MODELING THE ARITHMETIC OF STATISTICAL DISTRIBUTIONS

Leo H. Groner
International Business Machines Corporation
P. O. Box 390, Poughkeepsie, NY 12602, USA (914) 435-8199

Janice Cook
International Business Machines Corporation
Route 52, East Fishkill, NY 12533, USA (914) 892-2320

ABSTRACT

Examples of extended arithmetics include range, polynomial, dimensional, extended precision and fault tolerant arithmetic, and arithmetic on distributions of random variables. APL2's defined operators and general arrays enable the programmer to easily define such extensions. In this paper we will discuss some extended arithmetics and a prototype of arithmetic on statistical distributions that we have implemented in APL2 [1]. In this prototype, distributions take the place of numbers and the ordinary arithmetic scalar functions are replaced by derived functions that take distributions as arguments and return distributions as results. For ordinary addition, the natural derived function is convolution. Design issues and application examples are discussed.

INTRODUCTION

A major theme in the development of mathematics is the generalization of the concept of number and of operations defined on numbers. For example the concept of number came to embrace rationals, reals, zero, negatives, polynomials, imaginaries, vectors, quaternions, and matrices. Iverson's development of APL was very much in this tradition when it extended ordinary arithmetic and many other mathematical operations to the domain of arrays. The extension of functions and operators to nested arrays in APL2 continues this process. Furthermore, APL2's defined operators and nested arrays are generalizations that make it possible for the programmer to easily define further extensions.

EXTENDED ARITHMETICS

We will begin with a brief discussion of some of the simpler examples of extended arithmetics including arithmetic on rationals, ranges, polynomials, extended precision numbers, dimensional quantities and a fault tolerant arithmetic.

For rational arithmetic, we represent a number by an ordered pair: the numerator and denominator in lowest terms. We can then extend scalar operations so as to always yield exact rational results. For example:

```
{1 3} + {1 2}   yields   {5 6}
```

Or more conventionally,

```
1/3  +  1/2    yields    5/6
```

Rational arithmetic eliminates the inaccuracies of representing rationals as binary fractions in hardware.

In range arithmetic, we represent a number by an ordered pair representing the maximum and minimum values it can take. Scalar operations are extended to yield range results. For example:

```
{.9 1.1} - {.9 1.1} yields  {-.2 .2}
```

Range arithmetic can be used to assess the accuracy of numerical algorithms.

In polynomial arithmetic a number is represented by a vector of real (or complex) numbers representing the coefficients of a polynomial. Scalar operations are extended to yield polynomial results. For example:

```
{1 2 3} x {1 2} yields {1 4 7 6}
```

Or more conventionally,

```
  (1 + 2xX + 3xX*2) x (1 + 2xX)
```

yields

```
  1 + 4xX + 7xX*2 + 6xX*3
```

Extended precision arithmetic is polynomial arithmetic with the constraint that coefficients cannot exceed the largest integer representable by the primitive facilities of the language or hardware. When an excessively large coefficient is computed a "carry" operation occurs. This permits integer arithmetic on very large numbers. This representation is used in the search for large primes as well as in encryption or decryption algorithms.

In dimensional arithmetic, each number is associated with a unit or unit expression.

For example:

{5 inches} x {24 sec/cm} yields {304.8 sec}
{5 inches} + {1.29 hours} yields an error.

Dimensional arithmetic can be viewed as a very strongly typed system for debugging. Such a system has been proposed for PASCAL [2].

In fault tolerant arithmetic, the number system is extended to include a value of "undefined." For example, if we let "undefined" be represented by #

(1 2 3 ÷ 0 1 2) + 2 3 4 yields # 5 5.5

The advantage, of course, is that we can minimize the propagation of the "damage" that errors cause.

