47

Parallel Preconditioners for a Fourth-order Discretization of the Viscous Bürgers Equation

Samuel Kortas and Philippe Angot

1 Introduction

We present a parallel implementation of a new high precision conservative scheme "CFV4" that uses additive Schwarz domain decomposition methods to precondition three Krylov solvers. Parallel performance results are given for the Cray T3D. The additional use of a multigrid regular finite volume solver on each subdomain accelerates the observed elapsed times and appears optimal as soon as the number of unknowns in each subdomain is sufficient.

2 The Fourth-order Compact Conservative Scheme CFV4

We solve the 2D nonlinear unsteady viscous Bürgers equation in the bounded unit square domain $\Omega_0 = [0,1] \times [0,1]$ in the time interval $[0,T]$:

$$\begin{aligned}
\frac{\partial u}{\partial t}(\mathbf{x},t) + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - \nabla \cdot (\nu_x \nabla u)(\mathbf{x},t) &= f_u(\mathbf{x},t) \quad \text{for } (\mathbf{x},t) \in \Omega_0 \times [0,T] \\
\frac{\partial v}{\partial t}(\mathbf{x},t) + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} - \nabla \cdot (\nu_y \nabla v)(\mathbf{x},t) &= f_v(\mathbf{x},t) \qquad (2.1) \\
u|_{\partial\Omega_0}(\mathbf{x},t) = g_u(\mathbf{x},t) \quad & u(\mathbf{x},0) = u_0(\mathbf{x}) \\
v|_{\partial\Omega_0}(\mathbf{x},t) = g_v(\mathbf{x},t) \quad & v(\mathbf{x},0) = v_0(\mathbf{x})
\end{aligned}$$

where u_0, v_0, f_u, f_v, g_u, g_v are sufficiently regular functions and ν_x and ν_y are some given functions of space and time, possibly non-smooth. The 2D domain Ω_0 is meshed by uniform grid with $\Delta x = \Delta y = h$ and $[0,T]$ is divided into time steps Δt.

Because our final application is the unsteady incompressible Navier-Stokes system, we only consider divergence-free solutions satisfying (2.1). For these, the conservative

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

4^{th}-order time discretized formulation of (2.1) is integrated on a control volume $\mathcal{V}$ and the advective terms are linearized in time as follows:

$$\int_{\mathcal{V}} \frac{\Phi(u^{n+1},...,u^{n-3})}{12\Delta t} dv + \int_{\partial\mathcal{V}} u^m (u^*, v^*)^T \cdot\ \mathbf{n} d\sigma - \int_{\partial\mathcal{V}} \nu_x \nabla u^{n+1} \cdot\ \mathbf{n} d\sigma = \int_{\mathcal{V}} f_u^{n+1} dv \quad (2.2)$$

$$\int_{\mathcal{V}} \frac{\Phi(v^{n+1},..,v^{n-3})}{12\Delta t} dv + \int_{\partial\mathcal{V}} v^m (u^*, v^*)^T \cdot\ \mathbf{n} d\sigma - \int_{\partial\mathcal{V}} \nu_y \nabla v^{n+1} \cdot\ \mathbf{n} d\sigma = \int_{\mathcal{V}} f_v^{n+1} dv \quad (2.3)$$

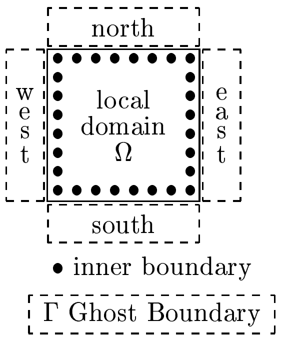
where $\Phi(\psi^{n+1},...,\psi^{n-3}) = 25\psi^{n+1} - 48\psi^n + 36\psi^{n-1} - 16\psi^{n-2} + 3\psi^{n-3}$ is chosen in order to obtain a 4^{th}-order discretization of $\frac{\partial\psi}{\partial t}$ at $(n+1)\Delta t$ and $\psi^* = 4\psi^n - 6\psi^{n-1} + 4\psi^{n-2} - \psi^{n-3}$ is the consistent 4^{th}-order Richardson extrapolation of ψ^{n+1}. We obtain two 4^{th}-order accurate time discretizations, whether the advective terms are treated implicitly $(m = n+1)$ or explicitly $(m = *)$.

