On a High-Order Compact Scheme and Its Utilization in Parallel Solution of a Time-Dependent System on a Distributed Memory Processor

Okon H. Akpan

Bowie State University, Bowie MD 20715, USA
oakpan@cs.bowiestate.edu

Abstract. The study resulting in this paper applied a parallel algorithm based on a fourth-order compact scheme and suitable for parallel implementation of scientific/engineering systems. The particular system used for demonstration in the study was a time-dependendent system solved in parallel on a 2-head-node, 224-compute-node *Apple Xserve G5* multiprocessor. The use of the approximation scheme, which necessitated discretizing in both space and time with h_x space width and h_t time step, produced a linear tridiagonal, *almost-Toeplitz* system. The solution used p processors with p ranging from 3 to 63. The speedups, s_p, approached the limiting value of p only when p was small but yieldd poor computations errors which became progressively better as p increases. The parallel solution is very accurate having good speedups and accuracies but only when p is within reasonable range of values.

1 Introduction

Finite difference methods are among the commonest approximation schemes used for numerical solution of ordinary and partial differential equations, mainly, because of their simplicity of use and the fact that they lend themselves quite easily to the Taylor series analysis of any incurred errors. Other approximation methods exist, and they include *finite elements*, *finite volumes*, and *spectral* methods. While there are a number of problems, for example, *elliptic* systems, which can be solved with low-order approximation methods (second or lower) with reasonable accuracies, there is also a large class of problems, including those of *acoustics* [1,2,3], and of *fluid dynamics* [5,6,7,9,13,14], the solutions of which typically require higher order approximation solution schemes for higher levels of accuracy.

A solution method is said to be of order h^n, where h is the mesh size of the problem domain, when its truncation error varies as h^n. In a second order approximation scheme, for instance, where error $\approx h^2$, halving the mesh size (h) reduces the error by a factor of approximately four. Low order approximations generally require *compact stencils* which utilize three nodal points in any direction. Thus 1-D, 2-D, or 3-D compact stencils require 3, 3×3, or $3 \times 3 \times 3$

K. Li et al. (Eds.): NPC 2007, LNCS 4672, pp. 1–12, 2007.

grid nodes respectively. Any approximation method which involves grid nodes outside those of a compact stencil is said to be *non-compact*. Higher order finite difference approximations (that is, those with approximation error $\approx h^n$, where $n > 2$) are possible but these methods typically require non-compact stencils, and application of non-compact stencils at or near boundaries of the problem domain usually requires inclusion of fictitious nodes thus complicating the resulting numerical formulations, and the usual consequences of those complications include increases in the overall number of grid points as well as increases in the bandwidths of the resulting system matrices. The latter problem, namely, increases in the bandwidths, precludes the use of implicit methods for solution of the resulting systems because those systems are usually not of *tridiagonal* form. Lastly, non-compact approximation methods do not easily allow for approximations with non-uniform grids thus excluding solution of certain problems notably *boundary-layer*, *driven-cavity*, and *turbulent flow* problems which typically involve wide ranges of space and time scales.

Approximation schemes which must retain a high accuracy without incurring most of the complications of non-compact methods must of necessity be compact, although they too must somehow deal with the problems imposed by having to apply the stencils at or near the problem boundaries and to come up with a numerical formulation with a high accuracy result. Gustafsson [15,16] has demonstrated that such a compact method must have *boundary closures* in which the boundary- and near-boundary points have about the same accuracy as that of the interior points or at most one order less. Abarbanel and Chertock [17] have successfully used a five-order *boundary closure* in their solution of hyperbolic initial boundary value problems in 1-D and 2-D spatial domains using a sixth-order compact difference scheme. In summary, a compact differencing scheme requires more work per grid point, but the result is higher approximation accuracy, a few grid points to compute with, and less computer memory requirements to store the computed result. As a result, compact approximation schemes are more efficient than both non-compact methods of the same order and also than low-order solution methods in general [18,19,20]. As stated by Orszak [21], at the cost of slight computational complexity, fourth-order compact schemes can achieve results in the 5% accuracy range with approximately half the spatial resolution in each space direction when compared with the second-order schemes.

