MULTIDIMENSIONAL PARALLEL SPECTRAL SOLVER FOR NAVIER-STOKES EQUATIONS

AMIR AVERBUCH*, LUDIMILA IOFFE†, MOSHE ISRAELI†, AND LEV VOZOVOI*

Abstract. In this paper we present a survey and new parallel algorithms for the solution of the incompressible two- and three-dimensional Navier-Stokes equations. We present a high-order parallel algorithms which require only minimum inter-processor communication which is dictated by the physical nature of the problem at hand. The parallelization is achieved via domain decomposition. We consider computational regions in the form of a 2-D or 3-D periodic box which is decomposed into parallel strips (slabs) and cells. The time discretization is performed via the semi-implicit splitting scheme of [33]. The splitting procedure in time results in solving in each time step two global elliptic equations: the Poisson equation for the determination of the pressure field and the Helmholtz equation for the implicit viscous step. The discretization in space is performed using the Local Fourier Basis method [23] and the multidomain local Fourier (MDLF) method that was developed in [1–3, 7–9, 12, 13, 29, 30, 37, 40]. Therefore, in the direction across the strip or cells we use the Local Fourier Basis technique which involves the overlapping of the neighboring subdomains and smoothing of local functions across the interior boundaries (interfaces). The discretization in the periodic directions is performed by the standard Fourier method. To avoid the Gibbs phenomenon, the global functions are decomposed into smooth local pieces. Then the Fourier method is applied on the extended local subdomains with spectral accuracy. The continuity conditions on the interfaces are enforced by adding homogeneous solutions. Therefore, the matching of the local solutions is performed by adding properly weighted interface Green's functions. Their amplitudes are found in terms of the jumps of the solution and its first derivatives at the interfaces for the Helmholtz equation. Such solutions often have fast decay properties which can be utilized to minimize interprocessor communication. In effect, the predominant part of the computation is performed independently in the subdomains (processors) by using only local communication.

We consider the complete Navier-Stokes system. The solution of the Poisson equation for pressure has the potential to degrade the performance and the achieved speedup of a parallel algorithm due to the global nature of this equation that necessitates global communication among the processors. However, as we showed in [5, 15, 40] only a few lowest harmonics require the global data transfer whereas the rest of harmonics can be treated locally. Therefore, most of the communication that is required for parallelization of the Navier-Stokes solver using the MDLF method is mainly local between adjacent subdomains. Moreover, the percentage of the time spent in global communication reduces as the size of the problem increases. Thus, the present parallel algorithm is highly scalable.

We proposed a new algorithm for the parallel solution of two-dimensional Navier-Stokes type equation with constant and non-constant coefficients which is mapped onto cell topology. This is a further development in the application of the local Fourier methods to the solutions of PDE's in multidomain regions. The extension of the above solution to problems with non-constant coefficients is suggested via spectral multidomain preconditioner. In addition, an appropriate Alternate Direction Implicit (ADI) scheme was applied. It enables the reduction of a 2-D problem to a collection of uncoupled 1-D ODE's. In effect, the 1-D solver becomes the basic routine to solve a 2-D problem using splitting of differential operators by ADI.

*School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, ISRAEL.
†Faculty of Computer Science, Technion, Haifa 3200, ISRAEL.

The Navier-Stokes solvers were implemented on four MIMD multiprocessors: 26-processors MEIKO, 60-processors IBM SP2, a 20-processors MOSIX, and a network of 10 Alpha workstations. The implementation on the last three machines used the same code written with PVM (parallel virtual machine) software package.

Key words. Spectral method, local Fourier basis, Navier-Stokes, Helmholtz and Poisson equations, parallel processing, domain decomposition, local and global communication, ADI, spectral multidomain preconditioner, SP2, MOSIX, network of PCs.

AMS(MOS) subject classifications. 65Y05, 58C40, 62M15, 35Q30, 76D05, 65M55.

1. Introduction. The numerical solution of multidimensional nonlinear evolution equations (and especially the particularly difficult case of the incompressible Navier-Stokes equations) is a heavy computational task and it is one of most challenging problems for high performance computing. The direct numerical simulation (DNS), i.e. solution of the complete Navier-Stokes equations, is out of reach for even today's fastest supercomputers. The major problem in applying DNS and related methods (such as based on the large eddy simulation model) is the huge amount of computational memory and CPU time needed for sufficiently accurate turbulence computation. The computational work requirement, which is the number of modes (or grid nodes) times the number of time steps, scales roughly like Re^3 [32] (the Reynolds number Re is a dimensionless measure of the degree of nonlinearity of a flow). Zang and Kriest [41] studied experimentally transient flows at subcritical and supercritical Reynolds numbers. They did a thorough investigation of the resolution necessary to capture certain fine details in the flow and found that extremely high resolution is necessary at the late transition phase. However, modern serial or vector computers are able to perform simulations with Reynolds numbers of several hundreds only, which is far from those required in many applications.

It is widely recognized that large-scale scientific computing can be realized only on massively parallel computers, especially the flexible MIMD-type message-passing systems, which are considered to be the most promising strategy in supercomputer hardware design. Rapid development of new architectures, processors, and communication networks makes such computers, with their distributed memory, more powerful than shared-memory machines.

The ability to maintain efficiency at a fixed size by simultaneously increasing the number of processors and the size of the problem is the main feature of *scalable* parallel system. The *scalability* of a parallel system measures its capacity to increase speedup in proportion to the number of processors. It reflects a parallel system's ability to use increasing processing resources effectively. Our goal is to demonstrate that the proposed parallel solvers for the 2-D and 3-D Navier-Stokes equations are scalable in the above sense.

The high performance of parallel machines can be realized only if efficient scalable parallelizing algorithms are supplied, otherwise, communication and synchronization overheads destroy considerably the parallel performance. The efficiency of a parallel algorithm implies:

- An efficient (fast, high order accuracy, stable time integration) serial solver underlying the parallel code.
- An efficient (low-communication, equal load balance) parallelizing approach.

Among the different discretization methods applied to simulation of flows the most prominent distinction is between those with formally infinite order of accuracy (usually denoted spectral or pseudospectral) and finite order accuracy (finite difference, finite volume, finite element methods). High order methods are preferred for turbulence computation since they converge rapidly as the number of degrees of freedom increases. As a consequence, for moderate to high accuracy a smaller number of degrees of freedom may be needed, especially in multi-dimensional problems. Simple fluid dynamics models indicate that for moderate accuracy second order methods would require 4, 5 times more points in each direction than spectral methods. It has been also proven that high order methods are beneficial for the long term integration necessary to attain the asymptotic turbulent regime.

During the last several years we have been working on local low communication parallel approaches for the solution of non-linear time dependent problems on coarse-grain multiprocessors [1–3, 7–16, 29, 30, 37–40]. We developed and implemented various Navier-Stokes solvers in two and three dimensions [4–6].

Our experiments were performed on a structured grids. Therefore, the geometric domain decomposition (DD) is intrinsically suited for the purposes of parallelization since neighboring subdomains can be allocated to neighboring processors with immediate communication links. The original problem is solved independently in each processor; then a matching procedure is used to enforce the continuity conditions on the interfaces. The objective of any DD method is to reduce the amount of interprocessor communications and data exchange as much as possible to avoid communication and synchronization bottlenecks that reduce the achieved speedup. Currently we are working on modifying these algorithms that can be solved on an unstructured grid [17]. This work, [17], showed how to parallelize a serial industrial solver for Navier-Stokes equations. The can handle a full compressible Navier-Stokes equations and it is applicable to realistic large size aerodynamic configurations. This is due to its ability to resolve the flow properly on relatively coarse grids. It possesses a complex infrastructure which ensures its flexibility and support to basic applicability to complex configurations.

For the long-term integration of PDEs and the resolution of the fine features in subdomains, it is preferable to use high-order methods, in particular spectral methods. However, the *global*[1] nature of spectral methods makes these difficult to parallelize while achieving good performance. In particular, this *global* transfer of data may destroy or degrade the possibility of achieving scalability in the above sense. Numerical parallel algorithms that are based on spectral methods require global data transfers (such as transpose in the application of fast Fourier transform (FFT)) and global communication. For a massively parallel algorithm this may cause communication and synchronization bottlenecks.

Low-communication parallel algorithms based on spectral methods have been developed for the solution of nonlinear time-dependent PDEs [1, 2, 29, 30, 37]. The parallelization is achieved by domain decomposition. Elemental solutions in subdomains are constructed using the local Fourier basis (LFB) method. In this method a global problem is decomposed into smooth local subproblems using a collection of overlapping bell functions. Then, the Fourier method is applied in each local interval with spectral accuracy (without exhibiting the Gibbs phenomenon). It has been shown in [1, 2, 29, 30, 37] that the multidomain local Fourier (MDLF) method is especially efficient for large problems using high resolutions in space. In this case the relative number of operations required to perform the projection is insignificant in comparison with what is needed to execute the FFT. Moreover, this extra work is reduced as the size of the problem grows in each domain [4]. Therefore, for the computation of large problems, which requires many degrees of freedom, the MDLF method is almost as efficient as the conventional Fourier method. The important advantage of the MDLF method, when compared to other multidomain spectral techniques, is that it enables a great simplification of the matching relations for the interface unknowns. It enables each harmonic to be matched separately, thus eliminating the global coupling of the interface unknowns. For problems with constant coefficients and in simple domains, the use of the Fourier basis permits the matching of each harmonic to be done separately, thus eliminating the global coupling of the interface unknowns. Since then, the method has been extended to handle complex geometries [38, 40]. The decomposition strategy of the MDLF method can easily be changed to reflect different resolutions in each subdomain or to treat an unstructured grids, which makes it valuable as an adaptive algorithm.

In previous work [4] the MDLF method was applied to the parallel solution of the model Navier-Stokes problem investigated previously by R. Temam [36]. In [4] the model that was used for the simulation did not contain the pressure term; therefore, the time discretization procedure results in the Helmholtz equation that had to be solved in each time step. It was shown [4, 11, 29, 30, 37] that certain elliptic operators have fast decaying Green's functions, which can be used to localize the Helmholtz

[1]Here *global* means that there is a transfer of data among many processors that are not adjacent to each other.

operator computations, and that the application of the MDLF method to such problems is therefore highly efficient. Matching of the solutions in subdomains requires only local communications between neighboring processors. Hence, the algorithm is fully scalable. These algorithms were implemented in [4–6].

In this paper we survey the algorithms and implementation of the parallel solution of the 2-D and 3-D Navier-Stokes equations with pressure. The pressure term in the Navier-Stokes equations adds the necessity for global data to be transferred in each time step to the parallel solution due to the need to solve the Poisson equation in each time step. The treatment and the solution of the global Poisson equation for pressure may degrade the performance and speedup of the parallel solution. We show that for an elliptic equations of the Poisson type the size of the required global data transfer can essentially be reduced. The reason, as will be demonstrated later, is that only the lowest harmonics of the pressure are treated and matched globally. Therefore, most of the communication that is required for parallelization of the Navier-Stokes equations is mainly local, between adjacent subdomains. The 2-D and 3-D Navier-Stokes solvers are implemented on three different MIMD message-passing multiprocessors: a 60-processor IBM SP2, a 20-processors MOSIX [18, 19], and a network of 10 Alpha workstations using the PVM (parallel virtual machine) software package [27]. The same program without any modification was ran on all these computational platforms, which demonstrates the portability of the algorithms and its software.

This paper is a further development of the local Fourier methods for the solution of PDE's in multidomain regions. The paper suggests a new algorithm for the parallel solution of two-dimensional equation of the Navier-Stokes type [36] with constant and non-constant coefficients which is mapped onto cell topology.

The extension of the above solution to problems with non-constant coefficients is suggested via spectral multidomain preconditioner. This approach is efficient when we have good local approximations in each subdomain (small variations in the coefficients from their mean values). By dividing the computational domain into a large enough number of subdomains we can guarantee it.

