Revista Mexicana de Astronomía y Astrofísica, Volumen 1, Abril 1974

APL IN ASTRONOMY

EUGENIO E. MENDOZA V.

Instituto de Astronomía
Universidad Nacional Autónoma de México
Received 1974 February 23

RESUMEN

En este trabajo se describe APL, un lenguaje algorítmico más extenso, preciso y consistente en su sintaxis que el lenguaje tradicional de las matemáticas elementales, con el objetivo principal de mostrar su aplicación en Astronomía. Para este fin se eligieron temas de mínimos cuadrados, integración numérica con abscisas de separación variable o constante y solución de sistemas de ecuaciones diferenciales ordinarias del primer orden. Cada tópico se ilustró con un ejemplo astronómico.

ABSTRACT

This paper describes APL, an algorithmic language more accurate, rich and consistent than that of classical elementary mathematics. Examples are given to demonstrate its versatility in the solving of astronomical problems. We select, for this purpose, topics in linear least squares, spline numerical quadrature, and solution of systems of ordinary differential equations of the first order.

Key words: APL — MATHEMATICAL LANGUAGE — ASTRONOMY.

I. INTRODUCTION

APL is an elegantly simple general purpose language, more accurate and consistent than that of classical elementary mathematics (for instance, it eliminates ambiguities, conflicts and anomalies that exist in arithmetic and elementary algebra —see Table 1). It represents a synthesis of mathematics from a variety of disciplines with a unified notation to describe many different processes precisely and concisely. It is, also, very rich in primitive functions (see appendices).

APL derives its name from the book of its originator K. E. Iverson: **A P**rogramming **L**anguage (Iverson 1962, 1971; Falkoff and Iverson 1968). It was initially designed for human communication, not for machines; however, now it can also be used with computers.

The main purpose of this paper is to introduce APL to astronomers, specially to those without programming experience, since its applications to machine use require virtually no knowledge of the internal functioning of the computer or programming experience.

Table 1 illustrates, very briefly, some of the above statements. The columns of this table contain, first, the name of familiar mathematical functions; second and third, the APL and the conventional notation for these functions, respectively; last, a remark on the traditional language (see also appendices).

In section II we describe the APL concepts that lead to linear least squares problems. We also give in this section a simple example taken from astronomy. In section III, we define a function to integrate numerically, specially when the data given points are not equally spaced. This technique is also illustrated with another example taken from astronomy.

In section IV, a seventh-order Runge-Kutta function is given to show an application in astronomy. The conclusions are presented in section V. The functions and operators of the language are summarized in the appendices.

TABLE 1

APL AND CONVENTIONAL NOTATION

Name of function	APL	Traditional	Remarks on traditional
Natural logarithm	⍟X	$\log_e X$	argument *right* to function
Factorial	!N	$N!$	argument *left* to function
Magnitude	∣X	$\lvert X \rvert$	argument between *two marks*
Power	X ⋆ 3	X^3	*no symbol* for the function
Times	X × Y	XY	*symbol* of the function *omitted*
Assign variable value	A ← 5	$A = 5$	same '=' symbol used for both,
Relationship statement	2 = 5 − 3	$2 = 5 - 3$	but *different* purposes.
Summation (sum over)	+/V	ΣV	*no indication* of function to
Times over	×/V	ΠV	be performed and *no extensions*
Maximum	⌈/V	$\max \{V\}$	to other dyadic functions, like
Minus over	−/V	$V_1 - V_2 + V_3 - \ldots$	alternating sum.

II. LINEAR LEAST SQUARES PROBLEMS

a) *Size, Take and Drop functions*

The number of elements in a vector V is called the *size* of the vector. It is denoted by ρ.

The monadic function ρ applied to a matrix yields a two element vector giving the number of rows in the matrix followed by the number of columns. In general, when applied to an array A, the function ρ yields a vector whose components are the dimensions of A.

The dyadic functions, *take* and *drop,* are denoted by ↑ and ↓, respectively. The take 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 (see Appendix E).

b) *Inner product*

The familiar matrix product of the matrices M and N is denoted in APL by M + . × N and is called the "plus times inner" product. In APL, also, the inner product is extended to all f and to all g primitive scalar dyadic functions (see Appendix C).

If X and Y are vectors of the same dimension then the expression Xf . gY is defined as equivalent to the expression f/XgY (see Table 1 and Appendix F).

The inner product is, also, defined on arrays A and B whose sizes satisfy conditions on *conformability.* A and B are conformable with the inner product Af.gB if at least one of the follwing conditions is satisfied:

1) A is a scalar
2) B is a scalar
3) ¯1 ↑ ρA is equal to 1 ↑ ρ B
4) ¯1 ↑ ρA is equal to one
5) 1 ↑ ρB is equal to one

The above definition of the inner product of two vectors can be extended to matrices and arrays. If M and N are two conformable matrices, then their inner product R ← Mf . gN is such that the element R [I;J] is given by the inner product of the vectors M [I;] and N [;J]

R[I;J] ← M [I;] f.gN [;J] (see Appendix G).

