

Algorithm 888: Spherical Harmonic Transform Algorithms

JOHN B. DRAKE, PAT WORLEY and EDUARDO D'AZEVEDO
Oak Ridge National Laboratory

A collection of MATLAB classes for computing and using spherical harmonic transforms is presented. Methods of these classes compute differential operators on the sphere and are used to solve simple partial differential equations in a spherical geometry. The spectral synthesis and analysis algorithms using fast Fourier transforms and Legendre transforms with the associated Legendre functions are presented in detail. A set of methods associated with a `spectral_field` class provides spectral approximation to the differential operators ∇ , $\nabla \times$, $\nabla \cdot$, and ∇^2 in spherical geometry. Laplace inversion and Helmholtz equation solvers are also methods for this class. The use of the class and methods in MATLAB is demonstrated by the solution of the barotropic vorticity equation on the sphere. A survey of alternative algorithms is given and implementations for parallel high performance computers are discussed in the context of global climate and weather models.

Categories and Subject Descriptors: F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—*Computation of transforms*; G.4 [Mathematical Software]; G.1.8 [Numerical Analysis] Partial Differential Equations; J.2 [Physical Sciences and Engineering]: —*Earth and atmosphere science*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Spectral transform methods, spherical, fluid dynamics, geophysical flow, high performance computing

This research at Oak Ridge National Laboratory was supported by the Climate Change Prediction Program, Office of Biological and Environmental Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with UT-Battelle, Inc. This research is sponsored by the Office of Advanced Scientific Computing Research and the Office of Biological and Environmental Research, U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract Number DE-AC0500OR22725. Authors' address: J. B. Drake, P. Worley, and E. D'Azevedo, Computer Science and Mathematics Division, Oak Ridge National Laboratory, P.O. Box 2008, MS 6367, Oak Ridge, Tennessee, 37831-6367; email: drakejb@ornl.gov.

© 2008 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by a contractor or affiliate of the [U.S.] Government. As such, the Government retains a nonexclusive royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credits is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permission@acm.org.

© 2008 ACM 0098-3500/2008/10-ART23 \$5.00 DOI: 10.1145/1391989.1404581. <http://doi.acm.org/10.1145/1391989.1404581>.

ACM Transactions on Mathematical Software, Vol. 35, No. 3, Article 23, Pub. date: October 2008.

ACM Reference Format:

Drake, J. B., Worley, P., and D’Azevedo, E. 2008. Algorithm 888: Spherical harmonic transform algorithms. *ACM Trans. Math. Softw.* 35, 3, Article 23 (October 2008), 23 pages. DOI = 10.1145/1391989.1404581. <http://doi.acm.org/10.1145/1391989.1404581>.

1. INTRODUCTION

The spherical harmonic transform is a critical computational kernel of the dynamics algorithms for numerical weather prediction and climate modeling. The announcement of sustained rates of 26.5 Tflops on the Japanese Earth Simulator (ES40, with NEC SX-6+ nodes) for an atmospheric simulation motivated a study of algorithmic options implementing spectral transforms and led to the development of this set of MATLAB classes to facilitate the study of alternative algorithms. The paper [Shingu et al. 2002] that won the 2002 Gordon Bell Award in Supercomputing used the full 640 nodes of the Japanese Earth Simulator with an atmospheric general circulation model (the AFES code) with a multi-level spectral transform algorithm. The dynamics part of the calculation accounted for 62% of the total time with the Legendre transform alone accounting for 51.8%. The columnar physics calculations that balance radiation and moist atmospheric processes only used 12% of the total run-time. The spectral resolution reported was a triangular truncation (T1279) with 96 levels on a 3840×1920 horizontal grid. This is high resolution (10km) for a weather prediction model and ultra-high for a climate model, which must be integrated in time for years instead of days. The horizontal resolution typically used for climate simulations in the U.S. research community is T85 with 26 vertical levels, which requires a 256×128 horizontal grid [Drake et al. 2005; Worley and Drake 2005]. The parallel algorithm used for these high resolution studies and benchmarking was given in Foster and Worley [1997]. The FFT algorithm used was given in Temperton [1983], a Fortran code specifically designed for vector computation of multiple blocked fast Fourier transforms.

The spatial resolution of a spectral model is referred to as a truncation and specifies the number of spectral modes retained in the representation of a scalar field. Spectral methods have been applied to a wide range of fluids problems and the theory of their application is given in Canuto et al. [1991]. For flows in a global domain, the preferred basis set for approximation of functions on the sphere is the spherical harmonic basis. The spherical harmonic transform is used to project grid point data on the sphere onto the spectral modes in an *analysis* step and an inverse transform reconstructs grid point data from the spectral information in a *synthesis* step. The synthesis step is described in Equation (1). The analysis step is described by Equations (2) and (3) consisting of the computation of the Fourier coefficient ξ^m and the Legendre transform that incorporates the Gaussian weights corresponding to the Gaussian latitudes $\mu_j = \sin(\theta_j)$.

$$\xi(\lambda, \mu) = \sum_{m=-M}^M \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu) e^{im\lambda}, \quad (1)$$

$$\xi_n^m = \sum_{j=1}^J \xi^m(\mu_j) P_n^m(\mu_j) w_j, \quad (2)$$

$$\xi^m(\mu_j) = \frac{1}{I} \sum_{i=1}^I \xi(\lambda_i, \mu_j) e^{-im\lambda_i}. \quad (3)$$

For a Gaussian grid the triangular spectral truncation requires the number of longitudes $I \geq 3M + 1$, and number of latitudes $J = I/2$, where M refers to the modal truncation number. In what follows we will assume a triangular truncation, though extension to other truncations is straightforward.

As atmospheric models push toward higher horizontal resolution, the algorithms used for the spectral transform are of interest and it is useful to have MATLAB implementations for testing and exploration of these algorithms. In this article, MATLAB classes are presented that implement the spherical harmonic transform along with methods based on the spectral method for approximating the standard differential operators in spherical geometry. The use of these classes and methods is illustrated by solving the barotropic vorticity equation on a sphere. A survey of alternative methods and a discussion of parallel algorithms on high performance computers concludes the study.

2. FORMULATION OPTIONS

2.1 BLAS Formulation

The recommended software package for the spherical harmonic transforms is SPHEREPACK [Adams and Swarztrauber 1999], a set of Fortran routines developed at the National Center for Atmospheric Research (NCAR). Another toolkit of Fortran routines is available from Wieczorek [2007]. For MATLAB computations, it is always possible to link with compiled routines using *mex* files. But there is an advantage in having native MATLAB code for exploring algorithms and testing performance. There are several collections of spherical harmonic routines currently available through the MATLAB software exchange [Kelbert 2007] but none offer methods for differential equations. The paper Simons et al. [2006], is a good reference for methods and algorithms using other spherical approximation methods.

The spherical harmonic transform can be formulated in terms of matrix operations. This follows from the fact that it is a linear transformation of one basis representation to another. Since the Fourier transform calculation is most efficiently organized with the FFT algorithm, what we describe here is a matrix formulation for the Legendre transform. A generalization to non-Gaussian grids is also possible as reported in Swarztrauber and Spatz [2000]. For high performance hardware, the efficiency of specialized matrix operations is well known and forms the basis of the LINPACK benchmark [Dongarra 2007]. The Basic Linear Algebra Subroutines (BLAS) have been optimized by most vendors and offer near peak rates. Since the Legendre transform can be expressed in matrix form we are led to explore the possibility of using BLAS routines for the computational kernel of the spherical harmonic transform.