The compact difference approximation methods that treat the approximated function and its derivatives as unknowns at grid points are fourth-order and they produce tridiagonal systems. These compact schemes generally fall into two classes. The first class consists of those solution methods which are best suited for uniform grids, and they include the Kreiss and Oliger's approximating scheme [18,4], and the Mehrstellen's scheme [1]. The second class consists of methods that allow for variable grids. These include the cubic spline methods of Rubin and Khosla [23,24] and the hermitian finite difference methods of Adam's [25,26]. Also belonging in this class are the fourth-order compact schemes for solution of *incompressible viscous flow* problems the most notable of which being the fourth-order, three-nodal stencil compact scheme of Gupta [27,28] useful for

solution of convection diffusion equations. Gupta's compact scheme, which does not seem to suffer excessively from the spurious oscillatory behavior as do other methods, has been applied by Yavneh to solve *convection-diffusion problems* [29], and also by Weinan to solve unsteady viscous flow problems [30].

While over the years there have been a large number of approximating schemes constructed for solution of many classes of scientific problems, the literature has reported relatively a few successful solution paradigms designed to fully exploit the capabilities of modern supercomputers for efficient solution of problems with these approximating schemes. The reported paradigms all attempt to solve complicated scientific problems in parallel harnessing the supercomputing resources in the process, and, if all goes well, the result is often an efficient solution in terms of both time (speed) and space (memory) requirements. The earliest parallel solution methods were designed for solution of *fine-grained* problems, that is, problems with $n \approx p$, where n is the size of the problem and p the number of processors (of a supercomputer), and, also, the methods were based on high-speed solution using tridiagonal solvers. The most known of these methods include the *recursive-doubling reduction* method of Stone [31] and its improved version [32], the *odd-even* or *cyclic reduction* technique of Hockney [33,34], and recently, the *prefix* scheme by Sun [35,36], which is a variation of the *cyclic reduction* method. Each of the cited parallel solution method is capable of solving n-dimensional tridiagonal system in $\mathbf{O}\log(n)$ time using n processors. More recent efforts are geared toward problem solution in various parallel computing environments, not just *fine-grained*. These include the methods of Lawrie and Sameh [37], Wang [38] designed for *median-* (that is, $p < n$) to *course-grained* (that is, $p \ll n$).

The thrust of this study is to apply the Kreiss and Oliger's fourth-order compact scheme to solve in parallel a time-dependent parabolic differential equation with Neumann boundary conditions. The focus is strictly computational. It is not concerned with developing another solution methodology, nor with modification of an existing one to solve any problem. Given the dearth of utilization of supercomputing resources to solve the kinds of problems mentioned here, the study clearly and painstakingly demonstrates the steps needed to numerically solve a given problem in parallel on a supercomputer in full consideration of both the inherent parallel properties of the problem and the architecture of the supercomputer involved.

2 Kreiss and Oliger's Fourth Order Compact Scheme

Kreiss and Oliger [18] suggested a fourth-order compact difference method in which the first ($f^{'}$) and second ($f^{''}$) derivatives for constant mesh size (h_x) are approximated by

$$f_n^{'} = \left(\frac{D_o}{1 + \frac{1}{6} h_x^2 D_+ D_-} \right) f_n, \tag{1}$$

and

$$f_n^{''} = \left(\frac{D_+ D_-}{1 + \frac{1}{12} h^2 D_+ D_-} \right) f_n, \tag{2}$$

where

$$D_o f_n = \frac{1}{2h_x} \left(f_{n+1} - f_{n-1} \right),$$

$$D_+ f_n = \frac{1}{h_x} \left(f_{n+1} - f_n \right),$$

$$D_- f_n = \frac{1}{h_x} \left(f_n - f_{n-1} \right).$$

Multiplying equations (1) and (2) with the respective denominators and simplifying yields

$$\frac{1}{6} f_{n-1}^{'} + \frac{2}{3} f_n^{'} + \frac{1}{6} f_{n+1}^{'} = \frac{1}{2h_x} \left(f_{n-1} \right), \tag{3}$$

$$\frac{1}{12} f_{n-1}^{''} + \frac{5}{6} f_n^{''} + \frac{1}{12} f_{n+1}^{''} = \frac{1}{2h_x^2} \left(f_{n+1} - 2f_n + f_{n-1} \right). \tag{4}$$