Thus, the inner product of two arrays A and B of equal rank, whose sizes are vectors of N-elements, can be defined in terms of arrays whose sizes are vectors of (N-1)-elements.

If X is a vector and M is a matrix, then the inner products, Mf . gX and Xf . gM, are defined by simply treating X much like a one-column and a one-row matrix, respectively.

A single element of an array A can be selected by specifying its *indices;* the number of indices required is called the *"rank"* or *"dimensionality"* of the array. Indexing is denoted by brackets. For example, if M is the matrix

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

then M [2;3] is 7, the element of the second row and the third column (the row index appears first).

M [2;] is 5 6 7 8 (a row index alone selects the entire vector in that row).

M [;3] is 3 7 11 (a column index alone selects the entire column).

c) *The Domino function*

APL includes a *matrix division* primitive function which is denoted by ⌹. The dyadic form of ⌹ (domino) is used for solving systems of linear equations.

The expression Y ← B ⌹ A produces a Y such that + / (, B − A + × Y) ★ 2 is minimized, where ρA is the vector with elements M and N (M ≥ N), 1 ↑ ρ B is M, and
ρ Y is N , 1 ↓ ρ B.

TABLE 2

HYADES PHOTOMETRY

VB	PD	m	V	B − V	O − C	VB	PD	m	V	B − V	O − C
1	1942	7.80	7.40	0.57	0.144	77	2728	7.16	7.03	0.50	−0.133
6	2338	6.34	5.97	0.34	0.095	78	2730	7.18	6.91	0.45	0.003
8	2391	6.78	6.37	0.42	0.139	80	2741	5.93	5.58	0.32	0.075
13	2511	6.98	6.62	0.42	0.090	82	2745	5.02	4.78	0.17	−0.035
14	2520	6.10	5.73	0.36	0.096	83	2746	5.66	5.48	0.26	−0.097
16	2550	7.26	7.05	0.42	−0.059	84	2747	5.66	5.41	0.26	−0.026
20	2570	6.61	6.32	0.40	0.018	85	2748	6.86	6.51	0.43	0.080
24	2592	5.82	5.65	0.28	−0.106	89	2761	6.32	6.02	0.34	0.025
28	2608	3.98	3.65	0.99	0.037	90	2767	6.66	6.40	0.41	−0.011
29	2610	7.17	6.89	0.56	0.020	94	2782	6.88	6.62	0.43	−0.009
30	2614	5.72	5.59	0.28	−0.146	95	2783	4.89	4.65	0.25	−0.034
32	2619	6.43	6.11	0.37	0.047	100	2809	6.23	6.02	0.38	−0.063
33	2621	5.44	5.26	0.22	−0.097	101	2813	6.90	6.65	0.44	−0.019
34	2625	6.47	6.17	0.46	0.031	103	2829	6.13	5.79	0.31	0.064
35	2630	7.06	6.80	0.44	−0.008	104	2831	4.66	4.27	0.12	0.118
36	2632	7.19	6.81	0.44	0.112	107	2841	5.64	5.39	0.25	−0.027
37	2635	6.94	6.61	0.41	0.059	108	2840	4.94	4.70	0.14	−0.035
38	2639	5.90	5.72	0.32	−0.095	111	2879	5.60	5.40	0.25	−0.077
41	2648	4.16	3.76	0.99	0.109	112	2895	5.64	5.37	0.19	−0.079
44	2649	7.39	7.18	0.45	−0.056	113	2900	7.38	7.26	0.56	−0.138
45	2653	5.93	5.64	0.30	0.014	119	2930	7.38	7.11	0.56	0.012
47	2662	5.12	4.80	0.15	0.045	121	2934	7.66	7.29	0.50	0.108
53	2670	6.38	5.97	0.37	0.137	122	2943	6.86	6.77	0.55	−0.171
54	2675	4.58	4.22	0.13	0.088	123	2940	5.32	5.11	0.21	−0.066
55	2676	5.53	5.28	0.25	−0.026	124	2950	6.52	6.29	0.50	−0.037
56	2678	4.54	4.28	0.40	−0.011	126	2983	6.63	6.37	0.29	−0.017
57	2681	6.74	6.46	0.49	0.014	128	3035	6.98	6.76	0.45	−0.048
58	2683	7.85	7.53	0.68	0.075	129	3060	4.90	4.64	0.16	−0.014
60	2686	4.52	4.28	0.26	−0.033	130	3126	5.69	5.43	0.24	−0.017
62	2687	7.62	7.38	0.54	−0.018	137	2403	6.20	5.89	0.32	0.035
66	2711	7.73	7.51	0.55	−0.037	141	2688	4.80	4.50	0.25	0.026
67	2708	5.98	5.72	0.27	−0.017	146	2974	7.48	7.24	0.53	−0.020
68	2716	6.19	5.90	0.32	0.015	154	1756	6.06	5.80	0.41	−0.012
70	2718	3.88	3.54	1.01	0.044	157	1319	5.96	5.79	0.44	−0.101
71	2720	3.93	3.83	0.95	−0.188	160	2438	5.80	5.46	0.36	0.066
72	2721	3.68	3.39	0.18	0.022	164	2894	6.22	6.01	1.21	−0.028
74	2724	5.41	5.03	0.23	0.104	168	3414	5.84	5.54	0.22	0.022
75	2726	6.85	6.59	0.53	−0.004	169	3695	4.36	4.13	0.16	−0.042

