Parallel Implementation of Multidomain Fourier Algorithm for 2-D and 3-D Navier-Stokes Equations[1]

0.1 Introduction

For long time integration of PDEs and resolving fine features of their solutions it is preferable to use high order methods, in particular spectral methods. But the *global* nature of spectral methods makes these methods difficult to parallelize with good performance. In particular, it can destroy the possibility of achieving scalability. Most numerical parallel algorithms based on spectral methods require global data transfers (such as transpose of a global matrix) and global communication. For massively parallel algorithm this may cause communication and synchronization bottlenecks.

A low communication parallel algorithms based on spectral methods were developed in [IVA1]-[IVA3] for the solution of non-linear time-dependent PDEs. The parallelization is achieved by domain decomposition. Elemental solutions in subdomains are constructed using the Local Fourier Basis (LFB) method. The idea of this method is to decompose a global problem 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 is shown in [IVA1]-[IVA3] 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 Fast Fourier Transform (FFT). Moreover, this extra work is reduced as the size of the problem grows in each domain[IVA4]. Therefore, for computation of large problems, which requires many degrees of freedom, the MDLF method is almost as efficient as the conventional Fourier method. An important advantage of this method, when compared to other multidomain spectral techniques, is that it enables a great simplification of the matching relations for the interface unknowns. For problems with constant coefficients and in simple domains, the use of the Fourier basis enables to fulfill the matching of each harmonic separately, and thus to eliminate the global coupling of the interface unknowns. Since then, the method was extended to handle complex geometries [IVA5]. The decomposition strategy of the the MDLF method can easily be changed to reflect different resolutions in each subdomain, which makes it valuable as an adaptive algorithm.

In [IVA4] the MDLF method was applied to the parallel solution of the model Navier-Stokes problem investigated previously in [T]. This model does not contain the pressure term, therefore, the time discretization procedure results in the Helmholtz equation to be solved in each time step. It was shown in [IVA4], that due to the locality properties of Helmholtz operator, application of the MLDF method to such problems is highly efficient. Matching of the solutions in subdomains requires only local communications between neighboring processors. Hence, the algorithm is fully

[1] A. Averbuch K. Ruvinsky
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
M. Israeli L. Vozovoi
Faculty of Computer Science, Technion, Haifa 32000, Israel

Domain Decomposition Methods in Sciences and Engineering, edited by R. Glowinski *et al.*

scalable. In [NS] a parallel algorithm, based on MDLF method for solution of the 2-D Navier-Stokes equations, was presented.

In the present paper the algorithm [NS] is generalized for 3-D Navier-Stokes equations and is applied to the parallel solution of the 2-D and 3-D Navier-Stokes equations. The treatment and the solution of the global Poisson equation for the pressure has the potential of degrading the performance and the speedup of the parallel solution. We show that for elliptic equation of Poisson type the size of the required global data transfer can essentially be reduced. The reason 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 (processors). The 2-D and 3-D Navier-Stokes equations are implemented on MIMD message-passing multiprocessor using PVM software package[PVM] and it achieves an almost linear speedup.

0.2 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

$$\frac{\partial \mathbf{v}}{\partial t} = Re^{-1}\nabla^2\mathbf{v} - \mathbf{N}(\mathbf{v}) - \nabla\Pi + \mathbf{F} \tag{0.1}$$

$$\nabla\Pi = \nabla\Pi' + q,\ \Gamma(x+2\pi, y+2\pi, z+2\pi) = \Gamma(x,y,z),\ \Gamma = (\Pi', \mathbf{v}),\ q = \text{const},$$

in the periodic domain $\Omega = [0, 2\pi]^3$. Here $\mathbf{v}(\mathbf{x}, t) = (u, v, w)$ is the velocity subject to the incompressibility constrain

$$\nabla \cdot \mathbf{v} = 0 \qquad \text{in} \quad \Omega, \tag{0.2}$$

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

$$\mathbf{N}(\mathbf{v}) = \mathbf{v} \times (\nabla \times \mathbf{v}). \tag{0.3}$$

The numerical solution of the problem (0.1)-(0.3) with periodic boundary conditions requires discretization in both time and space.