THE ARITHMETIC OF STATISTICAL DISTRIBUTIONS

Of particular interest is arithmetic on distributions of random variables. In this arithmetic probability distributions take the place of ordinary numbers. For example, if we let NORMAL (M,S) stand for a normal distribution with mean M and standard deviation S:

{NORMAL(1,3)} + {NORMAL(1.5,4)}

yields

{NORMAL(2.5,5)}

In other words, when we add a sample from the first distribution to an independent sample from the second distribution the sum is characterized by the third distribution. {NORMAL(2.5,5)} is called the convolution of {NORMAL(1,3)} with {NORMAL(1.5,4)}.

To fully extend ordinary arithmetic to the domain of distributions, we replace the ordinary arithmetic functions plus, minus, times and divide by derived functions that take distributions as arguments and return distributions. We have seen that for distributions convolution is the natural replacement function for +. In a similar way we can define -, x and ÷ "convolutions" to represent the distribution of the difference, product or quotient of independent samples from two distributions. For example, suppose that PRICE is a random variable denoting the future price of a product, VOLUME is an independent random variable denoting future sales volume, and REVENUE is a random variable given by PRICE*VOLUME. The distribution of REVENUE can be called the "times convolution" of the distributions of PRICE with VOLUME.

Such problems in statistical modeling require Monte Carlo techniques (usually implemented in a simulation language) or esoteric analytical techniques requiring skills foreign to the problem area. Monte Carlo techniques (frequently computationally expensive) yield numerical results which can be combined to produce numerical distributions. Pure analytic techniques can yield closed form algebraic solutions, but are frequently very difficult to obtain. The technique proposed here lies between these two extremes and may be applied to many problems considered to be too complex for analytical approaches. Unlike Monte Carlo it yields a representation of a distribution. It is not iterative. The compact representation of distributions makes it computationally efficient.

Applications of this statistical modeling technique include investments, reliability, yield forecasting, and queueing systems.

To implement arithmetic on distributions we have written an APL2 operator CON. This monadic operator takes an APL2 scalar function such as +,-,x,÷ or a defined functions as its operand. It takes representations of distributions as arguments. CON maps its operand into a function that yields a distribution in the same representation. The CON operator may be usefully applied to many of the arithmetic primitive functions of APL such =,>,<,∧,∨.

A Toy Example: Computing A Manufacturer's Profits

Let DEMAND be a vector of customer demands by product. Let SALEPRICE be a vector of sales prices by product. Let BOM be a bill of material matrix where BOM[I;J] gives the requirement for input J to produce a unit of product I. Let PURCHPRICE be a vector of the unit prices by input. Then profit is:

```
DEMAND +.x SALESPRICE - BOM +.x PURCHPRICE
```

(See Figure 1). All this is simple APL1 but suppose all four of the variables above have scalar components that are random variables instead of constants and that we wish to know the distribution of profit. If we replaced components of the arrays by representations of distributions (see Figure 2) and replaced the expression above with

```
DEMAND (+ CON).(x CON) SALESPRICE (- CON)
BOM (+ CON).(x CON) PURCHPRICE
```

where + CON, - CON and x CON are derived functions that operate on the domain of distributions, then we could compute the distribution of profits. See Figures 3 and 4.

Example: The Transient Behavior of a Queue

A second example is from queuing. Queuing systems arise in the study of transportation, manufacturing, and computer and communication facilities. Queuing systems are usually studied using either analytical models or simulation. Analytic solutions usually apply only to steady state behavior and assume one of a restricted set of possible arrival and service time distributions.

Transient behavior is usually much less amenable to analytic techniques. We will study the transient behavior of a single server, first come, first served queue. How long does the first, second,..., J'th customer wait for service? Let ARRIVAL and SERVICE be distributions of interarrival and service times at a facility. We wish to calculate WAIT[J], the wait time distribution seen by the J'th arrival. After initializing WAIT[0], the wait time distribution WAIT[J] for the J'th transaction may be calculated by:

```
WAIT[J]←0⌈CON ARRIVAL-CON SERVICE+CON WAIT[J-1]
```

The computation requires three "convolutions:" + CON, - CON and ⌈ CON . See Figures 5 and 6.

