ELEMENTARY FUNCTIONS

The elementary functions of mode PROC(REAL)REAL are:

sqrt	(square root)
cbrt	(cube root)
exp	(e to the power)
ln	(natural logarithm)
sin	
cos	
tan	
cot	
sec	
cosec	
sinh	hyperbolic sin, alternative identifier sh
cosh	cos ch
tanh	tan th
coth	
secn	
cosech	
arcsin	inverse sine in range (-pi/2, pi/2)
arccos	inverse cosine in range (0,pi)
arctan	inverse tangent in range (-pi/2, pi/2)
arccot	in range (-pi/2, pi/2)
arcsec	in range (0,pi)
arccosec	in range (-pi/2, pi/2)
sinc	the function sinc(x) is defined as (sin(pi*x))/(pi*x)

COMPLEX FUNCTIONS

The following functions take one COMPLEX parameter and deliver a COMPLEX result (mode PROC(COMPLEX)COMPLEX).

csqrt	delivers the square root z of a complex number, where -pi/2 < ARG z ⩽ pi/2
cexp	delivers the exponential of a complex number z. Fault display OVERFLOW IN LIBRARY PROC CEXP occurs if re OF z > 176 approx.
cln	delivers the natural logarithm (principal value). The parameter may have any value except zero.

BESSEL FUNCTIONS

The following procedures deliver the value of the Bessel functions $J_0(x)$, $J_1(x)$, $Y_0(x)$ etc.

```
PROC j0 = (REAL x) REAL
PROC j1 = (REAL x) REAL
PROC y0 = (REAL x) REAL
PROC y1 = (REAL x) REAL
PROC i0 = (REAL x) REAL
PROC i1 = (REAL x) REAL
PROC k0 = (REAL x) REAL
PROC k1 = (REAL x) REAL
```

All values are correct to at least 10 decimal places or 10 significant figures, whichever is the less accurate, for values of x normally encountered. However, for the oscillatory functions j0, j1, y0 and y1 the maximum number of correct decimal places decreases by 1 and the maximum number of correct significant figures decreases by 2 for every increase in x by a factor of 100 above unity, until all relative accuracy is lost when x is of the order of 1 & 10. The relative accuracy of the non-oscillatory functions (i0, i1, k0 and k1) is fairly constant at about 10 significant figures. Fault displays are:

ILLEGAL PARAMETER OF PROC
In procedures y0, y1, k0, k1 if x = 0.0

OVERFLOW IN LIBRARY PROC
In procedures y1, k1 if ABS x < 1.73 & -77 approx.
In procedures i0, i1 if ABS x > 176 approx.

The procedures k0 and k1 each deliver the value zero if x > 176 approx.

ERROR FUNCTION

```
PROC erf = (REAL x) REAL
```

This function delivers the value of $\frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt$

If x > +4.845 approx, the value +1.0 is delivered.
If x < -4.845 approx, the value -1.0 is delivered.

COMPLEMENTARY ERROR FUNCTION

```
PROC erfc = (REAL x) REAL
```

This function delivers the value of $\frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-t^2)\, dt$

If x > 13.228 approx, the value 0.0 is delivered.
If x < -4.845 approx, the value 2.0 is delivered.

The function should be used instead of the expression (1 - erf(x)) whenever the complementary error function is required, since significant figures are lost in the subtraction if x > 0.5.

For COMPLEX error functions, please see LIBRARY Functions 9.

GAMMA FUNCTION

```
PROC gamma = (REAL x) REAL
```

This procedure delivers the value of gamma(x).

If x is zero or a negative integer,
fault display ILLEGAL PARAMETER OF PROC GAMMA

If x > 58.088 approx,
fault display OVERFLOW IN LIBRARY PROC GAMMA

If x < -57.238 approx, unless a negative integer,
the value zero is delivered.

The procedure is faster for integers and half-integers, which are most commonly required, than for other values of x.

The factorial of n is given by gamma(n + 1), which gives a real result. The operator ROUND should be used if the integer value is required; overflow occurs if n > 10.

When calculating series whose terms include factorials, it is usually more efficient to calculate each term from the previous one, rather than use repeated calls of gamma.