Figure 1 Block structure of $\mathbf{A}_u$

$$\mathbf{A}_u = \begin{pmatrix} DDDPPPPP \\ DDDDPPPP \\ DDDDDPPP \\ PDDDDDPP \\ PPDDDDDP \\ PPPDDDDD \\ PPPPDDDD \\ PPPPPDDD \end{pmatrix}$$

with $\begin{cases} \text{D: Dense Matrix} \\ \text{P: Pentadiagonal Matrix} \end{cases}$

Figure 2 Ghost-cells and the local domain



In this conservative formulation, we need to evaluate the fluxes of the advective and diffusive terms at the surface $\partial\mathcal{V}$ of $\mathcal{V}$ with a 4^{th}-order spatial consistency. To do so, we derive linear "compact-like" relations [Lel92] linking discrete values of (u, v) defined at the center of each control volume on one line or column of the mesh, with their first derivatives taken at the center of the horizontal and vertical interfaces of these control volumes $\mathcal{V}$.

We obtain a matricial expression $\mathbf{A}_u.\ \mathbf{u}^{n+1}$, where $\mathbf{u}^{n+1}$ gathers all the u-unknowns at $(n+1)\Delta t$ stored in lexicographical order. Because the nonsymmetric and rather dense-profile matrix $\mathbf{A}_u$ (Figure 1) is expensive to build explicitly, we prefer to evaluate $\mathbf{A}_u.\ \mathbf{u}$ by calculating the solution of tridiagonal systems: 2 along each row and 2 along each column. For each point, we also need to perform four 5-point-stencil discrete 1D integration and one 5×5 point-stencil discrete 2D integration. Similar relations hold for v. Full details on the calculation of $\mathbf{A}.\ \mathbf{U} = (\mathbf{A}_u.\ \mathbf{u}, \mathbf{A}_v.\ \mathbf{v})^T$ are given in [KA96a] and the efficient parallel resolution of tridiagonal systems distributed over a row or column of processors is discussed in [KA96b].

3 Parallel Schwarz-Krylov Solvers

Parallel Partitioned Solvers

Because $\mathbf{A}$ is nonsymmetric, we implement some Krylov-type solvers to solve $\mathbf{A}.\ \mathbf{U} = \mathbf{f}$ at each time step: the straightforward implementation of BiCGSTAB and BiCGSTAB(2) solvers [Van95] on a MIMD machine, and the partitioned BiCGSTAB and GMRESR(k) methods [VV91] preconditioned by an additive Schwarz method.

Thanks to the use of "stencils of communication" and global reduction operations, these algorithms are implemented in a pleasant and easy-to-read form particularly well-suited to overlapping or nonoverlapping domain decomposition methods. As shown on Figure 2, each local subdomain Ω is extended with a ghost-cell boundary that contains either duplicate values of grid points known by the immediate neighbor processors or values set by the discretization of the boundary conditions on $\partial\Omega_0$.

A call to REFRESH_NEWS($\mathbf{u}$,$\mathcal{S}_{Default}$) duplicates the neighbor values of $\mathbf{u}$ contained in adjacent processors in the four cardinal directions (N,E,W,S). All operations preceded by GLOBAL_ are performed on each node before calling a global native reduction routine that sends back the global result to all nodes. A complete description of the implementation and performance of this model of programming on IBM SP2, Cray T3D, and IPSC/i860 can be found in [KA96b] or [AKF96].