Prior to this algorithm, when we decomposed into strips [4, 5], we were able to "kill" large variances in the non-constant coefficients only in the direction across the strips while along the strips the variance is still out of control.

The new achievement here is that we are able to handle decomposition of the domain into cells that is the decomposed in both spatial directions and thus we are able to have small variances on each "patch" of the domain.

An appropriate Alternate Direction Implicit (ADI) scheme was developed. It enables the reduction of a 2-D problem to a collection of uncoupled 1-D ODE's. In effect, the 1-D solver becomes the basic routine to solve a 2-D problem using splitting of differential operators by ADI.

Detailed performance analysis is given where the issue of the communication among the domains is examined. This is of particular importance when using iterative methods. We show that by using the Richardson method only local communication is required. On the other hand, this method converges slower than Conjugate Gradient (CG) type methods using "global" computation (scalar products). Therefore, there is a tradeoff between these approaches: "local" (Richardson method-slow convergence) and "global" (CG method-fast convergence).

The algorithm was implemented on IBM SP/2, network of Alpha workstations, and MOSIX which is a network of i586. All are implemented using the PVM software package and the same ADI program was running on these different multiprocessor configurations. It achieved efficiency of 55-70% depending on the multiprocessor.

The paper has the following structure: In Section 2 we present the governing equations and discuss their discretization procedures in space and time and the solution of the Helmholtz equation. In Section 2.4 we discuss how the global solution of the Poisson equation becomes "local". In Section 3 we describe the computational steps of the algorithm. Details of the implementation are presented in Section 4. Performance and scalability analyses are given in section 5. How this method is used to simulate turbulence is given in Section 5.4.

A new parallel solver for the solution of Navier-Equations with constant and non-constant coefficients is given in section 6 and its application to 2-D cell topology (Section 6.1). The alternate direction implicit (ADI) method as another time-discretization scheme is discussed in Section 6.2. Parallel algorithm for the cell computation with constant coefficients is given in Section 6.2.2. The solution with non-constant coefficients is described in Section 6.2.3 which is based on spectral preconditioner (Section 6.2.3). In Section 6.3 we describe the parallel algorithm for the cell computation with non-constant coefficients. The model problem is described in Section 6.4. In Section 6.5 we describe the performance analysis of the model problem implemented on SP2, MOSIX and network of ALPHA workstation.

2. Governing equations and numerical schemes. We are interested in the parallel solution of the Navier-Stokes equations, which governs the incompressible viscous flows with constant properties:

$$\frac{\partial \mathbf{v}}{\partial t} = Re^{-1}\nabla^2 \mathbf{v} - \mathbf{N}(\mathbf{v}) - \nabla\Pi + \mathbf{F}, \tag{2.1}$$

$\nabla\Pi = \nabla\Pi' + q$, $\Gamma = (\Pi', \mathbf{v})$, $q = \text{const}$, $\Gamma(x+2\pi, y+2\pi, z+2\pi) = \Gamma(x, y, z)$, in the periodic domain $\Omega = [0, 2\pi]^d$, $d = 2, 3$. Here $\mathbf{v}(\mathbf{x}, t) = (u, v, w)$ is the velocity subject to the incompressibility constraint

$$\nabla \cdot \mathbf{v} = 0 \qquad \text{in} \quad \Omega, \tag{2.2}$$

$\Pi = p + \frac{1}{2}|\mathbf{v}|^2$ is the total pressure (p is the hydrostatic pressure), $\mathbf{F}$ is the external forcing, and Re is the Reynolds number. The nonlinear term is written in rotational form:

$$\mathbf{N}(\mathbf{v}) = \mathbf{v} \times (\nabla \times \mathbf{v}). \tag{2.3}$$

The numerical solution of the problem Eqs. (2.1)–(2.3) with periodic boundary conditions requires discretization in both time and space.

2.1. Discretization in time. The discretization in time is performed using a splitting algorithm investigated in [33]:

$$\frac{\hat{\mathbf{v}} - \sum_{q=0}^{J_i-1} \alpha_q \mathbf{v}^{n-q}}{\Delta t} = \sum_{q=0}^{J_q-1} \beta_q (\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}), \tag{2.4}$$

$$\frac{\hat{\hat{\mathbf{v}}} - \hat{\mathbf{v}}}{\Delta t} = -\nabla \Pi^{n+1}, \tag{2.5}$$

$$\frac{\gamma_0 \mathbf{v}^{n+1} - \hat{\hat{\mathbf{v}}}}{\Delta t} = Re^{-1} \nabla^2 \mathbf{v}^{n+1}. \tag{2.6}$$

From Eqs. (2.2) and (2.6) it follows that $\nabla \hat{\hat{\mathbf{v}}} = 0$. Therefore, Eq. (2.5) can be rewritten in the form

$$\nabla^2 \Pi^{n+1} = \frac{1}{\Delta t} \nabla \cdot \hat{\mathbf{v}}. \tag{2.7}$$

The splitting procedures Eqs. (2.4), (2.6) and (2.7) can be implemented with a different order of accuracy, which depends on the numbers J_i and J_q of the time levels. The results presented in this paper are obtained using a third-order scheme with $J_i = J_q = 3$ and the coefficients $\gamma_0 = 11/6, \alpha_0 = 3, \alpha_1 = -3/2, \alpha_2 = 1/3, \beta_0 = 3, \beta_1 = -3, \beta_2 = 1$.

The splitting algorithm consists of an explicit advection step (2.4), a global pressure adjustment for incompressibility (2.5), (2.7), and an implicit viscous step (2.6). Semi-implicit schemes of this type have much less severe restrictions on the time step, $\Delta t = O(\Delta x)$ than do explicit schemes, where $\Delta t = O(\Delta x^2)$. However, parallelization of such schemes is considerably more difficult as they result in *global* elliptic equations of two types: the (monotonic) Helmholtz Equation (2.6) which can be rewritten in the form

$$\nabla^2 u - \tilde{\lambda}^2 u = f(\mathbf{x}), \tilde{\lambda}^2 = \gamma_0 Re/\Delta t, \tag{2.8}$$

and the Poisson Equation (2.7)

$$\nabla^2 u = g(\mathbf{x}). \tag{2.9}$$

Equations (2.8) and (2.9) are solved repeatedly in each time step. The discretization procedure and parallelization of these equations are discussed below.

2.2. Discretization in space: Domain decomposition. Our parallel algorithm is based on domain decomposition technique. In this paper we consider the computational domain $\Omega = [0, 2\pi]^d$, where $d = 2$ or 3 is decomposed into parallel 2-D strips or 3-D slabs (which are called *strips* in the rest of the paper), see Figure 1. The algorithm consists of two main steps. First, we construct the local solutions independently in each subdomain (strip) with some prescribed boundary conditions at the interior boundaries (interfaces). Second, we correct these local solutions to remove discontinuities on the interfaces. This matching step involves global coupling of the interface data due to the the global nature of the elliptic problems (2.8) and (2.9).

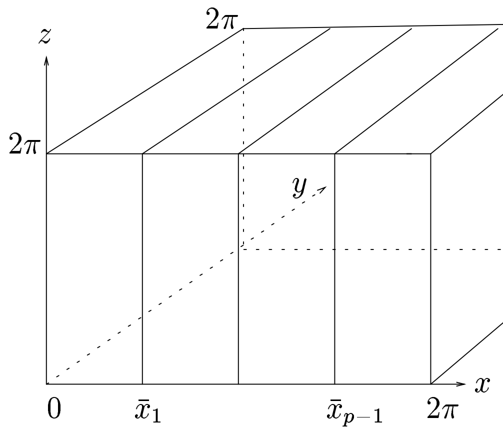

FIG. 1. *A 2-D domain ($z - x$ plane) and 3-D space ($x - y - z$) decomposed into parallel strips.*

In earlier works [1, 3, 4, 29, 30, 37] a parallel multidomain local Fourier (MDLF) method was developed. It was shown the parallel solution and implementation of the Helmholtz equation can be performed using only local communication between neighboring domains. In this paper, Section 2.4, we show that the parallel solution of the Poisson Equation (2.7) in a periodic domain is mostly composed of local problems since only the lowest harmonics of the pressure needed to be treated globally. Therefore, most of the communication that is needed for the parallelization of the Navier-Stokes Equation (2.1) is mainly local, between adjacent subdomains.

In the following sections we describe briefly the basic MDLF technique for the Helmholtz equation and, in more detail, for the Poisson equation.

2.3. Solution of the Helmholtz equation. In this Section we describe the algorithm as applied to the Helmholtz Equation (2.8). First, discretize the local problems on a uniform grid. After the application of the Fourier transform on the Helmholtz equation in the periodical directions y

and z (see Figure 1), we get the collection of uncoupled 1-D equations. In the nth strip we have

$$\frac{d^2\hat{u}^{(n)}}{dx^2} - \lambda^2 \hat{u} = \hat{f}^{(n)}(x). \tag{2.10}$$

where $\hat{u}^{(n)} = \hat{u}_k^{(n)}$, $\hat{f}^{(n)} = \hat{f}_k^{(n)}$, and $\lambda^2 = \tilde{\lambda}^2 + k^2$ in the 2-D case, and $\hat{u}^{(n)} = \hat{u}_{km}^{(n)}$, $\hat{f}^{(n)} = \hat{f}_{km}^{(n)}$, and $\lambda^2 = \tilde{\lambda}^2 + k^2 + m^2$ in the 3-D case. Here $k = -\frac{N_y}{2}, ..., \frac{N_y}{2}$ and $m = -\frac{N_z}{2}, ..., \frac{N_z}{2}$ are the wave numbers in the y and z directions. The MDLF algorithm for the solution of Equation (2.10) consists of two main steps:

1) Construction of the local particular solutions. Eq. (2.10) is solved independently in each subdomain for $x \in [\bar{x}_{n-1}, \bar{x}_n]$ with some prescribed boundary conditions on the interfaces. In order to obtain smooth local pieces $\hat{f}^{(n)}(x)$ of the source function, we perform extension and "folding" across the boundaries $\bar{x}_{n-1}$ and $\bar{x}_n$ (for additional details see [1–4, 6, 23, 29, 30, 37].

The extra pieces of the function $\hat{f}^{(n)}(x)$, required for smoothing, are provided by the overlapping of the neighboring subdomains over a small range 2ϵ. The data transfer between neighboring subdomains is demonstrated in Figure 2. The extended source function coincides with $\hat{f}^{(n)}(x)$ on the interval $x \in [\bar{x}_{n-1}, \bar{x}_n]$ and smoothly decays outwards over the distance 2ϵ from both sides of the interval.

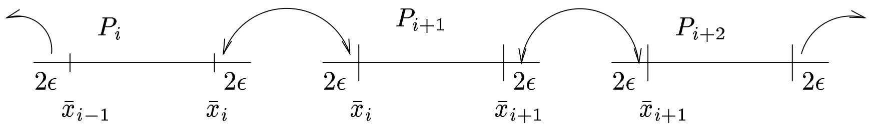

FIG. 2. *Data transfer between overlapped contiguous subdomains.*

The smooth source function in the right hand side can be represented by rapidly (in fact spectrally) converging trigonometric series. Thus, we integrate the local problems on extended intervals using the discrete cosine transform (DCT). The computational count for this step is $(n+N_\epsilon)\log(n+N_\epsilon)$, where n is the number of collocation points in a subdomain (across the strip) and N_ϵ is the number of points on the extended intervals ϵ. At the assumption that $N_\epsilon \ll n$, the "computational overhead" of the MDLF method, as compared to the conventional Fourier method, can be estimated as $(n + N_\epsilon)\log(n + N_\epsilon)/n \log n = O(N_\epsilon/n)$. This is the cost for having the spectral accuracy when applying the DCT transform to non periodic pieces of functions in the subdomains.

The values of the computed particular solutions $u_p^{(n)}$ have jumps on the interfaces and it is demonstrated in Figure 3.

