A Fast Spherical Harmonics Transform for Global NWP and Climate Models

NILS P. WEDI, MATS HAMRUD, AND GEORGE MOZDZYNSKI

European Centre for Medium-Range Weather Forecasts, Reading, United Kingdom

(Manuscript received 11 January 2013, in final form 16 March 2013)

ABSTRACT

Very high-resolution spectral transform models are believed to become prohibitively expensive because of the relative increase in computational cost of the Legendre transforms compared to the gridpoint computations. This article describes the implementation of a practical fast spherical harmonics transform into the Integrated Forecast System (IFS) at ECMWF. Details of the accuracy of the computations, of the parallelization, and memory use are discussed. Results are presented that demonstrate the cost effectiveness and accuracy of the fast spherical harmonics transform, successfully mitigating the concern about the disproportionally growing computational cost. Using the new transforms, the first T7999 global weather forecast (equivalent to ~2.5-km horizontal grid size) using a spectral transform model has been produced.

1. Introduction

Numerical weather prediction (NWP) requires an answer in real time with a window of approximately 1 h to run a medium-range global forecast that can be delivered in time to its customers. While computational efficiency remains one of the most pressing needs of NWP, there is an open question about how to make the most efficient use of the affordable computer power that will be available over the next decades, while seeking the most accurate forecast possible.

The spectral transform method has been successfully applied at the European Centre for Medium-Range Weather Forecasts (ECMWF) for approximately 30 years, with the first spectral model introduced into operations at ECMWF in April 1983. Spectral transforms on the sphere involve discrete spherical harmonics transformations between physical (gridpoint) space and spectral (spherical harmonics) space. The spectral transform method was introduced to NWP following the work of Eliasen et al. (1970) and Orszag (1970), who pioneered the efficiency obtained by partitioning the computations. One part of the computations is performed in physical space, where products of terms, the semi-Lagrangian or Eulerian advection, and the physical parameterizations are computed. The other part is solved in spectral space, where the Helmholtz equation arising from the semi-implicit time-stepping scheme can be solved easily and horizontal gradients on the (reduced) Gaussian grid are computed accurately, particularly the Laplacian operator that is so fundamental to the propagation of atmospheric waves. The success of the spectral transform method in NWP in comparison to alternative methods has been overwhelming, with many operational forecast centers having made the spectral transform their method of choice, as comprehensively reviewed in Williamson (2007).

A spherical harmonics transform is a Fourier transformation in longitude and a Legendre transformation in latitude, thus keeping a latitude–longitude structure in gridpoint space. The Fourier transform part of a spherical harmonics transform is computed numerically very efficiently by using the fast Fourier transform (FFT; Cooley and Tukey 1965), which reduces the computational complexity to $O(N^2 \log N)$, where $N$ symbolizes the cutoff spectral truncation wavenumber. However, the conventional Legendre transform has a computational complexity $\propto O(N^3)$, and with increasing horizontal resolution the Legendre transform will eventually become the most expensive part of the computations in terms of the number of floating-point operations and subsequently the elapsed (wall clock) time required. Because of the relative cost increase of the Legendre transforms compared to the gridpoint computations, very high-resolution spectral models are believed to become prohibitively expensive, and methods based on finite elements or finite volumes on alternative quasi-uniform
This article is organized as follows: section 2 describes the parallel spherical harmonics transform as implemented in ECMWF’s IFS transform package including the FLT acceleration. Section 3 discusses the computational performance of the FLT algorithm and shows the computational cost at different horizontal resolutions for actual NWP forecasts. Finally, section 4 discusses the use of the spectral transform method on future massively parallel computer architectures and draws some conclusions.

2. The fast spherical harmonics transform

a. The spherical harmonics transform

The truncated series expansion in spherical harmonics of a scalar variable $\zeta$ on a vertical level $l$ at or above the surface of a sphere is given by

$$\zeta_l(\theta, \lambda) = \sum_{m=-M}^{M} \sum_{n=|m|}^{N(m)} \zeta_{m,l}^{n} \overline{T_n^m}[\cos(\theta)],$$

(1)

where $\theta$ and $\lambda$ denote colatitude ($\phi = 90 - \theta$ is the geographical latitude with $\theta$ at the equator) and longitude, respectively; $\zeta_{m,l}^{n}$ are the spectral coefficients of the field $\zeta$ at level $l$; and the $T_n^m$ are the normalized (denoted by the overbar) associated Legendre polynomials of degree $n$ and order $m$ as a function of latitude only. The normalization is defined by

$$\frac{1}{2} \int_{-1}^{1} \overline{T_n^m}[\cos(\theta)]^2 d\cos(\theta) = 1.$$  

(2)

The indices $M$ and $N(m)$ symbolize the cutoff spectral truncation wavenumber in the spherical harmonics expansion and the choice of $N(m)$ specifies the truncation type. In the IFS the triangular truncation $N(m) = M$ is chosen, favoring a quasi-isotropic distribution (cf. Naughton et al. 1996). Equation (1) represents the discrete inverse spherical harmonics transform that resynthesizes the gridpoint representation from the spectral coefficients.