The performance study [Shingu et al. 2002] considered inner and outer product formulations with specific attention to vectorization, but did not use a BLAS approach. In an unpublished study, Li [personal communication] described the principal sums that form the Legendre transform in the synthesis and analysis phases and determined a dense matrix formulation that took advantage of the symmetry of the associated Legendre functions. The principal sum for the *synthesis* phase is:

$$s_j^m = \sum_{n=m}^{N(m)} \xi_n^m P_n^m(\mu_j), \quad (4)$$

where the ξ_n^m is the spectral coefficient of a field. The principal sum of the *analysis* phase is:

$$\xi_n^m = \sum_{j=1}^J s_j^m P_n^m(\mu_j), \quad (5)$$

where the $s_j^m = w_j \xi^m(\mu_j)$ represents the product of the Gauss weight and the m -th Fourier coefficient at latitude j .

The matrix-matrix multiplications representing these sums require some additional notation. Let

$$\mathbf{P}_m = \begin{bmatrix} P_m^m(\mu_1) & P_{m+1}^m(\mu_1) & \dots & \dots & P_{N(m)}^m(\mu_1) \\ P_m^m(\mu_2) & P_{m+1}^m(\mu_2) & & & P_{N(m)}^m(\mu_2) \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ P_m^m(\mu_{J/2}) & P_{m+1}^m(\mu_{J/2}) & \dots & \dots & P_{N(m)}^m(\mu_{J/2}) \end{bmatrix} \quad (6)$$

be the matrix of associated Legendre functions for mode m at half of the Gauss points. Since Legendre functions are symmetric about the equator and the Gauss points are anti-symmetric, the algorithm does not require computation of the functions at all the points. The operative identities are:

$$P_n^m(\mu_{J+1-j}) = P_n^m(-\mu_j) = (-1)^{n-m} P_n^m(\mu_j). \quad (7)$$

Introducing a vector notation for the spectral coefficients,

$$\mathbf{x}_m = \begin{bmatrix} \xi_m^m \\ \xi_{m+1}^m \\ \vdots \\ \xi_{N(m)}^m \end{bmatrix} \quad (8)$$

and

$$\tilde{\mathbf{x}}_m = \begin{bmatrix} \xi_m^m \\ -\xi_{m+1}^m \\ \vdots \\ (-1)^{N(m)-m} \xi_{N(m)}^m \end{bmatrix}, \quad (9)$$

the first principal sum can be represented in a matrix-matrix multiplication formulation as:

$$\begin{bmatrix} s_1^m & s_J^m \\ s_2^m & s_{J-1}^m \\ \vdots & \vdots \\ s_{J/2}^m & s_{J/2+1}^m \end{bmatrix} = \mathbf{P}_m [\mathbf{x}_m \tilde{\mathbf{x}}_m]. \quad (10)$$

In this equation the s_j^m 's and the ξ_n^m 's are complex, while the \mathbf{P} matrix is real. Computational performance may be enhanced by explicitly separating the complex vectors into real and imaginary parts, forming input and output matrices with four columns instead of two. The inverse transform (analysis phase) involves two steps. First, a matrix-matrix multiply step uses the transpose of the Legendre matrix,

$$\begin{bmatrix} \tau_1^m & \tilde{\tau}_J^m \\ \tau_2^m & \tilde{\tau}_{J-1}^m \\ \vdots & \vdots \\ \tau_{J/2}^m & \tilde{\tau}_{J/2+1}^m \end{bmatrix} = (\mathbf{P}_m)^T \begin{bmatrix} s_1^m & s_J^m \\ s_2^m & s_{J-1}^m \\ \vdots & \vdots \\ s_{J/2}^m & s_{J/2+1}^m \end{bmatrix}. \quad (11)$$

The intermediate quantities, τ_n^m and $\tilde{\tau}_n^m$, are then used to compute the spectral coefficients,

$$\xi_n^m = \tau_n^m + (-1)^{n-m} \tilde{\tau}_n^m. \quad (12)$$

The formulation has been implemented in MATLAB, where the fast BLAS from LAPACK [Anderson et al. 1999] are available when matrix notation is used. This allows us to test the formulation as well as the assumptions of advantage with specialized library routines. The MATLAB code and class structure used to express the formulation are described in Section 3.

By timing the computational portions of the transforms, we note that a considerable amount of time is spent in packing and unpacking Fourier and spectral coefficients and very little in the matrix multiply and FFT. Both of these computational steps are highly optimized in MATLAB, using FFTW for the FFTs and LAPACK [Anderson et al. 1999] for the matrix multiply. This is very similar to the situation with using math libraries on supercomputers since these are highly optimized but may in fact require incompatible storage orders.

2.2 Open Loop Formulation

A formulation that does not require explicit data movement to accommodate special purpose routines leaves much to the compiler. A good compiler will

recognize BLAS constructs and take appropriate action depending on the size of loops, and so on.

The basic computational loops in (4) and (5) are the same, but we split them to exploit the symmetry of the Legendre functions. This is done by partitioning into odd and even modes. The first sum can be written in two parts for ($1 \leq j \leq J/2$),

$$s_j^m = \sum_{n=m}^{N(m)} \xi_n^m P_n^m(\mu_j), \quad (13)$$

and

$$s_{J+1-j}^m = \sum_{n=m,2}^{N(m)} \xi_n^m P_n^m(\mu_j) - \sum_{m+1,2}^{N(m)} \xi_n^m P_n^m(\mu_j). \quad (14)$$

The second sum is represented in different ways when $(n - m)$ is odd or even,

$$\xi_n^m = \sum_{j=1}^{J/2} (s_j^m + s_{J+1-j}^m) P_n^m(\mu_j), \quad \text{mod}_2(n - m) = 0, \quad (15)$$

$$\xi_n^m = \sum_{j=1}^{J/2} (s_j^m - s_{J+1-j}^m) P_n^m(\mu_j), \quad \text{mod}_2(n - m) = 1. \quad (16)$$

See Algorithms 3.3 and 3.4 for MATLAB implementations of the open loop formulation.

2.3 Legendre Functions On-the-Fly Formulation

In Spitz and Swarztrauber [2001], a four term recursion is given for computing the normalized associated Legendre functions. Since the recursion can be applied to an entire column of the \mathbf{P}_m matrix with vector operations, it may be advantageous to compute the functions on the fly. This has the added advantage of reducing the storage required for the spectral transform from $O(M^3)$ to $O(M^2)$.

The normalized associated Legendre functions are defined by:

$$P_n^m(\theta) \equiv \frac{1}{2^n n!} \sqrt{\frac{(2n+1)(n-m)!}{2(n+m)!}} \cos^m(\theta) \frac{d^{n+m}}{d\mu^{n+m}} (\mu^2 - 1)^n, \quad (17)$$

where $\mu = \sin \theta$. The four-term recursion starts from precomputed and stored values of the matrices \mathbf{P}_0 and \mathbf{P}_1 . Denoting a single column of the \mathbf{P}_m matrix by

$$\vec{P}_n^m = \begin{bmatrix} P_n^m(\mu_1) \\ \vdots \\ P_n^m(\mu_{J/2}) \end{bmatrix}, \quad (18)$$

the recursion in [4, equation D.1] is given by:

$$\bar{P}_{n+1}^{m+1} = \frac{1}{a_{n+m+1}} (b_n a_{m+1} \bar{P}_{n-1}^{m-1} - a_{n-m+1} \bar{P}_{n+1}^{m-1} + b_n a_{n-m-1} \bar{P}_{n-1}^{m+1}). \quad (19)$$

The coefficients of the recursion are given explicitly by $a_n = \sqrt{n(n+1)}$ and $b_n = \sqrt{\frac{(2n+3)}{(2n-1)}}$. Using this recursion, the columns of the other matrices can be computed using vector operations. These can be computed on the fly as part of the m -loop of the Legendre transforms in either formulation. Since the recursion can be split into odd and even modes it fits well with the open loop formulation without requiring either duplicate computation or intermediate storage.

