A Fast Transform for Spherical Harmonics

Martin J. Mohlenkamp

Accepted August 1998 by
The Journal of Fourier Analysis and Applications

Affiliation. Department of Mathematics, Yale University; and Department of Applied Mathematics, University of Colorado, Campus Box 526, Boulder CO 80309-0526, mjm@colorado.edu.

Math Subject Classification. Primary 65T20; Secondary 42C10, 33C55.

Keywords and Phrases. Spherical Harmonics, Fast Transforms, Associated Legendre Functions.

Acknowledgements and Notes. I would like to thank my thesis advisor, R.R. Coifman, for his help and guidance.

Abstract

Spherical Harmonics arise on the sphere S^2 in the same way that the (Fourier) exponential functions $\{e^{ik\theta}\}_{k\in\mathbf{Z}}$ arise on the circle. Spherical Harmonic series have many of the same wonderful properties as Fourier series, but have lacked one important thing: a numerically stable fast transform analogous to the Fast Fourier Transform. Without a fast transform, evaluating (or expanding in) Spherical Harmonic series on the computer is slow—for large computations prohibitively slow. This paper provides a fast transform.

For a grid of $\mathcal{O}(N^2)$ points on the sphere, a direct calculation has computational complexity $\mathcal{O}(N^4)$, but a simple separation of variables and Fast Fourier Transform reduce it to $\mathcal{O}(N^3)$ time. Here we present algorithms with times $\mathcal{O}(N^{5/2}\log N)$ and $\mathcal{O}(N^2(\log N)^2)$.

The problem quickly reduces to the fast application of matrices of Associated Legendre Functions of certain orders. The essential insight is that although these matrices are dense and oscillatory, locally they can be represented efficiently in trigonometric series.

1 Introduction

Spherical Harmonics arise on the sphere S^2 in the same way that the (Fourier) exponential functions $\{e^{ik\theta}\}_{k\in\mathbf{Z}}$ arise on the circle. Spherical Harmonic series have many of the same wonderful properties as Fourier series, but have lacked one important thing: a numerically stable fast transform analogous to the Fast Fourier Transform. Without a fast transform, evaluating (or expanding in) Spherical Harmonic series on the computer is slow—for large computations prohibitively slow. This paper provides a fast transform.

Using spherical coordinates ($\phi \in (0,\pi)$, $\theta \in [0,2\pi)$), the Spherical Harmonics for S^2 can be represented as $\{P_n^m(\cos\phi)e^{im\theta}\}$ where $n \geq |m|$, m and n are integers, and the P_n^m are Associated Legendre functions. When properly normalized, this forms a basis for the sphere with measure $\sin\phi d\phi d\theta$.

On a grid of $2N$ points equispaced in θ by N gaussian nodes in $\cos\phi$, N^2 basis functions are resolved. If we expand (or evaluate) directly using these basis functions, it will take $2N^4$ computations. In the θ variable, our basis is simply Fourier series and we can use the FFT (Fast Fourier Transform). Applying the FFT for N values of ϕ requires $N \times 2N\log(2N)$ computations, and leaves us with a set of $2N$ problems indexed by m. Each problem is: on an interval of N points, expand (evaluate) a given function using the basis $\{P_n^m(\cos\phi)\}$ with measure $\sin\phi d\phi$. If we do each of these directly, it will take $2N \times \mathcal{O}(N^2)$ computations. Our overall cost is then $\mathcal{O}(N^3)$.

Here we present a method for expanding (evaluating) using $\{P_n^m(\cos\phi)\}$ in $\mathcal{O}(N\log^2 N)$ time, with constant independent of m. This method allows us to reduce the overall cost to $\mathcal{O}(N^2\log^2 N)$. We also present an $\mathcal{O}(N^{5/2}\log N)$ algorithm that, although asymptotically slower, performed better in our tests. Both methods use precomputed, compressed representations of the Associated Legendre function bases.

This paper is based on the thesis [Moh97]. More details may be found there.

1.1 Statement of the Problem

We are given a set of Spherical Harmonics:

$$\left\{\tilde{P}_n^m(\cos\phi)e^{im\theta}/\sqrt{2\pi}\right\}_{(m,n)\in\mathcal{I}_N} \tag{1}$$

defined on the sphere ($\phi \in (0,\pi)$, $\theta \in [0,2\pi)$). $\tilde{P}_n^m$ is the Associated Legendre function of degree n and order m, normalized so that our set of functions is orthonormal on the sphere. The indices allowed are restricted to the set

$$\mathcal{I}_N = \{(m,n) \in \mathbf{Z}\times\mathbf{Z} : 0 \le n < N, -n \le m \le n\}. \tag{2}$$

For the evaluation problem, we are also given a set of coefficients $\{\alpha_n^m\}_{(m,n)\in\mathcal{I}_N}$ and we form the sum

$$f(\phi,\theta) = \sum_{\mathcal{I}_N} \alpha_n^m \tilde{P}_n^m(\cos\phi)e^{im\theta}/\sqrt{2\pi}. \tag{3}$$

This sum is evaluated on a grid in $\phi\times\theta$ with $2N$ equispaced points in θ and N gaussian nodes in $\cos\phi$:

$$\left\{(\phi_j,\theta_k) = \left(\cos^{-1}(g_j^N), 2\pi\frac{k}{2N}\right) : j,k \in \mathbf{Z}; 0 \le j < N, 0 \le k < 2N\right\}. \tag{4}$$

(See e.g. [FPTV92] for discussion of the gaussian nodes.) The evaluation problem for Spherical Harmonics is to compute the $\mathcal{O}(N^2)$ output values $f(\phi_j,\theta_k)$ from the $\mathcal{O}(N^2)$ input values α_n^m in a fast, numerically stable way.

The expansion problem for Spherical Harmonics is dual to the evaluation problem. We are given the $\mathcal{O}(N^2)$ sampled values $f(\phi_j,\theta_k)$ of a function of the form (3) and wish to compute the $\mathcal{O}(N^2)$ output values α_n^m in a fast, numerically stable way.

1.2 Summary of Results

The problem of the Fast Transform quickly reduces to a set of $2N$ problems indexed by m ($-N < m < N$). (See Section 2.3.) The problems are to find a fast application for the matrices

$$M_{n,j}^m = \left(\sqrt{\sin\phi_j}\,\tilde{P}_n^m(\cos\phi_j)\right)_{n,j}. \tag{5}$$

These matrices are dense and oscillatory. (See Figure 2 in Section 2.4.)

To understand these matrices, we model them using quasi-classical (WKB) frequency estimates. (Section 2.4.) The model is

$$M^m_{n,j} \approx \left(\exp\left(\int^{\phi_j} i\sqrt{(n+1/2)^2 - \frac{m^2-1/4}{\sin^2 t}}dt\right)\right)_{n,j} = \left(\exp(in\phi_j \Phi^m(n,\phi_j))\right)_{n,j}. \tag{6}$$

We now fix m and analyze $\Phi^m(n,\phi)$. (See [BCG93] for a similar model analysis.) If for each n, $\Phi^m(n,\phi)$ is constant as a function of ϕ, this matrix application becomes the evaluation of a Fourier series and so can be done fast using the FFT. $\Phi^m(n,\phi)$ is not constant, but we can partition ϕ-space into a small number of intervals, on each of which $\Phi^m(n,\phi)$ is almost constant. Using this partition, we can write $\exp(in\phi\Phi^m(n,\phi))$ as a sum of $\mathcal{O}(\sqrt{N}\log N)$ localized exponentials. The partition, however, *depends on* n (and m). Restricting ourselves to dyadic partitions, we can find a suitable partition for each $\Phi^m(n,\phi)$, and yet the total collection of intervals used is manageable. We can now represent the entire matrix (6) in terms of $\mathcal{O}(N^{3/2}\log N)$ exponentials, apply it in this same $\mathcal{O}(N^{3/2}\log N)$ time, and return to normal coordinates with the FFT in $\mathcal{O}(N\log^2 N)$ time. We call this the One Dimensional Algorithm.

If for each ϕ, $\Phi^m(n,\phi)$ is constant as a function of n, this matrix application becomes an expansion into a Fourier series, which can also be done fast using the FFT. Thus we can use the same technique in the n coordinate. In fact, we can partition in both coordinates at the same time, which corresponds to breaking the matrix into dyadic rectangles. In ordinary coordinates, each rectangle gives the interaction of some interval in n with some interval in ϕ. We instead represent each rectangle by the interactions of a small number of exponentials in n with a small number of exponentials in ϕ. We can then represent the matrix (6) in terms of $\mathcal{O}(N\log N)$ interactions, apply it in $\mathcal{O}(N\log N)$ time, and return to normal coordinates with the FFT in $\mathcal{O}(N\log^2 N)$ time. The partition into rectangles will depend on m, and will not generally be a tensor product partition. We call this the Two Dimensional Algorithm.

This model gives the proper intuition, but is not suitable for proofs. To prove our compression estimates, we construct a quantitative type of Quasi-Classical (WKB) theory for solutions to Schrödinger equations in one dimension. Given a space localization, this theory gives us rigorous bounds on the number of local Fourier coefficients needed for a given precision. This theory also provides an intuition using 'instantaneous frequency' and the ever-elusive justification for this intuition.

We have implemented these algorithms, mainly for diagnostic purposes. Experiments have shown the algorithms to be stable and to work well at high precision. The one dimensional algorithm is outperforming the compression predictions, while the two dimensional is not performing as well. In our implementation the one dimensional algorithm becomes faster than the direct method at $N = 128$. By $N = 512$ it is faster by a factor of three.

2 Background

2.1 History

Spherical Harmonics arise naturally when one tries to generalize Fourier Series to the next dimension. Many of their properties, as well as references to their earlier history, may be found in Hobson [Hob31]. A more modern perspective can be found in [SW71]. The Spherical Harmonics can be built from Associated Legendre functions, which are considered 'special functions', and therefore appear in many handbooks of functions, e.g. [AS64].

Modern attempts at a fast transform were underway by the time of Dilts [Dil85], who reduced the operations count but not the order of the algorithm. Duvall, Elowitz, and Hill [DEH89] evaluate this method, and conclude the best current (1988) method is due to Brown [Bro85]. He separates variables and does an FFT in the ϕ variable to reduce the operations count to $\mathcal{O}(N^3)$ (See Section 2.3). This result is very likely much older, and indeed Brown does not claim originality.

Driscoll and Healy [DH94] have an $\mathcal{O}(N\log^2 N)$ algorithm for expansions (not evaluations) in $\{P_n^m(\cos\phi)\}$ with constant independent of m. Their algorithm becomes unstable as m increases, however.

Orszag [Ors86] proved a general result that one can expand or evaluate any set of Sturm-Liouville eigenfunctions (e.g. $\{P_n^m(\cos\phi)\}$ for fixed m) in $\mathcal{O}(N\log^2 N/\log\log N)$ time. The dependency on m is suppressed, however, so it does not yield a result for the full transform. Our approach is somewhat similar to Orszag's, but we are able to track down the m dependence.

Alpert and Rokhlin [AR91] have an $\mathcal{O}(N\log N)$ algorithm for Legendre Polynomials (P_n^0). As m increases, however, their technique would seem to break down. A frequency space version of our Section 4.4 would show that in fact one could have uniform bounds in m. We draw much from their ideas and approach.