EVALUATION OF REAL POLYNOMIAL

```
PROC polynomial = (REAL x, [ ]REAL coeffs) REAL
```

This procedure delivers the value of the polynomial

$$c_0 + c_1x + c_2x^2 + \ldots\ldots + c_nx^n$$

where the coefficients c_0 to c_n are given by the user in the array parameter "coeffs". The array of coefficients may have any bounds.

The procedure uses n floating point additions and n multiplications to evaluate an nth order polynomial. If floating point overflow is set during the evaluation the fault display is OVERFLOW IN LIBRARY PROC POLYNOMIAL.

COMPLETE ELLIPTIC INTEGRALS

```
PROC compellint = (REAL k, REF REAL kdash, first, second) VOID
```

Given the value of the modulus "k" as its first parameter, this procedure carries out 3 assignments:

1 It assigns to "kdash" the value of the complementary modulus

$$k' = \sqrt{1 - k^2}$$

2 It assigns to "first" the value of the complete elliptic integral of the first kind

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2\sin^2\theta}}$$

3 It assigns to "second" the value of the complete elliptic integral of the second kind

$$E(k) = \int_0^{\pi/2} \sqrt{1 - k^2\sin^2\theta}\, d\theta$$

If any of these three values are not required, NIL should be used for the corresponding actual parameters. Fault displays are:

ILLEGAL PARAMETER OF PROC COMPELLINT if $|k|>1.0$

OVERFLOW IN LIBRARY PROC if $|k|=1.0$
and the integral of the first kind is required

RATIO OF COMPLETE ELLIPTIC INTEGRALS OF THE FIRST KIND, AND ITS INVERSE

```
PROC ellkratio = (REAL t) REAL
```

If K, K' are the complete elliptic integrals of the first kind whose moduli are k and $k' = \sqrt{(1 - k^2)}$ respectively (see above), this procedure delivers the ratio K/K' corresponding to any value of $t = k/k'$.

```
PROC invellkratio = (REAL r, REF REAL kdash) REAL
```

If $r = K/K'$ where K, K' are defined as above, this procedure delivers the value of k and assigns to kdash the value of k'. The parameter r may have any non-negative value. If k' is not required, NIL may be used as the actual parameter for kdash.

These procedures should be used whenever the ratio K/K' (or its inverse) is required rather than the actual values of K and K', which covers a large percentage of cases in practice. The procedures are simpler and give a more accurate result than the use of "compellint", especially when k or k' is small (when k' or k will be nearly 1.0 and much accuracy is lost, and possibly overflow caused). There is no possibility of overflow with "ellkratio" and "invellkratio" however large or small the parameters are.

CONVERSION OF BINARY TO GRAY CODE AND VICE VERSA

```
PROC g of b = (INT b) INT
```

This procedure delivers the Gray code (sometimes called Cyclic Progressive or CP code) corresponding to any binary integer b. Its result will have the same sign as b. The Gray code has the property that when the binary integer b alters by one unit, only one binary digit of the corresponding Gray code alters. The code is also cyclic, ie the Gray code corresponding to +8388607 only differs in one binary digit from that corresponding to -8388608. For $p < 24$, the least significant p bits also form a cyclic code for positive p-bit integers in the range 0 to $2^p - 1$.

```
PROC b of g = (INT g) INT
```

This procedure delivers the binary integer b corresponding to any Gray code g (represented as an integer). It will have the same sign as g. If not more than the last p bits of g are non-zero ($p < 24$), the corresponding value of b will be in the range 0 to $2^p - 1$.

HIGHEST COMMON FACTOR AND LEAST COMMON MULTIPLE OF TWO INTEGERS

```
PROC hcf = (INT p, q) INT
```

This procedure delivers the highest common (positive) factor of two integers p and q. Although any number is a factor of zero, the value of hcf(0, 0) is given as 0. The procedure gives fault display OVERFLOW IN LIBRARY PROC if both p and q are -8388608 or if one of them is -8388608 and the other is zero.

```
PROC lcm = (INT p, q) INT
```

This procedure delivers the least common (positive) multiple of two integers p and q. If either p or q is zero the procedure delivers zero. Fault display OVERFLOW IN LIBRARY PROC occurs if the lcm is greater than 8388607.

COMPLEX ERROR FUNCTION

```
PROC cerf = (COMPLEX z) COMPLEX
```

This procedure delivers the value of $\frac{2}{\sqrt{\pi}} \int_0^z \exp(-t^2)\, dt$

As the integrand has no finite poles or other singularities, the value of the integral is independent of the path of integration. If y = 0.0, where z = x + iy, cerf(z) = erf(x) (see Functions 3).

