A Spectral Model for Process Studies of Rotating, Density-Stratified Flows

K. B. WINTERS, J. A. MACKINNON, AND BREN MILLS

Integrative Oceanography Division, Scripps Institution of Oceanography, University of California, La Jolla, California

(Manuscript received 5 February 2003, in final form 19 May 2003)

ABSTRACT

A numerical model designed for three-dimensional process studies of rotating, stratified flows is described. The model is freely available, parallel, and portable across a range of computer architectures. The underlying numerics are high quality, based on spectral expansions, and third-order time stepping. Optional submodels include accurate calculation of Lagrangian trajectories. Special consideration has been taken to ensure ease of use by geophysical, as distinguished from computational, scientists. The mathematical and computational methods underlying the model are presented here as are several illustrative applications highlighting the model capabilities and the types of flows amenable to simulation using the model. Sample applications include forced inertial gravity waves, parametric subharmonic instability, shear-driven mixing layers, instability of a low Froude number vortex street, and geostrophic adjustment of intermittent, isolated mixing patches.

1. Introduction

In this work, we describe the features and use of a numerical model designed to provide approximate solutions to the Navier–Stokes equations for density-stratified fluids in a rotating reference frame. The code is freely available and is intended for use as a tool for process-oriented simulations of stratified fluid flow, in particular for nonlinear interactions between internal gravity waves and transitional processes and instabilities resulting in disordered, three-dimensional motions. For our purposes here, we refer to this latter class of flows as turbulent. The algorithm is designed to be run on distributed memory multiprocessor computers using a data-parallel programming paradigm. Within the practical limits of memory and speed on the various platforms, the model can be run on machines ranging from Macintosh laptop computers, to workstation clusters, to large-scale community resources such as the Cray T3E.

The underlying numerical methods are based on spectral (Fourier, cosine, and sine) expansions of the flow variables and therefore rely heavily on fast Fourier transforms (FFTs). Consequently, spatial resolution and convergence characteristics greatly exceed those of models based on finite differences for discrete spatial differentiation. The trade-off, however, is that application of the model is limited to problems that can be formulated with simple boundary conditions. The model presented here is designed to solve problems either in (a) triply periodic spatial domains or (b) horizontally periodic domains with free-slip rigid lids at the top and bottom.

The remainder of the paper is organized as follows. The transformed equations of motion are derived in section 2, followed by a discussion of the discrete numerical algorithms in section 3. Parallel implementation is discussed in section 4. Routine diagnostics, illustrative, and validation simulations are presented in sections 5 and 6. Parallel performance benchmarks are provided in section 7. A brief guide to the configuration of the code for user-defined applications and the output files is included in appendixes A and B.

2. Spectral form of the Boussinesq equations

Throughout this document, we will use the following notation conventions:

$u(x, y, z, t)$, scalar function of position and time; (1)

$\mathbf{u}(x, y, z, t)$, vector function of position and time; (2)

$\tilde{u}(k, l, m, t)$,

wavenumber space transform of $u(x, y, z, t)$; (3)

$\mathbf{u}(k, l, m, t)$,

vector form in transformed space; and (4)

$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$; $\mathbf{k}$, $\mathbf{l}$, $\mathbf{m}$,

unit vectors in the x, y, z and k, l, m directions. (5)

The equations of motion for a density-stratified fluid on an f plane are

Corresponding author address: Dr. Kraig Winters, IOD/UCSD, 9500 Gilman Dr., Mail Code 0209, La Jolla, CA 92093-0209.
E-mail: kraig@coast.ucsd.edu

$$\frac{\partial u}{\partial t} = -\mathbf{u}\cdot\nabla u + fv - \frac{1}{\rho}\frac{\partial p}{\partial x} + F_1 + \nu\nabla^2 u, \quad (6)$$

$$\frac{\partial v}{\partial t} = -\mathbf{u}\cdot\nabla v - fu - \frac{1}{\rho}\frac{\partial p}{\partial y} + F_2 + \nu\nabla^2 v, \quad (7)$$

$$\frac{\partial w}{\partial t} = -\mathbf{u}\cdot\nabla w - \frac{1}{\rho}\frac{\partial p}{\partial z} + F_3 + \nu\nabla^2 w - g, \quad (8)$$

$$\frac{\partial \rho}{\partial t} = -\mathbf{u}\cdot\nabla\rho + F_4 + \kappa\nabla^2\rho, \quad \text{and} \quad (9)$$

$$\nabla\cdot\mathbf{u} = 0, \quad (10)$$

where u, v, w, ρ, p are functions of the spatial coordinates x, y, z (vertical, positive upward), and time t. In addition, ν, κ are the molecular viscosity and scalar diffusivity respectively; $f = 2\Omega\sin(\phi)$ is the Coriolis parameter; Ω is the earth's rotation frequency; ϕ is the latitude; and g is the gravitational acceleration; F_i are external forcing terms. Equations (6)–(10) are displayed with Laplacian dissipation and diffusion operators even though the code is implemented more generally, permitting a choice within a family of hyperviscosity operators. Because the numerical treatment of each of these schemes is similar, we present the algorithmic approach for the particular choice corresponding to Laplacian dissipation and diffusion.

For stably stratified geophysical flows, numerical accuracy can often be enhanced by focusing available precision on relatively small fluctuations about a steady background. We therefore decompose the (potential) density ρ into a reference value, a time-independent field, and a fluctuating component that varies in space and time:

$$\rho(x, y, z, t) = \rho_0 + \overline{\rho}(x, y, z) + \rho'(x, y, z, t). \quad (11)$$

A corresponding decomposition of pressure is defined in terms of the nonfluctuating components of density:

$$p = \overline{p}(x, y, z) + p'(x, y, z, t) \quad (12)$$

$$\frac{\partial \overline{p}}{\partial z} = -(\rho_0 + \overline{\rho})g. \quad (13)$$

We make the Boussinesq approximation assuming that density fluctuations are small compared to the reference value and hence use ρ_0 in place of ρ except where it is multiplied by g [see Gill (1982), Lighthill (1978) for a discussion of the physical underpinnings of this approximation], obtaining

$$\frac{\partial u}{\partial t} = -\mathbf{u}\cdot\nabla u + fv - \frac{1}{\rho_0}\frac{\partial \overline{p}}{\partial x} - \frac{1}{\rho_0}\frac{\partial p'}{\partial x} + F_1 + \nu\nabla^2 u, \quad (14)$$

$$\frac{\partial v}{\partial t} = -\mathbf{u}\cdot\nabla v - fu - \frac{1}{\rho_0}\frac{\partial \overline{p}}{\partial y} - \frac{1}{\rho_0}\frac{\partial p'}{\partial y} + F_2 + \nu\nabla^2 v, \quad (15)$$

$$\frac{\partial w}{\partial t} = -\mathbf{u}\cdot\nabla w - \frac{1}{\rho_0}\frac{\partial p'}{\partial z} + F_3 + \nu\nabla^2 w - \frac{g\rho'}{\rho_0}, \quad (16)$$

$$\frac{\partial \rho'}{\partial t} = -\mathbf{u}\cdot\nabla(\overline{\rho} + \rho') + F_4 + \kappa\nabla^2(\overline{\rho} + \rho'), \quad (17)$$

and

$$\nabla\cdot\mathbf{u} = 0. \quad (18)$$

Note that the vertical gradient of $\overline{p}$ does not appear in the equation for w once the hydrostatic relation (13) is invoked.

A few more manipulations are required before transforming the equations to wavenumber space. Because the flow is divergence free, we can write

$$\mathbf{u}\cdot\nabla\rho' = \nabla\cdot(\rho'\mathbf{u}) - \rho'(\nabla\cdot\mathbf{u}) = \nabla\cdot(\rho'\mathbf{u}). \quad (19)$$

The nonlinear terms can be expressed in terms of the vorticity $\boldsymbol{\omega} = \nabla\times\mathbf{u}$ using the identity

$$\mathbf{u}\cdot\nabla\mathbf{u} = -\mathbf{u}\times\boldsymbol{\omega} + \frac{1}{2}\nabla|\mathbf{u}|^2. \quad (20)$$

We define the new pressure variables,

$$P \equiv \frac{p'}{\rho_0} + \frac{\nabla|\mathbf{u}|^2}{2} \quad \text{and} \quad \overline{P} \equiv \frac{\overline{p}}{\rho_0}, \quad (21)$$

and, for notational convenience,

$$\mathbf{T} \equiv (T_1, T_2, T_3) = [\mathbf{u}\times\boldsymbol{\omega}]\cdot(\mathbf{x}, \mathbf{y}, \mathbf{z}) + (F_1, F_2, F_3) - \left(\frac{\partial\overline{P}}{\partial x}, \frac{\partial\overline{P}}{\partial y}, 0\right) \quad (22)$$

and obtain

$$\frac{\partial u}{\partial t} = T_1 - \frac{\partial P}{\partial x} + fv + \nu\nabla^2 u, \quad (23)$$

$$\frac{\partial v}{\partial t} = T_2 - \frac{\partial P}{\partial y} - fu + \nu\nabla^2 v, \quad \text{and} \quad (24)$$

$$\frac{\partial w}{\partial t} = T_3 - \frac{\partial P}{\partial z} + \nu\nabla^2 w - \frac{g}{\rho_0}\rho' \quad (25)$$

to go along with Eqs. (17) and (18).

The algorithmic objective is to compute approximate solutions to the stratified flow equations over a mesh of discrete grid points in Cartesian (x, y, z) space. The grid is defined by $x_i = (i-1)\Delta x$, $y_j = (j-1)\Delta y$, and $z_k = (k-1)\Delta z$ with $(\Delta x, \Delta y, \Delta z) = (L_x/nx, L_y/ny, L_z/nz)$ for $(i, j, k) = 1, 2, \ldots, (nx, ny, nz+1)$. We assume here that nx, ny, and nz can be represented as products of powers of 2, 3, or 5 so that discrete FFTs can be computed efficiently over the global data space.

a. Transformed equations: Triply periodic boundary conditions

Time integration is performed in wavenumber space to the spatially transformed version of the equations of

motion. The form of the variable expansions, and hence the form of the transformed equations, depend on the boundary conditions to be imposed. For triply periodic boundary conditions, we expand each of the primitive variables u, v, w, ρ', and P in discrete Fourier series of the form

$$u(x, y, z, t) = \sum_i \sum_j \sum_n \tilde{u}_{k,l,n}(t) e^{i(kx+ly+mz)}, \quad (26)$$

where the indices $i = [-nx/2 + 1, -nx/2 + 2, \ldots, nx/2]$, $j = [-ny/2 + 1, -ny/2 + 2, \ldots, ny/2]$ and $n = -[nz/2 + 1, -nz/2 + 2, \ldots, nz/2]$ define discrete wavenumber values $k = i(2\pi/L_x)$, $l = j(2\pi/L_y)$, and $m = n(2\pi/L_z)$. We assume here that $\partial\overline{\rho}/\partial z$ is a periodic function of the spatial variables though $\overline{\rho}$ need not be. This allows, for example, simulation of periodic flows in uniformly stratified fluids.

Spatial differentiation is analytic in wavenumber space with the operator transform pairs $(\partial/\partial x, \partial/\partial y, \partial/\partial z) \rightarrow (ik, il, im)$; for example,

$$\frac{\partial u}{\partial x} = \sum_i \sum_j \sum_n ik\tilde{u}_{k,l,n}(t) e^{i(kx+ly+mz)} \Leftrightarrow \frac{\partial u}{\partial x} = ik\tilde{u}. \quad (27)$$

For the triply periodic case, transformation of the equations and some straightforward algebra yield

$$\frac{\partial \tilde{u}}{\partial t} = \tilde{T}_1 - ik\tilde{P} + f\tilde{v} - \nu|\mathbf{k}|^2\tilde{u}, \quad (28)$$

$$\frac{\partial \tilde{v}}{\partial t} = \tilde{T}_2 - il\tilde{P} - f\tilde{u} - \nu|\mathbf{k}|^2\tilde{v}, \quad (29)$$

$$\frac{\partial \tilde{w}}{\partial t} = \tilde{T}_3 - im\tilde{P} - \frac{g}{\rho_0}\tilde{\rho}' - \nu|\mathbf{k}|^2\tilde{w}, \quad (30)$$

$$\frac{\partial \tilde{\rho}'}{\partial t} = -i\mathbf{k} \cdot \widetilde{(\rho\mathbf{u})} + (\tilde{F}_4 - \widetilde{\mathbf{u} \cdot \nabla\overline{\rho}} - \kappa\widetilde{\nabla^2\overline{\rho}}) - \kappa(|\mathbf{k}|^2\tilde{\rho}'), \quad \text{and} \quad (31)$$

$$\mathbf{k} \cdot \mathbf{u} = 0. \quad (32)$$

The transformed variables, for example, $\tilde{u}$ or $\tilde{T}_3$, represent sets of values at each discrete wavenumber location. The transformed equations therefore form a set of ordinary differential equations in time for the complex expansion coefficients at each value of the triplet (i, j, n) within the expansion range for a given discrete grid mesh.