2.2 What are Spherical Harmonics?

Spherical Harmonics can be generated in the same way as Fourier Series, simply in one dimension higher. A development which emphasizes these parallels may be found in Stein and Weiss [SW71, Chapter IV]. Our motivation for the consideration of Spherical Harmonics is the same as for Fourier Series, e.g. both diagonalize all linear operators which commute with rotations.

The Spherical Harmonics are eigenfunctions of the *Spherical Laplacian*

$$\Delta_{3S} = \csc^2\phi\frac{\partial^2}{\partial\theta^2} + \frac{\partial^2}{\partial\phi^2} + \cot\phi\frac{\partial}{\partial\phi} \tag{7}$$

which is the Laplacian in $\mathbf{R}^3$ restricted to the sphere. Δ_{3S} is self-adjoint and rotation-invariant, which implies that the eigenspaces are preserved by rotations and are orthogonal. The n^{th} eigenspace Λ_n has eigenvalue $-n(n-1)$, dimension $2n+1$, and consists of the homogeneous harmonic polynomials of degree n restricted to the sphere. All eigenfunctions are smooth, and together they span $L^2(S^2)$.

For fixed n, we can organize the Λ_n as $\{\tilde{P}_n^m(\cos\phi)e^{im\theta}/\sqrt{2\pi}\}_{-n\le m\le n}$. The condition on $\tilde{P}_n^m$ to make this a set of (smooth) spherical harmonics is that $\tilde{P}_n^m(1)\ne\pm\infty$ and

$$\left[\frac{\partial^2}{\partial\phi^2} + \cot\phi\frac{\partial}{\partial\phi} - \frac{m^2}{\sin^2\phi}\right]\tilde{P}_n^m(\cos\phi) = -n(n-1)\tilde{P}_n^m(\cos\phi). \tag{8}$$

This equation identifies the $\tilde{P}_n^m$'s up to a constant as the *Associated Legendre Functions (of the first kind) of order m and degree n.* We use the tilde ($\tilde{\cdot}$) to indicate the L^2 normalized version of the classically defined Associated Legendre functions, denoted P_n^m. We construct $\tilde{P}_n^m$ explicitly in Section 5.1.1.

2.3 First Reductions of the Problem

First note that we can use the FFT in θ (for each ϕ). Thus, for total computational cost of $\mathcal{O}(N^2\log N)$, we have reduced the evaluation problem to a set of N problems indexed by m ($-N < m < N$). Each problem is: Given the $\mathcal{O}(N)$ inputs α_n^m, compute the N outputs

$$f(\hat{m},\phi_j) = \sum_{n=|m|}^{N-1}\alpha_n^m\tilde{P}_n^m(\cos\phi_j). \tag{9}$$

Evaluated directly, each problem takes $\mathcal{O}(N^2)$ time, and so together they take $\mathcal{O}(N^3)$.

For the expansion problem we have learned first that these Spherical Harmonics are an orthonormal set, and so we should compute α_n^m by

$$\alpha_n^m = \int_0^\pi\int_0^{2\pi} f(\phi,\theta)\tilde{P}_n^m(\cos\phi)\frac{e^{-im\theta}}{\sqrt{2\pi}}\sin\phi\, d\phi\, d\theta. \tag{10}$$

Again we can use the FFT in θ and reduce the expansion problem to a set of N problem indexed by m, each of which is: Given the N sample values $f(\phi_j, \hat{m})$ compute the $\mathcal{O}(N)$ coefficients α_n^m. Our next task is to discretize the remaining integral. With our assumption (3) on the form of f, we only need be able to compute inner products of P_n^m's (m fixed). For $n, n' < N$, $\tilde{P}_n^m(\cos\phi)\tilde{P}_{n'}^m(\cos\phi) = (\sin\phi)^{2m}\tilde{P}_{n-m}^{(m,m)}(\cos\phi)\tilde{P}_{n'-m}^{(m,m)}(\cos\phi) = P(\cos\phi)$ is a polynomial in $\cos\phi$ of degree at most $2N-2$. Thus using gaussian nodes and weights in $\cos\phi$, we can capture this exactly using N points. (See e.g [FPTV92].) Each reduced problem can be solved by evaluating this sum, which takes $\mathcal{O}(N^2)$ time.

We will also find it convenient to modify our problems slightly. We define

$$\psi_n^m(\phi) = \sqrt{\sin\phi}\,\tilde{P}_n^m(\cos\phi). \tag{11}$$

The reduced evaluation and expansion problems then become matrix applications:

$$\left(\psi_n^m(\phi_j)\right)_{n,j}\alpha_n^m = \sqrt{\sin\phi_j}\,f(\phi_j,\hat{m}) \tag{12}$$

$$\left(\psi_n^m(\phi_j)\right)_{j,n}\left(\sqrt{\sin\phi_j}\,f(\phi_j,\hat{m})w_N(j)\right) = \alpha_n^m. \tag{13}$$

In our algorithm, we find a way to apply these matrices quickly. The method works equally well for the matrix transpose and so the evaluation and expansion problems become one problem. We will usually ignore the expansion problem and only consider the evaluation problem.

One property of $\tilde{P}_n^m(\cos\phi)$ that is only apparent from its explicit construction in Section 5.1.1 is that it is either even or odd across $\phi = \pi/2$ as $n-m$ is even or odd. We can solve the reduced problem separately for the even and odd components on $\phi \in (0, \pi/2)$ and then use simple reflections to produce the final solution. We use this explicitly in Section 4.3 when we prove compression results on $(0, \pi/2)$, in Section 5.1.3 where we use it to justify a specialized search, and to deal with edge effects. The use of parities also reduces computation time by a factor of two.

2.4 Quasi-Classical (WKB) Frequency Estimates

There is a method for estimating the solutions to some types of differential equations, in particular Schrödinger equations. See e.g. Landau and Lifschitz [LL77, Chapter VII] or Bender and Orszag [BO78]. We will use this to get a preliminary description of the Associated Legendre functions and to motivate the algorithms in Section 3.

With the reductions made in Section 2.3, we are considering $\{\psi_n^m(\phi) = \sqrt{\sin\phi}\,\tilde{P}_n^m(\cos\phi)\}$ as a basis with measure $d\phi$. From (8) we can deduce the differential equation:

$$\left[\frac{d^2}{d\phi^2} - \frac{m^2 - 1/4}{\sin^2\phi}\right]\psi_n^m(\phi) = -(n+1/2)^2\psi_n^m(\phi) \tag{14}$$

which is a nice Schrödinger equation, to which we can apply the Quasi-Classical approximation. This approximation yields 'instantaneous frequency'

$$\nu_n^m(\phi) = \sqrt{(n+1/2)^2 - \frac{m^2 - 1/4}{\sin^2\phi}} \tag{15}$$

valid when the argument of the root is positive, and the approximation $\psi_n^m(\phi) \approx \exp\left(i\int^\phi \nu_n^m(t)dt\right)$.

These estimates tell us that at the edges of the interval our functions decay rapidly and smoothly. As we move toward the center of the interval, they have instantaneous frequency increasing and concave down. See Figure 1.

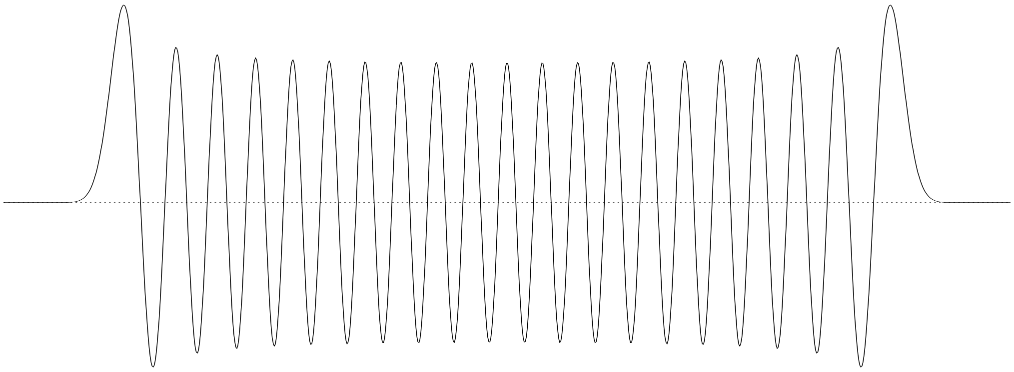

Figure 1: $\psi_{60}^{20}(\phi) = \sqrt{\sin\phi}\tilde{P}_{60}^{20}(\cos\phi)$ on $[0, \pi]$.

Globally this function is complicated, but locally it looks very much like a trigonometric function. This is the motivation behind our one dimesional algorithm (Section 3.2). We partition our function and represent each piece in localized trigonometric functions. These representations will be very efficient, and convert our matrix to a sparse or 'compressed' form. The partition we choose will depend on m and n. For fixed m, we wish to express the matrix $(\psi_n^m(\phi_j))_{n,j}$ in sparse (compressed) form. Since the partition changes with n, this is not simply a change of basis, but is an adaptive, non-standard representation of the matrix.

These estimates can also be used to model the entire matrix $(\psi_n^m(\phi_j))_{n,j}$ (m fixed). See Figure 2. Note that the matrix is also almost trigonometric as a function of n. With a proper two dimensional partition, we should be able to compress this matrix even further. This is the motivation for the two dimensional algorithm in Section 3.3. The partition chosen will depend on m.

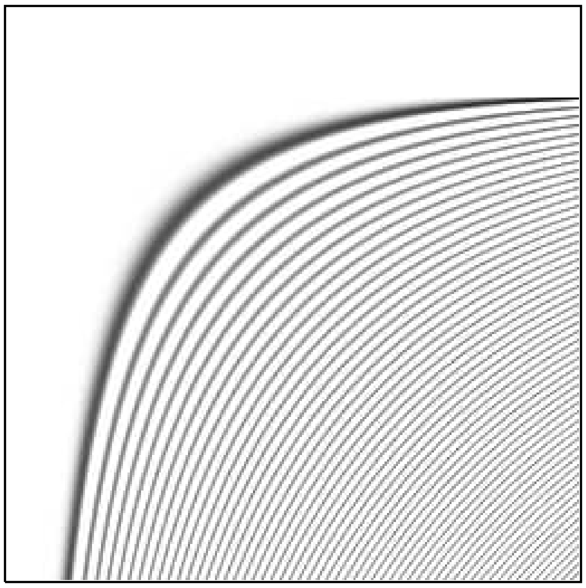

Figure 2: The positive part of $\psi_n^{40}(\phi) = \sqrt{\sin\phi}\tilde{P}_n^{40}(\cos\phi)$ for n in $[0, 256]$ and ϕ in $[0, \pi/2]$. The n axis points down, and the ϕ axis to the right.

The Quasi-Classical method is important in that it gives an understanding of the behavior of the Associated Legendre functions and motivation for our algorithms. It lacks sufficient rigor for use in proofs, however. In Section 4.1 we develop an alternative method which gives rigorous results of the type we need.

3 The Structure of the Algorithms

This section gives the structure of the algorithms used for the Spherical Harmonic transform. The fact that they do in fact give the transform follows from simple facts such as orthogonality and is explained here. The speed of these algorithms depends mostly on the compression which is achieved in the precomputation stage, and this is dealt with in Section 4. We deal specifically with the reduced form of the evaluation problem, from Section 2.3.

