Formex Configuration Processing II

Hoshyar Nooshin and Peter Disney

Space Structures Research Centre, Department of Civil Engineering, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom

ABSTRACT: This is the second paper in a series of papers that are intended to provide a comprehensive coverage of the concepts of formex configuration processing and their applications in relation to structural configurations. In the present paper, attention is focused on the configuration processing for a number of families of space structures, namely, pyramidal forms, towers, foldable systems and diamatic domes. Also included is a section on information export as well as an Appendix on basic formex functions. The section on information export describes the manner in which the information about the details of a configuration, generated by the programming language Formian, can be exported to graphics, draughting and structural analysis packages.

2.1 INTRODUCTION

Formex configuration processing provides a powerful medium for the processing of configurations of all kinds. Formex configuration processing uses the concepts of formex algebra through the programming language Formian to generate and process configurations. The preliminary concepts and ideas of formex configuration processing are described in the first instalment in this series of papers, Ref 1. The material in the present paper is highly dependent on that of Ref 1 and, therefore, the reader should be thoroughly familiar with the material of Ref 1 before attempting to study the present paper. For further information and downloading of Formian visit the web site: http://www.surrey.ac.uk/CivEng/research/ssrc/index.htm

2.2 PYRAMIDAL FORMS

Consider the pyramidal structure shown in Fig 2.2.1. The structure consists of 171 beam elements that are connected together forming a pyramid. The base of the pyramid is an equilateral triangle with each side being equal to L. The position of the global x-y-z coordinate system is assumed to be as shown in Fig 2.2.1. The base of the pyramid lies in the x-y plane with the origin of the coordinate system being at the centre of the base.

The height (altitude) of the pyramid is equal to H with the z axis passing through the apex. The pyramid has three identical triangular faces each of which consists of a planar arrangement of beam elements. The distance denoted by S in Fig 2.2.1a is
the distance between the apex of the pyramid and the midpoint of the base line of a face. The distance denoted by C in Fig 2.2.1c is the perpendicular distance from the centre to a side of the base of the pyramid. This distance is equal to the projection of S on the x-y plane.

The formulation of the configuration of the face shown in Fig 2.2.2a in terms of the indicated graduations along the x and y axes may be written as

$$E_1 = \text{genid}(1,6,2,1,-1,1) \mid \{(0,0,0; -1,1,0), [0,0,0; 1,1,0], [-1,1,0; 1,1,0]\}$$

In this equation, the part

$$\text{genid}(1,6,2,1,-1,1)$$

is a genid function that generates the triangulated arrangement of elements in Fig 2.2.2a, as described in Section 1.4.6 of Ref 1 and Section 2.A.4 of the Appendix. The argument of the genid function in the above equation, that is,

$$\{(0,0,0; -1,1,0), [0,0,0; 1,1,0], [-1,1,0; 1,1,0]\}$$

represents the three elements that are shown by thick lines in Fig 2.2.2a.

The configuration of Fig 2.2.2b is obtained by scaling the configuration of Fig 2.2.2a along the x and y axes. This may be achieved by writing

$$E_2 = \text{bb}(L/12, S/6) \mid E_1$$

The construct

$$\text{bb}(L/12, S/6)$$

is a basibifect retromorph that effects scaling by L/12 and S/6 in the x and y directions, respectively, as discussed in Section 1.4.4 of Ref 1 and Section 2.A.11 of the Appendix.

The configuration of Fig 2.2.2c may be obtained by writing

$$E_3 = \text{tran}(2, C-S) \mid E_2$$

In this equation, the part

$$\text{tran}(2, C-S)$$

is a translation function that effects a translation in the y direction by C - S, see Section 1.4.1 of Ref 1 and Section 2.A.3 of the Appendix.

The rotation of the face about its base line (Fig 2.2.2d) may be achieved by writing

$$E_4 = \text{verat}(C,0,-B) \mid E_3$$

The construct

$$\text{verat}(C,0,-B)$$

is a verat function that effects a rotation in the y-z plane about the base line of the face, see Section 1.4.7 of Ref 1 and Section 2.A.5 of the Appendix.

The distances C and S and the angle B may be obtained in terms of L and H as follows:

From Fig 2.2.3 that shows the base of the pyramid,

$$L/2C = \tan 60^\circ$$

and

$$C = L/(2 \tan 60^\circ)$$
Also, from Fig 2.2.2d,
\[ S = (C^2 + H^2)^{1/2} \]
and
\[ \tan B = H/C \]
or
\[ B = \arctan(H/C) \]

A Formian scheme for the generation of the configuration of the pyramidal structure of Fig 2.2.1 is shown in the editory display of Fig 2.2.4 (Formian schemes and editory displays are introduced in Ref 1, in particular, see Sections 1.3.3, 1.5 and 1.5.8). The scheme of Fig 2.2.4 contains Formian instructions that are based on the formulations presented above. However, the formulations in the scheme of Fig 2.2.4 are more general involving two additional parameters \( m \) and \( n \), where
- \( m \) denotes the number of elements along an edge of a face of the pyramid and
- \( n \) denotes the number of sides of the base of the pyramid.

The number of elements along an edge of a face of the pyramid (that is, \( m \)) is referred to as the 'frequency'. This is a measure of the 'density' of the pattern of the configuration of the pyramid.

The scheme of Fig 2.2.4 is a 'generic scheme'. The term 'generic' implies that the formulation is carried out in terms of parameters. This would allow the scheme to be used for exploring a variety of configurations rather than being restricted to a single configuration, see Sections 1.4.6 and 1.5.1 of Ref 1.

In the scheme of Fig 2.2.4, the formex variable \( E4 \) represents the configuration of a face of the pyramid as shown in Fig 2.2.2d. The configuration of the whole pyramid is then obtained by writing

\[ P = \text{pex} | \text{rosad}(0,0,n,360/n) | E4; \]

In this Formian statement, the construct
\[ \text{rosad}(0,0,n,360/n) \]
is a rosad function that generates the configuration of the entire pyramid by composing \( n \) rotations of the face represented by \( E4 \), see Section 1.4.7 of Ref 1 and Section 2.A.5 of the Appendix.

The part 'pex' in the above Formian statement is the pexum function that effects the removal of the superfluous overlapping elements along the edges of the pyramid, see Section 1.4.5 of Ref 1 and Section 2.A.7 of the Appendix.
Incidentally, to obtain the views shown in Fig 2.2.5, the use statement in the scheme of Fig 2.2.4 should be

\[ \text{use } \&.vm(2),vt(2),vh(0,-2*L,6*H,0,0,0,0,0,1); \]

for Figs 2.2.5a, 2.2.5b and 2.2.5c and

\[ \text{use } \&.vm(2); \]

for Fig 2.2.5d.

A major consideration in the design of pyramidal structures is the manner in which the elements are connected together. In the case of a 'single layer' pyramidal structure (like the ones in Figs 2.2.1, 2.2.5 and 2.2.6) the elements are, in general, under the effects of bending moments, shear forces, torques and axial forces. The connections should, therefore, be designed such that there are adequate rigidities in different directions to allow the transfer of the components of force and moment. The situation is rather different for a double layer pyramidal structure. In this case, the axial forces are usually the dominant effects in the members. Therefore, it is normally acceptable to use a connector that behaves more or less like a pin-joint. The above comments also apply in relation to single and double layer domes and barrel vaults.
Hoshyar Nooshin and Peter Disney

Double layer pyramidal structure of Fig 2.2.7 (*)

L=20; (*) length of each side of the base (*)

H=7; (*) height of pyramid (*)

D=1.5; (*) distance between two layers (*)

m=6; (*) frequency of top layer (*)

n=6; (*) number of sides of the base (*)

\[ C = \frac{L}{2 \tan\left(\frac{180}{n}\right)} \]

\[ S = \sqrt{C^2 + H^2} \]

\[ B = \tan^{-1}\left(\frac{H}{C}\right) \]

\[ E_T = \text{genid}(1,m,2,3,-1,1) \]

\[ E_B = \text{genid}(1,m-1,2,3,-1,1) \]

\[ E_W = \text{genid}(1,m,2,3,-1,1) \]

\[ E_1 = E_T \# E_B \# E_W \]

\[ E_2 = \text{bt}(L/(2*m),S/(3*m),D) \]

\[ E_3 = \text{tran}(2,C-S) \]

\[ E_4 = \text{verat}(0,0,-B) \]

Formulations in this scheme follow the same four-step procedure described for the single layer pyramidal structures, see Figs 2.2.2 and 2.2.4.

The step 1 of the formulation for the example of Fig 2.2.7 is carried out in terms of the graduations along the x, y and z axes as shown in Fig 2.2.9. The top layer of the face is assumed to be in the x-y plane and the bottom layer is assumed to be in the plane \( z = -1 \).

In the scheme of Fig 2.2.8, the formex variables \( ET, EB \) and \( EW \) represent the top layer elements, the bottom layer elements and the web elements of the face in Fig 2.2.9, respectively.

The only part of the scheme of Fig 2.2.8 that needs further explanation relates to the formulation for the elements that link the bottom layers of the neighbouring faces of the pyramid. One of these 'link elements' is denoted by \( ij \) in Fig 2.2.9. The position of node \( i \) relative to the reference system of Fig 2.2.9 is given by the signet

\[ Qi = [0,2,-1] \]

If \( Ri \) is a signet representing the final position of node \( i \) in the pyramid, then \( Ri \) is obtained from \( Qi \) through the following transformations:

- scaling by \( \text{bt}(L/(2*m),S/(3*m),D) \), corresponding to step 2 of the procedure described in Fig 2.2.2 (the function \( \text{bt}(L/(2*m),S/(3*m),D) \) is a basis function, as described in Section 1.7 of Ref 1 and Section 2.11 of the Appendix),
- translation by \( \text{tran}(2,C-S) \), corresponding to step 3 of the procedure described in Fig 2.2.2 and
- rotation by \( \text{verat}(0,0,-B) \), corresponding to step 4 of the procedure.

Therefore,

\[ Ri = \text{verat}(0,0,-B) \mid \text{tran}(2,C-S) \mid \text{bt}(L/(2*m),S/(3*m),D) \mid Qi \]

Also, if \( Rj \) is a signet representing the final position of node \( j \) in the pyramid, then \( Rj \) is obtained as

\[ Rj = \text{verad}(0,0,360/n) \mid Ri \]

The link element \( ij \) in its final position may be represented by

\[ K1 = \text{tig} \mid (Ri \# Rj) \]

The term 'tig' is the imprint of the 'tignum function' and the argument

\( (Ri \# Rj) \)

Fig 2.2.8 A generic scheme for the double layer pyramidal structure of Fig 2.2.7

Fig 2.2.9 Graduations along x, y and z axes for the formulation of a face of the pyramidal structure of Fig 2.2.7

Returning to the example of the double layer pyramidal structure of Fig 2.2.7, a scheme for the generation of the configuration of the structure is given in the editorial display of Fig 2.2.8.
is equal to the ingot

\[ \{[x_i,y_i,z_i], [x_j,y_j,z_j]\} \]

where \(x_i, y_i, z_i, x_j, y_j, z_j\) are the coordinates of nodes \(i\) and \(j\) in their final positions in the pyramid relative to the x-y-z coordinate system. The role of the tignum function is to create a cantle from the signets of an ingot, see Section 2.A.16 of the Appendix. Therefore, the construct

\[ \text{tig} | (R_i \# R_j) \]

is equivalent to the cantle

\[ [x_i,y_i,z_i; x_j,y_j,z_j] \]

This cantle represents the link element \(ij\) in its final position in the pyramidal structure of Fig 2.2.7.

Now, consider all the link elements in Fig 2.2.9. These elements in their final positions in the pyramidal structure of Fig 2.2.7 are given by

\[ K_2 = \text{rinax}(0,0,H, -L/(2*m), C/m, (m-1)*H/m, m) | K_1 \]

The construct

\[ \text{rinax}(0,0,H, -L/(2*m), C/m, (m-1)*H/m, m) \]

is a 'rinax function' which is a generalisation of the rindle function. A rinax function effects a translational replication in the direction given by a 'direction vector'. The direction vector is specified by the coordinates of its end points, as shown in Fig 2.2.10. Also, the amount of translation at each step of replication is given by the 'length' of the direction vector. Further information about the rinax function is given in Section 2.A.5 of the Appendix.

\[ \text{rinax}(X_1,Y_1,Z_1, X_2,Y_2,Z_2, m) \]

<table>
<thead>
<tr>
<th>number of</th>
<th>coordinates of the end point of the direction vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>replications</td>
<td>coordinates of the starting point of the direction vector</td>
</tr>
<tr>
<td>abbreviation for rindle proviax</td>
<td></td>
</tr>
</tbody>
</table>

Fig 2.2.10 Rinax function

In the case of the example under consideration, the coordinates of the starting point of the direction vector are specified as

\[ 0,0,H \]

and the co-ordinates of the end point of the direction vector are specified as

\[ -L/(2*m), C/m, (m-1)*H/m \]

This direction vector will be coincident with the final position of the top layer element indicated by \(ab\) in Fig 2.2.9.

Fig 2.2.11 A sectorate double layer grid generated by the scheme of Fig 2.2.8

\[
(*) \text{ Double layer grid of Fig 2.2.11 (*)} \\
L=20; \quad (*) \text{ length of each side (*)} \\
D=2.15; \quad (*) \text{ distance between two layers (*)} \\
m=6; \quad (*) \text{ frequency of top layer (*)} \\
n=8; \quad (*) \text{ number of sides (*)} \\
S=L/(2*\tan(180/n)); \\
ET=\text{genid}(1,m,2,3,-1,1) \{(0,0,0; -1,3,0), [0,0,0; 1,3,0], [-1,3,0; 1,3,0]\}; \\
EB=\text{genid}(1,m-1,2,3,-1,1) \{(0,2,1; -1,5,-1), [0,2,1; 1,5,-1], [-1,5,-1; 1,5,-1]\}; \\
EW=\text{genid}(1,m,2,3,-1,1) \{(0,0,0; 0,2,-1), [-1,3,0; 0,2,-1], [1,3,0; 0,2,-1]\}; \\
E=\text{bt}(L/(2*m), S/(3*m), D) | (ET#EB#EW); \\
Qi=[0,2,-1]; \\
Ri=\text{bt}(L/(2*m), S/(3*n), D) | Qi; \\
Rj=\text{verad}(0,0,360/n) | Ri; \\
K1=\text{tig} | (Ri \# Rj); \\
K2=\text{rinad}(0,0, -L/(2*m), S/m, m) | K1; \\
G=\text{pex} | \text{rosad}(0,0, n, 360/n) | (E#K2); \\
use \&, \text{vm}(2); \\
clear; \text{draw} G; \\
\text{\texttt{<<\texttt{>>}}}
\]

Fig 2.2.12 A generic scheme for the sectorate double layer grid of Fig 2.2.11

The scheme of Fig 2.2.8 may be used to generate a variety of double layer pyramids. It can also be used to create 'sectorate double layer grids'. These are 'degenerate' cases of double layer pyramids obtained...
by letting the height of the pyramid equal to zero. An example of a sectorate double layer grid is shown in Fig 2.2.11. This is generated by the scheme of Fig 2.2.8 using the parameter values

\[ L = 20, \ H = 0, \ D = 2.15, \ m = 6 \text{ and } n = 8 \]

A simplified version of the scheme of Fig 2.2.8 for the generation of sectorate double layer grids is shown in Fig 2.2.12. The simplification results from the fact that \( H = 0, \ S = C \) and steps 3 and 4 of the procedure of Fig 2.2.2 are not required for a sectorate double layer grid. Also, note that the rinax function of the scheme of Fig 2.2.8 is replaced by the simpler 'rinad function' in the scheme of Fig 2.2.12, see Section 2.A.5 of the Appendix.

### 2.3 TOWERS

The formulation of a number of lattice tower configurations is discussed in this section. As the first case, consider the tower a perspective view of which is shown in Fig 2.3.1. The tower consists of 144 elements that are connected together at 54 nodes. The nodes lie on a cylindrical surface that is referred to as the 'circumcylinder' of the tower. The height of the tower is \( H = 25 \) unit length and the radius of the cross-section of the circumcylinder is \( R = 5 \) unit length.

![A cylindrical tower](image)

The most convenient reference system for the formulation of the compret of the configuration of the tower of Fig 2.3.1 is a cylindrical normat. A cylindrical normat that suits the tower of Fig 2.3.1 is shown in Fig 2.3.2 (Cylindrical normats are discussed in Section 1.8 of Ref 1).

The three normat directions \( U_1, \ U_2 \) and \( U_3 \) in Fig 2.3.2 are shown together with the \( r-s-z \) cylindrical coordinate system as well as the \( x-y-z \) global Cartesian coordinate system. Also shown in Fig 2.3.2 is a rhombic unit of the tower. This is the part of the tower that is indicated by the letters A, B, C and D in Fig 2.3.1.

![A rhombic unit of the tower](image)

A formex \( F \) representing the compret of the configuration of the tower of Fig 2.3.1 relative to the normat of Fig 2.3.2 may be written as

\[
F = \text{rinit}(6,4,2,2) | \text{lamit}(0,1) | [1,0,0; 1,-1,1] \#
\]

\[
\{[1,-1,1; 1,1,1], [1,0,2; 1,2,2]\}
\]

In this formulation

- \( \text{lamit}(0,1) | [1,0,0; 1,-1,1] \)
  - represents elements AB, AC, BD and CD and
- \( \{[1,-1,1; 1,1,1], [1,0,2; 1,2,2]\} \)
  - represents elements BC and DE.

The constructs

- \( \text{rinit}(6,4,2,2) \)
- \( \text{lamit}(0,1) \)

are rinit and lamit functions, see Section 1.8 of Ref 1 and Section 2.A.4 of the Appendix.

The configuration of the tower of Fig 2.3.1 relative to the global \( x-y-z \) coordinate system can be obtained as

\[
T = \text{bc}(5,360/12,25/8) | F
\]

The construct

\[
\text{bc}(5,360/12,25/8)
\]

is a basicylindrical retronorm that transforms the \( U_1-U_2-U_3 \) normat coordinates into the equivalent \( x-y-z \) global coordinates, as discussed in Section 1.8 of Ref 1 and Section 2.A.11 of the Appendix.
In the above basic cylindrical representation:

- the first canonic parameter, that is, 5, specifies the length corresponding to every division in the U1 direction of the normat (this is the radius of the circumcylinder in the present example),
- the second canonic parameter, that is, 360/12, specifies the angle in degrees that corresponds to every division in the U2 direction of the normat and
- the third canonic parameter, that is, 25/8, specifies the length corresponding to every division in the U3 direction of the normat.

The reason for giving the second canonic parameter as 360/12, rather than 30, is to clarify the logic behind the specification. Namely, there are 12 subdivisions along the U2 direction (2 per rhombic unit) and the angle per subdivision must be such that these 12 subdivisions cover 360° (so that the rings of the tower close). Also, the third canonic parameter is given as 25/8 because there are 8 subdivisions along the U3 direction (2 per rhombic unit) and these 8 subdivisions should correspond to the full height of the tower which is 25 unit length.

A generic scheme for the creation of the configuration of the tower of Fig 2.3.1 is shown in the editor display of Fig 2.3.3. In this scheme, the formulations for the formex variables F and T are the parametric versions of the formulations given above. The parameters in the scheme are:

- H, denoting the height of the tower,
- R, denoting the radius of the circumcylinder of the tower,
- m, denoting the number of sides of a ring of elements in the tower and
- n, denoting the number of rhombic units in the vertical direction.

The statement following the formulations of the formex variables F and T in the scheme of Fig 2.3.3 is the use statement

```
use &.vm(2),vt(2),vh(14*R,14*R,2,0,0,0,0,0,1);
```

The role of this use statement is to set the viewing particulars for the perspective view of the tower in Fig 2.3.1, see Sections 1.5.2, 1.7.1 and 1.7.2 of Ref 1.

The scheme of Fig 2.3.3 may be used to generate a variety of cylindrical tower configurations by choosing different values for the parameters H, R, m and n. Three such examples are shown in Fig 2.3.4.

Three further examples of cylindrical tower configurations are shown in Fig 2.3.5. These may be obtained by slight modifications of the formulation of the formex variable F in the scheme of Fig 2.3.3. These modifications are left for the reader to carry out as exercise.
Now, consider the tower shown in Fig 2.3.6. This is a ‘tapered’ tower with the radius of its cross-section reducing gradually from the base radius of $R=5$ unit length to the top radius of $R_1=2.2$ unit length. Since the tower has seven ‘levels’, the reduction in the radius of the tower per level is

$$t = \frac{R-R_1}{7} = 0.4$$ unit length

The height of the tower is $H = 25$ unit length.

