## Calculation Title

Testing of Software Routine to Determine Deviate and Cumulative Probability: ModStandardNormal Version 1.0

### Revision History

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description of Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>REV 00</td>
<td>Initial Issue</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PURPOSE</td>
<td>3</td>
</tr>
<tr>
<td>2. METHOD</td>
<td>3</td>
</tr>
<tr>
<td>3. ASSUMPTIONS</td>
<td>4</td>
</tr>
<tr>
<td>4. USE OF COMPUTER SOFTWARE AND MODELS</td>
<td>4</td>
</tr>
<tr>
<td>5. CALCULATION</td>
<td>4</td>
</tr>
<tr>
<td>6. RESULTS</td>
<td>5</td>
</tr>
<tr>
<td>7. ATTACHMENTS</td>
<td>6</td>
</tr>
</tbody>
</table>
1. PURPOSE

The purpose of this calculation is to document that the software routine \textit{ModStandardNormal Version 1.0} which is a Visual Fortran 5.0 module, provides correct results for a normal distribution up to five significant figures (three significant figures at the function tails) for a specified range of input parameters. The software routine may be used for quality affecting work. Two types of output are generated in \textit{ModStandardNormal}: a deviate, \(x\), given a cumulative probability, \(p\), between 0 and 1; and a cumulative probability, \(p\), given a deviate, \(x\), between \(-8\) and 8. This calculation supports Performance Assessment, under Technical Product Development Plan, TDP-EBS-MD-000006 (Attachment I, DIRS 3) and is written in accordance with the AP-3.12Q \textit{Calculations} procedure (Attachment I, DIRS 4).

2. METHOD

The \textit{ModStandardNormal} routine performs the two functions stated above using three test driver programs. The first driver program, \textit{testfwdnom}, uses the cumulative probability function to numerically estimate the probability, \(p\), given the deviate \(x\). The procedure is referred to as the forward normal calculation. The second driver program, \textit{testinvnom}, uses the cumulative probability function to numerically estimate the deviate, \(x\), given the cumulative probability, \(p\) and the procedure is referred to as the inverse normal calculation. The third test driver program, \textit{testnormtable}, interpolates the deviate and cumulative probability from a table. The numerical functions and the table used can be found in Attachment II. A more detailed explanation on calculating the forward and inverse normals can be found in select journal articles (Attachment I, DIRS 1 and 2).

The \textit{ModStandardNormal} routine generates the normal distribution function by evaluating the error function as shown in Equation 1. Since the error function is only defined for deviate values between 0 and \(+\infty\), and since the normal distribution function is symmetrical about the y-axis, probability values for negative deviates are obtained by evaluating \(1-\text{erf}(x)\). To evaluate the integral numerically, a Chebyshev expansion, which is essentially a Fourier expansion, is used (Attachment I, DIRS 1).

\[
\text{erf}(x) = \frac{2}{\pi} \int_{0}^{x} e^{-t^2} dt
\]

(Eq. 1)

\textit{ModStandardNorm} uses the Chebyshev expansion to generate a normal table of 4900 equally sized increments of size 0.0001 for probabilities 0.01 < \(p\) < 0.50. Cumulative probability values of 0.50 < \(p\) < 1.0 are calculated by recognizing that the standard normal curve is symmetrical. Thus \(1-p = p\) for \(p\) between 0.50 < \(p\) < 1.0. For values within the range but not in the table, \textit{ModStandardNorm} uses linear interpolation to determine the deviate (or cumulative probability). For the small number of cumulative probabilities that fall outside the range 0.01 < \(p\) < 0.99, \textit{ModStandardNorm} uses the Chebyshev expansion to determine each deviate (or cumulative probability) value.
In order to document that *ModStandardNormal* provides correct results, the three driver programs used in *ModStandardNormal* are compared with the built-in cumulative probability functions in the MathCad software. Sixteen deviate values between −8 and 8 were calculated using both the built-in MathCad function `qnorm`, and the Fortran driver program, `testinvnorm`, and their absolute error was determined. Twenty-three cumulative probability values between $1 \times 10^{-15}$ and $9.99999999999999 \times 10^{-1}$ were also calculated using both the built-in MathCad function `pnorm`, and the Fortran driver program, `testfwdnorm`, and their absolute error determined. The absolute error from the output of the two MathCad functions were also evaluated in relation to the Fortran driver program, `testnomztable`, which interpolates the deviate and cumulative probability from a table.

### 3. ASSUMPTIONS

The following assumption was made in performing this calculation:

To test the consistency between the *ModStandardNormal* subroutine and the MathCad worksheets, it was assumed sufficient to have agreement up to five significant figures. Thus, an absolute error equal to or less than $1 \times 10^{-5}$ is taken to be a reasonable level of accuracy. However, for probabilities less than or equal to $1 \times 10^{-15}$ and greater than $9.99999999999999 \times 10^{-1}$, an absolute error equal to or less than $1 \times 10^{-3}$ is taken to be reasonable. Given that MathCad and the Fortran routine use slightly different polynomial expansions to numerically compute the cumulative distribution function, greater accuracy is not possible, especially at the tails of the distribution. This assumption is used throughout.

### 4. USE OF COMPUTER SOFTWARE AND MODELS

Appropriate industry standard software used in this calculation are MathCad Professional Version 8.02 and Excel 97 SR-2. This software was used to perform "hand calculation" verification of Visual Fortran 5.0 routines. Visual Fortran 5.0 is also industry standard software. Both software programs were executed on an IBM-compatible, DELL PowerEdge 2200 Workstation equipped with a Pentium II 266 MHz processor (CRWMS M&O tag 111593) in the Windows NT 4.0 operating system. Details of the MathCad worksheets used and the Fortran routines can be found in Attachment III (MathCad Worksheet) and Attachment II (Fortran Routines), respectively. A description of the governing formulas used in the *ModStandardNormal* routine was described in Section 2.

### 5. CALCULATION

A MathCad worksheet was written to duplicate the three Fortran driver programs used to run the *ModStandardNorm* subroutines. The driver program, `testfwdnorm` was emulated using the built-in MathCad function `pnorm`. The driver program, `testinvnorm` was emulated using the built-in MathCad function `qnorm`. The driver program `testnormtable` was emulated using both MathCad functions `pnorm` and `qnorm`. `Testnormtable` interpolates either the deviate or cumulative probability from a table. The Fortran driver programs and *ModStandardNormal* routines can be found in Attachment II and the MathCad worksheet is shown in Attachment III.
To test testfwdnorm, ModStandNorm and the testfwdnorm driver program were compiled and executed (a general-purpose module, moddefaultsize, is first compiled to define default byte size for reals, integers and logicals). Upon execution, the user is prompted to enter a deviate between −8 and 8. ModStandNorm then generates the corresponding cumulative probability value and prompts the user to enter another deviate between −8 and 8. To exit the program the user enters a value outside of the specified range. The input and output values are exported to the file, TestFwdNorm.out and the results are then copied into a column of an embedded Excel spreadsheet in the StandNorm MathCad worksheet. The same deviate values are used in the MathCad worksheet’s built-in function pnorm and the results saved to the preceding column of the embedded Excel sheet. The relative error between the two columns of numbers was then calculated.

