Faculty of Civil Engineering Belgrade
Master Study COMPUTATIONAL ENGINEERING
Fall semester 2004/2005

LECTURES

LESSON VI

Chapter 6. Special Functions

6.0 Introduction There is nothing particularly special about a special function, except that some person in authority or textbook writer (not the same thing!) has decided to bestow the moniker. Special functions are sometimes called higher transcendental functions (higher than what?) or functions of mathematical physics (but they occur in other fields also) or functions that satisfy certain frequently occurring second-order differential equations (but not all special functions do). One might simply call them useful functions and let it go at that; it is surely only a matter of taste which functions we have chosen to include in this chapter. Good commercially available program libraries, such as NAG or IMSL, contain routines for a number of special functions. These routines are intended for users who will have no idea what goes on inside them. Such state of the art black boxes are often very messy things, full of branches to completely different methods depending on the value of the calling arguments. Black boxes have, or should have, careful control of accuracy, to some stated uniform precision in all regimes. We will not be quite so fastidious in our examples, in part because we want to illustrate techniques from Chapter 5, and in part because we want you to understand what goes on in the routines presented. Some of our routines have an accuracy parameter that can be made as small as desired, while others (especially those involving polynomial fits) give only a certain accuracy, one that we believe serviceable (typically six significant figures or more). We do not certify that the routines are perfect black boxes. We do hope that, if you ever encounter trouble in a routine, you will be able to diagnose and correct the problem on the basis of the information that we have given. In short, the special function routines of this chapter are meant to be used we use them all the time but we also want you to be prepared to understand their inner workings. CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York) [full of useful numerical approximations to a great variety of functions]. IMSL Sfun/Library Users Manual (IMSL Inc., 2500 CityWest Boulevard, Houston TX 77042). NAG Fortran Library (Numerical Algorithms Group, 256 Banbury Road, Oxford OX27DE, U.K.), Chapter S.

Hart, J.F., et al. 1968, Computer Approximations (New York: Wiley). Hastings, C. 1955, Approximations for Digital Computers (Princeton: Princeton University Press). Luke, Y.L. 1975,Mathematical Functions and Their Approximations (New York: Academic Press). 6.1 Gamma Function, Beta Function, Factorials, Binomial Coefficients The gamma function is defined by the integral tz - 1 e-t dt (6.1.1) G(z)= 8 0 When the argument z is an integer, the gamma function is just the familiar factorial function, but offset by one, n!=G(n +1) (6.1.2) The gamma function satisfies the recurrence relation G(z +1)=zG(z) (6.1.3) If the function is known for arguments z¿1 or, more generally, in the half complex plane Re(z) ¿ 1 it can be obtained for z¡1 or Re (z) ¡ 1 by the reflection formula G(1 - z)= p G(z)sin(pz) = pz G(1 + z)sin(pz) (6.1.4) Notice that G(z) has a pole at z =0, and at all negative integer values of z. There are a variety of methods in use for calculating the function G(z) numerically, but none is quite as neat as the approximation derived by Lanczos [1]. This scheme is entirely specific to the gamma function, seemingly plucked from thin air. We will not attempt to derive the approximation, but only state the resulting formula: For certain integer choices of and N , and for certain coefficients c1 ,c2 ,...,cN , the gamma function is given by G(z +1)=(z + + 1 2 e-(z + + 1 2) 2)z + 1 (6.1.5) v2p c0 + c1 z + N + (z¿0) z +1+ c2 z +2 + + cN You can see that this is a sort of take-off on Stirlings approximation, but with a series of corrections that take into account the first few poles in the left complex plane. The constant c0 is very nearly equal to 1. The error term is parametrized by . For =5, N =6, and a certain set of cs, the error is smaller than — — ¡ 2 10-1 0 . Impressed? If not, then perhaps you will be impressed by the fact that (with these

206 Chapter 6. Special Functions Hart, J.F., et al. 1968, Computer Approximations (New York: Wiley). Hastings, C. 1955, Approximations for Digital Computers (Princeton: Princeton University Press). Luke, Y.L. 1975,Mathematical Functions and Their Approximations (New York: Academic Press). 6.1 Gamma Function, Beta Function, Factorials, Binomial Coefficients The gamma function is defined by the integral tz - 1 e-t dt (6.1.1) G(z)= 8 0 When the argument z is an integer, the gamma function is just the familiar factorial function, but offset by one, n!=G(n +1) (6.1.2) The gamma function satisfies the recurrence relation G(z +1)=zG(z) (6.1.3) If the

function is known for arguments z¿1 or, more generally, in the half complex plane Re(z) ¿ 1 it can be obtained for z¡1 or Re (z) ¡ 1 by the reflection formula G(1 - z)= p G(z)sin(pz) = pz G(1 + z)sin(pz) (6.1.4) Notice that G(z) has a pole at z =0, and at all negative integer values of z. There are a variety of methods in use for calculating the function G(z) numerically, but none is quite as neat as the approximation derived by Lanczos [1]. This scheme is entirely specific to the gamma function, seemingly plucked from thin air. We will not attempt to derive the approximation, but only state the resulting formula: For certain integer choices of and N , and for certain coefficients c1 ,c2 ,...,cN , the gamma function is given by G(z +1)=(z + + 1 2 e-(z + + 1 2) 2)z + 1 (6.1.5) v2p c0 + c1 z + N + (z¿0) z +1+ c2 z +2 + + cN You can see that this is a sort of take-off on Stirlings approximation, but with a series of corrections that take into account the first few poles in the left complex plane. The constant c0 is very nearly equal to 1. The error term is parametrized by . For =5, N =6, and a certain set of cs, the error is smaller than — — ¡ 2 10-1 0 . Impressed? If not, then perhaps you will be impressed by the fact that (with these

same parameters) the formula (6.1.5) and bound on apply for the complex gamma function, everywhere in the half complex plane Re z¿0. It is better to implement ln G(x) than G(x), since the latter will overflow many computers floating-point representation at quite modest values of x. Often the gamma function is used in calculations where the large values of G(x) are divided by other large numbers, with the result being a perfectly ordinary value. Such operations would normally be coded as subtraction of logarithms. With (6.1.5) in hand, we can compute the logarithm of the gamma function with two calls to a logarithm and 25 or so arithmetic operations. This makes it not much more difficult than other built-in functions that we take for granted, such as sin x or e x : FUNCTION gammln(xx) REAL gammln,xx Returns the value ln[G(xx)] for xx ¿ 0. INTEGER j DOUBLE PRECISION ser,stp,tmp,x,y,cof(6) Internal arithmetic wi ll be done in double precision, a nicety that you can omit if five-figure accuracy is good enough. SAVE cof,stp DATA cof,stp/76.18009172947146d0,-86.50532032941677d0, * 24.01409824083091d0,-1.231739572450155d0,.1208650973866179d-2, * -.5395239384953d-5,2.5066282746310005d0/ x=xx y=x tmp=x+5.5d0 tmp=(x+0.5d0)*log(tmp)-tmp ser=1.000000000190015d0 do 1 1 j=1,6 y=y+1.d0 ser=ser+cof(j)/y enddo 1 1 gammln=tmp+log(stp return END How shall we write a routine for the factorial function n!? Generally the factorial function will be called for small integer values (for large values it will overflow anyway!), and in most applications the same integer value will be called for many times. It is a profligate waste of computer time to call exp(gammln(n+1.0)) for each required factorial. Better to go back to basics, holding gammln in reserve for unlikely calls: FUNCTION factrl(n) INTEGER n REAL factrl C USES gammln Returns the value n! as a floating-point number. INTEGER j,ntop REAL a(33),gammln Table to be filled in only as required. SAVE ntop,a DATA ntop,a(1)/0,1./ Table initialized with 0! only. if (n.lt.0) then pause negative factorial in factrl else if (n.le.ntop) then Already in table. factrl=a(n+1) else if (n.le.32) then Fill in table up to desired value. do 1 1 j=ntop+1,n

a(j+1)=j*a(j) enddo 1 1 ntop=n factrl=a(n+1) else Larger value than size of table is required. Actually, this big a value is going to overflow on many computers, but no harm in trying. factrl=exp(gammln(n+1.)) endif return END A useful point is that factrl will be exact for the smaller values of n, since floating-point multiplies on small integers are exact on all computers. This exactness will not hold if we turn to the logarithm of the factorials. For binomial coefficients, however, we must do exactly this, since the individual factorials in a binomial coefficient will overflow long before the coefficient itself will. The binomial coefficient is defined by n k = n! k!(n - k)! 0 = k = n (6.1.6) FUNCTION bico(n,k) INTEGER k,n REAL bico C USES factln Returns the binomial coefficient n k as a floating-point number. REAL factln bico=nint(exp(factln(n)-factln(k)-factln(n-k))) return The nearest-integer function cleans up roundoff error for smaller values of n and k. END which uses FUNCTION factln(n) INTEGER n REAL factln C USES gammln Returns ln(n!). REAL a(100),gammln SAVE a DATA a/100*-1./ Initialize the table to negative values. if (n.lt.0) pause negative factorial in factln if (n.le.99) then In range of the table. if (a(n+1).lt.0.) a(n+1)=gammln(n+1.) If not already in the table, put it in. factln=a(n+1) else factln=gammln(n+1.) Out of range of the table. endif return END

209 6.2 Incomplete Gamma Function If your problem requires a series of related binomial coefficients, a good idea is to use recurrence relations, for example n k = n n +1 k = n +1 n - k +1 k + n k - 1 (6.1.7) n k n k +1 = n - k k +1 Finally, turning away from the combinatorial functions with integer valued arguments, we come to the beta function, tz -1 (1 - t)w -1 dt (6.1.8) B(z, w)=B(w, z)= 1 0 which is related to the gamma function by B(z,w)= G(z)G(w) (6.1.9) G(z + w) hence FUNCTION beta(z,w) REAL beta,w,z C USES gammln Returns the value of the beta function B(z, w). REAL gammln beta=exp(gammln(z)+gammln(w)-gammln(z+w)) return END CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Vol-

ume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 6. Lanczos, C. 1964, SIAM Journal on Numerical Analysis, ser. B, vol. 1, pp. 8696. [1] 6.2 Incomplete Gamma Function, Error Function, Chi-Square Probability Function, Cumulative Poisson Function The incomplete gamma function is defined by x P(a, x) = (a, x) e-t ta- 1 dt (a¿0) (6.2.1) G(a) = 1 G(a) 0

209 6.2 Incomplete Gamma Function If your problem requires a series of related binomial coefficients, a good idea is to use recurrence relations, for example n k = n n +1 k = n +1 n - k +1 k + n k - 1 (6.1.7) n k n k +1 = n - k k +1 Finally, turning away from the combinatorial functions with integer valued arguments, we come to the beta function, tz -1 (1 - t)w -1 dt (6.1.8) B(z, w)=B(w, z)= 1 0 which is related to the gamma function by B(z,w)= G(z)G(w) (6.1.9) G(z + w) hence FUNCTION beta(z,w) REAL beta,w,z C USES gammln Returns the value of the beta function B(z, w). REAL gammln beta=exp(gammln(z)+gammln(w)-gammln(z+w)) return END CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 6. Lanczos, C. 1964, SIAM Journal on Numerical Analysis, ser. B, vol. 1, pp. 8696. [1] 6.2 Incomplete Gamma Function, Error Function, Chi-Square Probability Function, Cumulative Poisson Function The incomplete gamma function is defined by x P(a, x) = (a, x) e-t ta- 1 dt (a¿0) (6.2.1) G(a) = 1 G(a) 0

210 Chapter 6. Special Functions 1.0 0.5 1.0 a = 3.0 a = 10 .2 .4 .6 .8 0 02 4 6 8101214 x Figure 6.2.1. The incomplete gamma function P (a, x) for four values of a. It has the limiting values P (a, 0) = 0 and P (a, 8)=1 (6.2.2) The incomplete

gamma function P (a, x) is monotonic and (for a greater than one or so) rises from near-zero to near-unity in a range of x centered on about a - 1, and of width about va (see Figure 6.2.1). The complement of P (a, x) is also confusingly called an incomplete gamma function, 8 Q(a, x) = 1 - P(a, x) = G(a, x) e-tta- 1 dt (a¿0) (6.2.3) G(a) = 1 G(a) x It has the limiting values Q(a, 0) = 1 and Q(a, 8)=0 (6.2.4) The notations P (a, x), (a, x), and G(a, x) are standard; the notation Q(a, x) is specific to this book. There is a series development for (a, x) as follows: 8 G(a) (a, x)=e-x xa G(a +1+n) xn (6.2.5) n =0 One does not actually need to compute a new G(a +1+n) for each n; one rather uses equation (6.1.3) and the previous coefficient.