The nodal points of the tower of Fig 2.3.6 lie on a ‘conical’ surface a vertical section of which is shown on the left of the tower in Fig 2.3.6. This surface is referred to as the ‘circumsurface’ of the tower. Actually, the term ‘circumsurface’ is a general term that can be used to refer to a surface that contains all the nodal points of a configuration. For example, one may say that the circumsurface of the tower of Fig 2.3.1 is a cylinder or, in relation to a dome, one may say that the dome has a spherical (elliptical, conical,...) circumsurface.

The formulation of the configuration of the tapered tower of Fig 2.3.6 may be carried out using the normat of Fig 2.3.2. In doing this, the tower is imagined to be positioned such that the base of the tower is in the $x$-$y$ plane with the centre of the base being at the origin and with node $A$ of the base being on the $x$ axis. One may then write

$$F = \text{rin}(2,6,2) \mid \text{lib}(i = 0, 6) \mid \{[R-t(i),i,i; R-t(i+1),i-1,i+1],$$
$$[R-t(i),i,i; R-t(i+1),i+1,i+1],$$
$$[R-t(i+1),i-1,i+1; R-t(i+1),i+1,i+1]\}$$

$$T = \text{bc}(1,360/12,25/7) \mid F$$

In the above formulation, the formex variable $F$ represents the configuration of the tower of Fig 2.3.6 relative to the cylindrical normat of Fig 2.3.2 and $T$ represents the configuration of the tower relative to the global $x$-$y$-$z$ coordinate system.

The above formulation involves a formex function that had not been encountered so far in the discourse, namely,$$
\text{lib}(i = 0, 6)
$$

This is a ‘libra function’ with the term ‘$i$’ being referred to as the ‘libra index’ or ‘libra variable’, Fig 2.3.7.

The libra index assumes a sequence of integer values starting from the ‘initial’ value of 0 through to the ‘final’ value of 6, in steps of 1. To explain the effect of the libra function, let

- $G_0$ denote the value of the argument of the above libra function for $i=0$, that is, $G_0 = \{(R,0,0; R-t,-1,1], [R,0,0; R-t,1,1], [R-t,-1,1; R-t,1,1]\}$,

- $G_1$ denote the value of the argument of the function for $i=1$, that is, $G_1 = \{(R-t,1,1; R-2t,0,2], [R-t,1,1; R-2t,2,2], [R-2t,0,2; R-2t,2,2]\}$

and so on.

The libra function will then produce the formex

$G_0 \# G_1 \# G_2 \# G_3 \# G_4 \# G_5 \# G_6$

The formex $G_0$ represents the elements AB, AC and BC of the tower in Fig 2.3.6, the formex $G_1$ represents the elements CD, CE and DE of the tower and so on. The libra designator

$\text{lib}(i = 0, 6) \mid \{(R-t,i,i; R-t(i+1),i-1,i+1], \}$
$$[R-t(i),i,i; R-t(i+1),i+1,i+1],$$
$$[R-t(i+1),i-1,i+1; R-t(i+1),i+1,i+1]\}$$

represents the ‘spiral’ of elements that is shown on the right of the tower in Fig 2.3.6.

The ‘looping mechanism’ of the libra function is similar to that of the ‘sigma operator’ in scalar algebra. However, the libra function is concerned with composition of formices whereas the sigma operator is concerned with summation of numbers.

The libra function is a valuable concept and provides a versatile mechanism for dealing with many
problems in formex configuration processing. In the case of the present example of the tapered tower, none of the previously introduced replicating functions could have dealt with the problem in a convenient manner. A more detailed description of the `libra` function is given in Section 2.A.12 of the Appendix.

A generic scheme which is based on the above given formulation for the tapered tower of Fig 2.3.6 is shown in the editory display of Fig 2.3.8. Also, three further examples of tapered towers are shown in Fig 2.3.9. These towers are generated using the scheme of Fig 2.3.8 with different choices of parameters as indicated in Fig 2.3.9.

<table>
<thead>
<tr>
<th>(<em>) Tapered tower of Fig 2.3.6 (</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H=25; (<em>) height of tower (</em>)</td>
</tr>
<tr>
<td>R=5; (<em>) radius of the base of tower (</em>)</td>
</tr>
<tr>
<td>R1=2.2; (<em>) radius at the top of tower (</em>)</td>
</tr>
<tr>
<td>m=6; (<em>) number of sides of the base (</em>)</td>
</tr>
<tr>
<td>n=7; (<em>) number of levels (</em>)</td>
</tr>
<tr>
<td>t=(R-R1)/n;</td>
</tr>
<tr>
<td>F=rin(2,m,2)lib(i=0,n-1)</td>
</tr>
<tr>
<td>(R-t*i,i,i,t;i+1,i+1,i+1),</td>
</tr>
<tr>
<td>(R-t*i,i,i,t;i+1,i+1,i+1),</td>
</tr>
<tr>
<td>(R-t*(i+1),i-1,i-1,i+1,i+1);</td>
</tr>
<tr>
<td>T=bc1360/(2*m),H/n;F;</td>
</tr>
<tr>
<td>use &amp;vm(2),vt(2),vh(10<em>R,10</em>R,2,0,0,0,0,0,1);</td>
</tr>
<tr>
<td>clear; draw T;</td>
</tr>
<tr>
<td>&lt;&gt;&lt;&gt;&lt;&gt;</td>
</tr>
</tbody>
</table>

Fig 2.3.8 A generic scheme for the tapered tower of Fig 2.3.6

| R=8, R1=4.4, m=10, n=6 | R=4, R1=2.2, m=3, n=6 | R=4, R1=1.5, m=6, n=5 |

Fig 2.3.9 Examples of tapered towers generated by the scheme of Fig 2.3.8 (in all the three examples H=25)

As the next example, consider the tower shown in Fig 2.3.10. This is a 'doubly curved' tower whose

circumsurface is a surface of revolution with a parabolic generator, as shown in Fig 2.3.10. The basic dimensions of the tower are as follows:

- The overall height is H=25 unit length.
- The height at the 'neck' of the tower is H1=18 unit length (The 'neck' of the tower is the position of the smallest cross-section of the circumsurface of the tower).
- The radius of the cross-section of the circumsurface at the base of the tower is R=10 unit length.
- The radius of the cross-section of the circumsurface at the neck of the tower is R1=5 unit length.

![Fig 2.3.10 A parabolic tower](image)

The equation of the generator of the circumsurface is obtained from the general parabolic equation

\[ x = az^2 + bz + c \]

with the following three conditions:

- when \( z=0 \) then \( x=R \),
- when \( z=H1 \) then \( x=R1 \) and
- when \( z=H1 \) then the derivative of \( x \) with respect to \( z \) is equal to zero, that is, \( (dx/dz) = 2az + b = 0 \).

From these conditions, the coefficients \( a \), \( b \) and \( c \) are found to be

\[ a = (R-R1)/H1^2, \quad b = -2aH1 \quad \text{and} \quad c = R \]

The configuration of the parabolic tower of Fig 2.3.10 may be formulated in terms of the normat of Fig 2.3.2 using an approach identical to that explained for the tapered tower of Fig 2.3.6. A generic scheme which is based on this approach is shown in the editory display of Fig 2.3.11.

The only new point here is that the \( U1 \) coordinates in the argument of the `libra` function are given as

\[ a(di)^2 + bdi + c \]
or 
\[ a(d(i+1))^2 + bd(i+1) + c \]
where 
\[ d = \frac{H}{n} \]
and 
\[ d_i \] and 
\[ d(i+1) \]
represent z coordinates. Therefore, the expression 
\[ a(d_i)^2 + bd_i + c \]
or 
\[ a(d(i+1))^2 + bd(i+1) + c \]
represents 
\[ az^2 + bz + c \]
and this specifies a coordinate of a node of the tower in the radial direction of the cylindrical normat of Fig 2.3.2.

\[
(*) \text{ Parabolic tower of Fig 2.3.10 (*)}
H=25; (**) \text{ height of tower (**)}
H1=18; (**) \text{ height at the neck of tower (**)}
R=10; (**) \text{ radius of the base of tower (**)}
R1=5; (**) \text{ radius at the neck of tower (**)}
m=10; (**) \text{ number of sides of the base (**)}
n=8; (**) \text{ number of levels (**)}
a=(R-R1)/H1; b=-2*a*H1; c=R; d=H/n;
F=rin(2,m,2)[lib[i=0,n-1]]
{[a*(d^2(i)) + 2*b*d^2(i+1)+c,i,i;}
[a*(d^2(i+1))^2 + 2*b*d^2*(i+1)+c,i+1,i+1],
[a*(d^2*(i+1)) + 2*b*d^2*(i+1)+c,i+1,i+1],
[a*(d^2*(i+1)) + 2*b*d^2*(i+1)+c,i+1,i+1],
[a*(d^2*(i+1)) + 2*b*d^2*(i+1)+c,i+1,i+1];
T=bc(1,360/[2*m],d)[F];
use &.vm(2),vt(2),vh(10*R,10*R,2,0,0,0,0,0,1); clear; draw T;

Fig 2.3.11 A generic scheme for the parabolic tower of Fig 2.3.10

Three further examples of parabolic towers are shown in Fig 2.3.12. These are generated by the scheme of Fig 2.3.11 with parameter values
\[ H = 30, R = 8, R1 = 3, m = 10 \] and \[ n = 8 \]
for all the three cases and with varying values for the parameter \( H1 \). In the case of the example on the left, the neck of the tower is at the height of \( H1 = 20 \) unit length. In the case of the example at the middle, the neck of the tower is at the height of \( H1 = 30 \) unit length. That is, the neck is at the very top of the tower. In the case of the example on the right, the neck is at the height of \( H1 = 50 \) unit length which is higher than the top of the tower. It is permissible for the position of the neck to be 'outside' the tower since the neck is defined with respect to the circumsurface of the tower and the circumsurface extends beyond the actual body of the tower.

The approach used in this section for the generation of tapered and doubly curved towers may also be employed to generate tapered (conical) and doubly curved barrel vaults. For instance, the doubly curved barrel vault of Fig 2.3.13 is generated using the scheme shown in the editory display of Fig 2.3.14. The barrel vault of Fig 2.3.13 has a saddle shaped body. The particulars of the barrel vault are as follows:
- It consists of 430 beam elements that are connected together at 231 nodes.
- Its sweep angle is 70°, see Section 1.8 of Ref 1.
- Its length is 40 unit length.
Its circumsurface is a surface of revolution with a parabolic generator where the radius of the cross-section of the circumsurface at the ends is 12 unit length and where the minimum radius is at the middle and is equal to 8 unit length.

The "frequency" (that is, the number of elements along an edge) in the circumferential direction is equal to 10.

The frequency in the longitudinal direction is equal to 20.

(*) Doubly curved barrel vault of Fig 2.3.13 (*)
P=70; (*) sweep angle of barrel vault (*)
L=40; (*) length of barrel vault (*)
R=12; (*) radius at the ends (*)
R1=8; (*) radius at the middle (*)
m=10; (*) frequency in U2 direction (*)
n=20; (*) frequency in U3 direction (*)
a=4*(R-R1)/L; b=-a*L; c=R; d=L/n;
F=rin(2,m,1) lib(i=0,n) [a*(d*i) A 2+b*d*i+c,0,i; a*(d*i) A 2+b*d*i+c,1,i];
B=verad(0,0,90-P) bc(1.2,P,m,d); 
use &,vm(2),vt(2),vh(0,3*L,-2*L,0,0,L,0,1,L);
clear; draw B;
<><><>

Fig 2.3.14 A generic scheme for the doubly curved barrel vault of Fig 2.3.13

Perspective view
Plan view
Side view

Fig 2.3.15 A convex barrel vault generated by the scheme of Fig 2.3.14 (with R1=16)

The scheme of Fig 2.3.14 may be used to generate a variety of doubly curved barrel vaults. Included in these are the 'convex' barrel vaults an example of which is shown in Fig 2.3.15.

In this case, the radius of the circumsurface of the barrel vault at the middle is R1=16 unit length which is larger than the radius at the ends, namely, R=12 unit length.

2.4 FOLDABLE SYSTEMS

A number of foldable configurations are considered in this section. These relate to the type of "foldable structures" (deployable structure) that consist of scissors-like units. To begin with, consider an example of a very simple foldable system, namely, the 'lazy-tongs' configuration shown in Fig 2.4.1a. The configuration consists of four scissors-like units. The corners of one of these units are indicated by letters i, j, k and l. The elements ik and jl are 'pivoted' together at the middle such that the unit can be opened and closed like a pair of scissors.
unit of this kind is referred to as a ‘duplet’ and each one of the parts $ik$ or $jl$ is referred to as a ‘uniplet’. The duplet $ijkl$ is ‘pinned’ (hinged) to its neighbouring duplets at points $i$, $j$, $k$ and $l$, so that the whole configuration of Fig 2.4.1a can be opened and closed like a pair of scissors. The extent of the folding of the configuration of Fig 2.4.1a may be specified by a single parameter. This parameter is chosen to be the angle $t$, as shown in the figure. Three folded states of the lazy-tongs configuration are shown in Fig 2.4.1. These correspond to $t=90^\circ$, $t=120^\circ$ and $t=60^\circ$, as shown in the figure. The angle $t$ is referred to as the ‘control angle’.

![Fig 2.4.1 Three folded states of lazy-tongs configuration](image)

To work out the geometry of a folding duplet, consider the arrangement shown in Fig 2.4.2. The duplet shown by full lines in this figure has the same horizontal and vertical dimensions $D$. The control angle for this duplet is equal to $90^\circ$. For any other value of the control angle the horizontal dimension $H$ of the duplet will be different from its vertical dimension $V$. This is illustrated for a folded state of the duplet shown by dotted lines in Fig 2.4.2.

![Fig 2.4.2 A folding duplet](image)

The horizontal and vertical dimensions of the dotted duplet are found to be:

\[ H = \sqrt{2} D \sin(t/2) \] and \[ V = \sqrt{2} D \cos(t/2) \]

The two-directional extension of the idea of lazy-tongs is a ‘foldable (double layer) grid’ an example of which is shown in Fig 2.4.3.

![Fig 2.4.3 A foldable grid](image)

The grid of Fig 2.4.3 consists of 71 identical duplets that are pinned together creating an assembly of duplets that can be folded into a ‘bundle’. The dimension $D$ (as shown in Fig 2.4.2) for the duplets in the grid of Fig 2.4.3 is equal to 0.8 unit length and the control angle of the duplets is equal to $100^\circ$. In practice, a foldable grid of the kind shown in Fig 2.4.3 is stabilised (and stiffened) by adding a number of elements and/or by constraining the grid at the support points when the grid is in the required final unfolded state.

A formex formulation for the configuration of the foldable grid of Fig 2.4.3 may be written as follows:

\[ E = \text{rinid}(6,6,1,1) \{[0,0,0; 1,0,1], [0,0,1; 1,0,0]\} \# \text{rinid}(7,5,1,1) \{[0,0,0; 0,1,1], [0,0,1; 0,1,0]\} \]
\[ F = \text{bt}(H,H,V) \mid E \]

In this formulation:
- the formex $\{[0,0,0; 1,0,1], [0,0,1; 1,0,0]\}$ represents the duplet in the left corner of the grid in the $U_1$ direction (shown by thick lines),
- the formex $\{[0,0,0; 0,1,1], [0,0,1; 0,1,0]\}$ represents the duplet in the left corner of the grid in the $U_2$ direction (shown by thick lines),
- $E$ represents the configuration of the grid relative to the given $U_1-U_2-U_3$ normat and
- $F$ represents the configuration of the grid relative to the global $x$-$y$-$z$ coordinate system, where $H$ and $V$ are as described for the duplet of Fig 2.4.2.
A generic scheme for the foldable grid of Fig 2.4.3 is shown in the display of Fig 2.4.4. The formex formulation in this scheme is based on the formulation given above with two differences. Firstly, the duplet frequencies in the x and y directions are given by parameters m and n. Secondly, to show an alternative approach in dealing with the problem, the formulation has been carried out directly in terms of the global x-y-z coordinate system without the involvement of an intermediate normat.

The scheme of Fig 2.4.4 may be used to generate foldable grids of the kind shown in Fig 2.4.3 with different values for the frequencies m and n and the size of the duplets represented by D. Also, the scheme can generate the configurations of different folded states of a grid by changing the value of the control angle t. For example, four different folded states of the grid of Fig 2.4.3 are shown in Fig 2.4.5. These are generated by the scheme of Fig 2.4.4 with the control angles

\[ t = 120^\circ, \quad t = 90^\circ, \quad t = 60^\circ \text{ and } t = 30^\circ \]

Now, consider the foldable barrel vault shown in Fig 2.4.6. The barrel vault has 42 duplets in the circumferential direction. These create 6 circular 'arches'. The arches are connected together by 40 duplets in the longitudinal direction. The duplets in the circumferential direction create arches because they have a 'trapezial' shape, as shown in Fig 2.4.7. In a 'trapezial duplet' the length L1 of the upper parts of the uniplets is different from the length L2 of the lower parts. In the present example, L1 = 0.8 unit length, L2 = 0.7 unit length and the control angle t is equal to 140°. Details of one of the arches of the barrel vault of Fig 2.4.6 are shown in Fig 2.4.8.

The duplets in the longitudinal direction of the barrel vault of Fig 2.4.6 are 'rectangular duplets'. In this case the uniplets are pivoted together at the middle, as shown in Fig 2.4.7 (The duplets in the lazy-tongs of Fig 2.4.1 and the grid of Fig 2.4.3 are rectangular duplets).

In order to work out the geometry of the foldable barrel vault of Fig 2.4.6, it is necessary to obtain the radius R of the top circumcylinder, the depth D of the barrel vault, the central angle C of a duplet in the circumferential direction and the distance D1 between the arches in terms of L1, L2 and the control angle t.
For any triangle with side lengths a, b, c and the corresponding opposite angles \( \alpha, \beta, \gamma \), the following rules apply:
\[
\frac{a}{\sin \alpha} = \frac{b}{\sin \beta} = \frac{c}{\sin \gamma} \\
a^2 = b^2 + c^2 - 2bc \cos \alpha
\]

From triangle 124 in Fig 2.4.7 and the second rule in Fig 2.4.9,
\[
D = (L_1^2 + L_2^2 - 2L_1L_2 \cos \alpha)^{1/2}
\]
Also, from the rectangular duplet in Fig 2.4.7,
\[
D_1 = (L^2 - D^2)^{1/2}
\]

From triangle 124 in Fig 2.4.7 and the first rule in Fig 2.4.9,
\[
L_2 / \sin A = D / \sin \alpha
\]
or
\[
A = \arcsin(L_2 \sin \alpha / D)
\]
Again, from triangle 124 in Fig 2.4.7,
\[
B_1 = 180^\circ - A - t_1 = 180^\circ - A - (180^\circ - t) = -A + t
\]
and
\[
B = 180^\circ - B_1 = 180^\circ + A - t
\]
From triangle 135 in Fig 2.4.7 and the first rule in Fig 2.4.9,
\[
R / \sin B = (R - D) / \sin A
\]
or
\[
R = (D \sin B) / (\sin B - \sin A)
\]
Also,
\[
L / \sin C = R / \sin B
\]
or
\[
C = \arcsin(L \sin B / R)
\]

A formex formulation for the barrel vault of Fig 2.4.6 may now be written as
\[
E_1 = \text{rin}(7,6,C,D_1) \big| \{[R-D,0,0; R,C,0], [R,0,0; R-D,C,0]\}
\]
\[
E_2 = \text{rin}(8,5,C,D_1) \big| \{[R-D,0,0; R,0,D_1], [R,0,0; R-D,0,D_1]\}
\]
\[
F = \text{bc}(1,1,1) \big| (E_1 \# E_2)
\]
\[
P = \frac{7C}{2}
\]
\[
BV = \text{verad}(0,0,90-P) \big| F
\]
In the above formulation:
- \( E_1 \) represents the duplets in the circumferential direction relative to the r-s-z cylindrical coordinate system,
- \( E_2 \) represents the duplets in the longitudinal direction relative to the r-s-z coordinate system,
- \( F \) represents the whole configuration relative to the global x-y-z coordinate system, where,
  \[
  \text{bc}(1,1,1)
  \]
is a basicylindrical retromor, as described in Section 1.8 of Ref 1 and Section 2.A.1.1 of the Appendix,
- \( P \) is the ‘sweep angle’ of the barrel vault, as described in Section 1.8 of Ref 1 and
- \( BV \) is a rotation of \( F \) representing the barrel vault of Fig 2.4.6 with the y axis in the vertical position, as discussed in Section 1.8 of Ref 1.