The same procedure described above for the testfwdnorm driver and pnorm function is used for the testinvnorm driver and qnorm function, and also the testnormtable driver and pnorm/qnorm functions. The Fortran input and output values for testinvnorm were exported to the file TestInvNor.out, and the input and output values for testnormtable were exported to the file TestNormTable.out. The Fortran output files are shown in Attachment IV. Since the inputs used in this calculation are not considered data, they do not require a TBV/TBD tracking number.

6. RESULTS

All input and output information relevant to this calculation are included in Attachments III and IV. For brevity, only a select portion is reproduced in hardcopy form within this section. This calculation does not contain information or assumptions that need to be confirmed prior to the use of the results of this calculation.

Shown below, is the embedded Excel worksheet within MathCad for the results of the inverse normal calculations (given a probability, p, find the deviate, x).

The input probability is shown in column A and the output of the built-in MathCad function qnorm is shown in column B. The Fortran outputs for testnormtable and testinvnorm are shown in columns C and D, respectively. Column E shows the absolute relative error between the MathCad output and testnormtable, while column F shows the absolute error between the MathCad output and testinvnorm. The absolute error for all input values are below $1 \times 10^{-5}$, with the exceptions of probabilities less than or equal to $1 \times 10^{-15}$ and greater than $9.99999999999999\times10^{-1}$, in which case the absolute error values are below $1 \times 10^{-3}$. The same is true for the forward normal calculations (given a deviate x, find the cumulative probability p) and the results are found in Attachment III.
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cum Prob. (p)</td>
<td>MathCad (x)</td>
<td>NormTable (x)</td>
<td>Function (x)</td>
<td>Absolute Error</td>
<td>Absolute Error</td>
</tr>
<tr>
<td>1.0000E-15</td>
<td>-7.9414E+00</td>
<td>N/A</td>
<td>-7.9420E+00</td>
<td>N/A</td>
<td>5.9444E-04</td>
</tr>
<tr>
<td>2.0000E-03</td>
<td>-2.8782E+00</td>
<td>N/A</td>
<td>-2.8782E+00</td>
<td>N/A</td>
<td>4.9370E-08</td>
</tr>
<tr>
<td>4.0000E-03</td>
<td>-2.6521E+00</td>
<td>N/A</td>
<td>-2.6521E+00</td>
<td>N/A</td>
<td>4.5278E-08</td>
</tr>
<tr>
<td>8.0000E-03</td>
<td>-2.4089E+00</td>
<td>N/A</td>
<td>-2.4089E+00</td>
<td>N/A</td>
<td>4.1107E-08</td>
</tr>
<tr>
<td>1.0000E-02</td>
<td>-2.3263E+00</td>
<td>-2.3263E+00</td>
<td>-2.3263E+00</td>
<td>3.9805E-08</td>
<td>3.9805E-08</td>
</tr>
<tr>
<td>1.0050E-02</td>
<td>-2.3245E+00</td>
<td>-2.3245E+00</td>
<td>-2.3245E+00</td>
<td>4.0153E-06</td>
<td>3.9775E-08</td>
</tr>
<tr>
<td>1.0070E-02</td>
<td>-2.3237E+00</td>
<td>-2.3237E+00</td>
<td>-2.3237E+00</td>
<td>3.3622E-06</td>
<td>3.9764E-08</td>
</tr>
<tr>
<td>1.0090E-02</td>
<td>-2.3230E+00</td>
<td>-2.3230E+00</td>
<td>-2.3230E+00</td>
<td>1.4164E-06</td>
<td>3.9752E-08</td>
</tr>
<tr>
<td>2.0000E-02</td>
<td>-2.0537E+00</td>
<td>-2.0537E+00</td>
<td>-2.0537E+00</td>
<td>3.5281E-08</td>
<td>3.5281E-08</td>
</tr>
<tr>
<td>6.0000E-02</td>
<td>-1.5548E+00</td>
<td>-1.5548E+00</td>
<td>-1.5548E+00</td>
<td>2.6589E-08</td>
<td>2.6589E-08</td>
</tr>
<tr>
<td>1.0000E-01</td>
<td>-1.2816E+00</td>
<td>-1.2816E+00</td>
<td>-1.2816E+00</td>
<td>2.1930E-08</td>
<td>2.1930E-08</td>
</tr>
<tr>
<td>2.0000E-01</td>
<td>-8.4162E-01</td>
<td>-8.4162E-01</td>
<td>-8.4162E-01</td>
<td>1.4393E-08</td>
<td>1.4393E-08</td>
</tr>
<tr>
<td>4.0000E-01</td>
<td>-2.5335E-01</td>
<td>-2.5335E-01</td>
<td>-2.5335E-01</td>
<td>4.3358E-09</td>
<td>4.3359E-09</td>
</tr>
<tr>
<td>5.0000E-01</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>7.0000E-01</td>
<td>5.2440E-01</td>
<td>5.2440E-01</td>
<td>5.2440E-01</td>
<td>8.9747E-09</td>
<td>8.9747E-09</td>
</tr>
<tr>
<td>9.999999999999990E01</td>
<td>7.94100472</td>
<td>N/A</td>
<td>7.94E+00</td>
<td>N/A</td>
<td>4.3963E-04</td>
</tr>
</tbody>
</table>

### 7. ATTACHMENTS

<table>
<thead>
<tr>
<th>Attachment Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>DIRS</td>
</tr>
<tr>
<td>II</td>
<td>Fortran Routines</td>
</tr>
<tr>
<td>III</td>
<td>MathCad Worksheet For Generating Random Numbers</td>
</tr>
<tr>
<td>IV</td>
<td>Fortran Output Files</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>2. Amos, D.L. and Daniel, S.L. 1972. “CDC 6600 Codes for the Error Function, Cumulative Normal and Related Functions.” SC-DR-72-0918. Albuquerque, New Mexico. TIC: 244294</td>
<td>Entire</td>
</tr>
<tr>
<td>Input Document</td>
<td>Change</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
</tr>
<tr>
<td>CRWMS M&amp;O 1999. <em>Testing of Software Routines Applicable to EBS Performance Assessment.</em> TDP-EBS-MD-000006 REV 00. Las Vegas, Nevada: CRWMS M&amp;O. ACC: MOL.19990811.0073</td>
<td>Entire</td>
</tr>
<tr>
<td>AP-3.12Q, Rev. 0. <em>Calculations.</em> Washington, D.C.: U.S. Department of Energy, Office of Civilian Radioactive Waste management. ACC: MOL.19990702.0312</td>
<td>Entire</td>
</tr>
</tbody>
</table>
### OFFICE OF CIVILIAN RADIOACTIVE WASTE MANAGEMENT

#### DOCUMENT INPUT REFERENCE SHEET

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Change:**
- Testing of Software Routine to Determine Deviate and Cumulative Probability:
- ModStandardNormal Version 1.0

