2.15 Procedures to Avoid Loss of Precision: $\ln(1+x)$, etc.

A. Purpose

These procedures compute expressions involving logarithm, exponential, sine, cosine, hyperbolic sine, hyperbolic cosine, and the gamma function, some having arguments that are expressions, more accurately than is possible using the Fortran intrinsic functions, or the gamma function described in Chapter 2.3. See also discussion of the function SRCVAL in Section C of Chapter 2.9.

B. Usage

B.1 Program Prototype, Single Precision, $\ln(1+x)$

REAL X, U, SLNREL

EXTERNAL SLNREL

Assign a value to X and obtain U = ln(1 + X) by using

U = SLNREL(X)

B.2 Program Prototype, Single Precision, $x - \ln(1+x)$

REAL X, U, SRLOG1

EXTERNAL SRLOG1

Assign a value to X and obtain U = X − ln(1 + X) by using

U = SRLOG1(X)

B.3 Program Prototype, Single Precision, $x - 1 - \ln(x)$

REAL X, U, SRLOG

EXTERNAL SRLOG

Assign a value to X and obtain U = X − 1 − ln(X) by using

U = SRLOG(X)

B.4 Program Prototype, Single Precision, $\exp(x) - 1$

REAL X, U, SREXP

EXTERNAL SREXP

Assign a value to X and obtain U = exp(X) − 1 by using

U = SREXP(X)

B.5 Program Prototype, Single Precision, $\sin(\pi x)$

REAL X, U, SSINPX

EXTERNAL SSINPX

Assign a value to X and obtain U = sin(πX) by using

U = SSINPX(X)

B.6 Program Prototype, Single Precision, $\cos(\pi x)$

REAL X, U, SCOSPX

EXTERNAL SCOSPX

Assign a value to X and obtain U = cos(πX) by using

U = SCOSPX(X)

B.7 Program Prototype, Single Precision, $(x - \sin(x))/x^3$

REAL X, U, SSIN1

EXTERNAL SSIN1

Assign a value to X and obtain U = (X − sin(X))/X^3 by using

U = SSIN1(X)

B.8 Program Prototype, Single Precision, $(1 - \cos(x))/x^2$

REAL X, U, SCOS1

EXTERNAL SCOS1

Assign a value to X and obtain U = (1 − cos(X))/X^2 by using

U = SCOS1(X)

B.9 Program Prototype, Single Precision, $\sinh(x) - x$

REAL X, U, SSINHM

EXTERNAL SSINHM

Assign a value to X and obtain U = sinh(X) − X by using

U = SSINHM(X)

B.10 Program Prototype, Single Precision, $\cosh(x) - 1$

REAL X, U, SCOSHM

EXTERNAL SCOSHM

Assign a value to X and obtain U = cosh(X) − 1 by using

U = SCOSHM(X)

B.11 Program Prototype, Single Precision, $\cosh(x) - 1 - \frac{1}{2}x^2$

REAL X, U, SCSHMM

EXTERNAL SCSHMM

Assign a value to X and obtain U = cosh(X) − 1 − $\frac{1}{2}$X^2 by using

U = SCSHMM(X)

B.12 Program Prototype, Single Precision, $1/\Gamma(x+1) - 1$

REAL X, U, SGAM1

EXTERNAL SGAM1

Assign a value to X and obtain U = 1/Γ(1 + X) − 1 by using

U = SGAM1(X)

B.13 Argument Definitions

X [in] Argument of functions. Argument values may be constrained as described in Section E below.

B.14 Modifications for Double Precision

For double precision computation, change the REAL type statement to DOUBLE PRECISION and change the initial letter of the function name to D. Since these functions are not generic intrinsic functions, it is important to declare them explicitly to be DOUBLE PRECISION, because the default implicit type would be REAL. The approximations used in SSINHM, SCOSHM and SCSHMM are accurate to 14 decimal digits, so the REAL versions of these procedures could be used to advantage on some platforms by changing the function type and all internal variable types, constants, and arithmetic to double precision.

C. Example and Remarks

See DRDLNREL and ODDLNREL for an example of the usage of these subprograms.

MATH77 does not include a special procedure to compute $\sin(x)/x$. On all machines tested, $\sin(x)/x$ could be computed with good relative accuracy, even for very small x, by using the Fortran intrinsic SIN function. The only difficulty is that the user must test whether x is exactly zero, and use 1.0 for the result. The procedure SSIN1, described above, can be used to compute $1 - \sin(x)/x$ without loss of precision.

