JOURNAL OF COMPUTATIONAL PHYSICS 57, 439–453 (1985)

Computation of Spherical Harmonic Expansion Coefficients via FFT's

GARY A. DILTS

Cooperative Institute for Research in Environmental Sciences, University of Colorado, Boulder, Colorado 80309

Received July 27, 1983; revised March 15, 1984

A method for numerically expanding an arbitrary function on the sphere in a series of spherical harmonics which makes use of the speed of a fast Fourier transform is described. Discussions of the operation count, storage requirements, accuracy, and an algebraic and a numerical example are included. A comparison with straightforward integration is made throughout. Also, a new method for evaluating the spherical harmonics is discussed. © 1985 Academic Press, Inc.

1. INTRODUCTION

It need hardly be said that the functions variously known as "spherical harmonics," "surface zonal harmonics," etc., described by the equation

$$Y_{lm}(\theta, \phi) \equiv C_{lm} P_l^m(\cos\theta)\, e^{im\phi}, \tag{1}$$

where $P_l^m(x)$ is the associated Legendre polynomial of degree l and order m and C_{lm} is a normalization constant given by

$$C_{lm} = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}}, \tag{2}$$

are of great utility [1] in many calculations in theoretical physics. These functions form an orthonormal set of basis functions on the unit sphere. That is, if

$$f\colon [0, \pi] \times [-\pi, \pi] \to R$$

with

$$\begin{aligned} f(\theta, -\pi) &= f(\theta, \pi) && \forall \theta \in [0, \pi] \\ f(0, \phi_1) &= f(0, \phi_2) && \forall \phi_1, \phi_2 \in [-\pi, \pi] \\ f(\pi, \phi_1) &= f(\pi, \phi_2) && \forall \phi_1, \phi_2 \in [-\pi, \pi] \end{aligned}$$

0021-9991/85 $3.00

and f is square integrable on $[0, \pi] \times [-\pi, \pi]$, then

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} F_{lm} Y_{lm}(\theta, \phi), \tag{3}$$

where F_{lm} is a complex constant given by

$$F_{lm} = \int_{-\pi}^{\pi} \int_{0}^{\pi} Y_{lm}^{*}(\theta, \phi) f(\theta, \phi) \sin\theta \, d\theta \, d\phi, \tag{4}$$

with * denoting complex conjugation, and

$$\int_{-\pi}^{\pi} \int_{0}^{\pi} Y_{l'm'}^{*} Y_{lm} \sin\theta \, d\theta \, d\phi = \delta_{ll'} \delta_{mm'}. \tag{5}$$

Expansions such as (3) will be referred to as *spherical harmonic expansions*, and F_{lm} will be called a *spherical harmonic expansion coefficient*. The aim of this paper is to present a method for numerically computing (4) for a given function of solid angle $f(\theta, \phi)$ which utilizes the speed of a two-dimensional fast Fourier transform (FFT).

To the author's knowledge, the only ways of computing (4) seem to be straightforward computation by two-dimensional integration and a method developed by W. Freeden [2, 12–14] which is analogous to Gaussian quadrature in one dimension. Freeden's method requires the solution of a linear system of equations for each degree l and each set of N^2 mesh points of dimension $N^2 + 2l + 1$. It will be shown that the method of this paper takes fewer operations than either straightforward integration or Freeden's method, making use of a two-dimensional FFT only once for all orders l and m, the remaining operations being only the evaluation of a double sum of roughly $(2l + 1)N$ terms, where N^2 corresponds to the number of mesh points in the integration method and in Freeden's procedure.

2. Derivation of the Algorithm

To begin the evaluation of (4) assume that the function f is represented by a finite Fourier series (for an arbitrary f a FFT approximates f by such a series),

$$f(\theta, \phi) = \sum_{a=-N}^{N} \sum_{b=-N}^{N} f_{ab} e^{i(a\theta + b\phi)}. \tag{6}$$

Suppose we could also represent the Y_{lm} by a finite Fourier series (this is the whole trick to the procedure) as follows,

$$Y_{lm}(\theta, \phi) = \sum_{j=-l}^{l} B_j^{lm} e^{i(j\theta + m\phi)}. \tag{7}$$

Then by (4)

$$F_{lm}=\int_{-\pi}^{\pi}\int_{0}^{\pi}\left[\sum_{j=-l}^{l} B_j^{lm*}e^{-i(j\theta+m\phi)}\right]\times\sum_{a=-N}^{N}\sum_{b=-N}^{N} f_{ab}e^{i(a\theta+b\phi)}\sin\theta\, d\theta\, d\phi.$$

Express $\sin\theta$ as $(1/2i)(e^{i\theta}-e^{-i\theta})$ and perform the integrations over θ and ϕ to get

$$\begin{aligned}F_{lm}=&\sum_{j=-l}^{l}\sum_{a,b} B_j^{lm*}f_{ab}\left(\frac{2\pi}{2i}\delta_{mb}\right)\\ &\times\left[\pi(\delta_{a,j-1}-\delta_{a,j+1})-(1-\delta_{a,j-1})\frac{i[(-1)^{a-j+1}-1]}{a-j+1}\right.\\ &\left.+(1-\delta_{a,j+1})\frac{[(-1)^{a-j-1}-1]}{a-j-1}\right].\end{aligned}$$