0.2.1 Discretization in Time

The discretization in time is performed via a splitting algorithm investigated in [KIO]:

$$\frac{\hat{\mathbf{v}} - \sum_{q=0}^{J_i-1} \alpha_q \mathbf{v}^{n-q}}{\Delta t} = \sum_{q=0}^{J_q-1} \beta_q (\mathbf{N}(\mathbf{v}^{n-q}) - \mathbf{F}^{n-q}), \tag{0.4}$$

$$\frac{\hat{\hat{\mathbf{v}}} - \hat{\mathbf{v}}}{\Delta t} = -\nabla\Pi^{n+1}, \tag{0.5}$$

$$\frac{\gamma_0 \mathbf{v}^{n+1} - \hat{\hat{\mathbf{v}}}}{\Delta t} = Re^{-1}\nabla^2\mathbf{v}^{n+1}. \tag{0.6}$$

From (0.6) it follows that $\nabla\hat{\mathbf{v}} = 0$. Therefore, Eq. (0.5) can be rewritten in the form

$$\nabla^2\Pi^{n+1} = \frac{1}{\Delta t}\nabla\cdot\hat{\mathbf{v}}, \tag{0.7}$$

This algorithm consists of an explicit advection step (0.4), a global pressure adjustment for incompressibility (0.5), (0.7) and implicit viscous step (0.6).

0.2.2 Discretization in Space

The splitting algorithm in time results in two types of elliptic equations: the Helmholtz equation

$$\nabla^2 u - \lambda^2 u = f(\mathbf{x}), \tag{0.8}$$

and the Poisson equation

$$\nabla^2 u = g(\mathbf{x}), \tag{0.9}$$

that have to be solved repeatedly (for each time step). The parameter λ in (0.8) is related to the time-stepping increment Δt, $\lambda^2 = \gamma_0 Re/\Delta t$. We consider computational domain $\Omega = [0, 2\pi]^d$, where $d = 2$ or 3 decomposed into parallel strips or slabs (further on called strips also) and the boundary conditions are periodic in all directions. We discretize the local problems in subdomains on a uniform grid. Then, we construct the local solutions using the multidomain local Fourier (MDLF) method of [IVA1]-[IVA3].

0.2.3 Solution of the Helmholtz Equation

After the application of the Fourier transform on Eq. (0.8) in the periodical directions y and z we get the 1-D equation in the nth strip:

$$\frac{d^2\hat{u}^{(n)}}{dx^2} - \tilde{\lambda}^2\hat{u} = \hat{f}^{(n)}(x). \tag{0.10}$$

In the 2-D case: $\hat{u}^{(n)} = \hat{u}_k^{(n)}$, $\hat{f}^{(n)} = \hat{f}_k^{(n)}$, $\tilde{\lambda}^2 = \lambda^2 + k^2$; in 3-D case: $\hat{u}^{(n)} = \hat{u}_{km}^{(n)}$, $\hat{f}^{(n)} = \hat{f}_{km}^{(n)}$, $\tilde{\lambda}^2 = \lambda^2 + k^2 + m^2$. Here $k = -\frac{N_y}{2}, ..., \frac{N_y}{2}$ and $m = -\frac{N_z}{2}, ..., \frac{N_z}{2}$ are harmonic numbers in y and z directions. According to the MDLF technique of [IVA1]-[IVA3] the solution of equation (0.10) in the whole domain consists of two main steps: Construction of the elemental particular solutions, matching step.

