ELEMENTARY ALGEBRA

IBM PHILADELPHIA SCIENTIFIC CENTER
Data Processing Division
TECH. REPORT NO. 320-3001 JUNE 1971

K. E. IVERSON

PHILADELPHIA§ SCIENTIFIC CENTER

TECHNICAL REPORTS

† 320-2900 *K. Spielberg*, An Algorithm for the Simple Plant Location Problem with Some Side Conditions, 48 p., November 1967

† ■ 320-2901 *J. Greenstadt*, Variations on Variable-Metric Methods, 23 p., May 1967

■ 320-2902 *Y. Bard*, A Function Maximization Method with Application to Parameter Estimation, 16 p., June 1967

■ * 320-2903 *A.N. Bomberault & W.W. White*, Scheduling Empty Box Cars, 23 p., December 1966

■ 320-2904 *B.D. Gavril*, S.C.U. Special Control Unit Information Brief No. 1, 12 p., July 1967

■ 320-2905 *B.D. Gavril*, S.C.U. Special Control Unit Information Brief No. 2, 29 p., July 1967

† ■ 320-2906 *O. Gurel & L. Lapidus*, Stability Via Liapunov's Second Method, 54 p., July 1967

320-2907 *O. Gurel*, Stability Analysis of N-Dimensional Systems Through Topology of Velocity-Space, 29 p., August 1967

† ■ 320-2908 *E.S. Savas*, Computers in Urban Air Pollution Control Systems, 51 p., August 1967

320-2909 *J. Greenstadt*, Cell Discretization II - Discrete Equations (Elliptic Case), 27 p., August 1967

† * 320-2910 *H. Eisenpress & J. Greenstadt*, The Estimation of Non-Linear Econometric Systems, 15 p., August 1967

† ■ * 320-2911 *C.E. Lemke & K. Spielberg*, Direct Search Zero-One and Mixed Integer Programming, 62 p., September 1967

† ■ * 320-2912 *J. Greenstadt*. A Richocheting Gradient Method for Non-Linear Optimization, 26 p., September 1967

† ■ 320-2913 *Y. Bard*, On a Numerical Instability of Davidon-Like Methods, 4 p., September 1967

320-2914 *J. Greenstadt*, Cell Discretization III - Treatment of Discrete Equations (Hyperbolic and Parabolic Case), 13 p., September 1967

† ■ * 320-2915 *J. Rubin*, Optimal Classification into Groups - An Approach for Solving the Taxonomy Problem, 62 p., November 1967

† 320-2916 *S.G. Hahn*, On the Optimal Cutting of Defective Glass Sheets, 20 p., October 1967

† 320-2917 *R.G. Loomis, & J.J. Lorenzo*, Experiments in Mapping with a Geo Space Plotter, 18 p., September 1967

† 320-2918 *L.S. Woo*, An Algorithm for Straight Line Representation of Simple Planar Graphs, 19 p., October 1967

† 320-2919 *O. Gurel & L. Lapidus*, The Maximum Principle and Discrete Systems, 22 p., November 1967

† ■ 320-2920 *M. Guignard*, Generalized Kuhn-Tucker Conditions for Mathematical Programming in a Banach Space, 17 p., November 1967

† ■ 320-2921 *O. Gurel*, Marker Layout Problem Via Graph Theory, 42 p., January 1968

■ † * 320-2922 *H.P. Friedman & J. Rubin*, On Some Invariant Criteria for Grouping Data, 69 p., November 1967

† * 320-2923 *P.G. Comba*, A Language for 3-Dimensional Geometric Processing - Written Form, 40 p., December 1967

† * 320-2924 *P.G. Comba*, A Procedure for Detecting Intersections of 3-Dimensional Objects, 18 p., January 1968

320-2925 *A.R. Colville*, Mathematical Programming Codes, 25 p., January 1968

320-2926 *A.R.D. Norman & M.J. Dowling*, Railroad Car Inventory: Empty Woodrack Cars on the Louisville and Nashville, 69 p., June 1967

■ 320-2927 *G. Gordon*, The Development of Simulation Languages, 34 p., January 1968

† 320-2928 *F. Freudenstein & L.S. Woo*, Kinematics of the Human Knee Joint, 30 p., January 1968

† 320-2929 *K. Spielberg*, Plant Location With Generalized Search Origin, 32 p., March 1968

† ■ 320-2930 *J.F. Raimond*, An Algorithm for the Exact Solution of the Machine Scheduling Problem, 40 p., March 1968

■ * 320-2931 *R. Shareshian & K. Spielberg*, The Mixed Integer Algorithm of N. Driebeek, 18 p., July 1966

* 320-2932 *R. Shareshian*, A Modification of the Mixed Integer Algorithm of N. Driebeek, 11 p., July 1966

■ * 320-2933 *Y. Bard*, Production-Transportation-Marketing Model, 22 p., October 1966

* 320-2934 *J. Greenstadt*, Further Experiments in Triangularizing Matrices, 19 p., November 1966

■ 320-2935 *G. Gordon & K. Zelin*, A Simulation Study of Emergency Ambulance Service in New York City, 37 p., February 1968

320-2936 *B.D. Gavril*, S.C.U. Special Control Unit Information Brief No. 3, xx+ 173 p., March 1968

† 320-2937 *O. Gurel & L. Lapidus*, A Guide to Methods for the Generation of Liapunov Functions, 82 p., March 1968

† ■ 320-2938 *K. Spielberg*, Enumerative Methods for Integer and Mixed Integer Programming, 115 p., March 1968

320-2939 *S. Poley*, Mesh Analysis of Piping Systems, 136 p., March 1968

* 320-2940 *R.G. Loomis*, A Design Study on Graphics Support in a FORTRAN Environment, 33 p., May 1966

■ * 320-2941 *J. Greenstadt*, On the Problem of Fairing, 58 p., December 1966

† ■ * 320-2942 *J. Greenstadt*, On the Relative Efficiencies of Gradient Methods, 16 p., January 1967

* 320-2943 *O. Gurel & M.M. Salah*, Stability Analysis Reports, Report No. 1, 25 p., Report No. 2, 16 p., February 1967

* 320-2944 *J. Greenstadt*, Cell Discretization I - Variational Basis, 27 p., February 1967

† ■ 320-2945 *O. Gurel*, Additional Considerations on Marker Layout Problem Via Graph Theory, 34 p., April 1968

320-2946 *D.W. Webber & W.W. White*, An Algorithm for Solving Large Structured Linear Programming Problems, 28 p., April 1968

† ■ 320-2947 *L.S. Woo & F. Freudenstein*, On the Curves of Synthesis in Plane, Instantaneous Kinematics, 40 p., May 1968

† ■ 320-2948 *Y. Bard & J.L. Greenstadt*, A Modified Newton Method for Optimization with Equality Constraints, 13 p., May 1968

■ 320-2949 *A.R. Colville*, A Comparative Study on Nonlinear Programming Codes, 77 p., June 1968

320-2950 *O. Gurel & M. Guignard*, Structure of Constrained Optimality With Respect to Higher Order Derivatives, 25 p., June 1968

320-2951 *H. Salkin & K. Spielberg*, Adaptive Binary Programming, 90 p., June 1968

† ■ 320-2952 *A.M. Bomberault & W.W. White*, Networks and Transportation: The Empty Freight Car Allocation Problem, 58 p., July 1968

† * 320-2953 *L.S. Woo*, Type Synthesis in Plane Linkages, 39 p., July 1966

■ 320-2954 *S. Poley & C. Strauss*, A Three Dimensional Piping Design Program, 30 p., July 1968

■ 320-2955 *Y. Bard*, Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems, 58 p., September 1968

320-2956 *H. Eisenpress & A. Bomberault*, Efficient Symbolic Differentiation using PL/I-Formac, 25 p., September 1968

320-2957 *O. Gurel*, Stability of the Pair (X,f), 14 p., November 1968

† 320-2958 *S. Gorenstein*, Printing Press Scheduling for Multi-Edition Periodicals, 22 p., November 1968

320-2959 *C. Hao, L. Woo, V. Vitagliano, & F. Freudenstein*, Analysis of Control-Mechanism Performance Criteria for an Above Knee-Leg Prosthesis, 51 p., December 1968

■ 320-2960 *E. Balas*, Project Scheduling with Resource Constraints, 23 p., January 1969

† 320-2961 *O. Gurel & L. Lapidus*, Topology of Velocity-Space for Stability Analysis, 12 p., January 1969

■ 320-2962 *J. Cord*, Generalized Upper Bounds with Upper Bounded Variables, 47 p., January 1969

(continued on back cover)

IBM PHILADELPHIA SCIENTIFIC CENTER TECHNICAL REPORT NO. 320-3001, JUNE 1971

ELEMENTARY ALGEBRA

K. E. IVERSON

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

PREFACE

The present text treats the usual topics expected in a second course in high school algebra. It differs from conventional treatments in the following respects:

1. The notation used is simple and precise and applies to arrays (vectors and matrices) in a simple and uniform manner.

2. Arrays are used extensively to give a graphic view of functions by displaying the patterns produced by applying them to vectors. They are also used to clarify topics which use vectors directly, such as linear functions and polynomials.

3. The precision of the notation permits an algorithmic treatment of the material. In particular, every expression in the book can be executed directly by simply typing it on an appropriate computer terminal. Hence if a computer is available, it can be used by students for individual or collective exploration of relevant mathematical functions in the manner discussed in Berry et al [7]. Even if a computer is not available, the algorithmic treatment presents the essentials of computer programming in a mathematical light, i.e., as the precise definition and application of functions.

4. The algorithmic approach is the same as that used in my *Elementary Functions* [3], a text which can be used as a continuation in topics such as the slope (derivative) of functions, and the circular, hyperbolic, exponential, and logarithmic functions.

5. The organization of topics follows a pattern suggested by considering algebra as a language; in particular, the treatment of formal identities is deferred until much work has been done in the reading and writing of algebraic sentences. These matters are discussed fully in the Appendix *Algebra as a Language*, and any teacher may be well-advised to begin by reading this appendix.

The pace of the text is perhaps best suited to a second year course, but it can also be used for a first year course since the early chapters contain all of the essentials such as the introduction of the negative and rational numbers. When used as a second year text, these early chapters can serve not only as a brief review, but also as an introduction to the notation used.

This text grew out of a summer project undertaken in 1969 in collaboration with my colleagues Adin Falkoff and Paul Berry of IBM, and with five high school teachers-- Mr. John Brown, now of Dawson College, Montreal; Mr. Nathaniel Bates, of Belmont Hill School, Belmont, Massachusetts; Miss Linda Alvord, of Scotch Plains High School, Scotch Plains, N.J.; and Sisters Helen Wilxman and Barbara Brennan, of Mary Immaculate School, Ossining, N.Y. I am indebted to all of these people for much fruitful discussion, and particularly to Messrs. Falkoff and Berry for helping to set and maintain the direction of the project.

TABLE OF CONTENTS

ILLUSTRATIONS

Chapter 1

THE LANGUAGE OF MATHEMATICS

1.1. INTRODUCTION

Algebra is the language of mathematics. It is therefore an essential topic for anyone who wishes to continue the study of mathematics. Moreover, enough of the language of algebra has crept into the English language to make a knowledge of some algebra useful to most non-mathematicians as well. This is particularly true for people who do advanced work in any trade or discipline, such as insurance, engineering, accounting, or electrical wiring. For example, instructions for laying out a playing field might include the sentence, "To verify that the corners are square, note that the length of the diagonal must be equal to the square root of the sum of the squares of the length and the width of the field," or alternatively, "The length of the diagonal must be $\sqrt{\ell^2 + w^2}$." In either case (whether expressed in algebraic symbols or in the corresponding English words), the comprehension of such a sentence depends on a knowledge of some algebra.

Because algebra is a language, it has many similarities to English. These similarities can be helpful in learning algebra, and they will be noted and explained as they occur. For instance, the integers or counting numbers (1, 2, 3, 4, 5, 6, . . .) in algebra correspond to the concrete nouns in English, since they are the basic things we discuss, and perform operations upon. Furthermore, functions in algebra (such as + (plus), × (times), and - (subtract) correspond to the verbs in English, since they <u>do</u> something to the nouns. Thus, 2+3 means "add 2 to 3," and (2+3)×4 means "add 2 to 3 and then multiply by 4." In fact, the word "function" (as defined, for example, in the American Heritage Dictionary), is descended from an older word meaning, "to execute," or "to perform."

When the language of algebra is compared to the language of English, it is in certain respects much simpler, and in other respects more difficult. Algebra is simpler in that the basic algebraic sentence is an instruction to do something, and algebraic sentences (usually called expressions) therefore correspond to <u>imperative</u> English sentences (such as "Close the door."). For example, 2+3 means "add 2 and 3," and *YEAR*←1970 means "assign to the name *YEAR* the value 1970," and *Y*←1970 means "assign to the name *Y* the value 1970." Since imperative sentences form only a small and relatively simple part of English, the language of algebra is in this respect much simpler.

Algebra is also simpler in that it permits less freedom in the ways you can express a particular function. For example, "subtract 2 from 4" would normally be written in algebra only as 4-2, whereas in English it could be expressed in many ways such as "take the number 2 and subtract it from the number 4," or "compute the difference of the integers 4 and 2."

The most difficult aspect of traditional presentations of algebra is the early emphasis on <u>identities</u>, or the equivalence of different expressions. For example, the expressions (5+7)×(5+7) and (5×5)+(2×5×7)+(7×7) are <u>equivalent</u> in the sense that, although they involve a different sequence of funtions, they each yield the same result. English also offers equivalent expressions. For example, "The dog bit the man" is equivalent to "The man was bitten by the dog." It is not that the rules for determining equivalence in algebra are more difficult than in English; on the contrary, they are so much simpler that their study is more rewarding and therefore more attention is given to equivalences in algebra than in English.

In the present treatment this aspect of algebra (that is, the study of identities or equivalence of expressions), is delayed until the student has devoted more attention to the reading, writing, and evaluation of algebraic expressions.

The exercises form an important part of the development, and the point at which the reader should be prepared to attempt each group of exercises is indicated in the right margin. For example, the first such marginal note appears as ⊞1-6 and indicates that Exercises 1 to 6 of this chapter may be attempted at that point.

1.2. EXPRESSIONS AND RESULTS

The expression 2+3 when evaluated produces the result 5. Such a fact will be written in the following form:

```
      2+3
5
```

and will be read aloud as "2 plus 3 makes 5." The following examples would be read in a similar way:

```
      7+12
19

      8×4
32
```

Where there is more than one function to be executed, parentheses are used to indicate which is to be done first. Thus the expression

```
(2+3)×4
```

is evaluated by first performing the function within the parentheses (that is, 2+3), and then multiplying the result by 4. The final result is therefore 20, as shown below:

```
      (2+3)×4
20
```

The foregoing is read aloud as "quantity 2+3, times 4." The word "quantity" indicates that the first expression following it is to be executed first. That is, you are to find the result of 2+3 before attempting to execute the function "times".

The steps in the execution of an expression may be displayed on successive lines, substituting at each line the value of part of the expression above it as illustrated below:

```
     |(2+3)×4
     |
     |  5  ×4
20
```

The vertical line drawn to the left of the first two lines indicates that they are <u>equivalent</u> statements, either of which would produce the result 20 shown on the final line. The whole statement would be read aloud as "Quantity 2 plus 3 times 4 is equivalent to 5 times 4 which makes 20. The following examples would be read in a similar way as shown on the right:

```
     |(2+3)×(5+4)          Quantity 2 plus 3 times quantity 5
     |                        plus 4
     |                     is equivalent to
     |  5  ×  9            5 times 9
45                         which makes 45

     |((2×3)+(5×4))×2      Quantity 2 times 3 plus quantity 5
     |                        times 4, all times 2
     |                     is equivalent to
     |(  6  +  20  )×2     quantity 6 plus 20 times 2
     |                     is equivalent to
     |     26       ×2     26 times 2
52                         which makes 52
```

The last example illustrates the difficulty of expressing in English the sequence of execution that is expressed so simply by parentheses in algebra, that is, when parentheses are "nested" within other parentheses even the use of the word "quantity" does not suffice and one resorts to expressions such as "all times 2". The main point is this: in learning any new language (such as algebra) it is important to re-express statements in a more familiar language (such as English); however, certain things are so awkward to express in the old language that it becomes important to learn to "think" in the new language. ⎕1-6

The expression 2+3×4, written without parentheses, could be taken to mean either (2+3)×4 (which makes 20), or 2+(3×4) (which makes 14). To avoid such ambiguity we make the following rule: when two or more functions occur in succession with no parentheses between them, the rightmost function is executed first. For example:

```
     |2+3×4
     |
     |2+ 12
14

     |1+2×3+4×5
     |
     |1+2×3+ 20
     |
     |1+2× 23
     |
     |1+  46
47

     |(1+2×3)+4×5
     |
     |(1+ 6)+ 20
     |
     |   7  +20
27
```

⎕7-12

1.3. NAMES

Consider the following statements:

```
      (1+3+5+7+9)×2
50
      (1+3+5+7+9)×3
75
      (1+3+5+7+9)×4
100
```

Since the expression 1+3+5+7+9 occurs again and again in the foregoing statements, it would be convenient to give some short name to the result produced by the expression, and then use that short name instead of the expression. This is done as follows:

```
      IT←1+3+5+7+9
      IT×2
50
      IT×3
75
      IT×4
100
      IT
25
```

The foregoing would be read aloud as follows: "The name *IT* is assigned the value of the expression 1+3+5+7+9. *IT* times 2 makes 50. *IT* times 3 makes 75. *IT* times 4 makes 100. *IT* makes 25."

Names can be chosen at will. For example:

```
      LENGTH←5
      WIDTH←4
      LENGTH×WIDTH
20
      AREA←LENGTH×WIDTH
      AREA
20

      PRICE←5
      QUANTITY←4
      PRICE×QUANTITY
20
```

Mathematicians usually prefer to use short names like *L* or *W* or *X* or *Y*, perhaps because this brings out the underlying structure or similarity of expressions which may deal with different names. Consider, for example, the following sequence:

```
      X←5
      Y←4
      X×Y
20
```


If *X* is taken to mean length and *Y* is taken to mean width, then the result is the area of the corresponding rectangle; but if *X* is taken to mean price and *Y* is taken to mean quantity, then the result is the total price. This makes clear that there is some similarity between the calculations of an area from length and width and the calculation of total price from price and quantity.

The names used in algebra are also called <u>variables</u>, since they may <u>vary</u> in the sense that the same name may represent different values at different times. For example:

```
      X←3
      X×X
9
      X←5
      X×X
25
```

This ability to vary distinguishes a name like *X* from a symbol like 5 which always represents the same value and is therefore called a <u>constant</u>.

It is interesting to note that the <u>variables</u> in algebra correspond to the <u>pronouns</u> in English. For example, the sentence "close it" is meaningless until one knows to what "it" refers. This reference is usually made clear by a preceding sentence. For example, "See the door. Close it" is unambiguous because the first sentence makes it clear that "it" refers to "the door". Similarly, in algebra the expression *IT*+5 cannot be evaluated unless the value to which *IT* refers is known. In algebra this reference is made clear in one way, by the use of the assignment represented by the symbol ←. For example:

```
      IT←3
      IT+5
8
```

The same name *IT* can stand for different values at different times just as the pronoun "it" can refer to different things at different times.

1.4. OVER NOTATION

It is often necessary to take the sum over a whole list of numbers. For example, if the list consists of the numbers 1 3 5 7 9 11, then their sum could be written as

```
      1+3+5+7+9+11
36
```

It is more convenient to use the following notation:

```
      +/1 3 5 7 9 11
36
```

The foregoing is read aloud as "Sum over 1 3 5 7 9 11", or as "Plus over 1 3 5 7 9 11."

The over notation can be used for other functions as well as for addition. For example:

```
                          READ AS

      ×/1 2 3             Times over 1 2 3
6                         makes 6

      ×/1 2 3 4           Times over 1 2 3 4
24                        makes 24

      +/1 2 3 4           Plus over 1 2 3 4
10                        makes 10

      (+/1 2 3 4)×6       Quantity plus over 1 2 3 4
                             times 6
60                        makes 60

      6× +/1 2 3 4        6 times plus over 1 2 3 4
60                        makes 60

      N←1 2 3 4           N assigned 1 2 3 4
      +/N                 Plus over N
10                        makes 10

      ×/N                 Times over N
24                        makes 24
```

⍠19-21

1.5. THE POSITIVE INTEGERS

The natural numbers 1 2 3 4 5 . . . are also called the positive integers. They may be produced as follows:

```
      ⍳3
1 2 3

      ⍳5
1 2 3 4 5

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

      N←6
      ⍳N
1 2 3 4 5 6
```

The symbol ⍳ is the Greek letter iota which corresponds to the English letter i. The expression ⍳N is read aloud as "the integers to N." Thus:

```
                          READ AS

      +/⍳5                Plus over the integers to 5
15                        makes 15

      ×/⍳5                Times over the integers to 5
120                       makes 120
```

⍠22-25

1.6. VECTORS

A list of numbers such as 3 5 7 11 is called a vector. The numbers in the list are called the elements of the vector. Thus the first element of the vector 3 5 7 11 is the number 3, the second element is 5, the third element is 7 and the fourth is 11. The number of elements in the vector is called the size of the vector. Thus the size of the vector 3 5 7 11 is 4.

Vectors can be added and multiplied as shown in the following examples:

```
                         READ AS

       3 5 7+1 2 3       Vector 3 5 7 plus vector 1 2 3
4 7 10                   makes 4 7 10

       1 2 3+3 2 1       Vector 1 2 3 plus vector 3 2 1
4 4 4                    makes 4 4 4

       1 2 3×3 2 1       Vector 1 2 3 times 3 2 1
3 4 3                    makes 3 4 3
```

From this it should be clear that when two vectors are added the first element is added to the first element, the second element is added to the second, and so on. Multiplication is performed similarily.

Like any other result, a vector can be assigned a name. For example:

```
                         READ AS

       V←1 2 3 4         The name V is assigned vector 1 2 3 4

       W←4 3 2 1         The name W is assigned vector 4 3 2 1

       V+W               V plus W
5 5 5 5                  makes 5 5 5 5

       V×W               V times W
4 6 6 4                  makes 4 6 6 4

       V×V               V times V
1 4 9 16                 makes 1 4 9 16
```

The following examples may be read similarly:

```
                         READ AS

       N←⍳5              N is assigned integers to 5

       N                 N
1 2 3 4 5                makes 1 2 3 4 5

       N×N               N times N
1 4 9 16 25              makes 1 4 9 16 25

       (⍳6)×⍳6           Quantity integers to 6 times
                           quantity integers to 6
1 4 9 16 25 36           makes 1 4 9 16 25 36
```


The addition of two vectors V and W means that the first element of V is to be added to the first element of W, the second element of V is to be added to the second element of W, and so on, and that an expression such as

```
       1 3 5+6 8 1 4 3
```

cannot be executed because the vectors are not of the same size.

However, expressions of the following form <u>can</u> be executed:

```
                         READ AS

       3 +1 3 5 7        3 plus vector 1 3 5 7
4 6 8 10                 makes 4 6 8 10

       1 2 3 4 5 +6      Vector 1 2 3 4 5 plus 6
7 8 9 10 11              makes 7 8 9 10 11
```

In other words, if one of the quantities to be added is a single number, it is added to each element of the vector quantity. The same holds for multiplication as follows:

```
                         READ AS

       3×1 3 5 7         3 times vector 1 3 5 7
3 9 15 21                makes 3 9 15 21

       3×⍳5              3 times integers to 5
3 6 9 12 15              makes 3 6 9 12 15

       2+3×⍳5            2 plus 3 times integers to 5
5 8 11 14 17             makes 5 8 11 14 17

       1+2×⍳6            1 plus 2 times integers to 6
3 5 7 9 11 13            makes 3 5 7 9 11 13

       +/1+2×⍳6          plus over 1 plus 2 times integers to 6
48                       makes 48

       1++/1+2×⍳6        1 plus plus over 1 plus 2 times
                           integers to 6
49                       makes 49
```

1.7. REPETITIONS

Consider the following statements and their verbalization:

```
                        READ AS

      3ρ2               3 repetitions of 2
2 2 2                   makes 2 2 2

      2ρ3               2 repetitions of 3
3 3                     makes 3 3

      5ρ7               5 repetitions of 7
7 7 7 7 7               makes 7 7 7 7 7
```

The symbol ρ is the Greek letter <u>rho</u> which corresponds to the English r.

The following two columns of statements show some interesting properties of repetitions, including the relation between multiplication and a sum of repetitions:

```
      +/3ρ2             2×3
6                 6

      +/4ρ2             2×4
8                 8

      +/5ρ7             7×5
35                35

      +/15ρ20           20×15
300               300

      ×/2ρ2             ×/2ρ3
4                 9

      ×/3ρ2             ×/3ρ3
8                 27

      ×/4ρ2             ×/4ρ3
16                81

      ×/5ρ2             ×/5ρ3
32                243
```

29-31

1.8. SUMMARY

This chapter has been concerned primarily with the language or notation of algebra, and the uses of the notation have been kept simple. Now that the language has been mastered, succeeding chapters can turn to more interesting uses of it. This does not imply that <u>all</u> the notation of algebra has now been covered, but rather that the main ideas have been introduced and that any further additions will be easy to grasp. The situation may be compared to the learning of a natural language such as French. Once the main ideas of the language have been learned (in months or years of study), the new French words needed for some particular purpose can be picked up more easily.

For example, the next chapter will treat the <u>maximum</u> function, represented by the symbol ⌈ and defined to yield the larger of its two arguments:

```
                        READ AS

      2⌈3               2 maximum 3
3                       makes 3

      2⌈4               2 maximum 4
4                       makes 4

      2⌈5               2 maximum 5
5                       makes 5

      5⌈2               5 maximum 2
5                       makes 5
```

The important point is that this new function is treated exactly like the functions plus and times, thus:

```
      2⌈1 2 3 4
2 2 3 4

      3⌈⍳5
3 3 3 4 5

      ⌈/8 1 7 10 3 10
10

      1 2 3 4 5⌈5 4 3 2 1
5 4 3 4 5
```

The main points of the notation introduced in this chapter will now be summarized in a few examples which should be useful for reference purposes:

EXAMPLE	READ AS	COMMENTS
(2+3)×4 20	Quantity 2 plus 3 times 4 makes 20	Function in parentheses is executed first
2+3×4 14	2 plus quantity 3 times 4 makes 14	Rightmost function is executed first if there are no intervening parentheses
N←3	*N* is assigned 3	Name *N* is assigned the value of the expression to the right of ←
N×4 12	*N* times 4 makes 12	
+/3 5 7 15	Plus over vector 3 5 7 makes 15	
×/2 3 5 2 60	Times over vector 2 3 5 2 makes 60	
1 2 3×3 2 1 3 4 3	Vector 1 2 3 times vector 3 2 1 makes 3 4 3	Element-by-element multiplication
3×1 2 3 3 6 9	3 times vector 1 2 3 makes 3 6 9	Single number multiplies each element
ι5 1 2 3 4 5	Integers to 5 makes 1 2 3 4 5	
5ρ4 4 4 4 4 4	5 repetitions of 4 makes 4 4 4 4 4	

Chapter 2

FUNCTION TABLES AND MAPS

2.1. INTRODUCTION

In Chapter 1, addition was spoken of as a "function" because it "does something" to the numbers it is applied to and produces some result. Multiplication was also referred to as a function, but the notion of function is actually much broader than these two examples alone might suggest. For example, the average or normal weight of a woman depends on her height and is therefore a function of her height. In fact, if one were told that the normal weight for a height of 57 inches is 113 pounds, the normal weight for a height of 58 inches is 115 pounds, and so on, then one could evaluate the function "normal weight" for any given height by simply consulting the list of corresponding heights and weights.

It is usually most convenient to present the necessary information about a function such as "normal weight" not by a long English sentence as begun above, but by a table of the form shown in Figure 2.1.

H	57	113	W
E	58	115	E
I	59	117	I
G	60	120	G
H	61	123	H
T	62	126	T
	63	130	
I	64	134	I
N	65	137	N
	66	141	
I	67	145	P
N	68	149	O
C	69	153	U
H	70	157	N
E	71	161	D
S	72	165	S

Table of Normal Weights Versus Heights

Figure 2.1

The quantity (or quantities) to which a function is applied is (are) called the <u>argument</u> (or <u>arguments</u>) of the function. For example, in the expression 3×4 the number 3 is the <u>left</u> (or <u>first</u>) argument of the function × and 4 is the <u>right</u> (or <u>second</u>) argument. Evaluation of the "normal weight" function (represented by Table 2.1) for a given argument (say 68 inches) is performed by finding the argument 68 in the first column and reading the weight (149 pounds) which occurs in the same row.

The <u>domain</u> of a function is the collection of all arguments for which it is defined. Addition is, of course, defined for any pair of numbers, but the function "normal weight" is certainly not defined for heights such as 2 inches or 200 inches. For practical purposes, the domain of a function such as "normal weight" is simply the collection of arguments in the table we happen to possess, even though information for other arguments might be available elsewhere. For example, the domain of the function of Table 2.1 is the set of integers from 57 to 70, that is, the set of integers 56 +114.

The <u>range</u> of a function is the collection of all the results of the function. For example, the range of the function of Figure 2.1 is the set of integers 113, 115, 117 120, etc., occurring in the second column.

⊡1-2

A table of normal weights often shows several columns of weights, one for small framed people, one for medium, and one for large. Such a table appears in Figure 2.2. In such a case the weight is a function of two arguments, the height and the "frame-class"; the first argument determines the row and the second argument determines the column in which the result appears. Thus the normal weight of a small-boned, 66-inch woman is 133 pounds.

		Frame			
		Small	Medium	Large	
H	57	105	113	121	W
E	58	107	115	123	E
I	59	109	117	125	I
G	60	112	120	128	G
H	61	115	123	131	H
T	62	118	126	135	T
	63	122	130	139	
I	64	126	134	143	I
N	65	129	137	147	N
	66	133	141	151	
I	67	137	145	155	P
N	68	141	149	158	O
C	69	145	153	162	U
H	70	149	157	165	N
E	71	153	161	169	D
S	72	157	165	173	S

Normal Weight as a Function
of Two Arguments

Figure 2.2

⊡3-4

An arithmetic function can also be represented by a table, as is illustrated by Figure 2.3 for the case of multiplication. Since the domain of multiplication includes all numbers, no table can represent the entire multiplication function; Figure 2.3, for example, applies only to the domain of the first few integers. The multiplication sign in the upper left corner is included simply to indicate the arithmetic function which the table represents.

×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80

Multiplication Table

Figure 2.3

In any table, the first column represents the domain of the first argument and the first row represents the domain of the second argument; the rest is called the body of the table. For example, in Figure 2.2, the body of the table is that part bordered on the left and top by the solid lines.

In any table representing a function of two arguments, any one column of the body (taken together with the column of arguments not in the body) represents a function of one argument. For example, if one takes the second column of the body of Figure 2.2, it represents the same function of one argument as does Figure 2.1.

Thus any function of two arguments can be thought of as a collection of functions of one argument. For example, the second column of the body of Figure 2.3 represents the "times two" function, the third column represents the "times three" function, etc.

Similarly, one row of the body of a function table represents a function of one argument. For example, the fifth row of the body of Figure 2.2 gives weights as a function of "frame" for 61 inch women.

5-10

2.2. READING FUNCTION TABLES

The basic rule for reading a function table is very simple - to evaluate a function, find the row in which the value of the first argument occurs (in the first column, not in the body of the table) and find the column in which the second argument occurs (in the first row) and select the element at the intersection of the selected row and the selected column. However, just as there is more to reading an English sentence than pronouncing the individual words, so a table can be "read" so as to yield useful information about a function beyond that obtained by simply evaluating it for a few cases.

For example, can the table of Figure 2.2 be "read" so as to answer the following questions:

1. Can two women of different heights have the same normal weight?

2. For a given frame type, does normal weight always increase with increasing height?

3. For a given height, does normal weight increase with frame type?

4. How many inches of height produce (about) the same change in weight as the change from small to large frame?
   Does this change remain about the same throughout the table?

Arithmetic functions are more orderly than a function such as that represented by Figure 2.2, and the patterns that can be detected in reading their function tables are more striking and interesting. Consider, for example, an attempt to read Figure 2.3 to answer the following questions:

5. The second column of the body (which was previously remarked to represent the "times two" function) contains the numbers 2 4 6, etc., which are encountered in "counting by twos".
   Can a similar statement be made about the other columns?

6. Is there any relation between corresponding rows and columns of the body, e.g., between the third row and the third column?

7. Can every result in the body be obtained in at least two different ways?
   Are there any results which can be obtained in only two ways?

Similarly, one can construct a function table for addition and read it to determine answers to the following questions:

8. In how many different ways can the result 6 be obtained by addition?
   Does the result 6 occur in the table in some pattern and if so does a similar pattern apply to other results such as 7, 8, etc.?

9. What is the relation between two successive rows of the table?

Because of the patterns they exhibit, function tables can be very helpful in gaining an understanding of unfamiliar mathematical functions. For this reason they will be used extensively in succeeding chapters.

11

2.3. EXPRESSIONS FOR PRODUCING FUNCTION TABLES

If

```
A←1 2 3 4 5 6 7 8
B←1 2 3 4 5 6 7 8 9 10
```

then the expression A.∘×B yields the body of the function table of Figure 2.3 as follows:

```
      A∘.×B
1   2   3   4   5   6   7   8   9  10
2   4   6   8  10  12  14  16  18  20
3   6   9  12  15  18  21  24  27  30
4   8  12  16  20  24  28  32  36  40
5  10  15  20  25  30  35  40  45  50
6  12  18  24  30  36  42  48  54  60
7  14  21  28  35  42  49  56  63  70
8  16  24  32  40  48  56  64  72  80
```

Similarly, the body of an addition table for the same set of arguments can be produced as follows:

```
      A∘.+B
2   3   4   5   6   7   8   9  10  11
3   4   5   6   7   8   9  10  11  12
4   5   6   7   8   9  10  11  12  13
5   6   7   8   9  10  11  12  13  14
6   7   8   9  10  11  12  13  14  15
7   8   9  10  11  12  13  14  15  16
8   9  10  11  12  13  14  15  16  17
9  10  11  12  13  14  15  16  17  18
```

The general rule is that the symbol ∘ (pronounced null) followed by a period followed by the symbol for a function produces the appropriate function table when applied to any arguments A and B. The expression "A∘.+B" may be read as "the addition table for A and B" or as "A addition table B", or even as "A null dot plus B". Similarly, "A∘.×B", may be read as "A times table B", etc.

It is important to note that the expression A∘.+B produces only the body of the addition table to which one may add a first column consisting of A and a first row consisting of B if this is found to make the table easier to read.

It is also important to note the difference between the expressions A∘.×B, which yields the multiplication table, and the expression A×B, which yields the element-by-element product of A and B. For example:

```
      A←1 3 5
      B←2 4 6

      A×B
2  12  30

      A∘.×B
  2   4   6
  6  12  18
 10  20  30
```

⊞12-13

The body of a table alone does not define a function. For example, the following tables define two distinct functions although the bodies of the tables are identical:

+	2	3	4	5
2	4	5	6	7
3	5	6	7	8
4	6	7	8	9
5	7	8	9	10

F	2	3	5	7
6	4	5	6	7
5	5	6	7	8
4	6	7	8	9
3	7	8	9	10

The name of the function represented by the first table is + (as shown in the upper left corner), and the table can be used to evaluate expressions as shown on the left below:

```
5 + 3 is 8        5 F 3 is 6
4 + 5 is 9        4 F 5 is 8
3 + 3 is 6        3 F 3 is 8
```

The function represented by the second table is called F (as indicated in the upper left corner) and the expressions on the right above shown the evaluation of the function F for the same arguments used on the left. Since the results differ, the two tables represent different functions.

The complete specification of a function table therefore requires the specification of four items:

1. The left domain (i.e., the domain of the left argument).

2. The right domain.

3. The body of the table.

4. The name of the function.

From these four items the table can be constructed and used as illustrated below:

```
Left domain:    2 + ⍳ 4
Right domain:   11 9 7 5 3 1
Body:           5 + (3×⍳4)∘.+(2×⍳6)
Name:           G

G  11   9   7   5   3   1
3  10  12  14  16  18  20
4  13  15  17  19  21  23
5  16  18  20  22  24  26
6  19  21  23  25  27  29

  4 G 5 is 19
  6 G 9 is 21
2×6 G 9 is 42
```

⎕14-16

2.4. THE FUNCTIONS DENOTED BY ⌈ AND ⌊

The advantages of the function table can perhaps be better appreciated by applying it to some unfamiliar functions than by applying it to functions such as addition and multiplication which are probably already well understood by the reader. For this purpose we will now introduce several simple new functions which will also be found to be very useful in later work.

It is sometimes instructive to introduce a new function as a puzzle - the reader must determine the general rule for evaluating the function by examining the results obtained when it is applied to certain chosen arguments. For example, the function ⌈ can be applied to certain arguments with the results shown below:

```
      3 ⌈ 8
8
      32 ⌈ 47
47
```

If one performs enough such experiments it should be possible to guess the general rule for the function. In attempting such a guess it is helpful to organize the

experiments in some systematic way, and the function table provides precisely the sort of organization needed. For example:

```
      I←1 2 3 4 5 6 7 8
      I∘.⌈I
1 2 3 4 5 6 7 8
2 2 3 4 5 6 7 8
3 3 3 4 5 6 7 8
4 4 4 4 5 6 7 8
5 5 5 5 5 6 7 8
6 6 6 6 6 6 7 8
7 7 7 7 7 7 7 8
8 8 8 8 8 8 8 8
```

From the foregoing the reader should be able to state the definition of the function and from that be able to apply it correctly to any pair of arguments.

The function ⌈ is called the <u>maximum</u> function because it yields the larger of its two arguments. The <u>minimum</u> function is denoted by ⌊ and is defined analogously. Its function table appears below:

```
      I∘.⌊I
1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 2
1 2 3 3 3 3 3 3
1 2 3 4 4 4 4 4
1 2 3 4 5 5 5 5
1 2 3 4 5 6 6 6
1 2 3 4 5 6 7 7
1 2 3 4 5 6 7 8
```

⎕17-18

2.5. THE POWER FUNCTION

Another very useful function is called the <u>power</u> function and is denoted by ⋆. Its function table is shown below:

```
      I←1 2 3 4 5 6 7
      I∘.⋆I
1  1   1    1     1      1      1
2  4   8   16    32     64    128
3  9  27   81   243    729   2187
4 16  64  256  1024   4096  16384
5 25 125  625  3125  15625  78125
6 36 216 1296  7776  46656 279936
7 49 343 2401 16807 117649 823543
```

The power function is defined in terms of multiplication in much the same way as multiplication is defined in terms of addition. To appreciate how multiplication is defined as "repeated additions", consider the following expressions:

```
      2ρ2
 2  2
      +/2ρ2               2×2
 4                   4
      3ρ2
 2  2   2
      +/3ρ2               2×3
 6                   6
      4ρ2
 2  2   2   2
      +/4ρ2               2×4
 8                   8
      +/5ρ2               2×5
10                  10
      +/6ρ2               2×6
12                  12
      +/8ρ3               3×8
24                  24
```

Comparing the results +/2ρ2 and 2×2 and the results +/3ρ2 and 2×3, etc., it should be clear that *M*×*N* is equivalent to adding *N* quantities each having the value *M*.

The corresponding definition of the power function ⋆ can be obtained by replacing each occurrence of + in the foregoing expressions by × and each occurrence of × by ⋆:

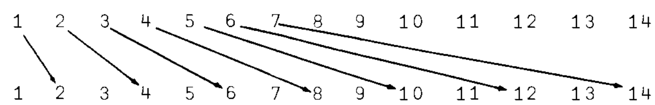
```
      2ρ2
2  2
      ×/2ρ2                  2⋆2
4                  4
      3ρ2
2  2  2
      ×/3ρ2                  2⋆3
8                  8
      4ρ2
2  2  2  2
      ×/4ρ2                  2⋆4
16                 16
      ×/5ρ2                  2⋆5
32                 32
      ×/6ρ2                  2⋆6
64                 64
      ×/8ρ3                  3⋆8
6561               6561
```


In general, *M* to the power *N* (that is, *M*⋆*N*) is obtained by multiplying together *N* factors each having the value *M*.

⎕19-22

2.6. MAPS

Figure 2.4 shows a map which represents the "times two" function. The rule for evaluating a function represented by a map is very simple: locate the specified argument in the top row, then follow the arrow from that argument to the result at the head of the arrow in the bottom row. For example, the result for the argument 3 is 6.

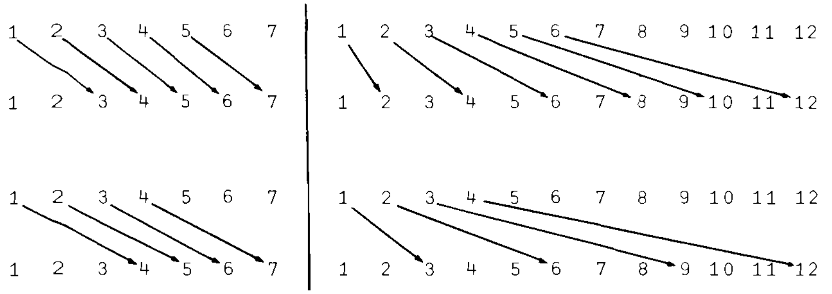


Map of "Times Two" Function

Figure 2.4

The rules for constructing a map are also simple. First consider all of the values in the domain of the function together with all of the results. Choose the smallest number and the largest number from this whole set of numbers. Write a row of numbers beginning with the smallest and continuing through each of the integers in order up to the largest. Repeat the same numbers in a row directly below the first row. For each argument in the top row now draw an arrow to the corresponding result in the bottom row.

Just as it is often helpful to read tables, so is it helpful to read such maps. Consider, for example, the four maps shown in Figure 2.5. From the first it is clear that in the map of addition of 2, the arrows are all parallel. From the map below this it is clear that the same is true for addition of 3, and that the slope of the arrows depends on the amount added. The maps on the right show multiplication. Here the slopes of the arrows are not constant, and the distance between successive arrowheads is seen to be equal to the multiplier.



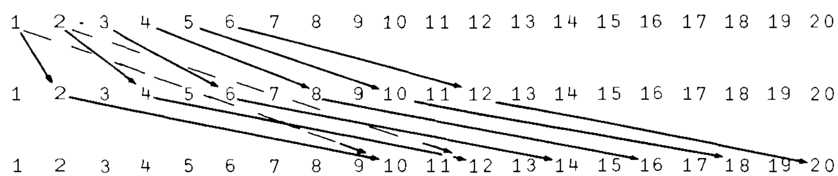
Maps for Addition and Multiplication

Figure 2.5

It is sometimes useful to show the maps of a sequence of functions such as the following:

```
      I←1 2 3 4 5 6
      2×I
 2   4   6   8   10   12
      8+(2×I)
10 12  14  16  18  20
```

The appropriate maps are shown in Figure 2.6. The broken lines show the map of the overall result produced, that is, the map of the function 8 + (2×*I*)



Maps of a Sequence of Functions

Figure 2.6

Maps will be used in the next chapter to introduce the function <u>subtraction</u> and the new <u>negative</u> numbers which this function produces.

Chapter 3

THE NEGATIVE NUMBERS

3.1. SUBTRACTION

The <u>subtraction</u> function is denoted by the <u>minus</u> sign (-). For example:

```
                        READ AS

      8-3               8 minus 3
5                       makes 5
      (5+3)-3           Quantity 5+3 minus 3
5                       makes 5
      (5-3)+3           Quantity 5-3 plus 3
5                       makes 5
```

The following examples illustrate the relation between addition and subtraction:

```
      5+3                 5+4
8                   9

      8-3                 9-4
5                   5

      6+3                 6+4
9                   10

      9-3                 10-4
6                   6

      7+3                 7+4
10                  11

      10-3                11-4
7                   7

      1 2 3 4 5+3         1 2 3 4 5+4
4 5 6 7 8           5 6 7 8 9

      4 5 6 7 8-3         5 6 7 8 9-4
1 2 3 4 5           1 2 3 4 5
```

From these examples it appears that subtraction will undo the work of addition. That is, if 3 is added to 5 to produce 8, and 3 is then subtracted from 8 the final result is the original value 5. This is true in general, and subtraction is therefore said to be the <u>inverse</u> of addition. Thus for any number X and any number A, the expression $(X+A)-A$ will yield X.

The converse is also true; that is, addition will undo the work of subtraction, and addition is therefore the inverse of subtraction. For example:

8-3
5

5+3
8

8 9 10 11 12 13-3
5 6 7 8 9 10

5 6 7 8 9 10+3
8 9 10 11 12 13

In other words, $(X-A)+A$ will also yield X.

In summary then:

$(X+A)-A$ makes X

$(X-A)+A$ makes X

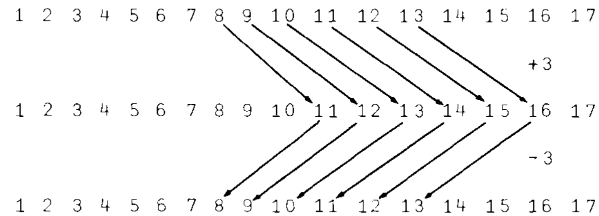
For example:

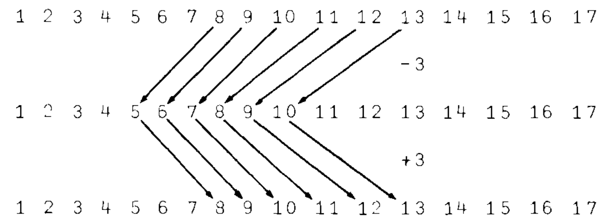
(8 9 10 11 12 13+3)-3
8 9 10 11 12 13

(8 9 10 11 12 13-3)+3
8 9 10 11 12 13

1-3

This inverse relation between addition and subtraction can also be exhibited in terms of maps as follows:

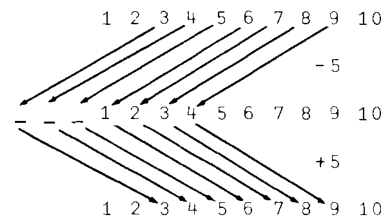




4

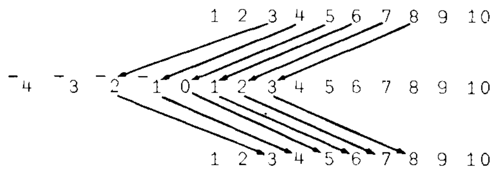
3.2. NEGATIVE INTEGERS

Consider a similar map for the case (3 4 5 6 7 8 9-5)+5 which should yield 3 4 5 6 7 8 9 as a final result:



A problem arises in some of the subtractions, since 3-5 and 4-5 and 5-5 do not yield positive integers. However, the map shows that if we keep track of the unnamed positions to the left of the first positive integer, the overall mapping for adding 5 and then subtracting 5 yields the correct final result.

The problem is resolved by assigning names to each of the new positions as follows:



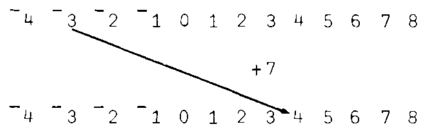
The first number to the left of 1 is named 0. This is read aloud as "zero," and means "nothing" or "none." The other new numbers, ¯1, ¯2, ¯3, and ¯4 are called *negative integers*, and are read aloud as "negative 1, negative 2, negative 3, and negative 4." Of course, the negative integers continue as far to the left as desired, just as the positive integers continue as far to the right as desired. The whole pattern including the negative integers, zero, and the positive integers, will be called the *integers*.

The effect of all this is to introduce new integers so that *every* subtraction has a proper result. Addition and subtraction are still defined as before by moving the proper number of places to the right or left in the pattern of the integers, but the pattern has now been expanded to include the negative integers and zero.

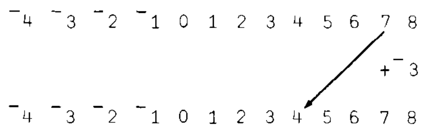
5-6

3.3. ADDITION AND SUBTRACTION

The expression 7+¯3 can be considered either as adding 7 to ¯3 as follows:

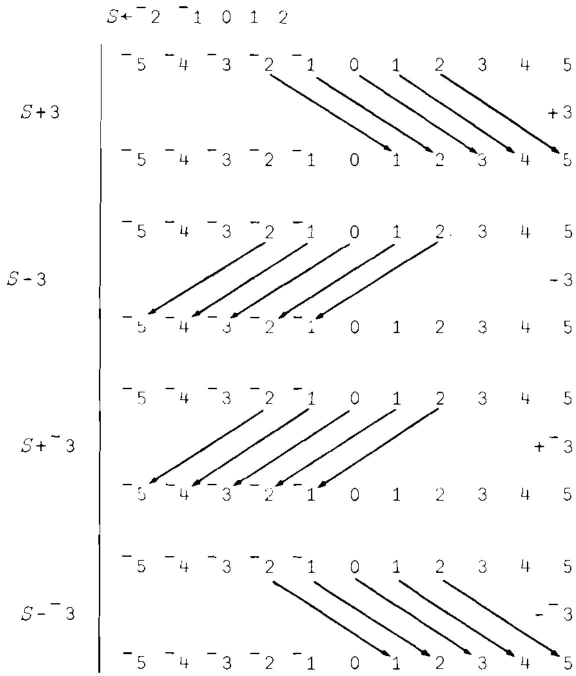


or as adding ¯3 to 7 as follows:



From the above it is clear that *adding* a *negative* number is equivalent to *subtracting* the corresponding *positive* number; that is, 7+¯3 yields the same result as 7-3.

The following examples each show an expression on the left and the corresponding map on the right for a variety of additions and subtractions involving both positive and negative integers:



The last example illustrates that subtraction of a negative number (¯3 in the example) is equivalent to *adding* the corresponding positive number (3 in the example). This follows from the fact that subtraction of ¯3 is inverse to addition of ¯3 which is equivalent to subtraction of 3. Hence subtraction of ¯3 is inverse to subtraction and is therefore equivalent to the addition of 3.

7-9

3.4. EXPRESSIONS FOR THE INTEGERS

The function ι introduced in Chapter 1 produces the positive integers as illustrated below:

```
    ι5
1 2 3 4 5

    ι7
1 2 3 4 5 6 7
```

The same function can be used to generate both positive and negative integers as follows:

```
      |(ι9)-5
      |1 2 3 4 5 6 7 8 9-5
¯4 ¯3 ¯2 ¯1 0 1 2 3 4

      ¯5+ι9
¯4 ¯3 ¯2 ¯1 0 1 2 3 4
```

The non-negative integers (that is the positive integers and zero), can be generated as follows:

```
      (ι6)-1
0 1 2 3 4 5

      ¯1+ι6
0 1 2 3 4 5
```

Non-positive integers can be generated as follows:

```
      ¯8+ι8
¯7 ¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0
```

The following examples illustrate some functions applied to a vector *S* of integers:

```
      S←¯5+ι9
      S
¯4 ¯3 ¯2 ¯1 0 1 2 3 4

      1+S
¯3 ¯2 ¯1 0 1 2 3 4 5

      ¯2+S
¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2

      S-S
0 0 0 0 0 0 0 0 0

      S+S
¯8 ¯6 ¯4 ¯2 0 2 4 6 8

      2×S
¯8 ¯6 ¯4 ¯2 0 2 4 6 8

      S+S+S
¯12 ¯9 ¯6 ¯3 0 3 6 9 12

      3×S
¯12 ¯9 ¯6 ¯3 0 3 6 9 12
```


Chapter 4

FUNCTION TABLES WITH NEGATIVE INTEGERS

4.1. INTRODUCTION

Function tables were used in Chapter 2 to explore the behavior of the functions plus and times. We can now apply them in the same manner to explore the new function subtraction introduced in Chapter 3. Moreover, they will be useful in re-examining the behavior of plus and times when applied to the new negative numbers also defined in Chapter 3.

4.2. SUBTRACTION

If *I*←ι9, then the body of a subtraction table for the arguments 1 to 9 is given by the expression *I*∘.-*I* as follows:

```
      I←ι9
      I
1  2  3  4  5  6  7  8  9
      S←I∘.-I
      S
 0 ¯1 ¯2 ¯3 ¯4 ¯5 ¯6 ¯7 ¯8
 1  0 ¯1 ¯2 ¯3 ¯4 ¯5 ¯6 ¯7
 2  1  0 ¯1 ¯2 ¯3 ¯4 ¯5 ¯6
 3  2  1  0 ¯1 ¯2 ¯3 ¯4 ¯5
 4  3  2  1  0 ¯1 ¯2 ¯3 ¯4
 5  4  3  2  1  0 ¯1 ¯2 ¯3
 6  5  4  3  2  1  0 ¯1 ¯2
 7  6  5  4  3  2  1  0 ¯1
 8  7  6  5  4  3  2  1  0
```

The subtraction table *S* has a number of interesting properties. For example, the zeros down the main diagonal of the table show that any number subtracted from itself yields 0. Moreover, each diagonal parallel to the main diagonal contains the same number repeated. For example, the diagonal two places below the main diagonal consists of all 2's.

Consider the arguments 5 and 3 in the expression 5-3. The result 2 is found in the circled position in the following subtraction table:

0	¯1	¯2	¯3	¯4	¯5	¯6	¯7	¯8
1	0	¯1	¯2	¯3	¯4	¯5	¯6	¯7
2	1	0	¯1	¯2	¯3	¯4	¯5	¯6
3	2	1	0	¯1	¯2	¯3	¯4	¯5
4	3	(2)	1	0	¯1	¯2	¯3	¯4
5	4	3	[2]	1	0	¯1	¯2	¯3
6	5	4	3	2	1	0	¯1	¯2
7	6	5	4	3	2	1	0	¯1
8	7	6	5	4	3	2	1	0

If each argument is increased by 1, the result is found in the next row and next column; in other words, one place down the diagonal as shown by the square in the above table. Since every entry in this diagonal is the same, we conclude that (5+1) - (3+1) yields the same result as 5-3. More generally, if we increase each argument by any number *N*, the result is found by moving *N* places down the diagonal. Hence we can conclude that (5+*N*) - (3+*N*) yields the same result as 5-3.

The conclusions made above for the arguments 5 and 3 will apply to arguments having any values whatever. Hence we conclude that (*X*+*N*) - (*Y*+*N*) yields the same result as *X*-*Y*.

The subtraction table *S* has another interesting property. If we choose the element in the third row and seventh column (which represents the result 3-7), we find that it is the negative of the result in the seventh row and third column (which represents 7-3). Hence the result of 3-7 is the negative of the result of 7-3. If any other pair of numbers is substituted for 7 and 3, the same relation will be observed in the table. We can therefore conclude that for any numbers *X* and *Y*, the result of *X*-*Y* is the negative of the result of *Y*-*X*.

From the above we may conclude the following: if we take the subtraction table *S* and form a new table *T* each of whose <u>columns</u> is equal to the corresponding <u>row</u> of *S*, then each element of *T* will be the negative of the corresponding element of *S*:

S

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

T

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

The sum of 4 and ¯4 is zero, and in general the sum of any number and its negative is zero. Hence we can state the foregoing result in another way; the sum of the tables *S* and *T* must be a table of all zeros:

S+*T*

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	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	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

⍠1

4.3. FLIPPING TABLES

In the previous section the table *T* was obtained from the table *S* by interchanging rows and columns. This interchange can be stated in a simple graphic way as follows: flip the table over about the axis formed by the main diagonal:

S

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

⇓

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

In examining the patterns exhibited by tables, it is also convenient to flip them in a similar way about a vertical axis and about a horizontal axis as follows:

S

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

⇓

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

S

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

⇓

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

Each of these three methods of flipping a table is a function which takes a table as argument and produces another table as a result. The symbols for each of these functions is a circle with a line through it which indicates the axis about which the table is flipped, thus: ⍉, ⌽, and ⊖. For example:

⍉S

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

⌽S

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

⊖S

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

⊖⌽S

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

The last of these four examples illustrates how the flipping functions can be applied in succession.

The function ⍉ is called transposition (because it transposes rows and columns), the function ⌽ is called row reversal (because it reverses each row vector in the table), and ⊖ is called column reversal.

A vector can be thought of much as a one-row table, and reversal can therefore be applied to it. For example:

```
      I←⍳9
      I
1 2 3 4 5 6 7 8 9
      ⌽I
9 8 7 6 5 4 3 2 1
```

The relation between the subtraction table *S* and its transpose *T* which was noted at the end of the preceding section can now be stated as follows:

```
      S+⍉S
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  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  0  0  0  0  0
0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0
```

⌹2-3

4.4. INDEXING TABLES

In discussing a table it is often necessary to refer to a particular row of the table (e.g., the fourth row), or to a particular column, or to a particular element. Such a reference will be called indexing the table, and the row and column numbers which refer to a given element are called its indices.

Indexing is denoted by brackets in the manner indicated by the following examples:

```
      M←(⍳6)∘.-⍳6
      M
 0 ¯1 ¯2 ¯3 ¯4 ¯5
 1  0 ¯1 ¯2 ¯3 ¯4
 2  1  0 ¯1 ¯2 ¯3
 3  2  1  0 ¯1 ¯2
 4  3  2  1  0 ¯1
 5  4  3  2  1  0
      M[3;4]
¯1
      M[4;3]
1
      M[3;]
2 1 0 ¯1 ¯2 ¯3
      M[;3]
¯2 ¯1 0 1 2 3
```

From the first two examples it should be clear that the row index appears first. From the third it appears that a row index alone selects the entire vector in that row.

From the fourth it appears that a column index alone selects the entire column. However, the column is displayed horizontally, not as a column. This emphasizes the fact that any single column or row selected from a matrix is simply a vector and is displayed as such.

Indexing can also be used to select an element from a vector, but in this case a single index only is required. For example:

```
      P←2 3 5 7 11
      P[4]
7
      P[2]
3
      2 3 5 7 11[2]
3
```

Moreover, a vector of indices can be used to select a vector of elements as follows:

```
      P[1 3 5]
2 5 11
      P[⍳4]
2 3 5 7
      P[5 4 3 2 1]
11 7 5 3 2
```

Finally, vectors can be used for both row and column indices to a table as follows:

```
      M[1 2;2 4 6]
¯1 ¯3 ¯5
 0 ¯2 ¯4

      M[1 3;]
 0 ¯1 ¯2 ¯3 ¯4 ¯5
 2  1  0 ¯1 ¯2 ¯3

      M[;2 4 6]
¯1 ¯3 ¯5
 0 ¯2 ¯4
 1 ¯1 ¯3
 2  0 ¯2
 3  1 ¯1
 4  2  0
```

⌹4

4.5. ADDITION

Consider the addition table *A* defined as follows:

```
I←⍳7
A←I∘.+I
A
```

2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12
7	8	9	10	11	12	13
8	9	10	11	12	13	14

It is clear that the transpose of the table *A* (that is, ⍉*A*) is equal to *A*. From this we may conclude that for any numbers *X* and *Y*, the sum *X*+*Y* is equal to the sum *Y*+*X*. The diagonals and counter-diagonals (running from upper right to lower left) of the addition table also show interesting patterns whose meanings can be examined in the manner illustrated in the discussion of the subtraction table in the preceding section.

It is also interesting to examine an addition table made for negative as well as positive arguments as follows:

```
J←(⍳15)-8
J
¯7 ¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5 6 7
B←J∘.+J
B
```

¯14	¯13	¯12	¯11	¯10	¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0
¯13	¯12	¯11	¯10	¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1
¯12	¯11	¯10	¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2
¯11	¯10	¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3
¯10	¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4
¯9	¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5
¯8	¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6
¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6	7
¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6	7	8
¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6	7	8	9
¯4	¯3	¯2	¯1	0	1	2	3	4	5	6	7	8	9	10
¯3	¯2	¯1	0	1	2	3	4	5	6	7	8	9	10	11
¯2	¯1	0	1	2	3	4	5	6	7	8	9	10	11	12
¯1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

One interesting point is that the main diagonal (consisting of all zeros) divides the positive numbers from the negative numbers. Other patterns noted in Table *A* can also be found in the extended Table *B*.

4.6. MULTIPLICATION

Again it will be convenient to consider two multiplication tables, a table *M* for positive arguments only, and a table *N* for negative arguments as well:

```
I←⍳7
M←I∘.×I
M
```

1	2	3	4	5	6	7
2	4	6	8	10	12	14
3	6	9	12	15	18	21
4	8	12	16	20	24	28
5	10	15	20	25	30	35
6	12	18	24	30	36	42
7	14	21	28	35	42	49

```
J←(⍳15)-8
J
¯7 ¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5 6 7
N←J∘.×J
N
```

49	42	35	28	21	14	7	0	¯7	¯14	¯21	¯28	¯35	¯42	¯49
42	36	30	24	28	12	6	0	¯6	¯12	¯18	¯24	¯30	¯36	¯42
35	30	25	20	15	10	5	0	¯5	¯10	¯15	¯20	¯25	¯30	¯35
28	24	20	16	12	8	4	0	¯4	¯8	¯12	¯16	¯20	¯24	¯28
21	18	15	12	9	6	3	0	¯3	¯6	¯9	¯12	¯15	¯18	¯21
14	12	10	8	6	4	2	0	¯2	¯4	¯6	¯8	¯10	¯12	¯14
7	6	5	4	3	2	1	0	¯1	¯2	¯3	¯4	¯5	¯6	¯7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
¯7	¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6	7
¯14	¯12	¯10	¯8	¯6	¯4	¯2	0	2	4	6	8	10	12	14
¯21	¯18	¯15	¯12	¯9	¯6	¯3	0	3	6	9	12	15	18	21
¯28	¯24	¯20	¯16	¯12	¯8	¯4	0	4	8	12	16	20	24	28
¯35	¯30	¯25	¯20	¯15	¯10	¯5	0	5	10	15	20	25	30	35
¯42	¯36	¯30	¯24	¯18	¯12	¯6	0	6	12	18	24	30	36	42
¯49	¯42	¯35	¯28	¯21	¯14	¯7	0	7	14	21	28	35	42	49

The zeros in *N* can be seen to divide the table into four quadrants, one in the upper right corner, one in the upper left, one in the lower left, and one in the lower right. For convenience in referring to them we will call these quadrant 1, quadrant 2, quadrant 3, and quadrant 4, assigning the numbers in a counter-clockwise order beginning with the upper right-hand corner as follows:

quadrant 2	quadrant 1
quadrant 3	quadrant 4

Each of the quadrants of *N* contains only positive numbers or only negative numbers, and the signs reverse as we proceed counter-clockwise through quadrants 1, 2, 3, and 4. It is also interesting to consider this change of sign by examining some row of the table.