Thus, Y is the least squares solution of the system or systems of simultaneous linear equations B = A + . × Y. If A is a non-singular square matrix, then Y is the solution of a well determined system of linear equations.

We see from the above that the domino function is very valuable in astronomy. A common astronomical problem is to transfer one photometric system into another. For example, let us transform the Potsdam visual photometric system into the standard B, V photoelectric system, taking those stars, cluster members of the Hyades group (Mendoza 1967), that have been observed in both photometric systems. Since they are located in a rather small area of the sky, the transfer equation becomes (Mendoza and Gómez 1969):

$$m - V = a + b\,(B - V) + s\,(V - \bar{V}) + p\,(V - \bar{V})\,(B - V),$$

where

m is Potsdam magnitude
V is Johnson magnitude
B − V is Johnson color
$\bar{V}$ is + /V ÷ ρ V (V − mean magnitude)

a is the zero point difference between both systems,
b is the spectral range difference between the eye and the RCA 1P21 photomultiplier plus a yellow filter,
s is the Pogson scale deviation, and
p is the Purkinje effect contribution (cf. Mendoza and Gomez 1969).

To find a, b, s and p, it is only necessary to apply the above domino function to solve the equation

X ⌹ M,

where X is the vector of all the m − V and M is the matrix of coefficients a, b, s and p. Thus X ⌹ M yields

for a	0.287
for b	−0.036
for s	0.010
for p	−0.030

These results indicate that Potsdam photometry is valuable. It is interesting to mention that this computation took 0.12 seconds with an IBM 370/155. The relevant data are listed in Table 2.

The columns of Table 2 contain: first, the VB number (Bueren 1952); second, the PD number (Müller and Kempf 1907); third, mean PD magnitude (Müller and Kempf 1907, column 9 of Table 1); fourth, V magnitude (Mendoza 1967); fifth, B − V color (Mendoza 1967); and last, the observed minus the computed m − V (O − C).

A more useful application would be in photoelectric reductions.

III. A SPLINE QUADRATURE FORMULA

a) *Function definition*

It would be impracticable and confusing to attempt to include as primitives in a language all of the functions which might prove useful in diverse areas of application. Instead, there should be the possibility of defining and naming functions.

In APL a function definition begins and ends with the simbol ∇ (del). Its name must begin with a letter but may include both letters and digits. It may have one argument (monadic), two arguments (dyadic), or zero arguments (niladic).

A defined function may contain both *local* and *global* variables. A variable is, normally, global in the sense that its name has the same significance, no matter what function or functions it may be used in. A variable is local when it has meaning only during the execution of the function and bears no relation to any object referred to by the same name at other times. Any number of variables can be made local to a function by appending each (preceded by a semicolon) to the function header.

b) *A Spline Function*

The name "spline function" comes from the fact that a third degree spline function behaves similarly to a mechanical spline (a device used by draughtsmen to draw a smooth curve) which consists of a flexible steel strip to which weights are attached at certain points, in order to force a fit to the given data points.

Following Greville (1967) we have derived a mechanical quadrature formula that uses a third degree spline function. This is given in Table 3a in terms of a monadic defined function named SPLINE. Its argument, X, is the matrix of the n given data points, n-rows and 2-columns (abscissas, first column; ordinates, second column). In addition to the primitive functions, SPLINE also uses three defined functions named DIF, S and SUM, respectively. They are listed in Table 3b in terms of primitive functions.

The ninth statement of SPLINE is prefaced by "ONE:" this name, at the beginning of the execution, is equal to 9 (the statement number). A variable specified in this way is called a *label.* Another label of SPLINE appears on line 12.