COMPLEX COMPLEMENTARY ERROR FUNCTION

```
PROC cerfc = (COMPLEX z) COMPLEX
```

This procedure delivers the value of $\frac{2}{\sqrt{\pi}} \int_z^\infty \exp(-t^2)\, dt = 1 - \text{cerf}(z)$

The integral is again independent of the path of integration as the integrand is zero at infinity and has no finite singularities. The function should be used instead of the expression (1 - cerf(z)) whenever the complementary error function is required, since many significant figures may be lost in the subtraction. Both cerf and cerfc may overflow giving fault display OVERFLOW IN LIBRARY PROC if ABS y > ABS x and ABS y > 13.35, where z = x + iy. If y = 0.0, cerfc(z) = erfc(x) (see Functions 3).

COMPLEX Φ - FUNCTION

```
PROC cphi = (COMPLEX z) COMPLEX
```

This procedure delivers the value of

$$\Phi(z) \equiv \frac{2}{\sqrt{\pi}} \exp(z^2) \int_z^\infty \exp(-t^2)\, dt = \text{cexp}(z * z) * \text{cerfc}(z)$$

It is often the function required instead of cerfc(z) itself, as provided x >= 0 (where z = x + iy), its real and imaginary parts vary smoothly with x and y instead of being oscillatory as are those of cerfc(z). Also its absolute value remains bounded provided x >= 0, and neither decreases exponentially as x tends to infinity nor increases exponentially as y tends to infinity as does that of cerfc(z). However, it does increase exponentially as x tends to minus infinity and will overflow, giving fault display OVERFLOW IN LIBRARY PROC, if x < 0 and $x^2 - y^2 > 176$ approx.

EXPONENTIATION OF A COMPLEX NUMBER

```
PROC cpower = (COMPLEX z, INT n) COMPLEX
```

This procedure delivers the value of z raised to the power n. The parameter n may be positive or negative, and z may have any value. The procedure will not overflow (eg in calculating the square of ABS z if n is negative) unless the result of the exponentiation is an overflowed number, in which case fault display OVERFLOW IN LIBRARY PROC CPOWER will occur.

For raising an integer or a real number to an integer power, the operator ↑ is provided in the system library. A similar operator for complex can be defined in a program by including the declaration

```
OP (COMPLEX, INT)COMPLEX ↑ = cpower;
```

Subsequently the expression z ↑ n may be used instead of cpower(z, n). The operator may of course be kept in an album if desired.

FRESNEL INTEGRALS

```
PROC fresnel = (REAL x, REF REAL cc, ss) VOID
```

This procedure evaluates the Fresnel integrals

$$C(x) = \int_0^x \cos(\pi t^2/2)\, dt$$

and $$S(x) = \int_0^x \sin(\pi t^2/2)\, dt$$

and assigns the value of C(x) to cc and the value of S(x) to ss. If either C(x) or S(x) is not required, NIL may be used as the corresponding actual parameter and the value will then not be calculated.

To assign the complex value C(x) + i S(x) to some COMPLEX variable z, simply write

```
fresnel(x, re OF z, im OF z)
```

The parameter x may have any value from -infinity to +infinity.

JACOBIAN ELLIPTIC FUNCTIONS

Each of the following procedures delivers the value of the corresponding Jacobian elliptic function:-

```
PROC sn = (REAL u, m) REAL
PROC cn = (REAL u, m) REAL
PROC dn = (REAL u, m) REAL
PROC ns = (REAL u, m) REAL
PROC nc = (REAL u, m) REAL
PROC nd = (REAL u, m) REAL
PROC sc = (REAL u, m) REAL
PROC sd = (REAL u, m) REAL
PROC cs = (REAL u, m) REAL
PROC cd = (REAL u, m) REAL
PROC ds = (REAL u, m) REAL
PROC dc = (REAL u, m) REAL
```

Each of the functions has two arguments, u and m. The second, m, is known as the *parameter* and is usually constant in any given application. In the first instance m will be restricted to the range $0.0 \leq m \leq 1.0$ and is then often denoted by k^2, where k is known as the *modulus*. The quantity $m1 = 1.0 - m$ is known as the *complementary parameter*, and $k' = \sqrt{m1}$ as the *complementary modulus*.