SERVICE + CON WAIT[J-1] is the distribution of the sum of the J-1'th wait and the J-1'th service time.

ARRIVAL - CON SERVICE + CON WAIT[J-1] is

```
the distribution of the interval between
completion of J-1'th customer's service
and arrival of the J'th customer.  Since
the J'th transaction may arrive after the
completion of the J-1'th transaction's
service, numbers from this distribution
may be negative.  Finally, ⌈ CON elimi-
nates negative intervals since "negative
wait time" does not offset positive waits.
```

```
SALESPRICE=  5 2 1

DEMAND=  20 50 200

PURCHPRICE=  0.1 0.2 0.3 0.4

BOM=  2 3 0 2.7
      0 4 1 0
      2 0 0 1.35

      PROFIT←DEMAND+.×SALESPRICE-BOM+.×PURCHPRICE

159
```

Figure 1. APL1 deterministic manufacturing model. Profit as a function of sales prices, raw material purchase prices.

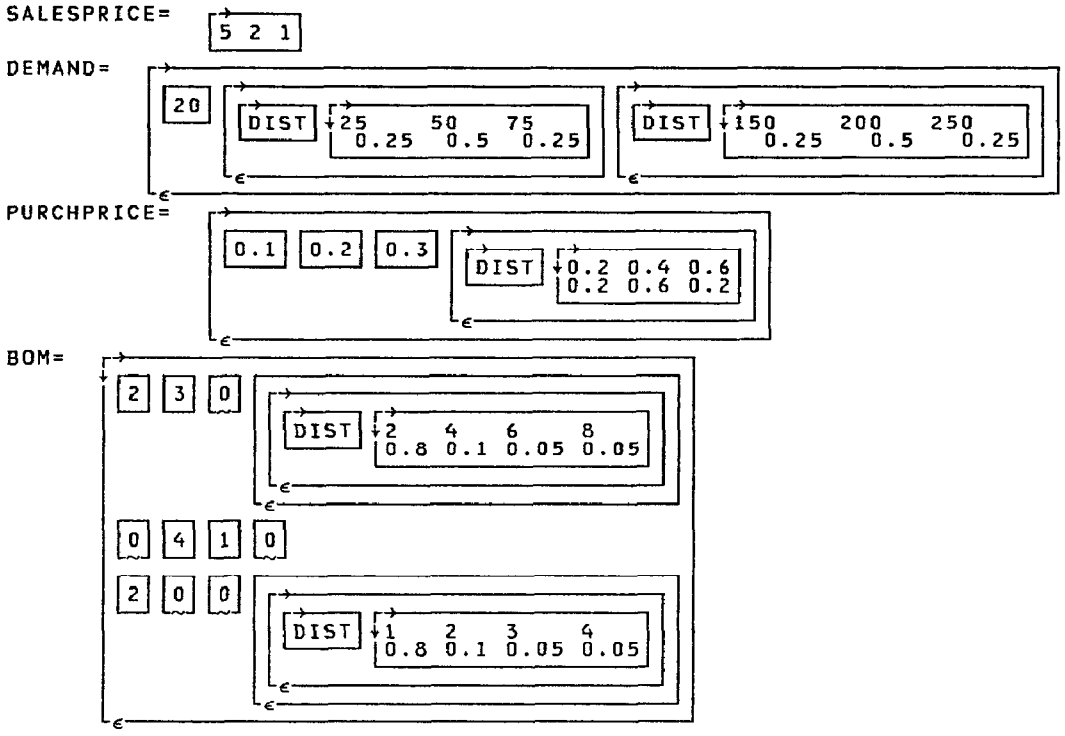

Figure 2. APL2 stochastic manufacturing modeling inputs.

```
    PROFIT←DEMAND(+CON).(×CON)SALESPRICE-CON BOM(+CON).(×CON)PURCHPRICE
PROFIT=
DIST  ¯389        ¯360       ¯323       ¯290      ¯241      ¯196      ¯127      ¯72.2   ¯13.2   41.2    106     150    187    220    251     276    291
        0.00000681  0.0000514  0.000454  0.00169   0.00206   0.00403   0.00835  0.0197  0.0275  0.0407  0.0913  0.233  0.317  0.185  0.0538  0.012  0.00368
```