A significant advantage of working with the transformed equations is that the pressure variable $\tilde{P}$ can be expressed in terms of the other variables by dotting the wavenumber vector $(\mathbf{k}, \mathbf{l}, \mathbf{m})$ into the vector form of the momentum equations and making use of (32). This analytical step is analogous to solving a Poisson equation for P in physical space, a task that often requires the use of iterative methods in purely grid-point-based formulations. The resulting expression for $\tilde{P}$ is then substituted into the momentum equations to give

$$\frac{\partial \tilde{u}}{\partial t} = \left(\frac{l^2 + m^2}{|\mathbf{k}|^2}\right)G_1 - \frac{kl}{|\mathbf{k}|^2}G_2 - \frac{km}{|\mathbf{k}|^2}G_3 - \nu|\mathbf{k}|^2\tilde{u}, \quad (33)$$

$$\frac{\partial \tilde{v}}{\partial t} = -\frac{kl}{|\mathbf{k}|^2}G_1 + \left(\frac{k^2 + m^2}{|\mathbf{k}|^2}\right)G_2 - \frac{lm}{|\mathbf{k}|^2}G_3 - \nu|\mathbf{k}|^2\tilde{v}, \quad \text{and} \quad (34)$$

$$\frac{\partial \tilde{w}}{\partial t} = -\frac{km}{|\mathbf{k}|^2}G_1 - \frac{lm}{|\mathbf{k}|^2}G_2 + \left(\frac{k^2 + l^2}{|\mathbf{k}|^2}\right)G_3 - \nu|\mathbf{k}|^2\tilde{w}, \quad (35)$$

where

$$G_1 = \tilde{T}_1 + f\tilde{v}, \quad (36)$$

$$G_2 = \tilde{T}_2 - f\tilde{u}, \quad \text{and} \quad (37)$$

$$G_3 = \tilde{T}_3 - \frac{g}{\rho_0}\tilde{\rho}'. \quad (38)$$

b. Transformed equations: Free-slip, solid surfaces at $z = 0, L_z$

We frequently encounter problem formulations in which the vertical extent of the domain is finite, though viscous boundary layers are not of primary interest. We can modify the variable expansions slightly to enable simulation of flows satisfying free-slip conditions at the rigid lids $z = 0, L_z$. In particular, we consider flows satisfying periodicity conditions in x and y with

$$w = 0 \qquad z = 0, L_z, \quad (39)$$

$$\frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = 0 \qquad z = 0, L_z, \quad (40)$$

$$\rho = \rho_{\text{bottom}}(x, y) \qquad z = 0, \quad \text{and} \quad (41)$$

$$\rho = \rho_{\text{top}}(x, y) \qquad z = L_z. \quad (42)$$

By choosing

$$\overline{\rho} = \frac{z}{L_z}\rho_{\text{top}}(x, y) + \left(1 - \frac{z}{L_z}\right)\rho_{\text{bottom}}(x, y), \quad (43)$$

we obtain the simple boundary condition $\rho' = 0$ at $z = 0, L_z$ and can therefore treat the primitive variables pairs (u, v) and (w, ρ') in terms of cosine and sine expansions in the vertical, respectively. Other choices could be made for specifying $\overline{\rho}$ consistent with these expansions. In general the function $\overline{\rho}$ must be of the form

$$\overline{\rho} = f(x, y) + g(x, y)z + \sum_n a_n(x, y) \sin\frac{n\pi}{L_z}z, \quad (44)$$

where f, g, and a_n are arbitrary periodic functions of x and y.

One application for which this formulation is useful is internal wave dynamics in a stratified, finite-depth ocean. For these problems, the vertical displacement of fluid particles from their equilibrium position is ap-

proximately given by $\rho'/\overline{\rho}_z$, where the fluctuations are defined relative to some ambient density stratification $\overline{\rho}$, for example, with $d\overline{\rho}/dz$ equal to a constant. Since the vertical motion must vanish at a rigid lid, we require that the vertical displacement, and hence, ρ' also vanish. Note, however, that these boundary conditions are not generally adiabatic because there can be a nonzero flux of density through the upper and lower bounding surfaces. Nevertheless, these boundary conditions are commonly prescribed for problems focusing on gravity wave interactions with reflective upper and lower boundaries. Convectively driven flow produced by a steady, spatially variable temperature difference between two free-slip plates can also be formulated using these boundary conditions. These and other examples are discussed further in section 6.

We refer to functions that vanish at $z = 0, L_z$ as odd functions; that is, their periodic extensions have odd symmetry across the boundaries. The functions ρ' and w are both odd and therefore expanded in the form

$$w(x, y, z, t) = \sum_i \sum_j \sum_n \tilde{w}_{k,l,n}(t) \sin(mz) e^{i(kx+ly)}, \quad (45)$$

where the i, j horizontal indices and k, l wavenumbers are defined as in the triply periodic case but in the vertical we take $n = [0, 1, \ldots, nz]$ and $m = n(\pi/L_z)$.

Functions with even periodic extensions, that is, u, v, and P, have z derivatives that vanish at $z = 0, L_z$ and are expanded in the form

$$u(x, y, z, t) = \sum_i \sum_j \sum_n \tilde{u}_{k,l,n}(t) \cos(mz) e^{i(kx+ly)}. \quad (46)$$

The vertical derivatives take the forms

$$\frac{\partial}{\partial z} w(x, y, z, t) = \sum_i \sum_j \sum_n m\tilde{w}_{k,l,n}(t) \cos(mz) e^{i(kx+ly)} \quad \text{and} \quad (47)$$

$$\frac{\partial}{\partial z} u(x, y, z, t) = \sum_i \sum_j \sum_n -m\tilde{u}_{k,l,n}(t) \sin(mz) e^{i(kx+ly)}. \quad (48)$$

The procedure for deriving the transformed equations for these boundary conditions is nearly identical to that for the triply periodic case except that the transformed space vertical derivative operator is either $\pm m$, depending on whether the function being differentiated has even or odd symmetry. Our notation is such that $\tilde{f}$ could refer to either the result of a Fourier–Fourier–sine transform or a Fourier–Fourier–cosine transform, with the symmetry properties implied by the context.

The transformed equations for domains of finite extent in z are

$$\frac{\partial \tilde{u}}{\partial t} = \left(\frac{l^2 + m^2}{|\mathbf{k}|^2}\right) G_1 - \frac{kl}{|\mathbf{k}|^2} G_2 + \frac{ikm}{|\mathbf{k}|^2} G_3 - \nu|\mathbf{k}|^2 \tilde{u}, \quad (49)$$

$$\frac{\partial \tilde{v}}{\partial t} = -\frac{kl}{|\mathbf{k}|^2} G_1 + \left(\frac{k^2 + m^2}{|\mathbf{k}|^2}\right) G_2 + \frac{ilm}{|\mathbf{k}|^2} G_3 - \nu|\mathbf{k}|^2 \tilde{v}, \quad (50)$$

$$\frac{\partial \tilde{w}}{\partial t} = -\frac{ikm}{|\mathbf{k}|^2} G_1 - \frac{ilm}{|\mathbf{k}|^2} G_2 + \left(\frac{k^2 + l^2}{|\mathbf{k}|^2}\right) G_3 - \nu|\mathbf{k}|^2 \tilde{w}, \quad \text{and} \quad (51)$$

$$\frac{\partial \tilde{\rho}'}{\partial t} = -ik\widetilde{(\rho' u)} - il\widetilde{(\rho' v)} + m\widetilde{(\rho' w)} + (\tilde{F}_4 - \widetilde{\mathbf{u} \cdot \nabla \overline{\rho}} - \kappa \widetilde{\nabla^2 \overline{\rho}}) - \kappa|\mathbf{k}|^2 \tilde{\rho}'. \quad (52)$$

3. Numerical algorithms

a. Treatment of nonlinear terms

Collocation methods (Canuto et al. 1988) are used for nonlinear terms. Terms involving products, for example, $\rho'\mathbf{u}$ and $\mathbf{T}$, are computed by local multiplication of values at the discrete grid points followed by transformation of the resulting product to wavenumber space. Care must be taken when using this approach however as aliasing errors are possible. Rather than employing relatively severe truncation rules to formally exclude the possibility of aliasing, we routinely monitor energy spectra in each of the three wavenumber directions as a means of detecting the development of aliasing errors. In well-resolved simulations of nonlinear flows, viscous and diffusive effects act to remove energy at small scales, at rates as fast or faster than the rate of energy transfer from larger scales. With insufficient grid resolution, however, energy tends to accumulate near the maximum resolvable wavenumbers because nonlinear transfers to unresolved scales are misrepresented at resolved scales. Accumulation of energy at small scales is a characteristic signature of an underresolved simulation. In practice, it is often difficult to predict how fine a mesh will be required for a given nonlinear flow. Monitoring the energy spectra is one tool by which an objective assessment of whether a given grid is sufficiently fine can be made.

b. Transform tools

As discussed further in section 4, the parallelization strategy is based on performing simultaneous transforms of different data on different processors rather than distributing the computations required for each transform across the processors. This ''coarse grained'' approach to parallelism allows the use of serial FFT code. We adopt the approach of Cooley et al. (1970) and use a discrete, complex to complex FFT as a computational kernel, around which the various specialized transforms (real to conjugate symmetric, real sine/cosine, etc.) are built. Specifically, we have implemented procedures 4–7 of Cooley et al. (1970) using the Temperton FFT package for the low-level FFTs.

c. Time integration and treatment of viscous/diffusive terms

At each discrete spatial wavenumber, each of the transformed ordinary differential equations has the form

$$\frac{d}{dt}f = F - \alpha(k^2 + l^2 + m^2)f, \tag{53}$$

where F is a function of the wavenumbers k, l, and m. An integrating factor can be introduced to obtain the condensed form

$$\frac{d}{dt}[fe^{\alpha(k^2+l^2+m^2)t}] = e^{\alpha(k^2+l^2+m^2)t}F. \tag{54}$$

The quantity $fe^{\alpha(k^2+l^2+m^2)t}$ is integrated discretely in time using an explicit, third-order Adams–Bashforth (AB3) scheme. Introducing superscripts to indicate the time level $t^n = (n - 1)dt$, the AB3 scheme is defined by

$$\frac{dy}{dt} = \psi \Rightarrow y^{n+1} = y^n + \frac{dt}{12}(23\psi^n - 16\psi^{n-1} + 5\psi^{n-2}) + O(dt^4). \tag{55}$$

Applying this scheme to (54), for time levels $n \geq 2$, yields

$$f^{n+1} = e^{-dt\alpha(k^2+l^2+m^2)} \times \left\{ f^n + \frac{dt}{12}[23F^n - 16e^{-dt\alpha(k^2+l^2+m^2)}F^{n-1} + 5e^{-2dt\alpha(k^2+l^2+m^2)}F^{n-2}] \right\}. \tag{56}$$

A simple startup procedure based on explicit, lower-order methods is used for $n < 2$. Because the AB3 is a multilevel scheme, the functions F from two previous time steps are required in addition to the current value in order to advance the fields. The AB3 method is asymptotically stable for wavelike problems as discussed by Canuto et al. (1988) and Slinn (1995).

We note here that simple generalizations to the diabatic terms in the governing equations, for example, anisotropic diffusion or dissipation operators, or ''hyperviscosity'' schemes such as biharmonic diffusion, are treated similarly. These generalizations lead to a slight modification of the exponential terms in (56). The numerical code is implemented in term of generalized operators, the parameters of which are specified by the user during configuration.

4. Parallelization

The parallelization strategy is based on the data parallel or single-process, multiple-data (SPMD) paradigm in which an identical copy of the program runs on each node of a parallel machine. Each process operates on distinct portions of the globally distributed data space. Exchange of data between the nodes is accomplished through calls to the message passing interface (MPI; Gropp et al. 1996; Gropp and Lusk 1996). To the extent that each processor can execute its assigned set of tasks without requiring an exchange of data with other processors, a parallel algorithm achieves a linear speedup; that is, the total execution time decreases linearly with the number of processors. For nontrivial algorithms, however, it is not possible to eliminate the need for data transfers between nodes entirely and communication overhead reduces the return of employing additional processors. Communication overheads are significant for algorithms based on global differentiation methods for which the value of a discrete derivative at a given point depends not only on values at neighboring points but on all values along the direction of differentiation. Implicit compact schemes and analytic differentiation in discrete Fourier space are two examples of global differentiation methods. The increase in accuracy and convergence rate comes at a cost in terms of the inherent degree of parallelism of the algorithm.

The global data space for the physical space representation of the primitive variables consists of four three-dimensional arrays of real valued elements corresponding to the three velocity components and the perturbation density at each of the $nx \times ny \times (nz + 1)$ spatial grid points. The apparently extra value in the z direction is required for functions that are even symmetric in z. For periodic or odd functions, the value at $z = L_z$ is equal to either the value at $z = 0$ or zero and would not generally require explicit storage. At certain stages of the algorithm, we will need to work with the transformed, wavenumber space representation of these, and similarly expanded, variables. The parallelization strategy therefore relies intrinsically on both a physical and a wavenumber space distribution of data across multiple processors.

The data partitioning in physical space is based on the logical allocation of a contiguous block of locnz + 1 = nz/np + 1 horizontal planes consisting of $(nx + 2) \times (ny + 1)$ real data values to each of np processor elements as shown in Fig. 1a. There are two (one) extra locations in the x (y) dimension that are not accessed in the physical space representation. They permit in-place real to complex FFTs in the x direction and a graceful collapse of the algorithm to two dimensions when $ny = 0$. We refer to the storage arrays in this orientation as data slabs. Each processor is assigned a unique identification label *myid*, with values ranging from 0 to $np - 1$. The uppermost (locnz + 1) plane is superfluous except for the uppermost processor myid = $(np - 1)$, where it is required if free-slip boundary conditions are imposed. For simplicity of the implementation, we use identical storage arrays for all processors and restrict the partition parameters locnz = nz/np and locnx = $(nx/2)/np$ to integer values.

Parallelization issues arise for sections of the algo-

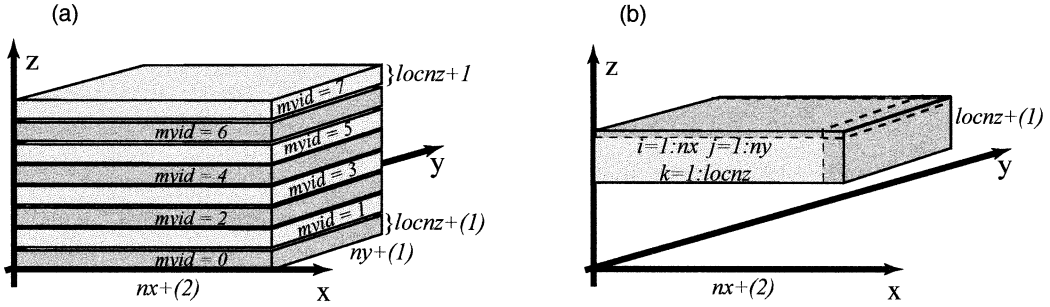

FIG. 1. Layout of data in slab orientation. (a) Physical space representation of each three-dimensional flow variable is distributed across *np* processors in slabs. (b) Each interior processor is assigned $nx \times ny \times nz/np$ real values stored within a slightly expanded array of size $(nx + 2) \times (ny + 1) \times (nz/np + 1)$.

rithm where the underlying numerical operations are global in scope rather than local. For example, Fourier transformation in the *z* direction of an initially real-valued, *z*-periodic variable would require $nx \times ny$ independent one-dimensional transforms over vectors of length *nz*. The operation is global in the sense that, at each discrete (x, y) location, all *nz* elements in the vertical direction are required to calculate the transform. Communication is an issue because the data needed to compute any one of the required transforms is distributed across the *np* processors. In contrast, *x* and *y* transforms are local in the sense that sets of these transforms can be performed over data already residing on each processor. Clearly, the operations required for each set of *x* or *y* transforms are independent of one another and the sets of transformations can therefore be executed in parallel.

We illustrate the approach by considering a three-dimensional transform of *w*, from physical to wavenumber space, for a simulation in a finite depth ocean.

