e^x, LOGARITHMS, AND THE NORMAL DISTRIBUTION

Dallas R. Hodgins

Medical Information Systems Department
Naval Health Research Center
P. O. Box 85122
San Diego, California 92138-9174

Report 88-9, supported by the Naval Medical Research and Development Command, Department of the Navy, under work unit M0095.005-1053. The views expressed in this article are those of the author and do not reflect the official policy or position of the Department of the Navy, Department of Defense, nor the U.S. Government. Approved for public release, distribution unlimited.

The author wishes to acknowledge the able assistance of Kathryn Medrano in preparing this paper.

Summary

Fast, accurate numerical algorithms for e^x and the logarithms of numbers are necessary to develop useful statistical models such as a rational polynomial approximation of the normal probability density function integral.

Exploiting the string functions $EXTRACT, $FIND, and $LENGTH of the MUMPS programming language, extremely precise algorithms are presented for e^x, the natural and common log of N, the error function, and the normal probability density function.

The standardized normal variable distribution routine presented is accurate to 1.5 parts in ten million - affording the analyst comfortable margins in models requiring extensive numeric manipulation.

e^X, LOGARITHMS, AND THE NORMAL DISTRIBUTION

Dallas R. Hodgins

Introduction

Since the Veteran's Administration File Manager (FM) does not presently have a very wide selection of statistical tools, the Naval Health Research Center initiated a program to enhance FM's statistical capabilities. To date, routines for basic descriptive statistics[1], a random number generator, multiple regression, and the routines presented herein have been completed. The philosophy has been to develop accurate, concise structures that industrial hygienists in shipyards, medical workers, and administrators can easily and comfortably use in dealing with small samples.

Much emphasis has been placed on using FM relationally. The dovetailing of relational data structure, the algorithms, and the coding is a remarkable outcome of the essential linearity of a relational logical view of the data. The routines presented in this paper are algebraic structures written using linear algebra notation. The correlation between the mathematical models used and the computer models developed is emphasized - not to pontificate, but to stress the inherent integrity of the methods and to underscore the accuracy and conciseness attainable in routines written in MUMPS. MUMPS is a flexible media that not only allows great latitude of expression, but is logically appealing, allowing code structures that are aesthetically attractive.

The goal is to place in the hands of researchers the ability to sample their data, order it, and derive basic inferences without the "blackbox" alienation induced by the magnificent, but overwhelming, services of some of the popular commercial packages. FM and the programs in this paper are in the public domain. The source code is presented for all to critique - which is the pleasure and challenge of an open forum.

The progression of this work has been to develop the elementary measures of central tendency with an emphasis on indexing so that in sorting, for example, entities can be compared across domains, missing data handled gracefully, and fast processing achieved. The taming of e^x in this paper, combined with the existing programs, allows the researcher to deal with probability distributions germane to his actual data. For example, the mean and standard deviation of the weight of the men (or women) of a particular shipyard can now be

computed, and the probability of the occurrence of an individual weight can be computed at once using NORMDIST.

Methods and Materials

The routines were developed using Intersystems MUMPS M/VX 3.1 on a Digital Equipment Corporation (DEC) VAX 11//85 machine. The programs meet or exceed the accuracy stated and have been thoroughly tested against values published in the Handbook of Mathematical Functions of National Bureau of Standards (Applied Mathematics Series 55, ninth printing. November 1970). In the discussion of the routines, N is any real number. The discrepancy between the symbols used in the discussion and the symbols in the routines stems from using conventional mathematical notation to write the algorithms ("x" is always the working variable, etc.) and use of symbols in the meta mathematical language that convey meaningful images.

We shall develop an algorithm to produce natural logarithms (ln), then a program to yield a number if given its logarithm, and finally relate these inverse processes to the normal probability density function. While the models are rigorous in a mathematical sense, the exposition is not.

It is of great convenience to represent numbers, N, by raising e (the Naperian or natural logarithm base: $e = \lim(1+1/n)^n$ as n goes to infinity) to a power b:

$$N = e^b$$

By definition, the ln of N is b

$$\ln N = b \ln e = b$$

as the ln e = 1 (if N = e, we have $e = e^b$ so ln e = b ln e, or b = 1; $e = e^1$).