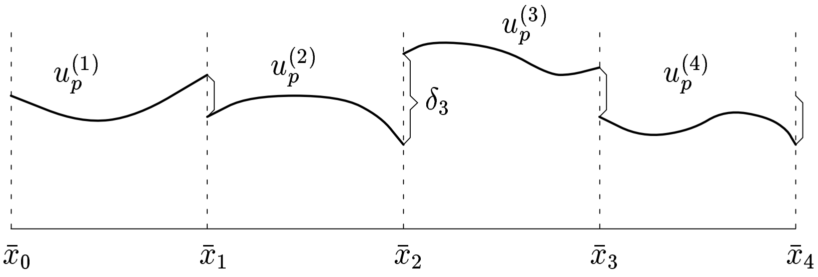

FIG. 3. *The jumps between the solution in each subdomain.*

This method preserves the spectral accuracy of the Fourier method and is very efficient for the evaluation of a particular solution in a subdomain.

2) Matching of the local solutions. In this step we correct the local solutions (independently for each harmonic k and m) in order to satisfy the continuity of the solution and its first derivative at the interfaces $x = \bar{x}_n$: $\hat{u}^{(n)} = \hat{u}^{(n+1)}$, $\frac{d}{dx}\hat{u}^{(n)} = \frac{d}{dx}\hat{u}^{(n+1)}$.
The matching procedure makes use of the properly weighted interface Green's functions which are the singular solutions of the homogeneous form of Eq. (2.10). For each n, this is a pair of exponential functions

$$h_+^{(n)} = e^{\lambda(x-x_n)}, \qquad h_-^{(n)} = e^{-\lambda(x-x_n)} \tag{2.11}$$

"attached" to the interface x_n and decaying away from $x = x_n$ on both side (see Figure 4). The smooth global solution is sought as a linear combination

$$\hat{u} = \bigcup_{n=1}^{P} \hat{u}^{(n)}, \qquad \hat{u}^{(n)} = \hat{u}_p^{(n)} + \sum_{m=1}^{P}\left(A_m h_+^{(m)} + B_m h_-^{(m)}\right). \tag{2.12}$$

where P is the number of subdomains (strips).
The amplitudes A_n and B_n are computed locally, in each subdomain, in terms of the jumps of $\hat{u}_p^{(n)}$, $d\hat{u}_p^{(n)}/dx$ at the interfaces $x = \bar{x}_n$,

$$A_n = -\alpha_{n-1}, \qquad B_n = -\beta_{n-1} \tag{2.13}$$

where

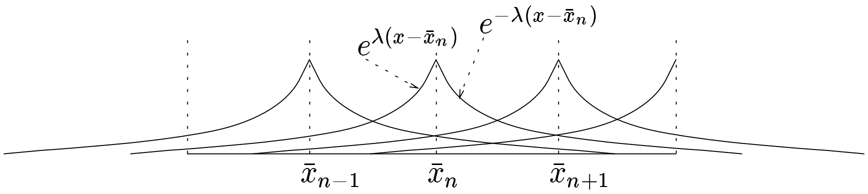

FIG. 4. *Decaying Green's functions on the interfaces.*

$$\alpha_n = \frac{1}{2}(\frac{\delta'_n}{\lambda} + \delta_n), \qquad \beta_n = \frac{1}{2}(\frac{\delta'_n}{\lambda} - \delta_n),$$

$$\delta_n = \hat{u}_p^{(n)}(x_{n+1}) - \hat{u}_p^{(n+1)}(x_{n+1}), \; \delta'_n = \frac{\hat{u}_p^{(n)}}{dx}(x_{n+1}) - \frac{\hat{u}_p^{(n+1)}}{dx}(x_{n+1}), \tag{2.14}$$

$\hat{u}_p^{(n)}$ and $\dfrac{\hat{u}_p^{(n+1)}}{dx}$ are the interface values of the particular solution and its derivative.

The above matching procedure is *exact*. Indeed, by matching the function and its first derivative at the interfaces, the higher derivatives will be matched automatically due to Eq. (2.10).

We observe that if $\lambda l \gg 1$, where $l = 2\pi/P$ is the size of a subdomain across the strip, then the exponential functions $h_\pm^{(n)} = e^{\pm\lambda|x-x_n|}$ decay rapidly away from the interface x_n so that they are well localized on the size of a subdomain. In effect, instead of (2.12) we arrive at the local matching relation

$$\hat{u}^{(n)} = \hat{u}_p^{(n)} + A_{n+1}h_+^{(n+1)} + B_n h_-^{(n)}. \tag{2.15}$$

The accuracy of the local matching is estimated as $O(e^{-\lambda l})$. The computational count of this step is $O(n)$.

In the case of the monotonic Helmholtz equation,

$$\lambda^2 = \tilde{\lambda}^2 + k^2 + m^2, \qquad \tilde{\lambda}^2 \propto Re\Delta t^{-1}. \tag{2.16}$$

the condition of locality $\lambda l \gg 1$ holds even for $k = m = 0$ provided that $Re/\Delta t$ is large enough. In such a case the local matching procedure (2.12) can be used, which requires only neighbor-to-neighbor communication.

To conclude, the performance of the MDLF method with the local matching of elemental solutions in subdomains is almost as good as that of the conventional Fourier method viewing accuracy, computational count and memory requirements. From the parallel performance point of view, this method is also advantageous as it requires only local interaction between neighboring subdomains. Thus, the corresponding parallel algorithm is scalable.

2.4. Solution of the Poisson equation. The solutions of the Poisson Equation (2.9) that describe equilibrium processes are determined, in principle, at each location by the combined effect of all non homogeneous forcing terms and boundary conditions. One may conclude, therefore, that these solutions must be obtained by global procedures requiring repeated global communications. The homogeneous solutions of Eq. (2.9) decay slowly as a function of the distance, so that the contribution of all remote interfaces seems to be necessary for the computation of the matching parameters in each location. Nevertheless, even in this "tough" case, a substantial reduction can be achieved in the magnitude of the interprocessor communication while preserving the desired accuracy.

The Fourier transform of Eq. (2.9) along the strips results in equation (2.10) where $\lambda^2 = k^2$ in the 2-D case ($\lambda^2 = k^2 + m^2$ in the 3-D case); k and m are wave numbers in the y and the z directions. The analysis below is related to the 2-D case and the 3-D case can be treated similarly.

For $\lambda \neq 0$, the homogeneous solutions of (2.10) decay exponentially as a function of x. The rate of decay becomes faster as k increases. Thus, a global matching procedure is required only for the long waves $\lambda \leq k_*$, whereas the short waves $\lambda \geq k^*$ can be treated by using local matching on the interfaces. For the intermediate wavenumbers $k_* < \lambda < k^*$ the number of connected subdomains (processors) can be controlled with the prescribed accuracy.

Thus, the solution of the Poisson Equation (2.9) can be obtained by the MDLF method using either global or local matching depending on the value of k.

The decomposition into "local" and "global" waves depends on the number of subdomains P, which determines the local length $l = 2\pi/P$, and thus affects the factor of locality $\mathcal{L} = e^{-kl}$. We now estimate the dependence of η on the number of processors P for a given accuracy ε. The processors are assumed to have a fixed memory size, so that the total number of collocation points in each subdomain $N_s = (N_x/P) \times N_y$ is independent of P. We define γ by $N_y/N_x = P^\gamma$. Then, the following relations hold:

$$N_s = \frac{N_x}{P} \cdot P^\gamma N_x = N_x^2 P^{\gamma-1},$$

$$N_x \propto P^{\frac{1-\gamma}{2}}, \quad N_y \propto P^{\frac{1+\gamma}{2}}.$$

In order to keep the locality parameter $\mathcal{L}$ constant (to ensure the prescribed accuracy ε), the cutoff wavenumber k^* must be proportional to P, so that

$$\eta \propto k^*/N_y \propto P/P^{\frac{1+\gamma}{2}} = P^{\frac{1-\gamma}{2}}.$$

For $\gamma = 0$ we have $N_x = N_y \propto \sqrt{P}$ and $\eta \propto \sqrt{P}$. Another extreme case, $\gamma = 1$, corresponds to a constant N_x and $N_y \propto P$ (the resolution is changed only in the direction of the strips, as in the plane $x-z$ of Figure 1.) In this case η is independent of P; that is, the algorithm is fully scalable.

3. Computational algorithm for the 2-D and 3-D Navier-Stokes parallel solver. In this section we present all the computational steps of the algorithm that was used to solve the 2-D and 3-D Navier-Stokes equations with strip topology. Each subdomain (strip) is mapped into a different processor. Initially, we define the velocity $\mathbf{U}_d(\mathbf{x},t)$ and the pressure $P_d(\mathbf{x})$ distributions in space (see Section 5.1). Then we compute the "forcing" $\mathbf{F}$ using Eq. (2.1). By using a time marching algorithm (Eqs. (2.4)–(2.6)) we compute the velocity evolution $\mathbf{v}(\mathbf{x},t)$ under the "forcing" $\mathbf{F}$ where the initial condition is $\mathbf{U}_d(\mathbf{x},t)$. It is clear that in this case the deviation of the computed solution $\mathbf{v}(\mathbf{x},t)$ from $\mathbf{U}_d(\mathbf{x},t)$ in each time step has to be small and depends on the time-stepping increment Δt and the number of collocation points. (For example, in our 3-D computations the maximum relative error is $\epsilon \sim 10^{-10}$ for double precision at $\Delta t = 10^{-3}$ where the number of collocation points $n = 480, N_y = 32$, and $N_z = 32$. n is the number of points in a subdomain across the strip where N_x and N_z are number of points in the y and z periodic direction (along the strips), respectively; see Section 5.)

As mentioned previously, all harmonics from the solution of Eqs. (2.4)–(2.6) in the y and z directions using strip topology are uncoupled. From now on, all the algorithmic work in the y and z directions will take place using specified harmonics except for the computation of the nonlinear terms $\mathbf{N}(\mathbf{v})$, which will be discussed later.

Our algorithm consists of eight main steps. After each step 2 to 6 we exchange data between the processors. However, only two data exchanges, before and after step 5, involve global data transfer (all processors transfer the data to one processor). All other data exchanges (communication) are local, that is they take place only between neighboring subdomains. In step 5, we match the particular solutions for the pressure Π and its derivative $\partial\Pi/\partial x$ in the whole domain. According to the previous analysis, only the lowest few harmonics have to be matched globally whereas higher harmonics are matched locally.

The time marching is performed in the Fourier domain independently for each harmonic k. The transformation to the physical domain is required only to compute the nonlinear term.

Step 1: Initialization

- Generation of the bell function for the RHS folding of Eqs. (2.4)–(2.6) [4, 23, 29, 30, 37]. It is needed for the construction of the elemental particular solutions of the Helmholtz equation in each subdomain (processor).
- Computation of the spatial part of the "exact" solution $\mathbf{U}_d(\mathbf{x},t)$ in each subdomain (for details see Section 5.1).

- Calculation of the spatial parts of the forcing $\mathbf{F}$ components using Eq. (2.1).
- Computation of the initial velocity fields $\mathbf{v}^n$ for $n = 0, -1, -2$ (correspondingly at $t = 0, -\Delta t, -2\Delta t$) as required by the $3d$ order time discretization scheme (2.4).
- Transform $\mathbf{v}^n$ into the Fourier space by applying the FFT in the periodic directions.
- Computation of the nonlinear term $\mathbf{N}(\mathbf{v}^n)$, $n = 0, -1, -2$ according to Eq. (2.3) using the pseudo-spectral technique (compute derivatives in the Fourier domain and multiply functions in the physical domain). This is performed in the following way: apply the inverse Fourier transform on $\nabla \times \mathbf{v}$ and $\mathbf{v}$. In physical space we compute $\mathbf{N}(\mathbf{v}) = \mathbf{v} \times (\nabla \times \mathbf{v})$ and return to Fourier space.