A direct spherical harmonics transformation is accomplished by a Fourier transformation in longitude as

$$\zeta_{m,l}(\lambda) = \frac{1}{2\pi} \int_{0}^{2\pi} \zeta_l(\theta, \lambda) e^{-im\lambda} d\lambda$$

(3)

and a Legendre transformation in latitude for each $m$ as

$$\zeta_{m,l}^{n} = \frac{1}{2} \int_{-1}^{1} \zeta_{m,l}^{n} \overline{T_n^m}[\cos(\theta)] d\cos(\theta).$$

(4)

The Fourier transform is computed using the FFT algorithm detailed in Temperton (1983). The Legendre transforms require the accurate discrete computation of the integral in (4), which is accomplished by Gaussian quadrature.
\[ \xi^{mj}_{nj} = \sum_{k=1}^{K} w_k \xi^{mj}(x_k) P^m_n(x_k) \]  

(5)

at the \( K = (2N + 1)/2 \) (linear grid) special quadrature points ("Gaussian latitudes") given by the roots of the ordinary Legendre polynomials \( P^m_n(x_k) = 0 \) and the Gaussian weights computed by the following formula (Schwarztrauber 2002):

\[ w_k = \frac{2N + 1}{[P^N_{n-1}(x_k)]^2} . \]  

(6)

b. Precomputations

For ultrahigh horizontal resolution (\( \approx N = 1000 \)) the cost and the accuracy of the precomputations also become important. In the IFS, the associated Legendre polynomials \( P^m_n \) evaluated at the roots of the ordinary Legendre polynomials \( P^m_n=0 \) are precomputed and stored, distributed in parallel space by wavenumber \( m \). Because of this parallel distribution, memory savings resulting from the FLT algorithm (Tygert 2008) are modest in our multiprocessor simulations (\( =10\% \) per compute node at T7999). However, at T7999 and higher resolutions this aspect may become significant on future computing architectures. When applying spectral transforms in post-processing and plotting applications (typically done on single processors) the memory requirements for \( N \approx O(1000) \) become a serious issue and the FLT algorithm will provide substantial benefits.

The recurrence formula, Eq. (17) in Belousov (1962),

\[ \bar{P}^m_n(x_k) = C^m_n \bar{P}^{m-2}_n(x_k) - D^m_n x_k \bar{P}^{m-2}_n(x_k) + E^m_n x_k^2 \bar{P}^{m-1}_n(x_k) \]  

(7)

with

\[ C^m_n = \sqrt{\frac{(2n+1)(m+n-1)(m+n-2)}{(2n-3)(m+n)(m+n-2)}} \]

\[ D^m_n = \sqrt{\frac{(2n+1)(m+n-1)(m+n-1)}{(2n-1)(m+n)(m+n-2)}} \]

\[ E^m_n = \sqrt{\frac{(2n+1)(n-m)}{(2n-1)(n+m)}} \]  

(8)

is used for the computation of the associated Legendre polynomials \( \bar{P}^m_n \) at the roots of the (ordinary) Legendre polynomials \( P^m_n=0 \). The Belousov recurrence relation is stable and accurate even at very large \( N \) provided that the initial values for the recurrence are computed sufficiently accurately. The accuracy for large \( N \) can be improved by computing the polynomials in extended (quadrupole) precision, but typically this bears a high computational penalty. Alternatively, Schwarztrauber (2002) suggested a Fourier representation of the ordinary Legendre polynomials together with simple formulas to evaluate the corresponding coefficients. The resulting \( \bar{P}^{m=0} \) and \( \bar{P}^{m=2} \) are then used for finding the roots of the ordinary polynomials via Newton iterations (to machine precision, typically with very few iterations) and they are used again for the initial values of the Belousov recurrence formula. Such a procedure has been implemented in the IFS transform library and provides improved accuracy at large \( N \) while maintaining all computations in double precision as opposed to extended precision.

