U.S. Department of Commerce
National Oceanic and Atmospheric Administration
National Weather Service
National Centers for Environmental Prediction
5200 Auth Road
Camp Springs, MD 20746-4304

Office Note 434

PARALLEL IMPLEMENTATION OF COMPACT NUMERICAL SCHEMES

Tsukasa Fujita*

R. James Purser†
Environmental Modeling Center

July 2001

* UCAR Visiting Scientist Program. On leave from the Japan Meteorological Agency; e-mail: fujita@npd.kishou.go.jp

† Science Applications International Corporation, Beltsville, Maryland; e-mail: jpurser@ncep.noaa.gov

Abstract

The formulation of compact schemes in a parallel environment is described in detail for the case of differencing. Compact numerical schemes are spatially implicit methods for obtaining spatial derivatives and partial integrations, interpolating from one grid to its staggered counterpart, and so on, where the target results occupy a regular grid as dense as the regular grid of source data. Among the numerical schemes to perform such operations at a given formal order of accuracy, the compact forms usually yield the smallest principal truncation error coefficients and, when properly coded for a shared-memory machine, require essentially minimal extra computational cost. However, the inherently recursive nature of the application of compact schemes poses a problem for their efficient implementation on distributed-memory machines, which we set out to address in this note.

Generally, a compact scheme is expressed for an entire line of data simultaneously by a linear system characterized by banded matrices on both sides of the equation. The solution involves a preliminary "L-D-U" decomposition of the system matrix on the left-hand side of the equation, where L and U are lower and upper triangular banded matrices with unit main diagonals, D is a diagonal matrix. For the common case in which the original system matrix is symmetric, factors L and U are the transposes of each other and the decomposition is a modified Cholesky factorization. The components of the resulting factors provide the coefficients for a pair of one-sided recursions required by the subsequent application of the scheme to lines of data. For unbounded (effectively infinite or cyclic) lines of data on a uniform grid, the recursions simplify to the extent of having constant coefficients and it is convenient to retain this simplicity, by appropriate manipulations of the scheme's end conditions, even when finite boundaries intrude. We show, using three different implementation strategies, that each such constant-coefficient recursion can then be carried out in a reasonably efficient manner across several processors of a distributed-memory computer although the cost now becomes significantly greater than the cost of applying the conventional difference scheme of the corresponding order of accuracy. Nevertheless, the compact schemes remain viable within a massively parallel atmospheric model where high-order numerics are desired. Spatial filters with recursive components, such as the various Butterworth filters, involve similar numerical considerations and, as we show, may be handled in a parallel computing environment in the same way.

1. Introduction

Numerical differencing is one of the most important aspects of atmospheric simulation models, for its truncation errors have a significant influence on the fidelity of the solution. The spectral method (Bourke 1972) gives a good solution in global models and in some limited area models in which the grid geometry is amenable to efficient spectral decomposition, but for the massively parallel computers recently developed and becoming more popular, finite differencing on grid coordinate systems, by eliminating the need to carry out spectral decompositions, might be a better choice for efficiency. In that case, preserving the quality of numerical differencing makes the adoption of high-order schemes very attractive, provided such schemes, with

their additional boundary conditions, can be accommodated efficiently within parallel computer architectures.

A compact scheme (Lele 1992) would be a possible choice in this situation. While in the conventionally used differencing schemes one approximates a derivative value by referring only to source values centered around the target point, in the compact schemes one uses a spatially 'implicit' stencil to equate linear combinations of both the derivatives and the source data (Purser 1998). As we show below, the truncation errors of compact schemes are considerably smaller than those of the conventional schemes of the same order. But, just as implicit schemes for time integration each result in a system of equations for the horizontally-distributed variables of a dynamical model, a compact differencing scheme similarly yields a system of equations for the target derivatives distributed along a grid line. When this line spans several processors, then, for the compact scheme to remain competitive compared to its conventional counterpart, a sufficiently efficient solution of the implied recursions generally requires special algorithmic techniques. Having established such techniques, they can be applied in other kinds of recursive numerical processes. These include integration, interpolation to a staggered grid and selective low-pass spatial filtering.

Compact methods have a special role to play in the construction of high-order accurate, fully conservative, semi-Lagrangian interpolation schemes. The lack of automatic conservation of first-order quantities (such as mass) in traditional formulations of semi-Lagrangian time-integration schemes (see Staniforth and Côté 1991) has frequently been held up as one of the major inherent defects of the semi-Lagrangian method as traditionally formulated. The problem occurs even in one dimensional interpolation between a pair of grids, identical except for a smooth but inhomogeneous distribution of stretching and compression of one grid relative to the other, such as will occur to a Lagrangian coordinate grid relative to the fixed Eulerian grid over the course of one time step. The direct interpolation of a density from one grid (the source grid) to the other (the target grid) by using, for example, Lagrange polynomial interpolation, does not generally conserve any objective measure of the total mass, regardless of the order of accuracy employed. Moreover, the discrepancies are typically amplified further in two or three dimensions.

Compact schemes come to the rescue in the following way. Assume we have expressed the original data distribution as a density per unit source grid coordinate. As the first step, we apply a compact integration scheme to evaluate the partial integrals, in effect, constructing the 'cumulative' distribution of the same data. At any given point, the cumulative represents the total mass on one side of this point. It is this cumulative distribution that we interpolate to the target grid and, in order to recover a coordinate-relative density, we complete the process by applying a compact differentiation to the interpolated cumulative. Provided the grid for the density and for the associated cumulative are mutually staggered, then at any order of accuracy, the compact differencing method has an exact numerical inverse expressible in terms of the correspondingly accurate compact integration scheme (without the mutual staggering, the two-grid wave constitutes a null-space of the differencing operator, preventing its stable inversion). The density distribution is conserved not just globally, but in a quasi-local way, when the integration and differencing steps in the modified interpolation process are such a pair of associated compact schemes. As shown by Purser and Leslie (1991), the multi-dimensional grid-to-grid interpolations of a semi-Lagrangian model can be profitably decomposed into a

sequence of separate stages, each treating one dimension of the problem at a time in the so-called 'Cascade interpolation' technique. The conservative form of interpolation incorporating mutually inverse compact integration and differencing schemes in the manner we have defined was successfully exhibited in the semi-Lagrangian model described by Leslie and Purser (1995) and seems to offer one of the easiest ways to achieve exact conservation in a semi-Lagrangian model of arbitrary formal order of accuracy.

In this note, we describe ways by which compact schemes can be realized in a parallel environment, specifically, on a computing platform adopting the Message Passing Interface (MPI). We begin with an assumed unbounded extension of the grid space, which simplifies the coefficients of the algorithm for parallel processing. We first focus on numerical differencing by the compact schemes in order to exemplify the main techniques involved, but we also discuss the other numerical operations of integration, midpoint interpolation and implicit low-pass filtering, which likewise benefit from the application of similar recursive procedures. Section 2 reviews the derivation of the coefficients of various numerical differencing and interpolation stencils using modified Vandermonde matrix systems and compares the accuracy of both conventional and compact formulations of these numerical operations, tabulating the coefficients of the various schemes for the first few orders of accuracy in each case in order to demonstrate the superior accuracy achieved by the compact versions. We also provide some discussion here of the related implicit technique of recursive Butterworth filtering, using the 'sine-Butterworth' filter as an example. Section 3 discusses the steps involved in implementing compact schemes and recursive filters on a serial processing (shared-memory) architecture. Then, in Section 4 we show how calculations proceed in the three distinct approaches to parallelization that we are comparing. The first approach makes no assumption concerning the scale of influence of the recursive operations involved. The second approach, which seems especially attractive for compact schemes with relatively small effective scales of influence, uses information about the rate at which the exponential 'tails' of the various recursive processes decay in order to economize on the amount of computation. The third approach simplifies the coding aspects of dealing with recursive processes by performing a global rearrangement ('transpose') of the data to ensure all mutually dependent data reside in the same processor for each operation. In Section 5 we present some results of timing experiments that provide some insight into the way in which such things as domain and subdomain dimensions, data quantity and the specific compact schemes themselves affect the performance of these schemes in the three approaches to parallelization. Finally, we discuss the implications of these experiments for numerical models of the atmosphere.

2. Conventional and compact numerical formulations

(*a*) *Differentiation*

We first assume the source data and target derivative values are deployed on the same simple (unstaggered) uniform grid. In a conventional differencing scheme the derivative values collected along the active direction to form a vector $\mathbf{d}$ are expressed in terms of the source data, similarly collected into a vector $\mathbf{c}$, through a set of equations that we can combine through the appropriate banded matrix $\mathbf{B}$:

$$\mathbf{d} = \mathbf{B}\mathbf{c}. \tag{2.1}$$

The corresponding compact schemes simply generalize this explicit formulation to an implicit one through the introduction of another banded matrix $\mathbf{A}$ on the left side of the equation (together with a *different* $\mathbf{B}$ on the right, of course):

$$\mathbf{A}\mathbf{d} = \mathbf{B}\mathbf{c}. \tag{2.2}$$

Rows of $\mathbf{A}$ and $\mathbf{B}$ are constructed from the stencil coefficients that characterize each scheme. For example, a stencil for a popular fourth-order compact scheme on an unstaggered grid (e.g., Navon and Riphagen 1979, Lele 1992) is written:

$$\frac{1}{6}(d_{n-1} + 4d_n + d_{n+1}) = \frac{1}{2h}(-c_{n-1} + c_{n+1}), \tag{2.3}$$

where d_n is the derivative estimate of source value c_n at grid point n, and h is the distance between each grid point, here assumed to be constant. For this particular example, $\mathbf{A}$ is a symmetric tridiagonal matrix with the generic entries of $(\frac{1}{6}, \frac{2}{3}, \frac{1}{6})$ and $\mathbf{B}$ is a tridiagonal matrix with $(-\frac{1}{2h}, 0, \frac{1}{2h})$ (first and last rows involve the boundaries and generally warrant special treatment, as we discuss in the next section). The sum of the bandwidths of $\mathbf{A}$ and $\mathbf{B}$ is what determines the attainable order of accuracy of such a scheme. It is convenient to index each possible centered scheme using the notation (p, q) to denote the scheme with p components of $\mathbf{A}$ and q components of $\mathbf{B}$ on each side of the points of symmetry of their respective rows, excluding the symmetry point itself. The order of accuracy in each case is $2(p + q)$. For example, by this notation, the compact scheme of (2.3) is indexed $(1, 1)$ and attains an order of accuracy of $4 \equiv 2 \times (1 + 1)$. Conventional explicit centered schemes are then just those of the form $(0, q)$, in which we can take the now redundant matrix $\mathbf{A}$ to be simply the identity. For example, the conventional fourth-order scheme, $(0, 2)$, for differencing on a simple grid has the following stencil:

$$d_n = \frac{1}{12h}(c_{n-2} - 8c_{n-1} + 8c_{n+1} - c_{n+2}), \tag{2.4}$$

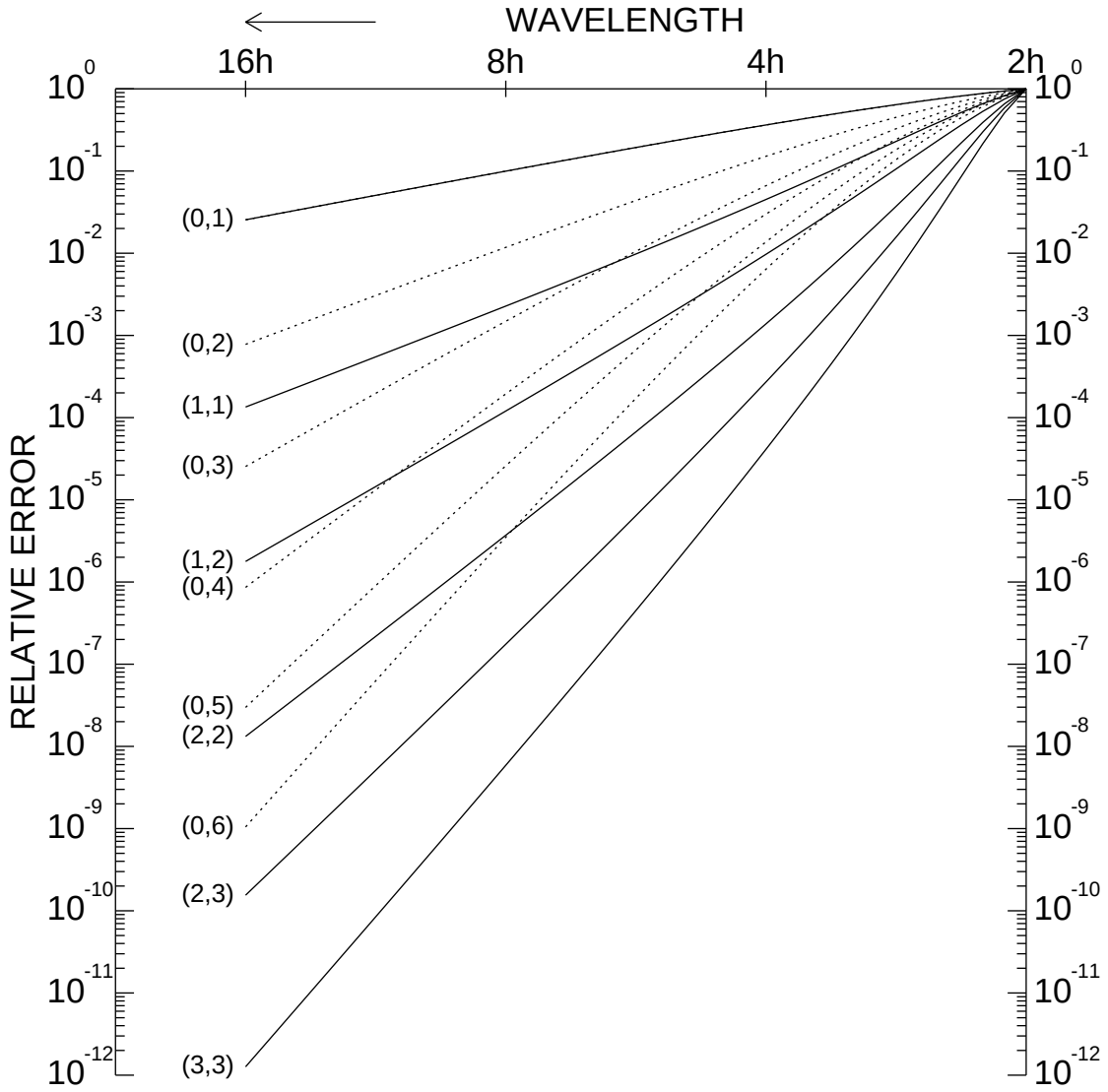

Figure 1. Log-log plot of the relative truncation error of unstaggered centered stencil differencing schemes as a function of the wavenumber of the sine-wave to which they are applied.