Given N, we want an expression that will produce b, the ln of N. We could use Taylor's series expansion for

$$1/(1+x) = 1 - x + x^2 - x^3 \ldots \text{ for } |x|<1$$

and integrate term by term to obtain

$$\ln(1+x) = x - x^2/2 + x^3/3 \quad x^4/4 \ldots \text{ for } |x|<1$$

The problem with this method is its slow convergence which leads to the need of many terms, thereby increasing the possibility of error. Hastings[2] approximated this power series with the polynomial

$$\ln(1+x) = a_1x + a_2x^2 + a_3x^3 \ldots a_8x^8 + \varepsilon(x) \qquad (1)$$

for $0 \leq x \leq 1$ where $|\varepsilon(x)| \leq 3E\text{-}8$ and

$a_1 = .99999\ 64239$ $\quad a_5 = .16765\ 40711$

$a_2 = -.49987\ 41238$ $\quad a_6 = -.09532\ 93897$

$a_3 = .33179\ 90258$ $\quad a_7 = .03608\ 84937$

$a_4 = -.24073\ 38084$ $\quad a_8 = .00645\ 35442$

The discerning reader will note that x greater than 1.0 are not fit for the formula. What do we do when we want the ln of a number greater than 2? We must convert numbers greater than 2 to the form (1+.bbb...) and subsequently retransform the value obtained from equation (1) (this is the only noteworthy activity in the routine LOG).

Turning to LOG (figure 1), let us step through the process as we look at the arithmetic for the ln of N = 22.345. At CHAR^LOG we set Z = 22.345 the N selected at SEL. The hub of the game in logarithms is dealing with the decimal position of N. The MUMPS $FIND(Z,".") function returns 0 if there is

```
LOG     ;NAPERIAN AND BRIGGSIAN LOGARITHMS,DRH,NHRC,1/12/88
        ;POLYNOMIAL APPROXIMATION - ABSOLUTE ERROR LESS THAN OR EQUAL TO
3*10E-8)
        S N(1)=0,N(2)=.6931471806,N(3)=1.0986122887,N(4)=1.3862943611,N(5)=
1.6094379124,N(6)=1.7917594692,N(7)=1.9459101491,N(8)=2.0794415417,N(9)=
2.1972245773
SEL     S L=0 R !!,"WHAT ARGUMENT ? (USE DECIMAL POINT): ",X I X="" K C1,L,X
Q
        I X<0!(X=0)!(X'?.N1".".N) W !,"NUMBER GREATER THAN 0 - NO COMMAS
USE DECIMAL POINT" G SEL
CHAR    S Z=X,M=$F(Z,".")
        I M=0 S C=$L(Z)-1 F J=1:1:C S X=X/10
        I M>2 S C=M-3 F J=1:1:C S X=X/10
        I M=2 S TM=$L(Z)-1 F J=1:1:TM S X=10*X
        I M=2 S TM2=$L(X),C=-(TM-TM2+1) F J=1:1:(TM2-1) S X=X/10
        S C1=$E(X,1),X=X/C1-1
LnX     S T(0)=1,T(1)=X F J=1:1:8 S T(J)=T(1)*T(J-1)
        S L=(.9999964239*T(1))-(.4998741238*T(2))+(.3317990258*T(3))-
(.2407338084*T(4))+(.1676540711*T(5))-(.0953293897*T(6))+(.0360884937*T(7))-
(.0064535442*T(8))
        W !,"THE LOG,BASE 10,OF ",Z," IS ",$J(.4342944819*(L+N(C1)+
(C*2.302585093)),10,8)
        W !,"THE LOG,BASE e, OF ",Z," IS ",$J((L+N(C1)+(C*2.302585093)),10,8)
        K A,C,J,L,M,S,T,TM,TM2,Z G SEL
        Q
```