Forward Fourier-Fourier-Sine transform of w

Given the $nx \times ny \times nz/np$ discrete values of *w* stored in the local data slab, each processor performs the following steps in parallel.

1) Perform $ny \times nz/np$ one-dimensional, real to complex FFTs over data vectors of length *nx*. Results from each transform are overwritten into the slab storage structure filling the extra two elements in the *x* direction.
2) Perform $(nx/2 + 1) \times (nz/np)$ complex to complex FFTs over data vectors of length ny.
3) Send $(np - 1)$ subblocks of the results, each consisting of $[(nx + 2)/2]/np \times ny \times (nz/np)$ complex elements to each of the remote processors.
4) Receive $(np - 1)$ like-sized subblocks of results from each of the remote processors, arranging the data to form a local column of horizontally transformed data with $[(nx + 2)/2)/np \times ny \times nz]$ complex data elements as shown in Fig. 2.
5) Perform sine transforms in *z* over each of the $(nx + 2)/np \times ny$ data vectors of length *nz* in the local data column.

Steps 3 and 4 consist of data transfers between processors and thus represent communication overhead to the algorithm for the three-dimensional transforms. In the implementation, these steps are isolated from all computation and performed in a single routine called slabs_to_columns. Inverse transforms are performed in a similar fashion, reversing the order of the transform directions and utilizing a communications routine called columns_to_slabs.

5. Diagnostics

a. Energy balance equations

Data enabling a time-dependent analysis of the energy balances Eqs. (57) and (58) (see, e.g., Winters et al.

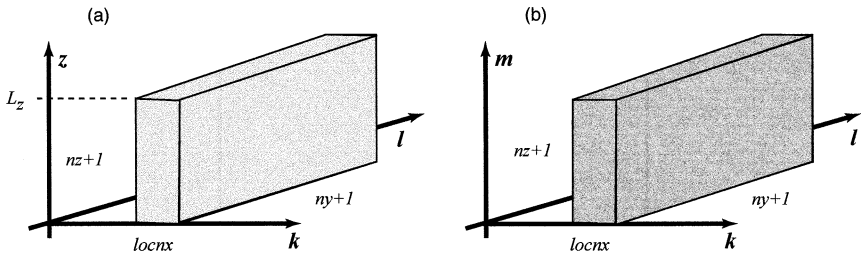

FIG. 2. Layout of data in wavenumber space representation. (a) Data arrangement after columns_to_slabs has been completed. (b) Data representation after *z* transforms are computed.

1995; Gill 1982) is routinely computed at user-specified time intervals as the integration proceeds:

$$\frac{d}{dt}E_k = -\frac{1}{V}\oint_S \left[\frac{p}{\rho_0} + \frac{1}{2}(u^2 + v^2 + w^2)\right]\mathbf{u}\cdot\hat{n}\, dS - \frac{g}{\rho_0 V}\int_V \rho w\, dV + \frac{1}{V}\int_V \nu\mathbf{u}\cdot\nabla^2\mathbf{u}\, dV + \frac{1}{V}\int_V uF_1 + vF_2 + wF_3\, dV \quad \text{and} \tag{57}$$

$$\frac{d}{dt}E_p = -\frac{g}{\rho_0 V}\oint_S gz\rho\mathbf{u}\cdot\hat{n}\, dS + \frac{g}{\rho_0 V}\int_V \rho w\, dV + \frac{g}{\rho_0 V}\int_V zF_4\, dV + \frac{\kappa g}{\rho_0 V}\oint_S z\nabla\rho\cdot\hat{n}\, dS - \frac{\kappa g}{\rho_0 L_z}(\overline{\rho}_{\text{top}} - \overline{\rho}_{\text{bottom}}), \tag{58}$$

where

$$E_k = \frac{1}{2V}\int_V (u^2 + v^2 + w^2)\, dV \quad \text{and} \tag{59}$$

$$E_p = \frac{g}{\rho_0 V}\int \rho z\, dV \tag{60}$$

are the kinetic and potential energies in joules per kilogram, respectively.

For horizontally periodic boundary conditions, the surface integral in Eqs. (57), that is, the term containing the pressure work and advection of kinetic energy, is always zero, regardless of which boundary condition set is chosen in the z direction. In (58), the surface integral quantifying energy changes associated with advection of fluid across the bounding surface S vanishes for the rigid-lid boundary conditions $w = 0$ at $z = 0, L_z$ but is nonzero for vertically periodic conditions. The quantities E_k and E_p, the terms on the right-hand sides of (57) and (58), and the corresponding values of time are written to an ascii output file at specified time steps. Discrete differentiation of $E_k(t)$ and $E_p(t)$ can then be used to check the accuracy of the volume-integrated energy balances and to diagnose bulk energy transfers within an evolving flow. Slightly modified forms of the dissipative terms are used if hyperviscosity is specified during configuration.

b. Energy spectra

As discussed previously, pseudospectral numerical methods are subject to aliasing errors if the underlying discrete grid is too coarse. To detect the onset of aliasing errors, the output data can be analyzed to compute energy spectra as functions of wavenumber in each dimension. Because calculation of spectra in each wavenumber direction requires significant interprocessor data exchanges, we do not perform the calculation as the simulation proceeds. Rather we monitor the output files using the spectral calculations as a form of a posteriori quality control.

Typically, a nonlinear simulation will start out well resolved and exhibit energy spectra that decay with wavenumber near the small-scale limits. As the simulation proceeds and the flow evolves, energy is generally transferred toward small, dissipative scales. Depending on the rate of the transfer, the grid may or may not be fine enough to resolve the evolving flow without aliasing. An insufficient grid can be diagnosed by examining the behavior of the spectra at small scales. If aliasing is detected, for example, upturned energy spectra near the cutoff wavenumbers is observed, the simulation can be rerun using a finer grid. To save computational effort, a well-resolved flow field at an intermediate time can be interpolated to a finer mesh and used as an ''initial'' condition for a higher-resolution continuation run.

c. Lagrangian trajectories

In addition to computing the flow fields, the underlying fields (and derived quantities) can be sampled in time along Lagrangian trajectories defined by

$$\frac{d\tilde{x}}{dt} = \mathbf{u}. \tag{61}$$

To compute the trajectories, velocities at positions that are not coincident with grid points are required. Rather than interpolating the required values from values at nearby gridpoint locations using locally based interpolation schemes, the direct-sum formulas Eqs. (26) and (27), or (45) and (46), are used. The values obtained in this way at arbitrary points are therefore just as accurate as the ''known'' values at the grid points. In this sense, the values are free of spatial interpolation errors. The summations are naturally distributed across multiple processors, as a distinct range of expansion coefficients are stored on each processor as shown in Fig. 2. For each distinct location (x, y, z) at which velocities are required, each processor computes a partial sum corresponding to the expansion coefficients within the processor's assigned portion of wavenumber space. The partial results are then globally summed to produce the required values. Though the summations are very efficiently computed, the summation algorithm itself is computationally intensive; it is not FFT based. For example, to compute values at $n = O(nx \times ny \times nz)$ arbitrary locations via direct summation requires $O(n^2)$ floating point operations. In contrast, if the N values were required at the grid points, as they are in the underlying flow calculations, they can be obtained using the FFT algorithm in $O(n \log n)$ operations. Our Lagrangian float model, while highly accurate, is therefore intrinsically computationally expensive within the overall FFT-based spectral algorithm. The model was de-

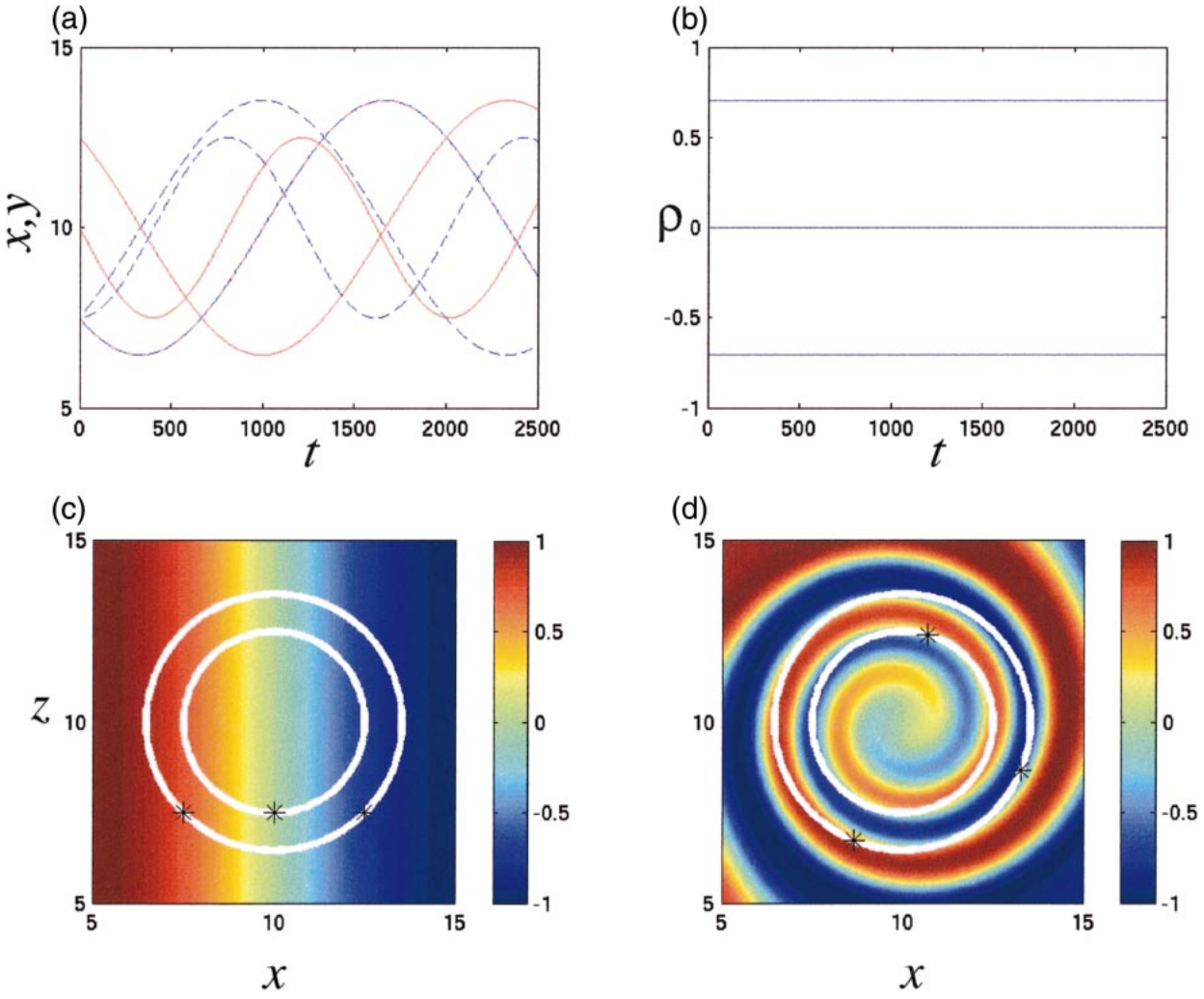

FIG. 3. Lagrangian advection. (a) Time series of position for three Lagrangian trajectories. (b) Density values along the same trajectories. (c) Initial (passive) density distribution, particle tracks traced out by the three trajectories over the simulation period. Initial positions shown as asterisks. (d) Final density distribution and trajectory locations.

signed to compute only a small number of trajectories (compared to the number of grid points), but to do so very accurately.

Slightly modified summation formulas are used to reconstruct, for example, the density field and its Laplacian $\nabla^2\rho$. The time series obtained by sampling ρ at each point on a given trajectory, $\rho_E(t)$, can then be compared to $\rho_L(t)$ obtained by integrating the equation

$$\frac{d\rho_L}{dt} = \kappa\nabla^2\rho. \tag{62}$$

Exact integration of this equation gives, in principle, a time series identical to ρ_E. In practice, however, accuracy depends on estimation of second derivatives and discrete integration in time. Estimation of the second derivative is a noise-amplifying operation; that is, noise in the high-wavenumber components of ρ will be amplified by the square of the wavenumber magnitude. Contamination of these components will lead to a discrepancy between the time series before the contamination is likely to be noticed in the underlying flow fields. Comparison of the two series is therefore a stringent test of the numerical quality not only of the ''Lagrangian float'' model, but for the underlying flow calculations as well. To minimize time-stepping errors, the Lagrangian float model is integrated in time using a fourth-order Runge–Kutta technique, which is inherently more accurate than the third-order method used to integrate the flow itself.