Figure 3. APL2 stochastic manufacturing model results.

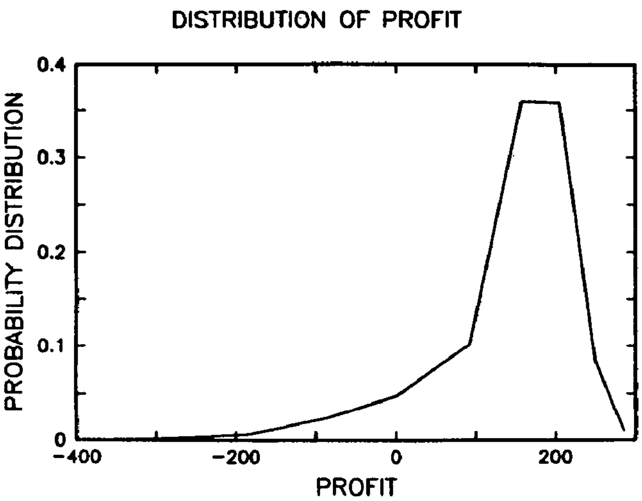

Figure 4. Stochastic manufacturing model results displayed as a plot of the cumulative distribution.

A warning: it is important to examine the properties of the extended numbers and functions. In particular:

A − CON A is not zero and

A × CON (B + CON C) is not equal to (A × CON B) − CON (A × CON C)

Discrete distributions under the operations + CON and × CON each form a monoid since they have closure, identity, and associativity, but no inverse and no distributive laws.

DESIGN ISSUES

Distributions can be represented in many ways: by samples, frequency tables, moments, and by algebraic expressions for the density function, the cumulative distribution function or the characteristic function. Steven Goldstein used sample sets to represent distributions in an approach implemented in APL1 [3]. An ideal representation is compact and easy to convert to its ultimate use. It should be closed under the relevant operations, i.e., the result of a calculation should have the same form as the arguments.

For this prototype, we have chosen to represent a distribution by a frequency table. We use a 2 by N array where the first row represents an ordered value set and the second row represents the corresponding probability of each value. Such tables can approximate continuous distributions. By increasing the number of elements in the frequency table we can come arbitrarily close to any distribution. See Figures 5 and 6. The CON operator transforms any scalar function on numbers to a function on frequency tables. The result is a frequency table. Closure is thus guaranteed.

```
CASE 1, Waiting
 ARRIVAL J-1
 |-------------------------|------------------------|
       WAIT(J-1)                 SERVICE(J-1)
                              ARRIVAL J
 |----------------------------|---------------------|
    INTERARRIVAL TIME(J-1,J)        WAIT(J) ≥0

CASE 2, No waiting
 ARRIVAL J-1
 |-----------------|----------------|---------------------|
       WAIT(J-1)       SERVICE(J-1)         WAIT(J)<0
                                                    ARRIVAL J
 |----------------------------------------------------|
                  INTERARRIVAL TIME(J-1,J)
WAIT[J]←0 ⌈ CON ARRIVAL - CON SERVICE + CON WAIT[J-1]
```

Figure 5. Queuing example with an illustration of the two cases.