Each of the above approximations utilizes a 1-D 3-node *compact* stencil

$$\left\{ 1 \quad a^{'} \quad 1 \right\}$$

where $a^{'} \in \{4, 10\}$. When (3) and (5) are correctly applied to approximate first or second order partial derivatives respectively, the system

$$\begin{bmatrix} c_0 & 1 & & & \\ 1 & c & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & c & 1 \\ & & & 1 & c_0^{'} \end{bmatrix} \mathbf{x} = \mathbf{A}x = \mathbf{k}, \tag{5}$$

results in which c_0, $c_0^{'}$ result from boundary conditions, and $x \in \{f^{'}, f^{''}\}$, $c \in \{4, 10\}$, whereas the conventional fourth-order scheme results in systems with larger bandwidths. A conventional fourth-order scheme approximating the same system with, for instance,

$$f_n^{''} = \frac{1}{12h_x^2} \left(-f_{n-2} + 16f_{n-1} - 30f_n + 16f_{n+1} - f_{n+2} \right)$$

applies a *non-compact* stencil (5-node stencil:

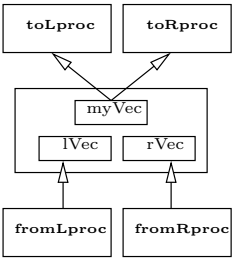
$$\{ -1 \quad 16 \quad -30 \quad 16 \quad -1 \} \tag{6}$$

in this case).

3 Parallel Solution

There are a number of parallel algorithms for solution of system (5). The most known among them are the *cyclic reduction* algorithm by Hockney, [34], the *recursive doubling* by Stone [32], and the *prefixed* method by Sun [36]. In this study, we used the *cyclic reduction* method as we found this algorithm to be very easily implementable on our supercomputing system.

A parallel *cyclic reduction*, which assumes availability of p processors where p and n (the system's order) need not be equal, uses l reduction levels, $1 \leq l \leq log2(n^{'})$. Every process, p_i, has 3 4-vectors, *myVec*, *leftVec*, and *rightVec*, where for any such vector (*vec*), $vec[0] \leftarrow$ d, $vev[1] \leftarrow$ e, $vec[2] \leftarrow$ f, and $vec[3] \leftarrow$ k, where d, e, f are the row components of **A** and k of (5). At a reduction level l, there is data transfer to process p_i from 2 processes with ranks *fromLProc* and *fromRProc*, and from p_i to 2 processes with ranks *toLProc* and *toRProc*. The interprocessor data transfers can be facilitated with the use of data structure:



Another data structure is the array **D** which holds rows $\boxed{\text{d}}\boxed{\text{e}}\boxed{\text{f}}\boxed{\text{k}}$ of **A** and **k** of (6). In this implementation, the first row of **D** contains a special vector $\boxed{0}\boxed{1}\boxed{0}\boxed{0}$, the second row of **D** contains the first row of **A** and of **k**, and so forth. Without the first row of **D** containing the special vector, the inter-process data transfers may not be correct or, if correct, at an unacceptable cost.

4 Computational Environment

The computational environment used for all parallel computations in this study is the Bowie State University's supercomputer facility locally dubbed *Xseed*. At the core of *Xseed* is Mac OS X Server operating system running a cluster configuration presently consisting of 2 ultradense, 1U rackmount XServe G5 servers and 224 G5 compute nodes each of which a dual, 64-bit, 2.0 GHz processor with 2 GB RAM and local store of 80 GB capacity. The 224 G5 compute nodes are clustered into shared-memory nodes with dual gigabit Ethernet on the motherboard which, in combination with the high bandwidth system control, effectively reduces contentions in both the network and i/o traffics. The 2 servers are the *XSeed's* administrative nerve center which is responsible for coordinating all computations and data routings, as well as managing data storage. *XSeed* is capable of theoretical performance rate of 2.104 Tflops and has a combined work storage space of over 6 TB.

XSeed is ideal for scientific and high performance computing as well as image rendering operations. Each G5 processor's superscalar, super-pipelined architecture supports up to 215 simultaneous instructions with a high-bandwidth execution core offering over 12 discrete functional units, including dual floating-point units and dual integer units, to process an immense number of instructions in parallel with 64-bit precision on 64-bit wide paths! Also *XSeed* is fully grid-enabled running a complete package of functioning *Globus Alliance*'s Globus Toolkit middleware (version 4.0.1). Thus packed with enormous floating-point computational muscle, plenty of memory for work space, and updated Grid infrastructure, *XSeed* is truly a distributed, self-contained, net-centric computing environment primed to harness (and share) federated computational resources globally for parallel solution of any type of scientific and engineering problems.