Figure 3 shows Lagrangian trajectories computed in a steady flow with circular streamlines in horizontal (x, y) planes. In this simulation, density was used as a passive scalar, that is, $g = 0$, and initialized with a uniform gradient in the x direction as shown in Fig. 3c. The viscosity and diffusivity were set to very small values and, over the time period shown here, their effects can

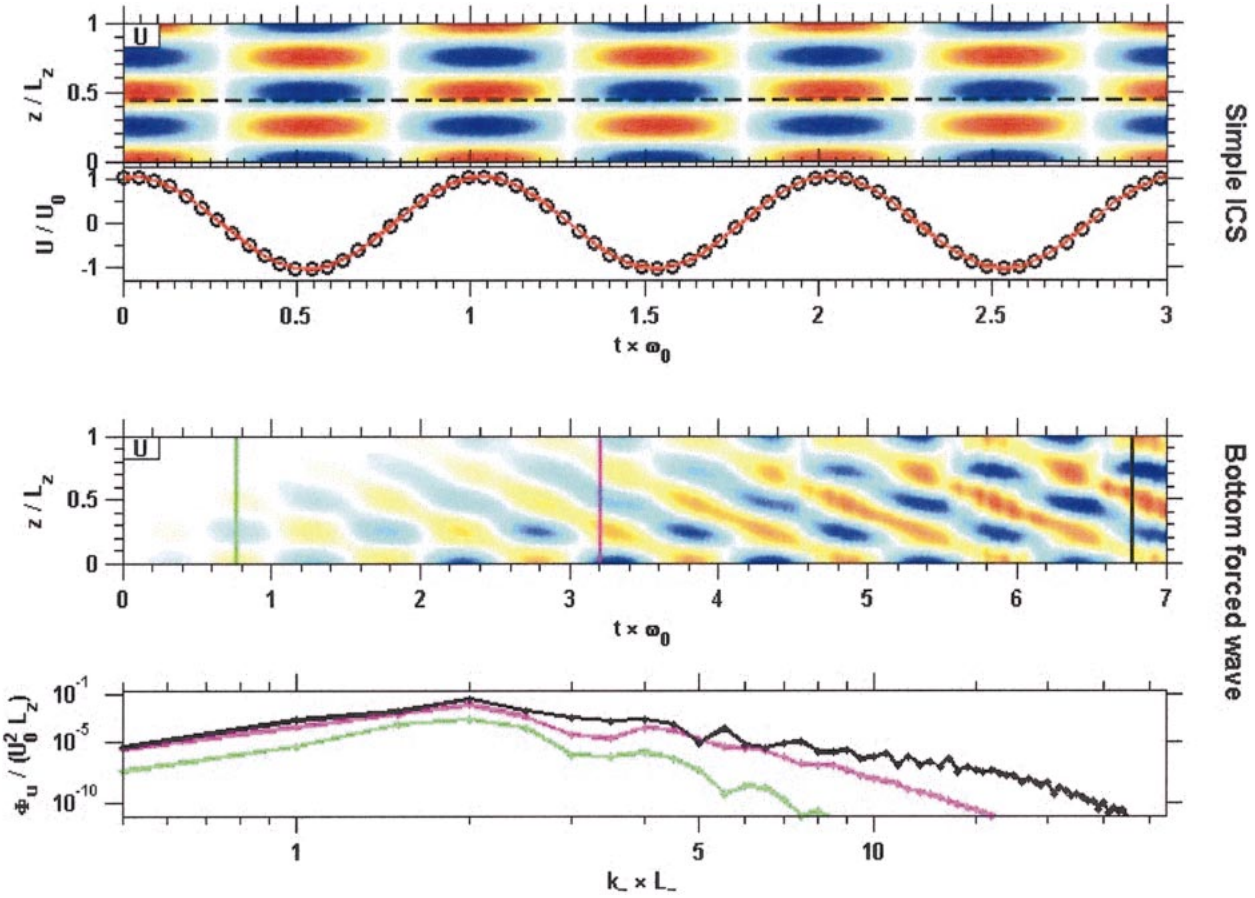

FIG. 4. (top) Depth and time series of horizontal velocity for an inertial gravity wave simulated as an initial value problem with free-slip boundary conditions. A time series at a fixed depth (circles) is shown in the second panel along with the analytical solution of the linearized equations (red line). (bottom) Depth and time series of an initially quiescent fluid subject to wavelike forcing in the near-bottom region. Vertical spectra at the three times indicated (color-matched vertical lines) are shown in the bottommost panel. All quantities have been nondimensionalized by the initial/forced wave period, total depth, and characteristic velocity.

be considered negligible. Time series of x and y position data are shown in Fig. 3a for three trajectories. Each trajectory traces out a circle in the (x, y) plane as shown in Fig. 3c. Though the circular motion of the fluid advects and distorts the density field (Fig. 3d) in the limit of vanishing diffusivity, the density along any given trajectory should remain constant. Figure 3b shows the computed density as a function of time for the same three trajectories. Both quantities ρ_E and ρ_L are shown but cannot be distinguished.

The float model can be configured to be imperfectly Lagrangian, that is, to use modified trajectory equations with residual or random components added to the right-hand side of (61). In addition, flow variables can be sampled at additional positions slightly offset from the location point of the infinitesimal float. This allows estimation of gradients of field quantities, for example, $\partial\rho/\partial z$ or $\partial\mathbf{u}/\partial z$, at each point along the trajectories. Such calculations are relatively expensive, however, as performing them is similar in workload to carrying nearby ''shadow'' floats in the calculation.

6. Applications

a. Internal gravity waves

The model is well suited to study nonlinear interactions between internal gravity waves. Simulations can be made in two or three dimensions, with or without rotation, and in the presence or absence of ambient flows. The ambient stratification need not be uniform, but can vary, for example, with depth. A variety of wave-related processes are therefore amenable to fully nonlinear simulation, for example, critical-level interactions (Winters and D'Asaro 1994), turning points, caustics, and wave instability.

Figure 4 shows two simple examples. In the first, the simulation is run as an initial value problem in a rotating reference frame, with initial conditions chosen as snapshot of a horizontally propagating, vertically standing wave. In the limit of small wave amplitude and vanishing viscosity, an analytic solution can be found. A comparison of the analytic solution, evaluated as a function of time at a fixed spatial position, with the computed

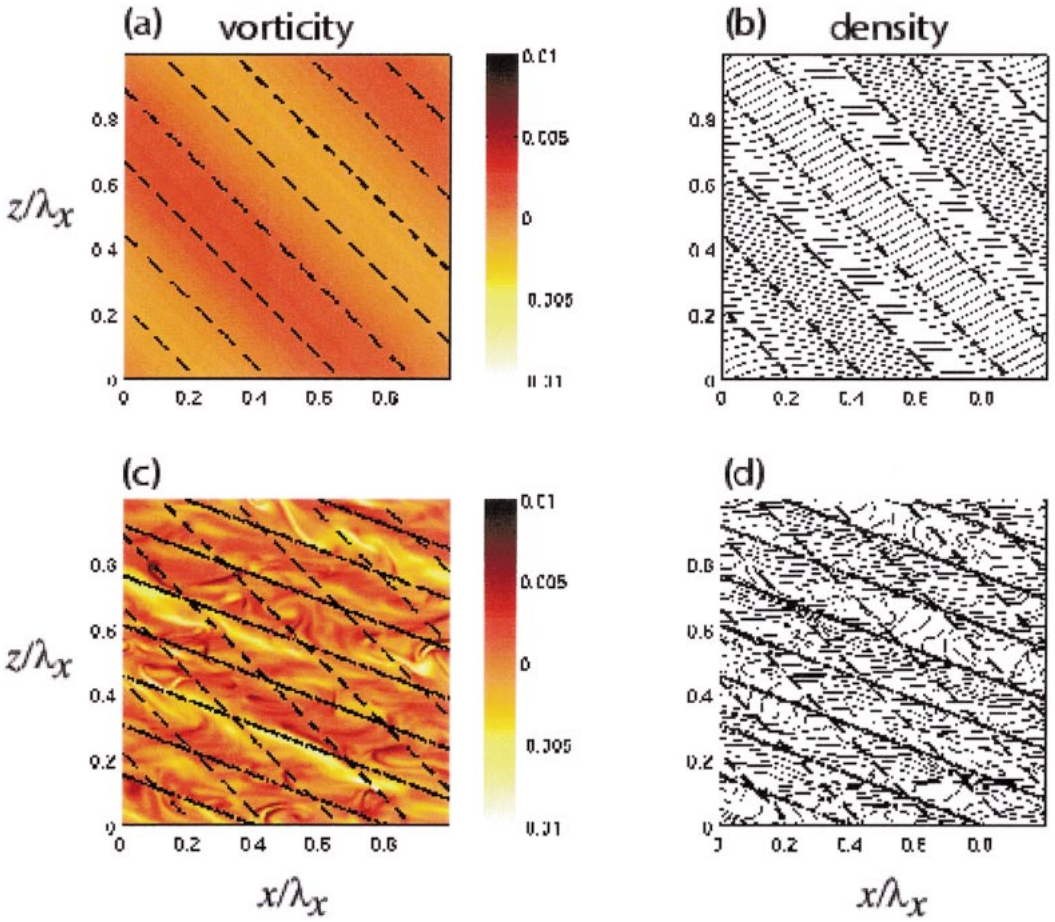

FIG. 5. (a) Initial vorticity ζ_y and density (b) ρ as functions of horizontal and vertical position, normalized by the horizontal wavelength $2\pi/k_x$. Also shown are the lines of constant phase (dashed) for the initialized wave motion. (c), (d) Same quantities at $t \approx 402\pi/\omega_0$. Phase lines (solid) corresponding to a linear wave of frequency $2\omega_0$ have been added.

solution at the same location is shown (Fig. 4, second panel from top).

Alternatively, forced waves can be simulated utilizing the functions F_i. In the example shown (Fig. 4, bottom panels), the (**x**, t) dependences of F_i are derived from a monochromatic, infinite-space planar gravity wave solution, but localized near the bottom by a windowing function. As in the unforced case, the wave frequency is set to the M_2 tidal frequency. Note the downward phase propagation, indicative of upward energy propagation. Even this relatively simple situation produces rich behavior, as nonlinear interactions between upward- and downward-propagating (reflected) waves increasingly transfer energy toward high frequencies and wavenumbers as shown in the bottom panel of Fig. 4. A similar approach might be used, for example, to simulate the nonlinear interactions between a spectrum of waves generated over variable topography.

Nonlinear interactions between gravity waves often lead to wave instabilities. We illustrate by recomputing the solution for run P1 in Bouruet-Aubertot et al. (2001). In this experiment, a two-dimensional, spatially periodic internal gravity wave propagates vertically through a linearly stratified fluid. The ambient flow field is infinitesimally disturbed by the presence of random velocity fluctuations such that the kinetic energy of the fluctuations is a factor of 10^4 smaller than the kinetic energy of the primary wave. As discussed in Bouruet-Aubertot et al. (2001), such waves are unstable, regardless of their initial amplitude. The instability is related to the parametric subharmonic instability (PSI) mechanism, transferring energy from the primary wave to one of half the frequency.

Figure 5a shows the y (into page) components of vorticity and Fig. 5b density, along with lines of constant phase for the primary wave at the start of the calculation. For linear gravity waves, the motion of fluid particles is along phase lines, with frequency proportional to the inclination angle from the horizontal. Figures 5c and 5d show the same quantities at a later time (after approximately 40 buoyancy periods). The overall orientation of the observed phase is consistent with energy transfer

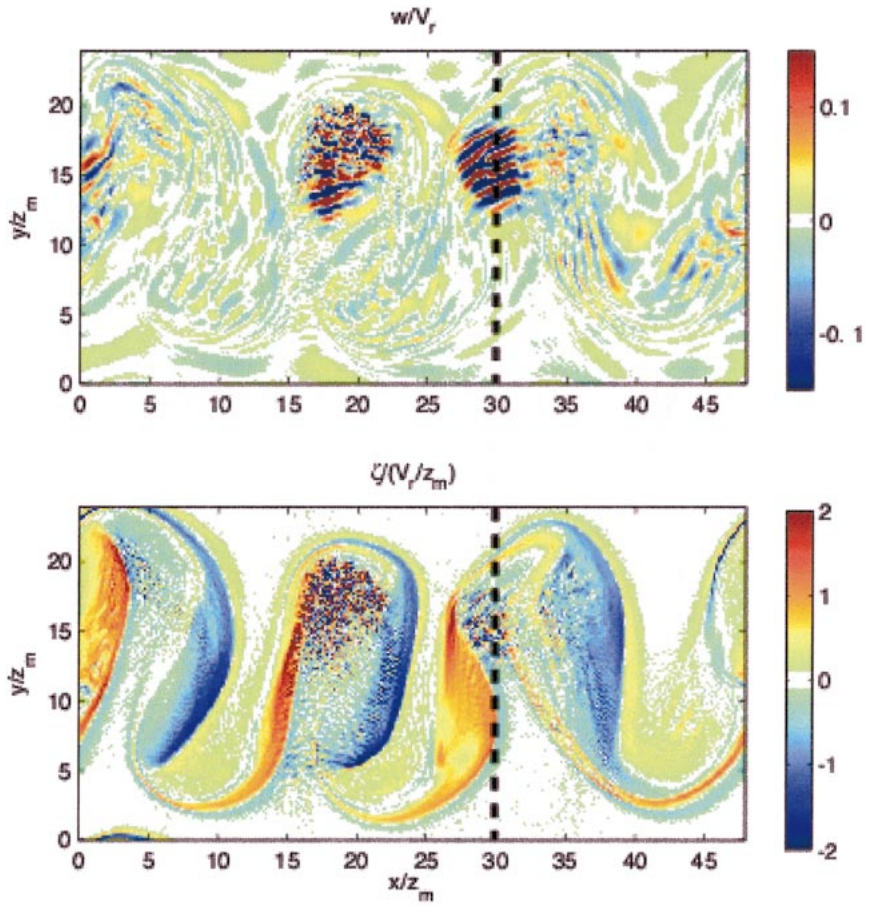

FIG. 6. (top) Vertical velocity in $z = L_z/2$ horizontal plane and (bottom) vertical vorticity in the same plane.

to lower frequency by PSI. Small-scale structure is also apparent in both the vorticity and density fields, which exhibit gravitationally unstable overturns.

b. *Vortex street instability*

In this example we consider a network of counterrotating vortex monopoles in a stably stratified fluid. A similar pattern of vorticity, sometimes called a vortex street, is seen after an object is towed through a stratified fluid and the initially turbulent wake collapses under the stabilizing effects of gravity. In the laboratory, these structures are typically observed to be long lived, surviving at least as long as it is feasible to conduct experiments (Spedding 1997).

We consider here a vorticity distribution comprising 12 pairs of vortex monopoles, each of which with an azimuthal speed V_θ given by

$$V_\theta = \pm \frac{r}{r_m} V \exp\left[1 - \left(\frac{r}{r_m}\right)^2 - \left(\frac{z}{z_m}\right)^2\right], \qquad (63)$$

where r and z are the radial and vertical distances from the vortex center, respectively; r_m and z_m are characteristic decay scales in the horizontal and vertical directions; and V is the characteristic speed of the swirling motions. Negative/positive vortex centers are distributed initially at $z = L_z/2$ along rows $y = L_y/2 \pm y_s/2$ with a spacing of x_s in the x direction. The precise locations and scales of the individual monopoles are perturbed by random fluctuations with magnitudes of about 5% of the corresponding scales. In addition, a random incompressible flow field is added at all resolvable scales. The total kinetic energy of this random component is set at 1×10^{-4} times the kinetic energy of the vortex motion.