A discussion of performance of these options on vector and scalar processors is given in D’Azevedo [2004].

3. MATLAB SPHERICAL HARMONIC CLASSES

The MATLAB programming language supports object-oriented programming by specifying a user class directory. The subdirectory is usually called *my_classes*. The software for spherical harmonic transforms described in this article can be added to a users directory and used to extend MATLAB’s functionality. The classes are included in the *my_classes* subdirectory with the folder naming convention *@class_name*. Following ESMF style conventions [Collins et al. 2005], the methods and data structures for the spectral transform fall into two classes: the *@gauss_grid* and the *@spectral_field*. The *@gauss_grid* class defines the evenly spaced longitudinal points and the latitudinal points corresponding to the Gauss points of the spherical grid. In addition, the geometric information, such as the radius of the sphere and the values of the associated Legendre functions at the points is stored. The standard *get*, *set* and *display* methods are provided. In the *private* methods are *grule*, which calculates the Gauss points and weights, and *shtraninit*, which computes the \mathbf{P}_m matrix to initialize the spherical harmonic transform. The initialization of an *@gauss_grid* object specifies the number of latitudinal points, n_j . A triangular truncation of the spectral coefficients is assumed, so for example, if $n_j = 32$, the spectral transform is initialized for T21 spectral fields.

Spherical harmonic transforms act on fields that are defined on spherical Gaussian grids. The *@spectral_field* class defines such a field. The analysis method that computes spectral coefficients from grid point values of a field, is *shtrana*. The inverse transform (synthesis), that computes grid point values from given spectral coefficients, is called *shtrans*. The Legendre transforms (forward and backward) are *private* methods. Algorithmic options such as previously discussed, may be included here, but the default is the open loop formulation. The MATLAB complex FFTW is used by the *shtrana(s)* methods to compute the real transforms. In addition to the basic *set*, *get*, and *display* methods, a variety of differential operators are implemented. The method library

that uses spherical transforms to discretize differential operators in spherical geometry includes:

<i>div</i> —the spherical (horizontal) divergence operator, $\nabla \cdot$	$\frac{1}{a} \left[\frac{1}{1-\mu^2} \frac{\partial U}{\partial \lambda} + \frac{\partial V}{\partial \mu} \right]$
<i>curl</i> —the curl operator, $k \cdot \nabla \times$	$\frac{1}{a} \left[\frac{1}{1-\mu^2} \frac{\partial V}{\partial \lambda} - \frac{\partial U}{\partial \mu} \right]$
<i>grad</i> —the spherical (horizontal) gradient operator, ∇	$\left(\frac{1}{a} \frac{\partial}{\partial \lambda}, \frac{1-\mu^2}{a} \frac{\partial}{\partial \mu} \right)$
<i>del2</i> —the spherical (horizontal) Laplacian operator, $\nabla^2 = \Delta$	$\frac{1}{a^2} \left[\frac{1}{1-\mu^2} \frac{\partial^2}{\partial \lambda^2} + \frac{\partial}{\partial \mu} \left((1-\mu^2) \frac{\partial}{\partial \mu} \right) \right]$
<i>del2inv</i> —the inverse Laplacian operator (Laplace equation solution)	Δ^{-1}
<i>helmholtz</i> —the solution of Helmholtz equation	$k^2 g + \nabla^2 g = f$
<i>UVinv</i> —the inversion operator for the vorticity, divergence and velocity	

To calculate the divergence of (U, V) the spectral coefficients are summed with the derivative, $H_n^m = (1-\mu^2) \frac{dP_n^m}{d\mu}$ of the spherical harmonic, according to the formula:

$$\text{div}(U, V)_n^m = \frac{1}{a} \sum_{j=1}^J [imU^m(\mu_j)P_n^m(\mu_j) - V^m(\mu_j)H_n^m(\mu_j)] \frac{w_j}{(1-\mu_j^2)}. \quad (20)$$

The U^m denotes the Fourier coefficient of the $U = u \cos \theta$ field. Note that normally multiplication by $\cos \theta$ is necessary so that the vector field is differentiable at the poles. All FFTs are computed in MATLAB using the native routines. This leads to some inefficiency due to use of complex transforms for real fields (see Exercise 3.6 of Trefethen [2000]).

The curl of (U, V) is calculated in spectral space from the formula:

$$\text{curl}(U, V)_n^m = -\frac{1}{a} \sum_{j=1}^J [imV^m(\mu_j)P_n^m(\mu_j) + U^m(\mu_j)H_n^m(\mu_j)] \frac{w_j}{(1-\mu_j^2)}. \quad (21)$$

The gradient operator is not invariant under choice of coordinate system. The operator is calculated in the (λ, μ) system as:

$$\nabla_{(\lambda, \mu)} \phi = \left(\frac{1}{a} \frac{\partial \phi}{\partial \lambda}, \frac{1-\mu^2}{a} \frac{\partial \phi}{\partial \mu} \right). \quad (22)$$

To convert to θ -coordinates we have $\nabla_{(\lambda,\mu)} = \cos\theta\nabla_{(\lambda,\theta)}$. The computation of the components uses the Legendre synthesis with H rather than P in the summation of coefficients.

$$\left\{ \frac{1}{a} \frac{\partial \phi}{\partial \lambda} \right\}_n^m = \frac{1}{a} \sum_m \sum_n im\phi_n^m P_n^m(\mu) e^{im\lambda} \quad (23)$$

and

$$\left\{ \frac{1 - \mu^2}{a} \frac{\partial \phi}{\partial \mu} \right\}_n^m = \frac{1}{a} \sum_m \sum_n \phi_n^m H_n^m(\mu) e^{im\lambda}. \quad (24)$$

Note that the first component computation could be done in Fourier space.

Similarly the Laplacian is calculated in spectral space using the eigenfunction relationship for the operator with the spherical harmonics,

$$\nabla^2 \psi_n^m = -\frac{n(n+1)}{a^2} \psi_n^m. \quad (25)$$

This relation is inverted for the solution of Laplace's equation using *del2inv*. A similar derivation applies for the Helmholtz equation.