A generic Formian scheme which is based on the above formulation is shown in the editory display of Fig 2.4.10. This scheme is used to generate three folded states of the barrel vault of Fig 2.4.6. These are shown in Fig 2.4.11 together with the corresponding values of the control angle.
(* Foldable barrel vault of Fig 2.4.6 (*)
t=140; (* control angle (*)
L1=0.8; (* length of upper part of uniplet (*)
L2=0.7; (* length of lower part of uniplet (*)
m=7; (* circumferential frequency (*)
n=5; (* longitudinal frequency (*)
L=L1+L2; t1=180-t;
D=sqrt(L1^2+L2^2-2*L1*L2*cos(t1));
D1=sqrt(L^2-D^2);
A=asin(L2*sin(t1)/D);
B=180+A-t;
R=D*sin(B)/(sin(B)-sin(A));
C=asin(L*sin(B)/R);
E=rinit(m,n+1,C,D1) I {[R-D,O,O; R-D/2,0,DI/2; R,O,DI],
[R-D,0,0; R-D/2,0,DI/2; R-D,O,DI]}#
E = rinit(m+1,n,C,D1) I {[R-D,O,O; R-D/2,0,DI/2; R,O,DI],
[R-D,0,0; R-D/2,0,DI/2; R-D,O,DI]};
F=bc(1,1,1)|E; P=m*C/2;
BV=verad(0,90-P)|F;
use &vm(2),vl(2).vh(5.8*R,-12*R, 0,0,0,1,R); clear; draw BV;
Radius=R; Depth=D; Side=n*D1; Sweep=P;
Span=2*R*sin|P; Rise=R*(1-cos|P); give Radius,Depth,Side,Sweep,Span,Rise;
<><><>>

Fig 2.4.10 A generic scheme for the foldable barrel vault of Fig 2.4.6

A useful feature of the scheme of Fig 2.4.10 is that it effects the display of information in a give box (see Section 1.3.4 of Ref 1). An example of a give box for the barrel vault of Fig 2.4.6 is shown in Fig 2.4.12. This provides a convenient tool for the design of a foldable system since the effects of changes in the parameters t, L1, L2, m and n can be easily observed and the parameters can be adjusted to suit the design requirements.

In the above example, the initial parameters are chosen to be L1, L2, t, m and n. However, if required, the problem may be formulated in terms of a different set of initial parameters. For instance, one may choose the span and rise of the barrel vault, instead of L1 and L2, as initial parameters.

The foldable systems described in this section are 'free-folding systems'. A free-folding system can be folded and unfolded freely without the need to exert any force to effect folding or unfolding (that is, if the system is assumed to be weightless and if the pivots and pins in the system are assumed to be frictionless). In the case of a free-folding system, the configuration of any folded state can be obtained by pure geometric considerations. In contrast, there are a number of different types of foldable systems for which the process of folding and unfolding involve storage and release of strain energy. For a system of this kind the folding and unfolding require exertion of force and the configuration of an intermediate folded state of the system can only be obtained through the structural analysis of the system.

Structural analysis (as well as configuration processing) of foldable systems is discussed at length in Ref 3. For the purposes of analysis, it may be required to represent the uniplets as 'three-noded' beam elements, with the nodes being at the two ends and the pivot point. The formex formulations in this section can be easily modified to cater for this requirement. For instance, to modify the scheme of Fig 2.4.10, the assignment statement that creates the formex variable E can be changed to

E = rinit(m,n+1,C,D1) I
{[R-D,0,0; L2*sinB/sin(C/2),C/2,0; R,C,0],
[R,0,0; L2*sinB/sin(C/2),C/2,0; R-D,C,0]#
rinit(m+1,n,C,D1)} I
{[R-D,0,0; R-D/2,0,D1/2; R,0,D1],
[R,0,0; R-D/2,0,D1/2; R-D,0,D1]}
2.5 DIAMATIC DOMES

Three families of lattice domes were discussed in Section 1.9 of Ref 1. These were the families of ribbed domes, Schwedler domes and lamella domes. Another family of lattice domes, namely, 'diamatic domes', will be described in this section. The material presented in this section is closely related to that of Section 1.9 of Ref 1 and the reader is expected to be thoroughly familiar with the material of that section before embarking on the study of the present section.

An example of a diamatic dome is shown in Fig 2.5.1, where the nodal points of the dome are assumed to be on a spherical cap.

Diamatic domes are one of the most efficient types of domes and are frequently used in practice. Two notable domes of this type are the Superdome in New Orleans and the Astrodome in Texas.

A diamatic dome consists of a number of 'sectors' with the characteristic pattern of elements an example of which is shown in Fig 2.5.1. The side boundaries of a sector are along two meridians of the circumsphere of the dome and the bottom boundary is along a parallel of the circumsphere. The number of elements along a boundary of a sector is referred to as the 'frequency'. The diamatic dome of Fig 2.5.1 has six sectors with the frequency being also six. In general, a diamatic dome may have any number of sectors and any frequency.

An important feature of a diamatic dome is that it does not present any problem regarding 'element cluttering' near the crown, as discussed in Section 1.9 of Ref 1. This may be clearly seen by comparing the dome of Fig 2.5.1 with a lamella dome of similar proportions shown in Fig 2.5.2.

To elaborate further on the problem of element cluttering, a sector of the dome of Fig 2.5.1 and the corresponding part of the dome of Fig 2.5.2 are shown side by side in Fig 2.5.3. The reason for the cluttering of elements near the crown in a lamella dome is that the pattern of the elements 'follows the meridians', as shown in Fig 2.5.3b (The meridians are shown by dotted lines). However, in the case of a diamatic dome, there is no such direct relationship between the pattern of the elements and the meridians, as can be seen from Fig 2.5.3a.

Formex formulation of the configuration of a lamella dome may be conveniently carried out in terms of a spherical normat, as described in Section 1.9 of Ref 1. However, a spherical normat is not the most suitable reference system for the formulation of a diamatic dome. Interestingly, the relationship between the meridians and the pattern of the elements in a lamella dome that creates the undesirable element cluttering is the very reason for the fact that the configuration of a lamella dome can be formulated in a convenient manner relative to a spherical normat. This comment also applies to the other types of domes discussed in Section 1.9 of Ref 1.

It is possible to formulate the configuration of a diamatic dome relative to a spherical normat and a number of examples of elegant formulations for diamatic domes in terms of spherical normats are found in Ref 6. However, the most convenient way...
of formulating a diamatic dome is through a modified form of a spherical normat\textsuperscript{7,8,9}. An example of this type of normat is shown in Fig 2.5.4. This is referred to as a 'diamatic normat'.

![Fig 2.5.4 A diamatic normat](image)

The first direction $U_1$ in a diamatic normat relates to the radial normat coordinates in exactly the same way as for a spherical normat, see Section 1.9 of Ref 1. Also, the third direction $U_3$ of a diamatic normat relates to the normat coordinates along a meridian in the same way as for a spherical normat, see Section 1.9 of Ref 1. The normat graduations in the third direction are shown encircled in Fig 2.5.4. In this figure, the normat divisions in the third direction are chosen to correspond to the rings of elements in the dome of Fig 2.5.1.

The difference between a spherical and a diamatic normat is in the arrangement of the normat coordinates in the second direction. To elaborate, in a spherical normat every second direction coordinate relates to a meridian, see Section 1.9 of Ref 1. In the case of a diamatic normat, the pattern governing the second direction coordinates is in accordance with the arrangement shown in Fig 2.5.4. That is, for every node along the line of elements on the left border of the first sector, the second normat coordinate is equal to zero, for every node along the line of elements to the right of the left border, the second normat coordinate is equal to one, and so on.

The diamatic normat of Fig 2.5.4 may be alternatively visualised in terms of 'normat lines', as shown in Fig 2.5.5. Here, every first direction normat coordinate $U_1$ represents a sphere of radius $U_1$, every second direction normat coordinate $U_2$ represents an inclined normat line on the sphere and every third direction normat coordinate $U_3$ represents a 'parallel' on the sphere.

The working of a diamatic normat is illustrated in terms of the plots of the formices:

- $E_1 = \{[1,0,2; 1,1,2], [1,0,2; 1,0,3]\}$
- $E_2 = \text{tran}(3,3)$ \|$E_1$
- $E_3 = \text{ref}(2,1.5)$ \|$E_2$
- $E_4 = \text{ref}(3,4.5)$ \|$E_3$
- $E_5 = \text{transit}(-2.3)$ \|$E_4$
- $E_6 = \text{lamit}(3.5,6.5)$ \|$E_3$

The plots of formices $E_1$ to $E_6$ are shown relative to the diamatic normat of Fig 2.5.6. In this figure, the plots of $E_1$ to $E_6$ are shown by thick lines and are denoted by $E_1'$ to $E_6'$, respectively. As far as the translational effects are concerned, it is seen that the movements take place along the normat lines. For instance, $E_2'$ is obtained by moving $E_1'$ by three divisions in the $U_3$ direction along a normat line.
Also, E5' is obtained by moving E4' by two divisions in the negative U2 direction and three divisions in the positive U3 direction along normal lines.

The reflectional effects are also governed by the normal lines. For example, E3' is obtained by reflecting E2' in the U2 direction with respect to the axis indicated by A1 (shown by dotted line). Also, E4' is obtained by reflecting E3' in the U3 direction with respect to the axis A2 and E6', consisting of four parts indicated by asterisks, is obtained from E3' by reflections in the U2 and U3 directions with respect to axes A3 and A4.

Now, the configre of the configuration of the first sector of the diamatic dome of Fig 2.5.1 may be formulated as follows:

\[
E = \text{genit}(1,6,1,1,0,1) \{[1,0,0; 1,0,1], [1,0,0; 1,1,1], [1,0,1; 1,1,1]\}
\]

\[
F = \text{bd}(30,60,6) | E
\]

In this formulation, E represents the configuration of the first sector (front sector) of the dome of Fig 2.5.1 relative to the normat of Fig 2.5.4. In the equation

\[
E = \text{genit}(1,6,1,1,0,1) \{[1,0,0; 1,0,1], [1,0,0; 1,1,1], [1,0,1; 1,1,1]\}
\]

the part

\[
\text{genit}(1,6,1,1,0,1)
\]

is a genit function that is used for the creation of the triangulated pattern of the elements of the first sector of the dome, see Section 2.A.4 of the Appendix and Section 1.4.6 of Ref 1. The argument of the genit function, that is,

\[
\{[1,0,0; 1,0,1], [1,0,0; 1,1,1], [1,0,1; 1,1,1]\}
\]

is a formex representing the top three elements of the first sector with the circumradius of the dome taken as 1.

The formex F in the above formulation represents the configuration of the first sector of the dome of Fig 2.5.1 relative to the global x-y-z coordinate system. In the equation

\[
F = \text{bd}(30,60,6) | E
\]

the part

\[
\text{bd}(30,60,6)
\]

is a ‘basidiamatic retronorm’. The effect of a basidiamatic retronorm is to transform the coordinates of a configuration that are given relative to a diamatic normat into equivalent global Cartesian x-y-z coordinates. A basidiamatic retronorm is a modified form of a basispherical retronorm, see Section 2.A.11 of the Appendix and Section 1.9 of Ref 1. A basidiamatic retronorm has three canonic parameters, as explained in Fig 2.5.7.

\[
\text{bd}(b1,b2,b3)
\]

- factor for scaling in the third (meridional) normat direction (angular scale factor)
- factor for scaling in the second (circumferential) normat direction, given as the sectoral angle, that is, the central angle of a sector in plan
- factor for scaling in the first (radial) normat direction (linear scale factor)

abbreviation for basidiamatic

Fig 2.5.7 Basidiamatic retronorm

The first and third canonic parameters of a basidiamatic retronorm work in the same way as the corresponding parameters in a basispherical retronorm, see Section 2.A.11 of the Appendix and Section 1.9 of Ref 1. However, the second canonic parameter of a basidiamatic retronorm is rather different. Here, the second canonic parameter specifies the ‘sectoral angle’ rather than the angle for each division in the U2 direction of the normat. The term ‘sectoral angle’ is used to refer to the central angle of a sector in plan.

A basidiamatic retronorm transforms the U1-U2-U3 normat coordinates into their equivalent r-s-t spherical coordinates using the relations:

\[
r = b1 \times U1
\]

\[
s = b2 \times U2 / |U3| \quad \text{if} \ U3 \neq 0
\]

\[
s = 0 \quad \text{if} \ U3 = 0
\]

\[
t = b3 \times U3
\]

The basidiamatic retronorm will then transform the resulting r-s-t spherical coordinates into their equivalent global Cartesian x-y-z coordinates using the standard transformation rules, see Section 1.9 of Ref 1.

Returning to the formulation of the diamatic dome of Fig 2.5.1, the formex representing the configuration of the first sector of the dome relative to the global x-y-z coordinate system is given by

\[
F = \text{bd}(30,60,6) | E
\]

where

- the value of 30 for the first canonic parameter of the retronorm gives the scale factor in the radial direction, and since the value of U1 in the formulation of E is given as 1 for all the nodal points, it follows that the radius of the
circumsphere of the dome is equal to 30 units of length,

- the value of 60 for the second canonic parameter indicates that the sectoral angle is equal to 60° and
- the value of 6 for the third canonic parameter indicates that every division in the U3 direction is 6°.

The third canonic parameter of the retronorm controls the 'bulginess' of the dome. In the present example, since the dome has 6 divisions along a meridian from the crown to the base ring, the 'sweep angle' of the dome will be $6 \times 6° = 36°$, see Section 1.9 of Ref 1.

The formulation of the diamic dome of Fig 2.5.1 has so far resulted in obtaining the formex F that represents the configuration of the first sector of the dome relative to the global x-y-z coordinate system. The whole dome may then be represented by

$$
G_{C} = \text{pex} \left[ \text{rosad}(0,0,6,60) \right] \text{F}
$$

In this equation

$$
\text{rosad}(0,0,6,60)
$$

is a 'rosad function' effecting a rotational replication in the x-y plane about the origin creating a combination of 6 sectors at 60° intervals, see Section 2.A.5 of the Appendix and Section 1.4.7 of Ref 1.

In the above equation, the part 'pex' before the rosad function is a 'pexum function' effecting the removal of the superfluous 'overlapping' border elements of the adjacent sectors, see Section 2.A.7 of the Appendix and Section 1.4.5 of Ref 1.

A Formian scheme for the diamic dome of Fig 2.5.1 is shown in the editory display of Fig 2.5.8. The scheme is 'generic', that is, it is written in terms of parameters. These parameters are:

- R, denoting the radius of the circumsphere of the dome,
- m, denoting the frequency, that is, the number of elements along each edge of a sector,
- n, denoting the number of sectors and
- A, denoting the sweep angle of the dome.

In the scheme of Fig 2.5.8, the assignment statements that create the formex variables E, F and G are the parametric versions of the formulations for E, F and G as discussed above.

The use statement following the assignment statements in the scheme of Fig 2.5.8 sets the viewing particulars for the perspective view of the dome of Fig 2.5.1, see Sections 1.5.2, 1.7.1 and 1.7.2 of Ref 1.

With the choice of $R = 30$, $m = 6$, $n = 6$ and $A = 36°$, the scheme of Fig 2.5.8 generates the diamic dome of Fig 2.5.1. However, many other diamic dome configurations may be generated through the scheme of Fig 2.5.8 by simply changing the values of the parameters. Four such examples are shown in Fig 2.5.9. These are generated by the scheme of Fig 2.5.8 with different values for the parameters m and n, as indicated in Fig 2.5.9.

Incidentally, diamic patterns of the type shown in Figs 2.5.1 and 2.5.9 may also be employed for finite element meshes. In this case, the component parts of the configurations will be 'tile' elements with three or more nodal points rather than 'linear' elements with two nodes at the ends. Such finite element meshes are used for the analysis of spherical shells.
A Formian scheme for the generation of diamatic finite element meshes is shown in the editory display of Fig 2.5.10. The schemes of Figs 2.5.8 and 2.5.10 produce 'similar looking' configurations. However, the configuration produced by the scheme of Fig 2.5.8 will consist of two-noded linear elements whereas the configuration produced by the scheme of Fig 2.5.10 will consist of triangular elements with three corner nodes.

The main difference between the schemes of Figs 2.5.8 and 2.5.10 is in the formulation of formex variable \( E \) representing the configuration of the first sector relative to the normat of Fig 2.5.4. Another difference between the two schemes is that the formulation for the formex variable \( G \) in the scheme of Fig 2.5.10 does not involve the 'pexum function'. This is due to the fact that in the finite element version of the configuration the neighbouring sectors do not have any overlapping parts, see Section 1.8 of Ref 1.

The schemes of Figs 2.5.8 and 2.5.10 have a further two minor differences. Firstly, the values given for the radii of circumsphere in the two schemes are different. Secondly, the use statement in the scheme of Fig 2.5.10 has an extra use-item, namely:

\[ c(3,40) \]

The effect of this use-item is that the finite-elements will be plotted with an infill colour. The infill colour will be the colour whose code number is 40, see Section 1.7.2 of Ref 1.

Returning to the discussion of 'lattice' diamatic domes, it should be mentioned that the examples considered so far (Figs 2.5.1 and 2.5.9) belong to a particular family of diamatic domes that are referred to as 'parallel lamella domes'. The distinguishing characteristic of this family of diamatic domes is that each sector is fully triangulated with the 'element lines' being approximately 'parallel' to the edges of the sector.

However, there are many other 'diamatic patterns' that can be used for domes. In general, a 'diamatic pattern' is defined as any pattern that can be obtained as a combination of elements whose 'nodal points' are on the 'normat points' of a diamatic normat. For example, consider the dome a perspective view of which is shown in Fig 2.5.11. This is a diamatic dome with a honeycomb pattern.

Included in Fig 2.5.11 is a generic formulation for the dome together with the diamatic normat with respect to which the formulation is carried out. The elements that are shown on the normat correspond to those elements that are shown by thick lines at the top of the first sector of the dome in Fig 2.5.11. To create a scheme for the generation of domes of the form shown in Fig 2.5.11, the formulations for the formex variables \( E \), \( F \) and \( G \) in the scheme of Fig 2.5.8 should be replaced by those given in Fig 2.5.11.

Using the same style of presentation as in Fig 2.5.11, another example of a diamatic dome is shown in Fig 2.5.12. A new feature in this example is the use of diamatic normat coordinates 'beyond' the borders of the first sector.
Further examples of diamatic domes are shown in Fig 2.5.14. The formulations of these domes are left for the reader to carry out as exercise.

Any diamatic dome may be transformed into an ellipsoidal form by scaling along coordinate axes, as discussed in Section 1.9 of Ref 1. For example, the diamatic dome whose plan view is shown in Fig 2.5.15b is obtained by scaling of the dome of Fig 2.5.15a in the x direction by the scale factor 0.8.

Also, one may create an 'ovate diamatic dome' by using different scale factors for the opposite halves of a diamatic dome in a manner similar to that discussed in Section 1.9 of Ref 1. For example, the ovate diamatic dome of Fig 2.5.15c is obtained by scaling the bottom half of the dome of Fig 2.5.15a in the y direction by the factor 1.3. Also, the ovate diamatic dome of Fig 2.5.15d is obtained by scaling the top and bottom halves of the dome of Fig 2.5.15a in the y direction by factors 1.1 and 0.85, respectively.

In an ovate dome, due to different scalings of the opposite halves, the pattern in one half is bound to be more 'compact' than that in the other half, as may be seen from Figs 2.5.15c and 2.5.15d. One way of eliminating (or lessening) this effect is to reduce the number of sectors in the half with the more compact pattern. This is illustrated in Fig 2.5.16. The ovate
dome shown in this figure is the same as the ovate dome of Fig 2.5.15d except that the number of the sectors in the 2nd half (bottom half) is reduced from four to three.

![Images of domes](a) (b) (c) (d)

Fig 2.5.15 Examples of ellipsoidal and ovate diamatic domes

Fig 2.5.16 Example of an ovate diamatic dome with unequal numbers of sectors in the opposite halves

The dome of Fig 2.5.16 may be obtained using the generic scheme shown in the editory display of Fig 2.5.17. This scheme is based on the diamatic pattern of Fig 2.5.11. However, the scheme may be modified to work with any other diamatic pattern. To do this, the only part of the scheme that needs modification is the formulation for the formex variable E that represents the configuration of the first sector relative to a diamatic normat.