Figure 1. Natural Logarithm

no decimal point in the number, 2 if N is of the form .xxx..., and an integer equal to two more than the number of integers preceding the point in all other cases--$F(22.345,".")=4. Alternate paths to this information are more difficult without direct access to the registers, as any Fortran programmer will testify. For instance, you can replace $F with

```
S N=22.345 F J=1:1 S C=J,Y=N/10,N=Y I Y<1 S C=C-1 W !,C Q
```

and a similar line for numbers less than 1. Setting M = $F(Z,".") has three possibilities:

1) when M=0, there is no decimal point in Z; therefore the characteristic of Z is $LENGTH(Z)-1 (logarithms are traditionally reported as .aaa...Ec (where E stands for "exponent"). The part .aaa... is called the mantissa. Ec is 10^c (which places the decimal) with c being known as the "characteristic");

2) if M>2, there is at least one digit in front of the decimal. We set c=M-3, as we want Z in the form a.aaa... as the ln(a.aaa...) is .bbb.... We subtract three in order to leave a digit, avoiding counting the decimal, and to account for the fact $F goes one beyond the decimal. Knowing the characteristic we now divide Z by 10^c

```
F J=1:1:C S Z=Z/10
```

to yield 2.2345E1 the logarithmic form we seek.

3) M=2 means we have a number like .aaa... or .000...aaa... which is slightly more complicated. If we wanted ln(.000123) we would do the following: S TM=$L(.000123)-1 which equals 6; multiply .000123 by 10 six times (see CHAR+3) giving us 123; setting TM2=$L(123)=3 allows us to calculate c, c=-(TM-TM2+1) or c=-(6-3+1)=-4 giving us 1.23E-4, the form we desire for ln(.000123).

At this point, we have the code to reduce all numbers to the form a.aaa...Ec, and we know the characteristic c. Our example 22.345 has become 2.2345E1 with c=1. What we need for the approximation in equation (1) is .aaa... as we are going to compute ln(1+.aaa...). Therefore, we set C1=$EXTRACT(2.2345) which gives us the 2 in front of the decimal and divide 2.2345 by C1(=2) to obtain 1.11725 from which we subtract 1 yielding .11725.

We are now ready to evaluate ln(1+.11725) with x=.11725. Line LnX+1 multiplies the terms, sums them, and places the result in L. The last chore is to untrack L=ln(1.11725) as we are seeking ln(22.345). What we did, actually, was first divide 22.345 by 10^C (where c was 1), then we divided by C1 (where C1 was 2); so L really is the value of $\ln(22.345/C1*10^C)$ or the ln(22.345/2*10). By generally accepted rules

$$\begin{aligned} L = \ln[22.345/(C1*C)] &= \ln(22.345) - \ln(C1*C) \\ &= \ln(22.345) - [\ln C1 + \ln C] \\ &= \ln 22.345 - \ln C1 - \ln C \end{aligned}$$

but C is actually 10^C, so

$$L = \ln 22.345 - \ln C1 - C \ln 10$$

Shifting the terms about, our answer has the form

$$\begin{aligned} \text{Ln } 22.345 &= L + \ln C1 + C \ln 10 \\ &= .11087 + .693147 + (1*2.302585) \\ &= 3.1066 \end{aligned}$$

The only item left needing explanation is C1 which takes on the values 0,1,2...9 and these lns are in N(1)...N(9). The code in LnX+3 reads

ln Z=$J((L+N(C1)+(C*2.30...)),10,8)

The Briggsian or common Log of N is simply the constant .434... times the ln X. Plainly, the efficacy of the process lies in the string functions $F, $L, and $E.

Computing the antilog e^x is more straight forward. Given x the natural logarithm of N, what is N? This time we "borrow" our polynomial approximation from Messrs. B. Carlson and M. Goldstein of the Los Almos Scientific Laboratory[3]:

$$e^x = 1 + a_1x + a_2x^2 + a_3x^3 \ldots a_7x^7 + \varepsilon(x) \qquad (2)$$

where $0 \le x \le \ln 2$, $|\varepsilon(x)| \le 2E\text{-}10$ and

a_1 =	.99999 99995	a_5 =	.00830 13598
a_2 =	.49999 99206	a_6 =	.00137 98820
a_3 =	-.16666 53019	a_7 =	-.00014 13161
a_4 =	.04165 73475		

Note that great care was taken to ensure numerical accuracy in the routine EXPX. The E(0) to E(20) values (e^x where x = 0,1,2...20 respectively) are 12 digits to ensure our 2 parts in 10 billion accuracy. There are two points to note about the polynomial approximation: 1) we are finding e^x so we must

compute the reciprocal of equation 2 eventually; and 2) the polynomial is accurate with augments less than or equal to ln 2 (.6931471).