1 Definitions

$$\text{Let } u = \int_{0.0}^{x} \frac{dt}{\sqrt{(1.0 - t^2)(1.0 - m.t^2)}} \quad \ldots\ldots\ldots\ (1)$$

$$\text{and } K = \int_{0.0}^{1.0} \frac{dt}{\sqrt{(1.0 - t^2)(1.0 - m.t^2)}}$$

where initially $0.0 < m < 1.0$, $-1.0 \leq x \leq +1.0$.

K is a function of m only, and hence of k. K(k) is in fact the complete elliptic integral of the first kind mentioned in Functions 6.

Equation (1) defines u as an odd function of x (regarding m as constant) which increases steadily from 0.0 to K as x increases from 0.0 to 1.0. Hence x is an odd function of u which increases steadily from 0.0 to 1.0 as u increases from 0.0 to K. This function is denoted by sn, so we have $x = sn(u)$, or recognising the fact that x also depends on m, $x = sn(u, m)$.

Functions cn, dn are then defined by

$$cn(u, m) = \sqrt{1.0 - sn^2(u, m)}$$

$$dn(u, m) = \sqrt{1.0 - m.sn^2(u, m)}$$

where both square roots are positive as long as u is confined to the interval $-K \leqslant u \leqslant +K$, so that cn and dn are even functions of u.

2 Other derived functions

To avoid repeating the parameter m each time, in what follows sn(u, m) will be denoted by snu, sn(v, m) by snv, sn(0.0, m) by sn0, sn(K, m) by snk, sn(2K, m) by sn2k, and similarly for the other functions. Then

sn0 = 0.0	cn0 = 1.0	dn0 = 1.0
snk = 1.0	cnk = 0.0	dnk = k'

Further derived functions are defined as follows:-

```
ns(u, m) = nsu = 1.0/snu
nc(u, m) = ncu = 1.0/cnu
nd(u, m) = ndu = 1.0/dnu
sc(u, m) = scu = snu/cnu
sd(u, m) = sdu = snu/dnu
cs(u, m) = csu = cnu/snu
cd(u, m) = cdu = cnu/dnu
ds(u, m) = dsu = dnu/snu
dc(u, m) = dcu = dnu/cnu
```

3 Identities

$$snu^2 + cnu^2 = 1.0 = dnu^2 + m.snu^2$$
$$m.cnu^2 + m1 = dnu^2 = cnu^2 + m1.snu^2$$

4 Derivatives

$$\left.\begin{aligned} d(snu)/du &= cnu.dnu \\ d(cnu)/du &= -snu.dnu \\ d(dnu)/du &= -m.snu.cnu \end{aligned}\right\} \text{ m constant}$$

5 Degenerate cases

(a) $m = 0.0$, $m1 = 1.0$, $K = \pi/2$
$snu = \sin(u)$, $cnu = \cos(u)$, $dnu = 1.0$

(b) $m = 1.0$, $m1 = 0.0$, K is infinite
$snu = \tanh(u)$, $cnu = dnu = \mathrm{sech}(u)$

6 Addition formulae

If $w = u + v$, and $-K \leqslant w \leqslant +K$,

$$snw = (snu.cnv.dnv + snv.cnu.dnu)/\Delta$$
$$cnw = (cnu.cnv - snu.snv.dnu.dnv)/\Delta$$
$$dnw = (dnu.dnv - m.snu.snv.cnu.cnv)/\Delta$$

where $\Delta = 1.0 - m.snu^2.snv^2$

(contd)

7 Extension to all real values of u

Assuming the functions still satisfy the above addition formulae when the range is extended, putting v = K in the formulae and using the above values of snk, cnk, dnk and simplifying gives

```
sn(u + K, m) = cdu
cn(u + K, m) = -k'.sdu
dn(u + K, m) = k'.ndu
```

Putting u = K in the above gives

```
sn2k = 0.0      cn2k = -1.0      dn2k = 1.0
```

Putting v = 2K in the addition formulae then gives

```
sn(u + 2K, m) = -snu
cn(u + 2K, m) = -cnu
dn(u + 2K, m) = +dnu
```

Thus,

```
sn(u + 4K, m) = snu      cn(u + 4K, m) = cnu
```

Hence snu and cnu are periodic in u with period 4K, and dnu is periodic in u with period 2K.