The manner in which scaling is effected in the scheme of Fig 2.5.17 needs some explanation. To elaborate, in Sections 1.8 and 1.9 of Ref 1, the basibifect and basitrifect retronorms were used for scaling of barrel vault and dome configurations to create elliptic and ovate forms. However, as explained in Section 1.8 of Ref 1, acting in the capacity of a scaling function is not the primary role of a basibifect or basitrifect retronorm. The fundamental role of a retronorm is to transform normat coordinates into global x-y-z coordinates.

<table>
<thead>
<tr>
<th>(*)</th>
<th>Ovate diamatic domes (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R=30; (*)</td>
<td>initial radius of circumsphere (*)</td>
</tr>
<tr>
<td>m=8; (*)</td>
<td>number of rhombic openings along a meridian (*)</td>
</tr>
<tr>
<td>n1=4; (*)</td>
<td>number of sectors in 1st half (*)</td>
</tr>
<tr>
<td>n2=3; (*)</td>
<td>number of sectors in 2nd half (*)</td>
</tr>
<tr>
<td>S1=1.1; (*)</td>
<td>scale factor for 1st half (*)</td>
</tr>
<tr>
<td>S2=0.85; (*)</td>
<td>scale factor for 2nd half (*)</td>
</tr>
<tr>
<td>A=45; (*)</td>
<td>sweep angle (*)</td>
</tr>
<tr>
<td>E=genit(1,m,3,3,0,1);{[1,0,0;1,1,2],[1,1,2;1,0,3],[1,1,2;1,3,3]};</td>
<td></td>
</tr>
<tr>
<td>F1=bd(R,180/n1,A/(3*m));E;</td>
<td></td>
</tr>
<tr>
<td>F2=bd(R,180/n2,A/(3*m));E;</td>
<td></td>
</tr>
<tr>
<td>G=dil(2,S1)</td>
<td>rosad(0,0,n1,180/n1)</td>
</tr>
<tr>
<td>clear;</td>
<td></td>
</tr>
<tr>
<td>use &amp;;vm(2);</td>
<td></td>
</tr>
<tr>
<td>draw G;</td>
<td></td>
</tr>
</tbody>
</table>

Fig 2.5.17 A generic scheme for ovate diamatic domes

The basic 'scaling function' in formex algebra is the 'dilatation function', where the term 'dilatation' implies increase or decrease in size. The particulars of the dilatation function are shown in Fig 2.5.18, also see Section 2.A.3 of the Appendix.

\[
dil(d,f) \]

- factor of dilatation (scale factor)
- direction of dilatation
- abbreviation for dilatation

Fig 2.5.18 Dilatation function

Now, referring to the scheme of Fig 2.5.17, the dilatation functions

\[
dil(2,S1)\]

and

\[
dil(2,S2)\]
are used in the formulation of the formex variable G. The dilatation function
dil(2,S1)
effects the scaling of the first half (top half) of the dome of Fig 2.5.16 by the factor $S1 = 1.1$ and the dilatation function
dil(2,S2)
effects the scaling of the second half (bottom half) of the dome by the factor $S2 = 0.85$.

As the last example in this section, consider the double layer diamatic dome of Fig 2.5.19. The plan view of the dome is shown on the left side of the figure with the top layer elements shown by thick lines and the bottom layer elements as well as the web elements shown by thin lines. Plan views of the top layer elements, web elements and bottom layer elements are also shown separately on the right side of Fig 2.5.19, indicated by T, W and B, respectively.

A Formian scheme for the generation of the double layer diamatic dome of Fig 2.5.19 is shown in the editor display of Fig 2.5.20. The scheme is generic involving the following parameters:
- $Rt$ denotes the radius of the top circumsphere of the dome.
- $Rb$ denotes the radius of the bottom circumsphere of the dome.
- $m$ denotes the frequency of the top layer of the dome.
- $n$ denotes the number of sectors of the dome.
- $A$ denotes the sweep angle of the top layer of the dome.

\[
\begin{align*}
(*) \text{ Double layer diamatic dome (*)} \\
Rt=36; \quad (*) \text{ radius of top circumsphere (*)} \\
Rb=34.5; \quad (*) \text{ radius of bottom circumsphere (*)} \\
m=4; \quad (*) \text{ frequency of top layer (*)} \\
n=6; \quad (*) \text{ number of sectors (*)} \\
A=36; \quad (*) \text{ sweep angle of top layer (*)} \\
TOP=\text{genit}(1,m,3,3,0,1) \{[Rt,0;0,Rt,0,3],[Rt,0;Rt,3,3],[Rt,3;Rt,3,3]\}; \\
WEB=\text{genit}(1,m,3,3,0,1) \{[Rb,1;2,Rt,0,0],[Rb,1;2,Rt,0,3],[Rb,1,2;Rt,3,3]\}; \\
BOT=\text{genit}(1,m-1,3,3,0,1) \{[Rb,1,2;Rb,1,5],[Rb,2,4;Rb,2,4,Rt,3,6],[Rb,2,4;Rt,0,3],[Rb,2,4;Rt,3,3]\}; \\
F=\text{bd}(1,360/n,A/(3*m))\{TOP#WEB#BOT\}; \\
G=\text{pex} \{\text{rosad}(0,0,n,360/n)\} F; \\
\text{clear}; \\
\text{use \\&.vm(2)}; \\
\text{draw G}; \\
<><><<>
\end{align*}
\]

Throughout the present section, the circumradii and sweep angles have been used to control the proportions of the domes. However, this control may also be effected by specifying the span and the rise of a dome, see Section 1.9 of Ref 1. The relationships between the circumradius $R$, sweep angle $A$, span $S$ and rise $H$ of a dome are given by (see Fig 1.9.2 of Section 1.9 of Ref 1):

\[
\begin{align*}
R &= \frac{S}{2 \sin A} \\
A &= 2 \arctan \left(\frac{2H}{S}\right) \\
S &= 2R \sin A \\
H &= \left(\frac{S}{2}\right) \tan \left(\frac{A}{2}\right)
\end{align*}
\]

In the case of the diamatic dome of Fig 2.5.19, the span and the rise are obtained as:

\[
\begin{align*}
S &= 2 \times 36 \times \sin 36^\circ = 42.321 \text{ unit length} \\
H &= \left(\frac{S}{2}\right) \tan 18^\circ = 6.875 \text{ unit length}
\end{align*}
\]

### 2.6 INFORMATION EXPORT

The objective of this section is to discuss the export of information from Formian into graphics and structural analysis packages. The idea is introduced in terms of the example of the double layer grid whose plan and elevation together with a perspective view are shown in Fig 2.6.1. In this figure, the top layer elements of the grid are shown by thick lines and the bottom layer elements as well as the web elements are shown by thin lines. The grid covers a square area of 28 m by 28 m.
Hoshyar Nooshin and Peter Disney

The grid is supported at 20 nodal points along the perimeter of the bottom layer. The support positions are indicated by little circles on the plan of the grid in Fig 2.6.1. There are two types of supports. The supports indicated by solid circles are constrained in x, y and z directions and the supports indicated by hollow circles have a single constraint in the z direction.

The grid is to be analysed using a commercial structural analysis package for the following three loading cases:

- The first loading case consists of equal vertical point loads applied at all the top layer nodes of the grid.
- The second loading case consists of equal vertical point loads applied at all the nodes of the left half of the top layer of the grid.
- The third loading case consists of a single point load applied at the central node of the bottom layer of the grid.

A Formian scheme for the generation of data for the analysis of the grid of Fig 2.6.1 is shown in the editory display of Fig 2.6.2. In this scheme, the formex variables TOP, BOT and WEB represent the top layer elements, bottom layer elements and web elements of the grid, respectively. Also, the formex variable GRID represents all the elements of the grid. Due to the simplicity of the geometry, the formulations are carried out directly in terms of the x-y-z global coordinate system (that is, without the aid of a separate normat, compare with the example of Fig 1.7.2, Ref 1).

The support positions are represented by formex variables S1 and S2, where, S1 represents the supports that have constraints in the x, y and z directions and S2 represents the supports that have a single constraint in the z direction.

The load positions are represented by formex variables L1, L2 and L3, where, L1 represents the load positions for the first loading case and L2 and L3 represent the load positions for the second and third loading cases, respectively.
Every one of the formex variables created by the scheme of Fig 2.6.2 may be considered to be the 'name' of a 'file' that contains the information about a formex. The information contained in such a file me be 'reformatted' to suit a graphics or structural analysis package.

The clicking of the 'reformat button' will now place the reformatted file GRID.plt in the indicated folder (that is, projects) in the indicated drive (that is, drive C). If it is required to place the file in a different folder then, before clicking the reformat button, the symbol [...] in the area below the indicated folder should be clicked. This will cause the list of all the current folders to be displayed and the required folder may be chosen by clicking its name. Also, if required, the indicated drive can be changed using the 'drive box' shown in Fig 2.6.3.

The next step after reformating is to 'import' the reformatted file into CorelDRAW. This can be done through the 'import' menu item of the 'file menu' in CorelDRAW. The imported material will resemble the perspective view of the double layer grid shown in Fig 2.6.1 and may be treated as a CorelDRAW graphic object for any further desired treatment in CorelDRAW.

A similar procedure may be followed to reformat the formex file GRID.fmx for use in the draughting package AutoCad. In this case, the formatting style to be highlighted in the 'format box' of Fig 2.6.3 is 'AutoCad(*.dxf). The result of reformatting will be a file in the DXF format called GRID.dxf. Subsequently, the file GRID.dxf may be opened in AutoCad and be used as though it was an AutoCad file.

To insert a formex plot as a 'picture' into a word processing package such as 'Microsoft Word', the metafile format, with the associated file name extension 'wmf', is the most suitable one. It is to be noted that an HPGL file (that is, a file with the extension 'plt') reflects the current viewing particulars in the Formian environment at the moment of reformating and the same applies to a metafile but a DXF file is independent of the viewing particulars.

All the formex files created by the scheme of Fig 2.6.2 may be transformed into DXF files for use in the AutoCad environment. Also, commercial structural analysis packages normally accept DXF files (either directly or through AutoCad) for data input. One may then use a DXF file to send the geometric information about each group of entities (elements, support positions, load positions, ...) to a package. Each group of entities will then be associated with the appropriate attributes (cross-sectional and material properties, constraint particulars, load components, ...) inside the package.
APPENDIX

Basic Formex Functions

2.A.1 Introduction

This appendix contains a description of the basic formex functions. The formex functions described here are updated and extended versions of the functions discussed in Ref 10. However, the definitions of the formex functions here are given in a concise form and only cover the essential particulars of the functions.

An important feature of the present definitions of the formex functions is that the functions are allowed to involve noninteger formices. This is in contrast with the definitions in Ref 10 where the functions may only involve integer formices.

The reader is assumed to be thoroughly familiar with the basic ideas of formex configuration processing as described in Ref 1.

2.A.2 Transflection Functions

'Transflection functions' constitute a major family of formex functions. These functions effect

- translation,
- reflection,
- vertition (rotation),
- glissation (shearing),
- dilatation (increase or decrease in size) and
- projection.

A graphical illustration of transflectional effects is shown in Fig 2.A.1.

For a transflection function to have an 'undistorted' graphical effect, the coordinate system used for the graphical representation must be of a Cartesian-type with identical uniform graduations along its axes. However, reference systems that do not satisfy these requirements are frequently used for the graphical representation of transflectional effects. In these cases, a term such as 'translation' should be interpreted in a more general sense than that of a simple 'rigid body' translational movement.

For example, consider the two directional curvilinear normat shown by thin lines in Fig 2.A.2. Here, the simple L-shaped configuration denoted by C2 is the 'translation' by 4 divisions in the U1 direction of the configuration denoted by C1. In this situation, the sense of 'translation' is more general than that of the basic geometric notion of translation. To elaborate, the translation of C1 is guided by the forms and positions of the 'normat lines' and the resulting configuration C2 has a shape which is a 'distorted' form of the shape of C1.

Similarly, in the environment of the normat of Fig 2.A.2, the notion of 'reflection' should be interpreted in a more general sense than that of a 'simple mirror image'. For instance, the configuration denoted by C3 in Fig 2.A.2 is the 'reflection' in the U2 direction of C2 with respect to the normat line denoted by a.
Also, the configuration C4 is the reflection in the U1 direction of C3 with respect to the normat line denoted by \( b \).

It should be borne in mind that, although the names of the transflection functions are suggestive of the basic geometric notions of translation, reflection, projection, ..., the definitions of the 'rules' of the transflection functions are 'algebraic' rather than 'geometric'. Therefore, the result of the application of a transflection function is in a 'numerical form' (in terms of a formex) and the graphical representation of the result can assume many different geometric shapes depending on the reference system used.

There are three families of transflection functions, namely, cardinal functions, tendial functions and provial functions. These will be described below.

### 2.A.3 Cardinal Functions

The family of 'cardinal functions' consists of 9 functions that are briefly described in Table 2.A.1. Cardinal functions effect transflections with respect to the main directions (cardinal directions) of the reference system. The operation rules for the basic cardinal functions are described in Table 2.A.2.

The general form of the function designator for the first cardinal function, that is, translation function, may be written as

\[
\text{tran}(d,t) | E
\]

where
- \( \text{tran}(d,t) \) is the 'function',
- the symbol \( | \) is the 'rallus symbol',
- \( E \) is the 'argument' of the function,
- \( \text{tran} \) is the 'imprint' of the function and
- \( d \) and \( t \) are the 'canonic parameters' of the function.

The above terminology is used for all the formex functions. However, the number and types of canonic parameters vary from function to function as will be discussed in each case.

The examples in Table 2.A.1 contain a number of formex plots. For these plots, the convention is used where a prime is added to a formex variable to indicate its plot. Thus, the plot of a formex variable \( G \) is denoted by \( G' \).

Another convention that is employed is for the indication of 'optional parts'. The compound symbols \( [ \) and \( ] \) are used for this purpose. These compound symbols are referred to as 'option brackets'. For example, the presence of the last canonic parameter of the vertition function is optional and, therefore, the general form of the vertition function in Table 2.A.1 is given as

\[
\text{ver}(d1,d2,c1,c2 \text{ \{r\}})
\]

This implies that the vertition function is either of the form

\[
\text{ver}(d1,d2,c1,c2,r)
\]

or of the form

\[
\text{ver}(d1,d2,c1,c2)
\]

It should be mentioned that in the case of cardinal functions that involve two directions, namely, vertition, rosette and glissation functions, the directions are specified by \( d1 \) and \( d2 \) where the value of \( d1 \) may be smaller or greater than that of \( d2 \) but the values of \( d1 \) and \( d2 \) may not be equal.

The last column of Table 2.A.1 contains information about the 'types' of the canonic parameters of the functions in the context of the programming language Formian. The terms 'numeric expression' and 'integer expression' in the last column of Table 2.A.1 are 'Formian grammatical terms' and are briefly described below.

A 'numeric expression' is a meaningful 'evaluable' combination of
- numeric constants,
- numeric variables, that is, variables that represent numeric values,
- function designators that have numeric values,
- numeric operators, namely,
  + (for addition),
  - (for subtraction and negation),
  * (for multiplication),
  / (for division),
  \(^\wedge\) (for exponentiation) and
- parenthenses.

Examples of valid numeric expressions in the programming language Formian are

\[
4 \times \text{tic} \mid A
\]
\[
-5.12e3 + \sin \mid (B / 12) - \text{ran} \mid 1.6
\]
\[
8.63 \times 3.2 / (75 + C)
\]

where
- \( A, B \) and \( C \) are numeric variables,
- \( -5.12e3 \) is a floatal constant representing the value \(-5120.00\),
- \( 8.63 \times 3.2 \) represents 8.63 to the power of 3.2 and
- \( \text{tic}, \sin \) and \( \text{ran} \) are numeric functions, as described in Section 1.5.3 of Ref 1.

The first of the above examples is an 'integer expression' since its value is an integer number and
the second and third examples are ‘floatal expressions’ since their values are noninteger, as discussed in Section 1.5.6 of Ref 1.

A numeric operator or a rallus symbol may not be followed by a plus or minus sign. Thus

$$m^{-2.5} \text{ and } \sin \left| -30 \right|$$

are not acceptable. However, the following forms are acceptable

$$m^{(-2.5)} \text{ and } \sin \left| (-30) \right|$$

Formian has a convention that is referred to as the ‘near-integer convention’. The convention provides an interpretation of what may be regarded as an integer value. In order to explain this convention, consider the following examples of rindle and rosette functions

$$\text{rin}(1, m^{\cdot2}, 7.5) \text{ and } \text{ros}(2, 3, 8, 6, n/2, -36),$$

where it is known that \(m\) and \(n\) are integer variables whose values are 4 and 20, respectively.

Thus, the value of \(m^{\cdot2}\) is 16 and it should be acceptable as the number of replications in the above rindle function. Also, the value of \(n/2\) is 10 and it should be acceptable as the number of replications in the above rosette function.

However, there is a problem with both of the above examples. Namely, the exponentiation and division operators \(^{\cdot}\) and / in Formian, as in most other programming languages, always give rise to ‘floatal’ values irrespective of whether their operands have integer or noninteger values. The problem would, therefore, be that numbers for ‘counting’ are given in floating point form. To avoid this problem, the following convention is used.

**Near-Integer Convention:** In any position in a Formian statement where an item with an integer value is required to appear, an item with a ‘nearly integer’ value will be also acceptable. In this context, a floatal value \(V\) is considered to be ‘near enough’ to an integer value provided that

$$\text{abs} \left| (V - \text{ric} V) \right| < 1E^{-5}$$

Here, ‘abs’ is the ‘absolute value function’ and ‘ric’ is the ‘rounded integer conversion function’ (Section 1.5.3 of Ref 1). That is, \(V\) is considered to be ‘near enough’ to an integer value provided that the absolute value of the difference between \(V\) and its nearest integer is less than 0.00001.

Now, referring to the above mentioned examples, namely,

$$\text{rin}(1, m^{\cdot2}, 7.5) \text{ and } \text{ros}(2, 3, 8, 6, n/2, -36),$$

with the near-integer convention, the terms

$$m^{\cdot2} \text{ and } n/2$$

will not create any problems since they represent floatal values that are very close to integer values.

However, it is to be emphasised that the near-integer convention only applies to situations when an integer value is ‘required’ to appear. Thus, with the same values for the integer variables \(m\) and \(n\) as in the above examples, the Formian statements

$$x = m^{\cdot2};$$

and

$$g = [4,3,n; 2,1,n/2];$$

will give rise to a floatal variable \(x\) (rather than an integer variable) and a floatal formex variable \(g\) (rather than an integer formex variable).

**2.A.4 Tendial Functions**

‘Tendial functions’ constitute the second major group of transflection functions. These functions are divided into five families, namely,

- tendic functions,
- tendid functions,
- tendis functions,
- tendit functions and
- tendix functions.

‘Tendic functions’ are extended forms of cardinal functions that allow multiple operations, as described in Table 2.A.3. The imprint of a tendic function is obtained by adding the suffix ‘ic’ to the imprint of the corresponding cardinal function.

There are no tendic functions corresponding to the cardinal functions that involve rotation or shearing, namely, vertition, rosette and glissation functions. On the other hand, the family of tendic functions include a member, namely, ‘gena tendic function’ that has no cardinal counterpart.

A gena tendic function always operates along two directions. In contrast, the number of directions along which any other tendic function can operate is unlimited.