Step 2: Begin the time marching: Explicit advection step

- Computation of the velocity $\hat{\mathbf{v}}$ according to Eq. (2.4).

Step 3: Compute the RHS of the Poisson equation

Computation of the function $\dfrac{1}{\Delta t}\nabla \cdot \hat{\mathbf{v}}$ in the RHS of (2.7). In order to compute the derivative $\partial \hat{u}/\partial x$ across the strips we have to extend and "fold" the velocity $\hat{u}$. Extension is provided by the overlapping of adjacent subdomains. Thus, the data exchange is taking place between neighboring processors (see Fig. 2).

- Computation of the derivatives $\partial \hat{u}/\partial x$, $\partial \hat{v}/\partial y$ and $\partial \hat{w}/\partial z$. To calculate $\partial \hat{u}/\partial x$ we extend and "fold" the velocity component $\hat{u}$ for the application of the Fourier transform in the x direction.
- Calculation of the RHS of the Poisson equation.

Step 4: Solution of the Poisson equation

We solve the Poisson Equation (2.7) and compute the pressure Π^{n+1} and its derivatives for the calculations of the RHS in Eq. (2.5). To accomplish this we have to exchange the overlapped areas of the RHS of Eq. (2.7) in the neighboring subdomains.

- Extension of the RHS function in each subdomain. In order to do this, exchange the data from the overlapped (neighboring) subdomains.
- Computation of the particular solution Π_p^{n+1} of the Poisson equation by the MDLF method: fold the RHS, apply the Discrete Cosine Transform (DCT) on the extended interval and integrate in the Fourier domain).
- Computation of the first derivative $\partial \Pi_p^{n+1}/\partial x$ by the MDLF method.

Step 5: Matching of $\mathbf{\Pi}_p^{n+1}$ and $\partial \mathbf{\Pi}_p^{n+1}/\partial x$ at the interfaces (local-global matching)

This describes the partial global matching for the particular solutions of the Poisson equation. Globally matched harmonics are processed in one processor after all the data have been channeled to it. For locally matched harmonics we need data exchange between neighboring subdomains.

- Exchange the boundary values of Π_p^{n+1} and $\partial\Pi_p^{n+1}/\partial x$ between adjacent subdomains (processors). We compute the coefficients A_n and B_n, using Eq. (2.13).
- *Global matching ($k < k_*$):* Exchange the computed values A_n, B_n between the processors so that each processor has the complete set of the coefficients. Correct the particular solutions using Green's functions from all the interfaces as in (2.12).
- *Local matching ($k > k_*$):* Correct the particular solutions using only local Green's functions as in (2.15).

Step 6: Computation of the velocity $\hat{\hat{\mathbf{v}}}$

- Computation of the functions Π^{n+1} and $\partial\Pi^{n+1}/\partial x$ in every subdomain, using Eq. (2.12).
- Computation of derivatives of Π^{n+1} in the periodic directions, $\partial\Pi^{n+1}/\partial y$ and $\partial\Pi^{n+1}/\partial z$.
- Evaluation $\hat{\hat{\mathbf{v}}}$ using Eq. (2.5)
- Evaluation the RHS of Eq. (2.6).

Step 7: Implicit viscous step

Solution of the Helmholtz Equation (2.6) for $\mathbf{v}_p^{n+1}$ using the MDLF method. This involves the following steps:

- Computation of the source term in the RHS, $Re\hat{\hat{\mathbf{v}}}$
- Extension the RHS by exchanging data between the neighboring subdomains (processors).
- Fold the RHS function.
- Application of the DCT (across the strips) on the extended intervals and integrate in the Fourier domain.

Step 8: Matching step of the particular solutions $\mathbf{v}_p^{n+1}$: Completion of the time step

To calculate the velocity $\mathbf{v}^{n+1}$ we match the particular solutions $\mathbf{v}_p^{n+1}$ and their derivatives $\partial\mathbf{v}_p^{n+1}/\partial x$. This requires the exchange of boundary values of these quantities between neighboring subdomains.

- Match the particular solutions $\mathbf{v}_p^{n+1}$ using local matching procedure as in step 5.
- Computation of the derivatives $\partial\mathbf{v}_p^{n+1}/\partial x$ across the strips.

This completes one marching step. To continue marching in time, go to step 2.

The accuracy and stability of the above MDLF method were tested in our previous papers. In [5] this method was applied for long time numerical simulation of turbulent flows.

4. Implementation. The proposed algorithms are appropriate for scalable distributed-memory MIMD multiprocessors.

To implement the above algorithms we used the PVM [27] software package, which currently is one of the most popular software packages for parallel processing. Under PVM the user defines a collection of serial, parallel, and vector computers that appears as one large distributed-memory computer. We will refer to this computer as a *virtual machine* and the term *host* will be used to designate one of the member computers. The virtual machine is scalable so that it can grow with the size of a problem. PVM supplies the functions to automatically start up tasks on the virtual machine and allows tasks to communicate and synchronize with each other. A *task* is defined as a unit of computation in PVM that is analogous to a UNIX process.

Effective implementation of the code on a distributed-memory parallel computer requires the development of an efficient concurrent algorithm, which should minimize communication costs and maintain a proper load balance. The parallel implementation strategy adopted here is based upon domain decomposition (DD) method. According to this method each domain is assigned to a specific processor.

During the computations there are several local data exchanges between neighboring processors, according to the parabolic domain decomposition (PDD) methodology and two partial global data exchanges for the matching of particular solutions for the Poisson equation. The 2-D and 3-D Navier-Stokes solvers are implemented on three different MIMD message-passing multiprocessors: a 60-processor IBM SP2, a 20-processor MOSIX [18, 19], and a network of 10 Alpha workstations (Alpha farm) using PVM software package. The same program without any modifications using PVM was running on all multiprocessors.

5. Performance analysis. We now present the performance analysis of our algorithm for the solution of the 2-D and 3-D Navier-Stokes equations (Eq. (2.1)). The algorithm was tested according to the methodology described in Section 3. We identify the evolution of the solution under an external force that has complicated behavior in space and time. This algorithm can be easily adapted to the solution of other real problems of turbulence, such as turbulence decay [21] and the long-term evolution of unstable exact solutions of Navier-Stokes Equation ([20]). Only the "forcing" term has to be modified for this application.

5.1. The 2-D and 3-D Navier-Stokes model problems. To simulate the solution with "complex" behavior in space and time we take the model suggested by R. Temam [36] for the investigation of the efficiency of the multilevel nonlinear Galerkin method. The general methodology and the theoretical results in this presentation are valid for wide range of problems and they are independent of the specific example below.

Different initial distributions of velocity and pressure were used to test the 2-D and 3-D algorithms. The results presented in this section are obtained for highly nonlinear distributions, which enable us to simulate the complex behavior of flow in space and time.

The initial distributions of velocity and pressure for the simulations are given below.

For the 2-D Navier-Stokes equations we have

$$v_x = \frac{\partial \psi}{\partial y}, \qquad v_y = -\frac{\partial \psi}{\partial x},$$

$$\psi(x, y) = g(t) e^{(\cos kx)(\cos ky)}, \tag{5.1}$$

$$g(t) = \frac{1}{40} \left(\frac{1}{10} e^{(sin2t - 3sin2\pi t)} + cos(2\sqrt{2}t) \right) + \frac{0.375}{5},$$

$Re = 100, k = 5$. We take the external pressure distribution to be

$$\Pi_d = \cos(2kx) \cos ky. \tag{5.2}$$

Components of the "forcing" term $\mathbf{F}$ were derived from Eq. (2.1) using the distributions of Eqs. (5.1) and (5.2).

For the 3-D Navier-Stokes equations we have

$$\mathbf{v} = \nabla \times \mathbf{\Psi},$$

$$\Psi_x = g(t) \cdot e^{(\cos ky)(\cos kz)},$$

$$\Psi_y = g(t) \cdot e^{(\cos kx)(\cos kz)},$$

$$\Psi_z = g(t) \cdot e^{(\cos kx)(\cos ky)},$$

$$g(t) = \frac{1}{40} \left(\frac{1}{10} e^{(sin2t - 3sin2\pi t)} + cos(2\sqrt{2}t) \right) + \frac{0,375}{5}, \tag{5.3}$$

$Re = 100, k = 5$. The pressure distribution is given by

$$\Pi_d = \cos(2kx) \cos ky \cos kz. \tag{5.4}$$

By using Eqs. (5.3) and (5.4) we compute the components of the forcing term $\mathbf{F}$ in Eq. (2.1).

Figure 5 demonstrates the solution (5.1) of the 2-D Navier-Stokes equations generated by the appropriate forcing term. It describes a contour plot of the vorticity in each domain (processor) before matching across neighboring interfaces is performed. It corresponds to the following parameters: the number of processors is 4, the viscosity coefficient ν is 0.0001, the time step Δt is 0.001, current time $t = 0.2$, the resolution in each domain is 32×128, and the number of points in the overlapping region $N_\epsilon = 16$.

Figure 6 demonstrates the solution (5.1) of the 2-D Navier-Stokes after matching.

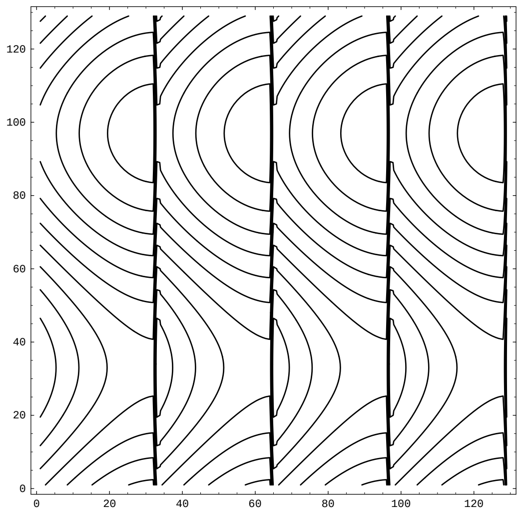

FIG. 5. *The solution of (5.1) before matching.*

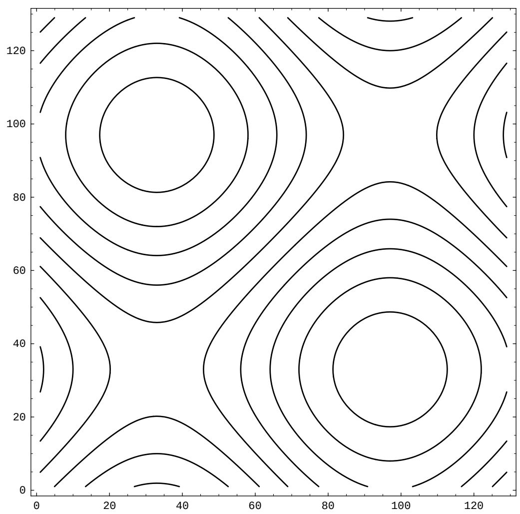

FIG. 6. *The solution of (5.1) after matching.*

5.2. Parallel performance – Speedup and efficiency. If we assume that we have p processors (which is also the number of strips), n is the number of points in a subdomain across the strip, and $N = np$ is the global number of points in one direction (across strips) then the achieved speedup and efficiency for the 2-D and 3-D cases is given by Tables 1 and 2, respectively.

TABLE 1
Two-dimensional problem: speedup and efficiency (in parentheses) of the 2-D parallel solver of the Navier-Stokes equations.

	$n = 488$			$n = 1000$
p	Alpha	MOSIX	SP2	SP2
3	1.5 (0.50)	1.7 (0.57)	2.1 (0.70)	2.2 (0.77)
6	4.1 (0.68)	4.2 (0.70)	4.7 (0.79)	5.0 (0.85)
10	6.8 (0.68)	7.3 (0.73)	8.3 (0.83)	9.2 (0.95)
20	—	16.2 (0.81)	19.1 (0.95)	20.1 (1.00)
30	—	—	32.2 (1.07)	33.5 (1.11)
40	—	—	43.8 (1.09)	46.1 (1.15)
50	—	—	57.8 (1.15)	60.2 (1.20)
60	—	—	69.0 (1.18)	73.8 (1.23)

TABLE 2
Three-dimensional problem: speedup and efficiency (in parentheses) of the 3-D parallel solver for the Navier-Stokes equations.