Figure 3 Parallel BiCGSTAB

SET $\mathbf{x}_0(=\ \mathbf{u}^*\ or\ \mathbf{u}^*_{rec})$, $\epsilon(=10^{-8})$, ϵ_{stop}
REFRESH_NEWS($\mathbf{x}_0$,$\mathcal{S}_{Default}$)
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\ \mathbf{x}_0,\ \widehat{\mathbf{r}_0} = \mathbf{r}_0,\ \mathbf{v} = \mathbf{p} = 0$
$\rho_0 = \alpha = \omega = 1$
WHILE $\frac{\text{GLOBAL\_}\|\ \mathbf{r}_k\|_2}{\text{GLOBAL\_}\|\ \mathbf{r}_0\|_2} > \epsilon$
  and GLOBAL_$\|\ \mathbf{r}_k\|_2 > \epsilon_{stop}$ DO
    $\rho =$ GLOBAL_($\widehat{\mathbf{r}_0}^T\ \mathbf{r}$)
    $\beta = \alpha\rho/\rho_0\omega,\ \rho_0 = \rho$
    $\mathbf{p} = \mathbf{r} + \beta(\ \mathbf{p} - \omega\ \mathbf{v})$
    SOLVE $\mathbf{M}\widehat{\mathbf{p}} = \mathbf{p}$
    REFRESH_NEWS($\widehat{\mathbf{p}}$,$\mathcal{S}_{Default}$)
    $\mathbf{v} = \mathbf{A}\widehat{\mathbf{p}}$
    $\alpha = \rho_1/$ GLOBAL_($\widehat{\mathbf{r}_0}^T\ \mathbf{v}$)
    $\mathbf{s} = \mathbf{r} - \alpha\ \mathbf{v}$
    SOLVE $\mathbf{M}\ \mathbf{z} = \mathbf{s}$
    REFRESH_NEWS($\mathbf{z}$,$\mathcal{S}_{Default}$)
    $\mathbf{t} = \mathbf{A}\ \mathbf{z}$
    $\omega = \frac{\text{GLOBAL\_}(\ \mathbf{t}^T\ \mathbf{s})}{\text{GLOBAL\_}(\ \mathbf{t}^T\ \mathbf{t})}$
    $\mathbf{x} = \mathbf{x} + \alpha\ \widehat{\mathbf{p}} + \omega\ \mathbf{z}$
    $\mathbf{r} = \mathbf{s} - \omega\ \mathbf{t}$
ENDWHILE

Figure 4 Parallel GMRESR(k)

SET $\mathbf{x}_0(=\ \mathbf{u}^*\ or\ \mathbf{u}^*_{rec})$, $\epsilon(=10^{-8})$, ϵ_{stop}
REFRESH_NEWS($\mathbf{x}_0$,$\mathcal{S}_{Default}$)
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\ \mathbf{x}_0$
$k = 0$
WHILE $\frac{\text{GLOBAL\_}\|\ \mathbf{r}_k\|_2}{\text{GLOBAL\_}\|\ \mathbf{r}_0\|_2} > \epsilon$
  and GLOBAL_$\|\ \mathbf{r}_k\|_2 > \epsilon_{stop}$ DO
    $k = k + 1$
    SOLVE $\mathbf{M}_k\ \mathbf{v}_k^{(1)} = \mathbf{r}_{k-1}$
    REFRESH_NEWS($\mathbf{v}_k^{(1)}$,$\mathcal{S}_{Default}$)
    $\mathbf{c}_k^{(1)} = \mathbf{A}\ \mathbf{v}_k^{(1)}$
    FOR $i = 1, \cdots, k-1$ DO
      $\alpha_i =$ GLOBAL_($\mathbf{c}_i^T\ \mathbf{c}_k^{(i)}$)
      $\mathbf{c}_k^{(i+1)} = \mathbf{c}_k^{(i)} - \alpha_i\ \mathbf{c}_i$
      $\mathbf{v}_k^{(i+1)} = \mathbf{v}_k^{(i)} - \alpha_i\ \mathbf{v}_i$
    $\mathbf{c}_k = \mathbf{c}_k^{(k)}/$ GLOBAL_ $\|\ \mathbf{c}_k^{(k)}\|_2$
    $\mathbf{v}_k = \mathbf{v}_k^{(k)}/$ GLOBAL_ $\|\ \mathbf{c}_k^{(k)}\|_2$
    $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{v}_k$ GLOBAL_($\mathbf{c}_k^T\ \mathbf{r}_{k-1}$)
    $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{c}_k$ GLOBAL_($\mathbf{c}_k^T\ \mathbf{r}_{k-1}$)
ENDWHILE

Parallel Preconditioners

Figure 3 shows the parallel version of a preconditioned BiCGSTAB algorithm. If $\mathbf{M} = \mathbf{I}$, we obtain a "natural" parallel partitioned implementation of this solver: **Part_BiCGSTAB**. Similarly, we implement **Part_BiCGSTAB(2)**.

Following [Meu91], [Le 94] or [Ang94], we precondition these solvers by an additive Schwarz domain decomposition method to solve at each iteration $\mathbf{M}\widehat{\mathbf{p}} = \mathbf{p}$ or $\mathbf{M}\ \mathbf{z} = \mathbf{s}$. In this preconditioning step, we first extend the local contribution of $\mathbf{s}$ or $\mathbf{p}$ to the extended overlapping local subdomain with calls to communication stencil routines. On each subdomain handled by a different node, we solve "exactly" a local problem, similar to (2.2–2.3) taken this time with homogeneous Dirichlet boundary condition. A BiCGSTAB(2) solver is used to reduce the local residual by eight orders of magnitude. During this step, no communication occurs. No coarse grid solver is implemented. The global solution vector $\widehat{\mathbf{p}}$ or $\mathbf{z}$ is finally rebuilt from the projections of the solution of each local problem.