Figs. 1 and 2, which are similar to Figs. 1 and 2 of Purser (1998), show the relative truncation errors of a few conventional and compact unstaggered and staggered differencing schemes, each identified by its (p, q) indices.

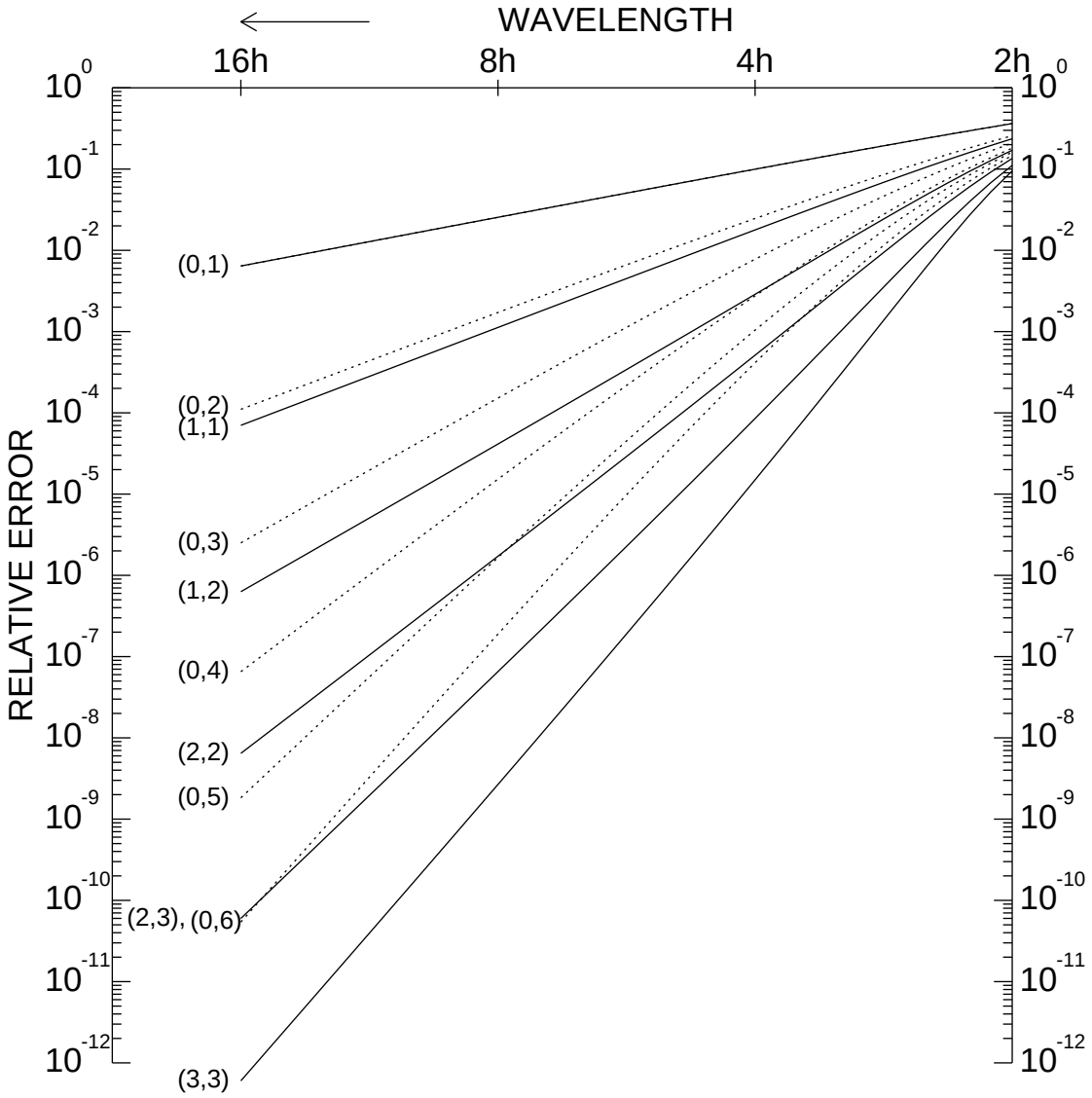

Figure 2. Same as Figure 1 but for centered schemes staggered with respect to the original data.

Concise analytic formulae for the coefficients of conventional centered differencing and midpoint interpolation schemes are given in Purser and Leslie (1988), but the coefficients of the more general partially-compact schemes do not appear to emerge from any simple algebraic formulae. Instead, we adopt the defining criterion that each scheme acts upon as many possible powers of $(x - x_n)/h$ without error, for x_n at or near the center of the stencil. The result is a modified form of Vandermonde linear system for the **A-** and **B-**coefficients. That is, we set $\mathbf{d}$ and $\mathbf{c}$ to exact values of the test functions in order to have a set of equations for the

row-coefficients of **A** and **B**:

$$\sum_i a_i \frac{d}{dx}\left(\frac{x-x_n}{h}\right)^m\Bigg|_{x=x_i} = \sum_i b_i \left(\frac{x_i-x_n}{h}\right)^m \qquad m=0,1,2,\cdots. \tag{2.5}$$

For the fourth-order case above, letting $n=0$ without a loss of generality, we find the following equation

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 0 & -1 \\ -2 & 0 & 2 & -1 & 0 & -1 \\ 3 & 0 & 3 & 1 & 0 & -1 \\ -4 & 0 & 4 & -1 & 0 & -1 \end{array}\right)\left(\begin{array}{c} a_{-1} \\ a_0 \\ a_1 \\ b_{-1}\times h \\ b_0 \times h \\ b_1 \times h \end{array}\right) = \left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right), \tag{2.6}$$

where, in row 1, we set an additional constraint on the sum of a_i being unity and use the test functions up to the fourth power to close the system of equations. For an unbounded domain the banded matrix **A** is generally symmetric about its principal diagonal and is positive-definite. In the case of unstaggered differencing, the symmetry point for a generic row of **B** is a matrix element but, in the cases of staggered differencing and for midpoint interpolation, the symmetry point is midway between two elements. About this symmetry point, **B** is symmetric if the operation is midpoint interpolation (or higher derivatives of even degree) but it is antisymmetric for first derivatives (or higher derivatives of odd degree). A characterization of the principal coefficient, ϵ, of truncation error is given for arbitrary smooth data, $c(x)$, by the defining identity for the scheme to be formally nth-order accurate:

$$d = \frac{dc}{dx} + \epsilon h^n \frac{d^n}{dx^n}\left(\frac{dc}{dx}\right) + \mathcal{O}(h^{n+1}), \tag{2.7}$$

in the case of estimates of the first derivative, dc/dx. Analogous expressions are found for midpoint interpolation and for derivatives of higher degree.

Appending an additional row of higher degree to (2.6) and evaluating the magnitude of the associated residual quantity from the left side of this extra row of the matrix equation is, apart from a factorial term, the coefficient, ϵ, for this scheme. For some numerical operations, this error coefficient obeys a simple algebraic expression. For example, the error coefficient $\epsilon_{(p,q)}$ for the unstaggered derivative of formal order of accuracy $n=2p+2q$ obeys:

$$(-)^{1+n/2}\epsilon_{(p,q)} \geq \left[(n+1)\binom{n}{n/2}\binom{n}{2p}\right]^{-1}, \tag{2.8}$$

with equality in the case, $q \geq p$, which makes the most accurate scheme of given order, n, the one for which $p=q=n/4$ or, if $n/2$ is odd, $p=q-1=(n-2)/4$. The combined stencil is then indeed as 'compact' as it can be made subject to attaining the required order of accuracy. If we pose the question, how great a resolution increase is required at a given order of accuracy to make the conventional differencing formula asymptotically as accurate as the corresponding compact scheme, then the answers for $n=4,6,8,10,12$ are factors of resolution increase of approximately, 1.57, 1.57, 1.70, 1.71, 1.77, respectively. By an application of Stirling's formula

(e.g., Abramowitz and Stegun 1965) and using the 'equality' case of the formula (2.8) for the asymptotic error, the required resolution increase can be shown ultimately to approach two in the limit of arbitrarily high-order accuracy. These observations help to justify the adoption of compact differencing when a high order of accuracy is sought, provided the computational cost remains comparable with the cost incurred to implement the corresponding conventional schemes.

Tables 1 and 2 provide a comparison of the stencil coefficients and principal errors of the conventional, and best compact schemes respectively for centered differencing up to twelfth-order.

TABLE 1. Coefficients of centered conventional differencing on a uniform unstaggered grid.

Order	β	$b_1 2h\beta$	$b_2 4h\beta$	$b_3 6h\beta$	$b_4 8h\beta$	$b_5 10h\beta$	$b_6 12h\beta$	ϵ
2	1	1						$\frac{1}{6}$
4	3	4	-1					$\frac{-1}{30}$
6	10	15	-6	1				$\frac{1}{140}$
8	35	56	-28	8	-1			$\frac{-1}{630}$
10	126	210	-120	45	-10	1		$\frac{1}{2772}$
12	462	792	-495	220	-66	12	-1	$\frac{-1}{12012}$

Coefficients for negative indices are $b_{-*}=-b_*$.

TABLE 2. Coefficients of centered compact differencing on a uniform unstaggered grid.

Order	β	$a_0\beta$	$a_1\beta$	$a_2\beta$	$a_3\beta$	γ	$b_1 2h\gamma$	$b_2 4h\gamma$	$b_3 6h\gamma$	ϵ
4	6	4	1			1	1			$\frac{-1}{180}$
6	5	3	1			15	14	1		$\frac{1}{2100}$
8	70	36	16	1		21	16	5		$\frac{-1}{44100}$
10	42	20	10	1		630	425	202	3	$\frac{1}{582120}$
12	924	400	225	36	1	220	125	88	7	$\frac{-1}{11099088}$

Coefficients for negative indices are $a_{-*}=a_*$ and $b_{-*}=-b_*$.

(*b*) *Compact integration as the inverse of staggered differentiation*

Some of the stencil and error coefficients for staggered derivatives are given in Table 3 for conventional schemes and in Table 4 for the best compact schemes. In the staggered case, there does not appear to be a simple generic algebraic expression for the error coefficients of the compact schemes but, from those that are tabulated, we find that the asymptotic errors, while consistently smaller than those of the unstaggered compact schemes, are not dramatically so. If now we ask how much of an increase in resolution would be required to match the magnitudes

of asymptotic error of each corresponding compact scheme, then at orders 4, 6, 8 and 10 the unstaggered grid resolution would only need to be increased by factors of approximately 1.17, 1.19, 1.09 and 1.10, respectively. In other words, when we adopt the most accurate methods for a given order of accuracy, the advantage of a staggered arrangement of variables, which amounts to an effective doubling of resolution for conventional second-order schemes (see the errors for $n=2$ in tables 1 and 3), rapidly diminishes to insignificance as we pass to higher orders of accuracy.

However, the staggered grid schemes do possess one crucially important property that is exploited in the conservative cascade algorithm for semi-Lagrangian advection (Leslie and Purser 1995): their invertibility in the form of compact quadrature schemes. A consequence of this property is that the transformation of a conserved density $\mathbf{d}$ into its corresponding partial integral $\mathbf{c}$ which, in this context, might be descriptively referred to as the 'cumulative' representation, entails no ambiguity or loss of information. As we shall show shortly, the mutually inverse operations of integration and differentiation by compact formulae when the source and target grids are staggered can be performed efficiently and conveniently 'in place' without the need to supply distinct arrays for the input and output fields.