However, the cost of the computation on each processor (responsible only for a subset of zonal wave numbers \( m \)) is still unnecessarily high because the Belousov recurrence formula involves varying degrees and orders of the "intermediate" associated polynomials, whereas ideally one would like a formula that only involves a single \( m \). While there are such formulas readily available in standard references (e.g., Abramowitz and Stegun 1972), these are numerically unstable beyond \( N \approx 20 \) (Belousov 1962). For all \( m \geq 2 \) we use the Eq. (42) in Tygert (2008) \(^1\) as

\[ \bar{P}^{m+2}_n(x_k) = (\epsilon^m_n)^{-1} \left( (x_k^2 - d^m_n \bar{P}^m_n(x_k) - c^m_n \bar{P}^{m-2}_n(x_k) \right) \]  

(9)

with

\[ c^m_n = \sqrt{\frac{(n-m+1)(n-m+2)(n+m+1)(n+m+2)}{(2n+1)(2n+3)(2n+5)}} \]

\[ d^m_n = \frac{2(n+1) - 2m^2 - 1}{(2n-1)(2n+3)} . \]  

(10)

Equation (9) is also unstable but can be made stable by tracking the values of the numbers and keeping them within an acceptable range that can be represented with double precision. Moreover, we have found that accurate starting values—\( \bar{P}^m, \bar{P}^{m+1}, \bar{P}^{m+2}, \) and \( \bar{P}^{m+3} \)—are crucial to achieve comparable accuracy with the Belousov algorithm. The formulas used for these initial values are derived from the (normalized) definition:

\[ \bar{P}^m_i(x) = \sqrt{2l + 1} \frac{(l-m)!}{(l+m)!} (1-x^2)^{m/2} \frac{d^m}{dx^m} P_l(x) . \]  

(11)

\(^1\) See also p. 100, Eq. (47) in Robin (1957), and normalize the polynomials according to (11).
Sufficient accuracy is obtained by using the closed form formulas (Egersdörfer and Egersdörfer 1936) for the \( m \)th derivative of the required ordinary Legendre polynomials \( P_l \) with \( l = m, m + 1, m + 2, m + 3 \) as

\[
\frac{d^m}{dx^m} P_m(x) = (2m-1)!!,
\]

\[
\frac{d^m}{dx^m} P_{m+1}(x) = (2m+1)!!x,
\]

\[
\frac{d^m}{dx^m} P_{m+2}(x) = \frac{(2m+1)!!}{2} [(2m+3)x^2 - 1],
\]

\[
\frac{d^m}{dx^m} P_{m+3}(x) = \frac{(2m+3)!!}{6} x[(2m+5)x^2 - 3],
\]

where \( (2m-1)!! = 1 \times 3 \times 5 \times \cdots \times (2m-3) \times (2m-1) \).

Furthermore, using the formulas for a single \( m \) halves the cost by requiring only even (odd) values depending on the symmetric (antisymmetric) part. The IFS model exploits symmetry properties and the computations are split into symmetric and antisymmetric components such that a field is given by a field split into symmetric and antisymmetric components.

The right-hand side represents the matrix of the associated Legendre polynomial coefficients \( \mathbf{P}_m(x_k) \) at given \( m \) for all total wavenumbers \( n \) and at all Gaussian latitudes, symbolically denoted by \( \mathbf{S} \), and multiplied with the spherical harmonics coefficient matrix for a given \( m \) at all levels and for all total wavenumbers \( n \). Notably, (13) exposes the parallelism of the problem, since all \( m, l \) are independent of each other and may be suitably distributed over the processors. Following Tygert (2010) the FLT algorithm is based on the fundamental idea that for a given zonal wavenumber \( m \) the matrix \( \mathbf{S} \) can be subdivided into smaller parts by doubling the columns and halving the rows with each level of subdivision and storing these level by level (of subdivision) in a hierarchical tree structure. This so-called butterfly algorithm is described in detail in O’Neil et al. (2010). The essence of the algorithm is that each of the new submatrices obtained in this way may be compressed using an ID (Cheng et al. 2005) (or, in principle, any other algorithm for rank reduction), such that

\[
|\mathbf{S}_r \times s - \mathbf{C}_{r \times k} \mathbf{A}_{k \times s}| \leq \epsilon,
\]