By keeping the same 4^{th}-order accurate spatial discretization for the local problems than for the initial one, we obtain a "classical" Krylov-Schwarz additive solver: **DDM4(l)_BiCGSTAB** where $\delta = l.h$ is half of the overlap within subdomains. As shown on Figure 4, we also implement a GMRESR solver preconditioned as above: **DDM4(l)_GMRESR(k)**.

We can also discretize and solve the local problems thanks to classical second-order finite volumes. These algorithms – **DDM2(l)_BiCGSTAB** and **DDM2(l)_GMRESR(k)** – are eventually accelerated by using a multigrid solver instead of BiCGSTAB(2). In parallel on each subdomain, and totally independently, V-Cycles are carried out on a hierarchy of grids. We perform 3 pre- and 2 post-smoothing Gauss-Seidel iterations per level, and we solve the 8×8 coarsest problem by BiCGSTAB to obtain the solvers: **DDM2MG(l)_BiCGSTAB** and **DDM2MG(l)_GMRESR(k)** (for the test problem studied here, $k = 15$ avoids restarting the GMRES algorithm).

We also test the efficiency of a reconjugation technique combined with the GMRESR-type solvers. Inspired by [Rou95], we store the "best" descent directions $(\mathbf{c}_{k\_rec}, \mathbf{v}_{k\_rec}) = (\mathbf{c}_k, \mathbf{v}_k)$ satisfying a bound on $\text{GLOBAL\_}(\mathbf{c}_k^T\,\mathbf{r}_{k-1})$. In the following time steps, these directions are reused to start the iteration process from a better initial guess $\mathbf{x}_0 = \mathbf{u}^*_{rec} = \mathbf{u}^* + \mathbf{v}_{k\_rec}\ \text{GLOBAL\_}(\mathbf{c}^T_{k\_rec}(\mathbf{b} - \mathbf{A}\ \mathbf{u}^*))$.

4 Convergence Results

As detailed in [KA96a], fourth-order accuracy in time and space is reached with such a discretization. In the following, we only present results on a test solution for a dipole diagonally crossing Ω_0 for $t \in [0, T = 1]$: $\big(u(x,y,t) = 10\,(t-y)\,e^{\frac{-(t-x)^2-(t-y)^2}{\nu}}, v(x,y,t) = 10\,(x-t)\,e^{\frac{-(t-x)^2-(t-y)^2}{\nu}}\big)$. All the tests are run with diffusive and advective terms both treated implicitly (i.e., $m = n+1$ in 2.2-2.3). Results when the advective terms are treated explicitly are collected in [Kor97].

Table 1 displays the convergence rate $\rho_{10} = \sqrt[N]{\frac{\|\mathbf{r}_N\|_2}{\|\mathbf{r}_0\|_2}}$ observed after ten time steps for the test solution solved on a 256×256 mesh with $\Delta t = 1.25 \times 10^{-2}$,

Table 1 Convergence Rate for 256×256 Dipole Problem with no Richardson Extrapolation ($\epsilon = 10^{-8}$, $\epsilon_{stop} = 10^{-4}$)

