NASA/CR-1999-209708
ICASE Report No. 99-40

Wavefront Cache-friendly Algorithm for Compact Numerical Schemes

Alex Povitsky
ICASE, Hampton, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

October 1999

WAVEFRONT CACHE-FRIENDLY ALGORITHM FOR COMPACT NUMERICAL SCHEMES

ALEX POVITSKY*

Abstract. Compact numerical schemes provide high-order solution of PDEs with low dissipation and dispersion. Computer implementation of these schemes requires numerous passes of data through cache memory that considerably reduces performance of these schemes. To reduce this difficulty, a novel algorithm is proposed here. This algorithm is based on a wavefront approach and sweeps through cache only twice.

Key words. cache locality, compact scheme, wavefront algorithm, banded linear systems

Subject classification. Computer Science, Applied and Numerical Mathematics

1. Introduction. Compact numerical schemes are widely used for challenging problems of computational physics [1]. Compact finite difference formulas are defined as expressions where derivatives at different mesh points appear simultaneously. These schemes mimic spectral schemes with low dissipation and dispersion.

In spite of the fact that the number of arithmetic operations per grid node is approximately equal for explicit and compact formulations of the same approximation order [2], the computational time is considerably larger for the compact schemes. Tam and Webb ([3], p. 278) reported about the order of magnitude computational time difference between explicit and compact formulations. The poor performance of compact schemes is explained by architectural features of modern computers.

The gap between formal CPU performance and actual performance is likely to increase because the CPU speed tends to increase much faster than the speed of memory access. To increase the computational efficiency of modern computers, the interface between a processor and its memory includes a number of cache memories that are placed (logically) between the processor and the physical main memory, giving the processor fast access to data stored in the cache [4]. Access to main memory typically requires dozens or hundreds of flops, and reduction of the main memory-cache exchange represents a challenge for scientific computing.

Compact schemes require solution of banded linear systems, which consider each spatial partial derivative separately. Solution of banded linear systems by Gaussian elimination requires forward and backward sweeps through data. These sweeps are repeated in all three spatial directions followed by a Runge-Kutta temporal update (RK) and computing the right-hand sides of compact formulations. Thus, compact 3-D solvers pass data through the memory cache eight times. On the contrary, an explicit central-difference algorithm may be easily written in such a way that data passes through the cache only once.

This study proposes a new formulation of a compact-scheme based numerical algorithm which sweeps through data only twice per stage of RK. The algorithm is based on a wavefront approach, where a current front nodes are computed using only the values at previous fronts. Then this algorithm is expanded to any number of levels of cache memory. Numerical solution at each time step is exactly the same as for a standard compact algorithm.

*Staff Scientist, ICASE, NASA Langley Research Center, Hampton, VA 23681-2199 (e-mail: aeralpo@icase.edu). This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199

2. High-order Numerical Methods. Consider a multi-dimensional partial differential equation (PDE):

$$\frac{dU}{dt} = S_x \frac{\partial U}{\partial x} + S_y \frac{\partial U}{\partial y} + S_z \frac{\partial U}{\partial z} \tag{1}$$

where t is the time, $k = 1, 2, 3$ are spatial coordinates. The right-hand side terms are approximated using compact finite difference schemes [1]:

$$\beta U'_{i-2} + \alpha U'_{i-1} + U'_i + \alpha U'_{i+1} + \beta U'_{i+2} = \frac{a}{2\Delta x}(U_{i+1} - U_{i-1}) + \frac{b}{2\Delta x}(U_{i+2} - U_{i-2}), \tag{2}$$

where Δx is the grid step and primes denote derivatives with respect to x. Expansion to systems with second spatial derivatives (Navier-Stokes type) is straight-forward as the compact formulation for derivatives and the method of their computation are similar to those for the first derivatives.

Equation (1) is discretized in time with an explicit Runge-Kutta scheme. The solution is advanced from time level n to time level $n + 1$ in several sub-stages [6]

$$\begin{aligned} H^M &= S_x \frac{\partial U^M}{\partial x} + S_y \frac{\partial U^M}{\partial y} + S_z \frac{\partial U^M}{\partial z} + a^M H^{M-1}, \\ U^{M+1} &= U^M + b^{M+1} \Delta t H^M, \end{aligned} \tag{3}$$

where M is the particular stage number; and the coefficients a^M and b^M depend upon the order of the RK scheme.