We present two algorithms. The first, the one dimensional algorithm, is asymptotically slower, but has performed better in numerical testing. The core of both algorithms is a non-standard matrix representation and multiplication. We note the similarities between our approach and the non-standard matrix multiplication in [BCR91].

3.1 Local Trigonometric Expansions

To compress the Associated Legendre functions we use variations of the standard Local Cosine basis. See [CM91] for a proof of its properties and [Wic94] (e.g.) for an exposition of its uses. A Local Cosine basis is constructed as follows: We begin with a sequence of points on the line (interval, circle) $\cdots a_i < a_{i+1} \cdots$. Let $I_i = [a_i, a_{i+1}]$. We have a set of compatible bells, indexed by their interval, $\{b_i(x)\}$. These bells have the properties that $b_i(x)b_{i-1}(x)$ is even about a_i, $b_i(x)b_{i'}(x) = 0$ if $i' \neq i \pm 1$, and $\sum_i b_i^2(x) = 1$. On each interval we have a set of cosines of the proper scaling and shift, denoted

$$\left\{ c_i^p(x) = \sqrt{\frac{2}{a_{i+1} - a_i}} \cos\left(\frac{(p+1/2)\pi(x - a_i)}{a_{i+1} - a_i} \right) \right\}_{p=0}^{\infty}. \tag{16}$$

The set $\{b_i(x)c_i^p(x)\}$ forms an orthonormal basis for the line. There is a fast LCT (Local Cosine transform) based on the FFT, using techniques similar to those in [FPTV92, Chapter 12].

3.2 The One Dimensional Algorithm

In the precomputation stage, we express each ψ_n^m in the Local Cosine basis which requires the least number of coefficients above some given ϵ. We require the intervals to have dyadic size and dyadic position (intervals of length 2^{-j} are $k2^{-j}$ from the edge, $k \in \mathbf{Z}$). We denote the set of dyadic intervals by $\mathcal{D}$ and the Local Cosine basis element for interval I with frequency p by $b_I(\phi)c_I^p(\phi)$. We can then write

$$\psi_n^m(\phi_j) = \sum_{I \in \mathcal{D}} \sum_{p=0}^{N|I|-1} \lambda_{Ip}^{mn} b_I(\phi_j) c_I^p(\phi_j). \tag{17}$$

Each ψ_n^m has λ_{Ip}^{mn}'s only for I's in some particular partition. We approximate the full sum with the sum where we discard all λ_{Ip}^{mn} that are smaller than ϵ.

For the reduced evaluation problem from Section 2.3, m is fixed, we are given $\{\alpha_n^m\}$, and we wish to form the sum

$$\sum_{n=|m|}^{N/2-1} \alpha_n^m \psi_n^m(\phi_j). \tag{18}$$

We can reorganize this as

$$\sum_n \alpha_n^m \sum_{I\in\mathcal{D}} \sum_p \lambda_{Ip}^{mn} b_I(\phi_j) c_I^p(\phi_j) = \sum_{I\in\mathcal{D}} b_I(\phi_j) \sum_p c_I^p(\phi_j) \sum_n \alpha_n^m \lambda_{Ip}^{mn} \tag{19}$$

and sum over n to obtain

$$= \sum_{I\in\mathcal{D}} b_I(\phi_j) \sum_p \gamma_{Ip}^m c_I^p(\phi_j). \tag{20}$$

To compute all the γ^m's costs as many computations as we have λ's. (Remember we have discarded $\lambda_{Ip}^{mn} < \epsilon$.) To evaluate the sums over I and p at all ϕ_j's, we do Local Cosine transforms at $\log N$ scales, for total cost $N \log^2 N$.

In Section 4.3 we will show that for each ψ_n^m we need at most $\mathcal{O}(\sqrt{m}\log N(\log\epsilon)^{7/2})$ coefficients. Adding this over n yields a total count of λ's of $\mathcal{O}(\sqrt{m}N\log N(\log\epsilon)^{7/2})$. The application of the compressed matrix (computation of γ) thus dominates the Local Cosine transforms. Doing this evaluation for N values of m gives us a total operations count of $\mathcal{O}(N^{5/2}\log N(\log\epsilon)^{7/2})$.

3.3 The Two Dimensional Algorithm

We again deal with the reduced evaluation problem (Section 2.3) for some fixed m. We consider it in its matrix form $(\psi_n^m(\phi_j))_{n,j}\, \alpha_n^m = \sqrt{\sin\phi_j} f(\phi_j, \hat{m})$. Define the matrix $M_{n,j} = \psi_n^m(\phi_j)$ with $\psi_n^m \equiv 0$ for $n < |m|$. Let α be the vector of values of α_n^m, and f be the vector of values of $\sqrt{\sin\phi_j} f(\phi_j)$. We can apply M to α directly with computation time equal to the number of nonzero entries of M, which is $N(N-m) = \mathcal{O}(N^2)$.

Instead we break M into disjoint rectangular submatrices. We can apply M to α by applying each submatrix to some part of α and putting the result in the appropriate place in f. Conceptually, it is best to think of this as applying the transpose of the vector α (α^t) to M from above, and getting out f^t, as in Figure 3. In the end, several rectangles contribute to each f_j, but only additively. We can treat each rectangle as a separate problem. In particular, we can do a distinct change of basis *on each rectangle.* As input for these rectangles, we must expand α locally into many different bases. Similarly, we have to recombine the outputs to get f.

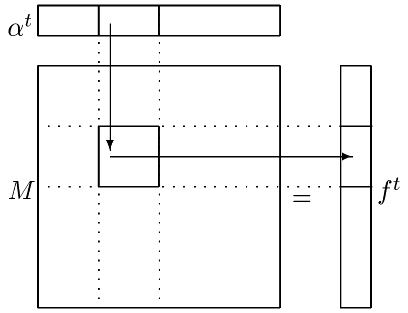

Figure 3: Applying a vector to a matrix from above.

We assure the fast computation of inputs from α by first requiring the rectangles be dyadic (in size and position). Second, we only allow our change of basis be to Local Cosine series. With these restrictions, preparing α and assembling f take at most $N\log^2 N$ computations.

The objective of this approach is to find a partition so that each rectangle can be represented by a small number of (Fourier) coefficients. The cost to apply this matrix is the total number of these which are greater than ϵ. Simply removing the rectangles creates a discontinuity at the edges, however, which makes the Fourier coefficients decay very slowly. We would instead like to separate the rectangles smoothly.

We consider the partitioning in n, which has to do with expansions of α. Fix j and a dyadic interval I. Let α_I be α restricted to I, I^* be the symmetric double of I, and b_I a smooth function supported on I^* with $1/2 \leq b(n) \leq 1$ when $n \in I$. As part of the matrix application, we wish to compute the inner product $\langle M_{n,j}|_I, \alpha\rangle = \langle M_{n,j}, \alpha_I\rangle = \langle M_{n,j} b_I, \alpha_I / b_I\rangle$. Let $\widehat{(\cdot)}$ represent a Fourier expansion on I^*. Then $\langle M_{n,j} b_I, \alpha_I / b_I\rangle = \langle \widehat{M_{n,j} b_I}, \widehat{\alpha_I / b_I}\rangle$. Since $M_{n,j} b_I$ is as smooth as M, this eliminates the discontinuity. We will store the coefficients $\widehat{M_{n,j} b_I}$ instead of $\widehat{M_{n,j}|_I}$. We refer to this trick as using 'double bells' or 'double sized intervals'.

Note that with this approach we lose orthogonality. All we really need is the ability to apply the matrix, which we retain. An advantage of this method is that adjacent rectangles do not interact. This removes issues of bell compatibility and greatly simplifies searches (Section 5.1.3). A disadvantage is a factor of two inefficiency in the bell cost.

In Section 4.4 we will show that for each m we need at most $\mathcal{O}(N \log N (\log \epsilon)^7)$ coefficients for M, with constants independent of m. The operations count of $\mathcal{O}(N \log^2 N)$ for the LCTs thus dominates. Doing this evaluation for N values of m gives us a total operations count of $\mathcal{O}(N^2 \log^2 N)$.

4 Compression Rates

This section gives rigorous proofs of bounds on the number of coefficients retained after compressing using the one or two dimensional partitioning schemes described in Section 3. We first construct a general theory of compression for solutions to Schrödinger equations, and then apply this theory to our specific case.

By 'compression' we mean the representation (to some prescribed accuracy) of a given function or matrix in a small number of parameters. We also require that we be able to effectively use these parameters in place of the function. This requirement means that a sparse representation of a function in terms of a basis with fast transform is compression, while the representation of the matrix $(\psi_n^m(\phi_j))_{n,j}$ by the two parameters m and N is not.

4.1 A Quantitative Quasi-Classical Theory

4.1.1 First Bounds

Theorem 4.1 *Let $\psi(x)$ be a solution to $\psi''(x) = -V(x)\psi(x)$ and $b(x)$ a smooth function supported on $[0,1]$. Then:*

$$|\widehat{b\psi}(\xi)| \leq \frac{||(V(x) - V^*)b||_p + ||b''||_p + 2|\xi| \cdot ||b'||_p}{|V^* - \xi^2|} ||\psi||_q \tag{21}$$

for any $V^* \in \mathbf{R}$ and $1 = (1/p) + (1/q)$; $1 \leq p, q \leq \infty$.

Proof: Integrating twice by parts, we obtain

$$\widehat{b\psi}(\xi) = \int_0^1 [b''(x)\psi(x) + 2b'(x)\psi'(x) + b(x)\psi''(x)] \frac{e^{-ix\xi}}{(-i\xi)^2} dx. \tag{22}$$

Considering the $b'\psi'$ term separately and integrating by parts again, we have:

$$\widehat{b\psi}(\xi) = \frac{1}{-\xi^2} \int_0^1 [-b''(x) - V(x)b(x) + 2i\xi b'(x)] \psi(x) e^{-ix\xi} dx. \tag{23}$$

Choosing any V^* we have

$$\widehat{b\psi}(\xi)\left(1+\frac{V^*}{-\xi^2}\right)=\frac{1}{-\xi^2}\int_0^1\left[-b''(x)-(V(x)-V^*)b(x)+2i\xi b'(x)\right]\psi(x)e^{-ix\xi}dx \tag{24}$$

$$\widehat{b\psi}(\xi)=\frac{1}{V^*-\xi^2}\int_0^1\left[-b''(x)-(V(x)-V^*)b(x)+2i\xi b'(x)\right]\psi(x)e^{-ix\xi}dx \tag{25}$$

and the theorem follows by Hölder's inequality. □

4.1.2 Measuring Localization

We have 'localized' $\widehat{b\psi}$ near $\xi=\pm\sqrt{V^*}$. The localization is better when we choose $V^*\approx V(x)$. In a rough sense then, we can claim ψ has local frequency $\pm\sqrt{V^*}$ and instantaneous frequency $\pm\sqrt{V(x)}$. To quantify these statements, we would like to know how many frequencies (near $\pm\sqrt{V^*}$) are significant. $\widehat{b\psi}$ cannot be compactly supported, so we must choose some $\epsilon>0$ and consider $\{\xi:|\widehat{b\psi}(\xi)|>\epsilon\}$ as being significant. We will call this set the *ϵ-support* of $\widehat{b\psi}(\xi)$.