D. Functional Description

D.1 Method

For xLNREL, when $1/\sqrt{2} < 1 + x < \sqrt{2}$, use approximation 2707 from Hart, et. al., [1]. Otherwise, use the Fortran intrinsic logarithm function.

For xRLOG1, when $-0.39 \le x \le 0.57$, use rational approximations due to Alfred H. Morris, Jr., [2]. Otherwise, use the Fortran intrinsic logarithm function.

For xRLOG, when $0.61 \le x \le 1.57$, use rational approximations due to Alfred H. Morris, Jr., [2]. Otherwise, use the Fortran intrinsic logarithm function.

For xREXP, when $|x| \le 0.15$, use rational approximations due to Alfred H. Morris, Jr., [2]. Otherwise, use the Fortran intrinsic exponential function.

For xSINPX and xCOSPX, for all values of x, use rational approximations due to Alfred H. Morris, Jr., [2].

For xSIN1, use Taylor's series for $|x| < 0.25$. Otherwise, use the Fortran intrinsic sine function.

For xCOS1, when $\cos(x) \ge 0$ use $1 - \cos(x) = \sin^2(x)/(1 + \cos(x))$. Otherwise, use $1 - \cos(x)$. In both cases, use the Fortran intrinsic functions.

For SSINHM, when $|x| < 1.65$, use rational approximations due to A. K. Cline and R. J. Renka, [2]. Otherwise, use the Fortran intrinsic SINH function.

For SCOSHM, when $|x| < 1.2$, use rational approximations due to A. K. Cline and R. J. Renka, [2]. Otherwise, use the Fortran intrinsic COSH function.

For SCSHMM, when $|x| < 2.7$, use rational approximations due to A. K. Cline and R. J. Renka, [2]. Otherwise, use the Fortran intrinsic COSH function.

For xGAM1, when $-0.5 \le x \le 1.5$, use rational approximations due to Alfred H. Morris, Jr., [2]. Otherwise, use the xGAMMA routine from Chapter 2.3.

D.2 Accuracy Tests

The single precision subprograms were tested on an IBM PC/AT using IEEE arithmetic, by comparison with the double precision Fortran intrinsic functions (except the reference value for SGAM1 used DGAMMA from Chapter 2.3) at 2000 points. For each function except SSINPX and SCOSPX, the points were chosen to cover the entire domain over which a special approximation is used. For SSINPX and SCOSPX, 2000 points were chosen randomly in the region $0 < x < 2$, and SSINPX(10^x),

	Percentage of samples with specified number of bits wrong										
Function	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	≥ 10
SLNREL	100.0										
$\ln(x+1)$	61.7	24.6	13.8								
SRLOG1	57.3	23.7	10.9	6.7	1.5						
$x - \ln(x+1)$	3.9	4.6	9.0	11.9	17.9	16.9	11.0	7.0	5.3	4.1	8.5
SRLOG	56.7	23.9	10.9	6.9	1.6						
$x - 1 - \ln(x)$	4.2	3.9	7.3	12.1	19.4	15.9	11.4	8.6	5.6	3.4	8.5
SREXP	71.7	26.5	1.9								
$\exp(x) - 1$	8.2	8.8	16.9	26.0	18.8	10.8	5.2	2.8	1.6	0.4	0.6
SSINPX	10.6	7.6	9.9	11.9	13.0	12.1	12.1	8.7	5.8	3.3	5.2
$\sin(\pi x)$	4.9	4.8	8.8	11.8	12.9	13.2	13.6	11.4	8.2	5.2	5.4
SCOSPX	11.4	7.1	11.4	11.8	11.4	12.2	11.1	10.1	6.0	3.9	3.9
$\cos(\pi x)$	5.1	4.3	9.4	10.9	12.7	13.7	13.4	11.9	7.8	5.3	5.6
SSIN1	91.1	8.9									
$(x - \sin(x))/x^3$	54.2	20.1	2.0	2.3	2.0	2.2	2.1	1.5	1.6	1.8	9.8
SCOS1	76.5	15.8	7.7	0.1							
$(1 - \cos(x))/x^2$	57.5	18.6	4.3	2.0	1.7	2.4	1.6	2.2	2.1	1.8	6.1
SSINHM	66.3	29.4	4.3								
$\sinh(x) - x$	13.4	12.3	23.6	17.0	8.5	7.7	5.0	3.7	2.5	2.0	4.3
SCOSHM	68.7	28.3	3.0								
$\cosh(x) - 1$	18.8	19.1	17.5	13.7	8.5	6.0	5.0	3.5	2.8	1.6	3.7
SCSHMM	46.5	32.9	19.8	0.8							
$\cosh(x) - 1 - x^2/2$	12.0	11.4	16.9	13.7	10.4	5.7	5.0	4.0	3.0	3.3	14.7
SGAM1	63.3	27.5	8.9								
$1/\Gamma(1+x) - 1$	8.2	8.7	13.9	20.7	22.1	14.7	5.4	3.7	1.5	0.6	0.7