Figure 6 shows the vertical velocity (top panel) and the vertical vorticity (bottom panel) on the central horizontal plane $z = L_z/2$ after the vortex street has been significantly deformed by the mutual interaction of the vortex monopoles. There has been an upscale energy transfer, producing a flow characterized by larger horizontal spatial scales than were present initially. Small-scale features are apparent in the vorticity field and are correlated with patches of highly structured regions of flow with alternating bands of upward and downward flow (Fig. 6 top). Owing to the vertical structure of the

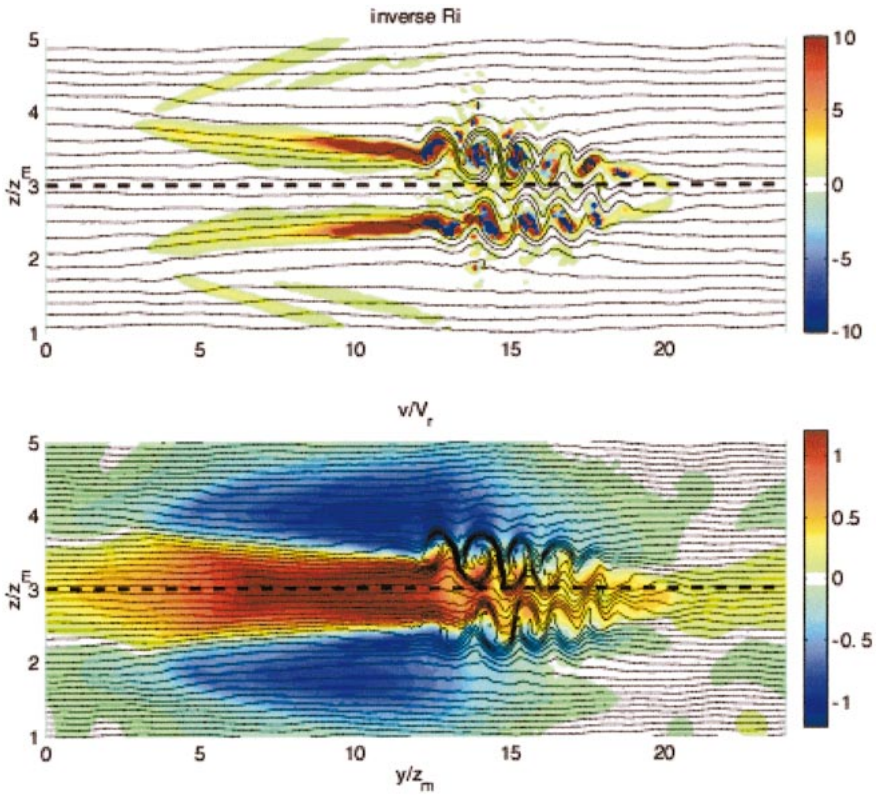

FIG. 7. (top) Inverse Richardson number (colors) and density contours. (bottom) Cross-stream velocity (colors) and density contours.

initial vorticity field, the vortex interactions and deformations are most pronounced in the central region of the flow. Figure 7 shows the Richardson number (a measure of stability in a stratified flow) and the cross-stream velocity component v in the y–z plane indicated in Fig. 6. The cross-stream flow has a jetlike structure, with high shears on the upper and lower flanks. These shears are unstable, as indicated by values of the inverse Richardson number greater than 4, producing a train of counterrotating Kelvin–Helmholtz (KH) billows on the flanks. The vertical motions associated with these secondary instabilities appear as the alternately signed bands in Fig. 6. Figure 8 shows some of the three-dimensional structure of the flow. Shown are the vertical vorticity on two separated horizontal planes as well as the planes $y = L_y$ and $x = L_x$ along with contours of density.

c. Shear-driven mixing layer

Very detailed studies of processes leading to turbulent transports in stratified flows can also be undertaken. We consider here a shear-driven mixing layer that develops from the unstable velocity and density profiles:

$$u = \frac{u_0}{2}\tanh\frac{2}{h_0}\left(z - \frac{L_z}{2}\right) \quad \text{and} \tag{64}$$

$$\rho = -\frac{\rho_0}{2}\tanh\frac{2}{h_0}\left(z - \frac{L_z}{2}\right). \tag{65}$$

The flow is unstable first to Kelvin–Helmholtz billows and subsequently to smaller-scale three-dimensional instabilities leading to turbulence. Eulerian aspects of this flow have been studied by Smyth and Moum (2002) and Smyth et al. (2001) as a model of small-scale turbulent patches observed by vertical profiling instruments in the ocean thermocline. The Lagrangian floats submodel was used by D'Asaro et al. (2004) to diagnose the signatures of such events detectable by Lagrangian instrumentation.

Figure 9 shows a snapshot of the density field in the $y = 0$ vertical plane, along with cross sections taken at the positions indicated in the top panel. The flow was computed on a grid with uniform mesh spacing at a resolution of $512 \times 256 \times 64$ for $\mathrm{Pr} = \nu/\kappa = 1$ on a Cray T3E using 32 processors. Thirty-six floats, each sampling the flow field at nearby shadow locations as well as their actual positions, were carried along for the

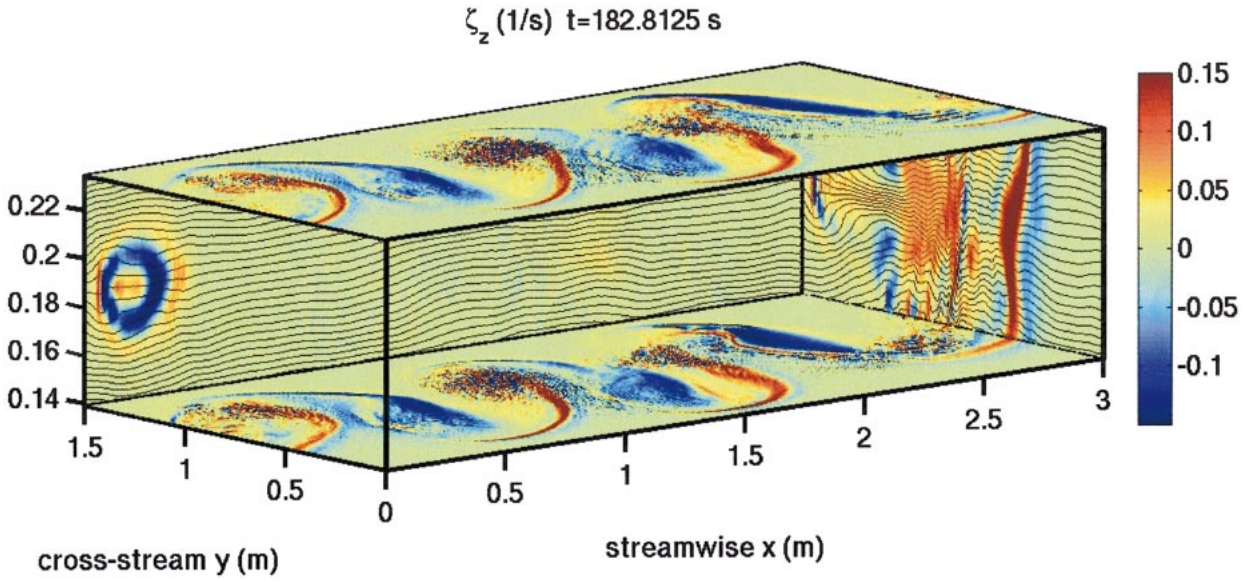

FIG. 8. Cutaway view of vertical vorticity and density contours when secondary instabilities are present.

Lagrangian study. Projections of the trajectories in streamwise, cross-stream, and horizontal planes are shown in Fig. 10. At early times, before the KH billows collapse and become turbulent, the trajectories trace out simple paths, particularly in the y–z and x–y planes, where the paths are vertical and horizontal lines, respectively. Significantly more structure is evident at later times.

d. Horizontal convection

We now consider the flow in an elongated container driven by an externally maintained temperature difference between the upper and lower bounding surfaces. We consider the response to a sinusoidal distribution of heating and cooling at the upper surface with the bottom surface maintained at the mean value imposed at the upper surface. A similar model has been examined recently (Paparella and Young 2002) in the context of the thermally driven component of the global circulation. Taking density as a surrogate for temperature, we prescribe

$$\rho = \Delta\rho \cos(2\pi x/Lx) \qquad z = L_z \quad \text{and} \qquad (66)$$

$$\rho = 0 \qquad z = 0, \qquad (67)$$

along with horizontal periodicity and free-slip rigid lids. The time-independent field $\overline{\rho}$ is prescribed as $\overline{\rho} = (z/L_z)\rho(x, y, L_z)$ and so ρ' is required to vanish at $z = 0, L_z$. In this simplified model, positive (negative) values of ρ correspond to cooling (heating) at the ''poles'' and ''equator,'' respectively.

Figure 11 shows the density field at several instants in time for a convectively driven flow with Rayliegh number $\text{Ra} = g'L_z^3/\nu\kappa = 1.5 \times 10^6$ and $\text{Pr} = \nu/\kappa = 3$, where $g' = (\Delta\rho/\rho_0)g$. Dense, convectively unstable fluid descends to depth at the poles $x = 0, L$ until it reaches the bottom surface, at which point it propagates toward the equator $x = 0$ as a gravity current. The simulation was done in two dimensions on a Macintosh laptop for a container with an aspect ratio of 10 using 1024×129 grid points in the horizontal and vertical directions, respectively.

e. Customization

Because the source is freely available, the code can be adapted for very specific research problems in fluid dynamics. Two examples are briefly presented here. W. D. Smyth (2003, personal communication) has modified the source to carry an additional passive scalar and to compute the active scalar ρ at higher resolution than that used for $\mathbf{u}$ and P. Such an approach is useful for flows with $\text{Pr} = \nu/\kappa > 1$, where the scale at which density fluctuations are diffusively smoothed are smaller than the scales at which kinetic energy is viscously dissipated. Figure 12 shows two scalars with different diffusivities computed at different resolutions within the same flow field.

M. A. Sundermeyer and M. P. Lelong (2003, personal communication) have used the model to study the formation of vortical modes through the relaxation of diapycnal mixing events, and their effect on the lateral dispersion of a passive tracer. To force the model, they have added a subroutine, which they call from within the main time-stepping loop of the code, to periodically inject potential energy (PE) in the form of randomly placed Gaussian-shaped stratification anomalies. Rather than explicitly simulating diapycnal mixing caused by breaking internal waves, they represent the effects of wave breaking by imposing a short-lived diapycnal dif-

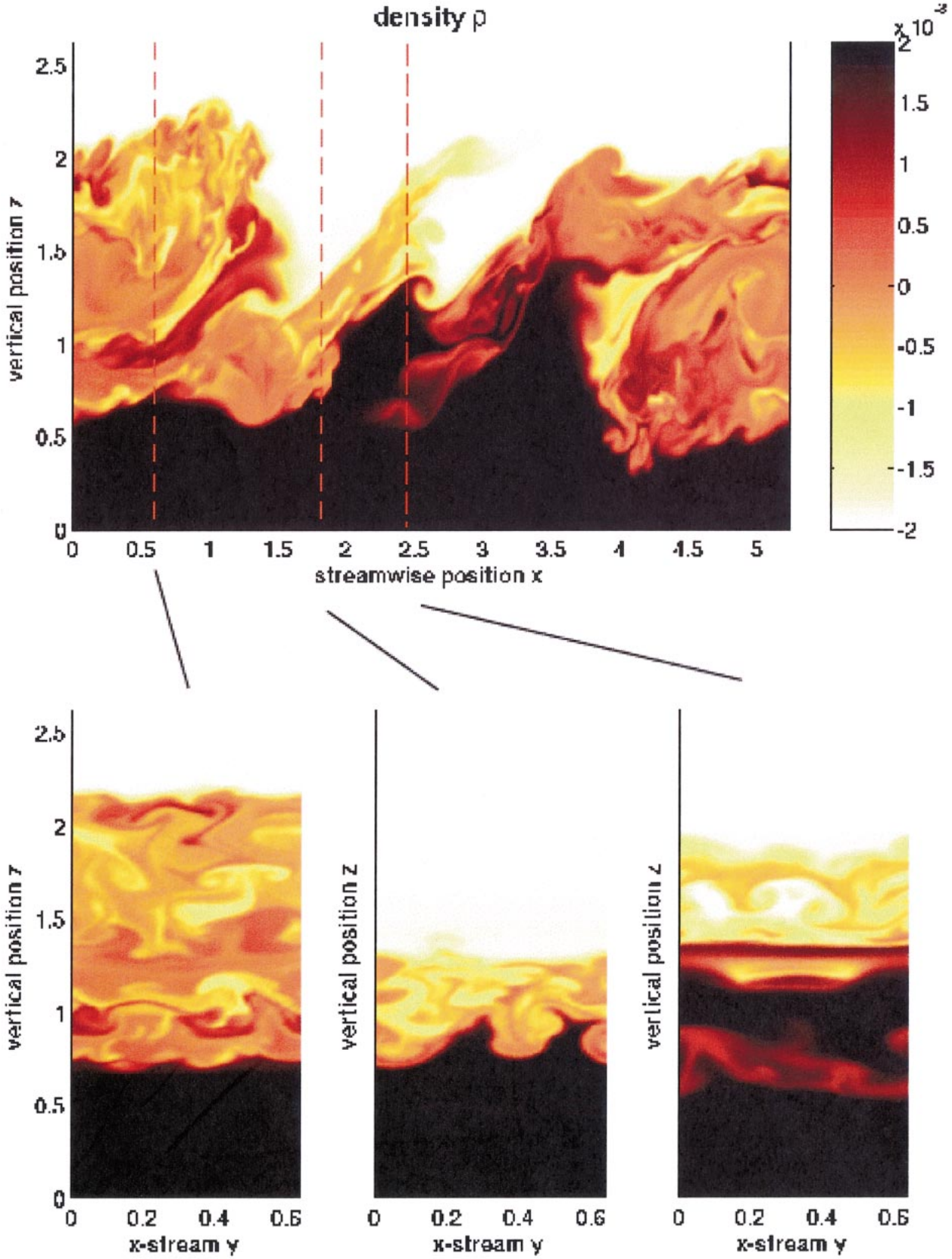

FIG. 9. Images of the density field from a turbulent mixing layer in the $y = 0$ streamwise–vertical plane and three spanwise–vertical cross sections.