In Table 2.A.3, the example for the first tendic function, that is, tranic function involves the function \(\text{pan}(1,0)\). This is a ‘pansion function’ that has the effect of adding a zero, as the first uniple, to all the signets of its argument.
Table 2.A.1 Cardinal Functions

<table>
<thead>
<tr>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td>E = {[1,1.5; 1,1], [1,1; 2,1]}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Translation Function</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ‘translation function’ is of the form tran(d,t)</td>
<td>G1 = tran(1,2)</td>
</tr>
<tr>
<td>where d specifies the ‘direction of translation’ and t specifies the ‘amount of translation’.</td>
<td>and</td>
</tr>
<tr>
<td>G2 = tran(2,1)</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rindle Function</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ‘rindle function’ is of the form rin(d,n,p)</td>
<td>G = rin(1,3,1.5)</td>
</tr>
<tr>
<td>where the term ‘rindle’ implies ‘translational replication’ and where d specifies the ‘direction of replication’, n specifies the ‘number of replications’ and p specifies the ‘pace’, that is, the ‘amount of translation at each step’.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reflection Function</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ‘reflection function’ is of the form ref(d,r)</td>
<td>G1 = ref(1,3)</td>
</tr>
<tr>
<td>where d specifies the ‘direction of reflection’ and r specifies the ‘position of the plane of reflection’.</td>
<td>and</td>
</tr>
<tr>
<td>G2 = ref(2,2.5)</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lambda Function</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ‘lambda function’ is of the form lam(d,r)</td>
<td>G1 = lam(1,3)</td>
</tr>
<tr>
<td>where the term ‘lambda’ implies ‘combination of an object with its reflection’ and where d specifies the ‘direction of reflection’ and r specifies the ‘position of the plane of reflection’.</td>
<td>and</td>
</tr>
<tr>
<td>G2 = lam(2,2.5)</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertition Function</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ‘vertition function’ is of the form ver(d1,d2,c1,c2)</td>
<td>G1 = ver(1,2,1,5.2,5)</td>
</tr>
<tr>
<td>where the term ‘vertition’ implies ‘rotation’ and where d1 and d2 specify the ‘plane of rotation’, c1 and c2 specify the coordinates, in d1 and d2 directions, of the ‘centre of rotation’ and r specifies the ‘rotation’ in degrees. The sense of rotation is such that when r is positive then the rotation by r of d1-axis about the origin will move the positive side of d1-axis towards the positive side of d2-axis. The presence of r is optional and its absence implies that r is equal to 90°.</td>
<td>and</td>
</tr>
<tr>
<td>G2 = ver(1,2,3,2,135)</td>
<td>E</td>
</tr>
</tbody>
</table>
Table 2.A.1 Cardinal Functions (Continued)

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rosette Function</strong>&lt;br&gt;A 'rosette function' is of the form&lt;br&gt;( \text{ros}(d_1,d_2,c_1,c_2,{n,p}) )&lt;br&gt;where the term 'rosette' implies 'rotational replication' and where ( d_1 ) and ( d_2 ) specify the 'plane of rotation' and ( c_1 ) and ( c_2 ) specify the coordinates, in ( d_1 ) and ( d_2 ) directions, of the 'centre of rotation' and where ( n ) specifies the 'number of replications' and ( p ) specifies the 'pace' (that is, rotation at each step) in degrees. The presence of ( n ) and ( p ) is optional and their absence implies that ( n ) is equal to 4 and ( p ) is equal to 90°. The sense of rotation is as defined for the vertition function.</td>
<td>Example:&lt;br&gt;( G = \text{ros}(1,2,3,1,5,-45) \mid E )&lt;br&gt;<img src="image" alt="Diagram of rosette function" /></td>
<td>( d_1, d_2 ) and ( n ) are integer expressions.( c_1, c_2 ) and ( p ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Glissation Function</strong>&lt;br&gt;A 'glissation function' is of the form&lt;br&gt;( \text{glis}(d_1,d_2,n,g) )&lt;br&gt;where the term 'glissation' implies 'shearing' and where ( d_1 ) specifies the 'direction of glissation', ( d_1 ) and ( d_2 ) specify the 'plane of glissation', ( n ) specifies the position of the 'neutral line' (that is, the line whose points are not affected by the glissation) and ( g ) specifies the 'angle of glissation' in degrees. The neutral line is the locus of the 'glissation centres' for all the points of the configuration to be glissated. The positive sense of glissation is defined in a manner similar to that of rotation for the vertition function.</td>
<td>Example:&lt;br&gt;( G = \text{glis}(1,2,3,5,40) \mid E )&lt;br&gt;<img src="image" alt="Diagram of glissation function" /></td>
<td>( d_1 ) and ( d_2 ) are integer expressions.( n ) and ( g ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Dilatation Function</strong>&lt;br&gt;A 'dilatation function' is of the form&lt;br&gt;( \text{dil}(d,f) )&lt;br&gt;where the term 'dilatation' implies 'elongation or contraction' and where ( d ) specifies the 'direction of dilatation' and ( f ) specifies the 'factor of dilatation'.</td>
<td>Examples:&lt;br&gt;( G_1 = \text{dil}(1,2.5) \mid E )&lt;br&gt;( G_2 = \text{dil}(2,3) \mid E )&lt;br&gt;<img src="image" alt="Diagram of dilatation function" /></td>
<td>( d ) is an integer expression.( f ) is a numeric expression.</td>
</tr>
<tr>
<td><strong>Projection Function</strong>&lt;br&gt;A 'projection function' is of the form&lt;br&gt;( \text{proj}(d,p) )&lt;br&gt;where ( d ) specifies the 'direction of projection' and ( p ) specifies the 'position of the plane of projection'.</td>
<td>Examples:&lt;br&gt;( G_1 = \text{proj}(1,4) \mid E )&lt;br&gt;( G_2 = \text{proj}(2,3) \mid E )&lt;br&gt;<img src="image" alt="Diagram of projection function" /></td>
<td>( d ) is an integer expression.( p ) is a numeric expression.</td>
</tr>
</tbody>
</table>
### Table 2.A.2 Operation Rules for Basic Cardinal Functions

<table>
<thead>
<tr>
<th>Functions</th>
<th>Operation rules</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,U_d+t,...,U_k]$</td>
<td>Example: $G = \text{trans}(3,2,1.5,2.5) \mid \text{pan}(1.0)^* \mid E$</td>
</tr>
<tr>
<td>Reflection Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,(2r-U_d),...,U_k]$</td>
<td>Example: $G = \text{ref}(1,3,2,1.5,1) \mid \text{pan}(2.0)^* \mid E$</td>
</tr>
<tr>
<td>Vertition Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,V_1,...,V_2,...,U_k]$ where $V_1 = c_1 + (U_d - c_1) \cos r - (U_d - c_2) \sin r$ $V_2 = c_2 + (U_d - c_1) \sin r + (U_d - c_2) \cos r$</td>
<td></td>
</tr>
<tr>
<td>Glissation Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,V_1,...,V_2,...,U_k]$ where $V = U_{dl} + (n - U_{d2}) \tan g$</td>
<td></td>
</tr>
<tr>
<td>Dilatation Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,fU_d,...,U_k]$</td>
<td></td>
</tr>
<tr>
<td>Projection Function</td>
<td>If a typical signet of $E$ is $[U_1,U_2,...,U_d,...,U_k]$ then the corresponding signet of $G$ is obtained as $[U_1,U_2,...,p,...,U_k]$</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2.A.3 Tendic Functions

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tranic Function</strong></td>
<td></td>
<td>$d_1,d_2,...,d_h$ are integer expressions.</td>
</tr>
<tr>
<td>(Translation Tendic Function)</td>
<td></td>
<td>$t_1,t_2,...,t_h$ are numeric expressions.</td>
</tr>
<tr>
<td>A 'tranic function' is of the form $\text{tranic}(d_1,d_2,...,d_h,t_1,t_2,...,t_h)$ The effect of the function is equivalent to that of the composite translation function $\text{tran}(d_h,t_h) \mid ... \mid \text{tran}(d_2,t_2) \mid \text{tran}(d_1,t_1)$</td>
<td>Example: $G = \text{tranic}(3,2,1.5,2.5) \mid \text{pan}(1.0)^* \mid E$</td>
<td></td>
</tr>
<tr>
<td><strong>Rinic Function</strong></td>
<td></td>
<td>$d_1,d_2,...,d_h$ are integer expressions.</td>
</tr>
<tr>
<td>(Rindle Tendic Function)</td>
<td></td>
<td>$n_1,n_2,...,n_h$ are integer expressions.</td>
</tr>
<tr>
<td>A 'rinic function' is of the form $\text{rinic}(d_1,d_2,...,d_h,n_1,n_2,...,n_h,p_1,p_2,...,p_h)$ The effect of the function is equivalent to that of the composite rindle function $\text{rin}(d_h,n_h,p_h) \mid ... \mid \text{rin}(d_2,n_2,p_2) \mid \text{rin}(d_1,n_1,p_1)$</td>
<td>Example: $G = \text{rinic}(1,3,3,2,1.5,1) \mid \text{pan}(2.0)^* \mid E$</td>
<td></td>
</tr>
</tbody>
</table>

* See Section 2.A.13
Table 2.A.3 Tendic Functions (Continued)

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Refic Function</strong> (Reflection Tendic Function)</td>
<td>Example:</td>
<td>d1,d2, ..., dh are integer expressions.</td>
</tr>
<tr>
<td>A ‘refic function’ is of the form</td>
<td>$G = \text{refic}(2,1,2,3) \mid E$</td>
<td>r1,r2, ..., rh are numeric expressions.</td>
</tr>
<tr>
<td>$\text{refic}(d_1,d_2, ... ,d_h, r_1,r_2, ... ,r_h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The effect of the function is equivalent to that of the composite reflection function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{ref}(d_h,r_h)$ ... $\text{ref}(d_2,r_2)$ $\text{ref}(d_1,r_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Lamic Function</strong> (Lambda Tendic Function)</td>
<td>Example:</td>
<td>d1,d2, ..., dh are integer expressions.</td>
</tr>
<tr>
<td>A ‘lamic function’ is of the form</td>
<td>$G = \text{lamic}(2,1,2,3) \mid E$</td>
<td>r1,r2, ..., rh are numeric expressions.</td>
</tr>
<tr>
<td>$\text{lamic}(d_1,d_2, ... ,d_h, r_1, r_2, ... ,r_h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The effect of the function is equivalent to that of the composite lambda function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{lam}(d_h,r_h)$ ... $\text{lam}(d_2,r_2)$ $\text{lam}(d_1,r_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dilic Function</strong> (Dilatation Tendic Function)</td>
<td>Example:</td>
<td>d1,d2, ..., dh are integer expressions.</td>
</tr>
<tr>
<td>A ‘dilic function’ is of the form</td>
<td>$G = \text{dilic}(2,3,2,5,2) \mid \text{pan}(1,0)^* \mid E$</td>
<td>f1,f2, ..., fh are numeric expressions.</td>
</tr>
<tr>
<td>$\text{dilic}(d_1,d_2, ... ,d_h, f_1,f_2, ... ,f_h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The effect of the function is equivalent to that of the composite dilatation function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{dil}(d_h,f_h)$ ... $\text{dil}(d_2,f_2)$ $\text{dil}(d_1,f_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Projic Function</strong> (Projection Tendic Function)</td>
<td>Example:</td>
<td>d1,d2, ..., dh are integer expressions.</td>
</tr>
<tr>
<td>A ‘projic function’ is of the form</td>
<td>$G = \text{projic}(1,2,4,2,5) \mid E$</td>
<td>p1,p2, ..., ph are numeric expressions.</td>
</tr>
<tr>
<td>$\text{projic}(d_1,d_2, ... ,d_h, p_1,p_2, ... ,p_h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The effect of the function is equivalent to that of the composite projection function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{proj}(d_h,p_h)$ ... $\text{proj}(d_2,p_2)$ $\text{proj}(d_1,p_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Genic Function</strong> (Gena Tendic Function)</td>
<td>Example:</td>
<td>d1, d2, n1,n2 and t are integer expressions.</td>
</tr>
<tr>
<td>A ‘genic function’ is of the form</td>
<td>$G = \text{genic}(2,3,4,3,1,5,1,0,5,-1) \mid \text{pan}(1,0)^* \mid E$</td>
<td>p1, p2 and b are numeric expressions.</td>
</tr>
<tr>
<td>$\text{genic}(d_1,d_2,n_1,n_2,p_1,p_2,b,t)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The effect of a genic function is to create a non-rectangular array of objects. If $b=0$ and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t=0$ then the effect of the above genic function will be equivalent to that of the rinic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>function $\text{rinic}(d_1,d_2,n_1,n_2,p_1,p_2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The parameter $b$ specifies the ‘bias’, that is, the movement in the $d_1$ direction for every</td>
<td></td>
<td></td>
</tr>
<tr>
<td>step in the $d_2$ direction. The parameter $t$ specifies the ‘taper’, that is, the ‘increment’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in $n_1$ for every step in the $d_2$ direction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See Section 2.A.13
<table>
<thead>
<tr>
<th>Tendid Functions</th>
<th>Tendix Functions</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tranid Function</td>
<td>Tranix Function</td>
<td>(t_1, t_2 \text{ and } t_3 ) are numeric expressions.</td>
</tr>
<tr>
<td>(Translation Tendid Function) (\text{A 'tranid function' is of the form} ) (\text{tranid}(1, t_2)) (\text{This is equivalent to the composite function} ) (\text{trans}(2, t_2)</td>
<td>\text{tran}(1, t_1))</td>
<td>(Translation Tendix Function) (\text{A 'tranix function' is of the form} ) (\text{tranix}(t_1, t_2, t_3)) (\text{This is equivalent to the composite function} ) (\text{tran}(3, t_3)</td>
</tr>
<tr>
<td>Rinid Function</td>
<td>Rinix Function</td>
<td>(n_1, n_2 \text{ and } n_3 ) are integer expressions.</td>
</tr>
<tr>
<td>(Rinkle Tendid Function) (\text{A 'rinid function' is of the form} ) (\text{rinid}(n_1, n_2, p_1, p_2)) (\text{This is equivalent to the composite function} ) (\text{rin}(2, n_2, p_2)</td>
<td>\text{rin}(1, n_1, p_1))</td>
<td>(Rindle Tendix Function) (\text{A 'rinix function' is of the form} ) (\text{rinix}(n_1, n_2, n_3, p_1, p_2, p_3)) (\text{This is equivalent to the composite function} ) (\text{rin}(3, n_3, p_3)</td>
</tr>
<tr>
<td>Refid Function</td>
<td>Refix Function</td>
<td>(p_1, p_2 \text{ and } p_3 ) are numeric expressions.</td>
</tr>
<tr>
<td>(Reflection Tendid Function) (\text{A 'refid function' is of the form} ) (\text{refid}(r_1, r_2)) (\text{This is equivalent to the composite function} ) (\text{ref}(2, r_2)</td>
<td>\text{ref}(1, r_1))</td>
<td>(Reflection Tendix Function) (\text{A 'refix function' is of the form} ) (\text{refix}(r_1, r_2, r_3)) (\text{This is equivalent to the composite function} ) (\text{ref}(3, r_3)</td>
</tr>
<tr>
<td>Lamid Function</td>
<td>Lamix Function</td>
<td>(r_1, r_2 \text{ and } r_3 ) are numeric expressions.</td>
</tr>
<tr>
<td>(Lambda Tendid Function) (\text{A 'lamid function' is of the form} ) (\text{lamid}(r_1, r_2)) (\text{This is equivalent to the composite function} ) (\text{lam}(2, r_2)</td>
<td>\text{lam}(1, r_1))</td>
<td>(Lambda Tendix Function) (\text{A 'lamix function' is of the form} ) (\text{lamix}(r_1, r_2, r_3)) (\text{This is equivalent to the composite function} ) (\text{lam}(3, r_3)</td>
</tr>
<tr>
<td>Dilid Function</td>
<td>Dilix Function</td>
<td>(f_1, f_2 \text{ and } f_3 ) are numeric expressions.</td>
</tr>
<tr>
<td>(Dilatation Tendid Function) (\text{A 'dilid function' is of the form} ) (\text{dilid}(f_1, f_2)) (\text{This is equivalent to the composite function} ) (\text{dil}(2, f_2)</td>
<td>\text{dil}(1, f_1))</td>
<td>(Dilatation Tendix Function) (\text{A 'dilix function' is of the form} ) (\text{dilix}(f_1, f_2, f_3)) (\text{This is equivalent to the composite function} ) (\text{dil}(3, f_3)</td>
</tr>
<tr>
<td>Projid Function</td>
<td>Projix Function</td>
<td>(p_1, p_2 \text{ and } p_3 ) are numeric expressions.</td>
</tr>
<tr>
<td>(Projection Tendid Function) (\text{A 'projid function' is of the form} ) (\text{projid}(p_1, p_2)) (\text{This is equivalent to the composite function} ) (\text{proj}(2, p_2)</td>
<td>\text{proj}(1, p_1))</td>
<td>(Projection Tendix Function) (\text{A 'projix function' is of the form} ) (\text{projix}(p_1, p_2, p_3)) (\text{This is equivalent to the composite function} ) (\text{proj}(3, p_3)</td>
</tr>
<tr>
<td>Genid Function</td>
<td>NOT APPLICABLE</td>
<td>(n_1, n_2 \text{ and } t ) are integer expressions.</td>
</tr>
<tr>
<td>(Gena Tendid Function) (\text{A 'genid function' is of the form} ) (\text{genid}(n_1, n_2, p_1, p_2, b, t)) (\text{This is equivalent to the tendic function} ) (\text{genic}(1, 2, n_1, n_2, p_1, p_2, b, t))</td>
<td></td>
<td>(p_1, p_2 \text{ and } b ) are numeric expressions.</td>
</tr>
</tbody>
</table>
Thus, for
\[ E = \{(1,1.5; 1,1), (1,1; 2,1)\} \]
the formex represented by
\[ \text{pan}(1,0) | E \]
will be
\[ E = \{(0,1,1.5; 0,1,1), (0,1,1; 0,2,1)\} \]
The pansion function is also used in another 3 examples in Table 2.A.3 with the aim of making the argument \( E \) conformable for the intended operations, see Section 2.A.13.

'Tendid functions' are special cases of tendic functions and are described in Table 2.A.4. A tendid function effects operations in directions 1 and 2 and its imprint is obtained by adding the suffix 'id' to the imprint of the corresponding cardinal function.

The families of tendis and tendit functions are similar to those of the tendid functions. A 'tendis function' operates in directions 1 and 3 and its imprint has the suffix 'is'. A 'tendit function' operates in directions 2 and 3 and its imprint has the suffix 'it'.

Except for the directions of the operations, the particulars of the tendis and tendit functions are identical to those of the tendid functions. Therefore, the tendis and tendit functions are not separately described since their particulars may be deduced from those of the tendid functions in Table 2.A.4. For instance, from the description of the tranid function in Table 2.A.4, it may be deduced that the function
\[ \text{tran}(a,b) \]
is equivalent to the composite function
\[ \text{tran}(3,b) | \text{tran}(1,a) \]
Also, it may be deduced that the function
\[ \text{tran}(a,b) \]
is equivalent to the composite function
\[ \text{tran}(3,b) | \text{tran}(2,a) \]
'Tendix functions' are special cases of tendic functions for operations in directions 1, 2 and 3. These functions are described in Table 2.A.4. The imprint of a tendix function is obtained by adding the suffix 'ix' to the imprint of the corresponding cardinal function.

The gena functions have no counterpart in the tendix family since the effects of the gena functions are confined to two directions. The term 'gena function' is used to refer to a gena tendic, gena tendid, gena tendis or gena tendit function.

It should be noted that the order of operations in tendid, tendis, tendit and tendix functions is always such that the operation along a direction that has a 'smaller number' is carried out before the operation along a direction that has a 'greater number'. This rule does not apply to the tendic functions, where, the order of the operations is dictated by the 'specified' order of the directions. The order of operations for the tendid, tendis and tendit functions is illustrated in Fig 2.A.3.

2.A.5 Provial Functions

'Provial functions' are generalisations of cardinal functions that allow operation in any direction. There are four families of provial functions, namely,
- 'provial functions', for operations involving directions 1 and 2,
- 'provia functions', for operations involving directions 1 and 3,
- 'provat functions', for operations involving directions 2 and 3 and
- 'proviax functions', for operations involving directions 1, 2 and 3.

Provial and proviax functions are described in Table 2.A.5. The imprint of a provial function is obtained by adding the suffix 'ad' to the imprint of the corresponding cardinal function. The imprint of a proviax function is obtained in a similar manner using the suffix 'ax'.

The terms A1, A2, A3, B1, B2 and B3 that appear as canonic parameters of provial and proviax functions in Table 2.A.5 represent the coordinates of the end points of a vector AB, as shown in Figs 2.A.4 and
2.A.5. This vector is referred to as the 'direction vector'.

For every proviad function in Table 2.A.5 there is a corresponding 'proviat function' that involves directions 1 and 3 and uses the suffix 'at'. Also, for every proviad function in Table 2.A.5 there is a corresponding 'proviax function' that involves directions 2 and 3 and uses the suffix 'at'. Proviads and proviats functions are not separately described since their particulars may be deduced from those of the proviad functions in Table 2.A.5.