The inversion of the (U, V) relationship with vorticity, $\xi = \mathbf{k} \cdot \nabla \times \mathbf{v}$, and divergence, $\delta = \nabla \cdot \mathbf{v}$, is given by the sums,

$$U(\lambda_i, \mu_j) = - \sum_{m=-M}^M \sum_{\substack{n=|m| \\ n \neq 0}}^{N(m)} \frac{a}{n(n+1)} [im\delta_n^m P_n^m(\mu_j) - \xi_n^m H_n^m(\mu_j)] e^{im\lambda_i},$$

$$V(\lambda_i, \mu_j) = - \sum_{m=-M}^M \sum_{\substack{n=|m| \\ n \neq 0}}^{N(m)} \frac{a}{n(n+1)} [im\xi_n^m P_n^m(\mu_j) + \delta_n^m H_n^m(\mu_j)] e^{im\lambda_i}.$$

A *test* method is provided along with a rudimentary *plot* method for spectral fields as a unit test for each method of the *spectral_field* class. Initialization of a spectral field requires a previous *gauss_grid* object. The *get* and *set* methods can be used to initialize the grid point values or the spectral coefficients. The plots in Figure 1 are from the *test* method using field values of the spherical harmonic, $Y_3^7(\lambda, \theta)$. These plots show values of Y_3^7 on the sphere along with a plot of the error in transforming the field values into spectral space, performing the backwards transformation and forming the difference. Errors are less than 10^{-14} , indicating an accurate double precision computation. To invoke the *test* method from MATLAB requires that the user *my_classes* subdirectory be added to the MATLAB path and that a *gauss_grid* and *spectral_field* are initialized. For example,

```
% addpath ~/Matlab/my_classes
% G = gauss_grid('T42',64);
% f = spectral_field('test function',G);
% test(f);
```

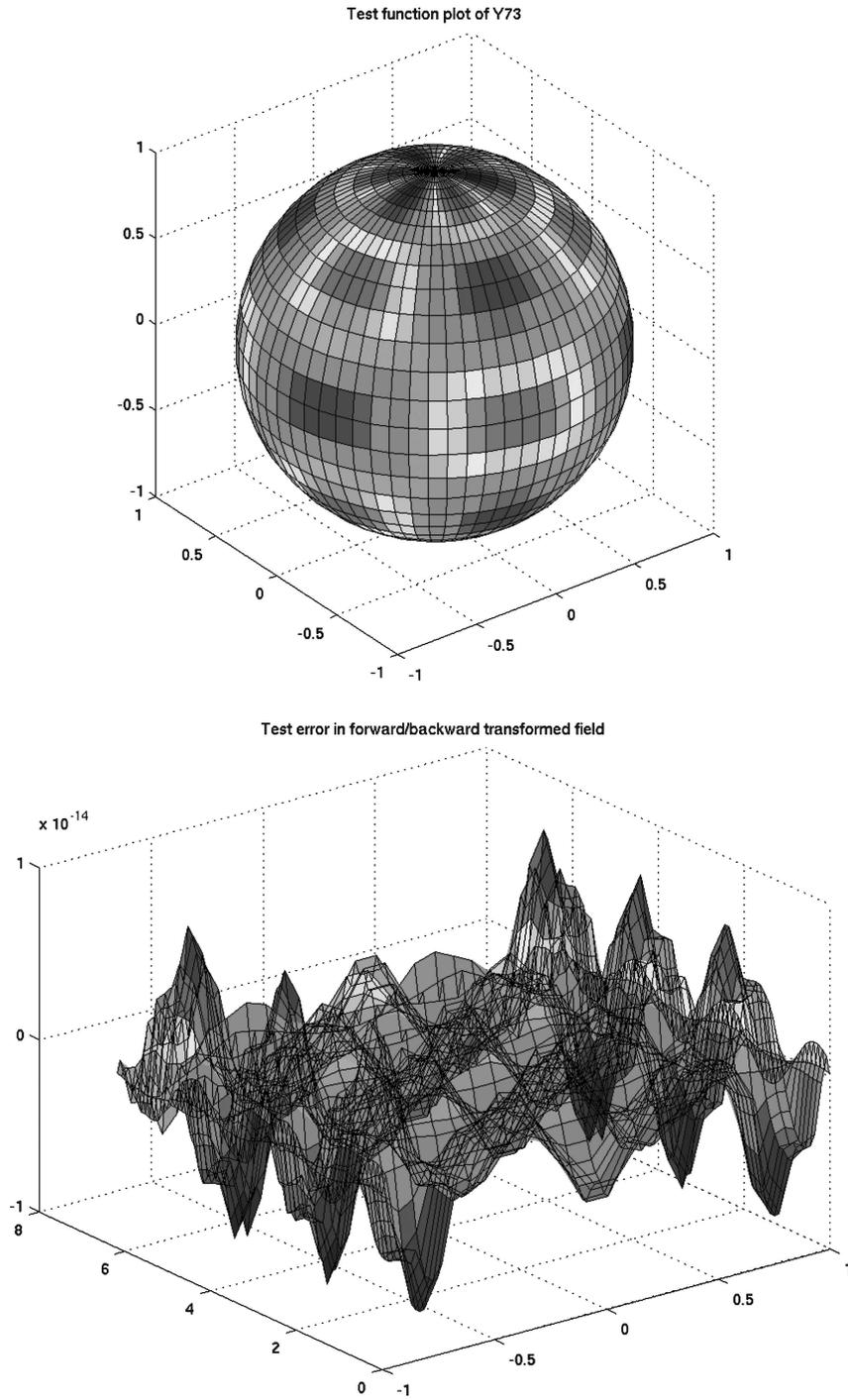


Fig. 1. MATLAB output from *test* showing the spherical harmonic function Y_7^3 and the error, a forward and backward transform.

MATLAB will use optimized versions of BLAS if certain environment variables are set when MATLAB is invoked. To take maximum advantage of the matrix formulation, an alias may be useful to set these environment variables. For example,

```
alias matlab='export BLAS_VERSION=atlas_P4.so;
export LD_ASSUME_KERNEL=2.3.98;
/usr/share/linux.x86/Matlab6/bin/matlab'
```

Vectorized versions for MATLAB of the Legendre open loop formulations are included in the software.

To illustrate the coding style, Algorithm 3.1 and Algorithm 3.2 are programs for the spherical harmonic analysis and synthesis as implemented in MATLAB. Algorithms 3.3 and 3.4 are examples of the private methods of the spherical harmonic field class that implement the open loop formulation of the Legendre transform.

ALGORITHM 3.1. *Spectral Analysis*

```
function f = shtrana(f)
% SHTRANA Spectral analysis of a spectral_grid field
% Compute the spectral coefficients from the field grid
% point values using a spherical harmonic transform analysis
%-----
% Input:
% f - spectral_field class object
% In particular:
% f.gp - field grid point values
% f.G - field gauss_grid
% Method of spectral_field class: Sept 2005
%-----
    ni = get(f.G,'ni'); nj = get(f.G,'nj');
    mm = get(f.G,'mm'); nn = get(f.G,'nn'); kk = get(f.G,'kk');
    wg = get(f.G,'wg'); P = get(f.G,'P'); gp = get(f,'gp');
%-----
    xf = fft(gp,ni)/ni; %note normalization for MATLAB FFT
%order ni= 2n real transform coming out as r0,r1,r2...i2,i1
%multiply Fourier coefficients by the Gauss weights
    for j=1:nj
        xf(:,j)=wg(j)*xf(:,j);
    end
    f.sc = legtranOLa(xf,nj,mm,nn,kk,P); % inverse Legendre
```