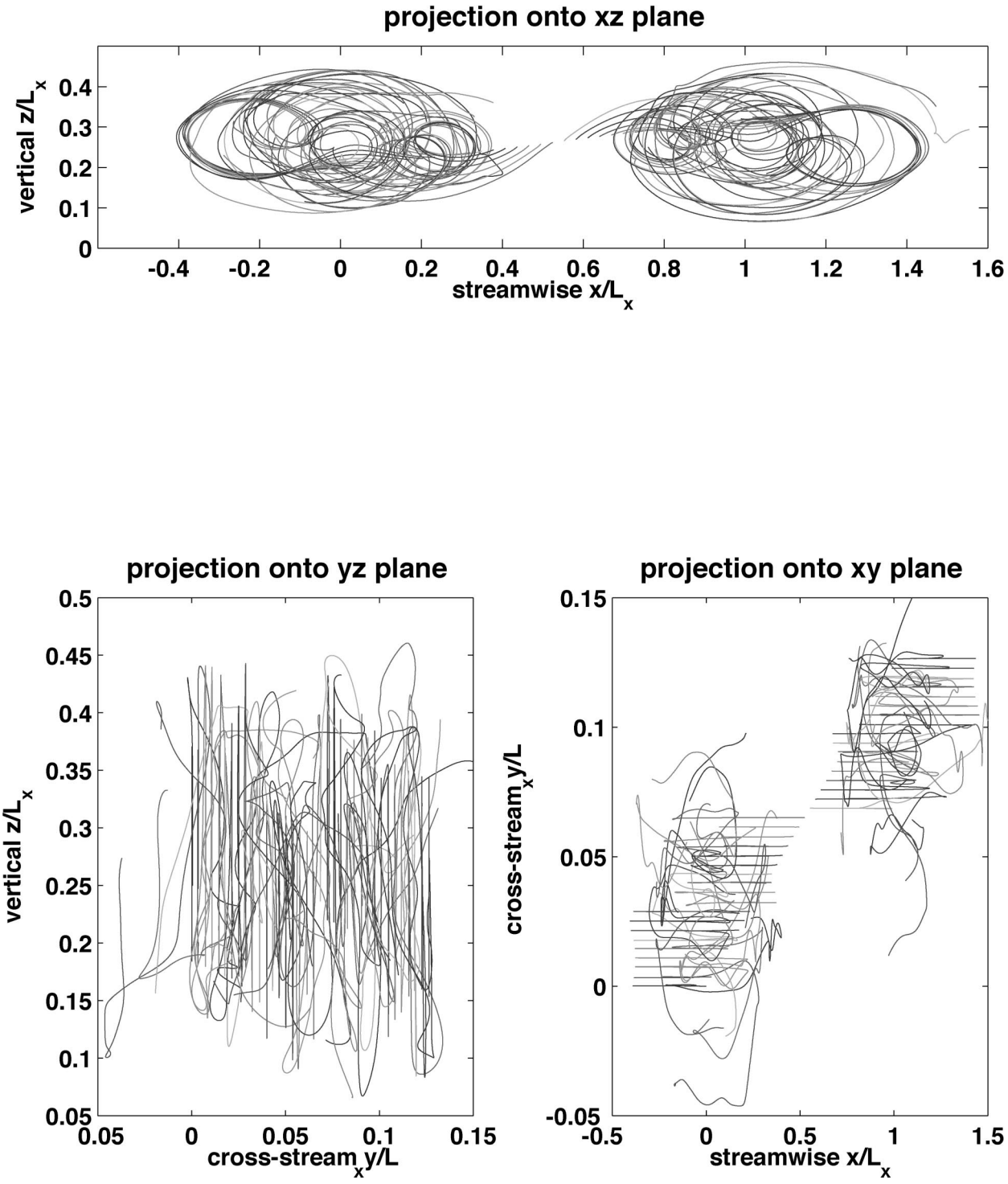

FIG. 10. Projections of Lagrangian trajectories in an evolving shear-driven mixing layer. Tracks are shown for the duration of the simulation.

fusivity profile at random locations in the model, the result of which subsequently adjusts to form small-scale vortical motions and radiating internal waves. Using this approach, they simulate flows that evolve from rest to a statistically stationary state in which the input of PE and subsequent conversion to kinetic energy (KE) is balanced by dissipation. The result is a random field of small-scale geostrophic eddies, or vortical modes, and a dispersing tracer field. The simulations are being used to test theoretical predictions of vortical mode stirring

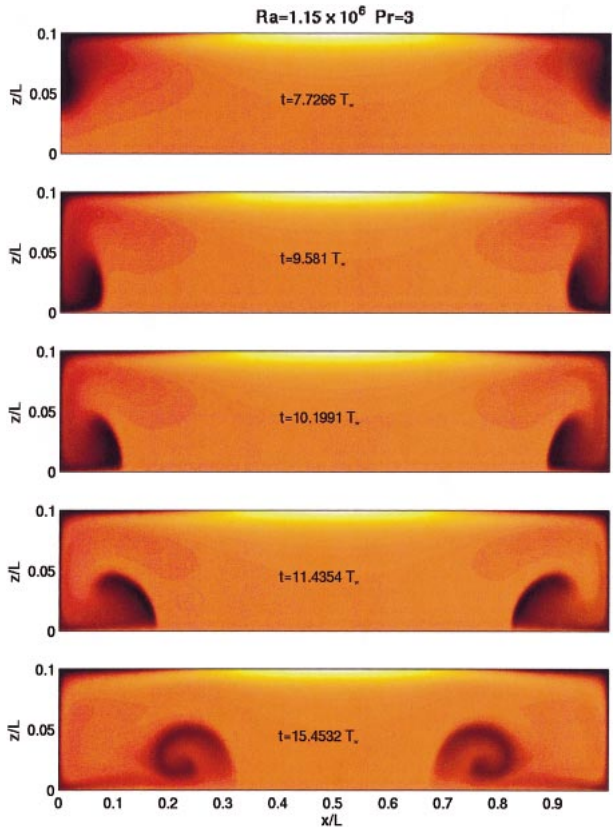

FIG. 11. Images of the density field at several instants in time for a fluid driven by an externally maintained density difference $\Delta\rho(x, y)$. The time scale $T_* = \sqrt{L_z/g'}$.

in the ocean (Fig. 13; see also Sundermeyer et al. 2003, manuscript submitted to *J. Phys. Oceanogr*).

7. Performance benchmarks

The run-time profile for a a single processor run using a grid mesh of 120^3 points, with the Lagrangian floats model switched off, was constructed. Nearly 88% of the execution time is spent computing multidimensional FFTs, which have an operation count that scales like $O(n \log n)$, where n is the total number of mesh points. Tasks such as computation of the wavenumber space derivatives needed to form the vorticity field, calculating various nonlinear products in physical space, combining various terms to form the right-hand sides of the transformed equations of motion, and time stepping the fields in wavenumber space scale like $O(n)$. The additional tasks of writing three-dimensional output fields to disk and computing the spatial integrals required for the energy balances also have $O(n)$ workloads. These tasks, however, are not executed at every time step but at user-specified frequencies. As the problem size n increases, the algorithm becomes increasingly dominated by the FFTs.

A parallel scalability analysis quantifies the expected improvement in performance as additional processors are added to a parallel computation. Such analyses seek to answer the question ''how many processors can I use before adding additional processors reduces the execution time too little to be worthwhile?'' Quantitative answers are specific to both the code and the architecture of the computing facility.

We assume that a given algorithm can be decomposed into a parallelizable component and a nonparallelizable or serial component. The total execution time T is then written as

$$T(np) = \frac{T_p}{np} + T_s, \tag{68}$$

where T_p is the time required for the parallelizable portion, T_s is the time required for the serial portion, and np is the number of processors. The total execution times for runs using different numbers of processors can be used to construct a linear relationship between T and $1/np$, yielding T_p and T_s for a given computing environment. Results are typically displayed as a plot of speedup S versus the number of processors np where $S = T(1)/T(np)$. In the limit $T_s/T_p \to 0$, the speedup is linear with respect to np and the algorithm scales infinitely well. For real algorithms, the ratio is finite and leads to a flattening of the speedup curve signifying diminishing returns for additional processors.

Figure 14 shows the speedup curves for the algorithm on two different architectures based on estimates of T_p and T_s from calculations with $O(128^3)$ grid points. The red curve gives results for a Beowulf cluster connected using an inexpensive 100-Mbit ethernet switch. The curve suggests that, for this algorithm, additional processors beyond about 30 yield only slight improvements in performance. On this cluster, the ratio $T_s/T_p \approx 0.0396$.Similar analyses were performed for a 128-node, dual-processor, Pentium Linux cluster with a Myrinet switch at the Scripps Institution of Oceanography (SIO) and the Cray T3E at the Texas Supercomputer Center (TACC). Both communications switches are significantly superior to a simple ethernet switch and so improvements in parallel scalability are expected. The results show that increasing the speed of communications produces a smaller T_s/T_p ratio and a wider range in np before the speedup curve flattens. Figure 14 shows that the algorithm scales reasonably well on the T3E to a few hundred processors. One might choose to run a given problem on more processors if memory limitations were the primary constraint, but the results show that additional processors beyond a few hundred would contribute relatively little in reducing the execution time.

Acknowledgments. This work was supported by the National Science Foundation (OCE 0242471) and the U.S. Office of Naval Research (Code 322, Physical Oceanography, Grant N0014-96-1-0616). We gratefully acknowledge the significant intellectual contribution to

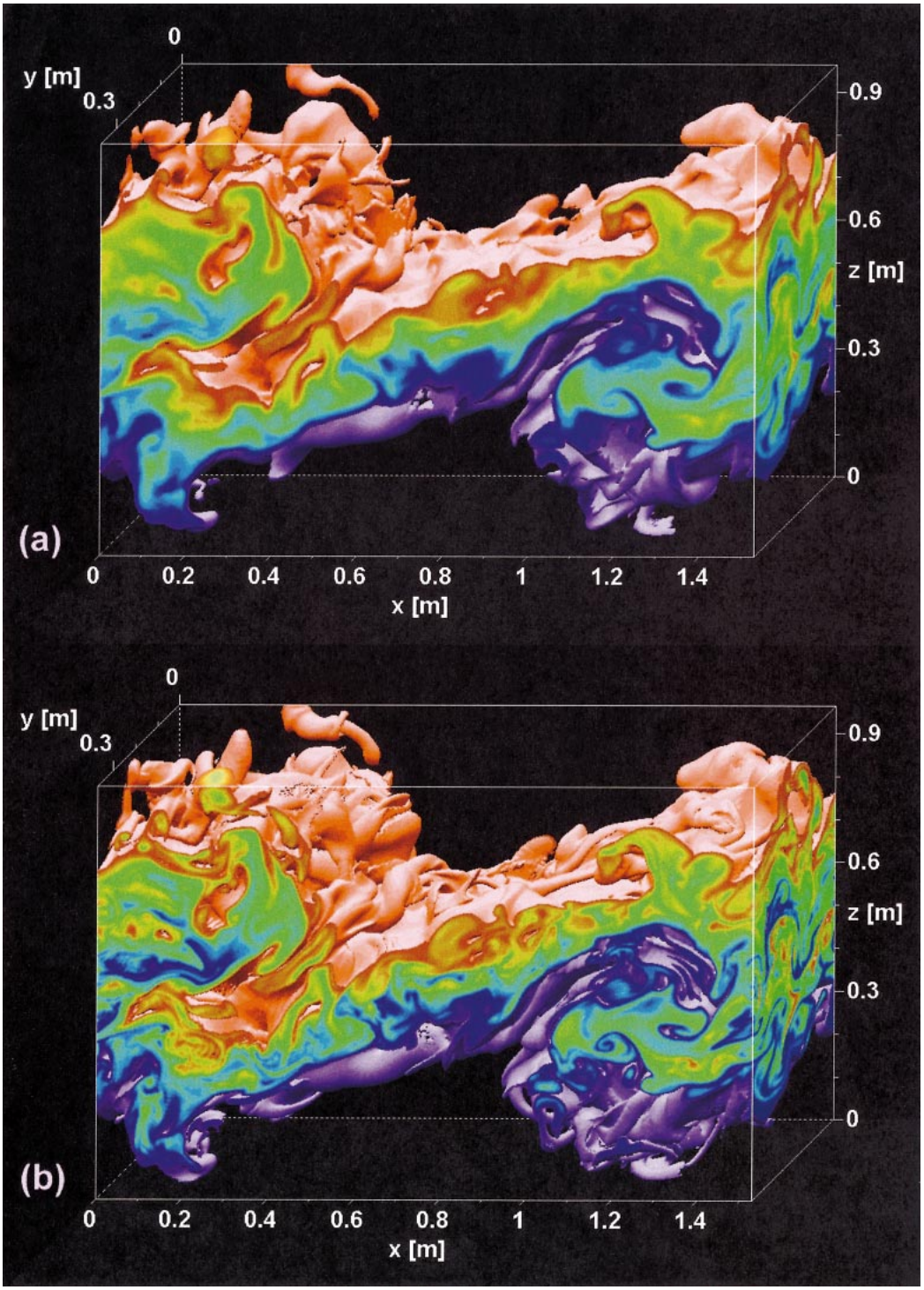

FIG. 12. Turbulent flow in a collapsing KH billow. The instability was driven by a maximum shear of 0.035 s^{-1} across a layer of initial depth 0.22 m. The viscosity was 0.995×10^{-6} m^2 s^{-1}, so the Reynolds number based on the total thickness of the shear layer was 1700. The flow carried two scalars differing only in their diffusivities: (a) 5.7×10^{-7} m^2 s^{-1} and (b) 1.4×10^{-7} m^2 s^{-1}. Density was controlled by the less-diffusive scalar such that the minimum gradient Richardson number was 0.04. The more diffusive scalar was resolved on a $192 \times 48 \times 120$ grid (with uniform spacing) as were the velocity and pressure fields. The less diffusive scalar was resolved on a $384 \times 96 \times 240$ grid. Interpolation between grids was done using FFTs.