Algorithm	2 dom.	4 dom.	8 dom.	16 dom.	32 dom.	64 dom.
Part_BiCGSTAB	5.70e-1	5.70e-1	5.80e-1	5.80e-1	5.80e-1	5.80e-1
Part_BiCGSTAB(2)	2.20e-1	2.20e-1	2.20e-1	2.10e-1	2.30e-1	2.20e-1
DDM4(2)_BiCGSTAB	-	-	-	1.60e-2	2.00e-2	1.90e-2
DDM4(4)_BiCGSTAB	-	-	7.80e-4	8.30e-4	1.40e-3	1.70e-3
DDM4(8)_BiCGSTAB	-	1.30e-5	6.60e-5	2.30e-5	2.50e-5	3.60e-5
DDM2(2)_BiCGSTAB	na	3.60e-2	3.70e-2	3.60e-2	4.50e-2	4.90e-2
DDM2(4)_BiCGSTAB	6.70e-3	9.80e-3	9.80e-3	1.00e-2	1.20e-2	1.20e-2
DDM2(8)_BiCGSTAB	6.70e-3	7.40e-3	6.40e-3	6.60e-3	8.00e-3	9.00e-3
DDM2MG(8)_BiCGSTAB	5.90e-3	7.60e-3	1.30e-2	1.40e-2	1.40e-2	1.80e-2
DDM4(2)_GMRESR	-	-	9.20e-2	9.40e-2	1.10e-1	1.20e-1
DDM4(4)_GMRESR	-	-	2.60e-2	2.70e-2	2.90e-2	3.30e-2
DDM4(8)_GMRESR	-	2.40e-3	2.50e-3	2.70e-3	3.20e-3	4.30e-3
DDM2(2)_GMRESR	1.30e-1	1.70e-1	1.70e-1	1.70e-1	1.90e-1	2.00e-1
DDM2(4)_GMRESR	7.10e-2	8.40e-2	8.50e-2	8.70e-2	9.40e-2	1.00e-1
DDM2(8)_GMRESR	5.10e-2	5.40e-2	5.60e-2	5.70e-2	6.30e-2	6.80e-2
DDM2MG(8)_GMRESR	na	7.20e-2	1.00e-1	1.00e-1	1.00e-1	9.50e-2

$\nu = 10^{-2}$, $\epsilon = 10^{-8}$, $\epsilon_{stop} = 10^{-4}$, and N the number of iterations needed to meet this convergence criteria ($N > 10$). No Richardson extrapolation is used and no reconjugation technique is activated to ameliorate the initial guess (i.e., $\mathbf{x}_0 = \mathbf{u}^n$). As awaited for a DDM-based method, the convergence rate appears better when the overlap δ increases, and worse when the number of involved subdomains increases. The better rate observed in the case of **DDM4_BiCGSTAB** solvers comes from the preconditioning step called twice for this algorithm instead of just once for the **DDM4_GMRESR** solvers.

Table 2 illustrates a classical result: if the overlap δ is kept proportional to the size of the local subdomains H, the number of iterations needed for a domain decomposition preconditioned solver is asymptotically bounded independently of h. With no preconditioner, one asymptotically needs $O(1/h)$ iterations as shown for **Part_BiCGSTAB**. Other calculations not presented here also show independence in H/h.

So far, we have not investigated the need for a coarse grid solver but a forthcoming article will further examine the "scaling" behavior of the convergence rates.

5 Parallel Performance Results

Even if the convergence rate is better, the elapsed times of an algorithm on up-to-date parallel computers can vary widely, being highly sensitive to the workload compared to the amount of communication needed to run an algorithm on several processors. Domain decomposition techniques are therefore particularly well-adapted to distributed computing because of the high data locality of the preconditioning step. We present here timing results measured on the Cray T3D.

In Table 3, we first observe a certain hierarchy among the solvers tested in Parallel.

Table 2 Behavior of the different algorithms when the relative overlap $\frac{\delta}{H}$ is kept constant. Test made on 64 Cray T3D nodes, to calculate one time step ($\epsilon = 10^{-16}$ $\epsilon_{stop} = 10^{-20}$)

Algorithm	global mesh	local mesh	l	# iterations	time(s)
Part_BiCGSTAB	128× 128	16× 16	-	34	1
Part_BiCGSTAB	256× 256	32× 32	-	75	8
Part_BiCGSTAB	384× 384	48× 48	-	107	23
Part_BiCGSTAB	512× 512	64× 64	-	147	55
Part_BiCGSTAB	640× 640	80× 80	-	204	117
Part_BiCGSTAB	768× 768	96× 96	-	223	181
DDM4(l)_BiCGSTAB	128× 128	16× 16	1	15	15
DDM4(l)_BiCGSTAB	256× 256	32× 32	2	10	73
DDM4(l)_BiCGSTAB	384× 384	48× 48	3	12	323
DDM4(l)_BiCGSTAB	512× 512	64× 64	4	9	537
DDM4(l)_BiCGSTAB	640× 640	80× 80	5	10	1233
DDM4(l)_BiCGSTAB	768× 768	96× 96	6	-	-
DDM2(l)_BiCGSTAB	128× 128	16× 16	1	16	3
DDM2(l)_BiCGSTAB	256× 256	32× 32	2	13	20
DDM2(l)_BiCGSTAB	384× 384	48× 48	3	12	60
DDM2(l)_BiCGSTAB	512× 512	64× 64	4	11	129
DDM2(l)_BiCGSTAB	640× 640	80× 80	5	11	249
DDM2(l)_BiCGSTAB	768× 768	96× 96	6	11	418
Part_BiCGSTAB	120× 120	15× 15	-	32	2
Part_BiCGSTAB	240× 240	30× 30	-	64	7
Part_BiCGSTAB	480× 480	60× 60	-	129	42
Part_BiCGSTAB	960× 960	120× 120	-	319	400
DDM2MG(l)_BiCGSTAB	120× 120	15× 15	1	16	3
DDM2MG(l)_BiCGSTAB	240× 240	30× 30	2	14	17
DDM2MG(l)_BiCGSTAB	480× 480	60× 60	4	14	86
DDM2MG(l)_BiCGSTAB	960× 960	120× 120	8	14	194