The first part of EXPX (see figure 2) is self explanatory. We accept only N less than 20 in SEL, as larger N cause errors greater than ± 2E-10. Since negative N are fair game, we set switch SW2=1, make N positive, and take the reciprocal later for them.

At DECIMAL^EXPX we start to explore our N. If N is negative and M=0 (i.e. $F found no decimal point), we simply print the appropriate reciprocal of E(N) (i.e. 1/E(N)). If M = 0 and N is positive, we again print the answer E(N) immediately. IF M>2, we set C = M-2 to capture the number of digits before the decimal; set CC = $E(N,1,C) to obtain the actual number before the decimal point (remember it can only be one of the set 2,3,4...20); and set MM = N-CC to obtain the decimal fraction. If M=2, indicating a number less than one, we set MM = N, and CC = 0.

```
EXPX    ; e RAISED TO POWER X (X<=20),DRH,NHRC,1/12/88
        ;POLYNOMIAL APPROXIMATION - ABSOLUTE ERROR LESS THAN OR EQUAL TO
2*E-10
        S E(0)=1,E(1)=2.71828182846,E(2)=7.3890560989,E(3)=20.0855369232,
E(4)=54.5981500331,E(5)=148.413159103,E(6)=403.428793493,E(7)=1096.63315843,
E(8)=2980.95798704,E(9)=8103.08392758,E(10)=22026.4657948,E(11)=59874.1417152,
E(12)=162754.791429,E(13)=442413.392009,E(14)=1202604.28416,E(15)=3269017.3724
7,E(16)=8886110.52051,E(17)=24154952.7536,E(18)=65659969.1373,E(19)=178482300.
963,E(20)=485165195.41
SEL     K X,Z S SW=0,SW2=0 R !!,"WHAT ARGUMENT (X<=20) ? ",X I X="" K
SW,SW2,E Q
        I X=-1 W !,"EXP -1 = .36787944117" G SEL
        I X<0 S SW2=1,X=-X
        I X=0 W !,"EXP 0 = 1" G SEL
        I X=1 W !,"EXP 1 = 2.71828182846 WHICH IS ""e""" G SEL
DECIMAL S Z=X,M=$F(Z,".")
        I (SW2)&(M=0) W !!,"EXP ",-Z," = ",$J(1/E(Z),10,10) G SEL
        I M=0 W !,"EXP ",Z," = ",E(Z) G SEL
        I M>2 S C=M-2,CC=$E(Z,1,C),MM=Z-CC
        I M=2 S MM=Z,CC=0
        I MM>.69314718056 S MM=MM/2,SW=1
POLY    S T(0)=1,T(1)=MM F J=1:1:7 S T(J)=T(1)*T(J-1)
        S EX=1-(.9999999995*T(1))+(.4999999206*T(2))-(.1666653019*T(3))+
(.0416573475*T(4))-(.0083013598*T(5))+(.001329882*T(6))-(.0001413161*T(7))
        I SW S EX=EX*EX
        I SW2 W !!,"EXP ",-X," = ",$J(1/((1/EX)*E(CC)),10,10) G KL
        W !!,"EXP ",X," = ",$J((1/EX)*E(CC),10,10)
KL      K C,CC,EX,J,M,MM,SW,SW2,T,X,Z G SEL
        Q
```

Figure 2. e^x

Now we must check to see if the decimal fraction is less than ln 2. If it is not, we divide by 2 and set switch SW = 1. In POLY we perform exactly the operations as in LOG, placing the evaluation of the polynomial in EX. If SW = 1, we undo our division by 2 by setting EX = EX*EX as $e^{-MM/2} * e^{-MM/2} = e^{-MM}$. If SW2 = 1, we change the sign of N back to -1 and print the reciprocal of the reciprocal of EX times E(CC). Nonnegative N are printed as the reciprocal of EX times E(CC) as EX is e^{-X} if you recall.