**Input Document Description:**
- **AP-3.12Q, Rev. 0. Calculations.**
# ATTACHMENT II – FORTRAN ROUTINES

## CONTENTS

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MODULE: MODDEFAULTSIZE</td>
<td>II-2</td>
</tr>
<tr>
<td>2. MODSTANDARDNORMAL</td>
<td>II-2</td>
</tr>
<tr>
<td>3. TESTFWDNORM</td>
<td>II-16</td>
</tr>
<tr>
<td>4. TESTINVNOR</td>
<td>II-17</td>
</tr>
<tr>
<td>5. TESTNORMTABLE</td>
<td>II-18</td>
</tr>
</tbody>
</table>
1. MODULE: MODDEFAULTSIZE

```
MODULE moddefaultsize
 c This module is general purpose to define default byte size for reals, c integers, and logicals. Consistent use throughout a program will c ensure argument list conformity, etc.
 c implicit none
 integer(l), PARAMETER :: RKind = 8
 integer(l), PARAMETER :: IKind = 4
 integer(l), PARAMETER :: LKind = 1

END MODULE moddefaultsize
```

2. MODSTANDARDNORMAL

```
MODULE modstandardnormal
 c The purpose of this module is to provide p given x or x given p in c the expression p = pr(z < x) where pr denotes probability and z c indicates the standard normal distribution. Which routines to call c depends on the number of calls that will be made.
 c 1. If fewer than about 5000 calls are to be made, the most efficient c way is: use InvNor(given p, find x) and FwdNorm(given x, find p).
 c 2. If more than about 5000 calls are to be made, the most efficient c way is to use StdNorDev(given p, find x) and StdNorProb (given x, c find p).
 c For the latter, tables of deviates are generated in a manner that the c associated cumulative probabilities can be ascertained from their c indexes. Then when StdNorDev is called with a probability argument, c deviate x is interpolated from the values in the tables. Similarly, c when StdNorProb is called with a deviate argument, probability p is c interpolated from the table indexes. To generate the table, array v c is allocated. If there is not sufficient memory for the allocation, c routines InvNor and FwdNorm will be used.
 c USE moddefaultsize
 Public  StdNorDev, StdNorProb, InvNor, FwdNorm
 Private MakeNormTable, InvERFC, BiSearchOnv, Erf, &
 &       ErfAbsLT2, Erf2To4, Erf4To6, ErfChebyPoly
 c 
 real(RKind), dimension(:,), allocatable, save, private :: v
 real(RKind), save, private :: MinProb4v, MaxProb4v
 logical(LKind), save, private :: TableMade
```
data TableMade / .false. /

CONTAINS  !MakeNormTable, StdNorDev, StdNorProb
         !InvNor, InvERFC, BiSearchOnv, FwdNorm, Erf
         !ErfAbsLT2, Erf2To4, Erf4To6, ErfChebyPoly

SUBROUTINE MakeNormTable

Make a normal table for the interior of the standard normal. The
'interior' for table v accounts for probabilities p such that
0.01 < p < 0.5 at 4900 equal increments of size 0.0001. Value v(i)
is the standard normal deviate at the cumulative probability
(i + 100) / 10,000 for indexes i such that 0 <= i <= 4900.
Input : (none)
Output: (none through the argument list, although v, MinProb4v,
MaxProb4v, and TableMade are defined at module-level)
Local : vDim, i, AllocStat, pr, ProbStep4v

Local variables

integer(1Kind) :: vDim, i, AllocStat
real(RKind)   :: pr, ProbStep4v

vDim     = 4900
MinProb4v = 0.01_RKind
MaxProb4v = 0.99_RKind
ProbStep4v = 0.0001_RKind

If insufficient memory to store v, return before setting TableMade
to true.
allocate(v(0: vDim), stat = AllocStat)
if(AllocStat .ne. 0) RETURN

pr    = MinProb4v
v(0)  = InvNor(pr)
v(vDim) = 0.0_RKind
do i = 1, vDim - 1
    pr = pr + ProbStep4v
    v(i) = InvNor(pr)
end do

TableMade = .true.
RETURN
END SUBROUTINE MakeNormTable

C******************************************************************************
C
real(RKind) function StdNorDev(parg)
C
Interpolation routine for a NORMAL lookup table. Argument parg is a
cumulative probability. The corresponding standard normal deviate is
found and returned as the function value. The value of v(i) is the
standard normal deviate at cumulative probability (i + 100) / 10,000
where 0 <= i <= 4900. For probabilities outside 0.01 to 0.99,
function InvNor is used to compute the deviate.
C
Input:  parg
Output: (function value)
Local:  KI, p, x, y, Rat
C
Arguments
C
real(RKind) :: parg
C
Local variables
C
integer(1Kind) :: KI
real(RKind) :: p, x, y, Rat
C
If the normal table is not yet made, do so now.
C
if(.not. TableMade) then
  CALL MakeNormTable
if(.not. TableMade) then     !Insufficient memory
  x = InvNor(p)
  StdNorDev = x
  RETURN
end if
end if
C
If the probability is on the tail, find the deviate directly using
function InvNor. Otherwise, linearly interpolate from the table,
taking advantage of symmetry.
C
p = parg
if(p < MinProb4v or. p > MaxProb4v) then
  x    = InvNor(p)
  StdNorDev = x
else if(p .gt. 0.5*RKind) then
  p   = 10000.0*RKind * (1.0*RKind - p)
  Rat = p - int(p)
  KI  = int(p - 100.0*RKind)
  y   = Rat * (v(KI + 1) - v(KI)) + v(KI)
  StdNorDev = -y
else if(p .lt. 0.5*RKind) then
  p   = 10000.0*RKind * p
  Rat = p - int(p)
  KI  = int(p - 100.0*RKind)
  y   = Rat * (v(KI + 1) - v(KI)) + v(KI)
  StdNorDev = y
else
  StdNorDev = 0.0*RKind
end if
RETURN
END FUNCTION StdNorDev

real(RKind) function StdNorProb(xarg)

Argument xarg is a deviate from the standard normal distribution.
Find the cumulative probability and return as the function value.
The value of v(i) is the standard normal deviate at cumulative probability (i + 100) / 10,000, where 0 <= i <= 4900. For probability outside 0.01 to 0.99, function FwdNorm is used.
Input : xarg
Output: (function value)
Local : CompProb, KI, FirstNdx, LastNdx, p, Rat, x
Arguments
real(RKind) :: xarg
Local variables
logical(LKind) :: CompProb
integer(IKind) :: KI, FirstNdx, LastNdx
real(RKind) :: p, Rat, x

If the normal tables are not yet made, do so now.
if (.not. TableMade) then
    CALL MakeNormTable
    if (.not. TableMade) then ! Insufficient memory
        if (abs(xarg) .gt. 0.0_RKind) then
            p = FwdNorm(xarg)
            StdNorProb = p
        else
            StdNorProb = 0.5_RKind
        end if
        RETURN
    end if
end if

For argument xarg, there are four tests of its value:
1. If xarg is such that the resulting probability p will be outside
the range from 1E-15 to 0.999999999999999, then either p = 0 or 1
will be returned, as a limiting value.
2. ElseIf xarg is such that p will be inside the range of (1) but
outside the range from 0.01 to 0.99, then function FwdNorm is
called to find the probability.
3. ElseIf xarg is such that p will be inside the range from 0.01 to
0.99 (but xarg is not zero), then find the stored entries in v
that surround xarg and linearly interpolate. The search for the
cvalues that surround xarg uses bisection (BiSearchOnv).
4. Else xarg must be exact zero so that p = 0.5.

