APL 82, W. H. Janko, W. Stucky (ed.),
APL Quote Quad, Vol. 13 No. 1,


# Computational Simplicial Homology in APL

**J. O. Shallit**
Department of Mathematics
and
Department of Computing Services
University of California, Berkeley
Berkeley, CA 94720
USA

**Abstract.**

In this paper, we describe a set of *APL* programs prepared by the author for computation of homology groups of simplicial complexes. These programs may be used in conjunction with a first course in algebraic topology in several ways: to help the student gain an intuitive feel for homology theory, to suggest plausible conjectures in algebraic topology, and to aid proofs.

## I. Introduction.

Use of the computer as an aid to the discovery and proof of mathematical theorems has increased greatly in the past ten years. One example that immediately comes to mind is the machine-aided proof of the four-color conjecture by Haken and Appel (see [1]). Computers have also been used extensively in group theory [2 3] and number theory [4 5].

In this paper, we describe a set of *APL* programs prepared by the author for computation of homology groups of simplicial complexes. The entire set of programs takes up little more than one sheet of paper, and most are written in the direct definition ($\alpha$–$\omega$) formalism (see the appendix). These programs may be used in conjunction with a first course in algebraic topology in several ways: to help the student gain an intuitive feel for homology theory, to suggest plausible conjectures in algebraic topology, and to aid proofs.



© 1982 ACM 0-89791-078-8/82/007/0332 $00.75

The programs exemplify the use of the computer to **compute**, not as a tutor or instructor. Use of *APL* in this way has been encouraged for more than ten years; for example, see [6].

In many ways, *APL* is the ideal language to express algebraic concepts. First, its function-oriented structure applies naturally to algebraic systems. Second, its powerful array processing allows easy manipulation of matrices. Finally, *APL*'s conciseness and power make development and testing easy.

## II. Algebraic Topology.

An **n-simplex** is the n-dimensional generalization of a triangle in 2-space, i. e. a set homeomorphic to

$$\{(t_0, t_1, \cdots, t_n) \mid t_i \geqslant 0,\ t_0 + t_1 + \cdots + t_n = 1\}$$

the standard simplex. The points (1,0,0,...,0), (0,1,0,...,0), etc. are called the **vertices** of the standard simplex.

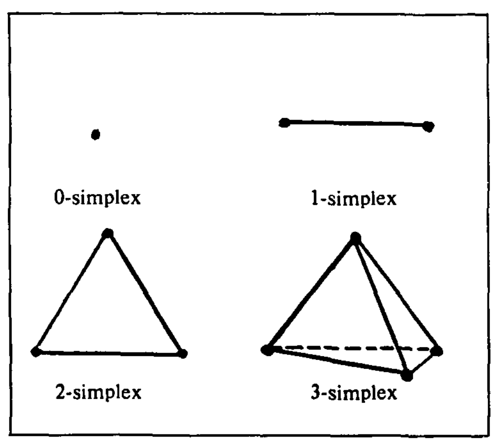



Figure 1: Simplices

A simplex $A$ is said to be a **face** of a simplex $B$ if $A$ is a subset of $B$ and every vertex of $A$ is also a vertex of $B$. For example, the 3-simplex above has 15 faces: the four points, the six lines, the four triangles, and the tetrahedron making up the 3-simplex itself. The dimension of a simplex is one less than the number of vertices defining it.

Two simplices $A$ and $B$ are said to be **properly joined** provided that either $A \cap B = \varnothing$ or $A \cap B$ is a face of both $A$ and $B$. See Figure 2.

Proper Joining    Improper Joining

Figure 2

A **simplicial complex** $K$ is a finite collection of properly joined simplices such that each face of a member of $K$ is also a member of $K$.

Through the process of **triangulation**, common topological objects may be represented by simplicial complexes. For example, the figure below is a triangulation of the Moebius strip, a one-sided surface.