Theorem 4.2 *With ψ and b as in Theorem 4.1, and with the assumptions $||\psi||_q=1$ and $0\le b(x)\le 1$, the length of the overall ϵ-support is bounded by*

$$\frac{12||b'||_p}{\epsilon}+\min\left\{2\sqrt{\frac{6||b''||_p}{\epsilon}},\frac{6\sqrt{2}||b''||_p}{\epsilon\sqrt{V^*}}\right\}+\min\left\{2\sqrt{\frac{6||V(x)-V^*||_p}{\epsilon}},\frac{6\sqrt{2}||V(x)-V^*||_p}{\epsilon\sqrt{V^*}}\right\}. \tag{26}$$

Proof: To compute the ϵ-support of our bounds, we compute the $(\epsilon/3)$-support of the three terms in Theorem 4.1 and take the union. For the $(V-V^*)b$ term we have

$$\frac{||V(x)-V^*||_p}{|V^*-\xi^2|}\ge\frac{\epsilon}{3} \tag{27}$$

when

$$\sqrt{V^*-\frac{3||V(x)-V^*||_p}{\epsilon}}\le|\xi|\le\sqrt{V^*+\frac{3||V(x)-V^*||_p}{\epsilon}} \tag{28}$$

which is a set of length

$$2\left(\sqrt{V^*+\frac{3||V(x)-V^*||_p}{\epsilon}}-\sqrt{V^*-\frac{3||V(x)-V^*||_p}{\epsilon}}\right). \tag{29}$$

When $V^*\le 3||V(x)-V^*||_p/\epsilon$ we drop the second term. This length is bounded by

$$\min\left\{2\sqrt{\frac{6||V(x)-V^*||_p}{\epsilon}},6\sqrt{2}\frac{||V(x)-V^*||_p}{\epsilon\sqrt{V^*}}\right\}. \tag{30}$$

This second bound is only useful when $V^*\ge 3||V(x)-V^*||_p/\epsilon$, so we call it a 'high frequency' bound.

Similarly, for the b'' term, we have a set of length

$$2\left(\sqrt{V^*+\frac{3||b''||_p}{\epsilon}}-\sqrt{V^*-\frac{3||b''||_p}{\epsilon}}\right)\le\min\left\{2\sqrt{\frac{6||b''||_p}{\epsilon}},\frac{6\sqrt{2}||b''||_p}{\epsilon\sqrt{V^*}}\right\}. \tag{31}$$

The b' term behaves differently, however.

$$\frac{2|\xi|||b'||_p}{|V^* - \xi^2|} \geq \frac{\epsilon}{3} \tag{32}$$

on a set of length

$$\frac{12||b'||_p}{\epsilon}.\square \tag{33}$$

Remark: The b' term yielded an ϵ-support of length $12||b'||_p/\epsilon$. We may be able to improve this by applying Theorem 4.1 again, but we will always be left with a term which, like this one, is independent of V and V^*. This term gives a minimum on the length of the ϵ-support, and can be viewed as a manifestation of the uncertainty principle.

4.1.3 Number of Terms

Up to now we have been considering the Fourier transform $\widehat{b\psi}$. Since we are actually on an interval, it makes sense to sample $\widehat{b\psi}$ and consider the Fourier series. Instead of considering the length of the ϵ-support as our measure of localization, we consider the number of Fourier coefficients above ϵ.

Definition 4.3 *The* **cost** *(or ϵ-cost) of an expansion is the number of Fourier coefficients above ϵ.*

For our application we choose $p = \infty$ and $q = 1$. We assumed earlier that $||\psi||_q = 1$. Actually, ψ will be a basis element, and so have $||\psi||_{L^2[0,1]} = 1$. We formulate our theorem on cost for our particular application because the scaling simplifies.

Theorem 4.4 *With ψ and b as in Theorem 4.1, and with the assumptions $||\psi||_2 = 1$ and $0 \leq b(x) \leq 1$, the cost of expansion on an interval of length l is at most*

$$2\max\left\{\frac{12||b'||_\infty}{\epsilon}, \min\left\{2\sqrt{\frac{6||b''||_\infty}{\epsilon}}, \frac{6\sqrt{2}||b''||_\infty}{\epsilon\sqrt{l^2V^*}}\right\},\right.$$
$$\left.\min\left\{2l\sqrt{\frac{6||V(x)-V^*||_\infty}{\epsilon}}, l\frac{6\sqrt{2}||V(x)-V^*||_\infty}{\epsilon\sqrt{V^*}}\right\}\right\}. \tag{34}$$

To deal with intervals of length l, we must adjust our previous bounds. We first note that we can bound $||\psi||_{L^1[x,x+l]} \leq ||\psi||_2||1||_{L^2[x,x+l]} = \sqrt{l}$. When we convert from a Fourier transform to Fourier series, we must also normalize the exponentials to be a basis on our interval, by dividing by $\sqrt{l}$. These two factors of $\sqrt{l}$ will cancel.

Converting the Fourier transform to Fourier series by sampling also means changing the ξ to k/l where k is an integer. (We ignore a factor of 2π.) The cost on an interval of length l can thus be computed from the ϵ-support as $l \cdot |\epsilon\text{-support}|$.

We must also shrink the bell to fit on this interval. We must consider $b(x/l)$ and thus $||(b(\cdot/l))'|| = ||b'||/l$ and $||(b(\cdot/l))''|| = ||b''||/l^2$. Our new b' cost bound is then

$$\frac{12||b'||_\infty}{\epsilon l} l = \frac{12||b'||_\infty}{\epsilon} \tag{35}$$

which is unchanged. Thus any expansion on any interval has at least this cost (using this method).

From b'' we have a new cost of

$$\min\left\{2\sqrt{\frac{6||b''||_\infty}{\epsilon}}, \frac{6\sqrt{2}||b''||_\infty}{\epsilon\sqrt{l^2V^*}}\right\}. \tag{36}$$

From $V - V^*$ we now have

$$\min\left\{2l\sqrt{\frac{6||V(x)-V^*||_\infty}{\epsilon}}, l\frac{6\sqrt{2}||V(x)-V^*||_\infty}{\epsilon\sqrt{V^*}}\right\}. \quad (37)$$

In this case both bounds have been multiplied by l and so improve linearly if we shrink l. Also note that the norm is taken over a different interval. Thus if we shrink l, for some (different) choice of V^*, we can likely make $||V(x) - V^*||_\infty$ somewhat smaller. How much smaller we can make it depends on our particular $V(x)$. We will later use our knowledge of $V(x)$ to choose appropriate l and V^*.

Remark: The b'' term is unimportant, since its cost in general can be dominated by the b' term. We will ignore its presence.

4.2 The Intuition

We would like to have a simple method for computing the cost of expanding a 'chirp', i.e. a function which has 'instantaneous frequency' some function $\nu(x)$. We graph $\nu(x)$ in the $x \times \xi$ phase plane. On the phase plane a Local Cosine basis element is viewed as a rectangle of area 1 with x-support on its base interval I shifted in ξ by its frequency. Intuitively, those boxes that intersect $\nu(x)$ should correspond to Local Cosine elements that yield significant coefficients. If we set $\Delta = \max_I \nu(x) - \min_I \nu(x)$ we can estimate the number of boxes by $\Delta|I| = \Delta l$. We must always intersect at least 1 box, however, so we need to consider $\max\{\Delta l, 1\}$. See Figure 4.

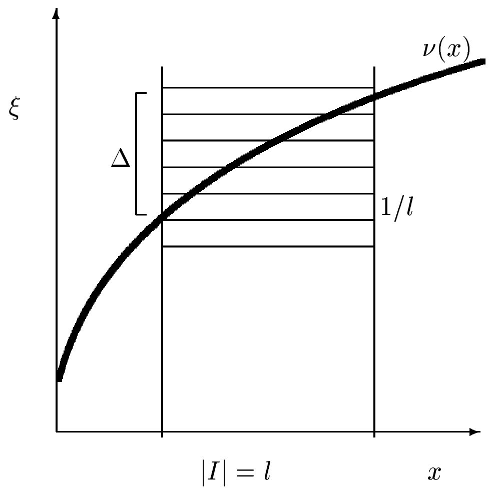

Figure 4: Instantaneous frequency cost intuition.

In Section 4.1.2 we bounded the cost of expansion by

$$\frac{C}{\epsilon}\frac{||V(x)-V^*||}{\sqrt{V^*}}. \quad (38)$$

Assuming $V(x) \geq 0$ let $\nu(x) = \sqrt{V(x)}$ and $\sqrt{V^*}$ be the median value of $\sqrt{V(x)}$. Then our cost is

$$\frac{C}{\epsilon}\left|\sqrt{V(x)} - \sqrt{V^*}\right|\frac{\left|\sqrt{V(x)} + \sqrt{V^*}\right|}{\sqrt{V^*}} \leq 2\frac{C}{\epsilon}\left|\sqrt{V(x)} - \sqrt{V^*}\right|. \quad (39)$$

In Section 4.1.3 we determined that if the interval is of length l, our cost is multiplied by l, to yield $(2C/\epsilon)l|\sqrt{V(x)} - \sqrt{V^*}|$. Again we let $\Delta = \max_I \nu(x) - \min_I \nu(x)$. The cost to expand on that interval is bounded by $(2C/\epsilon)\max\{l\Delta, B\}$, where B is the minimal bell cost ($||b'||$). Our intuition is therefore justified.

4.3 Associated Legendre Functions

If we consider the potential

$$V(\phi) = (n+1/2)^2 - \frac{m^2 - 1/4}{\sin^2 \phi} \tag{40}$$

on $[0, \pi/2]$, ψ will be an Associated Legendre function of order m, $\sqrt{\sin\phi}\tilde{P}_n^m(\cos\phi)$. We will eventually wish to take many derivatives of $V(x)$ but the $\sin\phi$ will introduce factors of $\cos\phi$ and make the calculations unpleasant. Instead we can consider the potential

$$V(x) = (n+1/2)^2 - \frac{m^2 - 1/4}{x^2} \tag{41}$$

which will yield Bessel functions. Qualitatively, these behave the same as the Associated Legendre functions, and so we will have the same overall cost estimate. Making this rigorous would simply introduce some factors of π. See Figure 5 for graphs comparing the behavior of an Associated Legendre function and the corresponding Bessel function.