	$n = 224$			$n = 480$
p	Alpha	MOSIX	SP2	SP2
3	2.01 (0.67)	1.71 (0.57)	1.81 (0.60)	1.87 (0.63)
6	4.02 (0.69)	3.85 (0.64)	3.91 (0.65)	5.21 (0.79)
10	7.04 (0.70)	6.80 (0.68)	7.14 (0.71)	8.31 (0.83)
20	—	14.03 (0.70)	18.09 (0.90)	18.80 (0.94)
30	—	—	26.75 (0.89)	27.63 (0.92)
40	—	—	41.23 (1.03)	43.62 (1.09)
50	—	—	52.51 (1.05)	55.53 (1.11)
60	—	—	65.04 (1.08)	68.39 (1.14)

From Tables 1 and 2 we can see that for $p > 20$ the efficiency stays almost constant when we increase the size of the problem and the number of processors which indicates that the algorithm is highly scalable.

Denote by N the size of the FFT, T_s the serial time (for a single processor) and T_p is the parallel time for p processors. We note that for $p > 20$ the efficiency $E > 1$. The reason is the following. The main routine in the present serial algorithm, which is based on the spectral Fourier method, is the DFFT. The computation count (or the performance time) for the

FFT is a nonlinear function of the resolution N, $T_{DFFT} \propto N \log N$. In this case the efficiency can be estimated roughly as follows:

$$E = \frac{T_s}{pT_p} \sim \frac{N \log N}{p\left(\dfrac{N}{p}\log\dfrac{N}{p} + \tau_{com}\right)} \sim \frac{\log N}{\log N - \log\, p + \dfrac{p\tau_{com}}{N}},$$

where τ_{com} is the total communication time. Evidently, for large enough N the last expression becomes larger than one. Thus, for a large enough resolution the gain in the computation count due to domain decomposition offsets the communication overheads and the loss due to additional operations inherent in the MDLF method (such as overlapping subdomains and smoothing operations).

It follows that for large resolutions it pays to use the domain decomposition strategy even for the serial computation in a single processor. If the MDLF method is used as a serial algorithm, the efficiency is always less than one:

$$E \sim \frac{N \log \dfrac{N}{p}}{p\left(\dfrac{N}{p}\log\dfrac{N}{p} + \tau_{com}\right)} \sim \frac{\log\dfrac{N}{p}}{\log\dfrac{N}{p} + \tau_{com}} < 1.$$

5.3. Additional tests on the SP2. The algorithm that was implemented with PVM was transferred to a team of researchers at IBM Science and Technology, Haifa, Israel, who are working in the area of debugging and performance monitoring for distributed parallel systems. The team's goals were to study and evaluate the tools for debugging and performance monitoring on the SP2 and to demonstrate and explore the capabilities and limitations of such tools on the Navier-Stokes 2-D and 3-D parallel solvers. The work reported in [42] focuses on the IBM SP2 platform, but the results are applicable to other distributed parallel systems as well. The algorithm was used as a benchmark to compare MPI, the IBM version of PVM, and PVM and the performance of each communication library.

The application was converted to MPI and evaluated [42]. The group used our algorithm as a "black box" and applied their monitoring and debugging tools to it and found no bottleneck in the algorithm or in the way it communicated with its different parts during the processing. The team concluded that MPI is more efficient than PVM and they obtained better performance with it.

5.4. A parallel spectral Fourier – Nonlinear Galerkin algorithm for simulation of turbulence. In [6] the Multidomain Local Fourier (MDLF) techniques that were described in Sections 2.1–2.3, and 3 was combined with the Nonlinear Galerkin (NLG) algorithm [24–26, 31, 35] to solve incompressible Navier-Stokes equations. Results were presented on direct numerical simulation of two-dimensional homogeneous turbulence using the NLG method. We derived a high-order parallel algorithm which requires only the minimum inter-processor communication dictated by the physical nature of the problem at hand. The parallelization is achieved by domain decomposition. The discretization in space is performed using the Local Fourier Basis method. The continuity conditions on the interfaces are enforced by adding homogeneous solutions. Such solutions often have fast decay properties which can be utilized to minimize interprocessor communication. In effect, the predominant part of the computation is performed independently in the subdomains (processors) or using only local communication.

A novel element this parallel algorithm is the incorporation of a Nonlinear Galerkin strategy to accelerate the computation and stabilize the time integration process. The basic idea of this approach consists of decomposition of the variables into large scale and small scale components with different treatment of these large and small scales.

6. 2-D parallel solver for the solution of Navier-Stokes equations with constant and variable coefficients using ADI on cells. In this section we present a new algorithm for the parallel solution of two-dimensional Navier-Stokes type equation with constant and non-constant coefficients which is mapped onto cell topology. This is a further development in the application of the local Fourier methods to the solutions of PDE's in multidomain regions as was presented in the above sections.

The extension of the above solution to problems with non-constant coefficients is suggested via spectral multidomain preconditioner. This approach is efficient when we have good local approximations in each subdomain (small variations in the coefficients from their mean values, i.e. variance $= <a> - a(x,y)$ is small enough). By dividing the computational domain into a large enough number of subdomains we can guarantee it.

Prior to this algorithm, when we decomposed into strips [4, 5], we were able to "kill" large variances in the non-constant coefficients only in the direction across the strips while along the strips the variance was still out of control.

The new achievement here is that we are able to handle decomposition of the domain into cells that is the decomposed in both directions, x and y, and thus we are able to have small variances on each "patch" of the domain.

In addition, an appropriate Alternate Direction Implicit (ADI) scheme was developed. It enables the reduction of a 2-D problem to a collection of uncoupled 1-D ODE's. In effect, the 1-D solver becomes the basic routine to solve a 2-D problem using splitting of differential operators by ADI. The ADI is only of second order accuracy when there exists at least one periodic direction, and it may be even of lower order when there is no such direction.

In addition, it has a perceivable splitting error. But for our purpose it is indeed an appropriate method to utilize.

We show that by using the Richardson method only local communication is required. On the other hand, this method converges slower than Conjugate Gradient (CG) type methods using "global" computation (scalar products). Therefore, there is a tradeoff between these approaches: "local" (Richardson method, slow convergence) and "global" (CG method, fast convergence).

The Multidomain Local Fourier Method (MDLF) approach that was described in Sections 2.1–2.3, and 3 is based on the local Fourier methods [23], overcomes most of the global coupling, inherent both in the use of a spectral method in space and in the implicit discretization of time in parallel processing.

As described above and in Section 6.1 we use two approaches in order to preserve spectral accuracy: a modified Local Fourier Basis (LFB) method [23], implemented in the interior subdomains where a smooth decomposition of the source function is performed by using a system of overlapping bell functions, and the Fourier-Gegenbauer (FG) method [39] which makes use of the re-expansion of the (local) Fourier partial sums into rapidly convergent Gegenbauer series. The FG technique is implemented in the subdomains adjacent to the boundaries where overlapping of two contiguous subdomains is not possible in the case of non-periodicity (e.g. for Dirichlet or Neumann boundary conditions). The FG method is a good choice for our purpose because it operates inside the interval where the function is defined, so that it does not require overlapping of subdomains. Also, the FG method has uniform resolution like the LFB method, so that both Fourier techniques are easily combined. Thus, the use of a non-overlapping FG method enables us to treat non-periodic problems.

This paper is a further development of the local Fourier methods for the solution of PDE's in multidomain regions. The paper suggests a new algorithm for the parallel solution of two-dimensional equation of the Navier-Stokes type [36] with constant and non-constant which is mapped onto cell topology.

The extension of the above solution to problems with non-constant coefficients is suggested via spectral multidomain preconditioner. This approach is efficient when we have good local approximations in each subdomain (small variations in the coefficients from their mean values). By dividing the computational domain into a large enough number of subdomains we can guarantee it. This is described in Section 6.2.3.

Detailed performance analysis is given where the issue of the communication among the domains (processors) is examined. This is of particular importance when using iterative methods. We show that by using the Richardson method only local communication is required. On the other hand, this method converges slower than Conjugate Gradient (CG) type methods using "global" computation (scalar products). Therefore, there is a tradeoff between these approaches: "local" (Richardson method-slow convergence) and "global" (CG method-fast convergence).

The algorithm was implemented on IBM SP/2, network of Alpha workstations, and MOSIX which is a network of i586. All are implemented using the PVM software package and the same ADI program was running on these different multiprocessor configurations. It achieved efficiency of 55–70% depending on the multiprocessor.

The proposed algorithm has the potential for treating efficiently complex geometries. This is a work in progress.

6.1. Application of the MDLF method to cell topology. We solve the two-dimensional Helmholtz equation

$$\nabla^2 u - \lambda^2 u = f(x, y). \tag{6.1}$$

Suppose that the source function $f(x, y)$ is periodic in the y direction and we apply the FFT along the strips. Then, the problem is reduced to a set of one-dimensional problems for the Fourier coefficients $\hat{u}_k^{(n)}(x)$:

$$\hat{u}_k^{(n)''} - (k^2 + \lambda^2)\hat{u}_k^{(n)} = \hat{f}_k^{(n)}(x) \quad \text{in } \omega_n, \quad k = -\frac{N_y}{2}, ..., \frac{N_y}{2} \tag{6.2}$$

(N_y is the number of collocation points in the y direction), which can be solved by using the algorithm with the local matching on the interfaces (as described in Sections 2.1–2.3, and 3).

If the function $f(x, y)$ is not periodic along the strips, as it was assumed before, it can be transformed to a periodic form by using the same "folding" procedure. Then, the previous situation is reproduced on an extended region $\bar{\Omega} = (0, L_x) \times (-\epsilon, L_y + \epsilon)$. The solution of this modified problem will coincide with the original one on Ω.

Assume that our problem is mapped into a domain which is decomposed into cells. We showed in [37] that the implementation of the LFB technique enables us to reduce the overall coupling of the interface points to a sequence of simple, one dimensional local matching relations. The idea is to apply this algorithm in the x and y directions separately.

In the first step we take a horizontal row of cells and transform the source function using the LFB in the y direction. (The procedure includes an ϵ-extension and smoothing across the horizontal interfaces, antisymmetrical reflection and application of FFT on the local period).

In the second step we solve the system of one-dimensional problems of type (6.2) for the Fourier coefficients with the local matching on the vertical interfaces and then transform back into the physical space. The resulting particular solution $u^{(I)}$ will be smooth in each horizontal band except for the horizontal interfaces.

To remove discontinuities from the horizontal interfaces we transform the function $u^{(I)}(x, y)$ using the LFB in the x direction in each vertical column and match Fourier coefficients on these lines. Then, after the application of the inverse transform into the physical space, we get a vertically smooth solution $u^{(II)}$.

Smoothing in the y direction introduces discontinuities on the vertical interfaces because each Fourier coefficient is matched independently. These jumps can be removed by another matching (in the x direction). Then, this is done in each new cycle along x and y alternatively. This results in a smoother solution.

We solve the two-dimensional Helmholtz Equation (6.1) in the rectangular cells $\Omega = \bigcup_{n=1}^{P_x} \bigcup_{m=1}^{P_y} \omega_{mn}$ (Fig. 7(a)) of sizes $l_x \times l_y$; $l_x = L_x/P_x$, $l_y = L_y/P_y$ $\big($Fig. 7(a)$\big)$. P_x and P_y are the number of processors along the x and y axis respectively.

Instead of *simultaneous* matching at all collocation points we rather match *successively* in x and y directions.