First consider the fourth row of table *M*, which represents the "four times" function for positive arguments:

```
      M[4;]
4  8  12  16  20  24  28
```

Reading this row from left to right is clearly "counting by 4's"; in other words, each entry is obtained from the one before it by adding 4. Similarly, reading backward is equivalent to "counting down by 4's", and each entry is obtained from the one to the right of it by subtracting 4.

Now consider the row of table *N* which represents the same "four times" function, that is, row 12:

```
      N[12;]
¯28  ¯24  ¯20  ¯16  ¯12  ¯8  ¯4  0  4  8  12  16  20  24  28
```

Reading from right to left is again "counting down by fours" and so the entry 4 is preceded by 0 which is in turn preceded by ¯4, and so on. Hence the zero entry separates the positive and negative entrys in this row. The same conclusion applies to any row, and a similar conclusion applies to any column. Hence the quadrants must alternate in sign, as already observed.

4.7. MAXIMUM AND MINIMUM

Consider the following set of positive and negative numbers:

```
      I←(⍳13)-7
      I
¯6  ¯5  ¯4  ¯3  ¯2  ¯1  0  1  2  3  4  5  6
```

For any pair of positive numbers such as 3 and 5, the value of their maximum 3⌈5 is the value of that one of the pair which lies farthest to the right in the vector *I*. The same rule applies to both positive and negative numbers. For example:

```
      3⌈5
5
      3⌈¯5
3
      3⌈I
3  3  3  3  3  3  3  3  3  3  4  5  6
      ¯3⌈I
¯3  ¯3  ¯3  ¯3  ¯2  ¯1  0  1  2  3  4  5  6
```

Therefore, the maximum table appears as follows:

```
      MAX←I∘.⌈I
      MAX
```

¯6	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6
¯5	¯5	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6
¯4	¯4	¯4	¯3	¯2	¯1	0	1	2	3	4	5	6
¯3	¯3	¯3	¯3	¯2	¯1	0	1	2	3	4	5	6
¯2	¯2	¯2	¯2	¯2	¯1	0	1	2	3	4	5	6
¯1	¯1	¯1	¯1	¯1	¯1	0	1	2	3	4	5	6
0	0	0	0	0	0	0	1	2	3	4	5	6
1	1	1	1	1	1	1	1	2	3	4	5	6
2	2	2	2	2	2	2	2	2	3	4	5	6
3	3	3	3	3	3	3	3	3	3	4	5	6
4	4	4	4	4	4	4	4	4	4	4	5	6
5	5	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6	6

The corresponding rule for the minimum function is obvious, and the minimum table appears as follows:

```
      MIN←I∘.⌊I
      MIN
¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6 ¯6
¯6 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5 ¯5
¯6 ¯5 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4 ¯4
¯6 ¯5 ¯4 ¯3 ¯3 ¯3 ¯3 ¯3 ¯3 ¯3 ¯3 ¯3 ¯3
¯6 ¯5 ¯4 ¯3 ¯2 ¯2 ¯2 ¯2 ¯2 ¯2 ¯2 ¯2 ¯2
¯6 ¯5 ¯4 ¯3 ¯2 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  0  0  0  0  0  0
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  1  1  1  1  1
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  2  2  2  2  2
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  2  3  3  3  3
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  2  3  4  4  4
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  2  3  4  5  5
¯6 ¯5 ¯4 ¯3 ¯2 ¯1  0  1  2  3  4  5  6
```

⎕8-11

4.8. RELATIONS

In the work thus far we have observed a number of <u>relations</u> among expressions. For example, `3+8` is equal to `8+3`, and in general `X+Y` is equal to `Y+X`. Such relations have also been observed between whole tables. For example, if `M` is any multiplication table it is equal to its transpose `⍉M`.

The symbol `=` is used to denote equality, and it will be used as a function which yields a `1` if the arguments are equal, and a `0` if they are not. For example:

```
      3=8
0
      3=3
1
      ¯3=3
0
      I←⍳5
      I
1  2  3  4  5
      ⌽I
5  4  3  2  1
      I=⌽I
0  0  1  0  0
```



```
      S←I∘.-I
      S
 0 ¯1 ¯2 ¯3 ¯4
 1  0 ¯1 ¯2 ¯3
 2  1  0 ¯1 ¯2
 3  2  1  0 ¯1
 4  3  2  1  0

      ⍉S
 0  1  2  3  4
¯1  0  1  2  3
¯2 ¯1  0  1  2
¯3 ¯2 ¯1  0  1
¯4 ¯3 ¯2 ¯1  0

      S=⍉S
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1

      S+⍉S
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=S+⍉S
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

```
      M←I∘.×I
      M
 1  2  3  4  5
 2  4  6  8 10
 3  6  9 12 15
 4  8 12 16 20
 5 10 15 20 25

      ⍉M
 1  2  3  4  5
 2  4  6  8 10
 3  6  9 12 15
 4  8 12 16 20
 5 10 15 20 25

      M=⍉M
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1

      M-⍉M
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=M-⍉M
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

The symbol `≠` is used to denote the <u>not-equal</u> function. For example:

```
      3≠8
1
      3≠3
0
      S≠⍉S
 0 1 1 1 1
 1 0 1 1 1
 1 1 0 1 1
 1 1 1 0 1
 1 1 1 1 0
```

From the foregoing it should be clear that a result of 1 implies that the indicated relation holds (that is, it is true), whereas a result of 0 implies that the relation does not hold (that is, it is false).

⊞12

There are other useful relations besides equal and not-equal. Thus the symbol < denotes the function less-than:

```
      3<5
1
      5<3
0
      3<3
0
      N←(⍳9)-5
      N
¯4  ¯3  ¯2  ¯1  0  1  2  3  4
      ⌽N
4  3  2  1  0  ¯1  ¯2  ¯3  ¯4
      N<⌽N
1  1  1  1  0  0  0  0  0
      (⌽N)<N
0  0  0  0  0  1  1  1  1
```

It should be clear that one integer is "less-than" another if it precedes it in a list of integers (such as N arranged in the usual ascending order.

The symbol > denotes the function greater-than. For example:

```
      N>⌽N
0  0  0  0  0  1  1  1  1
      (⌽N)>N
1  1  1  1  0  0  0  0  0
```

To remember which of the symbols < and > denotes "less-than" and which denotes "greater-than", it may be helpful to note that the large end of the symbol points to that argument which must be larger if the relation is to be true (that is, have the result 1).

Two further relations will also be employed - the less than or equal to (denoted by ≤) and the greater than or equal to (denoted by ≥). Their definitions should be clear from their names and from the following examples:

```
      I←(⍳7)-4
      I
¯3  ¯2  ¯1  0  1  2  3
      R←⌽I
      R
3  2  1  0  ¯1  ¯2  ¯3
      I≤R
1  1  1  1  0  0  0
      I<R
1  1  1  0  0  0  0
      I=R
0  0  0  1  0  0  0
```

```
      I≥R
0  0  0  1  1  1  1
      I>R
0  0  0  0  1  1  1
      I=R
0  0  0  1  0  0  0
```

⊞13-14

4.9. LOGICAL VALUES

From all of the examples in the preceding section it can be seen that every result of a relation function is either a 1 or a 0, or a vector or table of 1's and 0's. It will be convenient to use the term logical result or logical vector or logical table to refer to such results which consist of only 0's and 1's. The term "logical" arises from the fact that a 1 can be thought of as representing "true" and a 0 as representing "false".

The functions ⌈ and ⌊ (maximum and minimum) have interesting properties when applied to logical results. The maximum table restricted to such arguments appears as follows:

```
      0 1∘.⌈0 1
0  1
1  1
```

From this it appears that the result of $L \lceil K$ (when L and K are both logical values) is 1 if either one of the arguments (or both) is 1. In other words, $L \lceil K$ is true if either L is true or K is true. Hence the maximum function applied to logical results can be said to be the function or.

The following example may clarify the matter:

```
      X←1 2 3 4 5
      Y←5 4 3 2 1
      X<Y
1  1  0  0  0
      X=Y
0  0  1  0  0
      (X<Y)⌈(X=Y)
1  1  1  0  0
      X≤Y
1  1  1  0  0
```

For these values of X and Y it can be seen that the expression (X<Y) ⌈ (X=Y) has the same result as X≤Y. The expression X<Y) ⌈ (X=Y) may be read as "X is less than Y or X equals Y" and therefore the conclusion can be phrased as follows: "The expression X is less than Y or X equals Y has the same result as X≤Y."

In a similar manner it can be shown that the minimum functions applied to logical results is equivalent to "and".

```
      0 1∘.⌊0 1
0  0
0  1
```

In other words, the result $L \lfloor K$ is true only if L is true and K is true. For example, (X≤Y) ⌊ (X≥Y) is equivalent to X=Y.

The function ⌊/V (minimum over V) applied to any vector V yields the value of the smallest element in V. Hence if V is a logical vector, the expression ⌊/V yields a 0 if there is any zero in V, and the expression ⌊/V therefore is true (i.e., 1) only if all elements of V are true. Therefore ⌊/V can be thought of as "all of V".

Similarly ⌈/V is true if at least one element of V is true. For example:

```
      W←4 6 2 3 7
      1<W
1  1  1  1  1
      ⌊/1<W
1
      ⌈/1<W
1
      3<W
1  1  0  0  1
      ⌊/3<W
0
      ⌈/3<W
1
      8<W
0  0  0  0  0
      ⌊/8<W
0
      ⌈/8<W
0
```

⎕15

4.10 THE OVER FUNCTION ON TABLES

The over function has been frequently used on vectors in earlier chapters. For example:

```
      +/2 4 3
9
      ×/2 4 3
24
      ⌈/2 4 3
4
      ⌊/2 4 3
2
```

it is also useful to apply the over function to tables, and the method of doing this will now be defined.

A few examples will be given first:

```
      T←1 2 3 4∘.-1 2 3
      T
 0 ¯1 ¯2
 1  0 ¯1
 2  1  0
 3  2  1
      +/T
¯3 0 3 6
      ×/T
0 0 0 6
      ⌈/T
0 1 2 3
      ⌊/T
¯2 ¯1 0 1
```

The rule should be clear from the foregoing examples - apply the indicated function over each of the vectors formed by the rows of the table.

Sometimes one would like to apply a function over each of the vectors formed by the *columns* of a table. This can be done by first transposing the table. For example:

```
      ⍉T
 0  1  2  3
¯1  0  1  2
¯2 ¯1  0  1
      +/⍉T
6 2 ¯2
      ×/⍉T
0 0 0
      ⌈/⍉T
3 2 1
      ⌊/⍉T
0 ¯1 ¯2
```

Another *over* function can of course be applied to any vector resulting from an *over* function applied to a table. Hence one would obtain the sum of all elements of T by the following expression:

```
      +/+/T
6
```


Similarly, the expression ×/+/T yields the product of the sums of the rows of T:

```
      ×/+/T
0
```

In particular, the expression ⌊/⌊/L applied to any logical table L will yield a result of 1 (true) only if every element of L is true. This is useful in comparing tables. For example:

```
      I←1 2 3 4 5
      S←I∘.-I
      S=⍉S
 1 0 0 0 0
 0 1 0 0 0
 0 0 1 0 0
 0 0 0 1 0
 0 0 0 0 1
      ⌊/⌊/S=⍉S
0
```

```
      A←I∘.+I
      A=⍉A
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
      ⌊/⌊/A=⍉A
1
```

16

Chapter 5

THE RATIONAL NUMBERS

5.1. INTRODUCTION

In Chapter 3, the subtraction or minus function was introduced as a function which undid the work of addition, that is, for any positive integers, *X* and *A*, the expression

```
      (X+A)-A
```

would yield the result *X*. Subtraction was therefore said to be inverse to addition.

Since addition was also inverse to subtraction, it followed that the expression

```
      (X-A)+A
```

would also yield *X*. However, if *A* is larger than *X*, then *X-A* is not a positive integer, and the negative integers and zero were introduced to ensure that every subtraction would have a result.

In this chapter the division function will be introduced in a similar way, as a function which will undo the work of multiplication, that is,

```
      (X×A)÷A
```

yields the result *X*. Since multiplication will also undo the work of division, it follows that

```
      (X÷A)×A
```

also yields *X*. That is:

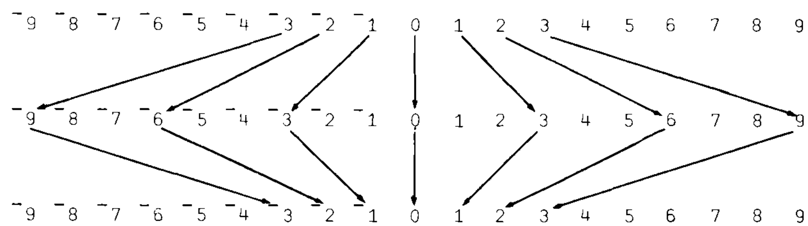
	READ AS
(X×A)÷A is X	Quantity X times A divided by A is X
and	
(X÷A)×A is X	Quantity X divided by A times A is X

For example:

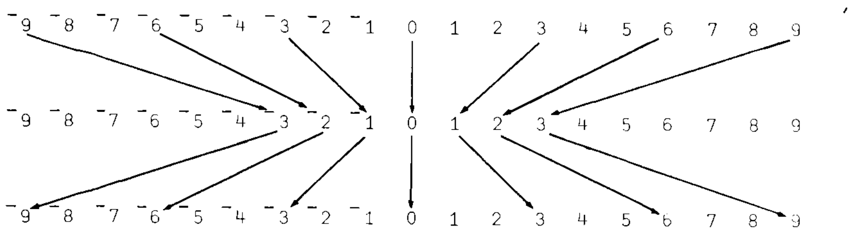
```
      3×8                          3×¯8
24                           ¯24
      (3×8)÷3                      (3×¯8)÷3
8                            ¯8
      24÷3                         ¯24÷3
8                            ¯8
      3×(24÷3)                     3×(¯24÷3)
24                           ¯24
      S←¯4+⍳7
      S
¯3 ¯2 ¯1 0 1 2 3
      S×3
¯9 ¯6 ¯3 0 3 6 9
      (S×3)÷3
¯3 ¯2 ¯1 0 1 2 3
      M←S×3
      M
¯9 ¯6 ¯3 0 3 6 9
      M÷3
¯3 ¯2 ¯1 0 1 2 3
      (M÷3)×3
¯9 ¯6 ¯3 0 3 6 9
```

1-2

Maps for the examples S×3 and (S×3)÷3 appear as follows:



The examples for $M \div 3$ and $(M \div 3) \times 3$ can be mapped similarly:



In discussing the expression $A \div B$, the first argument A is called the dividend (that which is to be divided), the second argument B is called the divisor (that which divides), and the result is called the quotient (how many times). For example, in the expression $12 \div 3$, the number 12 is the dividend, 3 is the divisor, and the result 4 is the quotient.

Just as the expression $X-A$ would sometimes yield a result which was not a positive integer, so the expression $X \div A$ will sometimes yield a result which is not an integer, and it becomes necessary to introduce a new class of numbers which are neither positive nor negative integers. These numbers are called rational numbers because they arise as a ratio of two integers. They are also called fractions, because a number such as $1 \div 3$ is considered to be one piece of a whole which is divided into 3 equal parts, that is, it is a fraction or "fractured part" of a whole. However, the question of these new numbers will be deferred until we have considered methods for performing division.

⊡3

5.2. LONG DIVISION

To divide a small number such as 8 into another small number such as 56, one can simply guess at the answer and then check the guess by multiplying it by the divisor (that is, 8) and comparing the resulting product with the original dividend 56. Thus if the guess is 7, the product 7×8 is 56 and the guess is correct; the quotient of 56 divided by 8 is 7. More generally, if DD is the name of the dividend, DR is the name of the divisor, and G is the name of the guess, then the product $DR \times G$ must agree with the dividend DD in order that the guess be the correct quotient resulting from $DD \div DR$.

For somewhat larger numbers one is less likely to guess right the first time, and the comparison of the product $DR \times G$ with the dividend DD can be used to determine whether the next guess should be larger or smaller. For example, in the division $40548 \div 124$, the value of DD is 40548, the value of DR is 124, and the first guess G might be slightly over three hundred, say 305. The product of G and DR may then be computed:

```
  124
 ×305
  620
 000
372
37820
```

Since the product 37820 is less than the dividend 40548, the next guess should be somewhat larger than 305.

One might take the next guess to be 330, in which case the product 124×330 would be 40920 and therefore too large. The third guess should be somewhere between 305 (which was too small) and 330 (which was too large). Guessing in this way will eventually lead to the desired quotient, but may take a lot of work.

⊡4

It would help to know not only that the next guess should be larger (or smaller) but by how much. It is easy to find how much the product $DR \times G$ should be increased; one merely subtracts it from the dividend. Thus in the example $40548 \div 124$ and the guess 305:

```
  124          40548
 ×305         -37820
  620           2728
 000
372
37820
```

The product should be increased by 2728. This can be done by increasing the guess by $2728 \div 124$.

We are thus faced with a new division problem (that is, $2728 \div 124$), but this time with a smaller dividend. Making a guess of 22 for the quotient would prove correct since 22×124 is euqal to 2728. The correct quotient is the sum of the first guess (305) and the correction to it (22), that is, 327. The whole process is shown below:

```
 40548÷124

 124        40548        124       2728        305
×305       -37820        ×22      -2728        +22
 620         2728        248          0        327
000                     248
372                     2728
37820
```

The work can be organized more conveniently as shown on the left below; the necessary multiplications are shown separately on the right and their results are transferred to the appropriate places on the left:

```
          327
          +22
          305          124        124
   124 |40548         ×305        ×22
       -37820          620        248
         2728         000        248
        -2728        372         2728
            0        37820
```

In the foregoing, the final result 327 is entered at the top of the column of guesses (305 and 22) of which it is the sum.

If the second guess is not correct a third can be made, and if that is not correct a fourth can be made, and so on. The final result is the sum of the guesses. For example, to compute 6704÷16:

```
          419             16          16
           +2           ×402         ×15
          +15             32          80
          402            00          16
    16 |6704            64           240
        -6432           6432
          272                         16
         -240                         ×2
           32                         32
          -32
            0
```

The quotient is 419. This result can be checked by multiplying it by 16 to see that the product is indeed equal to the dividend 6704.

If one chooses each guess to be a single digit, or a single digit followed by one or more zeros (that is, one chooses guesses which are single-digit multiples of 1, 10, 100, 1000, etc.) then the necessary multiplications become much simpler. For example, the division 40548÷124 (used in an earlier example) might begin with a guess of 300. Since 300×124 is equivalent to 3×124 followed by two zeros, this multiplication can be carried out on a single line and need not be done off to the side as was the case with the guess 305 used in the previous example:

```
          300
   124 |40548
       -37200
         3348
```

The next guess will be a multiple of 10, say 20:

```
          +20
          300
   124 |40548
       -37200
         3348
        -2480
          868
```

The next guess is a multiple of 1, say 7:

```
          327
           +7
          +20
          300
   124 |40548
       -37200
         3348
        -2480
          868
         -868
            0
```

This method of choosing multipliers not only simplifies the necessary multiplications, it also simplifies the addition of the guesses. In the previous example, the addition of 300 and 20 and 7 involves no carries, because each digit position has a single non-zero entry. This will always be the case provided that the leading digit in each guess is chosen as large as possible.

The preceding example (for the division 40548÷124) is repeated below on the left. It is also reproduced on the right but with all of the trailing zeros dropped from the calculations:

```
          327                     327
            7                       7
           20                      2
          300                     3
124 |40548               124 |40548
     -37200                  -372
       3348                    334
      -2480                   -248
        868                      868
       -868                     -868
          0                        0
```

From this it appears that the simpler scheme on the right will suffice to record the sequence of calculations. In fact, the sequence of guesses 3, 2, and 7 could be written on the same line, making the final addition unnecessary. The steps of this final scheme (called long division) are shown in the columns below:

```
       3                   32                    327
124 |40548         124 |40548          124 |40548
    -372               -372                 -372
      33                 334                  334
                        -248                 -248
                          86                   868
                                              -868
                                                 0
```

E6-7

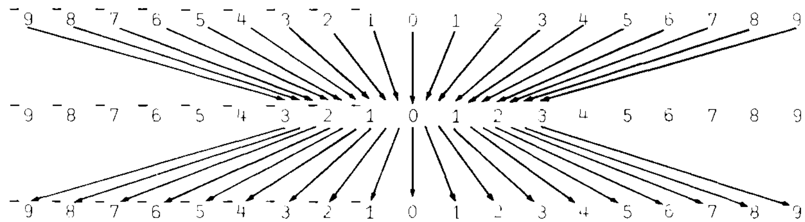
5.3. RATIONAL NUMBERS

In the preceding examples and exercises, each dividend used was an integer multiple of the divisor and the quotient was therefore an integer. However, the division 21÷4 cannot have an integer result since the quotient 5 is too small and the quotient 6 is too large. Rational numbers will now be introduced to ensure that every quotient of two integers has a result.

Consider the example

```
      P←¯10+⍳19
      P
¯9 ¯8 ¯7 ¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5 6 7 8 9
```

and the following map for P÷3 and (P÷3)×3



From this example, it appears that the number 6÷3 is less than 7÷3 which is less than 8÷3, and so on. In other words, the following sequence of four numbers is in ascending order:

6÷3 7÷3 8÷3 9÷3

Since 6÷3 is 2 and 9÷3 is 3, the above sequence may be written as:

2 7÷3 8÷3 3

In other words, the numbers 7÷3 and 8÷3 occur between the integers 2 and 3 and therefore cannot be integers. They are called rational numbers.

The negative integers and zero (introduced to make every subtraction have a result) are a set of numbers which precede the positive integers; the rational numbers (introduced to make every division have a result) are a set of numbers which occur between the integers.

Just as names were introduced for the negative numbers (for example ¯5 ¯4 ¯3), names can be introduced for rationals as follows: the result of 2÷3 is often written as 2/3, the result of 5÷2 is written as 5/2, etc. In this book we will make very little use of such names, but will instead simply write the expression which produces the rational number (for example, 2÷3 or 5÷2, or ÷/2 3 or ÷/5 2), or else write the rational number as a decimal fraction. Decimal fractions will be discussed later in this chapter.

Since the integer 2 is equal to 2÷1 or to 4÷2 or to 6÷3, etc., then the integer 2 itself can be considered to be a rational number. Similarly, 3 is equal to 3÷1 or 6÷2, etc. Therefore every integer can also be considered to be a rational number.

In discussing a rational such as *A*÷*B*, the terms dividend and divisor were introduced to refer to the parts *A* and *B*. The terms numerator (for *A*) and denominator (for *B*) are also used. To denominate means "to give a name to," and the second part of a rational gives a name to the result in the following sense: 3÷5 is called 3 fifths, 5÷7 is called 5 sevenths, etc. Similarly, the numerator gives the number of things named, as also illustrated in the examples of the preceding sentence.

⊞8-11

5.4. ADDITION OF RATIONAL NUMBERS HAVING THE SAME DIVISOR

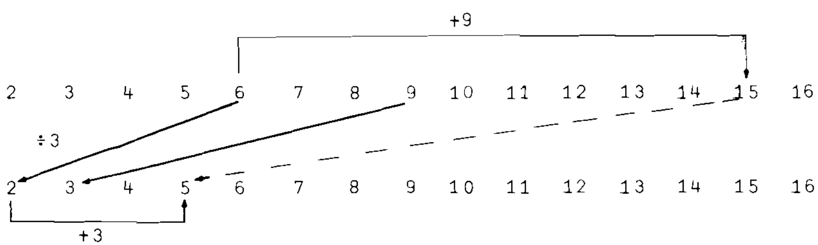
Consider the following pairs of examples:

(6÷3)+(9÷3)	(6+9)÷3
5	5
(20÷5)+(25÷5)	((20+25)÷5
9	9
(32÷4)+(8÷4)	(32+8)÷4
10	10

Since each of the results in the first column agree with the results in the second column, it appears that the expressions in each pair are equivalent, that is, (9÷3)+(6÷3) is equivalent to (9+6)÷3, and so forth. The general rule illustrated by the examples is this: If *A*, *B*, and *C* are any three integers, then

(*A*÷*C*)+(*B*÷*C*) is equal to (*A*+*B*)÷*C*

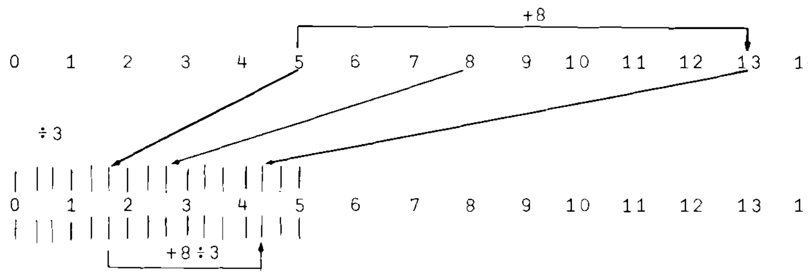
The first example may be diagrammed as follows:



Each division in the foregoing examples produces an integer, and so the rule for addition deduced above has only been shown to hold for such cases. It will, however, be assumed to hold for all rational numbers. For example:

(5÷3)+(8÷3) is equal to 13÷3

The diagram for this example follows:



It should be clear from the foregoing that similar rules apply to the subtraction of rationals having the same divisor, that is:

(*A*÷*C*)-(*B*÷*C*) is equal to (*A*-*B*)÷*C*

For example:

(13÷3)-(8÷3) is equal to 5÷3.

If the addition or subtraction of two rationals produces a dividend which is evenly divisible by the divisor, then the result may be further simplified to a single integer. For example:

|(8÷3)+(7÷3)
|15÷3
|5

|(8÷3)-(5÷3)
|3÷3
|1

The vertical lines above indicate, as usual, that the expressions to the right are equivalent. From here on the vertical lines will be omitted; that is, any list of expressions is to be read as a statement that the expressions are equivalent.

⊞12-14

5.5. MULTIPLICATION OF RATIONAL NUMBERS

The rules for multiplying two rational numbers will be explored by first considering a number of cases in which the division can actually be performed. Compare the corresponding examples in the following two columns:

$(10\div5)\times(12\div3)$	$(10\times12)\div(5\times3)$
2×4	$120\div15$
8	8
$(18\div3)\times(12\div6)$	$(18\times12)\div(3\times6)$
6×2	$216\div18$
12	12
$(32\div8)\times(35\div7)$	$(32\times35)\div(8\times7)$
4×5	$1120\div56$
20	20

Since the results in the two columns agree, it appears that $(10\div5)\times(12\div3)$ is equivalent to $(10\times12)\div(5\times3)$ and so on. In general, if A, B, C, and D are any integers, it appears that $(A\div B)\times(C\div D)$ is equivalent to $(A\times C)\div(B\times D)$. The above examples illustrate this only for cases where $A\div B$ and $C\div D$ each produce integer results. However, the rule will be assumed to apply for all rational numbers. For example:

$(3\div4)\times(5\div2)$ is equal to $15\div8$

$(4\div3)\times(2\div5)$ is equal to $8\div15$

$(3\div4)\times(4\div3)$ is equal to $12\div12$ (that is, 1).

The rule for multiplying rationals can therefore be stated as follows:

$(A\div B)\times(C\div D)$

$(A\times C)\div(B\times D)$

In words, the dividend of the result is the product of the dividends and the divisor of the result is the product of the divisors.

Applying this rule to the case where A, B, C, and D are equal to 4, 5, 3, and 3, respectively, yields

$(4\div5)\times(3\div3)$
$(4\times3)\div(5\times3)$
$12\div15$

However, since $3\div3$ is 1, then

$(4\div5)\times(3\div3)$
$(4\div5)\times1$
$4\div5$

Therefore, all members of the two sets of expressions above are equivalent, and $12\div15$ is equal to $4\div5$.

It therefore appears that for any three integers A, B, and C:

$A\div B$
$(A\div B)\times(C\div C)$
$(A\times C)\div(B\times C)$

In words, if the dividend and divisor of a rational number are multiplied by the same quantity C, the resulting rational number is equal to the original rational number. 15-18

5.6. MULTIPLICATION OF A RATIONAL BY AN INTEGER

Consider again the general rule for the multiplication of two ratios, that is:

$(A\div B)\times(C\div D)$
$(A\times C)\div(B\times D)$

If B has the value 1, we obtain the following simpler rule:

$A\times(C\div D)$
$(A\div1)\times(C\div D)$
$(A\times C)\div(1\times D)$
$(A\times C)\div D$

In other words, if a ratio $C\div D$ is to be multiplied by an integer A, the result is obtained by simply multiplying the numerator C by A. For example:

$5\times(3\div7)$
$15\div7$ 19

5.7. MULTIPLICATION EXPRESSED IN TERMS OF VECTORS

Since 3÷4 can be written as ÷/3 4, and 5÷2 can be written as ÷/5 2, etc., then any rational can be written as ÷/*V*, where *V* is a two-element vector. The first examples used in the multiplication of rational numbers will now be repeated but written in this new form:

```
      (÷/10 5)×(÷/12 3)          ÷/10 5×12 3
      2×4                        ÷/120 15
8                          8

      (÷/18 3)×(÷/12 6)          ÷/18 3×12 6
      6×2                        ÷/216 18
12                         12

      (÷/32 8)×(÷/35 7)          ÷/32 8×35 7
      4×5                        ÷/1120 56
20                         20
```

From the foregoing it appears that the rule for multiplying rationals can be written very neatly in terms of vectors: if *V* and *W* are each two-element vectors, then the product of the rationals (÷/*V*)×(÷/*W*) is equivalent to the rational ÷/*V*×*W*. For example:

```
      V←10 5
      W←12 3
      (÷/V)×(÷/W)
      2×4
8

      V×W
120 15
      ÷/V×W
8
```


5.8. ADDITION OF RATIONALS

The method for adding rationals given in Section 5.4 applied only to the addition of two rationals sharing the same divisor, that is,

(*A*÷*C*)+(*B*÷*C*) is equal to (*A*+*B*)÷*C*

It cannot be applied to add a pair of rationals such as 2÷3 and 4÷5. However, the results of the preceding section can be applied as follows:

2÷3 is equal to (2×5)÷(3×5)

4÷5 is equal to (4×3)÷(5×3)

Therefore 2÷3 and 4÷5 are equal to 10÷15 and 12÷15, respectively. But the last two rationals have the same divisor and can therefore be added as follows:

(10÷15)+(12÷15) is equal to 22÷15.

Therefore

(2÷3)+(4÷5) is equal to 22÷15.

Similarly:

```
      (2÷7)+(4÷5)
      ((2÷7)×(5÷5))+((4÷5)×(7÷7))
      (10÷35)+(28÷35)
      38÷35

      (1÷2)+(1÷3)+(1÷6)
      ((1÷2)×(3÷3))+((1÷3)×(2÷2))+(1÷6)
      (3÷6)+(2÷6)+(1÷6)
      6÷6
      1
```

In general, two rationals, (*A*÷*B*) and (*C*÷*D*) may be added as follows:

```
      (A÷B)+(C÷D)
      ((A÷B)×(D÷D))+((C÷D)×(B÷B))
      ((A×D)÷(B×D))+((C×B)÷(D×B))
      ((A×D)+(C×B))÷(B×D)
```

5.9. ADDITION OF RATIONALS IN TERMS OF VECTORS

Recall the rule for the addition of two rationals as follows:

```
(A÷B)+(C÷D)
((A×D)+(B×C))÷(B×D)
```

Recall also that if V is a two element vector, then ÷/V is the ratio V[1]÷V[2]. Consequently, the rule for the addition of two rationals ÷/V and ÷/W can be expressed as follows:

```
(÷/V)+(÷/W)
(+/V×⌽W)÷(V[2]×W[2])
```

For example:

```
V←3 5
W←7 2
(÷/3 5)+(÷/7 2)
(+/3 5×2 7)÷(5×2)
(+/6 35)÷10
41÷10
```

⎕22

5.10. THE QUOTIENT OF TWO RATIONALS

Consider the following examples of division:

```
      12÷4
3

      (12×5)÷(4×5)
3

      18÷2
9

      (18×7)÷2×7
9
```


They illustrate the fact, developed earlier, that the multiplication of both numerator and denominator by the same quantity leaves a fraction unchanged. That is:

```
P÷Q
(P×R)÷(Q×R)
```

Consider now the division of the rational number A÷B by the rational number C÷D, that is,

```
(A÷B)÷(C÷D)
```

The result will remain unchanged if the numerator A÷B and the denominator C÷D are each multiplied by the same number D÷C. That is:

```
(A÷B)÷(C÷D)
((A÷B)×(D÷C))÷((C÷D)×(D÷C))
```

The last half of the above expression (that is, (C÷D)×(D÷C)) can be simplified by applying the rule that the product of two rationals is the product of their numerators divided by the product of their denominators:

```
(C÷D)×(D÷C)
(C×D)÷(D×C)
```

Since C×D and D×C are equal, their quotient is 1. Therefore (C÷D)×(D÷C) makes 1.

Finally, then:

```
(A÷B)÷(C÷D)
((A÷B)×(D÷C))÷((C÷D)×(D÷C))
((A÷B)×(D÷C))÷1
(A÷B)×(D÷C)
```

Therefore the quotient (A÷B)÷(C÷D) is equivalent to the product (A÷B)×(D÷C). For example:

```
      (36÷3)÷(24÷4)
2

      (36÷3)×(4÷24)
2
```

This relation can also be expressed in terms of vectors as follows. If V is a two element vector and W is a two-element vector, then:

```
      (÷/V)÷(÷/W)
      (÷/V)×÷/⌽W
```

For example:

```
      (÷/36 3)÷(÷/24 4)
2

      (÷/36 3)×(÷/4 24)
2
```

⎕23

5.11. DECIMAL FRACTIONS

Any rational number having a denominator such as 10 or 100 or 1000, etc., can be represented as a decimal fraction in the manner illustrated below:

```
      1386÷10
      138.6

      1386÷100
      13.86

      1386÷1000
      1.386

      1386÷10000
      .1386

      1386÷100000
      .01386
```

The period occurring in a decimal fraction is called a decimal point. If the decimal point in a decimal fraction is followed by one digit, then the rational it represents is the integer represented by the same digits without a decimal point, divided by 10. If the decimal point is followed by two digits, the rational represented is the same integer divided by 100, and, in general, if the decimal point is followed by K digits, then the rational represented is the same integer divided by the integer formed by a 1 followed by K zeros.

⎕24-26

5.12. ADDITION AND SUBTRACTION OF DECIMAL FRACTIONS

The following examples show the addition of some pairs of decimal fractions in which the fractions in each pair have the decimal point in the same place, that is, they have the same number of digits following the decimal place:

```
      21.34+16.55
      (2134÷100)+(1655÷100)
      (2134+1655)÷100
      3789÷100
37.89

      13.659+82.546
      (13659+82546)÷1000
      96205÷1000
96.205

      12.700+39.615
      (12700+39615)÷1000
52.315
```

In other words, a pair of decimal fractions having the decimal point in the same place can be added just as if they were integers (i.e., by ignoring the decimal point), and then placing the decimal point in the same place in the result. This rule may be applied to the foregoing examples as follows:

```
      21.34          13.659          12.700
      16.55          82.546          39.615
      -----          ------          ------
      37.89          96.205          52.315
```

By the same reasoning, subtraction of such a pair of decimal fractions can be carried out in a similar manner. For example, the subtraction 21.34-16.55 can be carried out as follows:

```
      21.34
      16.55
      -----
       4.79
```

⎕27

It remains to add two decimal fractions which do not have the same number of digits following the decimal point. The value of a decimal fraction is not changed by appending zeros to the right of it; thus 12.7 and 12.70 and 12.700, etc., are all equal. This follows from the fact (established earlier) that the value of a rational is unchanged if the numerator and denominator are each multiplied by the same number. For example:

```
12.7
127÷10
(127×10)÷(10×10)
1270÷100
12.70
1270÷100
(1270×10)÷(100×10)
12700÷1000
12.700
```

Therefore, zeros may be appended to the right of any decimal fraction without changing its value. To perform the addition 12.7+39.615, one appends two zeros to the right of 12.7 (getting 12.700) and then adds them by the method for adding decimal fractions having the decimal point in the same place:

```
12.700
39.615
------
52.315
```

28

5.13. THE DECIMAL FRACTION REPRESENTATION OF A RATIONAL

Many rational numbers having denominators which are not of the form 10, 100, 1000, etc., can still be expressed as decimal fractions by simply multiplying both numerator and denominator by some integer which produces a denominator which is of the form 10, 100, 1000, etc. For example:

```
    1÷2
    (1×5)÷(2×5)
    5÷10
.5

    3÷5
    6÷10
.6

    7÷2
    35÷10
3.5

    1÷25
    4÷100
.04

    38÷4
    950÷100
9.5

    1÷125
    8÷1000
.008

    1÷16
    625÷10000
.0625

    1÷625
    16÷10000
.0016
```

From these examples, it should be clear that the ordinary long division process may be used to convert such

rationals to decimal fractions; all that is needed is to append to the integer numerator a decimal point followed by a sufficient number of zeros. For example, since 38 is equivalent to 38.0 then 38÷4 may be written as 38.0÷4 and the long division may be carried out as follows:

```
     9.5
 4 | 38.0
    -36
     20
    -20
      0
```

Similarly, ÷/1 16 may be converted to decimal fraction as follows:

```
      .0625
 16 | 1.0000
      -96
        40
       -32
         80
        -80
          0
```

29

5.14. DECIMAL FRACTION APPROXIMATIONS TO RATIONALS

The rational number 75÷64 can be converted to a decimal fraction by long division as follows:

```
      1.171875
 64 | 75.000000
     -64
      110
      -64
       460
      -448
        120
        -64
         560
        -512
          480
         -448
           320
          -320
             0
```

Therefore, 75÷64 is equivalent to 1.171875.

Suppose that one stopped the long division process just before the last digit, obtaining the quotient 1.17187 and leaving a non-zero remainder, that is, 320. The decimal fraction 1.17187 is not equal to 75÷64, but it is very nearly equal to it and is therefore said to be a good approximation to 75÷64. To see how close 1.17187 is to 75÷64 one may subtract the approximation 1.17187 from the true value of 1.171875 as follows:

```
 1.171875
-1.171870
 --------
 0.000005
```

The difference is therefore .000005 or 5÷1000000. This is only 5 millionths, a very small quantity.

The decimal fraction 1.17187 is said to be a 5-place approximation to 75÷64 because it is close to 75÷64 and has 5 digits following the decimal place. It is also a best 5-place approximation to 75÷64, since no other decimal fraction with only 5 places can be closer (although 1.17188 is just as close and is also a best approximation).

The decimal fraction 1.171 (obtained by stopping the long division after 3 places) is a three-place approximation to 75÷64, and is smaller than 75÷64 by the amount .000875. It is not, however, the best approximation, since the fraction 1.172 is larger than 75÷64 by only .000125 as may be seen from the following subtraction:

```
 1.172000
-1.171875
 --------
 0.000125
```

Therefore, to get a best approximation to a rational, one should continue the long division one place beyond the desired number of places. If the additional digit is less than 5, the additional digit should be discarded; if not, the additional digit should be discarded but a 1 should be added into the last place kept. For example:

```
      1.1718
64 | 75.0000
    -64
     110
     -64
      460
     -448
       120
       -64
        560
       -512
         48
```

The best 3-place approximation is 1.171+.001, or 1.172.

Similarly, the best two-place approximation to 115÷64 can be obtained as follows:

```
       1.796
64 | 115.000
     -64
      510
     -448
       620
      -576
        440
       -384
         56
```

The best two-place approximation to 115÷64 is therefore 1.79+.01, which is 1.80, or simply 1.8.

For many rationals, the long division process never terminates with a zero remainder. For example, for the rational 1÷3, the remainder is always 1:

```
    .333
3 | 1.000
    -9
     10
     -9
      10
      -9
       1
```

For such a case, the long division process can also be used to give a best approximation to the rational, thus .333 is the best three-place approximation for the rational 1÷3 and differs from it by only 1÷3000. For,

```
.333+(1÷3000)
(333÷1000)+(1÷3000)
(999÷3000)+(1÷3000)
1000÷3000
1÷3
```

Similarly, .667 may be obtained as the best three-place approximation to 2÷3 as follows:

```
    .6666
3 | 2.0000
   -1 8
      20
     -18
       20
      -18
        20
       -18
         2
```

Since the fourth digit of the result exceeds 5, the best three-place approximation is .666+.001, or .667.

The following table shows the four-place decimal fraction approximations to the rationals resulting from the expression (ι7)∘.÷ι7:

1	0.5	0.3333	0.25	0.2	0.1667	0.1429
2	1	0.6667	0.5	0.4	0.3333	0.2857
3	1.5	1	0.75	0.6	0.5	0.4286
4	2	1.333	1	0.8	0.6667	0.5714
5	2.5	1.667	1.25	1	0.8333	0.7143
6	3	2	1.5	1.2	1	0.8571
7	3.5	2.333	1.75	1.4	1.167	1

⊞30

5.15. MULTIPLICATION OF DECIMAL FRACTIONS

The following example shows the multiplication of two decimal fractions:

```
      1.3×2.14
      (13÷10)×(214÷100)
      (13×214)÷(1000)
      2782÷1000
2.782
```

From this it is clear that the following rule can be used: multiply the numbers as integers (ignoring the decimal point) and place a decimal point in the result so that the number of digits following it is equal to the sum of the number of digits following the decimal points in the two factors. For example:

```
  2.14      (2 decimal places)
 __1.3      (1 decimal place)

   642
 _214_

 2.782      (2+1 decimal places)
```

⊞31-32

5.16. DIVISION OF DECIMAL FRACTIONS

The following procedure can be used to find the quotient where the dividend and divisor are decimal fractions:

1. Perform the division as if the numbers were integers, ignoring the decimal points.

2. In the resulting quotient, move the decimal point as many places to the left as there are decimal places in the original dividend.

3. From there move the decimal point as many places to the right as there are decimal places in the original divisor.

For example, to evaluate the expression 11.025÷1.26, we first divide the integer 11025 by the integer 126:

```
         87.5
126 | 11025
     -1008
        945
       -882
        630
       -630
          0
```

The decimal point in the quotient 87.5 is now moved three places to the left (because the dividend 11.025 has three decimal places) to obtain .0875, and the decimal place is then moved two places to the right (because the divisor 1.26 has two decimal places) to obtain 8.75. This result can be checked by evaluating 8.75×1.26 to see that it yields 11.025 as required.

The justification for this procedure should be clear from the following equivalences:

```
11.025÷1.26
(11025÷1000)÷(126÷100)
(11025÷1000)×(100÷126)
((11025÷126)×(100÷1000)
```

⊞33

5.17. EXPONENTIAL NOTATION

Numbers such as 120000000 and .0000000017 are awkward to read and write because of the large number of zeros to be counted. Exponential notation allows one to write these numbers instead as 12*E*7 and 17*E*¯10.

More generally, one may write any decimal number (or integer) followed immediately by an *E* followed immediately by an integer. The value this denotes may be determined as follows: take the number before the *E* and move its decimal point by an amount determined by the integer following the *E*, moving it to the right if the integer is positive and to the left if the integer is negative. For example:

```
1.34E5          1.34E¯5
134000          .0000134
134E3           134E¯7
.134E6          .134E¯4
```

⌸34-35

5.18. DIVISION WITH NEGATIVE ARGUMENTS

A study of the map used in introducing rational numbers should make it clear that (¯1)÷3 is the negative of 1÷3, that (¯2)÷3 is the negative of 2÷3, etc. The result to be obtained when the divisor is negative is not so clear.

Consider the rational 3÷¯4 which has a negative divisor. We have seen that it is equivalent to the rational (3×*A*)÷(¯4×*A*), where *A* is any integer. If we choose *A* to be ¯1, then (3×*A*)÷(¯4×*A*) is equal (¯3)÷4. Similarly, (¯3)÷(¯4) is equal to 3÷4. From this it appears that the sign of the quotient *B*÷*C* is determined from the signs of the arguments *B* and *C* in exactly the same way that the sign of the product *B*×*C* is determined. ⌸36

5.19. DIVISION BY ZERO

The result of the division *A*÷*B* is a quotient *C* such that *C*×*B* is equal to *A*. If *A* is 4 and *B* is zero, then *C* must be a number such that *C*×0 is 4. Since 0 times anything is 0, there is no such number *C*. Hence division by zero is not possible.

Chapter 6

FUNCTION TABLES WITH RATIONAL NUMBERS

6.1. INTRODUCTION

In Chapter 4 we used function tables to examine the function of subtraction newly introduced in Chapter 3, and to re-examine familiar functions applied to the negative numbers also introduced in Chapter 3. In this chapter we will pursue a similar course with respect to the division function and the rational numbers introduced in Chapter 5.

In this chapter, the results of divisions are represented as decimal fractions correct to three places.

6.2. CATENATION

Catenation is a simple new primitive which will be needed in this and later chapters; it is denoted by the comma. "Catena" is a Latin word meaning "chain", and catenation is a function which chains its arguments together. For example:

```
        X←1 2 3
        Y←4 5
        X,Y
1 2 3 4 5
        Y,X
4 5 1 2 3
        +/X,Y
15
        X,7
1 2 3 7
        7,X
7 1 2 3
        7,8
7 8
```

⌸1

6.3. DIVISION TABLES

If I←⍳8, then the body of the division table for the arguments 1 to 8 is given by the expression I∘.÷I as follows:

```
      I←⍳8
      D←I∘.÷I
      D
 1.000  0.500  0.333  0.250  0.200  0.167  0.143  0.125
 2.000  1.000  0.667  0.500  0.400  0.333  0.286  0.250
 3.000  1.500  1.000  0.750  0.600  0.500  0.429  0.375
 4.000  2.000  1.333  1.000  0.800  0.667  0.571  0.500
 5.000  2.500  1.667  1.250  1.000  0.833  0.714  0.625
 6.000  3.000  2.000  1.500  1.200  1.000  0.857  0.750
 7.000  3.500  2.333  1.750  1.400  1.167  1.000  0.875
 8.000  4.000  2.667  2.000  1.600  1.333  1.143  1.000
```

This table has a number of interesting properties. For example, each row can be seen to be in descending order and each column can be seen to be in ascending order. Moreover, the main diagonal consists of all 1's, illustrating the fact that N÷N is equal to 1 whatever the value of N. Moreover, many other duplications occur in the table, showing that the same value may result from the division of different pairs of numbers. Thus the decimal fraction 0.333 occurs in two places, resulting from 1÷3 and 2÷6.

⌹2

The division table can be extended to negative arguments as well. However, as pointed out in Chapter 5, the number 0 is not permitted as the right argument of division:

```
      J←(⍳9)-5
      J
¯4  ¯3  ¯2  ¯1  0  1  2  3  4
      K←(0-⌽⍳4),⍳4
      K
¯4  ¯3  ¯2  ¯1  1  2  3  4
      J∘.÷K
 1.000  1.333  2.000  4.000 ¯4.000 ¯2.000 ¯1.333 ¯1.000
 0.750  1.000  1.500  3.000 ¯3.000 ¯1.500 ¯1.000 ¯0.750
 0.500  0.667  1.000  2.000 ¯2.000 ¯1.000 ¯0.667 ¯0.500
 0.250  0.333  0.500  1.000 ¯1.000 ¯0.500 ¯0.333 ¯0.250
 0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
¯0.250 ¯0.333 ¯0.500 ¯1.000  1.000  0.500  0.333  0.250
¯0.500 ¯0.667 ¯1.000 ¯2.000  2.000  1.000  0.667  0.500
¯0.750 ¯1.000 ¯1.500 ¯3.000  3.000  1.500  1.000  0.750
¯1.000 ¯1.333 ¯2.000 ¯4.000  4.000  2.000  1.333  1.000
```


6.4. COMPARISON

Two rationals such as 3÷7 and 4÷9 can be compared to see which is the larger by first converting them each to a decimal representation. For example:

```
      3÷7
0.429
      4÷9
0.444
      (3÷7)≤(4÷9)
1
```

It is also possible to compare two rationals without actually carrying out any division.

If two rationals have the same denominator, they can be compared by simply comparing their numerators. For example, 27÷63 is less then 28÷63. Moreover, for any pair of fractions one can find an equivalent pair which do have the same denominator. For example, 3÷7 is equivalent to (3×9)÷(7×9) (that is, 27÷63) and 4÷9 is equivalent to (7×4)÷(7×9) (that is, 28÷63).

In general, if N1, D1, N2, and D2 are any integers, then N1÷D1 and N2÷D2 can be compared by forming the equivalent pair (N1×D2)÷(D1×D2) and (D1×N2)÷(D1×D2), which have the same denominator. Hence it is only necessary to compare the numerators N1×D2 and D1×N2. For example:

```
      N1←3
      D1←7
      N2←4
      D2←9
      N1÷D1
0.429
      N2÷D2
0.444
      (N1÷D1)≤(N2÷D2)
1
      (N1×D2)≤(D1×N2)
1
```

The same relations will of course hold if *N*1, *D*1, *N*2, and *D*2 are vectors. For example:

```
      N1←1 1 1 2 2 2 3 3 3
      D1←1 2 3 1 2 3 1 2 3
      N2←4 4 4 5 5 5 6 6 6
      D2←4 5 6 4 5 6 4 5 6

      N1÷D1
1  0.5  0.333  2  1  0.667  3  1.5  1

     N2÷D2
1  0.8  0.667  1.25  1  0.833  1.5  1.2  1

     (N1÷D1)≤(N2÷D2)
1  1  1  0  1  1  0  0  1

      (N1×D2)≤(D1×N2)
1  1  1  0  1  1  0  0  1
```

Moreover, if one wants to compare each element of *N*1÷*D*1 with each element of *N*2÷*D*2, then the corresponding comparison tables agree as well:

```
      (N1÷D1)∘.≤(N2÷D2)         (N1∘.×D2)≤(D1∘.×N2)
 1 0 0 1 1 0 1 1 1          1 0 0 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 1          1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1          1 1 1 1 1 1 1 1 1
 0 0 0 0 0 0 0 0 0          0 0 0 0 0 0 0 0 0
 1 0 0 1 1 0 1 1 1          1 0 0 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 1          1 1 1 1 1 1 1 1 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 1 0 0          0 0 0 0 0 0 1 0 0
 1 0 0 1 1 0 1 1 1          1 0 0 1 1 0 1 1 1

      ⌊/⌊/((N1÷D1)∘.≤(N2÷D2)) = ((N1∘.×D2)≤(D1∘.×N2))
1
```

⌹4

6.5. THE POWER FUNCTION FOR NEGATIVE ARGUMENTS

In Chapter 4 the functions +, ×, ⌈, and ⌊ were re-examined to determine how they applied to the negative arguments introduced in Chapter 3. This was not done for the power function because the result of an expression such as 2*¯3 is a rational number, and rational numbers had not yet been introduced.

We will begin by recalling the definition of the power function as the product over a number of repetitions of a certain factor, that is, *A***B* is equivalent to ×/*B*⍴*A*. For example:

```
      3⍴2
2  2  2
      ×/3⍴2
8
      2*3
8
```

The power table for positive integers therefore appears as follows:

```
      I←2 3 4 5 6
      J←2 3 4 5 6 7
      I∘.*J
 4      8     16     32      64     128
 9     27     81    243     729    2187
16     64    256   1024    4096   16384
25    125    625   3125   15625   78125
36    216   1296   7776   46656  279936
```

A simple pattern emerges in each row of the table - any element of a row can be obtained from the element which precedes it by multiplying by a certain factor, that factor being the value of the left argument which produced that row. For example, the third row was produced by the expression:

```
      4*2 3 4 5 6 7
16  64  256  1024  4096  16384
```

and the third element in the row can be obtained from the one before it by multiplying by 4.

This same pattern can be stated in a different way - each element can be obtained from the one <u>following</u> it by <u>dividing</u> by the same factor. In this way the pattern can be extended to the left to obtain results for right arguments less than 2:

```
      I←2 3 4 5
      J←(⍳7)-4
      J
¯3 ¯2 ¯1 0 1 2 3
      I∘.*J
 0.125   0.250   0.500   1.000   2.000   4.000    8.000
 0.037   0.111   0.333   1.000   3.000   9.000   27.000
 0.016   0.062   0.250   1.000   4.000  16.000   64.000
 0.008   0.040   0.200   1.000   5.000  25.000  125.000
```

Two important results emerge from these patterns: (1) Any number *A* raised to the power 1 is equal to *A*, and (2) Any number raised to the power 0 is equal to 1. For example:

```
      1 2 3 4 5 6*1
1 2 3 4 5 6
      1 2 3 4 5 6*0
1 1 1 1 1 1
```

⌹5-6

The case of a zero left argument has not been considered. From the foregoing we may conclude that 0*0 should be 1 and that 0*1 should be 0. Further entries in the expression 0*0 1 2 3 4 will be obtained by multiplying by the factor 0 and are all zero:

```
      0*0 1 2 3 4 5
1 0 0 0 0 0
```

Recalling that *A**¯1 was obtained from *A**0 by dividing by *A*, we may now attempt to define a result for 0*¯1 by dividing the value for 0*0 (that is, 1) by the appropriate factor. But this factor is 0, and division by 0 is not allowed. Hence the function 0**R* is <u>not</u> defined for negative values of the right argument *R*.

The application of the power function to a negative <u>left</u> argument is straightforward. Recall that 3*4 is equivalent to ×/4ρ3, and that in general *A**4 is equivalent to ×/4ρ*A*. Hence if *A* is ¯3 we have:

```
      4ρ¯3
¯3 ¯3 ¯3 ¯3
      ×/4ρ¯3
81
      ¯3*4
81
      5ρ¯3
¯3 ¯3 ¯3 ¯3 ¯3
      ×/5ρ¯3
¯243
      ¯3*5
¯243
```

The foregoing results can now be used to construct a table of the power function for both positive and negative arguments, including 0 in the right argument only:

```
      I←(0-⌽⍳4),⍳4
      J←(⍳7)-4
      I
¯4 ¯3 ¯2 ¯1 1 2 3 4
      J
¯3 ¯2 ¯1 0 1 2 3
      I∘.*J
 ¯0.016  0.062  ¯0.250  1.000  ¯4.000  16.000  ¯64.000
 ¯0.037  0.111  ¯0.333  1.000  ¯3.000   9.000  ¯27.000
 ¯0.125  0.250  ¯0.500  1.000  ¯2.000   4.000   ¯8.000
 ¯1.000  1.000  ¯1.000  1.000  ¯1.000   1.000   ¯1.000
  1.000  1.000   1.000  1.000   1.000   1.000    1.000
  0.125  0.250   0.500  1.000   2.000   4.000    8.000
  0.037  0.111   0.333  1.000   3.000   9.000   27.000
  0.016  0.062   0.250  1.000   4.000  16.000   64.000
```

It should also be recalled that 0**A* is defined for non-negative values of *A*:

```
      0*0 1 2 3 4 5
1 0 0 0 0 0
```

⌹7

6.6. THE POWER FUNCTION FOR RATIONAL ARGUMENTS

When the power function is applied to a right argument consisting of successive integers, the successive elements of the result increase by a fixed factor. For example:

```
      4*0 1 2 3 4 5 6 7 8
1  4  16  64  256  1024  4096  16384  65536
```

The multiplying factor is 4. This same pattern is observed when the elements of the right argument are equally spaced, even though the spacing is not equal to 1. For example:

```
      4*0 2 4 6 8
1  16  256  4096  65536
```

The multiplying factor is now 16.

The first pattern above can be thought of as being obtained from the second by squeezing the odd integers between the even integers. Hence if the multiplying factor for the pattern 2*0 1 2 3 4 5 6 7 8 9 is 4, the factor for the pattern 2*0 2 4 6 8 must be 4×4, which agrees with the earlier observation.

Similarly the pattern 4*0 .5 1 1.5 2 2.5 3 3.5 4 4.5 5 can be thought of as being obtained by squeezing the entries .5, 1.5, 2.5, 3.5, and 4.5 between the integers 1, 2, 3, 4, and 5. In this case the multiplying factor must be 2, since the product of two factors (that is, 2×2) must be equal to the factor 4 which obtains for the pattern for the integers. Therefore:

```
      4*0 .5 1 1.5 2 2.5 3 3.5 4 4.5 5
1  2  4  8  16  32  64  128  256  512  1024
```

Similarly:

```
      9*0 1 2 3 4 5
1  9  81  729  6561  59049
      9*0 .5 1 1.5 2 2.5 3 3.5 4 4.5 5
1  3  9  27  81  243  729  2187  6561  19683  59049
      25*0 1 2 3 4 5
1  25  625  15625  390625  9765625
      25*0 .5 1 1.5 2 2.5 3 3.5 4 4.5
1  5  25  125  625  3125  15625  78125  390625  1953125
```


Each of the left arguments used above is a perfect square, that is, a number which is equal to some integer multiplied by itself. Thus 4 equals 2×2 and 9 equals 3×3 and 25 equals 5×5. Because of this property, the multiplying factor in each of the "squeezed" patterns is an integer. Since 3 is not a perfect square, a left argument of 3 gives a pattern in which the fractional powers are not integers:

```
      3*0 .5 1 1.5 2 2.5 3
   1.000   1.732   3.000   5.196   9.000  15.588  27.000
```

Nevertheless, the pattern is maintained, the multiplying factor is 1.732 (correct to 3 places) and 1.732×1.732 is (approximately) equal to 3.

From this it appears that 3*.5 is a number which multiplied by itself gives 3; it is called the square root of 3. Similarly, 2*.5 is the square root of 2, and (2*.5)×(2*.5) must equal 2.

The square root of a number can be obtained by "guessing and testing" much like the method described for division at the beginning of Chapter 3. For example, to obtain the square root of 2 we might try 1 (which is too small because 1×1 is less then 2), and 2 (which is too large since 2×2 is greater than 2), and then 1.5. Since 1.5×1.5 is 2.25, this is also too large. The next trial might be 1.4 (which is slightly too small), and the next might be 1.42. Better methods are developed in later chapters.

We can now produce a table of powers using right arguments of the form (⍳N)÷2:

```
      I←1 2 3 4 5 6 7 8 9
      J←0 .5 1 1.5 2 2.5
      I∘.*J
   1.000    1.000    1.000    1.000    1.000    1.000
   1.000    1.414    2.000    2.828    4.000    5.657
   1.000    1.732    3.000    5.196    9.000   15.588
   1.000    2.000    4.000    8.000   16.000   32.000
   1.000    2.236    5.000   11.180   25.000   55.902
   1.000    2.449    6.000   14.697   36.000   88.182
   1.000    2.646    7.000   18.520   49.000  129.642
   1.000    2.828    8.000   22.627   64.000  181.019
   1.000    3.000    9.000   27.000   81.000  243.000
```

The same reasoning can be applied to right arguments of the form (⍳*N*)÷*K* for any value of *K*:

```
      (⍳6)÷3
0.333  0.667  1  1.333  1.667  2

        I∘.*(⍳6)÷3
   1.000   1.000   1.000   1.000   1.000   1.000
   1.260   1.587   2.000   2.520   3.175   4.000
   1.442   2.080   3.000   4.327   6.240   9.000
   1.587   2.520   4.000   6.350  10.079  16.000
   1.710   2.924   5.000   8.550  14.620  25.000
   1.817   3.302   6.000  10.903  19.812  36.000
   1.913   3.659   7.000  13.391  25.615  49.000
   2.000   4.000   8.000  16.000  32.000  64.000
   2.080   4.327   9.000  18.721  38.941  81.000

      (⍳6)÷4
0.25  0.5  0.75  1  1.25  1.5
      I∘.*(⍳6)÷4
   1.000   1.000   1.000   1.000   1.000   1.000
   1.189   1.414   1.682   2.000   2.378   2.828
   1.316   1.732   2.280   3.000   3.948   5.196
   1.414   2.000   2.828   4.000   5.657   8.000
   1.495   2.236   3.344   5.000   7.477  11.180
   1.565   2.449   3.834   6.000   9.391  14.697
   1.627   2.646   4.204   7.000  11.386  18.520
   1.682   2.828   4.757   8.000  13.454  22.627
   1.732   3.000   5.196   9.000  15.588  27.000

      (⍳6)÷5
0.2  0.4  0.6  0.8  1  1.2
      I∘.*(⍳6)÷5
   1.000   1.000   1.000   1.000   1.000   1.000
   1.149   1.320   1.516   1.741   2.000   2.297
   1.246   1.552   1.933   2.408   3.000   3.737
   1.320   1.741   2.297   3.031   4.000   5.278
   1.380   1.904   2.627   3.624   5.000   6.899
   1.431   2.048   2.930   4.193   6.000   8.586
   1.476   2.178   3.214   4.743   7.000  10.330
   1.516   2.297   3.482   5.278   8.000  12.126
   1.552   2.408   3.737   5.800   9.000  13.967
```

The foregoing results have all involved applying the power function to non-integer right arguments and non-negative left arguments. In general it is not possible to apply it to non-integer right arguments together with *negative* left arguments. For example, to evaluate ¯4*.5 it would be necessary to determine a result *R* such that *R*×*R* equals ¯4. It is, however, impossible to find such a number, since the product of any number with itself is non-negative.