where matrix \( \mathbf{C}_{r \times k} \) constitutes a subset of the columns of submatrix \( \mathbf{S}_r \times s \) and where matrix \( \mathbf{A}_{k \times s} \) contains a \( k \times k \) identity matrix with \( k \) being called the \( \epsilon \) rank of submatrix \( \mathbf{S}_r \times s \) (Cheng et al. 2005; Martinsson and Rokhlin 2007). An important point to make is that the application of the ID compression directly on the full original matrix \( \mathbf{S} \) has no effect. Only when rank-deficient submatrices can be constructed by subdivision and by regrouping of columns and rows, will the algorithm save time and floating-point operations. For NWP applications the
advantage of the algorithm clearly lies in the fact that one can precompute and store the compressed images of the submatrices. Moreover, in the application stage one merely needs to apply the compressed representation, resulting in far fewer summations that need to be executed at every time step of the model simulation, provided that the rank reduction is effective. The cost increase involved in the additional precomputation step is negligible, with typically less than 0.1% of the total elapsed time of a 10-day forecast simulation.

In Tygert (2008) the FLT has been shown to have an asymptotic cost $O(N^2 \log N)$. However, as pointed out in Tygert (2010), there is no such formal proof for the butterfly algorithm. For wavemoth (Seljebotn 2012), an implementation of the butterfly algorithm for astrophysical applications, an asymptotic scaling of $\propto O(N^2 \log^2 N)$ is suggested. Our results confirm a substantial reduction of the cost in actual NWP simulations. While the horizontal truncations in our simulations are at the lower end of the projections in Tygert (2008), they are very high in the context of NWP and climate, and we find up to resolutions of T7999 that the spectral transforms scale according to $O(N^2 \log^3 N)$.

The “butterfly method” is described in Tygert (2010) and further illustrated in Seljebotn (2012). The subdivisions into smaller parts by doubling the columns and halving the rows with each level is illustrated in Fig. 1 (see also O’Neil et al. 2010). The main purpose of the subdivisions is to create rank-deficient blocks that can be compressed. At level 0 the submatrices are compressed and the residual is stored. At the next level in the tree the (compressed) left and right neighbors of the previous level form the submatrices of this level and are subsequently compressed. A schematic (pseudocode) illustration of the butterfly algorithm based on O’Neil et al. (2010) is given in the appendix. From our point of view, a heuristic explanation why the butterfly algorithm works well for applications on the sphere is due to the structure of spherical harmonics functions. Figure 2 illustrates two important aspects: first, the amplitude of the spherical harmonics for large $m$ is negligibly small toward the poles at many Gaussian latitudes—rows of the matrix $S$ to be subdivided—and the number of latitudes where the amplitudes are negligibly small is increasing with increasing truncation number $N$ (Fig. 2a).

![Fig. 1. Schematic butterfly algorithm with $2^L$ subdivisions at level 0 and $L = 3$ levels.](image1)

![Fig. 2. (a) Comparison of spherical harmonics functions at $n = 3999$ (solid) and $n = 3998$ (dashed) along a single meridian. (b) Zoom in of the near-equatorial region for one hemisphere.](image2)
Secondly, neighboring $n$—represented by neighboring columns of $S$ to be combined—are very similar (Fig. 2b). The former aspect is already used successfully in the application of a reduced grid, where the number of points (and Fourier modes) are reduced at each latitude toward the poles, requiring $\approx 30\%$ less grid points compared to an equivalent latitude–longitude grid and with the additional advantage of creating a quasi-uniform gridpoint distribution on the sphere, but without the penalty typically associated with reduced grids in other discretizations (e.g., finite volume; Jablonowski et al. 2009).

The butterfly FLT algorithm fits very well into the existing parallel transform package of the IFS. The only modification required in the precomputations is an additional step to setup and store the compressed symmetric and antisymmetric representations of the matrix $S$. The only other change is in the actual direct and inverse transform calls, where—depending on a switch that activates the FLT or not—the standard matrix–matrix multiplication $\text{dgemm}$ is replaced with a new routine that performs the matrix–matrix multiplication using the compressed representation ($\text{mult\_butm}$).

In the current implementation we do not make use of the “depth-first traversal” advocated in Tygert (2010), and due to the parallel distribution over the zonal wavenumbers $m$ and over the levels/fields, the memory usage per MPI task is sufficiently low, eliminating the need to recompute the coefficients every time step or otherwise change the programmatic flow. Nevertheless, the suggested optimizations in Seljebotn (2012) regarding the minimization of communications per time step through reorganization of the work flow may be explored in the future.