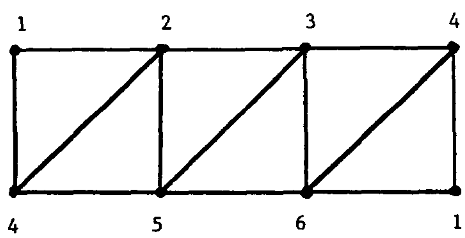



Figure 3: Triangulation of Moebius Strip

Associated with each simplicial complex is a set of groups called **homology groups**. These groups partially describe the structure of the complex, and methods for their computation form a large part of elementary algebraic topology.

For more on these concepts, see [7 8].

The *APL* workspace HOMOLOGY contains a set of functions and variables to manipulation representations of simplicial complexes, including the computation of their homology groups.

## III. Representations

An important question is: How should simplicial complexes be represented? We want a representation that is both **efficient** (in terms of storage requirements) and **easily manipulated**.

One representation (which we will call **B-representation**) is through Boolean matrices, where each row represents a face of a complex. For example, the Moebius strip could be represented by the following matrix of size 24 5:

```
      BFV VFP MOEBIUS
1 0 0 0 0 0
1 1 0 0 0 0
1 1 0 1 0 0
1 0 0 1 0 0
1 0 0 1 0 1
1 0 0 0 0 1
0 1 0 0 0 0
0 1 1 0 0 0
0 1 1 0 1 0
0 1 0 1 0 0
0 1 0 1 1 0
0 1 0 0 1 0
0 0 1 0 0 0
0 0 1 1 0 0
0 0 1 1 0 1
0 0 1 0 1 0
0 0 1 0 1 1
0 0 1 0 0 1
0 0 0 1 0 0
0 0 0 1 1 0
0 0 0 1 0 1
0 0 0 0 1 0
0 0 0 0 1 1
0 0 0 0 0 1
```

The presence of a 1 in column N indicates that vertex N is included in the face represented by the given row.

Another representation (which we call **V-representation**) is to again use rows to represent faces, but give the vertices explicitly (numbered from 1 to N), padded with zeros on the right, if necessary. The order in which rows appear is unimportant, but within each row the non-zero entries should appear in ascending order. The Moebius strip could be represented by the following matrix:

```
      VFP MOEBIUS
1 0 0
1 2 0
1 2 4
1 4 0
1 4 6
1 6 0
2 0 0
2 3 0
2 3 5
2 4 0
2 4 5
2 5 0
3 0 0
3 4 0
3 4 6
3 5 0
3 5 6
3 6 0
4 0 0
4 5 0
4 6 0
5 0 0
5 6 0
6 0 0
```

Another alternative, which is more efficient in terms of storage requirements, consists of representing complexes by listing only the **principal simplices**, i. e. those maximal with respect to inclusion. We call this the **P-representation**. To continue with our example of the Moebius strip, a P-representation might be

```
      MOEBIUS
124
146
235
245
346
356
```

The last possibility is a cross between P- and B-representations, where only principal simplices are given, but they are listed as rows in a Boolean matrix. This is the representation given in [9 10]. We will not use it in this paper.

We also want functions to convert between the various representations. These are given as follows:

| | |
|---|---|
| *BFV* | Convert to B-rep from V-rep |
| *VFB* | Convert to V-rep from B-rep |
| *VFP* | Convert to V-rep from P-rep |
| *PFV* | Convert to P-rep from V-rep |

Following the suggestion of Berry [11], these functions are designed to link together easily: for example, *PFV VFB M* converts *M* from Boolean to principal simplex representation.

Unless stated otherwise, most functions in the HOMOLOGY workspace assume arguments in V-representation. Although this representation is less efficient in terms of space needed, the ease of manipulation more than compensates for this deficiency.

## IV. Homology.

Suppose we are given a simplicial complex *A* in V-representation. The faces of *A* can be classified according to their dimension; for example, the Moebius strip has 6 faces of dimension 2.