Summing over b and rearranging the sum over a we obtain

$$\begin{aligned}F_{lm}=-\pi\sum_{j=-l}^{l} B_j^{lm*}&\left[2\sum_{a=-N}^{j-2}\frac{[(-1)^{a-j}+1]}{(a-j)^2-1}f_{am}+i\pi f_{j-1,m}-4f_{j,m}-i\pi f_{j+1,m}\right.\\ &\left.+2\sum_{a=j+2}^{N}\frac{[(-1)^{a-j}+1]}{(a-j)^2-1}f_{am}\right].\end{aligned}\tag{8}$$

It is understood that when $j=\pm N$, $\pm(N-1)$ the sums over a for which the lower limit exceeds the upper limit are omitted.

The algorithm then, is

Step 1
Obtain an approximation for f of the form (6) via a two-dimensional FFT.

Step 2 (9)
Evaluate the sums in (8).

The computation of the constants B_j^{lm} in (7), described in detail below, is not part of the algorithm, since they are independent of $f(\theta,\phi)$, needing to be calculated only once for all time.

3. Operation Count

It is now shown that in almost all cases of interest the present method has a smaller operation count than that of straightforward quadrature. Let n_f be the number of operations required to evaluate f; n_{lm} the number of operations to evaluate

$Y_{lm} \sin\theta$ by (7), which is shown in Section 7 to be approximately $20l$; $n_i N$ the number of operations required to perform a one-dimensional integration over an interval with N mesh points. For Newton–Cotes integration $n_i \approx 2$. It is assumed that N^2 mesh points in two dimensions will be used in both the FFT and the integration procedure. Then the total number of operations to evaluate (4) by quadratures is

$$N^2(n_f + n_{lm}) + n_i N^2 + n_i N \approx n_f N^2 + 20lN^2 + n_i N^2.$$

The number of operations in the evaluation of (8) is approximately $40lN$. The number of operations for a two-dimensional FFT is $n_{\mathrm{FFT}} N^2 \log N$, where n_{FFT} is a constant $\lesssim 25$ [3]. The total number of operations by the FFT method is

$$n_f N^2 + n_{\mathrm{FFT}} N^2 \log N + 40lN.$$

The FFT method takes fewer operations than quadrature when

$$n_{\mathrm{FFT}} N^2 \log N + 40lN \leqslant 20lN^2 + n_i N^2$$

or when

$$l \geqslant \frac{n_{\mathrm{FFT}} N \log N - n_i N}{20N - 40}.$$

Taking $n_{\mathrm{FFT}} \approx 25$, $n_i \approx 2$, we have

$$l \geqslant \frac{5}{4} N \left(\frac{\log N - 2/25}{N - 2} \right) \equiv l_N.$$

For $N = 10, 100, 1000$, l_N has the values 4, 7, 9, respectively, when rounded up to the nearest integer. The conclusion is that the algorithm (9), since (6) and (7) together imply one would be interested in all $l \leqslant N$, takes fewer operations than straightforward quadrature for most values of l of interest. Note though that for both the minimum number of operations is $n_f N^2$.

In contrast, the solution of a linear system of order n takes on the order of n^3 operations [6, Chap. 4]. Thus Freeden's scheme apparently requires much more computational effort. However, this extra work pays a dividend of being able to handle much more general mesh arrangements than the simple rectangular grid used in this paper. The method of this paper would be useful as a high-speed formula for obtaining the spherical harmonic expansion coefficients of a user-specified, easily computable function f, with not too much angular variation. Freeden's method, on the other hand, would be much more applicable to problems of empirical data interpolation, on the surface of the Earth, for example. Freeden's method also allows the possibility of computing error estimates.

Remaining questions concerning the justification of (7), the computation of the B_j^{lm}, the storage of them, and error estimates are addressed below.

4. Determining the B_j^{lm}

We shall need the following standard results concerning associated Legendre functions [4, 10]:

$$P_0^0(x)=1, \qquad P_1^0(x)=x, \qquad P_1^1(x)=-\sqrt{1-x^2} \tag{10}$$

$$P_l^{-m}(x)=(-1)^m\frac{(l-m)!}{(l+m)!}P_l^m(x) \tag{11}$$

$$P_{l+1}^m=xP_l^m(x)-(l+m)\sqrt{1-x^2}\cdot P_l^{m-1}(x) \tag{12}$$

$$P_l^l(\cos\theta)=\frac{(-1)^l\,(2l)!}{l!}\left(\frac{\sin\theta}{2}\right)^l. \tag{13}$$

