P6.5 PARALLEL IMPLEMENTATION OF COMPACT NUMERICAL SCHEMES

Tsukasa Fujita* and R. James Purser

Environmental Modeling Center, NCEP
Camp Springs, MD

1. INTRODUCTION

Compact numerical schemes for high-order spatial differencing and integration are planned to be incorporated into a non-hydrostatic semi-Lagrangian dynamical core (for example, see Purser et al. 2001) under development at NCEP as part of the multi-institution Weather Research and Forecasting (WRF) project. Since the schemes require recursive operations along entire lines of one dimension, they should be implemented with algorithms carefully devised not to impair their performance seriously. We tested three methods which feature different parallelization strategies for the recursive components, all of which are intended to be computationally efficient. The three methods differ significantly in their handling of communication. We describe these implementations, their limitations, and their relative computational performance. Only a brief survey of these techniques is provided in this preprint. A much more detailed treatment of this investigation will be found in Fujita and Purser (2001).

2. STRUCTURE OF COMPACT SCHEMES

A conventional high-order centered differencing operator, which can be thought of as applying to a vector $\boldsymbol{c}$ formed by the source data along the grid line to produce the target line of derivatives, vector $\boldsymbol{d}$, operates according to the explicit formula:

$$\boldsymbol{d} = \boldsymbol{B}\boldsymbol{c}. \tag{1}$$

The corresponding 'compact' scheme uses an alternative and more compact stencil of the derivative coefficients that form the band matrix $\boldsymbol{B}$, but also involves a second band matrix matrix, $\boldsymbol{A}$, of left-hand side stencil coefficients to make the defining operator for a line of data implicit:

$$\boldsymbol{A}\boldsymbol{d} = \boldsymbol{B}\boldsymbol{c}. \tag{2}$$

* UCAR Visiting Scientist. On leave from the JMA, Tokyo, Japan

Corresponding author address: R. J. Purser, W/NP2 RM 207, WWBG, 5200 Auth Road, Camp Springs, MD 20746

The compact schemes at a given order of accuracy tend to involve a slightly greater computational cost (typically between $10 - 20$ per cent), than their conventional counterparts on a serial shared-memory computer, but yield impressively smaller coefficients of principal truncation error. An example of a popular fourth-order compact scheme is given in Navon and Riphagen (1979). For a grid of uniform spacing, h, the stencil coefficients in a generic row of each matrix are $[1/6, 2/3, 1/6]$ for $\boldsymbol{A}$, and $(1/h)[-1/h, 0, 1/2]$ for $\boldsymbol{B}$. This scheme has a principal truncation error six times smaller than that of the conventional fourth-order differencing scheme, whose wider $\boldsymbol{B}$-stencil is $(1/h)[1/12, -2/3, 0, 2/3, -1/12]$. A variety of other compact schemes are discussed in Lele (1992).

For a general centered scheme, let p be the half band-width of the $\boldsymbol{A}$-stencil ($p = 1$ for the fourth-order compact scheme, and $p = 0$ for any conventional scheme for which we may implicitly regard the omitted $\boldsymbol{A}$ matrix as being the identity). Likewise, let q be the half-width of the $\boldsymbol{B}$-stencil ($q = 1$ for the fourth-order compact scheme, $q = n/2$ for any conventional centered scheme of order

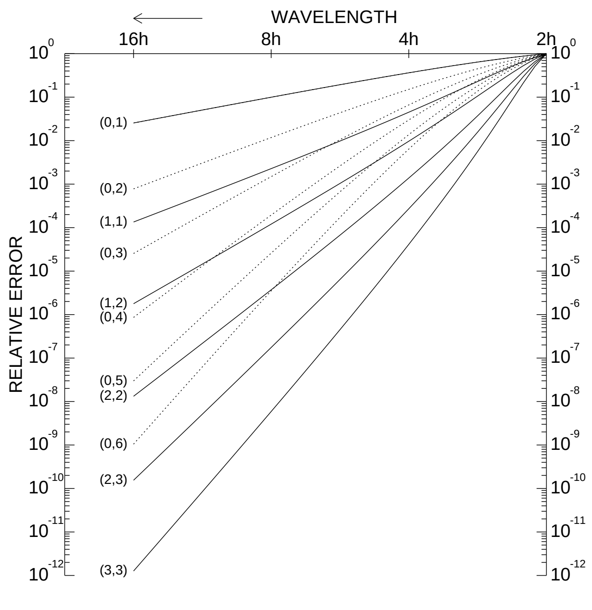

Figure 1. Log-log plot of the relative truncation error of unstaggered centered stencil differencing schemes as a function of the wavenumber of the sine-wave to which they are applied.

n). Then the most accurate compact scheme for a given order of accuracy, $n = 2(p+q)$, is the scheme for which either $q = p$ or else $q = p+1$. (It is owing to this fact that the 'compact' schemes derive their name.)

Fig. 1 graphs the accuracy (in terms of relative error) of various centered unstaggered difference schemes denoted by their (p, q) pairs plotted as a function of the wavelength of the sinusoidal function to which each scheme is applied. These graphs confirm that the accuracy of the compact schemes exceeds that of the conventional schemes for each of the equivalent orders of accuracy represented. It also shows that the relative improvement obtained with the compact scheme increases as the formal order of accuracy increases.