TABLE 3. Coefficients of conventional centered differencing on a uniform staggered grid.

Order	β	$b_{\frac{1}{2}}h\beta$	$b_{\frac{3}{2}}3h\beta$	$b_{\frac{5}{2}}5h\beta$	$b_{\frac{7}{2}}7h\beta$	$b_{\frac{9}{2}}9h\beta$	ϵ
2	1	1					$\frac{1}{24}$
4	8	9	-1				$\frac{-3}{640}$
6	128	150	-25	3			$\frac{5}{7168}$
8	1024	1225	-245	49	-5		$\frac{-35}{294912}$
10	32768	39690	-8820	2268	-405	35	$\frac{63}{2883584}$

Coefficients for negative indices are $b_{-*}=-b_*$.

TABLE 4. Coefficients of centered mutually-inverse compact differencing and quadrature on a uniform staggered grid.

Order	β	a_0	a_1	a_2	$b_{\frac{1}{2}}h\beta$	$b_{\frac{3}{2}}3h\beta$	$b_{\frac{5}{2}}5h\beta$	ϵ
4	24	22	1		24			$\frac{-17}{5760}$
6	80	62	9		63	17		$\frac{61}{358400}$
8	76160	51338	12228	183	46800	29360		$\frac{-69049}{6141542400}$
10	2951424	1731174	581100	29025	1366850	1515525	69049	$\frac{939109}{1396283277312}$

Coefficients for negative indices are $a_{-*}=a_*$ and $b_{-*}=-b_*$.

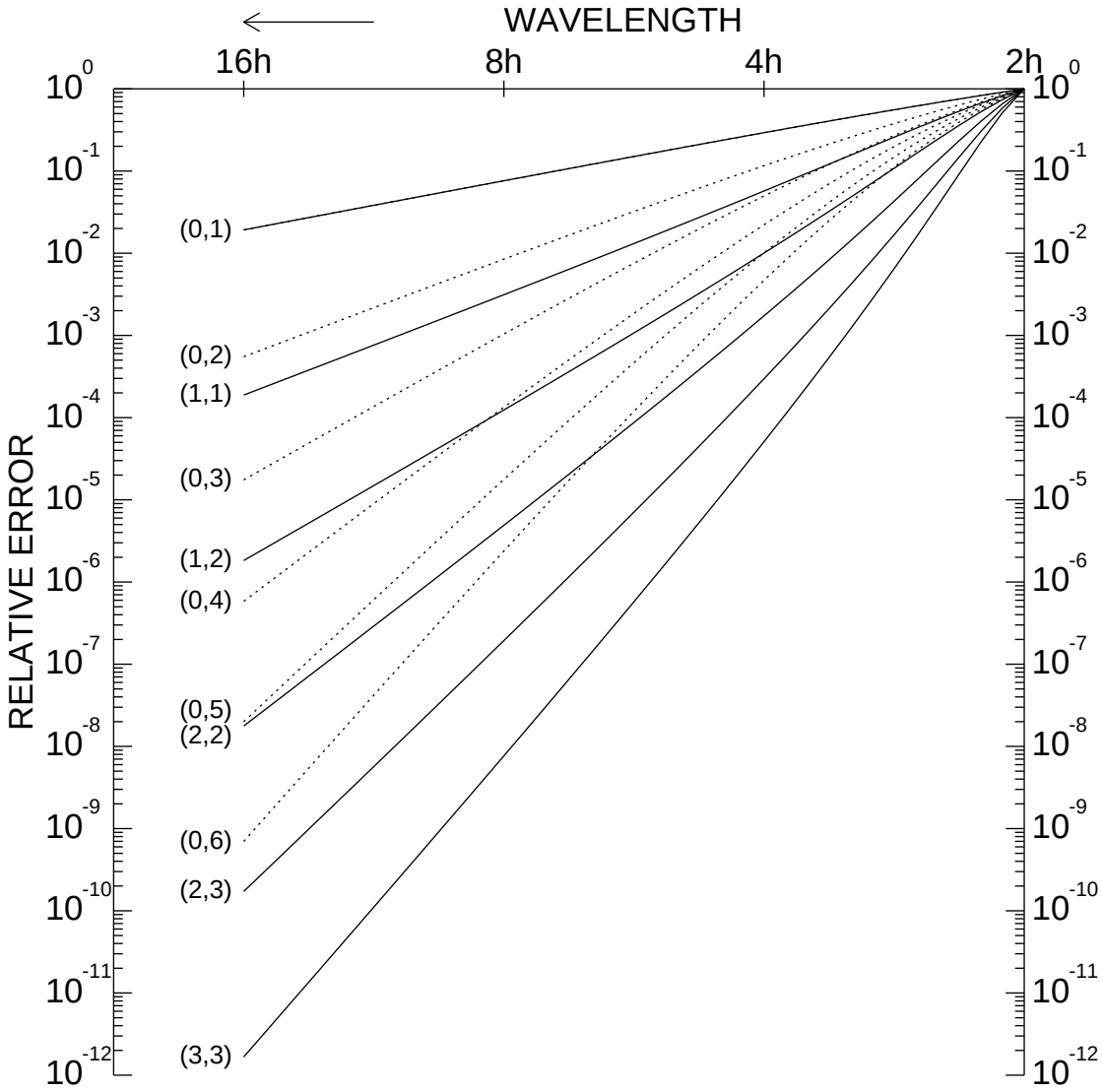

Figure 3. Same as Figure 1 but for the midpoint interpolation stencil.

(*c*) *Midpoint interpolation*

The compact formulation of interpolation superficially resembles that for staggered differencing (and some implementations may share a large proportion of the same code). Writing the vector of targets, $\mathbf{t}$, staggered with respect to the vector of source values, $\mathbf{s}$, then the centered formulae involve an odd number of $\mathbf{A}$-coefficients and an even number of $\mathbf{B}$ coefficients in each row of the matrix-vector expression,

$$\mathbf{At} = \mathbf{Bs}. \tag{2.9}$$

As before, a Vandermonde matrix system establishes the coefficients for each scheme of type (p, q), the order of accuracy again being $n = 2p + 2q$, and with a coefficient of error, $\epsilon_{(p,q)}$, defined:

$$t = s + \epsilon h^n \frac{d^n}{dx^n}(s) + \mathcal{O}(h^{n+1}). \tag{2.10}$$

The coefficient $\epsilon_{(p,q)}$ now satisfies the relatively simple algebraic formula:

$$\epsilon_{(p,q)} = (-)^{1+n/2} \binom{n}{n/2} \left[4^n \binom{n-1}{2p} \right]^{-1}. \tag{2.11}$$

TABLE 5. Coefficients of conventional centered midpoint interpolation.

Order	β	$b_{\frac{1}{2}}\beta$	$b_{\frac{3}{2}}\beta$	$b_{\frac{5}{2}}\beta$	$b_{\frac{7}{2}}\beta$	$b_{\frac{9}{2}}\beta$	$b_{\frac{11}{2}}\beta$	ϵ
2	2	1						$\frac{1}{8}$
4	16	9	-1					$\frac{-3}{128}$
6	256	150	-25	3				$\frac{5}{1024}$
8	2048	1225	-245	49	-5			$\frac{-35}{32768}$
10	65536	39690	-8820	2268	-405	35		$\frac{63}{262144}$
12	524288	320166	-76230	22869	-5445	847	-63	$\frac{-231}{4194304}$

Coefficients for negative indices are $b_{-*}=b_*$.

As in the case with differencing, the smallest coefficient $\epsilon_{(p,q)}$ at any given order n results from the choice of the most compact combined stencil. The stencil coefficients and principal errors for conventional midpoint interpolation and for the best compact schemes at up to twelfth-order can be compared in Tables 5 and 6. The stencil coefficients of the optimum compact schemes in this case are actually expressible simply as binomial coefficients:

$$\frac{1}{2^{n-1}} \binom{n}{n/2 + 2j} = \begin{cases} a_p & : \quad 2j \quad \text{even}, \\ b_p & : \quad 2j \quad \text{odd}. \end{cases} \tag{2.12}$$

A graphical comparison of the truncation errors for conventional and compact midpoint interpolation schemes is given in Fig. 3.

(*d*) *Implicit filtering*

In naively formulated meteorological simulations the energy cascade from larger to smaller scales frequently causes the shortest waves to accumulate too much energy, which can cause a nonlinear numerical instability. For low-order methods, the formulation of Arakawa-type advection schemes (for example, see Mesinger and Arakawa 1976) solves the problem, but this approach is generally not practical for high-order methods. Although numerical diffusion is one possible remedy, another solution that offers greater control over the spectral response is the application of carefully designed low-pass filters.

TABLE 6. Coefficients of compact midpoint interpolation.

Order	β	$a_0\beta$	$a_1\beta$	$a_2\beta$	$a_3\beta$	$b_{\frac{1}{2}}\beta$	$b_{\frac{3}{2}}\beta$	$b_{\frac{5}{2}}\beta$	ϵ
4	8	6	1			4			$\frac{-1}{128}$
6	32	20	6			15	1		$\frac{1}{2048}$
8	128	70	28	1		56	8		$\frac{-1}{32768}$
10	512	252	120	10		210	45	1	$\frac{1}{524288}$
12	2048	924	495	66	1	792	220	12	$\frac{-1}{8388608}$

Coefficients for negative indices are $a_{-*}=a_*$ and $b_{-*}=b_*$.

There are simple types of low-pass filters available, such as explicit (nonrecursive) and implicit (recursive) filters. However, for a given expenditure of computation, it is often the symmetric implicit filters that seem best fitted to the cases of concern because of their relatively sharp discrimination between the spectral region that is 'passed' and the unwanted spectral region that is substantially damped (Raymond and Garder 1991). Also, implicit filters have advantages in easily changing the degree of selectivity and by providing a continuous parameter to control the intended cut-off wavenumber, both being important parameters in practical use. As to the specific use in models with a terrain following vertical coordinate, a highly selective implicit filter can be profitably used to avoid spurious vertical transport caused by unintentional excessive smoothing in the vertical direction near orography (Raymond and Garder 1991).

Symmetric implicit filtering is carried out in practice by executing two one-sided recursive filters from opposite directions along each grid dimension separately. As Raymond (1988) shows, a multidimensional filter is possible and has been tested in numerical models. However, here we concentrate only on one-dimensional filtering.

We consider filters expressed in a matrix-vector notation analogous to (2.9) except that the target (filter output), $\mathbf{t}$, and the source (filter input), $\mathbf{s}$, now share the same grid. Both $\mathbf{A}$ and $\mathbf{B}$ are banded symmetric matrices containing the coefficients of the filter. For parameters (p, q, k_c) we set:

$$\mathbf{A} = \frac{\mathbf{C}^p}{(C_c)^p} + \frac{\mathbf{S}^q}{(S_c)^q}, \tag{2.13}$$

$$\mathbf{B} = \frac{\mathbf{C}^p}{(C_c)^p}, \tag{2.14}$$

where $(-\mathbf{S})$ is $h^2/4$ times the ordinary centered second-derivative operator for a grid of uniform spacing, h,

$$(\mathbf{Sy})_m = -\frac{1}{4}y_{m-1} + \frac{1}{2}y_m - \frac{1}{4}y_{m+1}, \tag{2.15}$$

while $\mathbf{C}$ is the basic '1–2–1' smoother, $\mathbf{C} \equiv \mathbf{I} - \mathbf{S}$:

$$(\mathbf{Cy})_m = \frac{1}{4}y_{m-1} + \frac{1}{2}y_m + \frac{1}{4}y_{m+1}, \tag{2.16}$$

and where the numbers S_c and C_c are the evaluations of the transfer functions of operators $\mathbf{S}$ and $\mathbf{C}$ at a 'critical wavenumber', the continuous parameter, k_c, chosen by the user. At this critical wavenumber, the transfer function of the complete filter has a value of one half. Suppose we apply operators (2.15) and (2.16) to a Fourier wave:

$$y_m = \tilde{y}(k)e^{imk}, \tag{2.17}$$

where k is the wavenumber in grid units (that is, h times the absolute wavenumber). Clearly, the responses are:

$$\mathbf{S}y_m = \sin^2\left(\frac{k}{2}\right)\tilde{y}(k)e^{imk}, \tag{2.18}$$

$$\mathbf{C}y_m = \cos^2\left(\frac{k}{2}\right)\tilde{y}(k)e^{imk}. \tag{2.19}$$

Hence, for input and output functions of the forms

$$s_m = \tilde{s}(k)e^{imk}, \tag{2.20}$$

$$t_m = \tilde{t}(k)e^{imk}, \tag{2.21}$$

the transfer function of this general form of discrete 'Butterworth filter' is

$$H_{(p,q,k_c)}(k) \equiv \frac{\tilde{t}(k)}{\tilde{s}(k)}, \tag{2.22}$$

$$= \frac{1}{1+\left(\frac{\sin(k/2)}{\sin(k_c/2)}\right)^{2q}\left(\frac{\cos(k_c/2)}{\cos(k/2)}\right)^{2p}}, \tag{2.23}$$

where k_c denotes the critical wavenumber in grid units.