Formulas (10)–(13) show, by induction, that

$$P_l^m(\cos\theta)=\sum_{k=-l}^{l}a_k^{lm}e^{ik\theta}, \tag{14}$$

where a_k^{lm} is a complex constant. Alternatively, one can use Rodrigues' formula:

$$P_l^m(x)=(-1)^m\,(1-x^2)^{m/2}\frac{d^m}{dx^m}P_l(x).$$

It is useful to think of (12) as a connection between three points of a grid denoting the possible values of l and m as shown in Fig. 1. If (14) is substituted into (12) with $x=\cos\theta$, a recursion relation is obtained for the a_k^{lm} after equating sums of coefficients of like powers of $e^{i\theta}$ to zero. It can be seen from Fig. 2 that all possible values for l and m are covered if in addition (10), (11) and (13) with $x=\cos\theta$ are utilized. The algebra is tedious but straightforward, so only the final recursion relations for the a_k^{lm} are presented:

$$\begin{aligned}
a_{l+1}^{l+1,m}&=\tfrac{1}{2}[a_l^{lm}+i(l+m)\,a_l^{l,m-1}]\\
a_l^{l+1,m}&=\tfrac{1}{2}[a_{l-1}^{lm}+i(l+m)\,a_{l-1}^{l,m-1}]\\
a_k^{l+1,m}&=\tfrac{1}{2}[a_{k-1}^{lm}+a_{k+1}^{lm}+i(l+m)(a_{k-1}^{l,m-1}-a_{k+1}^{l,m-1})],\\
&\qquad k=-(l-1),\ldots,l-1\\
a_{-l}^{l+1,m}&=\tfrac{1}{2}[a_{-(l-1)}^{lm}-i(l+m)\,a_{-(l-1)}^{l,m-1}]\\
a_{-(l+1)}^{l+1,m}&=\tfrac{1}{2}[a_{-l}^{lm}-i(l+m)\,a_{-l}^{l,m-1}].
\end{aligned} \tag{15}$$

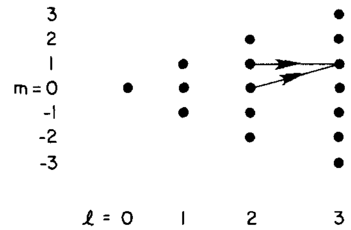

FIG. 1. Equation (12) connects points of $l-m$ space as shown by the solid lines.

The initialization, using (10), is

$$\begin{aligned} &a_0^{00}=1, \quad a_1^{10}=\frac{1}{2}, \quad a_0^{10}=0, \quad a_{-1}^{10}=\frac{1}{2} \\ &a_1^{11}=\frac{i}{2}, \quad a_0^{11}=0, \quad a_{-1}^{11}=\frac{-i}{2}. \end{aligned} \tag{16}$$

The coefficients a_k^{ll} can be determined from (13) and the binomial theorem:

$$a_{l-2j}^{ll}=\frac{i^l(2l)!}{4^l(l-j)!\,j!}, \qquad \text{and} \quad a_k^{ll}=0 \text{ otherwise.} \tag{17}$$

It is shown in Section 6 how the a_k^{lm} can be computed using mostly integer arithmetic. These recursion relations can be solved numerically by machine. Once that is done, from (1), (7) and (14) we have

$$B_k^{lm}=C_{lm}a_k^{lm}. \tag{18}$$

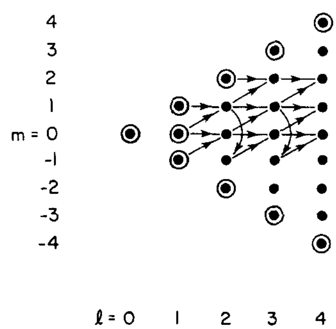

FIG. 2. Application of the recursive procedure (15). Circled dots indicate points at which (13) is used. For $m \leqslant -1$, a_k^{lm} is found from a_k^{l-m} by (11).