Suppose *A* has N faces of dimension P and M faces of dimension P-1. We can then form a matrix of size (N,M), say *P INCID A*, which contains $\pm 1$ in row I and column J iff the J-th subsimplex of dimension P-1 is a face of the I-th subsimplex of dimension P. All other entries are 0. The signs of the non-zero elements are chosen to alternate across rows. These matrices are called **incidence matrices**. For example,

```
      MSTRIP←VFP MOEBIUS
      1 INCID MSTRIP
1 ¯1  0  0  0  0
1  0  0 ¯1  0  0
1  0  0  0  0 ¯1
0  1 ¯1  0  0  0
0  1  0 ¯1  0  0
0  1  0  0 ¯1  0
0  0  1 ¯1  0  0
0  0  1  0 ¯1  0
0  0  1  0  0 ¯1
0  0  0  1 ¯1  0
0  0  0  1  0 ¯1
0  0  0  0  1 ¯1
      2 INCID MSTRIP
1 ¯1  0  0  1  0  0  0  0  0  0  0
0  1 ¯1  0  0  0  0  0  0  0  1  0
0  0  0  1  0 ¯1  0  1  0  0  0  0
0  0  0  0  1 ¯1  0  0  0  1  0  0
0  0  0  0  0  0  1  0 ¯1  0  1  0
0  0  0  0  0  0  0  1 ¯1  0  0  1
```

The rows of these incidence matrices can be considered to be generators of an abelian group; algebraically speaking, the rows span a submodule of $\mathbf{Z}^{(n)}$. The rows of the incidence matrix may not form a basis, however, and it is important later to have a minimal spanning set. A matrix with integer entries can be reduced to one in upper triangular form that spans the same submodule by a type of Euclidean algorithm; this is done by the *APL* function *UT*. *UT M* produces a matrix such that (*UT M*)+.×*M* is in upper triangular form. We can then delete rows containing all zeroes with the function *REDUCE*, which generates a minimal spanning set. Note that only integer operations are used.

For example:

```
REDUCE:  RZR TRI ω ⍝ MIN SPANNING ROW SET
RZR:     (∨/ω≠0)⌿ω ⍝ REMOVE ZERO ROWS
TRI:     (UT ω)+.×ω ⍝ CHANGE TO UPPER TRIANG FORM
      B←REDUCE 2 INCID MSTRIP
      B
1 ¯1  0  0  1  0  0  0  0  0  0  0
0  1 ¯1  0  0  0  0  0  0  0  1  0
0  0  0  1  0 ¯1  0  1  0  0  0  0
0  0  0  0  1 ¯1  0  0  0  1  0  0
0  0  0  0  0  0  1  0 ¯1  0  1  0
0  0  0  0  0  0  0  1 ¯1  0  0  1
      ⍴B
6 12
```

In a similar fashion, it is possible to obtain a basis for the null space of a given matrix; in fact, this is nothing more than the rows of *UT M* corresponding to the zero rows of (*UT M*)+.×*M*. For example:

```
NS:      (∧/0=T+.×ω)⌿T←UT ω ⍝ NULL SPACE
      Z←NS 1 INCID MSTRIP
      Z
 1 ¯1  0  0  1  0  0  0  0  0  0  0
 1 ¯1  0  1  0  0  1  0  0  0  0  0
 0  0  0  1  0 ¯1  0  1  0  0  0  0
 1  0 ¯1  1  0  0  0  0  1  0  0  0
¯1  1  0  0  0 ¯1  0  0  0  1  0  0
 0  1 ¯1  0  0  0  0  0  0  0  1  0
 1  0 ¯1  0  0  1  0  0  0  0  0  1
      ⍴Z
7 12
```

Note that the matrices given by *Z* and *B* above have the same number of columns. The group (or Z−module) represented by *Z* is called the **cycle group** and that represented by *B* is called the **boundary group**. We can find a function that maps *B* into *Z*; this amounts to expressing the rows of *B* as integer linear combinations of rows of *Z* and is neatly given by the ⌹ function. In theory, the result will be a matrix with integer entries, but round-off error occasionally obscures this.

```
      ⎕PP←1
      H←(⍉B)⌹⍉Z
      H
 1E0      ¯3E¯17   ¯5E¯17    1E0      3E¯16    2E¯17
¯1E¯18    ¯4E¯17    6E¯17   ¯5E¯17    1E0     ¯9E¯17
 1E¯34    ¯2E¯17    1E0      8E¯17   ¯8E¯17    1E0
¯2E¯18     8E¯17   ¯8E¯19   ¯2E¯17   ¯1E0     ¯1E0
 1E¯18     4E¯17   ¯1E¯17    1E0      4E¯17   ¯7E¯17
¯1E¯18     1E0     ¯1E¯18    1E¯17    1E0      1E¯16
¯2E¯18     2E¯17    1E¯17   ¯2E¯17    3E¯17    1E0
      ⌊H+.5
1  0  0  1  0  0
0  0  0  0  1  0
0  0  1  0  0  1
0  0  0  0 ¯1 ¯1
0  0  0  1  0  0
0  1  0  0  1  0
0  0  0  0  0  1
```