Of the parameters, indices p and q prescribe the selectivity of the filter, q determining the degree of 'roll-off' from a transfer function response of unity at $k=0$, and p determining the analogous rate of departure from zero exhibited by the transfer function at the 'Nyquist' $2h$-wave limit. The continuous parameter k_c fixes the wavenumber at which the amplitude of a sinusoidal wave is diminished by a half. The special class with $p=0$ and $\mathbf{B}\equiv\mathbf{I}$ is traditionally referred to as the 'sine-Butterworth', while the class with $p=q$ is referred to as the 'tangent-Butterworth', owing to the special algebraic forms taken by the transfer function (2.23) in these cases.

Fig. 4 illustrates the theoretical amplitude responses of sine-Butterworth filters with q from 1 to 6 plotted against the discrete wavenumbers of a cyclic grid of 100 points and shows how the index q affects the steepness of the response near the selected critical wavenumber. Fig. 5 shows similar plots but with $q=4$ and with values of k_c corresponding to the indicated discrete wavenumbers ranging between 4 and 24. For this form of filter we see that the shortest waves are not completely removed. In numerical modeling applications where the filter is applied repeatedly, this is not a serious imperfection; for applications where the complete removal of the shortest wave *is* important a better choice of filter might be one with $p\neq 0$, such as the 'tangent-Butterworth'. Otnes and Enochson (1972) discuss these filters in detail and describe

some of the numerical ill-conditioning problems that may occur when both the index q and the cut-off wavelength are relatively large. To a lesser extent, these difficulties of ill-conditioning apply also to the other compact numerical schemes we have discussed when their order index, n, becomes excessively large. Fortunately, since a sufficient accuracy is usually attained with a more modest n, such difficulties do not normally cause problems.

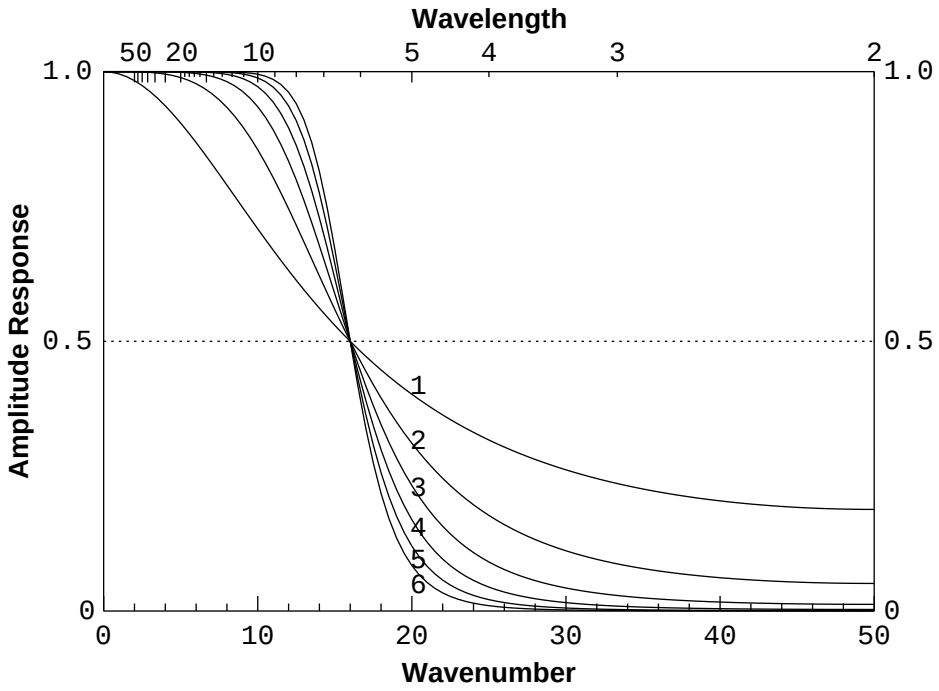

Figure 4. The amplitude responses of sine-Butterworth filters with different orders but the same critical wavenumber of 16.

3. Serial Implementation of compact schemes

(*a*) *Recursive solution of implicit operators*

Since $\mathbf{A}$ is positive symmetric, it can be decomposed as below (modified Cholesky decomposition)

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{3.1}$$

where $\mathbf{L}$ is a lower triangular matrix with unitary diagonal, $\mathbf{L}^T$ is its transpose, and $\mathbf{D}$ is a diagonal matrix. Because $\mathbf{A}$ is a banded matrix, the triangular factors $\mathbf{L}$ and $\mathbf{L}^T$ are also, and with the same band width defined by the order of accuracy of the differencing scheme. In an unbounded domain these factors act as commuting convolution operators, allowing us to recast (2.2) in the form,

$$\mathbf{L}\mathbf{L}^T\mathbf{d} = (\mathbf{D}^{-1}\mathbf{B})\mathbf{c}. \tag{3.2}$$

Having done this preliminary decomposition, the entire procedure for applying one of the compact operations can be reduced in general to the three steps summarized algebraically by

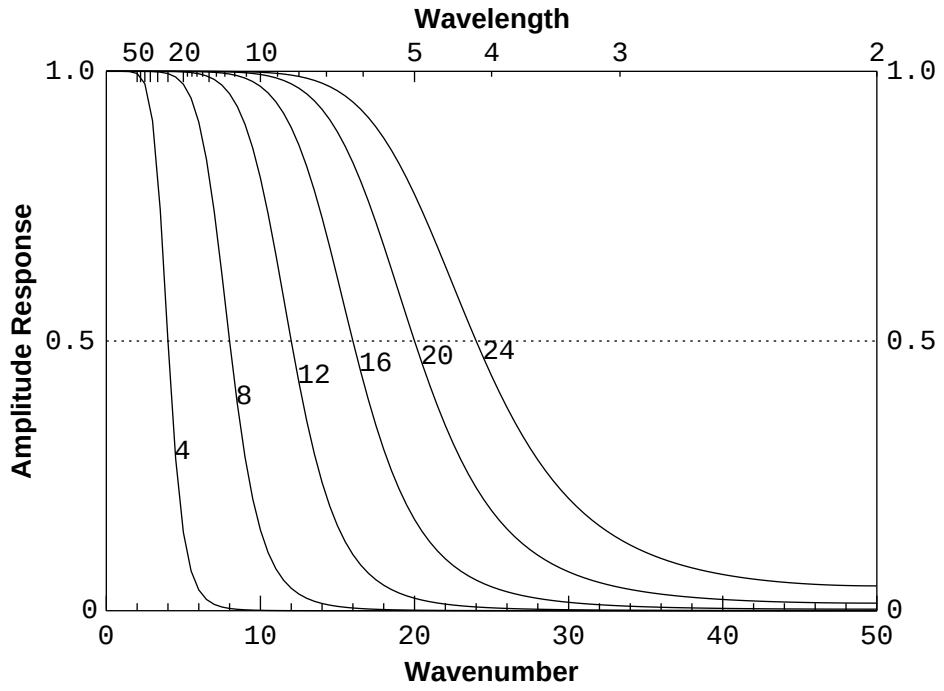

Figure 5. The amplitude responses of the $q = 4$ sine-Butterworth filter with different critical wavenumbers from 4 to 24.

the equations, (3.3), (3.4) and (3.5):

$$\boldsymbol{\chi} = (\mathbf{D}^{-1}\mathbf{B})\mathbf{c}, \tag{3.3}$$
$$\mathbf{L}\boldsymbol{\psi} = \boldsymbol{\chi}, \tag{3.4}$$
$$\mathbf{L}^T\mathbf{d} = \boldsymbol{\psi}, \tag{3.5}$$

where vectors $\boldsymbol{\psi}$ and $\boldsymbol{\chi}$ contain the results of intermediate calculations. Since the diagonal elements of $\mathbf{L}$ are simply unity, it turns out that the number of computations required to apply the compact scheme is essentially the same as for the corresponding conventional scheme in a serial environment. The difficulties, which we shall address in the next section, relate to the parallel implementation of the second and third steps, which are inherently recursive in nature. In the case of fourth-order unstaggered compact differencing, the three steps above become:

$$\chi_i = \gamma_1(c_{i+1} - c_{i-1}) \tag{3.6}$$
$$\psi_i = \chi_i - \alpha_1\psi_{i-1} \tag{3.7}$$
$$d_i = \psi_i - \alpha_1 d_{i+1}, \tag{3.8}$$

where, for this particular scheme, the constant coefficients are analytically defined:

$$\alpha_1 = 2 - \sqrt{3}, \tag{3.9}$$
$$\gamma_1 = 3\alpha_1. \tag{3.10}$$

In the general case, the invariant coefficients are obtained in practice through an iterative calculation which mimics the Cholesky decomposition of a matrix $\mathbf{A}$ of indefinitely large size,

the iterations proceeding until convergence of the factors' coefficients is achieved. We note that the recursive second step, (3.7), has to be performed by a sequence that progresses in the forward direction while the final step, (3.8) must proceed in the backward direction.

(*b*) *Staggered differencing and integration 'in place'*

The formalism we have just described applies also to the staggered differencing, midpoint interpolation and filtering operations discussed in section 2. However, for the case of the mutually inverse staggered differencing and partial integration operators, where the operations can be reversed at any time to recover the original data, it is sometimes convenient to adopt a numerical procedure that explicitly manifests this invertibility.

We begin by observing that, although the matrix $\mathbf{B}$ itself is not symmetric in the case of differencing and, in the staggered case, it does not even have a unique principal diagonal, we can recover both of these desirable properties by expressing it as the product:

$$\mathbf{B} = \mathbf{T}\mathbf{P}, \tag{3.11}$$

where $\mathbf{P}$ is the bidiagonal simple difference operator with square matrix representation:

$$\mathbf{P} = \begin{pmatrix} \ddots & \ddots & \ddots & \ddots & & \\ & 0 & -1 & 1 & 0 & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{3.12}$$

Assuming a uniform grid so that the coefficients of a representative row of $\mathbf{B}$ are:

$$(\cdots, 0, -b_{k-\frac{1}{2}}, \cdots, -b_{\frac{1}{2}}, b_{\frac{1}{2}}, \cdots, b_{k-\frac{1}{2}}, 0, \cdots), \tag{3.13}$$

we then find that $\mathbf{T}$ is a positive symmetric banded matrix with a representative row whose elements, t_i, for integer indices i centered on the principal diagonal element $i = 0$ satisfy:

$$t_i = \sum_{j>i} b_j. \tag{3.14}$$

We may now apply a modified Cholesky decomposition to matrix $\mathbf{T}$, so that performing a compact integration more directly resembles the other general compact operations we have described. While the final step of inverting the simple difference operation associated with $\mathbf{P}$ can use the accumulation operator, whose top-left corner defines the pattern:

$$\mathbf{P}^{-1} = \begin{pmatrix} 1, & 0, & 0, & 0, & \cdots \\ 1, & 1, & 0, & 0, & \cdots \\ 1, & 1, & 1, & 0, & \cdots \\ 1, & 1, & 1, & 1, & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{3.15}$$

in practice, this accumulation would be solved recursively like any other forward-progressing recursive filter.

There is an advantage in performing the modified Cholesky reduction of $\mathbf{T}$ in reverse order, so that the second recursive pass of the compact integration becomes a forward sweep that may be combined with forward-progressing accumulation operator, $\mathbf{P}^{-1}$. Thus, with $\mathbf{E}$ diagonal and $\mathbf{K}$ banded lower triangular, we may factorize $\mathbf{T}$:

$$\mathbf{T} = \mathbf{K}^T\mathbf{E}\mathbf{K} \tag{3.16}$$

Then, in the alternative sequence of three steps, which exhibits a remarkable symmetry when the sequence is reversed and the operations inverted, we have, for the staggered differencing steps:

$$\boldsymbol{\phi} = (\mathbf{KP})\mathbf{c}, \tag{3.17}$$

$$(\mathbf{LD})\boldsymbol{\psi} = (\mathbf{K}^T\mathbf{E})\boldsymbol{\phi}, \tag{3.18}$$

$$\mathbf{L}^T\mathbf{d} = \boldsymbol{\psi}. \tag{3.19}$$

Starting with $\mathbf{c}$, we apply (3.17) sequentially, but in the backward direction, which allows us to overwrite $\mathbf{c}$ with $\boldsymbol{\phi}$ as we go (we associate the relatively staggered grids here to make the operator $\mathbf{P}$ behave as a *lower* bidiagonal). The second step (3.18) is performed sequentially and progresses in the forward direction to allow $\boldsymbol{\psi}$ to overwrite $\boldsymbol{\phi}$. Note that the right-hand side of (3.18) is then explicit, since it refers to higher-indexed elements while the left-hand side is implicit, since it refers to the immediate target and lower-indexed values of the same target vector. The third step (3.19) is fully recursive and solved sequentially for the final derivative estimate, $\mathbf{d}$, by progressing in the backward direction once more, overwriting $\boldsymbol{\psi}$ in the process. Clearly, by reversing the order of these equations, reversing the direction of progression of the sequential steps in each, and transposing the designation of 'target' and 'source' vector in each step, we immediately have the requisite 'in place' algorithm by which the cumulative distribution $\mathbf{c}$ may be derived from the density $\mathbf{d}$. In cases where all the operators are of the diagonally-invariant 'convolution' type, then the placement of the diagonals, $\mathbf{D}$ and $\mathbf{E}$, is not critically important and it is often more convenient to combine them with the operator on the right-hand side of (3.17). For a bounded region with end conditions that preclude the convolution form, however, such a rearrangement is not possible and equations (3.17)–(3.19) must be solved as they stand.