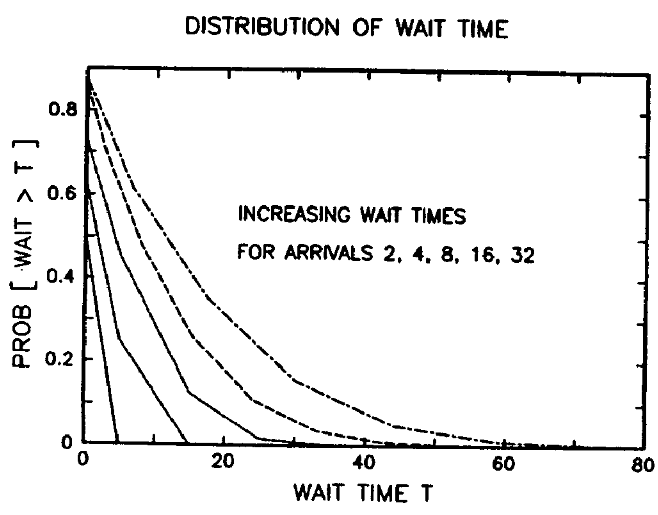

Figure 6. Queuing example, cumulative probability distribution of the J'th job's waiting time.

The frequency table representation of distributions has the disadvantage that with each application of the derived function the value sets, i.e., the number of table entries in the result can grow explosively. We prevent explosive growth by checking for a resulting table size larger than a globally defined limit called SIZE. If the number of table entries exceeds SIZE then table entries are merged until the table has no more than SIZE entries. This compact table then acts as an approximation of the full discrete distribution. Prony's method, suggested to us by Dr. Leo Hellerman of IBM Kingston, motivates our compaction technique [4]. Prony's method represents a distribution D with known moments by a discrete distribution D(K) with K value-probability pairs such that the first 2K-1 moments of D(K) are identical to those of D. To construct D(K) one first constructs a certain K by K matrix from the 2K-1 moments. One then solves an equation relating this matrix to the the K element vector

```
1,(-K)↓MOMENTS
```

The solution is a K element vector defining a polynomial whose K real roots, if they exist, are the desired value set for D(K). Complex roots imply that D(K) does not exist. The last step is to find the K corresponding probabilities by solving an equation relating the value set to the K required moments. We generalized Hellerman's APL1 implementation of Prony's method [5]. We take advantage of APL2 features such as ⌸ and the polynomal root function POLYZ in 1 MAT. (See the PRONY function in Appendix A.) Tables of size 11 or less are usually practical. Larger tables often generate unacceptable numerical error.

In our prototype for distributional arithmetic we at first used enclose to signal that a scalar is to be interpreted as a distribution. Unfortunately, this precluded any other use of enclosure. Now distributions specifically identify themselves as such. For example a distribution D may be formed by:

```
D ← (⊂'DIST') (⊂2 2 ⍴ 1 2 .75 .25)
```

At this time, our prototype assumes independent variates, i.e., every reference to a variate is treated as a separate independent sample. For example,

```
D ×CON 1 1 currently yields D D
```

Thus D D is a two-component vector whose components are independent variates. In principal, multivariate distributions may be treated similarly to univariate ones. Another level of enclose could be used to allow the values of univariate frequency tables to be replaced by enclosed vectors. For example,

```
D ×CON 1 1
```

would lead to a result equivalent to:

```
⊂(⊂'DIST') (2 2 ⍴ (⊂1 1) (⊂2 2) .75 .25)
```

This is a bivariate distribution whose terms have a correlation of one. If D represented the demand distribution for a product and D ×CON 1 1 represented the derived demand for a pair of raw materials used in the product the components of derived demand are clearly correlated and the interpretation of the multivariate implementation would be the correct one.