Chapter 7

THE RESIDUE FUNCTION AND FACTORING

7.1. THE RESIDUE FUNCTION

Consider the following expressions:

```
      3×0 1 2 3 4 5 6
0 3 6 9 12 15 18

      1+3×0 1 2 3 4 5 6
1 4 7 10 13 16 19

      2+3×0 1 2 3 4 5 6
2 5 8 11 14 17 20
```

From the first expression, it is clear that the numbers 0, 3, 6, 9, 12, 15 and 18 are each the product of 3 and some integer; they are therefore said to be *integer multiples* (or simply *multiples*) of 3. A number which is an integer multiple of 3 is also said to be *divisible by* 3.

The numbers 1, 4, 7, 10, 13, 16, and 19 are not divisible by 3; when divided by 3 they each yield an integer quotient and a *remainder* of 1. Similarly the numbers 2, 5, 8, 11, 14, 17, and 20 each yield a remainder of 2 when divided by 3. The remainder when dividing an integer by 3 must be either 2 or 1 or 0. If the remainder is 0 the number is, of course, divisible by 3.

The remainder obtained on dividing an integer *B* by an integer *A* is a function of *A* and *B*. This function is called the *remainder* or *residue* and is denoted by a vertical line as follows: *A*|*B*. For example:

```
      3|6
0

      3|7
1

      3|0 1 2 3 4 5 6 7 8 9 10
0 1 2 0 1 2 0 1 2 0 1

      5|0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 0 1 2 3 4 0
```

A function table for residue is shown in Figure 7.1. From this table it should be clear that the results of the expression A|B must be one of the integers 0, 1, 2, 3, etc., up to A-1. That is, the results belong to the vector ¯1+ιA.

|	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
3	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
4	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
5	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
6	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2
7	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0
8	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6

Left Domain:ι8
Right Domain:ι14
Body:(ι8)∘.|ι14
Symbol:|

Table of Residues

Figure 7.1

⌹1-2

7.2. NEGATIVE RIGHT ARGUMENTS

The following examples show how the residue function applies to negative right arguments:

```
      S←¯6+ι11
      S
¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5

      3×S
¯15 ¯12 ¯9 ¯6 ¯3 0 3 6 9 12 15

      3|3×S
0 0 0 0 0 0 0 0 0 0 0

      1+3×S
¯14 ¯11 ¯8 ¯5 ¯2 1 4 7 10 13 16

      3|1+3×S
1 1 1 1 1 1 1 1 1 1 1

      2+3×S
¯13 ¯10 ¯7 ¯4 ¯1 2 5 8 11 14 17

      3|2+3×S
2 2 2 2 2 2 2 2 2 2 2
```


It should be clear from these examples that the 3-residue of B (that is, 3|B) is obtained by adding or subtracting some integer multiple of 3 so that the result is the smallest non-negative number that can be so obtained. In general, the result A|B is the smallest non-negative integer that can be obtained by adding to, or subtracting from, B some integer multiple of A.

⌹3-4

7.3. DIVISIBILITY

The integer B is divisible by the integer A only if the A-residue of B is zero, that is, only if (A|B)=0. Since the expression (ι8)∘.|0,ι14 produced a table of residues (Table 7.1), the expression 0=(ι8)∘.|0,ι14 will produce the body of the corresponding divisibility table:

|	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
4	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
5	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
6	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
7	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
8	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

It is also interesting to arrange the integers 0 to 99 in a 10 by 10 table and then observe the patterns produced by first taking residues and then determining divisibility. For example:

```
      M←(10×0,ι9)∘.+0,ι9
      M
```

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

```
      5|M                                0=5|M
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0
0 1 2 3 4 0 1 2 3 4            1 0 0 0 0 1 0 0 0 0

      3|M                                0=3|M
0 1 2 0 1 2 0 1 2 0            1 0 0 1 0 0 1 0 0 1
1 2 0 1 2 0 1 2 0 1            0 0 1 0 0 1 0 0 1 0
2 0 1 2 0 1 2 0 1 2            0 1 0 0 1 0 0 1 0 0
0 1 2 0 1 2 0 1 2 0            1 0 0 1 0 0 1 0 0 1
1 2 0 1 2 0 1 2 0 1            0 0 1 0 0 1 0 0 1 0
2 0 1 2 0 1 2 0 1 2            0 1 0 0 1 0 0 1 0 0
0 1 2 0 1 2 0 1 2 0            1 0 0 1 0 0 1 0 0 1
1 2 0 1 2 0 1 2 0 1            0 0 1 0 0 1 0 0 1 0
2 0 1 2 0 1 2 0 1 2            0 1 0 0 1 0 0 1 0 0
0 1 2 0 1 2 0 1 2 0            1 0 0 1 0 0 1 0 0 1
```

⎕5-12

7.4. FACTORS

If *B* is divisible by *A*, then *A* is said to be a <u>factor</u> of *B*. For example, 3 is a factor of 12, and 5 is a factor of 15, and so on as shown below:

```
      12÷3                          3|12
4                              0

      15÷5                          5|15
3                              0

      9÷3                           3|9
3                              0

      24÷4                          4|24
6                              0

      24÷8                          8|24
3                              0
```


From these examples it is clear that the factors of any number *B* occur in <u>pairs</u> such that the product of the pair is equal to *B*. Thus, if 3 is a factor of 12 then 12÷3 (that is, 4) is also a factor and 3×4 is equal to 12. In general, if *A* is a factor of *B*, then *B*÷*A* is also a factor and the product of the pair of factors *A* and *B*÷*A* (that is, (*B*÷*A*)×*A*) is equal to *B*.

All possible factors of a number *B* can be found by simply trying to divide it by each of the integers from 1 up to and including *B*. For example, the number 24 has the following 8 factors:

```
      1 2 3 4 6 8 12 24
```

The factor pairs of 24 can be obtained by simply dividing 24 by the vector of its factors as follows:

```
      24÷1 2 3 4 6 8 12 24
24 12 8 6 4 3 2 1
```

Thus 1 and 24 are a pair; 2 and 12 are a pair, and so on.

The residue function can be used to determine which of the integers ⍳*B* are factors of *B*. For example, if *B* is 6, then:

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

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

The positions of the 1's in the last vector indicate which of the integers 1 2 3 4 5 6 are factors of 6. For example, since the third element is 1, then 3 is a factor, and since the fourth element is 0, then 4 is not a factor. The vector 1 1 1 0 0 1 can be used to pick out the actual factors 1 2 3 6 by means of the <u>compression</u> function discussed in the following section.

⎕13-16

7.5. COMPRESSION

The following examples show the behavior of the compression function:

```
      1 0 1 0 1/1 2 3 4 5
1 3 5

      1 0 1 0 1/2 3 5 7 11
2 5 11

      (⍳6)|6
0 0 0 2 1 0

      0=(⍳6)|6
1 1 1 0 0 1

      (0=(⍳6)|6)/⍳6
1 2 3 6

      (0=(⍳24)|24)/⍳24
1 2 3 4 6 8 12 24
```

The left argument of compression must be a vector of 1's and 0's and forms a "sieve" which picks up the element of the right argument wherever a 1 occurs in the left argument.

⌹17-18

7.6. PRIME NUMBERS

The following expressions yield all factors for each of the integers from 1 to 8:

```
      (0=(⍳1)|1)/⍳1
1

      (0=(⍳2)|2)/⍳2
1 2

      (0=(⍳3)|3)/⍳3
1 3

      (0=(⍳4)|4)/⍳4
1 2 4

      (0=(⍳5)|5)/⍳5
1 5

      (0=(⍳6)|6)/⍳6
1 2 3

      (0=(⍳7)|7)/⍳7
1 7

      (0=(⍳8)|8)/⍳8
1 2 4 8
```


Any number which has exactly two distinct factors is called a <u>prime</u> number. From the above examples it is clear that 2, 3, 5, and 7 are primes, but 1, 4, 6, and 8 are not. Thus a prime has no factors other than itself and 1.

If *K* is a vector of 0's and 1's, then +/*K* gives a count of the number of 1's in *K*. For example:

```
      +/1 1 0 1 0 0 0 1
4

      0=(⍳8)|8
1 1 0 1 0 0 0 1

      +/0=(⍳8)|8
4
```

The conditions for a prime number stated above in words can therefore be stated algebraically as follows: *B* is a prime number if the expression 2=+/0=(⍳*B*)|*B* has the value 1. For example:

```
      2=+/0=(⍳1)|1
0

      2=+/0=(⍳2)|2
1

      2=+/0=(⍳3)|3
1

      2=+/0=(⍳4)|4
0

      2=+/0=(⍳5)|5
1

      2=+/0=(⍳6)|6
0

      2=0+/=(⍳7)|7
1

      2=+/0=(⍳8)|8
0
```

This same test can be used to obtain all of the primes up to a certain value by applying it to a divisibility table. Consider, for example, the following tables:

|	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	0	1	0	1	0	1	0
3	1	2	0	1	2	0	1	2	0	1	2	0
4	1	2	3	0	1	2	3	0	1	2	3	0
5	1	2	3	4	0	1	2	3	4	0	1	2
6	1	2	3	4	5	0	1	2	3	4	5	0
7	1	2	3	4	5	6	0	1	2	3	4	5
8	1	2	3	4	5	6	7	0	1	2	3	4
9	1	2	3	4	5	6	7	8	0	1	2	3
10	1	2	3	4	5	6	7	8	9	0	1	2
11	1	2	3	4	5	6	7	8	9	10	0	1
12	1	2	3	4	5	6	7	8	9	10	11	0

Left D:⍳12
Right D:⍳12
Body:(⍳12)∘.|⍳12
Symbol:|

D	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	1	0	1	0	1	0	1	0	1
3	0	0	1	0	0	1	0	0	1	0	0	1
4	0	0	0	1	0	0	0	1	0	0	0	1
5	0	0	0	0	1	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1

Left D:⍳12
Right D:⍳12
Body:0=(⍳12)∘.|⍳12
Symbol:D

The last table shows divisibility. For example, the 1's in the 6th column show the position of the 4 factors of 6. Therefore the sum of the 6th column tells how many factors 6 has, and similarily for each column. The sum of the columns is obtained by summing the rows of the transpose of the table. Thus:

```
      +/⍉0=(⍳12)∘.|⍳12
1 2 2 3 2 4 2 4 3 4 2 6
```


The last result above gives the number of factors for each of the numbers 1 to 12. Therefore the expression 2=+/⍉0=(⍳12)∘.|⍳12 determines which numbers are primes:

```
      2=+/⍉0=(⍳12)∘.|⍳12
0 1 1 0 1 0 1 0 0 0 1 0
```

This vector of 0's and 1's can be used to compress the vector ⍳12 to finally pick out all of the primes up to 12:

```
      (2=+/⍉0=(⍳12)∘.|⍳12)/⍳12
2 3 5 7 11
```

⎕19-24

Chapter 8

MONADIC FUNCTIONS

8.1. INTRODUCTION

Each of the functions discussed thus far have applied to two quantities. Thus in the expressions 3×4 and 3+4 and 3⌈4, each of the functions ×, +, and ⌈ apply to the two quantities 3 and 4. These quantities are called the *arguments* of the function; the one to the left of the function is called the *first* or *left* argument, and the one to the right is called the *second* or *right* argument.

A function having two arguments is said to be *dyadic*, the prefix *dy* meaning two. There are also functions which apply to one argument; they are called *monadic* functions. The following examples show a monadic function which is called the *factorial* function:

```
      !1
1
      !2
2
      !3
6
      !4
24
```

```
      !5
120
      !6
720
      !7
5040
      !8
40320
```

From the examples it should be clear that factorial 3 is the product of the factors 1 2 3, factorial 4 is the product of the factors 1 2 3 4, and so on. The examples also illustrate a point which applies to all monadic functions - the symbol for the function (in this case, !) precedes its single argument.

The argument of a monadic function may (like the arguments of a dyadic function) be a vector. For example:

```
      !1 2 3 4 5 6 7 8
1 2 6 24 120 720 5040 40320
```


8.2. NEGATION

Negation is a monadic function denoted by the symbol -. For example:

```
      -3
¯3
      -5
¯5
      -¯5
5
      --5
5
      -2 3 5.8
¯2 ¯3 ¯5.8
```

```
      X←3
      -X
¯3
      S←2 3 5
      -S
¯2 ¯3 ¯5
      --S
2 3 5
```

From these examples it should be clear that negation of a number *B* is equivalent to subtracting *B* from zero; that is, -*B* is equivalent to 0-*B*. In other words, negation changes the sign of its argument.

It is also apparent from the examples that the symbol used for the monadic function of negation is the same as that already used for the dyadic function of subtraction. This might be expected to cause confusion, but it does not. For example:

```
      4-3
1
      4×-3
¯12
      4--3
7
```

Thus the symbol - denotes subtraction if it is preceded by an argument, but denotes negation if it is preceded by a function.

This double use of symbols (once for a dyadic function and once for a monadic function) will be applied to many other symbols as well as the -. For example, +, ×, ÷, ⌈, ⌊, and |, already used for dyadic functions, will be used to denote monadic functions as well.

8.3. RECIPROCAL

The reciprocal function is a monadic function denoted by ÷ and defined as follows: ÷*B* is equal to 1÷*B*. For example:

```
      ÷2
0.5

      ÷4
0.25

      S←⍳10
      S
1 2 3 4 5 6 7 8 9 10

      R←÷S
      R
1 0.5 0.3333 0.25 0.2 0.1667 0.1429 0.125 0.1111 0.1

      S×R
1 1 1 1 1 1 1 1 1 1
```

⍞4-5

8.4. MAGNITUDE

The numbers 5 and ¯5 are said to have the same size or magnitude, namely 5. In other words, the magnitude of a number is a function which ignores the sign of the number. For example:

```
      |5
5

      |¯5
5

      S←¯6+⍳11
      S
¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5

      |S
5 4 3 2 1 0 1 2 3 4 5

      T←6 ¯3 2 ¯5 4
      T÷T
1 1 1 1 1

      T÷|T
1 ¯1 1 ¯1 1
```

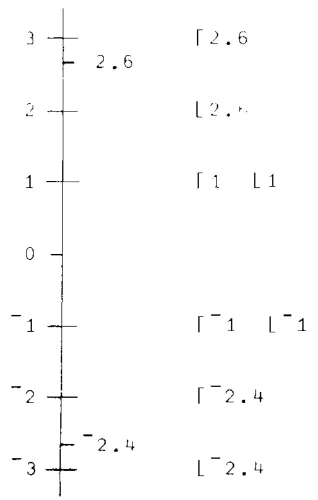
⍞6

8.5. FLOOR AND CEILING

The floor function is denoted by ⌊ and yields the next integer just below or equal to the argument. The ceiling function is denoted by ⌈ and yields the next integer just above or equal to the argument. For example:

```
      ⌊3                               ⌈3
3                                3

      ⌊3.14                            ⌈3.14
3                                4

      ⌊¯3.14                           ⌈¯3.14
¯4                               ¯3

      ⌊¯3                              ⌈¯3
¯3                               ¯3

      ⌊¯1.5 ¯1 ¯.5 0 .5 1 1.5          ⌈¯1.5 ¯1 0 .5 1 1.5
¯2 ¯1 ¯1 0 0 1 1                 ¯1 ¯1 0 1 1 2
```

The floor and ceiling functions are easily visualized by drawing the integers as the floors (and ceilings) in a building as follows:



The following examples illustrate how the monadic function floor is related to the dyadic function residue:

```
      17÷5
3.4

      ⌊17÷5
3

      (17-5|17)÷5
3
```

⌸7

8.6. COMPLEMENT

The complement function is denoted by ~ and applies only to logical arguments (that is, 0 and 1). When applied to 0 it produces 1, and when applied to 1 it produces 0. For example:

```
      ~1
0

      ~0
1

      ~1 0 1 0 1 1
0 1 0 1 0 0

      0=3|⍳12
0 0 1 0 0 1 0 0 1 0 0 1

      ~0=3|⍳12
1 1 0 1 1 0 1 1 0 1 1 0

      (~0=3|⍳12)/⍳12
1 2 4 5 7 8 10 11

      (0≠3|⍳12)/⍳12
1 2 4 5 7 8 10 11
```

The symbol ~ is called tilde.

⌸8-10

8.7. SIZE

The number of elements in a vector *V* is called the size of the vector. Size is therefore a monadic function and is denoted by ⍴. For example:

```
      V←2 3 5 7 11
      ⍴V
5
      X←⍳7
      ⍴X
7
      ⍴X[2 3 5]
3
      ⍴X[⍳2]
2
```

When applied to a table, the function ⍴ yields a two-element vector giving the number of rows in the table followed by the number of columns. For example:

```
      T←2 3 5∘.×⍳7
      ⍴T
3 7
      ⍴⍉T
7 3
```

⌸11

Chapter 9

FUNCTION DEFINITION

9.1. INTRODUCTION

The expression `0=3|X` was shown (in Chapter 7) to determine whether the argument *X* was divisible by 3. For example:

```
      0=3|9
1
      0=3|10
0
```

The expression `0=3|X` is therefore a monadic function of *X* in the sense that for any particular value assigned to *X*, the expression yields a particular corresponding value.

Unlike the functions floor, ceiling, and magnitude (which have the symbols ⌊, ⌈, and |), the function determined by the expression `0=3|X` has no special single symbol to denote it. It would, of course, be impractical to assign a special symbol to every possible such expression. However, it is important to be able to assign a name to any such expression which happens to be of interest at the moment, and then be able to use that name for the function just as ⌊, ⌈, and | are used for the floor, ceiling, and magnitude functions.

The name *DT* is assigned to the function determined by the expression `0=3|X` in the following manner:

```
      ∇Z←DT X
      Z←0=3|X ∇
```

The above is called <u>definition of the function</u> *DT*. Once the function *DT* has been so defined, it can be used like any other monadic function as follows:

```
      DT 9
1
      DT 10
0
      DT ⍳10
0 0 1 0 0 1 0 0 1 0
```


The symbol ∇ which begins and ends a function definition is called <u>del</u>.

Any number of such functions may be defined, but they must, of course, be given distinct names. These function names, like the names introduced for values in Chapter 1, must begin with a letter but may include both letters and digits. For example:

```
      ∇Z←D4 X
      Z←0=4|X ∇

      D4 ⍳10
0 0 0 1 0 0 0 1 0 0

      ∇ Z←D5 X
      Z←0=5|X ∇

      D5 ⍳10
0 0 0 0 1 0 0 0 0 1

      ∇Z←Q X
      Z←(X-3)×(X-5) ∇

      Q 6
3
      Q 7
8
      Q ⍳7
8 3 0 ¯1 0 3 8
```

The rules for determining the meaning of a function definition are very simple: when the function is applied to an argument, that argument is substituted for each occurrence of the name *X* in the second line of the function definition, and the result thereby assigned to the name *Z* is the result of the function. For example, to evaluate `Q 7`, the 7 is substituted for *X* to yield

```
      Z←(7-3)×(7-5)
```

This is evaluated to yield the result 8. Hence:

```
      Q 7
8
```

Functions such as floor and ceiling which have been assigned special fixed symbols will now be called <u>primitive functions</u> in order to distinguish them from the new class of <u>defined functions</u> just introduced. A defined function can be used within expressions, just as primitives are. For example:

```
      Q 6
3
      4×Q 6
12
      DT 12
1
      DT 4×Q 6
1
      Q Q 6
0
```

⌹5-7

9.2. DEFINITION OF DYADIC FUNCTIONS

The expression 0=*X*|*Y* was shown (in Chapter 7) to determine whether the argument *X* is a factor of the argument *Y*. For example:

```
      0=5|9
0
      0=7|21
1
```

The expression 0=*X*|*Y* is therefore a dyadic function of the arguments *X* and *Y* in the sense that for any particular values of *X* and *Y* the expression yields a particular corresponding value.

The name *F* is assigned to the dyadic function determined by the expression 0=*X*|*Y* in the following manner:

```
      ∇Z←X F Y
      Z←0=X|Y ∇
```


The function *F* can now be applied to pairs of arguments as illustrated below:

```
      5 F 9
0
      7 F 21
1
      5+7 F 21
6
      (5×7) F (5×21)
1
```

⌹8-13

9.3. A FUNCTION TO GENERATE PRIMES

In Chapter 7 it was shown that the expression

```
      (2=+/⍉0=(ιN)∘.|ιN)/ιN
```

would produce a vector of all the primes up to the integer *N*. Therefore a function *PR* can be defined to generate primes as follows:

```
      ∇Z←PR X
      Z←(2=+/⍉0=(ιX)∘.|ιX)/ιX∇
```

The following examples show the use of the function *PR*:

```
      PR 12
2 3 5 7 11
      +/PR 12
28
      PR 55
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
```

⌹14

9.4. TEMPERATURE SCALE CONVERSION FUNCTION

The Centigrade scale and the Fahrenheit scale are two different scales for measuring temperature. For any given temperature reading in Centigrade there is therefore a corresponding value in Fahrenheit; in other words, the Fahrenheit value is a function of the Centigrade value. This function will be expressed as a defined function called *CTOF* (for Centigrade TO Fahrenheit).

The Centigrade scale has 100 degrees between the freezing and boiling points of water, whereas the Fahrenheit scale has 180 degrees between these same points. Therefore any Centigrade reading *X* must be multiplied by 180 and divided by 100: that is, 180×*X*÷100. Moreover, 0 degrees Centigrade (the freezing point of water) corresponds to 32 degrees Fahrenheit and so it is necessary to add 32 to the foregoing expression, giving 32+180×*X*÷100. The conversion function *CTOF* may therefore be defined and used as follows:

```
      ∇Z←CTOF X
      Z←32+180×X÷100 ∇

      CTOF 0
32
      CTOF 100
212
      CTOF ¯40 ¯20 0 20 40 60 80 100
¯40 4 32 68 104 140 176 212
```

The function *CTOF* determines the Fahrenheit value as a function of the Centigrade value. It is, of course, also possible to define a function *FTOC* which determines the Centigrade value as a function of the Fahrenheit value:

```
      ∇Z←FTOC X
      Z←100×(X-32)÷180 ∇

      FTOC ¯40 4 32 68 104 140 176 212
¯40 ¯20 0 20 40 60 80 100
      CTOF FTOC ¯40 4 32 68 104 140 176 212
¯40 4 32 68 140 176 212
      FTOC CTOF ¯40 ¯20 0 20 40 60 80 100
¯40 ¯20 0 20 40 60 80 100
```

The last two lines above illustrate the fact that the function *FTOC* undoes the work of *CTOF*, and the preceding two lines illustrate that *CTOF* undoes the work of *FTOC*. The functions *FTOC* and *CTOF* are therefore inverse functions.

⌸15

9.5. FUNCTIONS ON RATIONALS

If *X* is a vector of two integer elements and *Y* is a vector of two integer elements, then ÷/*X* is a rational and ÷/*Y* is a rational. Moreover, as shown in Section 5.7, the product (÷/*X*)×(÷/*Y*) is equal to ÷/(*X*×*Y*). Therefore, the following function multiplies two rationals to produce the two element vector which represents their product:

```
      ∇Z←X P Y
      Z←X×Y ∇
```

For example:

```
      3 4 P 7 5
21 20
      ÷/3 4 P 7 5
1.05
      (÷/3 4)×(÷/7 5)
1.05
```

Similarly, the following function will add rationals:

```
      ∇ Z←X A Y
      Z←(+/X×⌽Y ),X[2]×Y[2] ∇
```

For example:

```
      3 4 A 7 5
43 20
      ÷/3 4 A 7 5
2.15
      (÷/3 4)+(÷/7 5)
2.15
```

⌸16-18

9.6. TRACING FUNCTION EXECUTION

A function can be defined by a single expression (as in the examples thus far), or it can be defined by a sequence of expressions. For example:

```
     ∇ Z←R X
[1]    T1←4×X
[2]    T2←3×X*2
[3]    T3←2×X*3
[4]    Z←T1+T2+T3∇

       R 2
36
       R 2 3 4
36 93 192
```

The statements are executed in the order in which they appear on the page, and each is identified by its number appearing in brackets on the left.

To understand the behavior of a function it is often helpful to examine some of the intermediate results produced by each of the individual statements in its definition. To indicate that each intermediate result produced in executing the function *R* is to be displayed, we would write

```
      TΔR←1 2 3 4
```

Thereafter, the execution of *R* would be accompanied by a display of the intermediate results as follows:

```
      Q←R 2
[1]   8
[2]   12
[3]   16
[4]   36

      Q
36
      Q←R 2 3 4
[1]   8 12 16
[2]   12 27 48
[3]   16 54 128
[4]   36 93 192

      Q
36 93 192
```

Such a display of the steps of execution of a function is called a trace of the function. The name *TΔR* used in initiating the trace of the function *R* denotes the trace control vector for *R*. In the preceding example, *TΔR* was set to trace every line of *R*, but it could be set to trace only some of them. For example:

```
      TΔR←1 3
      Q←R 2 3 4
[1]   8 12 16
[3]   16 54 128
```

Moreover, if *TΔR* is set to 0, no tracing is performed:

```
      TΔR←0
      Q←R 2 3 4
      Q
36 93 192
```


Chapter 10

THE ANALYSIS OF FUNCTIONS

10.1. INTRODUCTION

The problem of converting temperatures from the Centigrade to the Fahrenheit scale, which was handled by the function *CTOF* of Chapter 9, is often handled by simply providing a table covering the values of interest. For example, Table 10.1 would suffice for a range of temperatures just above the freezing point of water:

C	F
0	32
1	33.8
2	35.6
3	37.4
4	39.2
5	41
6	42.8
7	44.6
8	46.4
9	48.2
10	50

A Table Representation of the Function
CTOF for Centigrade Values Near Zero
Table 10.1

Such a table is often more convenient to use than to evaluate the expression 32+180×*C*÷100 (used in the definition of the function *CTOF*) for each conversion. However, such a tabular representation of a function also has its disadvantages; it provides only a limited set of values and could not, for example, be used directly to find the Fahrenheit equivalent of 25 *C* (which lies outside of the tabled values) or of 5.64 degrees Centigrade (which lies between two of the tabled values). For this reason it is often desirable to determine from such a table the algebraic expression which would produce the same function as that represented by the table.

To appreciate the problem of deriving an algebraic expression for a function represented only by a table, suppose that the expression 32+180×*C*÷100 is not known and that the only information known about the function is that contained in Table 10.1. One might begin by observing that each Fahrenheit value is at least 32 more than the corresponding Centigrade value, and therefore guess that the desired function is approximately $32+C$. The next step is to append to Table 10.1 a column of values for the function $32+C$ so that they can be compared with the tabled values of F:

C	F	$32+C$
0	32	32
1	33.8	33
2	35.6	34
3	37.4	35
4	39.2	36
5	41	37
6	42.8	38
7	44.6	39
8	46.4	40
9	48.2	41
10	50	42

Although the first entries in the columns F and $32+C$ agree (both are 32), the second entry falls short by 0.8, the third entry by 1.6, etc. It therefore appears that one should add $0.8\times C$ to the expression $32+C$, yielding $32+C+.8\times C$ or, more simply, $32+1.8\times C$. If a column of values for $32+1.8\times C$ is appended to the foregoing table and compared with the column F it will be seen that this is the required expression.

The process of determining an expression for a function from a table of the function will be referred to as analyzing the table or, alternatively, as analyzing the function represented by the table. The analysis of tables is not only an interesting puzzle, it is also a problem of the greatest importance, since it underlies every scientific discipline. The reason is that in every area of science and technology, one attempts to determine the functional relationships between various quantities of interest. Thus one wishes to know how the acceleration of an automobile depends on the power of the engine, how the gasoline consumption depends on the speed, how the length of life of the brakes depends on the area of the brake-shoes, how the electric current supplied to the headlamps depends on the battery voltage, how the weight limit of a suspension bridge depends on the size of the cable used, and so on. Moreover, it is important to be able to express these relations algebraically so that it becomes easy to calculate any new values needed.

However, the relationships between two quantities are normally determined by experiments in which the corresponding values of the quantities of interest are measured. Such experiments can only yield a table of

values--they do not yield an algebraic expression for the function. The algebraic function must be determined by analysis of the table.

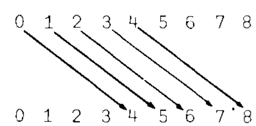
In practice one might do a few experiments, make a small table, derive from it an algebraic expression for the functional relationship, and then do a few more experiments to test (and perhaps revise) the derived expression. in a book this process cannot be simulated completely since we can only give fixed tables resulting from certain experiments, and cannot allow the reader to choose the values to be included in these tables. However, if a computer is available, one person (the teacher) can enter the definition of any function so that another person (the student) can "experiment" with the function at will by simply applying it to any desired arguments. If the student is not permitted to see the original definition of the function, then he can be given the problem of experimenting with the function, determining a table, and deriving from it his own definition of (i.e., algebraic expression for) the function.

The remainder of this chapter will be devoted to the analysis of tables. Three methods are treated: maps, graphs, and difference tables. Difference tables provide the most powerful method, but maps and graphs are treated first because they are easier to comprehend and because maps have already been used for other purposes in earlier chapters. A fourth and more powerful method (called curve-fitting) is treated in Chapter 16.

10.2. MAPS

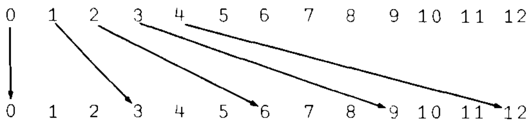
If one first makes a map of a table, then the map can be used as a guide in the analysis of the table. In order to see what guidance the map can provide, it is useful to recall the maps of two simple functions.

If $X\leftarrow 0,\iota 4$, then the map of the function $4+X$ against X appears as follows:



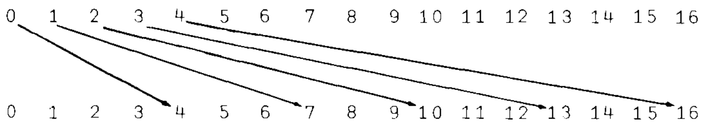
From this it is clear that the addition of a constant (in this case 4) appears in the map as a uniform translation, that is, each point is moved by the same amount, and the mapping arrows all have the same slope.

If, as before, X←0,ι4, then the map of the function 3×X appears as follows:



From this it is clear that multiplication by a constant (in this case 3) appears in the map as a uniform spreading, that is, the distance between the successive arrowheads (in this case 3) is the constant of multiplication.

Consider now the mapping of a function which involves both addition and multiplication, say 4+3×X:



The effects of uniform translation and uniform spreading are now superimposed, but it is still possible to recognize the individual effects of each. These observations will now be applied to the analysis of the function shown in Figure 10.2.

X	Y
2	1
3	3
4	5
5	7
6	9
7	11

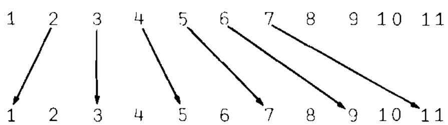
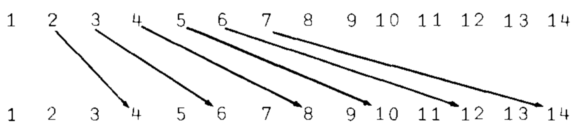


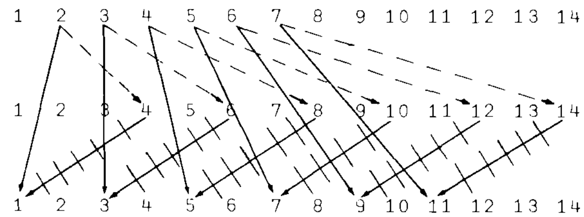
Table and Map of a Function

Figure 10.2

It is usually best to try to account for the multiplication (spreading) first. In this case adjacent arrowheads are separated by 2 units and so the multiplication factor is 2. Therefore we make a map of the function 2×X as follows:



The map of 2×X is now combined with the map of the original table as follows:



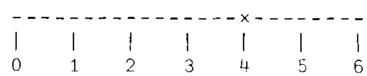
In this map, the original table is represented by normal lines as usual, and the approximating function 2×X is represented by broken lines. The scored lines lead from the results of 2×X to the results of the tabled function and therefore represent the function that must be applied to the function 2×X to yield the tabled function. Since the scored lines all have the same slope, this function must be a translation (by ¯3), that is, the addition of ¯3. The required function is therefore ¯3+2×X, as may be verified by computing the values for the case X←2 3 4 5 6 7 and comparing them with the second column of Figure 10.2.

The functions analyzed by maps thus far have all been of the form A+B×X where A and B are constants. In the analysis of more complex functions (such as 3+(5×X)+(2×X*2)), maps are of little help and one of the other methods should be used.

10.3. GRAPHS

Each row of a function table such as Table 10.1 consists of a pair or numbers representing an argument and a corresponding function value. Any other way of showing the pairing of the numbers in each of the rows is obviously a possible way of representing the function. For example, in a map, each pairing is shown by an arrow from the argument to the corresponding function value.

Any single number can be represented by marking off the integers at equal intervals along a line and then placing a cross on the line to show the desired value. For example 4 might by represented as follows:



A whole set of numbers could be represented by a set of crosses on such a line. Consider, for example, the function table of Table 10.3.

X	*Y*
1.5	5.5
2.0	4.5
2.5	3.5
3.0	2.5
4.0	0.5

Table of a Function

Table 10.3

The set of arguments shown in the first column would be represented as follows:

```
------x-x-x-x---x--------
|   |   |   |   |   |   |
0   1   2   3   4   5   6
```

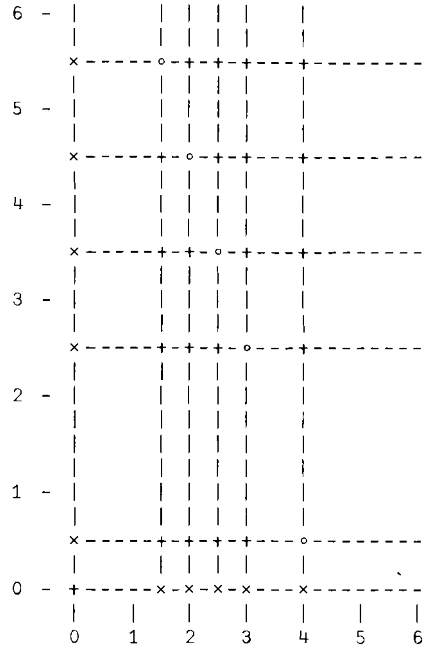
If the set of function values *Y* are now represented similarly along a vertical line rising from the 0-point of the first line, the picture appears as follows:

```
6 - |
    |
    x
    |
5 - |
    |
    x
    |
4 - |
    |
    x
    |
3 - |
    |
    x
    |
2 - |
    |
    |
    |
1 - |
    |
    x
    |
0 - |-----x-x-x-x---x--------
    |   |   |   |   |   |   |
    0   1   2   3   4   5   6
```

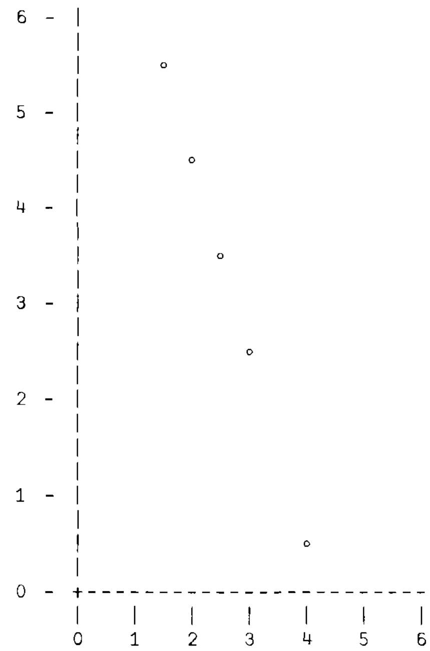

If vertical lines are drawn through the crosses on the horizontal line, and if horizontal lines are drawn through the crosses on the vertical line, the picture appears as follows:

```
6 - |     | | | |   |
    |     | | | |   |
    x-----+-+-+-+---+--------
    |     | | | |   |
5 - |     | | | |   |
    |     | | | |   |
    x-----+-+-+-+---+--------
    |     | | | |   |
4 - |     | | | |   |
    |     | | | |   |
    x-----+-+-+-+---+--------
    |     | | | |   |
3 - |     | | | |   |
    |     | | | |   |
    x-----+-+-+-+---+--------
    |     | | | |   |
2 - |     | | | |   |
    |     | | | |   |
    |     | | | |   |
    |     | | | |   |
1 - |     | | | |   |
    |     | | | |   |
    x-----+-+-+-+---+--------
    |     | | | |   |
0 - +-----x-x-x-x---x--------
    |   |   |   |   |   |   |
    0   1   2   3   4   5   6
```

The pairing of each argument with its particular function value can now be shown by placing a point at the intersection of the lines through them as follows:



In practice, one actually draws neither the lines nor the crosses, but simply marks the points of intersection, producing the following less cluttered picture:



This picture is called a *graph* or *plot* of the function of Table 10.3. Negative values are included by simply extending the horizontal line leftward from the zero and the vertical line downward from the zero.

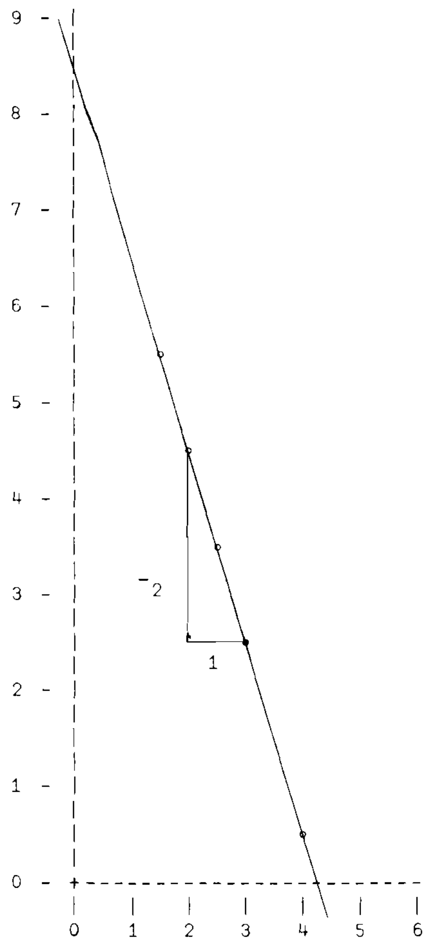
The vertical line of the graph (which passes through the zero point of the horizontal line) is called the *vertical axis* or *Y-axis*, and the horizontal line (through the zero of the vertical line) is called the *horizontal axis* or *X-axis*. The names are derived from the (arbitrary) convention that the argument of a function is often called X and the result is often called Y, so that the expression for a function is in the form $Y \leftarrow F\ X$.

⎕3-4

10.4. INTERPRETING A LINEAR GRAPH

If a ruler is laid along the points in the preceding graph, the points will be seen to lie in a straight line. If one graphs a number of functions of the form $A+B\times X$ (where A and B are fixed values), it will be seen that the points in the graph of any such function lie in a straight line. Conversely, every graph whose points all lie in one straight line represents a function of the form $A+B\times X$. Moreover, the values of A and B can be easily determined *from the graph*.

Consider, for example, Figure 10.4 which shows the graph of the function of Table 10.3 with a line drawn through the points. Any point on the line (not only the five taken from the table) represents a point of the function. For example, if the argument X is 1, then the function value Y is 6.5, and if X is 0, then Y is 8.5. But if X is 0, the value of the expression $A+B\times X$ is simply A. Hence, for this function A must have the value 8.5.



Graph of Function of Table 10.3

Figure 10.4

Moreover, B is clearly the amount that the function changes when the argument is changed from some value to a value greater by 1. Since the function is equal to 4.5 for X=2 and is equal to 2.5 for X=3 this change is equal to 2.5-4.5 or ¯2. Therefore B is equal to ¯2. Finally, the expression for the function must be 8.5+¯2×X. This may be verified by evaluating the expression for the values X←1.5 2 2.5 3 4 and comparing the results with the second column of Table 10.3.

To summarize, the values of A and B can be determined from a straight-line graph as follows:

(1) The value of A is the height at which the graph line crosses the vertical axis (where X=0).

(2) The value of B is the change in height corresponding to a change of 1 on the horizontal axis. ⎕5-6

A function table whose graph does not form a straight line is not as easy to interpret as a straight line graph. However, the graph can still provide some guidance.

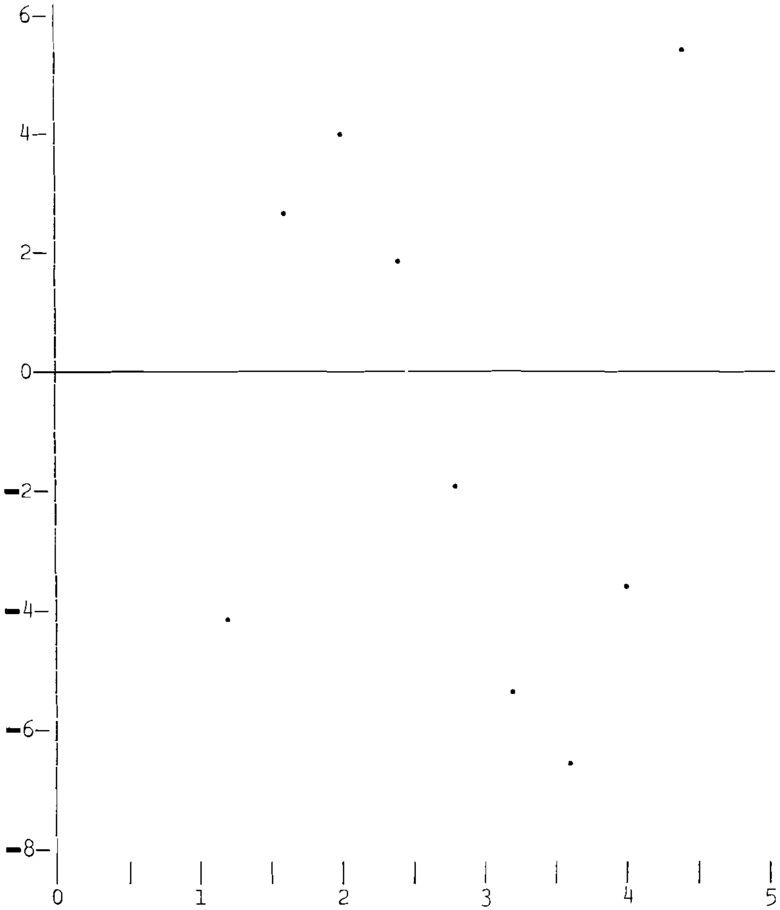
Consider, for example, Figure 10.5 which shows a function table and the corresponding graph. The points do not lie in a straight line, but have been joined by a smooth curve which suggests the function values which should be obtained between the points included in the table itself.

A number of interesting characteristics of the function can be seen clearly in its graph. For example, it is clear that the function reaches a low point for an argument value of X equal to approximately 3.5 and that it reaches a high point for a value of X a little less than 2. Moreover, it is easy to spot those argument values for which the function has a zero value, namely for X equal to 1.4 or 2.6 or 4.2.

Since X-1.4 is zero for X=1.4 and X-2.6 is zero for X=2.5 and X-4.2 is zero for X=4.2, then the expression

(X-1.4)×(X-2.6)×(X-4.2)

is zero for X equal to either 1.4 or 2.6 or 4.2. Hence it will agree with the given function at least for these three values of the argument X. In order to see how well this expression agrees with the given function for all points, it can be graphed together with the given function as shown in Figure 10.6.



X	Y
1.2	¯4.20
1.6	2.60
2.0	3.96
2.4	1.80
2.8	¯1.96
3.2	¯5.40
3.6	¯6.60
4.0	¯3.64
4.4	5.40

Table and Graph of a Function

Figure 10.5

A comparison of the two curves in Figure 10.6 shows that they have the same general shape, that is, the values for the given function appear to be larger than those of the expression by a fixed ratio. A value for this ratio can be determined from two corresponding points, say for an argument value of 2.4. The two corresponding function values are seen to be 1.8 and .36, and the ratio is therefore 1.8÷.36, that is, 5.

A better approximation to the given function is therefore given by 5 times the expression just tried, that is:

```
      5×(X-1.4)×(X-2.6)×(X-4.2)
```

Evaluation of this function for each of the argument values appearing in the first column of Table 10.5 shows that it agrees exactly with the function given in the second column.

⌹7-8

10.5. THE TAKE AND DROP FUNCTIONS

The dyadic functions <u>take</u> and <u>drop</u> are denoted by ↑ and ↓, respectively. The following expressions illustrate their use:

```
      Y←0 1 4 9 16 25 36

      3↑Y                 3↓Y
0  1  4           9  16  25  36
      2↑Y                 2↓Y
0  1              4  9   16  25  36
      ¯3↑Y                ¯3↓Y
16  25  36        0  1   4  9
      ¯2↑Y                ¯2↓Y
25  36            0  1   4  9  16
```

The <u>take</u> function takes from its right argument the number of elements determined by the left argument, beginning at the front end if the left argument is positive and at the back end if it is negative. The drop function behaves similarly, dropping the indicated number of elements from the right argument.

If the left argument is greater than the number of elements of the right argument, then the extra positions are filled with zeros. For example:

```
      X←2 3 5 7
      6↑X
2  3  5  7  0  0
      ¯6↑X
0  0  2  3  5  7
```

⌹9-10

X	Y	(X-1.4)×(X-2.6)×(X-4.2)
1.20	¯4.20	¯0.84
1.30	¯1.88	¯0.38
1.40	0.00	0.00
1.50	1.48	0.30
1.60	2.60	0.52
1.70	3.37	0.67
1.80	3.84	0.77
1.90	4.02	0.80
2.00	3.96	0.79
2.10	3.67	0.73
2.20	3.20	0.64
2.30	2.56	0.51
2.40	1.80	0.36
2.50	0.93	0.19
2.60	0.00	0.00
2.70	¯0.97	¯0.19
2.80	¯1.96	¯0.39
2.90	¯2.92	¯0.58
3.00	¯3.84	¯0.77
3.10	¯4.67	¯0.93
3.20	¯5.40	¯1.08
3.30	¯5.98	¯1.20
3.40	¯6.40	¯1.28
3.50	¯6.61	¯1.32
3.60	¯6.60	¯1.32
3.70	¯6.32	¯1.26
3.80	¯5.76	¯1.15
3.90	¯4.87	¯0.97
4.00	¯3.64	¯0.73
4.10	¯2.02	¯0.40
4.20	0.00	0.00
4.30	2.46	0.49
4.40	5.40	1.08

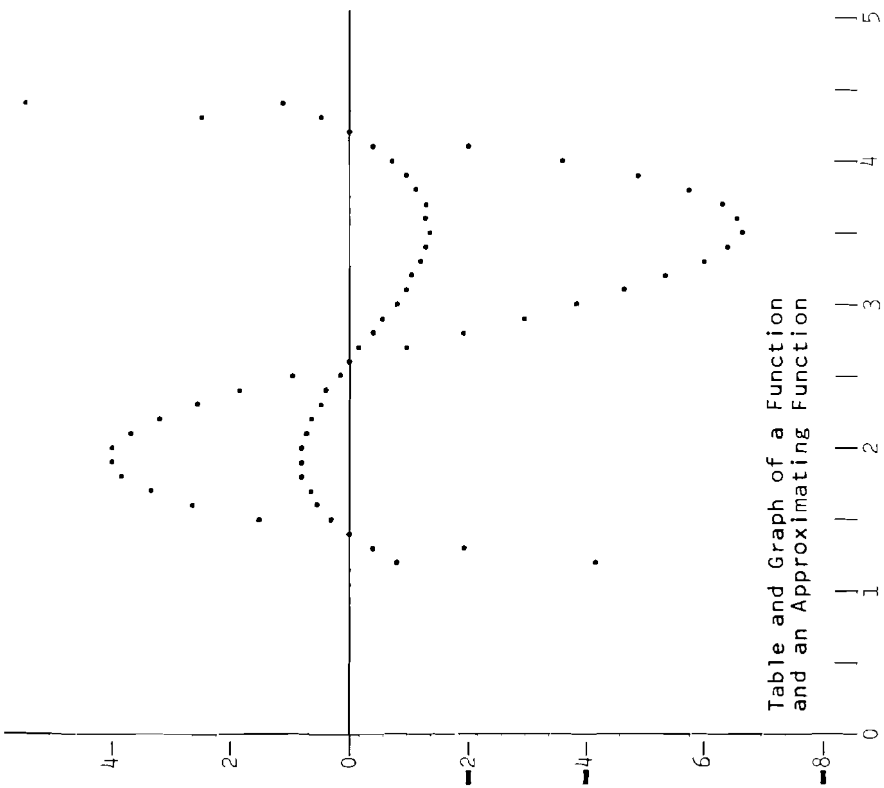


Table and Graph of a Function and an Approximating Function

Figure 10.6

10.6. DIFFERENCE TABLES

The __first difference__ of a vector *Y* is defined as the vector obtained by taking the difference between each of the pairs of adjacent elements of *Y*. For example, if *Y* is the vector

```
0 1 4 9 16 25 36 64 81 100
```

then the first difference of *Y* is the vector

```
1 3 5 7 9 11 13 15 17 19
```

More precisely, the first difference is the function *D* defined as follows:

```
∇Z←D Y
Z←(1↓Y)-(¯1↓Y)∇
```

For example:

```
      D Y
1 3 5 7 9 11 13 15 17 19
```

To understand the behavior of the function *D*, it may help to observe the effects of the terms 1↓*Y* and ¯1↓*Y* as follows:

```
      1↓Y
1 4 9 16 25 36 49 64 81 100

      ¯1↓Y
0 1 4 9 16 25 36 49 64 81
```

The subtraction of the second of these from the first clearly yields the differences between each of the adjacent elements of *Y*.

If *Y*←*F X* for some function *F* and some set of equally spaced arguments *X*, then the first difference of *Y* is also said to be the __first difference of the function__ *F*. For example, if *X*←0,⍳10 and *Y*←*X*⋆2 (that is, *Y* is the __square__ of *X*), then the vector

```
      D Y
1 3 5 7 9 11 13 15 17 19
```

is said to be the first difference of the square function (for the arguments *X*).

Viewed in terms of a function table, the vectors *X* and *Y* used in the preceding paragraph are the first and second columns of a function table. Attention will now be limited to function tables whose first column *X* is of the form 0,⍳*N*, that is, of the form 0 1 2 3 etc., up to some integer *N*. In the first section of Chapter 11, it will be shown how the methods developed can be applied to any set of equally spaced arguments such as 1.2 1.6 2.0 2.4 2.8 3.2, etc.

Since attention is being confined to argument sets of the form 0,⍳*N*, the argument column can be dropped from function tables without introducing ambiguity. For example, the single column on the left of Figure 10.7 shows this simplified form of the function table (for the function *CTOF*) of Table 10.1. The right side of the same figure shows a two-column table containing the function vector *F* and its first difference *D F*; such a table is called a <u>difference table</u>.

F
32
33.8
35.6
37.4
39.2
41
42.8
44.6
46.4
48.2
50

Abbreviated Function Table for Table 10.1

F	*D F*
32	1.8
33.8	1.8
35.6	1.8
37.4	1.8
39.2	1.8
41	1.8
42.8	1.8
44.6	1.8
46.4	1.8
48.2	1.8
50	1.8

Difference Table for the Function *CTOF* of Table 10.1

Function and Difference Table

Figure 10.7

10.7. FITTING FUNCTIONS OF THE FORM *A*+*B*×*X*

In using maps to analyze functions, it was found that any function of the form *A*+*B*×*X* could be recognized by the uniform spread between adjacent arrow points, and that the actual values of the constants *A* and *B* could be determined from the map. This type of function is analyzed even more easily with the aid of the difference table; the uniform spread is recognized by the fact that the elements of the first difference (which give the spacing between adjacent function values) are all the same. The constants *A* and *B* are simply the first row of the difference table, that is, 32 and 1.8 in Figure 10.7.

13-14

10.8. FACTORIAL POLYNOMIALS

In analyzing certain functions it will be found that the elements of the first difference are not all alike, and the function is therefore not of the form *A*+*B*×*X*. In such a case one may take a <u>second difference</u>, that is, the difference of the first difference. If this second difference is not constant, one takes a <u>third</u> difference, and so continues until a constant difference is reached.

For example, Table 10.8 shows a function table in which a constant difference is reached at the third difference.

Y	*D Y*	*D D Y*	*D D D Y*
5	¯2	8	¯6
3	6	2	¯6
9	8	¯4	¯6
17	4	¯10	¯6
21	¯6	¯16	¯6
15	¯22	¯22	
¯7	¯44		
¯51			

A Constant Third Difference

Table 10.8

The first row of the table is the vector `V←5 ¯2 8 ¯6`. The expression for the function is determined from the vector *V* as follows: *V* is first divided by the vector `!0 1 2 3` (that is, `1 1 2 6`) to obtain the vector *W* as follows:

```
      W←V÷!0 1 2 3
      W
5 ¯2 4 ¯1
```

The elements of *W* are then used to form the following expression:

```
      5+(¯2×X)+(4×X×(X-1))+(¯1×X×(X-1)×(X-2))
```

This expression represents the function exactly, as may be determined by evaluating it for the argument `0,⍳7` and comparing it with the first column of Table 10.8.

The method can be stated in general as follows: Calculate the successive columns of the difference table until a constant column is obtained. Then use the elements of the first row as follows:

Divide the first element by `!0` (that is, `1`).
Divide the second element by `!1` and multiply by *X*.
Divide the third element by `!2` and multiply by `X×(X-1)`.
Divide the fourth element by `!3` and multiply by `X×(X-1)×(X-2)`.
and so on.
Finally, add the expressions so obtained.

In other words, if the vector *V* is the first row of the difference table, then the expression

```
      (V[I]÷!I-1)× ×/X-¯1+⍳I-1
```

is evaluated for each value of *I* from `1` to `⍴V`, and the results are then added together.

The functions *X* and `X×(X-1)` and `X×(X-1)×(X-2)`, etc., are called <u>factorial polynomials</u>; *X* is called a factorial polynomial of <u>degree</u> `1`, and `X×(X-1)` is called a factorial polynomial of degree `2`, etc. In general, the factorial polynomial of degree *N* is given by the expression `×/X-¯1+⍳N`.

An explanation of why the method works will now be developed. The method is based on the fact that each of the functions *X* and `X×(X-1)` and `X×(X-1)×(X-2)`, etc., produce difference tables with particularly simple first rows, and on the fact that difference tables can be added and multiplied by constants in certain useful ways.

10.9. MULTIPLICATION AND ADDITION OF DIFFERENCE TABLES

The first difference of a vector has two very useful properties. If *Y* is any vector, if `D Y` is its first difference, and if *A* is any constant, then the first difference of the vector `A×Y` is equal to *A* times the first difference of *Y*; that is, `D A×Y` is equal to `A×D Y`. For example:

```
      Y←0 1 4 9 16 25 36 49
      D Y
1 3 5 7 9 11 13

      6×Y
0 6 24 54 96 150 216 294

      D 6×Y
6 18 30 42 54 66 78

      6×D Y
6 18 30 42 54 66 78
```

Clearly the same would be true of second differences, third differences, and so on. That is:

```
      |D A×Y        |D D A×Y        |D D D A×Y
      |A×D Y        |A×D D Y        |A×D D D Y
```

Therefore, if every element in a difference table is multiplied by some constant *A*, then it is still a proper difference table, but for the new function `A×Y` in its first column.

Similarly, if *Y1* and *Y2* are two vectors and if `D Y1` and `D Y2` are their first differences, then the first difference of the sum `Y1+Y2` is equal to the sum of the first differences; that is,

```
      D Y1+Y2
      (D Y1)+(D Y2)
```

Again, the same results apply to entire difference tables. Consequently, difference tables may be multiplied by constants and added together at will and the result is always a proper difference table.

10.10. DIFFERENCE TABLES FOR THE FACTORIAL POLYNOMIALS

The factorial polynomials of degrees 0 through 5 are shown below:

Degree	Polynomial
0	1
1	X
2	$X\times(X-1)$
3	$X\times(X-1)\times(X-2)$
4	$X\times(X-1)\times(X-2)\times(X-3)$
5	$X\times(X-1)\times(X-2)\times(X-3)\times(X-4)$

The polynomial of degree 2 has 2 occurrences of X, the polynomial of degree 3 has 3 occurrences of X, and so on. The function with a fixed value of 1 has been introduced as the polynomial of degree 0 in order to complete this pattern; it has 0 factors of X.

The difference tables for these factorial polynomials are shown in Figure 10.9. Previous tables shown have stopped at the first constant column, but these tables have been continued so that all have the same number of columns. Having the same number of columns, they can be added together. However, it is clear that any columns following a constant column will consist entirely of zeros.

Degree: 0
Function: 1

Y	DY	DDY			
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	
1	0	0	0		
1	0	0			
1	0				
1					

Degree: 1
Function: X

Y	DY	DDY			
0	1	0	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	
4	1	0	0		
5	1	0			
6	1				
7					

Degree: 2
Function: $X\times(X-1)$

Y	DY	DDY			
0	0	2	0	0	0
0	2	2	0	0	0
2	4	2	0	0	0
6	6	2	0	0	
12	8	2	0		
20	10	2			
30	12				
42					

Degree: 3
Function: $X\times(X-1)\times(X-2)$

Y	DY	DDY			
0	0	0	6	0	0
0	0	6	6	0	0
0	6	12	6	0	0
6	18	18	6	0	
24	36	24	6		
60	60	30			
120	90				
210					

Degree: 4
Function: $X\times(X-1)\times(X-2)\times(X-3)$

Y	DY	DDY			
0	0	0	0	24	0
0	0	0	24	24	0
0	0	24	48	24	0
0	24	72	72	24	
24	96	144	96		
120	240	240			
360	480				
840					

Degree: 5
Function: $X\times(X-1)\times(X-2)$
$\times(X-3)\times(X-4)$

Y	DY	DDY			
0	0	0	0	0	120
0	0	0	0	120	120
0	0	0	120	240	120
0	0	120	360	360	
0	120	480	720		
120	600	1200			
720	1800				
2520					

The Factorial Polynomials

Figure 10.9

The first row from each table is shown below, together with the degree of the polynomial it is taken from:

Degree	First Row of Difference Table					
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	2	0	0	0
3	0	0	0	6	0	0
4	0	0	0	0	24	0
5	0	0	0	0	0	120

Except for final zeros, the first row of the difference table for the factorial polynomial of order *N* is (*N*⍴0),!*N*, that is, *N* zeros followed by !*N*.

Consider now the function obtained as *A* times the zeroth order polynomial added to *B* times the first order polynomial, added to *C* times the second, etc.; that is, the function:

```
A+(B×X)+(C× ×/X-0 1)+(D× ×/X-0 1 2)+(E× ×/X-0 1 2 3)
    +(F× ×/X-0 1 2 3 4)
```

The difference table for this function will be *A* times the difference table for order 0, plus *B* times the difference table for order 1, etc. In particular, the first row of the difference table will be the sum of the following vectors:

```
A × 1   0   0   0   0   0
B × 0   1   0   0   0   0
C × 0   0   2   0   0   0
D × 0   0   0   6   0   0
E × 0   0   0   0  24   0
F × 0   0   0   0   0 120
```

This sum is clearly equal to (*A*,*B*,*C*,*D*,*E*,*F*)×1 1 2 6 24 120, or more simply (*A*,*B*,*C*,*D*,*E*,*F*)×!0,⍳5. Conversely, the values of *A*,*B*,*C*,*D*,*E*,*F* can be determined from the first row *V* of a difference table as follows: *A*,*B*,*C*,*D*,*E*, and *F* are the elements of the vector *V*÷!0,⍳5. This is the rule which was used in Section 10.8.