211 6.2 Incomplete Gamma Function A continued fraction development for G(a, x) is 1 - a 1 2 - a 2 G(a, x)=e-x xa 1 x + 1+ x + 1+ x + (x¿0) (6.2.6) It is computationally better to use the even part of (6.2.6), which converges twice as fast (see 5.2): 1 (1 - a) 2 (2 - a) G(a, x)=e- xxa 1 x +1- a - x +3- a - x +5- a - (x¿0) (6.2.7) It turns out that (6.2.5) converges rapidly for x less than about a +1, while (6.2.6) or (6.2.7) converges rapidly for x greater than about a +1. In these respective regimes each requires at most a few times va terms to converge, and this many only near x = a, where the incomplete gamma functions are varying most rapidly. Thus (6.2.5) and (6.2.7) together allow evaluation of the function for all positive a and x. An extra dividend is that we never need compute a function value near zero by subtracting two nearly equal numbers. The higher-level functions that return P (a, x) and Q(a, x) are FUNCTION gammp(a,x) REAL a,gammp,x C USES gcf,gser Returns the incompl ete gamma function P (a, x). REAL gammcf,gamser,gln if(x.lt.0..or.a.le.0.)pause bad arguments in gammp if(x.lt.a+1.)then Use the series representation. call gser(gamser,a,x,gln) gammp=gamser else Use

the continued fraction representation call gcf(gammcf,a,x,gln) gammp=1.-gammcf and take its complement. endif return END FUNCTION gammq(a,x) REAL a,gammq,x C USES gcf,gser Returns the incomplete gamma function Q(a, x) = 1 - P (a, x). REAL gammcf,gamser,gln if(x.lt.0..or.a.le.0.)pause bad arguments in gammq if(x.lt.a+1.)then Use the series representation call gser(gamser,a,x,gln) gammq=1.-gamser and take its complement. else Use the continued fraction representation. call gcf(gammcf,a,x,gln) gammq=gammcf endif return END

212 Chapter 6. Special Functions The argument gln is returned by both the series and continued fraction procedures containing the value ln G(a); the reason for this is so that it is available to you if you want to modify the above two procedures to give (a, x) and G(a, x), in addition to P (a, x) and Q(a, x) (cf. equations 6.2.1 and 6.2.3). The procedures gser and gcf which implement (6.2.5) and (6.2.7) are SUBROUTINE gser(gamser,a,x,gln) INTEGER ITMAX REAL a,gamser,gln,x,EPS PARAMETER (ITMAX=100,EPS=3.e-7) C USES gammln Returns the incomplete gamma function P (a, x) evaluated by its series representation as gamser. Also returns ln G(a) as gln. INTEGER n REAL ap,del,sum,gammln gln=gammln(a) if(x.le.0.)then if(x.lt.0.)pause x ¡ 0 in gser gamser=0. return endif ap=a sum=1./a del=sum do 1 1 n=1,ITMAX ap=ap+1. del=del*x/ap sum=sum+del if(abs(del).lt.abs(s 1 enddo 1 1 pause a too large, ITMAX too small in gser 1 gamser=sum*exp(-x+a*log(x)-gln) return END SUBROUTINE gcf(gammcf,a,x,gln) INTEGER ITMAX REAL a,gammcf,gln,x,EP PARAMETER (ITMAX=100,EPS=3.e-7,FPMIN=1.e-30) C USES gammln Returns the incomplete gamma function Q(a, x) eval uated by its continued fraction repre- sentation as gammcf. Also returns ln G(a) as gln. Parameters: ITMAX is the maximum allowed number of iterations; EPS is the

relative accu- racy; FPMIN i s a number near the smallest representable oating-point number. INTEGER i REAL an,b,c,d,del,h,gammln gln=gammln(a) b=x+1.-a Set up for evaluating continued fraction by modified Lentzs method (5.2) with b0 =0. c=1./FPMIN d=1./b h=d do 1 1 i=1,ITMAX Iterate to convergence. an=-i*(i-a) b=b+2. d=an*d+b if(abs(d).lt.FPMIN)d=FPMIN c=b+an/c if(abs(c).lt.FPMIN)c=FPMIN d=1./d del=d*c

213 6.2 Incomplete Gamma Function h=h*del if(abs(del-1.).lt.EPS)goto 1 enddo 1 1 pause a too large, ITMAX too small in gcf 1 gammcf=exp(-x+a*log(x)-gln)*h Put factors in front. return END Error Function The error function and complementary error function are special cases of the incomplete gamma function, and are obtained moderately efficiently by the above procedures. Their definitions are x erf(x)= 2 vp e-t2 dt (6.2.8) 0 and 8 vp erfc(x) = 1 - erf(x)= 2 e- t2 dt (6.2.9) x The functions have the following limiting values and symmetries: erf(0) = 0 erf(8)=1 erf(-x)=-erf(x) (6.2.10) erfc(0) = 1 erfc(8)=0 erfc(-x)=2- erfc(x) (6.2.11) They are related to the incomplete gamma functions by erf(x)=P 1 2 ,x2 (x = 0) (6.2.12) and erfc(x)=Q 1 2 ,x2 (x = 0) (6.2.13) Hence we have FUNCTION erf(x) REAL erf,x C USES gammp Returns the error function erf(x). REAL gammp if(x.lt.0.)then erf=-gammp(.5,x**2) else erf=gammp(.5,x**2) endif return END

215 6.3 Exponential Integrals Chi-Square Probability Function P (2 —) is defined as the probability that the observed chi-square for a correct model should be less than a value 2 . (We will discuss the use of this function in Chapter 15.) Its complement Q(2 —) is the probability that the observed chi-square will exceed the value 2 by chance even for a correct model. In both cases is an integer, the number of degrees of freedom. The functions have the limiting values P (0—)=0 P

(8—)=1 (6.2.16) Q(0—)=1 Q(8—)=0 (6.2.17) and the following relation to the incomplete gamma functions, P (2 —)=P 2 = gammp 2 (6.2.18) 2 , 2 2 , 2 Q(2 —)=Q 2 = gammq 2 (6.2.19) 2 , 2 2 , 2 CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapters 6, 7, and 26. Pearson, K. (ed.) 1951, Tables of the Incomplete Gamma Function (Cambridge: Cambridge University Press). 6.3 Exponential Integrals The standard definition of the exponential integral is e-x t En (x)= 8 tn dt, x ¿ 0,n=0, 1,... (6.3.1) 1 The function defined by the principal value of the integral et e- t Ei(x)=- 8 t dt, x ¿ 0(6.3.2) t dt = x -x - 8 is also called an exponential integral. Note that Ei(-x) is related to -E1 (x) by analytic continuation. The function En (x) is a special case of the incomplete gamma function En (x)=xn- 1 G(1 - n, x)(6.3.3)

216 Chapter 6. Special Functions We can therefore use a similar strategy for evaluating it. The continued fraction just equation (6.2.6) rewritten converges for all x¿0: n 1 n +1 2 En (x)=e-x 1 x + 1+ x + 1+ x + (6.3.4) We use it in its more rapidly converging even form, 1 n 2(n +1) En (x)=e-x 1 x + n - x + n +2- x + n +4- (6.3.5) The continued fraction only really converges fast enough to be useful for x ¿ 1. For 0 ¡x¡ 1, we can use the series representation 8 (-x)m En (x)= (-x)n -1 (n - 1)! [- ln x + (n)] - (m - n +1)m! (6.3.6) m= 0 m=n -1 The quantity (n) here is the digamma function, given for integer arguments by n- 1 1 (1) = - , (n)=- + m (6.3.7) m =1 where =0.5772156649...is Eulers constant. We evaluate the expression (6.3.6) in order of ascending powers of x: En (x)=- 1 (1 - n) - x (-1)(n - 2)! (2 - n) 1 + x2 (3 - n)(1 2) -+ (-x)n -2 + (-x)n -1 2 (n +1)!+ (n - 1)! [- ln x + (n)] - (-x)n 1 n! +

(-x)n +1 (6.3.8) The first square bracket is omitted when n =1. This method of evaluation has the advantage that for large n the series converges before reaching the term containing (n). Accordingly, one needs an algorithm for evaluating (n) only for small n, n ¡ 20 40. We use equation (6.3.7), although a table look-up would improve efficiency slightly. Amos [1] presents a careful discussion of the truncation error in evaluating equation (6.3.8), and gives a fairly elaborate termination criterion. We have found that simply stopping when the last term added is smaller than the required tolerance works about as well. Two special cases have to be handled separately: E0 (x)= e-x x (6.3.9) En (0) = 1 n - 1 ,n¿1

217 6.3 Exponential Integrals The routine expint allows fast evaluation of En (x) to any accuracy EPS within the reach of your machines word length for oating-point numbers. The only modification required for increased accuracy is to supply Eulers constant with enough significant digits. Wrench [2] can provide you with the first 328 digits if necessary! FUNCTION expint(n,x) INTEGER n,MAXIT REAL expint,x,EPS,FPMIN,EULER PARAMETER (MAXIT=100,EPS=1.e-7,FPMIN=1.e-30,EULER=.5772156649) Evaluates the exponential integral En (x). Parameters: MAXIT is the maximum allowed number of iterations; EPS is the desired relative error, not smaller than the machine precision; FPMIN is a number near the smallest representable oating-point number; EULER is Eulers constant . INTEGER i,ii,nm1 REAL a,b,c,d,del,fact,h,psi nm1=n-1 if(n.lt.0.or.x.lt.0..or.(x.eq.0..and.(n.eq.0.or pause bad arguments in expint else if(n.eq.0)then Special case. expint=exp(-x)/x else if(x.eq.0.)then Another special case. expint=1./nm1 else if(x.gt.1.)then Lentzs algorithm (5.2). b=x+n c=1./FPMIN d=1./b h=d do 1 1 i=1,MAXIT a=-i*(nm1+i) b=b+2. d=1./(a*d+b) Denominators cannot

be zero. c=b+a/c del=c*d h=h*del if(abs(del-1.).lt.EPS)then expint=h*exp(-x) return endif enddo 1 1 pause continued fraction failed in expint else Evaluate series. if(nm1.ne.0)then Setfirstterm. expint=1./nm1 else expint=-log(x)-EULER endif fact=1. do 1 3 i=1,MAXIT fact=-fact*x/i if(i.ne.nm1)then del=-fact/(i-nm1) else psi=-EULER Compute (n). do 1 2 ii=1,nm1 psi=psi+1./ii enddo 1 2 del=fact*(-log(x)+psi) endif expint=expint+del if(abs(del).lt.abs(expint)*EPS) return enddo 1 3

218 Chapter 6. Special Functions pause series failed in expint endif return END A good algorithm for evaluating Ei is to use the power series for small x and the asymptotic series for large x. The power series is Ei(x)= +lnx + x 2 2! + (6.3.10) 1 1! + x2 where is Eulers constant. The asymptotic expansion is Ei(x) ex x 1+ 1! x + 2! x2 + (6.3.11) The lower limit for the use of the asymptotic expansion is approximately — ln EPS—, where EPS is the required relative error. FUNCTION ei(x) INTEGER MAXIT REAL ei,x,EPS,EULER,FPMIN PARAMETER (EPS=6.e-8,EULER=.57721566,MAXIT=100,FPMIN=1.e-30) Computes the exponential integral Ei(x) for x¿0. Parameters: EPS is the relative error, or absolute error near the zero of Ei at x =0.3725; EULER is Eulers constant ; MAXIT is the maximum number of iterations allowed; FPMIN is a number near the smallest representable oating-point number. INTEGER k REAL fact,prev,sum,term if(x.le.0.) pause bad argument in ei if(x.lt.FPMIN)then Special case: avoid fail ure of convergence test be- cause of under ow. ei=log(x)+EULER else if(x.le.-log(EPS))then Use power series. sum=0. fact=1. do 1 1 k=1,MAXIT fact=fact*x/k term=fact/k sum=sum+term if(term.lt.EPS*sum)goto 1 enddo 1 1 pause series failed in ei 1 ei=sum+log(x)+EULER else Use asymptotic series. sum=0.

Start with second term. term=1. do 1 2 k=1,MAXIT prev=term term=term*k/x if(term.lt.EPS)goto 2 Since final sum is greater than one, term itself ap- proximates the relative error. if(term.lt.prev)then sum=sum+term Still convergi ng: add new term. else sum=sum-prev Diverging: subtract previous term and exit. goto 2 endif enddo 1 2 2 ei=exp(x)*(1.+sum)/x endif

219 6.4 Incomplete Beta Function, Students Distribution, F-Distribution, Cumulative Binomial Distribution return END CITED REFERENCES AND FURTHER READING: Stegun, I.A., and Zucker, R. 1974, Journal of Research of the National Bureau of Standards, vol. 78B, pp. 199216; 1976, op. cit., vol. 80B, pp. 291311. Amos D.E. 1980, ACM Transactions on Mathematical Software, vol. 6, pp. 365377 [1]; also vol. 6, pp. 420428. Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 5. Wrench J.W. 1952, Mathematical Tables and Other Aids to Computation, vol. 6, p. 255. [2] 6.4 Incomplete Beta Function, Students Distribution, F-Distribution, Cumulative Binomial Distribution The incomplete beta function is defined by x Ix (a, b) = Bx (a, b) ta-1 (1 - t)b -1 dt (a, b ¿ 0) (6.4.1) B(a, b) = 1 B(a, b) 0 It has the limiting values I0 (a, b)=0 I1 (a, b)=1 (6.4.2) and the symmetry relation Ix (a, b)=1- I1 -x (b, a)(6.4.3) If a and b are both rather greater than one, then Ix (a, b) rises from near-zero to near-unity quite sharply at about x = a/(a + b). Figure 6.4.1 plots the function for several pairs (a, b). The incomplete beta function has a series expansion 8 B (a +1,n +1) Ix (a, b)= xa (1 - x)b aB(a, b) 1+ B(a + b, n +1)xn +1 , (6.4.4) n =0 but this does not prove to be very useful in its numerical evaluation. (Note, however, that the beta functions

in the coefficients can be evaluated for each value of n with just the previous value and a few multiplies, using equations 6.1.9 and 6.1.3.) The continued fraction representation proves to be much more useful, 1 d1 d2 Ix (a, b)= xa (1 - x)b aB(a, b) 1+ 1+ 1+ 1+ (6.4.5)