The right-pointing arrows on lines 13, 16, 18, 19, and 20 of SPLINE are called *branches.* The

TABLE 3a

A SPLINE QUADRATURE FORMULA

```
    ∇ SPLINE X; B; DY; D2Y; ETA; G; H; HH; J; S2X; W; WW
[1]   H ← DIF X[;1]
[2]   DY ← (DIF X[;2]) ÷ H
[3]   HH ← SUM H
[4]   B ← 0.5×(¯1 ↓ H)÷HH
[5]   D2Y ← (DIF DY)÷HH
[6]   S2X ← 2×D2Y
[7]   G ← 3×D2Y
[8]   S2X ← 0, S2X, 0
[9]   ONE : ETA ← 0
[10]  J ← 2
[11]  W ← 4×2−3★÷2
[12]  TWO : WW ← W × G[J−1]− +/S2X[J], (B[J−1]×S2X[J−1]), (0.5−B[J−1])×S2X[J+1]
[13]  → 14+(|WW)≤ETA
[14]  ETA ← |WW
[15]  S2X[J] ← S2X[J]+WW
[16]  → 19−2×J≠¯1+(ρX)[1]
[17]  J ← J+1
[18]  → TWO
[19]  → 20+ETA<EPSILON
[20]  → ONE
[21]  '  ABSCISSAS    ORDINATES    S''''(X)'
[22]  S 1
[23]  ⍉ (3, ((ρX)[1]))ρ X[;1], X[;2], S2X
[24]  S 1
[25]  'SPLINE INTEGRAL = ';+/(0.5×H×(SUM X[;2]))−(H★3)×(SUM S2X)÷24
    ∇
```

TABLE 3b

AUXILIARY FUNCTIONS OF SPLINE

```
    ∇ Z ← DIF Y
[1]   Z ← (1 ↓ Y) − ¯1 ↓ Y
    ∇

    ∇ S N
[1]   (N−1)ρ '
    ∇

    ∇ Z ← SUM Y
[1]   Z ← (1 ↓ Y)+¯1 ↓ Y
    ∇
```

only effect of the expression "→TWO" (line 18) is to cause statement 12 (with the label TWO) to be executed next, i.e., the normal order can be modified by branches.

Labels are used to advantage in branches when it is expected that a function definition may be changed for one reason or another, since a label automatically assumes the new value of the statement number of its associated statement as statements are inserted or deleted.

The SPLINE function listed in Table 3a may be improved. The way it is presented shows a variety of uses of APL which may be of interest to the reader. A great advantage of SPLINE is that the given data points, X, are *not necessarily equally spaced.*

The photometric luminosity of a star derived from the fluxes measured over a range of wavelengths is given by

$$4\,\pi\,r^2 \int_0^\infty F\,(\lambda)\; d\,\lambda,$$

where r is the distance of the star and F (λ) is the flux at wavelength λ. In wide-band photometry the λ's, the effective wavelengths of the filters, are not equally spaced. Photometric luminosities may be easily calculated and duplicated by means of the SPLINE function. For instance, the observed

$$\int_{.36}^{5} F\,(\lambda)\; d\,\lambda$$

for T Tauri (Mendoza 1968), with the aid of SPLINE is:

SPLINE X

ABSCISSAS	*ORDINATES*	S″(X)
3.60000*E*¯1	5.62975*E*¯17	0.00000*E*0
4.40000*E*¯1	1.85923*E*¯16	¯6.20774*E*¯15
5.50000*E*¯1	3.11377*E*¯16	¯5.56997*E*¯15
7.00000*E*¯1	3.85046*E*¯16	¯1.97146*E*¯15
9.00000*E*¯1	4.04649*E*¯16	¯4.54455*E*¯16
1.25000*E*0	3.76252*E*¯16	¯6.46857*E*¯16
1.60000*E*0	2.89697*E*¯16	3.01159*E*¯16
2.20000*E*0	1.83259*E*¯16	1.60565*E*¯16
3.40000*E*0	1.38193*E*¯16	6.75254*E*¯17
5.00000*E*0	2.30368*E*¯16	0.00000*E*0

SPLINE INTEGRAL = 1.02418*E*¯15

The above result contains the original X and the second derivatives of the third degree spline function. This computation took 0.37 sec (see above).

IV. A RUNGE-KUTTA FUNCTION

Mendoza and Hacyan (1974) have derived classical fifth−, sixth−, seventh−, and eighth-order Runge-Kutta functions with step size control which hold for systems of n-differential equations. They have shown that the seventh-order function is the most suitable for use in APL because it is faster and more accurate than the other functions.

Below we present Mendoza and Hacyan's seventh-order function with minor changes to illustrate further APL. We, also, present the following astronomical problem:

Polytropes are very useful in the demonstration of some of the general concepts of stellar structure. The so called "Lane-Emden equation" is the basic equation in the study of polytropes. It can be written as two differential equations:

$$\left.\begin{aligned} \frac{dY}{dX} &= Z \text{ and} \\ \frac{dZ}{dX} &= -\frac{2Z}{X} - Y^n, \end{aligned}\right\} \qquad (1)$$

where n is the polytropic index.

The initial conditions are such that X = 0, Z = 0, and Y = 1. For n = 0, 1 and 5 the system (1) has an analytical solution. For other values of n we obtain a start from

$$Y_n = 1 - \frac{1}{6}X^2 + \frac{n}{120}X^4 - \cdots .$$

This series is valid only for small X. With values of Y and Z, at a point conveniently reached by the last equation, we can carry the solution with the seventh-order Runge-Kutta function (RK78P).