To compute spatial derivatives, we solve the sets of independent linear banded systems of equations where each system corresponds to one line of the numerical grid. For example, a system corresponding to a line in the x direction has a scalar tridiagonal matrix $N_x \times N_x$:

$$a_{k,l} Z_{k-1,l} + b_{k,l} Z_{k,l} + c_{k,l} Z_{k+1,l} = f_{k,l}, \tag{4}$$

where $k = 1, ..., N_x$, $l = 1, ..., N_y \times N_z$, $a_{k,l}, b_{k,l}$ and $c_{k,l}$ are the coefficients, $Z_{k,l}$ are the unknown variables, and N_x, N_y and N_z are the number of grid nodes in the x, y and z directions, respectively.

The first step of the Thomas algorithm is LU factorization

$$d_{1,l} = b_{1,l}, \quad d_{k,l} = b_{k,l} - a_{k,l} \frac{c_{k-1,l}}{d_{k-1,l}}, \quad k = 2, ..., N_x, \tag{5}$$

and forward substitution (FS)

$$g_{1,l} = \frac{f_{1,l}}{d_{1,l}}, \quad g_{k,l} = \frac{-a_{k,l} g_{k-1,l} + f_{k,l}}{d_{k,l}}, \quad k = 2, ..., N_x. \tag{6}$$

The second step of the Thomas algorithm is backward substitution (BS)

$$Z_{N_x,l} = g_{N_x,l}, \quad Z_{k,l} = g_{k,l} - Z_{k+1,l} \frac{c_{k,l}}{d_{k,l}}, \quad k = N_x - 1, ..., 1. \tag{7}$$

The coefficients a_k, b_k and c_k are constant for compact schemes; therefore, LU factorization is performed only once and the first step computations include only forward substitution (6).

The standard algorithm for compact numerical solution of the system (1) is performed as follows:
Algorithm **A**
Step 1. Compute the right-hand side of equation (2) using values of the governing variable U from the previous time step.
Step 2. Compute the spatial derivatives solving tridiagonal systems in all spatial directions.

Step 3. Compute the right-hand side of equation (1) using the spatial derivatives computed in Step 2 and update governing variables by Runge-Kutta scheme.
Step 4. Repeat computational Steps 1-3 for all Q stages of Runge-Kutta scheme.
Step 5. Repeat computational Steps 1-4 for all time steps.

This algorithm passes data through the cache to perform Step 1, then it passes data through the cache twice per direction to compute the spatial derivatives (Step 2), and finally the algorithm touches each grid point to compute the temporal update (Step 3). Therefore, the data pass through the cache $2 + 2D$ times, where D is the number of directions.

For explicit schemes, coefficients α and β are equal to zero; therefore, Step 2 in the above algorithm is reduced to local computations of spatial derivatives. Hence, explicit algorithms can be easily written in such a way that the data passes through the cache once.

3. Proposed Cache-friendly Algorithm. In this section we develop a cache-aware compact numerical algorithm where the data passes through the cache only twice.

Let us consider first the two-dimensional case. The wavefronts are defined as subsets of grid nodes (I, J) with $I + J = const$ within a wavefront. If a grid node (I, J) belongs to the front W, its neighbors $(I-1, J)$ and $(I, J-1)$ belong to the previous front WM and two other neighbors $(I, J+1)$ and $(I+1, J)$ belong to the next front WP.

The following Algorithm B sweeps through grid nodes only twice and performs exactly the same computations as Algorithm A (see the previous section).

Algorithm **B**

```
for iwf=IF,...,IL
{
for ign=1,...,IG(iwf)
{
Compute the right-hand side of equation (2)
Compute the forward step of the Thomas algorithm (6) in the $x$ and $y$ directions.
}
}
for iwf=IL,...,IF
{
for ign=1,...,IG(iwf)
{
Perform the backward step of the Thomas algorithm (7) in the $x$ and $y$ directions.
Compute the Runge-Kutta temporal update.
}
}
```

Here, IF and IL are the first and the last wavefronts; indexed variable IG defines the number of grid-nodes within a wavefront iwf. The forward-step computations (6) in both spatial directions use already computed forward-step coefficients in grid nodes $(I-1, J)$ and $(I, J-1)$, whereas the backward-step computations (7) use already computed values in grid nodes $(I+1, J)$ and $(I, J+1)$ (see Figure 1). This algorithm exploits the data-independence of the solution of banded linear systems in different spatial directions, i.e., the systems in the x and y directions are solved simultaneously. Additionally, the right-hand sides of compact

Table 1

Number of data passes through cache for basic compact scheme (Algorithm A), wavefront compact scheme (Algorithm B) and explicit scheme.