219 6.4 Incomplete Beta Function, Students Distribution, F-Distribution, Cumulative Binomial Distribution return END CITED REFERENCES AND FURTHER READING: Stegun, I.A., and Zucker, R. 1974, Journal of Research of the National Bureau of Standards, vol. 78B, pp. 199216; 1976, op. cit., vol. 80B, pp. 291311. Amos D.E. 1980, ACM Transactions on Mathematical Software, vol. 6, pp. 365377 [1]; also vol. 6, pp. 420428. Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 5. Wrench J.W. 1952, Mathematical Tables and Other Aids to Computation, vol. 6, p. 255. [2] 6.4 Incomplete Beta Function, Students Distribution, F-Distribution, Cumulative Binomial Distribution The incomplete beta function is defined by x Ix (a, b) = Bx (a, b) ta-1 (1 - t)b -1 dt (a, b ¿ 0) (6.4.1) B(a, b) = 1 B(a, b) 0 It has the limiting values I0 (a, b)=0 I1 (a, b)=1 (6.4.2) and the symmetry relation Ix (a, b)=1- I1 -x (b, a)(6.4.3) If a and b are both rather greater than one, then Ix (a, b) rises from near-zero to near-unity quite sharply at about x = a/(a + b). Figure 6.4.1 plots the function for several pairs (a, b). The incomplete beta function has a series expansion 8 B (a +1,n +1) Ix (a, b)= xa (1 - x)b aB(a, b) 1+ B(a + b, n +1)xn +1 , (6.4.4) n =0 but this does not prove to be very useful in its numerical evaluation. (Note, however, that the beta functions in the coefficients can be evaluated for each value of n with just the previous value and a few multiplies, using equations 6.1.9

and 6.1.3.) The continued fraction representation proves to be much more useful, 1 d1 d2 Ix (a, b)= xa (1 - x)b aB(a, b) 1+ 1+ 1+ 1+ (6.4.5)

220 Chapter 6. Special Functions 1 (0.5,5.0) (8.0,10.0) (1.0,3.0) (0.5,0.5) .2 .4 .6 .8 (5.0,0.5) 0 0 .2 .4 .6 1 .8 x Figure 6.4.1. The incomplete beta function Ix (a, b) for five different pairs of (a, b). Notice that the pairs (0.5, 5.0) and (5.0, 0.5) are symmetrically related as indicated in equation (6.4.3). where d2 m+1 = - (a + m)(a + b + m)x (a +2m)(a +2m +1) (6.4.6) d2 m = m(b - m)x (a +2m - 1)(a +2m) This continued fraction converges rapidly for x¡(a +1)/(a + b +2), taking in the worst case O(max(a, b)) iterations. But for x¿(a +1)/(a + b +2)we can just use the symmetry relation (6.4.3) to obtain an equivalent computation where the continued fraction will also converge rapidly. Hence we have FUNCTION betai(a,b,x) REAL betai,a,b,x C USES betacf,gammln Returns the incomplete beta function Ix(a, b). REAL bt,betacf,gammln if(x.lt.0..or.x.gt.1.)pause bad argument x in betai if(x.eq.0..or.x.eq.1.)then bt=0. else Factors in front of the continued fraction. bt=exp(gammln(a+b)-gammln(a)-gammln(b) * +a*log(x)+b*log(1.-x)) endif if(x.lt.(a+1.)/(a+ Use continued fraction directly.

221 6.4 Incomplete Beta Function, Students Distribution, F-Distribution, Cumulative Binomial Distribution betai=bt*betacf(a,b,x)/a return else betai=1.-bt*betacf(b,a,1.-x)/b Use continued fraction after making the symme- try transformation. return endif END which utilizes the continued fraction evaluation routine FUNCTION betacf(a,b,x) INTEGER MAXIT REAL betacf,a,b,x,EPS,FPMIN PARAMETER (MAXIT=100,EPS=3.e-7,FPMIN=1.e-30) Used by betai: Eval uates continued fraction for incomplete beta function by modified Lentzs method (5.2). INTEGER m,m2 REAL

aa,c,d,del,h,qab,qam,qap qab=a+b These qs will be used in factors that occur in the coe cients (6.4.6). qap=a+1. qam=a-1. c=1. First step of Lentzs method. d=1.-qab*x/qap if(abs(d).lt.FPMIN)d=FPMIN d=1./d h=d do 1 1 m=1,MAXIT m2=2*m aa=m*(b-m)*x/((qam+m2)*(a+m2)) d=1.+aa*d One step (the even one) of the recurrence. if(abs(d).lt.FPMIN)d=FPMIN c=1.+aa/c if(abs(c).lt.FPMIN)c=FPMIN d=1./d h=h*d*c aa=-(a+m)*(qab+m)*x/((a+m2)*(qap+m2)) d=1.+aa*d Next step of the recurrence (the odd one). if(abs(d).lt.FPMIN)d=FPMIN c=1.+aa/c if(abs(c).lt.FPMIN)c=FPMIN d=1./d del=d*c h=h*del if(abs(del-1.).lt.EPS)goto 1 Are we done? enddo 1 1 pause a or b too big, or MAXIT too small in betacf 1 betacf=h return END Students Distribution Probability Function Students distribution, denoted A(t—), is useful in several statistical contexts, notably in the test of whether two observed distributions have the same mean. A(t—) is the probability, for degrees of freedom, that a certain statistic t (measuring the observed difference of means) would be smaller than the observed value if the means were in fact the same. (See Chapter 14 for further details.) Two means are

222 Chapter 6. Special Functions significantly different if, e.g., A(t—) ¿ 0.99. In other words, 1 - A(t—) is the significance level at which the hypothesis that the means are equal is disproved. The mathematical definition of the function is 2 dx (6.4.7) t - + 1 A(t—)= 1 1+ x2

1 / 2 B(1 -t 2) 2 , Limiting values are A(0—)=0 A(8—)=1 (6.4.8) A(t—) is related to the incomplete beta function Ix (a, b) by

A(t—)=1- I 2 , 1 2 (6.4.9) +t 2 So, you can use (6.4.9) and the above routine betai to evaluate the function. F-Distribution Probability Function This function occurs in the statistical test of whether two observed samples have the same variance. A certain statistic F , essentially the ratio of the observed dispersion of the first sample to that of the second one, is calculated. (For further details, see Chapter 14.) The probability that F would be as large as it is if the first samples underlying distribution actually has smaller variance than the secondsis denoted Q(F — 1 , 2), where 1 and 2 are the number of degrees of freedom in the first and second samples, respectively. In other words, Q(F — 1 , 2) is the significance level at which the hypothesis 1 has smaller variance than 2 can be rejected. A small numerical value implies a very significant rejection, in turn implying high confidence in the hypothesis 1 has variance greater or equal to 2. Q(F— 1 , 2) has the limiting values Q(0— 1 , 2)=1 Q(8— 1 , 2)=0 (6.4.10) Its relation to the incomplete beta function Ix (a, b) as evaluated by betai above is 2 Q(F — 1 , 2)=I 2 2 (6.4.11) 2 , 1 2 + 1 F Cumulative Binomial Probability Distribution Suppose an event occurs with probability p per trial. Then the probability P of its occurring k or more times in n trials is termed a cumulative binomial probability, and is related to the incomplete beta function Ix (a, b) as follows: n n P = j pj (1 - p)n -j = Ip(k,n - k +1) (6.4.12) j =k

6.5 Bessel Functions of Integer Order 223 For n larger than a dozen or so, betai is a much better way to evaluate the sum in (6.4.12) than would be the straightforward sum with concurrent computation of the binomial coefficients. (For n smaller than a dozen, either method is acceptable.) CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapters 6 and 26. Pearson, E., and Johnson, N. 1968, Tables of the Incomplete Beta Function (Cambridge:

Cam- bridge University Press). 6.5 Bessel Functions of Integer Order This section and the next one present practical algorithms for computing various kinds of Bessel functions of integer order. In 6.7 we deal with fractional order. In fact, the more complicated routines for fractional order work fine for integer order too. For integer order, however, the routines in this section (and 6.6) are simpler and faster. Their only drawback is that they are limited by the precision of the underlying rational approximations. For full double precision, it is best to work with the routines for fractional order in 6.7. For any real , the Bessel function J (x) can be defined by the series representation 8 (- 1 4 x2)k J (x)= 1 2 x k!G(+ k +1) (6.5.1) k=0 The series converges for all x, but it is not computationally very useful for x 1. For not an integer the Bessel function Y (x) is given by (6.5.2) Y (x)= J (x)cos(p) - J- (x) sin(p) The right-hand side goes to the correct limiting value Y n (x) as goes to some integer n, but this is also not computationally useful. For arguments x¡ , both Bessel functions look qualitatively like simple power laws, with the asymptotic forms for 0 ¡x 1 2 x

= 0 J (x) 1 G(+1) (6.5.3) Y0 (x) 2 p ln(x) 1 2 x - ¿0 Y (x) -G( ) p

6.5 Bessel Functions of Integer Order 223 For n larger than a dozen or so, betai is a much better way to evaluate the sum in (6.4.12) than would be the straightforward sum with concurrent computation of the binomial coefficients. (For n smaller than a dozen, either method is acceptable.) CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapters 6 and 26. Pearson, E., and Johnson,

N. 1968, Tables of the Incomplete Beta Function (Cambridge: Cam- bridge University Press). 6.5 Bessel Functions of Integer Order This section and the next one present practical algorithms for computing various kinds of Bessel functions of integer order. In 6.7 we deal with fractional order. In fact, the more complicated routines for fractional order work fine for integer order too. For integer order, however, the routines in this section (and 6.6) are simpler and faster. Their only drawback is that they are limited by the precision of the underlying rational approximations. For full double precision, it is best to work with the routines for fractional order in 6.7. For any real , the Bessel function J (x) can be defined by the series representation 8 (- 1 4 x2)k J (x)= 1 2 x k!G(+ k +1) (6.5.1) k=0 The series converges for all x, but it is not computationally very useful for x 1. For not an integer the Bessel function Y (x) is given by (6.5.2) Y (x)= J (x)cos(p) - J- (x) sin(p) The right-hand side goes to the correct limiting value Y n (x) as goes to some integer n, but this is also not computationally useful. For arguments x¡ , both Bessel functions look qualitatively like simple power laws, with the asymptotic forms for 0 ¡x 1 2 x

= 0 J (x) 1 G(+1) (6.5.3) Y0 (x) 2 p ln(x) 1 2 x - ¿0 Y (x) -G( ) p

224 Chapter 6. Special Functions 1 J0 J1 J2 J3 .5 0 - .5 Y0 Y1 Y2 - 1 -1.5 - 2 246810 0 x Figure 6.5.1. Bessel functions J0 (x) through J3 (x) and Y0 (x) through Y2 (x). For x¿ , both Bessel functions look qualitatively like sine or cosine waves whose amplitude decays as x-1 / 2 . The asymptotic forms for x are J (x) 2 px cos x - 1 2 p - 1 4 p (6.5.4) Y (x) 2 px sin x - 1 2 p - 1 4 p In the transition region where x , the typical amplitudes of the Bessel functions are on the order 1 J ( ) 21 /3 1 /3 0 .4473 32 / 3 G(2 1 /3 3) (6.5.5) 1 Y ( ) - 21

/3 1 /3 -0.7748 31 /6 G(2 1 /3 3) which holds asymptotically for large . Figure 6.5.1 plots the first few Bessel functions of each kind. The Bessel functions satisfy the recurrence relations Jn +1 (x)= 2n x Jn (x) - Jn - 1 (x)(6.5.6) and Yn+ 1 (x)= 2n x Yn (x) - Yn -1 (x)(6.5.7) As already mentioned in 5.5, only the second of these (6.5.7) is stable in the direction of increasing n for x¡n. The reason that (6.5.6) is unstable in the

6.5 Bessel Functions of Integer Order 225 direction of increasing n is simply that it is the same recurrence as (6.5.7): A small amount of polluting Y n introduced by roundoff error will quickly come to swamp the desired Jn , according to equation (6.5.3). A practical strategy for computing the Bessel functions of integer order divides into two tasks: first, how to compute J 0 ,J1 ,Y0 , and Y1 , and second, how to use the recurrence relations stably to find other Js and Y s. We treat the first task first: For x between zero and some arbitrary value (we will use the value 8), approximate J0 (x) and J1 (x) by rational functions in x. Likewise approximate by rational functions the regular part of Y0 (x) and Y1 (x),defined as Y0 (x) - 2 x (6.5.8) p J0 (x)ln(x) and Y1 (x) - 2 p J1 (x)ln(x) - 1 For 8 ¡x¡8, use the approximating forms (n =0, 1) 8 8 Jn (x)= 2 x sin(Xn) (6.5.9) px Pn x cos(Xn) - Qn 8 8 Yn (x)= 2 x cos(Xn) (6.5.10) px Pn x sin(Xn)+Qn where Xn = x - 2n +1 4 p (6.5.11) and where P0 ,P1 ,Q0 , and Q1 are each polynomials in their arguments, for 0 ¡ 8/x ¡ 1. The Ps are even polynomials, the Qs odd. Coefficients of the various rational functions and polynomials are given by Hart [1], for various levels of desired accuracy. A straightforward implementation is FUNCTION bessj0(x) REAL bessj0,x Returns the Bessel function J0 (x) for any real x. REAL ax,xx,z DOUBLE PRECISION p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4, * r5,r6,s1,s2,s3,s4,s5,s6,y Well accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6, * s1,s2,s3,s4,s5,s6 DATA p1,p2,p3,p4,p5/1.d0,-.1098628627d-2,.2734510407d-4, * -.2073370639d-5,.2093887211d-6/, q1,q2,q3,q4,q5/-.1562499995d-1, * .1430488765d-3,-.6911147651d-5,.7621095161d-6,-.934945152d-7/ DATA r1,r2,r3,r4,r5,r6/57568490574.d0,-13362590354. * -11214424.18d0,77392.33017d0,-184.9052456d0/, * s1,s2,s3,s4,s5,s6/575 * 9494680.718d0,59272.64853d0,267.8532712d0,1.d0/ if(abs(x).lt.8.)then Direct rational function fit. y=x**2 bessj0=(r1+y*(r2+y*(r3+y*(r4+y* * /(s1+y*(s2+y*(s3+y*(s4+y*(s5+y*s6))))) else Fitting function (6.5.9). ax=abs(x) z=8./ax y=z**2 xx=ax-.785398164 bessj0=sqrt(.636619772/ax)*(cos(xx)*(p1+y*(p2+y*(p3+y*(p4+y