x = xarg
if (abs(x) .gt. 7.9414_RKind) then ! Case 1
    if (x .gt. 0.0_RKind) then
        StdNorProb = 1.0_RKind
    else
        StdNorProb = 0.0_RKind
    end if
elseif (abs(x) .ge. abs(v(0))) then ! Case 2
    p = FwdNorm(x)
    StdNorProb = p
else if (abs(x) .gt. 0.0_RKind) then ! Case 3
    if (x .gt. 0.0_RKind) then
        CompProb = .true. ! Find complementary probability first
        x = -x
    else
        CompProb = .false.
    end if
    FirstNdx = 0
LastNdx = 4900
KI = BiSearchOnv(x, FirstNdx, LastNdx)
Rat = (x - v(KI - 1)) / (v(KI) - v(KI - 1))
p = ((KI - 1) + 100.0_RKind + Rat) * 0.0001_RKind
if(CompProb) p = 1.0_RKind - p
StdNorProb = p
else
    ! Case 4
    StdNorProb = 0.5_RKind
end if
RETURN
END FUNCTION StdNorProb

real(RKind) FUNCTION InvNor(p)

Given cumulative probability p, find deviate x from the standard
c normal distribution such that p = pr(z < x). Use the numerical
c approximation to the complementary error function found in function
c InvERFC. Ensure the argument for InvERFC is valid.
Input: p
Output: function value of InvNor
Local : sqrt2
Argument
real(RKind) :: p
Local variable
real(RKind) :: sqrt2
save sqrt2
data sqrt2 / 1.4142135623731 /

Although function InvErfc is valid for arguments less than 1E-15 (and
greater than 6.63967719958073E-36), restrict the valid lower bound on
probability to 1E-15 to maintain symmetry with the largest possible
c probability, ie, 1 - 1E-15.
if (p .lt. 1.0E-15) then
    InvNor = -7.942_RKind
else if (p .lt. 0.5_RKind) then
    InvNor = -sqrt2 * InvERFC(2.0_RKind * p)
else if (p .gt. 0.999999999999999_RKind) then
    InvNor = 7.942_RKind
else
    InvNor = InvNor(p)
end if
RETURN
END FUNCTION InvNor

else if (p.gt.0.5-RKind) then
  InvNor = sqrt2 * InvERFC(2.0_RKind * (1.0_RKind - p))
else
  InvNor = 0.0_RKind
end if
RETURN
END FUNCTION InvNor
DATA(A3(I), I=1,22) / 1.10642888011036D+01, 4.34299147561447D+00,
& -2.33781774969295D-02, 4.23345215362947D-03, 8.68757084192089D-06,
& -5.98261113270881D-04, 4.50490139240298D-04, -2.54858131942102D-04,
& 1.27824189261340D-04, -5.97873878043957D-05, 2.66474012012582D-05,
& -1.14381836209267D-05, 4.75393030377615D-06, -1.91759589929610D-06,
& 7.50806465594834D-07, -2.84791180387123D-07, 1.04187791696225D-07,
& -3.64567243689145D-08, 1.20129296139030D-08, -3.61030126779729D-09,
& 9.12356140081759D-10, -1.36851363400914D-10

if (Y.ge. 0.5-RKind) then
  J = 1 - IKind
  D = 1.0_RKind - Y
  D = D + D
else if(Y.ge. 0.1_RKind) then
  J = 2 - IKind
  D = 5.0_RKind * Y - 1.5_RKind
else
  J = 3 - IKind
  W = SQRT(-log(Y))
  D = C1 * W - C2
endif

TD = D + D
VNP1 = 0.0_RKind
VN = 0.0_RKind
do K = 22 - IKind, 2 - IKind, -1 - IKind
  TEMP = VN
  VN = TD * VN - VNP1 + A(K, J)
  VNP1 = TEMP
end do
VN = D * VN - VNP1 + 0.5_RKind * A(1, J)
if (J .eq. 1 - IKind) VN = D * VN
InvERFC = VN
RETURN
END FUNCTION InvERFC

******************************************************************************

integer(IKind) FUNCTION BiSearchOnv(R, FirstNdx, LastNdx)

CAL-EBS-MD-000004 REV 00
Attachment II
II-9 October 1999
Use bisection to find the minimum value that exceeds \( R \) in array \( v \) between indexes \( \text{FirstNdx} \) and \( \text{LastNdx} \), and return its index as the function value. The logic assumes that:

1. The entries of array \( v \) are increasing
2. It is known that \( R \) is between the values given by the first and last indexes
3. \( \text{FirstNdx} < \text{LastNdx} \).

Input: \( R \), \( \text{FirstNdx} \), \( \text{LastNdx} \), (module level array \( v \))
Output: (function value)
Local: \( \text{StartNdx} \), \( \text{StopNdx} \), \( i \)

Arguments

\[
\text{real}(\text{RKind}) :: R \\
\text{integer}(\text{IKind}) :: \text{FirstNdx}, \text{LastNdx}
\]

Local variables

\[
\text{integer}(\text{IKind}) :: \text{StartNdx}, \text{StopNdx}, i
\]

\[
\text{StartNdx} = \text{FirstNdx} \\
\text{StopNdx} = \text{LastNdx}
\]

\[
\text{Do While} (\text{StartNdx} + 1 \_\text{IKind} .lt. \text{StopNdx}) \\
i = \text{StartNdx} + \text{Int}((\text{StopNdx} - \text{StartNdx}) / 2 \_\text{IKind}) \\
\text{If}(R .le. v(i)) \text{ then} \\
\text{StopNdx} = i \\
\text{Else} \\
\text{StartNdx} = i \\
\text{End If}
\]

BiSearchOnv = \text{StopNdx}
RETURN
END FUNCTION BiSearchOnv

Numerically approximate the cumulative probability of being less than deviate \( Z \) for the standard normal distribution, and return as the function value. Use the (complementary) error function in \( \text{Erf} \).

Input: \( Z \)
Output: function value
Local: \( \text{ErfOpt} \), \( \text{Root2} \), \( \text{TwoRoot2} \), \( \text{SixRoot2} \)
**Arguments**

- real(RKind) :: Z

**Local variables**

- logical(LKind) :: ErfOpt
- real(RKind) :: Root2, TwoRoot2, SixRoot2
- DATA Root2, TwoRoot2, SixRoot2 / 1.414213562373095, 2.828427124746191, 8.485281374238572 /

**Argument is tested for which interval it falls trying to avoid excessive loss of significance.**

if(Z.gt.SixRoot2) then
  FwdNorm = 1.0_RKind
else if(Z.lt.-SixRoot2) then
  FwdNorm = 0.0_RKind
else if(Z.gt.TwoRoot2) then
  ErfOpt = .false.
  FwdNorm = 1.0_RKind - 0.5_RKind * erf(ErfOpt, Z / Root2)
else if(Z.lt.-TwoRoot2) then
  ErfOpt = .false.
  FwdNorm = 0.5_RKind * erf(ErfOpt, -Z / Root2)
else
  ErfOpt = .true.
  FwdNorm = 0.5_RKind + 0.5_RKind * erf(ErfOpt, Z / Root2)
end if
RETURN
END FUNCTION FwdNorm