	Number of data passes through cache		
Dimension	Algorithm A	Explicit	Algorithm B
2-D	6	1	2
3-D	8	1	2

formulations are computed simultaneously with the forward-step computations and the RK computations are performed immediately after completion of the backward-step computations for a grid node.

This algorithm can be easily expanded to penta-diagonal matrices where two neighboring fronts from either side are used in computations.

This algorithm is expanded to the three-dimensional case where wavefronts represent planes of grid nodes (I, J, K) with $I+J+K = const$. Similar to the 2-D case, three previous neighbors belong to the plane WM (see Figure 2) and the next three neighbors belong to the plane WP. Still, the algorithm sweeps twice through the 3-D array (see Table 1).

4. Extension to Multi-level Cache. Let us first consider two cache levels, primary level L_1 and secondary level L_2. We cover the computational domain with boxes (squares in the 2-D case) that fit the cache size L_1, and renumber the boxes as super-nodes (I_b, J_b, K_b). Then, we define box wavefronts as subsets of boxes where $I_b + J_b + K_b = const$. Each box is considered as a computational domain where the forward and the backward steps of Algorithm B are applied. A computational domain covered with nine cache boxes is shown in Figure 3. These boxes form five wavefronts in the forward and backward directions. The first box wavefront includes the box $(1, 1)$, the second box wavefront includes boxes $(2, 1)$ and $(1, 2)$ and so on. The computed wavefronts within each box are shown for the first two box wavefront levels in the forward and backward directions. Let us define the two-level algorithm as follows:

Algorithm **C**

```
for ibf=IBF,...,IBL
{
for iwf=IF(ibf),...,IL(ibf)
{
Perform forward-step computations of algorithm B
}
}
for ibf=IBL,...,IBF
{
for iwf=IL(ibf),...,IF(ibf)
{
Perform backward-step computations of algorithm B
}
}
```

Algorithm C is consistent, i.e., the previous wavefront is completed by the time forward- or backward-step computations begin for the current wavefront. Algorithm C requires a different way of storage of governing array U than the traditional way of column-by-column placement of array in memory. Instead, here the array should be stored box-by-box.

This algorithm is easily expanded to cases with any number of cache levels. A cache box is covered with smaller boxes of the size of the next (smaller) level of cache. In this case the number of nested loops is equal to the number of cache levels. The inner loop sweeps through grid nodes that belong to a wavefront, whereas outer loops sweep through box wavefronts corresponding to different levels of cache.

5. Conclusion. The cache-aware compact numerical algorithm has been developed. The data pass through cache only twice for 2-D and 3-D cases. The algorithm is expanded to any number of cache levels. Interaction of the proposed algorithm with compilers will be studied in our future research.

REFERENCES

[1] S. K. Lele, Compact Finite Difference Schemes with Spectral Like Resolution, *Journal of Computational Physics*, **103** (1992), pp. 16-42.

[2] T. Colonius, Lectures on Computational Aeroacoustics, presented at the lecture series on Aeroacoustics and Active Noise Control, von Karman Institute of Fluid Dynamics, 1997, `http://green.caltech.edu/~colonius.`

[3] C. K. W. Tam and J. C. Webb, Dispersion-relation-preserving Finite Difference Schemes for Computational Acoustics, *Journal of Computational Physics*, **107** (1993), pp. 262-281.

[4] R. Y. Kain, *Advanced Computer Architecture (A System Design Approach),* Prentice-Hall, Inc., 1996.

[5] Tien-Pao Shih, Goal-directed Performance Tuning for Scientific Applications, Ph.D. Thesis, University of Michigan, 1996.

[6] R. V. Wilson, A. O. Demuren and M. Carpenter, High-order Compact Schemes for Numerical Simulation of Incompressible Flows, ICASE Report No. 98-13, 1998.