226 Chapter 6. Special Functions * *p5))))-z*sin(xx)*(q1+y*(q2+y*(endif return END FUNCTION bessy0(x) REAL bessy0,x C USES bessj0 Returns the Bessel function Y0 (x) for positive x. REAL xx,z,bessj0 DOUBLE PRECISION p1,p2,p3,p4,p5,q1, * q2,q3,q4,q5,r1,r2,r3,r4, * r5,r6,s1,s2,s3,s4,s5,s6,y Wel l accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5 * r5,r6,s1,s2,s3,s4,s5,s6 DATA p1,p2,p3,p4,p5/1.d0,-.1098628627d-2,.2734510407d-4, * -.2073370639d-5,.2093887211d-6/, q1,q2,q3,q4,q5/-.1562499995d-1, * .1430488765d-3,-.6911147651d-5,.7621095161d-6,-.934945152d-7/ DATA r1,r2,r3,r4,r5,r6/-2957821389.d0,7062834065.d0 512359803.6d0, * 10879881.29d0,-86327.92757d0,228.4622733d0/, * s1,s2,s3,s4,s5,s6/40076544269.d0,745249964.8d0, * 7189466.438d0,4744 if(x.lt.8.)then Rational function approximation of (6.5.8). y=x**2 bessy0=(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y * *(s3+y*(s4+y*(s5+y*s6)))))+.636619772*bessj0(x)*log(x) else Fitting function (6.5.10). z=8./x y=z**2 xx=x-.785398164 bessy0=sqrt(.636619772/x)*(sin(xx)*(p1+y*(p2+y*(p3+y*(p4+y* * p5))))+z*cos(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5))))) endif return END FUNCTION bessj1(x) REAL bessj1,x Returns the Bessel function J1 (x) for any real x. REAL ax,xx,z DOUBLE PRECISION p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4, *

r5,r6,s1,s2,s3,s4,s5,s6,y Well accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4,r5,r6, * s1,s2,s3,s4,s5,s6 DATA r1,r2,r3,r4,r5,r6/72362614232.d0,-7895059235.d0 * -2972611.439d0,15704.48260d0,-30.16036606d0/, * s1,s2,s3,s4,s5,s6/144 * 18583304.74d0,99447.43394d0,376.9991397d0,1.d0/ DATA p1,p2,p3,p4, 2,-.3516396496d-4,.2457520174d-5, * -.240337019d-6/, q1,q2,q3,q4,q5/.04 .2002690873d-3, * .8449199096d-5,-.88228987d-6,.105787412d-6/ if(abs(x).lt.8.)then Direct rational approximation. y=x**2 bessj1=x*(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6))))) * /(s1+y*(s2+y*(s3 else Fitting function (6.5.9). ax=abs(x) z=8./ax y=z**2 xx=ax-2.356194491

6.5 Bessel Functions of Integer Order 227 bessj1=sqrt(.636619772/ax) * *p5))))-z*sin(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5))))) * *sign(1.,x) endif return END FUNCTION bessy1(x) REAL bessy1,x C USES bessj1 Returns the Bessel function Y1 (x) for positive x. REAL xx,z,bessj1 DOUBLE PRECISION p1,p2,p3,p4,p5,q1,q2,q3,q4,q5, * r5,r6,s1,s2,s3,s4,s5,s6,s7,y Wel l accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,q1,q2,q3,q4,q5,r1,r2,r3,r4, * r5,r6,s1,s2,s3,s4,s5,s6,s7 DATA p1,p2,p3,p4,p5/1.d0,.183105d-2,-.3516396496d-4,.2457520174d-5, * -.240337019d-6/, q1,q2,q3,q4,q5/.04 .2002690873d-3, * .8449199096d-5,-.88228987d-6,.105787412d-6/ DATA r1,r2,r3,r4,r5,r6/-.4900604943d13,.1275274390d13,-.5153438139d11, * .7349264551d9,-.4237922726d7,.8511937935d4/, * s1,s2,s3,s4,s5,s6,s7/.2499580570d14,.4244419664d12, * .3733650367d10, if(x.lt.8.)then Rational function approximation of (6.5.8). y=x**2 bessy1=x*(r1+y*(r2+y*(r3+y*(r4+y*(r5+y*r6)))))/(s1+y*(s2+y* * (s3+y*(s4+y*(s5+y*(s6+y*s7))))))+.636619772 * *(bessj1(x)*log(x)-1./x) else Fitting function (6.5.10). z=8./x y=z**2 xx=x-2.356194491 bessy1=sqrt(.636619772/x)*(sin(xx)*(p1+y*(p2+y*(p3+y* * *p5))))+z*cos(xx)*(q1+y*(q2+y*(q3+y*(q4+y*q5))))) endif return END We now turn to the second task, namely how to use the recurrence formulas (6.5.6) and (6.5.7) to get the Bessel functions Jn (x) and Yn (x) for n = 2. The latter of these is straightforward, since its upward recurrence is always stable: FUNCTION bessy(n,x) INTEGER n REAL bessy,x C USES bessy0,bessy1 Returns the Bessel function Yn(x) for positive x and n = 2. INTEGER j REAL by,bym,byp,tox,bessy0,bessy1 if(n.lt.2)pause bad argument n in bessy tox=2./x by=bessy1(x) Starting values for the recurrence. bym=bessy0(x) do 1 1 j=1,n-1 Recurrence (6.5.7). byp=j*tox*by-bym bym=by by=byp enddo 1 1 bessy=by return END

6.6 Modified Bessel Functions of Integer Order 229 bjp=bjp*BIGNI bessj=bessj*BIGNI sum=sum*BIGNI endif if(jsum.ne.0)sum=sum+bj Accumulate the sum. jsum=1-jsum Change0to1orviceversa. if(j.eq.n)bessj=bjp Save the unnormalized answer. enddo 1 2 sum=2.*sum-bj Compute (5.5.16) bessj=bessj/sum and use it to normalize the answer. endif if(x.lt.0..and.mod(n,2).eq.1)bessj=-bessj return END CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 9. Hart, J.F., et al. 1968, Computer Approximations (New York: Wiley), 6.8, p. 141. [1] 6.6 Modified Bessel Functions of Integer Order The modified Bessel functions In (x) and Kn (x) are equivalent to the usual Bessel functions Jn and Yn evaluated for purely imaginary arguments. In detail, the relationship is In (x)=(-i)n Jn (ix) (6.6.1) Kn (x)= p 2 in +1 [Jn (ix)+iYn (ix)] The particular choice of prefactor and of the linear combination of J n and Yn to form Kn are simply choices that make the functions real-valued for real arguments x. For small arguments x n, both In (x) and Kn (x) become, asymptotically, simple powers of their argument n n = 0 x In (x) 1 n! 2 (6.6.2) K0 (x) -ln(x) -n n¿0 x Kn (x) (n - 1)! 2 2 These expressions are virtually identical to those for Jn (x) and Yn (x) in this region, except for the factor of -2/p difference between Y n (x) and Kn (x). In the region

230 Chapter 6. Special Functions I0 K0 K1 K2 I1 I2 I3 0 1 2 3 4 01234 x Figure 6.6.1. Modified Bessel functions I0 (x) through I3 (x), K0 (x) through K2 (x). x n, however, the modified functions have quite different behavior than the Bessel functions, v In (x) 1 2px exp(x) (6.6.3) v2px exp(-x) Kn (x) p The modified functions evidently have exponential rather than sinusoidal be- havior for large arguments (see Figure 6.6.1). The smoothness of the modified Bessel functions, once the exponential factor is removed, makes a simple polynomial approximation of a few terms quite suitable for the functions I 0 , I1 , K0 , and K1 . The following routines, based on polynomial coefficients given by Abramowitz and Stegun [1], evaluate these four functions, and will provide the basis for upward recursion for n¿1 when x¿n. FUNCTION bessi0(x) REAL bessi0,x Returns the modified Bessel functi on I0 (x) for any real x. REAL ax DOUBLE PRECISION p1,p2,p3,p4,p5,p6,p7,q1,q2,q3,q4,q5,q6,q7, * q8,q9,y Accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,p6,p7,q1,q2,q3,q4,q5,q6,q7,q8,q9 DATA p1,p2,p3,p4,p5,p6 * 0.2659732d0,0.360768d-1,0.45813d-2/ DATA q1,q2,q3,q4,q5,q6,q7,q8,q9 1, * 0.225319d-2,-0.157565d-2,0.916281d-2,-0.2057706d-1, * 0.2635537d- 1,-0.1647633d-1,0.392377d-2/

6.6 Modified Bessel Functions of Integer Order 231 if (abs(x).lt.3.75) then y=(x/3.75)**2 bessi0=p1+y*(p2+y*(p3+y*(p4+y* else ax=abs(x) y=3.75/ax bessi0=(exp(ax)/sqrt(ax))*(q1+y*(q2+y*(q3- * +y*(q5+y*(q6+y*(q7+y*(q8+y*q9)))))))) endif return END FUNCTION bessk0(x) REAL bessk0,x C USES bessi0 Returns the modified Bessel function K0 (x) for positive real x. REAL bessi0 DOUBLE PRECISION p1,p2,p3,p4,p5,p6,p7,q1,

* q2,q3,q4,q5,q6,q7,y Accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,p6,p7,q1,q2,q3,q4,q5,q6,q7 DATA p1,p2,p3,p4,p5,p6,p7/-0.57721566d0,0.42278420d0,0.23069756d0, * 0.3488590d-1,0.262698d-2,0.10750d-3,0.74d-5/ DATA q1,q2,q3,q4,q5,q6 0.7832358d-1,0.2189568d-1, * -0.1062446d-1,0.587872d-2,-0.251540d-2,0.53208d-3/ if (x.le.2.0) then Polynomial fit. y=x*x/4.0 bessk0=(-log(x/2.0)*bessi0(x))+(p1+y*(p2+y*(p3+ * y*(p4+y*(p5+y* else y=(2.0/x) bessk0=(exp(-x)/sqrt(x))*(q1+y*(q2+y*(q3+ * y*(q4+y*(q5+y*(q6+y*q7)))))) endif return END FUNCTION bessi1(x) REAL bessi1,x Returns the modified Bessel functi on I1 (x) for any real x. REAL ax DOUBLE PRECISION p1,p2,p3,p4,p5,p6,p7,q1,q2,q3,q4,q5,q6,q7, * q8,q9,y Accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,p6,p7,q1 DATA p1,p2,p3,p4,p5,p6,p7/0.5d0,0.87890594d0,0.51498869d0, * 0.15084934d0,0.2658733d-1,0.301532d-2,0.32411d-3/ DATA q1,q2,q3,q4,q5,q6,q7,q8,q9/0.39894228d0,-0.3988024d-1, * -0.362018d-2,0.163801d-2,-0.1031555d-1,0.2282967d-1, * -0.2895312d-1,0.1787654d-1,-0.420059d-2/ if (abs(x).lt.3.75) then Polynomial fit. y=(x/3.75)**2 bessi1=x*(p1+y*(p2+y*(p3+y*(p4+y*(p5+y*(p6+y*p7)))))) else ax=abs(x) y=3.75/ax bessi1=(exp(ax)/sqrt(ax))*(q1+y*(q2+y*(q3 * y*(q5+y*(q6+y*(q7+y*(q8+y*q9)))))))) if(x.lt.0.)bessi1=-bessi1 endif return END

232 Chapter 6. Special Functions FUNCTION bessk1(x) REAL bessk1,x C USES bessi1 Returns the modified Bessel function K1 (x) for positive real x. REAL bessi1 DOUBLE PRECISION p1,p2,p3,p4,p5,p6,p7,q1, * q2,q3,q4,q5,q6,q7,y Accumulate polynomials in double precision. SAVE p1,p2,p3,p4,p5,p6,p7 DATA p1,p2,p3,p4,p5,p6,p7/1.0d0,0.15443144d0,-0.67278579d0, * -0.18156897d0,-0.1919402d-1,-0.110404d-2,-0.4686d-4/ DATA q1,q2,q3,q4,q5,q6,q7/1.25331414d0,0.23498619d0,-0.3655620d-1, * 0.1504268d-1,-0.780353d-2,0.325614d-2,-0.68245d-3/ if (x.le.2.0) then Po lynomial fit. y=x*x/4.0 bessk1=(log(x/2.0)*bessi1(x))+(1.0/x)*