APL2 SUPPORT OF EXTENDED ARITHMETICS

The efficiency of the CON operator could be greatly improved by handling certain primitive scalar functions as special cases. The +, -, ⌈ and ⌊ convolutions have well-known algorithms that are more efficient than the general algorithm implemented in CON. In dimensional arithmetic, the conformability requirements and the action to be taken are different when + or × is the operand function. However, to make use of these distinct algorithms the code in the operator must be able to recognize the operand function. If the operand is a defined function, then ⎕CR applied to the operand returns the definition of the function as a character matrix. On the other hand, if the operand is a primitive function ⎕CR merely returns an empty matrix as a result. We propose that ⎕CR of a primitive function result in a one by one character matrix containing the symbol for the function. This would permit branches to optimun code, e.g.

```
→('+'=1 ↑,⎕CR 'F')/PLUS
```

CONCLUSIONS

There are many applications for extended arithmetics. Statistical modeling can be approached as an extended arithmetic of distributions. By developing a distributional arithmetic prototype we learned that APL2 has facilities that make it is easy to implement extended arithmetics; but APL2 could, with slight modification, do even better.

Some of the design issues for distributional arithmetic that we address include the represention of distributions and the approximation of bulky discrete distributions by compact ones. Major issues we hope to address in a later paper are correlated variates and multivariate distributions.

REFERENCES

[1] APL2 Programming Language Reference Manual, (SH20-9227-0). IBM Programming Publishing San Jose CA. August 1984.

[2] M.B. Agrawal and V.K. Garg, "Dimensional Analysis in PASCAL", ACM SIGPLAN Notices, Vol. 19, No. 3, pp. 7-11, March 1984.

[3] Stephen N. Goldstein, "Software for Explicitly Probabalistic Mathematics", Proceedings of the Sixth MIT/ONR Workshop on C3 Systems, LIDS-R-1354, July 25-29, 1983, Cambridge, Massachusetts.

[4] Carl-Eric Froberg, Introduction to Numerical Analysis, Addison-Wesley, 1965, pp. 288-290.