As an example, let us find the antilog of 3.415 (i.e., evaluate $e^{3.415}$ which is $e^{3} * e^{.415}$). We would read E(3) and evaluate EX for .415 and multiply E(3) times EX for the answer 20.0855369232 * 1.5143707 = 30.417.

The third routine, NORMDIST, is the least accurate, but not to worry, we will replace it later. It is the most interesting of the three - actually what this is all about. The standardized normal random variable probability function is

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \tag{3}$$

which is arrived at by setting the mean u=0 and standard deviation σ=1 in the Gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \tag{4}$$

If we know the mean μ and the standard deviation σ of a population we can use (3) by transforming our score x to a standardized normal variable $z=(x-\mu)/\sigma$.

For example, if we have a population mean u=2.5 and standard deviation σ=1.5 and select a subject whose raw score is 4.96, what is the probability this score is greater than zero and less than or equal to 4.96? First,

$$z = (4.96-2.5)/(1.5) = 1.64.$$

The answer is

$$P(0<Z<1.64) = \frac{1}{\sqrt{2\pi}} \int_{0}^{1.64} e^{-.1.3448} dx$$

the area under the density function from 0 to 1.64.

NORMDIST simply produces the familiar tabular values found at the back of every statistics text. Knowing the mean and standard deviation of any Gaus-

sian distribution, one can translate any measure to the standardized normal variable by $z = (x-u)/\sigma$ and plug it into NORMDIST to get:

i) P(0<Z<z)

ii) P(Z>z)

iii) P(Z<z)

iv) P(|Z|<z)

The values returned are in error to the extent of ± 5 units in the fourth decimal digit.

The numerical analysis is indirect. It turns out the integral of $e^{-x2}dx$ cannot be integrated in finite terms. We could, as with e^x, use a Taylor's series expansion around points of interest to ensure convergence, but it is much easier to pluck Hastings'[2] brain. The error function

$$\text{erf } z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-x^2} dx \tag{5}$$

is close to what we are after; namely,

$$f(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-t^2/2} dt \tag{6}$$

Hastings has approximated the erf with a rational approximation

$$\text{erf } x = 1 - 1/(1 + a_1x + a_2x^2 + a_3x^3 + a_4x^4)^4 + \varepsilon(X) \tag{7}$$

X real and $|\varepsilon(X)| \leq$ 5E-4 where

$a_1 = .278393$ $a_2 = .230389$

$a_3 = .000972$ $a_4 = .078108$

We must make the transformation

$$-t^2/2 = -x^2 \quad \text{or} \quad \frac{t}{\sqrt{2}} = x$$

letting

$$dt = \sqrt{2}\, dx$$

and substituting $-x^2$ for $-t^2/2$ and $\sqrt{2}\,dx$ for dt in equation (6) we arrive at

$$f(w) = \frac{1}{\sqrt{2\pi}} \int_0^w e^{(-x^2)} (\sqrt{2}\,dx)$$

which reduces to

$$f(w) = \frac{1}{\sqrt{\pi}} \int_0^w e^{-x^2} dx \qquad (8)$$

The expression (8) differs from the approximation (7) for the erf (5) by a factor of 2

$$\frac{2}{\sqrt{\pi}} \text{ is twice } \frac{1}{\sqrt{\pi}}$$

so we must divide equation (7) by 2 for the result in line SEL+6 in NORMDIST:

```
S ANS = $J((1-(1/S4))/2,6,4)
```

Turning to NORMDIST (see figure 3), we place in the A(i) the coefficients of the rational approximation (7). Entering an N at SEL, we make our variable transformation by setting $X = N/(2^{1/2})$. After computing the power terms T(J), multiplying the A(i)*T(J), adding 1 and summing, the sum is raised to the fourth power. Following the form of equation (7) we take the reciprocal of the sum, subtract the result from 1 and divide by 2 to reconcile the multiplicative terms of the integral.

