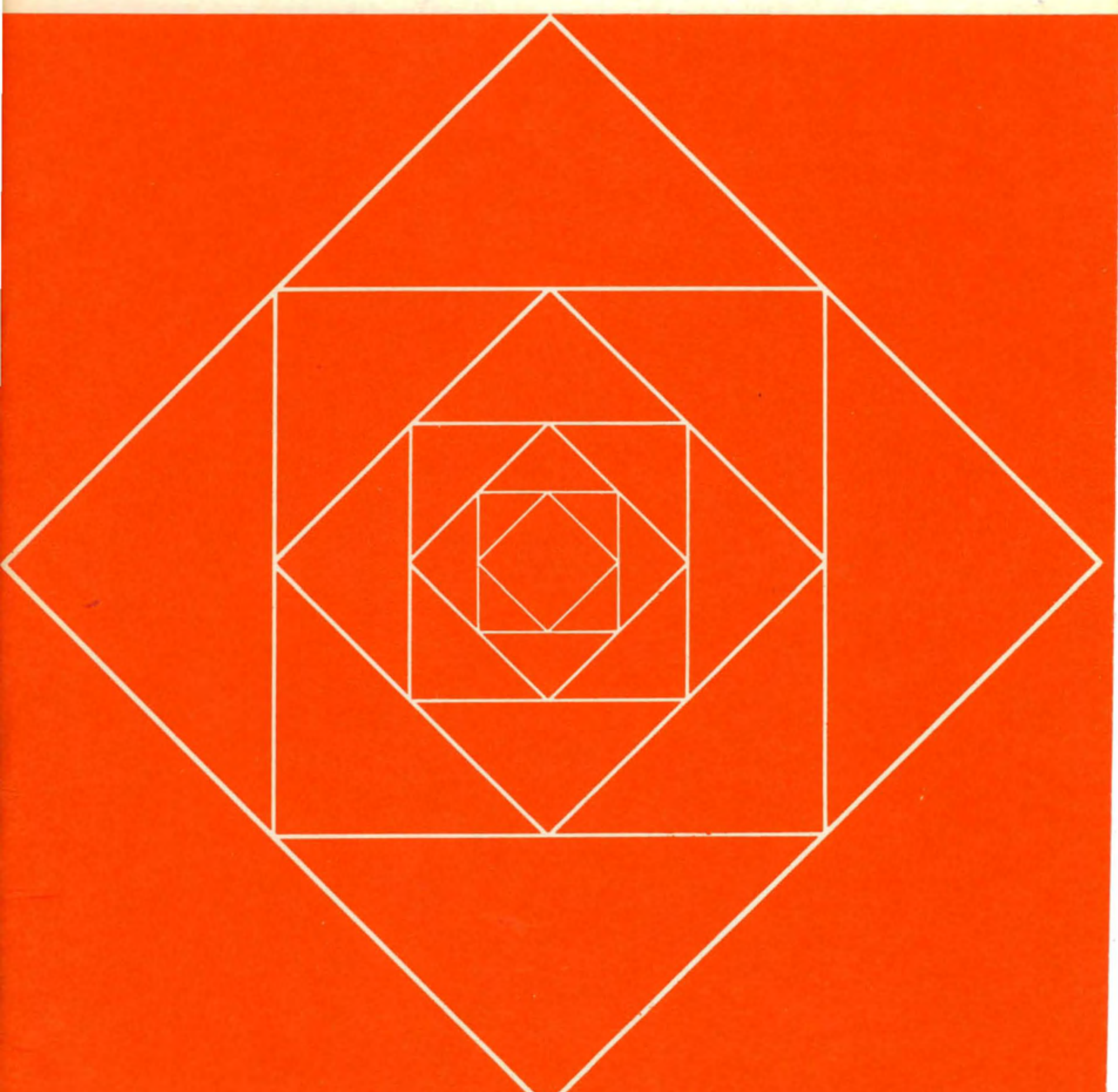
AN INTRODUCTION TO APL FOR SCIENTISTS AND ENGINEERS

IBM Philadelphia Scientific Center
Data Processing Division

TECHNICAL REPORT NO. 320-3019 MARCH 1973

K. E. IVERSON



IBM PHILADELPHIA SCIENTIFIC CENTER TECHNICAL REPORT NO. 320-3019 MARCH 1973

AN INTRODUCTION TO APL
FOR SCIENTISTS AND ENGINEERS

K. E. IVERSON

PHILADELPHIA SCIENTIFIC CENTER
IBM CORPORATION
3401 MARKET STREET, PHILADELPHIA, PENNSYLVANIA, 19104

TABLE OF CONTENTS

INTRODUCTION

This is an introduction to APL addressed to the scientist or engineer and designed to exploit any previous acquaintance with the very similar notation of vector algebra. A careful study of these pages should bring the reader to the point where he can begin to make serious use of APL in some topic of interest to him. The use of an APL terminal in this study, while not absolutely essential, adds greatly to the depth and interest of the work.

The pleasure and efficiency of learning by experimentation is not sufficiently appreciated, and the first six pages are designed to encourage this type of use of a terminal in learning APL. However, some readers are much inclined to experiment and to depart wildly from any prepared text; this cannot be proscribed but often wastes time. Undecipherable results obtained from the terminal by radical experiments or by mistyping may be either ignored or resolved with the aid of the four pages of reference material provided at the end of the paper.

It is usually advisable to attempt some independent use of the language rather soon, returning to a study of the language itself only to resolve difficulties and to open up new avenues of use. However, the reader may wish to consult *APL in Exposition* [1] for examples of use in a variety of areas, and the *APL\360 User's Manual* [2] for a fuller exposition of the language itself.