**END FUNCTION FwdNorm**

**FUNCTION Erf(ErfOpt, y)**

- Numerically approximate the (complementary) error function at Y using Chebyshev expansions on one of 3 intervals and return as the function value. Algorithm due to Amos and Daniel.
- If ErfOpt true, compute Erf(y), else compute Erfc(y) = 1 - Erf(y)
- Input : ErfOpt, y
- Output: function value
- Local : x, ANS, XLIM

**Arguments**
logical(LKind) :: ErfOpt
real(RKind) :: y

c Local variables
c
real(RKind) :: x, ANS, XLIM
DATA XLIM / 25.8408528684382 /
c
Argument out-of-bounds testing.
c
if(ErfOpt) then
  if(y .le. -6.0_RKind) then
    Erf = -1.0_RKind
    RETURN
  else if(y .ge. 6.0_RKind) then
    Erf = 1.0_RKind
    RETURN
  end if
else
  if(y .le. -6.0_RKind) then
    Erf = 2.0_RKind
    RETURN
  else if(y .ge. XLIM) then
    Erf = 0.0_RKind
    RETURN
  end if
end if

c Arguments must be positive. If original negative, then answers are
adjusted afterwards.
c
x = abs(y)
if(x .lt. 2.0_RKind) then
  ANS = ErfAbsLT2(y)
  if(.not. ErfOpt) then
    ANS = 1.0_RKind - ANS
  end if
  Erf = ANS
  RETURN
else if(x .lt. 4.0_RKind) then
  ANS = Erf2To4(x)
else
  ANS = Erf4To6(x)
end if

c Here the absolute argument is between 2 and the upper limit.
if(ErfOpt) then
  if(y .gt. 0.0_RKind) then
    ANS = 1.0_RKind - ANS
  else
    ANS = ANS - 1.0_RKind
  end if
else
  if(y .lt. 0.0_RKind) then
    ANS = 2.0_RKind - ANS
  end if
end if
Erf = ANS
RETURN
END FUNCTION Erf

real(RKind) FUNCTION ErfAbsLT2(X)

* Numerically approximate the error function at X using Chebyshev
* expansions for the interval -2 < X < 2 and return result as the
* function value.
* Input : X
* Output: function value
* Local : ANS, B1, B2, Z, A, N, I
* Argument
* Local variables

real(RKind) :: ANS, B1, B2, Z
real(RKind), save :: A(31)
integer(IKind) :: I, N
DATA N / 31/
DATA(A(I), I = 1, 31) /
& 2.96622112816961D+0, 0.0,-6.02142146773189D-1, 0.0,
& 1.37989661379662D-1, 0.0,-2.78325425294437D-2, 0.0,
& 4.84159904486783D-3, 0.0,-7.31727937169453D-4, 0.0,
& 9.7241968637174D-5, 0.0,-1.14985131161804D-5, 0.0,
& 1.22264871646933D-6, 0.0,-1.17982030973170D-7, 0.0,
& 1.04140177691278D-8, 0.0,-8.46595329454225D-10, 0.0,
& 6.37620443498960D-11, 0.0,-4.47177281962215D-12, 0.0,
& 2.93540222982101D-13, 0.0,-1.83283038964141D-14 /
Z = X / 2.0_RKind
CALL ErfChebyPoly(N, A, Z, B1, B2)
ANS = Z * (Z * B1 - B2 + A(1) / 2.0_RKind)
ErfAbsLT2 = ABS(ANS)
RETURN
END FUNCTION ErfAbsLT2

**********************************************************************************

real(RKind) FUNCTION Erf2To4(X)

Numerically approximate the complementary error function at X using
Chebyshev expansions for the interval 2 < X < 4 and return as the
function value.
Input: X
Output: function value
Local: ANS, B1, B2, Z, A, N, I
Argument
real(RKind) :: X

Local variables
real(RKind) :: ANS, B1, B2, Z
real(RKind), save :: A(16)
integer(1Kind) :: N, I

DATA N / 16 /
DATA(A(I), I = 1, 16) /
& 1.06663088531993D+0, 1.78876062094436D-2, -3.80175293809401D-3,
& 6.97111435023601D-4, -1.16368846063892D-4, 1.813675932619D-5,
& -2.67719939785138D-6, 3.77701329909996D-7, -5.12491142501402D-8,
& 6.71870395763107D-9, -8.54019646112644D-10, 1.05544302186899D-10,
& -1.2710899000124D-11, 1.49441348185064D-12,-1.71382907865335D-13,
& 2.08899564313469D-14 /

Z = X - 3.0_RKind
CALL ErfChebyPoly(N, A, Z, B1, B2)
ANS = Z * B1 - B2 + A(1) / 2.0_RKind
ANS = EXP(-X * X) * ANS / X
Erf2To4 = ANS
RETURN
END FUNCTION Erf2To4

CAL-EBS-MD-000004 REV 00
Attachment II
II-14

October 1999
**FUNCTION Erf4To6(X)**

Numerically approximate the complementary error function at X using Chebyshev expansions for the interval $4 < X < 6$ and return as the function value.

**Input**: X

**Output**: function value

**Local**:

- ANS, B1, B2, Z, RTPI, A, N, I

**Local variables**:

- real(RKind) :: X

- real(RKind) :: ANS, B1, B2, Z

- real(RKind), save :: A(27), RTPI

- integer(IKind) :: N, I

**DATA**

N / 27 /
DATA RTPI / 1.7724538509552 /
DATA(A(I), I = 1, 27) /
& 1.97070527225754, 0.0, -1.43397402717750D-2, 0.0,
& 2.97361692202619D-4, 0.0, -9.80351604336237D-6, 0.0,
& 4.3313342034728D-7, 0.0, -2.362150026241D-8, 0.0,
& 1.51549676581D-9, 0.0, -1.1084939856D-10, 0.0,
& 9.04259014D-12, 0.0, -8.0947054D-13, 0.0,
& 7.853856D-14, 0.0, -8.17918D-15, 0.0,
& 9.0715D-16, 0.0, -1.0646D-16 /

**Z** = 4.0_RKind / X
CALL ErfChebyPoly(N, A, Z, B1, B2)
ANS = Z * B1 - B2 + A(1) / 2.0_RKind
ANS = (EXP(-X * X) / (X * RTPI)) * ANS
Erf4To6 = ANS
RETURN
END FUNCTION Erf4To6

**SUBROUTINE ErfChebyPoly(N, A, Z, B1, B2)**

Evaluate a Chebyshev polynomial at Z using the N coefficients in A.