### 3. Results

There are essentially three tuning parameters that control the behavior of our implementation of the FLT algorithm. The first parameter is the number of columns $I_{\text{MAXCOLS}}$ in each submatrix at level 0 (cf. the appendix). The IFS simulations appear not to be very sensitive to this parameter so $I_{\text{MAXCOLS}} = 128$ is chosen for all horizontal resolutions greater than 159, to minimize wall-clock execution time. However, if the aim was to minimize the number of floating-point operations, $I_{\text{MAXCOLS}} = 64$ is found to be more optimal and we found that the choice can indeed be important on different computer architectures. The second parameter is a threshold value $I_{\text{thresh}}$. If (e.g., for the symmetric part) $(N - m + 2)/2 \approx I_{\text{thresh}}$, the standard matrix–matrix multiplication $\text{dgemm}$ is more efficient (in wall-clock time) and is applied instead of the FLT. The thresholds used are $(256, 512, 600, 900)$ for the resolutions $(799, 1279, 2047, \geq 3999)$, respectively. The corresponding butterfly precomputations are also omitted below the given threshold. The third parameter, which is the only parameter that introduces an approximation to the algorithm, is the accuracy required in the compression part of the algorithm. We find that with $\epsilon = 10^{-7}$ equivalent meteorological accuracy as measured in terms of hemispheric root-mean square error (rmse) and anomaly correlation of 500 geopotential height and other parameters (not shown)—typically used to verify technical model changes—is obtained. A stronger compression does reduce the computational cost further, but this does impact the meteorological forecast results negatively. To illustrate the meteorological impact of lossy compression of the ID algorithm (i.e., a loss of accuracy compared to not doing any compression), several T2047 10-day forecast simulations have been run using $\epsilon = 10^{-10}$, $\epsilon = 10^{-7}$, $\epsilon = 10^{-3}$, $\epsilon = 10^{-2}$ and a control simulation using $\text{dgemm}$. Figure 3 illustrates that with $\epsilon = 10^{-7}$ the accuracy (with respect to rmse and anomaly correlation of the 500-hPa geopotential height surface) is very close to the control, while still offering substantial computational savings at higher resolutions as discussed below. Figure 4 illustrates how the ID algorithm alters global prognostic variables by retaining the energetically dominant part (as determined by the lower wavenumbers of the spectrum) up to a selected accuracy. Notably, the ID is closely related to low-rank matrix approximations via singular value decomposition (SVD), where only the largest singular values are retained. In contrast to spectral truncation, the remainder (the smallest scales or largest wavenumbers in the spectrum) that may be viewed as less predictable, are perturbed but not set to zero. The effect of lossy compression in the transform part of the model can be identified in the noise visible in the resulting kinetic energy spectra for different values of $\epsilon$ (Fig. 4). Notably, the simulation with $\epsilon = 10^{-2}$ becomes numerically unstable after approximately 5 simulation days.

The simulations using the new FLTs are stable with $\epsilon \geq 10^{-3}$, and the FLT algorithm results below in actual global NWP simulations up to T7999 resolution have been obtained with $\epsilon = 10^{-7}$. Meteorological results from these ultrahigh-resolution simulations for NWP will be reported elsewhere. However, to illustrate the capabilities and the numerical stability of the simulations, Fig. 5a shows the global kinetic energy spectra of the IFS simulations using the FLTs at different resolutions averaged every 12 h over the first 5 forecast days. The T7999 spectrum obtained from the FLT simulation is an instantaneous spectrum after 6 h of simulation (the end of that simulation). The kinetic energy spectra of
equivalent runs using the standard matrix–matrix multiplication dgemm are shown in Fig. 5b, showing identical spectra at all resolutions up to T3999. Notably we did not obtain a spectrum for the T7999 simulation without FLTs, since postprocessing had to be disabled to save sufficient memory for that simulation to succeed, albeit producing only (ASCII) timing information used in the quantitative comparisons.

Figure 6 shows the measured number of floating-point operations used for the inverse or direct spectral transform of a single global field at different horizontal resolutions, using either the FLTs or the dgemm routine. The numbers have been scaled with $N^2 \log_3 N$. At T7999 resolution, approximately 4 times less floating-point operations are required using the FLTs, and the scaling implies an asymptotic increase of the number of operations $\propto (N^2 \log_3 N)$ with the FLTs and an increase of $N^3$ with the dgemm routine. At lower resolutions there are savings in the number of floating-point operations but not in wall-clock time, since the standard dgemm matrix–matrix multiplications are highly optimized and memory cache use is efficient. When applied to large matrices, these dgemm calls in IFS run at much higher computational rates (4–7 gigaflops sustained), compared to other parts of the IFS model (e.g., physical parameterizations such as the vertical diffusion scheme.