(*c*) *End conditions*

For the non-periodic cases, extrapolation is required at both ends, in order to have $\mathbf{c}$ valid for the calculation of *all* the right-hand side of (3.3) and to provide exterior starting values of $\boldsymbol{\chi}$ and $\boldsymbol{\psi}$ to initiate the recursions implied by the left-hand sides of (3.4) and (3.5). As one of the simplest methods, we choose polynomial extrapolation, since it enables both an arbitrary 'halo' width of extrapolation and derivatives to be kept continuous to any desired degree. In the modeling context, the invention of halo values in this manner is no substitute for the provision of truly informative values that would normally be supplied from a larger scale model. However, there are occasions when exact halo values are neither readily available, nor strictly necessary for the production of satisfactory interior values from a compact numerical process. On such occasions, it is still important not to stimulate noise at the boundaries that

might later propagate into the interior to spoil the solution. Smooth extrapolation then proves to be a useful numerical tool, but its validity needs to be checked on a case by case basis. In preparation for the explicit steps, such as (3.3), the simple approach of polynomial extrapolation will suffice to define the necessary extrapolation for $\mathbf{c}$. But the preparation of corresponding haloes of exterior values of $\boldsymbol{\psi}$ and $\mathbf{d}$ requires a more sophisticated extrapolation in order to be consistent with the recursive steps (3.4) and (3.5).

Taking first the simple extrapolation of uniformly spaced data $c_i, \qquad i = 1, \cdots$ to indices $i \leq 0$, we shall find that the first k extrapolated values, together with the first m of the original data, will fit a polynomial of degree $m-1$ provided that

$$\sum_{i=0}^{m} \pi_i c_{j+i} = 0 \qquad j = 0, -1, ..., -k+1, \tag{3.20}$$

where

$$\pi_i = (-1)^i \binom{m}{i}. \tag{3.21}$$

We may solve (3.20) recursively for each j, starting adjacent to the boundary and progressing outwards or, if we wish, we may expand the recursion and express the extrapolated values directly in terms of the first m interior values:

$$c_{1-i} = \sum_{j=1}^{m} X_{i,j}^{(m)} c_j, \tag{3.22}$$

where each matrix element $X_{i,j}^{(m)}$ is defined:

$$X_{i,j}^{(m)} = (-)^{j+1} \binom{i+j+2}{j-1} \binom{m+i-1}{m-j}. \tag{3.23}$$

With trivial modification of the indexing, the same direct formula applies to the extrapolation at the opposite end of the domain:

$$c_{N+i} = \sum_{j=1}^{m} X_{i,j}^{(m)} c_{N+1-j}. \tag{3.24}$$

In the case of extrapolating the end values for initiating a recursive process, such as solving (3.4), it is now the *output* of the process that we need to extrapolate. We seek a way of expressing a sufficient number k of the extrapolated exterior values of $\boldsymbol{\psi}$ in terms of the existing first m interior values of the input, $\boldsymbol{\chi}$, of the recursion. We achieve this by simultaneously requiring the k extrapolated and the first m interior values of $\boldsymbol{\psi}$ to satisfy the k linear constraints implied by these values fitting the same polynomial of degree $m-1$, together with the additional m linear constraints implied by the first m interior values of $\boldsymbol{\psi}$ obeying the recursion formula defined by a generic row of (3.4). This gives us as many equations as unknowns. Setting $m = 3$ and $k = 2$,

for example, we have

$$\begin{pmatrix} 1 & -3 & 3 & -1 & 0 \\ 0 & 1 & -3 & 3 & -1 \\ \alpha_2 & \alpha_1 & 1 & 0 & 0 \\ 0 & \alpha_2 & \alpha_1 & 1 & 0 \\ 0 & 0 & \alpha_2 & \alpha_1 & 1 \end{pmatrix} \begin{pmatrix} \psi_{-1} \\ \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \chi_1 \\ \chi_2 \\ \chi_3 \end{pmatrix}, \tag{3.25}$$

and, by inverting the matrix involved to extract the relevant elements, we may then express the k-vector of extrapolated $\boldsymbol{\psi}$ in terms of the m-vector of first interior $\boldsymbol{\chi}$.

In the more general example of a mixed explicit-implicit equation, such as (3.18), then the initial extrapolation requires the appropriate modification of the approach exemplified by (3.25) (except the input vector must now incorporate the explicit part of the calculation contained on the right-hand side of (3.18)). The boundary treatment at the opposite end of the domain, where the recursive progression ends, is the extrapolation appropriate to the explicit part of the calculation and is therefore of the form (3.20).

Extrapolation naturally deteriorates the accuracy of the scheme, but the influence does not penetrate too deeply into the interior region for typical compact numerical operators unless a very high formal order of accuracy is chosen. Figure 6 shows how far the boundary effect reaches and how large it is for eighth-order differencing. Panel (a) shows the average precision of the eighth-order differencing against analytic derivatives for a sinusoidal wave of various wave lengths over a bounded region of 100 grid points on a staggered grid. The numbers in brackets designate the base-10 logarithms of the absolute errors. From this panel we see that the errors are becoming small for longer waves, but generally they are constant in the inner region for any wave lengths. The effect of boundary extrapolation is almost confined to the vicinity of both ends, while it penetrates somewhat more deeply into the interior region for longer waves. Panel (b) shows the relative errors for the second order derivatives derived by

$$d_n = c_{n+1} - c_n. \tag{3.26}$$

It shows the errors are sufficiently smaller than those of the second order scheme even in most of the regions close to the ends. As the order of accuracy of the scheme becomes higher, more of the interior region tends to be influenced (not shown). This deterioration also depends on the extrapolating function.

Such boundary effects are inevitable if we truly have no direct knowledge of the profile of the data beyond the domain given. In the case of Butterworth filtering, the effects can propagate further in, and to an extend that must at least approximately scale as $1/k_c$. For the periodic cases, no extrapolation is required and therefore no degradation of accuracy occurs. It is also possible, and sometimes quite appropriate, to solve the extrapolation dilemma through the imposition of conditions of mirror-symmetry or antisymmetry at the boundaries. However, with this approach, it is very difficult to preserve the convenient property of spatial invariance of the coefficients of the recursive processes.

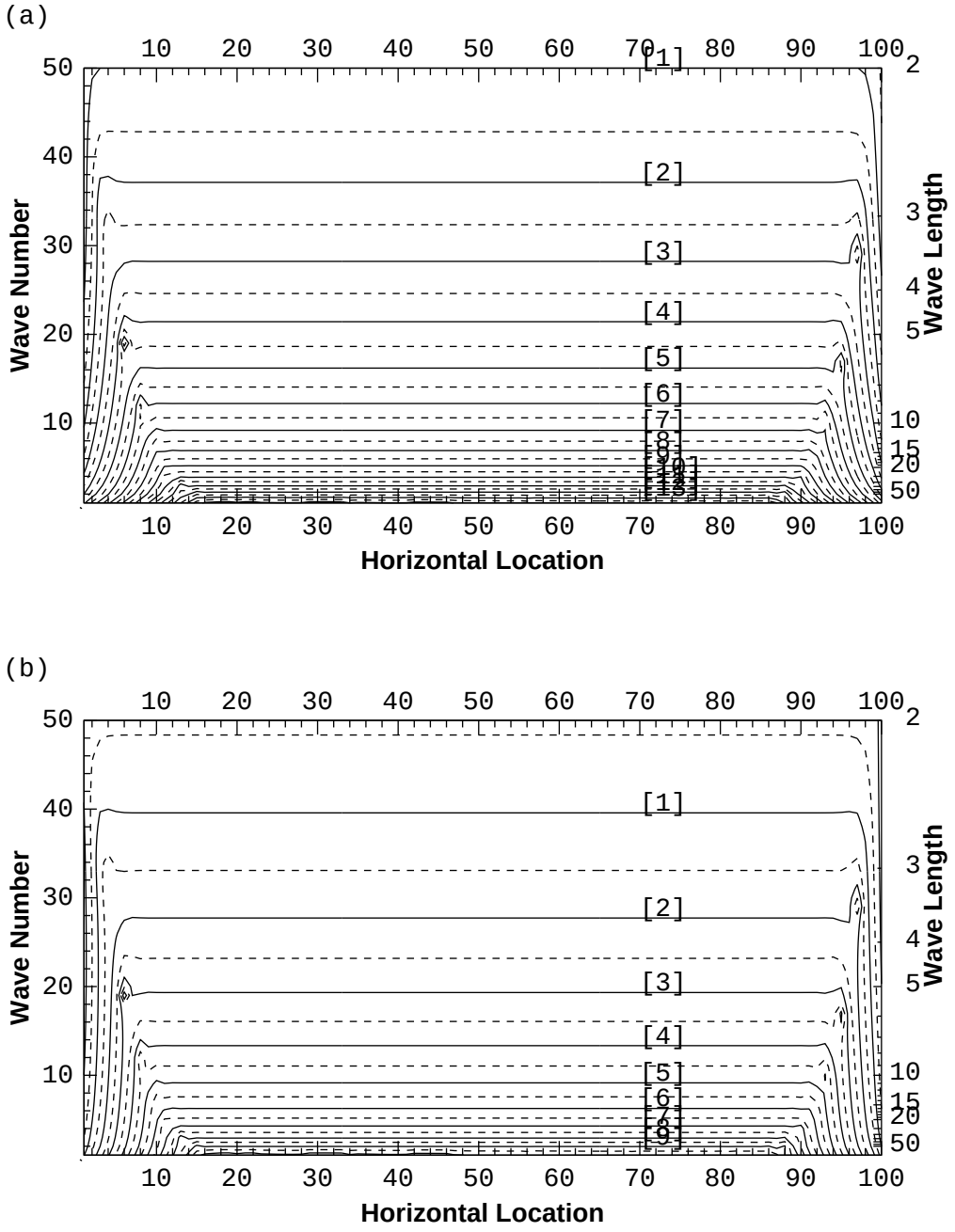

Figure 6. Accuracy of eighth-order differencing on a staggered grid in a 100-grid bounded region. The top panel shows the average errors of the scheme against the analytic solution in logarithm in the brackets with respect to wavenumbers 1 to 50 The bottom panel shows the average improvements of the scheme to the second order ordinary scheme. From these panels the substantial advantages of the compact scheme are clearly seen in the interior region, while the deterioration near the boundaries remains comparable to that found for the second order scheme.

4. Strategies for parallel implementation

The distribution of gridded data in a parallel environment is illustrated schematically in Figure 7. The entire region is usually divided into several subregions, and then, owing to the recursions inherent in the process of solving compact schemes or spatially-implicit filters,

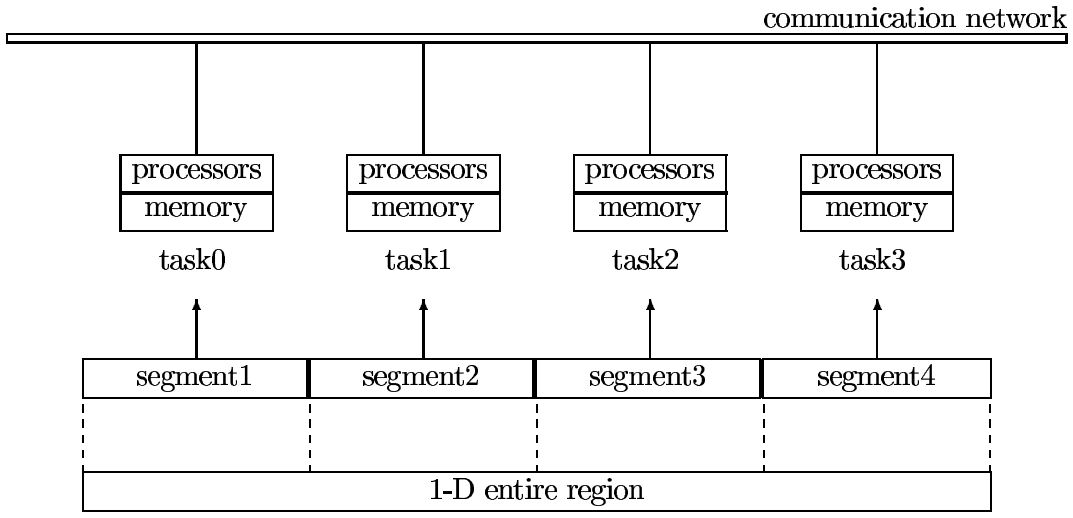

Figure 7. Concept of a parallel environment and region partitioning.

values in one subregion cannot be finally obtained until values in their neighboring regions have been set. Clearly, it is undesirable to have the efficiency of our most accurate finite differencing methods severely degraded by scheduling a line of processors to work only singly and sequentially on the recursive tasks of each numerical scheme, as the benefits of parallelization cannot then be realized. Instead, we must develop strategies by which the best use is made of every processor, if possible by having them simultaneously calculating useful values in each subregion in spite of the recursive dependency that superficially suggests the need for serial calculation.