TABLE I

l	m	k	Blmk	l	m	k	Blmk
0	0	0	(2.8209480643272400e-01, 0.0000000000000000e+00)				
1	0	-1	(2.4430125951766968e-01, 0.0000000000000000e+00)				
1	0	0	(0.0000000000000000e+00, 0.0000000000000000e+00)				
1	0	1	(2.4430125951766968e-01, 0.0000000000000000e+00)				
1	1	-1	(0.0000000000000000e+00,-1.7274707473566775e-01)	1	-1	-1	(0.0000000000000000e+00, 1.7274707473566775e-01)
1	1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)	1	-1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
1	1	1	(0.0000000000000000e+00, 1.7274707473566775e-01)	1	-1	1	(0.0000000000000000e+00,-1.7274707473566775e-01)
2	0	-2	(2.3654367029666901e-01, 0.0000000000000000e+00)				
2	0	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)				
2	0	0	(1.5769578516483307e-01, 0.0000000000000000e+00)				
2	0	1	(0.0000000000000000e+00, 0.0000000000000000e+00)				
2	0	2	(2.3654367029666901e-01, 0.0000000000000000e+00)				
2	1	-2	(0.0000000000000000e+00,-1.9313710101159480e-01)	2	-1	-2	(0.0000000000000000e+00, 1.9313710101159480e-01)
2	1	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)	2	-1	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)	2	-1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	1	1	(0.0000000000000000e+00, 0.0000000000000000e+00)	2	-1	1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	1	2	(0.0000000000000000e+00, 1.9313710101159480e-01)	2	-1	2	(0.0000000000000000e+00,-1.9313710101159480e-01)
2	2	-2	(-9.6568547189235687e-02, 0.0000000000000000e+00)	2	-2	-2	(-9.6568547189235687e-02, 0.0000000000000000e+00)
2	2	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)	2	-2	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	2	0	(1.9313709437847137e-01, 0.0000000000000000e+00)	2	-2	0	(1.9313709437847137e-01, 0.0000000000000000e+00)
2	2	1	(0.0000000000000000e+00, 0.0000000000000000e+00)	2	-2	1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	2	2	(-9.6568547189235687e-02, 0.0000000000000000e+00)	2	-2	2	(-9.6568547189235687e-02, 0.0000000000000000e+00)
3	0	-3	(2.3323521018028259e-01, 0.0000000000000000e+00)				
3	0	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)				
3	0	-1	(1.3994112610816956e-01, 0.0000000000000000e+00)				
3	0	0	(0.0000000000000000e+00, 0.0000000000000000e+00)				
3	0	1	(1.3994112610816956e-01, 0.0000000000000000e+00)				
3	0	2	(0.0000000000000000e+00, 0.0000000000000000e+00)				
3	0	3	(2.3323521018028259e-01, 0.0000000000000000e+00)				
3	1	-3	(0.0000000000000000e+00,-2.0198761507134417e-01)	3	-1	-3	(0.0000000000000000e+00, 2.0198761507134417e-01)
3	1	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-1	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	1	-1	(0.0000000000000000e+00,-4.0397523014268834e-02)	3	-1	-1	(0.0000000000000000e+00, 4.0397523014268834e-02)
3	1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	1	1	(0.0000000000000000e+00, 4.0397523014268834e-02)	3	-1	1	(0.0000000000000000e+00,-4.0397523014268834e-02)
3	1	2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-1	2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	1	3	(0.0000000000000000e+00, 2.0198761507134417e-01)	3	-1	3	(0.0000000000000000e+00,-2.0198761507134417e-01)
3	2	-3	(-1.2774819135665894e-01, 0.0000000000000000e+00)	3	-2	-3	(-1.2774819135665894e-01, 0.0000000000000000e+00)
3	2	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-2	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	2	-1	(1.2774819135665894e-01, 0.0000000000000000e+00)	3	-2	-1	(1.2774819135665894e-01, 0.0000000000000000e+00)
3	2	0	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-2	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	2	1	(1.2774819135665894e-01, 0.0000000000000000e+00)	3	-2	1	(1.2774819135665894e-01, 0.0000000000000000e+00)
3	2	2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-2	2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	2	3	(-1.2774819135665894e-01, 0.0000000000000000e+00)	3	-2	3	(-1.2774819135665894e-01, 0.0000000000000000e+00)
3	3	-3	(0.0000000000000000e+00, 5.2152977954098013e-02)	3	-3	-3	(0.0000000000000000e+00,-5.2152977954098013e-02)
3	3	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-3	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	3	-1	(0.0000000000000000e+00,-1.5645893386229404e-01)	3	-3	-1	(0.0000000000000000e+00, 1.5645893386229404e-01)
3	3	0	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-3	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	3	1	(0.0000000000000000e+00, 1.5645893386229404e-01)	3	-3	1	(0.0000000000000000e+00,-1.5645893386229404e-01)
3	3	2	(0.0000000000000000e+00, 0.0000000000000000e+00)	3	-3	2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	3	3	(0.0000000000000000e+00,-5.2152977954098013e-02)	3	-3	3	(0.0000000000000000e+00, 5.2152977954098013e-02)

The Y_{lm} for $l=0, 1, 2, 3$ are exhibited in [5]. From this list one can determine the B_k^{lm} by hand for $l=0, 1, 2, 3$. Table I shows the results of computing the B_k^{lm} for these degrees l by the scheme of (15)–(18) in FORTRAN double-precision arithmetic on a PDP 11/70 with a UNIX operating system. The underlined digits indicate agreement with hand calçulations made on a 10-digit/visible, 14-digit/internal calculator. This agreement is uniformly 8 digits or better, which attests to the accuracy of the procedure. Note that (11) implies one need only calculate a_k^{lm} for values of $m \geqslant 0$, and that $a_k^{l,-1}$, which is needed in (15) when $m=0$, can be found from $a_k^{l,1}$. This is also indicated in Fig. 2.

5. Storage of the B_k^{lm}

At first glance it might seem that the great numbers of B_k^{lm}'s needed to be stored in memory for the execution of the algorithm (9) would be a handicap. Since m has a range of $-l,\ldots, l$ for each l, and k has a range of $-l,\ldots, l$ for each l and m, the total number of real storage spaces (remember B_k^{lm} is complex) seemingly required is

$$2 \cdot \sum_{l=0}^{L} (2l+1)^2 = \frac{2}{3}(L+1)[4(L+1)^2 - 1], \tag{19}$$

where L is the maximum value of l being considered. Some values for various L are listed in Table II. As one can see the demands on memory become quite severe for even modest l. However, examination of Table I reveals some remarkable symmetries and regularities of the B_k^{lm}, which below will be proven to be true in general, which decrease these demands by a factor of 16. They are:

(S1) $B_k^{l,-m} = (-1)^m (B_{-k}^{lm})^*$.