Discussion

There are three pedestrian matters to dispose of. First, the accuracy of erf is not consistent with LOG and EXPX. The reader is urged to use Hastings' rational approximation

$$\text{erf } x = 1 - 1/(1 + a_1x + a_2x^2 + a_3x_3 \ldots a_6x^6)^{16} + \varepsilon(X)$$

$0 \leq x \leq \infty$ where $|\varepsilon(X)| \leq 3E\ 7$ and

$a_1 = .07052\ 30784$ $a_2 = .04228\ 20123$

$a_3 = .00927\ 05272$ $a_4 = .00015\ 20143$

$a_5 = .00027\ 65672$ $a_6 = .00004\ 30638$

This expression was not used originally, as there was no reasonable way to get the 16th power without risking grievious errors - motivating writing the LOG and EXPX routines.

```
NORMDIST ;STANDARDIZED NORMAL VARIABLE DISTRIBUTION,DRH,NHRC,1/12/88
         ;ABSOLUTE ERROR IN Z LESS THAN OR EQUAL TO 5*10E-4
         S A(1)=.278393,A(2)=.230389,A(3)=.000972,A(4)=.078108
SEL      S SN=1 R !!,"ENTER z SCORE : ",Z I Z="" K A,Z,ANS,SN Q
         I Z<0 S Z=-Z,SN=-1
         S X=Z/1.4142136,S=0
         S T(0)=1,T(1)=X F J=1:1:4 S T(J)=T(1)*T(J-1)
         F J=1:1:4 S S=S+(A(J)*T(J))
         S S=S+1,S4=S*S*S*S
         S R=(1-(1/S4))/1.1283792,ANS=$J(R*.3989472*1.4142136,6,4)
ANS      W !!,"IF Z IS THE STANDARD NORMAL RANDOM VARIABLE ",SN*Z," THEN :"
         I SN<1 G NEG
         W !,"P(0 < Z < ",Z,")= ",ANS
         W !,"P(Z > ",Z,")= ",.5-ANS
         W !,"P(Z < ",Z,")= ",.5+ANS,?33,"(NOTE: ERROR IN 4TH DIGIT ±5)"
         W !,"P(|Z| < ",Z,")= ",2*ANS G KL
NEG      W !,"P(Z < ",-Z,")= ",.5-ANS
         W !,"P(Z > ",-Z,")= ",.5+ANS
         W !,"P(",-Z," < Z < 0)= ",ANS
         W !,"P(|Z| > ",-Z,")= ",2*ANS,?33,"(NOTE: ERROR IN 4TH DIGIT ±5)"
KL       K J,R,S,S4,T,X,Z G SEL
         Q
```

Figure 3. The Standardized Normal Variable Distribution

Enter the new A(i), change SEL+3 to J=1:1:6, add 1 to S, and take the ln(S+1):

$$N=(S+1)^{16}$$

$$\ln N = 16\ln(S+1)$$

(You will have to change LOG and EXPX to operate as subroutines: i.e. instead of W !, you must set "ANS1=" to the value computed by LOG, etc.)

```
S = (S+1) D^LOG S S3=ANS1*16
S X=S3 D^EXPX S S4=ANS2
```

N is ANS2 from EXPX which computed e^{S3} for you. The rest of NORMDIST remains unchanged and you have now achieved 3 parts in ten million accuracy; your

answer is in error ± 3 units in the seventh decimal place.

If the erf is of no interest, Hastings'[2] rational approximation

$$P(x) = 1 - 1/2(1 + F_1x + F_2x^2 + F_3x^3 \ldots F_6x^6)^{-16} + \varepsilon(x) \qquad (9)$$

where $|\varepsilon(x)| < 1.5\times10^{-7}$ and

F_1 = .04986 73470 F_4 = .00003 80036
F_2 = .02114 10061 F_5 = .00004 88906
F_3 = .00327 76263 F_6 = .00000 53830

will produce

$$F(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2}\, dt$$

to an accuracy of one and one half parts in ten million. The code for (9) is displayed in the routine NORM (see figure 4). Note the line SEL+6 which assumes our LOG and EXPX routines have been converted to subroutines LGN and EXX respectively. Following the style of FM we enter subroutines with x and come out with y.