5 Application

An application of the Kreiss and Oliger's compact difference scheme for approximation solutions was demonstrated with a parabolic partial differential equation

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le \widehat{x}, \quad \widehat{x} > 0, \, t \ge 0, \tag{7}$$

having an initial condition prescribed at $t = 0$,

$$u(x, 0) = f(x), \quad 0 \le x \le \widehat{x}, \tag{8}$$

and Neumann boundary conditions (for positive κ)

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = g_1(0, t), \tag{9}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=\widehat{x}} = g_2(\widehat{x}, t). \tag{10}$$

To solve (7) computationally, discretization in both time and space is needed. Discretization in time is done by performing the Taylor's series expansion of (7) in time to obtain

$$u_x^{n+1} = u_x^n + \kappa h_t \frac{\partial^2 u_x^n}{\partial^2 x^2}, \tag{11}$$

in which n corresponds to a time level, and x a spatial location. Introducing a spatial mesh width, h_x, obtainable by dividing the given interval $[0, \, \widehat{x}]$ into N sub-intervals, that is, $h_x = \frac{\widehat{x}}{N}$ (where $N > 0$ is some integral value), and also the notations

$$x_i = ih_x, \quad t_j = jh_t, \tag{12}$$

where $i = 0, \cdots, N$, and $j = 0, 1, \cdots$, then (11) can be rewritten as

$$u_i^{j+1} = u_i^j + \kappa h_t \left.\frac{\partial^2 u_x^j}{\partial^2 x^2}\right|_{x \, = \, ih_x}. \tag{13}$$

The solution of (13) involves discretizing the spatial derivative $\frac{\partial^2}{\partial x^2}$ with the compact second-derivative operator provided in (2) to obtain the tridiagonal system (5), solving the resulting system by the parallel cyclic reduction whose code design and implementation are detailed above, substituting that solution into (13), and, finally, solving the resulting system to obtain u_i^{j+1}.

6 Experiments and Results

We solved (13) for case of $\widehat{x} = 2$, $g_1(0,t) = g_2(2,t) = 0$ using $0 < \kappa\frac{h_t}{h_x} \leq \frac{1}{2}$ (to avoid onset of instabilities in order to focus on the numerical solution, the objective of this study) for several h_x. A solution with a given value of p required time marching, and each time-step advance required a single compact solve in parallel. In order to be able to compare the parallel solutions to sequential solutions, we also solved (13) at the same time step, h_t, using the best sequential method, the LU decomposition:

$$\mathbf{L}g = k, \tag{14}$$

$$\mathbf{U}u = g, \tag{15}$$

where $\mathbf{L}$ and $\mathbf{U}$ are the lower and the upper triangular matrices respectively. Lastly, to compare approximate solutions to the *true* solutions, we first solved (7) analytically to obtain

$$u = 800\sum_{n=0}^{\infty}\frac{1}{n^2(2n+1)^2}cos\frac{(2n+1)(\pi(x-1)}{2}e^{-0.3738(2n+1)^2t},$$

and, second, we encoded and then solved the equation at time levels t to obtain analytical (*true*) results, u_{actual}. Executions were carried out with the number of processors, p, equal to 3, 7, 15, 31, and 63. All codes – parallel cyclic reduction and LU decomposition – were appropriately *instrumented* for execution parameters notably time. The results are given in Figure 4, Figure 5, and Figure 6 below. The parameters in the the solution figures are explained as follows:

1. p is the number of processors, n the order of the system, hence, $p = n = n'-1$, $n' = \frac{\widehat{N}}{h_x}$.
2. s_p is *speedup* given as $s_p = \frac{t_1}{t_p}$ $(1 \leq s_p \leq p)$, where t_1 is the time it took to execute the resulting tridiagonal system by the LU decomposition, t_p the time of solution of the same system with the parallel cyclic reduction with p processors.
3. ϵ_p is *efficiency* computed as $\epsilon_p = \frac{s_p}{p}$ $(\frac{1}{p} \leq \epsilon_p \leq 1)$.
4. Given h_x, the corresponding time step, h_t, is computed as $h_t = \frac{1}{2}\frac{h_x^2}{\kappa}$.
5. $Max.E_p$ is the *maximum relative error* determined from all solutions, u, during a system solve with p processors at a given time level. Here, $E_p = \frac{u_{actual}-u_{approx}}{u_{actual}}$, where u_{actual}, u_{approx} are the analytical solution and approximate solution respectively.