(S2) $B_{-k}^{l,m} = (B_k^{lm})^*$.

(S3) If m is odd B_k^{lm} is pure imaginary, while if m is even B_k^{lm} is real.

(S4) $B_l^{lm} \neq 0,\ B_{l-1}^{lm} = 0,\ B_{l-2}^{lm} \neq 0, \ldots, \begin{cases} B_0^{lm} = 0, & l \text{ odd}, \\ B_0^{lm} \neq 0, & l \text{ even}, m \text{ even}. \end{cases}$

That is, $B_{l-2i}^{lm} \neq 0$, and $B_{l-(2i+1)}^{lm} = 0$, for $i = 0, \ldots, [(l-1)/2]$, where $[x]$ denotes the greatest integer less than or equal to x.

(S5) $B_0^{lm} = 0$ if l even, m odd.

Before proving these note the effect on the number of B_k^{lm}'s needed to be stored. S1 and S2 imply one only need consider $m, k \geqslant 0$. S3 implies one only needs real storage locations. S4 implies that one only need consider half the positive values of k. Thus one expects a reduction by a factor of 16 in the number of real storage locations needed. With careful consideration of Fig. 3 and using the formula for the sum of the squares of consecutive integers it can be shown that the total number of real storage spaces that one needs, if L is the maximum degree l being considered, is

$$\sum_{j=1}^{(L+1)/2} (4j-1)\, j = (4L^3 + 21L^2 + 32L + 15)/24 \tag{20}$$

if L is odd and

$$\sum_{j=1}^{L/2} (4j-1)\, j + \left(\frac{L}{2} + 1\right)(L+1) = \frac{4L^3 + 21L^2 + 38L + 24}{24} \tag{21}$$

TABLE II

L	$2/3(L+1)(4(L+1)^2 - 1)$
3	168
9	2660
17	15540
33	104788

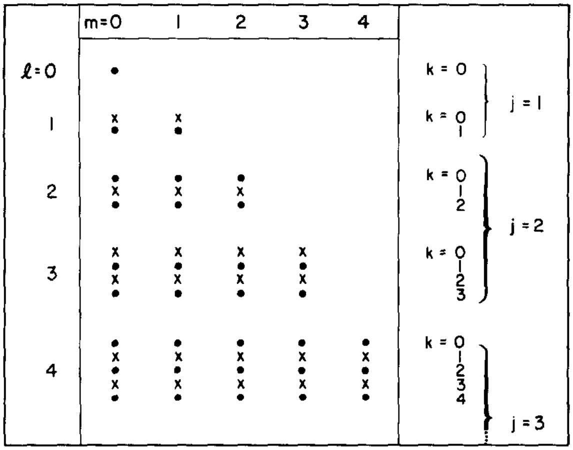

FIG. 3. Diagram of memory locations for storage of B_k^{lm}. Dots represent locations where B_k^{lm} is nonzero and crosses represent locations where B_k^{lm} is zero. For each j there are $4j-1$ dots, where j labels successive pairs of values of l. For each l there are $(l+1)j$ dots.

if L is even, which indeed shows a factor of 16 improvement over (19). Some values of the right-hand side of (20) are listed in Table III. These are much more manageable numbers.

The operational memory requirements may be even further diminished by noting that the calculation of the sum (8) requires, for a fixed l and m, only B_k^{lm}, $k=-l,\ldots,l$. Thus, for example, if one wanted to compute F_{lm} for several degrees l, one need only bring into memory the B_k^{lm} for the particular l being considered at the time, which requires on the order of $(l+1)\,l/2$ storage locations by Fig. 3.

Property S1 is derived by observing that (1), (2) and (11) imply that

$$Y_{l,-m} = (-1)^m Y_{lm}^*$$

whence

$$\sum_{j=-l}^{l} B_j^{l,-m} e^{i(j\theta - m\phi)} = (-1)^m \sum_{j=-l}^{l} B_j^{lm*} e^{-i(j\theta + m\phi)},$$

from which S1 follows by linear independence of the $e^{ij\theta}$.

TABLE III

L	$(4L^3+21L^2+32L+15)/24$
3	17
9	205
17	1095
33	6987

Property S2 follows from the observation that $P_l^m(\cos\theta)$ is real. For then, by (14),

$$\sum_{k=-l}^{l} a_k^{lm} e^{ik\theta} = \sum_{k=-l}^{l} (a_k^{lm})^* \, e^{-ik\theta},$$

from which S2 follows, using (18).

Properties S3 and S4 follow from (15), (16) and induction on l. Only $k \geqslant 0$ and $m \geqslant -1$ need be considered since S3 and S4 are preserved under the symmetries of S1 and S2. S3 is evident from (15), but S4 deserves comment. First note that S4 is true for $l=0$, $l=1$. Assume that it is true for l, and $m=0,..., l$. Then (15) shows that $a_l^{l+1,m}=0$, and if $k=l+1-(2i+1)=l-2i$ then $a_k^{l+1,m}=0$, for any m. Thus S4 is true for $l+1$.