The matrix $H$ given by the matrix division "imbeds" $B$ into $Z$; a well-known theorem says that we can change $H$ into a diagonal matrix through elementary row and column operations. This is done by the function *DIAG*.

```
DIAG:     ⍉TRI ⍉TRI ⍵ ⍝ REDUCE TO DIAGONAL FORM
      DIAG ⌊H+.5
 1  0  0  0  0  0
 0  1  0  0  0  0
 0  0  1  0  0  0
 0  0  0  1  0  0
 0  0  0  0 ¯1  0
 0  0  0  0  0 ¯1
 0  0  0  0  0  0
```

The interpretation of this matrix is as follows: The quotient group represented by $H$ is isomorphic to a sum of cyclic groups:

$$\mathbf{Z}/a_1\mathbf{Z} + \mathbf{Z}/a_2\mathbf{Z} + \cdots + \mathbf{Z}/a_n\mathbf{Z}$$

where the $a_n$ are entries on the diagonal of $H$. Hence the number of rows containing all zeros correspond to copies of **Z**, and other non-unit rows indicate the torsion (finite order) subgroups. In this case, $H$ is isomorphic to **Z**.

The entire procedure is automated by the function *CH*, which takes a left argument of the order of the homology group to be computed and a right argument of a simplicial complex in vertex form. The result is the non-unit entries on the diagonal of the $H$ matrix.

Usually it is more amenable to employ the function *CHOMP* (which stands for "Compute Homology, Please"); this has a left argument identical to that of *CH* and a right argument of the name of a complex in P-representation. The result is a character matrix describing the homology groups.

```
      CHOMP 'MOEBIUS'
H(0) = Z
H(1) = Z
H(2) = 0
```

## V. Discovering Theorems.

The HOMOLOGY workspace provides P-representations of triangulations of some common topological objects:

| | |
|---|---|
| *KLEIN* | Klein Bottle |
| *PROJPLANE* | Projective Plane |
| *TORUS* | Torus |
| *MOEBIUS* | Moebius Strip |
| *CYL* | Cylinder |

Also provided are the following functions, which generate sequences of topological objects:

| | |
|---|---|
| *SPHERE N* | The N-sphere |
| *K N* | Complete graph on N points |

It is useful to be able to join these objects in various ways. The following "conjunctive" functions are provided:

| | |
|---|---|
| *A DU B* | Disjoint union |
| *A CS B* | Connected sum |
| *A CP B* | Cartesian product |

Direct sum corresponds to placing A beside B; connected sum cuts a "hole" corresponding to the first simplex in A and B and gluing the complexes together at the "hole". Cartesian product forms a new complex by a method similar to APL outer product.

For example,

```
      CHOMP 'KLEIN'
H(0) = Z
H(1) = Z/2Z + Z
H(2) = 0
      CHOMP 'PROJPLANE CS PROJPLANE'
H(0) = Z
H(1) = Z/2Z + Z
H(2) = 0
```

Even if he doesn't know the theorem that 2-pseudomanifolds are isomorphic iff they have the same homology groups, the student might conjecture: **The Klein bottle is isomorphic to the connected sum of two projective planes.**

In a similar fashion, the student could compare the homology groups of the torus with those of the Cartesian product of two circles (i. e. 1-spheres):

```
      CHOMP 'TORUS'
H(0) = Z
H(1) = Z + Z
H(2) = Z
      CHOMP '(SPHERE 1) CP SPHERE 1'
H(0) = Z
H(1) = Z + Z
H(2) = Z
```

There are many similar examples.

