5.9 Derivatives or Integrals of a Chebyshev-approximated Function

If you have obtained the Chebyshev coefficients that approximate a function in a certain range (e.g., from chebft in §5.8), then it is a simple matter to transform them to Chebyshev coefficients corresponding to the derivative or integral of the function. Having done this, you can evaluate the derivative or integral just as if it were a function that you had Chebyshev-fitted ab initio.

The relevant formulas are these: If \( c_i, i = 1, \ldots, m \) are the coefficients that approximate a function \( f \) in equation (5.8.9), \( C_i \) are the coefficients that approximate the indefinite integral of \( f \), and \( c'_i \) are the coefficients that approximate the derivative of \( f \), then

\[
C_i = \frac{c_{i-1} - c_{i+1}}{2(i-1)} \quad (i > 1)
\]

(5.9.1)

\[
c'_{i-1} = c'_{i+1} + 2(i-1)c_i \quad (i = m - 1, m - 2, \ldots, 2)
\]

(5.9.2)

Equation (5.9.1) is augmented by an arbitrary choice of \( C_1 \), corresponding to an arbitrary constant of integration. Equation (5.9.2), which is a recurrence, is started with the values \( c'_m = c'_{m+1} = 0 \), corresponding to no information about the \( m+1 \)st Chebyshev coefficient of the original function \( f \).

Here are routines for implementing equations (5.9.1) and (5.9.2).

SUBROUTINE chder(a,b,c,cder,n)
INTEGER n
REAL a,b,c(n),cder(n)
Given \( a, b, c(1:n) \), as output from routine chebft §5.8, and given \( n \), the desired degree of approximation (length of \( c \) to be used), this routine returns the array \( cder(1:n) \), the Chebyshev coefficients of the derivative of the function whose coefficients are \( c(1:n) \).

INTEGER j
REAL con

\[
cder(n)=0.
\]

\[
cder(n-1)=2*(n-1)*c(n)
\]
do j=n-2,1,-1
\[
cder(j)=cder(j+2)+2*j*c(j+1)
\]
enddo
con=2./(b-a)
do j=1,n
\[
cder(j)=cder(j)*con
\]
endo
return
END

SUBROUTINE chint(a,b,c,cint,n)
INTEGER n
REAL a,b,c(n),cint(n)
Given \( a, b, c(1:n) \), as output from routine chebft §5.8, and given \( n \), the desired degree of approximation (length of \( c \) to be used), this routine returns the array \( cint(1:n) \), the Chebyshev coefficients of the integral of the function whose coefficients are \( c \). The constant of integration is set so that the integral vanishes at \( a \).

INTEGER j
REAL con,fac,sum

\[
cint(n)=0.
\]
\[
cint(n-1)=c(n)
\]
do j=n-2,1,-1
\[
cint(j)=cint(j+2)+2*j*c(j+1)
\]
enddo
con=2./(b-a)
do j=1,n
\[
cint(j)=cint(j)*con
\]
endo
return
END
con = 0.25 * (b - a)  
Factor that normalizes to the interval b - a.

sum = 0.  
Accumulates the constant of integration.

fac = 1.  
Will equal \pm 1.

do 11  
j = 2, n - 1

\text{cint}(j) = \text{con} \times \frac{\text{c}(j - 1) - \text{c}(j + 1)}{j - 1}  
\text{Equation (5.9.1)}.

sum = sum + fac * cint(j)

fac = -fac
enddo 11

cint(n) = \text{con} \times \frac{\text{c}(n - 1)}{n - 1}  
\text{Special case of (5.9.1) for n}.

sum = sum + fac * cint(n)

cint(1) = 2. * sum  
Set the constant of integration.

return

END

Clenshaw-Curtis Quadrature

Since a smooth function’s Chebyshev coefficients \( c_i \) decrease rapidly, generally exponentially, equation (5.9.1) is often quite efficient as the basis for a quadrature scheme. The routines \text{chebft} and \text{chint}, used in that order, can be followed by repeated calls to \text{chebev} if \( \int_a^b f(x) \, dx \) is required for many different values of \( x \) in the range \( a \leq x \leq b \).

If only the single definite integral \( \int_a^b f(x) \, dx \) is required, then \text{chint} and \text{chebev} are replaced by the simpler formula, derived from equation (5.9.1),

\[
\int_a^b f(x) \, dx = (b - a) \left[ \frac{1}{2} c_1 - \frac{1}{3} c_3 - \frac{1}{5} c_5 - \cdots - \frac{1}{(2k + 1)(2k - 1)} c_{2k + 1} - \cdots \right]
\]

(5.9.3)

where the \( c_i \)'s are as returned by \text{chebft}. The series can be truncated when \( c_{2k + 1} \) becomes negligible, and the first neglected term gives an error estimate.