To estimate the gain of the local matching for the partial pressure harmonics in the 3-D case we compare the performance of the parallel solution of the Poisson equation (0.8) in two cases: complete global matching and local matching for partial pressure harmonics. We take the forcing in (0.8) to be $g(\mathbf{x}) = \cos kx \cdot e^{(\cos ky)(\cos kz)}$. The results are presented in Table 0.1. Local matching for partial pressure harmonics produces solution with order of accuracy of $\epsilon \sim 10^{-8}$ in comparison with complete solution that utilizes global matching. We need the following notations:

Notations:

N_g	size of square matrix of interface values of the lowest Fourier coefficients in y and z directions (included mean value) that are globally matched
T_g	executable time for the parallel algorithm where all the pressure harmonics are matched globally
T_g^c	communication time of the parallel algorithm where all the pressure harmonics are matched globally
$T_{g/l}$	executable time for the local matching of partial pressure harmonics
$T_{g/l}^c$	communication time of the local matching of partial pressure harmonics
$S_{g/l} = \frac{T_g}{T_{g/l}}$	speedup

Table 0.1 indicates that we achieve considerable savings in executable time by using local matching with partial pressure harmonics while preserving the accuracy of the solution (columns 6 and 11 in Table 0.1). This gain grows with the increase of the machine size and the amount of transferred data. This is due to the fact that the bottlenecks that are part of the complete global matching of pressure harmonics. We

P	$N_y = N_z = 128$					$N_y = N_z = 256$				
	$T_{g/l}^c$	$T_{g/l}$	T_g^c	T_g	$S_{g/l}$	$T_{g/l}^c$	$T_{g/l}$	T_g^c	T_g	$S_{g/l}$
4	17	54	41	80	1.5	73	182	168	289	1.6
6	19	50	65	91	1.8	75	185	253	405	2.2
8	20	51	89	130	2.6	74	183	362	553	3
10	22	53	121	164	3.1	75	195	507	747	3.8

Table 0.1 Timing (in ticks) and speedup $S_{g/l} = T_g/T_{g/l}$ for parallel algorithms for the Poisson equation: $n = 48$, $N_x = n \cdot P$, $N_\epsilon = 8$, $N_g = 8$.

would like to emphasize the fact that even in the case when global communication is required, a careful choice of the method may results in a lesser communication. Thus, the overhead is not increased.

0.3 Computational Algorithm for 2-D and 3-D Navier-Stokes Equations

In this section we present the algorithm that was used for solving 2-D and 3-D Navier-Stokes equations using strip topology. Initially, we define the velocity $\mathbf{U}_d(\mathbf{x}, t)$ and the pressure $P_d(\mathbf{x})$ distributions in space (see Sections 0.5). Then, we compute the "forcing" $\mathbf{F}$ using Eq. (0.1). By using time marching algorithm (Eqs. (0.4) - (0.6)) 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 deviation of the computed solution $\mathbf{v}(\mathbf{x}, t)$ from $\mathbf{U}_d(\mathbf{x}, t)$ in each time step have 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 $N_x = 128, N_y = 32, N_z = 32$, see Sec. 0.5).

Our algorithm consists of eight main steps (for detailed description of the algorithm see [AVIV]):

Step 1: Initialization
Step 2: Explicit advection step: beginning of time marching
Step 3: Calculation of the RHS of the Poisson equation
Step 4: Solution of the Poisson equation
Step 5: Matching of Π_p^{n+1} and $\partial\Pi_p^{n+1}/\partial x$ in the whole domain (partial global matching)
Step 6: Computation of the velocity $\hat{\mathbf{v}}$
Step 7: Implicit viscous step
Step 8: Matching of the particular solutions $\mathbf{v}_p^{n+1}$:Completion of the time step

After each step (from 2-6) we have to exchange data between processors. But only two data exchanges, before and after step 5, have some globality of data transfer: in step 5, we match particular solutions for the pressure Π and its derivative $\partial\Pi/\partial x$ in the whole domain. In this step only the lowest few harmonics have to be matched globally. The other four data exchanges are purely local. According to MDLF methodology the data transfer takes place between neighboring subdomains (processors).

0.4 Implementation

The proposed algorithms are appropriate for scalable distributed memory MIMD multiprocessors. Currently, the code is ported onto a distributed memory farm of DEC 3000 Model 400/400S AXP system called, from now on, as *Alpha Farm*. Each node in the farm is a high-performance desktop system that uses Digital DECchip 21064 RISC microprocessor running at 133.33 MHz. It is based on Digital Alpha AXP architecture and has ~75MB of memory space available to the user. This *Alpha Farm* consists of 10 computers connected through FDDI (Fiber Distributed Data Interface). Effective data and messages transfer through this interface is realized via *Giga Switch*, with averaged data transfer rate 3-4 MB/s.