APPENDIX A

The CON operator and supporting code:

```
[0]   Z←A(F CON)B;C;P;AA;BB;⎕IO;IDA;IDB
[1]  ⍝APPLY GENERALIZED CONVOLUTION OPERATOR
[2]  ⍝TO FUNCTION F AT ARGS A AND B
[3]   ⎕IO←1
[4]  ⍝ 2 1⍴(DISPLAY A)(DISPLAY B)
```

```
[5]   ⍝  SIMPLE SCALAR OR ARRAY
[6]   L:→((1≥≡⍴A)∧1≥≡⍴B)/'Z←A F B'
[7]    A←REDDEPTH A
[8]    B←REDDEPTH B
[9]   ⍝  ENCLOSED ARRAY,MIGHT BE DISTRIBUTION
[10]   IDA←ISDIST A
[11]   IDB←ISDIST B
[12]   →(IDA∧IDB)/'Z←DIST (2⊃⍴A)F CON0 2⊃⍴B'
[13]  ⍝MUST CHANGE DEPTH OF ARGS
[14]   →(ISDIST⊂A)/'L A←⊂A'
[15]   →(ISDIST⊂B)/'L B←⊂B'
[16]  ⍝ ARRAY, APPLY CON RECURSIVELY TO ITEMS
[17]   →((0<⍴⍴A)∨0<⍴⍴B)/'Z←(⊂⍤A)(F CON)⍤⊂⍤B'
[18]  ⍝   IF  ONE ARG IS SIMPLE SCALAR,
[19]  ⍝   CONVERT IT TO DIST
[20]   IDB←ISDIST B
[21]   →(IDB∧~HASDIST A)/'L A←DIST 2 1⍴(⊂A),1'
[22]   L B←DIST 2 1⍴(⊂B),1
    ∇ 1986-03-13 13.42.29 (GMT-5)

[0]    Z←A(F CON0)B
[1]    C←,A[1;]∘.F B[1;]
[2]    P←,A[2;]∘.×B[2;]
[3]    Z←Z,[.1]⊃⍤((Z←((C⍳C)=⍳⍴C)/C)∘.=C)+.×P
[4]    →(SIZE<1↓⍴Z)/'Z←SIZE COMPACT Z'
[5]    Z←Z[;⍋⊃⍤Z[1;]]
    ∇ 1984-10-31 15.40.22 (GMT-5)

[0]    Z←ISDIST X
[1]    Z←0
[2]    →(3>≡⍴X)/0
[3]    →(0≠⍴⍴X)/0
[4]    →(0=⍴⍴⊃X)/0
[5]    '→0' ⎕EA '→(~''DIST''=⊃1⊃⍴X)/0'
[6]    Z←1
    ∇ 1984-11-01 11.07.02 (GMT-5)

[0]    Z←HASDIST X
[1]   ⍝ DOES X CONTAIN A DISTRIBUTION
[2]    Z←0
[3]    →(3>≡⍴X)/0
[4]    →((3=≡⍴X)∧ISDIST X)/YES
[5]   ⍝→(0<⍴⍴X)/'Z←∨/HASDIST⍤⊂⍤X'
[6]    →(0=⍴⍴X)/'Z←∨/HASDIST⍤X'
[7]    →0
[8]   YES:Z←1
    ∇ 1986-03-13 14.21.29 (GMT-5)

[0]    Z←REDDEPTH X
[1]    Z←X
[2]   L:→(0≠⍴⍴Z)/0
[3]    →(3≥≡⍴Z)/0
[4]   ⍝SCALAR AND DEPTH>3
[5]    →(0=⍴⍴⊃Z)/'Z←⊃Z'
[6]    →L
    ∇ 1984-11-01 11.40.59 (GMT-5)

[0]    Z←SIZE COMPACT D;MAXV
[1]    D[1;]←D[1;]÷MAXV←⌈/|D[1;]
[2]    Z←SIZE PRONY(¯1+2×SIZE←SIZE⌊11)GMOM D
[3]    Z[1;]←Z[1;]×MAXV
    ∇ 1984-03-14 17.37.36 (GMT-5)

[0]    Z←K PRONY M;SKEWM;V;P;COEF;TEMM
[1]   ⍝Z IS THE DESIRED DISCRETE DISTRIBUTION
[2]   ⍝K IS THE NUMBER OF ENTRIES(COLS) IN Z
[3]   ⍝M IS A VEC OF ≥ 2K+1 MOMENTS OF Z
[4]   ⍝Z IS A 2×K MATRIX WHOSE:
[5]   ⍝1ST ROW IS VALUES AND WHOSE
[6]   ⍝2ND ROW IS PROBABILITIES
[7]   ⍝PRONY'S METHOD, FROBERG,
[8]   ⍝INTRO TO NUM ANAL,
[9]   ⍝ADDISON WESLEY 1965, P288-2
[10]  L:→((⍴M)<¯1+2×K)/'ERROR IN LENTH OF M'
[11]   SKEWM←(K,K)↑(¯1+⍳K)⌽(K⍴0)∘.+TEMM←1,(¯1+2×K)↑M
[12]   '→ERROR' ⎕EA 'COEF←1,⌽(-K↓TEMM)⌹SKEWM'
[13]   V←POLYZ COEF
[14]   →(1E¯8<⌈/11○V)/ERRORC  ⍝ IMAGINARY TOO BIG
[15]   V←9○V ⍝ REAL PART
[16]   Z←V,[.1]P←(1,(K-1)↑M)⌹⍉V∘.*0,⍳K-1
[17]   →0
[18]  ERRORC:'COMPLEX ROOTS'
[19]   'NEW K: ' K←K-1
[20]   →0
[21]  ERROR:K←K-1
[22]   ⎕EM
[23]   'NEW K: ',K
[24]   →L
    ∇ 1986-03-13 14.44.33 (GMT-5)

[0]    Z←N GMOM X
[1]    Z←X[2;]+.×X[1;]∘.*⍳N
    ∇ 1984-03-13 12.44.57 (GMT-5)
```