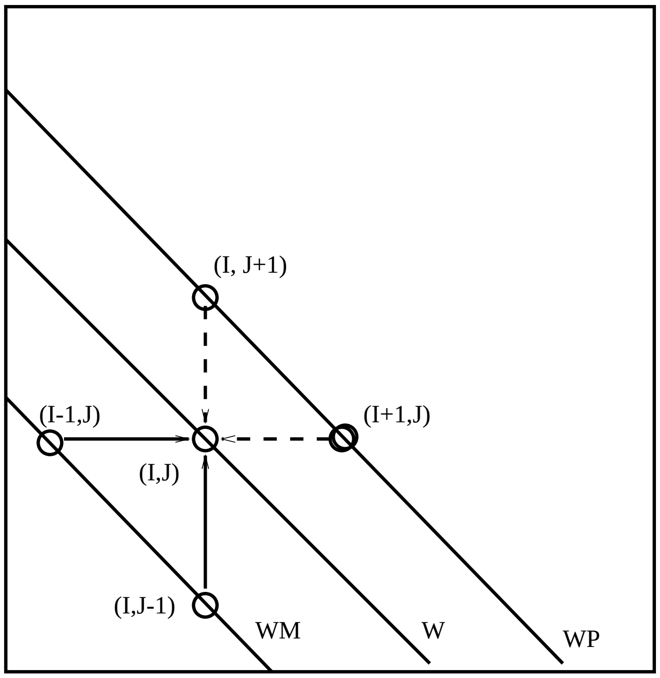

FIG. 1. *Wavefront algorithm in a 2-D case, where WM, W and WP are three consequent wavefronts. Solid arrows represent forward-step computations and dashed arrows represent backward-step computations*

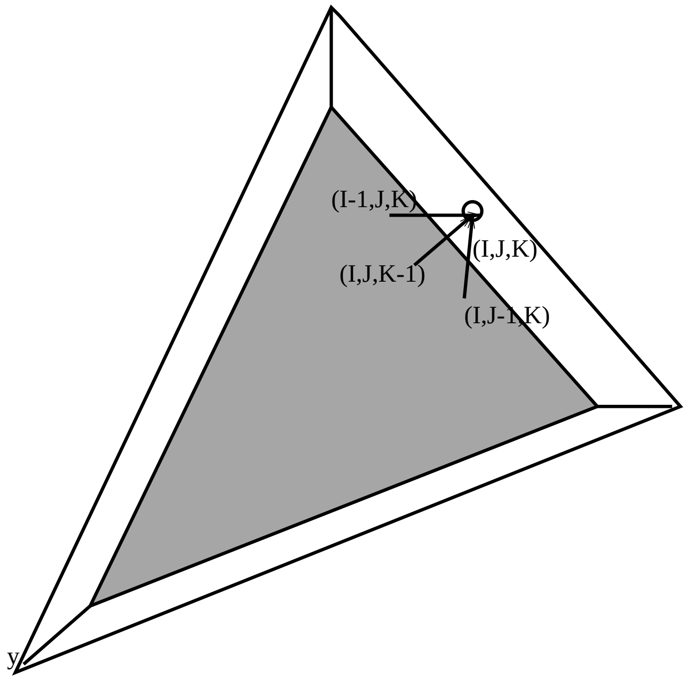

FIG. 2. *Wavefront algorithm in a 3-D case. Shaded plane is the previous wavefront.*

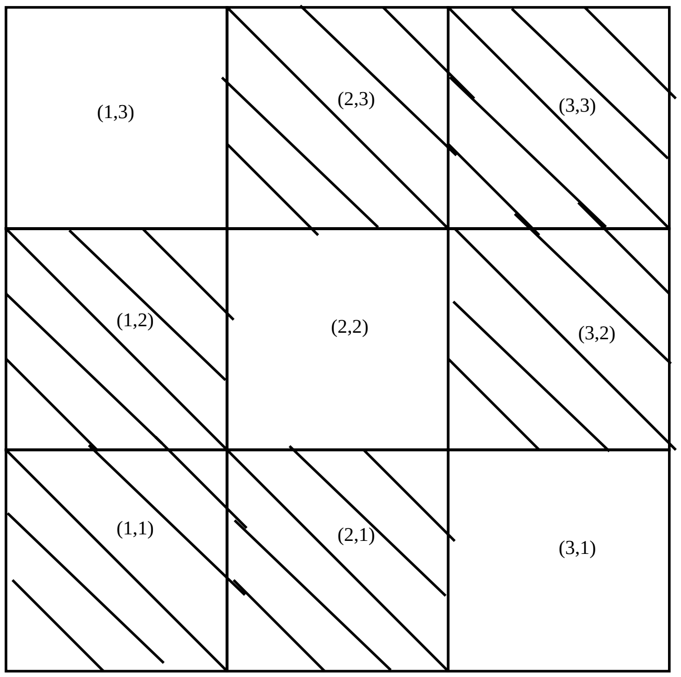

FIG. 3. *Domain partitioning for two-level cache.*