h_x	h_t	p	s_p	ϵ_p	$Max.E_p$
0.2500	0.8250	3	2.9103	0.9701	0.072
0.2500	0.2063	7	6.5727	0.9390	0.051
0.1250	0.0515	15	13.6581	09105	0.020
0.0625	0.0129	31	27.2871	0.8802	0.001
0.0313	0.0032	63	54.1605	0.8597	0.000

Fig. 1. Execution results for $p = 3, 7, 15, 31$, and 63

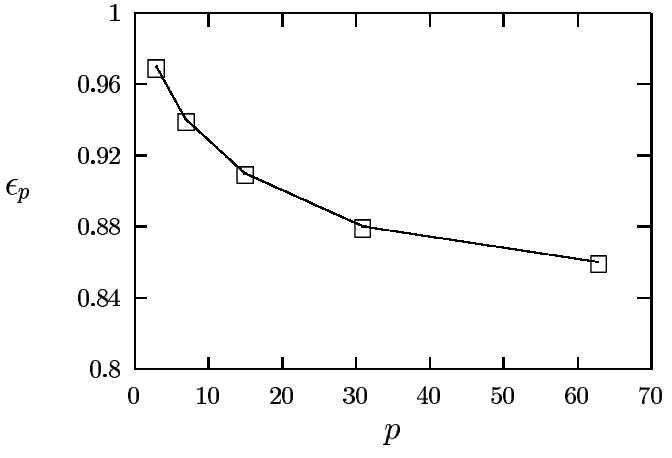


Fig. 2. Results for $p = 3, 7, 15, 31$, and 63

7 Observations

1. The Kreiss and Oliger's fourth order compact scheme produced very accurate solution results only when solved with reasonably small spatial and time steps and when the solution was within the *stability* regimes, that is, $\kappa\frac{h_t}{h_x^2} \leq \frac{1}{2}$, but inaccurate results with error as large as 10% when the steps in time and space became large. More specifically, when h_x became large (implying small p because $p = \frac{\widehat{N}}{h_x} - 1$), the results at all time levels were unacceptably poor with the worst errors occuring at $h_x = 0.5$ or $p = 3$, that is, $Max.E_3 \approx 10\%$. But as h_x became small, the errors became correspondingly small until perfect results occured at about $h_x = 0.0313$ with $p = 63$, that is, $Max.E_{63} \approx 0\%$).
2. When $p \leq 7$, the speedups s_p were very high approaching their limiting values of p, but as p increased (with decreasing h_x), the speedups became smaller tending to a limiting (but acceptable) value of about 82.5%.
3. For $n > 15$, the parallel cyclic reduction completed reduction computation, (that is, with every process i obtaining its final vector, $\boxed{\text{d}}\boxed{\text{e}}\boxed{\text{f}}\boxed{\text{k}}$, at $\log_2 n - 1$ levels, that is, one level short of the theoretical maximum of $\log_2 n$.

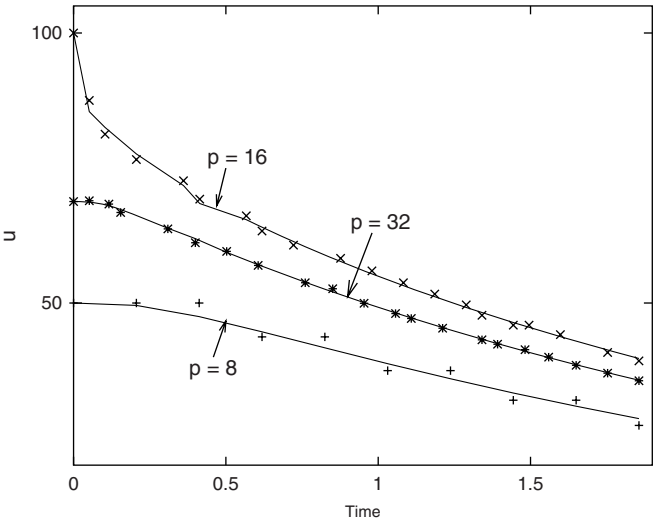


Fig. 3. Efficiency versus *Time*

8 Conclusion

1. Solution of the tridiagonal system resulting from application of the Kreiss and Oliger's compact approximation scheme produced very high speedups but very inaccurate results at small values of p, and produced low speedups but very accurate results at high value of p. The reason for this is that, although the parallel cyclic reduction exhibits 100% load balance at every reduction level,
   (a) there is no parallelism between reduction levels, that is, level-level computation is purely *sequential* supported by *MPI_Barrier* and similar freeze commands which ensure completion of all reduction operations by all processes at the current level before the same operations are started at the next reduction level,
   (b) even at a reduction level where computation is parallel at 100% load balance, the interprocessor inter-communication is quite high