Return adjacent terms in B1 and B2.
c Input: N, A, Z
c Output: B1, B2
c Local: I, TZ, S
c
arguments
real(RKind) :: A(*), Z, B1, B2
integer(IKind) :: N
c
local variables
integer(IKind) :: I
real(RKind) :: TZ, S

TZ = Z + Z
B1 = 0.0_RKind
B2 = 0.0_RKind
DO I = N, 2_IKind, -1_IKind
   S = B1
   B1 = TZ * B1 - B2 + A(I)
   B2 = S
end do
RETURN
END SUBROUTINE ErfChebyPoly

END MODULE modstandardnormal

3. TESTFWDNORM

PROGRAM testfwdnorm
  c
  c This program is designed for testing of the numerical approximation
c to the forward normal. The test asks for a deviate and the code
c responds with the cumulative probability. Probabilities will be
c written to screen, and optionally to file.
c
  Use modstandardnormal
  logical(LKind) :: WriteToFile
  real(RKind) :: p, z

  write(*, 9000)
9000 format(’Save results to file (T/F)? ’)
  read(*, *) WriteToFile
  if(WriteToFile) then
    open(10, file = ’TestFwdNorm.out’)
    write(10, 9005)
  end if
END PROGRAM testfwdnorm
9005  format(10x,'z','22x','p')
end if

9010  format('Respond with a deviate outside (-8, 8) to stop.')
z = 0.0

9020  format('Enter deviate (-8, 8).')
read(*, *) z
if(z .le. -8.0 .or. z .ge. 8.0) then
  exit
endif

p = FwdNorm(z)
write(*, 9050) z, p

9050  format(1pe21.14,2x,1pe21.14)
if(WriteToFile) then
  write(10, 9050) z, p
end if
end do
if(WriteToFile) then
  close(10)
end if

END PROGRAM testfwdnorm

4. TESTINVNOR

PROGRAM testinvnor
  logical(LKind) :: WriteToFile
  real(RKind) :: p, z
  read(*, *) WriteToFile
  if(WriteToFile) then
    open(10, file = 'TestInvNor.out')
  end if

CAL-EBS-MD-000004 REV 00
Attachment II
II- 17 October 1999
5. TESTNORMTABLE

PROGRAM testnormtable
  c
  c This program is designed for testing of the interpolation of the
  c standard normal table. The test first asks for a probability and the
  c code responds with two deviates, 1 from InvNor and 1 from the table.
  c It then asks for a deviate and code responds with 2 probabilities,
  c 1 from FwdNorm and 1 from the table.
  c Deviates and probabilities will be written to screen, and
  c optionally to file.
  c
  Use modstandardnormal
  logical(LKind) :: WriteToFile
  real(RKind) :: p, p1, p2, z, z1, z2
  c
  write(*, 9000)
9000 format('Save results to file (T/F)?
   read(*, *) WriteToFile
   if(WriteToFile) then
       open(10, file = 'TestNormTable.out')
       write(10, 9005)
   end if
   c
   write(*, 9010)
9010 format('Respond with probability outside (0.01, 0.99) to stop.
   p = 0.5
   c
   do while (p .ge. 0.01 .and. p .le. 0.99)
       write(*, 9020)
9020 format('Enter probability (0, 1)
   read(*, *) p
   if(p .lt. 0.01 .or. p .gt. 0.99) then
       exit
   endif
   c
   z1 = InvNor(p)
   z2 = StdNorDev(p)
   write(*, 9050) p, z1, z2
9050 format(1pe21.14,2x,1pe21.14,2x,1pe21.14)
   if(WriteToFile) then
       write(10, 9050) p, z1, z2
   end if
   end do
   c
   if(WriteToFile) then
       write(10, 9105)
9105 format(10x,'z',17x,'FwdNorm-p',16x,'Table-p')
   end if
   c
   write(*, 9110)
9110 format('Respond with a deviate outside (-2.33, 2.33) to stop.
   z = 0.0
   c
   do while (z .gt. -2.33 .and. z .lt. 2.33)
       write(*, 9120)
9120 format('Enter deviate (-2.33, 2.33)
   read(*, *) z
   if(z .le. -2.33 .or. z .ge. 2.33) then
       exit
   endif
   c
p1 = FwdNorm(z)
p2 = StdNorProb(z)
write(*, 9150) z, p1, p2
9150 format(1pe21.14,2x,1pe21.14,2x,1pe21.14)
if(WriteToFile) then
  write(10, 9150) z, p1, p2
end if
end do
if(WriteToFile) then
  close(10)
end if

END PROGRAM testnormtable
ATTACHMENT III

In this MathCad spreadsheet, the built in MathCad functions are used to evaluate the cumulative probability, p, for a deviate, x, and to evaluate the deviate, x, for a cumulative probability, p.

---

<table>
<thead>
<tr>
<th>Input</th>
<th>pnorm(x, 0, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7.9</td>
<td>1.39451714665928·10⁻¹⁵</td>
</tr>
<tr>
<td>-7.89</td>
<td>1.51093268139137·10⁻¹⁵</td>
</tr>
<tr>
<td>-6.05</td>
<td>7.24229170513766·10⁻¹⁰</td>
</tr>
<tr>
<td>-6</td>
<td>9.86587645037701·10⁻¹⁰</td>
</tr>
<tr>
<td>-5</td>
<td>2.86651571879195·10⁻⁷</td>
</tr>
<tr>
<td>-4</td>
<td>3.16712418331200·10⁻⁵</td>
</tr>
<tr>
<td>-3</td>
<td>1.34989803163010·10⁻³</td>
</tr>
<tr>
<td>-2</td>
<td>9.87667984102212·10⁻³</td>
</tr>
<tr>
<td>-1</td>
<td>1.00359801002741·10⁻²</td>
</tr>
<tr>
<td>0</td>
<td>2.27501319481792·10⁻²</td>
</tr>
<tr>
<td>1</td>
<td>1.58655253931457·10⁻¹</td>
</tr>
<tr>
<td>2</td>
<td>5.00000000000000·10⁻¹</td>
</tr>
<tr>
<td>2.325</td>
<td>8.41344746068543·10⁻¹</td>
</tr>
<tr>
<td>2.331</td>
<td>9.77249868051821·10⁻¹</td>
</tr>
<tr>
<td>3</td>
<td>9.89964019899726·10⁻¹</td>
</tr>
<tr>
<td>4</td>
<td>9.90123320158978·10⁻¹</td>
</tr>
<tr>
<td>5</td>
<td>9.98650101968370·10⁻¹</td>
</tr>
<tr>
<td>6.05</td>
<td>9.99968328758167·10⁻¹</td>
</tr>
<tr>
<td>7</td>
<td>9.9999713348428·10⁻¹</td>
</tr>
<tr>
<td>7.89</td>
<td>9.999999275771·10⁻¹</td>
</tr>
<tr>
<td>7.9</td>
<td>9.999999998720·10⁻¹</td>
</tr>
<tr>
<td>9</td>
<td>9.999999999998·10⁻¹</td>
</tr>
<tr>
<td></td>
<td>9.999999999999·10⁻¹</td>
</tr>
</tbody>
</table>