FIG. 3. Comparison of (a) 500-hPa geopotential height rmse and (b) anomaly correlation in the Northern Hemisphere for several T2047 10-day forecast simulations with selected values of the compression accuracy $\epsilon$. The control forecast (no FLT, solid, open triangles) is compared with FLT simulations using $\epsilon = 10^{-10}$ (dotted, squares), $\epsilon = 10^{-7}$ (dash–dotted, open circles), and $\epsilon = 10^{-3}$ (dashed, full circles).
that run at 0.7–1 gigaflops or less). Figure 7 illustrates that the savings in the number of floating-point operations are not evenly distributed across the wavenumber space, with the largest savings for large $N^2$ values. Therefore the threshold value $I_{\text{thresh}}$ has been introduced, below which the standard matrix–matrix multiplication dgemm is used for maximum efficiency in wall-clock time. Notably, all smaller-sized submatrices within the FLT tree level structure also use dgemm.

Figure 8 shows the average total wall-clock time computational cost (in milliseconds) per spectral transform during actual NWP simulations at different horizontal resolutions, comparing simulations with the new FLT and simulations using dgemm. Both show a significant cost increase when increasing the horizontal resolution globally from T3999 (~5-km horizontal grid length) to T7999 (~2.5 km). In Fig. 9 the cost of $10^7$ transforms (derived from the same simulations) is scaled by $N^2 \log^3 N$. The results imply an asymptotic increase that is equal to the results for the number of floating-point operations [i.e., $\propto O(N^2 \log^3 N)$] when using the FLT up to the largest tested resolution T7999. Comparing Figs. 9 and 6 it is clear that there is a delay in obtaining actual benefits in terms of reducing the wall-clock execution time by the relative reduction of floating-point operations because of the apparent improved efficiency of dgemm for larger matrices. Notably, for the optimized dgemm the wall-clock time increase with increased resolution scales much better than the $\propto O(N^3)$ increase of the floating-point operations.

There is also a long history in improving the efficiency of the spectral transform method by reducing the number of transforms actually required (cf. Temperton 1991).
In particular, the introduction of the semi-Lagrangian advection removed the requirement to explicitly evaluate the nonlinear advective derivatives and thus eliminated the need to spectrally transform the moist prognostic variables and the chemical tracers, representing a substantial cost saving. The typical average number of spectral transforms per time step in the nonhydrostatic IFS, $N_{\text{TFs}}$, can be calculated as $N_{\text{TFs}} = I \times [I_{\text{tot}} \times (N_{\text{prog3D}} + N_{\text{der3D}} + N_{\text{filt}}) + N_{\text{prog2D}} + N_{\text{der2D}}]$, where $N_{\text{prog3D}}$ and $N_{\text{prog2D}}$ denote the minimum number of three-dimensional and two-dimensional prognostic variables required. The sum of $N_{\text{der3D}}$ and $N_{\text{der2D}}$ denote the number of horizontal derivatives requiring extra transforms, and $N_{\text{filt}} = 2$ denotes extra transforms related to a dealiasing filter applied in the IFS. All these numbers are different for inverse and direct Legendre transforms and are also different for the Fourier transforms, since for example for the zonal and meridional winds ($u, v$) meridional derivatives can be derived from zonal derivatives and thus only require two extra Fourier transforms.

**FIG. 6.** Number of floating-point operations used for the inverse or direct spectral transform of a single global field at different horizontal resolutions using the FLTs (right bars) compared to using dgemm (left bars). Numbers are scaled with $N^2 \log^3 N$.

**FIG. 7.** Number of floating-point operations ($\times 10^6$) required for the different zonal wavenumbers ($m$). In this example taken from a T3999 simulation, $(N - m + 2)/2 \leq I_{\text{thresh}} = 900$, which means that for all $m < 2200$ the butterfly algorithm is applied. The simulation has been optimized for wall-clock time, resulting in a sharp drop in the number of floating-point operations used when switching to the FLT algorithm (solid line). The small steps in the curve relate to the levels of subdivisions in the butterfly algorithm.

**FIG. 8.** Average total wall-clock time computational cost (in milliseconds) per transform as obtained from actual NWP simulations with 91 vertical levels and different horizontal resolutions for simulations with FLTs (right bars) and with the standard dgemm (left bars), respectively.