   thus contributing to low speedups and corresponding low efficiencies.
2. The time for inter-process communication (for data exchanges), t_{comm}, which was quite significant when p was high, dominated the actual computation time, t_{calc}, at all reduction levels and this accounted for low speedups and efficiencies at high p, but, at low p, t_{calc} dominated t_{comm} thus resulting in high speedups but at the expense of accuracy.
3. The solution methods used in this study are highly recommended for parallel solution of time-dependent systems provided such solutions use reasonable number of processors and appropriate steps in time and space whose range can be determined from observations. (For the system used in the study,

solutions at $7 \leq p \leq 31$ were deemed reasonable with error range of $10\% \leq Max.E_p1 \leq 1\%$, speedup range of $95\% \leq s_p \leq 88\%$). On the other hand, solutions with $p < 7$ yielded very inaccurate results, while solutions with $p \geq 31$ were *excessive* resulting in waste of computational resources with little gain in accuracy.)

4. On account of the fact that the cyclic reduction is best suited for *fine-grained* parallel solution such as the one used in this study, we conclude that the parallel solution of problems approximated with the fourth compact scheme would yield even better results if solved with a *massively parallel processor* instead of a *distributed processor* as is the case in this study.

Acknowledgment

I am very grateful to Dr. Sadanand Srivastava, professor and chair of *Computer Science Department*, and to Dr. Joan S. Langdon, Dr. Manohar Mareboyana, and Dr. Paul Chi who read, edited this paper and offered other invaluable help without which a timely completion of this study would have been impossible. I am also indebted to Dr. Mark A. Matties, director of the *Xseed* project, whose technical support and suggestions significantly propelled this study to its successful completion.

References

1. Krause, E., Kordulla, W.: Fourth-Order Mehrstellen Integration for Three-Dimensional Turbulent Boundary Layers. In: AIAA Comp. Fluid Dyn. Conf. (1973)
2. Gustaffsson, B.: Time Compact Difference Methods for Wave Propagation in Discontinuous Media, Tech. Reports., Dept. of Inf. Tech., Uppsala Univ., 2003-023 (2003)
3. Cohen, G., Joly, P.: Construction and Analysis of Fourth-Order Finite Difference Schemes for the Acoustic Wave Equation in Nonhomogeneoues Media. SIAM J. Num. Anal. 33 (1996)
4. Hirsh, R.: Higher Order Accurate Difference Solutions of Fluid Mechanics Problems by a Compact Differencing Technique. J. Comp. Phys. 19, 1 (1975)
5. Tolstykh, A.I.: Higher Order Accurate Non-Centered Compact Difference Schemes for Fluid Dynamics Applications. Series on Adv. in Math. for Appl. Sc. 21 (1994)
6. Ratnesh, K., Shukla, S.: Derivation of High-Order Compact Finite Difference Schemes for Non-Uniform Grid Using Polynomial Interpolation. Jour. Comp. Phys. 204, 2 (2005)
7. Joslin, R., Streett, C., Chang, C.: Validation of Three-Dimensional Incompressible Spatial Direct Numerical Simulation Code, NASA Tech. Report, TP-3025, NASA Langley Research Center (1992)
8. Spotz, W., Garey, G.: High-Order Compact Scheme for the Stream-function Vorticity Equations. Int'l J. for Num. Method. in Eng. 38, 20 (1995)
9. Haiwei, S., Kang, N., Zhang, J., Carlson, E.: A Fourth-Order Comapct Difference Scheme on Face Centered Cubic Grids with Multigrid Method for Solving 2D Convection Diffusion Equation. Math. and Comp. in Simul. 63, 6 (2003)