S5 follows immediately after setting $k=0$ in (15) and using S2 and S3.

6. Errors

Observe that if both sides of (15) are multiplied by 2^{l+1} a recursion relation for the quantities $\tilde{a}_k^{lm} = 2^l a_k^{lm}$ is obtained which is the same as (15) except the factor of $\frac{1}{2}$ on the right side is missing. Next, observe that (13) can be rewritten as $(2l-1)!!$ $(-\sin\theta)^l$, which means $\tilde{a}_k^{ll}$ is an integral complex number. Thus the recursion procedure for the $\tilde{a}_k^{lm}$ is performed *entirely* in integer arithmetic, because the initial values a_0^{00}, $2a_k^{1m}$ are integers by (16). Hence, the only error in a_k^{lm} is the round-off error incurred by the integer division in $\tilde{a}_k^{lm}/2^l$. The only remaining step in calculating the B_k^{lm} after finding the a_k^{lm} is (18). The error in C_{lm} can be assumed to be very small, therefore the B_k^{lm} can be assumed to be determined with very high accuracy. In the evaluation of (4) then, the only significant errors come from the truncation of the Fourier series of f in (6) to $(2N+1)^2$ terms, any error that might occur in the FFT procedure in determining the f_{ab}, and round-off errors occurring in the evaluation of (8). The relative error due to the truncation of the Fourier series of f is $O(1/N^{2r+2})$, if f is r times continuously differentiable, by a two-dimensional extension of the argument in [6]. Gentleman [7] states that the relative rms error for an FFT is bounded by $1.06 \times \sum_i (2n_i)^{3/2} \times \text{eps}$, where $2N+1=\prod_i n_i$, each n_i is prime, and "eps" is the machine precision. It is conjectured that the evaluation of the sum (8) is numerically stable since the f_{am} decrease as $1/(am)^{r+1}$ as in [6], and the coefficients of f_{am} decrease as $1/a^2$. Thus, it appears that the major source of error in the algorithm (9) is the FFT itself.

7. Evaluation of $Y_{lm}(\theta, \phi)$

To obtain $f(\theta, \phi)$ from the F_{lm} one uses (3), which requires the evaluation of $Y_{lm}(\theta, \phi)$. Equation (7) provides a means to do this. If the Goertzel–Reinsch algorithm [8] is used to evaluate (7) the operation count is about $20l$, and numerical stability is guaranteed.

The definition (1) can be used as well to evaluate $Y_{lm}(\theta, \phi)$, however, it requires knowledge of $P_l^m(\cos\theta)$, which can be obtained by means of a recursion relation such as

$$(l-m+1)\, P_{l+1}^m(x) = (2l+1)\, x P_l^m(x) - (l+m)\, P_{l-1}^m(x) \tag{22}$$

with perhaps (13) as an initialization. The operation count for this method is roughly $10l$-$8m$, about twice as fast as (7). Reference [11] shows that this recursion scheme is stable and also describes codes based on it which calculate the associated Legendre functions in extended-precision arithmetic.

Since $P_l^{-m}(x)$ is proportional to $P_l^m(x)$, the equation

$$\frac{dP_l^{-m}(x)}{dx} P_l^m(x) - P_l^{-m}(x) \frac{dP_l^m(x)}{dx} = 0 \tag{23}$$

holds. Using the expression [4]

$$\frac{dP_l^m(x)}{dx} = (l+1)\, x P_l^m(x) - (l-m+1)\, P_{l+1}^m(x)$$

and various recurrence relations for the P_l^m, one can consider (23) a check on the accuracy of (7) or (22). A similar idea was used in [11].

Codes based on (7) and (22) were written, and programs to compare the two methods and to check (23) were run for all values of the degree $l \leqslant 15$. The agreement of (23) with zero was uniformly better than 16 decimal places for each method, as was the agreement of the two methods with each other. The computations were performed on a DEC VAX-11/780 in FORTRAN DOUBLE-PRECISION arithmetic.

8. Examples

This paper will close with a few examples.

First, a hand example is considered. Let $f(\theta, \phi) = Y_{11}(\theta, \phi)$. By (4) and (5), (8) should give

$$F_{00} = F_{1,-1} = F_{1,0} = F_{l,m} = 0 \qquad \text{for} \quad l \geqslant 2, |m| \leqslant l$$
$$F_{11} = 1.$$

That this is so shall be shown presently. Let $N = \infty$ in (8).