$\sin(10^x\pi)$, SCOSPX(10^x) and $\cos(10^x\pi)$ were then computed. For SSIN1 and SCOS1, 2000 points were chosen randomly in the region $-8 < x < -1$, then $y = 10^x$, SSIN1(y), $(y - \sin(y))/y^3$, SCOS1(y) and $(1 - \cos(y))/y^2$ were computed. The results of these tests may be summarized as follows:

Function	**avg |E|**	max |E|	**std. dev.**
SLNREL	0.25	0.50	0.14
$\ln(x+1)$	0.50	1.49	1.49
SRLOG1	0.46	3.28	0.43
$x - \ln(x+1)$	8075.58	$\approx 7.9 \times 10^6$	$\approx 1.9 \times 10^5$
SRLOG	0.47	3.10	0.43
$x - 1 - \ln(x)$	3341.24	$\approx 2.9 \times 10^6$	$\approx 7.3 \times 10^4$
SREXP	0.37	1.39	0.26
$\exp(x) - 1$	16.51	7402.74	190.13
SSINPX	58.32	5795.79	253.07
$\sin(\pi x)$	119.20	48061.18	1193.09
SCOSPX	84.64	35894.27	965.42
$\cos(\pi x)$	94.27	19712.38	588.76
SSIN1	0.32	0.97	0.13
$(x - \sin(x))/x^3$	407.81	28944.33	2091.66
SCOS1	0.25	2.02	0.38
$(x - \cos(x))/x^2$	102.36	7931.0	525.75
SSINHM	0.35	1.36	0.35
$\sinh(x) - x$	2584.93	$\approx 2.6 \times 10^6$	$\approx 6.7 \times 10^4$
SCOSHM	0.39	1.35	0.27
$\cosh(x) - 1$	2302.36	$\approx 3.7 \times 10^6$	$\approx 8.3 \times 10^4$
SCSHMM	0.65	23.00	0.68
$\cosh(x) - 1 - x^2/2$	$\approx 1.9 \times 10^5$	$> 10^7$	$\approx 1.6 \times 10^6$
SGAM1	0.46	2.44	0.37
$1/\Gamma(1+x) - 1$	32.94	28016.70	686.03

For the functions SLNREL, SRLOG1, SRLOG, SREXP, SSINHM, SCOSHM and SGAM1 the error in units of the least significant digit was approximately constant over the entire region. For SSINPX and SCOSPX the error in units of the least significant digit and the absolute error both increased as the argument increased. For SCSHMM the error in units of the least significant digit was nearly constant over the entire region, except that one sample had 5 wrong bits near the zero of the function.

When $\ln(1+x)$, $x-\ln(1+x)$, $x-1-\ln(x)$, $(x-\sin(x))/x^3$, $(1-\cos(x))/x^2$, $\sinh(x)-x$, $\cosh(x)-1$, $\cosh(x)-1-x^2/2$ and $1/\Gamma(1+x)-1$ were computed using single precision intrinsic functions and single precision arithmetic, there was substantial increase in the error in units of the least significant digit near the zeros of the functions, even though the absolute error approached zero. For $\sin(\pi x)$ and $\cos(\pi x)$ the error in units of the least significant digit and the absolute error increased as the argument increased. The average, maximum and standard deviation of the errors in units of the least significant digit are summarized in the table at left, in units of $\rho = 2^{-23} \approx 1.192 \times 10^{-7}$, the relative precision of IEEE single precision arithmetic.

For arguments in the range $0 < x < 1$, SSINPX and SCOSPX commit approximately the same error as equivalent expressions constructed from the intrinsic SIN and COS functions.