* y*(p3+y*(p4+y*(p5+y*(p6+y*p7)))))) else y=2.0/x bessk1=(exp(-x)/sqrt(x))*(q1+y*(q2+y*(q3+ * y*(q4+y*(q5+y*(q6+y*q7)))))) endif return END The recurrence relation for In (x) and Kn (x) is the same as that for Jn (x) and Yn (x) provided that ix is substituted for x. This has the effect of changing a sign in the relation, In +1 (x)=- 2n x In (x)+In - 1 (x) (6.6.4) Kn+ 1 (x)=+ 2n x Kn (x)+Kn -1 (x) These relations are always unstable for upward recurrence. For K n , itself growing, this presents no problem. For In , however, the strategy of downward recursion is therefore required once again, and the starting point for the recursion may be chosen in the same manner as for the routine bessj. The only fundamental difference is that the normalization formula for I n (x) has an alternating minus sign in successive terms, which again arises from the substitution of ix for x in the formula used previously for Jn 1=I0 (x) - 2I2 (x)+2I4 (x) - 2I6 (x)+ (6.6.5) In fact, we prefer simply to normalize with a call to bessi0. With this simple modification, the recursion routines bessj and bessy become the new routines bessi and bessk: FUNCTION bessk(n,x) INTEGER n REAL bessk,x C USES bessk0,bessk1 Returns the modified Bessel function Kn(x) for positive x and n = 2. INTEGER j REAL bk,bkm,bkp,tox,bessk0,bessk1 if (n.lt.2) pause bad argument n in bessk tox=2.0/x

6.6 Modified Bessel Functions of Integer Order 233 bkm=bessk0(x) Upward recurrence for all x... bk=bessk1(x) do 1 1 j=1,n-1 ...and here it is. bkp=bkm+j*tox*bk bkm=bk bk=bkp enddo 1 1 bessk=bk return END FUNCTION bessi(n,x) INTEGER n,IACC REAL bessi,x,BIGNO,BIGNI PARAMETER (IACC=40,BIGNO=1.0e10,BIGNI=1.0e-10) C USES bessi0 Returns the modified Bessel function In (x) for any real x and n = 2. INTEGER j,m REAL bi,bim,bip,tox,bessi0 if (n.lt.2) pause bad argument n in bessi if (x.eq.0.) then bessi=0. else tox=2.0/abs(x) bip=0.0 bi=1.0 bessi=0. m=2*((n+int(sqrt(float(IACC* Downward recurrence from even m. do 1 1 j=m,1,-1 Make IACC larger to increase accuracy. bim=bip+float(j)*tox*bi The downward recurrence. bip=bi bi=bim if (abs(bi).gt.BIGNO) then Renormalize to prevent overflows. bessi=bessi*BIGNI bi=bi*BIGNI bip=bip*BIGNI endif if (j.eq.n) bessi=bip enddo 1 1 bessi=bessi*bessi0(x)/bi Normalize with bessi0. if (x.lt.0..and.mod(n, bessi=-bessi endif return END CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), 9.8. [1] Carrier, G.F., Krook, M. and Pearson, C.E. 1966, Functions of a Complex Variable (New York: McGraw-Hill), pp. 220ff.

234 Chapter 6. Special Functions 6.7 Bessel Functions of Fractional Order, Airy Functions, Spherical Bessel Functions Many algorithms have been proposed for computing Bessel functions of fractional order numerically. Most of them are, in fact, not very good in practice. The routines given here are rather complicated, but they can be recommended wholeheartedly. Ordinary Bessel Functions The basic idea is Steeds method, which was originally developed [1] for Coulomb wave functions. The method calculates J , J , Y , and Y simultaneously, and so involves four relations among these functions. Three of the relations come from two continued fractions, one of which is complex. The fourth is provided by the Wronskian relation W = J Y - Y J = 2 px (6.7.1) The first continued fraction, CF1, is defined by = f = J J J x - J + 1 (6.7.2) 1 = x - 1 2(+1)/x - 2(+2)/x - You can easily derive it from the three-term recurrence relation for Bessel functions: Start with equation (6.5.6) and use equation (5.5.18). Forward eval-

uation of the continued fraction by one of the methods of 5.2 is essentially equivalent to backward recurrence of the recurrence relation. The rate of convergence of CF1 is determined by the position of the turning point xt p = (+1) , beyond which the Bessel functions become oscillatory. If x ¡ xt p , convergence is very rapid. If x ¿ xt p, then each iteration of the continued fraction effectively increases by one until x ¡ xt p ; thereafter rapid convergence sets in. Thus the number of iterations of CF1 is of order x for large x. In the routine bessjy we set the maximum allowed number of iterations to 10,000. For larger x, you can use the usual asymptotic expressions for Bessel functions. One can show that the sign of J is the same as the sign of the denominator of CF1 once it has converged. The complex continued fraction CF2 is defined by (1/2)2 - 2 (3/2)2 - 2 = - 1 p + iq = J + iY 2x + i + i x 2(x + i)+ 2(x +2i)+ (6.7.3) J + iY (We sketch the derivation of CF2 in the analogous case of modified Bessel functions in the next subsection.) This continued fraction converges rapidly for x ¿ xt p , while convergence fails as x 0. We have to adopt a special method for small x, which we describe below. For x not too small, we can ensure that x ¿ xt p by a stable recurrence of J and J downwards to a value = ¡ x, thus yielding the ratio f at this lower value of . This is the stable direction for the recurrence relation. The initial values for the recurrence are J = arbitrary,J = f J , (6.7.4) with the sign of the arbitrary initial value of J chosen to be the sign of the denominator of CF1. Choosing the initial value of J very small minimizes the possibility of overflow during the recurrence. The recurrence relations are J - 1 = xJ + J (6.7.5) J - 1 = - 1 x J - 1 - J

6.7 Bessel Functions of Fractional Order 235 Once CF2 has been evaluated at = , then with the Wronskian (6.7.1) we have enough relations to solve for all four quantities. The formulas are simplified by introducing the quantity (6.7.6) = p - f q Then 1/ 2 (6.7.7) J = W q + (p - f) (6.7.8) J = f J (6.7.9) Y = J (6.7.10) Y = Y p + q

The sign of J in (6.7.7) is chosen to be the same as the sign of the initial J in (6.7.4). Once all four functions have been determined at the value = , we can find them at the original value of .ForJ and J , simply scale the values in (6.7.4) by the ratio of (6.7.7) to the value found after applying the recurrence (6.7.5). The quantities Y and Y can be found by starting with the values in (6.7.9) and (6.7.10) and using the stable upwards recurrence Y +1 = 2 x Y - Y -1 (6.7.11) together with the relation (6.7.12) Y = xY - Y +1 Now turn to the case of small x, when CF2 is not suitable. Temme [2] has given a good method of evaluating Y and Y +1 , and hence Y from (6.7.12), by series expansions that accurately handle the singularity as x 0. The expansions work only for — —=1/2, and so now the recurrence (6.7.5) is used to evaluate f at a value = in this interval. Then one calculates J from (6.7.13) J = W Y - Y f and J from (6.7.8). The values at the original value of are determined by scaling as before, and the Y s are recurred up as before. Temmes series are 8 8 Y = - ck gk Y + 1 = - 2 ck hk (6.7.14) x k = 0 k = 0 Here (6.7.15) ck = (-x2 /4)k k! while the coefficients gk and hk are defined in terms of quantities pk , qk , and fk that can be found by recursion: gk = fk + 2 sin2 p 2 qk hk = -kgk + pk (6.7.16) pk = pk - 1 k - qk = qk - 1 k + fk = kfk -1 + pk -1 + qk - 1 k2 - 2

236 Chapter 6. Special Functions The initial values for the recurrences are - G(1 +) x p0 = 1 p 2 G(1 -) x (6.7.17) q0 = 1 p 2 p f0 = 2 p s ln 2 sin p cosh sG1 ( )+sinhs x G2 ( ) with s = ln 2 x 1 (6.7.18) G1 ( )= 1 2 G(1 -) - 1 G(1 +) 1 G2 ( )= 1 2 G(1 -) + 1 G(1 +) The whole point of writing the formulas in this way is that the potential problems as 0

can be controlled by evaluating p/sin p, sinh s/s, and G1 carefully. In particular, Temme gives Chebyshev expansions for G1 ( ) and G2 ( ). We have rearranged his expansion for G1 to be explicitly an even series in so that we can use our routine chebev as explained in 5.8. The routine assumes = 0. For negative you can use the reflection formulas J- =cos p J - sin p Y (6.7.19) Y- =sin p J +cos p Y The routine also assumes x¿0.Forx¡0 the functions are in general complex, but expressible in terms of functions with x¿0.Forx =0, Y is singular. Internal arithmetic in the routine is carried out in double precision. To maintain portability, complex arithmetic has been recoded with real variables. SUBROUTINE bessjy(x,xnu,rj,ry,rjp,ryp) INTEGER MAXIT REAL rj,rjp,ry,ryp,x,xnu,XMIN DOUBLE PRECISION EPS,FPMIN,PI PARAMETER (EPS=1.e-10,FPMIN=1.e-30,MAXIT=10000,XMIN=2., * PI=3.141592653589793d C USES beschb Returns the Bessel functions rj = J , ry = Y and their derivatives rjp = J , ryp = Y , for positive x and for xnu = = 0. The relative accuracy is within one or two significant digi ts of EPS, except near a zero of one of the functi ons, where EPS controls its absolute accuracy. FPMIN is a number close to the machines smallest floating-point number. All internal arithmetic i s in double precision. To convert the entire routine to double precision, change the REAL declaration above and decrease EPS to 10- 1 6 . Also convert the subroutine beschb. INTEGER i,isign,l,nl DOUBLE PRECISION a,b,br,bi,c,cr,ci,d,del,del1,den,di,dlr,dli, * dr,e,f,fact,fact2,fact3,ff,gam,gam1,gam2,gammi,gampl,h, * p,pimu,pimu2 * rymu,rymup,rytemp,sum,sum1,temp,w,x2,xi,xi2,xmu,xmu2 if(x.le.0..or pause bad arguments in bessjy if(x.lt.XMIN)then nl is the number of downward recurrences of the Js and upward recurrences of Y s. xmu lies between -1/2 and 1/2 for x ¡ XMIN, while it is chosen so that x is greater than the turning point for x

= XMIN. nl=int(xnu+.5d0) else nl=max(0,int(xnu-x+1.5d0)) endif xmu=xnu-nl xmu2=xmu*xmu xi=1.d0/x xi2=2.d0*xi w=xi2/PI The Wronskian.

6.7 Bessel Functions of Fractional Order 237 isign=1 Evaluate CF1 by modified Lentzs method (5.2). isign keeps track of sign changes in the denominator. h=xnu*xi if(h.lt.FPMIN)h=FPMIN b=xi2*xnu d=0.d0 c=h do 1 1 i=1,MAXIT b=b+xi2 d=b-d if(abs(d).lt.FPMIN)d=FPMIN c=b-1.d0/c if(abs(c).lt.FPMIN)c=FPMI d=1.d0/d del=c*d h=del*h if(d.lt.0.d0)isign=-isign if(abs(del-1.d0).lt.EPS)goto 1 enddo 1 1 pause x too large in bessjy; try asymptotic expansion 1 continue rjl=isign*FPMIN Initialize J and J for downward recurrence. rjpl=h*rjl rjl1=rjl Store values for later rescaling. rjp1=rjpl fact=xnu*xi do 1 2 l=nl,1,-1 rjtemp=fact*rjl+rjpl fact=fact-xi rjpl=fact*rjtemp-rjl rjl=rjtemp enddo 1 2 if(rjl.eq.0.d0)rjl=EPS f=rjpl/rjl Now have unnormalized J and J . if(x.lt.XMIN) then Use series. x2=.5d0*x pimu=PI*xmu if(abs(pimu).lt.EPS)then fact=1.d0 else fact=pimu/sin(pimu) endif d=-log(x2) e=xmu*d if(abs(e).lt.EPS)the fact2=1.d0 else fact2=sinh(e)/e endif call beschb(xmu,gam1,gam2,gampl Chebyshev evaluation of G1 and G2 . ff=2.d0/PI*fact*(gam1*cosh(e)+ga f0 . e=exp(e) p=e/(gampl*PI) p0 . q=1.d0/(e*PI*gammi) q0 . pimu2=0.5d0*pimu if(abs(pimu2).lt.EPS)then fact3=1.d0 else fact3=sin(pimu2)/pimu2 endif r=PI*pimu2*fact3*fact3 c=1.d0 d=-x2*x2 sum=ff+r*q sum1=p

238 Chapter 6. Special Functions do 1 3 i=1,MAXIT ff=(i*ff+p+q)/(i*i-xmu2) c=c*d/i p=p/(i-xmu) q=q/(i+xmu) del=c*(ff+r*q) sum=sum+del del1=c*p-i*del sum1=sum1+del1 if(abs(del).lt.(1.d0+abs(sum))*EPS)goto 2 enddo 1 3 pause bessy series failed to converge 2 continue rymu=-sum ry1=-sum1*xi2 rymup=xmu*xi*rymu-ry1 rjmu=w/(rymup-f*rymu) Equation (6.7.13). else Evaluate CF2 by modified Lentzs method (5.2). a=.25d0-xmu2 p=-.5d0*xi q=1.d0 br=2.d0*x

bi=2.d0 fact=a*xi/(p*p+q*q) cr=br+q*fact ci=bi+p*fact den=br*br+b
dr=br/den di=-bi/den dlr=cr*dr-ci*di dli=cr*di+ci*dr temp=p*dlr-
q*dli q=p*dli+q*dlr p=temp do 1 4 i=2,MAXIT a=a+2*(i-1)
bi=bi+2.d0 dr=a*dr+br di=a*di+bi if(abs(dr)+abs(di).lt.FPMIN)dr=F
fact=a/(cr*cr+ci*ci) cr=br+cr*fact ci=bi-ci*fact if(abs(cr)+abs(ci).lt.F
den=dr*dr+di*di dr=dr/den di=-di/den dlr=cr*dr-ci*di dli=cr*di+ci*d
temp=p*dlr-q*dli q=p*dli+q*dlr p=temp if(abs(dlr-1.d0)+abs(dli).lt.E
3 enddo 1 4 pause cf2 failed in bessjy 3 continue gam=(p-f)/q Equations (6.7.6) (6.7.10). rjmu=sqrt(w/((p-f)*gam+q)) rjmu=sign(rjmu,rjl) rymu=rjmu*gam rymup=rymu*(p+q/gam) ry1=xmu*xi*rymu-rymup endif fact=rjmu/rjl