10. Lambiotte, J., Voigt, R.: The Solution of Tridiagonal Linear Systems on CDC Star-100 Computer. ACM Trans. Math. Soft. 10 (1984)
11. Ortega, J., Voigt, R.: Solution of Partial Differential Equations on Vector and Parallel Computers. SIAM Review (1985)
12. Spotz, W.: Accuracy and Performance of Numerical Wall-Boundary Conditions for Steady 2D Incompressible Stream-Function Vorticity. Int'l J. for Num. Meth. in Fluids 28, 4 (1998)
13. Berikelashvili, G., Gupta, M.M., Mirianashvili, M.: Convergence of Fourth Order Compact Difference Schemes for Three-Dimensional Convection Diffusion Equations. SINUM (SIAM Jour. on Num. Anal.) 45, 1 (2007)
14. Spotz, W., Garey, G.: Formulation and Experiments with High-Order Compact Schemes for Nonuniform Grids. Int'l J. for Heat & Fluid Flow 8, 3 (1998)
15. Gustafsson, B.: The Convergence Rate for Difference Approximations to Mixed Initial Boundary Value Problems. Math. Comp. 29, 396 (1975)
16. Gustafsson, B.: The Convergence Rate for Difference Approximations to General Mixed Initial Boundary Value Problems. SIAM J. Numer. Anal. 18, 179 (1981)
17. Abarbanel, S., Chertock, A.: Strict Stability of High-Order Compact Implicit Finite Difference Schemes: The Role of Boundary Conditions for Hyperbolic PDEs. J. of Comp. Phys. 160 (2000)
18. Kreiss, O., Oliger, J.: Methods for the Approximate Solution of Time-Dependent Problems, GARP Report, No. 10 (1973)
19. Zhang, J.: Multigrid Solution of the Convection-Diffusion Equation with High Reynolds Number. In: Proc. Copper Mountain Conf. on Iter. Meth. (1996)
20. Zhang, J.: On Convergence and performance of iterative Methods with Fourth-Order Compact Schemes. Num. Meth. for Partial Diff. Eqns. 14 (1998)
21. Orszak, S., Isreal, M.: Numerical Simulation of Viscous Incompressible Flows. Annual Rev. of Fluid Mech. 6 (1974)
22. Hirsh, R.: Higher Order Accurate Difference Solutions of Fluid Mechanics Problems by a Compact Differencing Technique. J. Comp. Phys. 19, 1 (1975)
23. Rubin, S., Khosla, P.: High-Order Numerical Solutions Using Cubic Splines, NASA CR-2653 (1976)
24. Rubin, S., Khosla, P.: High-Order Numerical Methods Derived from Three-Point Polynomial Interpolation, NASA CR-2735 (1976)
25. Adam, Y.: A Hermitian Finite Difference Method for the Solution of Parabolic Equations. Comp & Math. Appl. 1 (1975)
26. Adam, Y.: Highly Accurate Compact Implicit Mehods and Boundary Conditions. J. Comp. Phys. 24 (1977)
27. Gupta, M., Manohar, R., Stephenson, J.: A Single Cell High Order Scheme for the Convection-Diffusion Equation with Variable Coefficients. Int'l J. Num. Meth. Fluids, vol. 4 (1984)
28. Gupta, M.: High Accuracy Solutions of Incompressible Navier-Stokes Equations. J. Comp. Phys. 93 (1991)
29. Yavneh, I.: Analysis of Fourth-Order Compact Scheme for Convection-Diffusion. J. Comp. Phys. 133 (1997)
30. Weinan, E., Liu, J.: Essentially Compact Schemes for Unsteady Viscous Incompressible Flows. J. Comp. Phys. 126 (1996)
31. Stone, H.: An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. J. of ACM 20 (1973)
32. Stone, H.: Parallel Tridiagonal Solvers. Assoc. Comp. Mach. Trans. Soft 1 (1975)
33. Hockney, R.: A Fast Direct Solution of Poisson's Equation using Fourier Analysis. J. of ACM 12 (1965)

34. Hockney, R., Jeshoppe, C.R.: Parallel Computers 2: Architecture, Programming, and Algorithm, 2nd edn. Inst. of Physics Pub., Bristol, Philadelphia (1988)
35. Sun, X., Joslin, R.: A Parallel Algorithm for Almost-Toeplitz Tridiagonal Systems. Int'l Jour. of High Speed Comp. 4 (1995)
36. Sun, X., Gustafson, J.: Toward a Better Parallel Performance Metric. Par. Comp. 17 (1991)
37. Laurie, D., Sameh, A.: The Computation and Communication Complexity of a Parallel Banded System Solver. ACM Trans. Math. Soft. 10 (1984)
38. Wang, H.: A Parallel Method for Tridiagonal Equations. ACM Trans. Math. Soft. 7 (1981)