References

1. J. F. Hart et al., **Computer Approximations**, J. Wiley and Sons, New York (1968).
2. Alfred. H. Morris, Jr., **NSWC Library of Mathematics Subroutines**. Technical Report NSWCDD/TR-92/425, Naval Surface Warfare Center, Dahlgren, VA 22448-5000 USA (Jan. 1993).

E. Error Procedures and Restrictions

If the argument of xLNREL or xRLOG1 ≤ -1.0 or the argument of xRLOG ≤ 0.0, the Fortran intrinsic logarithm function will produce an error message. If the argument of xGAM1 ≤ -1.0 the xGAMMA function will produce an error message. If the magnitude of the argument of SSINPX, DSINPX, SCOSPX or DCOSPX $> 1/\rho$, where ρ is the smallest number that can be added to one and give a result different from one, these routines will issue an error message using the error message processor, described in Chapter 19.2, at level 2. If error termination is suppressed by using ERMSET, SSINPX and DSINPX return 0.0, while SCOSPX and DCOSPX return 1.0.

F. Supporting Information

Entry	Required Files
DCOS1	DCOS1
DCOSHM	AMACH, DCOSHM
DCOSPX	AMACH, ERFIN, ERMSG, DCOSPX, DERM1, DERV1
DCSHMM	AMACH, DCSHMM
DGAM1	AMACH, DERM1, DERV1, DGAM1, DGAMMA, ERFIN, ERMSG
DLNREL	DLNREL
DREXP	DREXP
DRLOG	DRLOG
DRLOG1	DRLOG
DSIN1	AMACH, DSIN1
DSINHM	AMACH, DSINHM
DSINPX	AMACH, ERFIN, ERMSG, DERM1, DERV1, DSINPX
SCOS1	SCOS1
SCOSHM	SCOSHM
SCOSPX	AMACH, ERFIN, ERMSG, SCOSPX, SERM1, SERV1
SCSHMM	SCSHMM
SGAM1	AMACH, ERFIN, ERMSG, SERM1, SERV1, SGAM1, SGAMMA
SLNREL	SLNREL
SREXP	SREXP
SRLOG	SRLOG
SRLOG1	SRLOG
SSIN1	AMACH, SSIN1
SSINHM	SSINHM
SSINPX	AMACH, ERFIN, ERMSG, SERM1, SERV1, SSINPX

Designed and programmed by W. V. Snyder, JPL 1991 and 1993.