Given the deviate x, find the cumulative probability, p. This is called the Forward Normal.
Excel Sheet (below) shows output from MathCad, and the two different Fortran algorithms (NormTable, Function and the error between the two and MathCad).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deviate (x)</td>
<td>MathCad (p)</td>
<td>NormTable (p)</td>
<td>Function (p)</td>
<td>Absolute Error</td>
<td>Absolute Error</td>
</tr>
<tr>
<td>-7.9</td>
<td>1.3945E-15</td>
<td>N/A</td>
<td>1.3945E-15</td>
<td>N/A</td>
<td>1.5556E-21</td>
</tr>
<tr>
<td>-7.89</td>
<td>1.5109E-15</td>
<td>N/A</td>
<td>1.5109E-15</td>
<td>N/A</td>
<td>1.6814E-21</td>
</tr>
<tr>
<td>-6.05</td>
<td>7.2423E-10</td>
<td>N/A</td>
<td>7.2423E-10</td>
<td>N/A</td>
<td>4.8778E-16</td>
</tr>
<tr>
<td>-6</td>
<td>9.8659E-10</td>
<td>N/A</td>
<td>9.8659E-10</td>
<td>N/A</td>
<td>6.5430E-16</td>
</tr>
<tr>
<td>-5</td>
<td>2.8665E-07</td>
<td>N/A</td>
<td>2.8665E-07</td>
<td>N/A</td>
<td>1.2722E-13</td>
</tr>
<tr>
<td>-4</td>
<td>3.1671E-05</td>
<td>N/A</td>
<td>3.1671E-05</td>
<td>N/A</td>
<td>9.1616E-12</td>
</tr>
<tr>
<td>-3</td>
<td>1.3499E-03</td>
<td>N/A</td>
<td>1.3499E-03</td>
<td>N/A</td>
<td>2.2754E-10</td>
</tr>
<tr>
<td>-2.331</td>
<td>9.8767E-03</td>
<td>N/A</td>
<td>9.8767E-03</td>
<td>N/A</td>
<td>1.0518E-09</td>
</tr>
<tr>
<td>-2.325</td>
<td>1.0036E-02</td>
<td>1.0036E-02</td>
<td>1.0036E-02</td>
<td>9.9036E-08</td>
<td>1.0638E-09</td>
</tr>
<tr>
<td>-2</td>
<td>2.2750E-02</td>
<td>2.2750E-02</td>
<td>2.2750E-02</td>
<td>4.4455E-08</td>
<td>1.8480E-09</td>
</tr>
<tr>
<td>-1</td>
<td>1.5866E-01</td>
<td>1.5866E-01</td>
<td>1.5866E-01</td>
<td>9.6537E-10</td>
<td>4.1412E-09</td>
</tr>
<tr>
<td>0</td>
<td>5.00E-01</td>
<td>5.0000E-01</td>
<td>5.0000E-01</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>1</td>
<td>8.41E-01</td>
<td>8.4134E-01</td>
<td>8.4134E-01</td>
<td>9.6537E-10</td>
<td>4.1412E-09</td>
</tr>
<tr>
<td>2</td>
<td>9.77E-01</td>
<td>9.7725E-01</td>
<td>9.7725E-01</td>
<td>4.4455E-08</td>
<td>1.8480E-09</td>
</tr>
<tr>
<td>2.331</td>
<td>9.90E-01</td>
<td>N/A</td>
<td>9.9012E-01</td>
<td>N/A</td>
<td>1.0518E-09</td>
</tr>
<tr>
<td>3</td>
<td>9.99E-01</td>
<td>N/A</td>
<td>9.9865E-01</td>
<td>N/A</td>
<td>2.2754E-10</td>
</tr>
<tr>
<td>4</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>9.1630E-12</td>
</tr>
<tr>
<td>5</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.2701E-13</td>
</tr>
<tr>
<td>6.05</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>9.9920E-16</td>
</tr>
<tr>
<td>7</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>7.89</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>7.9</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>1.00E+00</td>
<td>N/A</td>
<td>9.9920E-16</td>
</tr>
</tbody>
</table>
Given the cumulative probability $p$, find the deviate $x$
This is called the Inverse Normal.

\[
p := \begin{bmatrix}
  0.002 \\
  0.004 \\
  0.008 \\
  0.010 \\
  0.01005 \\
  0.01007 \\
  0.01009 \\
  0.02 \\
  0.06 \\
  0.1 \\
  0.2 \\
  0.4 \\
  0.5 \\
  0.7
\end{bmatrix}
\]

\[
qnorm(p, 0, 1) = \begin{bmatrix}
  -7.94140555777135 \\
  -2.87816173909549 \\
  -2.6520698079022 \\
  -2.40891554581546 \\
  -2.32634787404084 \\
  -2.32447593301054 \\
  -2.32372943172158 \\
  -2.3229842231219 \\
  -2.05374891063183 \\
  -1.55477359459685 \\
  -1.2815515655446 \\
  -0.841621233572915 \\
  -0.253347103135799 \\
  0 \\
  0.52440051270804 \\
  7.94100472087549
\end{bmatrix}
\]
Excel Sheet shows output from MathCad, and the two different Fortran algorithms (NormTable and and the error between the two.

<table>
<thead>
<tr>
<th>Cum Prob. (p)</th>
<th>MathCad (x)</th>
<th>NormTable (x)</th>
<th>Function (x)</th>
<th>Absolute Error (B-C)</th>
<th>Absolute Error (B-D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000E-15</td>
<td>-7.9414E+00</td>
<td>N/A</td>
<td>-7.9420E+00</td>
<td>N/A</td>
<td>5.9444E-04</td>
</tr>
<tr>
<td>2.0000E-03</td>
<td>-2.8782E+00</td>
<td>N/A</td>
<td>-2.8782E+00</td>
<td>N/A</td>
<td>4.9370E-08</td>
</tr>
<tr>
<td>4.0000E-03</td>
<td>-2.6521E+00</td>
<td>N/A</td>
<td>-2.6521E+00</td>
<td>N/A</td>
<td>4.5278E-08</td>
</tr>
<tr>
<td>8.0000E-03</td>
<td>-2.4089E+00</td>
<td>N/A</td>
<td>-2.4089E+00</td>
<td>N/A</td>
<td>4.1107E-08</td>
</tr>
<tr>
<td>1.0000E-02</td>
<td>-2.3263E+00</td>
<td>-2.3263E+00</td>
<td>-2.3263E+00</td>
<td>3.9805E-08</td>
<td>3.9805E-08</td>
</tr>
<tr>
<td>1.0050E-02</td>
<td>-2.3245E+00</td>
<td>-2.3245E+00</td>
<td>-2.3245E+00</td>
<td>4.0153E-06</td>
<td>3.9775E-08</td>
</tr>
<tr>
<td>1.0070E-02</td>
<td>-2.3237E+00</td>
<td>-2.3237E+00</td>
<td>-2.3237E+00</td>
<td>3.3622E-06</td>
<td>3.9764E-08</td>
</tr>
<tr>
<td>1.0090E-02</td>
<td>-2.3230E+00</td>
<td>-2.3230E+00</td>
<td>-2.3230E+00</td>
<td>1.4164E-06</td>
<td>3.9752E-08</td>
</tr>
<tr>
<td>2.0000E-02</td>
<td>-2.0537E+00</td>
<td>-2.0537E+00</td>
<td>-2.0537E+00</td>
<td>3.5281E-08</td>
<td>3.5281E-08</td>
</tr>
<tr>
<td>6.0000E-02</td>
<td>-1.5548E+00</td>
<td>-1.5548E+00</td>
<td>-1.5548E+00</td>
<td>2.6589E-08</td>
<td>2.6589E-08</td>
</tr>
<tr>
<td>1.0000E-01</td>
<td>-1.2816E+00</td>
<td>-1.2816E+00</td>
<td>-1.2816E+00</td>
<td>2.1930E-08</td>
<td>2.1930E-08</td>
</tr>
<tr>
<td>2.0000E-01</td>
<td>-8.4162E-01</td>
<td>-8.4162E-01</td>
<td>-8.4162E-01</td>
<td>1.4393E-08</td>
<td>1.4393E-08</td>
</tr>
<tr>
<td>4.0000E-01</td>
<td>-2.5335E-01</td>
<td>-2.5335E-01</td>
<td>-2.5335E-01</td>
<td>4.3358E-09</td>
<td>4.3359E-09</td>
</tr>
<tr>
<td>5.0000E-01</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>7.0000E-01</td>
<td>5.2440E-01</td>
<td>5.2440E-01</td>
<td>5.2440E-01</td>
<td>8.9747E-09</td>
<td>8.9747E-09</td>
</tr>
<tr>
<td>1.0000E+00</td>
<td>7.941000472</td>
<td>7.94E+00</td>
<td>7.94E+00</td>
<td>4.3963E-04</td>
<td></td>
</tr>
</tbody>
</table>
### ATTACHMENT IV
FORTRAN OUTPUT FILES