REFERENCES

1. Iverson, K. E., *APL in Exposition*, IBM Philadelphia Scientific Center, Technical Report 320-3010, January, 1972.

2. Falkoff, A. D., and K. E. Iverson, *APL\360 User's Manual*, IBM Corporation, 1968.

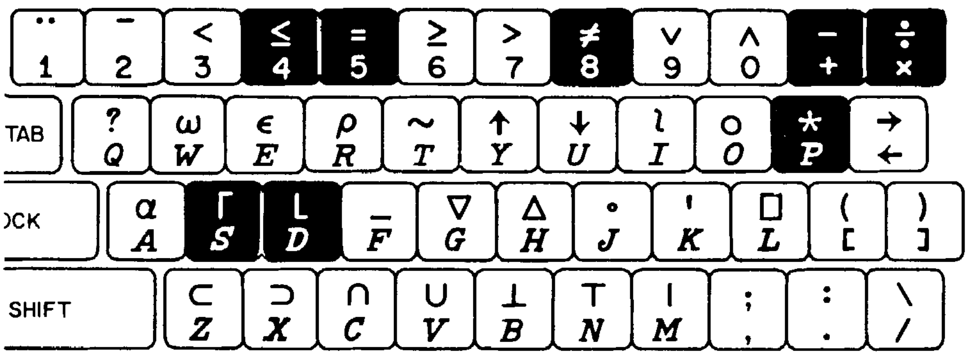
EXPERIMENTATION

A. Simple expressions:

```
      3+4              Carriage Return
7

      3×4.7            Carriage Return
14.1
```

B. Determine the meanings of the following eight functions (whose locations on the keyboard are identified by shading of the keys):

```
        -  ÷  *  ⌈  ⌊  ≤  =  ≠
```



For example, enter

```
      3-4
¯1
```

to verify that - represents subtraction, and

```
      3÷4
0.75
```

to verify that ÷ represents division.

SYSTEMATIC EXPERIMENTATION

A. On single quantities:

```
      2|1
1
      2|2
0
      2|3
1
```

Vary one argument systematically.

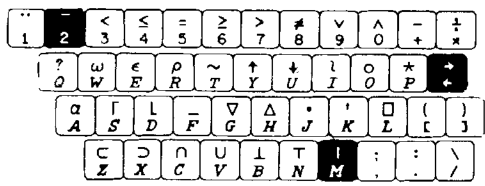
B. On lists of numbers:

```
      3|1 2 3 4 5 6 7
1 2 0 1 2 0 1

      3|¯3 ¯2 ¯1 0 1
0 1 2 0 1
```

Negative sign (uppercase 2) is distinct from the minus sign used for subtraction.

C. Use names for convenience:

```
      X←5
      X*2
25
      S←1 2 3 4 5 6 7
      3|S
1 2 0 1 2 0 1
      S*3
1 8 27 64 125 216 343
      S×S
1 4 9 16 25 36 49
```



D. Explore the functions of page 2, part B for negative numbers. For example:

```
      T←S-4
      T*2
9 4 1 0 1 4 9
```

E. To correct any entry <u>before</u> striking the carriage return, backspace to the point of error and strike the attention button (which "erases" everything from there to the right) and continue typing. For example:

```
      S←1 2 4 4
            v
            3 4 5
```

MULTIPLICATION AND OTHER FUNCTION TABLES

A. Expressions for tables:

```
      S←1 2 3 4 5 6 7

      S∘.×S                                    S∘.+S
 1  2  3  4  5  6  7                      2  3  4  5  6  7  8
 2  4  6  8 10 12 14                      3  4  5  6  7  8  9
 3  6  9 12 15 18 21                      4  5  6  7  8  9 10
 4  8 12 16 20 24 28                      5  6  7  8  9 10 11
 5 10 15 20 25 30 35                      6  7  8  9 10 11 12
 6 12 18 24 30 36 42                      7  8  9 10 11 12 13
 7 14 21 28 35 42 49                      8  9 10 11 12 13 14

      B←2 3

      B∘.×S                                    2 3∘.+S
 2  4  6  8 10 12 14                      3  4  5  6  7  8  9
 3  6  9 12 15 18 21                      4  5  6  7  8  9 10
```

B. Produce function tables for ⌈ ⌊ < = and |.

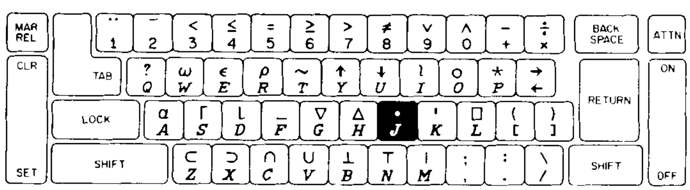
To aid in reading the tables you may wish to enter (by hand) the first argument in a column at the left of the table and the second in a row along the top.

C. Examine the tables for patterns and try to see why each function generates the particular pattern.

D. Repeat parts A-C with the vector *T*←*S*-4 replacing *S*.

E. The <u>outer product</u> (∘.+ and ∘.× and ∘.⌈, etc.) applies to higher-dimensional arrays in an obvious way. Try, for example:

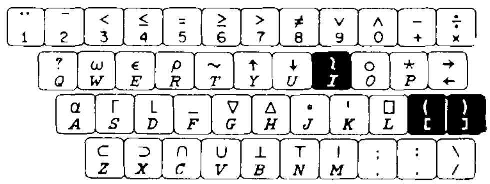
```
      Q←1 2 3 4

      Q∘.×Q∘.×Q
```



GRAPHS AND BAR CHARTS

A. Graph of a parabola:

```
      X←1 2 3 4 5 6 7

      V←(X-3)×(X-5)

      V
8  3  0 ¯1  0  3  8

      R←8 7 6 5 4 3 2 1 0 ¯1

      R∘.=V
 1 0 0 0 0 0 1
 0 0 0 0 0 0 0
 0 0 0 0 0 0 0
 0 0 0 0 0 0 0
 0 0 0 0 0 0 0
 0 1 0 0 0 1 0
 0 0 0 0 0 0 0
 0 0 0 0 0 0 0
 0 0 1 0 1 0 0
 0 0 0 1 0 0 0
```