APPENDIX B

Some simple examples of the CON operator:

```
      P← DIST 2 5⍴.1 1 8 8.1 20 .2 .2 .2 .2 .2
      P
DIST    0.1 1   8   8.1 20
        0.2 0.2 0.2 0.2  0.2
```

Compare the effects of setting SIZE to 999 and to 3. Note that the results are identical up to the first five moments.

```
      SIZE←999
      MOM ⎕←P + CON P
DIST    0.2   1.1   2     8.1   8.2   9     9.1   16    16
        0.04  0.08  0.04  0.08  0.08  0.08  0.08  0.04  0

       .1   16.2   20.1   21     28     28.1   40
       .08   0.04   0.08   0.08   0.08   0.08   0.04
1 14.88 17.971 20.292 22.193 23.81 25.212 26.441
  27.524

      SIZE←3
      MOM ⎕←P + CON P
DIST    3.6619   18.268    35.701
        0.36773  0.51853   0.11374
1 14.88 17.971 20.292 22.193 23.81 25.179 26.325
  27.279

      P - CON P
DIST    16.388    2.9637E¯16 16.388
          0.18905 6.2190E¯1    0.18905

      P × CON P
DIST    7.763    138.02     396.21
        0.71891    0.23857    0.042517

      P ÷ CON P
DIST    1.9945   74.702     199.38
        0.86616   0.093048    0.040794

      P ⌈ CON P
DIST    0.71472 8.039   19.998
        0.15763 0.4822  0.36018

      1 + CON P
DIST    1.4757  8.9811   20.992
        0.39297 0.40651  0.20051

      1 + CON 2
3

      + CON/4⍴P
 DIST    11.452    33.293    59.065
           0.29485   0.59236   0.11279

      ⌈ CON/4⍴P
 DIST    0.92329  8.0635  19.999
         0.025169 0.38431  0.59052

      ⎕←P4←4⍴P
DIST    0.1 1   8   8.1 20       DIST   0.1 1   8
        0.2 0.2 0.2 0.2  0.2             0.2 0.2 0.2

      8.1 20       DIST   0.1 1   8   8.1 20       DIS
      0.2  0.2            0.2 0.2 0.2 0.2  0.2

      T   0.1 1   8   8.1 20
          0.2 0.2 0.2 0.2  0.2

      P4+ CON P4
 DIST    3.6619   18.268    35.701       DIST   3.661
         0.36773  0.51853   0.11374             0.367
```

```
      9  18.268    35.701        DIST    3.6619  18.268
      73  0.51853   0.11374              0.36773  0.518

         35.701         DIST    3.6619  18.268    35.701
      53  0.11374               0.36773  0.51853   0.113

      74

    P4 + CON P
DIST    3.6619  18.268    35.701         DIST    3.661
        0.36773  0.51853   0.11374               0.367

      9  18.268    35.701        DIST    3.6619  18.268
      73  0.51853   0.11374              0.36773  0.518

         35.701         DIST    3.6619  18.268    35.701
      53  0.11374               0.36773  0.51853   0.113

      74

    P4 + CON 0
DIST    0.47567 7.9811  19.992         DIST    0.4756
        0.39297 0.40651  0.20051               0.3929

      7 7.9811  19.992         DIST    0.47567 7.9811
      7 0.40651  0.20051               0.39297 0.40651

      19.992         DIST    0.47567 7.9811  19.992
       0.20051               0.39297 0.40651  0.20051

    1 2 +CON 1 (DIST (2 1p2 1))
2   DIST    4
            1
```

Some examples of the interaction of the CON operator with other operators:

```
    + CON/P4 xCON P4

DIST    74.757  358.83    796.34
         0.5454   0.41456   0.04004

    P4 (+ CON).(x CON) P4

DIST    74.757  358.83    796.34
         0.5454   0.41456   0.04004
```