Table 4a contains the dyadic function RK78P. Its left argument indicates the number of differential equations. The right argument is a vector of size N + 5. The components of this vector give:

TABLE 4a

A SEVENTH-ORDER RUNGE-KUTTA FUNCTION

```
∇ N RK78P Y;AC;CRP;EFE;ERR;H;HH;I;J;P;XX;YRES
[1]  J ← 1
[2]  Z ← ((1+(Y[N+2]−Y[1])÷Y[N+3]),N+1)ρ 0
[3]  AC ← 0
[4]  EFE ← (13,N) ρ 0
[5]  XX ← Y[1]
[6]  H ← Y[N + 3] × 0.1
[7]  FIVE: CRP ← 0
[8]  → ONE IF Y[1] ≠ XX
[9]  Z[J;] ← Y[ιN+1]
[10] J ← J+1
[11] XX ← XX+Y[N+3]
[12] → TWO IF Y[1]≥Y[N+2]
[13] ONE : P ← | P+Y[N+3]×0=P ← XX−Y[1]+H
[14] SIX : EFE[1;] ← Y[N+5] DER(N+1)ρY
[15] I ← 2
[16] THREE : EFE[I;] ← Y[N+5] DER((Nρ 1 ↓ Y)+(((I−1)ρBETA[(ιI−1)+
     0.5×(I−2)×I−1])+.×EFE[ιI−1;]×H),Y[1]+ALPHA[I−1]×H
[17] → THREE IF 13≥I←I+1
[18] YRES ← (Nρ1 ↓ Y)+H×GAMMA+.×EFE[1,5+ι6;]
[19] ERR ← (41÷840)× ⌈/|EFE[1;]+EFE[11;]−EFE[12;]+EFE[13;]
[20] HH ← H
[21] H ← ⌊/P,HH×(Y[N+4]÷(ERR×10)+1E¯7×Y[N+4])★÷7
[22] → FOUR IF ERR≥Y[N+4]
[23] Y[ιN+1] ← Y[1], YRES+HH
[24] AC ← AC+ERR×HH
[25] → FIVE
[26] FOUR : → SIX IF 2≥CRP ← CRP+1
[27] TWO : S 1
[28] Z[;3] ← −Z[;3]
[29] Z
   ∇
```

TABLE 4b

AUXILIARY FUNCTIONS OF RK78P

```
∇ COEFF
[1] ALPHA ← (2÷27),(÷9),(÷6),(5÷12),0.5,(5÷6),(÷6),(2÷3),(÷3), 1 0 1
[2] BETA ← (2÷27),(÷36),(÷12),(÷24),0,(÷8),
    (5÷12),0,(¯25÷16),(25÷16),(÷20), 0 0, 0.25 0.2,(¯25÷108), 0 0
[3] BETA ← BETA,(125÷108),(¯65÷27),(125÷54),(31÷300), 0 0 0,(61÷225),
    (¯2÷9),(13÷900), 2 0 0,(¯53÷6),(704÷45)
[4] BETA ← BETA,(¯107÷9),(67÷90),3,(¯91÷108), 0 0,(23÷108),(¯976÷135),
    (311÷54),(¯19÷60),(17÷6),(−÷12)
[5] BETA ← BETA,(2383÷4100), 0 0,(¯341÷164),(4496÷1025),(¯301÷82),
    (2133÷4100),(45÷82),(45÷164),(18÷41),(3÷205)
[6] BETA ← BETA, 0 0 0 0,(¯6÷41),(¯3÷205),(¯3÷41),(3÷41),(6÷41), 0,
    (¯1777÷4100), 0 0,(¯341÷164),(4496÷1025)
[7] BETA ← BETA,(¯289÷82),(2193÷4100),(51÷82),(33÷164),(12÷41), 0 1
[8] GAMMA ← (41÷840),(34÷105),(9÷35),(9÷35),(9÷280),(9÷280),(41÷840)
  ∇
```

```
∇ Z ← N DER Y
[1] Z ← ((¯2×Y[1]÷Y[3])−Y[2]★N),Y[1]
  ∇
```

```
∇ Z ← A IF B
[1] Z ← B/A
  ∇
```

1,···,N	The initial values of the dependent variables.
N + 1	The initial value of the independent variable.
N + 2	The last value of the independent variable.
N + 3	Step size for printing the new N + 1 elements of Y.
N + 4	Tolerance
N + 5	Polytropic index

Table 4b shows the defined functions used by RK78P, DER and IF. In addition, Table 4b lists the coefficients of the seventh-order Runge-Kutta function as a niladic function (COEFF). Again, the listed functions may be improved. However, they are given primarily to show a variety of uses of APL.

Table 4c shows the results from 0 to 3.6 of the Lane-Emden function for the case n = 1.5, which are accurate at least to the seventh place (Wrubel 1958). This computation took 27 seconds (see above).

