Normal probabilities for the standard normal can be computed using an integrated Taylor polynomial. Symmetry of the probability density function about the mean, $\mu = 0$, reduces the problem to positive x. Smoothness and concavity above and below the inflection point lead to the use of two Taylor polynomials.

```
{---------------------------------------------------------------------+
|   Function PROBNORM                                   5/2/88  jlb   |
+---------------------------------------------------------------------+
|   Computes P(X<x) for X distributed N(0,1).  Based upon Algorithm   |
|   209 (D. Ibbetson) Comm. A.C.M. Oct. 1963.                         |
|                                                                     |
|   Argument : x - real                                               |
|   Result   : real                                                   |
|                                                                     |
|   No error checking.  All values of x assumed valid.                |
+---------------------------------------------------------------------}
  function ProbNorm(x : REAL) : REAL;

  Var w, y, z : real;  {Used for intermediate results}

  Begin
    if (x = 0)  then z := 0  {Special case requires no real computation}
      else
        begin
          y := x/2;
          if (y < 0) then y := -y; {Uses symmetry of normal distribution}
          if (y >= 3) then z := 1  {Values this large, |x|>6, are
                                    trivially extreme}
            else if (y >= 1) then
               begin
                 y := y - 2;
                 z := (( -0.000045255659 *y+0.000152529290)*y
                       -0.000019538132);
                 z := (((z*y-0.000676904986)*y+0.001390604284)*y
                       -0.000794620820)*y;
                 z := (((z-0.002034254874)*y+0.006549791214)*y
                       -0.010557625006)*y;
                 z := (((z+0.011630447319)*y-0.009279453341)*y
                       +0.005353579108)*y;
                 z := ((z-0.002141268741)*y+0.000535310849)*y
                       +0.999936657524
               end
              else
               begin
                 w := y*y;
                 z := ((0.000124818987*w-0.001075204047)*w
                       +0.005198775019)*w;
                 z := (((z-0.019198292004)*w+0.059054035642)*w
                       -0.151968751364)*w;
                 z := (((z+0.319152932694)*w-0.531923007300)*w
                       +0.797884560593)*y*2
               end
        end;
      if (x < 0) then
        probnorm := (1-z)/2
      else
        probnorm := (1+z)/2
  end;
```

Computation of chi-square probabilities. Note that for even degrees of freedom, the standard normal can be used to generate probabilities.

```
{---------------------------------------------------------------------+
|   Function PROBCHISQ                                  5/2/88  jlb   |
+---------------------------------------------------------------------+
|   Computes P(X<x) for X distributed Chi-square with df degrees of   |
|   freedom.                                                          |
|   Based upon Algorithm 299 (I.D. Hill and M.C. Pike) Comm. A.C.M.   |
|   April 1967.                                                       |
|                                                                     |
|   Argument : x - real                                               |
|              df - integer                                           |
|   Result   : real                                                   |
|                                                                     |
|   No error checking.  Returns 0 when x < 0 and/or df < 1.           |
+---------------------------------------------------------------------}
  function ProbChisq(x : REAL; df : INTEGER) : REAL;

  Var a, c, e, s, y, z : real;  {Used for intermediate results}
```

```
    big, small : boolean;

Begin
  if ((x <= 0) or (df < 1)) then
      ProbChisq := 0
    else
      begin
        a := x / 2;
        y := 0;
        if (x >= 50) then
            begin
              big := true;
              small := false
            end
          else if (df > 2) then
                   begin
                     small := true;
                     big := false
                   end
                 else
                   begin
                     small := false;
                     big := false
                   end;
        if ((not(odd(df))) or small) then
          y := exp(-a);
        s := y;
        if (odd(df)) then s := 2*probnorm(-sqrt(x));
        if (df <= 2) then
           ProbChisq := 1 - s
          else
            begin
              x := (df-1)/2;
              z := 0.5;
              if (not(odd(df))) then z := 1;
              if (big) then
                 begin
                   if (odd(df)) then
                       e := 0.572364942925
                     else
```

```
                       e := 0;
                   c := ln(a);
                   while (z <= x) do
                     begin
                       e := ln(z) + e;
                       s := exp(c*z-a-e)+s;
                       z := z+1
                     end;
                   ProbChisq := 1 - s
                 end
                else
                 begin
                   if(odd(df)) then
                       e := 0.564189583548/sqrt(a)
                     else
                       e := 1;
                   c := 0;
                   while (z <= x) do
                     begin
                       e := e*a/z;
                       c := c+e;
                       z := z+1
                     end;
                   ProbChisq := 1-(c*y+s)
                 end
            end
      end
end;
```