**FIG. 9.** Average total wall-clock time computational cost (in milliseconds) of $10^7$ transforms for the same simulations as in Fig. 8, but scaled with $N^2 \log^3 N$, illustrating the approximate scaling found when using the FLTs (right bars) up to the largest tested T7999 horizontal resolution. Notably, with the optimized dgemm routine a scaling in wall-clock time that is much better than $N^3$ is obtained in the tested range.
(Temperton 1991). The iteration counter $I$ indicates the number of iterations in the iterative-centered-implicit (ICI) time-stepping scheme (Bénard 2003; Bénard et al. 2010). The primary reason why the hydrostatic model is approximately half the overall cost of the nonhydrostatic IFS model is the substantially reduced number of required spectral transforms in the former. Specifically, $I = 1$ in the hydrostatic model with $(N_{\text{prog3D}}, N_{\text{prog2D}}, N_{\text{der3D}}, N_{\text{der2D}}) = (4, 2, 1, 2)$ for the inverse Legendre transforms, and $I = 2$ with $(7, 2, 3, 2)$ in the nonhydrostatic case; $N_{\text{der3D}} = N_{\text{der3D}} + 2$ for the inverse Fourier transforms, whereas there are no derivatives in the direct Fourier and Legendre transforms with $N_{\text{prog3D}} = N_{\text{prog3D}} - 2$. All simulations in the comparison have been run with the nonhydrostatic IFS model with $l_{\text{tot}} = 91$ vertical levels for each three-dimensional prognostic variable. It should be noted, however, that simulations up to T2047 are usually run with the hydrostatic option with indiscernible meteorological results. The above numbers are not considering additional transforms at initial time and for postprocessing purposes.

4. Discussion and conclusions

The European Centre for Medium-Range Weather Forecasts (ECMWF) plans to implement a global horizontal resolution of approximately 10 km by 2015 for its assimilation and high-resolution forecasts, and approximately 20 km for the ensemble forecasts. The scales resolved at these resolutions are still hydrostatic and the efficiency and accuracy of the hydrostatic, semi-Lagrangian, semi-implicit solution procedure using the spectral transform method is shown to remain a relevant benchmark for NWP applications at this resolution. The computational efficiency of the hydrostatic IFS model has been further illustrated in the Athena project, a NSF funded initiative to determine the feasibility of using dedicated supercomputing resources to rapidly accelerate progress in modeling climate variability out to decadal and longer time scales (Kinter et al. 2013; Jung et al. 2012). Over a 6-month period, IFS was run for $3 \times 47$ yr at T159 resolution, $1 \times 47$ yr at T511, $3 \times 47$ yr at T1279, and an additional $19$ yr at T2047 resolution. With the results in section 3 and given the necessity to often perform ensembles at medium-range, monthly, seasonal or climate-scale lengths, the semi-implicit semi-Lagrangian, spectral transform model appears to be still a competitive method in the medium term, at least if only hydrostatic scales are resolved.

Moreover, the results presented in this paper suggest that the concern about the disproportionately growing computational cost of the Legendre transforms with increasing resolution has been mitigated. At T2047 ($\approx 10$ km), T3999 ($\approx 5$ km), and at T7999 ($\approx 2.5$ km) horizontal resolutions in actual NWP simulations, we find that the spectral transform computations scale $\propto O(N^2 \log^2 N)$. The computational cost of all the spectral computations (the spectral transforms and the spectral computations associated with solving the Helmholtz equation arising from the semi-implicit solution procedure) relative to the total computational cost of the model is 33% at T3999 resolution with 91 vertical levels for the nonhydrostatic model with $I = 2$. Here the MPI parallelization over the wavenumbers $m$ efficiently distributes the work to reduce the overall wall-clock time and the memory footprint.