To implement the above algorithms we used the Parallel Virtual Machine (PVM)[PVM] software package. Implementation results for Navier-Stokes equations on the *Alpha Farm* using PVM are presented in the next section.

0.5 Performance analysis

In this section we present the performance analysis of our algorithm for the solution of Navier-Stokes equations (Eq. (0.1)). The algorithm was tested according to the methodology described in Sec. 0.3. We identify the evolution of the solution under external force that has complicated behaviour in space and time. This algorithm can be easily adapted to the solution of other real problems of turbulence. For example, turbulence decay [BC], long time evolution of instable exact solution of Navier-Stokes

equation [BMO], etc. Elimination of the "forcing" term is the only thing that has to be done for this adaption.

After the completion of the extension and folding of the MDLF method [IVA1, IVA2] we apply the Fourier transform to even functions in the subdomains. Therefore, we can use the discrete cosine transform (DCT) instead of the FFT (as was done in [IVA4]). It saves 40-50% of the overall time needed for the application of the Fourier transform.

Detailed performance analysis of MDLF algorithm for 2-D and 3-D Navier- Stokes equations are presented in [AVIV]. Here we dwell briefly on the results for 3-D Navier-Stokes equations.

Notations:

$N_x,\ N_y,\ N_z$	number of collocation points in x, y and z directions
P	number of processors
$n = N_x/P$	number of points in a processor in x
N_ϵ	number of points on an overlapped interval ϵ
$N = 2(n + 2N_\epsilon)$	the size of the local FFT
T_s	best serial time
T_{DCT}	parallel time in each processor using the Discrete Cosine Transform (DCT)
$S = T_s/T_{DCT}$	speedup
T_c	communication time
$E = S/P,\ 0 < E_p < 1$	efficiency

Parallel timing

The parallel measurements for the 3-D Navier-Stokes equations are presented in Tables 0.2.

Scalability: From Table 0.2 we see that the 3-D algorithm is highly scalable: when we increase the number of processors and the problem size in each processor remains the same (i.e. the problem size increases $\sim P$), the computation time of the whole problem stays the same (T_{DCT} in columns 2 and 10 in Table 0.2. This is due to the fact that ~90% of the communication between subdomains is local.

Communication time: Our results indicate that the total computation time, T_{DCT}, strongly depends on the data transfer rate between processors. Actually, from Table 0.2(columns 3 and 11) we see that the contribution of the communication time T_c to T_{DCT} varies from 30% to 50% depending upon the problem size. In our particular *Alpha Farm* configuration the average internal data transfer rate V_c is about 3 MB/s. It should be mentioned that, currently, the data transfer rate for internal network can be ten times higher than what is available by the Giga Switch of our *Alpha Farm*. So we extrapolate our results by taking $V_c = 9$ MB/s. The results of the extrapolation are presented in Table 0.2 (under $V_c = 9$ MB/s) and in the Figs. 0.1 From this tables we can see that by increasing V_c to 9 MB/s results in 10-3 0% increase in the speedup (S) and efficiency (E).

P	$n = 488$ $N = 1024$							
	$V_c = 3$ MB/s				$V_c = 9$ MB/s			
	T_{DCT}	T_c	S	E	T_{DCT}	T_c	S	E
4	18	13	0.93	0.23	9	4	1.9	0.45
6	16	11	1.63	0.27	8	4	3.3	0.55
8	18	15	2	0.25	8	5	4.5	0.56
10	17	12	2.7	0.27	9	4	5.1	0.51
	$n = 2024$ $N = 4096$							
4	34	14	2.3	0.58	25	5	3.1	0.79
6	39	15	3.1	0.52	29	5	4.2	0.7
8	37	15	4.5	0.56	27	5	6.1	0.77
10	33	11	6.4	0.64	25	3	8.4	0.84