The inherently recursive nature of the equations we need to solve when the compact schemes are used suggests initially factorizing the matrix $\boldsymbol{A}$ into lower-triangular and upper-triangular factors. The symmetry of matrix $\boldsymbol{A}$ may be exploited to make these factors each other's transpose and a constant scaling of both sides of (2) lets us assume that the principal diagonal of the resulting Cholesky factor, $\boldsymbol{L}$, consists of ones. The solution for a line of compact derivatives may now be obtained by carrying out the explicit operation associated with the right-hand side matrix $\boldsymbol{B}$:

$$\boldsymbol{\chi} = \boldsymbol{B}\boldsymbol{c}, \tag{3}$$

followed by a pair of back-substitution steps:

$$\boldsymbol{L}\boldsymbol{\psi} = \boldsymbol{\chi}, \tag{4a}$$

$$\boldsymbol{L}^T\boldsymbol{d} = \boldsymbol{\psi}. \tag{4b}$$

Eq. (3) is an explicit operation which is straight-forward to apply on either a serial or parallel machine. However, (4a) and (4b) are recursive and would appear to require the calculations to proceed in the forward and backward directions respectively, since in component form, these steps would be executed as:

$$\psi_i = \chi_i - \sum_{j=1}^{p} \alpha_j \psi_{i-j}, \tag{5a}$$

$$d_i = \psi_i - \sum_{j=1}^{p} \alpha_j d_{i+j}, \tag{5b}$$

where the elements of $\boldsymbol{L}$ j steps from the principal (unit)-diagonal are here denoted α_j.

The back-substitution steps present no problems on a serial processor but we are immediately faced with non-trivial practical difficulties as soon as we attempt to apply these recursions efficiently across the several subdomains of a grid that is split up for parallel processing on a distributed-memory computer. The sequential mode of processing, which is acceptable for a serial machine where the calculations are in any case performed one at a time, is unacceptably inefficient in a parallel architecture where we want *all* processors to be simultaneously performing useful work, and not waiting in line for their own stage of each recursion to be reached before they begin computing.

We have investigated several methods for overcoming this problem, using various configurations of domain decomposition and several different compact differencing, integration and midpoint interpolation procedures (all of which have a potential role to play in the mass-conserving semi-Lagrangian atmospheric model referred to in the introduction). We have also been able to apply our methods on a small number of different parallel computers. Only a brief descriptive outline of three of these methods can be given in this preprint; additional details will be presented at the conference.

3. STRATEGIES FOR PARALLEL IMPLEMENTATION

The majority of the timing experiments were carried out for the case of eighth-order unstaggered differencing - the compact scheme $(2, 2)$ and, for comparison, the explicit scheme $(4, 0)$ in the indexing notation used in section 2. The relevant dimensions of the grid for these experiments are the 'active' dimension (parallel to the direction in which the differencing operator is applied) and the product of all transverse dimensions, including the 'dimension' over different field variables that are simultaneously treated by the same operator. By applying the operators to all such fields in a combined calculation in a numerical model one reduces the number of inter-processor messages initiated and hence the overall time attributable to message latency. Also relevant is the manner in which the grid is carved up into subdomains, especially the number of divisions in the longitudinal direction. At each interior boundary formed by the division into subdomains, it is necessary to transfer a sufficiently wide strip (that is, at least p grid points, in the case of the recursive processes, q grid points for the explicit steps of the compact scheme) of the updated interior data from the end of one subdomain to the corresponding neighboring domain. This transferred copy of the data becomes available for calculations in the neighboring processor as its so-called 'halo'. The cost of transferring the halo data is a significant proportion of the computations in a parallel computer. The strategies we apply seek to ensure that all processors remain busy throughout the computations while trying to minimize the cost associated with the transfer of halo messages between processors.

3.1 *Method 1*

The first strategy, which we refer to as 'Method 1' is to apply the recursive computations in all subdomains simultaneously, regardless of the fact that erroneously zero values are being used to initiate these recursions in the subdomains' respective halo strips. Upon completing these recursions for each subdomain, a set of numerical discrepancies can be collected from each of these haloes. Then, by redistributing the resulting array of errors equitably among processors, in such a way that all the discrepancies belonging to any given complete grid line parallel to the active direction are assigned to the same processor, it is possible to reconcile the mismatches and infer for each halo region precisely what the *correct* results of the recursive sweep should have been. Following this substitution, a second sweep of the same recursion simultaneously in every subdomain, but now beginning with the *correct* halo values, will yield a self-consistent end result.

One obvious defect of this strategy is that it requires the recursive calculations throughout the domain to be repeated. Also, the 'reconciliation' step, by which the corrected halo values are obtianed, involves a significant amount of message passing in order to be able distribute the work of calculating the results for this intermediate step evenly among the processors available.