Observe that since

$$Y_{11}(\theta, \phi) = \frac{i}{2}\sqrt{\frac{3}{8\pi}}\,(e^{i\theta} - e^{-i\theta})\, e^{i\phi}$$

we have

$$f_{1,1}=\frac{i}{2}\sqrt{\frac{3}{8\pi}}, \qquad f_{-1,1}=\frac{-i}{2}\sqrt{\frac{3}{8\pi}}, \qquad \text{all other} \quad f_{ab}=0. \tag{24}$$

We immediately see $F_{lm}=0$ for all $m\neq 1$ in (8). The first sum over a contributes nothing if $j<1$ and the second sum if $j>-1$. Thus (8) becomes, for $l\geqslant 2$,

$$\begin{aligned} F_{l,1}= -\pi\Bigg[&\sum_{j=1}^{l} 2B_j^{l,1*} \sum_{a=-\infty}^{j-2} \frac{[(-1)^{a-j}+1]}{(a-j)^2-1} f_{a,1} \\ &+ i\pi B_0^{l,1*} f_{-1,1} + i\pi B_2^{l,1*} f_{1,1} - 4B_{-1}^{l,1}{}^{*} f_{-1,1} \\ &- 4B_1^{l,1*} f_{1,1} - i\pi B_{-2}^{l,1}{}^{*} f_{-1,1} + -i\pi B_0^{l,1*} f_{1,1} \\ &+ \sum_{j=-l}^{-1} 2B_j^{l,1*} \sum_{a=j+2}^{\infty} \frac{[(-1)^{a-j}+1]}{(a-j)^2-1} f_{a,1}\Bigg]. \end{aligned} \tag{25}$$

If $l=2$ this gives, using S1, S4, and (24),

$$\begin{aligned} F_{21}= -\pi[&\tfrac{4}{3}B_1^{21} f_{-1,1} \\ &+ i\pi B_0^{21*}(f_{-1,1}-f_{1,1}) + i\pi(B_2^{2,1*}+B_2^{2,1})(f_{-1,1}+f_{1,1}) - 4(B_{+1}^{21}{}^{*} - B_{-1}^{21}{}^{*}) f_{1,1} \\ &+ \tfrac{4}{3}B_{-1}^{21} f_{1,1}], \end{aligned}$$

which is zero by S1–S5. If $l\geqslant 3$ (25) becomes

$$\begin{aligned} F_{l1}= -\pi\Bigg[&2\sum_{j=3}^{l} B_j^{l1*}(\alpha_- f_{-1,1}+\alpha_+ f_{1,1}) \\ &+ \tfrac{4}{3}B_1^{l1*} f_{-1,1} + i\pi B_0^{l1*}(f_{-1,1}-f_{1,1}) + \tfrac{4}{3}B_{-1}^{l1}{}^{*} f_{1,1} \\ &+ 2\sum_{j=-l}^{-3} B_j^{l1*}(\alpha_- f_{-1,1}+\alpha_+ f_{1,1})\Bigg], \end{aligned}$$

where $\alpha_\pm=[(-1)^{1-j}+1]/[(\pm 1-j)^2-1]$. If l is odd $B_0^{l1*}=0$ by S4. If l is even $B_0^{l1*}=0$ by S5. Since $f_{-1,1}=-f_{1,1}$ the middle terms are proportional to

$$\begin{aligned} B_1^{l1*} f_{-1,1} + B_{-1}^{l1}{}^{*} f_{1,1} &= (-B_1^{l1*} + B_{-1}^{l1}{}^{*}) f_{1,1} \\ &= (B_{-1}^{l1} + B_{-1}^{l1}{}^{*}) f_{1,1}, \end{aligned}$$

which is zero, by S3.

The remaining terms give, also using (24),

$$F_{l1}=2\pi\sum_{j=3}^{l} B_j^{l1}[-\alpha_- + \alpha_+ - \alpha_+ + \alpha_-] f_{1,1},$$

TABLE IV

l	m	Flm
0	0	(-8.1381983767419099e-17, 9.9587350402627157e-18)
1	-1	(8.0116929555045708e-17,-9.7780130099236597e-18)
1	0	(2.2828045937564377e-19,-1.1414022968782386e-19)
1	1	(9.9944979319713916e-01, 1.2776187076386269e-17)
2	-2	(-7.1285329282950260e-17,-1.1784978715267913e-17)
2	-1	(3.0437394583420775e-19,-6.0074789166840060e-19)
2	0	(2.1125454178054911e-17,-3.2625082319103262e-18)
2	1	(1.1262908691486206e-17, 1.0677115941351754e-17)
2	2	(-8.4568398512870875e-17,-8.5070139939206939e-18)
3	-3	(7.4315395692944235e-17,-2.3647239877720289e-17)
3	-2	(1.1794490401075142e-18,-1.6169865872442015e-19)
3	-1	(-4.1851417552202415e-19, 2.4730383099028704e-19)
3	0	(7.2288812135623564e-19,-4.2802586132935490e-20)
3	1	(-2.8609315859251788e-03,-2.5496146320160456e-17)
3	2	(-1.9023371614638539e-19,-2.4849279171620209e-18)
3	3	(-6.9800317336282625e-17,-1.9107193345814467e-17)