This scheme is known as Clenshaw-Curtis quadrature\footnote{Clenshaw, C.W., and Curtis, A.R. 1960, Numerische Mathematik, vol. 2, pp. 197–205. \cite{Clenshaw1960}}. It is often combined with an adaptive choice of \( N \), the number of Chebyshev coefficients calculated via equation (5.8.7), which is also the number of function evaluations of \( f(x) \). If a modest choice of \( N \) does not give a sufficiently small \( c_{2k + 1} \) in equation (5.9.3), then a larger value is tried. In this adaptive case, it is even better to replace equation (5.8.7) by the so-called “trapezoidal” or Gauss-Lobatto (§4.5) variant,

\[
c_j = \sum_{k=0}^{N} f \left( \cos \left( \frac{\pi k}{N} \right) \cos \left( \frac{\pi (j - 1) k}{N} \right) \right) \quad j = 1, \ldots, N
\]

(5.9.4)

where (N.B.) the two primes signify that the first and last terms in the sum are to be multiplied by 1/2. If \( N \) is doubled in equation (5.9.4), then half of the new function evaluation points are identical to the old ones, allowing the previous function evaluations to be reused. This feature, plus the analytic weights and abscissas (cosine functions in 5.9.4), give Clenshaw-Curtis quadrature an edge over high-order adaptive Gaussian quadrature (cf. §4.5), which the method otherwise resembles.

If your problem forces you to large values of \( N \), you should be aware that equation (5.9.4) can be evaluated rapidly, and simultaneously for all the values of \( j \), by a fast cosine transform. (See §12.3, especially equation 12.3.17.) (We already remarked that the nontrapezoidal form (5.8.7) can also be done by fast cosine methods, cf. equation 12.3.22.)

CITED REFERENCES AND FURTHER READING:


5.10 Polynomial Approximation from Chebyshev Coefficients

You may well ask after reading the preceding two sections, “Must I store and evaluate my Chebyshev approximation as an array of Chebyshev coefficients for a transformed variable $y$? Can’t I convert the $c_k$’s into actual polynomial coefficients in the original variable $x$ and have an approximation of the following form?”

$$f(x) \approx \sum_{k=1}^{m} g_k x^{k-1} \quad (5.10.1)$$

Yes, you can do this (and we will give you the algorithm to do it), but we caution you against it: Evaluating equation (5.10.1), where the coefficient $g$’s reflect an underlying Chebyshev approximation, usually requires more significant figures than evaluation of the Chebyshev sum directly (as by chebft). This is because the Chebyshev polynomials themselves exhibit a rather delicate cancellation: The leading coefficient of $T_n(x)$, for example, is $2^{n-1}$; other coefficients of $T_n(x)$ are even bigger; yet they all manage to combine into a polynomial that lies between ±1. Only when $m$ is no larger than 7 or 8 should you contemplate writing a Chebyshev fit as a direct polynomial, and even in those cases you should be willing to tolerate two or so significant figures less accuracy than the roundoff limit of your machine.

You get the $g$’s in equation (5.10.1) from the $c$’s output from chebft (suitably truncated at a modest value of $m$) by calling in sequence the following two procedures:

```fortran
SUBROUTINE chebpc(c,d,n)
INTEGER n,NMAX
REAL c(n),d(n)
PARAMETER (NMAX=50)  ! Maximum anticipated value of n.

Chebyshev polynomial coefficients. Given a coefficient array c(1:n) of length n, this routine generates a coefficient array d(1:n) such that \( \sum_{k=1}^{n} d_k y^{k-1} = \sum_{k=1}^{n} c_k T_{k-1}(y) - c_1/2 \). The method is Clenshaw’s recurrence (5.8.11), but now applied algebraically rather than arithmetically.

INTEGER j,k
REAL sv,dd(NMAX)

do 11 j=1,n
   d(j)=0.
   dd(j)=0.
11 enddo

d(1)=c(n)

do 13 j=n-1,2,-1
   do 12 k=n-j+1,2,-1
      sv=d(k)
      d(k)=2.*d(k-1)-dd(k)
      dd(k)=sv
12   sv=d(1)
   d(1)=-dd(1)+c(j)
   dd(1)=sv
13 enddo

do 14 j=n-2,-1
   d(j)=d(j-1)-dd(j)
14 enddo

d(1)=-dd(1)+0.5*c(1)
return
END
```