P	$n = 1000$ $N = 2048$							
	$V_c = 3$ MB/s				$V_c = 9$ MB/s			
	T_{DCT}	T_c	S	E	T_{DCT}	T_c	S	E
4	22	15	1.7	0.42	12	5	3	0.77
6	22	15	2.6	0.43	12	5	4.8	0.8
8	20	13	3.9	0.49	12	4	6.5	0.81
10	20	13	4.9	0.5	12	4	8.2	0.82
	$n = 4072$ $N = 8192$							
4	67	14	2.5	0.63	58	5	2.9	0.72
6	67	20	3.8	0.64	54	7	4.8	0.79
8	67	20	5.2	0.65	54	7	6.5	0.81
10	65	15	6.8	0.68	55	5	8	0.8

Table 0.2 3-D timing (in ticks), speedup and efficiency for 10 iterations: $N_y = 8$, $N_z = 8$, $N_x = n \cdot P$, $N = 2(n + 2N_\epsilon)$, $N_\epsilon = 16$.

Speedup: From Tables 0.2 and Figs. 0.1 we can see that the achieved speedup grows almost linearly with the increase of the number of processors P and this is done while the problem size in each processor stays invariant. This linear growth is explained by the fact that algorithm is highly scalable. When the number of processors increases and, therefore, the size of the problem also increases, the parallel computation time T_{DCT} stays almost the same, but the serial computation time T_s increases.

Efficiency: The efficiency of the algorithm is high and increases with the increase of the problem size. Thus, it varies from 0.3 to 0.7-0.9 (Table 0.2).

Local matching of partial pressure harmonics: In [AVIV] we had a detailed explanation that for 2-D and 3-D Poisson equation only few (lowest) harmonics of the pressure in the directions along the strips have to be matched globally. The number of globally matched

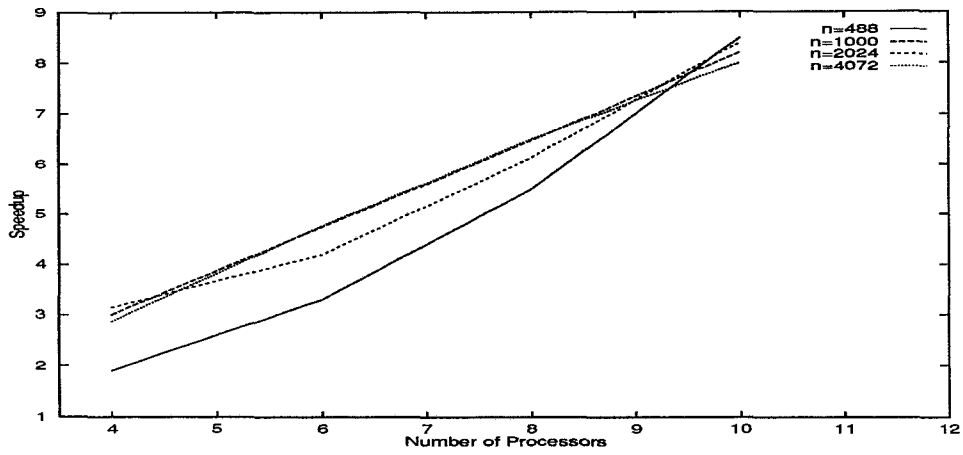

Figure 0.1 Achieved speedup when 3-D domain is decomposed into parallel strips for $N_y = 8$, $N_z = 8$.

harmonics depends on the required accuracy of the solution. Here we verify this observation on highly nonlinear solution with complex dynamics in space and time. To investigate the advantage of using local matching of partial pressure harmonics we take another extreme case, in comparison with the results in Tables 0.2. For this extreme case, we take the maximum number of harmonics (the only constraint is the memory size) that are being matched along strips. The results of these computations are presented in Table 0.3. The notations are the same as in Table 0.1. All the results reflect local matching of partial pressure harmonics at distinct N_g produce the same order of accuracy ($\epsilon \sim 10^{-4}$) as was achieved by using complete global matching of pressure harmonics. From this table we see that we need only the three lowest harmonics and mean value of the pressure to be matched globally. This results in ~20% saving in the executable time in comparison with the complete (full version) of the global algorithm. We get almost the same savings when the number of globally matched harmonics is increased (row 2 at $N_g = 32$ at Table 0.3). This is due to the fact that a packet size in the *Alpha Farm* is 16KB. Therefore, the transfer time of 4 and 32 harmonics is almost the same. From Table 0.3 we see that algorithm with local matching of partial pressure harmonics is scalable. Execution time of complete global (full version) algorithm increases with the number of processors. Global exchange of data in global algorithms can degrade the performance. Therefore, the speedup $S_{g/l}$ increases with machine size (see Table 0.3).