Second, the use of the error function to evaluate the standardized normal variable function does not preclude us from using it, also, as part of our armament. The erf(ha) is the probability that the error of a single measurement lies between ± a, where h is the precision index. Those using a digital approach to neural nets will appreciate the utility of both the error function and the normal probability density function in adjusting weights in logical threshold units.

Third, standardizing a random variable is, in itself, a powerful ploy. Setting $z=(x-u)/\sigma$ conveys a wealth of information about the location and status of a score in the distribution, if one keeps in mind that the distribution mean is zero and the standard deviation is one. Negative z are less than the population mean - positive z greater. Given a z of 1.64 you know instantly this is a score almost two standard deviations above the mean of the population - a reasonably rare event. Sixty-eight percent of the distribution lies between ±1 standard deviations. Ninety-five percent between ±2 standard deviations. Knowing these relationships and using NORMDIST can rationalize a myriad of guessing situations.

Great care has been taken to make these routines, and those previously developed, as accurate as practically possible. The concept of algorithms for small samples is not whimsical. If one has a small sample with missing data

and assumes an N which counts missing data as present, the mean will consistently be underestimated, a serious matter in the health care business.

```
NORM    ;STANDARDIZED NORMAL VARIABLE DISTRIBUTION,DRH,NHRC,1/12/88
        ;ABSOLUTE ERROR IN Z LESS THAN OR EQUAL TO 1.5*10E-7
        S F(1)=.049867347,F(2)=.0211410061,F(3)=.0032776263,F(4)=.0000380036,
F(5)=.0000488906,F(6)=.000005383
SEL     S SN=1 R !!,"ENTER z SCORE : ",Z I Z="" K A,Z,PZ,SN Q
        I Z>6 W !,"OUT OF RANGE- SELECT A ""z"" LESS THAN OR EQUAL TO 6 " G
SEL
        I Z<0 S Z=-Z,SN=-1
        S (X,ZZ)=Z,S=0
        S T(0)=1,T(1)=X F J=1:1:6 S T(J)=T(1)*T(J-1)
        F J=1:1:6 S S=S+(F(J)*T(J))
        S S=S+1,X=S D ^LGN S S3=16*Y,X=S3 D ^EXX S S4=Y
        S R=(1-(1/(2*S4))),PZ=$J(R,9,7)
        W !!,"IF Z IS THE STANDARDIZED NORMAL RANDOM VARIABLE AND z =
",SN*ZZ," THEN :"
        I SN<1 G NEG
        W !,"P(0 < Z < ",ZZ,")= ",PZ-.5
        W !,"P(Z > ",ZZ,")= ",1-PZ
        W !,"P(Z < ",ZZ,")= ",PZ
        W !,"P(|Z| < ",ZZ,")= ",2*(PZ-.5) G KL
NEG     W !,"P(Z < ",-ZZ,")= ",1-PZ
        W !,"P(Z > ",-ZZ,")= ",PZ
        W !,"P(",-ZZ," < Z < 0)= ",PZ-.5
        W !,"P(|Z| > ",-ZZ,")= ",2*(PZ-.5)
KL      K J,R,S,S4,T,X,Z G SEL
        Q
```

Figure 4. Precision Standardized Normal Variable Distribution

Large samples take care of themselves. The law of large numbers, or the central limit theorem, grant reprieve to shoddy data analysis practices. In the FM descriptive statistics programs, each field is restricted to numeric values in a definite range for that domain, ensuring data attribute integrity. The existence or nonexistence of an entity in that field is ascertained and only then is N augmented or decreased. These are minimal mechanical safe guards.

When looking at the normal distribution function

$$f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-(x-\mu)^2/2\sigma^2}$$

it becomes apparent that one must minimize the error in $(x-\mu)^2$ in order to minimize the error in the integration process. This is why we needed to

develop LOG and EXPX to write a reasonable standardized normal distribution function. Multiplying a slight error in X sixteen times is untenable (for example .99 to the sixteenth power = .85). On the other hand, underestimating u by including missing data in the count is equally serious and very misleading in small samples.