Table 2.A.5 Proviad and Proviat Functions

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 'tranad function' (translation proviad function) is of the form</td>
<td>Examples:</td>
<td>A1, A2, A3, B1, B2, B3 and t are numeric expressions.</td>
</tr>
<tr>
<td>tranad(A1, A2, B1, B2, t) and a 'tranax function' (translation proviab function) is of the form</td>
<td>G1 = tranad(1,1,3,2.5)</td>
<td>E</td>
</tr>
<tr>
<td>tranax(A1, A2, A3, B1, B2, B3, t)</td>
<td>G2 = tranad(1,1,5.5,2)</td>
<td>E</td>
</tr>
<tr>
<td>where the direction of translation is given by vector AB and the amount of translation is</td>
<td></td>
<td>A1, A2, A3, B1, B2, B3 and t are numeric expressions.</td>
</tr>
<tr>
<td>given by t or the length of AB, in the absence of t.</td>
<td></td>
<td>A1, A2, A3, B1, B2, B3 and p are numeric expressions.</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------------------------</td>
<td>Example:</td>
<td>n is an integer expression.</td>
</tr>
<tr>
<td>A 'rinad function' (rindle proviad function) is of the form</td>
<td>G = rinad(1,1,2.5,1.3,4)</td>
<td>E</td>
</tr>
<tr>
<td>rinad(A1, A2, A3, B1, B2, n) and a 'rinax function' (rindle proviab function) is of the form</td>
<td></td>
<td>A1, A2, A3, B1, B2, B3 and p are numeric expressions.</td>
</tr>
<tr>
<td>rinax(A1, A2, A3, B1, B2, B3, n)</td>
<td></td>
<td>n is an integer expression.</td>
</tr>
<tr>
<td>where the direction of replication is given by vector AB, the number of replications is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>given by n and the amount of translation at each step is given by p or the length of AB,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in the absence of p.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Table 2.A.5 Proviad and Proviax Functions (Continued)**

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 'refad function' (reflection proviad function) is of the form refad(A1,A2,B1,B2),</td>
<td>Examples: G1 = refad(1,0,2.75,1)</td>
<td>A1, A2, A3, B1, B2 and B3 are numeric expressions.</td>
</tr>
<tr>
<td>a 'lamad function' (lambda proviad function) is of the form lamad(A1,A2,B1,B2),</td>
<td>and G2 = lamad(1,0,2.75,1)</td>
<td></td>
</tr>
<tr>
<td>a 'refax function' (reflection proviix function) is of the form refax(A1,A2,A3,B1,B2,B3) and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a 'lamax function' (lambda proviix function) is of the form lamax(A1,A2,A3,B1,B2,B3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the direction of reflection is given by vector AB, with the plane of reflection being perpendicular to AB at B.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'verad function' (vertition proviad function) is of the form</td>
<td></td>
<td>c1, c2 and r are numeric expressions.</td>
</tr>
<tr>
<td>verad(c1,c2,-r-)</td>
<td>Examples: G1 = verad(1.5,2.5)</td>
<td></td>
</tr>
<tr>
<td>where the coordinates of the centre of rotation in plane U1-U2 are given by c1 and c2, the amount of rotation is given by r (in degrees), the sense of rotation is as explained for the cardinal vertition function in Table 2.A.1 and where the absence of r implies that r is equal to 90°.</td>
<td>and G2 = verad(3,2,135)</td>
<td></td>
</tr>
<tr>
<td>A 'verax function' (vertition proviix function) is of the form</td>
<td>Example: G = verax(2,2,2,0,0,0,0,50)</td>
<td>A1, A2, A3, B1, B2, B3 and r are numeric expressions.</td>
</tr>
<tr>
<td>verax(A1,A2,A3,B1,B2,B3 -[r]-)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the axis of rotation is given by vector AB and the amount of rotation is given by r (in degrees). The sense of rotation is such that when the value of r is positive then the rotation causes a right-handed screw to move from A towards B. The absence of r implies that r is equal to 90°.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'rosad function' (rosette proviad function) is of the form</td>
<td>Example: G = rosad(3,1,5,,-45)</td>
<td>c1, c2 and p are numeric expressions.</td>
</tr>
<tr>
<td>rosad(c1,c2,-[n,p]-)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the coordinates of the centre of rotation in plane U1-U2 are given by c1 and c2, the number of replications is given by n, the amount of rotation at each step (pace) is given by p (in degrees), the sense of rotation is as explained for the cardinal vertition function in Table 2.A.1 and the absence of n and p implies that n is equal to 4 and p is equal to 90°.</td>
<td></td>
<td>n is an integer expression.</td>
</tr>
</tbody>
</table>
Table 2.A.5 Proviad and Proviap Functions (Continued)

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 'rosax function' (rosette proviap function) is of the form</td>
<td></td>
<td>A1, A2, B1, B2 and B3 are numeric expressions.</td>
</tr>
<tr>
<td>rosax(A1, A2, A3, B1, B2, B3 <del>[n,p]</del>)</td>
<td>Example: G = rosax(2,2,2,0,0,7,25)</td>
<td>n is an integer expression.</td>
</tr>
<tr>
<td>where the axis of rotation is given by vector AB, the number of replications is given by n, the amount of rotation at each step (pace) is given by p (in degrees) and the sense of rotation is as described for the verax function. The absence of n and p implies that n is equal to 4 and p is equal to 90°.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'glisad function' (glissation proviad function) is of the form</td>
<td>Example: G = glisad(0,2,5,4,4,45)</td>
<td>A1, A2, B1, B2 and g are numeric expressions.</td>
</tr>
<tr>
<td>glisad(A1, A2, B1, B2, g)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the direction of glissation as well as the neutral line is given by vector AB and the angle of glissation is given by g in degrees. The positive sense of glissation is as described for the glissax function in Table 2.A.1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'glisax function' (glissation proviap function) is of the form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>glisax(A1, A2, A3, B1, B2, B3, C1, C2, C3, g)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where A1, A2, A3, B1, B2, B3, C1, C2 and C3 specify the coordinates of three noncollinear points A, B and C and where g specifies the 'glissation angle' in degrees. The points A, B and C define a plane that is referred to as the 'base plane'. For a point P whose glissation is Q, the 'glissation plane' is the plane that contains point P and is parallel to the base plane, and the 'glissation direction' as well as the 'neutral line' is given by the vector DE which is the projection of AB onto the glissation plane. The positive sense of glissation is as indicated by the 'broken vector' ABC. All the canonic parameters of a glisax function are numeric expressions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'dilad function' (dilatation proviad function) is of the form</td>
<td>Example: G = dilad(0,1.5,1.5,2.5,2)</td>
<td>A1, A2, A3, B1, B2, B3 and f are numeric expressions.</td>
</tr>
<tr>
<td>dilad(A1, A2, B1, B2, f) and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a 'dilax function' (dilatation proviap function) is of the form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dilax(A1, A2, A3, B1, B2, B3, f)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the direction of dilatation is given by vector AB, with point A acting as the 'origin' and where the factor of dilatation is given by f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A 'projad function' (projection proviad function) is of the form</td>
<td>Example: G = projad(1,0,3,5,1)</td>
<td>A1, A2, A3, B1, B2 and B3 are numeric expressions.</td>
</tr>
<tr>
<td>projad(A1, A2, B1, B2) and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a 'projax function' (projection proviap function) is of the form</td>
<td></td>
<td></td>
</tr>
<tr>
<td>projax(A1, A2, A3, B1, B2, B3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>where the direction of projection is given by vector AB, with the plane of projection being perpendicular to AB at B.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For example, consider the formex variable
\[ F = [0,0,0; 1,1,1; 1,1,2] \]
This is a formex representing a triangular 'tile' a plot of which is shown in Fig 2.A.6, denoted by \( F' \). The equation
\[ G_{12} = \text{tranad}(0,0,1.5,1.5) \mid F \]
will transform \( F \) into the formex variable \( G_{12} \) whose plot is shown in Fig 2.A.6 as \( G_{12}' \). The above equation uses the tranad function acting in plane 1-2 with the direction vector being indicated by the dotted vector AB. The equation
\[ G_{13} = \text{tranas}(0,0,2.5,1.5) \mid F \]
will work in a similar way using the tranas function acting in the 1-3 plane. A plot of the resulting formex is shown in Fig 2.A.6, denoted by \( G_{13}' \). Similarly, the equation
\[ G_{23} = \text{tranat}(0,0,1.5,1.5) \mid F \]
will give rise to the formex variable \( G_{23} \) whose plot is shown as \( G_{23}' \) in Fig 2.A.6.

It should be pointed out that the grade of the argument of a proviad function must be greater than or equal to 2 but as far as a provias, proviat or proviax function is concerned, the grade of its argument must be greater than or equal to 3.

2.A.6 Introflection Functions

'Introflection functions' are formex functions that effect curtailment of formices in various ways. There are a number of introflection functions as follows:
- pexum function,
- family of rendition functions and
g- relection functions
These functions are briefly described below.

2.A.7 Pexum Function

The effect of the 'pexum function' is to transform a formex \( E \) into another formex by deleting every cantle of \( E \) that has the same signets as a preceding cantle of \( E \). The pexum function has no canonical parameter and its imprint is 'pex'. For example, if \( E \) is given by
\[ \{[4,3; 1,2], [5,4; 2,3], [1,2; 4,3], [4,3; 5,4], [4,3; 1,2] \} \]
then the value of the function designator pex \mid E is
\[ \{[4,3; 1,2], [5,4; 2,3], [4,3; 5,4] \} \]
Here, the third and fifth cantles of \( E \) are deleted because their signets are the same as those of the first cantle.

Note that a cantle that has the same signets as a preceding cantle is deleted irrespective of the order in which the signets appear in the cantles. Thus, in the above example, the third cantle of \( E \) is deleted even though its signets are not in the same order as those of the first cantle.

The role of the pexum function is to remove the superfluous repeated elements of a configuration. The pexum function has been used in a number of examples throughout the discourse. In particular, see Section 1.4.5 of Ref 1.

2.A.8 Rendition Functions

The family of 'rendition functions' consists of six functions, as described in Table 2.A.6. The description of the functions in Table 2.A.6 are given in two different forms. In the left-hand column, the descriptions are in terms of numerical procedures and in the right-hand column the descriptions are in terms of graphical effects.

A rendition function has a single canonic parameter. In the context of 'pure' formex algebra, all that one needs to say about this canonic parameter is that it is a formex. However, in the context of the programming language Formian, one has to be more specific about the type of the parameter. In Formian, the canonic parameter of a rendition function is required to be a formex expression.

A 'formex expression' is a meaningful 'evaluable' combination of
### Table 2.A.6 Rendition Functions

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Luxum Function</strong>&lt;br&gt;A ‘luxum function’ is of the form&lt;br&gt;lux(F)&lt;br&gt;where the term ‘luxum’ implies ‘disconnected parts’ and where F is a formex expression. If G = lux(F)</td>
<td>The plot of lux(F)</td>
</tr>
<tr>
<td><strong>Coluxum Function</strong>&lt;br&gt;A ‘coluxum function’ is of the form&lt;br&gt;col(F)&lt;br&gt;where F is a formex expression. If G = col(F)</td>
<td>The plot of col(F)</td>
</tr>
<tr>
<td><strong>Nexum Function</strong>&lt;br&gt;A ‘nexum function’ is of the form&lt;br&gt;nex(F)&lt;br&gt;where the term ‘nexum’ implies ‘connected parts’ and where F is a formex expression. If G = nex(F)</td>
<td>The plot of nex(F)</td>
</tr>
<tr>
<td><strong>Conexum Function</strong>&lt;br&gt;A ‘conexum function’ is of the form&lt;br&gt;con(F)&lt;br&gt;where F is a formex expression. If G = con(F)</td>
<td>The plot of con(F)</td>
</tr>
<tr>
<td><strong>Pactum Function</strong>&lt;br&gt;A ‘pactum function’ is of the form&lt;br&gt;pac(F)&lt;br&gt;where the term ‘pactum’ implies ‘matching parts’ and where F is a formex expression. If G = pac(F)</td>
<td>The plot of pac(F)</td>
</tr>
<tr>
<td><strong>Copactum Function</strong>&lt;br&gt;A ‘copactum function’ is of the form&lt;br&gt;cop(F)&lt;br&gt;where F is a formex expression. If G = cop(F)</td>
<td>The plot of cop(F)</td>
</tr>
</tbody>
</table>
• formex constants,
• formex variables,
• formex function designators,
• duplus symbols,
• parentheses and
• formex formations.

A ‘formex formation’ is a construct that has the same basic form as a formex constant and in which one or more uniples are given as numeric expressions. For example, if $i, j$ and $k$ are numeric variables, then

$$[3, i+k; 2, j]$$

and

$$[[1,4,3], [7,2,\text{ric}(k/5)], [j^3.5,-6,11]]$$

are formex formations.

Note that a single formex constant, formex variable, formex function designator or formex formation is counted as a formex expression.

Returning to the description of rendition functions, the effect of a rendition function is to ‘curtail’ its argument, as dictated by the signets or cantles of the value of its canonic parameter. The first four rendition functions, namely, luxum, coluxum, nexum and conexum functions use the information provided by their canonic parameters as ‘lists of signets’, irrespective of the manner in which the signets are grouped into cantles. In contrast, the last two rendition functions, namely, pactum and copactum functions work in terms of the ‘cantles’ of their canonic parameters.

Rendition functions may be subdivided into three ‘complementary pairs’. To elaborate, the luxum and coluxum functions form a ‘complementary pair’ because their effects are of a complementary nature. In fact, the name ‘coluxum’ implies the ‘complement’ of the luxum function. Similarly, the nexum and conexum functions constitute a complementary pair of functions and so do the pactum and copactum functions.

As a consequence of the complementary nature of the effects of the luxum and coluxum functions, the combination of the plots of

$$\text{lux}(F) \mid E \quad \text{and} \quad \text{col}(F) \mid E$$

will give rise to the plot of $E$, as may be seen from the plots given in Table 2.A.6. This, in fact, is a general rule that applies whatever the values of $E$ and $F$. This rule is also applicable in the case of the complementary function pairs ‘nexum/conexum’ and ‘pactum/copactum’.

2.A.9 Restition Functions

The family of ‘restition functions’ consists of six functions, as described in Table 2.A.7. The term ‘restition’ implies ‘cordoning’ (from the Latin word ‘restis’ meaning a cord). A restition function effects curtailment of a formex as guided by a ‘region’. An example of a region is shown in Fig 2.A.8. Also shown in this figure is a diagonal grid together with a two directional normat.

The formex representing the diagonal grid of Fig 2.A.8 is the same as formex $E$ formulated for the grid of Fig 2.A.7.

The border of the region in Fig 2.A.8 may be specified by the formex

$$b = [4,8; 14,8; 14,2,5; 9,5; 2,5]$$

The border is specified by listing the signets that represent the corners of the region. A formex such as $b$ above is referred to as a ‘border specifier’.

The elements of a configuration may be divided into three different types, depending on their dispositions with respect to a region, as follows:
• ‘interior element’ whose nodes are inside or on the border of the region,
• ‘exterior element’ whose nodes are outside or on the border of the region with at least one node outside the region and
• ‘border element’ which has at least one node inside and at least one node outside the region.

Examples of these types of elements are shown in Fig 2.A.8. Here, the interior, exterior and border elements are exemplified by ‘two-noded’ elements but the above definitions of interior, exterior and border elements apply to elements with any number of nodes.

If E is a formex representing a configuration and R is a region then a cantle of E is said to be
• an ‘interior cantle’ with respect to R,
• an ‘exterior cantle’ with respect to R or
• a ‘border cantle’ with respect to R depending on the disposition of the plot of the cantle with respect to the region R.

The region shown in Fig 2.A.8 is an example of an ‘ambit region’. The border of an ambit region is defined by specifying the coordinates of its corner points. However, there are two other types of regions, namely, ‘rectangular’ and ‘circular’ regions. The border of a rectangular region is defined by specifying the coordinates of two diagonally opposite corners. The border of a circular region is defined by specifying
• the coordinates of its centre and a point on its circumference or
• the coordinates of its centre and its radius.

A rectangular region may also be specified as an ambit region by giving the coordinates of all four corners but the specification as a ‘rectangular region’ is, obviously, more convenient.

To define a region, in addition to its border specifier, it is necessary to provide information about the type and the directions associated with the region. As far as the type of a region is concerned, the ‘region codes’ 1, 2 and 3 are used to specify ambit, rectangular and circular regions, respectively. The reason for the need to specify the directions associated with a region is that regions are not only used in the 1-2 plane, as exemplified so far, but also in planes 1-3 or 2-3 or indeed any other pair of directions.

The type and the associated directions of a region are specified by a construct of the form
\[ \{c,d_1,d_2\} \]
where c is the region code and d1 and d2 are the directions associated with the region. This construct is referred to as the ‘signature’ of the region. For example, the signature of the ambit region of Fig 2.A.8 may be given as
\[ \{1,1,2\} \]

A region may consist of a combination of regions. For instance, the region whose boundaries are shown by thick lines in Fig 2.A.10 consists of two rectangular regions and a circular region. This ‘compound region’ may be specified by listing the signatures and border specifiers of its constituent regions one after the other, as follows:
\[ \{2,2,3\}, \{1,1; 2.5,2.5\}, \{2,2,3\}, \{2,2; 3.5,3.5\}, \{3,2,3\}, \{3.5,3.5,1\} \]

When a signature in the specification of a compound region is the same as its preceding one then it may be omitted. Thus, the above specification may, alternatively, be written as
\[ \{2,2,3\}, \{1,1; 2.5,2.5\}, \{2,2; 3.5,3.5\}, \{3,2,3\}, \{3.5,3.5,1\} \]

A region may have a ‘negating’ effect. For instance, the compound region whose boundaries are shown by thick lines in Fig 2.A.10b consists of two ‘normal’ rectangular regions and a ‘negating’ circular region. A region with a ‘negating’ effect is referred to as an ‘antiregion’. The specification of an antiregion is identical to that of a normal region except for its region code which is given with a minus sign. For example, the specification of the compound region of Fig 2.A.10b may be given as
\[ \{2,1,3\}, \{1,1; 2.5,2.5\}, \{2,2; 3.5,3.5\}, \{3.5,3.5,1\} \]

Thus, the ‘border specifier’ for the rectangular region shown in Fig 2.A.9 may be given by
\[ b = [p_1,q_1; p_2,q_2] \]
Also, the border specifier for the circular region shown in Fig 2.A.9 may be given by
\[ b = [p_1,q_1; p_2,q_2] \]
Also, the specification of the compound region of Fig 2.A.10c may be written as
\[\{-2,1,2\}, [1,1; 2.5,2.5], \{2,1,2\}, \{2; 3.5,3.5\}, \{3,1,2\}, [3.5,3.5,1]\]

An antiregion has no effect by itself unless it overlaps a normal region.

A region need not necessarily be ‘in one piece’. For example, the compound region of Fig 2.A.10d consists of two separate sections as shown by thick lines. This compound region is obtained as a combination of three simple regions one of which is an antiregion. The specification of the ‘disjointed’ region of Fig 2.A.10d may be written as
\[\{2,1,2\}, [1,1; 2.5,2.5], \{-2,1,2\}, \{2,2; 3.5,3.5\}, \{3,1,2\}, [3.5,3.5,1]\]

Incidentally, it is interesting to note that a two-noded element such as e in Fig 2.A.10d will be regarded as an interior element since both of its nodes are inside the region.

In general, the specification of a compound region may be written as

\[s_1,b_1, s_2,b_2, ... , s_k,b_k \quad (k \geq 1)\]

where \(s_1, s_2, ... , s_k\) are region signatures and \(b_1, b_2, ... , b_k\) are their corresponding border specifiers and where it is understood that a signature that is the same as its preceding one can be omitted.

In the specification of a compound region, the ‘order’ in which the simple regions are listed has no particular significance and the shape of the resulting region will not be affected by this order.

The above general form of the specification of a compound region is used for the canonic parameters of the restition functions in Table 2.A.7.

The family of restition functions consists of three pairs of complementary functions, namely,
- ducture and coducture functions,
- juncture and cojuncture functions and
- vecture and covecture functions.

The term ‘ducture’ implies the ‘parts outside the region’, the term ‘juncture’ implies the ‘parts inside the region’ and the term ‘vecture’ implies the ‘parts cut by the region’.

The first four restition functions, namely, ducture, coducture, juncture and cojuncture functions are applicable to configurations with elements having any number of nodes. However, the last two restition functions, namely, vecture and covecture functions, are only relevant in relation to configurations that involve two-noded elements.