Thus, we show that local matching of partial pressure harmonics leads to a reduction in executable time while preserving the accuracy of the solution

P	N_g	$T^c_{g/l}$	$T_{g/l}$	T^c_g	T_g	$S_{g/l}$
10	4	600	681	694	792	1.16
4	32	568	660	621	726	1.1
6	32	546	637	632	726	1.14
8	32	559	652	680	762	1.17
10	32	562	651	694	792	1.21

Table 0.3 Timing (in ticks) and speedup $S_{g/l} = T_g/T_{g/l}$ for parallel algorithms for ten iterations; $n = 8$, $N_x = n \cdot P$, $N_y = N_z = 128$, $N_\epsilon = 4$.

Stability and Numerical Accuracy : Our implementation is stable. For example for parameters $P = 3$, $N_x = 128$, $N_y = 32$, $N_z = 32$, $N_\epsilon = 16$ and the time step $\Delta t = 10^{-3}$ the numerical error $\varepsilon = \max_{0<x,y,z<2\pi} |\mathbf{v} - \mathbf{v}_{ex}| \sim 10^{-9}$ during 7500 time units.

REFERENCES

[IVA1] Israeli M., Vozovoi L., Averbuch A., Spectral multi-domain technique with Local Fourier Basis, *J. Scientific Computing*, **8**, No. 2, (1993), pp. 181-195.

[IVA2] M. Israeli, L. Vozovoi, A. Averbuch, Parallelizing Implicit algorithms for time-dependent problems by parabolic domain decomposition, *J. Scientific Computing*, **8**, No. 2, (1993), pp. 197-212

[IVA3] Vozovoi L., Israeli M., Averbuch A., Spectral multidomain technique with Local Fourier Basis II: decomposition into cells, *J. Scientific Computing*), **9**, No. 3, (1994), pp. 311-326.

[IVA4] Averbuch A., Israeli M., Vozovoi L., Parallel implementation of non-linear evolution problems using parabolic domain decomposition, *Parallel Computing*, **21**, No. 7 (1995), pp. 1151-1183.

[IVA5] L. Vozovoi, M. Israeli, A. Averbuch, Multidomain local Fourier method for PDEs in complex geometries, to appear in *J. of Comput. Applied Math.*

[NS] L. Vozovoi, M. Israeli, A. Averbuch, Multidomain Fourier Algorithms for Parallel solution of the Navier-Stokes equation, *Contemporary Mathematics*, **180**, (1994), pp.539-546.

[KIO] G.E. Karniadakis, M. Israeli, S.A. Orszag, High order splitting methods for the incompressible Navier-Stokes Equations, *J. Computational Physics*, **97**, No. 2, (1991), pp. 414-443.

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

[BC] G.L. Browning, H.O. Kreiss, Comparison of numerical methods for the calculation of two-dimensional turbulence, *Mathematic of Computation*, **52**, No. 186, (1989), pp. 369-388.

[BMO] M. Brachet, D.I. Meiron, S. A. Orszag et. al. Small-scale structure of the Taylor-Green vortex, *J. Fluid Mech.*, **130**, (1983), pp. 411-452.

[T] R.Temam, Stability Analysis of the Nonlinear Galerkin Method, *Math. of Computation*, **57**, No. 196, (1991), p. 477-505.

[AVIV] A. Averbuch, K. Ruvinsky, M. Israeli, L. Vozovoi, Highly scalable 2-D and 3-D Navier-Stokes Parallel Solver on MIMD Multiprocessors, *submitted.*