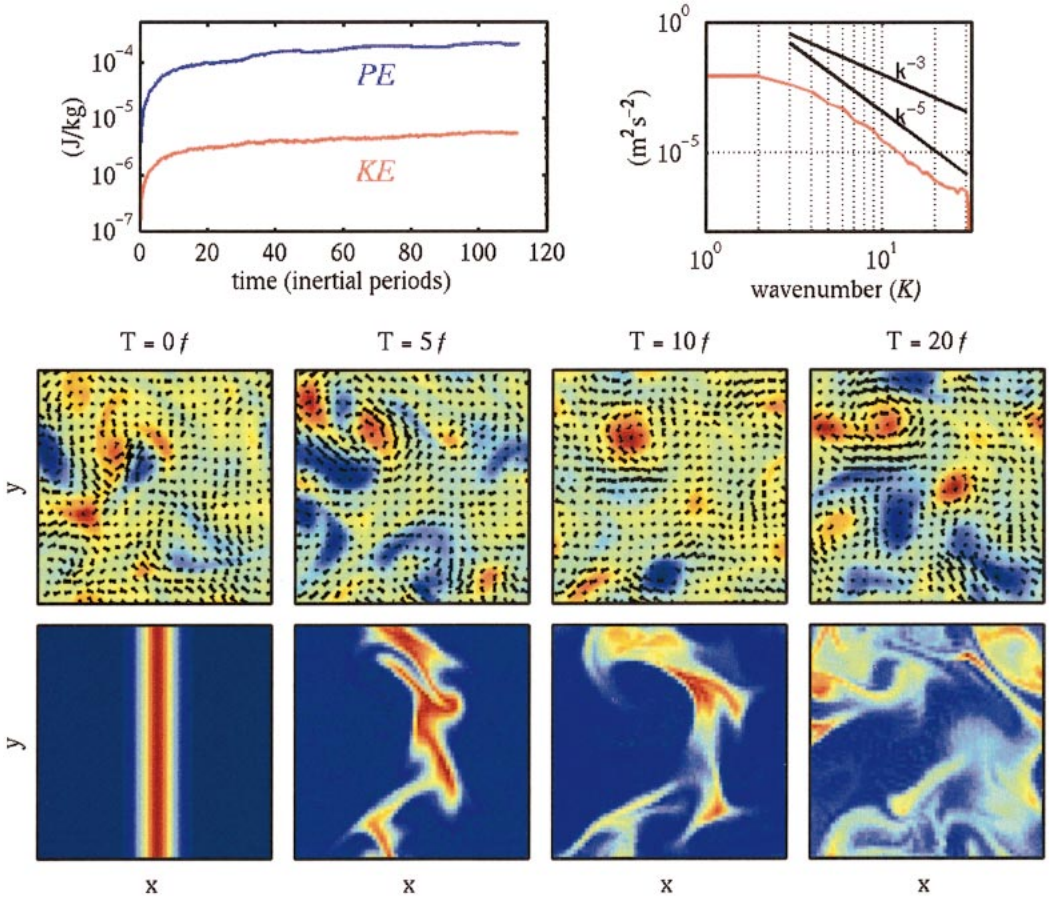

FIG. 13. (top left) Time series of PE and KE for a typical model spinup and equilibration, and (top right) KE spectrum after statistical equilibrium has been reached. Bottom two rows: plan views of potential vorticity with (subsampled) velocity vectors overlaid, and tracer at $t = 0$, 5, 10, and 20 inertial periods after KE equilibrium is reached.

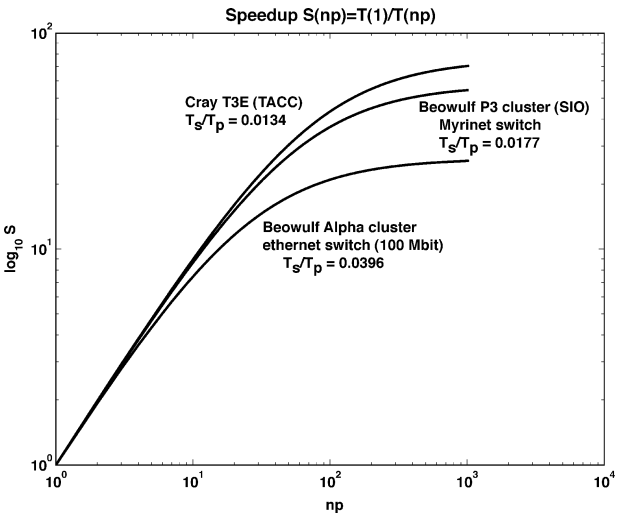

FIG. 14. Speedup characteristics on three facilities with different communications infrastructures.

this work by James J. Riley and thank Bill Smyth, Miles Sundermeyer, and Pascale Lelong for their valuable contributions.

APPENDIX A

Configuration for User-Defined Problems

The code is written in Fortran77 and makes calls to both the MPI and NETCDF libraries. The model has been succesfully run on the following platforms: single- and dual-processor Pentium workstations, Linux; a small cluster of Digital Equipment Corporation (DEC) Alpha workstations, Linux; a Cray T3E, Unicosmk; a 256-processor Pentium cluster, Myrinet, Linux; and a Macintosh PowerBook G4, OS X.

For our systems, we use the mpich implementation of MPI available from Argonne National Laboratory (information available online at http://www-unix.mcs.anl.gov/mpi/mpich). The NETCDF libraries are available from the National Center for Atmospheric Research (NCAR) (information available online at http://www.

TABLE A1. Parameters in input/problem_params.h.

Parameter	Value	Comments
netcdf_file	None	Initial conditions not read in from existing netcdf file (*filename*)
force_flag	No	No user-defined forcing terms added to equations of motion (*yes*)
bc_flag	zslip	Free-slip, rigid lid at $z = 0, L_z$ (*zperiodic*)
Lx	20 000	All fields periodic in x over scale *Lx*
Ly	20 000	All fields periodic in y over scale *Ly*
Lz	2000	z boundary conditions applied at $z = 0, L_z$
g	9.81	Gravitational acceleration in m s^{-2}
*x*lat	$\pi/4$	Latitude ϕ in rad (45°)
*x*Omega	$\Omega = 2\pi/(24 \times 3600)$	Earth's rotation frequency in s^{-1}
f	$f = 2\Omega \sin(\phi)$	Coriolis parameter
rho_0	1027	Reference density ρ_0 in kg m^{-3}
nu	1×10^{-6}	Fluid viscosity in m^2 s^{-1}
kappa	$1 \times 10^{-6}/0.75$	Diffusivity of density in m^2 s^{-1}
diss_flag	Isotropic	$\nu = \nu_{x,y} = \nu_z$, no stretching of dissipation operators (*anisotropic*)
p	2	Use standard Laplacian dissipation and diffusion (DNS)
T_diss	−999 999 99	Option unused for case isotropic, $p = 2$
DGRAD	$2 \times 10^{-6}(\rho_0/g)$	Characteristic scale of density gradient in kg m^{-4}
U0	0.01	Characteristic velocity scale in m s^{-1}
*b*freq	$\sqrt{(g/\rho_0) \times \text{DGRAD}}$	Derived quantity, do not alter
dt	46	Time step in s
*t*_start	0	Start time in s for integration
*t*_end	$4000 \times dt$	End time for integration
*t*_write_all	$50 \times dt$	Time increment in s between creation of output files
vort_flag	No	Do not write the three vorticity components to the output files (*yes*)
*t*_stat	$50 \times dt$	Time increment in s between calculation of volume-integrated statistics
float_switch	Off	Do not calculate trajectories of Lagrangian floats (*on*)
*t*_floats_on	*t*_start	Time in s at which floats are released, ignored if float_switch is off
*z*_close	$0.5 \times dz/Lz$	Min distance from $z = 0, L_z$ allowed for floats, ignored if float_switch is off
*z*_offset	$-0.5 \times dz/Lz$	Relative location of second float sensor, ignored if float_switch is off
delta_*w*	$0.0/U0$	Dimensionless residual velocity of imperfect floats
efactor	0	No random noise added to initial conditions

unidata.ucar.edu/packages/netcdf/). Our expectation is that, provided the MPI and NETCDF libraries are installed, the source code should compile and run on any UNIX-like system with a Fortran compiler. Because the model is designed to be run in a parallel environment, calls will be made to MPI even when the number of processors is set to one. The MPI library is therefore required even if the code is to be run only in a single-processor environment. Fortunately, the widely used implementations of MPI have been designed to function efficiently in the single-processor limit.

There are three main steps to configuring the model for a user-defined problem: configuring the parameter file, configuring the user-defined routines, and compiling the executable code. The procedure is explained here using the unforced internal wave example discussed in section 6b as an illustrative example. This configuration computes the time evolution of an internal wave mode in a rigid-lid, free-slip ocean using a snapshot of the linearized analytical solution as an initial condition.

a. Configure the main parameter file

The first step is to specify basic parameters such as the domain size, the boundary conditions, fluid properties, and run-time control parameters, for example, the time step, the total number of time steps in the integration, and how frequently to write the results to output files. The specification is accomplished by setting values in the text file input/problem_params.h, which are then incorporated during compilation.

Table A1 gives a brief description of each of the parameter settings for the default configuration. The file supplied with the code can be used as a template for developing new configurations. Comments in parenthesis give other legal values for text-based parameters.

If the parameter float_switch is set to ''on,'' an additional file called float_positions.dat must be placed in the input directory. Each line of the file must contain an integer float label (i.e., 1, 2, 3. . .) followed by the dimensional x, y, and z starting locations for each float. A sample file is included with the code.

b. Configure user-supplied subroutines

There are three user-supplied subroutines that require configuration prior to compilation. All three reside in the input directory. Again, the routines supplied with the code should be used as templates for constructing more elaborate definitions.

The first routine, rho_bar_func.f, is used to specify the ambient density profile $\overline{\rho}(x, y, z)$, its partial derivatives, and its Laplacian. The horizontal gradients of the corresponding hydrostatic pressure field are also specified here. Recall from section 2 that the total (potential) density is decomposed as

$$\rho = \rho_0 + \overline{\rho} + \rho'(x, y, z, t), \quad \text{(A1)}$$

where ρ_0 is a constant reference value. By splitting the remainder of the density field into a time-invariant part and a time- and space-dependent fluctuating part, the available resolution of the algorithm is focused on the relatively small unknown component rather than on a larger invariant component. Note that this is a mathematical decomposition rather than a purely physical one. The same problem can be solved using different choices of $\overline{\rho}$. Different fluctuating components will be solved for numerically but the combination $\overline{\rho} + \rho'$ will be the same (up to roundoff errors). The fluid is assumed to obey the Boussinesq approximation (fluctuations are much smaller than the characteristic value) and so ρ_0 is neither carried explicitly in the calculation nor included in the output data.

The routine below prescribes a linear density profile, that is, a fluid with constant buoyancy frequency N.

```
      subroutine rho_bar_func(xprime,yprime,zprime,rho_bar,
     *   rho_bar_x,rho_bar_y,rho_bar_z,g2rb,
     *   p_bar_x,p_bar_y,Lz,DGRAD,U0,rho_0)
c
c  *****input arguments xprime,yprime,zprime are DIMENSIONLESS as are the
c  *****output variables rho_bar,rho_bar_x,p_bar_x etc
c
c  (xprime,yprime,zprime):
c    d·less vertical position
c  Lz:
c    vertical domain length [m]
c  DGRAD:
c    characteristic density gradient used to scale variables [kg/m^4]
c  U0:
c    characteristic velocity scale used in nondimensionalization
c  rho_0:
c    reference density [kg/m3]
c  rho_bar, rho_bar_x, rho_bar_y, rho_bar_z, g2rb
c    density, and spactial gradients, g2rb=grad2(rho_bar)

c  N.B.
c  LZ,DGRAD, & U0 ARE OBTAINED BY THE CALLING MODULE
c  FROM THE INPUT FILE input/problem_params.h. THEY ARE
c  USED TO NONDIMENSIONALIZE THE RETURNED VALUES. YOU
c  MAY OR MAY NOT WANT/NEED TO USE THEM IN YOUR DEFINITIONS

      implicit none
      real xprime,yprime,zprime
      real Lz,DGRAD,U0,rho_0
      real rho_bar,rho_bar_x,rho_bar_y,rho_bar_z,g2rb,p_bar_x,p_bar_y

c  local variables
      real x,y,z

c  **** USER DEFINED DEFINITIONS START HERE **********************************
      x = xprime*Lz            ! use dimensional value to evaluate formulae
      y = yprime*Lz            ! use dimensional value to evaluate formulae
      z = zprime*Lz            ! use dimensional value to evaluate formulae

      rho_bar = DGRAD*(Lz-z)   ! linear stratification, fn of z only
      rho_bar_x = 0.0
      rho_bar_y = 0.0
      rho_bar_z = -DGRAD
      g2rb = 0.0

c  p_bar = -g rho_0 z - g* integral rho_bar (x,y,z') dz'
```

```
c  only the quantities p_bar_x and p_bar_y are needed
   p_bar_x=0.0
   p_bar_y=0.0

c  **** USER DEFINED DEFINITIONS END HERE ***********************************
c  return nondimensional values
   rho_bar=rho_bar/(DGRAD*Lz)
   rho_bar_x = rho_bar_x/(DGRAD)
   rho_bar_y = rho_bar_y/(DGRAD)
   rho_bar_z = rho_bar_z/(DGRAD)
   g2rb = g2rb/(DGRAD/Lz)
   p_bar_x = p_bar_x/( (rho_0*U0**2/Lz) )
   p_bar_y = p_bar_y/( (rho_0*U0**2/Lz) )

   return
   end
```

There are two ways in which the initial conditions can be prescribed. The first is used in the example configuration. Here, the values of the flow velocities and the perturbation density are calculated and specified in a user-defined subroutine, given the spatial location at which the values are required. The internal wave example here is relatively simple; it evaluates some straightforward analytical functions of position derived from the linearized internal wave equations. Much more complicated user-defined routines could be used, for example, using random number generators to introduce random components of flow at various scales. The routine user_defined_ics is called for each spatial grid point whenever the value of netcdf_file is set to ''none'' in input/problem_params.h. The user-defined routine need not be particularly efficient because it is called only once prior to the real computational workload of integrating the equations forward in time. Local variables may be defined and used for storing intermediate results provided they are declared with respect to type as shown in the following example:

```
   subroutine user_defined_ics(x,y,z,Lx,Ly,Lz,u,v,w,pd)
c
c  inputs
c    x,y,z                 positions in spatial domain              [m]
c    Lx,Ly,Lz              domain lengths in x,y,z                   [m]
c  outputs
c    u,v,w                 speeds in x,y,z directions at (x,y,z)    [m/s]
c    pd                    perturbation density                     [kg/m3]
c
c    rho_total=rho_0 + rho_bar(z) + pd(x,y,z)
c    rho_bar defined in user supplied rho_bar_func.f
c    rho_0 specified input/problem_params.h
   implicit none
   real x,y,z,u,v,w,pd
c  ***************************************************************************
c  **user declares any additional local variables to be used here
     real k_ics,l_ics,m_ics,omega2,omega_ics,A_ics,c(2,2),arg
     real ambient_density_gradient,Lx,Ly,Lz,N2_ics,pi,rho_0,g,f
c  ***************************************************************************

   pi=4.*atan(1.)

c  WAVE PARAMETERS FOR SIMPLE TEST WAVE
c    n.b. wave is defined in uniform stratification
c    ambient_density_stratification should be consistent with
```

```
c    the definition of rho_bar in rho_bar_func

   ambient_density_gradient = -(rho_0/g)*2e-6      ! [kg/m4]
   rho_0=1027.                                     ! [kg/m3]
   g=9.81                                          ! [m/s2]
   f=2*(2*pi/(24*3600))*sin(pi/4.)                 ! [1/s] 45 deg latitude

   A_ics=3.                                        ! displacement [m]
   k_ics=((1)*(2.*pi/Lx))                          ! [1/m]
   l_ics=((0)*(2.*pi/Ly))
   m_ics=((4)*(pi/Lz))
   N2_ics=(-g/rho_0)*ambient_density_gradient      ! [1/s^2]
   omega2 = ( ((f)**2)*m_ics**2 + N2_ics*(k_ics**2+l_ics**2) )/
   *  ( k_ics**2 + l_ics**2 + m_ics**2 )
   omega_ics=sqrt(omega2)                          ! (1/s)

c  constants in front of u & v expressions [m/s]
   c(1,1)=A_ics*k_ics*m_ics*omega_ics/(k_ics**2+l_ics**2)
   c(1,2)=A_ics*l_ics*m_ics*f/(k_ics**2+l_ics**2)
   c(2,1)=A_ics*l_ics*m_ics*omega_ics/(k_ics**2+l_ics**2)
   c(2,2)=A_ics*k_ics*m_ics*f/(k_ics**2+l_ics**2)

   arg=k_ics*x + l_ics*y        ! generally -omega_ics*t but set t=0

   w = A_ics*omega_ics*sin(m_ics*z)*sin(arg)
   pd =-A_ics*ambient_density_gradient*sin(m_ics*z)*cos(arg)
   u = c(1,1)*cos(m_ics*z)*cos(arg)
   *  -c(1,2)*cos(m_ics*z)*sin(arg)
   v = c(2,1)*cos(m_ics*z)*cos(arg)
   *  +c(2,2)*cos*(m_ics*z)*sin(arg)

   return
   end
```

A second method is to use the values in an existing NETCDF data file as initial conditions. This method is invoked whenever netcdf_file is set to the pathname of an existing NETCDF file rather than to none. We assume here that the NETCDF file is global, that is, that it contains initialization data over the full computational domain, and is consistent with the run for which it is being used to initialize [i.e., matching resolution and reference profile $\rho(\overline{z})$]. Local NETCDF files, that is, those containing only data allocated to individual processors, can be concatenated together using calls to the NETCDF libraries (or using other tools designed to manipulate NETCDF files). We require a global data file for initialization so that a given NETCDF file can be used to initialize new runs independent of the number of processors used to carry out the simulations.