Comparing the groups of the the Klein bottle, projective plane, and objects like the torus, another plausible (in fact, true) conjecture would be: **A 2-pseudomanifold is non-orientable iff its second homology group is {0}.**

Let's now look at the homology groups of N-dimensional spheres. For example:

```
      CHOMP 'SPHERE 1'
H(0) = Z
H(1) = Z
      CHOMP 'SPHERE 2'
H(0) = Z
H(1) = 0
H(2) = Z
      CHOMP 'SPHERE 3'
H(0) = Z
H(1) = 0
H(2) = 0
H(3) = Z
```

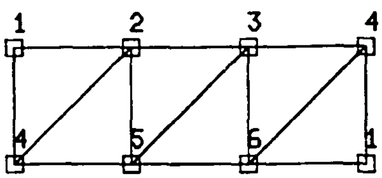
Judging from these examples, a plausible conjecture might be: **The 0-th and N-th homology groups of the N-sphere are both {Z}; all others are {0}.**

These are just a few of the theorems that can be discovered easily by an inquiring student.

## VI. Interface with Graphics.

It is worthwhile to mention a simple experimental interface of the HOMOLOGY workspace with graphics workspaces suitable for use with a Tektronix 4013 or 4015 terminal. This allows interactive graphics input of simplicial complexes, where the user sketches a triangulation on the screen and the groups are computed automatically.

For example, below is the sample input from the graphical homology program for the Moebius strip.




$H_0(MSTRIP) = Z$

$H_1(MSTRIP) = Z$

$H_2(MSTRIP) = 0$

## VII. Topics for Further Development.

The HOMOLOGY workspace has been used successfully in conjunction with an elementary course in algebraic topology at the University of California, Berkeley. However, there are many ways in which the workspace could be improved. We list just a few:

A. Rewrite the functions that compute with incidence matrices to use sparse representation methods.

B. Develop good interactive methods for input and display of higher-dimensional complexes on graphics terminals.

C. Find upper bounds on the computation time and space needed to determine homology groups based on the number of simplexes.

## VIII. Acknowledgements.

The author acknowledges with thanks conversations with John Hughes.

## IX. References.

[1] K. Appel and W. Haken, The solution of the four-color map problem, *Scientific American* (October, 1977) pp. 108-121.

[2] J. A. Gallian, "Computers in Group Theory", *Mathematics Magazine*, V. 49 (1976) pp. 69-73.

[3] J. M. Kane, "Distribution of Orders of Abelian Groups", *Mathematics Magazine*, V. 49 (1976) pp. 132-135.

[4] H. Zimmer, *Computational Problems, Methods, and Results in Algebraic Number Theory*, Springer-Verlag, New York: 1972.

[5] D. H. Lehmer, "Computer Technology Applied to the Theory of Numbers", in *Studies in Number Theory*, W. J. Leveque, editor, Prentice-Hall, New York: 1969.

[6] P. C. Berry, A. D. Falkoff, and K. E. Iverson, "Using the Computer to Compute: A Direct but Neglected Approach to Teaching Mathematics", IBM New York Scientific Center Technical Report No. 320-2988, (May, 1970).

[7] Fred H. Croom, *Basic Concepts of Algebraic Topology*, Springer-Verlag, New York: 1978.

[8] W. S. Massey, *Algebraic Topology: An Introduction*, Springer-Verlag, New York: 1967.

[9] T. Pinkerton, "An Algorithm for the Automatic Computation of Integral Homology Groups", *Mathematical Algorithms*, V. I, No. I (January, 1966) pp. 27-44.

[10] T. Pinkerton, "An Algorithm for the Automatic Computation of Integral Homology Groups, Part II", *Mathematical Algorithms*, V. I, No. II (April, 1966) pp. 33-39.

[11] P. C. Berry, personal communication.