B. Bar chart:

```
      R∘.≤V
 1 0 0 0 0 0 1
 1 0 0 0 0 0 1
 1 0 0 0 0 0 1
 1 0 0 0 0 0 1
 1 0 0 0 0 0 1
 1 1 0 0 0 1 1
 1 1 0 0 0 1 1
 1 1 0 0 0 1 1
 1 1 1 0 1 1 1
 1 1 1 1 1 1 1
```

C. Graph other functions of one argument.

D. Determine the significance of the function ⍳ (iota) by the following experiments:

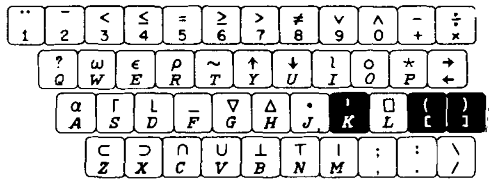
```
      ⍳5
      .1×⍳6
      ⍳1000
```

Pressing the attention button will interrupt any activity of the computer.

INDEXING AND CHARACTERS

A. Indexing:

```
      X←2 3 5 7 11
      X[4]
7
      X[1 2 3]
2  3  5
      X[5 4 3 2 1]
11 7  5  3  2
      X[4 1 3]
7  2  5
```



B. Characters:

```
      W←'DOG'
      W[3]
G
      W[3 2 1]
GOD
```

(If your computer gives no response to your entries you may be "in an open quote". Try entering a single quote to escape.)

```
      'ABCDEFGHI '[8 5 1 4 10 3 8 9 5 6]
HEAD CHIEF

      ' *'[2 1 2 2 1 2 2 1 2]
* ** ** *
```

C. Plotting:

Enter the following:

```
      X←1 2 3 4 5 6 7

      V←(X-3)×(X-5)

      R←8 7 6 5 4 3 2 1 0 ¯1

      R∘.=V

      ' *'[1+(R∘.=V)]

      ' *'[1+(2≥(X∘.-X))]
```

EXPLORING FUNCTIONS OF ONE ARGUMENT

A. Negation:

```
      X←3
      -X
¯3

      P←1 2 3 4 5 6 7
      Q←P-4
      R←P÷2
      -P
¯1 ¯2 ¯3 ¯4 ¯5 ¯6 ¯7
      -Q
3 2 1 0 ¯1 ¯2 ¯3
      -R
¯.5 ¯1 ¯1.5 ¯2 ¯2.5 ¯3 ¯3.5
```

B. Explore the following functions of one argument:

```
      ÷ | ⌊ ⌈ *
```

[Note that each of these symbols denotes either a function of two arguments (as in `X÷Y`) or of one argument (as in `÷Y`) just as the symbol `-` denotes either subtraction (as in `X-Y`) or negation (as in `-Y`) in conventional notation.]

C. Enter the following expressions:

```
      T←¯3 ¯2 ¯1 0 1 2 3
      T∘.×T
      ×T∘.×T
      '- +'[2+×(T∘.×T)]
```

Use these results (and any other experiments you wish to try) to determine the meaning of the function `×` when applied to one argument.

DEFINING NEW FUNCTIONS

A. A parabola with zeros at 3 and 5:

```
      X←7

      (X-3)×(X-5)
8

      ∇ Z←F X
[1]   Z←(X-3)×(X-5) ∇

      F 7
8

      2×F 7
16

      F F 7
15

      F 1 2 3 4 5 6 7
8  3  0  ¯1  0  3  8
```

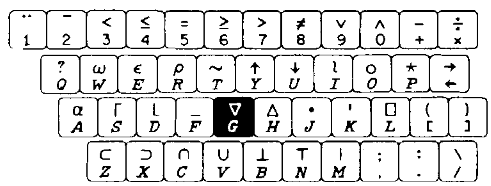
If you wish to change a function *F* after having defined it, type:

```
)ERASE F
```

Then begin your new definition of *F*. More convenient, but more complex, ways of revising functions are presented on page 20.

B. A test for divisibility by 7:

```
      ∇ Z←D X
[1]   Z←0=(7|X) ∇

      D 13
0

      D 868
1

      D 6 7 8 9 10 11 12 13 14
  0  1  0  0  0  0  0  0  1
```



C. A plotting function.

Enter the following:

```
      ∇ Z←PLOT T
[1]   Z←' *'[1+T] ∇

      R←8 7 6 5 4 3 2 1 0 ¯1

      PLOT R∘.=F 1 2 3 4 5 6 7
```

RELATION TO VECTOR ALGEBRA

APL is a simplification and extension of vector algebra.

ELEMENT-BY-ELEMENT EXTENSION OF FUNCTIONS

```
      X←2 3 5 7
      Y←4 3 2 1
      X+Y
6 6 7 8
      X⋆Y
16 27 25 7
      X×Y
8 9 10 7
```

The model provided by vector addition applies without exception to <u>all</u> dyadic scalar functions, i.e., all scalar functions of two arguments.

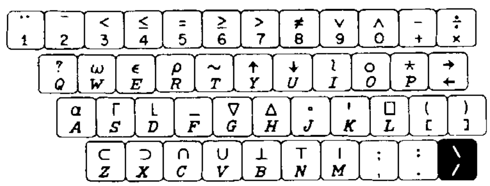
```
      -Y
¯4 ¯3 ¯2 ¯1
      ÷Y
0.25 0.3333 0.5 1
      !Y
24 6 2 1
```

The model provided by negation applies to all monadic scalar functions.

The symbol ! is formed by the sequence backspace '

```
      3×X
6 9 15 21
      3+X
5 6 8 10
      3⋆Y
81 27 9 3
      3⌈Y
4 3 3 3
```

The model provided by scalar multiplication applies without exception to all dyadic scalar functions.



REDUCTION

```
      +/X
17
      ×/X
210
      ⌈/X
7
      +/(X×Y)
34
```

Summation is denoted by the plus sign followed by a slash. This model is followed for all dyadic functions.