ALGORITHM 3.2. *Spectral Synthesis*

```

function f = shtrans(f)
% SHTRANS Spectral synthesis to a spectral_grid field
% Compute the grid point values from the spectral coefficients using
% a spherical harmonic transform synthesis
%-----
% Input:
% f - spectral_field class object
% In particular:
% f.gp - field grid point values
% f.G - field gauss_grid
% Output:
% f.sc - Output array of complex spectral coefficients (n,m)
% Local
% xf - matrix of Fourier coefficients ordered (m,j)
% Method of spectral_field class: Sept 2005
%-----
    ni = get(f.G,'ni'); nj = get(f.G,'nj');
    mm = get(f.G,'mm'); nn = get(f.G,'nn'); kk = get(f.G,'kk');
    wg = get(f.G,'wg'); P = get(f.G,'P');
%-----
    xf = legtranOLs(f.sc,nj,mm,nn,kk,P); % inverse Legendre transform
%make xf into a Hermitian array for real transform back
    for m=1:ni/2
        xf(ni-m+1,:) = conj(xf(m+1,:));
    end
    f.gp =real(ifft(xf,ni))*ni; % inverse Fourier transform, note normalization

```

ALGORITHM 3.3. *Legendre Transform Analysis - Open Loop*

```

function x = legtranOLa(s,nj,mm,nn,kk,P)
% Compute a Legendre transform analysis
% Input:
% s - complex Fourier coefficients ordered (m,j)
% nj = number of Gauss latitudes
% mm, nn,kk are the truncation parameters
% P - associated Legendre functions ordered (j,n,m)
% Output:
% x - matrix of spectral coefficients ordered (n,m)
%-----
% Based on Spherical harmonic transform formulation with open loops (OL)
%-----
    njo2=nj/2;
    x = zeros(nn+1,mm+1);
    for m=0:mm
        j=1:njo2;
        for n=m:2:nn
            x(n+1,m+1) = x(n+1,m+1)
                + (s(m+1,j) + s(m+1,nj-j+1))*P(j,n+1,m+1);
        end
        for n=m+1:2:nn
            x(n+1,m+1) = x(n+1,m+1)
                + (s(m+1,j) - s(m+1,nj-j+1))*P(j,n+1,m+1);
        end
    end
end

```

```

ALGORITHM 3.4. Legendre Transform Synthesis - Open Loop
function s = legtranOLs(x,nj,mm,nn,kk,P)
% Compute a Legendre transform synthesis – OPEN LOOP formulation
% Input:
% x - complex spectral coefficients (n,m)
% nj = number of Gauss latitudes
% mm, nn,kk are the truncation parameters
% P - associated Legendre functions ordered (j,n,m)
% Output:
% s - matrix of Fourier coefficients ordered (m,j)
%-----
% Based on Spherical harmonic transform formulation as an open loop
%-----
    nj02=nj/2;
    ni=2*nj;
%
    s=zeros(mm+1,nj);
%
    for m=0:mm
        for j=1:nj02
            for n = m:nn
                s(m+1,j) = s(m+1,j) + x(n+1,m+1)*P(j,n+1,m+1); %first half
            end
            for n=m:2:nn
                s(m+1,nj-j+1) = s(m+1,nj-j+1)
                    + x(n+1,m+1)*P(j,n+1,m+1); %second even
            end
            for n=m+1:2:nn
                s(m+1,nj-j+1) = s(m+1,nj-j+1)
                    - x(n+1,m+1)*P(j,n+1,m+1); %second odd
            end
        end
    end
end

```

4. BAROTROPIC VORTICITY EQUATION: MATLAB EXAMPLE

To demonstrate the use of the spectral transform in dynamics equations for the atmosphere, the simplest setting is a barotropic vorticity equation. The velocity is related to a horizontal stream function by $\mathbf{v} = \mathbf{k} \times \nabla \psi$. This stream function is calculated by inverting the elliptic equation:

$$\nabla^2 \psi = \xi, \quad (26)$$

where ξ is the (absolute) horizontal vorticity, $\xi = \mathbf{k} \cdot \nabla \times \mathbf{v}$. The governing equation of motion is given in terms of the potential vorticity:

$$\frac{d\eta}{dt} = 0. \quad (27)$$

The potential vorticity is related to the absolute vorticity by $\eta = \xi + f$, where $f = 2\Omega \sin\theta$ is the Coriolis term. The potential vorticity equation is written in advective form with the material derivative:

$$\frac{d\eta}{dt} \equiv \frac{\partial\eta}{\partial t} + \mathbf{v} \cdot \nabla\eta. \quad (28)$$

The time integration algorithm directly treats the material derivative operator using a semi-Lagrangian transport scheme. Particle tracking to determine departure points uses the MATLAB `@slt_grid` class, which extends the Gaussian grid with extra halo points for interpolation methods.

The barotropic vorticity solution algorithm also follows the ESMF style [Collins et al. 2005] specification of a gridded component model with *begin*, *run*, *finalize* steps. In brief, the solution algorithm is as follows:

ALGORITHM 4.1. *Barotropic Vorticity Integration*

Begin method

- integration control initialization
- gauss_grid initialization
- slt_grid initialization
- spectral_field initialization (prognostic initial conditions)
- spectral_field initialization (diagnostic initial conditions)

Run method (loop until solution end time)

- slt_particle tracking and departure point calculation
- slt_interpolation of η at departure points
- slt_interpolation of right hand side at departure points
- update η to new time level
- UVinv invert diagnostic relation to get new velocity \mathbf{v}
- del2inv invert Laplace operator to get stream function ψ

Finalize method

The example MATLAB program *BV.m* integrates the barotropic vorticity equation for three days starting from an initial Rossby-Haurwitz wave number four. The plots in Figures 2, 3,4 show the final solution after three days for potential vorticity, stream function and velocities using a T10 spectral truncation for the *gauss_grid*. The correct symmetries are apparent though a higher resolution solution would better display the persistence of the wave number four in the solution.

5. ALTERNATIVE ALGORITHMS

A number of papers propose alternative algorithms for the spherical harmonic transform. In this section we will briefly survey these in an effort to gauge their appropriate use for high resolution climate and weather modeling. The possibility of replacing the transform kernel with a lower operation count