V. CONCLUSION

We have presented a partial description of APL to show two main characteristics, namely, its virtues as a mathematical language and its versatility in the solving of astronomical problems.

For the sake of simplicity and briefness, we did not give more complicated examples. Needless to say, APL will handle them with no problem.

It is interesting to point out that programming time and execution time are as a rule shorter, in APL than other major computer languages, see for instance, Kolsky (1969).

We are grateful to the IBM Latin American Scientific Center for providing computing facilities.

TABLE 4c

LANE-EMDEN FUNCTION

(Polytrope n = 1.5)

X	Y	Y′
0	1	0
0.1	0.99833458	0.033283375
0.2	0.99335329	0.066267997
0.3	0.98510075	0.098660069
0.4	0.97365051	0.13017558
0.5	0.95910386	0.16054489
0.6	0.94158813	0.18951693
0.7	0.9212547	0.21686297
0.8	0.89827654	0.2423798
0.9	0.87284558	0.26589233
1	0.84516976	0.28725554
1.1	0.81546995	0.30635568
1.2	0.78397682	0.32311089
1.3	0.75092764	0.33747108
1.4	0.7165631	0.34941725
1.5	0.68112433	0.35896018
1.6	0.64484991	0.36613866
1.7	0.60797328	0.37101729
1.8	0.57072021	0.37368393
1.9	0.53330663	0.37424694
2	0.49593676	0.37283214
2.1	0.45880147	0.36957988
2.2	0.42207699	0.36464189
2.3	0.38592395	0.35817841
2.4	0.35048663	0.35035528
2.5	0.31589258	0.34134136
2.6	0.2822524	0.33130609
2.7	0.24965981	0.32041742
2.8	0.21819187	0.30884
2.9	0.18790943	0.29673374
3	0.15885761	0.28425273
3.1	0.13106644	0.27154467
3.2	0.10455153	0.25875075
3.3	0.079314641	0.24600631
3.4	0.055344243	0.23344261
3.5	0.032615729	0.22119086
3.6	0.011090995	0.20939266

APPENDIX A

PUNCTUATION MARKS

Symbol	Description	Example
.	The same as in arithmetic.	2.71828
.	Catenates some functions to define new operators.	inner product
¯	Used only as a part of a constant to represent negative numbers, immediately preceding the number.	¯2
E	An integer, following the "E" specifies the power of ten by which the part preceding the E is to be multiplied.	2.718E2 is 271.8
'	An enclosed expression in "quotes" defines a character expression. The quotes do not form part of the expression.	'APL\360' is APL\360
( )	An enclosed expression in "parentheses" must be completely evaluated before its results can be used.	(2 + 3) × 4 is 20 (2 + 3× 4) is 14
[ ]	To the right of a variable, for indexing.	M[2;] is 8 6 4 2
[ ]	To the left of a variable, for indicating under which dimension is the execution (see also Appendix E).	+/[1] M is 9 9 9 9
[ ]	In defined functions "brackets" are statement numbers.	See text
;	In brackets for separating indices.	M [2;3] is 4
;	To define local variables in defined functions.	See text
;	Catenates a variable with an expression.	'SUM IS ' ; + / ⍳5 gives SUM IS 15
:	For labeling statements in defined functions.	See text
_	Letters "underlined" are composite letters.	A̲ P̲ L̲
⍝	The "lamp" symbol signifies what follows it is a comment, for illuminatior only and not to be executed.	See Falkoff and Iverson (1968)

APPENDIX B

PRIMITIVE SCALAR MONADIC FUNCTIONS

Function	Name	Definition	Example	Result
+	Plus	+ X ← → 0 + X	+ 4	4
−	Negation	− X ← → 0 − X	− 4	¯4
×	Signum	1 for × > 0	× 4	1
		0 for × = 0	× 0	0
		¯1 for × < 0	× ¯4	¯1
÷	Reciprocal	÷ X ← → 1 ÷ X	÷ 4	0.25
⌈	Ceiling	Smallest integer not exceeded by X	⌈ 4.13	5
⌊	Floor	Largest integer not exceeding X	⌊ 4.13	4
⋆	Exponential	⋆ X ← → (2.718281828. . .) ⋆ X	⋆ 4	54.59815. . .
⍟	Natural Logarithm	⍟ × ← → (2.718281828. . .) ⍟X	⍟4	1.38629. . .
|	Magnitude	|X ← → X × × X	| 4	4
!	Factorial	Gamma and factorial functions of the arithmetic	! 4	24
?	Roll	? N ← → random selection among ⍳N	? 4	2
∼	Complement	1 for X = 0	∼ 0	1
		0 for X = 1	∼ 1	0
○	Pi times	○ X ← → (3.14159. . .) × X	○ 4	12.56637. . .