1. We take horizontal row of cells $\Omega_\nu = \bigcup_{m=1}^{P_x} \omega_{\nu m}, \nu = 1, \ldots, P_y$ and transform the source function $f(x,y)$ into the LFB in the y direction (the procedure includes an ϵ-extension, smoothing across the boundaries $\bar{y} = (\nu - 1)l_y - \epsilon$ and $\bar{y} = \nu l_y + \epsilon$, antisymmetric reflection and FFT on the local period $\tau = 2(l_y + 2\epsilon)$- see Sections 2.1, 2.2 and Figs. 7 (c) and (d)).
2. We solve the system of one-dimensional problems of the type
$$\frac{d^2 \hat{U}_k^{(n)}}{dx^2} - (k^2 + \lambda^2)\hat{U}_k^{(n)} = \hat{f}_k^{(n)}(x) \quad \text{in} \ \ \omega_n, \quad k = -\frac{N_y}{2}, \ldots., \frac{N_y}{2}$$
3. (N_y is the number of collocation points in the y direction) for the Fourier coefficients with the local matching on the vertical interfaces, after which we transform back into the physical space.
4. The resulting particular solution $U_p(x,y)$ will be smooth in each horizontal band except for the interfaces $\bar{y}_n = nl_y,\ n = 1, \ldots, P_x$ (Figure 7(c,d)).
5. To remove discontinuities from the horizontal interfaces we use again our one-dimensional matching routine. We take the vertical column $\Omega^\mu = \bigcup_{n=1}^{P_y} \omega_{n\mu},\ \mu = 1, \ldots, P_x$, transform U_p into the LFB (by analogy with the source function on the first step) and implement the matching on the lines $\bar{y}_n$ for each Fourier coefficient. By inverting the transform into the physical space we get the vertically smooth solution ${U^I}_p$.
   Smoothing in the y direction induces discontinuities on the vertical lines $\bar{x}_n = nl_x,\ n = 1, \ldots, P_x$, because each Fourier coefficient is matched independently. These jumps are removed by another matching (in the x direction). Each new "ironing" along x and y alternatively results in a smoother solution.

The proposed algorithm is an extension (generalization) of the strip [4] computational algorithm since it can be done in the following way:

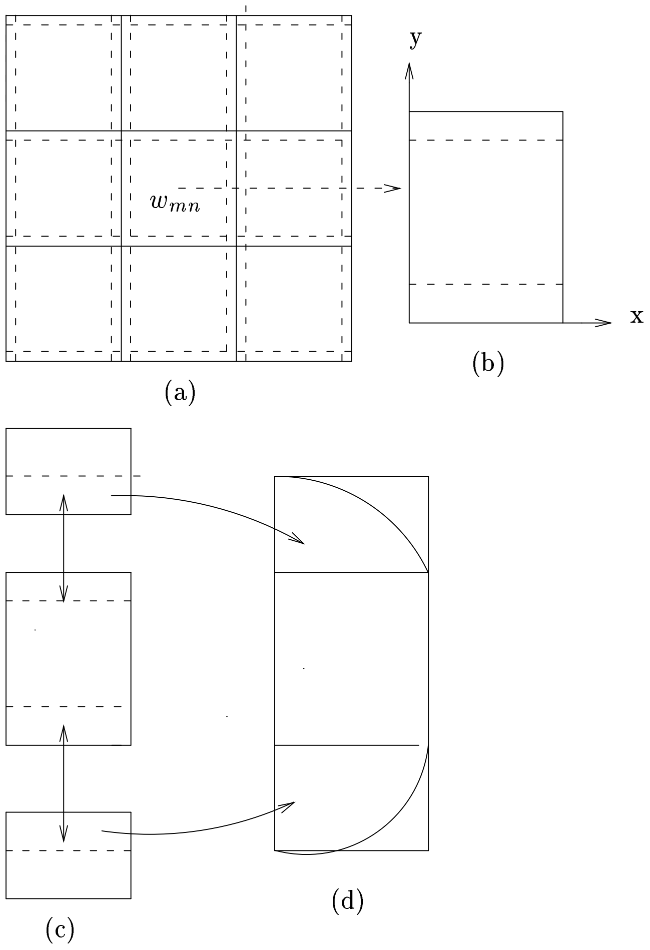

FIG. 7. *Implementation of the cell algorithm: (a) The domain is divided into $P_x \times P_y$ processors. (b) The overlapped area of each cell in one direction marked by the dotted lines. (c) and (d) show the extension and smoothing of the local solution.*

1. Do steps (a)–(d) as described in Fig. 7. In other words, as in the LFB, we take extension and perform smoothing of the local particular solution U_p.
2. As in the strip computation, we match in the x direction.
3. Extension, smoothing and FFT as explained in the strip computation.
4. As in the strip computation, we match in the y direction.
5. Go to step 2.

6.2. The ADI method. The alternate direction implicit (ADI) method is another time-discretization scheme. In this section we describe how the ADI method was implemented for the domain decomposition in the form of cells.

6.2.1. Discretization in time. An efficient time-discretization scheme can be constructed by using the alternate direction implicit (ADI) method. This is a two-step method which can be used as

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t/2} = \nu \left(\mathbf{v}^*_{xx} + \mathbf{v}^n_{yy}\right) - \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \tag{6.3}$$

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t/2} = \nu \left(\mathbf{v}^*_{xx} + \mathbf{v}^{n+1}_{yy}\right) - \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \tag{6.4}$$

When the 2-d order Adams-Bashford scheme is employed for the approximation of the non-linear and the non-homogeneous terms. The following parameters are used:
$J_e = 2,\ \beta_0 = 1.5,\ \beta_1 = 0.5.$

The ADI method allows us to split a 2-D problem into two *ordinary* 1-D equations such that the first one (6.3) is solved in the x direction, and the second Equation (6.4) is solved in the y direction. The main advantage of this approach is that it enables to perform the time-advancing step in the *physical* space without the need initially to apply the transform into the spectral space in the transverse direction (as was done in the previous sections and in [4]).

We rewrite the system (6.3), (6.4) in the form

$$\mathbf{v}^*_{xx} - \lambda^2 \mathbf{v}^* = -\lambda^2 \mathbf{v}^n - \mathbf{v}^n_{yy} + \frac{1}{\nu}\mathbf{G} \equiv \mathcal{R}_1 \tag{6.5}$$

$$\mathbf{v}^{n+1}_{yy} - \lambda^2 \mathbf{v}^{n+1} = -\lambda^2 \mathbf{v}^* - \mathbf{v}^*_{xx} + \frac{1}{\nu}\mathbf{G} \equiv \mathcal{R}_2 \tag{6.6}$$

where

$$\lambda^2 = \frac{2}{\nu \Delta t}, \qquad \mathbf{G} = \frac{3}{2}(\mathbf{N} - \mathbf{F})^n - \frac{1}{2}(\mathbf{N} - \mathbf{F})^{n-1} .$$

The most precise and efficient way to evaluate the non-homogeneous terms in the RHS of (6.5), (6.6), which does not require any additional calculation of derivatives, is done in the following way:

$$\mathcal{R}_1 = -\mathcal{R}_2 - 2\lambda^2 \mathbf{v}^{n+1} + \mathbf{G}/\nu, \tag{6.7}$$

$$\mathcal{R}_2 = -\mathcal{R}_1 - 2\lambda^2 \mathbf{v}^* + \mathbf{G}/\nu \tag{6.8}$$

Equation (6.5) is solved for $\mathbf{v}^*(x_i, y_j)$ at each $y_j = j\Delta y,\ \Delta y = 2\pi/N_y,$ $j = 0, 1, ..., N_y$ and Eq. (6.6) is solved for $\mathbf{v}^{n+1}$ at each $x_i = i\Delta x,\ \Delta x = 2\pi/N_x,\ i = 0, 1, ..., N_x$ by using the LFB method.

The ADI method can be implemented for domain decomposition in the form of cells as well as strips [4, 5]. In the case of cells, the matching is performed both on the vertical (for Eq. (6.5)) and the horizontal interfaces $\bar{y}_r = r\ l_y,\ \ r = 1, 2, ..., P_y,\ l_y = 2\pi/P_y$ for Eq. (6.6).

Therefore, the general ADI computational algorithm utilizing domain decomposition in the form of cells is identical to one described in Section 3 and in [4] but the time-advancing step is performed in physical space without transforming it into spectral space.

6.2.2. Parallel algorithm for the cell computation with constant coefficients.

- **Initial computation in each processor (no time marching yet)**
  - **Initial**
    - Computation of the exact solution (for verifying the result)
    - Computation of the forcing term according to (Eq. (6.5)) which corresponds to exact solution
    - Computation of $\mathcal{R}_1$ without the non-linear terms $\mathbf{N}(\mathbf{v}^{n-q})$)
  - **Computation of the non-linear terms**
  - **Adding the non-linear terms**

    Adding the non-linear terms to $\mathcal{R}_1$ (without any time stepping)
- **Start of the main loop (advance in time)**
  - **Adams-Bashford scheme:**

    Adams-Bashford scheme is employed for the approximation of the non-linear term. It is performed for both directions (x and y).
  - **Computation of the RHS:**

    Computation of the RHS according to Eq. (6.7). It is done for the two directions.
  - **Exchange of the overlapped $\boldsymbol{\mathcal{R}_1}$ in the $\boldsymbol{x}$ direction (communication):**

    Transfer of the overlapped RHS from the two neighbors
  - **Computation of the partial solution:**

    Computation of the partial solution from Eq. (6.5).
  - **Matching in the $\boldsymbol{x}$ direction (communication):**

    As explained in Section 6.1.
  - **Computation of real velocity**
  - **Computation of $\boldsymbol{\mathcal{R}_2}$:**

    Computation of $\mathcal{R}_2$ according to Eq. (6.8).
  - **Exchange of overlapped $\boldsymbol{\mathcal{R}_2}$ in the $\boldsymbol{y}$ direction (communication):**

    Transfer of the overlapped RHS from the two neighbors
  - **Computation of the partial solution:**

    We compute the partial solution from Eq. (6.6).
  - **Exchange of boundary values (communication)**

Matching in the y direction (communication):
As explained in Section 6.1.

- **Going back to the start of the main loop**
In order to compute the non-linear term in the next time-step.

6.2.3. Solution with non-constant coefficients-preconditioned iterative method with spectral preconditioner. The following section suggests solution to problems with non-constant coefficients via spectral multidomain preconditioner. This approach is efficient when we have good local approximations in each subdomain (small variations in the coefficients from their mean values). By dividing the computational domain into a large enough number of subdomains we can guarantee it.

If we have

$$\mathcal{L}u = f \quad \text{in} \quad \omega \tag{6.9}$$

where

$$\mathcal{L}u = \frac{\partial^2 u}{\partial \eta^2} + a(\xi,\eta)\frac{\partial^2 u}{\partial \xi \partial \eta} + b(\xi,\eta)\frac{\partial^2 u}{\partial \xi^2} + c(\xi,\eta)\frac{\partial u}{\partial \eta} + d(\xi,\eta)u. \tag{6.10}$$

then, spectral approximations to the boundary value problem lead to a full $N \times N$ matrix equations for the N expansion coefficients. The solution of this algebraic system requires $O(N^2)$ storage locations and $O(N^3)$ operations to invert the matrix (using the Gauss elimination or other direct methods). Obviously, for multi-dimensional problems of realistic size (typically, with $N \sim 10^6$) the direct solution of such a system is prohibitively expensive.

The straightforward application of the iterative methods to the solution of big linear algebraic systems is also inefficient. The reason is that the typical number of iterations required to reduce the error by an order of magnitude (it can be characterized by the *condition number* κ) grows with the size N of the system. For the second order operator (6.9), $\kappa \propto N^2$ no matter whatever discretization method is being used.

The convergence of the iterations can be improved substantially, if an appropriate *preconditioner* to the original differential operator is used [22]. In this section we construct an efficient *spectral* preconditioner to $\mathcal{L}$ to obtain a rapidly convergent iterative algorithm.