246 Chapter 6. Special Functions 6.8 Spherical Harmonics Spherical harmonics occur in a large variety of physical problems, for ex- ample, whenever a wave equation, or Laplaces equation, is solved by separa- tion of variables in spherical coordinates. The spherical harmonic Y lm (,f), -l = m = l, is a function of the two coordinates , f on the surface of a sphere. The spherical harmonics are orthogonal for different l and m, and they are normalized so that their integrated square over the sphere is unity: 2 p d(cos)Yl m *(,f)Ylm (,f)=dl l dm m (6.8.1) df 1 0 - 1 Here asterisk denotes complex conjugation. Mathematically, the spherical harmonics are related to associated Legendre polynomials by the equation (l - m)! (l + m)! P m Yl m(, f)= 2l +1 l (cos)ei mf (6.8.2) 4p By using the relation Yl, -m(, f)=(-1)m Ylm *(,f)(6.8.3) we can always relate a spherical harmonic to an associated Legendre polynomial with m = 0. With x = cos , these are defined in terms of the ordinary Legendre polynomials (cf. 4.5 and 5.5) by P m l (x)=(-1)m(1 - x2)m/2 dm dxm Pl (x)(6.8.4) The first few associated Legendre polynomials, and their corresponding normalized spherical harmonics, are P 0 0 (x)= 1 Y0 0 = 1 4 p P 1 1 (x)= - (1 - x2)1 / 2 Y1 1 = - 3 8 p sin eif P 0 4 p cos 1 (x)=

xY1 0 = 3 1 5 P 2 2 (x)= 3(1- x2) Y2 2 = 1 4 2 p sin2 e2 i f P 1 2 (x)=-3(1- x2)1 / 2 xY2 1 = - 1 5 8 p sin cos ei f P 0 2 (x)= 1 2 cos2 - 1 2) 2 (3x2 - 1) Y2 0 = 5 4 p (3 (6.8.5) There are many bad ways to evaluate associated Legendre polynomials numer- ically. For example, there are explicit expressions, such as 1 - x P m l (x)= (-1)m (l + m)! 1!(m +1) 2 2mm!(l - m)! (1 - x2)m/2 1 - (l - m)(m + l +1) 2 1 - x + (l - m)(l - m - 1)(m + l +1)(m + l +2) - 2!(m +1)(m +2) 2 (6.8.6)

247 6.8 Spherical Har monics where the polynomial continues up through the term in (1 - x) l -m. (See [1] for this and related formulas.) This is not a satisfactory method because evaluation of the polynomial involves delicate cancellations between successive terms, which alternate in sign. For large l, the individual terms in the polynomial become very much larger than their sum, and all accuracy is lost. In practice, (6.8.6) can be used only in single precision (32-bit) for l up to 6 or 8, and in double precision (64-bit) for l up to 15 or 18, depending on the precision required for the answer. A more robust computational procedure is therefore desirable, as follows: The associated Legendre functions satisfy numerous recurrence relations, tab- ulated in [1-2]. These are recurrences on l alone, on m alone, and on both l and m simultaneously. Most of the recurrences involving m are unstable, and so dangerous for numerical work. The following recurrence on l is, however, stable (compare 5.5.1): (l - m)P m l- 2 (6.8.7) l = x(2l - 1)P m l -1 - (l + m - 1)Pm It is useful because there is a closed-form expression for the starting value, Pm m =(-1)m(2m - 1)!!(1 - x2)m/2 (6.8.8) (The notation n!! denotes the product of all odd integers less than or equal to n.) Using (6.8.7) with l = m +1, and setting P m m -1 =0, we find P m m (6.8.9) m+1 = x(2m +1)Pm Equations (6.8.8) and (6.8.9) provide the two starting values required for (6.8.7)

for general l. The function that implements this is FUNCTION plgndr(l,m,x) INTEGER l,m REAL plgndr,x Computes the associated Legendre polynomial Pm l (x).Herem and l are integers satisfying 0 = m = l, while x lies i n the range -1 = x = 1. INTEGER i,ll REAL fact,pll,pmm,pmmp1,somx2 if(m.lt.0.or.m.gt.l.or.abs(x).gt.1.)pause bad arguments in plgndr pmm=1. Compute Pm m . if(m.gt.0) then somx2=sqrt((1.-x)*(1.+x)) fact=1. do 1 1 i=1,m pmm=-pmm*fact*somx2 fact=fact+2. enddo 1 1 endif if(l.eq.m) then plgndr=pmm else pmmp1=x*(2*m+1)*pmm Compute Pm m+ 1 . if(l.eq.m+1) then plgndr=pmmp1 else Compute P m l , l¿m+1. do 1 2 ll=m+2,l

248 Chapter 6. Special Functions pll=(x*(2*ll-1)*pmmp1-(ll+m-1)*pmm)/(ll-m) pmm=pmmp1 pmmp1=pll enddo 1 2 plgndr=pll endif endif return END CITED REFERENCES AND FURTHER READING: Magnus, W., and Oberhettinger, F. 1949, Formulas and Theorems for the Functions of Mathe- matical Physics (New York: Chelsea), pp. 54ff. [1] Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 8. [2] 6.9 Fresnel Integrals, Cosine and Sine Integrals Fresnel Integrals The two Fresnel integrals are defined by cos p sin p C(x)= x 2 t2 dt (6.9.1) 2 t2 dt, S(x)= x 0 0 The most convenient way of evaluating these functions to arbitrary precision is to use power series for small x and a continued fraction for large x. The series are 2 x5 4 x9 C(x)=x - p 2 5 2! + p 2 9 4! - (6.9.2) 3 x7 5 x1 1 x3 S(x)= p 2 3 1! - p 2 7 3! + p 2 11 5! - There is a complex continued fraction that yields both S(x) and C(x) simul- taneously: vp C(x)+iS(x)= 1+i 2 erf z, z = 2 (1 - i)x (6.9.3) where 1 1/2 1 3/2 2 vp ez 2 erfc z = 1 z + z + z + z +

z + (6.9.4) 1 1 2 3 4 = 2z vp 2z2 +1- 2 z2 +5- 2z 2 +9-

248 Chapter 6. Special Functions pll=(x*(2*ll-1)*pmmp1-(ll+m-1)*pmm)/(ll-m) pmm=pmmp1 pmmp1=pll enddo 1 2 plgndr=pll endif endif return END CITED REFERENCES AND FURTHER READING: Magnus, W., and Oberhettinger, F. 1949, Formulas and Theorems for the Functions of Mathe- matical Physics (New York: Chelsea), pp. 54ff. [1] Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 8. [2] 6.9 Fresnel Integrals, Cosine and Sine Integrals Fresnel Integrals The two Fresnel integrals are defined by cos p sin p C(x)= x 2 t2 dt (6.9.1) 2 t2 dt, S(x)= x 0 0 The most convenient way of evaluating these functions to arbitrary precision is to use power series for small x and a continued fraction for large x. The series are 2 x5 4 x9 C(x)=x - p 2 5 2! + p 2 9 4! - (6.9.2) 3 x7 5 x1 1 x3 S(x)= p 2 3 1! - p 2 7 3! + p 2 11 5! - There is a complex continued fraction that yields both S(x) and C(x) simul- taneously: vp C(x)+iS(x)= 1+i 2 erf z, z = 2 (1 - i)x (6.9.3) where 1 1/2 1 3/2 2 vp ez 2 erfc z = 1 z + z + z + z + z + (6.9.4) 1 1 2 3 4 = 2z vp 2z2 +1- 2 z2 +5- 2z 2 +9-

6.9 Fresnel Integrals, Cosine and Sine Integrals 249 In the last line we have converted the standard form of the continued fraction to its even form (see 5.2), which converges twice as fast. We must be careful not to evaluate the alternating series (6.9.2) at too large a value of x; inspection of the terms shows that x =1.5 is a good point to switch over to the continued fraction. Note that for large x C(x) 1 2 + 1 px sin p 2 - 1 px cos p 2 x2 ,S(x) 1 2 x2 (6.9.5) Thus the precision of the routine frenel may be limited by the precision of the library routines for sine and cosine for large x. SUBROUTINE frenel(x,s,c) INTEGER

MAXIT REAL c,s,x,EPS,FPMIN,PI,PIBY2,XMIN PARAMETER (EPS=6.e-8,MAXIT=100,FPMIN=1.e-30,XMIN=1.5, * PI=3.1415927,PIBY2=1.5707963) Computes the Fresnel integrals S(x) and C(x) for all real x. Parameters: EPS is the relative error; MAXIT is the maximum number of iterations allowed; FPMIN is a number near the smallest representable oating-point number; XMIN is the dividing line between using the series and continued fraction; PI = p; PIBY2 = p/2. INTEGER k,n REAL a,absc,ax,fact,pix2,sign,sum,sumc,sums,term,test COMPLEX b,cc,d,h,del,cs LOGICAL odd absc(h)=abs(real(h))+abs(ain Statement function. ax=abs(x) if(ax.lt.sqrt(FPMIN))then Special case: avoid failure of convergence test because of under ow. s=0. c=ax else if(ax.le.XMIN)then Evaluate both series simultaneously. sum=0. sums=0. sumc=ax sign=1. fact=PIBY2*ax*ax odd=.true. term=ax n=3 do 1 1 k=1,MAXIT term=term*fact/k sum=sum+sign*term/n test=abs(sum)*EPS if(odd)then sign=-sign sums=sum sum=sumc else sumc=sum sum=sums endif if(term.lt.test)goto 1 odd=.not.odd n=n+2 enddo 1 1 pause series failed in frenel 1 s=sums c=sumc else Evaluate continued fraction by modified Lentzs method (5.2). pix2=PI*ax*ax b=cmplx(1.,-pix2)

250 Chapter 6. Special Functions cc=1./FPMIN d=1./b h=d n=-1 do 1 2 k=2,MAXIT n=n+2 a=-n*(n+1) b=b+4. d=1./(a*d+b) Denominators cannot be zero. cc=b+a/cc del=cc*d h=h*del if(absc(del-1.).lt.EPS)goto 2 enddo 1 2 pause cf failed in frenel 2 h=h*cmplx(ax,-ax) cs=cmplx(.5,.5)*(1.-cmplx(cos(.5*pix2),sin(.5*pix2))*h) c=real(cs) s=aimag(cs) endif if(x.lt.0.)then Use antisymmetry. c=-c s=-s endif return END Cosine and Sine Integrals The cosine and sine integrals are defined by cos t - 1 Ci(x)= +lnx + x t dt (6.9.6) 0 sin t Si(x)= x t dt 0 Here 0.5772... is Eulers constant. We only need a way to calculate the functions for x¿0, because Si(-x)=-

Si(x), Ci(-x)=Ci(x) - ip (6.9.7) Once again we can evaluate these functions by a judicious combination of power series and complex continued fraction. The series are Si(x)=x - x3 5 5! - 3 3! + x5 (6.9.8) Ci(x)= +lnx + - x2 4 4! - 2 2! + x4 The continued fraction for the exponential integral E1 (ix) is E1 (ix)=- Ci(x)+i[Si(x) - p/2] 1 1 2 2 = e-i x 1 (6.9.9) ix + 1+ ix + 1+ ix + 12 22 = e-i x 1 1+ix - 3+ix - 5+ix -

6.9 Fresnel Integrals, Cosine and Sine Integrals 251 The even form of the continued fraction is given in the last line and converges twice as fast for about the same amount of computation. A good crossover point from the alternating series to the continued fraction is x =2in this case. As for the Fresnel integrals, for large x the precision may be limited by the precision of the sine and cosine routines. SUBROUTINE cisi(x,ci,si) INTEGER MAXIT REAL ci,si,x,EPS,EULER,PIBY2,FPMIN,TMIN PARAMETER (EPS=6.e-8,EULER=.57721566,MAXIT=100,PIBY2=1 * FPMIN=1.e-30,TMIN=2.) Computes the cosi ne and sine integrals Ci(x) and Si(x). Ci(0) is returned as a large negative number and no error message is generated. For x¡0 the routine returns Ci(-x) and you must supply the - ip yourself. Parameters: EPS is the relative error, or absolute error near a zero of Ci(x); EULER = ; MAXIT is the maximum number of iterations allowed; PIBY2 = p/2; FPMIN is a number near the smallest representable oating-point number; TMIN is the dividing line between usi ng the series and continued fraction. INTEGER i,k REAL a,err,fact,sign,sum,sumc,sums,t,term,absc COMPLEX h,b,c,d,del LOGICAL odd absc(h)=abs(real(h))+abs(aimag(h)) Statement function. t=abs(x) if(t.eq.0.)then Special case. si=0. ci=-1./FPMIN return endif if(t.gt.TMIN)then Evaluate continued fraction by modified Lentzs method (5.2). b=cmplx(1.,t) c=1./FPMIN d=1./b h=d do 1 1 i=2,MAXIT a=-(i-1)**2 b=b+2. d=1./(a*d+b) Denominators cannot be zero. c=b+a/c del=c*d h=h*del if(absc(del-1.).lt.EPS)goto 1 enddo 1 1 pause cf failed in cisi 1 continue h=cmplx(cos(t),-sin(t))*h ci=-real(h) si=PIBY2+aimag(h) else Evaluate both series simultaneously. if(t.lt.sqrt(FPMIN))then Special case: avoid failure of convergence test because of under ow. sumc=0. sums=t else sum=0. sums=0. sumc=0. sign=1. fact=1. odd=.true. do 1 2 k=1,MAXIT fact=fact*t/k term=fact/k

252 Chapter 6. Special Functions sum=sum+sign*term err=term/abs(sum) if(odd)then sign=-sign sums=sum sum=sumc else sumc=sum sum=sums endif if(err.lt.EPS)goto 2 odd=.not.odd enddo 1 2 pause maxits exceeded in cisi endif 2 si=sums ci=sumc+log(t)+EULER endif if(x.lt.0.)si=-si return END CITED REFERENCES AND FURTHER READING: Stegun, I.A., and Zucker, R. 1976, Journal of Research of the National Bureau of Standards, vol. 80B, pp. 291311; 1981, op. cit., vol. 86, pp. 661686. Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapters 5 and 7. 6.10 Dawsons Integral Dawsons Integral F (x) is defined by x et2 dt (6.10.1) F (x)=e-x 2 0 The function can also be related to the complex error function by F(z)= ivp 2 e- z 2 [1 - erfc(-iz)] . (6.10.2) A remarkable approximation for F (z), due to Rybicki [1],is 1 F (z) = lim e-(z -n h)2 vp n od d n (6.10.3) h 0 What makes equation (6.10.3) unusual is that its accuracy increases exponentially as h gets small, so that quite moderate values of h (and correspondingly quite rapid convergence of the series) give very accurate approximations.