The inner product of X and Y.

<u>Exercises</u>: Experiment with variants of the inner product using functions other than + and ×. In particular, use ⌈ (maximum) and ⌊ (minimum).

MONADIC FUNCTIONS

POSITION OF FUNCTION SYMBOL

```
      X←1 2 3 4 5

      -X
¯1 ¯2 ¯3 ¯4 ¯5

      Y←¯2 ¯1 0 1 2
```

The monadic functions (i.e., functions of a single argument) in APL follow the model of negation in algebra: the function symbol appears before its argument. This model is applied strictly to all functions.

```
      |Y
2 1 0 1 2
```

Absolute value or magnitude.

```
      !X
1 2 6 24 120
```

The symbol for factorial is formed by overstriking a quote (uppercase *K*) with a period by the sequence ' backspace .

DOUBLE USE OF SYMBOLS

```
      X-Y
3 3 3 3 3

      -Y
2 1 0 ¯1 ¯2
```

The minus sign denotes both subtraction (dyadic) and negation (monadic). This model is followed for other symbols in APL.

```
      Y÷X
¯2 ¯0.5 0 0.25 0.4

      ÷X
1 0.5 0.3333333333 0.25 0.2

      ⌊.5 1 1.5 2 2.5
0 1 1 2 2
```

Exercises: Experiment with various arguments to determine what monadic function is represented by each of the following symbols: * ⌊ ⌈ ×

ORDER OF EXECUTION

PARENTHESES

```
      (3+4)×(5+6)
77
      (3+(4⌈5))×6
48
```

Parentheses are used to specify the order of execution in a compound expression exactly as in algebra.

RIGHT TO LEFT EXECUTION

Except for the order imposed by parentheses, expressions are evaluated from right to left, following the pattern provided by expressions of the form F G H y (or Log Sin Arctan y) in algebra. For example:

```
      X←¯2 ¯1 0 1 2
      |X
2 1 0 1 2
```

```
      -!|X
¯2 ¯1 ¯1 ¯1 ¯2
      ÷-!|X
¯0.5 ¯1 ¯1 ¯1 ¯0.5
```

The same rule applies to dyadic functions. In particular, there is no hierarchy (such as × is executed before +) among the functions; all are treated alike. For example:

```
      3×4⌈5
15
      3×4+5
27
```

```
      +/X*2
10
      (+/X)*2
0
```

The main advantage of the hierarchy of +, ×, and * in conventional notation is in writing polynomials. However, a polynomial can be written in terms of its vector of coefficients and vector of exponents as follows:

```
      X←5
      +/3 1 4 2×X*0 1 2 3
358
```

Horner's form of the polynomial (for efficient evaluation) and the expression for a continued fraction can be written without parentheses:

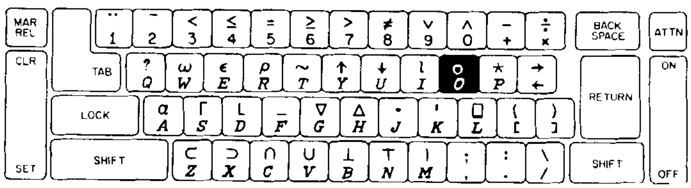
```
      3+X×1+X×4+X×2
358
      3+÷1+÷4+÷2
3.818181818
```

<u>Exercises</u>: Show how the order of execution implies that -/X will yield the alternating sum of X and ÷/X will yield the alternating product.

EVALUATION OF SERIES

The general term of the series expansion of the exponential function is written as (X⋆K)÷!K. Thus:

```
      X←.5                  For a single term.
      K←3

      (X⋆K)÷!K
0.02083333333

      K←0 1 2 3 4           For a set of terms.

      (X⋆K)÷!K
1  0.5  0.125  002083333333  0.002604166667

      S←+/(X⋆K)÷!K          Sum of the set of terms.
      S
1.6484375

      ⋆X                    Correct value of the exponential.
1.648721271

      AS←-/(X⋆K)÷!K         Alternating sum.
      AS
0.6067708333

      ⋆-X
0.6065306597

      S×AS
1.000223796

      C←2×K
      C
0 2 4 6 8

      +/(X⋆C)÷!C            Hyperbolic cosine.
1.127625965
```



<u>Exercises</u>: 1. Use the foregoing scheme to approximate Sinh X, Sin X, and Cos X.

2. Repeat exercise 1 using more terms of the series. For convenience, use the <u>index generator</u> function denoted by ι.

3. Use the expression 1○X to check the result obtained for the approximation to Sin above. Consult page 22 for the notation for the whole family of circular and hyperbolic functions.

4. Evaluate the expression 1 2 3∘.○ι5.

FUNCTION DEFINITION

An expression such as *(X⋆K)÷!K* is a function of two arguments; it can be assigned a name (in this case the name *TERM*) and then used like a primitive function as follows:

```
      ∇ Z←X TERM K
[1]   Z←(X⋆K)÷!K
[2]   ∇

      .5 TERM 3
0.02083333333

      .5 TERM 0 1 2 3 4
1 0.5 0.125 0.02083333333 0.002604166667

      +/.5 TERM ¯1+⍳5
1.6484375
```

A defined function can be used within the definition of another function:

```
      ∇ Z←X SUM K
[1]   Z←+/X TERM K ∇

      .5 SUM ¯1+⍳5
1.6484375

      ∇ Z←COSH X
[1]   Z←X SUM 2×¯1+⍳28 ∇

      COSH 3
10.067662
```

Exercises: 1. Define functions *SIN* and *COS*.

2. Define functions of two arguments using the following example of length of hypotenuse as a model:

```
      ∇ Z←X HYP Y
[1]   Z←((X⋆2)+(Y⋆2))⋆.5 ∇

      3 5 6 HYP 4 12 8
5 13 10
```

3. Explore the scalar functions listed on page 24, particularly the logical functions (and, or, etc.) and use them in function definitions.