10.11. EXPRESSIONS FOR GRAPHS

Consider the function *F* defined and used as follows:

```
      ∇Z←F X
[1]    Z←(X-5)×(X-3)∇

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

A graph of the function *F* for the arguments *X* is shown in Figure 10.10. The pattern shown by the points of this graph is also shown by the 1's in the following result:

```
      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
```

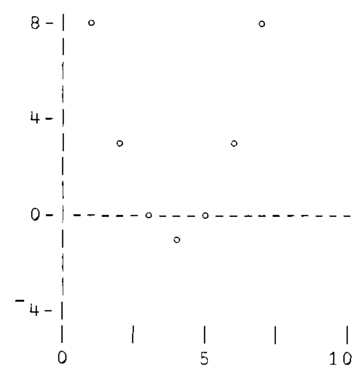


Figure 10.10

The vector *R* is simply the range of the function for the argument *X*, and the comparison between it and the set of values *V* will clearly yield a 1 at each point to be plotted in the graph.

A <u>bar chart</u> for the same function can be obtained by replacing the comparison for equality by a comparison for less-than-or-equal:

```
      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
```

⊞23-26

The expression *R*∘.=*V* will identify only those elements of *V* which agree exactly with elements of the range. For example:

```
      Y←X+.1
      Y
1.1  2.1  3.1  4.1  5.1  6.1  7.1
      W← F Y
      W
7.41  2.61  ¯0.19  ¯0.99  0.21  3.41  8.61
      R∘.=W
 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 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 0 0 0 0 0
 0 0 0 0 0 0 0
```


However, one might want to plot points where the argument is close. This could be done by taking the integer parts of the function values as follows:

```
      ⌊W
7  2  ¯1  0  3  8
      R∘.=⌊W
 0 0 0 0 0 0 1
 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 0 0 0 1 0
 0 1 0 0 0 0 0
 0 0 0 0 0 0 0
 0 0 0 0 1 0 0
 0 0 1 1 0 0 0
```

The comparison can also be made as loose or as tight as desired by simply computing the table |*R*∘.-*W* and then comparing it with any desired quantity. For example:

```
      T←|R∘.-W
      T
0.59  5.39  8.19  8.99  7.79  4.59  0.61
0.41  4.39  7.19  7.99  6.79  3.59  1.61
1.41  3.39  6.19  6.99  5.79  2.59  2.61
2.41  2.39  5.19  5.99  4.79  1.59  3.61
3.41  1.39  4.19  4.99  3.79  0.59  4.61
4.41  0.39  3.19  3.99  2.79  0.41  5.61
5.41  0.61  2.19  2.99  1.79  1.41  6.61
6.41  1.61  1.19  1.99  0.79  2.41  7.61
7.41  2.61  0.19  0.99  0.21  3.41  8.61
8.41  3.61  0.81  0.01  1.21  4.41  9.61

      .5≥T               1≥T                 2≥T
 0 0 0 0 0 0 0     1 0 0 0 0 0 1       1 0 0 0 0 0 1
 1 0 0 0 0 0 0     1 0 0 0 0 0 0       1 0 0 0 0 0 1
 0 0 0 0 0 0 0     0 0 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 1 0
 0 0 0 0 0 0 0     0 0 0 0 0 1 0       0 1 0 0 0 1 0
 0 1 0 0 0 1 0     0 1 0 0 0 1 0       0 1 0 0 0 1 0
 0 0 0 0 0 0 0     0 1 0 0 0 0 0       0 1 0 0 1 1 0
 0 0 0 0 0 0 0     0 0 0 0 1 0 0       0 1 1 1 1 0 0
 0 0 1 0 1 0 0     0 0 1 1 1 0 0       0 0 1 1 1 0 0
 0 0 0 1 0 0 0     0 0 1 1 0 0 0       0 0 1 1 1 0 0
```

⊞27

10.12. CHARACTER VECTORS

If *P* is a vector of the first five prime integers, then one can index it as shown in the following examples:

```
      P[2]
3
      P[3 1 2]
5  2  3
      P[2 5 4]
3  11  7
      P
2  3  5  7  11
```

Similarly, if *L* is a vector of the first five letters of the alphabet it may be indexed as follows:

```
      L[2]
B
      L[3 1 2]
CAB
      L[2 5 4]
BED
      L
ABCDE
```

The original value of the vector *L* could be assigned by the following expression:

```
      L←'ABCDE'
```

The quotes are necessary to indicate that the result is to be the actual string of characters *ABCDE* rather than some value which has been assigned to the name *ABCDE*. For example:

```
      PRIMES←2 3 5 7 11
      A←PRIMES
      B←'PRIMES'
      A[4 3 2 5]
7  5  3  11
      B[4 3 2 5]
MIRE
      ⍴A
5
      ⍴B
6
```

⌸28

Characters other than letters can also be used. For example:

```
      C←'*+ABCD'
      C[2 2 1 5 1 3 1 6 1 2 2]
++*C*A*D*++
      ' *'[2 2 1 2 2 1 2 2]
** ** **
```


The last example above illustrates how the space may be used as a character.

⌸29

Indexing of a character vector can also be used to display the graphs produced in Section 10.9 in a more pleasing and more readable form. For example, if *R* and *V* are the vectors defined in Section 10.9, then:

```
      R
8  7  6  5  4  3  2  1  0  ¯1
      V
8  3  0  ¯1  0  3  8
      M←R∘.=V
      M
 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

      1+M
2 1 1 1 1 1 2
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 2 1 1 1 2 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 2 1 2 1 1
1 1 1 2 1 1 1

      ' *'[1+M]
*     *




 *   *


  * *
   *
```

In order to make such graphing easy we might even define a graphing function *GR* as follows:

```
      ∇Z←GR X
[1]   Z←' *'[1+X]∇

      GR M
*     *




 *   *


  * *
   *

      GR (⍳8)∘.≥⍳8
*
**
***
****
*****
******
*******
********
```

⌹30-31

Chapter 11

INVERSE FUNCTIONS

11.1. INTRODUCTION

The functions *CTOF* (for Centigrade TO Fahrenheit), and *FTOC*, introduced in Chapter 10, are an example of a pair of mutually inverse functions; that is, *FTOC* undoes the work of *CTOF*, and *CTOF* undoes the work of *FTOC*. This may be stated as follows:

FTOC CTOF X yields *X* for any *X*.
CTOF FTOC X yields *X* for any *X*.

Examples of the foregoing for particular values of *X* appear in Chapter 10.

Inverse functions are very important. The reason is that whenever one needs to use a certain function, the need for the inverse almost invariably arises. Suppose, for example, that *F* is a function which yields the amount of heat produced by an electric heater as a function of the voltage applied to it. Then for any given voltage *V* one can determine the heat produced by using the expression *F V*. However, if one wants to produce a specified amount of heat *H*, it will be necessary to determine what voltage will produce it. This requires the use of the function inverse to *F* which will yield the voltage as a function of the heat. If this inverse function is called *G*, then the necessary voltage is given by *G H*. Moreover:

G F X yields *X* for any *X*.
F G X yields *X* for any *X*.

It is therefore important to investigate methods for determining the inverse of any given function *F*. If *F* is represented by a function table, then the inverse function is represented by the same table, but with the argument and function columns interchanged. For example, Table 10.1 (reproduced in the left side of Figure 11.1) represents the function *CTOF* for a certain set of arguments. To apply the function *CTOF* to the argument 3, one locates the value 3 in the first column of the table and then takes the second value in that row (that is, 37.4) as the result. To apply the inverse function *FTOC*, to the argument 41, one locates 41 in the second column and takes the first element in that row (that is, 5) as the result. In other words, the appropriate function table for the inverse function is obtained from the function table for the original function by interchanging the two columns as shown on the right of Figure 11.1.

C	F
0	32
1	33.8
2	35.6
3	37.4
4	39.2
5	41
6	42.8
7	44.6
8	46.4
9	48.2
10	50

F	C
32	0
33.8	1
35.6	2
37.4	3
39.2	4
41	5
42.8	6
44.6	7
46.4	8
48.2	9
50	10

A Pair of Inverse Functions

Figure 11.1

11.2. INVERSE OF THE FUNCTION A+B×X

If F is the function A+X, that is:

```
∇Z←F X
Z←A+X ∇
```

then the inverse function is given by X-A or, equivalently, by (-A)+X. Thus the inverse function G is defined as follows:

```
∇Z←G X
Z←(-A)+X ∇
```

It is easy to see that F and G are inverse, for G F X is equivalent to (-A)+A+X and since (-A)+A is zero, this is equivalent to 0+X, or simply X as required. Similarly, F G X is equivalent to A+(-A)+X which is equivalent to 0+X or X.

If H is the function B×X, the inverse function K is the function X÷B, or (÷B)×X. Thus:

```
∇Z←H X              ∇Z←K X
Z←B×X ∇             Z←(÷B)×X ∇
```


From the foregoing results for addition and multiplication, it should be clear that the inverse of the function A+B×X is the function (÷B)×(-A)+X. Thus if L and M are defined as follows:

```
∇A←L X              ∇Z←M X
Z←A+B×X ∇           Z←(÷B)×(-A)+X ∇
```

then:

```
L M X                    M L X
A+B×(÷B)×(-A)+X          (÷B)×(-A)+A+B×X
A+1×(-A)+X               (÷B)×0+B×X
A+(-A)+X                 (÷B)×B×X
0+X                      1×X
X                        X
```

⊞1-2

11.3. DIFFERENCE TABLES

These results will now be applied to extend the applicability of the difference table method of function analysis developed in Chapter 10. Recall that the method developed applies only to a set of arguments of the form 0, 1, 2, 3, etc. Thus the difference table for a function whose values are 4 ¯1 ¯2 1 8 19 would appear as follows if the argument column was added:

X	Y	D Y	D D Y
0	4	¯5	4
1	¯1	¯1	4
2	¯2	3	4
3	1	7	4
4	8	11	
5	19		

The function F represented by the table is obtained by using the first row of the difference table (that, is 4 ¯5 4) divided by the vector 1 1 2 to obtain the coefficients 4 ¯5 2 for the following expression: 4+(¯5×X)+2×X×(X-1). Therefore, the required function F is defined as follows:

```
∇Z←F X
Z←4+(¯5×X)+2×X×(X-1) ∇
```

Evaluation of the expression F 0,⍳5 serves as a check as follows:

```
      F 0 1 2 3 4 5
4 ¯1 ¯2 1 8 19
```

Suppose now that the desired arguments were the equally spaced values `P←2.0 2.2 2.4 2.6 2.8 3.0`. The following table shows these arguments appended to the difference table as a leftmost column:

P	X	Y	D Y	D D Y
2	0	4	¯5	4
2.2	1	¯1	¯1	4
2.4	2	¯2	3	4
2.6	3	1	7	4
2.8	4	8	11	
3	5	19		

Suppose that one were able to determine a function *G* which yields the column *X* as a function of *P*, that is:

```
      G 2 2.2 2.4 2.6 2.8 3
0 1 2 3 4 5
```

Then *F G P* would yield *Y*; that is:

```
      F G 2 2.2 2.4 2.6 2.8 3
4 ¯1 ¯2 1 8 19
```

In other words, the function *H* defined as follows is the required function:

```
      ∇ Z←H X
      Z←F G X ∇
```

It remains to determine the function *G* which yields the column *X* as a function of the column *P*. Since *X* is of the form `0 1 2 3 4 5`, it is easy to determine *P* as a function of *X*, that is, to determine the function <u>inverse</u> to *G*. This is done by forming the difference table for *P* as follows:

X	P	D P
0	2	.2
1	2.2	.2
2	2.4	.2
3	2.6	.2
4	2.8	.2
5	3	

The coefficients `2 .2` in the first row yield the expression `2+.2×X` for the function inverse to *G*. This is of the form `A+B×X` and its inverse (that is, *G*) is therefore `(÷B)×(-A)+X`. Hence *G* is defined as follows:

```
      ∇ Z←G X
      Z←5×¯2+X ∇
```

Finally:

```
      G 2 2.2 2.4 2.6 2.8.3
0 1 2 3 4 5

      F G 2 2.2 2.4 2.6 2.8 3
4 ¯1 ¯2 1 8 19

      H 2 2.2 2.4 2.6 2.8 3
4 ¯1 ¯2 1 8 19
```

Instead of defining and using the separate functions *F* and *G*, their effect could be combined in a single (but cumbersome) expression by substituting for each occurrence of *X* in the expression for *F*, the expression occurring in the function *G*. Thus, for each *X* in the expression

```
      4+(¯5×X)+2×X×(X-1)
```

substitute the expression

```
      5×¯2+X
```

to obtain the single expression

```
      4+(¯5×(5×¯2+X))+2×(5×¯2+X)×((5×¯2+X)-1)
```

⊞3-4

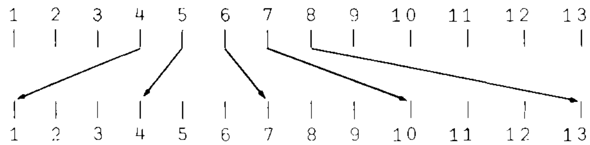
11.4. MAPS

In Chapter 10, it was shown how maps and graphs could be useful guides in the analysis of functions. They can also be useful guides in determining inverse functions.

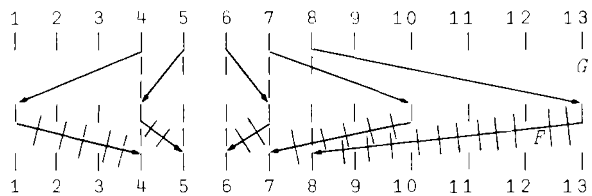
If *F* and *G* are each monadic functions, then we will write *F G* to denote the function defined by applying *F* to the result of *G*. That is, the function *F G* applied to *X* yields *F G X*. If *F* and *G* are inverses, then *F G* must be the <u>identity</u> function, that is, the function which applied to any argument *X* yields *X*.

Consider a function G represented by the following function table and the corresponding map:

X	Y
4	1
5	4
6	7
7	10
8	13



A map of the identity function clearly consists of a set of vertical arrows. Therefore, if the identity function is represented by broken line arrows and superimposed on the preceding map, the picture appears as follows:



The function F represented by the crossed lines is clearly the inverse of G, since the application of F to the results of G produces the equivalent of the identity function.

5

11.5. GRAPHS

In a graph, the values of the argument X are represented by distances measured along a horizontal line, and the values of the function values Y are represented by distances measured along a vertical line. Since an inverse function is obtained by exchanging the roles of argument and result in the original function, the graph of the inverse is obtained from the graph of the original function by interchanging the horizontal and vertical lines in the graph.

This interchange is easily visualized as follows:

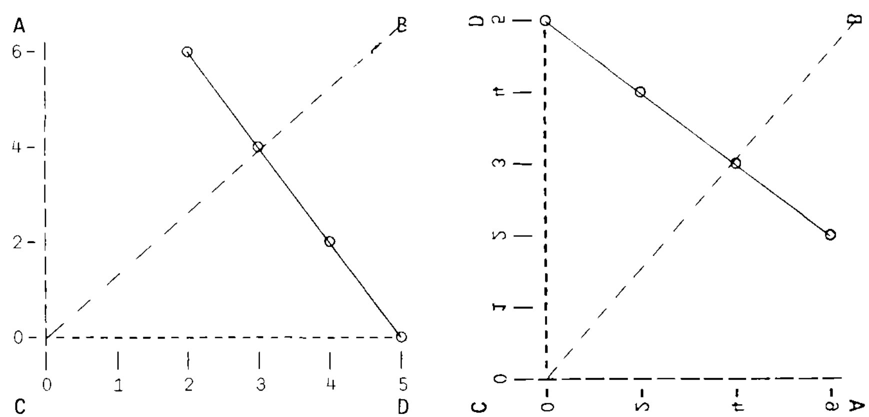
(1) Draw the graph of the original function on translucent paper (which can be read through from the obverse side of the paper).

(2) Label the top two corners of the paper with A and B, and the bottom two corners with C and D (both pairs in order from left to right).

(3) Grasp the paper by corners B and C and flip it over without changing the positions of the two corners held.

The result is a graph of the inverse function.

For example, the left side of Figure 11.2 shows a function table and the corresponding graph. The right side shows the table for the inverse function together with the graph obtained by the process just described. The broken line midway between the X-axis and the Y-axis shows the line through the points B C about which the paper is flipped. It is the one line in the graph whose position remains unchanged.



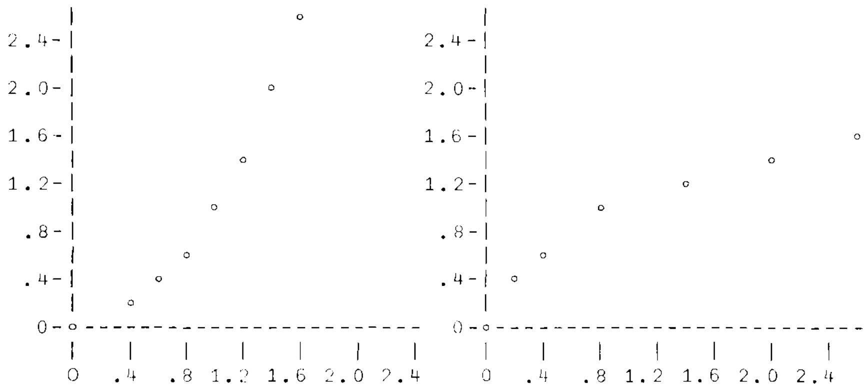
X	Y
2	6
3	4
4	2
5	0

X	Y
6	2
4	3
2	4
0	5

Graphs of a Pair of Inverse Functions

Figure 11.2

The graph of an inverse function can, of course, be obtained without using translucent paper, by simply plotting it from the table for the inverse function. One advantage of this is that the scales (the numbers along the horizontal and vertical axes) do not appear lying on their sides and printed backwards as in Figure 11.2. Figure 11.3 shows a pair of functions (the square function *X*⋆2 and its inverse) in which the graph of the inverse has been drawn in this manner.



X	*Y*
0	0
.2	.04
.4	.16
.6	.36
.8	.64
1.0	1.00
1.2	1.44
1.4	1.96
1.6	2.56

X	*Y*
0	0
.04	.2
.16	.4
.36	.6
.64	.8
1.00	1.0
1.44	1.2
1.96	1.44
2.56	1.6

Inverse Graph by Reflection

Figure 11.3

The function inverse to the square function is called the square root function. It was treated briefly in Section 6.6 where it was shown that the square root of *X* is equivalent to *X*⋆.5.

11.6. DETERMINING THE INVERSE FOR A SPECIFIC ARGUMENT

For any function whose graph is a straight line, it is easy to find an expression for the function since it is only necessary to determine the values of the constants *A* and *B* in the expression *A*+*B*×*X*. It is equally easy to obtain the expression for the inverse function since this is given by (÷*B*)×(-*A*)+*X*. For example, the function graphed on the left of Figure 11.1 is given by the expression 10+¯2×*X* and the inverse on the right is given by ¯.5×¯10+*X*.

For a function whose graph is not a straight line, it may be impossible to obtain an expression for the inverse function. However, it is possible to determine the inverse function in the following sense: for any given argument in the domain of the inverse function it is possible to determine the corresponding value of the result of the inverse function.

For example, in the case of the square function (*X*⋆2) graphed on the left of Figure 11.3 we have no expression for the inverse function, the square root, graphed on the right. However, for any particular argument it is possible to find the result approximately from the graph of the inverse; for example, if the argument is 2, the result of the inverse function is clearly slightly greater than 1.4. Moreover, one can achieve the same without the graph of the inverse, by working directly from the graph of the original function. Thus one locates the argument 2 on the <u>vertical</u> axis and determines the approximate corresponding result on the horizontal axis.

Finally, one can work directly from the expression for the original function without even graphing it. For example, the expression for the function on the left of Figure 11.2 is *X*⋆2. To obtain the value of the inverse function applied to the argument 2, one must determine a value of *X* such that *X*⋆2 is equal to 2. If one determines a value *C* such that *C*⋆2 is less than 2 and another value *D* such that *D*⋆2 is greater than 2, then the required value of the square root of 2 must lie between *C* and *D*.

Thus, if *C* is 1.4 and *D* is 1.42, then *C*⋆2 is 1.96 and *D*⋆2 is 2.0164 and the required value lies between 1.4 and 1.42. The point midway between them is (1.4+1.42)÷2, that is 1.41. Since 1.41⋆2 is equal to 1.9881, the required value is greater than 1.41. Since it is already known to be less than 1.42, we now choose the value midway between 1.41 and 1.42, that is, 1.415. The value of 1.415⋆2 is 2.012225 which is very near to 2. Hence 1.415 is a very good approximation to the value of the square root function applied to the argument 2. Moreover, the same process could be continued to determine better and better approximations as long as desired.

Although we have not obtained an expression for the square root function, we have devised a process which determines the value of the square root when applied to the particular argument 2. Moreover, the process could be applied for any argument other than 2 which lies in the domain of the square root. Finally, the process uses only the expression for the original square function.

The procedure used to determine the square root had to be repeated or *iterated* a number of times to obtain a sufficiently good approximation to the desired result. Such a process is called *iterative*. Functions which are defined by iterative procedures will be discussed more fully in the succeeding chapter.

⊞10-11

11.7. THE SOLUTION OF EQUATIONS

If *G* is the function inverse to *F*, and one wishes to obtain the value of *G N*, then the required value *Y* must be such that *F Y* is equal to *N*. In other words, the following expression must be true (that is, have the value 1):

```
N=F Y
```

Such an expression which is required to be true is called an *equation*, and a value of *Y* which makes it true is called a *solution* or *root* of the equation.

The problem of determining the value of the inverse function *G* applied to the argument *N* is therefore equivalent to finding a solution to the equation *N*=*F Y*. It is for this reason that the solution of equations is a very important topic in the study of algebra. For example, finding the square root of 2 is equivalent to solving the equation 2=*X*⋆2, and finding the square root of 10 is equivalent to solving the equation 10=*X*⋆2.

The origin of the term "square root" for the function inverse to the square function should now be clear; the square root of the argument *N* is the solution or *root* of the equation *N*=*X*⋆2 in which the square function occurs to the right of the equal sign.

⊞12-13

Chapter 12

ITERATIVE PROCESSES

12.1. INTRODUCTION

The iterative process used in Section 11.6 for finding the square root of 2 is only one of many possible iterative processes for achieving the same end. The following procedure is, in fact, more effective than the procedure of Chapter 11 in the sense that it closes in on the desired value in fewer iterations.

Suppose that *S* is the square root of a given number *X*, that *Z* is any other number, and that *Y* is equal to *X*÷*Z*. Then *Z*×*Y* is equal to *X*, and *S*×*S* is also equal to *X*. Hence if *Z* is less than *S*, then *Y* must be greater than *S*, and if *Z* is greater than *S*, then *Y* must be less. In any case, the correct square root *S* must lie between *Z* and *Y*. Consequently, the point midway between *Z* and *Y* (that is, .5×*Z*+*Y*) should furnish a good new approximation to the square root *S*. Since *Y* is equal to *X*÷*Z*, this expression can be written simply as .5×*Z*+*X*÷*Z*.

Suppose, for example, that we wish to find the square root of 3, that is, *X* has the value 3. If we choose a value of 1 for *Z*, then the next approximation is given as follows:

```
      X←3
      Z←1
      .5×Z+X÷Z
2
```

Using the new approximation 2 for *Z* yields the next approximation:

```
      Z←2
      .5×Z+X÷Z
1.75
```

Again:

```
      Z←1.75
      .5×Z+X÷Z
1.732142857

      Z←1.732142857
      .5×Z+X÷Z
1.73205081
```

Squaring this last result yields:

```
      1.73205081*2
3.000000008
```

showing that it is a good approximation to the square root of 3.

The foregoing procedure can be made clearer by simply assigning the value of the new approximation to the name Z each time as follows:

```
      X←3
      Z←1
      Z←.5×Z+X÷Z
      Z
2

      Z←.5×Z+X÷Z
      Z
1.75

      Z←.5×Z+X÷Z
      Z
1.732142857

      Z←.5×Z+X÷Z
      Z
1.73205081
```

From this it is clear that the iteration consists of repeating the execution of the expression Z←.5×Z+X÷Z enough times, the line containing only the expression Z being inserted solely to allow us to see the successive values of the approximation Z.

Such iteration can be specified in a function definition as follows:

```
      ∇Z←SQRT X
[1]   Z←1
[2]   Z←.5×Z+X÷Z
[3]   →2∇
```


The right-pointing arrow on line 3 of the function definition is called a branch; the only effect of the expression →2 is to cause statement number 2 to be executed next. Hence statements 2 and 3 are executed again and again in sequence. This behavior can be seen from a trace of the function as follows:

```
      TΔSQRT←1 2 3
      P←SQRT 3
SQRT[1] 1
SQRT[2] 2
SQRT[3] 2
SQRT[2] 1.75
SQRT[3] 2
SQRT[2] 1.732142857
SQRT[3] 2
SQRT[2] 1.73205081
```

The trouble with the function SQRT is that it never terminates. It would be desirable to make it terminate when a certain condition becomes satisfied, say when the magnitude of the difference between Z*2 and the argument X becomes less than .00001. This is achieved in the function SQT defined as follows:

```
      ∇Z←SQT X
[1]   Z←1
[2]   Z←.5×Z+X÷Z
[3]   →2×.00001<|X-Z*2∇
```

As long as X and Z*2 differ by .00001 or more, the expression following the branch arrow is equal to 2×1 and statement 2 is executed next. When Z*2 becomes close enough to X, the expression has the value 2×0, (that is, 0), indicating that statement 0 should be executed next. Since there is no statement 0, the process terminates.

The function SQT can now be applied to any non-negative argument. For example:

```
      SQT 2
1.4142156862745

      (SQT 2)*2
2.0000060073049

      SQT 10
3.1622776651757

      (SQT 10)*2
10.000000031668
```

The detailed behavior of the function *SQT* can be seen in a trace as follows:

```
      TΔSQT←1 2 3
      P←SQT 10
SQT[1] 1
SQT[2] 5.5
SQT[3] 2
SQT[2] 3.65909090909
SQT[3] 2
SQT[2] 3.1960050818746
SQT[3] 2
SQT[2] 3.1622776651757
SQT[3] 0
      P
3.1622776651757
```

Iteration is of great importance in mathematics and its uses are by no means limited to root-finding. The remaining sections of this chapter illustrate a few of its uses. Others occur in later chapters.

12.2. GENERAL ROOT FINDER

The iterative method used in Section 11.6 to determine the square root of 2 can now be expressed as a formal function definition by using branching. The method consists of using two quantities *C* and *D* which bound the desired value in the following sense: *C*⋆2 is less than 2 and *D*⋆2 is greater than 2, and the desired value therefore lies between *C* and *D*. The method procedes by computing the point *Z* midway between *C* and *D* and then computing *Z*⋆2 to see whether it lies above or below 2. If it lies below 2, then *C* is respecified by *Z* (that is, *C*←*Z*) and the process is repeated; otherwise *D* is respecified by *Z* and the process is repeated.

It will be more convenient to combine the bounding quantities *C* and *D* in a single vector *B* so that *Z* respecifies either *B*[1] or *B*[2]. The complete definition follows:

```
      ∇Z←Q X
[1]   B←1.4 1.42
[2]   A←.5×+/B
[3]   I←1+X<Z⋆2
[4]   B[I]←Z
[5]   →2×.00001<|X-Z⋆2∇
```


The behavior of the function is illustrated by the following trace:

```
      TΔQ←⍳5
      P←Q 2
Q[1]  1.4 1.42
Q[2]  1.41
Q[3]  1
Q[4]  1.41
Q[5]  2
Q[2]  1.415
Q[3]  2
Q[4]  1.415
Q[5]  2
Q[2]  1.4125
Q[3]  1
Q[3]  1
Q[4]  1.4125
Q[5]  2
Q[2]  1.41375
Q[3]  1
Q[4]  1.41375
Q[5]  2
Q[2]  1.414375
Q[3]  2
Q[4]  1.414375
Q[5]  2
Q[2]  1.4140625
Q[3]  1
Q[4]  1.4140625
Q[5]  2
Q[2]  1.41421875
Q[3]  2
Q[4]  1.41421875
Q[5]  2
Q[2]  1.414140625
Q[3]  1
Q[4]  1.414140625
Q[5]  2
Q[2]  1.4141796875
Q[3]  1
Q[4]  1.1414796875
Q[4]  1.4141796875
Q[5]  2
Q[2]  1.41419921875
Q[3]  1
Q[4]  1.41419921875
Q[5]  2
Q[2]  1.414208984375
Q[3]  1
Q[4]  1.414208984375
Q[5]  2
Q[2]  1.4142138671875
Q[3]  2
Q[4]  1.4142138671875
Q[5]  0
      P
1.4142138671875
```

The foregoing function will determine a root of the equation *X*=*Z*⋆2, that is, for a given value of *X* it will determine a value of *Z* such that the equation is true. In order to obtain a general root finder which would solve the equation *X*=*F Z* for any desired function *F*, it is necessary to replace every occurrence of the expression *Z*⋆2 in the function *Q* by the expression *F Z*.

It will also be convenient to have the bounding vector *B* as an argument of the function so that one can specify suitable initial bounding values. The general root-finder is therefore defined as follows:

```
      ∇Z←B GRF X
[1]   Z←.5×+/B
[2]   B[1+X<F Z]←Z
[3]   →.00001<|X-F Z∇
```

Suppose, for example, that F is the cube function defined as follows:

```
      ∇Z←F X
[1]    Z←X*3∇
```

Then, since 4*3 is less than 100 and 5*3 is greater than 100, the expression 4 5 *GRF* 100 yields a solution of the equation 100=Z*3 as follows:

```
      4 5 GRF 100
4.6415887878967

      (4 5 GRF 100)*3
99.999990581929
```

⎕3

There are two reasons for including the bounding values *B* as an argument of the general root finder function *GRF*. The first is that for some functions *F* it is very difficult to compute suitable initial bounding values and it may be necessary to provide them, possibly from information obtained from a rough graph. The second reason is that for some functions *F* the equation *X*=*F Z* has more than one solution, and the initial bounding values permit us to isolate any one of the several roots as desired.

For example, suppose that *F* is defined as follows:

```
      ∇ Z←F X
        Z←¯76.44+(102.2×X)+(¯41×X*2)+(5×X*3) ∇
```

Then several different values of *X* can be determined for which *F X* is zero:

```
      1 2 GRF 0
1.4

      3 2 GRF 0
2.6

      4 5 GRF 0
4.2
```


It can be verified that this function is equivalent to the function 5×(X-1.4)×(X-2.6)×(X-4.2) whose graph appears in Figure 10.5. This graph will therefore be helpful in appreciating how the different bounding values lead to different roots. Two further solutions appear below:

```
      1 2 GRF 3
1.65639

      3 2 GRF 3
2.23409
```

⎕4-6

12.3. GREATEST COMMON DIVISOR

The integer 7 is a divisor of 42 and a divisor of 63 and is therefore said to be a common divisor of the pair of integers 42 and 63. The largest integer which is a common divisor of a pair of integers is said to be their greatest common divisor. Thus 7 is a common divisor of the pair 42 63 but is not their greatest common divisor since 21 is also a common divisor and is greater than 7.

An interesting and efficient method for finding the greatest common divisor of a pair of integers *X* and *Y* is based on the following fact: If *Z* is the remainder obtained on dividing *X* into *Y* (that is, *Z*←*X*|*Y*), then the greatest common divisor of *X* and *Y* is also the greatest common divisor of *X* and *Z*. For example, if *X* is 48 and *Y* is 66, then *Z* is 18 and the greatest common divisor of 48 and 66 is the same as the greatest common divisor of 18 and 48. The process can now be repeated since the greatest common divisor of 18 and 48 is the greatest common divisor of 18 and their remainder, which is 12. Thus we look for the greatest common divisor of 12 and 18. The remainder 12|18 is 6 and we now look at the pair 6 and 12. The remainder 6|12 is zero. This indicates that 6 is a divisor of 12 and therefore 6 is the greatest common divisor of 6 and 12. Hence, 6 is also the greatest common divisor of the original pair 48 and 66.

The foregoing is an iterative process which can obviously be defined as follows:

```
      ∇Z←X GD Y
[1]    Z←X
[2]    X←X|Y
[3]    Y←Z
[4]    →X≠0∇
```

The behavior of the function *GD* can be seen from the following trace:

```
      TΔGD←⍳4
      P←48 GD 66
GD[1] 48
GD[2] 18
GD[3] 58
GD[4] 1
GD[1] 18
GD[2] 12
GD[3] 18
GD[4] 1
GD[1] 12
GD[2] 6
GD[3] 12
GD[4] 1
GD[1] 6
GD[2] 0
GD[3] 6
GD[4] 0
      P
6
```

The greatest common divisor function can also be defined in terms of a single argument (which is expected to be a two-element vector) as follows:

```
    ∇ Z←GCD X
[1]   Z←X[1]
[2]   X←(|/X),X[1]
[3]   →X[1]≠0 ∇
```

For example:

```
      TΔGCD←⍳3
      P←GCD 48 66
GCD[1] 48
GCD[2] 18 48
GCD[3] 1
GCD[1] 18
GCD[2] 12 18
GCD[3] 1
GCD[1] 12
GCD[2] 6 12
GCD[3] 1
GCD[1] 6
GCD[2] 0 6
GCD[3] 0
      P
6
```


The function *GCD* can be used in the treatment of rational numbers as follows. If *V* is any two-element vector of integers it can be used to represent the rational number ÷/*V*. Moreover, if *V* is multiplied by any scalar integer *S* it still represents the same rational number. For example:

```
      V←48 66
      ÷/V
0.727273

      3×V
144 198

      ÷/3×V
0.727273
```

Similarly, if *V* is divided by any integer which is a divisor of both elements, the result is a pair of integers which also represent the same rational number. For example:

```
      V÷2
24 33

      ÷/V÷2
0.727273
```

Moreover, if *V* is divided by the greatest common divisor of *V*[1] and *V*[2], one obtains the smallest pair of integers which represent the same rational. For example:

```
      V÷GCD V
8 11

      ÷/V÷GCD V
0.727273
```

⌹8-11

12.4. THE BINOMIAL COEFFICIENTS

Binomial coefficients are of importance in many areas of mathematics. In this section they will be introduced as a further example of the use of iteration in the function which defines them. They will be used and studied more thoroughly in later chapters in the treatment of polynomials.

The binomial coefficients of order *N* are the *N*+1 elements of the vector produced by the expression *BIN N* using the function *BIN* defined as follows:

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

The following examples illustrate the behavior of the function:

```
       BIN 0
1
       BIN 1
1 1
       BIN 2
1 2 1
       BIN 3
1 3 3 1
       BIN 4
1 4 6 4 1
       BIN 5
1 5 10 10 5 1
       BIN 6
1 6 15 20 15 6 1

       TΔBIN←ι4
       P←BIN 3
BIN[1] 1
BIN[2] 3
BIN[3] 1 1
BIN[4] 2
BIN[2] 3
BIN[3] 1 2 1
BIN[4] 2
BIN[3] 1 3 3 1
BIN[4] 2
BIN[2] 0
       P
1 3 3 1
```

⌹12-19

Chapter 13

INNER PRODUCTS AND POLYNOMIALS

13.1. INTRODUCTION

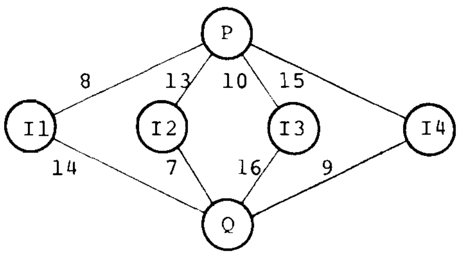
Each of the expressions +/*D*×*W* and ⌊/*A*+*B* and ⌈/*A*⌊*B* involve a dyadic function applied to the two arguments, followed by a reduction of this result by a second dyadic function applied over the result. These expressions are therefore said to be of the same form, although they do differ in the actual dyadic functions employed. Thus the first uses + and ×, the second uses ⌊ and +, and the third uses ⌈ and ⌊.

Expressions of this form are so important that they will be assigned a special notation known as inner product. Their importance is due largely to the fact that they arise very frequently in practical problems. Consider, for example, the following expressions:

```
       D←5 2 4
       W←36 12 1
       +/D×W
208

       A←8 13 10 15
       B←14 7 16 9
       ⌊/A+B
20
```

The expression +/*D*×*W* may arise from a practical problem as follows. Suppose that the elements of *D* express a certain distance in terms of yards, feet, and inches, that is, *D* represents the distance 5 yards, 2 feet, and 4 inches. One could express tha same distance in inches alone by multiplying the first element by 36, the second by 12, the third by 1, and then summing the results. In other words, if *W* is the weighting vector as specified above, then the distance in inches is given by the expression +/*D*×*W*.

The second expression ⌊/*A*+*B* may arise as follows. Suppose that one wishes to travel from station *P* to station *Q* and has a choice of four different routes, via the four different intermediate stations, *I*1, *I*2, *I*3, and *I*4 as shown in Figure 13.1. Suppose further that the distances from *P* to the four intermediate stations are given by the four elements of the vector *A*, and that the distances from the intermediate stations to the destination *Q* are given by the vector *B*. Then the expression *A*+*B* gives the total distances for each of the four possible routes, and ⌊/*A*+*B* gives the smallest of these distances, that is, the shortest distance possible by the available routes.



Minimum Distance

Figure 13.1

⊞1-2

13.2. THE INNER PRODUCT OF TWO VECTORS

If *X* and *Y* are vectors of the same dimension, then the expression *X*+.×*Y* is called the plus times inner product of *X* and *Y*, and is defined to be equivalent to the expression +/*X*×*Y*. Similarly, *X*⌊.+*Y* is called the minimum plus inner product and is defined as ⌊/*X*+*Y*, and so on for every pair of dyadic functions. For example:

```
      X←2 3 5 7 11
      Y←2 1 2 0 1
      X+.×Y              +/X×Y
28                 28
      X⌊.+Y              ⌊/X+Y
4                  4
      X×.*Y              ×/X*Y
3300               3300
      X+.-Y              +/X-Y
22                 22
      X+.≠Y              +/X≠Y
4                  4
      X⌈.=Y              ⌈/X=Y
1                  1
```

⊞3-5

13.3. MATRICES

What we have been calling a table is in mathematics more usually called a matrix; we will call it so from now on. We will also generalize the dyadic repetition function (introduced in Section 1.7 and denoted by ρ) so that it will permit the specification of a matrix with any shape and having any desired elements.

The dyadic repetition function ρ was defined only for scalar arguments, but it will now be defined for vector arguments as well. For example:

```
      3ρ5
5 5 5
      5ρ3
3 3 3 3 3
      3ρ1 2 3 4
1 2 3
      10ρ1 2 3 4
1 2 3 4 1 2 3 4 1 2
```

From these examples it is clear that the left argument determines the size of the result and that the elements of the result are chosen from the right argument, repeating them over and over if necessary.

If the left argument *A* is a two-element vector it again determines the size of the result, that is, the result is a matrix *M* such that ρ*M* (that is, the size of *M*) is equal to *A*. In other words, *M* has *A*[1] rows and *A*[2] columns. For example:

```
      2 3ρ1 2 3 4 5 6
1 2 3
4 5 6

      3 4ρι12
 1  2  3  4
 5  6  7  8
 9 10 11 12

      3 5ρ0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
```

⊞6-7

13.4. INNER PRODUCT WITH MATRIX ARGUMENTS

The inner product also applies to matrix arguments. For example:

```
      M←3 4⍴3 0 4 2 4 6 5 1 0 5 2 4
      N←4 5⍴6 7 2 1 7 5 6 5 0 5 7 2 3 6 3 1 2 2 1 3
      M
3  0  4  2
4  6  5  1
0  5  2  4
      N
6  7  2  1  7
5  6  5  0  5
7  2  3  6  3
1  2  2  1  3

      M+.×N                      M⌊.+N
48  33  22  29  39 3  4  4  0  5
90  76  55  35  76 2  3  3  2  4
43  42  39  16 43  5  4  2  1  5

      M+.=N                      M+.≠N
0  1  1  1  0       4  3  3  3  4
1  1  0  1  0       3  3  4  3  4
1  1  1  0  1       3  3  3  4  3

      (M+.=N)+(M+.≠N)
4  4  4  4  4
4  4  4  4  4
4  4  4  4  4
```

The result of an inner product applied to matrices *M* and *N* is a matrix having the same number of rows as the first argument and as many columns as the second argument. The elements of the results are the results obtained by applying the inner product to each <u>row</u> vector of the first argument paired with each <u>column</u> vector of the second argument. More specifically, if *R←M+.×N*, then the element *R[I;J]* is given by the expression *M[I;]+.×N[;J]*. For example:


```
      R←M+.×N
      R
48  33  22  29  39
90  76  55  35  76
43  42  39  16  43

      R[2;3]
55
      M[2;]
4  6  5  1
      N[;3]
2  5  3  2
      M[2;]+.×N[;3]
55

      (M⌊.+N)[3;5]
5
      M[3;]⌊.+N[;5]
5
```

⊞8-10

If *X* is a vector and *M* is a matrix, then the inner product *M+.×X* is defined by simply treating *X* much like a 1-column matrix. For example:

```
      X←0 3 2 4
      M+.×X
16  32  35
      M⌊.+X
3  4  0
      M+.≠X
4  4  1
      M+.=X
0  0  3
```

If *Y* is a vector and *M* is a matrix, then the inner product *Y+.×M* is defined by treating *Y* much like a 1-row matrix. For example:

```
      Y←0 4 2
      Y+.×M
16  34 24  12
      Y⌊.+M
2  0  4  2
      Y+.≠M
2  2  2  3
```

⊞11-18

13.5. POLYNOMIALS

If *C* is a vector and *X* is a scalar, then an expression of the form +/C×X*¯1+⍳ρC is a function of *X* which is called a <u>polynomial of degree</u> ¯1+ρC. For example, if C←2 5 ¯3 1, then +/C×X*¯1+⍳ρC is a polynomial of degree 3 and is equivalent to the expression +/2 5 ¯3 1×X*0 1 2 3. This expression is clearly equal to the sum of the following quantities:

```
 2×X*0
 5×X*1
¯3×X*2
 1×X*3
```

Each of these quantities is called a <u>term</u> of the polynomial; each of the constant multipliers is called a <u>coefficient</u>.

Figure 13.2 shows a graph of each of the terms of the polynomial +/2 5 ¯3 1×X*0 1 2 3, together with a graph of their sum, that is, of the polynomial itself.

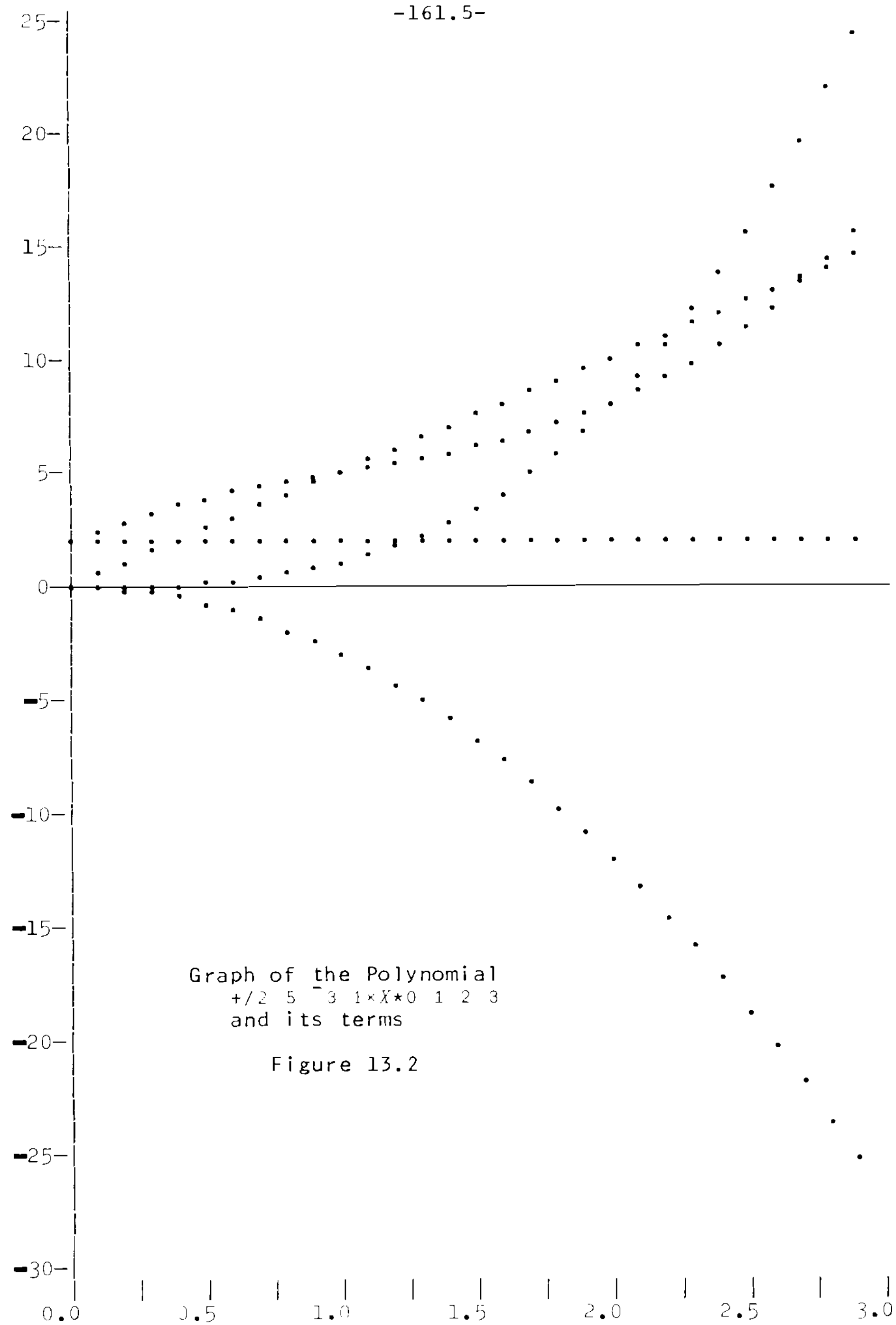
Since a polynomial may have any number of terms and since each of the coefficients may have any value, these graphs suggest (correctly) that coefficients can be chosen so as to make a polynomial which approximates any function of practical interest. This ability to approximate a wide variety of functions is one of the main reasons for the overwhelming importance of polynomials. A second reason is the ease of evaluation, which involves only addition, multiplication, and powers. A third reason is the ease with which polynomial functions can be analyzed.

⎕19-21

13.6. POLYNOMIALS EXPRESSED AS INNER PRODUCTS

Since P×Q is equivalent to Q×P, the expression +/C×(X*¯1+⍳ρC) for a polynomial can be written equivalently as +/(X*¯1+⍳ρC)×C. Moreover, since +/Q×P can be written in the inner product form as Q+.×P, the polynomial can be written as the inner product (X*¯1+⍳ρC)+.×C.

It should be clear that none of these equivalent expressions for a polynomial apply correctly to a <u>vector</u>



Graph of the Polynomial
+/2 5 ¯3 1×X*0 1 2 3
and its terms

Figure 13.2

argument X in order to evaluate the polynomial applied separately to each element of X. For example:

```
      C←1 2 1
      X←3
      +/C×X*¯1+⍳⍴C
16
      X←4
      +/C×X*¯1+⍳⍴C
25
      X←5
      +/C×X*¯1+⍳⍴C
36
      X←3 4 5
      +/C×X*¯1+⍳⍴C
34
      X←3 4
      +/C×X*¯1+⍳⍴C
```

(cannot be evaluated because the vectors X and -1+⍳⍴C are not of the same size)

To obtain the correct result of 16 25 36 when applying the polynomial with coefficients C←1 2 1 to the vector argument 3 4 5, it requires a different expression for the polynomial. This can be obtained by a slight modification of the inner product expression (X*¯1+⍳⍴C)+.×C, namely, (X∘.*¯1+⍳⍴C)+.×C. For example:

```
      C←1 2 1
      X←3 4 5

      X∘.*¯1+⍳⍴C
1   3    9
1   4   16
1   5   25

      (X∘.*¯1+⍳⍴C)+.×C
16  25  36
```

The following definition will therefore be adopted for the polymonial function:

```
      ∇Z←C POL X
      A←(X∘.*¯1+⍳⍴C)+.×C∇
```

The following examples illustrate its use:

```
      1 2 1 POL 3 4 5 6
16  25  36  49

      1 3 3 1 POL 3 4 5 6
64  125  216  343
```


THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 14

IDENTITIES

14.1. INTRODUCTION

Two expressions are said to be equivalent if they represent the same function, that is, if they both yield the same value for any specified argument (lying within their domains). For example, X×Y and Y×X are equivalent, as are X⌈Y and Y⌈X, but X-Y and Y-X are not equivalent.

If two equivalent expressions are joined by an equal sign, the resulting single expression is true (i.e., has the value 1) for every possible value of the argument or arguments. It is therefore called an identity. For example, the expression (X×Y)=(Y×X) is always true, as are (X⌈Y)=(Y⌈X) and (X⌊(Y⌊Z))=((X⌊Y)⌊Z).

For convenience in discussion, many of the more useful identities are given names. For example, the identity (X×Y)=(Y×X) is said to express the commutativity of times, and (X⌊(Y⌊Z))=((X⌊Y)⌊Z) expresses the associativity of minimum. The following list shows (together with their names) a number of identities which the reader should either find already familiar, or be able to verify by evaluating them for a few sample values of the arguments:

Identity	Name
(X+Y)=(Y+X)	Commutativity of plus
((X⌈Y)⌈Z)=(X⌈(Y⌈X))	Associativity of maximum
(X×(Y+Z))=((X×Y)+(X×Z))	Distributivity of times over plus
(X⌈(Y⌊Z))=((X⌈Y)⌊(X⌈Z))	Distributivity of maximum over minimum
(X⌈Y)=(-(-X)⌊(-Y)) (X⌊Y)=(-(-X)⌈(-Y))	Duality of maximum and minimum
(X∨Y)=(~(~X)∧(~Y)) (X∧Y)=(~(~X)∨(~Y))	Duality of and and or

Identities are very useful in mathematics, primarily because they allow one to easily express the same function in a variety of ways, each of the different ways possessing some particular advantage such as being easy to evaluate, or providing some particular insight into the behavior of the function. Consider, for example, the function +/(⍳X)*2

which yields the sum of the squares of the integers up to and including X. The difference table for this function appears as follows:

X	+/(⍳X)*2	D +/(⍳X)*2	D D +/(⍳X)*2	D D D+/(⍳X)*2
0	0	1	3	2
1	1	4	5	2
2	5	9	7	2
3	14	16	9	2
4	30	25	11	
5	55	36		
6	91			

According to the method of analyzing a function by difference tables developed in Chapter 10, the first row of the difference table (that is, 0 1 3 2) can be divided by !0 1 2 3 (that is, 0 1 2 6) to obtain the coefficients 0, 1, 3÷2, and 2÷6 used in the following expression:

```
0+X+((3÷2)×X×(X-1))+(2÷6)×X×(X-1)×X-2
```

The expression is equivalent to +/(⍳X)*2. Moreover, for large values of X it is much easier to evaluate than +/(⍳X)*2. For example, the sum of the squares up to 100 is given by:

```
0+100+((3÷2)×100×99)+(2÷6)×100×99×98
0+100+14850+323400
338350
```

Moreover, by methods to be developed in this chapter, the expression 0+X+((3÷2)×X×X-1)+(2÷6)×X×(X-1)×X-2 can be shown to be equivalent to the polynomial:

```
(÷6)×(X*0 1 2 3)+.×0 1 3 2
```

This can be evaluated even more easily. For example:

```
X←100
(÷6)×(X*0 1 2 3)+.×0 1 3 2
(÷6)×1 100 10000 1000000+.×0 1 3 2
(÷6)×0+100+30000+2000000
(÷6)×2030100
338350
```

14.2. COMMUTATIVITY

Since X+Y yields the same result as Y+X, the function + is said to commute, or to be commutative. The word commute implies that the two arguments can be commuted (i.e., interchanged) without changing the result. The function × is also commutative; that is, (X×Y)=(Y×X). To see why this is so, consider the way in which multiplication is defined as repeated addition, that is, 3×4 can be considered as the addition of three groups of objects each containing four items.

This can be pictured in terms of the array

```
      3 4 ρ '□'
□□□□
□□□□
□□□□
```

which consists of three rows, each containing four boxes. The total number of boxes is then 3×4. It is clear that the array

```
      ⍉3 4 ρ '□'
□□□
□□□
□□□
□□□
```

contains the same number of boxes. It is equally clear that this is the same array as

```
      4 3 ρ '□'
□□□
□□□
□□□
□□□
```

which represents the product 4×3. Hence, (3×4)=(4×3).

The functions maximum and minimum are both commutative, that is,

```
      (X⌈Y)=(Y⌈X)
```

and

```
      (X⌊Y)=(Y⌊X)
```

It is equally clear that equality is commutative, that is,

```
      (X=Y)=(Y=X).
```

To show that a function is not commutative, it is sufficient to exhibit one pair of arguments for which it does not commute. For example, 4-3 yields 1 and 3-4 yields ¯1. Since these results differ, it is clear that subtraction is not commutative. Similarly 3≤4 yields 1 and 4≤3 yields 0 and the function ≤ therefore does not commute.

The results thus far can be summarized in a table as follows:

+	-	×	⌈	⌊	≤	=
1	0	1	1	1	0	1

A zero lying below a function symbol indicates that the function is not commutative, and a 1 indicates that it is.

⌹1

The 1's and 0's in the foregoing table can be thought of as the results of a function *COM* which determines the commutativity of its argument, that is, *COM* '+' yields 1, and *COM* '-' yilelds 0, and so on. This function could be defined as follows:

```
      ∇Z←COM X
      Z←(X='+-×⌈⌊≤=')/1 0 1 1 1 0 1
```

For example, in the evaluation of the expression *COM* '⌈', the argument *X* has the value '⌈', and the expression X='+-×⌈⌊≤=' therefore has the value 0 0 0 1 0 0 0. Consequently, (X='+-×⌈⌊≤=')/1 0 1 1 1 0 1 yields 1, indicating that the function maximum is commutative.

⌹2

Function Tables. Consider the subtraction table *S* and its transpose T←⍉S shown in Figure 14.1. The circled element in *S* is the result of the subtraction 5-3. The corresponding element of *T* (enclosed in a square) is clearly the result of 3-5. More generally, if one uses table *S* to evaluate any subtraction X-Y, then the corresponding element of table *T* is the result of the commuted expression Y-X. Consequently, a function is commutative only if its function table *A* agrees with its transpose ⍉A.

S

0	¯1	¯2	¯3	¯4	¯5	¯6
1	0	¯1	¯2	¯3	¯4	¯5
2	1	0	¯1	¯2	¯3	¯4
3	2	1	0	¯1	¯2	¯3
4	3	2	1	0	¯1	¯2
5	4	3	2	1	0	¯1
6	5	4	3	2	1	0

S←(⍳7)∘.-⍳7

T

0	1	2	3	4	5	6
¯1	0	1	2	3	4	5
¯2	¯1	0	1	2	3	4
¯3	¯2	¯1	0	1	2	3
¯4	¯3	¯2	¯1	0	1	2
¯5	¯4	¯3	¯2	¯1	0	1
¯6	¯5	¯4	¯3	¯2	¯1	0

T←⍉S

Function Tables for Subtraction

Figure 14.1

Most functions of interest are defined on a limitless domain (e.g., all numbers) and any function table therefore represents only a part of the domain. Consequently, the fact that a function table agrees with its transpose does not prove that the function is commutative, since an enlarged table might show that it is not. However, some important functions are defined for a limited domain (i.e., for only a small number of argument values), and for such a function it is possible to make a complete function table and determine the properties of the function directly from the table.

We will illustrate this by defining four important logical functions, i.e., functions whose domains are limited to logical values 0 and 1. They are called and, or, not-and, and not-or, and are denoted by $\wedge$, $\vee$, ⍲, and ⍱, respectively. They are completely defined by the function tables of Figure 14.2. These tables are all symmetric (i.e., agree with their transposes), and these functions are therefore all commutative.

∧	0	1
0	0	0
1	0	1

and

∨	0	1
0	0	1
1	1	1

or

⍲	0	1
0	1	1
1	1	0

not-and

⍱	0	1
0	1	0
1	0	0

not-or

Function Tables for Logical Functions

Figure 14.2

⊞3

The Method of Exhaustion. The process of examining all possible cases to determine some property of a function (used above on the logical functions) is called the method of exhaustion. It can often be applied even if the number of possible values of the arguments is unlimited. For example, the arguments of the function $\leq$ can take on an unlimited number of values, but it is only necessary to consider three cases: if the arguments are arranged in ascending order according to value, then the order is either

X Y, in which case the result of the function $X \leq Y$ is 1, or the order is Y X in which case the result of $X \leq Y$ is 0 or the two are equal, in which case the result is 1. This may be summarized in a table as follows:

Case	$X \leq Y$
X Y	1
Y X	0
$Y=X$	1

Moreover, if a column for the expression $Y \leq X$ is added, the table appears as shown in Table 14.3. This table shows that the function $\leq$ is not commutative.

Case	$X \leq Y$	$Y \leq X$
X Y	1	0
Y X	0	1
$Y=X$	1	1

Non-Commutativity of $\leq$

Table14.3

The same scheme of exhaustion can be used to determine the commutativity of the other relations $<$ $=$ $\geq$ $>$ and $\neq$, and of the functions ⌈ and ⌊. For example, Table 14.2 shows that maximum is commutative.

Case	$X⌈Y$	$Y⌈X$
X Y	Y	Y
Y X	X	X
$Y=X$	X	X

Commutativity of ⌈

Table 14.4

⊞4

14.3. ASSOCIATIVITY

Since *X*+(*Y*+*Z*) yields the same result as (*X*+*Y*)+*Z*, the function + is said to be associative. Multiplication is also associative, that is,

(*X*×(*Y*×*Z*))=((*X*×*Y*)×*Z*)

It is easy to show that subtraction and division are not associative. For example, 4-(3-2) yields 3 and (4-3)-2 yields ¯1.

The associativity of the maximum function can be established by examining all possible cases. If three names *X*, *Y*, and *Z* are arranged in non-decreasing order according to their values, they can occur in exactly six possible arrangements. These are shown in Table 14.5, together with columns showing the evaluation of the expression *X*⌈(*Y*⌈*Z*) and (*X*⌈*Y*)⌈*Z*. This evaluation proceeds as follows. The first column shows the values of the expression *X*⌈*Y*, and the second shows the maximum of these values and *Z*; the third column shows the values of *Y*⌈*Z*, and the fourth column shows the maximum of *X* and these values. Since columns 2 and 4 agree, the function ⌈ is associative.

Case	*X*⌈*Y*	(*X*⌈*Y*)⌈*Z*	*Y*⌈*Z*	*X*⌈(*Y*⌈*Z*)
X *Y* *Z*	*Y*	*Z*	*Z*	*Z*
X *Z* *Y*	*Y*	*Y*	*Y*	*Y*
Y *X* *Z*	*X*	*Z*	*Z*	*Z*
Y *Z* *X*	*X*	*X*	*Z*	*X*
Z *X* *Y*	*Y*	*Y*	*Y*	*Y*
Z *Y* *X*	*X*	*X*	*Y*	*X*

Associativity of ⌈

Table 14.5

⊞5-7

14.4. DISTRIBUTIVITY

The identity

(*X*×(*Y*+*Z*))=((*X*×*Y*)+(*X*×*Z*))

is said to represent the distributivity of multiplication over addition, since it shows that the effect of multiplication by *X* on the sum *Y*+*Z* (shown to the left of the equal sign) can be said to distribute equally over each of the arguments *Y* and *Z* as shown on the right.

To see why multiplication distributes over addition, it is helpful to use the picture of multiplication presented in the discussion of commutativity, that is, the product of two factors *P* and *Q* is pictured as the number of elements in the array (*P*,*Q*) ⍴ '⎕'. The left side of the identity of the preceding paragraph is then represented by the array (*X*,*Y*+*Z*)⍴'⎕', and the right side by the sum of the arrays (*X*,*Y*)⍴'⎕' and (*X*,*Z*)⍴'⎕'. For example, if *X*←4 and *Y*←9 and *Z*←5, then:

```
 (X,Y+Z)⍴'⎕'
⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕


(X,Y)⍴'⎕'  (X,Z)⍴'⎕'
⎕⎕⎕⎕⎕⎕⎕⎕⎕  ⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕  ⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕  ⎕⎕⎕⎕⎕
⎕⎕⎕⎕⎕⎕⎕⎕⎕  ⎕⎕⎕⎕⎕
```

If the last two arrays are pushed together they form an array identical to the first and therefore contain the same total number of elements as the first.

⊞8-9

The function and distributes over or, that is:

(*X*∧(*Y*∨*Z*))=((*X*∧*Y*)∨(*X*∧*Z*))

Since the arguments *X*, *Y*, and *Z* are each limited to the values 0 and 1, this identity can be examined by evaluating the expressions for each of the eight possible cases as shown in Table 14.6.

X	*Y*	*Z*	*Y*∨*Z*	*X*∧(*Y*∨*Z*)	*X*∧*Y*	*X*∧*Z*	(*X*∧*Y*)∨(*X*∧*Z*)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributivity of ∧ over ∨

Table 14.6

⊞10-12

The function $\lceil$ distributes over $\lfloor$, that is,

$$(X\lceil(Y\lfloor Z))=((X\lceil Y)\lfloor(X\lceil Z))$$

To examine this putative identity, it is necessary to consider the six possible arrangements of the arguments X, Y, and Z when arranged in non-decresing order according to value. This is shown in Table 14.7

Case			$Y\lfloor Z$	$X\lceil(Y\lfloor Z)$	$X\lceil Y$	$X\lceil Z$	$(X\lceil Y)\lfloor(X\lceil Z)$
X	Y	Z	Y	Y	Y	Z	Y
X	Z	Y	Z	Z	Y	Z	Z
Y	X	Z	Y	X	X	Z	X
Y	Z	X	Y	X	X	X	X
Z	X	Y	Z	X	Y	X	X
Z	Y	X	Z	X	X	X	X

Distributivity of $\lceil$ over $\lfloor$

Table 14.7

A function may distribute over itself. For example, the function $\lfloor$ does so:

$$(X\lfloor(Y\lfloor Z))=((X\lfloor Y\lfloor)(X\lfloor Z))$$

This fact can be examined by means of a table similar to Table 14.7. It can easily be shown that plus does not distribute over itself. For example, $3+(4+5)$ is not equal to $(3+4)+(3+5)$.

The distributivity properties of functions can be summarized conveniently in a table. For example, for the functions $+$ $\times$ $\lceil$ and $\lfloor$, the results derived thus far are shown in Table 14.8. For example, the second row (labelled $\times$), shows that $\times$ distributes over $+$. The blank entries of the table could be filled in by further analysis. For example, plus does not distribute over either itself or times, but it does distribute over both maximum and minimum; the complete first row of Table 14.8 would therefore be 0 0 1 1.

	$+$	$\times$	$\lceil$	$\lfloor$
$+$	0			
$\times$	1			
$\lceil$				1
$\lfloor$				1

Some distributivity properties

Table 14.8

13-15

14.5. IDENTITIES BASED ON COMMUTATIVITY, ASSOCIATIVITY, AND DISTRIBUTIVITY

It is important to recognize that an identity such as $(X\times Y)=(Y\times X)$ applies not only to the simple names X and Y, but also to any expression that may be substituted for them. For example, if the expression $(P\times Q-R)$ is substituted for X, and the expression $(M+R\times Q)$ is substituted for Y, then the foregoing identity (representing the commutativity of multiplication) ensures that

$$(P\times Q-R)\times(M+R\times Q)$$

is equivalent to

$$(M+R\times Q)\times(P\times Q-R)$$

The combined use of the properties of commutativity, associativity and distributivity leads to a host of identities too numerous to list. For example, $(A+B)\times C$ is equivalent to $C\times(A+B)$ (since $\times$ is commutative), which is equivalent to $(C\times A)+(C\times B)$ (since $\times$ distributes over $+$), which is equivalent to $(A\times C)+(B\times C)$ (since $\times$ is commutative). Consequently, $(A+B)\times C$ is equivalent to $(A\times C)+(B\times C)$.

In order to show the derivation of such a result clearly, it is convenient to simply list the successive equivalent statements, one below the other, together with notes to the right of them showing what property was used to derive each new equivalent statement. For example, the derivation used in the preceding paragraph would be shown as follows:

```
(A+B)×C
C×(A+B)          Commutativity of ×
(C×A)+(C×B)      Distributivity of × over +
(A×C)+(B×C)      Commutativity of ×
```

⍠16-18

For convenience, the notes written to justify each step in a derivation will be abbreviated; the symbols C̲, A̲, and D̲ will be used to denote commutativity, associativity and distributivity. Thus C̲+ means that + is commutative, A̲× means that × is associative, and ×D̲+ means that × distributes over +.

The following shows the use of these abbreviations in the derivation of a rather important identity:

```
(A+B)×(C+D)
((A+B)×C)+((A+B)×D)                ×D+
(C×(A+B))+(D×(A+B))                C×
((C×A)+(C×B))+((D×A)+(D×B))        ×D+
((A×C)+(B×C))+((A×D)+(B×D))        C×
(A×C)+((B×C)+(A×D))+(B×D)          A+
(A×C)+((A×D)+(B×C))+(B×D)          C+
(A×C)+(A×D)+(B×C)+(B×D)            A+
```

Consequently, the first expression, (A+B)×(C+D), is equivalent to the last, (A×C)+(A×D)+(B×C)+(B×D), that is:

```
(A+B)×(C+D)
(A×C)+(A×D)+(B×C)+(B×D)
```

In other words, each element of the first sum is multiplied by each element of the second sum and the four resulting terms are added together.

⍠19-21

The foregoing result will be used in deriving further results, and to make it easy to refer to, it will be given the name <u>Theorem 1</u>. One reason for the importance of Theorem 1 is that it has some useful special cases. For

example, if A and C both have the same value X, then according to Theorem 1, the expression (X+B)×(X+D) is equivalent to (X×X)+(X×D)+(B×X)+(B×D). This leads to the following derivation:

```
(X+B)×(X+D)
(B+X)×(D+X)                             C+
(B×D)+(B×X)+(X×D)+(X×X)                 Theorem 1
(B×D)+((B×X)+(X×D))+(X×X)               A+
(B×D)+((X×B)+(X×D))+(X×X)               C×
(B×D)+(X×(B+D))+(X×X)                   ×D+
(B×D)+((B+D)×X)+(X×X)                   C×
(B×D)+((B+D)×X)+(X⋆2)                   (X⋆2)=(X×X)
(B×D)+((B+D)×(X⋆1))+(X⋆2)               (X⋆1)=X
((B×D)×X⋆0)+((B+D)×X⋆1)+(X⋆2)           (X⋆0)=1
+/((B×D)×X⋆0),((B+D)×X⋆1),(X⋆2)         (P+Q+R)=+/P,Q,R
+/((B×D),(B+D),1)×X⋆0 1 2               ((P[1]×Q[1])+(P[2]×Q[2])
                                          +(P[3]×Q[3]))=+/P×Q
```

Finally then:

```
(X+B)×(X+D)
+/((B×D),(B+D),1)×X⋆0 1 2
```

In other words, (X+B)×(X+D) is equivalent to a polynomial in X with the coefficients B×D and B+D and 1.

For example, if B is 2 and D is 3, the polynomial has the coefficients 6, 5, and 1. In other words:

```
((X+2)×(X+3))=(+/6 5 1×X⋆0 1 2)
```

The product (X+2)×(X+3) can also be expressed in the form ×/X+2 3. In general if V is any two-element vector, then ×/X+V is equivalent to (X+V[1])×(X+V[2]). Moreover, the coefficients of the equivalent polynomial are given by ×/V and +/V and 1. That is:

```
(×/X+V)=+/((×/V),(+/V),1)×X⋆0 1 2
```

⍠22-23

14.6. IDENTITIES ON VECTORS

Thus far, the identities considered have been applied only to scalar arguments. However, many of them apply equally to vectors. For example, the commutativity of ×

ssures that (A×B)=(B×A) and that 3×5 is therefore equal to ×3. However, if A is the vector 3 5 7 and B is the vector 0 ¯1, it is still true that (A×B)=(B×A). For example:

```
      A←3  5  7
      B←5  0  ¯1
      A×B
5  0  ¯7
      B×A
5  0  ¯7
```

ommutativity of × applies for vectors because it applies or each of the corresponding pairs of elements of the rguments.

For the same reason, the associativity and istributivity of functions applies to vectors as well. For xample:

```
      A←3  5  7
      B←5  0  ¯1
      C←6  4  2
      (A⌈B)⌈C
  5  7
      A⌈(B⌈C)
  5  7
      A×(B+C)
3  20  7
      (A×B)+(A×C)
3  20  7
      A⌊B
  0  ¯1
      C+(A⌊B)
  4  1
      C+A
  9  9
      C+B
1  4  1
      (C+A)⌊(C+B)
  4  1
```

⎕24-25

There are also some important identities concerning ne reduction of vectors. Thus (+/A)+(+/B) is equivalent to /A,B. For example:

```
      (+/1 2 3)+(+/4 5 6 7)
      (1+2+3)+(4+5+6+7)        Definition of +/
      1+2+3+4+5+6+7            A+
      +/1 2 3 4 5 6 7          Definition of +/
      +/(1 2 3),(4 5 6 7)      Definition of ,
```


Moreover, if the vectors A and B are of the same dimension so that A+B is meaningful, then (+/A)+(+/B) is equivalent to +/A+B. For example, if A is 1 2 3 and B is 4 5 6:

```
(+/1 2 3)+(+/4 5 6)
(1+2+3)+(4+5+6)             Definition of +/
1+2+(3+4)+5+6               A+
1+2+(4+3)+5+6               C+
1+(2+4)+(3+5)+6             A+
1+(4+2)+(5+3)+6             C+
(1+4)+(2+5)+(3+6)           A+
+/(1+4),(2+5),(3+6)         Definition of +/
+/1 2 3+4 5 6               Definition of vector addition
```

Since the only properties of addition used in the foregoing derivations were its commutativity and associativity, the same results hold for any function which is both commutative and associative. For example:

```
((⌈/A)⌈(⌈/B))=(⌈/A,B)
((⌈/A)⌈(⌈/B))=(⌈/A+B)
((×/A)×(×/B))=(×/A,B)
((×/A)×(×/B))=(×/A×B)
```

Thus if F is any function which is both associative and commutative, then

```
((F/A)F(F/B))=(F/A,B)
```

Since this is a very useful result which will be referred to again in later derivations, it will be given the name Theorem 2.

Moreover, if F is any function which is both associative and commutative, and A and B are vectors of the same dimension, then

```
((F/A)F(F/B))=(F/A F B)(Theorem 3)
```

This result will be called Theorem 3, as indicated by the note to the right of the identity.

⎕26-27

Since × distributes over +, a product of sums can be expressed as a sum of products. More explicitly, if V and W are two vectors, then

```
((+/V)×(+/W))=+/+/V∘.×W      (Theorem 4)
```

For example:

```
      V←3  1  4
      W←5  0  2  6
      (+/V)×(+/W)
104
      V∘.W
15  0  6  18
 5  0  2   6
20  0  8  24

      +/V∘.×W
39 13 52
      +/+/V∘.×W
104
```

⊞28

The preceding identity (Theorem 4) and the following one will both be useful in the treatment of products of polynomials:

```
      ((A×P)∘.(B×Q))=((A∘.×B)×(P∘.×Q))      (Theorem 5)
```

For example:

```
      A←1  2  3
      B←4  5  6  7
      P←2  0  2
      Q←3  1  3  1
      A×P
2  0  6
      B×Q
12  5  18  7
      (A×P)∘.×(B×Q)
24   10   36   14
 0    0    0    0
72   30  108   42

      A∘.×B
 4    5    6    7
 8   10   12   14
12   15   18   21

      P∘.×Q
6  2  6  2
0  0  0  0
6  2  6  2

      (A∘.×B)×(P∘.×Q)
24   10   36   14
 0    0    0    0
72   30  108   42
```


Each side of the identity of Theorem 5 is a table; the identity will be derived by showing that (for any value of *I* and any value of *J*) the element in the *I*th row and *J*th column of the table on the left is identical with the corresponding element of the table on the right:

```
 ((A×P∘.×(B×Q))[I;J]
 ((A×P)[I])×((B×Q)[J])               Definition of ∘.×
 (A[A]×P[I])×(B[J]×Q[J])             Multiplication of vectors
 A[I]×(P[I]×B[J])×Q[J]               A×
 A[I]×(B[J]×P[I])×Q[J]               C×
 (A[I]×B[J])×(P[I]×Q[J])             A×
 ((A∘.×B)[I;J])×((P∘.×Q)[I;J])       Definition of ∘.×
 ((A∘.×B)×(P∘.×Q))[I;J]              Multiplication of tables
```

The only properties of the function × used in this derivation are its associativity and commutativity. Therefore, the same derivation would apply for any function which is both assocative and commutative. Hence Theorem 5 remains true if any such function is substituted for ×. For example:

```
      ((A⌈P)∘.⌈(B⌈Q))=((A∘.⌈B)⌈(P∘.⌈Q))
```

⊞29-31

14.7. THE POWER FUNCTION

Consider the following expressions:

```
      2*3
8
      2*4
16
      (2*3)×(2*4)
128
      2*(3+4)
128
      (2*(3+4))=((2*3)×(2*4))
1
```

The foregoing result suggests the following identity:

```
      (A*(B+C))=((A*B)×(A*C))(Theorem 6)
```

It can be derived as follows:

```
      (A*B)×(A*C)
      (×/BρA)×(×/CρA)          (P*Q)=×/QρP
      ×/(BρA),×(CρA)           Theorem 2
      ×/(B+C)ρA                Definitions of ρ and ,
      A*(B+C)                  (P*Q)=×/QρP
```

⊞32

Theorem 6 leads to a very useful identity on vectors. If X is a scalar and E and F are any vectors, then:

```
((X⋆E)∘.×(X⋆F))=(X⋆E∘.+F)   (Theorem 7)
```

For example:

```
      E←0  1  2
      F←0  1  2  3
      X←2
      X⋆E
1  2  4
      X⋆F
1  2  4  8
      (X⋆E)∘.×(X⋆F)
1   2   4   8
2   4   8  16
4   8  16  32
      E∘.+F
0  1  2  3
1  2  3  4
2  3  4  5
      X⋆E∘.+F
1   2   4   8
2   4   8  16
4   8  16  32
```

33

14.8. SUM OF POLYNOMIALS

The polynomial function introduced in Chapter 13 was defined as the function P whose definition appears below:

```
∇Z←C P X
Z←(X∘.⋆¯1+⍳⍴C)+.×C∇
```

Consider the polynomials `1 3 5 P X` and `6 1 4 P X`. Their sum can be shown to be equivalent to the polynomial `7 4 9 P X` whose coefficient vector is the sum of the coefficient vectors of the given polynomials, that is:

```
((1 3 5 P X)+(6 1 4 P X))=((1 3 5+6 1 4)P X)
```


In general, if X is a scalar and A, B and E are vectors of the same dimension, then

```
((+/A×X⋆E)+(+/B×X⋆E))=(+/(A+B)×X⋆E)
```

In particular, if E is the vector `¯1+⍳⍴A`, then the left side of the foregoing identity is the sum of the polynomial with coefficients A and the polynomial with coefficients B, and the right side is the polynomial with coefficients $A+B$. The derivation of the identity follows:

```
(+/A×X⋆E)+(+/B×X⋆E)
+/(A×X⋆E)+(B×X⋆E)          Theorem 3
+/((X⋆E)×A)+((X⋆E)×B)      C×
+/(X⋆E)×(A+B)              ×D+
+/(A+B)×(X⋆E)              C×
```

The polynomials `C P X` and `(C,0) P X` are clearly equivalent, since an extra term in the polynomial with a zero coefficient will contribute nothing to the sum. For example, if `C←1 2 3`, and `X←4`, then:

```
C P X
+/1 2 3×4⋆0 1 2
+/1 2 3×1 4 16
+/1 8 48
57
```

and

```
(C,0) P X
+/1 2 3 0×1 4 16 64
+/1 8 48 0
57
```

More generally, any number of zeros may be appended to the right of a vector of coefficients without changing the polynomial, that is, `((C,N⍴0) P X)=(C P X)`. Consequently, two polynomials with coefficients C and D of different dimensions may be added by first appending enough zeros to the shorter of the two to yield a vector of the same dimension as the longer. For example, if `(⍴D)<⍴C`, then:

```
((C+(⍴C)↑D) P X)=(C P X)+(D P X)
```

The following identity applies to every case, that is, for (⍴D) less than, equal to, or greater than ⍴C:

```
M←(⍴C)⌈(⍴D)
(((M↑C)+(M↑D)) P X)=(C P X)+(D P X)
```

⊞34-35

14.9. THE PRODUCT OF POLYNOMIALS

The product of two polynomials is equivalent to another polynomial whose coefficients are easily determined from the coefficients of the given polynomials. In other words,

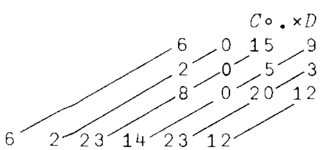
```
(E P X)=((C P X)×(D P X))
```

and the coefficients E can be determined from C and D. The method will first be described by means of an example and the derivation will be shown later.

Suppose that C←3 1 4 and D←2 0 5 3. First form the multiplication table C∘.×D:

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

Then draw diagonal lines through the table and sum the numbers on each diagonal, placing each sum at the end of its diagonal as shown below:



The result is the vector of coefficients 6 2 23 14 23 12; that is:

```
(6 2 23 14 23 12 P X)=(3 1 4 P X)×(2 0 5 3 P X)
```

The reasons why the method works will now be examined. The product of the polynomials C P X and D P X may be written as:

```
(+/C×X*¯1+⍳⍴C)×(+/D×X*¯1+⍳⍴D)
```

In this form it is clear that the product is a product of the sums of two vectors V and W, where V←C×X*¯1+⍳⍴C and W←D×X*¯1+⍳⍴D, that is, (+/V)×(+/W). The results of Theorem 4 can therefore be applied to express the result in terms of the multiplication table for V and W:

```
((+/V)×(+/W))=+/+/V∘.×W
```

Since V is the product of two vectors (that is, C and X*¯1+⍳⍴C) and W is the product of two vectors, Theorem 5 can be applied to write the table V∘.×W as the product of the two tables C∘.×D and (X*¯1+⍳⍴C)∘.×(X*¯1+⍳⍴D). That is:

```
(V∘.×W)=(C∘.×D)×((X*¯1+⍳⍴C)∘.×(X*¯1+⍳⍴D))
```

But Theorem 7 allows us to write X*(¯1+⍳⍴C)∘.+(¯1+⍳⍴D) for the second table; that is,

```
(V∘.×W)=(C∘.×D)×X*(¯1+⍳⍴C)∘.+(¯1+⍳⍴D)
```

For example, if C and D are as defined in the earlier example (that is, C←3 1 4 and D←2 0 5 3), then:

```
      C∘.×D              (¯1+⍳⍴C)∘.+(¯1+⍳⍴D)
6   0  15   9            0   1   2   3
2   0   5   3            1   2   3   4
8   0  20  12            2   3   4   5
```

The table on the right gives the exponents of X.

To summarize:

(C P X)×(D P X)	
(+/C×X*¯1+⍳⍴C)×(+/D×X*¯1+⍳⍴D)	Definition of polynomial
+/+/(C×X*¯1+⍳⍴C)∘.×(D×X*¯1+⍳⍴D)	Theorem 4
+/+/(C∘.×D)×(X*¯1+⍳⍴C)∘.×(X*¯1+⍳⍴D)	Theorem 5
+/+/(C∘.×D)×X*(¯1+⍳⍴C)∘.+(¯1+⍳⍴D)	Theorem 7

It is clear that the table of exponents (¯1+⍳⍴C)∘.+(¯1+⍳⍴D) will always be of the form shown in the example in the preceding paragraph, that is, it contains a zero in the upper left corner, 1's in the next diagonal, 2's in the next diagonal, and so on. Hence the element of the table C∘.×D that is multiplied by X*0 is in the upper left hand corner, the elements multiplied by X*1 are on the next diagonal, etc. Hence the appropriate coefficients for X*0 and X*1, and X*2, etc., in the product polynomial are obtained as the upper left corner of C∘.×D, the sum of the next diagonal of C∘.×D, the sum of the next diagonal, etc. This is the pattern shown in the rule given at the outset for multiplying polynomials.

⊞36

14.10. THE PRODUCT ×/X+V

In Section 14.5 it was shown that the product (X+2)×(X+3) could be expressed in the form ×/X+2 3, and that, more generally, if V were any 2-element vector, then ×/X+V was equivalent to (X+V[1])×(X+V[2]). Moreover, it was shown that ×/X+V was equivalent to the polynomial with coefficients (×/V),(+/V),1. The case of a vector V of arbitrary dimension will now be considered.

The expression X+2 is equivalent to the polynomial with coefficients 2 1, that is, (X+2)=+/2 1×X*0 1. Similarly, X+3 is equivalent to the polynomial with coefficients 3 1. Therefore, the product (X+2)×(X+3) can be treated as a product of polynomials. The coefficients of the product polynomial may then be obtained by the method of Section 14.9 as follows:

```
      2 1∘.×3 1

        6  2
        3  1
6   5   1
```

This result agrees with that obtained in Section 14.5.

Consider now the product ×/X+4 2 3:

```
×/X+4 2 3
(X+4)×(X+2)×(X+3)        Definition of ×/
(X+4)×(6 5 1 P X)        Preceding result
(4 1 P X)×(6 5 1 P X)    X+4 as a polynomial
```

This last product of polynomials can again be evaluated by the method of the earlier section:

```
           4 1∘.×6 5 1

           24 20  4
            6  5  1
24  26  9   1
```

Hence (×/X+4 2 3)=(24 26 9 1) P X

It should now be clear that the product ×/X+V is a product of polynomials with coefficients V[1],1 and V[2],1 and V[3],1, etc. The coefficients of a polynomial equivalent to ×/X+V can therefore be obtained by multiplying these polynomials together in turn. The following function Q produces the desired coefficients as a function of the vector V:

```
    ∇Z←Q V
[1]  Z←1
[2]  I←ρV
[3]  Z←(V[I]×Z,0)+(0,Z)
[4]  I←I-1
[5]  →3×I≠0∇
```

For example:

```
    TΔQ←3
    Q 4 2 3
Q[3] 3 1
Q[3] 6 5 1
Q[3] 25 26 9 1
24 26 9 1
```

⊞37-38

14.11. THE FACTORIAL POLYNOMIALS

The factorial polynomials introduced in Section 10.7 for the purpose of fitting functions were defined as follows:

Degree of Factorial Polynomial	Factorial Polynomial
0	1
1	X
2	X×(X-1)
3	X×(X-1)×(X-2)
4	X×(X-1)×(X-2)×(X-3)

Such a polynomial can also be written in the form ×/X+V, where V is the vector 1-⍳N and N is the degree of the polynomial.

The coefficients of a polynomial equivalent to the factorial polynomial of degree *N* can therefore be obtained by applying the function *Q* to the argument 1-⍳*N*. For example:

```
      Q -0
0 1
      Q -0 1
0 ¯1 1
      Q -0 1 2
0 2 ¯3 1
      Q -0 1 2 3
0 ¯6 11 ¯6 1
```

Hence:

```
          (0 1 P X)=X
        (0 ¯1 1 P X)=X×(X-1)
      (0 2 ¯3 1 P X)=X×(X-1)×(X-2)
  (0 ¯6 11 ¯6 1 P X)=X×(X-1)×(X-2)×(X-3)
```

In the introduction to this chapter it was shown that the function +/(⍳*X*)⋆2 (that is, the sum of the squares of the integers to *X*) was equivalent to the following sum of factorial polynomials:

```
0+X+((3÷2)×X×(X-1))+(2÷6)×X×(X-1)×(X-2)
```

Moreover, it was stated that this expression was equivalent to the polynomial (÷6)×(*X*⋆0 1 2 3)+.×0 1 3 2. This statement can now be proven as follows:

```
0+X+((3÷2)×X×(X-1))+(2÷6)×X×(X-1)×(X-2)
(÷6)×6×(X+((3÷2)×X×(X-1))+(2÷6)×X×(X-1)×(X-2))     ((÷6)×6)=1
(÷6)×((6×X)+(9×X×(X-1))+(2×X×(X-1)×(X-2)))         ×D+
(÷6)×((6×0 1 P X)+(9×0 ¯1 1 P X)+(2×0 2 ¯3 1 P X))Note 1
(÷6)×((0 6 P X)+(0 ¯9 9 P X)+(0 4 ¯6 2 P X))       Note 2
(÷6)×((0 6 0 0 P X)+(0 ¯9 9 0 P X)+(0 4 ¯6 2 P X))Note 3
(÷6)×(0 1 3 2 P X)                                  Note 4
(÷6)×+/0 1 3 2×X⋆0 1 2 3                            Note 5
(÷6)×+/(X⋆0 1 2 3)×0 1 3 2                          C×
(÷6)×(X⋆0 1 2 3)+.×0 1 3 2                          Note 6
```

Note 1: Polynomial equivalent of factorial polynomials
Note 2: (*A*×(*C P X*))=(*A*×*C*) *P X*
Note 3: ((*C*,0) *P X*)=*C P X*
Note 4: Sum of polynomials
Note 5: Definition of Polynomials
Note 6: Definition of +.×

14.12. MATHEMATICAL INDUCTION

The function +/⍳*X* can be analyzed by constructing a difference table as follows:

X	+/⍳*X*	*D*+/⍳*X*	*D D*+/⍳*X*	*D D D*+/⍳*X*
0	0	1	1	0
1	1	2	1	
2	3	3		
4	10			

The results of Section 10.7 may then be applied to conclude that the function +/⍳*X* was equivalent to the following sum of factorial polynomials:

```
0+X+(.5×X×(X-1))
```

In drawing this conclusion it is <u>assumed</u> that every one of the third differences (in the last column) would be 0. This happens to be true for the function +/⍳*X*, but the calculations of this table do not <u>prove</u> it to be so.

For example, suppose one attempted to analyze the function

```
X+(.5×X×X-1)+X×(X-1)×(X-2)×(X-3)×(X-4)
```

The first five entries in the difference table would appear exactly the same as the table shown for +/⍳*X*, and one might erroneously conclude that all third differences would be zero. However, if one considered one further row, the table would appear as follows:

X	+/⍳*X*	*D*+/⍳*X*	*D D*+/⍳*X*	*D D D*+/⍳*X*	*D D D D*+/⍳*X*	*D D D D D*+/⍳*X*
0	0	1	1	0	0	120
1	1	2	1	0	120	
2	3	3	1	120		
3	6	4	121			
4	10	125				
5	135					

A difference table can yield the coefficients of a polynomial which fits a given function exactly for a certain number of values of the argument and which <u>probably</u> fits it very nearly or exactly for all values of the argument, but study of the difference table alone cannot ensure that it fits for all points. It is therefore desirable to develop other means of verifying that an expression derived from a difference table does in fact agree with the given function for points other than those actually used in the table.

Let us suppose that the functions +/⍳*X* and *X*+.5×*X*×*X*-1 do agree for some integer value *K*, that is, we suppose that

```
(+/⍳K)=K+.5+K×K-1
```

From this assumption alone, we will now show that they must agree for the argument *K*+1.

We have undertaken to show that +/⍳*K*+1 is equal to (*K*+1)+.5×(*K*+1)×(*K*+1)-1, in other words to show that

```
(+/⍳K+1)-((K+1)+.5×(K+1)×(K+1)-1)
```

is zero.

Let the functions *F* and *G* be defined as follows:

```
∇Z←F X                  ∇Z←G X
Z←+/⍳X∇                 Z←X+.5×X×X-1∇
```

We wish to show that *F* and *G* agree for all integer values of their argument, that is, that (*F X*)-(*G X*) is zero for every integer *X*. We begin by expressing the difference for the argument *K*+1 in terms of the difference for argument as follows:

```
(F K+1)-(G K+1)
(+/⍳K+1)-((K+1)+.5×(K+1)×(K+1)-1)  Definitions of F and G
((+/⍳K)+(K+1))-((K+1)+.5×(K+1)×K)  (+/⍳K+1)=(+/⍳K)+K+1
((+/⍳K)+(K+1)-(K+1))-.5×(K+1)×K
((+/⍳K)+0)-.5×(K+1)×K
(+/⍳K)-.5×(2×K)+(K-1)×K
(+/⍳K)-K+.5×K×K-1
(F K)-(G K)                        Definitions of F and G
```

Hence the difference between *F K*+1 and *G K*+1 must be the same as the difference between *F K* and *G K*. In other words, if *F K* and *G K* are equal, then *F K*+1 and *G K*+1 must also be equal.

But for *K*=1, *F K* and *G K* are obviously equal; that is +/⍳1 is equal to 1+.5×1×0. Hence *F* 1+1 must equal *G* 1+1, that is, *F* 2 equals *G* 2. Thus, for *K*=2, *F K* equals *G K*. Therefore *F* 2+1 equals *G* 2+1, and so on for all possible integer arguments. Hence *F X* equals *G X* for all positive integer values of *X*.

This method of proof is called <u>mathematical</u> <u>induction</u>. To prove that two function *F* and *G* are equivalent, proceed as follows:

1) Show that the difference (*F K*+1)-(*G K*+1) is equal to the difference (*F K*)-*G K*.

2) Show that *F* 1 is equal to *G* 1.

If items 1 and 2 can both be shown to be true then the functions must agree for all positive integer arguments.

⍠41

Chapter 15

LINEAR FUNCTIONS

15.1. INTRODUCTION

The expression 4+3×X is said to be a linear function. The reason for the term "linear" becomes evident on plotting the function; as shown in Figure 15.1, the plot forms a straight line.

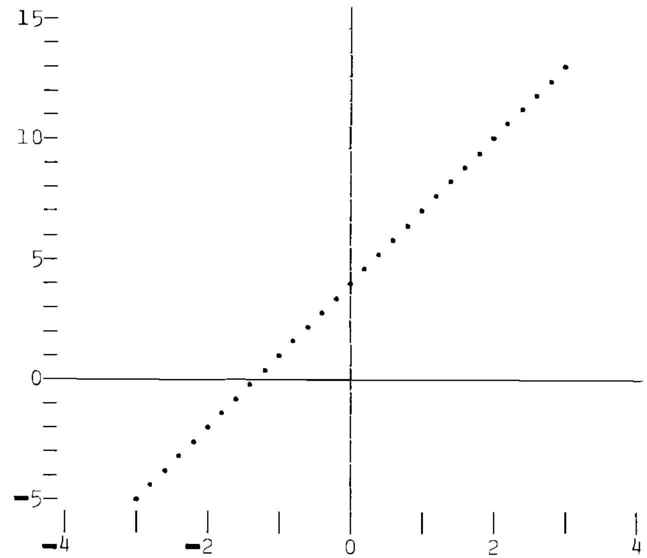
More generally, if A and B are any scalar constants, then the expression A+B×X is a linear function. A plot of several linear functions sharing the same value of B and having different values of A (Figure 15.2) shows that the graphs have the same slope (i.e., they are parallel), but that they intercept the Y-axis at different points determined directly by the value of A. That is, the Y-intercept of the function 5+3×X is 5, the Y-intercept of 2+3×X is 2, and so on.

A plot of the function A+B×X for a common value of A and different values of B (Figure 15.3) shows that the functions share the same Y-intercept but have different slopes which are directly determined by B, that is, the vertical distance between any two points on the graph is B times the horizontal distance between them.

If A, B, and C are scalar constants, then the expression A+(B×X)+(C×Y) is a function of two arguments X and Y. For any fixed value of X the expression is a linear function of Y. For example, the function 1+(2×X)+(3×Y) is equivalent to 1+(2×4)+(3×Y) if X is given the fixed value 4. This in turn is equivalent to 9+3×Y, which is clearly a linear function of Y.

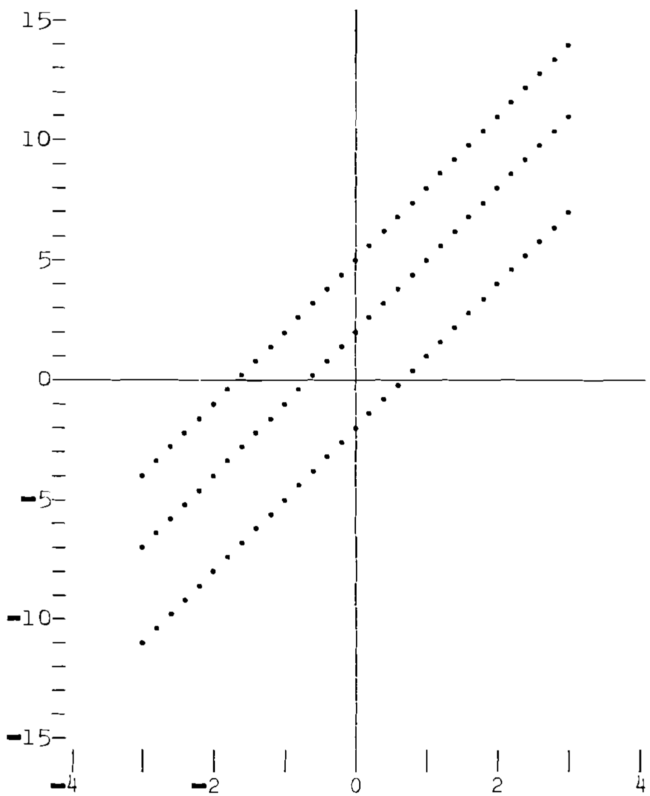
Similarly, for a fixed value of Y, the expression A+(B×X)+(C×Y) is a linear function of X. Consequently it is said to be a linear function of two arguments.

If the two arguments X and Y are combined in a single two-element vector V, then the linear function 1+(2×X)+(3×Y) can be written more concisely as 1+(2 3+.×V). more generally, for any scalar A and any two-element vector B, the expression A+B+.×V represents a linear function of the two arguments V[1] and V[2].



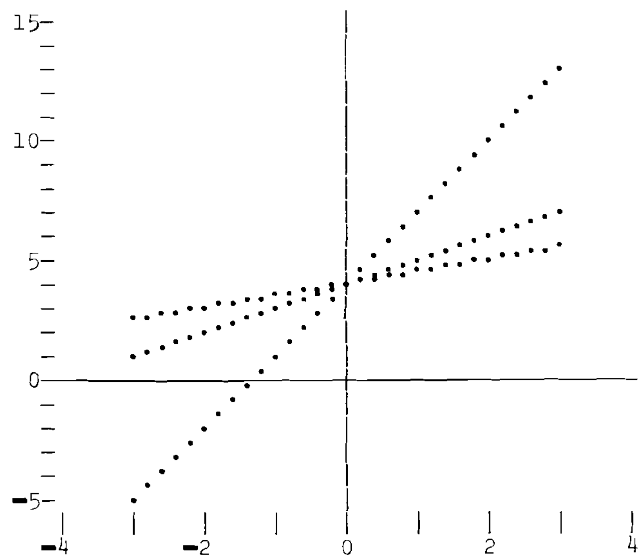
The linear Function 4+3×X

Figure 15.1



Linear Functions $A+3\times X$ (Common Slope)

Figure 15.2



Linear Functions $4+B\times X$ (Common Intercept)

Figure 15.3

This vector form of writing linear equations possesses three important advantages. First, the expression *A+B+.×V* applies for a linear function of any number of arguments; it is only necessary that *B* and *V* each have the same number of elements as there are arguments. For example, the expression 1+2 3 4+.×*V* represents a linear function of the three arguments *V*[1], *V*[2], and *V*[3]. It could be written in terms of these individual arguments as follows:

```
1+(2×V[1])+(3×V[2])+(4×V[3])
```

or, if the three arguments are called *X*, *Y*, and *Z* it could be written as:

```
1+(2×X)+(3×Y)+(4×Z)
```

⌹1-3

The second advantage of using the expression *A+B+.×V* is that it can express not only one linear function, but several. For example, if *B* is the matrix

```
      B←2 2⍴2 3 1 4
      B
 2  3
 1  4
```

and *A* is the vector 5 7, then *A+B+.×V* yields two results:

```
5+2 3+.×V
```

and

```
7+1 4+.×V
```

Hence *A+B+.×V* expresses two linear functions in two arguments.

In general, if *A* is a vector of *M* elements and *B* is an *M* by *N* matrix, then *A+B+.×V* expresses *M* linear functions in *N* arguments.

⌹4-6

15.2. MAPPINGS

If *A* is a two-element vector and *B* is a 2 by 2 matrix, then the expression *A+B+.×V* applies to a two-element vector *V* and yields a two-element vector as a result. For example:

```
      A←¯2 ¯4
      B←2 2⍴1 2 3 2
      B
 1  2
 3  2
      B+.×1 2
 5  7
      A+B+.×1 2
 3  3
```

The vector 1 2 can be shown as a point on the graph as can the vector 3 3 which results from applying the linear function *A+B+.×V* to it. Hence the effect of the linear function can be shown as a map by drawing an arrow from the point representing the vector 1 2 to the point representing the result 3 3. This is shown in Figure 15.4.

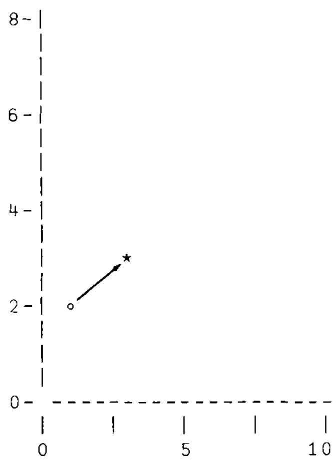
A more complete picture of the effect of the linear function *A+B+.×V* can be obtained by computing and plotting the results from applying it to a number of points. Figure 15.5 shows the mapping from the points 1 2 and 1 5 and 5 5 and 5 2.

⌹7

The effects of *A* and *B* can be studied separately by considering certain special cases. For example, if *A* has the value 0 0, then *A+B+.×V* is equivalent to *B+.×V*.

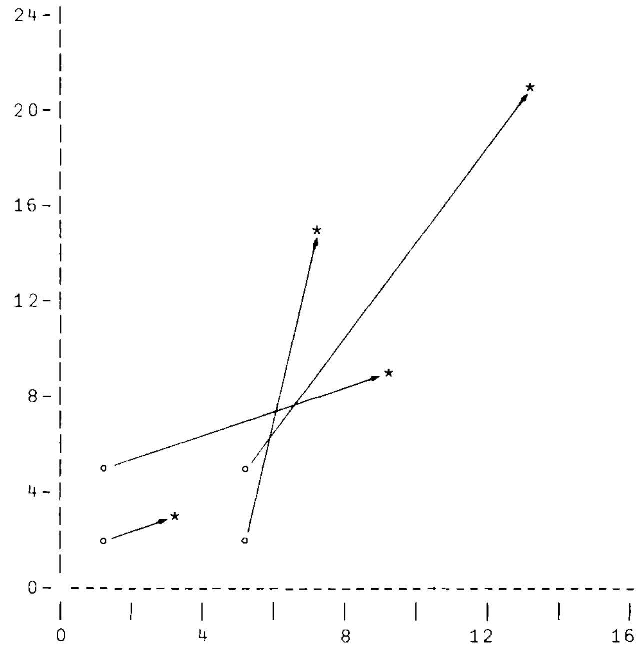
The linear function *B+.×V* always leaves the origin (the point 0 0) unchanged, that is, *B+.×*0 0 is 0 0 no matter what *B* is. Apart from this simple fact, the mapping produced by *B+.×V* can be quite complicated. For example, if

```
      B←2 2⍴2.5 ¯.5 1.5 .5
      B
    2.5   ¯0.5
    1.5    0.5
      B+.×1 7
¯1  5
      B+.×2 6
2 6
      B+.×3 5
5 7
      B+.×4 4
8 8
```



A Linear Mapping
on One Point

Figure 15.4



A Linear Mapping on Several Points

Figure 15.5

then the mapping produced by B+.×V is shown in Figure 15.6. From this figure it appears that the effects on different points may be quite different. For example, the last point S is "stretched" (that is, it maps into a point straight away from the origin in the same direction as S), the second point Q maps into itself, and the arrows from P and R lead in opposite directions. Points (such as P, Q, R, and S) which lie on a line do, as remarked before, map into points which also lie on a line.

⌹8

15.3. ROTATIONS

There is a certain class of matrices which yield a very simple and important mapping. If B is a 2 by 2 matrix of the form

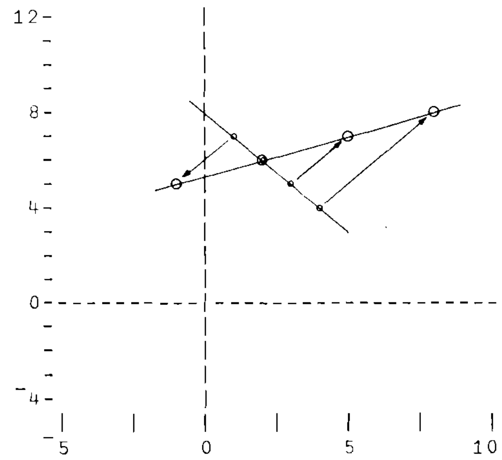
```
      S   C
     -C   S
```

and C is equal to either (1-S*2)*.5 or -(1-S*2)*.5, then the mapping B+.×V is a <u>rotation</u> about the origin. That is, each point maps into a point the same distance from the origin but displaced by rotation through a certain angle. Such a matrix will be called a <u>rotation matrix</u>. For example, if S←.5, then (1-S*2)*.5 is equal to (3÷4)*.5 (which is approximately .866), and B is the matrix:

```
 0.5      0.866
¯0.866    0.5
```

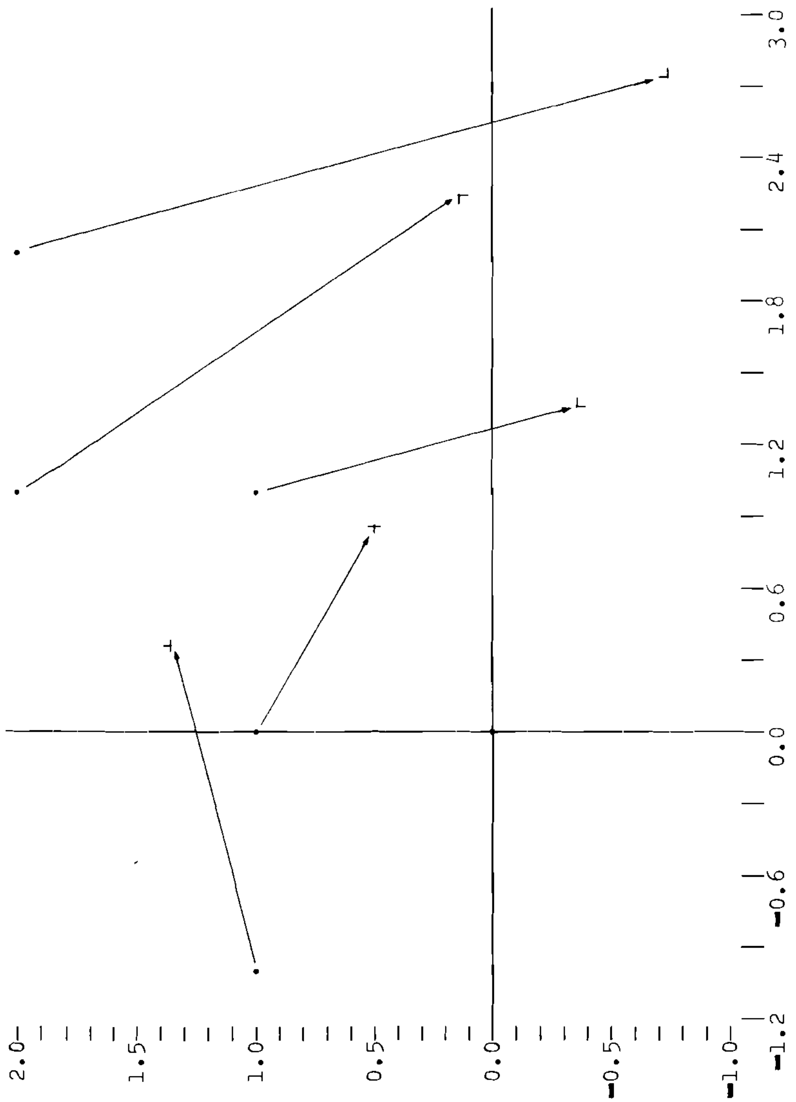
Figure 15.7 shows the mapping B+.×V applied to the following set of points:

```
      B+.×0 0
0   0
      B+.×1 1
1.37  ¯0.366
      B+.×2 2
2.73  ¯0.732
      B+.×¯1 1
0.366  1.37
      B+.×0 1
0.866  0.5
      B+.×1 2
2.23  0.134
```



A Linear Mapping

Figure 15.6



A Rotation

Figure 15.7

To see why this mapping is called a rotation, lay a sheet of translucent paper over the plot and copy onto it the original points V and the axes. Then place a pin through the origin and rotate the translucent overlay until one of the points V coincides with the point B+.×V into which it maps. It will then be seen that all points in V lie over the corresponding points B+.×V. Moreover, the angle of rotation is the angle formed between the new and old positions of the axes.

If S is equal to 1, then (1-S⋆2)⋆.5 is equal to zero, and the rotation matrix B becomes

```
1  0
0  1
```

In this case it is clear that B+.×V yields V for any V. The mapping B+.×V is therefore called the identity mapping, and the matrix B is called the identity matrix. ⎕9-13

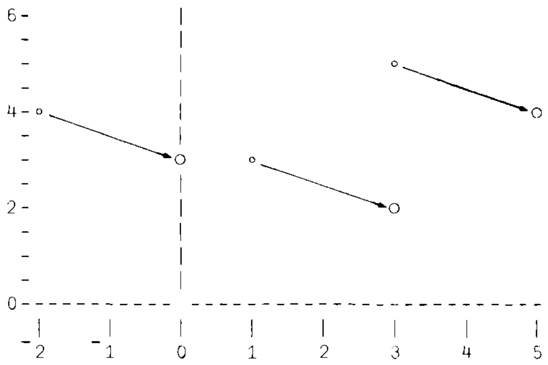
15.4. TRANSLATION

The effect of the vector A in the linear function A+B+.×V is most easily seen if B is chosen to be the identity matrix. In that case B+.×V yields V and the expression A+B+.×V is therefore equivalent to the expression A+V. This mapping is shown in Figure 15.8 for the case A←2 ¯1. All of the mapping arrows are parallel and of the same length. This sort of mapping is called a translation.

If the first element of A is zero, the translation is vertical, moving upward if A[2] is positive and downward if it is negative. Likewise, if the second element is zero the translation is horizontal, to the right if A[1] is positive, and to the left if it is negative. ⎕14

15.5. LINEAR FUNCTION ON A SET OF POINTS

It is often necessary to apply the expression B+.×V to a number of points, that is, for a number of different values of V. This can be done comveniently by assembling the values into a single matrix M such that each of the points appear as a column of M. Then the expression B+.×M



Translation

Figure 15.8

yields a matrix whose columns are the results of applying the linear function to each column of *M*. For example, if the required points are 2 3 and 4 2 and 1 5, then

```
      M←⍉3 2ρ2 3, 4 2, 1 5
      M
  2  4  1
  3  2  5
```

Moreover, if

```
      B←2 2ρ1 2 3 2
      B
  1  2
  3  2
```

then

```
      B+.×M
   8   8  11
  12  16  13
```

⊞15

The translation *A*+*V* does not extend to a matrix of points quite so neatly as does the expression *B*+.×*V*. For example, if *A*←1 2 and *M* is the matrix of the preceding paragraph, then *A*+2 3 is a translation of the vector 2 3 but *A*+*M* cannot be evaluated because *A* and *M* are not of the same shape. What is needed is a matrix *P* of the same shape as *M* and having each column equal to *A*, that is:

```
      P
  1  1  1
  3  3  3
```

Then *P*+*M* yields the desired translation of the columns of *M*;

```
      P+M
  3  5  2
  6  5  8
```

The matrix *P* can be obtained by the expression ⍉(⌽ρ*M*)ρ*A*. Hence the translation of a set of points *M* can be expressed as:

```
      (⍉(⌽ρM)ρA)+M
```

and the general linear function *A*+*B*+.×*V* can be expressed for a set of points *M* as:

```
      (⍉(⌽ρM)ρA)+B+.×M
```

⊞16

15.6. ROTATION AND TRANSLATION

If *B* is a rotation matrix, then the function *B*+.×*V* is a rotation and the function *A*+*B*+.×*V* is a rotation followed by a translation. Similarly, *B*+.×*A*+*V* is a translation followed by a rotation. A few experiments with these expressions for some chosen values of *A* and *B* applied to a number of points *V* will show that the two expressions are <u>not</u> equivalent.

However, the same experiments will be seen to suggest that *B*+.×*A*+*V* <u>is</u> equivalent to rotation by *B* (that is, *B*+.×*V*) followed by some translation. The amount of the translation will be found to be not *A* but rather *B*+.×*A*. In other words:

```
B+.×A+V
(B+.×A)+(B+.×V)
```

The foregoing identity expresses the fact that the inner product function +.× distributes over +. This identity holds for any matrix *B* (i.e., it is not limited to rotation matrices). A proof of this for 2 by 2 matrices is fairly simple and is outlined in an exercise. The identity also holds for matrices *B* of any dimension. The proof of this is more involved and will not be attempted here, although the reader should be able to extend the method of proof used for a 2 by 2 matrix to the case of a 3 by 3 matrix. Any reader not wishing to work through the proofs may wish to shore up his faith in the identity by performing a number of experiments.

⌹17

15.7. STRETCHING

If *B* is the matrix

```
3 0
0 3
```

then the expression *B*+.×*V* "stretches" the point *V* by a factor of 3, since each element of the result is 3 times the corresponding element of *V*. In a plot, such stretching is equivalent to extending the line from the origin to the point *V* to 3 times its length. If *I* is the identity matrix and *T* is any scalar value, then *T*×*I* is a stretching matrix whose degree of stretch is equal to *T*.

A more general stretching is illustrated by the matrix *B* below:

```
3 0
0 2
```

For such a matrix, the expression *B*+.×*V* stretches by a different amount for each coordinate.

⌹18

15.8. IDENTITIES ON THE INNER PRODUCT +.×

The inner product +.× has been seen to be central to the treatment of linear functions. Certain identities involving the inner product are also important in the study of linear functions. One of these has already been established, namely, the distributivity of +.× over +:

```
B+.×A+V
(B+.×A)+(B+.×V)
```

A second important fact is that this inner product +.× is associative, that is:

```
M+.×(B+.×V)
(M+.×B)+.×V
```

A proof of this will be outlined in exercises for the case of 2 by 2 matrices *M* and *B*.

⌹19-21

15.9. LINEAR FUNCTIONS ON 3-ELEMENT VECTORS

If *V* is a 3-element vector, *B* is a 3 by 3 matrix and *A* is a 3-element vector, then *A*+*B*+.×*V* is again a linear function of *V* which produces a 3-element result.

In order to get a clear picture of the mapping produced by the function *A*+*B*+.×*V* for vectors *V* of dimension 3, it is necessary to devise a way of plotting a point having 3 coordinates. This can be done as follows: Draw the usual coordinates for a graph on a flat piece of thick styrofoam and obtain a set of wires of various lengths. Stick a wire into the point 3 4 on the graph so that it extends straight up to a length of 5 units. The tip of the wire then represents the point (that is, the vector) 3 4 5. Other points can be represented similarly.

The points plotted in 3-dimensions will be easier to see if the wires are tipped with colored beads. Moreover, if two different colors are used to plot the points *V* and the points *A*+*B*+.×*V*, then the effect of a linear mapping can be observed easily. Light tape can be used to connect each point to the corresponding point produced by the linear function. Alternatively, numeric labels identifying the points can be attached to them.

For example:

```
      B←3 3⍴2 0 ¯1 1 ¯2 1 1 1 1
      M←⍉5 3⍴1 1 1, 2 2 2, 3 3 3, 0 1 1, 0 2 2

      B                        M
 2  0 ¯1                1  2  3  0  0
 1 ¯2  1                1  2  3  1  2
 1  1  1                1  2  3  1  2

      B+.×M
 1  2  3 ¯1 ¯2
 0  0  0 ¯1 ¯2
 3  6  9  2  4
```

The plot of this mapping is shown in Figure 15.9.

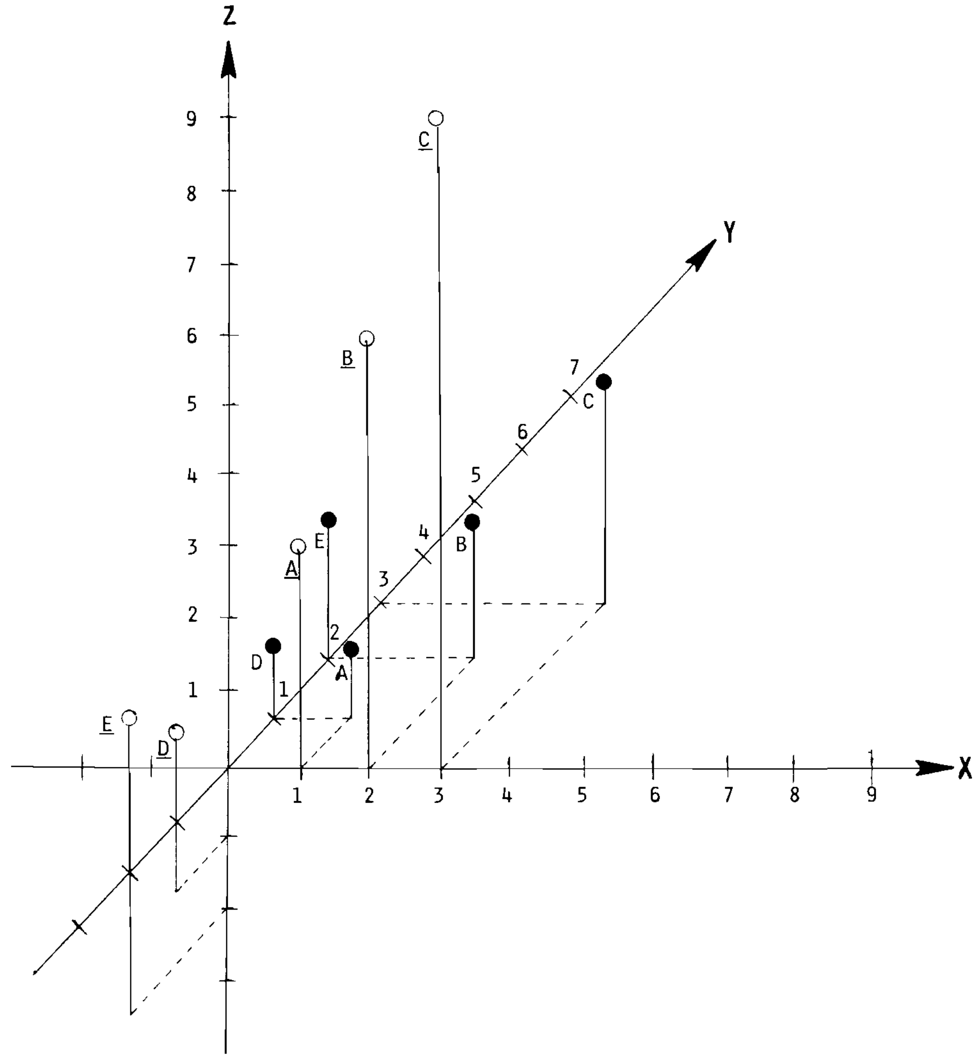
Most of the properties of linear functions observed for 2-element vectors carry over to the case of 3-dimensions. For example, points lying on any line map into points lying on a line. Since this is true for a line in any direction it is also true for any plane, that is, points lying in the same plane map into points lying in a plane. performing and plotting experiments for various values of *B* and *V* should make this clear.

The identity matrix for 3-dimensions is the matrix *I* shown below:

```
1  0  0
0  1  0
0  0  1
```

It is easy to show that this is the identity matrix by showing that *I*+.×*V* yields *V* for any 3-element vector *V*.

22-23



A Mapping in Three Dimensions

Figure 15.9

15.10. ROTATIONS IN THREE DIMENSIONS

In an earlier section it was shown that the expression `B+.×V` produced a rotation (in two-dimensions) if `B` was a matrix of the form:

```
 S  C
-C  S
```

where `C` is equal to `(1-S*2)*.5` or to `-(1-S*2)*.5`.

It was also shown (in Exercise 15.13) that for such a matrix `B`, multiplication by its transpose yields the identity matrix, that is: `B+.×⍉B` is equal to the identity matrix. This is the essential property of a rotation matrix and applies in 3-dimensions as well. Thus any 3 by 3 matrix `B` such that `B+.×⍉B` yields the identity matrix is a rotation matrix.

it is easy to assemble a matrix `B` which meets these specifications. If `S` and `C` satisfy the requirements imposed in the first paragraph, then the following matrix `R` is a rotation matrix.