**TestFwdNorm.out:**

<table>
<thead>
<tr>
<th>$z$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7.90000000000000E+00</td>
<td>1.39451559106967E-15</td>
</tr>
<tr>
<td>-7.89000000000000E+00</td>
<td>1.51093100002460E-15</td>
</tr>
<tr>
<td>-6.05000000000000E+00</td>
<td>7.24228682733315E-10</td>
</tr>
<tr>
<td>-6.00000000000000E+00</td>
<td>9.86586990739334E-10</td>
</tr>
<tr>
<td>-5.00000000000000E+00</td>
<td>2.86651444658299E-07</td>
</tr>
<tr>
<td>-4.00000000000000E+00</td>
<td>3.16712326714731E-05</td>
</tr>
<tr>
<td>-3.00000000000000E+00</td>
<td>1.34989780408652E-03</td>
</tr>
<tr>
<td>-2.33100000000000E+00</td>
<td>9.87667878923520E-03</td>
</tr>
<tr>
<td>-2.32500000000000E+00</td>
<td>1.00359790364380E-02</td>
</tr>
<tr>
<td>-2.00000000000000E+00</td>
<td>2.27501301001457E-02</td>
</tr>
<tr>
<td>-1.00000000000000E+00</td>
<td>1.58655249790304E-01</td>
</tr>
<tr>
<td>0.00000000000000E+00</td>
<td>5.00000000000000E-01</td>
</tr>
<tr>
<td>1.00000000000000E+00</td>
<td>8.41344750209696E-01</td>
</tr>
<tr>
<td>2.00000000000000E+00</td>
<td>9.7724986989854E-01</td>
</tr>
<tr>
<td>2.32500000000000E+00</td>
<td>9.89964020963562E-01</td>
</tr>
<tr>
<td>2.33100000000000E+00</td>
<td>9.90123321210765E-01</td>
</tr>
<tr>
<td>3.00000000000000E+00</td>
<td>9.98650102195913E-01</td>
</tr>
<tr>
<td>4.00000000000000E+00</td>
<td>9.99968328767329E-01</td>
</tr>
<tr>
<td>5.00000000000000E+00</td>
<td>9.9999713348555E-01</td>
</tr>
<tr>
<td>6.05000000000000E+00</td>
<td>9.9999999275771E-01</td>
</tr>
<tr>
<td>7.00000000000000E+00</td>
<td>9.9999999898720E-01</td>
</tr>
<tr>
<td>7.89000000000000E+00</td>
<td>9.9999999999998E-01</td>
</tr>
<tr>
<td>7.90000000000000E+00</td>
<td>9.9999999999999E-01</td>
</tr>
</tbody>
</table>

**TestInvNor.out:**

<table>
<thead>
<tr>
<th>$p$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00000000000000E-15</td>
<td>-7.94200000000000E+00</td>
</tr>
<tr>
<td>2.00000000000000E-03</td>
<td>-2.87816168972574E+00</td>
</tr>
<tr>
<td>4.00000000000000E-03</td>
<td>-2.65206976262383E+00</td>
</tr>
<tr>
<td>8.00000000000000E-03</td>
<td>-2.40891550470837E+00</td>
</tr>
<tr>
<td>1.00000000000000E-02</td>
<td>-2.32634783423627E+00</td>
</tr>
<tr>
<td>1.00500000000000E-02</td>
<td>-2.32447589323511E+00</td>
</tr>
<tr>
<td>1.00700000000000E-02</td>
<td>-2.32372939195777E+00</td>
</tr>
<tr>
<td>1.00900000000000E-02</td>
<td>-2.32298418336969E+00</td>
</tr>
<tr>
<td>2.00000000000000E-02</td>
<td>-2.05374887535086E+00</td>
</tr>
<tr>
<td>6.00000000000000E-02</td>
<td>-1.55477356800796E+00</td>
</tr>
<tr>
<td>p</td>
<td>InvNor-z</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>1.0000000000000E-02</td>
<td>-2.32634783423627E+00</td>
</tr>
<tr>
<td>1.0050000000000E-02</td>
<td>-2.32447589323511E+00</td>
</tr>
<tr>
<td>1.0070000000000E-02</td>
<td>-2.32372939195777E+00</td>
</tr>
<tr>
<td>1.0090000000000E-02</td>
<td>-2.32298418336969E+00</td>
</tr>
<tr>
<td>2.0000000000000E-02</td>
<td>-2.05374887535086E+00</td>
</tr>
<tr>
<td>6.0000000000000E-02</td>
<td>-1.55477356800796E+00</td>
</tr>
<tr>
<td>1.0000000000000E-01</td>
<td>-1.28155154361505E+00</td>
</tr>
<tr>
<td>2.0000000000000E-01</td>
<td>-8.41621219179845E-01</td>
</tr>
<tr>
<td>4.0000000000000E-01</td>
<td>-2.53347098799945E-01</td>
</tr>
<tr>
<td>5.0000000000000E-01</td>
<td>0.00000000000000E+00</td>
</tr>
<tr>
<td>7.0000000000000E-01</td>
<td>5.24400503733308E-01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>z</th>
<th>FwdNorm-p</th>
<th>Table-p</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.3250000000000E+00</td>
<td>1.00359790364380E-02</td>
<td>1.00360791360384E-02</td>
</tr>
<tr>
<td>-2.0000000000000E+00</td>
<td>2.27501301001457E-02</td>
<td>2.27501764027591E-02</td>
</tr>
<tr>
<td>-1.0000000000000E+00</td>
<td>1.58655249790304E-01</td>
<td>1.58655254896831E-01</td>
</tr>
<tr>
<td>0.0000000000000E+00</td>
<td>5.00000000000000E-01</td>
<td>5.00000000000000E-01</td>
</tr>
<tr>
<td>1.0000000000000E+00</td>
<td>8.41344750209696E-01</td>
<td>8.41344745103169E-01</td>
</tr>
<tr>
<td>2.0000000000000E+00</td>
<td>9.77249869899854E-01</td>
<td>9.77249823597241E-01</td>
</tr>
<tr>
<td>2.3250000000000E+00</td>
<td>9.89964020963562E-01</td>
<td>9.89963920863962E-01</td>
</tr>
</tbody>
</table>