HIGHER-DIMENSIONAL ARRAYS

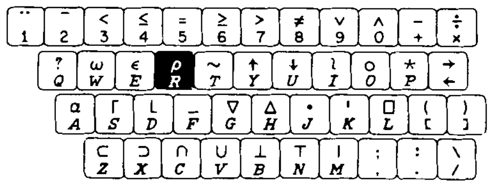
FORMATION

```
      M←3 4⍴⍳12
      M
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

Reshape function.

```
      T←2 3 4⍴⍳24
      T
 1  2  3  4
 5  6  7  8
 9 10 11 12

13 14 15 16
17 18 19 20
21 22 23 24
```

Three-dimensional array.




```
      M×M
  1   4   9  16
 25  36  49  64
 81 100 121 144
```

Scalar functions apply element-by-element.

```
      +/[1]M
15 18 21 24
      ⌈/[2]M
4 8 12
```

Reduction applies over specified coordinate.

```
      ×/M
24 1680 11880
```

Or over last coordinate if none is specified.

SHAPE

```
      ⍴M
3 4
      ⍴T
2 3 4
```

The shape of an array is given by the monadic function ⍴.

```
      ×/⍴T
24
```

Total number of elements in *T*.

Exercises: 1. Determine the behaviour of the reshape function when the right argument is too short to fill the shape specified by the left argument, e.g.: `4 4⍴1 2 3`

2. Experiment with the expressions `N⍴X` and `(N,N)⍴1,N⍴0` where *N* is a scalar integer.

INNER PRODUCT

The ordinary matrix product is a special case of the inner product in which each element of the result is obtained from an expression of the form `+/R×C`, where *R* is the appropriate row of the first argument and *C* is the appropriate column of the second. The role of the functions `+` and `×` is reflected in the notation `+.×` used for the matrix product. For example:

```
      M←(⍳4)∘.≥⍳4
      N←4 4⍴⍳16

      M                          N
 1 0 0 0                     1  2  3  4
 1 1 0 0                     5  6  7  8
 1 1 1 0                     9 10 11 12
 1 1 1 1                    13 14 15 16

      M+.×N
  1  2  3  4
  6  8 10 12
 15 18 21 24
 28 32 36 40


      1 2 3 4+.×M
10 9 7 4

      M+.×1 2 3 4
1 3 6 10
```

In the general inner product, the functions + and × can be replaced by any primitive dyadic functions f and g and each element of the result is then obtained from an expression of the form `f/RgC` For example:

```
      M∨.∧M                Connections of length two in the
 1 0 0 0                   directed graph represented by the
 1 1 0 0                   connection matrix M.
 1 1 1 0
 1 1 1 1

      M∧.=1 1 1 0          Determination of which rows of M
0 0 1 0                    equal the right argument.
```

Exercise: Explore the significance of the expression `(X∘.*¯1+⍳⍴C)+.×C` for vectors *C* and *X* of differing lengths, and also for matrices *C* and *X*.

LINEAR EQUATIONS

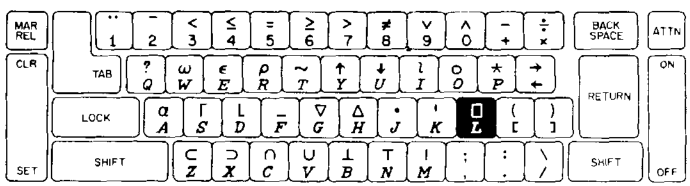
```
      A←?4 4ρ9
      X←3 2 5 7
      A
 6 4 8 5
 9 4 6 7
 8 4 2 7
 7 8 1 7

      B←A+.×X
      B
51 91 117 82

      B⌹A
3 2 5 7
```

Yields result of solving the set of linear equations expressed conventionally as Ax=b. The symbol ⌹ is formed by overstriking the symbol ÷ by the symbol ⎕ (uppercase *L*).

```
      ⌹A
```

Inverse of *A*.

```
 0.16091954023  0.09195402299 ¯0.17241379310  0.08045977011
¯0.09195402299 ¯0.19540229885  0.24137931034 ¯0.04597701149
¯0.19827586207  0.17241379310 ¯0.19827586207  0.27586206897
 0.13936781609 ¯0.06321839080  0.30603448276 ¯0.36781609195

      (⌹A)+.×B
3 2 5 7
```



CURVE FITTING

As shown in the discussion of linear equations, the expression *X←B⌹A* yields a vector *X* such that ∧/*B*=*A*+.×*X* if *A* is nonsingular. If *A* is singular such a value of *X* is not attainable, but *X* is determined so as to minimize (in a least squares sense) the difference between *B* and *A*+.×*X*. In other words, the value of the expression +/(*B*-*A*+.×*X*)⋆2 is minimized. This implies that *A*+.×*B*⌹*A* is the projection of *B* on the subspace spanned by the column vectors of *A*.

LEAST SQUARES POLYNOMIAL FIT

If *X* is a vector and *Y←F X* for some function *F*, and *A* is the matrix *X*∘.⋆0,⍳*D*, then *C←Y⌹X*∘.⋆0,⍳*D* yields the coefficients of the polynomial of degree *D* which best fits the function *F*. For example:

```
      X←1 2 3 4
      Y←X⋆3
      Y
1 8 27 64
      X∘.⋆0,⍳2
     1     1     1
     1     2     4
     1     3     9
     1     4    16
      C←Y⌹X∘.⋆0,⍳2
      C
10.5 ¯16.7 7.5
      (X∘.⋆0,⍳2)+.×C
1.3 7.1 27.9 63.7
      Y⌹X∘.⋆0,⍳3
1.372E¯14 ¯2.422E¯14 1.232E¯14 1
```

OTHER FUNCTIONS

The coefficients for sets of functions other than powers can be obtained in a similar way. For example:

```
      X∘.×⍳4
  1  2  3  4
  2  4  6  8
  3  6  9 12
  4  8 12 16
```

contains the multiples (harmonics) of *X* up to *D* and the matrix 1○*X*∘.×0,⍳*D* therefore contains the sines of the harmonics and the expression *Y*⌹1○*X*∘.×⍳*D* yields the coefficients for a best fit to *Y* by a linear combination of sines of multiples of *X*.

SELECTION FUNCTIONS

INDEXING

```
      P←2 3 5 7 11
      M←3 4ρι12
      M
  1  2  3  4
  5  6  7  8
  9 10 11 12

      P[3]
5
      P[2 3 4]
3 5 7

      P[4 1 5 2 3]        Permutation.
7 2 11 3 5

      M[2;3]              Single element.
7

      M[2 3;3 2 1]        Set of rows and columns.
  7  6  5
 11 10  9

      M[2;]               Entire row.
5 6 7 8

      M[;3]               Entire column.
3 7 11

      M[;3 2]             Entire columns.
  3  2
  7  6
 11 10
```

TAKE AND DROP

```
      2↑P                 ¯3↑P
2 3                   5 7 11
      2↓P                 2 3↑M
5 7 11                 1 2 3
                       5 6 7
```

Exercise: Explore the selection and other functions in the table of mixed functions on page 23, particularly the decode, transpose, compress, and rotate functions. Use vectors of characters in some of your examples (see page 6).

ITERATION

BRANCHING

A sequence of lines occurring in a function definition is executed in sequence except that a branch (denoted by an expression of the form →*S*) causes line number *S* to be executed next. For example:

```
      ∇ Z←F X
[1]   Z←X
[2]   Z
[3]   Z←3+X
[4]   →2 ∇

      F 7
7
10
13
16
```

This function will repeat lines 2-4 without stopping unless interrupted by depressing the Attention button at the upper right of the keyboard.

CONDITIONAL BRANCH

A change in the value of the argument of a branch will cause a branch to a different line and the sequence can therefore be controlled. A branch to a non-existent line terminates the function. For example:

```
      ∇ Z←BIN N
[1]   Z←1
[2]   Z←(Z,0)+0,Z
[3]   →2×N≥ρZ ∇

      BIN 4
1 4 6 4 1
```

TRACING

The execution of any desired lines of a function can be traced as shown in the following example:

```
      TΔBIN←2 3
      Z←BIN 2
BIN[2] 1 1
BIN[3] 2
BIN[2] 1 2 1
BIN[3] 0

      TΔBIN←0
      BIN 2
1 2 1
```

IDENTITIES

APL is rich in useful identities, and the serious user should become familiar with the more important of them.

DUALITY

The following expressions are identities, i.e., they have the value 1 (true) for any vector arguments within the domains of the indicated functions:

```
(⌈/A)=-⌊/-A                    (∧/L)=~∨/~L
(⌊/A)=-⌈/-A                    (≠/L)=~=/~L
```

Duality also applies to matrix arguments in inner products:

```
∧/,(C∨.∧D)=~(~C)∧.∨~D
∧/,(M⌊.⌈N)=-(-M)⌈.⌊-N
∧/,(C∧.=D)=~(~C)∨.≠~D
```

ASSOCIATIVITY

```
∧/,((M+.×N)+.×P)=M+.×(N+.×P)
∧/,((C∨.∧D)∨.∧E)=C∨.∧(D∨.∧E)
∧/,((M⌊.+N)⌊.+P)=M⌊.+(N⌊.+P)
```

DISTRIBUTIVITY

```
∧/,(M+.×(N+P))=(M+.×N)+(M+.×P)
∧/,(C∨.∧(D∨E))=(C∨.∧D)∨(C∨.∧E)
∧/,(M⌊.+(N⌊P))=(M⌊.+N)⌊(M⌊.+P)
```

PARTITIONING

If *U* is a logical vector then:

```
∧/,(M+.×N)=((U/M)+.×U/[1]N)+((~U)/M)+.×(~U)/[1]N
∧/,(C∨.∧D)=((U/C)∨.∧U/[1]D)+((~U)/C)∨.∧(~U)/[1]D
```

<u>Exercises</u>: Test the identities by evaluating them for sample values of the arguments. Then attempt to generalize them. For example:

What is the dual of the <u>not</u>-<u>and</u> function ⍲?

What is the rule for determining whether any inner product (such as ⌈.+ or ∧.=) is associative?

To what inner products does the partitioning identity apply?

Test (and generalize) the following relation between inner and outer products: `∧/,(M+.×N)=+/1 3 3 2⍉M∘.×N`

PROOFS

The following format will be used for proofs: a listing of two or more expressions on successive lines asserts that the expressions are equivalent. Notes at the right give the bases of the assertions. For example, the following develops a general form for the distributivity of × over + for vectors *V* and *W* and scalar *S*:

```
Thm 1:  +/S×W
        S×+/W            Distributivity of × over +

Thm 2:  +/V∘.×W
        V×+/W            Thm 1 applied to each element of V

Thm 3:  +/+/V∘.×W
        +/V×(+/W)        Thm 2
        (+/V)×(+/W)      Thm 1 (with +/W for S and V for W)
```

Exercises: 1. Illustrate the foregoing theorems by evaluating the expressions for assigned values of the arguments.

2. Illustrate and prove the following theorems (for scalar *X* and vectors *A*, *B*, *P*, and *Q*):

```
Thm 4:  (A×P)∘.×(B×Q)       (Show that the I,Jth element of
        (A∘.×B)×(P∘.×Q)     the first matrix equals the I,Jth
                            element of the second)
Thm 5:  (X*A)∘.×(X*B)
        X*(A∘.+B)
```

THE PRODUCT OF POLYNOMIALS

Let *P* be the following polynomial function:

```
      ∇ Z←C P X
[1]     Z←+/C×X*¯1+⍳⍴C∇
```

The coefficient vector of the product of polynomials with coefficients *C* and *D* is obtained by summing the table *C*∘.×*D* as shown on the right. The rationale for this rests on the matrix of exponents *M* also shown on the right. The necessary theorem follows:

```
      C←3 1 4
      D←2 0 5 3
      C∘.×D
6  0 15  9
2  0  5  3
8  0 20 12

6  2 23 14 23 12

      M←(¯1+⍳⍴C)∘.+(¯1+⍳⍴D)
      M
0 1 2 3
1 2 3 4
2 3 4 5
```

```
(C P X)×(D P X)
(+/C×X*¯1+⍳⍴C)×(+/D×X*¯1+⍳⍴D)          Definition of P
+/+/(C×X*¯1+⍳⍴C)∘.×(D×X*¯1+⍳⍴D)        Thm 3
+/+/(C∘.×D)×(X*¯1+⍳⍴C)∘.×(X*¯1+⍳⍴D)    Thm 4
+/+/(C∘.×D)×X*(¯1+⍳⍴C)∘.+(¯1+⍳⍴D)      Thm 5
```

REFERENCE MATERIAL

For complete reference material (including the establishment and use of libraries of work to be saved for later use) the reader is referred to the manual mentioned on page 1. The following three pages contain a table of all error reports, a table of all scalar functions, and a table of all mixed functions. This page offers advice on difficulties frequently encountered by the beginner.

CORRECTIONS

Every entry must be concluded by a carriage return to signal the end of the entry to the computer. To correct any typing error detected before striking the carriage return, backspace to the beginning of the error, strike the attention button (which effectively erases everything from that point to the right, marks the point with a caret, and spaces the paper up) and then continue typing.

If the computer gives no response to one or more entries, you have probably entered an unmatched quote; try entering a single quote (uppercase *K*) followed by a carriage return.

REVISION AND DISPLAY OF FUNCTIONS

To revise or display a function already defined first enter a ∇ followed by the <u>name</u> of the function only. This <u>reopens</u> the definition.

The function may then be displayed by entering [⎕]. For example, if the function *BIN* of page 19 is already defined then:

```
      ∇BIN
[4]    [⎕]
    ∇ Z←BIN N
[1]   Z←1
[2]   Z←(Z,0)+0,Z
[3]   →2×N≥ρZ
    ∇
[4]
```

Line 2 can now be revised by entering