When this second initialization method is chosen, the dimensional flow variables are extracted from the file, nondimensionalized using the scales specified in problem-params.h for the new run, and stored in arrays to be used as initial conditions. This technique can be used, for example, to restart a simulation from any saved output point.

Finally, the external forcing functions F_i (see section 2) need to be defined if the parameter force_flag is set to ''yes.'' The routine is not called, and need not be modified, if force_flag is set to ''no.'' The routine user_defined_forcing allows these functions to be specified as functions of space and time. The example below lists the routine used for the forced wave run discussed in section 6b. In this case, body forces are applied near the bottom boundary $z = 0$ with magnitude and phase chosen to project strongly onto a monochromatic, upward-propagating internal gravity wave. Efficiency is somewhat more critical in this case because the routine is called for each grid point at each time step. Compared to multidimensional transform operations, however, the computational workload is small, that is, $O(n)$ compared to $O(n \log n)$, where n is the total number of spatial grid points:

```
      subroutine user_defined_forcing(x,y,z,t,F1,F2,F3,F4,Lx,Ly,Lz,f,g,rho_0)

c     ***********************************************************************
c     User defined subroutine to add time/space dependent forcing to the
c     equations of motion. Inputs and outputs for this routine are all
c     DIMENSIONAL.
c

c     inputs:
c       x,y,z      spatial location at which forcing is to be defined, [m]
c       t          time at which forcing is to be defined              [s]
c       f,g,rho_0  Coriolis parameter, gravity, reference density [1/s,m/s2,kg/m3]
c                    f,g and rho_0 specified in input/problem_params.h
c     outputs:
c       F1-3       acceleration at x,y,z & t in x,y,z directions       [m/s2]
c       F4         rhs of perturbation density eqn                     [(kg/m^3)/s]
c                  careful with F4, you must take into account what you are
c                  using for rho_bar when you define this term
c     ***********************************************************************

      implicit none
      real x,y,z,t,F1,F2,F3,F4
      real Lx,Ly,Lz,f,g,rho_0

c     declare local variables
        real A_force,omega_force,k_force,l_force,m_force,b,F_of_z,Fprime,
     &    envelope_uv,envelope_wp,arg,pi
        real ambient_density_gradient,N2_force,omega2_force,F_of_t

c       define rhs values

c       properties of forcing wave, dimensional
c       it is most convenient to specify horizontal wavenumber and frequency

      pi=4.*atan(1.)
      ambient_density_gradient = -(rho_0/g)*2e-6      ! [kg/m4]
      N2_force=(-g/rho_0)*ambient_density_gradient    ! [1/s^2]

      A_force=3.                                      ! displacement [m]
      k_force=((1)*(2.*pi/Lx))                        ! [1/m]
      l_force=((0)*(2.*pi/Ly))
      m_force=((4)*(pi/Lz))
      omega2_force = (((f)**2)*m_force**2 + N2_force*(k_force**2+l_force**2) )/
     &  ( k_force**2 + l_force**2 + m_force**2 )
      omega_force=sqrt(omega2_force) ! [1/s]

      beta=0.2e-4                  ! width of guassian forcing wavenumber packet
      bfactor=1.0                  ! height of vertical amplitude window compared
                                     to m_force
      b=bfactor*m_force/(2.*pi)    ! characteristic height of vertical amplitude
                                     window
      F_of_z=exp(-(b**2)*(z**2))   ! Vertical amplitude window
      Fprime=-2.*(b**2)*z*F_of_z   ! derivative of vertical amplitude window

c       loop through wavenumbers and force in an envelope around k_force

      F1=0
      F2=0
```

```
F3=0
F4=0

do i=1,32
   k=(i*2.*pi/Lx)
   dk=2.*pi/Lx
   A=A_force*exp(-((k-k_force)/(sqrt(2.)*beta))**2)
   m=k/alpha
   l=l_force
   arg=k*x+l*y-omega_force*t
   envelope_uv=*F_of_z*cos(m*z)-(Fprime)*sin(m*z)  ! [] rhs forcing envelope
                                                      for u and v
   envelope_wp=*F_of_z*sin(m*z)
   F1=F1+A*(omega_force**2/k)*sin(arg)*envelope_uv
   F2=F2-A*(omega_force**2/k)*(f/omega_force)*cos(arg)*envelope_uv
   F3=F3-A*(omega_force**2/m)*cos(arg)*envelope_wp
   F4=F4+A*(omega_force/m)*(rho_0*N2/g)*sin(arg)*envelope_wp
enddo

return
end
```

c. Compile: Setup

Once the main parameter file and the user-defined routines have been configured, all that remains is to compile and run the code. The compilation, along with various organization tasks, is invoked by executing the Perl script setup with the appropriate switches. The routine setup performs several tasks. In particular, setup

- modifies source code so that arrays are properly dimensioned given the requested resolution and number of processors;
- constructs a makefile and compiles the code;
- creates an output directory and archives source and parameter files in a subdirectory codes_etc;
- creates a file input/float_positions.dat containing random starting locations for *nfloats* Lagrangian trajectories (the file can be edited if different starting locations are desired);
- calculates and reports run-time memory usage, comparing it to the memory available on the local machine;
- calculates and reports total disk space required for the output files, comparing it to disk space currently available in the requested output location; and
- constructs a short shell script *run* containing the command needed to launch a multiprocessor job under MPI.

The setup script processes information obtained from command line switches and from the system.conf file. A typical setup invocation is

```
./setup -np 4 -nx 128 -ny 128 -nz 128 -nfloats 3 -output_dir /work/kraig/
  spectral_model/floats_test.
```

This command configures and compiles the code for a four-processor run at a spatial resolution of (128 × 128 × 128), with three infinitesimal Lagrangian floats, with output sent to the specified location. The system_conf supplied with the code contains several example configurations for different computing environments. The file is used to specify locations of the NETCDF and MPI libraries, the Fortran compiler and desired compilation switches, and the syntax of the run command for MPI jobs. The example below can be used on machines with Intel processors running Linux, assuming that the libraries have been installed in the customary location/usr/local/lib. The syntax of the run command is appropriate for the mpich implementation of MPI. A log file will be written as the code is executed. We are assuming in this example that mpich has been configured for a cluster consisting of at least four processors. Changing the −np flag to 1 will generate code for a uniprocessor run:

```
# x86 linux g77. (NOTE: If using an f77 compiler, define F77 below, but do not
  define F90!)
FFLAGS = $FFLAGS -O3 -I/usr/local/include -ffixed-line-length-none -x f77-
  cpp-input
FLIBS = -L/usr/local/lib -lnetcdf -lmpich
F77 = g77
RUN_COMMAND = /usr/local/mpich/bin/mpirun -np NUMPROCS $PWD/flow_solve.x >&
  output/runlog &
```

APPENDIX B

Model Output

Results are output at a user-specified frequency in the form of NETCDF files. In a multiprocessor run, each processor writes files containing data over the portion of physical space allocated to that processor. The naming convention of the files is cfd3D_xxxxxx_yyy, where xxxxxx is a six-digit integer string corresponding to the time step at which the file was written and yyy is a three-digit string corresponding to the identification number of the processor that wrote the file. For single-processor runs the files are simply named cfd3D_xxxxxx. Each file contains data values for the velocity components u, v, and w; the perturbation density ρ'; the ambient profile $\overline{\rho}(z)$; and the pressure P. The files also contain values for the independent variables x, y, z, and t, along with descriptive labels and units for each variable. If the user-specified parameter vort_flag was set to ''yes'' during configuration, the three components of the vorticity vector will also be present.

There are many tools available to manipulate, view, or extract data from NETCDF files, though a detailed discussion is beyond the scope of this paper. We do mention a few particularly useful tools, however: Ncview (information available online at http://meteora.ucsd.edu/pierce/ncview_home_page.htmland) and NCO (for NETCDF operators; information online at http://nco.sourceforge.net). Ncview is a visual browser for data stored in NETCDF format. NCO allows operations such as extraction, appending, new file creation, and averaging to be applied to data stored in NETCDF files. We also import NETCDF files, either those written directly by our model or new files created by extraction or concatenation, directly into the commercial software package MATLAB for further analysis or display. A freely available NETCDF interface can be obtained from the U.S. Geological Survey (available online at http://woodshole.er.usgs.gov/operations/modeling/mexcdf.html).

A diagnostic file *energetics* is also written out containing estimates of the volume integrated kinetic and potential energies in addition to the energy transfer rates on the right-hand sides of Eqs. (57) and (58). The frequency at which the data are computed and stored is controlled by the parameter t_stat in problem_params.h. Note that t_stat should be specified small enough to allow reasonable estimates of the time derivatives of E_k and E_p. The file is formatted as shown in Table B1 with white-space-separated values. The energy balances can then be evaluated on a term by term basis. Generally, we have found that for well-resolved simulations, the balances are accurate enough that errors in estimation of the time derivatives and spatial integrals, neither of which affect the quality of the underlying simulation itself, dominate the residual.

TABLE B1. Format of *energetics* file.

Column	Quantity	Units
1	Time t	s
2	E_k	J kg^{-1} = m^2 s^{-2}
3	E_p	J kg^{-1}
4	$\frac{g}{\rho_0 V}\int_V \rho w \, dV$	W kg^{-1} = m^2 s^{-3}
5	$\frac{1}{V}\int_V \nu \mathbf{u}\cdot\nabla^2\mathbf{u} \, dV$	W kg^{-1}
6	$-\frac{g}{\rho_0 V}\oint_S gz\rho\mathbf{u}\cdot\hat{n} \, dS$	W kg^{-1}
7	$\frac{\kappa g}{\rho_0 V}\oint_S z\nabla\rho\cdot\hat{n} \, dS$	W kg^{-1}
8	$-\frac{\kappa g}{\rho_0 L_z}(\overline{\rho}_{\text{top}} - \overline{\rho}_{\text{bottom}})$	W kg^{-1}
9	$\frac{1}{V}\int_V uF_1 + vF_2 + wF_3 \, dV$	W kg^{-1}
10	$\frac{g}{\rho_0 V}\int_V zF_4 \, dV$	W kg^{-1}

REFERENCES

Bouruet-Aubertot, P., C. Koudella, C. Staquet, and K. B. Winters, 2001: Particle dispersion and mixing induced by breaking internal gravity waves. *Dyn. Atmos. Oceans,* **33,** 95–134.

Canuto, C., M. Y. Hussaini, A. Quarteroni, and T. A. Zang, 1988: *Spectral Methods in Fluid Dynamics.* Springer-Verlag, 557 pp.

Cooley, J. W., P. A. W. Lewis, and P. D. Welch, 1970: The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine and Laplace transforms. *J. Sound Vib.,* **12,** 315–337.

D'Asaro, E. A., K. B. Winters, and R.-C. Lien, 2004: Lagrangian

estimates of diapycnal mixing in simulated K–H instability. *J. Atmos. Oceanic Technol.*, in press.
Gill, A. E., 1982: *Atmosphere–Ocean Dynamics.* Academic Press, 662 pp.
Gropp, W. D., and E. Lusk, 1996: User's guide for mpich, a portable implementation of MPI. Mathematics and Computer Science Division, Argonne National Laboratory, ANL-96/6, 102 pp.
——, ——, N. Doss, and A. Skjellum, 1996: A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Comput.*, **22,** 789–828.
Lighthill, J., 1978: *Waves in Fluids*. Cambridge University Press, 504 pp.
Paparella, F., and W. R. Young, 2002: Horizontal convection is non-turbulent. *J. Fluid Mech.*, **446,** 205–214.
Slinn, D. N., 1995: Numerical simulation of turbulent mixing caused by internal wave reflection from sloping boundaries. Ph.D. thesis, University of Washington, 398 pp.
Smyth, W. D., and J. N. Moum, 2002: Anistropy of turbulence in stably stratified mixing layers. *Phys. Fluids*, **12,** 1343–1362.
——, ——, and D. R. Caldwell, 2001: The efficiency of mixing in turbulent patches: Inferences from direct simulations and microstructure observations. *J. Phys. Oceanogr.*, **31,** 1969–1992.
Spedding, G. R., 1997: The evolution of initially-turbulent bluff-body wakes at high internal Froude number. *J. Fluid Mech.*, **337,** 283–301.
Winters, K. B., and E. A. D'Asaro, 1994: Three-dimensional wave instability near a critical level. *J. Fluid Mech.*, **272,** 255–284.
——, P. N. Lombard, J. J. Riley, and E. A. D'Asaro, 1995: Potential energy and mixing in density-stratified flows. *J. Fluid Mech.*, **289,** 115–128.