Remark: In tabulated results, a dash "-" means that the calculation was not pursued because of a predicted high elapsed time of computation. An "na" means that the result is not available.

Table 3 Elapsed time (s) to perform 10 time steps for a 256×256 Problem on Cray T3D with no Richardson Extrapolation ($\epsilon = 10^{-8}$, $\epsilon_{stop} = 10^{-4}$)

Algorithm	2 dom.	4 dom.	8 dom.	16 dom.	32 dom.	64 dom.
Part_BiCGSTAB	629	307	158	78	46	25
Part_BiCGSTAB(2)	582	293	153	78	44	25
DDM4(2)_BiCGSTAB	-	-	-	1432	778	396
DDM4(4)_BiCGSTAB	-	-	1745	909	519	255
DDM4(8)_BiCGSTAB	-	2155	1239	711	435	245
DDM2(2)_BiCGSTAB	-	775	400	208	106	56
DDM2(4)_BiCGSTAB	833	655	352	190	105	57
DDM2(8)_BiCGSTAB	873	666	359	218	125	80
DDM2MG(8)_BiCGSTAB	628	327	177	86	49	26
DDM4(2)_GMRESR	-	-	-	1190	685	329
DDM4(4)_GMRESR	-	-	1704	890	513	258
DDM4(8)_GMRESR	-	2151	1244	712	443	247
DDM2(2)_GMRESR	1059	739	379	197	108	57
DDM2(4)_GMRESR	845	552	295	159	85	52
DDM2(8)_GMRESR	771	506	291	168	99	59
DDM2MG(8)_GMRESR	na	267	166	73	41	22

For all purely partitioned solvers, **Part_BiCGSTAB(2)** and **Part_BiCGSTAB** show the same efficiency and score surprisingly well. In comparison, the "classical" domain decomposition preconditioning technique performs poorly: in all configurations of overlap and amount of nodes, **Part_BiCGSTAB** and **Part_BiCGSTAB(2)** greatly exceed **DDM4(l)_BiCGSTAB** and **DDM4(l)_GMRESR** in term of elapsed time. With a 4^{th}-order discretization, the better rate of convergence obtained thanks to DDM-preconditioning doesn't counterbalance the overload of computation demanded to solve each local problem quasi-exactly.

One can significantly improve this disappointing result with a conventional second-order finite volume solver combined with a multigrid acceleration technique. As shown on Tables 4 and 3, the resolution of the local problems discretized at the order 2 scores better than classical DDM4-preconditioned solver. Only a multigrid acceleration of this DDM2-solver equals or performs better than **Part_BiCGSTAB** or **Part_BiCGSTAB(2)**. Table 2 clearly tilts in favor of a multigrid second order preconditioner as soon as the amount of local points is sufficiently important to achieve a decent acceleration (194s vs 400s for the 960×960 mesh).

The influence of the size of the overlapping δ exhibits different optimal values for each algorithm. However, for the Krylov additive Schwarz solvers taken with an optimal overlapping, the GMRESR-based solver always appears slightly faster than BiCGSTAB-global solvers. It may come from the different number of synchronization points present in both algorithms: they are more numerous in GMRESR but localized in the orthogonalization loop whereas they are spread out all over the algorithm for BiCGSTAB. The collected global reduction operations are not indeed as penalizing as the alternation of stencil calls.