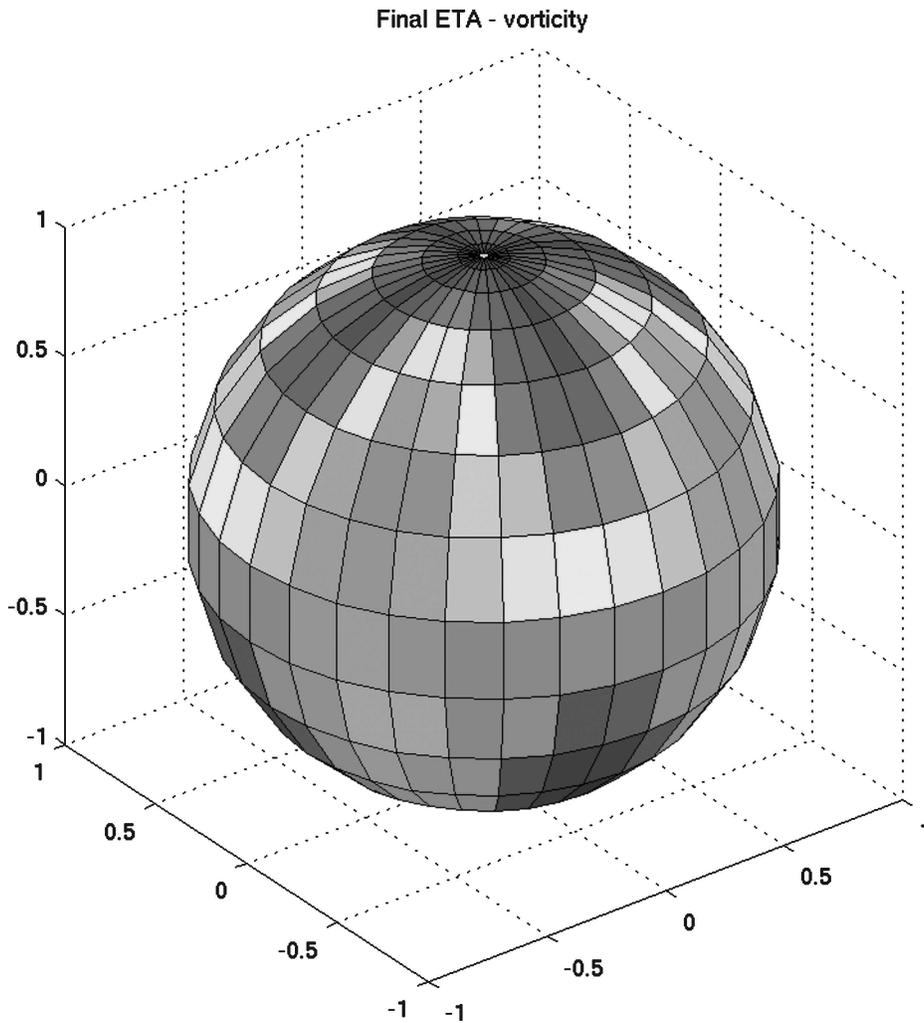


Fig. 2. MATLAB output from *BV.m* barotropic vorticity example showing potential vorticity.

algorithm, or of finding a more efficient method of computation is what motivates the survey. Our conclusion is that new formulations of the larger dynamic problem and other means of evaluating derivatives for the partial differential equations do indeed offer more promising algorithms. These new algorithms represent the progress in the numerical analysis of spectral methods that has occurred over the last decade but has not yet found its way into productive use.

5.1 The Fast Spherical Harmonic Algorithms

Driscoll and Healy [1989] introduced the first exact fast spherical harmonic transform. Subsequent refinements [Healy et al. 2003; Inda et al. 2001;

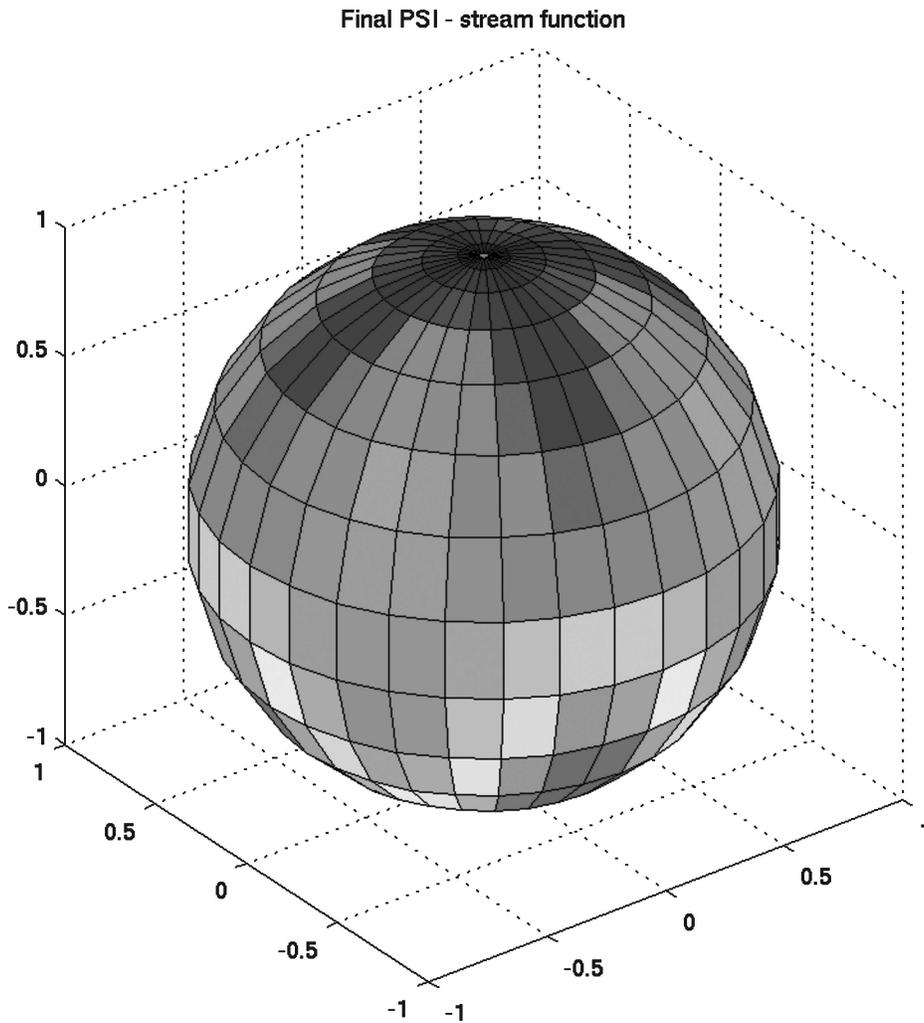


Fig. 3. MATLAB output from *BV.m* barotropic vorticity example showing stream function.

Mohlenkamp 1999] have resulted in parallel versions with reasonable accuracy, stability, and performance. The asymptotic operation count for the fast algorithm is $M^2(\log M)^2$. A crossover point for performance in comparison with the direct method occurs at $M=128$ with a savings by a factor of three at $M=512$. For $M=1279$ and other high resolution cases, the operation count of the transform can be reduced dramatically.

An approximate fast transform has been developed using the ideas in Boyd [1992]. The new algorithm [Suda and Takami 2002] is based on fast polynomial interpolation accelerated by the Fast Multipole Method. The asymptotic operation count is proportional to $M^2 \log M$, an improvement on the Driscoll and Healy algorithm. The crossover point with the direct method is observed