Vecture and covecture functions use the concepts of ‘subcantles’ and ‘subelements’. These concepts are explained with the aid of the example shown in Fig 2.A.11. In this figure, a region \(R\) is shown together with a two-noded border element 1-2. This element is the plot of the cantle
\[[p_1,q_1; p_2,q_2]\]

Now, the ‘interior part’ of element 1-2 may be represented by the cantle
\[[p_1,q_1; p_3,q_3]\]

where \(p_3\) and \(q_3\) are the coordinates of the ‘border point’ of the element. The above cantle is referred to
Table 2.A.7 Restitution Functions

<table>
<thead>
<tr>
<th>Brief descriptions of functions</th>
<th>Examples</th>
<th>Types of canonic parameters in Formian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ducture Function</strong></td>
<td>![Plot of duc(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'ducture' function is of the form</td>
<td>$\text{duc}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is not an exterior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Coducture Function</strong></td>
<td>![Plot of cod(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'coducture' function is of the form</td>
<td>$\text{cod}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is an exterior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Juncture Function</strong></td>
<td>![Plot of jun(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'juncture' function is of the form</td>
<td>$\text{jun}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is an interior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cojuncture Function</strong></td>
<td>![Plot of coj(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'cojuncture' function is of the form</td>
<td>$\text{coj}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is an interior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$ and by replacing every cantle of its argument that is a border cantle by its interior subcantle.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vecture Function</strong></td>
<td>![Plot of vec(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'vecture' function is of the form</td>
<td>$\text{vec}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is an exterior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$ and by replacing every cantle of its argument that is a border cantle by its interior subcantle.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Covecture Function</strong></td>
<td>![Plot of cov(s,b)</td>
<td>E*](#)</td>
</tr>
<tr>
<td>A 'covecture' function is of the form</td>
<td>$\text{cov}(s_1, b_1, s_2, b_2, \ldots, s_k, b_k)$ $k \geq 1$</td>
<td></td>
</tr>
<tr>
<td>The effect of the function is to produce a formex by deleting every cantle of its argument that is an interior cantle of the region represented by $s_1, b_1, s_2, b_2, \ldots, s_k, b_k$ and by replacing every cantle of its argument that is a border cantle by its exterior subcantle.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* $E$ is the formex whose plot is given in Fig 2.A.8 and $s$ and $b$ are, respectively, the signature and border specifier of the region shown in Fig 2.A.8. ** For an element that is not two-noded, vecture and covecture functions act like juncture and cojuncture functions, respectively.
as the 'interior subcantle' of element 1-2. Also, the part 1-3, considered as a separate element, is referred to as the 'interior subelement' of element 1-2.

Similarly, the cantle

\[ [p3,q3; p2,q2] \]

is referred to as the 'exterior subcantle' of element 1-2 and the part 3-2, considered as a separate element, is referred to as the 'exterior subelement' of element 1-2. The concepts of 'subcantles' and 'subelements' only apply to two-noded elements.

Note that the normat directions in Fig 2.A.11 are given as \( U_i \) and \( U_j \) implying that the concept discussed is applicable in relation to any pair of directions.

It is now necessary to bring out a number of points relating to regions. To begin with, it should be understood that an ambit region is always a 'convex' polygon. This convex polygon is obtained by examining the line passing through every pair of points listed in the border specifier of the region. Such a line will be considered to be an edge of the polygon provided that all the listed points are to 'one side' of the line (except for two or more of the points that are on the line).

For example, consider the ambit region shown in Fig 2.A.12. The signature of this region is

\{1,Ui,Uj\}

and its border specifier contains the coordinates of nine points, numbered 1 to 9 in Fig 2.A.12. It can be noticed that not all the specified points are corners of the region. The points that fall within the polygon, namely, points 1, 6 and 9 are then 'superfluous' and will be disregarded. It should also be noted that the 'order' in which the points are listed in the border specifier of an ambit region has no significance and will not affect the shape of the region.

The basic simple regions, namely, ambit, rectangular and circular regions are all 'convex'. However, compound regions that consist of combinations of simple regions may be used to create 'non-convex' regional shapes, as may be seen from the examples shown in Fig 2.A.10.

In general, the border specifier of an ambit region is of the form

\[ [p_1,q_1; p_2,q_2; \ldots ; p_r,q_r] \]

where \((p_1,q_1), (p_2,q_2), \ldots , (p_r,q_r)\) are the coordinates of \( r \) points and where \( r \geq 3 \). If all the \( r \) points are coincident or collinear then the region is a 'null region'. Also, the following regions are regarded as null regions:

- a circular region with a zero radius and
- a rectangular region whose border specifier contains the coordinates of two coincident points or two points that give rise to a degenerate rectangle consisting of a line segment.

A null region is considered to have no interior. Therefore, with respect to a null region, any element (cantle) is an exterior element (cantle).

Now, consider a cantle \( C \) and let this cantle be represented by

\[ \{U_{11}, U_{12}, \ldots , U_{1i}, \ldots , U_{1j}, \ldots , U_{1m}; U_{21}, U_{22}, \ldots , U_{2i}, \ldots , U_{2j}, \ldots , U_{2m}; \ldots \ldots \ldots \}

\[ U_{n1}, U_{n2}, \ldots , U_{ni}, \ldots , U_{nj}, \ldots , U_{nm}\] Let it be required to determine the 'status' of this cantle with respect to a region \( R \) whose directions are \( i \) and \( j \). What is meant by determining the status of the cantle, is to find out if the cantle is an interior, an exterior or a border cantle.
be effectively like an infinitely long prism a part of which is shown in Fig 2.A.13b.

Therefore, although as (a part of) the canonic parameter of a refection function, a simple region is always defined with respect to two directions, the 'effective directions' of the region will consist of all the directions associated with the argument of the function.

Finally, it should be mentioned that the concept of regions in this section has been described in terms of Cartesian-type normats. However, the idea of a region may be employed in relation to any kind of normat. With normats that are not of Cartesian-type, the borders of regions will assume shapes that are conformable with the forms of the normat lines and surfaces. For instance, in a spherical normat, the borders of a rectangular region with directions 2 and 3 will be along the parallels and meridians of the normat.

2.A.10 Relection Function

A 'relection function' effects curtailment of its argument, as dictated by a 'condition'. For example, consider the configuration shown in Fig 2.A.14a. The configuration consists of an arrangement of two-noded elements. A formex representing the configuration relative to the normat of Fig 2.A.14a is given by

\[ F_1 = \text{rinid}(7,5,1,1) | \{[1,1; 2,1], [1,1; 1,2]\} \]

Now, suppose that it is required to remove the elements sticking out on the right-hand side of the configuration of Fig 2.A.14a. This can be achieved by writing

\[ F_2 = \text{rel}(U(2,1) < 7.5) | F_1 \]

where \( F_2 \) represents the configuration shown in Fig 2.A.14b.

In the above equation, the construct

\[ \text{rel}(U(2,1) < 7.5) \]

is a 'relection function' where 'rel' is the imprint of the function, standing for 'relection', and

\[ U(2,1) < 7.5 \]

is the canonic parameter of the function. The term 'relection' is a Latin based word meaning 're-selection'.

The canonic parameter of a relection function is a 'condition' relating to the cantles of the argument of the function. The effect of the function is to 'keep' the cantles that satisfy the condition and 'delete' the cantles that do not satisfy the condition. For instance, for a cantle of the argument of a relection function, the relation

\[ U(2,1) < 7.5 \]

is TRUE if the 1st uniple of the 2nd signet of the cantle is less than 7.5 and is FALSE otherwise.

The construct

\[ U(2,1) \]

is referred to as a 'brevant'. The term 'brevant' means a 'shorthand indicator' (from Latin 'brevis' meaning short).

![Fig 2.A.14 Examples of the application of the relection function](image)

The general form of a brevant is

\[ \text{U}(i,j) \]

representing the jth uniple of the ith signet of a cantle. In the context of the programming language Formian, \( i \) and \( j \) are integer expressions, as explained in Section 2.A.3.

A relation appearing as the canonic parameter of a relection function is of the general form

\[ E \quad R \quad E \]
where \( E \) is a numeric expression that may include one or more brevants and where \( R \) is a 'relational operator'.

Formian has six relational operators, as shown in Table 2.A.8.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>( \geq )</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>( \Rightarrow )</td>
<td>Equal to</td>
</tr>
<tr>
<td>( \neq )</td>
<td>Not equal to</td>
</tr>
<tr>
<td>( \leq )</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
</tbody>
</table>

Returning to the examples of Fig 2.A.14, the configuration in Fig 2.A.14c may be represented by

\[
F_3 = \text{rel}(U(2,2) < 5.5) \mid F_2
\]

Here, the effect of the relection function

\[
\text{rel}(U(2,2) < 5.5)
\]

is to produce formex \( F_3 \) from those cantles of \( F_2 \) that satisfy the condition

\[
U(2,2) < 5.5
\]

That is, \( F_3 \) will be created from all the cantles of \( F_2 \) in which the \( 2^{nd} \) uniple of the \( 2^{nd} \) signet is less than 5.5. The effect will be to remove the cantles that represent the top row of the vertical elements in Fig 2.A.14b, as shown in Fig 2.A.14c.

The configuration of Fig 2.A.14c may also be represented by

\[
F_3 = \text{rel}(U(2,1) < 7.5 \&\& U(2,2) < 5.5) \mid F_1
\]

as shown in Fig 2.A.14d. In this case, the above discussed relations are combined using the compound symbol \( \&\& \) which is the 'logical AND operator' in Formian.

Formian has another logical operator, namely, the 'OR operator' which is denoted by the compound symbol \( | \). The use of the OR operator is exemplified in Figs 2.A.14e and 2.A.14f.

In general, the canonic parameter of a relection function is a 'perdicant', where a 'perdicant' is defined as any meaningful evaluable combination of

- relations that may include brevants,
- logical operators and
- parentheses.

### 2.A.11 Elementary Retronorms

The term 'retronormic function', or 'retronorm', is used to refer to a function that transforms the 'normat coordinates' of a configuration into 'global Cartesian coordinates'. In this process, the 'configuration' remains the same but the 'formex' that represents the configuration will change.

Retronorms play a central role in formex configuration processing and there are many retronorms that are frequently used in practice. The objective of this section is to describe a group of 12 basic retronorms that are referred to as the 'elementary retronorms'. These retronorms are described in Table 2.A.9.

In general, the canonic parameter of a relection function is a 'perdicant', where a 'perdicant' is defined as any meaningful evaluable combination of

- relations that may include brevants,
- logical operators and
- parentheses.

To explain the idea of a retronorm, consider the web-like configuration shown by thick lines in Fig 2.A.15a. This configuration consists of 58 line elements that are connected together at 35 nodal points. The web-like configuration in Fig 2.A.15a is shown together with a 'polar normat'. This normat is to be used as the reference system for the formulation of the compret of the configuration. The web-like configuration and the polar normat are shown together with the corresponding r-s polar coordinate system and the x-y global coordinate system in Fig 2.A.15b.
A formex representing the web-like configuration of Fig 2.A.15 relative to the U1-U2 polar normat may be written as

\[ E = \text{rinid}(6,5,1,1) \{ [1,1; 2,1] \} # \text{rinid}(7,4,1,1) \{ [1,1; 1,2] \} \]

If this formex is plotted with respect to the global x-y coordinate system, the result will be as shown in Fig 2.A.16.

![Plot of E relative to the x-y global Cartesian coordinate system](image)

The configuration in Fig 2.A.16 has the same 'compret' as that in Fig 2.A.15 but the 'normic' properties of these two configurations are quite different, see Section 1.4.4 of Ref 1. To clearly see the relationship between the configurations of Figs 2.A.15 and 2.A.16, the letters A, B, C and D are used to indicate the corresponding corners of the configurations.

A formex whose uniples are the global x-y coordinates of the nodal points of the web-like configuration of Fig 2.A.15 may be written as

\[ G = \text{bp}(1.5,30) \{ E \} \]

The effect may be described as follows: If

[U1,U2]

is a signet of E, with U1 and U2 being the normat coordinates of a nodal point of the configuration of Fig 2.A.15, then

\[ [x,y] \]

will be the corresponding signet of G, with x and y being the global coordinates of the same nodal point.

The transformation is effected through the function

\[ \text{bp}(1.5,30) \]

This is a 'basipolar' retronorm which has two canonic parameters. The general form of a basipolar retronorm is

\[ \text{bp}(b_1,b_2) \]

where, 'bp' stands for 'basipolar' and where \( b_1 \) specifies the 'scale factor' in the first direction and \( b_2 \) specifies the 'scale factor' in the second direction. The scale factor in the first direction is 'linear' and the scale factor in the second direction is 'angular'.

In the case of the basipolar retronorm for the example of Fig 2.A.15, that is,

\[ \text{bp}(1.5,30) \]

the first canonic parameter is given as 1.5, indicating that the scale factor for transforming the coordinates along U1 into those along the radial polar axis r is 1.5. The second canonic parameter of the retronorm is given as 30. This indicates that a division along the second normat direction U2 corresponds to 30° on the circumferential polar axis s.

In general, the effect of a basipolar retronorm is to find the r-s polar coordinates of the nodal points of a configuration from the U1-U2 normat coordinates using the scale factors \( b_1 \) and \( b_2 \) and then find the equivalent global x-y coordinates using the standard relations

\[
\begin{align*}
x & = r \cos s \\
y & = r \sin s
\end{align*}
\]

These relations, in terms of the U1-U2 normat coordinates will be of the form

\[
\begin{align*}
x & = b_1 xU1 \times \cos (b_2 xU2) \\
y & = b_1 xU1 \times \sin (b_2 xU2)
\end{align*}
\]

The prefix 'basi' in the term 'basipolar' implies that the divisions along the U1 and U2 directions are 'uniform'. Six of the retronorms in Table 2.A.9 have the prefix 'basi'. Each of these retronorms has a corresponding retronorm with a different prefix, namely, the prefix 'metri' that implies 'rhythmically' increasing or decreasing. This 'rhythm' is governed by a 'geometric progression'.

To elaborate, consider the sequence of the divisions along an axis, as shown in Fig 2.A.17. Here, the first division in the sequence is equal to b and the subsequent divisions are equal to

\[ b \times m, \ b \times m^2, \ b \times m^3, \ldots \]

as shown in the figure. If \( m>1 \) then the divisions become progressively larger as n increases and if \( m<1 \) then the divisions become progressively smaller as n increases. The terms 'b' and 'm' are referred to as the 'basifactor' and 'metrifactor', respectively.
With a 'rhythm' of the divisions along an axis as shown in Fig 2.A.17, the 'sum' of the divisions between 0 and \( n \) is given by

\[
b(1 - m^n)/(1 - m)
\]

This represents the sum of the first \( n \) terms of the geometric progression

\[
b, b \times m, b \times m^2, b \times m^3, ...
\]

Formian has a (numeric) function that can be used to find the above sum. The function is referred to as the 'metril' function. In terms of this function, the sum of the first \( n \) terms of the above geometric progression may be written as

\[
b \times \text{met}(m) \mid n
\]

The construct

\[
\text{met}(m)
\]

is a 'metril function' with 'met' being the imprint of the function and with the canonic parameter \( m \) being the metrifactor.

In general, the canonic parameter \( m \) and the argument \( n \) of a metril function are numeric expressions, and the value of

\[
\text{met}(m) \mid n
\]

is determined as follows:

- If \( m = 1 \) or \( n = 0 \) then \( \text{met}(m) \mid n \) is equal to \( n \).
- If \( m \neq 1 \) and \( n > 0 \) then \( \text{met}(m) \mid n \) is equal to
  
  \[
  (1 - m^n)/(1 - m)
  \]
  
- If \( m \neq 1 \) and \( n < 0 \) then \( \text{met}(m) \mid n \) is equal to
  
  \[
  -(1 - m^{-n})/(1 - m)
  \]

One of the retronorms with a 'metri' prefix in Table 2.A.9 is the 'metripolar retronorm'. To exemplify the application of this retronorm, let it be required to create the web-like configuration whose details are given in Fig 2.A.18. In this configuration the divisions in the radial direction have a 'metri-rhythm' with a metrifactor of 1.2. Also, the divisions in the circumferential direction have a metri-rhythm with a metrifactor of 0.8. Therefore, the divisions in the radial direction become progressively larger and the divisions in the circumferential direction become progressively smaller.

To formulate the configuration, it will be convenient to use the U1-U2 normat that follows the rhythms of the variations of the divisions of the configuration in the radial and circumferential directions, as shown in Fig 2.A.18. In terms of this normat, the configuration is represented by the formex E given in Fig 2.A.16.

The web-like configuration of Fig 2.A.18, relative to the global x-y coordinate system, may then be represented by

\[
G = \text{mp}(1,30,1.2,0.8) \mid E
\]

The construct

\[
\text{mp}(1,30,1.2,0.8)
\]

is a 'metripolar' retronorm with 'mp' being the imprint of the function. The first two canonic parameters of a metripolar retronorm are the 'basifactors' specifying the first divisions in the U1 and U2 directions, respectively. The remaining two canonic parameters are the 'metrifactors' specifying the 'metri-rhythms' along the U1 and U2 directions, respectively.

Now, turning the attention to Table 2.A.9, the first six elementary retronorms that are described in the table are for use in relation to Cartesian-type normats. This group of retronorms consists of:

- 'basiumifact' retronorm that effects uniform scaling in the first direction,
- 'metriunifact' retronorm that effects metri-rhythmic scaling in the first direction,
- 'basibifect' retronorm that effects uniform scaling in the first and second directions,
- 'metribifect' retronorm that effects metri-rhythmic scaling in the first and second directions,
- 'basitrifect' retronorm that effects uniform scaling in the first, second and third directions and
- 'metritrifect' retronorm that effects metri-rhythmic scaling in the first, second and third directions.

A number of examples involving the use of basibifect and basitrifect retronorms are found in Sections 1.4.4, 1.4.6 and 1.7 of Ref 1.
Table 2.A.9 Elementary Retronorms

<table>
<thead>
<tr>
<th>Brief descriptions of retronorms</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basiusifect Retronorm</strong></td>
<td>![Plot of bu(1.5)]</td>
</tr>
<tr>
<td>A ‘basiusifect retronorm’ is of the form $bu(b)$</td>
<td>The effect of a basiusifect retronorm is to replace the 1\textsuperscript{st} uniple $U_1$ of every signet of its argument by $b \times U_1$</td>
</tr>
<tr>
<td>The canonic parameter $b$ is a numeric expression.</td>
<td></td>
</tr>
<tr>
<td><strong>Metriusifect Retronorm</strong></td>
<td>![Plot of mu(0.8,1.2)]</td>
</tr>
<tr>
<td>A ‘metriusifect retronorm’ is of the form $mu(b,m)$</td>
<td>The effect of a metriusifect retronorm is to replace the 1\textsuperscript{st} uniple $U_1$ of every signet of its argument by $b \times \text{met}(m) \times U_1$</td>
</tr>
<tr>
<td>The canonic parameters $b$ and $m$ are numeric expressions.</td>
<td></td>
</tr>
<tr>
<td><strong>Basibusifect Retronorm</strong></td>
<td>![Plot of bb(1.5,0.9)]</td>
</tr>
<tr>
<td>A ‘basibusifect retronorm’ is of the form $bb(b_1,b_2)$</td>
<td>The effect of a basibusifect retronorm is to replace the 1\textsuperscript{st} and 2\textsuperscript{nd} uniples $U_1$ and $U_2$ of every signet of its argument by $b_1 \times U_1$ and $b_2 \times U_2$</td>
</tr>
<tr>
<td>The canonic parameters $b_1$ and $b_2$ are numeric expressions.</td>
<td></td>
</tr>
<tr>
<td><strong>Metribusifect Retronorm</strong></td>
<td>![Plot of mb(0.8,1.6,1.2,0.8)]</td>
</tr>
<tr>
<td>A ‘metribusifect retronorm’ is of the form $mb(b_1,b_2,m_1,m_2)$</td>
<td>The effect of a metribusifect retronorm is to replace the 1\textsuperscript{st} and 2\textsuperscript{nd} uniples $U_1$ and $U_2$ of every signet of its argument by $b_1 \times \text{met}(m_1) \times U_1$ and $b_2 \times \text{met}(m_2) \times U_2$</td>
</tr>
<tr>
<td>The canonic parameters $b_1$, $b_2$, $m_1$ and $m_2$ are numeric expressions.</td>
<td></td>
</tr>
<tr>
<td><strong>Basitrisifect Retronorm</strong></td>
<td>![Plot of bt(1.2,1.5,1.3)]</td>
</tr>
<tr>
<td>A ‘basitrisifect retronorm’ is of the form $bt(b_1,b_2,b_3)$</td>
<td>The effect of a basitrisifect retronorm is to replace the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} uniples $U_1$, $U_2$ and $U_3$ of every signet of its argument by $b_1 \times U_1$, $b_2 \times U_2$ and $b_3 \times U_3$</td>
</tr>
<tr>
<td>The canonic parameters $b_1$, $b_2$ and $b_3$ are numeric expressions.</td>
<td></td>
</tr>
<tr>
<td><strong>Metritrisifect Retronorm</strong></td>
<td>![Plot of mt(1,1.1,1.05,1.3,2.5)]</td>
</tr>
<tr>
<td>A ‘metritrisifect retronorm’ is of the form $mt(b_1,b_2,b_3,m_1,m_2,m_3)$</td>
<td>The effect of a metritrisifect retronorm is to replace the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} uniples $U_1$, $U_2$ and $U_3$ of every signet of its argument by $b_1 \times \text{met}(m_1) \times U_1$, $b_2 \times \text{met}(m_2) \times U_2$ and $b_3 \times \text{met}(m_3) \times U_3$</td>
</tr>
<tr>
<td>The canonic parameters $b_1$, $b_2$, $b_3$, $m_1$, $m_2$ and $m_3$ are numeric expressions.</td>
<td></td>
</tr>
</tbody>
</table>