252 Chapter 6. Special Functions sum=sum+sign*term err=term/abs(sum) if(odd)then sign=-sign sums=sum sum=sumc else sumc=sum sum=sums endif if(err.lt.EPS)goto 2 odd=.not.odd

enddo 1 2 pause maxits exceeded in cisi endif 2 si=sums ci=sumc+log(t)+EULER endif if(x.lt.0.)si=-si return END CITED REFERENCES AND FURTHER READING: Stegun, I.A., and Zucker, R. 1976, Journal of Research of the National Bureau of Standards, vol. 80B, pp. 291311; 1981, op. cit., vol. 86, pp. 661686. Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapters 5 and 7. 6.10 Dawsons Integral Dawsons Integral F (x) is defined by x et2 dt (6.10.1) F (x)=e-x 2 0 The function can also be related to the complex error function by F(z)= ivp 2 e- z 2 [1 - erfc(-iz)] . (6.10.2) A remarkable approximation for F (z), due to Rybicki [1],is 1 F (z) = lim e-(z -n h)2 vp n od d n (6.10.3) h 0 What makes equation (6.10.3) unusual is that its accuracy increases exponentially as h gets small, so that quite moderate values of h (and correspondingly quite rapid convergence of the series) give very accurate approximations.

253 6.10 Dawsons Integral We will discuss the theory that leads to equation (6.10.3) later, in 13.11, as an interesting application of Fourier methods. Here we simply implement a routine for real values of x based on the formula. It is first convenient to shift the summation index to center it approximately on the maximum of the exponential term. Define n 0 to be the even integer nearest to x/h, and x0 = n0 h, x = x - x0 , and n = n - n0 , so that N F (x) 1 vp e- (x -n h)2 , (6.10.4) n + n0 n =- N n o dd where the approximate equality is accurate when h is sufficiently small and N is sufficiently large. The computation of this formula can be greatly speeded up if we note that e-(x -n h)2 = e-x 2 e-(n h)2 e2 x h n . (6.10.5) The first factor is computed once, the second is an array of constants to be stored, and the third can be computed recursively, so that only

two exponentials need be evaluated. Advantage is also taken of the symmetry of the coefficients e -(n h) 2 by breaking the summation up into positive and negative values of n separately. In the following routine, the choices h =0.4 and N =11are made. Because of the symmetry of the summations and the restriction to odd values of n, the limits on the do loops are 1 to 6. The accuracy of the result in this REAL version is about 2 10- 7 . In order to maintain relative accuracy near x =0, where F (x) vanishes, the program branches to the evaluation of the power series [2] for F (x), for —x— ¡ 0.2. FUNCTION dawson(x) INTEGER NMAX REAL dawson,x,H,A1,A2,A3 PARAMETER (NMAX=6,H=0.4,A1=2./3.,A2=0.4,A3=2./7.) Returns Dawsons integral F (x)=exp(-x2) x 0 exp(t2)dt for any real x. INTEGER i,init,n0 REAL d1,d2,e1,e2,sum,x2,xp,xx,c(NMAX) SAVE init,c DATA init/0/ Flag is 0 if we need to initialize, else 1. if(init.eq.0)then init=1 do 1 1 i=1,NMAX c(i)=exp(-((2.*float(i)-1.)*H)**2) enddo 1 1 endif if(abs(x).lt.0.2)then Use series expansion. x2=x**2 dawson=x*(1.-A1*x2*(1.-A2*x2*(1.-A3*x2))) else Use sampling theorem representation. xx=abs(x) n0=2*nint(0.5*xx/H) xp=xx-float(n0)*H e1=exp(2.*xp*H) e2=e1**2 d1=float(n0+1) d2=d1-2. sum=0. do 1 2 i=1,NMAX

254 Chapter 6. Special Functions sum=sum+c(i)*(e1/d1+1./(d2*e1) d1=d1+2. d2=d2-2. e1=e2*e1 enddo 1 2 dawson=0.5641895835*sign(exp xp**2),x)*sum Constant is 1/vp. endif return END Other methods for computing Dawsons integral are also known [2,3]. CITED REFERENCES AND FURTHER READING: Rybicki, G.B. 1989, Computers in Physics, vol. 3, no. 2, pp. 8587. [1] Cody, W.J., Pociorek, K.A., and Thatcher, H.C. 1970, Mathematics of Computation, vol. 24, pp. 171178. [2] McCabe, J.H. 1974, Mathematics of Computation, vol. 28, pp. 811816. [3] 6.11 Elliptic Integrals and Jacobian Elliptic Functions Elliptic

integrals occur in many applications, because any integral of the form R(t, s) dt (6.11.1) where R is a rational function of t and s, and s is the square root of a cubic or quartic polynomial in t, can be evaluated in terms of elliptic integrals. Standard references [1] describe how to carry out the reduction, which was originally done by Legendre. Legendre showed that only three basic elliptic integrals are required. The simplest of these is dt I1 = x (a1 + b1 t)(a2 + b2 t)(a3 + b3 t)(a4 + b4 t) (6.11.2) y where we have written the quartic s2 in factored form. In standard integral tables [2], one of the limits of integration is always a zero of the quartic, while the other limit lies closer than the next zero, so that there is no singularity within the interval. To evaluate I1 , we simply break the interval [y, x] into subintervals, each of which either begins or ends on a singularity. The tables, therefore, need only distinguish the eight cases in which each of the four zeros (ordered according to size) appears as the upper or lower limit of integration. In addition, when one of the bs in (6.11.2) tends to zero, the quartic reduces to a cubic, with the largest or smallest singularity moving to 8; this leads to eight more cases (actually just special cases of the first eight). The sixteen cases in total are then usually tabulated in terms of Legendres standard elliptic integral of the 1st kind, which we will define below. By a change of the variable of integration t, the zeros of the quartic are mapped to standard locations

254 Chapter 6. Special Functions sum=sum+c(i)*(e1/d1+1./(d2*e1)) d1=d1+2. d2=d2-2. e1=e2*e1 enddo 1 2 dawson=0.5641895835*sign(ex xp**2),x)*sum Constant is 1/vp. endif return END Other methods for computing Dawsons integral are also known [2,3]. CITED REFERENCES AND FURTHER READING: Rybicki, G.B. 1989, Computers in Physics, vol. 3, no. 2, pp. 8587. [1] Cody, W.J., Pociorek, K.A., and Thatcher, H.C. 1970, Mathe-

matics of Computation, vol. 24, pp. 171178. [2] McCabe, J.H. 1974, Mathematics of Computation, vol. 28, pp. 811816. [3] 6.11 Elliptic Integrals and Jacobian Elliptic Functions Elliptic integrals occur in many applications, because any integral of the form R(t, s) dt (6.11.1) where R is a rational function of t and s, and s is the square root of a cubic or quartic polynomial in t, can be evaluated in terms of elliptic integrals. Standard references [1] describe how to carry out the reduction, which was originally done by Legendre. Legendre showed that only three basic elliptic integrals are required. The simplest of these is dt I1 = x (a1 + b1 t)(a2 + b2 t)(a3 + b3 t)(a4 + b4 t) (6.11.2) y where we have written the quartic s2 in factored form. In standard integral tables [2], one of the limits of integration is always a zero of the quartic, while the other limit lies closer than the next zero, so that there is no singularity within the interval. To evaluate I1 , we simply break the interval [y, x] into subintervals, each of which either begins or ends on a singularity. The tables, therefore, need only distinguish the eight cases in which each of the four zeros (ordered according to size) appears as the upper or lower limit of integration. In addition, when one of the bs in (6.11.2) tends to zero, the quartic reduces to a cubic, with the largest or smallest singularity moving to 8; this leads to eight more cases (actually just special cases of the first eight). The sixteen cases in total are then usually tabulated in terms of Legendres standard elliptic integral of the 1st kind, which we will define below. By a change of the variable of integration t, the zeros of the quartic are mapped to standard locations

on the real axis. Then only two dimensionless parameters are needed to tabulate Legendres integral. However, the symmetry of the original integral (6.11.2) under permutation of the roots is concealed in Legendres notation. We will get back to Legendres notation below. But first, here is a better way: Carlson [3] has given a new definition of a standard elliptic integral of the first kind, dt 8 RF (x, y, z)= 1 2 (t + x)(t + y)(t + z) (6.11.3) 0 where x, y, and z are nonnegative and at most one is zero. By standardizing the range of integration, he retains permutation symmetry for the zeros. (Weierstrass canonical form also has this property.) Carlson first shows that when x or y is a zero of the quartic in (6.11.2), the integral I1 can be written in terms of RF in a form that is symmetric under permutation of the remaining three zeros. In the general case when neither x nor y is a zero, two such RF functions can be combined into a single one by an addition theorem, leading to the fundamental formula I1 =2RF (U2 14)(6.11.4) 12 ,U2 13 ,U2 where Ui j =(Xi Xj Yk Ym + Yi YjXk Xm)/(x - y)(6.11.5) Xi =(ai + bi x)1 / 2 ,Yi =(ai + bi y)1 /2 (6.11.6) and i, j, k, m is any permutation of 1, 2, 3, 4. A short-cut in evaluating these expressions is U2 1 2 - (a1 b4 - a4 b1)(a2 b3 - a3 b2) 1 3 = U2 (6.11.7) U2 1 2 - (a1 b3 - a3 b1)(a2 b4 - a4 b2) 1 4 = U2 The U s correspond to the three ways of pairing the four zeros, and I1 is thus manifestly symmetric under permutation of the zeros. Equation (6.11.4) therefore reproduces all sixteen cases when one limit is a zero, and also includes the cases when neither limit is a zero. Thus Carlsons function allows arbitrary ranges of integration and arbitrary positions of the branch points of the integrand relative to the interval of integration. To handle elliptic integrals of the second and third kind, Carlson defines the standard integral of the third kind as dt 8 (6.11.8) RJ (x, y, z, p)= 3 2 (t + p) (t + x)(t + y)(t + z) 0 which is symmetric in x, y, and z. The degenerate case when two arguments are equal is denoted RD (x, y, z)=RJ (x, y, z, z)(6.11.9) and is symmetric in x and y. The function RD replaces Legendres integral of the second kind. The degenerate form of RF is denoted RC (x, y)=RF (x, y, y)(6.11.10) It embraces logarithmic, inverse circular, and inverse hyperbolic functions. Carlson [4-7] gives integral tables in terms of the exponents of the linear factors of the quartic in (6.11.1). For example, the integral where the exponents are (1 2 ,- 1 2 ,-3 2) 2 ,1 can be expressed as a single integral in terms of RD ; it accounts for 144 separate cases in Gradshteyn and Ryzhik [2]! Refer to Carlsons papers [3-7] for some of the practical details in reducing elliptic integrals to his standard forms, such as handling complex conjugate zeros.