```
1  0  0
0  S  C
0 -C  S
```

For `⍉R` is equal to

```
1  0  0
0  S -C
0  C  S
```

and `R+.×⍉R` therefore equals

```
1        0                  0
0    (S*2)+(C*2)       (S×-C)+(C×S)
0   (-C×S)+(S×C)        (C*2)+(S*2)
```

which (since `(S*2)+(C*2)` equals 1) is the identity matrix.

Similarly,

```
 S  C  0          S  0  C
-C  S  0   and    0  1  0
 0  0  1         -C  0  S
```

are rotation matrices. Moreover, if `R` and `T` are any two rotation matrices then the product `R+.×T` is also a rotation matrix.

Chapter 16

INVERSE LINEAR FUNCTIONS

16.1. INTRODUCTION

The importance of inverse functions was noted in Chapter 11 where it was remarked that whenever one finds use for a particular function, the need for the inverse of that function usually arises. This is true of linear functions, and this chapter will be devoted to methods for obtaining the inverse of a linear function.

For a linear function of a single argument `X`, the inverse has already been determined in Chapter 11, where it was shown that the inverse of the function

```
A+B×X
```

was

```
(÷B)×(-A)+X
```

For example, if `A` is 3 and `B` is 4 and `X` is 7, then `A+B×X` makes 31. Applying the inverse function to this result yields:

```
(÷4)×(-3)+31
(÷4)×28
7
```

Hence the result is the original value of `X` as required.

An important point is that the inverse function `(÷B)×(-A)+X` is itself a linear function. To show that this is so, we write the expression in an equivalent form as follows:

```
(÷B)×(-A)+X
((÷B)×(-A))+((÷B)×X
```

The last expression is a linear function since it is a constant (that is, `(÷B)×(-A))` added to a constant (that is, `÷B`) times *X*. For example, if *A* is 8 and *B* is 4, then the original linear function `A+B×X` is

```
      8+4×X
```

and the inverse is

```
      ((÷4)×(-8))+((÷B)×X)
      ¯2+.25×X
```

Chapter 11 dealt only with the inverses of functions of a single argument and, strictly speaking, the notion of inverse functions applies only to such a case. However, as shown in Chapter 15, a linear function of several arguments *X*, *Y*, and *Z* can be treated as a function of the single vector argument *V*, where `V←X,Y,Z`. In this sense, a linear function of several arguments does possess an inverse. As was just shown for the case of a single argument *X*, the inverse of any linear function is itself a linear function.

16.2. SOME INVERSE FUNCTIONS

As we did in the study of linear functions in Chapter 15, we will begin with a simple case in which *A* is zero, that is, we will consider the linear function `B+.×V`. Suppose that *B* and *IB* are defined as follows:

```
      B←2 2ρ3 1 5 2
      IB←2 2ρ2 ¯1 ¯5 3
      B             IB
 3  1            2 ¯1
 5  2           ¯5  3
```

Then the linear function `IB+.×V` is the inverse of the function `B+.×V`. This can be tested on a number of examples as follows:

```
      B+.×1 2
5  9
      IB+.×5 9
1  2
      B+.×¯3 4
¯5 ¯7
      IB+.×¯5 ¯7
¯3  4
      B+.×IB+.×2 5
2  5
      IB+.×B+.×2 5
2  5
```


Similarly, in 3 dimensions the following matrices *B* and *IB* define inverse functions:

```
      B←3 3ρ1 0 2 2 1 3 4 0 4
      IB←3 3ρ¯1 0 .5 ¯1 1 ¯.25 1 0 ¯.25
      B              IB
 1  0  2        ¯1  0  .5
 2  1  3        ¯1  1 ¯.25
 4  0  4         1  0 ¯.25

      B+.×1 2 4
9  16  20
      IB+.×9 16 20
1  2  4
```

⌹1-2

The foregoing illustrates how the linear function `B+.×V` may have an inverse `IB+.×V` which is also a linear function. It does not show how to go about finding a suitable inverse *IB* for any given matrix *B*. This is a rather difficult matter which will be addressed in subsequent sections.

In these later sections we will be considering the problem of finding an inverse for the function `B+.×V` and will ignore the more general problem of finding an inverse to the general linear function `A+B+.×V`. The reason is that the inverse to `A+B+.×V` can be easily obtained once we find an inverse to `B+.×V`. This will now be shown.

Suppose a matrix *IB* has been found which is inverse to *B*, that is,

`IB+.×B+.×V`   yields *V*.

Then `IB+.×(-A)+V` is the function inverse to `A+B+.×V`. For:

```
      IB+.×(-A)+(A+B+.×V)
      IB+.×((-A)+A)+(B+.×V)      Associativity of +
      IB+.×0+(B+.×V)
      IB+.×B+.×V
      V                          Because IB is inverse of B
```

Consequently, attention will be restricted to the problem of finding an inverse to the function `B+.×V`.

16.3. THE SOLUTION OF LINEAR EQUATIONS

In Section 11.7 it was remarked that even though a general expression for a function *G* inverse to *F* could not be found, yet one could find the value of *G N* for any argument *N* by simply finding a value of *Y* such that

```
      N=F Y
```

This value satisfies the only requirement on *G*, namely, that *F G N* must be equal to *N*, for if *G N* is *Y*, then *F G N* is *F Y* which in turn is equal to *N* since *Y* was so chosen.

Finding a value of *Y* such that *N=F Y* is called "solving the equation *N=F Y*". It is often easier to solve such an equation than to find a general expression for the inverse function *G*. Moreover, solving such an equation for several different values of *N* may give some clues to an expression for *G*.

In any case, we shall approach the problem of finding an inverse to the function *B+.×V* by developing methods for solving the equation *N=B+.×V*. Since *N* is a vector, we require a value of *V* such that <u>each</u> element of *N* agrees with <u>each</u> element of *B+.×V*. This can be expressed by saying that the following expression is required to be true:

```
      ∧/N=B+.×V
```

For example, if

```
      B←2 2⍴1 2 2 3
      B
  1  2
  2  3
      N←3 4
      V←1 1
      B+.×V
3  5
      N=B+.×V
1  0
      ∧/N=B+.×V
0
```

then the first element of *B+.×V* agrees with the first element of *N*, but *V* is not a solution of the equation *N=B+.×V* since the elements do not all agree, as shown by the zero value resulting from the expression *∧/N=B+.×V*. However, the vector ¯1 2 is a solution as shown below:

```
      V←¯1 2
      B+.×V
3  4
      N=B+.×V
1  1
      ∧/N=B+.×V
1
```


16.4. BASIC SOLUTIONS

A solution of the equation

```
      ∧/1 0 = B+.×V
```

or of the equation

```
      ∧/0 1 = B+.×V
```

will be called a <u>basic</u> solution. Basic solutions have two important properties:

They are rather easy to obtain.

They can be used to determine solutions to the equation *∧/N=B+.×V* for any value of *N*.

The second matter will be explored first, that is, we will first assume that we know two basic solutions *V1* and *V2* such that

```
      ∧/1 0=B+.×V1
      ∧/0 1=B+.×V2
```

and will show how *V1* and *V2* can be used to determine a solution to the general equation *∧/N=B+.×V*. The matter of how to determine *V1* and *V2* themselves will be deferred to the succeeding section.

if *V1* and *V2* are basic solutions for a matrix *B*, then the vector

```
      V←(N[1]×V1)+(N[2]×V2)
```

is a solution of the equation *∧/N=B+.×V*. For example, if *B* is the matrix

```
 4  2
 1  3
```

then

```
      V1←.3 ¯.1
      V2←¯.2  .4
```

are basic solutions, for:

```
      B+.×V1
 1 0
      B+.×V2
 0 1
```

Moreover, if N←3 5, then:

```
      V←(N[1] ×V1)+(N[2] ×V2)
      V
¯0.1  1.7
      B+.×V
3  5
```

and V is indeed a solution of the equation ∧/N=B+.×V.

⌸5-6

The method is based on two simple facts:

1) B+.×S×V is equal to S×B+.×V for any scalar S

2) B+.×P+Q is equal to (B+.×P)+(B+.×Q)
   (Distributivity of +.× over +)

The first of these facts is easily established and the second was established in Exercises in Chapter 15.

The following arguments can now be used to show that V←(N[1]×V1)+(N[2]×V2) is in fact a solution of the equation ∧/N=B+.×V:

```
B+.×V
B+.×((N[1]×V1)+(N[2]×V2))      Definition of V
(B+.×N[1]×V1)+(B+.×N[2]×V2)    Fact 2
(N[1]×B+.×V1)+(N[2]×B+.×V2)    Fact 1
(N[1]×1 0)+(N[2]×0 1)          Definition of V1 and V2
(N[1],0)+(0,N[2])
N
```

16.5. DETERMINING BASIC SOLUTIONS

We now address the problem of finding basic solutions, that is, finding solutions V1 and V2 for the following set of equations:

```
∧/1 0 = B+.×V1
∧/0 1 = B+.×V2
```


If one has a vector VA such that B+.×VA is equal to S,0 then V1←(÷S)×VA is a basic solution. For example:

```
      B
 1  3
 4  2
      VA
 2 ¯4
      B+.×VA
¯10 0
      V1←(÷¯10)×VA
      V1
¯.2  .4
      B+.×V1
 1  0
```

The foregoing is a simple application of Fact 1 of the preceding section. Moreover, the expression (÷S)×VA can be written equivalently as VA÷S.

To find a basic solution we can therefore begin with the simpler problem of finding a vector VA such that B+.×VA is equal to S,0 for any value of S. It is easy to choose a value of VA such that the second element of B+.×VA is zero; simply take the second row of B, reverse the sign of its first element, and then reverse the order of its elements. In other words:

```
      VA←⌽ ¯1 1×B[2;]
```

For example, if B is the matrix

```
  1  3
  4  2
```

then

```
  4  2          Second row of B (that is, B[2;])
 ¯4  2          Reversal of sign (¯1 1×B[2;])
  2 ¯4          Reversal of order (⌽¯1 1×B[2;])
```

```
      B+.×2 ¯4
¯10 0
```

Hence if VA←2 ¯4, then B+.×VA is ¯10 0. Moreover, V1←VA÷¯10 is a basic solution:

```
      V1←VA÷¯10
      V1
¯.2  .4
      B+.×V1
 1  0
```

⌸7-9

The following set of equivalent statements show why the second element of B+.×VA is zero when VA is determined by the foregoing procedure:

```
(B+.×VA)[2]                      Second element of B+.×VA
B[2;]+.×VA                       Definition of inner product
+/B[2;]×VA                       Definition of inner product
+/B[2;]×⌽¯1 1×B[2;]              Choice of VA
+/B[2;]×B[2;2],-B[2;1]           Reversals of sign and order
+/(B[2;1],B[2;2])×(B[2;2],-B[2;1])
(B[2;1]×B[2;2])+(B[2;2]×-B[2;1])
0
```

The entire procedure for determining the basic solution V1 can therefore be summarized as follows:

```
      VA←⌽¯1 1×B[2;]
      R1←B+.×VA
      V1←VA÷R1[1]
```

It should be clear that a similar procedure applies to the second basic solution V2 such that ∧/0 1 = B+.×V2. It is only necessary to interchange the roles of the first and second elements as may be seen by comparing the pair of procedures below:

```
      VA←⌽¯1 1×B[2;]              VB←⌽1 ¯1×B[1;]
      R1←B+.×VA                   R2←B+.×VB
      V1←VA÷R1[1]                 V2←VB÷R2[2]
```

For example:

```
        B
  3  5
  2  4
      VA←⌽¯1 1×B[2;]              VB←⌽1 ¯1×B[1;]
      VA                          VB
4  ¯2                          ¯5 3
      R1←B+.×VA                   R2←B+.×VB
      R1                          R2
2  0                           0  2
      V1←VA÷R1[1]                 V2←VB÷R2[2]
      V1                          V2
2  ¯1                          ¯2.5  1.5
      B+.×V1                      B+.×V2
1  0                           0  1
```


16.6. SIMPLIFIED CALCULATIONS FOR BASIC SOLUTIONS

Examination of the procedures for determining basic solutions shows that certain simplifications can be made. For example, in calculating R1←B+.×VA, only the first element of R1 need be calculated since it is the only one used in the expression V1←VA÷R1[1]. Thus R1[1] can be computed as B[1;]+.×VA, which requires only half as much computing as does B+.×VA. On the other hand, it may be wise to do the whole calculation B+.×VA since the value of the second element (which must be zero if VA has been computed correctly) is a check on the work thus far.

Similar remarks apply to the calculation of R2[2] for the second basic solution; that is, R2[2] is B[2;]+.×VB. Moreover, R2[2] need not be computed at all since it is equal to R1[1], as you may have noticed in previous examples and exercises. The reason for this appears in the following identity, in which the first line is the expression for R1[1] and the second line is the expression for R2[2]:

```
      +/(B[1;1],B[1;2])×(B[2;2],-B[2;1])
      +/(B[2;1],B[2;2])×((-B[1;2]),B[1;1])
```

Taking either of these expressions for R1[1], it is clear that if B is a matrix having the elements P, Q, R, and S as follows:

```
      P  Q
      R  S
```

then R1[1] is equal to (P×S)-(Q×R). In other words, one takes the product of the first element with the one diagonally opposite and subtracts from it the product of the remaining two elements. For example, if B is the matrix

```
   5  2
   7  4
```

then the value of R1[1] is (5×4)-(2×7), that is, 6

Continuing with this example, the whole computation of V1 can be expressed as follows:

```
      V1←4 ¯7÷(5×4)-(2×7)
```

Similarly, V2 is obtained as follows:

```
      V2←¯2 5÷(5×4)-(2×7)
```

16.7. THE DETERMINANT FUNCTION

The expression for R1[1] (or for R2[2]) developed in the preceding section is a very important function called the <u>determinant</u>. It was also shown that if B is the matrix

```
P  Q

R  S
```

then the determinant of B is the expression (P×S)-(Q×R).

The determinant function may be defined formally as follows:

```
    ∇ Z←DET B
[1]   Z←(B[1;1]×B[2;2])-(B[1;2]×B[2;1])∇
```

For example:

```
      B←2 2⍴5 2 7 4
      B
 5  2
 7  4

      DET B
6
```

The function DET will be used throughout the remainder of this chapter. The notion of determinant is used for square matrices of dimensions higher than 2 by 2, but it must be emphasized that the function DET applies only to 2 by 2 matrices.

⍈12-15

16.8. MATRIX FORM OF THE BASIC SOLUTIONS

It is convenient to represent the basic solutions V1 and V2 as a single matrix BS whose first column is V1 and whose second column is V2. For example, if B is the matrix

```
3  5
2  4
```

then V1←2 ¯1 and V2←¯2.5 1.5 and the matrix BS is

```
 2  ¯2.5
¯1   1.5
```

Since B+.×V1 is 1 0, the first column of B+.×BS is 1 0 and similarly the second column is 0 1. Thus

```
      B+.×BS
 1  0
 0  1
```

Recalling the names VA and VB used in first deriving basic solutions:

```
VA←⌽¯1  1×B[2;]
VB←⌽ 1 ¯1×B[1;]
```

and the fact that V1 and V2 are obtained by dividing these vectors by the determinant of B:

```
V1←VA÷DET B
V2←VB÷DET B
```

Then if M is the matrix whose columns are the vectors VA and VB, it follows that the matrix BS of basic solutions can be obtained from M as follows:

```
BS←M÷DET B
```

The matrix M can be determined as follows. Suppose that the elements of B are called P, Q, R, and S as follows:

```
P  Q
R  S
```

then the first column of M is (S,-R) and the second column is ((-Q),P). Hence M is

```
 S  -Q
-R   P
```

In other words M is obtained from B by simply interchanging the first element of B with the one diagonally opposite, and reversing the signs of the remaining two elements. Finally, the matrix of basic solutions BS is obtained by dividing M by the determinant of B.

To summarize, if B is the matrix

```
P  Q
R  S
```

form the matrix

```
 S  -Q
-R   P
```

and divide it by the determinant (P×S)-(Q×R) to obtain the matrix of basic solutions.

For example:

```
     B          M           DET B        BS
   9  8       ¯6 ¯8         ¯10        ¯.6  .8
   8  6       ¯8  9                     .8 ¯.9

     B+.×BS
 1  0
 0  1
```

⎕16-17

16.9. THE GENERAL SOLUTION FROM THE MATRIX OF BASIC SOLUTIONS

In section 16.4 we saw that the solution of the general linear equation

```
      ∧/N=B+.×V
```

could be obtained from the basic solutions *V1* and *V2* as follows:

```
      V←(N[1]×V1)+(N[2]×V2)
```

This can be written more neatly in terms of the matrix of basic solutions *BS* as follows:

```
      V←BS+.×N
```

For example, if

```
      N←5  6
      V1←2  3
      V2←4  5
```

then *BS* is

```
      2  4
      3  5
```

and

```
      N[1] ×V1
10  15
      N[2] ×V2
24  30
      (N[1] ×V1)+(N[2] ×V2)
34  45
      BS+.×N
34  45
```

⎕18-19

We will now show that *BS+.×N* is equivalent to *(N[1]×V1)+(N[2]×V2)* by showing that each of their two elements agree. Beginning with the first element:

```
 (BS+.×N)[1]
 BS[1;]+.×N                       Definition of inner product
 (BS[1;1]×N[1])+(BS[1;2]×N[2])    Definition of inner product
 (V1[1]×N[1])+(V2[1]×N[2])        Definition of BS
 (N[1]×V1[1])+(N[2]×V2[1])        Commutativity of ×
 ((N[1]×V1)+(N[2]×V2))[1]         Definition of indexing
```

A similar proof applies for the second element.

16.10. THE INVERSE LINEAR FUNCTION

In the preceeding section we saw that if *BS* is the matrix of basic solutions for the matrix *B*, then *BS+.×N* is a solution of the general equation

```
      ∧/N=B+.×V
```

Consequently if *V* is any vector and *N←B+.×V* then *BS+.×N* yields *V*. In other words

```
      BS+.×(B+.×V)
```

yields *V*. Therefore the function *BS+.×V* is the linear function inverse to the function *B+.×V*.

Since the inverse relationship is mutual, the expression

```
      B+.×(BS+.×V)
```

also yields *V*.

⎕20-21

16.11. PROPERTIES OF THE INVERSE LINEAR FUNCTION

As noted in the preceding section

```
      BS+.×(B+.×V)      B+.×(BS+.×V)
      V
```

Since the inner product +.× is associative, it also follows that

```
      (BS+.×B)+.×V
      (B+.×BS)+.×V
      V
```

But the only matrix which multiplied by any vector V yields V is the identity matrix I which has the value

```
      1  0
      0  1
```

Hence

```
      BS+.×B
      B+.×BS
      I
```

It is already clear that B+.×BS yields the identity matrix, since the columns of BS are the basic solutions for B and the columns of B+.×BS are therefore 1 0 and 0 1. The reader may wish to verify that BS+.×B is also equal to the identity matrix for each of the corresponding values of BS and B determined in earlier examples and exercises.

⌹22-23

16.12. ALTERNATIVE DERIVATION OF THE INVERSE LINEAR FUNCTION

The linear function BS+.×V inverse to B+.×V was first determined by computing BS as the matrix of basic solutions for B. The method used applies only for vectors V of dimension 2 and cannot be applied for higher dimensions. We will now develop an alternative method which is somewhat more difficult but which has the important advantage that it applies to higher dimensions.

Since BS+.×V is inverse to B+.×V only if BS+.×B is the identity matrix, we can pose the problem as follows: find a matrix BS such that BS+.×B is the identity matrix. We will determine BS in several steps. Thus if H1 is a matrix such that H1+.×B is "closer" to the identity than B itself, we may find a second matrix H2 such that H2+.×(H1+.×B) is even closer to the identity. Suppose that in four such steps the result

```
      H4+.×(H3+.×(H2+.×(H1+.×B)))
```

is equal to the identity matrix. Then (because +.× is associative):

```
      (H4+.×H3+.×H2+.×H1)+.×B
```

is also equal to the identity matrix. Hence

```
      BS←H4+.×H3+.×H2+.×H1
```

is the required inverse matrix.

For example:

```
      B←2 2 ρ 5 3 4 2
      B
 5  3
 4  2
      H1←2 2 ρ .2 0 0 1
      H1
.2  0
 0  1
      H1+.×B
 1 .6
 4  2
      H2←2 2 ρ 1 0 ¯4 1
      H2
 1  0
¯4  1
      H2+.×(H1+.×B)
 1  .6
 0 ¯.4
      H3←2 2 ρ 1 0 0 ¯2.5
      H3
 1  0
 0 ¯2.5
      H3+.×(H2+.×(H1+.×B))
 1 .6
 0  1
      H4←2 2 ρ 1 ¯.6 0 1
      H4
 1 ¯.6
 0   1
      H4+.×(H3+.×(H2+.×(H1+.×B)))
 1  0
 0  1
      BS←H4+.×H3+.×H2+.×H1
      BS
¯1  1.5
 2 ¯2.5
      BS+.×B
 1  0
 0  1
```

There are a number of points to observe in the foregoing sequence. Each of the matrices H themselves differ from the identity in only one element. H1+.×B is closer to the identity than B in the sense that the first element is 1; thus the first element of H1 was chosen as the reciprocal of 5 so as to divide the first row of B by 5.

The matrix *H*2 was chosen so that the second row of the result would be obtained by adding ¯4 times the first row to the second row, thus making the first element in the second row of the result zero. Thus the element *H*2[2;1] was chosen as -(*H*1+.×*B*)[2;1]. The result *H*2+.×(*H*1+.×*B*) therefore agrees with the identity in the entire first column.

The matrices *H*3 and *H*4 are chosen similarly to make the second column agree; *H*3 multiplies the second row by the reciprocal of the last element of the matrix *H*2+.×*H*1+.×*B*, and *H*4 adds the appropriate multiple of the second row to the first so as to make the upper right element of the result zero.

It will be instructive to repeat the foregoing sequence using a name *BT* for the intermediate results produced so that we write *BT*←*B* and *BT*←*H*1+.×*BT* and *BT*←*H*2+.×*BT*, etc. Moreover, if we first set *BS* to be the identity matrix, and then write *BS*←*H*1+.×*BS* and *BS*←*H*2+.×*BS*, etc., the final value of *BS* will be the required product of the *H* matrices. Thus:

```
      BT←B                                  BS←2 2 ρ 1 0 0 1
      BT                                    BS
 5  3                                 1  0
 4  2                                 0  1

      BT←H1+.×BT                            BS←H1+.×BS
      BT                                    BS
 1 .6                                .2  0
 4  2                                 0  1
      BT       BT←H2+.×BT                   BS←H2+.×BS
      BT                                    BS
1  .6                                .2    0
0 ¯.4                               ¯.8    1
      BT←H3+.×BT                            BS←H3+.×BS
      BT                                    BS
1   .6                               .2    0
0   1                                2  ¯2.5
      BT←H4+.×BT                            BS←H4+.×BS
      BT                                    BS
1   0                               ¯1   1.5
0   1                                2  ¯2.5
      BS+.×B
1   0
0   1
```


Finally, since *BS* and *BT* are subjected to the same sequence of multiplications, we can combine the matrices *BT* and *BS* into a single matrix *M* whose first two columns represent *BT* and whose last two columns represent *BS*. The foregoing computation then appears as follows:

```
      I←2 2 ρ 1 0 0 1
      I
1     0
0     1
      M←B,I
      M
5     3     1     0
4     2     0     1
     M←H1+.×M
      M
1    .6    .2     0
4     2     0     1
      M←H2+.×M
1    .6    .2     0
0   ¯.4   ¯.8     1
      M←H3+.×M
1    .6    .2     0
0     1     2  ¯2.5
      M←H4+.×M
      M
1     0    ¯1   1.5
0     1     2  ¯2.5
```

The last two columns of *M* are the required inverse.

In other words, if we append the identity matrix to the right of *B* and multiply the resulting matrix by any sequence of matrices such that the first two columns become the identity matrix, then the last two columns will be the inverse of the matrix *B*.

It may be noted that each of the matrices *H* were chosen such that each multiplication *H*+.×*M* affected only one row and affected that row in one of two simple ways:

> It multiplied the row by a scalar (chosen so as to make the diagonal element of the row equal to 1
>
> It added to the row some multiple of another row (chosen so as to make one of the elements zero).

We can perform such a sequence of calculations without actually writing out the matrices *H* which produce them. To illustrate this we repeat the preceding example in this form together with notes showing what calculations were performed:

```
      B,I
5    3    1    0
4    2    0    1

1   .6   .2    0         Row 1 is multiplied by ÷5
4    2    0    1

1   .6   .2    0
0  ¯.4  ¯.8    1         ¯4 times row 1 is added to row 2

1   .6   .2    0
0    1    2 ¯2.5         Row 2 is multiplied by ÷¯.4

1    0   ¯1  1.5         ¯.6 times row 2 is added to row 1
0    1    2 ¯2.5
```

The foregoing should be compared carefully with the earlier example which used the matrices *H*1, *H*2, etc. This method for determining the inverse of a matrix is called the Gauss-Jordan method.

⎕24

16.13. EFFICIENT SOLUTION OF A LINEAR EQUATION

A solution to the equation ∧/*N*=*B*+.×*V* can be obtained by determining the matrix *BS* which is inverse to *B* and then computing *V*←*BS*+.×*N* to obtain the solution. A modification of the Gauss-Jordan method can provide the solution more efficiently as follows: apply the Guass-Jordan method to the matrix *B*,*N* instead of to *B*,*I* and the last column of the result will be the desired solution. For example, if *N* is the vector 4 6 and *B* is the matrix of the preceding example, then:

```
      B,N
5    3    4
4    2    6

1   .6   .8
4    2    6

1   .6   .8
0  ¯.4  2.8

1   .6   .8
0    1   ¯7

1    0    5
0    1   ¯7
```

The solution is therefore 5 ¯7. This may be checked as follows:

```
      B+.×5 ¯7
4    6
      N
4    6
```

⎕25

16.14. INVERSE LINEAR FUNCTIONS IN THREE DIMENSIONS

If *V* is a vector of 3 elements and *B* is a 3 by 3 matrix, then *B*+.×*V* is a linear function of *V*. The inverse function *BS*+.×*V* can be determined by the Gauss-Jordan method. The reason it works is the same as in the case of two elements, namely, if *B* is multiplied by a sequence of matrices until the result becomes the identity matrix, then the product of that sequence of matrices is a matrix *BS* such that *BS*+.×*B* is the identity. In other words, *BS* is the inverse of *B*. The Gauss-Jordan method is simply an efficient way of keeping track of the product of the sequence of matrices applied to *B*.

The general scheme is to first reduce the first column to 1 0 0, then reduce the second column to 0 1 0, then the third column to 0 0 1. The first operation for the first column is to divide the first row by its first element. The next is to add a multiple of the first row to the second, and the next is to add a multiple of the first row to the third. On the second column we first divide the second row by its second element and then add multiples of it to rows 1 and 3. On the third column we first divide the third row by its third element and then add multiples to rows 1 and 2. For example:

```
      B←3 3ρ2 1 3 1 0 2 4 0 4
      B
2  1  3
1  0  2
4  0  4
      B,3 3ρ1 0 0 0 1 0 0 0 1
2  1   3   1  0  0
1  0   2   0  1  0
4  0   4   0  0  1

1  .5  1.5  .5  0  0       Multiply row 1 by ÷2
1  0   2    0   1  0
4  0   4    0   0  1

1  .5  1.5  .5  0  0
0 ¯.5  .5  ¯.5  1  0       Add ¯1 times row 1 to row 2
0 ¯2  ¯2   ¯2   0  1       Add ¯4 times row 1 to row 3

1  .5  1.5  .5  0  0
0  1  ¯1    1  ¯2  0       Multiply row 2 by ÷¯.5
0 ¯2  ¯2   ¯2   0  1

1  0   2   0   1  0        Add ¯.5 times row 2 to row 1
0  1  ¯1   1  ¯2  0
0  0  ¯4   0  ¯4  1        Add 2 times row 2 to row 3

1  0   2   0   1  0
0  1  ¯1   1  ¯2  0
0  0   1   0   1 ¯.25      Multiply row 3 by ÷¯4

1  0   0   0  ¯1  .5       Add ¯2 times row 3 to row 1
0  1   0   1  ¯1 ¯.25      Add 1 times row 3 to row 2
0  0   1   0   1 ¯.25
```

The desired inverse is in the last 3 columns, that is:

```
BS←3 3ρ0 ¯1 .5 1 ¯1 ¯.25 0 1 ¯.25
      BS
0 ¯1  .5
1 ¯1 ¯.25
0  1 ¯.25
      BS+.×B
1  0  0
0  1  0
0  0  1
      B+.×BS
1  0  0
0  1  0
0  0  1
```

⌹26-28

16.15. THE INVERSE FUNCTION

We have seen that if *BS*+.×*B* is the identity matrix, then the function *BS*+.×*V* is inverse to the function *B*+.×*V*. For this reason the matrix *BS* is said to be the <u>inverse of the matrix</u> *B*. The inverse of a matrix is an important function which will be assigned the symbol ⌹. Thus if *P*←⌹*Q*, then *P*+.×*Q* and *Q*+.×*P* are both equal to the identity matrix.

Moreover, (⌹*Q*)+.×*N* is the solution of the equation ∧/*N*=*B*+.×*V*. This is easily seen by substituting the solution (⌹*Q*)+.×*N* for *V* obtaining:

```
∧/N=Q+.×(⌹Q)+.×N
∧/N=(Q+.×⌹Q)+.×N          Associativity of +.×
∧/N=I+.×N                 Q+.×⌹Q is the identity I
∧/N=N
1
```

The solution of the equation ∧/*N*=*Q*+.×*V* is also an important function of *N* and *Q* and will be assigned the symbol ⌹ as a dyadic function; that is, *N*⌹*Q* yields the solution of the equation ∧/*N*=*Q*+.×*V*. In other words:

```
N⌹Q
(⌹Q)+.×N
```

⌹29-32

16.16. CURVE FITTING

In Chapter 10, the problem of fitting a function *F* was posed as follows: given a table of a vector of arguments *X* and the corresponding vector *Y*←*F X*, determine a function *E* defined by some expression such that *E X* is equal to *Y*. In Chapter 10 this problem was solved by constructing a difference table and using its first row to determine multipliers of factorial polynomials whose sum became the required expression. This solution applied only to a set of arguments *X* of the form 0,⍳*N*.

In Chapter 11 the method was extended to apply to any set of equally spaced arguments, that is, to any set of arguments *X* of the form *A*+*B*×⍳*N*. Moreover, in Chapter 14 a simpler equivalent expression was found which involved a polynomial rather than the factorial polynomials. However, the method still applied only to equally spaced arguments.

The inverse linear function can now be applied in a simple manner to obtain a solution for any set of arguments

X. We seek a vector of coefficients *C* such that the polynomial *C POL X* is equal to the required set of function values *Y*, that is:

```
∧/Y=C POL X
```

Recalling the definition of the polynomial function from Section 13.6, this requirement may be written as follows:

```
∧/Y=(X∘.*¯1+⍳⍴C)+.×C
```

Furthermore, because *C* must have the same number of elements as *X*, the expression ⍳⍴*C* may be replaced by ⍳⍴*X* so that the outer product in the foregoing expression becomes a function of *X* only. Thus:

```
∧/Y=(X∘.*¯1+⍳⍴X)+.×C
```

This is clearly a linear equation with a given value of *Y*, a given matrix *X*∘.*¯1+⍳⍴*X*, and an argument *C* whose values are to be determined. Hence the required value of *C* is given by the expression:

```
Y⌹(X∘.*¯1+⍳⍴X)
```

For example, if *X*←0 3 4 6 8 (not equally spaced) and if *F* is the function +/(⍳*X*)*3, then *Y* has the value 0 36 100 441 1296, and the square matrix *X*∘.*¯1+⍳⍴*C* has the value:

1	0	0	0	0
1	3	9	27	81
1	4	16	64	256
1	6	36	216	1296
1	8	64	512	4096

The solution may then be obtained by appending the vector *Y* as a final column on this matrix and applying the efficient method of Section 16.13 to the resulting matrix shown below:

1	0	0	0	0	0
1	3	9	27	81	36
1	4	16	64	256	100
1	6	36	216	1296	441
1	8	64	512	4096	1296

The solution is:

```
C←0 0 0.25 0.5 0.25
```

This result may be checked by evaluating the polynomial *C P* 0 3 4 6 8.

Appendix

ALGEBRA AS A LANGUAGE

INTRODUCTION

Although few mathematicians would quarrel with the proposition that the algebraic notation taught in high school is a language (and indeed the primary language of mathematics), yet little attention has been paid to the possible implications of such a view of algebra. This paper adopts this point of view to illuminate the inconsistencies and deficiencies of conventional notation and to explore the implications of analogies between the teaching of natural languages and the teaching of algebra. Based on this analysis it presents a simple and consistent algebraic notation, illustrates its power in the exposition of some familiar topics in algebra, and proposes a basis for an introductory course in algebra. Moreover, it shows how a computer can, if desired, be used in the teaching process, since the language proposed is directly usable on a computer terminal.

ARITHMETIC NOTATION

We will first discuss the notation of arithmetic, i.e., that part of algebraic notation which does not involve the use of variables. For example, the expressions 3-4 and (3+4)-(5+6) are arithmetic expressions, but the expressions 3-*X* and (*X*+4)-(*Y*+6) are not. We will now explore the anomalies of arithmetic notation and the modifications needed to remove them.

<u>Functions and symbols for functions</u>. The importance of introducing the concept of "function" rather early in the mathematical curriculum is now widely recognized. Nevertheless, those functions which the student encounters first are usually referred to not as "functions" but as "operators". For example, absolute value (|-3|) and arithmetic negation (-3) are usually referred to as operators. In fact, most of the functions which are so fundamental and so widely used that they have been assigned some graphic symbol are commonly called operators (particularly those functions such as plus and times which apply to two arguments), whereas the less common functions which are usually referred to by writing out their names (e.g., Sin, Cos, Factorial) are called functions.

This practice of referring to the most common and most elementary functions as operators is surely an unnecessary obstacle to the understanding of functions when that term is first applied to the more complex functions encountered. For this reason the term "function" will be used here for all functions regardless of the choice of symbols used to represent them.

The functions of elementary algebra are of two types, taking either one argument or two. Thus addition is a function of two arguments (denoted by $X+Y$) and negation is a function of one argument (denoted by $-Y$). It would seem both easy and reasonable to adopt one form for each type of function as suggested by the foregoing examples, that is, the symbol for a function of two arguments occurs between its arguments, and the symbol for a function of one argument occurs before its argument. Conventional notation displays considerable anarchy on this point:

1. Certain functions are denoted by any one of several symbols which are supposed to be synonomous but which are, however, used in subtly different ways. For example, in conventional algebra $X\times Y$ and XY both denote the product of X and Y. However, one would write either $3\times Y$ or $3X$ or $X\times 3$, or 3×4, but would not likely accept $X3$ as an expression for $X\times 3$, nor 3 4 as an expression for 3×4. Similarly, $X:Y$ and X/Y are supposed to be synonomous, but in the sentence "Reduce 8/6 to lowest terms", the symbol / does not stand for division.

2. The power function has no symbol, and is denoted by position only, as in X^N. The same notation is often used to denote the Nth element of a family or array X.

3. The remainder function (that is, the integer remainder on dividing X into Y) is used very early in arithmetic (e.g., in factoring) but is commonly not recognized as a function on a par with addition, division, etc., nor assigned a symbol. Because the remainder function has no symbol and is commonly evaluated by the method of long division, there is a tendency to confuse it with division. This confusion is compounded by the fact that the term "quotient" itself is ambiguous, sometimes meaning the quotient and sometimes the integer part of the quotient.

4. The symbol for a function of one argument sometimes occurs before the argument (as in -4) but may also occur after it (as in 4! for factorial 4) or on both sides of it (as in $|X|$ for absolute value of X).

Table 1 shows a set of symbols which can be used in a simple consistent manner to denote the functions mentioned thus far, as well as a few other very useful basic functions such as maximum, minimum, integer part, reciprocal, and exponential. The table shows two uses for each symbol, one to denote a <u>monadic</u> function (i.e., a function of one argument), and one to denote a <u>dyadic</u> function (i.e., a function of two arguments). This is simply a systematic exploitation of the example set by the familiar use of the minus sign, either as a dyadic function (i.e., subtraction as in 4-3) or as a monadic function (i.e., negation as in -3). No function symbol is permitted to be elided; for example, *X*×*Y* may not be written as *XY*.

Monadic form f*B*		f	Dyadic form *A*f*B*	
Definition or example	Name		Name	Definition or example
+3 ↔ 0+3	Plus	+	Plus	2+3.2 ↔ 5.2
-3 ↔ 0-3	Negative	-	Minus	2-3.2 ↔ ¯1.2
×3 ↔ (3>0)-(3<0)	Signum	×	Times	2×3.2 ↔ 6.4
÷3 ↔ 1÷3	Reciprocal	÷	Divide	2÷3.2 ↔0.625
B ⌈*B* ⌊*B* 3.14 4 3 ¯3.14 ¯3 ¯4	Ceiling	⌈	Maximum	3⌈7 ↔ 7
	Floor	⌊	Minimum	3⌊7 ↔ 3
⋆3 ↔ (2.71828∘∘)⋆3	Expon-ential	⋆	Power	2⋆3 ↔ 8
⍟⋆5 ↔ 5 ↔ ⋆⍟5	Natural logarithm	⍟	Loga-rithm	10⍟3 ↔ Log 3 base 10 10⍟3 ↔ (⍟3)÷⍟10
|¯3.14 ↔ 3.14	Magnitude	|	Remain-der	3|8 ↔ 2

TABLE 1

A little experimentation with the notation of Table 1 will show that it can be used to express clearly a number of matters which are awkward or impossible to express in conventional notation. For example, *X*÷*Y* is the quotient of *X* divided by *Y*; either ⌊(*X*÷*Y*) or ((*X*-(*Y*|*X*))÷*Y* yield the integer part of the quotient of *X* divided by *Y*; and *X*⌈(-*X*) is equivalent to |*X*.

In conventional notation the symbols <, ≤, =, ≥, >, and ≠ are used to state relations among quantities; for example, the expression `3<4` asserts that 3 is less than 4. It is more useful to employ them as symbols for dyadic functions defined to yield the value `1` if the indicated relation actually holds, and the value zero if it does not. Thus `3≤4` yields the value `1`, and `5+(3≤4)` yields the value `6`.

Arrays. The ability to refer to collections or arrays of items is an important element in any natural language and is equally important in mathematics. The notation of vector algebra embodies the use of arrays (vectors, matrices, 3-dimensional arrays, etc.) but in a manner which is difficult to learn and limited primarily to the treatment of linear functions. Arrays are not normally included in elementary algebra, probably because they are thought to be difficult to learn and not relevant to elementary topics.

A vector (that is, a 1-dimensional array) can be represented by a list of its elements (e.g., `1 3 5 7`) and all functions can be assumed to be applied element-by-element. For example:

```
      1 2 3 4 × 4 3 2 1   produces
4 6 6 4
```

Similarly:

```
      1 2 3 4 + 4 3 2 1
5 5 5 5
      ! 1 2 3 4
1 2 6 24
      1 2 3 4 * 2
1 4 9 16
      2 * 1 2 3 4
2 4 8 16
```

In addition to applying a function to each element of an array, it is also necessary to be able to apply some specified function to the collection itself. For example, "Take the sum of all elements", or "Take the product of all elements", or "Take the maximum of all elements". This can be denoted as follows:

```
      +/2 5 3 2
12
      ×/2 5 3 2
60
      ⌈/2 5 3 2
5
```


The rules for using such vectors are simple and obvious from the foregoing examples. Vectors are relevant to elementary mathematics in a variety of ways. For example:

1. They can be used (as in the foregoing examples) to display the patterns produced by various functions when applied to certain patterns of arguments.

2. They can be used to represent points in coordinate geometry. Thus `5 7 19` and `2 3 7` represent two points, `5 7 19 - 2 3 7` yields `3 4 12`, the displacement between them, and `(+/(5 7 19 - 2 3 7)*2)*.5` yields `13`, the distance between them.

3. They can be used to represent rational numbers. Thus if `3 4` represents the fraction three-fourths, then `3 4 × 5 6` yields `15 24`, the product of the fractions represented by `3 4` and `5 6`. Moreover, `÷/3 4` and `÷/5 6` and `÷/15 24` yield the actual numbers represented.

4. A polynomial can be represented by its vector of coefficients and vector of exponents. For example, the polynomial with coefficients `3 1 2 4` and exponents `0 1 2 3` can be evaluated for the argument `5` by the following expression:

```
      +/3 1 2 4 × 5 * 0 1 2 3
558
```

Constants. Conventional notation provides means for writing any positive constant (e.g., `17` or `3.14`) but there is no distinct notation for negative constants, since the symbol `-` occurring in a number like `-35` is indistinguishable from the symbol for the negation function. Thus negative thirty-five is written as an expression, which is much as if we neglected to have symbols for five and zero because expressions for them could be written in a variety of ways such as `8-3` and `8-8`.

It seems advisable to follow Beberman [1] in using a raised minus sign to denote negative numbers. For example:

```
      3 - 5 4 3 2 1
¯2 ¯1 0 1 2
```

Conventional notation also provides no convenient way to represent numbers which are easily expressed in expressions of the form 2.14×10^{8} or $3.265\times10^{¯9}$. A useful practice widely used in computer languages is to replace the symbols `×10` by the symbol *E* (for exponent) as follows: `2.14E8` and `3.265E¯9`.

Order of execution. The order of execution in an algebraic expression is commonly specified by parentheses. The rules for parentheses are very simple, but the rules which apply in the absence of parentheses are complex and chaotic. They are based primarily on a hierarchy of functions (e.g., the power function is executed before multiplication, which is executed before addition) which has apparently arisen because of its convenience in writing polynomials.

Viewed as a matter of language, the only purpose of such rules is the potential economy in the use of parentheses and the consequent gain in readability of complex expressions. Economy and simplicity can be achieved by the following rule: parentheses are obeyed as usual and otherwise expressions are evaluated from right to left with all functions being treated equally. The advantages of this rule and the complexity and ambiguity of conventional rules are discussed in Berry [2], page 27 and in Iverson [3], Appendix A. Even polynomials can be conveniently written without parentheses if use is made of vectors. For example, the polynomial in X with coefficients 3 1 2 4 can be written without parentheses as +/3 1 2 4 × X ⋆ 0 1 2 3. Moreover, Horner's expression for the efficient evaluation of this same polynomial can also be written without parentheses as follows:

```
3+X×1+X×2+X×4
```

__Analogies with Natural Languange__. The arithmetic expression 3×4 can be viewed as an order to __do__ something, that is, multiply the arguments 3 and 4. Similarly, a more complex expression can be viewed as an order to perform a number of operations in a specified order. In this sense, an arithmetic expression is an imperative sentence, and a function corresponds to an imperative verb in natural language. Indeed, the word "function" derives from the latin verb "fungi" meaning "to perform".

This view of a function does not conflict with the usual mathematical definition as a specified correspondence between the elements of domain and range, but rather supplements this static view with a dynamic view of a function as that which __produces__ the corresponding value for any specified element of the domain.

If functions correspond to imperative verbs, then their arguments (the things upon which they act) correspond to nouns. In fact, the word "argument" has (or at least had) the meaning topic, theme, or subject. Moreover, the positive integers, being the most concrete of arithmetical objects, may be said to correspond to proper nouns.

What are the roles of negative numbers, rational numbers, irrational numbers, and complex numbers? The subtraction function, introduced as an inverse to addition, yields positive integers in some cases but not in others, and negative numbers are introduced to refer to the results in these cases. In other words, a negative number refers to a process or the result of a process, and is therefore analogous to an abstract noun. For example, the abstract noun "justice" refers not to some concrete object (examples of which one may point to) but to a process or result of a process. Similarly, rational and complex numbers refer to the results of processes; division, and finding the zeros of polynomials, respectively.

ALGEBRAIC NOTATION

__Names.__ An expression such as 3×X can be evaluated only if the variable X has been assigned an actual value. In one sense, therefore, a variable corresponds to a __pronoun__ whose referent must be made clear before any sentence including it can be fully understood. In English the referent may be made clear by an explicit statement, but is more often made clear by indirection (e.g., "See the door. Close it."), or by context.

In conventional algebra, the value assigned to a variable name is usually made clear informally by some statement such as "Let X have the value 6" or "Let X=6". Since the equal symbol (that is, '=') is also used in other ways, it is better to avoid its use for this purpose and to use a distinct symbol as follows:

```
      X←6
      Y←3×4
      X+Y
18
      (X-3)×(X-5)
3
```

__Assigning Names to Expressions__. In the foregoing example, the expression (X-3)×(X-5) was written as an instruction to evaluate the expression for a particular value already assigned to X. One also writes the same expression for the quite different notion "Consider the expression (X-3)×(X-5) for any value which might later be assigned to the argument X." This is a distinct notion which should be represented by distinct notation. The idea is to be able to refer to the expression and this can be done by assigning a name to it. The following notation serves:

```
      ∇ Z ← G X
      Z←(X-3)×(X-5)∇
```

The ∇'s indicate that the symbols between them define a function; the first line shows that the name of the function is *G*. The names *X* and *Z* are dummy names standing for the argument and result, and the second line shows how they are related.

Following this definition, the name *G* may be used as a function. For example:

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

Iterative functions can be defined with equal ease (Iverson [3]) but the mechanics will not be discussed here.

<u>Form of Names</u>. If the variables occurring in algebraic sentences are viewed simply as names, it seems reasonable to employ names with some mnemonic significance as illustrated by the following sequence:

```
      LENGTH←6
      WIDTH←5
      AREA←LENGTH×WIDTH
      HEIGHT←4
      VOLUME←AREA×HEIGHT
```

This is not done in conventional notation, apparently because it is ruled out by the convention that the multiplication sign may be elided; that is, *AREA* cannot be used as a name because it would be interpreted as $A \times R \times E \times A$.

This same convention leads to other anomalies as well, some of which were discussed in the section on arithmetic notation. The proposal made there (i.e., that the multiplication sign cannot be elided) will permit variable names of any length.

ANALOGIES WITH THE TEACHING OF NATURAL LANGUAGE

If one views the teaching of algebra as the teaching of a language, it appears remarkable how little attention is given to the reading and writing of algebraic sentences, and how much attention is given to identities, that is, to the analysis of sentences with a view to determining other equivalent sentences; e.g., "Simplify the expression $(X-4) \times (X+4)$." It is possible that this emphasis accounts for much of the difficulty in teaching algebra, and that the teaching and learning processes in natural languages may suggest a more effective approach.

In the learning of a native language one can distinguish the following major phases:

1. An informal phase, in which the child learns to communicate in a combination of gestures, single words, etc., but with no attempt to form grammatical sentences.

2. A formal phase, in which the child learns to communicate in formal sentences. This phase is essential because it is difficult or impossible to communicate complex matters with precision without imposing some formal structure on the language.

3. An analytic phase, in which one learns to analyze sentences with a view to determining equivalent (and perhaps "simpler" or "more effective") sentences. The extreme case of such analysis is Aristotelian Logic, which attempts a formal analysis of certain classes of sentences. More practical everyday cases occur every time one carefully reads a composition and suggests alternative sentences which convey the same meaning in a briefer or simpler form.

The same phases can be distinguished in the teaching of algebraic notation:

1. An informal phase in which one issues an instruction to add 2 and 3 in any way which will be understood. For example:

```
2+3          Add 2 and 3

 2             2
 3            +3
```

Add two and three

Add // and ///

The form of the expression is unimportant, provided that the instruction is understood.

2. A formal phase in which one emphasizes proper sentence structure and would not accept expressions such as $6 \times \begin{matrix}2\\3\end{matrix}$ or 6×(add two and three) in lieu of 6×(2+3). Again, adherence to certain structural rules is necessary to permit the precise communication of complex matters.

3. An analytic phase in which one learns to analyze sentences with a view to establishing certain relations (usually identity) among them. Thus one

learns not only that 3+4 is equal to 4+3 but that the sentences *X*+*Y* and *Y*+*X* are equivalent, that is, yield the same result whatever the meanings assigned to the pronouns *X* and *Y*.

In learning a native language, a child spends many years in the informal and formal phases (both in and out of school) before facing the analytic phase. By this time she has easy familiarity with the purposes of a language and the meanings of sentences which might be analyzed and transformed. The situation is quite different in most conventional courses in algebra - very little time is spent in the formal phase (reading, writing and "understanding" formal algebraic sentences) before attacking identities (such as commutativity, associativity, distributivity, etc.). Indeed, students often do not realize that they might quickly check their work in "simplification" by substituting certain values for the variables occurring in the original and derived expressions and comparing the evaluated results to see if the expressions have the same "meaning", at least for the chosen values of the variables.

It is interesting to speculate on what would happen if a native language were taught in an analogous way, that is, if children were forced to analyze sentences at a stage in their development when their grasp of the purpose and meaning of sentences were as shaky as the algebra student's grasp of the purpose and meaning of algebraic sentences. Perhaps they would fail to learn to converse, just as many students fail to learn the much simpler task of reading.

Another interesting aspect of learning the non-analytic aspects of a native language is that much (if not most) of the motivation comes not from an interest in language, but from the intrinsic interest of the material (in children's stories, everyday dialogue, etc.) for which it is used. It is doubtful that the same is true in algebra - ruling out statements of an analytic nature (identities, etc.), how many "interesting" algebraic sentences does a student encounter?

The use of arrays can open up the possibility of much more interesting algebraic sentences. This can apply both to sentences to be read (that is, evaluated) and written by students. For example, the statements:

```
2*1 2 3 4 5
2×1 2 3 4 5
2÷1 2 3 4 5
1 2 3 4 5÷2
1 2 3 4 5*2
1 2 3 4 5×5 4 3 2 1
```

produce interesting patterns and therefore have more intrinsic interest than similar expressions involving only

single quantities. For example, the last expression can be construed as yielding a set of possible areas for a rectangle having a fixed perimeter of 12.

More interesting possibilities are opened up by certain simple extensions of the use of arrays. One example of such extensions will be treated here. This extension allows one to apply any dyadic function to two vectors *A* and *B* so as to obtain not simply the element-by-element product produced by the expression *A*×*B*, but a table of all products produced by pairing each element of *A* with each element of *B*. For example:

```
A←1 2 3
B←2 3 5 7

A∘.×B              A∘.+B              A∘.*B

 2  3  5  7         3  4  6  8         1  1   1    1
 4  6 10 14         4  5  7  9         4  8  32  128
 6  9 15 21         5  6  8 10         9 27 243 2187
```

If *S*←1 2 3 4 5 6 7, then the following expressions yield an addition table, a multiplication table, a subtraction table, a maximum table, an "equal" table, and a "greater than or equal" table:

```
S∘.+S                         S∘.⌈S
 2  3  4  5  6  7  8          1 2 3 4 5 6 7
 3  4  5  6  7  8  9          2 2 3 4 5 6 7
 4  5  6  7  8  9 10          3 3 3 4 5 6 7
 5  6  7  8  9 10 11          4 4 4 4 5 6 7
 6  7  8  9 10 11 12          5 5 5 5 5 6 7
 7  8  9 10 11 12 13          6 6 6 6 6 6 7
 8  9 10 11 12 13 14          7 7 7 7 7 7 7

S∘.×X                         S∘.=S
 1  2  3  4  5  6  7          1 0 0 0 0 0 0
 2  4  6  8 10 12 14          0 1 0 0 0 0 0
 3  6  9 12 15 18 21          0 0 1 0 0 0 0
 4  8 12 16 20 24 28          0 0 0 1 0 0 0
 5 10 15 20 25 30 35          0 0 0 0 1 0 0
 6 12 18 24 30 36 42          0 0 0 0 0 1 0
 7 14 21 28 35 42 49          0 0 0 0 0 0 1

S∘.-S                         S∘.≥S
 0 ¯1 ¯2 ¯3 ¯4 ¯5 ¯6          1 0 0 0 0 0 0
 1  0 ¯1 ¯2 ¯3 ¯4 ¯5          1 1 0 0 0 0 0
 2  1  0 ¯1 ¯2 ¯3 ¯4          1 1 1 0 0 0 0
 3  2  1  0 ¯1 ¯2 ¯3          1 1 1 1 0 0 0
 4  3  2  1  0 ¯1 ¯2          1 1 1 1 1 0 0
 5  4  3  2  1  0 ¯1          1 1 1 1 1 1 0
 6  5  4  3  2  1  0          1 1 1 1 1 1 1
```

Moreover, the graph of a function can be produced as an "equal" table as follows. First recall the function *G* defined earlier:

```
      ∇Z←G X
       Z←(X-3)×(X-5)∇

      G S
8  3  0  ¯1  0  3  8
```

The range of the function for this set of arguments is from 8 down to ¯1, and the elements of this range are all contained in the following vector:

```
      R←8 7 6 5 4 3 2 1 0 ¯1
```

Consequently, the "equal" table *R*∘.=*G S* produces a rough graph of the function (represented by 1's) as follows:

```
      R∘.=G S
 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
```

A PROGRAM FOR ELEMENTARY ALGEBRA

The foregoing analysis suggests the development of an algebra curriculum with the following characteristics:

1. The notation used is unambigious, with simple and consistent rules of syntax, and with provision for the simple and direct use of arrays. Moreover, the notation is not taught as a separate matter, but is introduced as needed in conjunction with the concepts represented.

2. Heavy use is made of arrays to display mathematical properties of functions in terms of patterns observed in vectors and matrices (tables), and to make possible the reading, writing, and evaluation of a host of interesting algebraic sentences before approaching the analysis of sentences and the concomitant development of identities.

Such an approach has been adopted in Iverson [4], where it has been carried through as far as the treatment of polynomials and of linear functions and linear equations. The extension to further work in polynomials, to slopes and derivatives, and to the circular and hyperbolic functions is carried forward in Chapters 4-8 of Iverson [3].

It must be emphasized that the proposed notation, though simple, is not limited in application to elementary algebra. A glance at the bibliography of Falkoff and Iverson [5] will give some idea of the wide range of applicability.

<u>The Role of the Computer</u>. Because the proposed notation is simple and systematic it can be executed by automatic computers and has been made available on a number of time-shared terminal systems. The most widely used of these is described in Falkoff and Iverson [6], IBM Corporation, 1968. It is important to note that the notation is executed directly, and the user need learn nothing about the computer itself. In fact, each of the examples in the present paper are shown exactly as they would be typed on a computer terminal keyboard.

The computer can obviously be useful in cases where a good deal of tedious computation is required, but it can be useful in other ways as well. For example, it can be used by a student to explore the behavior of functions and discover their properties. To do this a student will simply enter expressions which apply the functions to various arguments. If the terminal is equipped with a display device, then such exploration can even be done collectively by an entire class. This and other ways of using the computer are discussed in Berry et al [7].

ACKNOWLEDGMENTS

I am indebted to my colleagues at the Philadelphia Scientific Center for many fruitful discussions and suggestions, particularly to Messrs. Adin Falkoff and Paul Berry.

REFERENCES

1. Beberman, M., and H. E. Vaughan, High School Mathematics Course 1, Heath, 1964.

2. Berry, P. C., APL\360 Primer, IBM Corp., 1969.

3. Iverson, K. E., Elementary Functions: an algorithmic treatment, Science Research Associates, 1966.

4. Iverson, K. E., "Elementary Algebra", Report of the IBM Philadelphia Scientific Center.

5. Falkoff, A. D., and K. E. Iverson, "APL Bibliography", Proceedings of IEEE NEREM Conference, November 1970.

6. Falkoff, A. D., and K. E. Iverson, APL\360 User's Manual, IBM Corp., 1968.

7. Berry, P. C., A. D. Falkoff, and K. E. Iverson "Using the Computer to Compute: A Direct but Neglected Approach to Teaching Mathmatics", IFIP World Conference on Computer Education, Amsterdam, August 24-28, 1970.

EXERCISES

CHAPTER 1

1.1 Evaluate the following expressions, entering the result in the position indicated by the underscore:

```
      (3+4)×6
___
      3+(4×6)
___
      3+(4+6)
___
      (3+4)+6
___
      3×(4×6)
___
      (3×4)×6
___
      (3+5)×(6+4)
___
      (9+19)×(42+8)
___
      (18+10)+5
___
      (16×13)+49
___
      49+(16×13)
___
      3×((5×6)+4)
___
      (3×(5×6))+4
___
      ((2+3)×(4+6))+(2×5)
___
      1+(2×(3+(4×(5+6))))
___
      ((((1+2)×3)+4)×5)+6
___
```

1.2 Check your answers to Exercise 1.1 and repeat each one which is incorrect, filling in the steps of the evaluation in the manner shown in the text. For example, the last exercise would appear as follows:

```
      ((((1+2)×3)+4)×5)+6
      (((   3   ×3)+4)×5)+6
      ((        9  )+4)×5)+6
      (            13   ×5)+6
                        65 +6
71
```

1.3 Enter numbers in the underscored positions such that each expression gives the indicated result:

```
      (3+___)×6
42
      3+(___×6)
27
      (7+___)×3
30
      (7+3)×___
30
      (42+___)×4
200
      (___+6)+4
17
      (2×___)+19
49
      ___+(45×6)
274
      (4+___)×(5+6)
77
      (3+(4×(___+2))+7
38
      (2×(((3+___)+(2×2))+5))+3
33
```

1.4 Check your answers for Exercise 1.3. For each one that is incorrect, show every step of the evaluation using the number that you entered in the underscored position.

1.5 Write an equivalent algebraic expression for each of the following sentences:

Quantity 7 plus 1 multiplied by 3.

17 added to the product of 6 and 2.

5 times the quantity 17×6.

Add the quantity 3+2 to the product of 8 and 5.

The product of the quantities 6+10 and 7+3.

The sum of 4 and 14 added to the product of 3 and 13.

29 plus the product of 19 and 6.

Quantity 9+20 added to the sum of 7 and 6.

Increase the quantity 8×3 by 7.

Add 15 to the sum of 14 and 8.

Multiply 6 times itself and then add 3.

Quantity 1+2+3 times 8.

The product of 3+4 and 8.

2 plus twice the quantity 9+5.

Six more than the product of 2 and 8.

1.6 Write an equivalent English expression for each algebraic expression in Exercise 1.1.

1.7 Evaluate the following expressions:

```
      2×3+4
___   2+3×4
___   1+2×3+4×5
___   1+(2×3)+4
___   1+(2×3)+4×5
___   (2+9+20)×16
___   14×15×13+6+20
___   2×10+10
___   9×(2+7)×3
___   23+7×2+1
___   1+(9×11)+11×1
___   1+(2×3+4)×5+6
___   1+(2×3)+(4×5)+6
___
```

1.8 For each wrong answer obtained in Exercise 1.7 take the given expression and modify it by inserting all of the parentheses implied by the rule to evaluate the rightmost function first. Then evaluate the resulting expression. For example:

```
      1+(2×3)+4×5
      1+((2×3)+(4×5))
27
```


1.9 Enter a number in each of the underscored positions such that the expression gives the indicated result:

```
      2+___×3+5
162
      2+(___×3)+5
67
      2×(3+___)×5+3
144
      2×(___+3)×5+3
144
      10+6×4+___×2
130
      10+(6×4)+___×2
130
      10×25+___+45
800
      ___×9×3×1×7
9072
      (___+40+10)×2
118
      10+17+___×17×5
197
      43+9×6+___
160
```

1.10 For each wrong answer obtained in Exercise 1.9, fill into the given expression your answer and all of the implied parentheses and then evaluate the resulting expression.

1.11 Using as few parentheses as possible, write algebraic expressions for each of the English expressions of Exercise 1.5.

1.12 Write equivalent English expressions for each of the expressions of Exercise 1.7.

1.13 Evaluate each of the indicated expressions:

```
      A←2
      B←3
      A+B
___   A×B
___   A+3
___   4×B+8×A
___   (10+B)×A
___   P←9
      B←2
      A+P×3
___   (B+B)×P
___   A+B+B+B
___   A+(3×B)
___   A+3×B
___   A+(P+7)
___   SPEED←60
      TIME←5
      DISTANCE←SPEED×TIME
      DISTANCE
___   SPEED×7
___   SPEED←40
      SPEED×TIME
___   3×(4×A)
___   (4×A)×3
___   (A×4)×3
___   A×4×3
___   CAT←1
      KITTENS←4
      TOTAL←CAT+KITTENS
      TOTAL
___   NEWKITTENS←KITTENS×5
      TOTAL←TOTAL+NEWKITTENS
      TOTAL
___   2×TOTAL+(4×7)
___   (5+(CAT×TIME)+3)×3
___   CAT×CAT+5
___
```

```
      CAT
____
      17+(17+TOTAL)×2
____
      T←4
      V←7
      V×(T+3)
____
      (T+3)×V
____
      (T×V)+(3×V)
____
      (V×T)+(V×3)
____
      V×T+V×3
____
      DO←3
      DO+6×7
____
      3+DO×4+5
____
      DO
____
      X←3
      X×X
____
      X←5
      X×X
____
```

1.14 For each wrong answer in Exercise 1.13, repeat the work showing every step of the evaluation.

1.15 Fill in the underscored positions so that the expressions give the indicated result:

```
      WIDTH←9
      (___+WIDTH)×3
93
      8+(___×WIDTH)
44
      LEN←___
      WIDTH+LEN
13
      (LEN×3)+(WIDTH+___)
22
      10+LEN×___
18
      HEIGHT←5
      20+HEIGHT+___
37
      VOLUME←LEN×WIDTH×HEIGHT
      ___×VOLUME
360
      (LEN+VOLUME)+___
190
      (3+4+LEN)×___
55
      (3+4)+(LEN×___)
55
```

1.16 For each wrong answer in Exercise 1.15, write in your answer and every step in the evaluation of the expression.

1.17 Translate each of the following sentences into a sequence of algebraic expressions:

The length of a playing field is 100 yards. Its width is 50 yards. The area is the length times the width.

A weightlifter has a steel bar weighing 20 lbs. He also has two weights, each weighing 50 lbs. The total weight that he will be lifting is the sum of the bar and the two weights.

A triangle has three sides. Side a is 3 inches long, side b is 4 inches, and side c is 5 inches long. The perimeter of the triangle is the sum of the lengths of the sides.

A nickel has a value of five cents. A dime is worth ten cents. A quarter is worth two dimes and one nickel.

An airplane is flying directly east with a heading of 90 degrees. He turns right 30 degrees. The new heading will be the sum of the old heading and the amount that the plane turned.

On a trip across the country, the Smiths travelled for six days, covering 500 miles each day. The total distance travelled is the daily mileage times the number of days in transit.

John weighed 100 lbs. He then ate three pieces of steak, each weighing 1 lb. His new weight is the sum of his old weight and all that he ate.

1.18 Make up "word problems" to correspond to each of the following groups of algebraic sentences:

```
      X←100
      Y←50
      X×Y
5000
      INCHES←7
      FEET←2
      YARDS←4
      (YARDS×36)+(FEET×12)+INCHES
175
```

1.19 Evaluate the following expressions:

```
      +/9 7 19 19
____
      ×/4 2 1 6 3
____
      ×/20 5 7
____
      18+(×/20 3 1)
____
      (×/2 4)+39
____
      (+/10 20)×3
____
      +/43 7 19 21 28
____
      +/16 15 50 36
____
      +/30 4
____
      3+3+3
____
      3+3
____
      3
____
      +/3 3 3
____
      +/3 3
____
      +/3
____
      +/10 19
____
      +/30 7 45
____
      (+/3 4 1)×7
____
      +/7
____
      ×/8 3 7
____
      ABC←1 3 5
      DE←2 4 6 8 10
      +/ABC
____
      ×/DE
____
      ×/ABC
____
      ABC
____
      +/DE
____
      3++/ABC
____
      2×2×2
____
      2×2
____
      2
____
      ×/2 2 2
____
      ×/2 2
____
      ×/2
____
      ×/19 19 5
____
      +/9 10 1
____
      7+×/3 5 7
____
```

```
×/3
(+/9 43 46 4)+7
×/13 5
E←2
E+×/ABC
(+/DF)×E
E←3+×/ABC
E+(+/ABC)+(+/DE)
(E×3)++/ABC
+/ABC++/DE
+/5
+/9 26 42 15
×/2 6 9 27 19
(×/12 49 45)×8
+/15 34 14
×/9
```

1.20 Use the over notation to write an equivalent algebraic expression for each of the following sentences:

Plus over 4 6 8 9.

Times over 2 4 6.

The sum over 20 15 4.

6 plus the product over 4 1 2.

2 plus the sum over 3 12 4 20.

The product of 3 and 7.

Ten times the product over 8 3.

Four plus 3 plus 7.

Three times the sum over 1 2 3 4 5 6.

Six times seven times one times three.

Quantity 4+3 times the sum over 20 17 4 7.

The sum of 3 4 and 5, all times the product over 2 8 3 4.

1.21 Write an equivalent English expression for each of the first 10 expressions in Exercise 1.19.

1.22 Evaluate the following expressions:

```
ι4
+/ι4
×/ι4
ι5
+/ι5
×/ι5
ι1
+/ι1
N←3
+/ιN
+/ιN+1
+/ιN+2
+/ι2×N
```


1.23 Fill in the underscored position so that each of the expressions give the indicated results:

```
ι___
1 2 3 4
+/ι___
10
+/ι___
15
+/ι___
55
×/ι___
24
×/ι3+___
720
+/ι4+___
78
N←___
×/ιN
120
+/ι___
1
×/ι___
1
×/ι___
3628800
```

1.24 Write an equivalent algebraic expression for each of the following sentences:

The first three integers.

Iota 5.

The integers to nine.

The sum of the first three integers.

Times over the integers to 4.

Plus over the integers to 7.

Q is assigned the value 4.

The integers to Q.

The one digit integers.

1.25 Write an equivalent English expression for each of the expressions of Exercise 1.22.

1.26 Evaluate the following expressions:

```
3 5 7 4 + 6 2 9 15
4 3 2 1 + 1 2 3 4
3 5 7 9 + ι4
+/3 5 7 9 + ι4
3 5 7 9 + 3
3 + 3 5 7 9
3 + ι4
5×ι4
3+5×ι4
(ι4)×(ι4)
+/(ι4)×(ι4)
N←3 5 7 9
M←4
N+M
N+ιM
N×N
M+M×ιM
+/3×ι6
3×+/ι6
3×4+ι5
12+3×ι5
```

1.27 Fill in the underscored positions so that each of the expressions give the indicated result (Note that each entry may be either a vector or a scalar):

```
      2 3 5 7 +___
5 10 6 10
      2 3 5 7 +___
6 7 9 11
      2 8 1 6 ×___
6 32 4 30
      2 8 1 6 ×___
10 40 5 30
      ___+ι4
6 7 8 9
      ___×ι4
20 40 60 80
      ___+ι___
8 9 10 11 12
      ___×ι___
7 14 21 28 35
      ___+___×ι___
8 13 18 23 28 33
      3×___+ι___
18 21 24 27 30 33
      ___×___+ι___
16 20 24 28 32
      ___+___×ι___
16 20 24 28 32
```

1.28 Write an equivalent algebraic expression for each English expression:

The first five integers following 4.

Every third integer beginning with 3 and ending with 21.

Every third integer beginning with 7 and ending with 31.

1.29 Evaluate the following expressions:

```
      3ρ5
___
      +/3ρ5
___
      3×5
      ×/2ρ4
___
      ×/10ρ4
___
      (4ρ1)+2 3 5 7
___
       1+2 3 5 7
___
      (4ρ2)×ι4
___
      ×/9ρ10
___
      4××/3ρ7
___
      3++/3ρ7
___
      16 9 13 10++/4ρ4
___
```

1.30 Fill in the blanks so that the expressions give the printed result:

```
      +/4ρ___
12
      8ρ___
8 8 8 8 8 8 8 8
      +/8ρ___
64
      6ρ___
6 6 6 6 6 6
      ×/___ρ3
243
      ×/5ρ___
100000
      2ρ___
3 3
      +/___ρ10
80
      +/___ρ4
28
      +/___ρ6
60
      10ρ___
2 2 2 2 2 2 2 2 2 2
      ×/3ρ___
343
      ___ρ7
7 7 7 7 7 7 7 7 7
      8ρ___
5 5 5 5 5 5 5 5
      ___ρ2
2 2 2 2 2 2 2 2
```