The idea of preconditioning is to solve the equation

$$H^{-1}\mathcal{L}u = H^{-1}f \tag{6.11}$$

instead of Eq. (6.9). Instead of solving the problem directly we solve it iteratively using a preconditioned iterative method The preconditioned iterations (for the particularly simple Richardson scheme) look as follows:

$$Hu^{n+1} = f - (\mathcal{L} - H)u^n. \tag{6.12}$$

Both Eqs. (6.11) and (6.9) are equivalent in view of the solution but the iterative method may converge much faster for Eq. (6.11) if the preconditioner H approximates $\mathcal{L}$ in the sense that the condition number of the "compound" operator $H^{-1}\mathcal{L}$ be bounded (independent of N).

In [34], a low order (finite-difference or finite-element) approximation to $\mathcal{L}$ is commonly used in the capacity of H. The use of a higher order finite-difference preconditioner improves the rate of convergence (in comparison with a lower order approximations) but is partially offset by the additional cost of inverting a denser matrix (representing the operator, discretized on a larger stencil). Therefore, high-order finite-difference preconditioners are not useful here.

More sophisticated iterative schemes, such as conjugate gradient types method, can be used as well. If the operator $\mathcal{L}$ is approximated in a spectral basis, upon convergence of iterations (6.12) a spectrally accurate solution is obtained. The rate of convergence depends on the choice of a preconditioner H.

If the coefficients in $\mathcal{L}$ vary in small ranges, then the *spectral* preconditioner (Eq. (6.12)) obviously provides a better approximation to $\mathcal{L}$ than any low-order preconditioner. However, if the coefficients vary substantially on the computational region, then Eq. (6.12) is a poor approximation to $\mathcal{L}$.

The situation will be changed if the preconditioned iterative method is combined with the domain decomposition technique. Domain decomposition provides us with a new possibility for using *different* preconditioners in subdomains, where each one is a good constant-coefficient approximation to $\mathcal{L}$ on a small enough patch of the region. In a local preconditioner H_n the coefficients are computed by averaging over the region of subdomains ω_n. As a result, the convergence rate for the iterations can be improved considerably.

We propose to construct H in the same form as $\mathcal{L}$ but with *constant* coefficients in place of variable coefficients in $\mathcal{L}$. For instance, the preconditioner to the operator (6.9) should be

$$Hu = \frac{\partial^2 u}{\partial \eta^2} + \bar{a}\frac{\partial^2 u}{\partial \xi \partial \eta} + \bar{b}\frac{\partial^2 u}{\partial \xi^2} + \bar{c}\frac{\partial u}{\partial \eta} + \bar{d}u. \tag{6.13}$$

The coefficients $\bar{a}, \bar{b}, \bar{c}$ and $\bar{d}$ can be obtained, for example, by the averaging of the corresponding non-constant coefficients over the region ω:

$$\bar{b} = \frac{1}{L}\int_{\omega} J^{-1}(\xi,\eta)\, d\xi\, d\eta. \tag{6.14}$$

The operator H, defined in a rectangular domain ω, is trivially inverted by the Fourier method.

If the coefficients in $\mathcal{L}$ vary in small ranges, then the *spectral* preconditioner (6.13) obviously provides a better approximation to $\mathcal{L}$ than any low-order preconditioner. However, if the coefficients vary substantially on the computational region, then (6.13) is going to be a poor approximation to $\mathcal{L}$.

The situation will change if the preconditioned iteration method is combined with the domain decomposition technique. Domain decomposition provides us with the new possibility of using *different* preconditioners in subdomains, each one being a good constant-coefficient approximation to $\mathcal{L}$ on a small enough patch of the region. As a result, the convergence rate of iterations can be improved considerably.

We illustrate this conclusion by the following 1-D example

$$u'' + a(x)u' - \lambda^2 u = f(x), \qquad x \in [0,1], \quad u(0) = u(1) = 0, \tag{6.15}$$

where $a(x) = 10\ (1 + \tan\ 20(x - 0.5))$. Function $a(x)$ changes rapidly from 0 to 20 on a small interval $\Delta x \sim 1/20$ near the point $x = 0.5$. The forcing function has the form $f(x) = 2 - a + (2a + \lambda^2)x - \lambda^2 x^2$ which corresponds to the exact solution $u_{ex}(x) = x(x-1)$.

Following our approach, we divide the computational interval $x \in [0,1]$ into a number of subintervals $[x_{n-1}, x_n]$, $n = 1, ..., P$ and define a set of constant coefficient operators H_n as follows:

$$H_n = u'' + \bar{a}_n u' - \lambda^2 u, \qquad x \in [x_{n-1}, x_n], \tag{6.16}$$

where

$$\bar{a}_n = \frac{1}{x_n - x_{n-1}} \int_{x_{n-1}}^{x_n} a(x)\, dx.$$

Fig. 8 shows the variance of the coefficient $a(x)$ on the interval $x \in [0,1]$ and its step-wise approximation $\bar{a}_n$ for $P = 15$.

The convergence history of the preconditioned conjugate gradient iterations with the spectral preconditioner (6.16) is given in Fig. 9 for several domain decompositions (parameter $\lambda = 1$). When the number of subdomains varies from $P = 11$ to $P = 15$, the same asymptotic accuracy is eventually attained. However, the convergence is much *faster* for larger P, i.e., for smaller deviations $|a(x) - a_n|$ within the intervals $[x_{n-1}, x_n]$. If the deviations $|a(x) - a_n|$ are too large ($P = 9$), then a *smaller* accuracy is achieved after an affordable (some tens) number of iterations.

Note that the computation of inner products, needed when conjugate-gradient type methods is being used, leads to additional global communications which degrade somewhat the parallel performance.

6.3. Parallel implementation of cell computation with non-constant coefficients.

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t/2} = \nu(x,y)(\mathbf{v}^*_{xx} + \mathbf{v}^n_{yy}) - \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \tag{6.17}$$

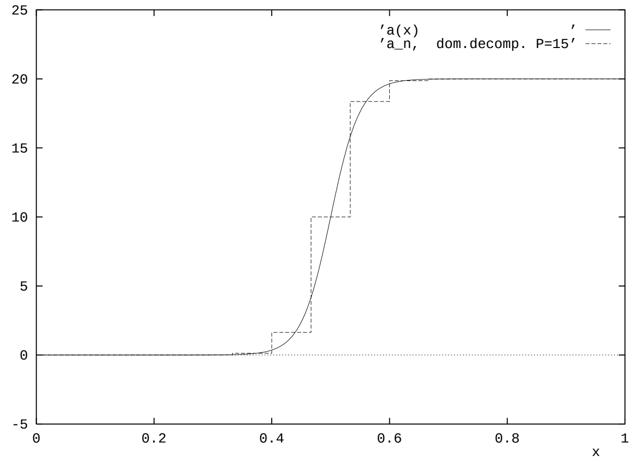

FIG. 8. *Step-wise approximation of the variable coefficient* $a(x) = 10\big(1 + \tan\, 20\,(x - 0.5)\big)$.

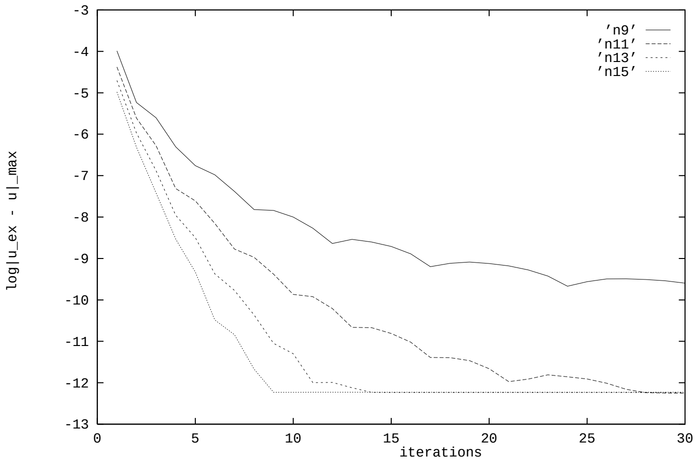

FIG. 9. *Acceleration of convergence by splitting the computational region into an increasing number of subdomains P.*

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t/2} = \nu(x,y)(\mathbf{v}^*_{xx} + \mathbf{v}^{n+1}_{yy}) - \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \tag{6.18}$$

Equation (6.17) is equivalent to Eq. (6.3) and Eq. (6.18) to Eq. (6.4) but the difference is that $\nu(x,y)$ is not constant and for example can be as the Eq. (6.29) in Section 6.4.

We divide Eqs. (6.17) and (6.18) by $\nu(x,y)$ and denote $\lambda_0^2 \triangleq (\frac{\Delta t}{2})^{-1}$. Then, Eqs. (6.17) and (6.18) become

$$\begin{aligned} &\lambda_0^2 \nu(x,y)^{-1}(\mathbf{v}^* - \mathbf{v}^n) \\ &\qquad = \mathbf{v}^*_{xx} + \mathbf{v}^n_{yy} - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.19}$$

$$\begin{aligned} &\lambda_0^2 \nu(x,y)^{-1}(\mathbf{v}^{n+1} - \mathbf{v}^*) \\ &\qquad = \mathbf{v}^*_{xx} + \mathbf{v}^{n+1}_{yy} - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.20}$$

Then, we have to solve the following pair of equations:

$$\begin{aligned} \mathbf{v}^*_{xx} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^* &= -\mathbf{v}^n_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^n \\ &\quad - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.21}$$

which was derived from Eq. (6.19), and

$$\begin{aligned} \mathbf{v}^{n+1}_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^{n+1} &= -\mathbf{v}^*_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^* \\ &\quad - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.22}$$

which was derived from Eq. (6.20).

We rewrite Eq. (6.21) as:

$$\mathbf{v}^*_{xx} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^* \triangleq \mathcal{R}_1 \tag{6.23}$$

and Eq. (6.22) as:

$$\mathbf{v}^{n+1}_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^{n+1} \triangleq \mathcal{R}_2 \tag{6.24}$$

where

$$\begin{aligned} \mathcal{R}_1 &= -\mathbf{v}^n_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^n \\ &\quad - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.25}$$

and

$$\begin{aligned} \mathcal{R}_2 &= -\mathbf{v}^*_{yy} - \lambda_0^2 \nu(x,y)^{-1} \mathbf{v}^* \\ &\quad - \nu(x,y)^{-1} \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}\right) \end{aligned} \tag{6.26}$$

In each processor (domain) we assume that $\bar{\lambda}_0^2 = \lambda_0^2 < \nu >$ where $<>$ denotes the spatial average. Then, we can use the results from Section 6.2.3 and apply them as was done in Section 6.2.2.

6.4. Model problem. The algorithm was tested and implemented using the following model problem which is a two-dimensional equation of the Navier-Stokes type [36]

$$\begin{gathered} \frac{\partial \mathbf{v}}{\partial t} = \nu \nabla^2 \mathbf{v} - \mathbf{N}(\mathbf{v}) + \mathbf{F}(\mathbf{x}, \mathbf{t}) \quad \text{in } \Omega = (0, 2\pi)^2 \ , \\ \nabla \cdot \mathbf{v} = 0 \end{gathered} \tag{6.27}$$

$$\begin{gathered} \mathbf{N}(\mathbf{v}) = (\mathbf{v} \cdot \nabla)\mathbf{v} + \frac{1}{2}(\nabla \cdot \mathbf{v})\mathbf{v} \ , \\ \mathbf{v} = (v_x, v_y), \qquad \mathbf{F} = (F_x, F_y, t), \qquad \mathbf{x} = (x, y) \end{gathered} \tag{6.28}$$

and

$$\nu(x, y) = \frac{1}{1 + \epsilon \cos kx \cos ky} \tag{6.29}$$

where the boundary conditions are

$$\mathbf{v}(x + L_x, y) = \mathbf{v}(x, y + L_y) = \mathbf{v}(x, y) \tag{6.30}$$

and the viscosity for the constant coefficients case is $Re^{-1} = \nu$ and for the non-constant coefficients $\nu = \nu(x, y)$.