DRDLNREL

```
c       program DRDLNREL
c>> 1999-07-27 DRDLNREL Krogh analyz => DANAL for global analysis.
c>> 1996-07-09 DRDLNREL Krogh Moved formats up.
c>> 1994-11-02 DRDLNREL Krogh  Changes to use M77CON
```

```
c>> 1994-07-06 DRDLNREL WVS set up for chgtyp
c
c     Demonstration driver for more accurate routines from chapter 2-15.
c     To illustrate worst-case, expressions are computed little-by-
c     little.  If possible, the routine should be compiled with an
c     option that causes results not to be stored in registers from one
c     assignment to the next.  For Lahey Fortran on a PC, use /7.
c
c--D replaces "?": DR?LNREL, ?GAMMA, ?LNREL, ?RLOG1, ?RLOG, ?REXP
c--   &             ?SINPX, ?COSPX, ?SINHM, ?COSHM, ?CSHMM, ?GAM1
c--   &             ?COS1, ?SIN1, ?ANAL
      double precision DPI
      parameter (DPI = 3.14159265358979323846264338327950288419 7D0)
      double precision D, D1MACH, DGAMMA, U, W
      external D1MACH, DGAMMA
      double precision DLNREL, DRLOG1, DRLOG, DREXP, DSINPX, DCOSPX
      double precision DSINHM, DCOSHM, DCSHMM, DGAM1
      double precision DCOS1, DSIN1
      external DLNREL, DRLOG1, DRLOG, DREXP, DSINPX, DCOSPX
      external DSINHM, DCOSHM, DCSHMM, DGAM1
      external DCOS1, DSIN1
10    format (' Error is (relative error) / (round off level).'/
     1' Round off level is (smallest number r such that 1 + r .NE. 1).'/
     2' For the present machine, r =',1p,g13.6/
     3' Errors less than 0.5 r should be considered to be zero.')
c
c     DLNREL
      d = 1.0d0 / 2.0d0**12
      w = 1.0d0 + d
      w = log(w)
      u = dlnrel(d)
      call DANAL ('DLNREL',d,u,w)
c
c     DRLOG1
      w = d - w
      u = drlog1(d)
      call DANAL ('DRLOG1',d,u,w)
c
c     DRLOG
      w = d - 1.0d0
      w = w - log(d)
      u = drlog(d)
      call DANAL ('DRLOG ',d,u,w)
c
c     DREXP
      w = exp(d)
      w = w - 1.0d0
      u = drexp(d)
      call DANAL ('DREXP ',d,u,w)
c
c     DSINPX
      d = 25.125d0
      w = sin(d*dpi)
      u = dsinpx(d)
      call DANAL ('DSINPX',d,u,w)
c
c     DCOSPX
      w = cos(d*dpi)
      u = dcospx(d)
```

```
      call DANAL ('DCOSPX',d,u,w)
c
c     DSIN1
      d = 0.5d0**22
      w = sin(d)
      w = (d - w) / d**3
      u = dsin1(d)
      call DANAL ('DSIN1 ',d,u,w)
c
c     DCOS1
      w = cos(d)
      w = (d - w) / d**2
      u = dcos1(d)
      call DANAL ('DCOS1 ',d,u,w)
c
c     DSINHM
      d = 0.25d0
      w = sinh(d)
      w = w - d
      u = dsinhm(d)
      call DANAL ('DSINHM',d,u,w)
c
c     DCOSHM
      w = cosh(d)
      w = w - 1.0d0
      u = dcoshm(d)
      call DANAL ('DCOSHM',d,u,w)
c
c     DCSHMM
      w = cosh(d)
      w = w - 1.0d0
      w = w - 0.5*d*d
      u = dcshmm(d)
      call DANAL ('DCSHMM',d,u,w)
c
c     DGAM1
      w = 1.0d0/dgamma(1.0d0+d)
      w = w - 1.0d0
      u = dgam1(d)
      call DANAL ('DGAM1 ',d,u,w)
c
      print 10, d1mach(4)
      stop
      end

      subroutine DANAL (ROUTIN, X, U, W)
      character*6 ROUTIN
      double precision X, U, W, EW
      double precision D1MACH
      external D1MACH
      double precision ROUND
      save ROUND
      data ROUND /-1.0d0/
10    format (
     1' Function                    Result          ---- Not Using ----'/
     2' From This                   Using This      --- This Package ---'/
     3' Package        Argument     Package         Result         Error')
20    format (1x,a6,5x,1p,e12.4,2x,e13.6,2x,e13.6,1x,e9.2)
      if (round .lt. 0.0d0) then
```

```
    round = d1mach(4)
    print 10
end if
ew = abs(w-u)/abs(u*round)
print 20, routin,x,u,w,ew
return
end
```

ODDLNREL

Function From This Package	Argument	Result Using This Package	——— Not Using ——— This Package —— Result	Error
DLNREL	2.4414E−04	2.441108E−04	2.441108E−04	0.00E+00
DRLOG1	2.4414E−04	2.979747E−08	2.979747E−08	2.68E+02
DRLOG	2.4414E−04	7.318010E+00	7.318010E+00	0.00E+00
DREXP	2.4414E−04	2.441704E−04	2.441704E−04	1.36E+03
DSINPX	2.5125E+01	−3.826834E−01	−3.826834E−01	9.15E+00
DCOSPX	2.5125E+01	−9.238795E−01	−9.238795E−01	1.62E+00
DSIN1	2.3842E−07	1.666667E−01	1.660156E−01	1.76E+13
DCOS1	2.3842E−07	5.000000E−01	−1.759218E+13	1.58E+29
DSINHM	2.5000E−01	2.612317E−03	2.612317E−03	9.72E+00
DCOSHM	2.5000E−01	3.141310E−02	3.141310E−02	1.99E+00
DCSHMM	2.5000E−01	1.630999E−04	1.630999E−04	4.21E+02
DGAM1	2.5000E−01	1.032627E−01	1.032627E−01	1.21E+00

Error is (relative error) / (round off level).
Round off level is (smallest number r such that 1 + r .NE. 1).
For the present machine, r = 2.220446E−16
Errors less than 0.5 r should be considered to be zero.