```
      ×/9ρ___
134217728
      ___ρ5
5 5 5 5 5 5
      2ρ___
10 10
      ___ρ3
3 3 3 3 3 3 3 3 3 3
      ×/___ρ4
262144
      1ρ___
1
      7ρ___
3 3 3 3 3 3 3
      ___ρ7
7 7 7 7 7 7 7 7
      +/4ρ___
12
      ×/___ρ1
1
      +/___ρ5
40
      ___ρ3
3 3 3 3
      ___ρ9
9 9 9
      ___ρ8
8 8 8 8 8 8
      +/___ρ5
5
```

1.31 Write an equivalent algebraic expression for each of the following sentences:

Three repetitions of 5.

5 repetitions of 3.

Plus over 6 repetitions of 4.

The product of 3 repetitions of 7.

Seven repetitions of six.

The sum of ten repetitions of four.

Times over vector 3 6 plus 2 repetitions of 5.

Vector 5 7 9 times 3 repetitions of 1.

4 repetitions of 7 plus 4 repetitions of 3.

3 times 6 repetitions of 5.

1.32 Evaluate the following expressions:

```
      N←2 3 5 7
      M←8 7 6 5
      M+N
___
      M+N×M
___
      (M+4)×M
___
      (M+N)×M
___
      (M+ι4)×N
___
      ((3×M)+(2×N))×2
___
      +/3×M
___
      3×+/M
___
      +/M×N
___
       M×+/N
___
      (+/M)×N
___
      (+/M)×+/N
___
      ×/M+N
___
      (×/M)+N
___
      (×/M)+×/N
___
      ι+/N
___
      +/ι+/N
___
      +/ι+/ι3
___
      +/ι+/3ρ2
___
      ×/ι×/2ρ3
```

2⌈3

_____ M⌈N

_____ ⌈/M

_____ ⌈/N

_____ ⌈/M+N

_____ ⌈/M×N

_____ (⌈/M)+⌈/N

_____ (⌈/M)×N

_____ +/M⌈N

_____ ×/M⌈N

_____ 4⌈N

_____ +/4⌈N

_____ ×/4⌈N

_____

CHAPTER 2

2.1 Use Table 2.1 to evaluate the function "normal weight" for the following arguments:

59 63 69 60

2.2 We will use the term "two times" for a function whose result is twice the argument. Thus a table for this function for the arguments ⍳4 would appear as follows:

Argument	Result
1	2
2	4
3	6
4	8

a) Make a table for the "two times" function for the same set of a⁻guments as used in Table 2.1.

b) Is the "two times" function a good approximation to the "normal weight" function of Table 2.1?
Over what set of arguments do the two functions differ by not more than 2?

c) One could add a certain "correction" to each result of the "two times" function to obtain the exact normal weight. For example, for the argument 63 the value of "two times" is 126 and a correction of 4 is needed to give the actual normal weight of 130. Make a table to represent an appropriate "correction" function for the arguments from 60 to 70.

2.3 Evaluate the function represented by Table 2.2 for each of the following cases:

61 inches medium frame
58 inches large frame
63 inches small frame
65 inches all frames
68 inches small and large

2.4 Use the information in Table 2.2 to make tables to represent each of the following functions:

a) Normal weights for large frame and heights 60 to 66.

b) Normal weights for all frames and heights 66 to 70.

c) Normal weights for small frame and for even numbered heights from 58 to 68, that is, for heights 56+2×⍳6.

d) Normal weights for height 67 and all frames.

2.5 a) Extend the table of Figure 2.3 to include arguments up to 12 (for both arguments).

b) Circle the result in the table which results from the expression 6×8.

c) Underscore the result of the expression 8×6.

d) Pick out all occurrences of the number 40 in your table and label each with a different letter of the alphabet. Then write these letters in a column and beside each write the expression (e.g., 5×8) which corresponds to that particular entry in the table.

e) Repeat part (d) for the number 24.

2.6 a) Construct an addition table for the arguments 1 to 12.

b) Label each occurrence of the result 9 in the table with a different letter. Then list the letters and show with each the expression which corresponds to that entry.

c) Repeat part (b) for the number 20.

2.7 Let *X* denote the domain of the first argument of the multiplication table of Figure 2.3 (that is, *X*←⍳8), and let *Y* denote the domain of the second argument (that is, *Y*←⍳10). Then the function represented by the third row of the body of Figure 2.3 can also be represented as 3×*Y*, and the function represented by the fourth column can be represented as *X*×4. Use this scheme to write expressions which represent each of the functions represented by the following parts of the body of Table 2.3:

a) Row 2.
b) Column 10.
c) Row 5.
d) Column 5.

2.8 Make a table whose body consists of one column taken from the 8th column of the body of the multiplication table of Figure 2.3, and whose first column (that is, the arguments lying outside the body) is taken from the second column of the body of Figure 2.3. Call the function represented by this table *F*.

a) Evaluate the function *F* for the arguments 4, 6, and 10.

b) What is the domain of *F*?

c) What is the range of *F*?

d) Write an expression (in the manner of Exercise 2.7) which represents *F*.

2.9 Repeat Exercise 2.8 using row 9 of the body of Figure 2.3 as the one-column body of the table, and row 3 as the arguments. If any of the arguments in part (a) do not lie in the domain of this function, indicate that they cannot be evaluated.

2.10 Repeat Exercise 2.9 using rows from the addition table constructed in Exercise 2.6 rather than Figure 2.3.

2.11 (Parts a-i)Answer the nine questions posed in Section 2.2.

2.12 Let *A*←1 2 3 4 and *B*←1 2 3 4 5. Then evaluate the following expressions:

a) *A*∘.×*B*

b) *A*∘.+*B*

c) *B*∘.×*A*

d) *B*∘.+*A*

e) *B*∘.×*B*

f) *A*∘.+*A*

2.13 Evaluate the following expressions:

a) (⍳3)∘.×(⍳4)

b) (2×⍳5)∘.+⍳3

c) (2×⍳5)∘.+(2×⍳5)

d) 2×(⍳3)∘.×(⍳4)

e) 5+(⍳3)∘.×(⍳4)

f) 2×(⍳5)∘.+⍳5

2.14 a) Construct a function table according to the following specifications:

Left domain:	⍳4
Right domain:	⍳6
Body:	(3×⍳4)∘.+⍳6
Name:	*H*

b) Evaluate the following expressions:

3 *H* 5

5 *H* 3

1 *H* 1

4 *H* (1 *H* 1)

4 *H* 1 *H* 1

2 *H* 1 *H* 2

2.15 a) Construct a function table according to the following specifications:

Left domain:	56+⍳14
Right domain:	1 2 3
Body:	Same as Fig. 2.2
Name:	*W*

b) Evaluate the following expressions:

68 *W* 1

68 *W* 2

63 *W* 3

c) State clearly the relation between the function *W* and the function represented by Figure 2.2.

2.16 a) Construct the following function table:

Left domain:	⍳8
Right domain:	⍳8
Body:	(⍳8)∘.+⍳8
Name:	*PLUS*

b) Evaluate the following expressions:

3 *PLUS* 5

4 *PLUS* 6

3×4 *PLUS* 6

2 *PLUS* 2×3

4×2 *PLUS* 2×3

(4×2) *PLUS* 2×3

2+3 *PLUS* 4

2 *PLUS* 3+4

2+3+4

2 *PLUS* 3 *PLUS* 4

2.17 Evaluate the following expressions:

3⌈3

3⌊8

8⌈3

8⌊3

2×5⌈7

(5+2)⌈9

(5×2)⌈9

3⌈5⌊2

(3⌈5)⌊2

2.18 Evaluate the following expressions:

10⌈8⌈6⌈14⌈7⌈9

⌈/10 8 6 14 7 9

⌊/10 8 6 14 7 9

A←10 8 6 14 7 9
B←17 4 13 2 19
⌈/B

⌊/B

(⌊/A)+⌊/B

⌊/A+B

(+/A)⌊+/B

(+/A)⌈+/B

⌊/A⌈B

⌈/A⌊B

+/A⌊B

A∘.⌈B

B∘.⌈A

B∘.⌊A

2.19 a) Evaluate the following expressions:

+/3ρ4

4×3

+/5ρ3

5×3

+/10ρ10

10×10

b) Use multiplication to evaluate the following expressions:

+/6ρ3

+/25ρ16

+/100ρ13

+/20ρ20

+/2000ρ512

2.20 Evaluate the following expressions:

×/3ρ2

2⋆3

×/5ρ2

2⋆5

×/6ρ4

4⋆6

×/10ρ2

2⋆10

×/2ρ10

10⋆2

2.21 Evaluate the following expressions:

A←2 3 4 5 6 7 8 9

1⋆A

2⋆A

3⋆A

4⋆A

A∘.⋆A

2.22 Evaluate the following expressions:

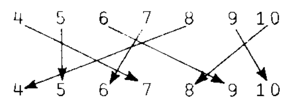
B←1 2 3 4 5 6

B⋆2

B⋆3

B⋆4

2.23 a) Let F be the function represented by the following map:



Then evaluate the following expressions:

F 4

F 6

F 9

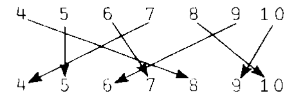
F F 6

F 2×3

2×F 3

F 4 5 6 7 8 9 10

b) Let G be the function represented by the following map:



Then evaluate the following expressions:

G 4

G 6

G 7

G G 6

G 2×3

G 4 5 6 7 8 9 10

F G 4

G F 4

F G 6

G F 6

F G 4 5 6 7 8 9 10

G F 4 5 6 7 8 9 10

c) How are the functions F and G related?

d) Make maps of some other pair of functions H and K which are related in the same manner that F and G are.

e) Construct a function table to represent the function F.

f) Repeat part (e) for each of the functions G, H, and K.

2.24 Let *F* and *G* be the functions defined by maps in Exercise 2.23. Then if *X* is any argument value, the expression *F G X* means to apply the function *G* to *X* and then apply the function *F* to the result.

a) Make maps to show the sequence of functions *F G X*.

b) Make a single map to show the overall result of the expression *F G X*.

c) State the overall effect of applying *F* to the result of *G*.

d) Repeat parts (a-c) for the expression *G F X*.

e) Repeat parts (a-d) for the functions *H* and *K* of Exercise 2.23.

CHAPTER 3

3.1 Evaluate the following expressions:

```
      8-6
____
      13-6
____
      13-6 5 4 3 2 1
____
       6 7 8 9 10-5
____
      1 2 3 4 5+5
____
      8-ι4
____
      +/8-ι4
____
      M←8 12 7 11 43
      N←6 7 2 1 20
      M-N
____
      M+N
____
      (M-N)+N
____
      (M+N)-N
____
      M∘.+N
____
      ι5
____
      6-ι5
____
      +/ι5
____
      +/6-ι5
____
      2×+/ι5
____
      (ι5)+(6-ι5)
____
      +/(ι5)+(6-ι5)
____
      5×6
____
      2×+/ι8
____
      8×9
____
      2×+/ι10
____
      10×11
____
      P←7+ι5
      P∘.-ι5
____
```

3.2 Fill in the blanks so that the expressions will give the indicated results. Note that each entry may be either a scalar or a vector:

```
      8-____
5
      (8-____)+6
10
      (8+6)-____
10
      ____-2 3 4 5
6 9 1 ¯8
      ____-ι5
2 4 6 ¯8
      +/8-ι____
25
      M←2 3 5 7
      ____-M
8 7 14 ¯2
      ____-M
6 5 3 ¯1
```

3.3 In defining the over notation it was shown that +/14 10 8 7 2 means 14+10+8+7+2. Similarly, -/14 10 8 7 2 means 14-10-8-7-2, where the expression is evaluated from the right as usual. That is, -/14 10 8 7 2 is equivalent to 14-(10-(8-(7-2))), or 7. Use this fact to evaluate the following expressions:

```
      -/8 6 4 2
____
      -/12 9 8 4 3
____
      -/20 14 12 10 18 9
____
```

```
      (20+12+18)-(14+10+9)
____
      -/8 7 6 5 4 3 2 1
____
      (8+6+4+2)-(7+5+3+1)
____
      -/7 6 5 4 3 2 1
____
      (7+5+3+1)-(6+4+2)
____
```

3.4 Make a map to represent each of the following expressions:

```
  7 8 9 10 11-5

  2 3 4 5 6+5

  10 11 12 13 14-8

  2 3 4 5 6+8

  ((ι5)+6)-6
```

3.5 Evaluate the following expressions:

```
      5-8
____
      5-ι8
____
      1-ι8
____
      8-ι8
____
      0-ι8
____
      S←ι8
      S+S
____
      S-S
____
      S∘.-S
____
      T←S+S
      T∘.-T
____
      T∘.-S
____
      S∘.-T
____
      -/10 8 6 4
____
      -/1 2 3 4
____
      -/ι5
____
      -/ι6
____
      -/ι7
____
      -/ι8
____
```

3.6 Fill in the blanks so that the expressions will yield the indicated results:

```
      8-____
¯3
      8-____
3
      (8+____)-48
8
      3 4 5 12-____
0 2 ¯5 ¯8
      3 4 7 12-____
¯2 ¯1 2 7
      (ι5)-____
¯3 3 ¯3 3 ¯3
      +/3-ι____
0
      +/5-ι____
0
      +/7-ι____
0
      -/ι____
¯3
      -/ι____
¯4
      -/ι____
¯8
```

3.7 Make maps to represent the following expressions:

```
      (ι5)-3

      (ι5)+¯3

      (ι7)-9

      (ι7)+¯9

      (ι7)+9

      (ι7)-¯9
```

3.8 Evaluate the following expressions:

```
      (ι5)-3
____
      (ι5)+¯3
____
      (ι7)-¯9
____
      (ι7)+9
____
      N←0-ι6
      N
____
      P←ι6
____
      P+N
____
      P-N
____
      N-P
____
      P∘.+N
____
      P∘.-N
____
      N∘.+P
____
      N∘.-P
____
```

3.9 Fill in the blanks so as to make the expressions yield the indicated results:

```
      ¯3 2 1 5-____
¯8 6 ¯4 18
      ¯4 1 ¯3 7 5+____
¯8 ¯5 ¯3 2 12
      S←¯8 ¯5 ¯3 2 12
      S-____
¯4 1 ¯3 7 5
      S+____
14 2 ¯8 3 ¯7
      S+____
¯5 ¯2 0 5 15
      S+____
¯11 ¯8 ¯6 1 9
      S-____
0 0 0 0 0
      S+____
0 0 0 0 0
      +/0-ι____
¯10
      +/0-ι____
¯21
      -/ι____
3
      -/ι____
¯3
      -/ι____
4
      -/ι____
¯4
```

3.10 Write algebraic expressions for each of the following:

The integers from ¯8 to 8

The integers from ¯4 to 15

Every third integer from ¯12 to 12

Every second integer from ¯9 to 7

The positive integers to 6

The positive integers to 6 in descending order

The negative integers from ¯6 in ascending order (that is, running from ¯6 to ¯1)

The negative integers greater than ¯7 in descending order

CHAPTER 4

4.1 a) Construct a subtraction table with a left domain of ⍳12 and a right domain of ⍳12.

b) Make a clear statement of each property you observe in the table of Part (a).

4.2 Let

```
A←⍳3
B←⍳4
S←A∘.-B
```

a) Evaluate the following expressions:

```
⍉S

⌽S

⊖S

B∘.-A

0-B∘.-A
```

b) Without using any of the flipping functions ⍉, ⊖, or ⌽, write an expression to yield a result equivalent to ⍉S.

c) Evaluate the following expressions:

```
⌽B

⌽S

A∘.-⌽B

⌽A

(⌽A)∘.-B

⊖S
```

d) State any relations you observe among the expressions of Part (c).

e) Write an expression using S (but not A or B) to yield a result equal to the result of the expression (⌽A)∘.-⌽B.

4.3 The following simple table M will be used to observe the behavior of the flipping functions:

```
      M←0 2∘.+1 2
      M
1  2
3  4
```

a) Evaluate the following expressions:

```
⌽⊖M

⊖⌽M

⍉⌽M

⌽⍉M

⍉⊖M

⊖⍉M

⌽⍉⌽M
```

b) The expressions of Part (a) produce several different results although some pairs produce the same result. Using sequences of flipping functions as long as you like, how many <u>different</u> results can you produce?

c) Can any sequence of flipping functions applied to M produce the result

```
1  2
4  3
```


d) Can you give a convincing argument to show that the different results you produced in Part (b) are <u>all</u> that can be produced?

e) Write the shortest possible expressions you can find for each of the different results produced in Part (b). For example, the expression ⌽⍉M produces the result

```
3  1
4  2
```

and is therefore equivalent to rotating M clockwise by one position. Hence a re-application of the pair ⌽⍉ (that is, ⌽⍉⌽⍉M) will effect a second rotation to produce the result

```
4  3
2  1
```

However, this can also be produced by the shorter expression ⊖⌽M.

f) From the preceding parts of this exercise it should be clear that ⊖⌽M is <u>not</u> equivalent to ⍉M. Nevertheless, for the subtraction table S it was obvious from the examples given in the text that ⊖⌽S is equivalent to ⍉S. What is there about the table S that makes this so?

4.4 Let

```
A←3+⍳6
B←2×⍳5
M←A∘.-B
```

a) Evaluate the following expressions:

```
A[4]

B[2]

M[3;5]

M[5;3]

(⍉M)[3;5]

(⍉M)[5;3]

M[2;]

M[;4]

(M[3;])[5]
```

b) Evaluate the following expressions:

```
A[2 4]

A[⍳3]

A[3+⍳3]

M[2 4;1 3 5]

M[2 4;]

M[;1 3]

A[3]

B[4]

A[3]+B[4]

(A∘.+B)[3;4]
```

4.5 Consider the addition table B given in the text. State any patterns you observe in the table. Where possible make your statements in both English and algebra. For example:

⍉B is equal to B.

0-⌽B[2;] is equal to B[2;].

B[I;] is equal to B[;I] for any value of I.

B[5;] is equal to 2+B[3;].

4.6 Repeat the work of Exercise 4.5 for the multiplication table N given in the text.

4.7 Quadrant 2 of the multiplication table N given in the text consists of the first seven rows and first seven columns of N. Hence Quadrant 2 is the table Q2 defined as follows:

```
Q2←N [⍳7;⍳7]
```

Quadrant 4 can be specified similarly:

```
Q4←N[8+⍳7;8+⍳7]
```

a) Write similar expressions to define the remaining quadrants Q1 and Q3.

b) State any relations you observe among the quadrants.

4.8 Repeat the work of Exercise 4.5 on the table MAX defined in the text.

4.9 Repeat the work of Exercise 4.5 on the table MIN defined in the text.

4.10 Evaluate the following expressions and compare the results:

```
I←⍳6

J←0-I

I∘.⌈I

J∘.⌊J

I∘.⌊I

J∘.⌈J
```

4.11 a) Repeat Exercise 4.10 with I←(⍳13)-7.

b) Evaluate the following expressions and comment on the patterns in the table T:

```
K←⍳8
R←K∘.⌈K
T←R⌈⊖⌽R
```

4.12 Evaluate the following expressions:

```
3=7

3=3

X←⍳7
Y←⌽X
X=Y

X≠Y

X∘.=X

X∘.≠X

X∘.=Y
```

4.13 Evaluate the following expressions:

```
X←⍳7
X∘.>X

X∘.≥X

X∘.≤X

X∘.≤⌽X

⍉X∘.≥X
```


4.14 Evaluate the following expressions:

```
I←(⍳11)-6
A←I∘.+I

4≤A

16≤A⋆2

S←I∘.-I
4≤S

M←I∘.×I

12≥M

144≥M⋆2
```

4.15 Evaluate the following expressions:

```
X←8 4 3 5 7 6
Y←4 3 10 8 2 5
X≤Y

⌈/X≤Y

⌊/X≤Y

5≤X+Y

⌈/5≤X+Y

⌊/5≤X+Y

⌈/9≤X+Y

⌊/9≤X+Y

⌈/15≤X+Y

⌊/15≤X+Y
```

4.16 Evaluate the following expressions:

```
A←(⍳6)∘.+⍳6
A=⍉A

⌊/A=⍉A

⌊/⌊/A=⍉A

S←(⍳6)∘.-⍳6
S=⍉S

⌊/⌊/S=⍉S

⌈/S=⍉S

⌊/⌈/S=⍉S

C←(⍳6)∘.≥⍳6
+/C

+/⍉C
```

CHAPTER 5

5.1 Evaluate the following expressions:

4×8

___

32÷4

___

32÷8

___

48÷8

___

(32÷8)+(48÷8)

___

(32+48)÷8

___

S←6×⍳7
S

___

S÷2

___

S÷3

___

S÷6

___

S÷1

___

S∘.÷1 2 3 6

___

(-S)÷2

___

¯3 ¯6 ¯9 ¯12 × 2

___

T←S-24
T

___

T∘.÷1 2 3 6

___

P←¯8 12 ¯10 21
Q←16 15 35 49
R←2 3 5 7
P÷R

___

Q÷R

___

(P÷R)+(Q÷R)

___

(P+Q)÷R

___

(P-Q)÷R

___

5.2 Fill in each underscored position giving either the result of evaluating the expression or a value such that the expression will yield the indicated result:

___×3
24

24÷3

___

___×15
300

300÷15

___

___÷20
25

25×20

___

___÷7
32

32×7

___

(25÷5)⌈(35÷5)

___

(25⌈35)÷5

___

(28+___)÷5
40

___÷5
¯40

(28+___)÷5
¯40

(¯28+___)÷5
40

(¯28+___)÷5
¯40

22 21 32÷___
2 7 16

22 21 32÷___
22 21 32

5.3 Make maps to represent each of the following expressions, where S←⍳6 and N←¯4+⍳7 and M←4×S:

S×5

S×5 followed by (S×5)÷5

N×2 followed by (N×2)÷2

M÷4 followed by (M÷4)×4

5.4 Evaluate each of the following expressions using the method of guessing, first obtaining two guesses which "bracket" the result (that is, one is too high and the other is too low), and then closing in on the result by successive guesses which lie between the guesses which bracket the result most closely. Make your guesses as good as possible to shorten the work, but show all of your work:

256÷8

378÷7

4096÷16

5040÷42

40320÷105

362880÷144

362880÷27

362880÷48

362880÷36

5.5 Evaluate the following expressions, using the method of guessing at a quotient, subtracting from the dividend the product of this guess with the divisor, making a guess at the quotient of the new remainder divided by the divisor, and so on. Show all of your work.

40548÷124

51324÷78

971203÷257

2511930÷1095

5764896÷2164

1505625÷1375

751424÷3184

5.6 Repeat the examples of Exercise 5.5 using the method of long division.

5.7 Fill in the blanks in the following, using long division where necessary.

241724÷178

___

___×314
853452

314×___
1174046

(___+15)×624
144144

(___-48)×176
457248

(___÷3)÷167
416331

2578647÷(167×3)

___

(268000÷4)÷250

___

268000÷(250×4)

___

(238750×5)÷50

___

23870÷(50÷5)

___

1728÷12

___

1728÷12*2

___

1728÷12*3

___

5.8 Make maps to represent each of the following, where S←¯4+⍳9:

```
S÷4 followed by (S÷4)×4

S÷3 followed by (S÷3)×6

S÷6 followed by (S÷6)×3
```

5.9 State the values of the divisor, dividend, and quotient for each of the following expressions:

```
8÷4
10÷2
196÷14
2048÷64
1728÷144
```

5.10 State the values of the numerator and denominator for each of the expressions of Exercise 5.9.

5.11 Give an appropriate name for each of the following fractions:

```
1÷2
1÷3
2÷5
7÷5
2÷6
3÷6
4÷6
6÷6
¯7÷12
```

5.12 Under each expression below enter a simpler equivalent expression of the form A÷B (where A and B are integers), as shown by example in the first four lines:

```
(2÷8)+(5÷8)
7÷8
(7÷3)+(8÷3)
15÷3

(10÷7)+(4÷7)

(¯6÷13)+(32÷13)

(32÷13)-(6÷13)

(42÷15)+(¯42÷15)

(26÷3)-(¯22÷3)

(38-47)÷19

(25+14)÷7

(25+9)÷(4+5)

(19+¯38)÷(7+8)

(3÷9)-(25÷9)+(¯20÷9)

(10+27)÷(4-3)

(¯32÷12)-(¯32÷12)

(¯1÷18)-(¯19÷18)-(6÷18)

(2÷11)+(2÷11)+(2÷11)

(3×2)÷11
```

5.13 Review each of the results obtained in the preceding exercise and add a third line giving an equivalent integer if there is such an integer. For example:

```
(7÷3)+(8÷3)
15÷3
5
```

5.14 Fill in the underscored expressions with integers such that the indicated equivalences will hold:

```
(5÷13)+(___÷13)
19÷13

(5÷13)+(___÷13)
2

(16÷31)+(___÷31)
8÷31

(___÷17)+(21÷17)
2

(31÷99)-(___÷99)
22÷99

(64÷19)-(___÷19)
64÷19

(29÷___)+(19÷___)
4
```

5.15 Under each expression enter a simpler equivalent expression of the form integer ÷ integer:

```
(2÷3)×(5÷7)

(3÷5)×(5÷3)

(¯10÷17)×(51÷2)

(¯2÷3)×(¯2÷3)

(4÷7)×(7÷9)+(15÷9)

(13÷8)×(14÷6)-(¯17÷6)

((13÷8)×(11÷3))+((7÷12)×(5÷2))

((3÷4)+(10÷4))×(35÷52)-(19÷15)

(¯2÷8)×(¯5÷3)

(0÷4)×(¯15÷3)

(¯7÷5)×(5÷5)

(3÷4)×(12÷12)
```

5.16 Review each result obtained in the preceding exercise and give an equivalent integer where possible.

5.17 Fill in the underscored positions appropriately:

```
(3÷5)×(___÷12)
18÷60

(17÷8)×(2÷___)
34÷120

(15÷___)×(___÷20)
120÷80

(17÷24)×(___÷___)
85÷96

(5÷___)×(6÷___)
30÷54

(5÷3)×(___÷___)
4

(5÷3)×(___÷___)
1

(17÷23)×(___÷___)
1

(___÷___)×(39÷41)
1
```

5.18 Under each expression enter an equivalent expression of the form integer ÷ integer:

```
(2÷3)×(2÷2)

(2÷3)×(3÷3)

(3÷4)×(5÷5)

(7÷9)+(2÷3)×(3÷3)

(7÷9)+(2÷3)×1

(7÷9)+(2÷3)

((8÷4)×(5÷5))+(¯13÷20)

((3÷4)×(5÷5))+((3÷5)×(4÷4))

((2÷3)×(2÷2))+((1÷2)×(3÷3))
```

5.19 For each expression write an equivalent expression of the form integer ÷ integer:

```
3×(4÷5)

4×(3÷5)

5×(3÷5)

(7÷9)×11

3×(7÷9)×3

3×7÷9×3

(7÷9)×(3÷3)

5×14÷13×2

1×2÷3×4

1×(2÷3)×4

4×3÷2×1
```

5.20 As shown in the first example, write equivalent expressions of the form ÷/*V* where *V* is a vector whose two elements are integers:

```
(÷/3 5)×(÷/2 3)

÷/6 15

(÷/16 28)×(÷/10 20)

(16÷28)×(10÷20)

(10÷7)×(¯12÷3)

(÷/23 4)×(÷/4 23)

(÷/12 25)×÷/4 4)

(3÷12)+(5÷12)

(÷/3 12)+(÷/5 12)

(÷/15 28)+(÷/¯1 28)

(÷/17 29)-(÷/¯32 29)

(÷/2 5)×(÷/3 7)

÷/2 5×3 7

2×÷/4 5

÷/2×4 5

5×÷/2 3×4 7
```

5.21 For each expression write an equivalent expression which involves not more than two integers:

```
(2÷7)+(4÷5)

(3÷5)+(4÷6)

(12÷24)+(¯3÷17)

(12÷24)-(3÷17)

(12÷24)-(¯3÷17)

(2÷5)+(3÷10)

(÷/2 5)+(÷/3 10)

(÷/5 2)+(÷/10 3)

2×(÷/5 2)-(÷/¯10 3)

2 7×(÷/5 2)+(÷/3 11)

3 3×(÷/5 7)-(÷/¯11 6)

(1÷2)+(3÷4)+(5÷6)

(1÷2)+(2÷3)+(3÷4)

(÷/⍳2)+(÷/1+⍳2)+(÷/2+⍳2)
```

5.22 Under each expression write a series of equivalent expressions showing the steps in simplifying to a final expression of the form *X*÷*Y*:

```
A←4 7
B←2 5
(÷/A)+(÷/B)

(÷/A)-(÷B)

(÷/B)-(÷/A)

(÷/B)+(÷/A)

(÷/A)+(÷/A)

G←10 9
(÷/B)-(÷/G)

(÷/A)+(÷/G)-(÷/G)

(÷/G)-(÷/B)

W←(÷/A)+(÷/G)
(÷/W)-(÷/A)

(÷/W)+(2×(÷/A))

(÷/B)+(÷/W)

(÷/B)×(÷/A)

(÷/W)×(÷/G)

(÷/W)×(÷/W)

(÷/G)×(÷/B)

(÷/W)+(÷/B)
```


5.23 For each expression write a simpler equivalent expression involving at most two integers:

```
(9÷2)÷(4÷3)

(7÷3)÷(4÷9)

(¯7÷3)÷(4÷9)

3÷(4÷9)

5÷(5÷6)

A←3 4
B←5 6
(÷/A)÷(÷/B)

(÷/B)÷(÷/A)

(÷/A)×(÷/B)÷(÷/A)

(÷/78 23)÷(÷/45 3)

(4÷7)÷(÷/3 1)

(4÷7)÷3

(7÷8)÷2
```

5.24 Write the following rational numbers as decimal fractions:

```
5÷10

2÷10

8÷1

34÷100

34÷1000

34÷10

7÷10000

234÷10

÷/234 100

234÷1000

45÷10

÷/294 10000

38÷10

50÷10

÷/23 100

÷/¯8 1000

÷/¯567 1

0000÷100

4567÷100
```

```
28345÷1000
79÷1000
÷/¯78 1000
÷/293847 10
29÷1
9287654÷100000
9÷100000
23÷100
36887÷10
```

5.25 Write decimal fractions equivalent to each of the following expressions:

```
(÷/14 10)×(÷/7 100)
(÷/14 10)÷(÷/100 7)
(÷/24 100)×(÷/74 10)
(÷/14 100)+(÷/27 100)
(÷/64 100)+(÷/136 100)
(÷/164 100)+(÷/135 10)
(÷13.6 10)
(÷/14.82 10)
(÷/15.66 10)×(÷/256.4 100)
```

5.26 Evaluate the following expressions showing each rational result as a decimal fraction:

```
V←6 27 135
E←10*⍳4
V∘.÷E

F←10*(⍳7)-4
V∘.÷F
```

5.27 Evaluate the following:

```
34.3+6.3
2.5+5.6
19.4-3.2
38.6-¯10.3
(÷/48 10)+4.6
6.00+3.87
4.7300+9.4529+98.0000
¯7.50+68.90-¯548.21
5.78-2.40
¯67.8+3.6
866.00+(4÷100)
¯13.6-¯7.2
```

5.28 Evaluate the following expressions:

```
5.3+8.27
8.6+5.14+1.26
870.3458+¯78.2
(÷/34 10)+21.7-¯44.4
45.23+(÷/37 10)
(÷/56 100)+(4÷10)
5.6-(45÷10)+¯4.12
19.5-279.69
58.3-23.45
¯67.8+¯692.5678
(÷/¯93 1000)+2.45
```



```
(÷/98 100)+(12÷1)-(÷/98 10)
36.5-¯578.4
77.777-66.66
¯46.9-26.879
```

5.29 Obtain decimal fraction equivalents for the following expressions:

```
3÷4
1728÷25
1728÷16
153÷12
2 3÷5 25
3÷5 25
(⍳8)÷8
(⍳16)÷16
(⍳32)÷32
(⍳25)÷25
(⍳25)÷4
1÷2*⍳6
1÷5*⍳6
1÷10*⍳6
1-(⍳8)÷8
1-(⍳32)÷32
```

5.30 Obtain the best 3-place decimal fraction approximation to each of the following expressions:

```
1÷3
2÷3
(⍳9)÷9
(⍳32)÷32
(⍳10)∘.÷(⍳10)
(65÷24)÷(12÷44)
(71÷3)÷(7÷8)
(46÷9)÷(11÷19)
(32÷21)÷(12÷10)
(24÷28)÷16
(4÷9)÷(21÷8)÷(32÷6)
(7÷12)÷(25÷71)
(8÷1)×(6÷37)
(8÷13)÷(20÷9)
(7÷14)÷(31÷6)
(66÷2)÷(2÷3)
(9÷16)÷(6÷6)
(7÷2)÷(8÷3)÷(9÷4)
```

5.31 Evaluate the following expressions:

```
2.41×1.48
3.27×16.4
1.287×14.321
234.56×12.34
2.4×3.5×4.6×5.7
13.287×4.8+5.6
1.125×.32
```

5.32 Obtain the best 2-place decimal approximation to each of the expressions of the preceding exercise.

5.33 Find the best 3-place approximation to each of the expressions of Exercise 5.31 but with each multiplication replaced by division.

5.34 Write each of the results of Exercise 5.31 in exponential notation.

5.35 Write each of the results of Exercise 5.33 in exponential notation with the value 3 for the integer following the *E*.

5.36 Obtain the best 3-place approximation to each of the following expressions:

```
2÷3

2÷¯3

¯2÷3

¯2÷¯3
```


CHAPTER 6

6.1 Evaluate the following expressions:

```
A←2 3 5 7
B←4 1 2
C←9 8
A,B

B,A

(A,B),C

A,(B,C)

(-⌽ι4),ι4
```

6.2 Let *D* be the 8-by-8 division table shown in the text.

a) Evaluate the following expressions:

```
D=1

D=1÷2

D=1÷3
```

b) Examine the results of Part (a) and state the pattern produced by expressions of the form *D=R*, where *R* is any value which occurs more than once in *D*. (if necessary evaluate further cases, possibly extending the table *D* itself)

6.3 a) Give expressions of the form used in Exercise 4.7 (for the multiplication table *N*) to define four suitable quadrants of the division table *J*∘.÷*K* given in Section 6.3.

b) State any relations observed among the quadrants.

6.4 a) Evaluate the following expressions:

```
A←6 7 8 9 10 11
B←7 8 9 10 11 12
C←9 10 11 12 13 14
D←10 11 12 13 14 15
A÷B

C÷D

T←(A÷B)∘.≤(C÷D)
```

b) Use the table *T* to determine which is the larger of each of the following pairs of rationals:

8÷9 and 9÷10
9÷10 and 10÷11

c) Without using division write an expression which will yield a table identical with *T*. Evaluate the expression and compare the result with *T*.

6.5 Evaluate the following expressions:

```
2 3∘.*1+ι10

2*¯4+ι12

3*¯4+ι12

2 3∘.*¯4+ι12

2 3 4 5 6∘.*¯4+ι7
```

6.6 a) Evaluate the following expressions to five decimal places:

```
A←⍳5
B←0-A
2*A

2*B

(2*A)×2*B

(2*⍳8)×2*0-⍳8

+/(2*⍳100)×2*0-⍳100
```

b) Evaluate the following expressions to five decimal places:

```
A←⍳4
3*A

3*0-A

(3*A)×(3*0-A)
```

c) Evaluate the following expressions to five decimal places:

```
A←⍳4
5*A

1÷5*A

5*0-A

(1÷5*A)=5*0-A
```

6.7 Evaluate the following expressions:

```
A←⍳5

10*A

10*0-A

¯10*A

¯10*0-A

20*A

¯20*A

0*A

0*0-A
```

6.8 Evaluate the following expressions:

```
A←(⍳6)÷2

9*A

B←0-A

9*B

49*A

49*B
```

6.9 a) Determine a number *A* which when multiplied by itself yields 10 (correct to three decimal places).

b) Use the result of Part (a) to evaluate the following expressions:

```
10*(⍳6)÷2

10*0-(⍳6)÷2
```

6.10 Evaluate the following expressions:

```
A←⍳6
3*A÷3

3*A÷4

3*A÷5

3*A÷6

5*A÷6
```

CHAPTER 7

7.1 Evaluate the following expressions:

```
3|17

3|17 593 18 42

9|⍳9

9|+/⍳9

+/3|⍳3

+/4|⍳4

+/12|⍳12

(8+7)|6 1 654 42 752 9104

(5|77 5 24 2750 4 2)-660

3|+/6 8

(3+9≤6)|726 10 9 234 3064 36

2|-/2 4 9 653 1504 7

(4⌈6)|6 2522 5 5

5|5 1312 9 1 162×5 9 932 7 3

7|7 4394⌊955 8513

(9|7418 20 887)+200 10 866

(3×6)|7 1 1 8 4+7 21 82 4 10

|/3 9 1365+10 258 3

7|60 3

(6|2)×755

6|4≥3

6|3216 5 5172

|/4 19

|/7 37
```

7.2 Evaluate the following:

```
1 2 3 ∘.| 4 5 6 7

2 4 6 ∘.| 6 12 18 24 72

1 4 ∘.| 10 4 520 831

4 7 8 10 ∘.| 920 6540 42

7≥10 50 3 1 10 ∘.| 3 360 2 5

8 8 10 89 3 ∘.| 9410 6 8

10 3 ∘.| 220 7148 14 910 7

10 56 2 ∘.| 4 388

7| 1 17 1 4 26 ∘.+ 3 1

10 7 5 9 7 ∘.| 3 680 9090 26

22 5 4 6 89 ∘.| 8 48 67 7 2

7-8 9 10 ∘.| 8 320

2 1 2 ∘.| 10 4 3 6922

(1+2≥1 6 3 4) ∘.| 3 7

5 4 3 1 ∘.| 529 4 6|2 486 9

8 10 7 3 8 ∘.| 10 25 85 69 5

7 7 8 6 ∘.| 4365 7 585

6|6 7 34 1 ∘.+ 3 1 5 458

70 5 3 ∘.|36 84 10 26 2516

5 8 ∘.| 69 4 9
```

```
(3×10 5) ∘.| 24 66 2 8

6 2 9 4 5 ∘.| 3 2 373+8 1 45
```

7.3 Evaluate the following:

```
3|¯10

(3+7)|9 5270 ¯1 ¯7 ¯4160

10|¯727 ¯7 ¯9 ¯3

(4+2)|¯61

6|+/¯465 0

1|¯8

6|8416⌊0 0 4 6 5

16|×/1 5 3

66|4998 1476⌊¯10

(10×5)|2⌊9 25 48

(26×10)|350 46 9 ¯94 6517

|/¯5+7023 99 ¯10

8|¯6 2 ¯451 990 ¯216

(5+5≥9)|-6164 1 ¯1

1|¯3 1×¯3899

(3|3 ¯8198)×¯9 2

7|=/¯7 ¯1

(4⌈3≤6)|¯9 9 6 0≥44 3 38 40

6|¯9402 3216
```

7.4 Make a table of the results of the expression (⍳9)∘.|¯10+⍳19. Do you notice any patterns in the table? Are they similar to the patterns in Table 7.1? Draw circles around all the 0's in the table. Connect groups of these circles by straight lines. Does it seem that one half of the table is the mirror image of the other half with respect to these lines?

7.5 Evaluate the following expressions:

```
0=3|⍳16

0=5|⍳25

M←(10×0,⍳9)∘.+0,⍳9
4|M

9|M

7|M
```

7.6 Make the table 0=(⍳10)∘.|⍳10. Circle the positions of all the 1's in the table. Why are there no 1's in half of the table? What is the significance of the line of 1's that divides the table in half?

7.7 In the table of the preceding exercise, the number 3 will be seen to have exactly two divisors (1 and 3). Find all the other numbers in the table which have exactly two divisors. Find four more numbers not in the table which have this property.

7.8 Make the table 0=(⍳10)∘.|¯11+⍳21. Note all of the interesting properties of the table that you can observe; for example, is the left half a mirror image of the right half? Where does the split occur? Is ¯8 divisible by the same numbers as 8?

7.9 Which of the following numbers is divisible by 3:

```
12 45 34 87 ¯10 5 76543 76
¯567 9876543 39 ¯149 9378 345 83
86 ¯237 873 3482 ¯93754
```


Add up the digits of each number. Are these sums divisible by 3? Can you find a rule that will tell quickly whether a number is divisible by 3 or not? Can you find a relationship between the 3-residue of the number and the residue of the sum of its digits? Does this relationship hold for integers other than 3?

7.10 Which of the following numbers is divisible by 5?

```
56 25 90 1234 1000 ¯595
98765 234 ¯3591 63 55 80 ¯390 48
240
```

Is there any relationship between the 5-residue of the number and the 5-residue of its final digit?

7.11 Which of the following numbers is divisible by 2?

```
8 24 86 ¯456 9870 34592 ¯237
162 1000 645 343 926 ¯427 1445 92
```

Is there any relationship between the 2-residue of a number and the 2-residue of its final digit?

7.12 Write down in your own words a definition for the | function. According to your definition, what would the result of 0|*N* be, where *N* is any integer?

Now suppose you defined *A*|*B* as the repeated subtraction of *A* from *B* until a result is obtained that is 0 or larger but also less than *A*. Will this definition produce the same results as the definition introduced in the text? Using this new definition, 0|*B* would be a never ending process. Would it seem reasonable to let 0|*B* have the result *B*?

7.13 Evaluate the expression (⍳*N*)|*N* for each of the following values of *N*:

```
9 12 15 17 24 32 36
```

7.14 Use the results of the preceding exercise to determine all of the factors of each of the numbers 9,12, etc., listed in that exercise.

7.15 For each list of factors obtained in the preceding exercise write the list of corresponding factor pairs. For example, the factors of 6 are 1 2 3 6 and the corresponding factors are 6 3 2 1.

7.16 From your answers to the preceding exercise, does it seem reasonable that every number has an even number of factors? Can you find any numbers that have an odd number of factors? If a number has an odd number of factors, what are its factor pairs?

7.17 Evaluate the following expressions:

```
1 0 1 0 1/3 5 7 9 11

0 1 0 1 0/3 5 7 9 11

X←12 17 ¯4 5 ¯3 0 ¯4 0

1 1 1 1 0 0 0 0/X

(X>0)/X

(X≥0)/X

(0=2|X)/X

(0=3|X)/X

((0=2|X)⌈(0=3|X))/X
```

```
((0=2|X)⌊(0=3|X))/X

((0=2|X)⌊(0≠3|X))/X

(0=5|⍳25)/⍳25

(1=5|⍳25)/⍳25

(2<5|⍳25)/⍳25

+/X∘.=X

(1=+/X∘.=X)/X

(1≠+/X∘.=X)/X
```

7.18 Write expressions which will select from the positive integers up to N those numbers satisfying the stated properties. For example, the expression (0=4|⍳N)/⍳N would be appropriate for the property "all integers up to N which are divisible by 4".

a) All integers up to N which are divisible by either 3 or 5

b) All integers up to N which are divisible by both 3 and 5

c) All integers up to N which are divisible by 15

d) All integers up to N which are greater than M

e) All integers up to N which are greater than M and divisible by 5

f) All integers up to N which are divisible by every element of the vector V

g) All integers up to N which are divisible by exactly K elements of the vector V

7.19 Use the expression (2=+/⍉0=(⍳N)∘.|⍳N)/⍳N to determine all of the prime numbers up to 20. Show each step of the calculation.

7.20 Evaluate the following expressions:

```
P←(2=+/⍉(⍳12)∘.|⍳12)/⍳12
P*2 0 2 0 1

×/P*2 0 2 0 1

×/P*0 0 0 0 0

×/P*1 0 0 0 0

×/P*0 1 0 0 0

×/P*2 0 0 0 0

×/P*0 0 1 0 0

×/P*1 1 0 0 0
```

7.21 The expressions of the preceding exercise were all of the form ×/P*E, and the last five of them yielded the first five positive integers. Determine further values of E to continue the process for integers 7, 8, 9, etc. What is the first integer impossible to represent in this way?

7.22 Take the first integer which cannot be represented in the form ×/2 3 5 7 11*E and append it (it is a prime number) to the list P and then continue the process of Exercise 7.21 for a few more integers. Can every integer be represented as ×/P*E where P is a vector of prime numbers?

7.23 a) If P is a vector of primes and if M←×/P*E and N←×/P*F and G←×/P*E⌊F, then G is a divisor of both M and N. Choose a number of different values of E and F and verify that this is so for the cases chosen.

b) Explain why G is a divisor of both M and N.

c) Is it possible to find a number larger than G which divides both M and N? Why?

7.24 a) If P, M, and N are as defined in the preceding exercise, and if L←×/P*E⌈F, then both M and N divide L. Verify this for a few values of E and F.

b) Explain why M and N divide L.

c) Is it possible to find a number smaller than L which is divisible by both M and N? Why?

CHAPTER 8

8.1 Evaluate the following expressions:

```
!3

×/⍳3

!4

×/⍳4

!⍳10

(!5)÷(!4)

(!6)÷(!5)

(!1+⍳10)÷(!⍳10)

(1+⍳10)×(!⍳10)

(!⍳10)÷⍳10

1,!⍳9
```

8.2 Comparison of the last two results of Exercise 8.1 suggests a definition for the value of !0. What is the value? Would its adoption agree with the obvious requirement that !N+1 is equal to (N+1)×!N? What value would the same line of reasoning give for !¯1?

8.3 Evaluate the following expressions:

```
-⍳6

X←2 ¯5 3 ¯7 4

-X

X-3

X+¯3

X+-X

X--X

X×-X

X⌈-X

X⌊-X
```

8.4 Evaluate the following expressions correct to 3 decimal places:

```
÷4

÷5

÷6

÷⍳8

-÷⍳8

÷-⍳8

÷!⍳5

+/÷!⍳5

÷2*⍳5

+/÷2*⍳5
```

8.5 Evaluate the expression +/÷2*⍳N for the first few positive values of N. What integer do these results seem to be approaching? Can you choose a positive value of N large enough so that +/÷2*⍳N is larger than 1?

8.6 a) Evaluate the following expressions:

```
|3 ¯4 7 ¯9 ¯10

X←3 ¯4 7 ¯9 ¯10
|X

|-X
```



```
-|X

X÷|X

+/|X

|+/X

X=|X

(X=|X)/X

(X≠|X)/X
```

b) Evaluate the following expressions:

```
P←7.2 ¯3.4 8.1 ¯6
|P

P⌈-P
```

c) What is the relation between the expressions |P and P⌈-P appearing in Part (b)? Would this relation remain true for any value assigned to P?

8.7 Evaluate the following expressions:

```
⌊3.5 ¯2.6 2 ¯4.9

⌈3.5 ¯2.6 2 ¯4.9

⌊(⍳10)÷2

⌈(⍳10)÷2

⌊(⍳10)÷3

⌈(⍳10)÷3

X←1.8 ¯2.7 ¯6 4.9 7

X=⌊X

(X=⌊X)/X

(X≠⌊X)/X

N←⍳12
⌊N÷3

(N-3|N)÷3

⌊N÷5

(N-5|N)÷5
```

8.8 Evaluate the following expressions:

```
~1 1 0 1 0 1

~~1 1 0 1 0 1

X←¯2 3 ¯5 7 11

X>3

~X>3

X≤3

~0=5|⍳12

0≠5|⍳12
```

8.9 Evaluate the following expressions and compare their results:

```
L←0 1
L∘.⌊L
~(~L)∘.⌈(~L)

L∘.⌈L

~(~L)∘.⌊(~L)

L∘.≠L

~(~L)∘.=(~L)

L∘.<L

~(~L)∘.≤(~L)
```

8.10 If *L* is any logical vector (i.e., each of its elements is either 0 or 1) of any dimension, then the expressions ⌊/*L* and ~⌈/~*L* yield the same result.

a) Verify this for a number of values of *L*.

b) Perform a similar verification of the equivalence of ⌈/*L* and ~⌊/~*L*.

c) Find similar relations among the functions <, ≤, =, ≥, and ≠. For example, ≠/*L* is equivalent to ~=/~*L*.

8.11 Evaluate the following expressions:

```
A←2 3 5
B←1 3 5 7 9
ρA

ρB

+/A=A

+/B=B

M←A∘.+B
ρM

×/ρM

ρ⍉M

ρB∘.+A
```


CHAPTER 9

9.1 Define a function called *D6* to determine divisibility of its argument by 6. Then evaluate the following expressions:

```
D6 12

D6 ⍳12

D6 (⍳10)∘.+(⍳10)

D6 (⍳10)∘.×(⍳10)

D6 (⍳10)∘.-(⍳10)
```

9.2 Define a function called *B* which determines the square of its argument. Then evaluate the following expressions:

```
B ⍳6

B (⍳6)∘.+(⍳6)
```

9.3 Define a function called *R7* which yields the remainder when its argument is divided by 7. Then evaluate the expression *R7* ⍳12.

9.4 Define a function called *IQ7* which yields the integer part of the quotient of its argument when divided by 7. Then evaluate the expression *IQ7* 3 74 23 49.

9.5 Using the functions defined in the preceding exercises, evaluate the following expressions:

```
3×D6 ⍳10

+/D6 ⍳10

⌊/D6 72 138 252

3×B 2+⍳5

X←12+2×⍳8

7×IQ7 X

(7×IQ7 X)+R7 X
```

9.6 a) Using the functions defined in preceding exercises, evaluate the expression *D6 R7 B* ⍳8

b) Let *C* be the function defined as follows:

```
∇Z←C X
Z←D6 R7 B X∇
```

Now evaluate the expression *C* ⍳8

9.7 Define monadic functions to yield each of the following results:

a) The area of a square as a function of the length of its side.

b) The area of a circle as a function of its radius (Use 3.1416 as an approximation to pi).

c) The area of a circle as a function of its diameter.

d) The volume of a sphere as a function of its radius.

e) The length of a rope in inches as a function of its length in feet.

9.8 Use the dyadic function *F* defined in the text to evaluate the following expressions:

```
2 4 6 8 F 13 14 15 16

4 F 13 14 15 16

2 4 6 8 F 13
```

```
M←(⍳5)∘.+(⍳5)

M F 7+M
```

9.9 Define a dyadic function called *H* which gives the area of the rectangle whose length is given by the first argument and whose width is given by the second argument. Then evaluate the following expressions:

```
3 H 4

3 4 5 H 5 6 7

3 H 5 6 7

3 4 5 H 5
```

9.10 Define a dyadic function *K* which yields the volume of the square cylinder, where the first argument represents the height of the cylinder and the second argument represents the length of the square base.

9.11 Define dyadic functions to yield each of the following results (the first argument mentioned is to be the first argument of the function):

a) The area of a triangle as a function of its base and altitude.

b) The perimeter of a rectangle as a function of its length and width.

c) The width of a rectangle as a function of its area and length.

d) The width of a rectangle as a function of its length and area.

e) The volume of a circular cylinder as a function of its height and the radius of its base.

f) The altitude of a triangle as a function of its area and base.

9.12 a) A rectangular plot is to be enclosed with 432 yards of fencing. Define a function to give the area of the enclosed plot (in square yards) as a function of the length of one of the sides (in yards).

b) Evaluate the function for a number of arguments to determine that value which yields the largest possible area.

9.13 a) A rectangular plot is to be enclosed with a fence of length *L*. Define a function which gives the area enclosed as a function of *L* and of the length *S* of one of the sides.

b) Evaluate the function for a number of values of *L* and *S* and determine the largest possible value of the area for a given fence length *L*.

c) How do the values of *L* and *S* compare when *S* has been chosen to give maximum area for some fixed value of *L*?

9.14 Using the function *PR* defined in the text, determine the value of the expression ⍴*PR X* for the following values of *X*: 10, 15, and 20.

9.15 Using the functions *FTOC* and *CTOF* defined in the text, evaluate the following expressions:

```
FTOC 20+⍳10

CTOF FTOC 20+⍳10

FTOC 20+⍳10

CTOF FTOC 20+⍳10
```


9.16 Using the function *A* defined for adding rationals, evaluate the following expressions:

```
3 4 A 1 2

÷/3 4 A 1 2

(÷/3 4)+(÷/1 2)

5 7 A 4 6

21 3 A 15 8

27 7 A 1 10

14 13 A 26 29
```

9.17 Define a function *M* which multiplies rationals in the same manner that the function *P* adds then. Then evaluate the following expressions:

```
3 4 M 1 2

÷/3 4 M 1 2

(÷/3 4)×(÷/1 2)

5 7 M 4 6

21 3 M 15 8

27 7 M 1 10
```

9.18 Define a function *D* which divides one rational by a second. Then evaluate the following expressions:

```
3 4 D 2 1

÷/3 4 D 2 1

(÷/3 4)÷(÷/1 2)

5 7 D 4 6
```

9.19 Using the function *R* of the text, show the results produced by the following execution traces:

```
TΔR←⍳4
Q←R 3
Q←R 4

TΔR←2 4
Q←R 3
Q←R 4
```

CHAPTER 10

10.1 Analyze each of the four following function tables, that is, determine a function to fit each table:

0	.4
1	2.1
2	3.8
3	5.5
4	7.2
5	8.9

0	¯3.9
1	¯2.7
2	¯1.5
3	¯0.3
4	0.9
5	2.1

0	¯4.7
1	¯1.9
2	0.9
3	3.7
4	6.5
5	9.3

0	15
1	19
2	23
3	27
4	31
5	35

10.2 For each of the tables of Exercise 10.1 make a corresponding map and use it to determine an expression representing the table. Compare the results with the results of Exercise 10.1.

10.3 Graph each of the functions of Exercise 10.1.

10.4 Graph each of the following two functions:

0	¯12.4
1	¯8.9
2	¯5.6
3	¯2.5
4	0.4
5	3.1
6	5.6
7	7.9
8	10.0
9	11.9
10	13.6
11	15.1
12	16.4
13	17.5
14	18.4
15	19.1
16	19.6
17	19.9
18	20.0
19	19.9
20	19.6
21	19.1
22	18.4
23	17.5
24	16.4
25	15.1
26	13.6
27	11.9
28	10.0
29	7.9
30	5.6
31	3.1
32	0.4
33	¯2.5
34	¯5.6
35	¯8.9
36	¯12.4
37	¯16.1
38	¯20.0
39	¯24.1

0	¯61
1	¯50.59
2	¯41.32
3	¯33.13
4	¯25.96
5	¯19.75
6	¯14.44
7	¯9.97
8	¯6.28
9	¯3.31
10	¯1.00
11	0.71
12	1.88
13	2.57
14	2.84
15	2.75
16	2.36
17	1.73
18	0.92
19	¯0.01
20	¯1.00
21	¯1.99
22	¯2.92
23	¯3.73
24	¯4.36
25	¯4.75
26	¯4.84
27	¯4.57
28	¯3.88
29	¯2.71
30	¯1.00
31	1.31
32	4.28
33	7.97
34	12.44
35	17.75
36	23.96
37	31.13
38	39.32
39	48.59

10.5 Use the graphs of Exercise 10.3 to analyze each of the functions they represent. Compare the results with those of Exercise 10.1.

10.6 Consider the function *L* as defined below:

```
∇Z←C L X
Z←C[1]+C[2]×X∇
```

When applied to any two-element vector left argument and any vector right argument it produces a function which plots as a straight line. For example, if *X*←0,⍳5, then *X* is the first column of the first table of Exercise 10.1 and .4 1.7 *L X* is the second column.

a) Write expressions using *L* to produce the second column of each of the tables of Exercise 10.1.

b) Use the function *L* to produce a number of new function tables. Then graph each function and use the graph to analyze the function (i.e., determine an expression for it). It is best if you do not know or remember the expression which produced the table - either exchange tables with fellow students or lay your tables aside for a few days before analyzing them.

10.7 Use the graphs produced in Exercise 10.4 to answer the following questions about each of the functions they represent:

a) For what value (or values) of the argument does the function have the value 0?

b) For what values of the argument is the function equal to 3, to ¯3, to 100?

c) For what argument values does the function reach a local high point?

d) For what argument value does the function appear to be changing most rapidly.

10.8 For each of the function tables of Exercise 10.4 attempt to find an expression which represents the function. For each expression you try, evaluate it for some or all of the argument values in the table to see how closely your proposed function fits the given function. You may find some of the results of Exercise 10.7 useful.

10.9 Evaluate the following expressions:

```
3↑⍳5

¯3↑⍳5

3↓⍳5

¯3↓⍳5

7↑⍳5

¯7↑⍳5

A←1 2 3 4 5
B←6 7 8
ρA

ρB

(ρB)↑A

B+(ρB)↑A

A+(ρA)↑B
```

10.10 a) Evaluate the following expressions:

```
Y←0 1 4 9 16 25 36
1↓Y

¯1↓Y

V←(1↓Y)-(¯1↓Y)
V

W←(1↓V)-(¯1↓V)
W

(1↓W)-(¯1↓W)
```

b) Repeat Part (a) with

```
Y←(0,⍳6)*3
```

c) Repeat Part (a) with *Y* specified as the column of Fahrenheit values from Table 10.1.

d) Repeat Part (a) with *Y* specified as the second column of the first table of Exercise 10.4.

e) Repeat Part (a) with *Y*←⍳8

10.11 Make a difference table for each of the functions of Exercise 10.1.

10.12 Make a difference table for each of the function tables produced in Exercise 10.6.

10.13 Use the difference tables produced in Exercise 10.11 to determine expressions to fit each of the functions. Compare the results with those of Exercise 10.1.

10.14 Use the difference tables produced in Exercise 10.12 to determine expressions to fit each of the functions. Compare the results with those of Exercise 10.6.

10.15 Make a difference table for each of the functions of Exercise 10.4. Be sure to include enough columns in the table so that the last column has a constant value.

10.16 Use the difference tables of Exercise 10.15 to determine an expression for each of the functions represented. Evaluate your expressions for a few arguments (say, 0 5 10 20 30) to see if your expressions do properly represent the functions.

10.17 Extend each of the difference tables produced in 10.15 by appending two further columns. What can you say about any column which follows a constant column?

10.18 Consider the following function:

```
∇Z←C QUADRATIC X
Z←(X-C[1])×(X-C[2])∇
```

When applied to any two-element vector left argument and any vector right argument it produces a function called a *quadratic* function. Choose various values of the left argument and the value 0,⍳6 for the right argument to produce tables for a number of functions. Make difference tables to analyze each of the functions produced and apply each of the expressions produced to the argument 0,⍳6 to see if the expressions properly represent the functions.

10.19 Repeat Exercise 10.18, replacing the quadratic function by the *cubic* function defined as follows:

```
∇Z←C CUBIC X
Z←(X-C[1])×(X-C[2])×(X-C[3])∇
```

The left argument must, of couse, be a 3-element vector.

10.20 Extend one of the difference tables of Exercise 10.15 by one column (of zeros) to make two tables of the same size to be used as follows:

a) Multiply the first table by 3 and verify that the resulting table is a proper difference table.

b) Multiply the second table by 4 and verify that the result is a proper difference table.

c) Add the two tables and verify that the result is a proper difference table.

d) Add 3 times the first table to 4 times the second table and verify that the result is a proper difference table.

10.21 a) Use the difference table produced in Exercise 10.20(a) to determine an expression for the function it represents. Compare this expression with 3 times the expression produced in Exercise 10.16.

b) Repeat Part (a) for each of the difference tables produced in Exercise 10.20, comparing each result with an appropriate expression from the results of Exercise 10.16.

10.22 Evaluate the factorial polynomial of order 7 for the arguments 0,⍳7 and from the results form the difference table for the polynomial.

10.23 Evaluate the following expressions:

```
∇Z←G X
Z←¯3+X*2∇

X←¯4+⍳7
X

V←G X
V

L←⌈/V
S←⌊/V
R←⌽(¯1+S)+⍳1+L-S
R

M←R∘.=V
M
```

10.24 A logical table containing many zeros can be displayed more easily using squared paper, drawing lines to enclose a rectangle of the same shape as the table and entering a 1 in each square corresponding to a 1 element in the table. The zeros need not be entered. Display the matrix *M* of Exercise 10.23 in this manner.

10.25 a) Evaluate the following expressions, using the scheme of Exercise 10.24 to display any logical tables produced:

```
∇Z←H X
Z←X*3∇

X←¯4+⍳7
V←H X
R←⌽(¯1+⌊/V)+⍳1+(⌈/V)-⌊/V
M←R∘.=V
M
```

b) Repeat Part (a), replacing each use of the function *H* by use of the following function *K*:

```
∇Z←K X
Z←(X-1)×(X+2)∇
```

10.26 Evaluate the following expressions, using the scheme of Exercise 10.24 to display the logical tables produced:

```
X←¯9+⍳17
2>|X∘.-X

5<|X∘.-X

(2>|X∘.-X)⌈(5<|X∘.-X)

7≥|X∘.-X

7<|X∘.-X

6=X∘.+X

12=X∘.×X

12=|X∘.×X
```

10.27 Evaluate the following expressions, using the scheme of Exercise 10.24 to display any logical tables produced:

```
X←0,.1×⍳10
V←X*2
R←0,.05×⍳20
W←|R∘.-V
.01≥W

.02≥W

.1≥W
```

10.28 Evaluate the following expressions:

```
ALPH←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
ALPH[8 9 7 8]

ALPH[⍳4]

ALPH[⌽⍳4]

⌽ALPH[⍳4]

ALPH[6⍴24]
```

10.29 Evaluate the following expressions, assuming that *ALPH* has the value assigned in Exercise 10.28:

```
B←'*⎕ +-×'
B[7⍴1 2]

B[7⍴2 3]

B[1+6|(⍳7)∘.+⍳7]

A←ALPH,B

A[9 29 19 9 14 7 29 15 6]
```

10.30 Use the graphing function *GR* of Section 10.12 to evaluate the following expressions:

```
X←⍳8
T←X∘.≤X
GR T

GR ⍉T

M←X∘.⌈X
GR 4<M

GR 5<M

GR ⊖⌽5<M

GR (5<M) ⌈ ⊖⌽5<M
```

10.31 Evaluate the following expressions:

```
M←(⍳8)∘.⌈⍳8
C←' ∘-+×○*⎕'
C[M]

C[5⌊M]

C[5⌈M]

C[M⌈⊖⌽M]
```

CHAPTER 11

11.1 The phrase "define *F* by the expression 3+4×*X*" will be used to mean "Define the function *F* as follows":

```
∇Z←F X
Z←3+4×X∇
```

a) Define *P* by the expression 8+4×*X*

b) Define *Q* as the function inverse to *P*

c) Evaluate the following expressions:

```
Q 0,⍳5

P Q 0,⍳5

P 0,⍳5

Q P 0,⍳5
```

11.2 a) Define *F1*, *F2*, etc., by the following expressions:

```
¯3+2×X

¯8+10×X

¯2+¯10×X

4+3×X

4×X

5+X
```

b) Define functions *G1*, *G2*, etc., which are inverse to the functions *F1*, *F2*, etc.

c) Evaluate the following expressions:

```
X←¯3+⍳5

F1 X

G1 F1 X

G1 X

F1 G1 X
```

d) Repeat Part (c) for each of the other function pairs *F2* and *G2*, *F3* and *G3*, etc.

11.3 Take the four function tables of Exercise 10.1 and replace the first column of each by the vector 2 2.2 2.4 2.6 2.8 3. Analyze each of the functions represented by the new tables. Verify your work by applying each of the resulting expressions to the arguments 2 2.2 2.4 2.6 2.8 3.

11.4 Repeat Exercise 11.3 but replacing the first columns by each of the following vectors:

```
¯7 ¯4 ¯1 2 5 8

¯2.5 ¯1 0.5 2 3.5 5
```

11.5 Make maps to show the application of each of the pairs of inverse functions of Exercise 11.2.

11.6 Draw graphs to represent each of the pairs of inverse functions of Exercise 11.2.

11.7 Define *Q* by the expression *X*⋆3. Graph the function *Q* for argument values from ¯2.5 to 2.5. Draw the graph of the function *R* which is inverse to *Q* and use it to evaluate (approximately) the expression *R* ¯1.3 0 1.27 2.15. Check these results by applying the function *Q* to them.

11.8 Graph the function -*X* and from it obtain the graph for the inverse function. What is the expression for the inverse function?

11.9 Repeat Exercise 11.8 for the function ÷*X*.

11.10 The function *X*⋆2 is called the <u>square</u> function and its inverse is called the <u>square root</u>. Determine the square root of each of the arguments 3, 5, 6, and 4096. Check your results by applying the square function.

11.11 The expression *X*⋆3 is called the <u>cube</u> function and its inverse is called the <u>cube root</u>. Determine the cube root of each of the arguments 3, 5, 6, and 4096. Check your results by applying the cube function.

11.12 Solve each of the following equations:

```
5=3+X

7=4×X

18=4+3×X

248=13+2×X-3

164=¯8+(2×X)-8

164=¯8+(2×X)÷8
```

11.13 Solve each of the following equations:

```
5=X⋆2

6=X⋆3

4096=X⋆3

256=(X-4)⋆2

343=(X+15)⋆3
```


CHAPTER 12

12.1 Show the complete trace of the first four iterations of the function *SQRT* (defined in the text) when applied to each of the arguments 5 and 25 and .25. Check the results by applying the square function to them.

12.2 Show the complete trace of the function *SQT* when applied to the arguments 5 and 25 and .25 (carry all calculations to 7 decimal digits.)

12.3 Show the complete trace of the execution of the expression 4 5 *GRF* 20 for the case where *F* is the square function.

12.4 Show the complete trace of the execution of the expression 3 2 *GRF* 3, where the function *F* is defined as follows:

```
    ∇Z←F X
    Z←5×(X-1.4)×(X-2.6)×(X-4.2)
∇
```

12.5 Write an expression using the function *GRF* which would yield a solution to the equation

```
17=X⋆4
```

and show the appropriate definition of the function *F* used within *GRF*.

12.6 Repeat exercise 12.5 for each of the following equations:

```
29=(X-2)⋆3

265=X⋆5

19=(3+2×X)⋆2

47=(¯2+.5×X)⋆6
```

12.7 Show the complete trace of the execution of the following expressions:

```
GCD 35 133

GCD 133 35

GCD 140 35

GCD 1728 840
```

12.8 a) Evaluate the expression *V*÷*GCD V* for each of the following values of the argument *V*:

```
  6    8

 35  133

 54  318

175 2025

1024  128
```

b) For each of the cases of Part (a) verify that *V* and *V*÷*GCD V* both represent the same rational number, that is, (÷/*V*)=(÷/*V*÷*GCD V*)

c) Apply the function *GCD* to each of the results of Part (a) to verify that the elements of the result have no common factor, that is, their greatest common divisor is 1.

12.9 a) Use the function *A* defined in Section 9.5 (to add rationals) to evaluate the following expressions:

```
3    4 A  1   2

7   20 A  8  45

3    8 A  5  16

74 100 A 13  50
```

b) Apply the function *GCD* to each of the results of Part (a).

12.10 a) Define a dyadic function *PLUS* which adds two rationals (in the manner of the function *A* of Section 9.5), but which yields the result in "reduced form", that is, with the smallest integers possible. Use the functions *A* and *GCD* in the definition.

b) Redefine the function of Part (a) so that the functions *P* and *GCD* are not used within it but are each replaced by statements like those in their definitions.

12.11 Define a function *TIMES* which multiplies rationals and produces the result in reduced form.

12.12 Evalute the expression +/*BIN N* for integer values of *N* from 0 to 7. Give a simple expression which is equivalent to the function +/*BIN N* and test it by evaluating both expressions for the case *N*←12.

12.13 Evaluate the expression -/*BIN N* for values of *N* from 0 to 7. Give a simple expression which is equivalent to the function -/*BIN N*.

12.14 Each of the following functions is equivalent to some primitive function. Evaluate each for a few scalar arguments and identify the equivalent primitive function:

```
    ∇ Z←X A Y
[1]   Z←1
[2]   →3×Y≠0
[3]   Y←Y-1
[4]   Z←X×Z
[5]   →2 ∇
```

```
    ∇ Z←B X
[1]   Z←1
[2]   I←0
[3]   →4×I=X
[4]   I←I+1
[5]   Z←I×Z
[6]   →3 ∇
```

```
    ∇ Z←X C Y
[1]   Z←X
[2]   →3×X<Y
[3]   Z←Y ∇
```

12.15 Without using the complement function (~) itself, define a function *D* which is equivalent to the complement function.

12.16 Repeat Exercise 12.15 for each of the following functions:

Minimum (⌊)

Magnitude (|)

Not-equal (≠)

12.17 a) Without using the residue function (|) itself define a function equivalent to the residue function, at least for non-negative right and left arguments.

b) Modify the function defined in Part (a) so that it is equivalent to the residue function for negative as well as positive right arguments.

12.18 a) Use the ceiling function (⌈) to define a function equivalent to the floor function (⌊).

b) Without using any of the ceiling, floor, or residue functions, define a function which is equivalent to the floor function for non-negative arguments.

c) Modify the function defined in Part (a) to make it apply to negative as well as non-negative arguments.

12.19 Consider the function *W* defined as follows:

```
    ∇ Z←W N
[1]   Z←2
[2]   I←2
[3]   I←I+1
[4]   →5+3×I>N
[5]   →6-3×⌈/0=Z|I
[6]   Z←Z,I
[7]   →3 ∇
```

Evaluate *W N* for a few different values of *N* and state in words what the function *W* does. (For integer arguments greater than 1 it is equivalent to a function defined in an earlier chapter).

CHAPTER 13

13.1 Evaluate the following expressions:

```
A←1 2 3 4 5
B←5 4 3 2 1
+/A×B

+/A⌈B

⌊/A⌈B

+/A≤B

⌊/A≤B

⌈/A≤B

×/A-B

+/A|B

+/A*B

+/B*A

C←¯10 3 14 ¯8 0 2
D←5 ¯7 2 ¯6 ¯1 3
+/C×D

⌈/C⌊D

⌊/C⌈D

⌈/(|C)⌊(|D)

⌊/(|C)⌊(|D)

+/C≤D

+/C=D

+/C-D
```

13.2 State in words what the following expressions mean. For example, the first one means the number of positions in which the elements of *Q* exceed the corresponding elements of *P*:

```
+/P<Q

+/P=Q

⌊/P≠Q

⌈/P=Q

×/P+Q

⌈/P+Q
```

13.3 Rewrite each of the expressions of Exercise 13.1 in inner product form.

13.4 Evaluate the following expressions:

```
P←2 3 5 7 11
E←2 0 2 0 1
F←1 1 1 1 0
P×.*E

P×.*F

P×.*E⌊F

P×.*E⌈F

2 3 5 7 11×.*2 0 2 0 1

1 0 1 1 0+.×ι5

¯1 1 ¯1 ¯1 1+.×ι5

(¯1*1 0 1 1 0)+.×ι5

(¯1*1 0 1 1 0)+.×P

(P<7)+.×P

(P≠5)+.×P

(¯1*P≠5)+.×P

P⌊.=E

P⌊.=P
```



```
P⌊.=F

1 2 3 4⌊.=ι4
```

13.5 Evaluate each of the following expressions:

```
1 3 3 1+.×(5*0 1 2 3)

X←5
C←1 3 3 1
E←0 1 2 3
C+.×(X*E)

(X*¯1+ιρC)+.×C

(X+1)*3

D←1 2 1
(X*¯1+ιρD)+.×D

(X+1)*2

B←1 4 6 4 1
(X*¯1+ιρB)+.×B

(X+1)*4

X←7
(X*¯1+ιρD)+.×D

(X+1)*2

(X*¯1+ιρC)+.×C

(X+1)*3

(X*¯1+ιρB)+.×B
```

13.6 Evaluate the following expressions:

```
X←1 2 3 4 5 6
A←3ρX
A

ρA

B←8ρX
B

ρB

1,(5ρ4 2),1

7ρ1 0
```

13.7 Evaluate the following expressions:

```
X←1 2 3 4 5 6
M←2 3ρX
M

ρM

N←3 5ρX
N

ρN

4 3ρι12

⍉3 4ρι12

ρ4 3ρι12

ρ⍉4 3ρι12
```

13.8 Let *M* and *N* be the following matrices:

```
      M                    N

 4 ¯6  3          2  0  5 ¯1
¯1  0 ¯4         ¯3  1  6  0
                  1 ¯2  3 ¯4
```

Then evaluate the following expressions:

```
M+.≤N

M⌊.+N

M+.⌊N

M+.×N

⍉(⍉N)+.×(⍉M)

M+.=N

⍉(⍉N)+.=⍉M

M+.≤N

⍉(⍉N)+.≤(⍉M)

⍉(⍉N)+.>(⍉M)
```

13.9 State in words what each of the first six expressions of the preceding exercise represent.

13.10 Let Q and C be specified as follows:

```
Q←1 5⍴0 1 2 3 4 and
C←5 5⍴1 1 1 1 1 0 1 2 3 4 0 0 1 3
      6 0 0 0 1 4 0 0 0 0 1
```

Then Q and C are the following matrices:

Q

0	1	2	3	4

C

1	1	1	1	1
0	1	2	3	4
0	0	1	3	6
0	0	0	1	4
0	0	0	0	1

Now evaluate the following expressions:

```
X←3
(X*Q)+.×C

(X+1)*Q

X←4
(X*Q)+.×C

(X+1)*Q

(7*Q)+.×C

(7+1)*Q
```

13.11 Evaluate the following expressions:

```
M←(⍳5)∘.≤⍳5
M

X←2 3 5 7 11
X+.×M

(+/1↑X),(+/2↑X),(+/3↑X),
     (+/4↑X),(+/5↑X)

M+.×X

(⍉M)+.×X

X×.*M

(×/1↑X),(×/2↑X),(×/3↑X),
     (×/4↑X),(×/5↑X)

X×.*⍉M
```

13.12 Let the matrices I and D be defined as follows:

I

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

D

1	0	0	0	0
¯1	1	0	0	0
0	¯1	1	0	0
0	0	¯1	1	0
0	0	0	¯1	1

Then evaluate the following expressions:

```
X←2*⍳5
X

I+.×X

I+.×14 3 16 ¯7 0

D+.×X

D+.×14 3 16 ¯7 0
```

13.13 a) Write an expression using outer product to define the matrix I of Exercise 13.12.

b) Write an expression using outer products to define the matrix D of Exercise 13.12.

c) Modify the expressions derived in Parts (a) and (b) to define similar matrices of any specified dimension N.

d) The expression `I+.×X` is a function of the vector X. State in words what this function is.

e) The function `D+.×X` is closely related to the difference function defined in Section 10.6. State exactly what this relationship is.

f) State in words how the matrix D should be modified to produce a matrix $D1$ such that the function `D1+.×X` is exactly the difference function of Section 10.6.

g) Write an expression using outer products to define the matrix $D1$ of part (f).

13.14 Let D be the matrix defined in Exercise 13.12, and let S be the following matrix:

S

1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1

a) Evaluate the following expressions:

```
X←1 4 9 16 25
D+.×X

S+.×(D+.×X)

S+.×D

(S+.×D)+.×X

S+.×X

D+.×(S+.×X)

D+.×S

(D+.×S)+.×X
```

b) State in words the relation between the functions `D+.×X` and `S+.×X`.

13.15 Let M be the following matrix:

M

2	3	1	4
0	1	2	0
2	3	2	4
0	1	0	0
1	0	0	1

a) Evaluate the following expressions:

```
P←2 3 5 7 11
N←P×.*M
N

GCD←P×.*⌊/M
GCD

N÷GCD
```

b) Verify that GCD is the greatest common divisor of the elements of N.

c) Choose any other value for M, except that the matrix must have 5 rows and must contain only non-negative integer elements. Then repeat Parts (a) and (b).

13.16 a) Using the matrix M of Exercise 13.15, evaluate the following expressions:

```
P←2 3 5 7 11
N←P×.*M
N

LCM←P×.*⌈/M
LCM

LCM÷N
```

b) Verify that LCM is the least common multiple of the elements of N.

c) Choose another value for M (as in Exercise 13.15 (c)) and repeat Parts (a) and (b).

13.17 Let *M* be the matrix:

```
      M
  2   3  ¯5
  0   1   2
  4  ¯2   2
```

a) Evaluate the following expressions:

```
A←(M[;1]×2)+(M[;2]×1)+(M[;3]×3)
B←M+.×2 1 3
V←2 ¯4 3
C←(M[;1]×V[1])+(M[;2]×V[2])
     +(M[;3]×V[3])
D←M+.×V
```

b) Display and compare the values of *A* and *B* and of *C* and *D*. State in words the relationship this comparison suggests.

c) Test the relationship you expressed in Part (b) by evaluating *C* and *D* for several different values of *V* and of *M*.

13.18 Follow the steps of Exercise 13.17 to establish a similar relationship between the expression *V*+.×*M* and expressions involving the <u>rows</u> of *M*.

13.19 a) Evaluate the following expressions:

```
X←4
X*0 1 2 3

5 2 0 1×X*0 1 2 3

+/5 2 0 1×X*0 1 2 3

E←0 1 2 3
+/5 0 0 0×X*E

+/0 5 0 0×X*E

5×X*1

+/0 0 7 0×X*E

7×X*2
```

b) Identify each of the curves of Figure 13.1, labelling each as a "first term", "second term", etc.

13.20 Let the functions *SUM* and *TERMS* be defined as follows:

```
    ∇ Z←SUM X
[1]   Z←+/X∇

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

Evaluate the following expressions:

```
C←2 1 0 4
X←5
C TERMS X

SUM C TERMS X
```

13.21 Repeat Exercise 13.20 for the following values of *X* and *C*:

```
X      C
4    1 3 3 1
5    0 0 0 1
5    1 3 3 1
6    0 0 0 1
0    1 3 3 1
1    0 0 0 1
2    1 4 6 4 1
3    0 0 0 0 1
```

13.22 Use the function *POL* defined in Section 13.6 to evaluate the following expressions:

```
5 0 7 2 POL 0 1 2 3 4 5

¯5 0 ¯7 2 POL 0 1 2

¯5 0 ¯7 2 POL ¯4 ¯3 ¯2
```



```
1 1 POL 0,⍳5

0 1 POL 1+0,⍳5

1 2 1 POL 0,⍳5

0 0 1 POL 1+0,⍳5

1 3 3 1 POL 0,⍳5

0 0 0 1 POL 1+0,⍳5
```

13.23 Use the difference function *D* defined in Section 10.6 to evaluate the following expressions:

```
V←5 0 2 3 POL 0,⍳5
V

D V

D D V

D D D V

W←5 0 2 3 1 POL 0,⍳7
W

D D D D W
```

13.24 Use the matrix *S* defined in Exercise 13.14 to evaluate the following expressions:

```
N←⍳5
S+.×N

0 0.5 0.5 POL N

S+.×N*2

(0 1 3 2÷6) POL N

S+.×N*3

0 0 0.25 0.5 0.25 POL N

S+.×N*0

0 1 POL N
```

CHAPTER 14

14.1 For each of the dyadic functions + - × ÷ ∧ ∨ < ≤ = | ⌈ and ⌊, state:

a) Whether you think it is commutative or not.

b) An example proving that the function is non-commutative for each case you declare to be non-commutative.

14.2 Modify the function *COM* defined in Section 14.2 so as to include in its domain all of the function symbols appearing in Exercise 14.1.

14.3 a) Make tables to prove that the functions <u>and</u> and <u>or</u> are commutative.

b) Evaluate the following expressions:

```
0∧1

0∨1

0⍲1

0⍱1

X←0 0 1 1
Y←0 1 0 1
X∧Y

X∨Y

X⍲Y

X⍱Y
```

14.4 Use the method of exhaustion to examine the commutativity (or non-commutativity) of the dyadic functions < ≤ = ≥ > ≠ and ⌊.

14.5 Make a table similar to Table 14.5 to prove that the minimum function is associative.

14.6 Make a table (of 8 cases labelled 0 0 0 and 0 0 1 and 0 1 0, etc., to 1 1 1) which will show whether the <u>and</u> function is associative.

14.7 Repeat Exercise 14.6 for each of the following functions: ∨ ⍲ ⍱.

14.8 a) Write an example to show that addition does not distribute over multiplication.

b) Write an example to show that addition does not distribute over itself.

c) Write an example to show that multiplication does not distribute over itself.

d) Write a few examples to <u>illustrate</u> that multiplication distributes over addition (include some negative numbers in the example).

e) Complete the following table so as to summarize the foregoing results, using a 1 to denote commutativity and a 0 to denote non-commutativity:

	+	×
+		
×		

14.9 Extend the table of Exercise 14.8 (e) to include the functions + × - ⌈ and ⌊. You are not expected to provide proofs of commutativity, but test the matter thoroughly by evaluating a number of expressions looking for values which will prove non-commutativity. Be sure to use some negative values in this search. For each function stated to be non-commutative, give an example which proves it so.

14.10 Make tables to determine whether:

a) ∨ distributes over ∧

b) ∨ distributes over ∨

c) ∧ distributes over ∧.

14.11 Summarize the results of Table 14.6 and of Exercise 14.10 in a distributivity table of the form

	∨	∧
∨		
∧		

entering a 1 in the *I*th row and *J*th column of the table if the function heading the *I*th row distributes over the function heading the *J*th column, and a 0 otherwise.

14.12 Extend the distributivity table of Exercise 14.11 to include the functions ∨ ∧ ⍲ and ⍱. Make tables of the form of Table 14.6 to develop any results you may need for this table.

14.13 a) Make a table similar to Table 14.7 to prove that addition distributes over maximum.

b) make a table to test whether subtraction distributes over maximum.

c) If in Exercise 14.9 you concluded that multiplication distributes over maximum, then evaluate the following pair of expressions and compare the results:

```
¯6×4⌈9

(¯6×4)⌈(¯6×9)
```

14.14 Repeat Exercise 14.13 substituting minimum for maximum.

14.15 Make a table of the form of Table 14.8 to summarize all of the results obtained thus far. Enter 0's and 1's <u>only</u> for results that have been <u>proven</u>, and leave other entries blank. Include the dyadic functions + - × ÷ ⌈ ⌊ ∨ ∧ ⍱ and ⍲. Fill out blank spaces in the table by constructing further proofs if you wish.

14.16 The proof (i.e., derivation) that *(A+B)×C* is equivalent to *(A×C)+(B×C)* which was given in Section 14.5 can be illuminated by evaluating each expression occurring in it for some chosen value of *A*, *B*, and *C*. For example, if *A*←3 and *B*←7 and *C*←4, the illumination would appear as follows:

```
      (3+7)×4
40
      4×(3+7)
40
      (4×3)+(4×7)
40
      (3×4)+(4×7)
40
```

Illuminate the proof for each of the following values of *A*, *B*, and *C*:

A	B	C
3	14	¯8
¯3	5	7
¯3	¯5	¯7

14.17 a) Prove that `(P⌊Q)⌈R` is equivalent to `(R⌈P)⌊(Q⌈R)`. Use the first such proof in Section 14.5 as a model, writing the justification of each step to the right of it.

b) Choose values of *P*, *Q*, and *R* and illuminate the proof in the manner defined in Exercise 14.16.

14.18 Repeat Exercise 14.17 to show the equivalence of each of the following pairs of expressions:

```
A∧(B∧C)
C∧(B∧A)

A+(B+C)
C+(B+A)

A×B×C×D
D×C×B×A
```

14.19 For each of the proofs of Exercises 14.17 and 14.18 add the abbreviated form of the note to the right of each note in the proof.

14.20 Choose values of *A*, *B*, *C*, and *D* and use them to illuminate the proof (given in the text) that `(A+B)×(C+D)` is equivalent to `(A×C)+(A×D)+(B×C)+(B×D)`

14.21 Make (and illuminate) proofs for the following pairs of equivalent statements:

```
(A⌊B)+(C⌊D)
(A+C)⌊(A+D)⌊(B+C)⌊(B+D)

A∧(B∨C∨D)
(A∧B)∨(A∧C)∨(A∧D)
```

14.22 a) Determine a value of the vector *C* such that the expression `+/C×X*0 1 2` is equivalent to the expression `×/X+4 1`.

b) Evaluate the expressions in Part (a) for several values of *X* and compare the results (which should agree).

14.23 Repeat Exercise 14.22 for each of the following expressions:

```
(X+4)×(X+1)

×/X-4 1

×/X+1 1

×/X+1 0

×/X+0 1

(X+¯1)×(X+¯1)

(X-1)×(X-1)

R←3 5
×/X+R

×/X+(-R)

×/X-R
```

14.24 Choose vector values of the arguments to illuminate the proof illuminated in Exercise 14.16.

14.25 Chose vector values to illuminate each of the proofs of Exercise 14.18.

14.26 Evaluate the following expressions:

```
A←3 ¯8 15 6
B←¯5 0 18 43 ¯7
+/A,B

(+/A)+(+/B)

⌈/A,B

(⌈/A)⌈(⌈/B)

⌊/A,B

(⌊/A)⌊(⌊/B)

×/A,B

(×/A)×(×/B)

-/A,B

(-/A)-(-/B)

C←1 0 1 0 1
D←0 1 1
∨/C,D

(∨/C)∨(∨/D)
```

14.27 Evaluate the following expressions:

```
A←3 ¯8 15 6
B←4 2 ¯1 4
+/A+B

(+/A)+(+/B)

×/A×B

(×/A)×(×/B)

⌈/A⌈B

(⌈/A)⌈(⌈/B)

-/A-B

(-/A)-(-/B)
```

14.28 Use each of the following pairs of values of *V* and *W* to illuminate the identity expressed by Theorem 4:

V	W
1 10 2 3	2 0 5
2 0 5	1 5 2 3
¯3 10 2 ¯8	2 0 ¯2 ¯3 1

14.29 Use the following values to illuminate Theorem 5:

```
A← 3 ¯1 0 4 2
B← 5 ¯2 6
P←¯2 2 1 0 5
Q← 7 2 ¯4
```

14.30 a) Repeat Exercise 14.29, substituting the function + for every occurrence of × in Theorem 5.

b) Repeat Part (a) using ⌈ instead of +.

14.31 Use the values of *A*, *B*, *P*, and *Q* from Exercise 14.29 and the values `I←4` and `J←2` to illuminate the proof of Theorem 5.

14.32 Use the following sets of values of *A*, *B*, and *C* to illuminate Theorem 6:

A	B	C
3	2	4
¯2	3	5
3	4	¯4

14.33 Choose some values for *X*, *E*, and *F* and use them to illuminate Theorem 7.

14.34 a) For each of the following pairs of values of *A* and *B*, determine a vector *D* such that the expression *D P X* is equivalent to (*A P X*)+(*B P X*) (where *P* is the polynomial function defined in Section 14.8):

A	B
2 1 4	3 ¯2 5
6 18 4 2	¯3 ¯3 8 ¯4
2 0 4 8	0 0 0 2

b) Verify each of the foregoing results by evaluating the expressions *D P X* and (*A P X*)+(*B P X*) for *X*←¯3+⍳5.

14.35 Repeat Exercise 14.34 for the following values of *A* and *B*:

A	B
6 1 2	3 0 ¯4 8 2
2 1 3 ¯2 4	2 0 1

14.36 Repeat Exercises 14.34 and 14.35 but with the expression (*A P X*)+(*B P X*) replaced by (*A P X*)×(*B P X*).

14.37 For each of the following expressions determine the coefficients of an equivalent polynomial:

```
×/X+2 3

×/X+4 7

×/X+7 4

×/X+(-7 4)

×/X-7 4

×/X+¯7 ¯4

×/X+2 3 4

×/X+4 3 2

×/X+3 2 4

×/X-0 1

×/X-0 1 2

×/X-0 1 2 3
```

14.38 a) For each of the following expressions determine the coefficients of an equivalent polynomial:

```
×/X+1

×/X+1 1

×/X+1 1 1

×/X+4⍴1

×/X+5⍴1

×/X+6⍴1
```

b) Compare the results of Part (a) with the binomial coefficients of Section 12.4.

14.39 Let *M* be the following matrix:

```
1  0  0  0
0  1 ¯1  2
0  0  1 ¯3
0  0  0  1
```

a) Compare the columns of *M* with the coefficients of polynomials equivalent to the factorial polynomials and state how the columns correspond to the degrees of the factorial polynomials. (Note that final zeros appended to a vector of coefficients make no difference to the value of the polynomial).

b) Evaluate the following expression:

```
V←0,1,(3÷2),(2÷6)
A←M+.×V
A
```

c) Use the results of Exercise 13.17 (in Chapter 13) to state in words the relation between the result of Part (b) and a certain weighted sum of the columns of *M* (that is, of the coefficients of polynomials equivalent to the factorial polynomials).

d) Use the vector *A* of Part (b) and the polynomial function *P* defined in the text to evaluate the expression *A P X* for several values of *X*. Compare the results with the evaluation of +/(⍳*X*)*2 for the same values of *X*.

e) Explain the agreements obtained in Part (d).

14.40 Exercise 14.39 illustrated how the expression *M*+.×*V* would yield the coefficients of a polynomial equivalent to the sum of *V*[1] times the 0-degree factorial polynomials, *V*[2] times the 1-degree factorial polynomial, etc. Apply this result to obtain the coefficients of a polynomial equivalent to +/(⍳*X*)*3 as follows:

a) Extend the matrix *M* to be a 5 by 5 matrix incorporating the coefficients for the next factorial polynomial.

b) Evaluate +/(⍳*X*)*3 for a number of values of *X* beginning with 0.

c) Use the difference table method of Section 10.8 to determine an equivalent function (expressed as a weighted sum of factorial polynomials).

d) Evaluate the expression *Q*←*M*+.×*R*÷!0 1 2 3 4, where *R* is the first row of the difference table.

e) Compare *Q P X* and +/(⍳*X*)*3 for a number of values of *X*.

14.41 Use mathematical induction to prove that the functions +/(⍳*X*)*2 and (+/0 1 3 2×*X**0 1 2 3)÷6 are equivalent.

CHAPTER 15

15.1 For each of the following linear expressions, write an equivalent expression in terms of a single vector argument *V*, where *V*←*X*,*Y* or *V*←*X*,*Y*,*Z* or *V*←*W*,*X*,*Y*,*Z* as appropriate:

```
3+(4×X)+(5×Y)

¯4+(6×X)+7×Y

¯4+(6×Y)+7×X

3+(¯6×X)+0×Y

3+(¯6×X)

¯8+(0×X)+¯9×Y

¯8+¯9×Y

-(8+9×Y)

0+(3×X)+(¯6×Y)

(3×X)+(¯6×Y)

(3×X)-(6×Y)

4-(3×X)+7×Y

8+(2×X)+(5×Y)+(10×Z)

8+(2×X)+(0×Y)+(10×Z)

¯4+(2×X)+(10×Z)

18+10×Z

4+(3×X)+(0×Y)+(0×Z)

4+(3×X)

X+Y+Z

Z+(2×Y)+(4×X)

X-Y-Z

X+Y+Z+W
```

15.2 Take each result of Exercise 15.1 and (without looking at the original expression in the exercise) write an equivalent expression in terms of the arguments *X* and *Y* (and if necessary, *Z* and *W*). Compare your results with the original expressions.

15.3 Let *X*←3 and *Y*←2 and *Z*←4 and *W*←15 and let *V*←*X*,*Y* or *V*←*X*,*Y*,*Z* or *V*←*X*,*Y*,*Z*,*W* as appropriate. Then evaluate each expression of Exercise 15.1 and evaluate each equivalent expression which you obtained and compare the results.

15.4 a) Determine a vector *A* and a matrix *B* such that the expression *A*+*B*+.×*X*,*Y* is equivalent to the following pair of expressions:

```
3+(2×X)+(¯4×Y)
4+(¯3×X)+(2×Y)
```

More precisely, *A*+*B*+.×*X*,*Y* is equivalent to the catenation of these expressions, that is:

```
(3+(2×X)+(¯4×Y)),4+(¯3×X)+(2×Y)
```

b) Evaluate *A*+*B*+.×*X*,*Y* and compare the result with the result of evaluating the given expressions for each of the following pairs of values of *X* and *Y*:

X	Y
2	5
3	0
0	3
0	0
¯4	2
3	¯7
¯9	¯3

c) Take the result of Part (a) and from it write the equivalent expressions in terms of *X* and *Y* and compare with the original expressions.

15.5 Repeat Exercise 15.4 for the following pairs of expressions:

```
¯3+(4×X)+(¯2×Y)
6+(2×X)+(7×Y)

¯3-(¯4×X)+(2×Y)
6-(¯2×X)+(¯7×Y)

(3×X)+(7×Y)
(4×Y)+(8×X)

2+3×X
8+7×Y
```

15.6 Choosing any values that you wish for *Z* in the evaluations, repeat Exercise 15.4 for the following set of expressions:

```
18+(3×X)+(¯4×Y)+(7×Z)
¯13+(2×Y)
2+(0×X)+(3×Y)+(¯4×Z)
```

15.7 a) Plot the mapping produced by the expression *A*+*B*+.×*V* for the following set of values:

```
   A            B         V
3 ¯5          2 ¯1       2 1
             ¯3  4
```

b) Add to the plot of Part (a) the mappings for each of the following 7 values of *V* (shown in columns to save space):

```
¯2  0  0  1  1  ¯1  ¯1.4
¯1  0  1  0  1  ¯1    .2
```

c) Make other maps for any values of *A* and *B* that you wish to choose. For each case try to find some value of *V* which (like the last one in Part (b)) maps into the origin (that is, the point 0 0).

15.8 Repeat Exercise 15.7 but with *A* assigned the value 0 0 in every case.

15.9 Let *B* be the following matrix:

```
 .5   .866
¯.866  .5
```

a) Plot the mapping *B*+.×*V* when applied to each of the set of points *V* listed in exercise 15.7 (b).

b) Verify that this mapping is a rotation.

15.10 Repeat Exercise 15.9 for each of the following values of the matrix *B*:

```
 0 1     0 ¯1     ¯1  0     1 0
¯1 0     1  0      0 ¯1     0 1

 .707 .707        .707 ¯.707
¯.707 .707        .707  .707
```

15.11 a) Let *B* be the matrix of Exercise 15.9. Then plot the mappings produced by repeated applications of *B* to the point *V*←1 2, that is:

```
B+.×V
B+.×B+.×V
B+.×B+.×B+.×V
```

and so forth.

b) How many applications of *B* are equivalent to the identity function?

c) Write an expression of the form B+.×B+.×B+.×B, with N occurrences of B, where N denotes the answer to Part (b). Evaluate this expression and compare the result with the identity matrix.

15.12 a) Repeat Exercise 15.11 for each of the matrices of Exercise 15.10.

b) Determine a rotation matrix whose first and last elements are equal to .2 and repeat Exercise 15.11 for this matrix.

15.13 a) Let B be a rotation matrix with elements S, C, -C, and S as defined at the beginning of Section 15.3. Show that the product B+.×⍉B is the identity matrix.

b) Show that (⍉B)+.×B is the identity matrix.

c) Test these results by applying them to the rotation matrices of Exercise 15.10

15.14 Plot the mapping produced by the translation 3 ¯5+V applied to each of the points V of Exercise 15.7 (b).

15.15 Let M be the matrix given for V in Exercise 15.7 (b), that is, the columns of M are the values of V in the order shown.

a) Evaluate the expression B+.×M, where B is the matrix of Exercise 15.9. Compare the results with those of Exercise 15.9.

b) Repeat Part (a) for the matrices B listed in Exercise 15.10.

15.16 Define a matrix P to be used with the matrices B and M of Exercise 15.15 in the expression P+B+.×M to produce the translation 3 ¯5.

15.17 Use the matrices P and M of Exercise 15.16 and the matrix B←2 2⍴0 1 ¯1 0 and plot the mappings produced by each of the following expressions:

```
P+B+.×M

B+.×P+M

(B+.×P)+(B+.×M)
```

15.18 a) Define a stretching matrix B and apply it to the matrix M of Exercise 15.15, that is, evaluate the expression B+.×M.

b) Compare the matrices M and B+.×M and state the relation between them.

c) Repeat Part (a) for a number of stretching matrices which you choose.

15.19 a) Choose a number of matrices and use them to test the distributivity of the inner product +.× over +.

b) Choose a number of matrices and use them to test the associativity of the +.× inner product.

15.20 Let A,B, and C be 2-by-2 matrices and give names to each of the elements according to the following scheme:

```
A11 A12     B11 B12     C11 C12
A21 A22     B21 B22     C21 C22
```


a) For each of the following expressions write an equivalent expression in terms of the names A11, A12, etc:

```
B+.×C

A+.×(B+.×C)

(A+.×B)+.×C
```

b) Prove that the expression obtained for the second case of Part (a) is equivalent to the expression obtained for the third case. (This proves the associativity of +.× for 2-by-2 matrices.)

15.21 Repeat Exercise 15.20, replacing the second and third expressions of Part (a) by the following expressions

```
A+.×(B+C)

(A+.×B)+(A+.×C)
```

(This proves that +.× distributes over + for 2-by-2 matrices.)

15.22 a) Make a 3-dimensional plot of the eight points represented by the following matrix M:

```
            M
1  2  3  0  0  0 ¯1  1
1  2  3  1  2  0  2 ¯1
1  2  3  1  2  0 ¯3  1
```

b) Evaluate the expression B+.×M for the following matrix B:

```
   B
2  0 ¯1
1 ¯2  1
1  1  1
```

c) Add to the plot the points determined in Part (b) and show the mapping produced by the matrix B.

15.23 a) Choose any three 3 by 3 matrices C D and E and use them to test the associativity of the +.× inner product in three dimensions.

b) Use the same matrices to test the distributivity of +.× over +.

15.24 a) Make a plot to show the mapping B+.×M, where B is the following 3-dimensional rotation matrix:

```
1     0      0
0   ¯.707   .707
0   ¯.707   .707
```

and M is the matrix of points given in Exercise 15.22.

b) Repeat Part (a) for any 3-dimensional rotation matrices you may wish to construct.

15.25 a) Evaluate the following expressions:

```
X←0 1 2 3 4 5 6 7 8 9 10
Y←⌽X
Y

M←(2×Y)∘.+(X-12)
M

N←Y∘.+¯1×X
N

0=M

0=N

' ⋆'[1+0=M]
```

```
' *'[1+0=N]

(0=M)∨(0=N)

(0=M)∧(0=N)
```

b) Discuss the results of Part (a), stating as clearly as you can what each of the logical matrices represent.

c) Repeat Part (a) for various values of *X* and *Y* and for various linear functions of your own choosing.

CHAPTER 16

16.1 a) Test the fact that the 2-dimensional matrices *B* and *IB* given in Section 16.2 actually produce inverse functions by applying them to the set of points represented by the following matrix *M*:

```
1  2  0  1 ¯3 ¯5  1  0
2  5  0  1  5 ¯2  0  1
```

b) Evaluate the expressions `B+.×IB` and `IB+.×B` and compare them with the identity matrix.

16.2 Repeat Exercise 16.1 for the 3-dimensional matrices *B* and *IB* given in Section 16.2 and for the following matrix *M*:

```
 9 ¯3  1  0  0 ¯8  0
16  5  0  1  0  1  0
20 ¯7  0  0  1  5  0
```

16.3 a) Evaluate the expression `∧/3 ¯7=B+.×V` for the matrix `B←2 2ρ1 ¯3 ¯2 4` and for each of the following values of the 2-element vector *V*:

```
1  0.5  4.5 ¯3.2  1  0
2  3.5  0.5  4.2  0  1
```

b) Use the results of Part (a) to determine which of the given values of *V* is a solution of the equation `3 ¯7=B+.×V`.

16.4 Let *M* and *N* be the following matrices:

```
            M
¯7  5 ¯1  1 ¯5 ¯9
 3  3  8 ¯2  0  6

                N
¯18 ¯10  10 ¯14   2  ¯2
 39   5  10  22 ¯11  41
```

Each column of *M* (that is `M[;I]`) is a solution of the equation `N[;J]=B+.×M[;I]` for some *J*th column of *N*, where *B* is the matrix `B←2 2ρ2 0 ¯1 5`. Determine which column of *M* gives the solution of the equation for each column of *N*.

16.5 If `B←2 2ρ2 3 3 5`, then the basic solutions *V1* and *V2* are among the columns of the following matrix:

```
1  1 ¯3  0  5 ¯1.5  2
0 ¯2  2  1 ¯3  3.5  0
```

a) Determine the basic solutions of *B*

b) Using the values of *V1* and *V2* obtained in Part (a), evaluate the following expressions:

```
N←(4×0 1)+(¯2×1 0)
N

V←(4×V1)+(¯2×V2)
V

B+.×V

∧/N=B+.×V
```

c) Use the scheme suggested by Part (b) to determine a solution to the equation `N=B+.×V` for each of the following values of *N*:

```
 5  7
¯3  8
 0  0
¯7  0
 0  4
```

16.6 The basic solutions for the matrix *B*←2 2⍴4 2 7 3 also occur among the columns of the matrix give in Exercise 16.5. Use this fact to repeat the work of Exercise 16.5 for this value of *B*.

16.7 Let *B* be the following matrix:

```
2  3
3  5
```

a) Determine a value for *VA* such that the second element of *B*+.×*VA* is zero.

b) Determine a value of *K* such that if *V1*←*VA*÷*K*, then *V1* is a basic solution of *B*.

16.8 The vector *VA*←0 0 would satisfy the requirement imposed in Part (a) of Exercise 16.7, namely that the second element of *B*+.×*VA* must be zero. Try to use this value of *VA* to determine a basic solution *V1* as in Part (b) of the same exercise. Why does it not work?

16.9 Repeat Exercise 16.7 for each of the following values of *B*:

```
4  2      2  3       8  6
7  3      2  8      ¯6  8
```

16.10 a) Repeat the steps of Exercise 16.7 but modified to determine the <u>second</u> basic solution *V2*.

b) Repeat Part (a) for the matrices of Exercise 16.9.

16.11 Determine basic solutions for each of the following matrices:

```
2 7     4  3     16  5     6 9
1 3     8 11     ¯8 10     3 5
```

16.12 a) Evaluate the determinant of each matrix of Exercise 16.11

b) Evaluate the determinant of each matrix of Exercise 16.9

16.13 a) construct a matrix *B* whose determinant is 4

b) If the determinant of *B* is 4, what is the determinant of the matrix -*B*?

c) Modify the matrix *B* of Part (a) to obtain a matrix whose determinant is ¯4

d) Construct at least 3 different matrices whose determinants have the same value 100

e) Construct at least 3 different matrices whose determinants have the value 1.

16.14 What effect does each of the following changes to a matrix have on the value of its determinant:

a) Interchanging its two rows?

b) Interchanging its columns?

c) Interchanging the rows and then interchanging the columns?

e) Changing the sign of every element?

16.15 a) Evaluate the determinant of the following matrix:

```
6  12
4   8
```

b) Is it possible to determine basic solutions for this matrix?

c) Construct at least three different matrices for which it is impossible to determine basic solutions.

16.16 Determine the matrix *BS* which gives the basic solution in matrix form for each of the following matrices:

```
3  7     8  4
1  3     5  3
```

16.17 Determine the matrix of the basic solutions for each of the matrices of Exercise 16.11 and compare the results with those of Exercise 16.11.

16.18 a) Use the results of Exercises 16.16 and 16.17 to determine the solution of the equation 3 13=*B*+.×*V* for each of the matrices *B* involved in those exercises.

16.19 Find solutions to the equation

∧/*N*=(2 2⍴7 5 5 3)+.×*V*

for each of the following values of *N*:

```
10  23
14  12
17   3
 1   0
 0   1
```

16.20 a) Determine *BS* as the matrix of basic solutions for the matrix *B*←2 2⍴9 4 4 2

b) Evaluate the expressions:

B+.×*M*

BS+.×*B*+.×*M*

BS+.×*M*

B+.×*BS*+.×*M*

for the matrix *M* given below:

```
1  3  1  ¯7  0  0  25
1  5  0   6  1  0   3
```

16.21 Repeat Exercise 16.20 for each of the following values of the matrix *B*:

```
4  7     13 ¯3     12  2
8 11      3  7     11  6
```

16.22 a) For the matrices *B* and *BS* of Exercise 16.20, evaluate the following expressions:

B+.×*BS*

BS+.×*B*

b) Repeat Part (a) for each of the pairs *B* and *BS* of Exercise 16.21

16.23 If *BS* is the matrix of basic solutions for *B*, then *B*+.×*BS* is always equal to *BS*+.×*B* (since each is equal to the identity matrix). This might suggest that the function +.× is commutative. Show that this is not so by constructing at least one pair of matrices *C* and *D* such that *C*+.×*D* is <u>not</u> equal to *D*+.×*C*.

16.24 a) Use the Gauss-Jordan method to determine the matrix *BS* of basic solutions for the matrix *B* of Exercise 16.20. Show all of your work.

b) Repeat Part (a) for each of the matrices of Exercise 16.21.

16.25 a) Apply the efficient method of Section 16.13 to solving the equation

∧/3 ¯11=B+.×V

for the matrix *B* of Exercise 16.20. Show all of your work.

b) Repeat Part (a) for each of the matrices of Exercise 16.21.

16.26 a) Use the Gauss-Jordan method to determine the matrix *BS* which is inverse to the following matrix *B*:

```
3  1  4
5  8  2
1  7  1
```

carry all calculations to 4 decimal places.

b) Check your result by evaluating the expression *B*+.×*BS*.

c) Use the matrix *BS* to obtain the solution to the equation ∧/2 ¯5 6=B+.×V

16.27 Repeat Exercise 16.26 for each of the following matrices:

```
¯5  2  ¯7       12  ¯8   4
 8  1  ¯3        3  17   2
 1  4   2        1   9  16
```

16.28 Apply the efficient method of solution to solve the following equation:

∧/12 3 14=B+.×V

where *B* is the following matrix:

```
 10   3  14
 ¯2  12   1
 ¯4   7  15
```

16.29 Evaluate the expression ⌹*B*, where *B* is the matrix of Exercise 16.28.

16.30 Define a function *F* which is equivalent to the function ⌹ when applied to a 2 by 2 matrix argument.

16.31 Define a function *G* which is equivalent to the function ⌹ when applied to a 3 by 3 matrix argument. Base the function definition on the Gauss-Jordan method and use iteration as much as possible.

16.32 Modify the definition of the function *G* of Exercise 16.31 so that it applies to a square matrix argument of any dimension.

16.33 Apply the efficient method of Section 16.13 to the 5 by 6 matrix given in Section 16.16. Compare the result with the solution *C* given in the same section.

16.34 Apply the general curve fitting process to the following function table:

X	Y
1	1
3	6
8	36

	NAME	SYMBOL	DEFINITION OR EXAMPLE	SECTION #
D	Addition	+	3+4←→7	1.2
Y	Multiplication	×	3×4←→12	1.2
A	Subtraction	-	3-4←→¯1	3.1
D	Division	÷	3÷4←→.75	5.1
I	Maximum	⌈	3⌈4←→4	2.4
C	Minimum	⌊	3⌊4←→3	2.4
	Power	⋆	3⋆4←→81 A⋆B←→×/BρA	2.5 6.5-6
F	Remainder	|	3|4←→1	7.1
U	Relations	<≤=≥>≠	3<4←→1 4<3←→0	4.8
N	Or	∨		14.2
C	And	∧	∨ 0 1 ∧ 0 1 ⍱ 0 1 ⍲ 0 1	14.2
T	Not-or	⍱	0 0 1 0 0 0 0 1 0 0 1 1	14.2
I	Not-and	⍲	1 1 1 1 0 1 1 0 0 1 1 0	14.2
O	Domino	⌹	B⌹M is soln of B=M+.×X	16.15
N				
S	Repetition	ρ	3ρ5←→5 5 5	1.7 13.3
	Catenation	,	4 2,1 3 5←→4 2 1 3 5	6.2
	Take	↑	2↑4 5 6←→4 5	10.5
	Drop	↓	2↓4 5 6←→6	10.5
	Compression	/	0 1 1 0/1 2 3 4←→2 3	7.5
M	Negation	-	-4←→¯4	8.2
O	Reciprocal	÷	÷4←→.25	8.3
N	Magnitude	|	|¯4←→4	8.4
A	Factorial	!	!4←→1×2×3×4	8.1
D	Ceiling	⌈	⌈3.4←→4	8.5
I	Floor	⌊	⌊3.4←→3	8.5
C	Complement	~	~1←→0 ~0←→1	8.6
	Matrix Inverse	⌹	M+.×⌹M is the identity	16.15
	Integers	⍳	⍳4←→1 2 3 4	1.5
	Size	ρ	ρ4 1 3 6 2←→5	8.7
	Flipping	⌽ ⊖ ⍉	Flip table about axis	4.3
O	Assignment	←	X←6	1.3
T	Indexing	X[I]	2 3 5 7[2 4]←→3 7	4.4
H		M[I;J]		
E	Function	∇Z←F X		9.1
R	Definition	∇Z←X F Y		9.2
	Parentheses			1.2
	Execution order		3×4+5-7←→3×(4+(5-7))	1.2
	Vectors		2 3 5×1 2 3←→2 6 15	1.6
	Tables, Matrices			2.1 13.3
	Reduction (Over)	f/	+/2 3 5←→10 ×/3 4←→12	1.4 4.10
	Outer Product	∘.f		2.3
	Inner Product	f.g		13.2 13.4

SUMMARY OF NOTATION

TECHNICAL REPORT INDEXING INFORMATION

1. AUTHOR(S): IVERSON, K. E.	9. INDEX TERMS FOR THE IBM SUBJECT INDEX: APL, Algebra, Education; 05 - Computer Application, 16 - Mathematics, 21 - Programming
2. TITLE: Elementary Algebra	
3. ORIGINATING DEPARTMENT: Philadelphia Scientific Center	
4. REPORT NUMBER: 320–3001	
5a. NO. OF PAGES: 330	5b. NO. OF REFERENCES: 7

6a. DATE COMPLETED	6b. DATE OF INITIAL PRINTING	6c. DATE OF LAST PRINTING
June 1971	June 1971	

7. ABSTRACT:

A text for a second year course in High School Algebra which employs a simple and precise notation (APL) suitable for direct use on a computer terminal, and emphasizes the use of arrays and explicit algorithms. The organization of topics follows a pattern suggested by considering algebra as a language; in particular, the treatment of formal identities is deferred until much later than usual.

8. DISTRIBUTION LIMITATIONS:

PUBLICATIONS

(previously appeared as Technical Reports)

(as of June 1970)

BALINSKI, M. & K. SPIELBERG, "Methods for Integer Programming: Algebraic, Combinatorial and Enumerative", *Progress in Operations Research, V. III; Relationship between Operations Research and the Computer*, (edited by J. Arnofsky), John Wiley & Sons, 195-292, 1969 [320-2938]

BARD, Y., "On a Numerical Instability of Davidon-Like Methods", *Math. of Computation*, V. 22, No. 103, 665-666, July 1968 [320-2913]

BARD, Y. & J. GREENSTADT, "A Modified Newton Method for Optimization with Equality Constraints", *OPTIMIZATION - Symp. of the Inst. of Mathematics and Its Applications, Univ. of Keele, England, 1968*, (edited by R. Fletcher & A.E.R.E. Harwell), Academic Press, 299-307, 1969 [320-2948]

BARD, Y., "Proof that $H_k \to G^{-1}$ for the E_I Correction", (an Appendix to "Variations on Variable-Metric Methods" by J. Greenstadt), *Math. of Computation*, V. 24, No. 109, 19-22, January 1970 [320-2901]

COMBA, P.G., "A Language for Three-Dimensional Geometry", *IBM Syst. J.*, V. 7, Nos. 3 & 4, 292-306, 1969 [320-2923]

COMBA, P.G., "A Procedure for Detecting Intersections of Three-Dimensional Objects", *J. of ACM*, V. 15, No. 3, 354-366, July 1968 [320-2924]

EISENPRESS, H. & J. GREENSTADT, "The Estimation of Nonlinear Econometric Systems", *Econometrica*, V. 34, No. 4, 851-861, October 1966 [320-2910]

FREUDENSTEIN, F. & L. WOO, "Kinematics of the Human Knee Joint", *Bull. of Math. Biophysics*, V. 31, No. 2, 215-232, June 1969 [320-2928]

FRIEDMAN, H.P. & J. RUBIN, "On Some Invariant Criteria for Grouping Data", *Amer. Statist. Assoc. J.*, V. 62, 1159-1178, Dec. 1967 [320-2922]

GORENSTEIN, S., "Printing Press Scheduling for Multi-Edition Periodicals", *Mgmt. Sci.*, V. 16, No. 6, B-373–B-383, February 1970 [320-2958]

GREENSTADT, J., "A Ricocheting Gradient Method for Nonlinear Optimization", *SIAM J. of Appl. Math.*, V. 14, No. 3, 429-445, May 1966 [320-2912]

GREENSTADT, J., "On the Relative Efficiencies of Gradient Methods", *Math. of Computation*, V. 21, No. 99, 360-367, July 1967 [320-2942]

GREENSTADT, J., "Variations on Variable-Metric Methods", *Math. of Computation*, V. 24, No. 109, 1-19, January 1970 [320-2901]

GUIGNARD, M., "Generalized Kuhn-Tucker Conditions for Mathematical Programming Problems in a Banach Space", *SIAM J. of Control*, V. 7, No. 2, 232-241, May 1969 [320-2920]

GUREL, O. & L. LAPIDUS, "Stability Via Liapunov's Second Method", *Ind. & Eng. Chem.*, V. 60, No. 6, 13-26, June 1968 [320-2906]

GUREL, O. & L. LAPIDUS, "The Maximum Principle and Discrete Systems", *Ind. & Eng. Chem. Fund.*, V. 7, 617-621, November 1968 [320-2919]

GUREL, O. & L. LAPIDUS, "A Guide to Methods for Generation of Liapunov Functions", *Ind. & Eng. Chem.*, V. 61, No. 3, 30-41, March 1969 [320-2937]

GUREL, O., "Marker Layout Problem Via Graph Theory", *Computing Methods in Optimization Problems-2*, (edited by L.A. Zadeh, L.W. Neustadt & A. V. Balakrishnan), Academic Press, 133-141, 1969 [320-2921]

GUREL, O., "Qualitative Study of Unstable Behavior of Cancerous Cells", *Cancer*, V. 24, No. 5, 945-947, November 1969 [320-2976]

GUREL, O. & L. LAPIDUS, "Topology of Velocity-Space for Stability Analysis", *Int'l. J. of Control*, V. 11, No. 1, 19-26, 1970 [320-2961]

HAHN, S., "On the Optimal Cutting of Defective Glass Sheets", *Operations Research*, V. 16, No. 6, 1100-1114, November-December 1968 [320-2916]

LEMKE, C. & K. SPIELBERG, "Direct Search Algorithms for Zero-One and Mixed Integer Programming", *Operations Research*, V. 15, No. 5, 892-914, October 1967 [320-2911]

LOOMIS, R. & J. LORENZO, "Experiments in Mapping with a Geo Space Plotter", *Urban & Regional Information Systems for Social Programs*, (edited by John E. Rickert), Center for Urban Regionalism, Kent State University, Ohio, 219-232, 1967 [320-2917]

LOURIE, J., "Loom Constrained Designs: An Algebraic Solution", *Proceedings of the 24th Nat'l. Conf. ACM*, ACM Publication No. P-69, 185-192, 1969 [320-2969]

LOURIE, J., "The Computation of Connected Regions in Interactive Graphics", *Proceedings of the 24th Nat'l. Conf. ACM*, ACM Publication No. P-69, 369-377, 1969 [320-2975]

RAIMOND, J., "Minimaximal Paths in Disjunctive Graphs by Direct Search", *IBM J. of Res. & Development*, V. 13, No. 4, 391-399, July 1969 [320-2930]

RUBIN, J., "Optimal Classification into Groups: An Approach to Solving the Taxonomy Problem", *J. of Theoret. Biol.*, V. 15, 103-144, 1967 [320-2915]

SAVAS, E.S., "Computers in Urban Air Pollution Control Systems", *1) Proceedings of the IBM Scientific Computing Symp. on Water and Air Resource Mgmt., Oct. 23-25, 1967, Yorktown Heights, N.Y.*, 141-173; *2) Socio-Economic Planning Science*, V. 1, 157-183, 1967; *3) Ekistics*, 45-64, 1968 [320-2908]

SPIELBERG, K., "Algorithms for the Simple Plant Location Problem with Some Side Conditions", *Operations Research*, V. 17, No. 1, 85-111, January-February 1969 [320-2900]

SPIELBERG, K., "Plant Location with Generalized Search Origin", *Mgmt. Sci.*, V. 16, No. 3, 165-178, November 1969 [320-2929]

STRAUSS, C.M. & S. POLEY, "A Three-Dimensional Piping Design Program", *IFIPS Conf. Proceedings, Edinburg, Scotland, 1968*, Information Processing 68 - North Holland Publishing Co., Amsterdam, 1431-1440, 1969 [320-2954]

WHITE, W. & A. BOMBERAULT, "A Network Algorithm for Empty Freight Car Allocation", *IBM Sys. J.*, V. 8, No. 2, 147-169, 1969 [320-2952]

WOO, L., "Type Synthesis of Plane Linkages", *J. Eng. for Ind., Trans. of ASME*, V. 89(B), 159-162, February 1967 [320-2953]

WOO, L., "An Algorithm for Straight Line Representation of Simple Planar Graphs", *J. of the Franklin Institute*, V. 287, No. 3, 197-208, March 1969 [320-2918]

WOO, L. & F. FREUDENSTEIN, "On the Curves of Synthesis in Plane, Instantaneous Kinematics", *Proceedings of the 12th Int'l. Cong. of Appl. Mechanics, Stanford University, Aug. 26-31, 1969*, (edited by M. Hetenyi & W.G. Vincenti), Springer-Verlag, 400-414, 1969 [320-2947]

TECHNICAL REPORTS

(continued from front cover)

- ■ 320-2963 *O. Gurel,* The Structure of the Genetic Code, 28 p., January 1969
- ■ 320-2964 *S. Gorenstein,* Generating Random Normal Numbers, 19 p., February 1969
- ■ 320-2965 *O. Gurel,* Circular Graph of Marker Layout, 63 p., February 1969
- 320-2966 *W. White,* Dynamic Transshipment Networks: An Algorithm and Its Application to the Distribution of Empty Containers, 40 p., February 1969
- 320-2967 *F. Freudenstein, V. Vitagliano, L. Woo & C. Hao,* Dynamic Response of Mechanical Systems, 29+p., March 1969
- 320-2968 *Proceedings of the Third Annual Technical Management Science/Operations Research Conference,* 206+p., April 1969
- † 320-2969 *J. Lourie,* Loom-Constrained Design: An Algebraic Solution, 32 p., April 1969
- 320-2970 *L. Bodin,* The Catalogue Ordering Problem-2, 33 p., April 1969
- ■ 320-2971 *E. Balas,* Machine Sequencing: Disjunctive Graphs and Degree-Constrained Subgraphs, 31 p., April 1969
- ■ 320-2972 *J. Colmin & K. Spielberg,* Branch and Bound Schemes for Mixed Integer Programming, 27 p., May 1969
- 320-2973 *M. Grigoriadis,* A Dual Generalized Upper Bounding Technique, 38 p., June 1969
- ■ 320-2974 *L. Bodin & T. Roefs,* Determination of the Operating Policy of a General Reservoir System, 24 p., June 1969
- † 320-2975 *J. Lourie,* The Computation of Connected Regions in Interactive Graphics, 26 p., June 1969
- † 320-2976 *O. Gurel,* Qualitative Study of Unstable Behavior of Cancerous Cells, 9 p., July 1969
- 320-2977 *S. Gorenstein,* An Algorithm for Project (Job) Sequencing with Resource Constraints, 60 p., July 1969
- 320-2978 *G. Moreau & P. Tarbe de Saint Hardouin,* Marker Layout Problem: An Experimental Attempt, 67 p., September 1969
- 320-2979 *C. Lemke, H. Salkin & K. Spielberg,* Set Covering by Single Branch Enumeration with Linear Programming Subproblems, 51 p., October 1969
- 320-2980 *Y. Bard & L. Lapidus,* On Nonlinear System Identification, 21 p., October 1969
- 320-2981 *O. Gurel,* Algebraic Theory of Scheduling, 32 p., November 1969
- 320-2982 *L. Woo & F. Freudenstein,* Application of Line Geometry to Theoretical Kinematics and the Kinematic Analysis of Mechanical Systems, 103 p., November 1969
- 320-2983 *O. Gurel,* Fundamental Weak Topologies in Living Systems, 18 p., March 1970
- 320-2984 *H. Hellerman & Y. Ron,* A Time Sharing System Simulation and Its Validation, 36 p., April 1970
- 320-2985 *O. Gurel,* Functional Groups of Amino Acids and Structural Groups in the Genetic Code, 14 p., April 1970
- 320-2986 *A. Falkoff & K. Iverson,* The Use of Computers in Teaching Mathematics, 10 p., April 1970
- 320-2987 *O. Gurel,* Cancer: An Unstable Biodynamic Field, 22 p., April 1970
- 320-2988 *P. Berry, A. Falkoff & K. Iverson,* Using the Computer to Compute: A Direct But Neglected Approach to Teaching Mathematics, 20 p., May 1970
- 320-2989 *M. Jenkins,* The Solution of Linear Systems of Equations and Linear Least Squares Problems in APL, 14 p., June 1970
- 320-2990 *W. W. White,* On the Computational Status of Mathematical Programming, 61 p., June 1970
- 320-2991 *J. Greenstadt,* A Variable-Metric Method Using No Derivatives, 24 p., June 1970
- 320-2992 *M. Grigoriadis,* A Projective Method for a Class of Structured Nonlinear Programming Problems, 142 p., June 1970
- 320-2993 *W. W. White & E. Wrathall,* A System for Railroad Traffic Scheduling, 101 p., August 1970
- 320-2994 *K. M. Brown & J. E. Dennis, Jr.,* Derivative Free Analogues of the Levenberg-Marquart and Gauss Algorithms for Nonlinear Least Squares Approximation, 21 p., August 1970
- 320-2995 *S. Gorenstein,* Programming for Economic Lot Sizes with Precedences between Items - An Assembly Model, 76 p., October 1970
- 320-2996 *M. Grigoriadis & W. W. White,* A Partitioning Algorithm for the Multicommodity Network Flow Problem, 25 p., October 1970
- 320-2997 *L. H. Scott, E. Wrathall, & S. Poley,* A Railroad Freight Operations Control System, 22 p., December 1970
- 320-2998 *D. Goldfarb,* Modification Methods for Inverting Matrices and Solving Systems of Linear Algebraic Equations, 58 p., January 1971
- 320-2999 *S. Gorenstein, S. Poley, & W. W. White,* On the Scheduling of Railroad Freight Operations, 21 p., January 1971
- 320-3000 *M. Guignard & K. Spielberg,* The State Enumeration Method for Mixed Zero-One Programming, 21 p., February 1971

† Appeared in a professional publication, see separate list. These are no longer available in technical report form, but reprints are available upon request.

†■ No copies of either the report or reprint are available from the PSC.

■ Report is out of print. No copies are available from the PSC .

* Reprinting of reports originally issued with different numbers.

§ In September, 1970, the New York Scientific Center was transferred to Philadelphia and is now the Philadelphia Scientific Center. Reports formerly distributed by the NYSC, and those now published by the PSC may now be obtained by writing to: IBM Corporation, Philadelphia Scientific Center, 3401 Market Street, Philadelphia, Pa. 19104

IBM