* $E$ is the formex given in Fig 2.A.16. ** This is a pansion function, as described in Section 2.A.13.
Table 2.A.9 Elementary Retronorms (Continued)

<table>
<thead>
<tr>
<th>Brief descriptions of retronorms</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baspolar Retronorm</strong> A 'baspolar retronorm' is of the form ( bp(b_1,b_2) )</td>
<td>![Plot of ( bp(1.5,8) )</td>
</tr>
<tr>
<td>The effect of a baspolar retronorm is to replace the 1(^{st} ) and 2(^{nd} ) uniples ( U_1 ) and ( U_2 ) of every signet of its argument by the global x-y coordinates corresponding to the r-s polar coordinates ( r=b_1 \times U_1 ) and ( s=b_2 \times U_2 )</td>
<td>The canonic parameters ( b_1 ) and ( b_2 ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Metripolar Retronorm</strong> A 'metripolar retronorm' is of the form ( mp(b_1,b_2,m_1,m_2) )</td>
<td>![Plot of ( mp(1.20,0.9,1.2) )</td>
</tr>
<tr>
<td>The effect of a metripolar retronorm is to replace the 1(^{st} ) and 2(^{nd} ) uniples ( U_1 ) and ( U_2 ) of every signet of its argument by the global x-y coordinates corresponding to the r-s polar coordinates ( r=b_1 \times met(m_1) \times U_1 ) and ( s=b_2 \times met(m_2) \times U_2 )</td>
<td>The canonic parameters ( b_1, b_2, m_1 ) and ( m_2 ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Basicylindrical Retronorm</strong> A 'basicylindrical retronorm' is of the form ( bc(b_1,b_2,b_3) )</td>
<td>![Plot of ( bc(1,15,3) )</td>
</tr>
<tr>
<td>The effect of a basicylindrical retronorm is to replace the 1(^{st} ), 2(^{nd} ) and 3(^{rd} ) uniples ( U_1, U_2 ) and ( U_3 ) of every signet of its argument by the global x-y-z coordinates corresponding to the r-s-z cylindrical coordinates ( r=b_1 \times U_1, s=b_2 \times U_2 ) and ( z=b_3 \times U_3 )</td>
<td>The canonic parameters ( b_1, b_2, b_3 ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Metricylindrical Retronorm</strong> A 'metricylindrical retronorm' is of the form ( mc(b_1,b_2,b_3,m_1,m_2,m_3) )</td>
<td>![Plot of ( mc(1.25,1,1.0,8,1.5) )</td>
</tr>
<tr>
<td>The effect of a metricylindrical retronorm is to replace the 1(^{st} ), 2(^{nd} ) and 3(^{rd} ) uniples ( U_1, U_2 ) and ( U_3 ) of every signet of its argument by the global x-y-z coordinates corresponding to the r-s-z cylindrical coordinates ( r=b_1 \times met(m_1) \times U_1, s=b_2 \times met(m_2) \times U_2 ) and ( z=b_3 \times met(m_3) \times U_3 )</td>
<td>The canonic parameters ( b_1, b_2, b_3, m_1, m_2 ) and ( m_3 ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Basispherical Retronorm</strong> A 'basispherical retronorm' is of the form ( bs(b_1,b_2,b_3) )</td>
<td>![Plot of ( bs(1,30,5) )</td>
</tr>
<tr>
<td>The effect of a basispherical retronorm is to replace the 1(^{st} ), 2(^{nd} ) and 3(^{rd} ) uniples ( U_1, U_2 ) and ( U_3 ) of every signet of its argument by the global x-y-z coordinates corresponding to the r-s-t spherical coordinates ( r=b_1 \times U_1, s=b_2 \times U_2 ) and ( t=b_3 \times U_3 )</td>
<td>The canonic parameters ( b_1, b_2, b_3 ) are numeric expressions.</td>
</tr>
<tr>
<td><strong>Metrispherical Retronorm</strong> A 'metrispherical retronorm' is of the form ( ms(b_1,b_2,b_3,m_1,m_2,m_3) )</td>
<td>![Plot of ( ms(1.30,3,1.0,9,1.2) )</td>
</tr>
<tr>
<td>The effect of a metrispherical retronorm is to replace the 1(^{st} ), 2(^{nd} ) and 3(^{rd} ) uniples ( U_1, U_2 ) and ( U_3 ) of every signet of its argument by the global x-y-z coordinates corresponding to the r-s-t spherical coordinates ( r=b_1 \times met(m_1) \times U_1, s=b_2 \times met(m_2) \times U_2 ) and ( t=b_3 \times met(m_3) \times U_3 )</td>
<td>The canonic parameters ( b_1, b_2, b_3, m_1, m_2 ) and ( m_3 ) are numeric expressions.</td>
</tr>
</tbody>
</table>

\* \( E \) is the form given in Fig 2.A.16. \*\* This is a expansion function, as described in Section 2.A.13.
The remaining six elementary retronorms described in Table 2.A.9 are for use in relation to the polar, cylindrical and spherical normats. Some examples involving the use of basipolar, basicylindrical and basispherical retronorms are found in Sections 1.4.4, 1.8 and 1.9 of Ref 1.

2.A.12 Libra Function

A 'libra function' is a mechanism that effects 'serial formex composition'. The operation of a libra function has some similarities with that of a 'sigma summation' in scalar algebra. The idea of a libra function is introduced in terms of an example.

Let it be required to create a configuration consisting of a 'chain' of six connected line segments whose ends are on the parabola

\[ y = x^2 \]

in the range

\[ x = -1 \text{ to } x = 1 \]

This 'line-chain' is shown in Fig 2.A.19.

A formex representing the line-chain of Fig 2.A.19 may be written as

\[ C_1 = \text{lib}(i = -3, 2) \left\{ i/3, (i/3)^2; \frac{(i+1)}{3}, \left(\frac{(i+1)}{3}\right)^2 \right\} \]

The construct

\[ \text{lib}(i = -3, 2) \]

is a 'libra function' where

- 'lib' is the imprint of the function, standing for libra,
- \( i \) is the 'libra index' (or 'libra variable') and
- \(-3\) and \(2\) are the 'libra bounds'.

The part between the parentheses in the libra function, that is,

\[ i = -3, 2 \]

represents a 'looping mechanism'. The effect is that the libra index \( i \) successively assumes the values

\[ -3, -2, -1, 0, 1 \text{ and } 2 \]

The first item in the sequence is equal to the first libra bound and then the value is incremented by 1 successively until the value of the second libra bound is reached.

The argument of the above libra function is the cantle

\[ \left\{ i/3, (i/3)^2; \frac{(i+1)}{3}, \left(\frac{(i+1)}{3}\right)^2 \right\} \]

The first signet in this cantle, that is,

\[ i/3, (i/3)^2 \]

represents the \( x \) and \( y \) coordinates of the first end (left end) of a typical line segment in Fig 2.A.19. For instance, when \( i = -3 \) then \( x = -1 \) and \( y = 1 \). These are the \( x \) and \( y \) coordinates of the left end of the first line segment (leftmost line segment) in Fig 2.A.19.

The second signet in the argument of the above libra function, that is,

\[ \left(\frac{(i+1)}{3}, \left(\frac{(i+1)}{3}\right)^2 \right) \]

represents the \( x \) and \( y \) coordinates of the second end (right end) of a typical line segment. For instance, when \( i = -3 \) then \( x = -\frac{2}{3} \) and \( y = \frac{4}{9} \). These are the \( x \) and \( y \) coordinates of the right end of the first line segment in Fig 2.A.19.

The effect of the libra function is to replace the libra index \( i \) in its argument by the numbers in the sequence

\[ -3, -2, -1, 0, 1 \text{ and } 2 \]

successively and then compose the resulting formices. This will give rise to the formex

\[ C_1 = \left\{ \left(\frac{1}{3}, \frac{1}{9} \right); \left(\frac{-2}{3}, \frac{4}{9} \right); \ldots \right\} \]

Each of the above cantles represents one of the line segments in Fig 2.A.19 and the whole line-chain is represented by \( C_1 \).

Fig 2.A.20 shows the plot of

\[ C_2 = \text{lib}(i = -10, 9) \left\{ \frac{i}{10}, \left(\frac{i}{10}\right)^2; \frac{(i+1)}{10}, \left(\frac{(i+1)}{10}\right)^2 \right\} \]

This equation is similar to the one representing the line-chain in Fig 2.A.19 except for the libra bounds that are \(-10\) and \(9\). In this case, the line-chain consists of 20 line segments. The effect of the 'divisor' \( 10 \) (factor \( 0.1 \)) in the argument of the above libra function is to change the values of the libra index \( i \) into \( x \) coordinates.

As another example, a line-chain consisting of 20 line segments on the cubic parabola.
y = x^3

in the range

x = -1 to x = 1

is shown in Fig 2.A.21. This line-chain is the plot of the formex

C3 = lib(i = -10, 9) | [i/10, (i/10)^3;

((i+1)/10, ((i+1)/10)^3]}

Fig 2.A.20 A line-chain consisting of twenty line segments on the parabola y = x^2

The general form of a libra function designator is

lib(i = m, n) | E

where

• the libra index i is a ‘name’ (in Formian, i is an identifier, see Section 1.3.2 of Ref 1),
• the libra bounds m and n are integers (in Formian, m and n are integer expressions, see Section 2.A.3) and
• the argument E (the ‘librand’) is a formex expression that may contain the libra index i at one or more positions.

If E(r) denotes the value of the argument E when every occurrence of i in E is replaced by an integer r, then

if m < n

lib(i = m, n) | E = E(m) # E(m+1) # E(m+2) # ...

#E(n-2) # E(n-1) # E(n),

and

if m > n

lib(i = m, n) | E = E(m) # E(m-1) # E(m-2) # ...

#E(n+2) # E(n+1) # E(n)

The argument of a libra function need not necessarily contain the libra index. For example

lib(i = 1, 5) | [7,9]

is a valid libra designator whose value is

{{[7,9], [7,9], [7,9], [7,9], [7,9]}}

Libra functions may be ‘nested’. That is, they may follow each other in a sequence. An example of a ‘3-nested’ libra function designator is

lib(i = m, n) | lib(j = i + 2, p) | lib(k = t, j - i + 8) | E

In a nested libra function, a libra bound may be given in terms of one or more of the preceding libra indices. For example, in the above nested libra function, the first libra bound of the second libra function is given as i + 2 and the second libra bound of the third libra function is given as j - i + 8.

Many examples of the use of the libra function are found in Refs 6 and 10. In these references, a libra function such as

lib(i = m, n)

is represented by

\[
\left[ \begin{array}{c}
\frac{n}{i=m}
\end{array} \right]
\]

This is referred to as a ‘libra operator’. With libra operators, the above 3-nested libra function designator is represented by

\[
\left[ \begin{array}{c}
\frac{n}{i=m} \frac{p}{j=i+2} \frac{j-i+8}{k=t}
\end{array} \right] E
\]

Note that the libra operator does not involve any rullus symbol.

It is not possible to use the libra operator in Formian. However, when outside the domain of Formian, the libra operator provides a convenient notation for the libra mechanism.

2.A.13 Pansion and Depansion Functions

A ‘pansion function’ is of the form

pan(d,p)

where ‘pan’ is the imprint of the function standing for ‘pansion’. The argument of the pansion function
should be a 'formex entity', that is, something that has a 'formex value'. In Formian, the argument of a pansion function is a formex expression.

The canonic parameter \( d \) of a pansion function represents a 'position' in a signet and the canonic parameter \( p \) represents a numeric value. The effect of a pansion function is to produce a formex by inserting a uniple equal to \( p \) at position \( d \) of every signet of its argument. For example, if

\[
E = \{[4,1; 3,2], [2,2], [6,2; 2,3]\}
\]

then

\[
\text{pan}(2,5) | E = \{[4,5,1; 3,5,2], [2,5,2], [6,5,2; 2,5,3]\}
\]

In Formian, the canonic parameters \( d \) and \( p \) of a pansion function are an integer expression and a numeric expression, respectively.

A 'depansion function' is of the form

\[
\text{dep}(d)
\]

where 'dep' is the imprint of the function standing for 'depansion'. The argument of a depansion function should be a formex entity. In Formian, the argument of a depansion function is a formex expression.

The canonic parameter \( d \) of a depansion function represents a position in a signet. The effect of a depansion function is to produce a formex by removing the uniple at position \( d \) of every signet of its argument. For instance, if

\[
E = \{[4,5,1; 3,5,2], [2,5,2], [6,5,2; 2,5,3]\}
\]

then

\[
\text{dep}(1) | E = \{[5,1; 5,2], [5,2], [5,2; 5,3]\}
\]

In Formian, the canonic parameter of a depansion function is an integer expression.

2.A.14 Medulla Function

The 'medulla function' is of the form

\[
\text{med}
\]

where 'med' is the imprint of the function standing for 'medulla'. The term 'medulla' is a Latin word meaning the 'middle'. The term is used in formex algebra to refer to an ingot that consists of the 'distinct' signets of a formex. The argument of the medulla function should be a formex entity. In Formian, the argument of the medulla function is a formex expression.

The effect of the medulla function is to produce an ingot that consists of all the distinct signets of its argument. For example, if

\[
E = \{[2,1; 3,2], [3,2; 4,3; 2,1], [4,3]\}
\]

then

\[
\text{med} | E = \{[2,1], [3,2], [4,3]\}
\]

2.A.15 Novation Function

The idea of the 'novation function' is explained with the aid of an example. Consider the planar diagonal grid shown in Fig 2.A.22. The grid consists of 96 line elements that are connected together at 58 nodal points. The grid has regular horizontal and vertical divisions with the exception of the rightmost horizontal division which is equal to 1.65 m rather than 2.25 m.

![Fig 2.A.22 A diagonal grid with an irregular side](image)

In formulating the configuration of the grid, it is best to ignore the irregularities to begin with. A formex representing the compret of the 'regularised' grid, relative to the \( U1-U2 \) normat of Fig 2.A.23 may be written as

\[
E = \text{rinid}(6,4,2,2) | \text{larnid}(1,1) | [1,0; 0,1]
\]

The regularised grid relative to the x-y coordinate system is given by

\[
F = \text{bb}(2.25,1.95) | E
\]

Fig 2.A.23 The regularised grid

Now, to obtain the original shape of the grid the nodes indicated by asterisks in Fig 2.A.23 should be
moved to the left by 0.6 m. This will make the
elements on the right-hand side of the grid to assume
their correct positions indicated by the dotted lines.

To effect the movement of the nodes, one may write
\[ N = \text{rin}(2, 4, 2 \times 1.95) | [12 \times 2.25, 1.95] \]
\[ P = \text{tran}(1, -0.6) | N \]
\[ G = \text{nov}(1, N, P) | F \]

In the above formulation
- \( N \) represents the four nodes that are indicated by
  asterisks in Fig 2.A.23,
- \( P \) represents the points to which the nodes
  represented by \( N \) should be moved and
- \( G \) represents the configuration of the original
  grid as shown in Fig 2.A.22.

The construct
\[ \text{nov}(1, N, P) \]
is a ‘novation function’ where ‘nov’ is the imprint of
the function standing for ‘novation’ (The term
‘novation’ means ‘renewing’). The effect of the
function is to replace every signet of its argument
that is equal to a signet of \( N \) by the corresponding
signet of \( P \).

The general form of a novation function is
\[ \text{nov}(m, N, P) \]
where
- \( N \) is an ingot representing the list of nodes that
  are to be moved (in Formian, \( N \) is a formex
  expression whose value is an ingot),
- \( P \) is an ingot of the same order and grade as \( N \),
  representing the points to which the nodes
  represented by \( N \) should be moved (in Formian,
  \( P \) is a formex expression whose value is an
  ingot) and
- \( m \) is an integer representing the ‘mode’ of the
  operation (in Formian, \( m \) is an integer
  expression).

The value of the mode in the above example is given
as 1. This represents the mode of operation described
in this section. However, the novation function has
other modes of operation that will be discussed in a
subsequent paper in this series.

2.A.16 Tignum Function

The ‘tignum function’ is of the form
\[ \text{tig} \]
where ‘tig’ is the imprint of the function standing for
‘tignum’. The term ‘tignum’ is a Latin word meaning
a ‘log’. The term is used in formex algebra to mean a
cantle that consists of the signets of an ingot. The
argument of the tignum function should be an ingot.
In Formian, the argument of the tignum function is a
formex expression whose value is an ingot.

The effect of a tignum function is to produce a cantle
from the signets of its argument. For example, if
\[ E = \{[4,5,1], [3,4,2], [2,3,3]\} \]
then
\[ \text{tig} | E = [4,5,1; 3,4,2; 2,3,3] \]

As an example of the application of the tignum
function, consider the equation
\[ p = \text{tig} | \text{lib}(i = 1,10) | \text{ran} | 1, \text{ran} | 1 \]

Here, the argument
\[ \text{ran} | 1, \text{ran} | 1 \]
is a signet consisting of two uniples each of which is
a ‘random number function designator’ whose value
is a random number between –1 and 1.

The part
\[ \text{lib}(i = 1,10) | \text{ran} | 1, \text{ran} | 1 \]
in the above equation is a libra function designator
(see Section 2.A.12) whose value is an ingot of the
form
\[ \{[R1, R2], [R3, R4], \ldots, [R19, R20]\} \]

Each of the uniples R1 to R20 in this ingot is a
random number between –1 and 1. The variable \( p \)
in the above equation will then represent the cantle
\[ [R1,R2; R3,R4; \ldots; R19,R20] \]

The plot of \( p \) for an instance of the execution of the
Formian statement
\[ p = \text{tig} | \text{lib}(i = 1,10) | \text{ran} | 1, \text{ran} | 1; \]
is shown in Fig 2.A.24a. The plot of another instance
of the execution of the statement is shown in Fig
2.A.24b. The positions of the randomly chosen
points in Figs 2.A.24a and 2.A.24b are indicated by
little circles.

Fig 2.A.24 Two examples of 10-point polybates
A randomly generated shape of the kind shown in
Fig 2.A.24a or 2.A.24b is referred to as a ‘polybate’
(from 'poly' + Greek 'bates' meaning 'that which steps'). A polybate with n randomly chosen points is referred to as an 'n-point polybate'. Thus, the polybates in Fig 2.A.24 are '10-point polybates'.

The reason for the 'chequered-like pattern' of the polybate of Fig 2.A.24a is that, when a multi-point cantle is plotted, the overlapping of infilled areas cancels the infill.

Each polybate has a unique shape and the probability of another polybate having exactly the same shape is very small.

(*) 12-point Polybate with a Frame (*)
frame=rosad(0,0) [-5,5; 5,5];
use &,vs(33),lw(2.5),c(1,43);
clear; draw tranid(4.75,4.75) frame;
polybate=tig Ilib(i=1,12) [ran 14.6, ran 14.6];
use lw(0.6),c(1,23),c(3,ric abs ran 50); draw tranid(4.75,4.75) polybate;
<><><>

Fig 2.A.25 A scheme for the generation of 12-point polybates

Watching polybates on the Formian screen can be an engaging pastime. A sequence of polybates may be generated by repeated execution of a scheme such as the one shown in the edityor display of Fig 2.A.25. This scheme generates 12-point polybates with random infill colour.

ACKNOWLEDGEMENTS

The early work in formex configuration processing was greatly helped by substantial donations from a group of Iranian Engineers. These are A. Sarshar, A. Jahanshahi, C. G. Abkarian, G. A. Mirzareza, M. S. Yazdani and J. Hassanein and their contributions are gratefully acknowledged. In the early nineties, the Taiyo Kogyo Corporation of Japan played a crucial role in supporting research in formex configuration processing and, more recently, the Tomoe Corporation of Japan has been instrumental in supporting research in this field. Their generous help is gratefully acknowledged.

REFERENCES