## Appendix: Function Listings

### A. Functions to Compute and Display Homology Groups.

```
BBASE:    REDUCE (α+1) INCID ω
CHAR:     -/,(0,⍳DIM ⍎ω) APPLY 'NF VFP ',ω ⍝ EULER-POINCARE CHARACTERISTIC
CHOMP:    (0,⍳DIM ⍎ω) APPLY 'CHOM ',ω ⍝ COMPUTE HOMOLOGY, PLEASE
CHOM:     α HEAD FMT α CH VFP ω
CH:       1 REMOVE | 1 1 ⍉ SQUARE DIAG α HBASE ω  ⍝ COMPUTE HOMOLOGY
DIAG:     ⍉TRI ⍉TRI ω ⍝ REDUCE TO DIAGONAL FORM
DIM:      (⌈/+/ω≠0)-1
FMT:      REF ¯2↓,(ω ZERO 'Z/' WITH (RLF ⍕COL ω),'Z') WITH ' + ' : 0=ρω : ,'0'
HBASE:    ⌊.5+(⍉α BBASE ω)⌹⍉α ZBASE ω
HEAD:     'H(',(⍕α),') = ',ω
INCID:    (∧/∨/[2] (RTF α SIMP ω)∘.=RTF (α-1) SIMP ω) INCI α
INCI:     (ρα)ρ(,α)\((1↑ρα)×ω+1)ρ(ω+1)ρ1 ¯1
NF:       (ρα SIMP ω)[1] ⍝ NO. OF FACES OF ω OF DIM α
NS:       (∧/0=T+.×ω)⌿T←UT ω ⍝ NULL SPACE
REDUCE:   RZR TRI ω ⍝ MIN SPANNING ROW SET
SIMP:     ((α+1)=+/ω≠0)⌿ω ⍝ GIVEN SC ω IN V-FORM, DETERMINES α-SIMPLICES
SQUARE:   (2ρ1↑ρω)↑ω ⍝ SELECT SQUARE SUBMATRIX
TRI:      (UT ω)+.×ω ⍝ CHANGE TO UPPER TRIANG FORM
ZBASE:    NS α INCID ω: α=0 : ID ⌈/0,,ω
ZERO:     ω[;1],(α≠0) BLANK 0 1↓ω
```

### B. Functions to Convert Representations.

```
BFV:      +/[2] ω∘.=⍳⌈/0,,ω ⍝ B-REP FROM V-REP
BSUB:     ⍉(ωρ2)⊤⍳¯1+2*ω ⍝ BOOLEAN MATRIX REPRESENTING ALL NON-NULL SUBSETS
MASK:     ,(+/ω)∘.≥⍳1↓ρω
SORT:     ω[ORDER ω;]
SUBSETS:  (0,ω)[1+VFB BSUB ρω] ⍝ SUBSETS OF ω, PADDED WITH 0'S
VFB:      RTF (ρω)ρ(MASK ω)\(A≠0)/A←,ω×(ρω)ρ⍳1↓ρω ⍝ V-REP FROM B-REP
```

```
    ∇ V←VFP P;K
[1]  ⍝ V-REP FROM P-REP
[2]   V←(0,1↓ρP)ρK←0
[3]  L1:→((1↑ρP)<K←K+1)/L2
[4]   V←V CAT SUBSETS 0 REMOVE P[K;]
[5]   →L1
[6]  L2:V←RDR V
    ∇
```

```
    ∇ P←PFV V
[1]  ⍝ CONVERTS TO P-REP FROM V-REP
[2]   V←V[⍒+/V≠0;]
[3]  ⍝ ORDER BY LARGEST SIMPLICES
[4]   P←(0,1↓ρV)ρ0
[5]  L0:→(0=1↑ρV)/0
[6]   P←P,[1] V[1;]
[7]   V←(~∧/V∊V[1;],0)⌿V
[8]   →L0
    ∇
```