```
      [2] Z←(Z,0)-0,Z
```

Finally, the function definition may be closed by entering ∇.

TYPE	Cause; CORRECTIVE ACTION
CHARACTER	Illegitimate overstrike.
DEPTH	Excessive depth of function execution. CLEAR STATE INDICATOR.
DOMAIN	Arguments not in the domain of the function.
DEFN	Misuse of ∇ or ⎕ symbols: 1. ∇ is in some position other than the first. 2. The function is pendent. DISPLAY STATE INDICATOR AND CLEAR AS REQUIRED. 3. Use of other than the function name alone in reopening a definition. 4. Improper request for a line edit or display.
INDEX	Index value out of range.
LABEL	Name of already defined function used as a label, or colon used other than in function definition and between label and statement.
LENGTH	Shapes not conformable.
RANK	Ranks not conformable.
RESEND	Transmission failure. RE-ENTER. IF CHRONIC, REDIAL OR HAVE TERMINAL OR PHONE REPAIRED.
SYNTAX	Invalid syntax; e.g., two variables juxtaposed; function used without appropriate arguments as dictated by its header; unmatched parentheses.
SYMBOL TABLE FULL	Too many names used. ERASE SOME FUNCTIONS OR VARIABLES, THEN SAVE, CLEAR, AND COPY.
SYSTEM	Fault in internal operation of APL\ 360. RELOAD OR SAVE, CLEAR, AND COPY. SEND TYPED RECORD, INCLUDING ALL WORK LEADING TO THE ERROR, TO THE SYSTEM MANAGER.
VALUE	Use of name which has not been assigned a value. ASSIGN A VALUE TO THE VARIABLE, OR DEFINE THE FUNCTION.
WS FULL	Workspace is filled (perhaps by temporary values produced in evaluating a compound expression). CLEAR STATE INDICATOR, ERASE NEEDLESS OBJECTS, OR REVISE CALCULATIONS TO USE LESS SPACE.

ERROR REPORTS

(Reprinted from reference [2])

Monadic form *fB*: Definition or example	Monadic form *fB*: Name	f	Dyadic form *AfB*: Name	Dyadic form *AfB*: Definition or example
+B ↔ 0+B	Plus	+	Plus	2+3.2 ↔ 5.2
-B ↔ 0-B	Negative	-	Minus	2-3.2 ↔ ¯1.2
×B ↔ (B>0)-(B<0)	Signum	×	Times	2×3.2 ↔ 6.4
÷B ↔ 1÷B	Reciprocal	÷	Divide	2÷3.2 ↔ 0.625
(see Ceiling/Floor table below)	Ceiling	⌈	Maximum	3⌈7 ↔ 7
	Floor	⌊	Minimum	3⌊7 ↔ 3
*B ↔ (2.71828..)*B	Exponential	*	Power	2*3 ↔ 8
⍟*N ↔ N ↔ *⍟N	Natural logarithm	⍟	Logarithm	A⍟B ↔ Log B base A A⍟B ↔ (⍟B)÷⍟A
∣¯3.14 ↔ 3.14	Magnitude	∣	Residue	(see Residue table below)
!0 ↔ 1 !B ↔ B×!B-1 or !B ↔ Gamma(B+1)	Factorial	!	Binomial coefficient	A!B ↔ (!B)÷(!A)×!B-A 2!5 ↔ 10 3!5 ↔ 10
?B ↔ Random choice from ⍳B	Roll	?	Deal	A Mixed Function (See Table 2)
○B ↔ B×3.14159...	Pi times	○	Circular	See Table at left
∼1 ↔ 0 ∼0 ↔ 1	Not	∼		
		∧	And	(see logical table below)
		∨	Or	
		⍲	Nand	
		⍱	Nor	
		<	Less	Relations
		≤	Not greater	Result is 1 if the
		=	Equal	relation holds, 0
		≥	Not less	if it does not:
		>	Greater	3≤7 ↔ 1
		≠	Not Equal	7≤3 ↔ 0

B	⌈B	⌊B
3.14	4	3
¯3.14	¯3	¯4

Case	A∣B
A≠0	B-(∣A)×⌊B÷∣A
A=0,B≥0	B
A=0,B<0	Domain error

A	B	A∧B	A∨B	A⍲B	A⍱B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

(-A)○B	A	A○B
(1-B*2)*.5	0	(1-B*2)*.5
Arcsin B	1	Sine B
Arccos B	2	Cosine B
Arctan B	3	Tangent B
(¯1+B*2)*.5	4	(1+B*2)*.5
Arcsinh B	5	Sinh B
Arccosh B	6	Cosh B
Arctanh B	7	Tanh B

Table of Dyadic ○ Functions

SCALAR FUNCTIONS

(Reprinted from reference [2])

Name	Sign[1]	Definition or example[2]
Size	ρA	ρP ↔ 4 ρE ↔ 3 4 ρ5 ↔ ι0
Reshape	VρA	Reshape A to dimension V 3 4ρι12 ↔ E 12ρE ↔ ι12 0ρE ↔ ι0
Ravel	,A	,A ↔ (×/ρA)ρA ,E ↔ι12 ρ,5 ↔ 1
Catenate	V,V	P,ι2 ↔ 2 3 5 7 1 2 'T','HIS' ↔ 'THIS'
Index[3,4]	V[A] M[A;A] A[A;.. ..;A]	P[2] ↔3 P[4 3 2 1] ↔7 5 3 2 E[1 3;3 2 1] ↔ 3 2 1 11 10 9 E[1;] ↔ 1 2 3 4 E[;1] ↔ 1 5 9 'ABCDEFGHIJKL'[E] ↔ ABCD EFGH IJKL
Index generator[3]	ιS	First S integers ι4 ↔ 1 2 3 4 ι0 ↔ an empty vector
Index of[3]	VιA	Least index of A in V, or 1+ρV Pι3 ↔2 4 4ι4 ↔ 1 PιE ↔ 5 1 2 5 3 5 4 5 5 5 5 5
Take	V↑A	Take or drop |V[I] first (V[I]≥0) or last (V[I]<0) elements of coordinate I 2 3↑X ↔ ABC EFG ¯2↑P ↔ 5 7
Drop	V↓A	
Grade up[3,5]	⍋A	The permutation which would order A (ascending or descending) ⍋3 5 3 2 ↔ 4 1 3 2
Grade down[3,5]	⍒A	⍒3 5 3 2 ↔ 2 1 3 4
Compress[5]	V/A	1 0 1 0/P ↔ 2 5 1 0 1 0/E ↔ 1 3 5 7 9 11 1 0 1/[1]E ↔ 1 2 3 4 ↔ 1 0 1⌿E 9 10 11 12
Expand[5]	V\A	1 0 1\ι2 ↔ 1 0 2 1 0 1 1 1\X ↔ A BCD E FGH I JKL
Reverse[5]	⌽A	⌽X ↔ DCBA HGFE LKJI ⌽[1]X ↔ ⊖X ↔ IJKL EFGH ABCD ⌽P ↔ 7 5 3 2
Rotate[5]	A⌽A	3⌽P ↔ 7 2 3 5 ↔ ¯1⌽P 1 0 ¯1⌽X ↔ BCDA EFGH LIJK
Transpose	V⍉A	Coordinate I of A becomes coordinate V[I] of result 2 1⍉X ↔ AEI BFJ CGK DHL 1 1⍉E ↔ 1 6 11
	⍉A	Transpose last two coordinates ⍉E ↔ 2 1⍉E
Membership	A∊A	ρW∊Y ↔ ρW E∊P ↔ 0 1 1 0 1 0 1 0 0 0 0 0 P∊ι4 ↔ 1 1 0 0
Decode	V⊥V	10⊥1 7 7 6 ↔ 1776 24 60 60⊥1 2 3 ↔ 3723
Encode	V⊤S	24 60 60⊤3723 ↔ 1 2 3 60 60⊤3723 ↔ 2 3
Deal[3]	S?S	W?Y ↔ Random deal of W elements from ιY

MIXED FUNCTIONS

(Reprinted from reference [2])

1. Restrictions on argument ranks are indicated by: S for scalar, V for vector, M for matrix, A for Any. Except as the first argument of $S\iota A$ or $S[A]$, a scalar may be used instead of a vector. A one-element array may replace any scalar.

2. Arrays used in examples: $P \leftrightarrow 2\ 3\ 5\ 7$ $\quad E \leftrightarrow \begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{matrix}$ $\quad X \leftrightarrow \begin{matrix} ABCD \\ EFGH \\ IJKL \end{matrix}$

3. Function depends on index origin.

4. Elision of any index selects all along that coordinate.

5. The function is applied along the last coordinate; the symbols ⌿, ⍀, and ⊖ are equivalent to /, \, and ⌽, respectively, except that the function is applied along the first coordinate. If $[S]$ appears after any of the symbols, the relevant coordinate is determined by the scalar S.

Notes to Table of Mixed Functions

(Reprinted from reference [2])

IBM

IBM Cambridge Scientific Center
545 Technology Square
Cambridge, Massachusetts 02139

IBM Houston Scientific Center
6900 Fannin Street
Houston, Texas 77025

IBM Los Angeles Scientific Center
1930 Century Park W.
Los Angeles, California 90067

IBM Palo Alto Scientific Center
2670 Hanover Street
Palo Alto, California 94304

IBM Philadelphia Scientific Center
3401 Market Street
Philadelphia, Pennsylvania 19104