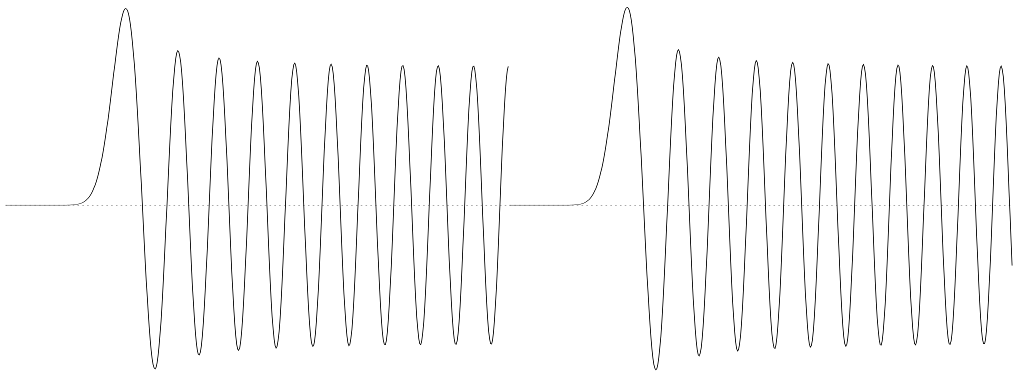

Figure 5: $\sqrt{\sin\phi}\tilde{P}_{60}^{20}(\cos\phi)$ and $\sqrt{r}J_{20}(60r)$ on $[0, \pi/2]$

Remark: In the next few sections we will prove compression results for Associated Legendre functions by proving them for Bessel functions. As a consequence, we also obtain a fast version of the Hankel transform for f with compact support. (See [EMOT54, p.3].) The Hankel tranform is defined as

$$\mathcal{H}_\nu f(y) = \int_0^\infty f(r) J_\nu(ry)\sqrt{ry}\,dr. \tag{42}$$

4.3.1 Bessel Functions

We have made a simplification to consider the potential (41) on $[0,1]$. $\psi(x)$ is then the Bessel function of order m, $\sqrt{x}J_m(nx)$. To simplify the following calculations further, we make a further reduction to $V(x) = n^2 - (m/x)^2$. We will not use any algebraic properties of the values of m and n, so this reduction is unimportant.

The tools we developed above give bounds on the cost (number of coefficients) for a given interval. Our only real freedom is how we partition $[0,1]$. Our total cost is the sum of the costs from each subinterval. We would like to find the partition which minimizes our total cost.

Theorem 4.5 *Let ψ be a solution to $\psi'' = (n^2 - (m/x)^2)\psi$ on N points in $[0,1]$, normalized so $||\psi||_2 \leq 1$. Then there exists a partition of $[0,1]$ which yields total expansion cost less than*

$$2\sqrt{m}\frac{\sqrt{||b'||}}{\epsilon}\log N. \tag{43}$$

The first part of our scheme is a stopping time argument. We fix a cost C to be the maximum allowed cost. Given an interval, if its cost is less than C we keep it; otherwise split it in two and pass them to the next stage. Start with $[0,1]$ and continue until stopped. In deciding whether we stop there are two considerations. First is the bell cost. If we split an interval in two, our non-bell cost (37) shrinks by a factor of at least two, but the bell cost (35) is unchanged. We will stop if and only if C is greater than the bell cost $||b'||/\epsilon$. The second consideration is that we will be working on a finite number of points N. Once the number of points in an interval is less than C, it cannot cost more than C in any case. We will also use this fact when we count intervals.

Before trying to count the intervals, we consider the structure of the partition we expect to be chosen. When $x < m/n$, ψ decays rapidly and smoothly and can be expanded into Local Cosine with negligible cost. We thus consider this a 'nothing' interval and only worry about $[m/n, 1]$. For a fixed interval length $l = 2^{-j}$ there will be some set in $[m/n, 1]$ such that the interval $[x, x+l]$ has cost less than C. Using the high frequency bound in (30), we have cost

$$\leq \frac{2^{-j}||V - V^*||}{\epsilon\sqrt{V^*}}. \tag{44}$$

Since V is concave down, we can bound $||V - V^*||$ on $[x, x+2^{-j}]$ by $2^{-j}V'(x)$, choosing $V^* = V(x)$. We then have cost

$$\leq \frac{2^{-2j}m^2}{\epsilon x^2\sqrt{(nx)^2 - m^2}}. \tag{45}$$

Note that this cost shrinks as x increases. The 'good' set (G_j) for size 2^{-j} is thus some interval $[x_j, 1]$. The remaining interval $[m/n, x_j]$ is the 'bad' set (B_j). See Figure 6.

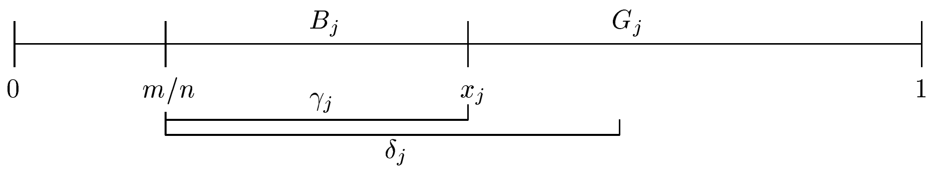

Figure 6: Good and bad intervals for size 2^{-j}

It will be most efficient if we break the interval $B_{j-1} \cap G_j = [x_j, x_{j-1}]$ into intervals of size 2^{-j}. It will take $2^j|B_{j-1} \cap G_j|$ such intervals, yielding cost $C2^j|B_{j-1} \cap G_j|$. To estimate $|B_{j-1} \cap G_j|$, we first dominate it by $|B_{j-1}| = x_{j-1} - m/n = \gamma_{j-1}$. Setting our cost bound equal to C and manipulating, we find

$$\frac{2^{-j}m}{\sqrt{C\epsilon}} = x_j((x_jn)^2 - m^2)^{1/4} = (\gamma_j + m/n)(\gamma_j n)^{1/4}(\gamma_j n + 2m)^{1/4}. \tag{46}$$

It is difficult to solve for γ_j, but it will be dominated by δ_j, which satisfies

$$\frac{2^{-j}m}{\sqrt{C\epsilon}} = \delta_j^{3/4}(m/n)^{1/4}(\delta_j n)^{1/4}m^{1/4} = \delta_j m^{1/2}, \tag{47}$$

which means $\delta_j = 2^{-j}\sqrt{m/C\epsilon}$.

We can now bound $[x_j, x_{j-1}]$ by δ_{j-1}. Our cost for this interval using subintervals of size 2^{-j} is thus at most

$$2^{-j+1}\sqrt{\frac{m}{C\epsilon}}\cdot 2^j \cdot C = 2\sqrt{m}\sqrt{\frac{C}{\epsilon}}. \tag{48}$$

As noted above, $j \leq \log N$ and thus we can sum over j and bound our overall cost by $2\sqrt{m}\sqrt{C/\epsilon}\log N$. Plugging in the minimal cost per interval $||b'||/\epsilon$, we obtain total cost

$$2\sqrt{m}\frac{\sqrt{||b'||}}{\epsilon}\log N. \tag{49}$$

Remark: According to these estimates, our best results are obtained by shrinking the intervals until the non-bell cost equals the bell cost. Since the bell cost is independent of the interval size, this indicates that our choice to have all intervals of the same cost should be optimal.

4.3.2 Higher Order Estimates

The above estimates are fine, but depend on ϵ like ϵ^{-1}. Here we apply Theorem 4.1 repeatedly to get higher order decay in ξ, and thus more benign dependence on ϵ.

If we apply Theorem 4.1 one time, we convert a single term into 3 terms. One of these has multiplicity 2, so we'll consider there to be 4 terms. If we apply the theorem k times, we expect 4^k terms. Along the way we take derivatives of products and powers, which will give us further multiplicities. In going from step k to step $k+1$, we multiply by at most $(2k)(2k+1)$. Our total number of terms, including multiplicities, is thus bounded by $4^k(2k)!$. Within some terms we will also generate a factor of $k!$. We account for this now so that when it appears later, we can ignore it. We therefore proceed as if there were $4^k(2k)!k!$ terms.

Each term has the following form:

$$\frac{|\xi|^a \cdot ||b^{(b)}|| \cdot l^{2k-(a+b)} \cdot ||V - V^*||^c \cdot |V^{(1)}|^{d_1} \cdot |V^{(2)}|^{d_2} \cdots}{|\xi^2 - l^2 V^*|^k} \tag{50}$$

with $a + b + 2c + \sum d_i(i+2) = 2k$, $a \leq k$, and if $a = k$ then $b = k$. We wish to know when this is greater than $\epsilon/4^k(2k)!k!$. We will separate each term into k factors and determine when each is greater than $(\epsilon/4^k(2k)!k!)^{1/k} \approx \epsilon^{1/k}/k^3 = \eta$. (We use Stirling's formula $n! \approx \sqrt{2\pi}n^{n+1/2}e^{-n}$.) The η-support of each factor is two intervals, containing the points $\pm l\sqrt{V^*}$. The union of these intervals is essentially the largest pair of intervals, and will contain the intervals when our term is greater than $\epsilon/4^k(2k)!k!$.

Following Section 4.3.1, we fix a sub-cost C and determine the smallest x which, given an l, yields cost C for this factor. We will have some factors whose cost is independent of l and x (and $V(x)$) and these will act as a minimum on C. When we can affect the cost with l, we will show that l is a linear function of x. It may have x-intercept at 0, in which case its slope is independent of m and n. It could instead have x-intercept at m/n, in which case its slope depends on m like $m^{-1/2}$. As we saw in Section 4.3.1 and using (48), a linear dependence between l and x (or $(x - m/n)$) means our total cost will depend on N like $\log N$. The total cost also depends on the slope of the line like its inverse. For us this is either independent of m or depends on m like $\sqrt{m}$. Given this collection of lines, we can choose a line which is under all of them on $[m/n, 1]$, and the inverse of its slope will be $\mathcal{O}(\sqrt{m})$. This line gives us a partition of $[m/n, 1]$ for which all factors obey the sub-cost limit C. A closer examination of its slope will tell us our dependence on C, ϵ, etc.

We organize the various types of factors in tables. Table 1 contains the basic factors, the bound used for that factor, the corresponding line, and the total cost if that was the dominant line. The bounds are analogous to those in (30) and (33). The lines are found by either setting the bound equal to C and solving for l, or by using the δ construction as in (47). The total cost is found by taking the inverse of the slope and multiplying by C. We will also use the shorthand

$$b_k = \sup_{1\leq j\leq 2k} ||b^{(j)}||^{1/j}. \tag{51}$$

Factor	Bound used	Line	Total cost	
$\dfrac{\lvert\xi\rvert b_k}{\lvert\xi^2 - l^2V^*\rvert}$	$\dfrac{b_k}{\eta}$			(52)
$\dfrac{b_k^2}{\lvert\xi^2 - l^2V^*\rvert}$	$\dfrac{b_k}{\sqrt{\eta}}$			(53)
$\dfrac{\lvert\xi\rvert lx^{-1}}{\lvert\xi^2 - l^2V^*\rvert}$	$\dfrac{lx^{-1}}{\eta}$	$l = \eta Cx$	$\dfrac{1}{\eta}$	(54)
$\dfrac{l^3m^2x^{-3}}{\lvert\xi^2 - l^2V^*\rvert}$	$\dfrac{l^2m^2}{\eta x^2\sqrt{(nx)^2 - m^2}}$	$l = \sqrt{\dfrac{C\eta}{m}}\left(x - \dfrac{m}{n}\right)$	$\sqrt{\dfrac{mC}{\eta}}$	(55)
$\dfrac{lx^{-1}}{\lvert\xi^2 - l^2V^*\rvert}$	$\sqrt{\dfrac{lx^{-1}}{\eta}}$	$l = \eta C^2 x$	$\dfrac{1}{C\eta}$	(56)
$\dfrac{l^2x^{-2}}{\lvert\xi^2 - l^2V^*\rvert}$	$\dfrac{lx^{-1}}{\sqrt{\eta}}$	$l = \sqrt{\eta}Cx$	$\dfrac{1}{\sqrt{\eta}}$	(57)

Table 1: Basic factors in the higher order estimates.

Next, in Table 2, we list the more complicated, composite terms and show how to factor them. The terms of type (58), (62), and (63) have factorials, but these were accounted for at the beginning and incorporated into η, and so can be ignored.