To attain our purpose, we retain our assumption of diagonal invariance of the operators $\mathbf{B}$ and $\mathbf{L}$ that appear in the algorithm expressed by (3.3)–(3.5). The treatment of end conditions for bounded domains carries over to the parallel case essentially unchanged, but the treatment of the interior data must be planned with great care in order to extract the optimum computational efficiency from a parallel distributed-memory computer. For the explicit portion, (3.3), which may be written for a generic target χ_i:

$$\chi_i = \sum_{j=-k}^{k} \gamma_j c_{i+j}, \tag{4.1}$$

the communication of the necessary halo of k elements of $\mathbf{c}$ at both ends of each segment may be carried out essentially at the same time and independently. For conventional differencing, this constitutes the *only* communication required, and the implementation is relatively straight-forward. The harder problem concerns the communication associated with the recursive portions,(3.4) and (3.5), of the algorithm. We shall devote most of this section to descriptions of two possible strategies, which we refer to as "Method 1" and "Method 2", that address the problem of parallel implementation of the twin processes of forward and backward recursions.

(*a*) *Method 1*

The generic recursion (3.4) expressed for a particular target element, ψ_i, becomes

$$\psi_i = \chi_i - \sum_{j=1}^{k} \alpha_j \psi_{i-j}, \tag{4.2}$$

and the adjoint of this recursion, (3.5), is similarly written:

$$d_i = \psi_i - \sum_{j=1}^{k} \alpha_j d_{i+j}. \tag{4.3}$$

But instead of processing each calculation in a purely sequential way, we first partition the data into segments assigned to different processors and attempt to perform the calculations in parallel. Fig. 8 shows a schematic partitioning of the data into four segments containing N_1, N_2, N_3 and N_4 points respectively. But appended to the front of each segment is a halo of k additional points, the 'head' of each segment, comprising the additional data $\boldsymbol{\psi}$ required to evaluate (4.2) for *all* of the targets interior to the segment in question. Ideally, the 'head' of one segment should be an exact copy of the last k values, or 'tail', of the previous segment, except in the case of the left boundary, where the extrapolatory procedure of Section 3c can come into play instead. Likewise, in solving the backward recursion (3.4), the same partioning applies, but the positions of the 'head' and 'tail' data of each segment is then as shown schematically in Fig. 9.

It is convenient to refer to the N_j data in segment j using the local indexing convention:

$$\boldsymbol{\psi}^{(j)} \equiv (\psi_1^{(j)}, \cdots, \psi_{N_j}^{(j)})^T \tag{4.4}$$

and to distinguish the 'head' data for the forward recursion:

$$\boldsymbol{\psi}_h^{(j)} = (\psi_{-k+1}^{(j)}, \cdots, \psi_0^{(j)})^T \qquad j = 1, 2, 3, 4, \tag{4.5}$$

and the 'tail' data by:

$$\boldsymbol{\psi}_t^{(j)} = (\psi_{N_j-k+1}^{(j)}, \cdots, \psi_{N_j}^{(j)})^T \qquad j = 1, 2, 3, 4. \tag{4.6}$$

Method 1 requires us to start with all the 'head' data set to zero:

$$\boldsymbol{\psi}_h^{(j)} = \mathbf{0}, \tag{4.7}$$

and to proceed with the recursions in all the segments simultaneously. Upon the completion of these forward recursions, the 'tail' data of each segment is, in general, inconsistent with the 'head' data of the following segment, but the time taken to get to this stage is only the time required to calculate the recursions on one segment, since the computations have proceeded in parallel. The next step, which we refer to as the 'reconciliation' of end conditions, seeks to adjust the 'head' data in response to the entire set of diagnosed inconsistencies so that a repeat of the recursions in each segment terminates with the complete elimination of any such

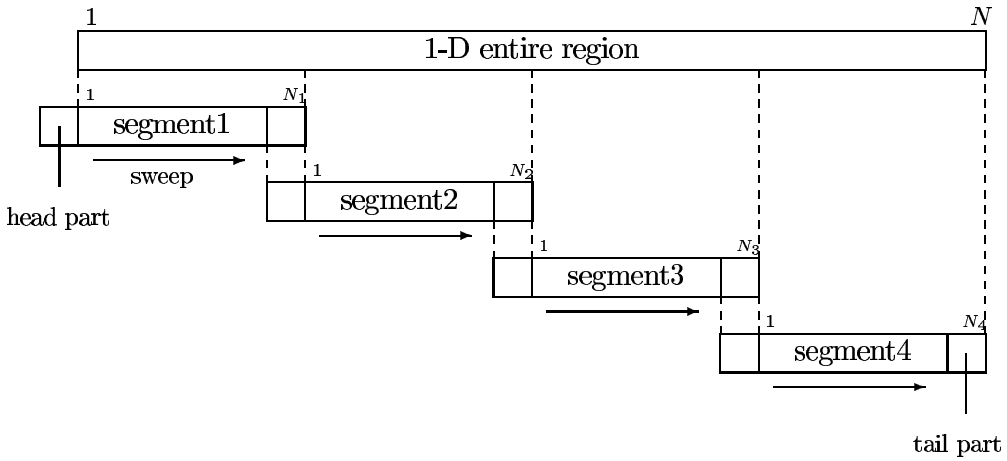

Figure 8. Region partitioning for the forward recursions

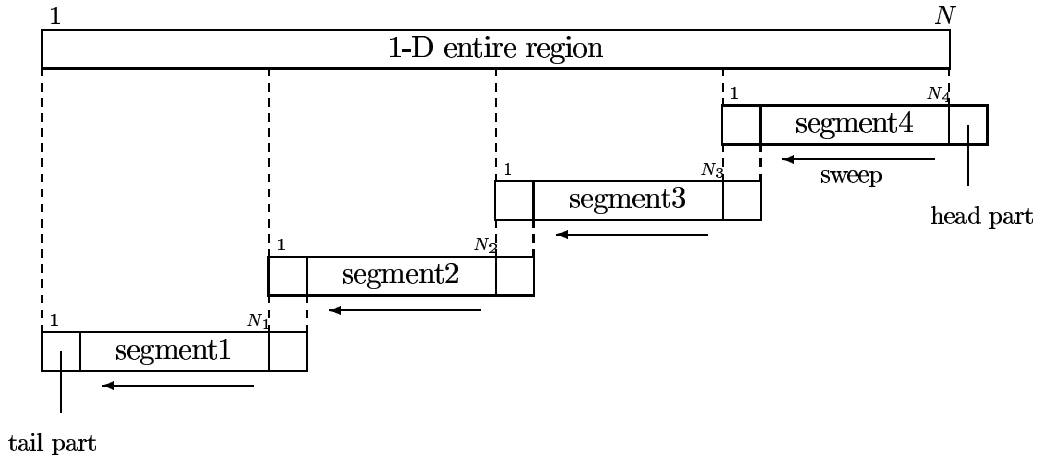

Figure 9. Region partitioning for the backward recursions

inconsistency. In other words, all the recursions are actually performed twice, doubling the amount of calculation compared to the corresponding sequential code even without counting the calculations associated with the reconciliation step. However, the benefits of parallel computations over segments should still render the overall computation more efficient than a serial code, provided the communications are sufficiently fast and there are enough segments into which each line of data is divided.

The data inconsistency is measured by error vectors,

$$\mathbf{e}^{(j+1)} = \boldsymbol{\psi}_t^{(j)} - \boldsymbol{\psi}_h^{(j+1)}. \tag{4.8}$$

For the first segment of a bounded domain, $\mathbf{e}^{(1)}$ is not defined with this equation, but assuming that we can rationally define the head values $\boldsymbol{\psi}_h^{(1)}$, through the application of an appropriate extrapolation, as proposed in Section 3c, or via an interpolation from the data grid belonging

to a model or analysis covering a more extensive domain, it is legitimate to assume the error in the first segment vanishes: $\mathbf{e}^{(1)} = \mathbf{0}$.

We consider how the changes of ψ_i, denoted as $d\psi_i$, affect values at other points ahead and, finally, at the points in the tail of the segment. First of all, we find that the sensitivity of k consecutive grid points to another k consecutive grid points which are displaced by only a unit distance can be described by,

$$\begin{pmatrix} d\psi_{i-k+1}^{(j)} \\ \cdots \\ d\psi_{i-1}^{(j)} \\ d\psi_{i}^{(j)} \end{pmatrix} = \mathbf{p} \begin{pmatrix} d\psi_{i-k}^{(j)} \\ \cdots \\ d\psi_{i-2}^{(j)} \\ d\psi_{i-1}^{(j)} \end{pmatrix}, \tag{4.9}$$

where $\mathbf{p}$ denotes a sensitivity, or 'Jacobian' matrix of the vector on the left-hand side of the equation with respect to the vector on the right-hand side, that is,

$$\mathbf{p} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_k & -\alpha_{k-1} & -\alpha_{k-2} & \cdots & -\alpha_1 \end{pmatrix}. \tag{4.10}$$

We can now make a sensitivity equation of values at the head part to values at the tail part by repeatedly using (4.9)

$$d\boldsymbol{\psi}_t^{(j)} = \mathbf{Q}_j d\boldsymbol{\psi}_h^{(j)}, \tag{4.11}$$

where

$$d\boldsymbol{\psi}_h^{(j)} = (d\psi_{-k+1}^{(j)}, \cdots, d\psi_{-1}^{(j)}, d\psi_0^{(j)})^T, \tag{4.12}$$

$$d\boldsymbol{\psi}_t^{(j)} = (d\psi_{N_j-k+1}^{(j)}, \cdots, d\psi_{N_j-1}^{(j)}, d\psi_{N_j}^{(j)})^T, \tag{4.13}$$

$$\mathbf{Q}_j = \mathbf{p}^{N_j}. \tag{4.14}$$

Note that

$$\mathbf{Q}_j \to \mathbf{0} \qquad \text{when} \qquad N_j \to \infty, \tag{4.15}$$

for stable recursions. In practice, we often find for the interesting compact schemes that the $\mathbf{Q}_j$ attain magnitudes comparable with round-off error after only a relatively small number of displacements N_j. This is a feature that we exploit in the alternative method discussed in the next subsection.

We may ask how the error vector (4.8) depends on the head values when the latter influence the tail vectors through the identity (4.11). Clearly,

$$d\mathbf{e}^{(j+1)} = \mathbf{Q}_j d\boldsymbol{\psi}_h^{(j)} - d\boldsymbol{\psi}_h^{(j+1)}, \tag{4.16}$$

so, in order to nullify the actual error, the changes required to the $\boldsymbol{\psi}_h$ are such as to provoke corresponding changes $d\mathbf{e}$ that satisfy:

$$d\mathbf{e}^{(j)} = -\mathbf{e}^{(j)} \tag{4.17}$$

simultaneously for all the j. We obtain as many individual equations for the changes to the head components as there are segments times k and, by concatenating the head and error vectors, the reconciliation problem for a single complete line of data may be expressed in the single lower-bidiagonal block-matrix equation:

$$\begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ -\mathbf{Q}_1 & \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{Q}_2 & \mathbf{I} & 0 \\ 0 & 0 & -\mathbf{Q}_3 & \mathbf{I} \end{pmatrix} \begin{pmatrix} d\boldsymbol{\psi}_h^{(1)} \\ d\boldsymbol{\psi}_h^{(2)} \\ d\boldsymbol{\psi}_h^{(3)} \\ d\boldsymbol{\psi}_h^{(4)} \end{pmatrix} = \begin{pmatrix} \mathbf{e}^{(1)} \\ \mathbf{e}^{(2)} \\ \mathbf{e}^{(3)} \\ \mathbf{e}^{(4)} \end{pmatrix}, \tag{4.18}$$

where $\mathbf{I}$ is the identity matrix of order k. Equation (4.18) can itself be solved as a simple block-recursive procedure progressing in the forward direction.