The (energy) cost and latency of the parallel communications within the spectral transforms and the communications within the spectral computations, however, have not been addressed in this paper and remain a concern. Spectral-to-gridpoint transformations require data-rich (and energy inefficient) global communications at every time step that may become too expensive on future massively parallel computers. The latter aspect is investigated as part of the Collaborative Research into Exascale Systemware, Tools and Applications (CRESTA) project, and preliminary results suggest a way forward in the medium term through the use of modern computer language concepts (e.g., PGAS, coarray FORTRAN) overlapping computations and communications. Comparing the relative computational cost of the spherical harmonics transforms plus all the spectral computations (i.e., solving the Helmholtz equation and other spectral computations) as a percentage of the overall model cost, it
is found that at all resolutions and configurations tested the overhead due to the MPI communications required is \( \approx 12\% \) on the IBM cluster, as can be deduced by comparing the left and right columns in Fig. 10 for each horizontal resolution. Each bar in Fig. 10 has been derived considering all gridpoint computations (e.g., related to semi-Lagrangian advection) and physics computations (including the wave model) but without considering IO, synchronization costs (barriers, in particular in the transpositions), and other ancillary costs. All runs used the nonhydrostatic version of the IFS with 91 vertical levels. The relatively small overhead indicates a good potential for “hiding” the communications behind the computations. However, communication cost may increase substantially with increased numbers of compute threads, and only up to T2047 the tested configurations reflect the operational use situation of \( \approx 240 \) simulated forecast days per day. With the current implementation there is still a small overall relative growth of the total spectral computations cost with increasing resolution. However, it should be noted that (i) the maximum overall cost of all aspects associated with the spectral part of the model is less than 50\% of the overall model cost, and (ii) that all simulations were done with the existing nonhydrostatic model. For the hydrostatic model the relative cost of all aspects associated with the spectral part is approximately half of the values shown for each horizontal resolution in Fig. 10, so for the hydrostatic model the maximum overall cost of all aspects associated with the spectral part of the model is less than 25\% of the overall model cost. Given the large number of transforms required in the existing nonhydrostatic model, further research in the medium term will thus also focus on removing iterations that involve a duplication of all spectral transforms.

Moreover, there is some evidence that a substantial further reduction in the wall-clock time computational cost of the spectral transforms may be achieved through applying low energy general purpose graphics processor units (GPGPUs). In the latter context it will be of particular interest to explore the additional parallelism offered by the described butterfly tree level structure.

Finally, we note the potential opportunities offered by lossy compression in the FLTs as a paradigm for trading numerical accuracy at the smallest scales for energy efficiency in ensemble-based-uncertainty estimation.

Acknowledgments. We thank M. Tygert for his suggestion to implement the butterfly algorithm instead of the Cuppen-based FLT. We also gratefully acknowledge the implementation of a very elegant recursive Cuppen algorithm by Mike Fisher used in an earlier attempt to implement a FLT. Moreover we thank Agathe Untch, Erland Källén, Peter Dueben, and two anonymous reviewers for their comments and suggestions. We are also grateful to Rob Hine for his help with preparing the figures. The author Nils Wedi benefited from a stay at the Newton Institute, Cambridge, during the program on “Multiscale Numerics for the Atmosphere and Ocean.”

APPENDIX

Sketch of the Butterfly Algorithm

During the precomputation, the butterfly structure is set up following the algorithm for the rapid evaluation of special function transforms described in O’Neil et al. (2010). In this algorithm, submatrices are extracted from the matrix \( \mathbf{S} \) with its coefficients \( P^m_n(x_k) \) in (13) according to Fig. 1, where at level 0 each submatrix contains \( I_{\text{MAXCOLS}} \) columns (except the last one, in the general case). In our implementation the following code is used to prepare for the compressed matrix representation of the form \( \mathbf{S}_{r\times s} = \mathbf{C}_{r\times k} \mathbf{A}_{k\times s} \):

for \( l = 0 \rightarrow L \) do

for all \( j, k \) boxes do

if \( l = 0 \) then

\[ \mathbf{S}_{0,k} = \text{extract_sub_matrix}( ) \]

\[ \text{compr_sub_matrix}(\mathbf{S}_{0,k}, \mathbf{A}_{0,k}) \]

\[ \text{store } \mathbf{A}_{0,k} \]

else

\[ \mathbf{S}_{l,j,k} = \text{comb_compr}_l\text{-and}_r\text{-neighb}(\mathbf{C}, l - 1) \]

\[ \text{compr_sub_matrix}(\mathbf{S}_{l,j,k}, \mathbf{A}_{l,j,k}) \]

\[ \text{store } \mathbf{A}_{l,j,k} \]

end if

if \( l = L \) then

\[ \text{store } \mathbf{C}_{L,j} \]

end if

end for

end for.

During the application stage \( \mathbf{f} = \mathbf{S}\mathbf{a} \) is computed to precision \( \epsilon \), where \( \mathbf{a} \) is an arbitrary vector:
for \( l = 0 \to L \) do
  for all \( j, k \) boxes do
    if \( l = 0 \) then
      store \( \beta_{0,k} = A_{0,k} \alpha_k \)
    else
      store \( \beta_{l,j,k} = A_{l,j,k} \times \text{comb}_l \text{and}_r \text{neigh}(\beta, l - 1) \)
    end if
  end for
  if \( l = L \) then
    store \( f_{L,j} = C_{L,j} \beta_{L,j} \)
  end if
end for.

Finally, \( f \) is obtained by concatenation of the \( f_{L,j} \).

REFERENCES