For large m, the worst total is in terms of type (55) and is $\sqrt{mC/\eta}$. The factors of type (52) and (53) give us bounds independent of l, and thus tell us we should choose $C = b_k/\eta$. Plugging in $C = b_k/\eta$ and $\eta = \epsilon^{1/k}/k^3$ yields $\sqrt{mb_k}k^3\epsilon^{1/k}$. To estimate b_k, we note $b^{(j)}$ should have j peaks, about $1/j$ apart. The height of these peaks can be found from the height of $b^{(j-1)}$'s peaks by multiplying by the previous separation $(1/(j-1))$ and an extra factor of two. Inductively this gives $|b^{(j)}| \sim 2^j j!$ and so $b_k \sim 4k$. Plugging in this estimate yields cost $\sqrt{m}k^{7/2}\epsilon^{1/k}$. Choosing $k = \log(1/\epsilon)$ gives us a compression cost of $\mathcal{O}(\sqrt{m}\log N(\log\epsilon)^{7/2})$.

Remark: The estimate $b_k \sim 4k$ is far from rigorous. In applications, we use bells which are optimized (Section 5.1.2) using a method that does not look at derivatives of the bell. We have found these bells to be highly efficient.

4.3.3 One Dimensional Compression Conclusion: $\mathcal{O}(N^{5/2}\log N(\log\epsilon)^{7/2})$

We concluded above that we can compress each Associated Legendre function into at most

$$\mathcal{O}(\sqrt{m}\log N(\log\epsilon)^{7/2}) \leq \mathcal{O}(\sqrt{N}\log N(\log\epsilon)^{7/2}) \tag{64}$$

coefficients. We must add this count over N values of m and N values of n. For the complete Spherical Harmonic transform, this yields a bound of

$$\mathcal{O}(N^{5/2}\log N(\log\epsilon)^{7/2}). \tag{65}$$

Term	Factorization	
$\frac{\lvert\xi\rvert^p \cdot \lvert\lvert V^{(p)}\rvert\rvert \cdot l^{p+2}}{\lvert\xi^2 - l^2V^*\rvert^{p+1}}$	$= \frac{m^2l^3x^{-3}}{\lvert\xi^2 - l^2V^*\rvert} \frac{\lvert\xi\rvert}{\lvert\xi^2 - l^2V^*\rvert} \left(\frac{\lvert\xi\rvert lx^{-1}}{\lvert\xi^2 - l^2V^*\rvert}\right)^{p-1} (p+2)!$	(58)
$\frac{\lvert\lvert V - V^*\rvert\rvert}{\lvert\xi^2 - l^2V^*\rvert}$	$\leq \frac{m^2l^3x^{-3}}{\lvert\xi^2 - l^2V^*\rvert}$	(59)
$\frac{\lvert\xi\rvert^p\lvert\lvert b^{(p)}\rvert\rvert}{\lvert\xi^2 - l^2V^*\rvert^p}$	$\leq \left(\frac{\lvert\xi\rvert b_k}{\lvert\xi^2 - l^2V^*\rvert}\right)^p$	(60)
$\frac{\lvert\xi\rvert^p\lvert\lvert b^{(p+q)}\rvert\rvert}{\lvert\xi^2 - l^2V^*\rvert^{p+q/2}}$	$\leq \left(\frac{\lvert\xi\rvert b_k}{\lvert\xi^2 - l^2V^*\rvert}\right)^p \left(\frac{b_k^2}{\lvert\xi^2 - l^2V^*\rvert}\right)^{q/2}$	(61)
$\frac{l^{p+2}\lvert\lvert V^{(p)}\rvert\rvert}{\lvert\xi^2 - l^2V^*\rvert^{1+p/2}}$	$= \frac{m^2l^3x^{-3}}{\lvert\xi^2 - l^2V^*\rvert} \frac{lx^{-1}}{\lvert\xi^2 - l^2V^*\rvert} \left(\frac{l^2x^{-2}}{\lvert\xi^2 - l^2V^*\rvert}\right)^{p/2-1} (p+2)!$	(62)
$\frac{\lvert\xi\rvert^p l^{p+q+2}\lvert\lvert V^{(p+q)}\rvert\rvert}{\lvert\xi^2 - l^2V^*\rvert^{p+1+q/2}}$	$= \frac{m^2l^3x^{-3}}{\lvert\xi^2 - l^2V^*\rvert} \frac{\lvert\xi\rvert}{\lvert\xi^2 - l^2V^*\rvert} \left(\frac{\lvert\xi\rvert lx^{-1}}{\lvert\xi^2 - l^2V^*\rvert}\right)^{p-1} \left(\frac{l^2x^{-2}}{\lvert\xi^2 - l^2V^*\rvert}\right)^{q/2} (p+q+2)!$	(63)

Table 2: Composite terms in the higher order estimates.

4.4 Two Dimensional Compression: $\mathcal{O}(N^2 \log N (\log \epsilon)^7)$

The compression scheme described above involves compressing $\sqrt{\sin\phi}\tilde{P}_n^m(\cos\phi)$ as a function of ϕ, and so is one-dimensional. We are actually applying a matrix of $\sqrt{\sin\phi}\tilde{P}_n^m(\cos\phi)$ (m fixed) and so can consider this as a function of both n and ϕ and compress in both directions. In the n direction we don't have a differential equation, but instead a recurrence relation (see Section 5.1.1). If we model with Bessel functions, however, we are dealing with $\sqrt{nx}J_m(nx)$ for which n and y are interchangeable. Setting $y = n/N$ we consider $\sqrt{yxN}J_m(yxN)$ on $(x, y) \in [0,1] \times [0,1]$ which is fully symmetric. Obtaining rigorous results for the Associated Legendre functions from results for Bessel functions in the n variable seems difficult, but we expect the behavior to be the same. We state the result for Bessel functions:

Theorem 4.6 *Let $\psi(x, y) = \sqrt{yxN}J_m(yxN)$ on the unit square, normalized so $||\psi(\cdot, y)||_2 \leq 1$. There exists a partition into squares with total expansion cost bounded by*

$$\mathcal{O}(N \log N (\log \epsilon)^7). \tag{66}$$

In Section 4.1 we constructed tools for one dimensional compression and in Section 4.3 we applied them. Here we will try to use these tools and techniques as much as posssible and only highlight the differences.

We are allowed to choose any partition of the unit $(x \times y)$ square into rectangles, but for now we restrict ourselves to squares of sidelength 2^{-j}. We first note that the turning point occurs when $x = m/yN$. Anything above or to the left of this hyperbola is either rapidly decaying or zero, so we don't have to worry about this region. (Recall that since M is a matrix, the x axis points down.)

First we fix a sub-cost C^2 as our goal on the subsquares. Fixing j, we can break the unit square into three regions. The first is a 'good' region for the size 2^{-j}, denoted G_j, where the cost is less than C^2. The second is a 'bad' region (B_j) where the bounds fail, and the third is a 'nothing' region beyond the turning point. We break the unit square into the nothing region and $G_0 \cup_{j=0}^{\infty} (B_j \cap G_{j+1})$. To compute the number of squares needed for a region, we can compute the area of that region and divide it by the area of the squares used. Thus for the region $B_j \cap G_{j+1}$ we need $|B_j \cap G_{j+1}| \cdot 2^{2j+2}$ squares. Multiplying by C^2 will give the number of coefficients. Our problem is now reduced to estimating the area of these regions.

The first step is to assess the cost associated with the square $[x, x+2^{-j}] \times [y, y+2^{-j}]$. This cost will be bounded by

$$\left| \bigcup_{0 \le t \le 2^{-j}} 2^{-j}\epsilon\text{-supp}\Big((b_{[x,x+2^{-j}]}\psi^m_{y+t}(\cdot))^{\widehat{}} \Big) \right| \cdot \left| \bigcup_{0 \le s \le 2^{-j}} 2^{-j}\epsilon\text{-supp}\Big((b_{[y,y+2^{-j}]}\psi^m_{(\cdot)}(x+s))^{\widehat{}} \Big) \right| \tag{67}$$

If we require this to be less than C^2 it suffices to have each factor less than C. When computing the ϵ-supports, the only dependence on m was through $||V(x) - V^*||$. Recalling now that V also depends on y, we instead consider $\sup |V_y(x) - V^*|$ taken over the square $[x, x+2^{-j}] \times [y, y+2^{-j}]$ with V^* independent of y. The cost computed using this will dominate the first factor in (67).

For our particular potential, $V_y(x) = (yN)^2 - m^2/x^2$, we can choose $V^* = (yN)^2 - m^2/x^2$ and achieve the supremum with

$$V_{y+2^{-j}}(x+2^{-j}) - V^* = y^2N^2 + 2y2^{-j}N^2 + l^2N^2 - \frac{m^2}{(x+2^{-j})^2} - yN^2 + \frac{m^2}{x^2} \tag{68}$$

$$= m^2 \frac{2^{-j}(2x+2^{-j})}{x^2(x+2^{-j})^2} + 2y2^{-j}N^2 + l^2N^2. \tag{69}$$

We will consider the costs from these three terms separately. We will show the three bad regions shrink at least at a certain rate. The analysis of the second factor in (67) yields the same rate, and so the union of all of these regions also has this same rate. We then use $|B_j|$ to dominate $|B_j \cap G_{j+1}|$.

The first term gives cost

$$\frac{m^2 2^{-j}(2x+2^{-j})}{x^2(x+2^{-j})^2} \frac{2^{-j}}{\sqrt{(yN)^2 - m^2/x^2\epsilon}}. \tag{70}$$

We set this equal to C, define $\gamma = x - m/yN$ and manipulate to find

$$\frac{m^2(2^{-j})^2}{C\epsilon} = \frac{(\gamma + 2^{-j} + m/yN)^2 \sqrt{\gamma y N}\sqrt{\gamma y N + 2m}}{2\gamma + 2^{-j} + 2m/yN}. \tag{71}$$

Following the argument from Section 4.3.1, we note it is sufficient to use δ such that

$$\frac{m^2(2^{-j})^2}{C\epsilon} = \delta^{1/2}\left(\frac{m}{yN}\right)^{1/2} \frac{m}{N}\sqrt{\delta y N}\sqrt{m}, \tag{72}$$

which means $\delta = N(2^{-j})^2/C\epsilon$ will suffice. This $\delta = \delta_j$ bounds the thickness of the strip B_j. Since this strip has length at most 1, δ_j also bounds its area. Our cost for the region $B_j \cap G_{j+1}$ is thus less than $\delta(2^{-j})^{-2} = NC/\epsilon$. Since a square must contain a point, $j \le \log N$, and we can add our cost over $\log N$ strips to obtain total cost $(C/\epsilon)N \log N$. See Figure 7 for a schematic of the different regions and Figure 2 for the original matrix.

The second term gives cost

$$y2^{-j}N^2 \frac{2^{-j}}{\sqrt{(yN)^2 - m^2/x^2\epsilon}}. \tag{73}$$

We set this equal to C, define $\gamma = x - m/yN$ and manipulate to find

$$\frac{(2^{-j})^2 yN^2}{C\epsilon} = \frac{\sqrt{\gamma y N}\sqrt{\gamma y N + m^2}}{(\gamma + m/yN)^2}. \tag{74}$$

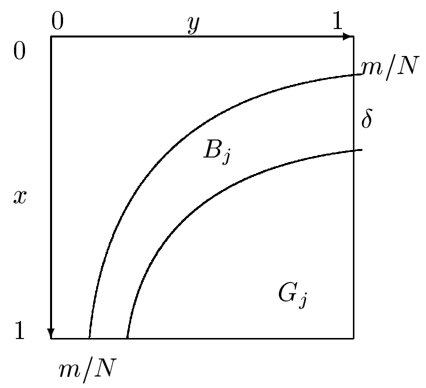

Figure 7: The good and bad regions for a certain sized square.