For the case of a cyclic domain the system of equations would take a form a bit different from that for the bounded case (4.18) and the solution generally requires some extra steps. The cyclic counterpart to (4.18) is:

$$\begin{pmatrix} \mathbf{I} & 0 & 0 & -\mathbf{Q}_4 \\ -\mathbf{Q}_1 & \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{Q}_2 & \mathbf{I} & 0 \\ 0 & 0 & -\mathbf{Q}_3 & \mathbf{I} \end{pmatrix} \begin{pmatrix} d\boldsymbol{\psi}_h^{(1)} \\ d\boldsymbol{\psi}_h^{(2)} \\ d\boldsymbol{\psi}_h^{(3)} \\ d\boldsymbol{\psi}_h^{(4)} \end{pmatrix} = \begin{pmatrix} \mathbf{e}^{(1)} \\ \mathbf{e}^{(2)} \\ \mathbf{e}^{(3)} \\ \mathbf{e}^{(4)} \end{pmatrix}, \tag{4.19}$$

where the condition (4.8) for the first segment must now become:

$$\mathbf{e}^{(1)} = \boldsymbol{\psi}_t^{(4)} - \boldsymbol{\psi}_h^{(1)}. \tag{4.20}$$

Eq. (4.19) can be solved by successive elimination of the $d\boldsymbol{\psi}_h$ until we finally obtain a system of equations for the last head vector, in this case, $d\boldsymbol{\psi}_h^{(4)}$:

$$\begin{aligned} d\boldsymbol{\psi}_h^{(4)} &= \mathbf{Q}_3 d\boldsymbol{\psi}_h^{(3)} + \mathbf{e}^{(4)} \\ &= \mathbf{Q}_3 \left(\mathbf{Q}_2 d\boldsymbol{\psi}_h^{(2)} + \mathbf{e}^{(3)} \right) + \mathbf{e}^{(4)} \\ &\vdots \\ &= \mathbf{Q}_3\mathbf{Q}_2\mathbf{Q}_1\mathbf{Q}_4 d\boldsymbol{\psi}_h^{(4)} + \mathbf{e}^{(4)} + \mathbf{Q}_3 \left(\mathbf{e}^{(3)} + \mathbf{Q}_2 \left(\mathbf{e}^{(2)} + \mathbf{Q}_1 \mathbf{e}^{(1)} \right) \right), \end{aligned} \tag{4.21}$$

or equivalently,

$$(\mathbf{I} - \mathbf{Q})\, d\boldsymbol{\psi}_h^{(4)} = \mathbf{e}^{(4)} + \mathbf{Q}_3 \left(\mathbf{e}^{(3)} + \mathbf{Q}_2 \left(\mathbf{e}^{(2)} + \mathbf{Q}_1 \mathbf{e}^{(1)} \right) \right), \tag{4.22}$$

where $\mathbf{Q} = \mathbf{Q}_3\mathbf{Q}_2\mathbf{Q}_1\mathbf{Q}_4 = \mathbf{p}^N$ and N is the sum of N_1, N_2, N_3 and N_4, the total number of grid points in the entire region. In typical practice it is very unusual for $(\mathbf{I} - \mathbf{Q})$ to be numerically distinguishable from the identity (except for some of the Butterworth filters whose scales of influence, if chosen sufficiently large, can 'wrap around' the entire domain). This system (4.22) can be solved for $d\boldsymbol{\psi}_h$, and subsequently, the remaining head increments, $d\boldsymbol{\psi}_h^{(1)}$, $d\boldsymbol{\psi}_h^{(2)}$ and $d\boldsymbol{\psi}_h^{(3)}$ are obtained by the forward recursion as in the bounded-domain case.

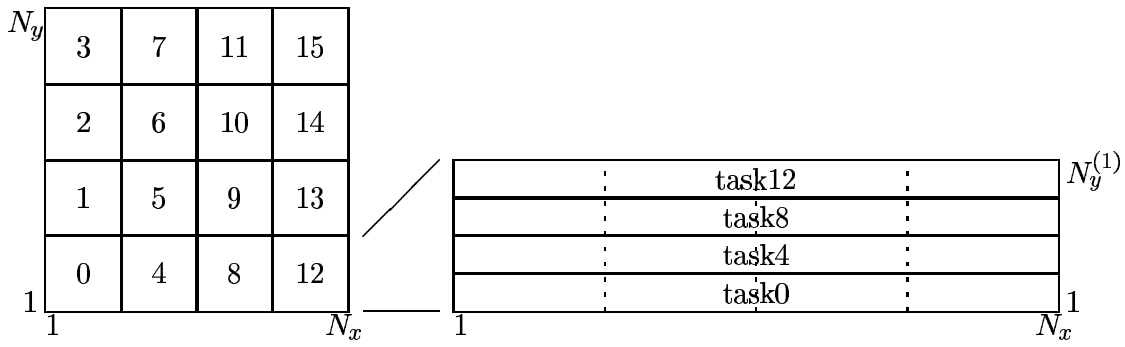

Figure 10. 2-D partitioning (left) and sharing of reconciliation burden (right). Each task execute sweeps for the subregion assigned, while it reconciles an entire region in x-direction.

The computational work associated with the reconciliation is not insignificant for a large number of segments and therefore we should strive to parallelize this burden if possible. Although our description of the algorithm of Method 1 has, until now, concerned only the one-dimensional aspects, the typical applications we are interested in involve two-dimensional or three-dimensional spatial grid arrays and, more often than not, several fields of data that require treatment by the same differencing operator. Combining the other dimensions of the data into the single transverse dimension, 'y', then a more complete schematic depiction of the typical partitioning of data employed in practice is as shown in Fig. 10

Assuming we apply the algorithm to a 2-dimensional problem, we suppose the entire region is divided horizontally into (Nsx x Nsy), where Nsx is the number of segments in x, and Nsy in y. Fig. 10, left, shows the partitioning for 16 tasks with Nsx=Nsy=4.

Because the compact scheme operation is one-dimensional and requires communication solely in this direction, we naturally split the two-dimensionally deployed tasks into Nsy subsets of Nsx sequential tasks of the differencing direction and restrict our consideration in each set of the tasks and their responsible subregions. As any calculation in each subregion is carried out by the task responsible for the region, sweeps in each subregion can be done in parallel. On the other hand, the reconciliation process handles an entire line at a time, so that the split region must be combined and redivided in the differencing direction for each task to share the burden (Fig. 10, right).

This presentation of Method 1 for the forward recursive processes is modified in the obvious way for application to the backward recursions. Instead of the lower-bidiagonal block-matrix equation of (4.18) (or its cyclic counterpart) the reconciliation involves the solution of an upper-bidiagonal block-matrix system, and therefore a backward recursion in the reconciliation process. The relevant pictorial representation of the process in this case is given by Fig. 9.

(*b*) *Method 2*

As suggested in the previous subsection, there is another way to parallelize the procedure if $\mathbf{Q}_j$ effectively vanishes (that is, when the magnitudes of its elements are comparable with the machine precision). It is therefore important to investigate, for each of the principal

compact numerical procedures, the characteristic rate of 'decay' of the exponential (or mixed exponential-oscillatory) tails with grid displacement. The relevant statistics are listed in Table 7 for unstaggered and staggered differencing, staggered integration, midpoint interpolation and implicit filtering. The 'decay rate' is the asymptotic factor, or geometric mean of the decay factor per grid unit, for the slowest decay mode of each scheme's recursions. The columns headed 'REAL(4)' and 'REAL(8)' contain estimates of the number of grid spaces required for these decay modes to render the amplitude indistinguishable from round-off at 4-byte and 8-byte precision. (The 'accumulation' step of the compact integration schemes is not considered here; it is clearly a recursive process without finite decay scale, but neglecting it only changes the values of the 'constant of integration' in each data segment, for which allowances can be made in most practical applications). The scales of influence are shortest for staggered differencing and integration, longest for unstaggered differencing and midpoint interpolation, but in no case are the effective scales of influence so large that they would extend over more than a small fraction of the horizontal grid of a meteorological simulation model at today's typical resolutions and domain sizes.

The case of a periodic domain is the most convenient one to consider here. In the situations where the decay to numerical insignificance occurs within the width of a single subdomain, the correction vectors $d\boldsymbol{\psi}_h^{(j)}$ are defined instantly once the difference vectors $\mathbf{e}^{(j)}$ are found. The errors $\mathbf{e}^{(j)}$ are then practically defined by the upstream sweeps. To exploit this situation efficiently it is natural to divide each computational subregion transversally, assigning one of them as an initiation area in a pattern that staggers the transverse locations of these initiation areas from one subdomain to the next in line, as indicated schematically, again for the four-segment domain-length decomposition, by Fig. 11. The calculations begin simultaneously in the sub-regions ('stripes') marked '1, 5' initiated with 'head' values set (erroneously) to zero. While the errors incurred propagate forward along with the recursions, the magnitude of the errors, compared to the globally consistent ideal numerical solution, dwindle and effectively vanish by the time the 'tail' values are being computed at the far end of each of these stripes. Except for these essentially 'correct' tail values, the rest of the results for this preliminary step are simply discarded. The tail values of one subdomain are communicated without alteration to provide the matching 'head' values in the neighboring subdomain's stripe marked '2', which involves the same subset of grid lines. Because of the pattern of staggering for these stripes, this second period of calculation also keeps all processors busy. The pattern of computation is likewise repeated to stripes marked '3', then '4' and, finally, in order to provide the *consistent* output values in the original set of stripes (where the first outputs were thrown away), the fifth period of computations returns to this set of stripes (marked '1, 5'). This time, the 'head' values *are* consistent, since they are supplied by the 'tail' values of the stripes marked '4'. The same pattern of calculation, but with the appropriately qualified rules of communication at the domain boundaries, applies to the parallelization of recursions over a bounded (nonperiodic) domain, even though this appears to entail redundant calculations in the two subdomains adjacent to the boundaries. Again, the application of these ideas to recursions proceeding in the backward direction requires the obvious modifications to the staggered pattern of stripes.

Communications are simpler to configure in this case than in the distributed reconciliation of Method 1 (clearly, Method 2 requires no extra reconciliation). In cases where the scale of

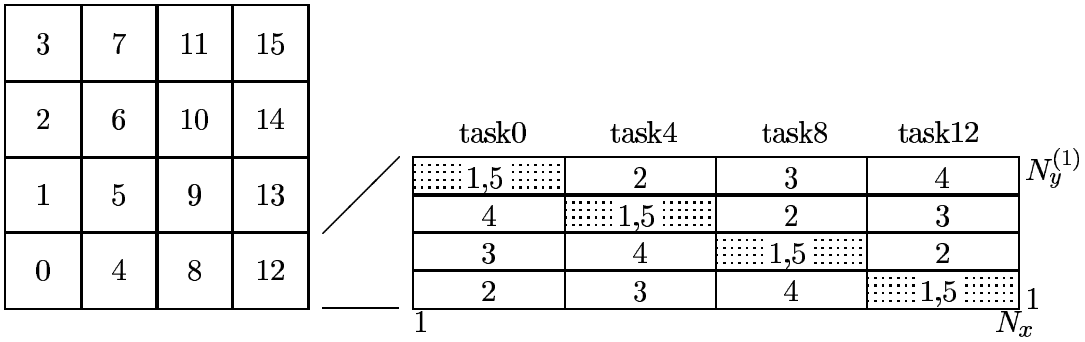

Figure 11. Same as Fig.10 but for the forward sweep in the case of Method 2 when, effectively, $\mathbf{Q} = 0$. Each task sweeps its own four-partitioned subregion once except for the repetition in each stripe where sweeping is initiated (dotted region marked by '1,5'). For the corresponding backward sweep, the initiation stripes are also staggered, but lie diagonally but in the opposite orientation to ensure proper alignment and ordering of the sequence-numbered stripes.

decay of the recursive processes exceeds the width of a subdomain, the region of overlapping or repeated calculations can be extended to cover more than one subdomain (at the cost of the implied additional calculations and communications). But, typically, we might expect the overall amount of calculation required by this Method 2 to be significantly less than the more than two-fold redundancy that is intrinsically and inescapably a feature of Method 1. For very large domains subdivided length-wise into numerous segments, the quantity of calculation in Method 2 can theoretically reduce to only slightly more than the optimum attained by serial processing. If there is a sufficient quantity of data in the transverse direction to ensure that the stripes remain wide enough to keep the communications efficient, then this method seems undoubtedly to offer the better of the two strategies proposed. However, it is possible that, with insufficient lines of data in the transverse direction and a large number of segments longitudinally, the lengths of the messages communicated become too small to avoid a severe 'latency' penalty associated with the initiation of each short message. (A similar hazard can befall the reconciliation steps of Method 1). Should this pose a threat to the method's overall efficiency it might be worth considering reducing the number of stripes in the transverse direction and making each one wider by selecting more than one location along each complete line of data where the initiation of the recursions is allowed to occur. In this way, the communication overheads are reduced (fewer, but longer messages) at the expense of slightly higher calculation costs (additional overlap regions of the kind marked in Fig. 11 by '1, 5'). We reiterate that the success of Method 2 hinges on the assumption that the influences of the recursive processes involved decay away relatively fast; Method 2 is probably *not* appropropriate for general filtering codes where the possibility of arbitrarily large scales of influence must be allowed for.

(*c*) *Method 3*

By Methods 1 and 2, we concentrate on diminishing the message lengths between tasks, either by way of the rather complicated calculations for reconciliation steps (Method 1), or by staggering the initiation location of each recursion from one processor to the next in line (Method 2) to keep all processors busy. Instead, if we invest the effort initially to place all

TABLE 7. The scales of influence of various functions and orders of accuracy. Decay rate depends on the type of functions, the order of accuracy, the coordinate and the precision of variables used. The names of function show compact differencing by Diff., integration by Integ., midpoint interpolation by Intplt. and sine-Butterworth filter by S-B. For the S-B, only the waves of wavelengths 2h and 3h are considered.