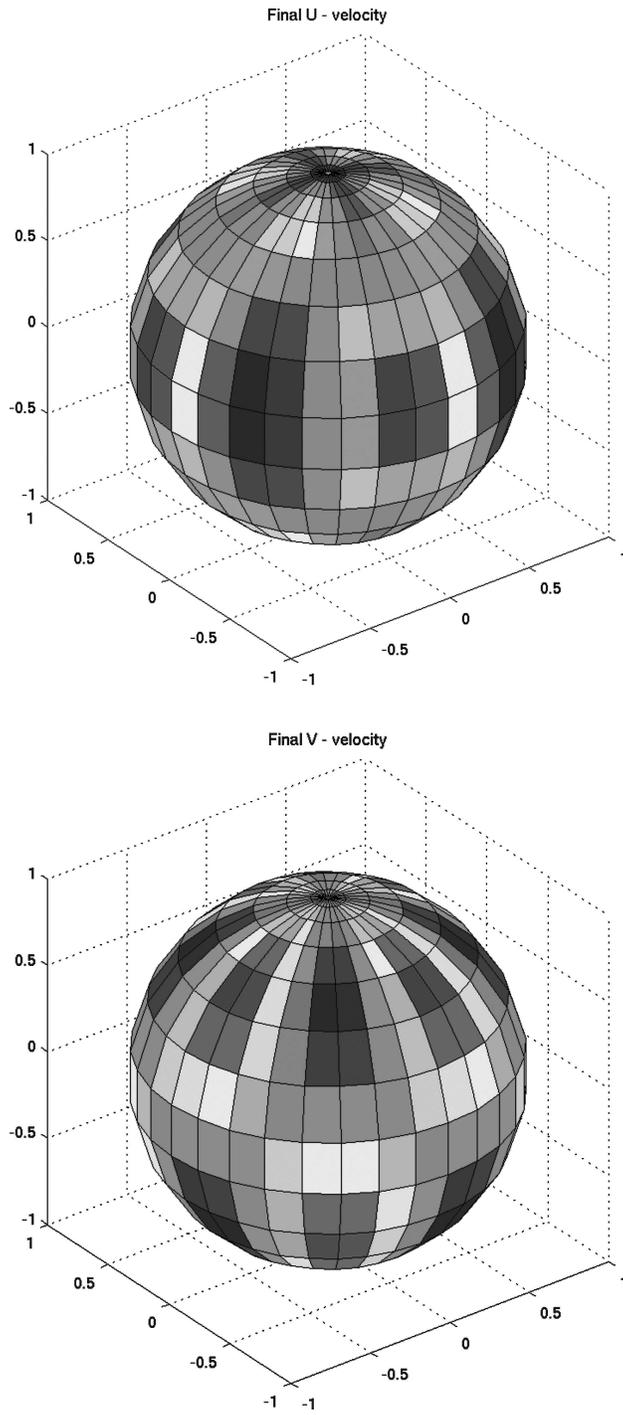


Fig. 4. MATLAB output from *BV.m* barotropic vorticity example showing velocity field.

at $M=512$. A speedup factor of 1.8 is observed at $M=1365$. Another approximate method with an $O(M^2 \log M)$ operation count is proposed in Rokhlin and Tygert [2006].

5.2 FFT2D Derivative Evaluations

The main use of the spectral method is in the highly accurate and efficient representation of differential terms in numerical approximations to partial differential equations. For the flow equations of the atmosphere used in weather and climate modeling, the spherical harmonic spectral representation leads to diagonal forms for the Laplacian operator. The solution to a Helmholtz equation is the critical step in a semi-implicit time discretization that effectively filters gravity waves and stabilizes long time integrations. The spectral methods offer an important advantage for the solution of this Helmholtz equation over more standard grid point (finite difference, control volume, and finite element) methods.

An alternative way to evaluate the differential terms uses a 2-D (lon-lat) FFT where the latitude direction is taken on great circles. The paper Spatz et al. [1998] studies Merilee's pseudospectral model and finds a FFT-based algorithm that exactly matches results from a spherical harmonic transform model. The evaluation of derivatives can be accomplished with $I^2 \log I$ operations. The argument is made that, with the addition of a fast projection algorithm to ensure all modes remain in the spherical harmonic space, the FFT2D based model is the fastest spherical harmonic algorithm. This method was coupled with two and three level semi-Lagrangian time-stepping methods for a very efficient, high order solution algorithm [Layton and Spatz 2002]. The key new element that makes these algorithms attractive is the existence of a fast spherical harmonic projection that stabilizes the Fourier method.

5.3 Fast Spherical Harmonic Projection Algorithms

The projection of a function onto the spherical harmonic modes (analysis) followed by the inverse transformation back to grid point space is a filter of the original function. If there is no need to use the spectral coefficients to evaluate derivatives, then the forward and inverse transforms can be combined algebraically to produce an explicit projection operator. Several algorithms have been studied in Spatz and Swarztrauber [2001] and compared in single processor and parallel implementations.

An interesting method for evaluating the projection utilizes a fast multipole method. The multipole projection was proposed as an approximate, fast projection method. It is based on an application of the Christoffel-Darboux formula relating the sum of products of associated Legendre functions. Several variants have been developed including Holmes et al. [1996] and Yarvin and Rokhlin [1998]. The study in Spatz and Swarztrauber [2001] proposes several other algorithms and concludes that the Weighted Orthogonal Complement (WOC) algorithm developed in Swarztrauber and Spatz [2000] is the most efficient in terms of operation count, cache utilization and overall

performance for the resolutions studied. This algorithm for Legendre projection is faster than the multipole projection method in its known implementations to date.

5.4 Operation Counts and Parallel Algorithm Performance Model

A performance model of the parallel spectral transform can be developed to estimate the time for a multi-level calculation. The computational operation counts and communication cost estimates are based on a model in Foster and Worley [1997] for a one dimensional decomposition and modified by Rich Loft (NCAR) to reflect a simple transpose between FFT and Legendre transform phases including vertical levels. The time for the FFT, the Legendre transform and the communication overhead are estimated using machine-dependant rate constants a, b, d , and e .

<p>Time for FFT = $5a(6L + 1)IJ\log_2(I)$ Time for LT = $2b(6L + 1)JM^2$ Time in COMM = $dP + 2e(6L + 1)J(2M + 1)$ Nomenclature: <i>M</i> wave number resolution, eg. TM <i>I</i> number of longitudes ($I \geq 3M + 1$) <i>J</i> number of latitudes ($J = I/2$) <i>L</i> number of vertical levels <i>P</i> number of nodes (computational unit doing FFT or LT) <i>a</i> computational rate of FFT in flops/node <i>b</i> computational rate for LT in flops/node <i>d</i> latency factor <i>e</i> bandwidth factor</p>

Using this model with estimates of network bandwidth and the speed of a node in computing FFTs and Legendre transforms, we can project the overall, sustained computational rate of a computer for performing spherical harmonic transforms. Refinement of the performance model requires an experimental determination of the a, b, d , and e parameters. This can be done with kernel tests or by fitting performance data. Figure 5 shows the system balance required to sustain a petaflop performance on the spherical harmonic transform. The balance of latency (nanoseconds), bandwidth (GigaBytes per second) and computational performance (GigaFlops/sec) on the Legendre transform and FFTs, required by three systems is illustrated using spherical harmonic transforms for an atmospheric model with 96 levels at the T1279 resolution using 1920 processing nodes. In this figure, what is striking is that the balance is so sensitive to latency and bandwidth of the nodal interconnect. Improving the latency, from the yellow machine with a latency of 1 microsecond, to the blue machine with a latency of 0.5 microseconds, significantly reduces the need for computational speed and bandwidth. If bandwidth cannot be obtained above 500 GB/s (the purple machine), then the computational speed of the nodes must be 1 Tflop in order to achieve a sustained petaflop.