which is zero, as required. For $l=1$ we have, instead of (25),

$$\begin{aligned} F_{11} &= -\pi\left[\tfrac{4}{3}B_1^{11}{}^* f_{-1,1} + i\pi B_0^{11}{}^* f_{-1,1} - 4B_{-1}^{11}{}^* f_{-1,1} - 4B_1^{11}{}^* f_{1,1} \right. \\ &\qquad \left. - i\pi B_0^{11}{}^* f_{1,1} + \tfrac{4}{3}B_{-1}^{11}{}^* f_{1,1}\right] \\ &= -\pi\left[\tfrac{-4}{3}B_1^{11}{}^* + 4B_{-1}^{11}{}^* - 4B_1^{11}{}^* + \tfrac{4}{3}B_{-1}^{11}{}^*\right] f_{11} \\ &= -\pi\left[\tfrac{-4}{3} - 4 - 4 - \tfrac{4}{3}\right] B_1^{11}{}^* f_{11} \\ &= \pi\tfrac{32}{3}\cdot |f_{11}|^2 = 1, \qquad \text{since} \quad B_1^{11} = f_{11}. \end{aligned}$$

The sum (8) is validated for this example.

Now some numerical examples are in order. Table IV shows the result of calculating (4) by integration over the sphere using a two-dimensional extension of Simpson's rule, for the case $f = Y_{11}(\theta, \phi)$. The machine used in this and the following computations is the same that produced Table I. The domain of θ, $[0, \pi]$ was divided into 10 equal intervals and the domain of ϕ, $[-\pi, \pi]$, into 20. This corresponds to N between 10 and 14. Values for F_{11}, F_{31} are correct only to 3 decimal places, all other coefficients to 16. Table V shows the result of calculating (4) by algorithm (9) for the same function f. F_{11} has 7-decimal place accuracy; all

TABLE V

l	m	Flm
0	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
1	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)
1	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
1	1	(9.9999994039535522e-01, 0.0000000000000000e+00)
2	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	1	(0.0000000000000000e+00, 0.0000000000000000e+00)
2	2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	-3	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	-2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	-1	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	0	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	1	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	2	(0.0000000000000000e+00, 0.0000000000000000e+00)
3	3	(0.0000000000000000e+00, 0.0000000000000000e+00)

other values are exact. It is remarkable, in light of the many cancellations occurring in (8), that the zero values are exactly zero and not just within the machine "eps" of zero, as is the case in Table IV. These two tables represent the maximum attainable accuracy for each method, since for Table V the values f_{lm} were not obtained by a FFT, but were obtained from a program which calculates the B_k^{lm}. The proposed method is evidently superior in accuracy, although the accuracy of the integration procedure could be improved by using a method of higher order than Simpson's rule. As for the computational time for the two methods, Table IV took 69.7 seconds to make, while Table V took only about 0.7 seconds to make. If one includes the time it takes to produce the f_{ij} by a FFT, which is about 4.7 seconds, the time required to produce Table V by algorithm (9) is less than 6 seconds, as opposed to 70 seconds for integration. The FFT used was written by R. C. Singleton, and is described in [9].

9. Conclusion

It appears that algorithm (9) has two major advantages over integration: speed and accuracy. The major drawback is that it cannot be applied to problems with irregular spacing of the data f over the sphere, for then a FFT cannot be applied in the determination of the f_{ij}. Another method for finding them must be used, such as Goertzel–Reinsch. In this case it is conceivable that Freeden's method may be more desirable, since he also provides a way of minimizing the error for $2l+1<N$.

Acknowledgments

The author would like to thank J. A. Scales and C. B. Archambeau for suggesting the basic idea of the paper to him.

References

1. J. D. Jackson, "Classical Electrodynamics," Chaps. 3 and 16, Wiley, New York, 1975.
2. W. Freeden, *Computing* **25** (1980), 131.
3. J. W. Cooley, P. A. W. Lewis, and P.D. Welch, "The Fast Fourier Transform and Its Applications," IBM Research Paper RC-1743, IBM Watson Research Center, Yorktown Heights, N.Y., 1967.
4. I. S. Gradshteyn and I. M. Ryzhik, "Table of Integrals, Series, and Products," Sects. 8.73, 8.75, 8.81, and 8.91, Academic Press, New York/London, 1980.
5. Jackson, *op. cit.*, p. 100.
6. J. Stoer and R. Bulirsch, "Introduction to Numerical Analysis," p. 89, Springer-Verlag, New York, 1980.
7. W. M. Gentleman, *Comput. J.* **12** (1969), 160.
8. Stoer, *op. cit.*, p. 86.

9. Digital Signal Processing Committee, Eds., "Programs for Digital Signal Processing," p. 1.4-1, IEEE Press, New York, 1979.
10. A. ERDÉLYI, Ed., "Higher Transcendental Functions," Vol. 1, McGraw-Hill, New York, 1953.
11. F. W. J. OLVER AND J. M. SMITH, *J. Comput. Phys.* **51** (1983), 502–518.
12. W. FREEDEN, *Boll. Geodesia Sci. Affini* **1** (1982), 105–120.
13. W. FREEDEN, Report No. 341 of the Department of Geodetic Science and Surveying, The Ohio State University, Columbus, 1982.
14. W. FREEDEN AND R. REUTER, *Manuscripta Mathematica* **35** (1981), 371–386.