It will suffice to take δ such that

$$\frac{(2^{-j})^2 y N^2}{C\epsilon} = \delta y N \tag{75}$$

and so $\delta = (2^{-j})^2 N/C\epsilon$ will work. This yields NC/ϵ as the cost for a strip and total cost $(C/\epsilon)N \log N$.

The third term gives cost

$$(2^{-j})^2 N^2 \frac{2^{-j}}{\sqrt{(yN)^2 - m^2/x^2\epsilon}}. \tag{76}$$

We set this equal to C, define $\gamma = x - m/yN$ and manipulate to find

$$\frac{(2^{-j})^3 N^2}{C\epsilon} = \frac{\sqrt{\gamma y N}\sqrt{\gamma y N + m^2}}{(\gamma + m/yN)^2}. \tag{77}$$

It will suffice to take δ such that

$$\frac{(2^{-j})^3 N^2}{C\epsilon} = \delta y N \tag{78}$$

which means $\delta y = (2^{-j})^3 N/C\epsilon$ will work. Anticipating the results from the analysis of the first term in the compression in y, we assume $y \geq \delta + m/x \geq \delta$ (for the same δ) and so we can take $\delta^2 = (2^{-j})^3 N/C\epsilon$ and

$$\delta = \frac{(2^{-j})^2 N}{\sqrt{C\epsilon}} \frac{1}{\sqrt{2^{-j}N}}. \tag{79}$$

Noting $2^{-j} \geq 1/N$ and so $1/\sqrt{2^{-j}N} < 1$, we can take $\delta = (2^{-j})^2 N/\sqrt{C\epsilon}$, yielding total cost $(C/\epsilon)N\sqrt{C\epsilon}$.

As in the one dimensional case, we can use Theorem 4.1 iteratively to obtain higher order decay. Excluding the case (55) dealt with above, all the basic factors in Table 1 yielded lines of the form $x = lA = 2^{-j}A$ with A independent of m, n, y, etc. These lines give us strip cost of $A/2^{-j}$ and total cost $A\sum_{j=0}^{\log N}(2^{-j})^{-1} = AN$. We expect ϵ dependence like $((\log\epsilon)^{7/2})^2$. For each m we can compress into $\mathcal{O}(N\log N(\log\epsilon)^7)$ coefficients and so have total coefficient count

$$\mathcal{O}(N^2 \log N(\log\epsilon)^7). \tag{80}$$

5 Numerical Results

The proof of the theoretical result of the fast transform for Spherical Harmonics is now complete. When attempting to implement the algorithms, several issues arise. This section first gives the solutions to those problems which proved most troublesome. It then gives the results of numerical testing on an implementation of the algorithms from Section 3.

Our implementation was mainly for diagnostic purposes. When evaluating run times we attempted an efficient implementation, but our main goal was to provide a fair comparison with the direct method. We seek to answer the following questions:

1. Does the cost of expansion agree with the prediction in Section 4?
   (a) As a function of N?
   (b) As a function of ϵ?
2. When are our algorithms better than the direct method, specifically:
   (a) At what N do they become faster?
   (b) Are there other considerations we need to take into account?
3. Are our algorithms stable, and how large are the errors?

To summarize the results:

1. The one dimensional algorithm performs better than predicted, as a function of both N and ϵ. The two dimensional algorithm performs well in ϵ, but poorly in N, in the range of N we tested.
2. The one dimensional algorithm becomes faster than the direct method at $N = 128$. At this size, the overhead cost for full adaptation is too great, so we had to modify our adaptation process. By $N = 512$, we have improved run times by a factor of three.
3. Since we perform only orthogonal operations, the one dimensional algorithm is stable and has errors about size ϵ.

5.1 Computational Details

5.1.1 Generating the $\tilde{P}_n^m$'s

Before we compress a matrix of Associated Legendre functions (or ψ_n^m's) we must be able to construct the matrix. Let $\tilde{P}_k^{(m,m)}$ be the L^2 normalized Jacobi Polynomial. We can construct the L^2 normalized Associated Legendre functions for $m \geq 0$ as:

$$\tilde{P}_n^m(\cos\phi) = (\sin\phi)^m \tilde{P}_{n-m}^{(m,m)}(\cos\phi) \tag{81}$$

and for $m < 0$ using $\tilde{P}_n^m(\cos\phi) = \tilde{P}_n^{-m}(\cos\phi)$. Using the recurrence relation and normalization from Szegö, [Sze75, p.68,71] we can construct $\tilde{P}_k^{(m,m)}$ with the recurrence initialized by

$$\tilde{P}_{-1}^{(m,m)}(x) \equiv 0 \tag{82}$$

$$\tilde{P}_0^{(m,m)}(x) = \sqrt{\frac{\Gamma(2m+2)}{2}} \frac{1}{2^m\Gamma(m+1)} = \frac{1}{\sqrt{2}} \prod_{j=1}^{m} \sqrt{1 + \frac{1}{2j}} \tag{83}$$

and with general term

$$\tilde{P}_k^{(m,m)}(x) = 2x\tilde{P}_{k-1}^{(m,m)}(x)\left(1+\frac{m-1/2}{k}\right)^{1/2}\left(1-\frac{m-1/2}{k+2m}\right)^{1/2}$$
$$- \tilde{P}_{k-2}^{(m,m)}(x)\left(1+\frac{4}{2k+2m-3}\right)^{1/2}\left(1-\frac{1}{k}\right)^{1/2}\left(1-\frac{1}{k+2m}\right)^{1/2}. \quad (84)$$

For large m this recurrence is poorly conditioned, especially when $m \leq n \leq 2m$. The solution we want corresponds to the larger eigenvalue, however, so the conditioning acts in our favor. The recurrence is also highly prone to underflows. We fix this by using scientific notation, $\sqrt{\sin\phi}\tilde{P}_n^m(\cos\phi) = A2^B$, $1/2 < A \leq 1$, $B \in \mathbf{Z}$, and storing A and B separately.

5.1.2 Bells

In determining the order of our algorithm, the bell used is unimportant, except perhaps that it have a few bounded derivatives. In determining the constants involved, however, it it crucial. We use bells developed by Matviyenko [Mat96], which are optimized for a given precision. These bells are bi-orthogonal, so we expand using the bells $\{b_i(x)\}$ and reconstruct with the dual bells $\{b_i(x)/\sum_j b_j^2(x)\}$. The use of these bi-orthogonal bells has condition number $\sqrt{2}$, so we will pretend as though we are using orthogonal bells.

5.1.3 Searches

In Section 4.3 we developed a theory which said we can achieve a certain amount of compression if we can find an optimal partitioning of the interval. We find the optimal partition by searching, and here present fast searches.

First consider the one dimensional partition from Section 4.3.1. We know that the interval size should grow slowly as we move to the right, so we explicitly restrict to partitions where an interval's left neighbor is either the same size or one size smaller. Our compression bounds only hold if all our bells are dilates of a single bell. With our restricted partition we can have good (broadly supported) bells and yet only use two different bells. When an interval's left neighbor is of the same size, we use a 'full bell', which will be supported on the double of the interval. When the left neighbor is half the size, we adjust the left half of our bell to be compatible with the full bell of the smaller neighbor, creating an 'asymmetric bell'.

We place the root node to the left of our interval. It has $\log(N)$ children, which correspond to the leftmost dyadic intervals. Now working left to right, we assign each node children corresponding to allowed right neighbors of the current interval. (See Figure 8.) The cost of a partition is the cost of a path from the root node to a leaf node (right edge), so we want to find the path of least cost. For our purposes, the cost is the number of coefficients above ϵ, but the search generalizes to other cost functions.

The generic decision step is as follows: We assume the current node's children contain the minimal cost of a path from them to the right. We compare the costs for the symmetric child of the same size and the asymmetric child of the next larger size (if it exists), and *add* the smaller to the symmetric and asymmetric costs of the current node. To each box associate its center. Process the boxes in order of the x coordinate of their center, starting at the right. This reduces us to the children of the root node, which we compare by hand. It costs $2N\log_2^2 N$ to compute the coefficients and $4N$ to run the search.

For the two dimensional compression scheme in Section 4.4 we also require a search algorithm. Our use of double bells allows us to use an algorithm for determining the best partition into dyadic rectangles developed by Nick Bennett [Ben97]. The search using squares only is given in [Wic94, p.299]. See also [TV96] for a similar algorithm.

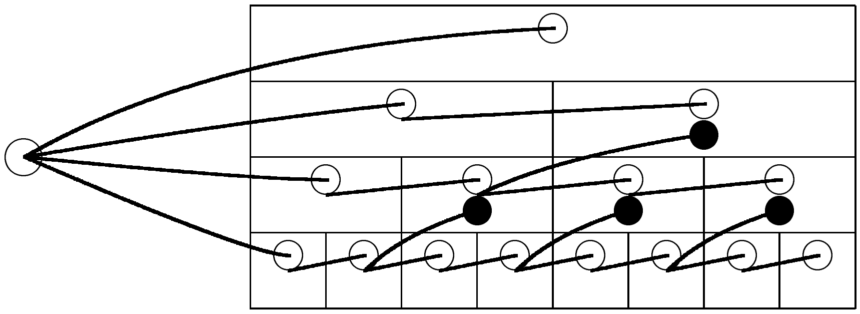

Figure 8: The one dimensional search graph. Empty circles represent full bells and solid circles asymmetric bells.

5.2 Coefficient Counts

5.2.1 One Dimensional

First we consider the coefficents counts (costs) obtained from the one dimensional compression scheme from Section 4.3. These coefficients would be used in the one dimensional algorithm in Section 3.2. Our algorithm to obtain the counts is as follows:

1. Fix some ϵ to serve as a cutoff, and the number of points N. This ϵ determines which bell to use, as in Section 5.1.2.

2. Loop through $0 \leq m < N$ and $m \leq n < N$. For each value do a best non-decreasing dyadic partition search as in Section 5.1.3 in ϕ to $\psi_n^m(\phi)$ on $[0, \pi/2]$. This search gives us a cost for each (m, n) which we add to a running total.

If we did no compression, we would be adding over $N/2$ points, $0 \leq m < N$, and $m \leq n < N$, for total cost $N^2(N+1)/4$. Dividing this number by the number of coefficients we kept gives us the compression ratio. Our prediction in Section 4.3.3 is that the number we keep for each Associated Legendre function should be $\mathcal{O}(\sqrt{N})$. To test this, we divide the number kept by $\sqrt{N}N(N+1)/2$ to obtain the 'effective constant'. The results for this method are contained in Table 3.