8 Extension to all real values of m

So far m has been restricted to the range (0.0, 1.0). The functions may, however, be defined and have real values for all real values of m, whereas k is imaginary if m < 0.0, and k' is imaginary if m > 1.0. This was the reason for choosing m as the second parameter of the procedures, rather than k which is perhaps more usual. There is no restriction on the value of m in the library procedures, nor on the value of u.

8.1 m < 0.0

Functions with negative parameters may be defined in terms of functions with parameters in the range (0.0, 1.0) as follows:-

If m < 0.0, let mm = -m/m1, v = u.k'. Then

```
sn(u, m) = sd(v, mm)/k'
cn(u, m) = cd(v, mm)
dn(u, m) = nd(v, mm)
```

8.2 m > 1.0

Functions with parameters greater than unity may be defined in terms of functions with parameters in the range (0.0, 1.0) as follows:-

If m > 1.0, let mm = 1.0/m, v = u.k. Then

```
sn(u, m) = sn(v, mm)/k
cn(u, m) = dn(v, mm)
dn(u, m) = cn(v, mm)
```

9 Complex arguments

The functions may be defined for pure imaginary arguments by using the relations

```
sn(iv, m) = i.sc(v, m1)
cn(iv, m) =   nc(v, m1)
dn(iv, m) =   dc(v, m1)
```

Values for complex arguments (u + iv) may then be obtained if required by using the addition formulae. It will be found that sn is periodic in v with period 2K', and cn and dn are periodic in v with period 4K', where K' = K(k'). Functions with complex arguments have not yet been implemented in the system library.

Many other duplication, integral and other formulae are available, analagous to those for ordinary circular and hyperbolic functions. For further details see, for example, Introduction to Elliptic Functions with Applications, by F Bowman (Dover, 1961).

10 Accuracy

Absolute accuracy (or relative accuracy for small values of u) of all functions is generally of the order of 1 &-11, but may be worse if u is large, in a similar manner to the circular functions.

11 Efficiency

As a substantial part of the work in calculating any function consists of determining constants (used in evaluating the functions) which depend only on m, this is done only once for any particular value of m no matter how many function calls are made, until a different value of m is used. Most of the remainder of the calculation is common to all the functions, so the first time any particular value of u is used the values of sn, cn and dn are calculated and stored and the appropriate value delivered. Any subsequent function calls using the same values of u and m then only involve a trivial amount of calculation, until a new value of u or m is used.

A simple iterative method is used which normally converges in 3 or 4 iterations depending on the value of m, varying from 1 when m is near zero to a maximum of 7 for values of m near unity.

12 Fault displays

Fault display OVERFLOW IN LIBRARY PROC XY may occur in some procedures for certain isolated values of the argument u (especially zero for procedures ns, cs, ds, but unlikely elsewhere), where XY is the name of the procedure concerned. Overflow should not occur for the basic functions sn, cn or dn for any value of m or u, nor for the functions nd, sd or cd provided m <= 1.0.

SUM OF CHEBYSHEV SERIES

```
PROC chebsum = (REAL x, [ ]REAL coeffs) REAL
```

This procedure delivers the sum of the Chebyshev series

$$\sum_{r=0}^{n}{}' c_r T_r(x) \equiv \tfrac{1}{2}c_0 + c_1 T_1(x) + c_2 T_2(x) + \ldots + c_n T_n(x)$$

the prime after the Σ denoting that the first term is halved. The coefficients c_0 to c_n are given by the user in the array parameter "coeffs" which may have any bounds, and $T_r(x)$ is the rth order Chebyshev polynomial most simply defined by the recurrence relations

$$T_0(x) \equiv 1 \qquad T_1(x) \equiv x$$

$$T_r(x) \equiv 2x.T_{r-1}(x) - T_{r-2}(x) \qquad r > 1$$

The procedure uses (n + 2) floating-point multiplications, additions and subtractions to evaluate an nth-order series, using the recurrence relations

$$b_{n+2} = b_{n+1} = 0.0$$

$$b_r = 2x.b_{r+1} - b_{r+2} + c_r \qquad (r = n, n-1, \ldots 0)$$

$$\text{sum} = \tfrac{1}{2}(b_0 - b_2)$$

If floating-point overflow is set during the evaluation, the fault display OVERFLOW IN LIBRARY PROC CHEBSUM occurs.