The non-linear term is written in the form (6.28) which preserves the energy.

The exact solution $\mathbf{v} = (v_x, v_y)$ is:

$$\begin{aligned} v_x^{exact} &= \cos k_x x \cos k_y y \ , \\ v_y^{exact} &= \cos L_x x \cos L_y y \end{aligned} \tag{6.31}$$

6.5. Parallel speedup and efficiency. Speedup and efficiency of the 2-D Navier-Stokes parallel solver are presented in Table 3.

From Table 3 we can see that the ability to maintain efficiency at a fixed value by simultaneously increasing the number of processors and the size of the problem is the main feature of a *scalable* parallel systems. The *scalability* of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processors. It reflects a parallel system's ability to utilize increasing processing resources effectively. We demonstrated that the new proposed parallel solver for 2-D Navier-Stokes equations using ADI discretization is scalable in the above sense as we examined Table 3. When the number of processors increases and, therefore,

TABLE 3
Speedup and efficiency (in brackets) of the 2-D parallel solver of the Navier-Stokes equations using ADI. p is the number of processors in each direction and n is the number of points in one direction.

	$n = 480$			$n = 992$
p	ALPHA	MOSIX	SP2	SP2
2×2	2.12 (0.53)	2.27 (0.57)	2.52 (0.63)	2.30 (0.58)
3×3	4.66 (0.52)	4.97 (0.55)	5.71 (0.64)	5.44 (0.60)
4×4		8.92 (0.55)	10.52 (0.66)	9.95 (0.62)
6×6			24.06 (0.67)	23.20 (0.66)
7×7			34.35 (0.70)	33.32 (0.68)
8×8			46.06 (0.72)	45.44 (0.71)

the size of the problem also increases, the parallel efficiency stays almost the same, but the serial computation time increases.

From these results we see that the scalability function is approximately constant for all considering computers. It can be explained by realizing first that the communication time is a small part of the total computation time, and secondly, the time spent for synchronization of the different parts of our computational scheme was not too large. Hence, we can deduce that the 2-D ADI parallel algorithm is scalable. The efficiency of ADI algorithm is lower than of 2-D and 3-D Navier-Stokes schemes [5]. This is because in the computation in the cell topology we need to apply the local Fourier basis in both directions.

REFERENCES

[1] A. AVERBUCH, M. ISRAELI, AND L. VOZOVOI, *Domain Decomposition Methods for Solving Parabolic PDEs on Multiprocessors*, Applied Numerical Methods, 193–212, May 1993.

[2] A. AVERBUCH, M. ISRAELI, AND L. VOZOVOI, *Domain Decomposition methods with local Fourier basis for parabolic problems*, The Contemporary Mathematics volume on Domain Decomposition edited by Quarteroni et al., **157**, 223–230, 1994.

[3] A. AVERBUCH, M. ISRAELI, AND L. VOZOVOI, *Multidomain Fourier algorithms for parallel solutions of the Navier-Stokes equations*, The Contemporary Mathematics volume on Domain Decomposition Methods in Scientific and Engineering Computing, edited by D. Keyes and J. Xu, **180**, 539–546, AMS 1994.

[4] A. AVERBUCH, M. ISRAELI, AND L. VOZOVOI, *Parallel Implementation of Non-Linear Evolution Problems using Parabolic Domain Decomposition*, Parallel Computing, **21**, No. 7, 1151–1183, 1995.

[5] A. AVERBUCH, L. IOFFE, M. ISRAELI, AND L. VOZOVOI, *Highly Scalable two- and Three-Dimensional Navier-Stokes Parallel Solvers on MIMD Multiprocessors*, J. of Supercomputing, **11**, No. 1, 7–39, 1997.

[6] A. AVERBUCH, L. IOFFE, M. ISRAELI, AND L. VOZOVOI, *A Parallel Spectral Fourier – Nonlinear Galerkin Algorithm for Simulation of Turbulence*, Numerical Methods for Partial Differential Equations, 699–715, 1997.

[7] A. Averbuch, M. Israeli, and L. Vozovoi, *Parabolic domain decomposition techniques for parallel computation*, International Conf. on Spectral and High Order Methods (ICOSAHO), Montpellier, France, June 22–26, 1992.

[8] A. Averbuch, M. Israeli, an L. Vozovoi, *Domain Decomposition methods with local Fourier basis for parabolic problems*, 40–41, Domain Decomposition methods in Science and Engineering, Villa Olmo, Como, Italy, June 15–19, 1992.

[9] A. Averbuch, M. Israeli, and L. Vozovoi, *Multidomain Fourier Algorithms for Fluid Dynamics Problems*, Sixth International Congress on Computational and Applied Mathematics (ICCAM 94), Katholieke Universiteit Leuven, Belgium, July 25–30, 1994.

[10] A. Averbuch, M. Israeli, and L. Vozovoi, *Parallel Implementation of Non-Linear Evolution Problems using Parabolic Domain Decomposition*, Seventh SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, CA, Feb. 15–17, 379–384, 1995.

[11] A. Averbuch, M. Israeli, and L. Vozovoi, *Parallel Spectral Fourier Algorithm for Fluid Dynamics Problems*, ICFD Conf. on Numerical Methods for Fluid Dynamics, April 3–6, Institute for Computational Fluid Dynamics, Oxford University Computing Lab, 1995.

[12] A. Averbuch, M. Israeli, and L. Vozovoi, *Parallelization of High Order Schemes in Large Scale Scientific Computing*, Parallel CFD 95, California Institute of Technology, CA, June 26–28, 1995.

[13] A. Averbuch, M. Israeli, and L. Vozovoi, *Parallel Spectral Fourier Algorithm for Fluid Dynamics Problems*, International Conference on Spectral and High Order Methods (ICOSAHOM 95), June 5–9, Houston, 605–612, Texas, 1995.

[14] A. Averbuch, K. Ruvinsky, M. Israeli, and L. Vozovoi, *Parallel Implementation of Multidomain Fourier Algorithms for 2D and 3D Navier-Stokes Equations*, "Domain Decomposition Methods in Science and Engineering", the Eight International Conference on Domain Decomposition, May 16–20, 1995, Beijing, China, 433–442.

[15] A. Averbuch, L. Ioffe, M. Israeli, and L. Vozovoi, *Highly Scalable 2-D and 3-D Navier-Stokes Parallel Solver on MIMD Multiprocessors*, Eighth SIAM Conference on Parallel Processing for Scientific Computing, March 14–17, 1997, Minneapolis, Minnesota, USA.

[16] A. Averbuch, L. Ioffe, M. Israeli, and L. Vozovoi, *2-D Parallel Solver for the Solution of Navier-Stokes Equations with Constant and Variable Coefficients using ADI on Cells*, Parallel Computing, **24**, 673–699, 1998; also in SIAM Annual Meeting, July 14–18, 1997, Stanford University, USA.

[17] A. Averbuch, B. Epstein, L. Ioffe, and I. Yavneh, *Efficient Parallelization of a Three-Dimensional Navier-Stokes Solvers on MIMD Multiprocessors*, submitted.

[18] A. Barak, S. Guday, and R. Wheeler, *The MOSIX Distributed Operating System, Load Balancing for UNIX*, Lecture Notes in Computer Science, **672**, Springer-Verlag, 1993.

[19] A. Barak, O.Laden, and Z. Yarom, *The NOW MOSIX and its Preemptive Process Migration Scheme*, IEEE TCOS, **7**, 2:5–11, 1995.

[20] M. Brachet, D.I. Meiron, S.A. Orzag et al., *Small-scale structure of the Taylor-Green vortex*, Journal of Fluid Mechanics, **130**:411–452, 1983.

[21] G.L. Bronwing and H.O. Kreiss, *Comparison of numerical methods for the calculation of two-dimensional turbulence*, Mathematic of Computation, **52**, 186:369–388, 1989.

[22] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang, *Spectral methods in fluid dynamics*, Series in Computational Physics, Springer-Verlag, 1989.

[23] R. Coifman and Y. Meyer, *Remarques sur l'analyse de Fourier à fenêtre, série*, C. R. Acad. Sci. Paris, **312** (1991), 259–261.

[24] A. Debussche, T. Dubois, and R.Temam, *The Nonlinear Galerkin Method: A multi-scale method applied to the simulation of homogeneous turbulent flows*, Theoretical and Computational Fluid Dynamics, **7**, 4 (1995), 941–967.

[25] T. Dubois and R.Temam, *Separation of scales in turbulence using the Nonlinear Galerkin method*, Advances in Computational Fluid Dynamics, W.G. Habashi et M. Hafez, ed., Gordon and Breach Publishers, 1993.

[26] C. Foias, O. Manley, and R. Temam, *On the interaction of small and large eddies in two dimensional turbulent flows*, Math. Model. and Num. Anal., (M2AN) **22** (1988), 93–114.

[27] A. Geist, A. Beguelin et al., *PVM: parallel virtual machine*, The MIT Press, Cambridge, Mass., 1994.

[28] D. Gottlieb and R.S. Hirsh, *Parallel pseudo-spectral domain decomposition techniques*, Journal of Scientific Computing, **4**:309–325, 1989.

[29] M. Israeli, L. Vozovoi, and A. Averbuch, *Spectral Multidomain Technique with Local Fourier Basis*, Journal of Scientific Computing, **8**, No. 2, 135–149, June 1993.

[30] M. Israeli, L. Vozovoi, and A. Averbuch, *Parallelizing Implicit Algorithms for Evolution Equations by Parabolic Domain Decomposition*, Journal of Scientific Computing, **8**, No. 2, 151–166, June 1993.

[31] F.Jaubertau, C.Roisier, and R.Temam, *The Nonlinear Galerkin method in computational fluid dynamics*, Applied Numerical Mathematics, **6** (1989/90), 361–370.

[32] G. Karniadakis and S. Orszag, *Nodes, nodes and flow codes*, Physics Today, No. 3, 34–42 (1993).

[33] G.E. Karniadakis, M. Israeli, and S.A. Orszag, *High order splitting methods for the incompressible Navier-Stokes Equations*, Journal Computational Physics, **97**, 2:414–443, 1991.

[34] S.A. Orszag, *Spectral methods for problems in complex geometries*, Numerical methods for PDE's, 1979.

[35] R. Temam, *Inertial manifolds and multigrid methods*, SIAM J. Math. Anal., **21**, 154–178 (1990).

[36] R. Temam, *Stability analysis of the nonlinear Galerkin method*, Mathematics of Computation, **57**:196, 477–505, 1991.

[37] L. Vozovoi, M. Israeli, and A. Averbuch, *Spectral Multidomain Technique with Local Fourier Basis II: Decomposition into Cells*, Journal of Scientific Computing, **9**, No. 3, 311–326, 1994.

[38] L. Vozovoi, M. Israeli, A. Averbuch, *Multidomain Local Fourier Method for PDEs in Complex Geometries*, J. Computational and Applied Mathematics, **66**, 543–555, 1996.

[39] L. Vozovoi, M. Israeli, and A. Averbuch, *Analysis and Application of Fourier-Gegenbauer Method to Stiff Differential Equations*, SIAM J. Numerical Analysis, **33**, No. 5, 1844–1863, October 1996.

[40] L. Vozovoi, M. Israeli, and A. Averbuch, *Application of the Multidomain Local Fourier method for CFD in complex geometries*, Computational Fluid Dynamics, Roesner's Anniversary Volume, 245–256, Springer, 1994.

[41] T. Zang, S.E. & Krisr, *Numerical experiments on stability and transition in plane flow*, Theoret. Comput. Fluid Dyn., **1**, 41–64.

[42] D. Zernik, A. Agbaria, and Z. Yehudai, *Parallel programming tools*, IBM, Haifa Research Lab, Technical Report 88.372, October 1996.