APPENDIX C

PRIMITIVE SCALAR DYADIC FUNCTIONS

Function	Name	Defintion	Example	Result
+	Addition	Same as in arithmetic	2 + 4	6
−	Substraction	Same as in arithmetic	2 − 4	¯2
×	Multiplication	Same as in arithmetic	2 × 4	8
÷	Division	Same as in arithmetic	2 ÷ 4	0.5
⌈	Maximum	X ⌈ Y ←→ largest between X and Y	2 ⌈ 4	4
⌊	Minimum	X ⌊ Y ←→ smallest between X and Y	2 ⌊ 4	2
★	Power	Same as in arithmetic	2 ★ 4	16
⊛	Logarithm	X ⊛ Y ←→ log Y base X	2 ⊛ 4	2
|	Residue	X | Y ←→ Y − (|X) × ⌊ Y ÷ | X for X ≠ 0	2 | 4	0
		X | Y ←→ Y for X = 0, Y ⩾ 0	0 | 4	4
		X | Y ←→ not defined for X = 0, Y < 0	0 | ¯4	Domain error
!	Binomial coefficient	X ! Y ←→ (! Y) ÷ (! X) × ! Y − X	2 ! 4	6
=	Equal	Same as in arithmetic	2 = 4	0
≠	Not equal	Same as in arithmetic	2 ≠ 4	1
>	Greater	Same as in arithmetic	2 > 4	0
⩾	Not less	Same as in arithmetic	2 ⩾ 4	0
<	Less	Same as in arithmetic	2 < 4	1
⩽	Not greater	Same as in arithmetic (see Table 1).	2 ⩽ 4	1

Logical functions Table

Function	Name	X	Y	X ∧ Y	X ∨ Y	X ⍲ Y	X ⍱ Y
∧	And	0	0	0	0	1	1
∨	Or	0	1	0	1	1	0
⍲	Nand	1	0	0	1	1	0
⍱	Nor	1	1	1	1	0	0

Circular functions Table

(− X) ○ Y	X	X ○ Y
(1 − Y ★ 2) ★ 0.5	0	(1 − Y ★ 2) ★ 0.5
Arcsin Y	1	Sine Y
Arccos Y	2	Cosine Y
Arctan Y	3	Tangent Y
(¯1 + Y ★ 2) ★ 0.5	4	(1 + Y ★ 2) ★ 0.5
Arcsinh Y	5	Sinh Y
Arccosh Y	6	Cosh Y
Arctanh Y	7	Tanh Y

APPENDIX D

PRIMITIVE MIXED MONADIC FUNCTIONS

Function	Name	Definition	Example	Result
⍴	Size	⍴ X ← → dimension of X	⍴ M	2 4
,	Ravel	, X ← → (× / ⍴ X) ⍴ X	, M	1 3 5 7 8 6 4 2
⍳	Index generator	⍳ N ← → first N integers	⍳ 4	1 2 3 4
⍋	Grade up	The permutation which orders X ascendingly	⍋ 3 5 2 4	3 1 4 2
⍒	Grade down	The permutations which order X descendingly	⍒ 3 5 2 4	2 4 1 3
⌽	Row reversal	⌽ X ← → X is reflected on last coordinate	⌽ M	7 5 3 1 2 4 6 8
⊖	Column reversal	⊖ X ← → X is reflected on first coordinate	⊖ M	8 6 4 2 1 3 5 7
⍉	Transposition	⍉ X ← → is transposed on last two coordinates	⍉ M	1 8 3 6 5 4 7 2
⌹	Matrix inverse	⌹ X ← → It is such that X + . × ⌹ X is the identity	⌹ 2 2 ⍴ 1 3 5 7	0.875 0.375 0.625 ¯0.125