	$\epsilon = 10^{-6}$		$\epsilon = 10^{-12}$	
N	Ratio	Constant	Ratio	Constant
32	1.09	2.59	1.00	2.82
64	1.32	3.04	1.07	3.71
128	1.67	3.38	1.30	4.34
256	2.18	3.67	1.64	4.88
512	3.03	3.74	2.08	5.44
1024	4.39	3.65	2.81	5.68
2048	*6.47	*3.49	*3.97	*5.70
4096	*9.67	*3.31	*5.51	*5.81

Table 3: Compression ratios and effective constants for compression in ϕ only. A * indicates an estimate by sampling in m.

Preliminary timing results indicated that the cost of the LCTs at multiple scales is prohibitive at $N \approx 256$. As an alternative to using multiple scales, we can choose a single partition for each m

and parity. (See Section 5.3.2.) In Table 4 we give compression ratios and effective constants for this method.

	$\epsilon = 10^{-6}$		$\epsilon = 10^{-12}$	
N	Ratio	Constant	Ratio	Constant
32	1.09	2.59	1.00	2.82
64	1.28	3.12	1.07	3.71
128	1.59	3.56	1.28	4.41
256	2.07	3.86	1.56	5.14
512	2.89	3.91	1.97	5.75
1024	4.26	3.75	*2.69	*5.94
2048	*6.31	*3.59	*3.88	*5.83

Table 4: Compression ratios and effective constants for compression in ϕ only using a single partition per m (and parity). A * indicates an estimate by sampling in m.

5.2.2 Two Dimensional

Next, in Table 5, we consider the two dimensional compression scheme, as in Section 4.4. Our algorithm is now:

1. Fix some ϵ to serve as a cutoff, and the number of points N. This ϵ determines which bell to use, as in Section 5.1.2.
2. Loop through $0 \leq m < N$. For each value do the two dimensional search (Section 5.1.3) to $(\psi_n^m(\phi))$ as a function of ϕ and n on $[0, \pi/2] \times [0, N]$. We allow the search to choose not to try to compress any particular rectangle in either direction. The search produces a cost for each m which we add to a running total.

We compute the compression ratios and effective constants as above. The 'constant' is computed by dividing by $\sqrt{N}N(N+1)/2$. This is *not* the rate predicted for this algorithm, but gives us a good way to compare it with the one dimensional algorithm. For the two dimensional algorithm to work as predicted, we want the constant to decrease at a rate like $1/\sqrt{N}$ or perhaps $\log N/\sqrt{N}$.

	$\epsilon = 10^{-6}$		$\epsilon = 10^{-12}$	
N	Ratio	Constant	Ratio	Constant
32	1.13	2.50	1.04	2.72
64	1.31	3.05	1.13	3.55
128	1.59	3.55	1.32	4.30
256	1.98	4.03	1.55	5.16
512	2.60	4.35	1.90	5.96
1024	3.56	4.49	*2.39	*6.69

Table 5: Compression ratios and effective constants (using one dimensional rates) for compression in ϕ and n. A * indicates an estimate by sampling in m.

The two dimensional algorithm is not performing very well in this range of N. The analysis in Section 4.4 predicts the constant in the two dimensional algorithm to be the square of that in the one dimensional. The double bells will give an extra factor of 4 inefficiency. At smaller N, other constraints enable us to still have compression, but we don't expect to see the predicted rate until much larger N. Ad hoc methods allow us to outperform the one dimensional compression, but the effect is marginal: a 25% reduction in coefficients at $N = 1024$. At some point in the future, this algorithm will have to be revisited.

5.3 Program Timings

The coefficient counts in the previous section are essentially implementation-invarient. When trying to compare the speed of these algorithms to the direct method, there are implementation-dependent and hardware-dependent factors. We attempt here to give fair comparisons.

All programs were implemented in ANSI C, compiled with `cc -O -native -dalign` and run on a Sun Ultra2 with UltraSPARCII cpu running at 300Mhz. The timing mechanism is based on internal system queries as to how much CPU time has been used. The times given are the average over multiple runs in an attempt to give two significant digits.

5.3.1 The Direct Method

We first time the direct method and those components which are common to both the direct and fast methods. We deal with the collection of reduced evaluation problems using parities, as stated in Section 2.3. These results appear in Table 6. The component operations are:

Reflect Using parities, reflect the even and odd parts of our function to regain the function itself.

Transpose Transpose the matrix in $\phi \times \theta$.

FFTθ Perform the N real FFTs of length $2N$ in the θ variable.

Total (C) The total of the common elements: Reflect + Transpose + FFTθ. Abbreviated as C.

Apply The time to apply the matrices in ϕ for all m.

Apply+C The total cost for a transform using the direct method.

	Common				Direct	
N	Reflect	Transpose	FFTθ	Total (C)	Apply	Apply+C
64	.00039	.00042	.0021	.0029	.0062	.0091
128	.0015	.0015	.0094	.012	.065	.077
256	.011	.035	.042	.088	.60	.69
512	.043	.19	.19	.42	*8.5	*8.9

Table 6: Timing for common elements and the direct method, in seconds. A * indicates an estimate by sampling in m.

Remark: We expect times for the direct method to grow by a factor of 8 when we double N, but they are growing faster than that. The number of operations is growing exactly by 8, so the discrepancy should be in system dependent factors such as pipelining, cache size, and the nuaces of compiler optimization. We cannot hope to account for these factors, so we instead tried to structure the direct and fast algorithms as much alike as possible, to provide a fair comparison.

5.3.2 The One Dimensional Algorithm

For the values of N we have tested, the one dimensional algorithm is performing best in coefficient counts, so we use it for our performance testing. Again we consider the collection of reduced evaluation problems using parities, as stated in Section 2.3. Preliminary timings demonstrated that at N of size around 256, performing $\log_2 N$ FFTs takes far longer than applying the direct method. We are forced to perform the equivalent of one FFT of length N for each scale we allow. As noted in Table 4, at N this size, choosing a single partition for each m (and even/odd) hardly hurts the compression rates. Since this greatly reduces the number of FFTs, it is a good trade off. As N becomes larger, FFTs become relatively cheap, and it will be worthwhile to allow more partitions. It is possible to modify the search algorithm in Section 5.1.3 to take the cost of additional FFTs into account when choosing the partitions.

The times for the one dimensional algorithm using a single partition per m and parity are shown in Table 7. The component operations are:

AC (Apply Compressed) Apply the matrix in compressed form for all m.

LCTs Perform the Local Cosine transforms to return to normal coordinates.

Total AC + LCTs. Abbreviated as F6 when $\epsilon = 10^{-6}$ and F12 when $\epsilon = 10^{-12}$. This takes the place of the Apply step in Table 6.

F6+C, F12+C The total cost for a transform using this method.

	Fast at $\epsilon = 10^{-6}$				Fast at $\epsilon = 10^{-12}$			
N	AC	LCTs	Total(F6)	F6+C	AC	LCTs	Total(F12)	F12+C
64	.0058	.0036	.0093	.012	.0067	.0036	.011	.013
128	.042	.015	.057	.069	.055	.015	.070	.082
256	.28	.072	.35	.44	.37	.066	.43	.52
512	1.6	.28	1.9	2.3	*2.5	.28	*2.8	*3.2

Table 7: Timing for the one dimensional algorithm, in seconds, using a single partition per m (and parity). A * indicates an estimate by sampling in m.

To ease the comparison of the direct and fast methods, we give ratios of run-times in Table 8.

	$\epsilon = 10^{-6}$		$\epsilon = 10^{-12}$	
N	Apply/F6	(Apply+C)/(F6+C)	Apply/F12	(Apply+C)/(F12+C)
64	0.67	0.76	0.56	0.70
128	1.14	1.12	0.92	0.93
256	1.71	1.57	1.40	1.33
512	*4.47	*3.87	*3.04	*2.78

Table 8: Ratios of run times of the one dimensional algorithm versus the direct method. A * indicates an estimate by sampling in m.

References

[AR91] Bradley Alpert and Vladimir Rokhlin. A fast algorithm for the evaluation of Legendre expansions. *SIAM J. Sci. Statist. Comput.*, 12(1):158–179, January 1991.

[AS64] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*, volume 55 of *Applied Math Series*. National Bureau of Standards, 1964.

[BCG93] B. Bradie, R. Coifman, and A. Grossmann. Fast numerical computations of oscillatory integrals related to acoustic scattering, i. *Appl. Comput. Harmon. Anal.*, 1:94–99, 1993.

[BCR91] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms i. *Comm. Pure Appl. Math.*, 44:141–183, 1991.

[Ben97] Nicholas N. Bennett. *Signal Analysis of Chirps: Detection, Oscillatory Kernels, and Anisotropic Wavelets*. PhD thesis, Yale University, New Haven CT, May 1997.

[BO78] Carl M. Bender and Steven A. Orszag. *Advanced mathematical methods for scientists and engineers*. International series in pure and applied Mathematics. McGraw–Hill, New York, 1978.

[Bro85] Timothy M. Brown. Solar rotation as a function of depth and latitude. *Nature*, 317(17):591–594, October 1985.

[CM91] Ronald R. Coifman and Yves Meyer. Remarques sur l'analyse de Fourier à fenêtre. *C. R. Académie des Sciences*, 312(1):259–261, 1991.

[DEH89] Thomas L. Duvall, Jr., Mark Elowitz, and Frank Hill. A test of a modified algorithm for computing spherical harmonic coefficients using an fft. *J. Comput. Phys.*, 80:506–511, 1989.

[DH94] James R. Driscoll and Dennis M. Healy, Jr. Computing Fourier transforms and convolutions on the 2–sphere. *Adv. in Appl. Math.*, 15:202–250, 1994.

[Dil85] Gary A. Dilts. Computation of spherical harmonic expansion coefficients via fft's. *J. Comput. Phys.*, 57:439–453, 1985.

[EMOT54] Arthur Erdélyi, Wilhelm Magnus, Fritz Oberhettinger, and Francesco G. Tricomi. *Tables of Integral Transforms*, volume 2 of *Bateman Manuscript Project*. McGraw-Hill, New York, 1954.

[FPTV92] B. Flannery, W. Press, S. Teukolsky, and W. Vettering. *Numerical Recipies in C*. Cambridge University Press, Cambridge UK, 2nd edition, 1992.

[Hob31] E. W. Hobson. *The Theory of Spherical and Ellipsoidal Harmonics*. Chelsea, New York, 1931.

[LL77] L. D. Landau and E. M. Lifschitz. *Quantum Mechanics (Non–relativistic Theory)*. Pergamon Press, New York, 3rd edition, 1977.

[Mat96] Gregory Matviyenko. Optimized local trigonometric bases. *Appl. Comput. Harmon. Anal.*, 3(4):301–323, 1996.

[Moh97] Martin J. Mohlenkamp. *A Fast Transform for Spherical Harmonics*. PhD thesis, Yale University, New Haven CT, May 1997.

[Ors86] Steven A. Orszag. Fast eigenfunction transforms. *Advances in Mathematics Supplementary Studies*, 10, 1986.

[SW71] Elias Stein and Guido Weiss. *Fourier Analysis on Euclidean Spaces*. Princeton University Press, Princeton NJ, 1971.

[Sze75] Gabor Szegö. *Orthogonal Polynomials.* AMS, Providence, RI, 4th edition, 1975.

[TV96] Christoph M. Thiele and Lars F. Villemos. A fast algorithm for adapted time–frequency tilings. *Appl. Comput. Harmon. Anal.*, 3(2):91–99, 1996.

[Wic94] Mladen Victor Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* A.K. Peters, Wellesley, MA, 1994.