The DEC machines have 18 decimal digit accuracy (they actually carry 19 digits) with 64 bit precision. This is automatic double precision arithmetic. A VAX 750 running under the UNIX operating system scored the lowest error rating in a PC Tech Journal accuracy benchmarking test based on stringent numerical criteria [4].

With accurate algorithms and accurate machines, all that remains is an accurate compiler. The MUMPS community must pay attention to the IEEE p. 754/854 standards for numerical computation which define procedures for dealing with a discontinuous number space. If MUMPS is to gain the preeminence it deserves, it must handle numbers with precision and efficiency. Some companies utilizing co-processors are certainly headed in the right direction, producing native mode machine code and using runtime systems that handle indirection and the Xecute command[5].

The one fly in the ointment of the MUMPS language itself is the order of arithmetic operation in an expression. Countless hours have been spent discovering MUMPS has left to right arithmetical precedence! Otherwise, after many years of massaging numbers, it can be truthfully reported that doing numerical analysis with MUMPS is a pleasure. I/O is the easiest of any language used. The string manipulators are without parallel in examining numbers. The fact that one can simulate the normal probability density integral in four or five lines of code speaks for itself.

REFERENCES

1. Hodgins, D.R. Descriptive Statistics Using the Veterans Administration File Manager as a Relational Database Management System. NHRC Report No. 87-22, 1987.

2. Hastings, Jr., C. Approximations for Digital Computers. Princeton University Press, Princeton, NJ, 1955.

3. Carlson, B., Goldstein, M. Rational Approximations of Functions, Los Alamos Scientific Laboratory Report No. LA-1943, Los Alamos, N. Mexico, 1955.

4. Roberts, J. Measuring Numerical Accuracy. PC Tech Journal, January 1988, Vol. 6:1, pp. 142-158

5. Dayhoff, Ruth. New Developments in MUMPS. MUMPS News, Vol 5:1, February 1988.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS None
2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NHRC Report No. 88-9	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Health Research Center 6b OFFICE SYMBOL (If applicable) Code 20	7a NAME OF MONITORING ORGANIZATION Commander, Naval Medical Command
6c ADDRESS (City, State, and ZIP Code) P. O. Box 85122 San Diego, CA 92138-9174	7b ADDRESS (City, State, and ZIP Code) Department of the Navy Washington, DC 20372
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Medical Research & Development Command 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Naval Medical Command National Capitol Region Bethesda, MD 20814-5044	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 63706N PROJECT NO: M0095 TASK NO: 005 WORK UNIT ACCESSION NO: 1053

11 TITLE (Include Security Classification)

e^x, LOGARITHMS, AND THE NORMAL DISTRIBUTION

12 PERSONAL AUTHOR(S)
HODGINS, Dallas R.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM _____ TO _____	1988 February 19	20

16 SUPPLEMENTARY NOTATION
Sub x

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			Normal distribution, logarithms, error function

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The occurrence of e, the natural logarithm base, in widely used probability density functions, necessitates having elementary computer algorithms to evaluate numbers N represented $N=e^x$. Programs are presented that find x given N, find N given x, and use these operations in developing a rational approximation for the normal (Gaussian) probability density function integral.

Accuracy of the order of three parts in ten million are assured in the logarithm routine. The exponential process is accurate to two parts in ten billion and the density function is in error by only one and one half units in the seventh decimal digit.

The MUMPS (Massachusetts General Hospital Utility Multi-Programming System) string functions $EXTRACT, $FIND and $LENGTH are shown to be efficient aids in examining numbers. MUMPS, numerical analysis, and compiler standards for number manipulation are briefly discussed.

(Continued on reverse)

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dallas R. Hodgins	22b TELEPHONE (Include Area Code) 619/553-9291 22c OFFICE SYMBOL Code 20

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

*U.S. Government Printing Office: 1988—607-047

UNCLASSIFIED

Item 19 (continued).

Fast, efficient numerical algorithms are realizable in the MUMPS environment. The string manipulation operators, in particular, allow concise and readable code. MUMPS is a vastly underrated language with respect to numerical analysis. It is appealing to one's intuition, logically compelling, and parsimonious.