APPENDIX E

PRIMITIVE MIXED DYADIC FUNCTIONS

Function	Name	Definition	Example	Result
ρ	Reshape	X ρ Y ←→ "reshapes" Y to dimension X	2 4 ρ ι 8	1 2 3 4 5 6 7 8
,	Row catenation	Joins two variables along the last coordinate	P , Q	10 8 6 1 3 5 4 2 0 7 11 13
, [1]	Column catenation	Joins two variables along the first coordinate	P , [1] Q	10 8 6 4 2 0 1 3 5 7 11 13
, [R]	Lamination	Joins two variables along a new coordinate	P , [1.5] Q	10 8 6 1 3 5 4 2 0 7 11 13
ι	Index of	X ι Y ←→ Least "index of" X in Y, or 1 ↑ ρ Y	2 3 5 7 ι 5 2	3 1
↑	Take	See text	2 ↑ 7 3 0	7 3
↓	Drop	See text	2 ↓ 7 3 0	0
/	Row compression	X/Y ←→ "compressed" on last coordinate (X, logical vector)	0 1 0 1 / M	3 7 6 2
⌿	Column compression	X ⌿ Y ←→ "compressed" on first coordinate (X, logical vector)	0 1 ⌿ M	8 6 4 2
\	Row expansion	X \ Y ←→ "expanded" on last coordinate	1 1 0 1 1 \ M	1 3 0 5 7 8 6 0 4 2
⍀	Column expansion	X ⍀ Y ←→ "expanded" on first coordinate	1 0 1 ⍀ M	1 3 5 7 0 0 0 0 8 6 4 2
⌽	Row rotation	X ⌽ Y ←→ "rotated" on last coordinate	2 1 ⌽ P	6 10 8 2 0 4
⊖	Column rotation	X ⊖ Y ←→ "rotated" on first coordinate	1 ⊖ P	4 2 0 10 8 6
⍉	Transposition	X ⍉ Y ←→ coordinate I of Y becomes coordinate Y [I] of result	2 1 ⍉ P	10 4 8 2 6 0
∈	Membership	ρX ∈ Y ←→ ρX	7 3 5 2 ∈ 2 4	0 0 0 1
⊥	Decode	X ⊥ Y ←→ Y is transformed to base X	10 ⊥ 3 1 4 2	3142
⊥	Encode	X ⊤ Y ←→ Y is represented in system X	24 60 60 ⊤ 3142	0 52 22
?	Deal	X ? Y ←→ random "deal" of X elements for ιY	3 ? 10	9 2 5
⌹	Domino	See text	(⍉ P) ⌹ ⍉ Q	¯3.671 ¯2.214 1.833 0.833

APPENDIX F
PRIMITIVE OPERATORS

Operator	Name	Definition	Example	Result
f/	Row reduction	f/X ←→ (X[1; . . . ; 1] fX [1; . . . ; 2] f . . . fX [1; . . . ; ρX [ρρX]), (X[2; . . . ; 1] fX [2; . . . ; 2] f . . . fX [2; . . . ; ρX [ρρX]), ... X[ρX [1]; . . . ; 1] fX [ρX [1]; . . . 2] fX [ρX [1]; . . . ; ρX [ρρX]	+/M	16 20
f ⌿	Colum reduction	f⌿X ←→ (X[1; . . . ; 1] fX [2;. . . ; 1] f . . . fX [ρX [ρρX]; . . . ; 1), (X[1; . . . ; 2] fX [2; . . . ; 2] f . . . fX [ρX [ρρX]; . . . ; 2), ... X[1; . . . ; ρX[1]]fX [2; . . . ; ρX [1]]f. . .f [ρX [ρρX]; . . . ; ρX [1]	+⌿M	9 9 9 9
∘ . f	Outer product	X ∘ . fY ←→ yields an array of dimension (ρX), ρY, formed by applying f to every pair of components of X and Y	2 4 ∘ . + 7 5 3	9 7 5 11 9 7
f . g	Inner product	See text	P + . × ⍉ Q	64 236 10 50

APPENDIX G
SPECIAL FUNCTIONS

Function	Name	Description	Example
←	Assignment	A variable to the left of the arrow receives the value specified by the expression to the right of the arrow (see Table 1).	X ← 10
→	Branch	See text	
⎕	Quad	⎕ ← X ←→ prints and storages X X ← ⎕ ←→ accepts any valid numerical expression as keyboard input.	⎕ ← 2 × 7 is 14 3 × ⎕ + 2 ⎕: 7 (input) (27 is the result)
⍞	Quote quad	X ← ⍞ ←→ accepts any valid character expression as keyboard input.	NAME ← ⍞ John (input)

In the appendices

M	P	Q
1 3 5 7	10 8 6	1 3 5
8 6 4 2	4 2 0	7 11 13

and R, non integer such that $3 > R > 0$, and ← → means "is defined as".

REFERENCES

Bueren, H. G., van 1952, *B. A. N.*, **11** No. 432.

Falkoff, A. D. and Iverson, K. E. 1968, *APL\360 User's Manual.* IBM Corp. GH20-0906.

Greville, T. N. E. 1967, *Math. Meth. for Digital Computers*, Vol. 2, ed. A. Ralston and H. S. Wilf (New York: Wiley).

Iverson, K. E. 1962, *A Programming Language* (New York: Wiley).

Iverson, K. E. 1971, *Elementary Algebra,* IBM Philadelphia Scientific Center Technical Report 320-3001.

Kolsky, H. G. 1969, *IBM System Journal,* **8**, No. 3.

Mendoza, E. E. 1967, *Bol. Obs. Tonantzintla y Tacubaya,* **4**, 149.

Mendoza, E. E. 1968, *Ap. J.*, **151**, 977.

Mendoza, E. E. and Gómez T. 1969, *Bol. Obs. Tonantzintla y Tacubaya,* **5**, 111.

Mendoza and Hacyan S. 1974, *Rev. Mex. Astron. Astrof.* **1**, 35.

Müller, G. and Kempf P. 1907, *Publ. des Astrophys. Obs. Potsdam,* **17**.

Wrubel, M. H. 1958, *Handbuch der Physik,* **51**, 1, ed. S. Flugge (Springer-Verlag).