Petaflop Balance for SHTrans

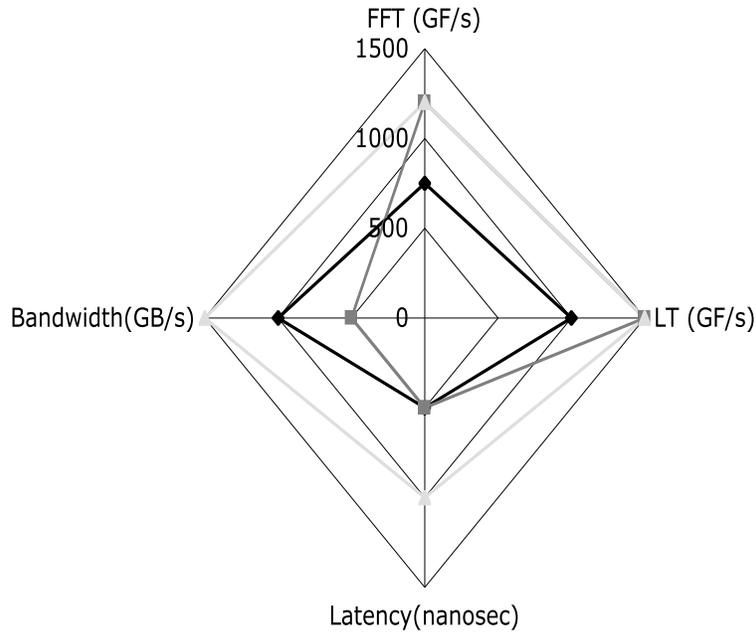


Fig. 5. System balance required for a sustained petaflop calculating T1279 transforms on 1920 processing nodes. Each closed figure represents a machine configuration that would sustain a petaflop on the parallel spherical harmonic transform. The low bandwidth of the purple machine requires higher FFT and LT performance per node than the blue machine. Similarly, the high latency of the yellow machine requires higher bandwidth than the purple or blue machines.

6. CONCLUSIONS

A set of algorithms and MATLAB classes for computing spherical harmonic transforms was described. This set of classes enables a number of algorithmic studies and computational experiments important for the development of weather and climate models on high performance computers. The typical use of the spectral transform is illustrated using the differential operator methods of these classes to solve the barotropic vorticity equation.

ACKNOWLEDGMENTS

The authors wish to thank the DOE Scientific Discovery through Advanced Computing (SciDAC) program and the DOE Climate Change Prediction Program (CCPP) for continuing interest in the relationship of mathematics and computation to the sciences. The authors also wish to thank the anonymous reviewers whose comments added to the organization and clarity of the presentation.

REFERENCES

- ADAMS, J. AND SWARZTRAUBER, P. N. 1999. Spherepack3.0: A model development facility. *Mon. Wea. Rev.* 127, 1872–1878.
- ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK Users' Guide*, Third ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.
- BOYD, J. 1992. Multipole expansions and pseudospectral cardinal functions: a new generalization of the fast Fourier transform. *J. Comp. Phys.* 103, 1, 184–186.
- CANUTO, C., HUSSAINI, M., QUARTERONI, A., AND ZANG, T. 1991. *Spectral Methods in Fluid Dynamics*, 3rd ed. Springer-Verlag, New York.
- COLLINS, N., THEURICH, G., DELUCA, C., SUAREZ, M., TRAYANOV, A., BALAJI, V., LI, P., YANG, W., HILL, C., AND DA SILVA, A. 2005. Design and implementation of components in the Earth System Modeling Framework. *Int. J. High Perf. Comput. Appl.* 19, 3, 341–350.
- D'AZEVEDO, E. 2004. Performance of the spherical harmonic transform on modern architectures. In *Proceedings of the SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, Philadelphia, PA.
- DONGARRA, J. 2007. Performance of various computers using standard linear equations software. Tech. Rep. CS - 89, University of Tennessee, Knoxville, TN.
- DRAKE, J., JONES, P., AND CARR, G. 2005. Overview of the software design and parallel algorithms of the CCSM. *Int. J. High Perf. Comput. Appl.* 19, 3, 177–186.
- DRISCOLL, J. AND HEALY, JR., D. 1989. Computing Fourier transforms and convolutions on the 2-sphere. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*. IEEE, Los Alamitos, CA, 344–349.
- FOSTER, I. AND WORLEY, P. 1997. Parallel algorithms for the spectral transform method. *SIAM J. Sci. Stat. Comput.* 18, 3, 806–837.
- HEALY, JR., D., ROCKMORE, D., KOSTELEK, P., AND MOORE, S. 2003. An FFT for the 2-sphere: improvements and variations. *J. Fourier Anal. Appl.* 9, 4, 341–385.
- HOLMES, J., WANG, Z., DRAKE, J., LYON, B., AND CHEN, W.-T. 1996. A fast multipole transformation for global climate calculations. Tech. Rep. ORNL TM-13135, Oak Ridge National Laboratory.
- INDA, M., BISSELING, R., AND MASLEN, D. 2001. On the efficient parallel computation of Legendre transforms. *SIAM J. Sci. Comput.* 23, 1, 271–303.
- KELBERT, A. 2007. Shtools - spherical harmonics toolbox. Download from Matlab Central File Exchange. available at <http://www.mathworks.com/matlabcentral/fileexchange>.
- LAYTON, A. T. AND SPOTZ, W. F. 2002. A semi-Lagrangian double Fourier method for the shallow water equations on the sphere. *J. Comp. Phys.* 189, 1, 180–196.
- MOHLENKAMP, M. 1999. A fast transform for spherical harmonics. *J. Fourier Anal. Appl.* 5(2/3), 159–184.
- ROKHLIN, V. AND TYGERT, M. 2006. Fast algorithms for spherical harmonic expansions. *SIAM J. Sci. Stat. Comput.* 27, 6, 1903–1928.
- SHINGU, S., TAKAHARA, H., FUCHIGAMI, H., YAMADA, M., TSUDA, Y., OHFUCHI, W., SASAKI, Y., KOBAYASHI, K., HAGIWARA, T., HABATA, S., YOKOKAWA, M., ITOH, H., AND OTSUKA, K. 2002. A 26.58 terflop global atmospheric simulation with the spectral transform method on the Earth Simulator. In *Proceedings of the 2002 ACM/IEEE Supercomputing Conference*. ACM/IEEE, Baltimore, Maryland, 1–19.
- SIMONS, F. J., DAHLEN, F., AND WIECZOREK, M. 2006. Spatiospectral localization on a sphere. *SIAM Rev.* 48, 2, 504–536.
- SPOTZ, W. F. AND SWARZTRAUBER, P. 2001. A performance comparison of associated Legendre projections. *J. Comp. Phys.* 168, 339–355.
- SPOTZ, W. F., TAYLOR, M. A., AND SWARZTRAUBER, P. 1998. Fast shallow-water equation solvers in latitude-longitude coordinates. *J. Comp. Phys.* 145, 432–444.
- SUDA, R. AND TAKAMI, M. 2002. A fast spherical harmonic transform algorithm. *Math. Comp.* 7, 703–715.

- SWARZTRAUBER, P. N. AND SPOTZ, W. F. 2000. Generalized discrete spherical harmonic transforms. *J. Comp. Phys.* 145, 213–230.
- TEMPERTON, C. 1983. Fast mixed-radix real Fourier transform. *J. Comp. Phys.* 52, 340–350.
- TREFETHEN, L. N. 2000. *Spectral Methods in MATLAB*. SIAM Books, Philadelphia, PA.
- WIECZOREK, M. 2007. Shtools: Tools for working with spherical harmonics. <http://www.ipgp.jussieu.fr/~wieczor/SHTOOLS>. available at <http://www.ipgp.jussieu.fr/~wieczor/SHTOOLS>.
- WORLEY, P. H. AND DRAKE, J. 2005. Software design for performance portability in the Community Atmosphere Model. *Int. J. High Perf. Comput. Appl.* 19, 3, 187–201.
- YARVIN, N. AND ROKHLIN, V. 1998. A generalized one-dimensional fast multipole method with applications to filtering of spherical harmonics. *J. Comp. Phys.* 159, 2, 594–609.

Received October 2006; revised December 2007; accepted April 2008