Finally, if compared with Table 3, Table 4 underlines the important influence of the Richardson extrapolation taken as initial guess: for the same stopping criteria ϵ_{stop},

Table 4 Elapsed time observed (s) to perform 10 time steps for a 256×256 Dipole Problem on Cray T3D with Richardson Extrapolation ($\epsilon = 10^{-8}$ and $\epsilon_{stop} = 10^{-4}$)

Algorithm	2 dom.	4 dom.	8 dom.	16 dom.	32 dom.	64 dom.
Part_BiCGSTAB	455	226	119	59	33	19
Part_BiCGSTAB(2)	467	230	123	60	35	19
DDM4(8)_BiCGSTAB	-	2183	1269	719	447	246
DDM2(8)_BiCGSTAB	683	432	240	144	84	48
DDM2MG(8)_BiCGSTAB	411	238	146	74	40	22
DDM2(8)_GMRESR	553	370	212	123	78	50
DDM2MG(8)_GMRESR	na	205	128	53	30	16
DDM4(8)_GMRESR/Reconj	-	1649	946	542	330	185
DDM2(8)_GMRESR/Reconj	568	360	207	119	72	43
DDM2MG(8)_GMRESR/Reconj	na	190	117	55	30	17

then observed elapsed time highly decreases for all tested algorithms. One can also note the insignificant impact of the reconjugation technique.

6 Acknowledgement

The authors gratefully acknowledge use of the Cray T3D located at the C.E.A. Grenoble and I.D.R.I.S., and of the IBM SP2 at C.N.U.S.C. The research of the first author is supported by the CEA-CISI-CNRS BDI scholarship contract #740 384 CEA/DAM/CEL-V #3633.

REFERENCES

[AKF96] Angot P., Kortas S., and Fürst J. (1996) Parallel and distributed multi-domain methods for numerical fluid dynamics. In Bubak M. and Mościński J. (eds) *High Performance Computing in Europe on IBM Platforms, Sup'Eur 96 Conf. Proc.*, pages 111–120. ACC CYFRONET, Kraków.

[Ang94] Angot P. (1994) Parallel multi-level and domain decomposition methods. *Calculateurs Parallèles, L.T.C.P.* 6: 9–14.

[KA96a] Kortas S. and Angot P. (1996) A new class of high order compact finite volume schemes. *Numerical Methods for PDEs* To submit.

[KA96b] Kortas S. and Angot P. (June 1996) A practical and portable model of programming for iterative solvers on distributed memory machines. *Parallel Computing* 22: 487–512.

[Kor97] Kortas S. (1997) *Résolution Haute Précision des équations de Navier-Stokes sur machines parallèles à mémoire distribuée.* PhD dissertation in applied mathematics, Université de Provence, C.M.I. in progress.

[Le 94] Le Tallec P. (1994) Domain decomposition methods in computational mechanics. *Computational Mechanics Advances* 1: 121–220.

[Lel92] Lele S. K. (1992) Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics* 103: 16–42.

[Meu91] Meurant G. A. (1991) Numerical experiment with a domain decomposition method for parabolic problems on parallel computers. In Glowinski R., Kuznetsov Y. A., Meurant G. A., Périaux J., and Widlund O. B. (eds) *Proc. Fourth Int. Conf.*

on Domain Decomposition Meths., chapter 32, pages 394–408. SIAM, Philadelphia.

[Rou95] Roux F.-X. (1995) Parallel implementation of a domain decomposition method for non-linear elasticity problems. In Keyes D. E., Saad Y., and Truhlar D. G. (eds) *Domain-Based Parallelism and Problem Decomposition Methods in Computational Sciences and Engineering*, chapter 10, pages 161–175. SIAM, Philadelphia.

[SBG96] Smith B., Bjørstad P., and Gropp W. (1996) *Domain Decomposition, Parallel Multilevel Methods for Elliptic Differential Equations.* CAMBRIDGE University Press.

[Van95] Van der Vorst H. A. (May 1995) Parallel iterative solution methods for linear systems arising from discretized PDE's. Lecture Notes on Parallel Iterative Methods for discretized PDE's. AGARD Special Course on Parallel Computing in CFD, available from http://www.math.ruu.nl/people/vorst/#lec.

[VV91] Van der Vorst H. A. and Vuik C. (1991) GMRESR: A family of nested GMRES methods. Technical Report DUT-TWI-91-80, Delft University of Technology, Department of Technical Mathematics and Informatics, Delft, The Netherlands. avalaible from ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1991/DUT-TWI-91-80.ps.gz.