256 Chapter 6. Special Functions Turn now to the numerical evaluation of elliptic integrals. The traditional methods [8] are Gauss or Landen transformations. Descending transformations decrease the modulus k of the Legendre integrals towards zero, increasing transformations increase it towards unity. In these limits the functions have simple analytic expressions. While these methods converge quadratically and are quite satisfactory for integrals of the first and second kinds, they generally lead to loss of significant figures in certain regimes for integrals of the third kind. Carlsons algorithms [9,10], by contrast, provide a unified method for all three kinds with no significant cancellations. The key ingredient in these algorithms is the duplication theorem: RF (x, y, z)=2RF (x + , y + , z +) (6.11.11) x + = RF 4 , y + 4 , z + 4 where =(xy)1 / 2 +(xz)1 / 2 +(yz)1/ 2 (6.11.12) This theorem can be proved by a simple change of variable of integration[11]. Equation (6.11.11) is iterated until the arguments of RF are nearly equal. For equal arguments we have RF (x, x, x)=x-1 / 2 (6.11.13) When the arguments are close enough, the function is evaluated from a fixed Taylor expansion about (6.11.13) through fifth-order terms. While the iterative part of the algorithm is only linearly convergent, the error ultimately decreases by a factor of 46 = 4096 for each

iteration. Typically only two or three iterations are required, perhaps six or seven if the initial values of the arguments have huge ratios. We list the algorithm for RF here, and refer you to Carlsons paper [9] for the other cases. Stage 1: For n =0, 1, 2,... compute n =(xn + yn + zn)/3 Xn =1- (xn /n),Yn =1- (yn /n),Zn =1- (zn /n) n =max(—Xn —, —Yn —, —Zn —) If n ¡ tol go to Stage 2; else compute n =(xn yn)1 / 2 +(xn zn)1 / 2 +(yn zn)1 / 2 xn+ 1 =(xn + n)/4,yn + 1 =(yn + n)/4,zn +1 =(zn + n)/4 and repeat this stage. Stage 2: Compute E2 = Xn Yn - Z2 n ,E3 = Xn Yn Zn RF =(1- 1 2 - 3 1 0 E2 + 1 1 4 E3 + 1 2 4 E2 4 4 E2 E3)/(n)1/ 2 In some applications the argument p in RJ or the argument y in RC is negative, and the Cauchy principal value of the integral is required. This is easily handled by using the formulas RJ (x, y,z, p)= [(- y)RJ (x, y, z,) - 3RF (x, y, z)+3RC (xz/y, p /y)] /(y - p) (6.11.14) where = y + (z - y)(y - x) (6.11.15) y - p

6.11 Elliptic Integrals and Jacobian Elliptic Functions 257 is positive if p is negative, and 1/ 2 RC (x - y, -y)(6.11.16) RC (x, y)= x x - y The Cauchy principal value of RJ has a zero at some value of p¡0, so (6.11.14) will give some loss of significant figures near the zero. FUNCTION rf(x,y,z) REAL rf,x,y,z,ERRTOL,TINY,BIG,THIRD,C1,C2,C3,C4 PARAMETER (ERRTOL=.08,TINY=1.5e-38,BIG=3.E37,THIRD=1./3., * C1=1./24.,C2=.1,C3=3./44.,C4=1./14.) Computes Carlsons elliptic integral of the first kind, RF (x, y,z). x, y,andz must be nonnegative, and at most one can be zero. TINY must be at least 5 times the machine under ow limit, BIG at most one fifth the machine over ow limit. REAL alamb,ave,delx,dely,delz,e2,e3,sqrtx,sqrty,sqrtz,xt,yt,zt if(min(x,y,z).lt.0.. * max(x,y,z).gt.BIG)pause invalid arguments in rf xt=x yt=y zt=z 1 continue sqrtx=sqrt(xt) sqrty=sqrt(yt) sqrtz=sqrt(zt) alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz xt=.25*(xt+alamb) yt=.25*(yt zt=.25*(zt+alamb) ave=THIRD*(xt+yt+zt) delx=(ave-xt)/ave dely=(ave-yt)/ave delz=(ave-zt)/ave if(max(abs(delx),abs(dely),abs(delz 1 e2=delx*dely-delz**2 e3=delx*dely*delz rf=(1.+(C1*e2-C2-C3*e3)*e2+C4*e3)/sqrt(ave) return END A value of 0.08 for the error tolerance parameter is adequate for single precision (7 n , we see that 0.0025 will yield double precision significant digits). Since the error scales as 6 (16 significant digits) and require at most two or three more iterations. Since the coefficients of the sixth-order truncation error are different for the other elliptic functions, these values for the error tolerance should be changed to 0.04 and 0.0012 in the algorithm for RC , and 0.05 and 0.0015 for RJ and RD . As well as being an algorithm in its own right for certain combinations of elementary functions, the algorithm for RC is used repeatedly in the computation of RJ . The Fortran implementations test the input arguments against two machine-dependent constants, TINY and BIG, to ensure that there will be no underflow or overflow during the computation. We have chosen conservative values, corresponding to a machine minimum of 3 10-39 and a machine maximum of 1.7 103 8 . You can always extend the range of admissible argument values by using the homogeneity relations (6.11.22), below. FUNCTION rd(x,y,z) REAL rd,x,y,z,ERRTOL,TINY,BIG,C1,C2,C3,C4,C5,C6 PARAMETER (ERRTOL=.05,TINY=1.e-25,BIG=4.5E21,C1=3./14.,C2= * C3=9./22.,C4=3./26.,C5=.25*C3,C6=1.5*C4) Computes Carlsons elliptic integral of the second kind, RD (x, y, z). x and y must be nonnegative, and at most one can be zero. z must be positive. TINY must be at least twice the negative 2/3 power of the machine overflow limit. BIG must be at most 0.1 ERRTOL times the negative 2/3 power of the machine underflow limit. REAL alamb,ave,delx,dely,delz,ea,eb,ec,ed,ee,fac,sqrtx,sqrty,

258 Chapter 6. Special Functions * sqrtz,sum,xt,yt,zt

if(min(x,y).lt.0..or.min(x+y,z).lt.TINY.or. * max(x,y,z).gt.BIG)pause invalid arguments in rd xt=x yt=y zt=z sum=0. fac=1. 1 continue sqrtx=sqrt(xt) sqrty=sqrt(yt) sqrtz=sqrt(zt) alamb=sqrtx*(sqrty+ sum=sum+fac/(sqrtz*(zt+alamb)) fac=.25*fac xt=.25*(xt+alamb) yt=.25*(yt+alamb) zt=.25*(zt+alamb) ave=.2*(xt+yt+3.*zt) delx=(ave-xt)/ave dely=(ave-yt)/ave delz=(ave-zt)/ave if(max(abs(delx) 1 ea=delx*dely eb=delz*delz ec=ea-eb ed=ea-6.*eb ee=ed+ec+ec rd=3.*sum+fac*(1.+ed*(-C1+C5*ed-C6*delz*ee) * +delz*(C2*ee+delz* C3*ec+delz*C4*ea)))/(ave*sqrt(ave)) return END FUNCTION rj(x,y,z,p) REAL rj,p,x,y,z,ERRTOL,TINY,BIG,C1,C2,C3,C4,C5,C6,C7, PARAMETER (ERRTOL=.05,TINY=2.5e-13,BIG=9.E11,C1=3./14.,C * C3=3./22.,C4=3./26.,C5=.75*C3,C6=1.5*C4,C7=.5*C2,C8=C3+C3) C USES rc,rf Computes Carlsons elliptic integral of the third kind, RJ (x,y, z, p). x, y,andz must be nonnegative, and at most one can be zero. p must be nonzero. If p¡0, the Cauchy principal value is returned. TINY must be at least twice the cube root of the machine under ow limit, BIG at most one fifth the cube root of the machine over ow limit. REAL a,alamb,alpha,ave,b,beta,delp,delx,dely,delz,ea,eb,ec, * ed,ee,fac,pt,rcx,rho,sqrtx,sqrty,sqrtz,sum,tau,xt, * yt,zt,rc,rf if(min(x,y,z) * max(x,y,z,abs(p)).gt.BIG)pause invalid arguments in rj sum=0. fac=1. if(p.gt.0.)then xt=x yt=y zt=z pt=p else xt=min(x,y,z) zt=max(x,y,z) yt=x+y+z-xt-zt a=1./(yt-p) b=a*(zt-yt)*(yt-xt) pt=yt+b rho=xt*zt/yt

263 6.12 Hypergeometric Functions CITED REFERENCES AND FURTHER READING: Erd elyi, A., Magnus, W., Oberhettinger, F., and Tricomi, F.G. 1953, Higher Transcendental Functions, Vol. II, (New York: McGraw-Hill). [1] Gradshteyn, I.S., and Ryzhik, I.W. 1980, Table of Integrals, Series, and Products (New York: Academic Press). [2] Carlson, B.C. 1977, SIAM Journal on Mathematical Analysis, vol. 8, pp. 231242. [3] Carlson, B.C. 1987, Mathematics of Computation, vol. 49,

pp. 595606 [4]; 1988, op. cit., vol. 51, pp. 267280 [5]; 1989, op. cit., vol. 53, pp. 327333 [6]; 1991, op. cit., vol. 56, pp. 267280. [7] Bulirsch, R. 1965, Numerische Mathematik, vol. 7, pp. 7890; 1965, op. cit., vol. 7, pp. 353354; 1969, op. cit., vol. 13, pp. 305315. [8] Carlson, B.C. 1979, Numerische Mathematik, vol. 33, pp. 116. [9] Carlson, B.C., and Notis, E.M. 1981, ACM Transactions on Mathematical Software, vol. 7, pp. 398403. [10] Carlson, B.C. 1978, SIAM Journal on Mathematical Analysis, vol. 9, p. 524528. [11] Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 17. [12] Mathews, J., and Walker, R.L. 1970, Mathematical Methods of Physics, 2nd ed. (Reading, MA: W.A. Benjamin/Addison-Wesley), pp. 7879. 6.12 Hypergeometric Functions As was discussed in 5.14, a fast, general routine for the the complex hyperge- ometric function 2 F1 (a, b, c; z), is difficult or impossible. The function is defined as the analytic continuation of the hypergeometric series, z z 2 2 F1 (a, b, c; z)=1+ ab c 1! + a(a +1)b(b +1) c(c +1) 2! + zj + a(a +1)...(a + j - 1)b(b +1)...(b + j - 1) c(c +1)...(c + j - 1) j! + (6.12.1) This series converges only within the unit circle —z— ¡ 1 (see [1]), but ones interest in the function is not confined to this region. Section 5.14 discussed the method of evaluating this function by direct path integration in the complex plane. We here merely list the routines that result. Implementation of the function hypgeo is straightforward, and is described by comments in the program. The machinery associated with Chapter 16s routine for integrating differential equations, odeint, is only minimally intrusive, and need not even be completely understood: use of odeint requires a common block with one zeroed variable, one subroutine call, and a prescribed format for the derivative routine hypdrv. The subroutine hypgeo will fail, of course, for values of z too close to the singularity at 1. (If you need to approach this singularity, or the one at 8, use the linear transformation formulas in 15.3 of [1].) Away from z =1, and for moderate values of a, b, c, it is often remarkable how few steps are required to integrate the equations. A half-dozen is typical.

264 Chapter 6. Special Functions FUNCTION hypgeo(a,b,c,z) COMPLEX hypgeo,a,b,c,z REAL EPS PARAMETER (EPS=1.e-6) Accuracy parameter. C USES bsstep,hypdrv,hypser,o
Complex hypergeometric function 2 F1 for complex a, b,c,andz , by direct integration of the hypergeometric equation in the complex plane. The branch cut is taken to lie along the real axis, Re z¿1. INTEGER kmax,nbad,nok EXTERNAL bsstep,hypdrv COMPLEX z0,dz,aa,bb,cc,y(2) COMMON /hypg/ aa,bb,cc,z0,dz COMMON /path/ kmax Used by odeint. kmax=0 if (real(z)**2+aimag(z)**2.le.0.25) then Use series... call hypser(a,b,c,z,hypgeo,y(2)) return else if (real(z).lt.0.) then ...or pick a starting point for the path inte- gration. z0=cmplx(-0.5,0.) else if (real(z).le.1.0) then z0=cmplx(0.5,0.) else z0=cmplx(0.,sign(0.5,aimag(z))) endif aa=a Load the common block, used to pass pa- rameters over the head of odeint to hypdrv. bb=b cc=c dz=z-z0 call hypser(aa,bb,cc,z0,y(1),y(2)) Get starting function and derivative. call odeint(y,4,0.,1.,EPS,.1,.0001,nok,nb
The arguments to odeint are the vector of independent variabl es, its length, the starting and ending values of the dependent variable, the accuracy parameter, an initial guess for stepsize, a minimum stepsize, the (returned) number of good and bad steps taken, and the names of the derivative routine and the (here Bulirsch-Stoer) stepping routine. hypgeo=y(1) return END SUBROUTINE hypser(a,b,c,z,series,deriv) INTEGER n COMPLEX a,b,c,z,series,deriv,aa,bb,cc,fac,temp Returns the

hypergeometric series 2 F1 and its derivative, iterating to machine accuracy. For cabs(z) = 1/2 convergence is quite rapid. deriv=cmplx(0.,0.) fac=cmplx(1.,0.) temp=fac aa=a bb=b cc=c do 1 1 n=1,1000 fac=((aa*bb)/cc)*fac deriv=deriv+fac fac=fac*z/n series=temp+fac if (series.eq.temp) return temp=series aa=aa+1. bb=bb+1. cc=cc+1. enddo 1 1 pause convergence failure in hypser END

265 6.12 Hypergeometric Functions SUBROUTINE hypdrv(s,y,dyds) REAL s COMPLEX y(2),dyds(2),aa,bb,cc,z0,dz,z Derivative subrouti ne for the hypergeometric equation, see text equation (5.14.4). COMMON /hypg/ aa,bb,cc,z0,dz z=z0+s*dz dyds(1)=y(2)*dz dyds(2)=((aa*bb)*y(1)-(cc-((aa+bb)+1.)* z)) return END CITED REFERENCES AND FURTHER READING: Abramowitz, M., and Stegun, I.A. 1964, Handbook of Mathematical Functions, Applied Mathe- matics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York). [1]