### C. Functions to Generate and Join Objects.

```
K:        (1,ω+1)ρ⍳ω+1 ⍝ COMPLETE GRAPH ON ω+1 POINTS, P-REP
SPHERE:   ⊖VFB (⍳ω+2)∘.≠⍳ω+2 ⍝ PRINCIPAL SIMPLICES FOR N-SPHERE
CS:       SORT ROWSORT RENUM (1 0↓α REPL α[1;],[0.5] T[1;]),[1] 1 0↓T←ω+⌈/0,,α
DU:       α CAT ω+⌈/0,,α ⍝ DISJOINT UNION
ECP:      +\1,[3] (ID 2)[;1+(ω-1) COMB α+ω-2] ⍝ ELEMENTARY CART PROD
```

```
    ∇ Z←A CP B;R;U;T;P;J
[1]  ⍝ CARTESIAN PRODUCT
[2]   T←⌈/,A
[3]   U←⌈/,B
[4]   R←(T,U)ρ⍳T×U
[5]   P←(⍳1↑ρA) PAIR⍳1↑ρB
[6]   Z←(0,¯1+(1↓ρA)+1↓ρB)ρ0
[7]   J←0
[8]  L0:→((1↑ρP)<J←J+1)/0
[9]   C←0 REMOVE A[P[J;1];]
[10]  D←0 REMOVE B[P[J;2];]
[11]  Z←Z CAT R[C;D] INDEX(ρC) ECPρD
[12]  →L0
    ∇
```

### D. Utility Functions.

```
APPLY:    (⍎ (⍕1↑α),' ',ω) CAT (1↓α) APPLY ω : 0=ρα : 0 0ρ''
BLANK:    α⍀α⌿ω
CAT:      (MAT α) JOIN MAT ω ⍝ CATENATE TWO ARRAYS, TREATED AS MATRICES
COL:      ((ρω),1)ρω ⍝ CONVERT TO COLUMN
COMB:     (0,α COMB ω-1),[1] 1,(α-1) COMB ω-1 : ∨/α=ω,0 : (1,ω)ρα=ω
EXP:      (1,(1↑ρω)ρ0),0,[1] ω ⍝ ADD ROW AND COLUMN TO MATRIX ω
FILL:     1↑0ρω ⍝ FILL CHARACTER FOR ω
ID:       (⍳ω)∘.=⍳ω  ⍝ SQUARE IDENTITY MATRIX
INDEX:    (,α)[⎕IO+(ρα)⊥ω-⎕IO]
JOIN:     (((1↑ρα),T)↑α),[1] ((1↑ρω),T←(1↓ρα)⌈1↓ρω)↑ω ⍝ JOIN TWO MATRICES
MAT:      (¯2↑1 1,ρω)ρω
PAIR:     (((ρα)×ρω)ρα),[1.5] ,⍉((ρα),ρω)ρω ⍝ PAIR GENERATOR
RDR:      (∨/T≠¯1 0↓0,[1] T)⌿T←SORT ω ⍝ REMOVE DUPLICATE ROWS
REF:      (T∨¯1↓0,T←(FILL ω)≠ω,FILL ω)/ω,FILL ω
REMOVE:   (ω≠α)/ω
RENUM:    (UNI ω)⍳ω
REPLACE:  (α,ω[2;])[(⍳ρα)⌈(α∊ω[1;])×(ρα)+ω[1;]⍳α]
```

```
REPL:     (ρα)ρ(,α) REPLACE ω
RLF:      (0,¯1+(∨⍀ω≠FILL ω)⍳1)↓ω ⍝ REMOVE LEADING FILL CHARACTERS
ROWSORT:  ω INDEX 2 GRADEUP ω
RTF:      (0,1-(⌽∨⍀ω≠FILL ω)⍳1)↓ω ⍝ REMOVE TRAILING FILL CHARACTERS
RZR:      (∨/ω≠0)⌿ω ⍝ REMOVE ZERO ROWS
SORT:     ω[ORDER ω;]
UNI:      (ω≠1↓ω,1+¯1↑ω)/ω←ω[⍋ω←,ω]
WITH:     α,((1↑ρα),ρω)ρω : (ρρω)>ρρα : (((1↑ρω),ρα)ρα),ω
```

```
    ∇ T←ORDER A;I;K              ∇ Z←AXIS GRADEUP A;T;V
[1]   T←⍳1↑ρA              [1]  ⍝ RETURNS INDICES SUCH THAT
[2]   I←(ρA)[2]            [2]  ⍝      A INDEX I GRADEUP A
[3]  L1:→(0≥I)/0           [3]  ⍝ SORTS THE ELEMENTS OF A ALONG THE I-TH AXIS
[4]   T←T[K←⍋A[;I]]        [4]   Z←(ρA)⊤(⍋,A)-⎕IO
[5]   A←A[K;]              [5]   T←AXIS≠⍳ρρA
[6]   I←I-1                [6]   Z←⎕IO+Z[;⍋(T/ρA)⊥T⌿Z]
[7]   →L1                  [7]   V←(T/⍳ρρA),AXIS