3.2 *Method 2*

The second strategy, 'Method 2', seeks to minimize the redundancy associated with the repetition of each recursion in Method 1 by exploiting the fact that the characteristic reponse functions of the recursive operations associated with actual compact numerical methods involve rapidly decaying exponentials. This implies that, given the finite and known machine precision, the *effective* range of influence of these operations is never more than a few tens of grid spaces. In fact, this implied decay scale is typically less than the width of a single subdomain, so the elaborate 'reconciliation' step of Method 1 can be substantially simplified.

Initially we subdivide each of the subdomains in the transverse direction into 'stripes' that match up for the length of the entire domain. For any given stripe the recursive operations are begun at a designated processor that need not be the one at the starting edge of the whole domain. Having chosen a starting processor for recursive calculations of a particular stripe, each processor in cyclic sequence along the length of this stripe waits its turn to process the recursion in the proper relative order until a full cycle, plus the requisite decay-width overlap necessary to overwrite the earliest of these recursive calculations, has been completed. Although the recursions begin erroneously (unless the starting point for this stripe happens to be at the appropriate domain boundary) for want of the proper values with which to prime the recursions, the error incurred diminishes to numerical insignificance within the scheme's characteristic decay-scale. But this extent of the earliest results is precisely what is overwritten when the cycle of one-sided recursive calculations for this stripe becomes complete. The halo communication is simpler than in Method 1; each halo message is passed only to the adjacent subdomain.

Since many stripes thread any given subdomain, it is always possible to stagger the starting locations of the recursions across the bundle of stripes in a staggered sequence that ensures that, at any given time, every processor has at least one stripe threading it on which it is actively calculating the recursions in (5a) or (5b). Moreover, for domains of large size in the active direction of the recursion, the remaining inefficiency associated with the necessary overlap of this method becomes very small in proportionate terms.

3.3 *Method 3*

The last of the strategies we have studied is simply the 'brute-force' approach of performing a global ('all-to-all') transposition of the gridded data initially to arrange that all of the points of each given domain-straddling line parallel to the active direction are placed in the same processor. The recursive computations then involve no further

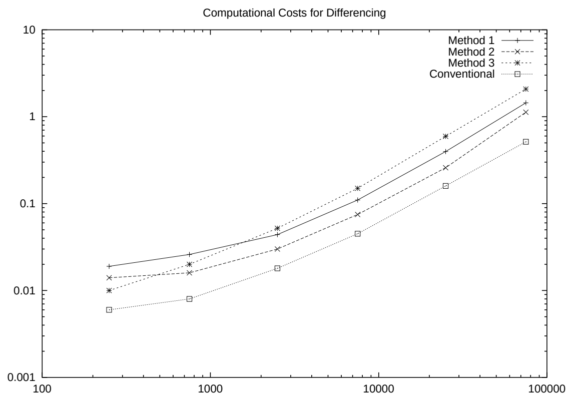

Figure 2. Log-log plot of the timing experiment results for Methods 1, 2, 3, and conventional centered scheme, all of eighth-order, using 4×4 processors. The total numbers of grid points in directions transverse to the direction of application of the differencing scheme are plotted along the x-axis. Times required to repeat each operation five times over the entire grid are indicated by the y-axis.

message-passing until, at the end of the computation phase for this compact scheme, the data can be re-arranged (by another global transposition) back to their original processors.

Of the three methods, this 'Method 3' is clearly the most straight-forward to implement. However, it involves the movement of practically *all* of the gridded data.

4. RESULTS AND CONCLUSIONS

Various grid and subdomain configurations have been tested. The three methods summarized in section 3 were compared together with an implementation of the conventional differencing method of corresponding order. The results shown in Fig. 2 are rather typical as far as the relative rankings go. The conventional scheme is always computationally cheaper than any of the compact schemes. Among the compact schemes, Method 2 is consistently superior to Method 1. Usually, Method 3 is the most costly strategy. The results depicted are for NCEP's IBM SP computer, but similar relative rankings have been obtained on other parallel platforms. For large domains combined with large numbers of domain subdivisions in the active direction, the gap between the conventional scheme and compact Method 2 closes somewhat, making this compact implementation a more attractive option (given its substantially smaller formal truncation error coefficients).

Despite the usually poor performance of Method 3 when only a single compact operation is sandwiched between global transposes of data, this choice becomes much more appealing for some applications in numerical weather prediction where it is possible to concatenate a short sequence of the required compact operations that all share the same active direction. In this case, the communications are restricted to one traspose at the start and one at the end of such a chain of operations. The communications overheads then become much less burdensome.

5. ACKNOWLEDGMENTS

The authors are grateful to Mr. Jim Tuccillo of IBM and Mr. John Michalakes of ANL for valuable discussions and assistance. This work was partially supported by the UCAR Visiting Scientist Program and 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

Fujita, T., and R. J. Purser, 2001: Parallel implementation of compact numerical schemes. NOAA/NCEP Office Note, 33 pp. (To appear.)

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

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.

Purser, R. J., T. Fujita, S. K. Kar, and J. G. Michalakes, 2001: A semi-lagrangian dynamical core for the non-hydrostatic wrf model. (Preprint) AMS Ninth Conference on Mesoscale Processes, Jul. 30th–Aug. 2nd 2001, Fort Lauderdale, FL.