Function	Coord/ Wavelength	Order	Decay Rate	REAL(4)	REAL(8)
Diff.	Unstag.	4	0.268	12.1	27.4
		8	0.493	22.5	50.9
Diff.	Stag.	4	0.046	5.2	11.7
		6	0.148	8.4	18.9
		8	0.236	11.1	25.0
		10	0.267	13.8	31.2
Integ.	Stag.	6	0.083	6.4	14.5
		8	0.178	9.3	20.9
		10	0.267	12.1	27.3
Intplt.	Stag.	4	0.172	9.0	20.4
		6	0.333	14.5	32.8
		8	0.446	19.8	44.7
		10	0.528	25.0	56.4
S-B	3h	1	0.209	10.2	23.0
		2	0.268	12.1	27.4
		3	0.338	14.7	33.2
		4	0.397	17.3	39.0
		5	0.448	19.8	44.9
		6	0.491	22.4	50.7
S-B	2h	1	0.172	9.0	20.4
		2	0.217	10.4	23.6
		3	0.268	12.1	27.4
		4	0.311	13.6	30.8
		5	0.346	15.0	34.0
		6	0.376	16.3	36.9

communicating data of a line into the same processor, employing what is known as a global transpose, the reconciliation steps can be dispensed with. Afterward, the inverse transpose of the data puts them back into the processors to which they were assigned originally. This leads to a rather simple way of implementing the compact schemes and is hereafter called 'Method 3'. When we can assume $\mathbf{Q}$ vanishes below round-off significance in the course of a full sweep of the domain (generally the case in practice), then Method 3 will further simplify and become more efficient even for a cyclic region. As in the previous methods, Method 3 is realized by partitioning the subregions into stripes as shown in Fig 10, right. But now each stripe is

assigned to a single processor. The reversal of the transpose of data is then necessary at the end and, since each message is fairly large, communication costs are subsequently expected to be relatively large per message. Computational costs, on the other hand, are expected to be as small as in Method 2. Method 3 requires a large movement of data at the start and at the end while Method 2 moves a significantly smaller amount of data, but fragmented into many smaller messages, so it is not obvious which strategy will prove superior in a given situation. This is the question we attempt to answer in the next section.

5. Computational Costs

In this section, the performances of parallel programs using compact schemes of Method 1, 2, 3 and conventional centered schemes on a parallel machine (IBM RS/6000 SP) are compared. The sample programs, highly tuned for the machine, execute numerical differencing five times, and the following results are the best ones of the repetitive trials. Since resulting times are only moderately stable and possibly fluctuate, especially when longer times are required, the results for the larger domain are measured for a single operation and simply multiplied by 5 for easy comparison with other results.

Table 8 shows the timings by Method 1, 2, 3 and the centered conventional differencing schemes for various data sizes over the two dimensional domain, basically assuming a three dimentional domain with 250×250 grid points in the horizontal, extending in the non-processing direction assuming multiple elements and vertical levels are folded into a two dimentional array to process comprehensive data at a single call. The timings for 'Total' are the durations required for the five calls of the differencing subroutine, and they are broken down into 'Comm.' and 'Calc.' which show the required time for communication and calculation excluding small costs for the establishment and dismissal of communication groups. 'Min.' shows the minimum timings of a single differencing.

In Table 8, 'Total' times appear to scale in proportion to the size of the entire domain by any scheme for the large data ($N_y \geq 2500$), while they do not for the smallest data ($N_y = 250$). From the comparisons, Method 2 is clearly better than Methods 1 and 3. Theoretically, the calculation times for sweeping in Methods 2 and 3 are 5/8 of Method 1 with four processors in the operating direction, and for the case of $N_y = 25000$ they would be $0.355 \times 5/8 \approx 0.222$ sec, though they are actually 0.256 sec and 0.300 sec for Methods 2 and 3. The inefficiency of Method 3 compared with Method 2 and Method 2 against the theoretical estimation may result from the particular way of implementating the schemes, which incurs its own handling costs for arrays. In the meantime, the communication requires longer times in Method 3 than in Method 1, so that the merit of the reduced calculation costs in total time appears only for Method 2, and not at all for Method 3. From 'Total' times we can safely say that, for this computational platform, at least, Method 2 is the best choice for computational performance and, for a single compact operation in isolation, Method 3 would be recommended only in a limited situation. (When a string of consecutive compact operations must be applied in the *same* direction, Method 3 obviously becomes more attractive, since the cost of the data transposions performed once at the beginning and once at the end of such a sequence is effectively shared.) Method 1 would only be a reasonable choice when the rapid decay to insignificance of the influence function

of the recursive operators, as described in the previous sections, cannot be guaranteed for the particular recursions under consideration.

TABLE 8. Timing of compact numerical differencing schemes of the three methods and conventional centered differencing schemes for cyclic regions of different sizes. Timings are measured for five iterations of differencing.

	Processors	Order	Nx	Ny	Total	Comm.	Calc.	Min.
Method 1	4×4	8	250	25000	0.397	0.056	0.337	0.079
	4×4	8	250	2500	0.044	0.013	0.027	0.009
	4×4	8	250	250	0.019	0.012	0.003	0.004
Method 2	4×4	8	250	25000	0.259	0.033	0.222	0.052
	4×4	8	250	2500	0.030	0.007	0.018	0.006
	4×4	8	250	250	0.014	0.006	0.002	0.002
Method 3	4×4	8	250	25000	0.595	0.250	0.340	0.119
	4×4	8	250	2500	0.052	0.026	0.020	0.010
	4×4	8	250	250	0.010	0.004	0.002	0.002
Conventional	4×4	8	250	25000	0.160	0.025	0.130	0.032
	4×4	8	250	2500	0.018	0.003	0.011	0.004
	4×4	8	250	250	0.006	0.001	0.001	0.001

Table 8 also shows comparison of conventional and compact numerical differencing schemes. It is proved that conventional schemes are about two to six times more efficient in time than compact schemes. Although the formal algebraic operation counts are not so much different from each other, compact schemes require more frequent communications and data handling, which significantly affects the performance.

Table 9 shows the performance of Method 2 with various orders of accuracy. When the order of accuracy is raised from 4 to 6, and 8 to 10, the right-hand side of the compact schemes has more coefficients and thereby requires more calculation and the exchanging of larger messages. The left-hand side has more coefficients when the order is raised from 6 to 8. In both cases, computational costs increase moderately with order of accuracy.

TABLE 9. Same as Table 8 but only for Method 2 with various orders of accuracy.

	Processors	Order	Nx	Ny	Total	Comm.	Calc.	Min.
Method 2	4×4	4	250	25000	0.275	0.050	0.220	0.055
	4×4	6	250	25000	0.295	0.050	0.235	0.059
	4×4	8	250	25000	0.320	0.055	0.260	0.064
	4×4	10	250	25000	0.350	0.070	0.280	0.070

Table 10 shows the results with different numbers and assignments of processors. While the computational costs of Method 2 with a fixed number of processors diminish when mode processors are assigned so as to divide the domain in the direction of processing, the timings seemingly diminish with fewer processors in the operational direction.

TABLE 10. Same as Table 8 but only for Method 2 with different numbers of processors.

	Processors	Order	Nx	Ny	Total	Comm.	Calc.	Min.
Method 2	2×2	8	250	25000	1.220	0.050	1.165	0.244
	4×2	8	250	25000	0.695	0.120	0.570	0.139
	2×4	8	250	25000	0.575	0.030	0.540	0.115
	8×2	8	250	25000	0.415	0.090	0.320	0.083
	4×4	8	250	25000	0.320	0.055	0.260	0.064
	2×8	8	250	25000	0.260	0.015	0.240	0.052

6. Discussion and concluding remarks

Compact schemes are most easily expressed by assuming that the lines on which they are to operate extend without bounds in both directions. Under this assumption, the matrix representation of each scheme possesses diagonally invariant form, implying that the coefficients do not depend upon the location within the grid. We have shown that boundary conditions can be constructed to be consistent with this assumed invariance, in both bounded and cyclic domains. One consequence of the simplifying assumption is that it allows the recursive processes to be broken down, or 'factored', into pieces each associated with a subdomain corresponding to a subdivision of the grid required by parallel processing. Our Methods 1 and 2 exploit this factoring directly, but in different ways, as described in section 4. The results of timing experiments show that Method 2 is uniformly superior to Method 1 in implementing the 'standard' compact operations. It would therefore only be for very special circumstances, such as the implementation of recursive filtering codes in which the length-scale of the filters were unknown *a priori*, where Method 2 might fail, that one would prefer Method 1.

In comparison with conventional numerical methods of corresponding orders of accuracy, even the Method 2 timings were found to be somewhat disappointing, owing in part to the slightly greater computation burden inherent in this method, but also owing to the significantly greater costs associated with the communications between processors which, for compact schemes, necessarily involve more messages. In an attempt to replace the transmission a many short messages throughout the recursions by fewer but longer messages exchanged at the beginning and end of the entire process, we also investigated the cost of this approach, 'Method 3'. Unfortunately, in no cases was this method found to be competitive with Method 2 for single compact operations, although it would clearly pay off in cases where *several* recursive, or broad-stencil numerical operations, directed along the same lines of the grid are required to be executed consecutively. This is actually the case in some algorithms for the conserving grid-to-grid interpolations of a semi-Lagrangian model. Another advantage of the global transposes effected by Method 3 is that they significantly eliminate much of the code complexity of the other two methods.

The present study was conducted on a single parallel platform, the IBM SP at NOAA/NCEP. However, preliminary indications suggest that the qualitative rankings do not differ significantly when some similar timing experiments were conducted using a nine-node Compaq cluster at NCAR, Boulder, and the Compaq XP1000 Linux cluster at FSL, Boulder. Since many of the

rankings among the competing methods can depend upon hardware considerations, especially concerning the latency time for message passing relative to the speed of performing floating-point operations, same of our conclusions may require some qualification or amendment when compact schemes are implemented on alternative computing platforms. We hope to be able to extend the scope of this study at a future time to include detailed timing data for a wider variety of other parallel machines.

Acknowledgments

The authors are grateful to Drs. David Parrish, Mark Iredell and Zavisa Janjić for many useful discussions about recursive computations and to Messrs. Jim Tuccillo of IBM and John Michalakes of ANL for independently suggesting the inclusion of timing comparisons with the global transposes ('Method 3'). We would also like to thank the computer support staff at FSL, Boulder, and at NCAR for their assistance in setting up accounts that enabled the main timing results of this note to be substantiated on two other parallel architectures. This work was partially supported by the NSF/NOAA Joint Grants Program of the US Weather Research Program. This research is also in response to requirements and funding by the Federal Aviation Administration (FAA). The views expressed are those of the authors and do not necessarily represent the official policy or position of the FAA.

References

Abramowitz, M., and I. A. Stegun 1965 *Handbook of Mathematical Functions*, Dover, New York. 1046 pp.

Bourke, W. 1972 An efficient, one-level, primitive-equation spectral model. *Mon. Wea. Rev.*, **100,** 683–689.

Hamming, R. W. 1998 *Digital Filters*, Third Edition: Dover Publications, Inc., 284 pp.

Lele, S. K. 1992 Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.*, **103,** 16–42.

Leslie, L. M., and R. J. Purser 1995 Three-Dimensional Mass-Conserving Semi-Lagrangian Scheme Employing Forward Trajectories: *Mon. Wea. Rev.*, **123,** 2551–2566

Mesinger, F., and A. Arakawa 1976 Numerical methods used in atmospheric models. *GARP Publication Series No. 14,* WMO/ICSU Joint Organizing Committee, 64 pp.

Navon, I. M., and H. A. Riphagen 1979 An implicit compact fourth-order algorithm for solving the shallow-water equations in conservation-law form. *Mon. Wea. Rev.*, **107,** 1107–1127.

Otnes, R. K., and L. Enochson 1972 *Digital Time Series Analysis*: John Wiley & Sons, Inc., 467 pp.

Purser, R. J., and L. M. Leslie 1988 A semi-implicit semi-Lagrangian finite-difference scheme using high-order spatial differencing on a nonstaggered grid. *Mon. Wea. Rev.*, **116,** 2067–2080.

Purser, R. J., and L. M. Leslie 1991 An Efficient Interpolation Procedure for High-Order Three-Dimensional Semi-Lagrangian Models: *Mon. Wea. Rev.*, **119,** 2492–2498

Purser, R. J. 1998 Efficient High-Order Semi-Lagrangian Methods: *Seminar Proceedings on Recent Developments in Numerical Methods for Atmospheric Modeling*, ECMWF, Reading, UK, 7-11 September 1998, 73–94

Raymond, W. H.	1988	A Spatial Filter for Use in Finite Area Calculations: *Mon. Wea. Rev.*, **116,** 209–222
Raymond, W. H., and A. Garder	1991	A Review of Recursive and Implicit Filters: *Mon. Wea. Rev.*, **119,** 477–495
Staniforth, A., and J. Côté	1991	Semi-Lagrangian integration schemes for atmospheric models — a review. *Mon. Wea. Rev.*, **119,** 2206–2223.