    ∇                      [8]   Z←(⎕IO,1+V)⍉((ρρA),(ρA)[V])ρZ
                                 ∇
```

```
    ∇ Z←UT M;K;R;A;S;T;C
[1]  ⍝ RETURNS A MATRIX <UT M> SUCH THAT (UT M)+.×M IS UPPER
[2]  ⍝ TRIANGULAR.  BOTH THE ARGUMENT AND RESULT HAVE INTEGER
[3]  ⍝ ENTRIES.
[4]   Z←ID 1↑ρM
[5]   →(0=1↑ρM)/0
[6]  ⍝ FIND FIRST COLUMN WITH A NON-ZERO ENTRY
[7]   K←(∨⌿0≠M)⍳1
[8]  ⍝ IF THERE ARE NONE, WE ARE DONE
[9]   →(K>1↓ρM)/0
[10] ⍝ FIND THE SMALLEST NON-ZERO ENTRY IN THAT COLUMN...
[11]  S←⌊/|(0≠M[;K])/M[;K]
[12] L2:
[13]  R←S
[14] ⍝ ...AND FIND WHERE THAT ENTRY OCCURS
[15]  A←(|M[;K])⍳R
[16] ⍝ SWITCH ROWS IN BOTH MATRICES
[17] ⍝
[18]  M[1,A;]←M[A,1;]
[19]  Z[1,A;]←Z[A,1;]
[20] ⍝ AND SUBTRACT MULTIPLES OF THE FIRST ROW FROM ALL
[21] ⍝ THE OTHER ROWS
[22]  T←1+⍳1↑ρM
[23]  C←⌊M[T;K]÷M[1;K]
[24]  M[T;]←M[T;]-C∘.×M[1;]
[25]  Z[T;]←Z[T;]-C∘.×Z[1;]
[26] ⍝ WHAT IS THE NEW SMALLEST NON-ZERO ENTRY IN THAT COLUMN?
[27]  S←⌊/|(0≠M[;K])/M[;K]
[28] ⍝ IF IT'S SMALLER, CONTINUE; OTHERWISE WE'RE DONE WITH THE 1ST ROW
[29]  →(R>S)/L2
[30] ⍝ AT THIS POINT, WE RECURSE TO GET THE REST OF THE ROWS
[31] ⍝ AND THEN PRODUCE THE RESULT BY A SIMPLE +.×
[32]  Z←(EXP UT 1 1 ↓M)+.×Z
    ∇
```

## E. Variables

```
      TORUS                         KLEIN
 1 2 5                         1 2 5
 1 2 8                         1 2 7
 1 3 4                         1 3 7
 1 3 7                         1 3 9
 1 4 8                         1 4 5
 1 5 7                         1 4 9
 2 3 6                         2 3 6
 2 3 9                         2 3 8
 2 5 9                         2 5 6
 2 6 8                         2 7 8
 3 4 6                         3 6 7
 3 7 9                         3 8 9
 4 5 6                         4 5 8
 4 5 9                         4 6 7
 4 8 9                         4 6 9
 5 6 7                         4 7 8
 6 7 8                         5 6 9
 7 8 9                         5 8 9
```

```
      PROJPLANE                       MOEBIUS
 1 2 3                           1 2 4
 1 2 4                           1 4 6
 1 3 5                           2 3 5
 1 4 6                           2 4 5
 1 5 6                           3 4 6
 2 3 6                           3 5 6
 2 4 5
 2 5 6
 3 4 5
 3 4 6
```

```
      CYL
 1 2 4
 1 3 4
 3 4 6
 3 5 6
 2 5 6
 1 2 5
```