*Proc. Natl. Sci. Counc. ROC(A)*
Vol. 25, No. 3, 2001. pp. 172-184

# Domain-based Parallel Flow Computation with Interface Preconditioner

DANIEL LEE AND MULDER YU

*Computational Mathematics Group*
*National Center for High-Performance Computing*
*Hsinchu, Taiwan, R.O.C.*

(Received March 3, 2000; Accepted July 12, 2000)

**ABSTRACT**

An interface problem is tackled as a preconditioner for the nonlinear block Jacobi domain decomposition (DD) approach. Various preconditioners are investigated in solving convection-diffusion and incompressible Navier-Stokes (NS) equations, with an optional fine level interface problem solved as a further preconditioner. In addition, a (global) coarse level problem is designed as a preconditioner. Examined also is the relaxation type preconditioner. The Successive Over-Relaxation (SOR) type strategy, in simple or hybrid form, can be used to accelerate the convergence of the interface variables, so as to provide an interface preconditioner for the global problem. Furthermore, one can over-relax the setup of the interface problems, resulting in an accelerated interface preconditioner. The nature of these preconditioners is quite different from that in linear DD theory and its application. These preconditioned nonlinear DD methods exhibit impressive improvement over the basic non-preconditioned parallel Newton-Jacobi method.

**Key Words:** domain decomposition (DD), Newton method, finite volume (FV), parallel computation

## I. Introduction

This paper is concerned with parallel computation for solving the convection-diffusion equation (Morton, 1996) and the incompressible Navier-Stokes (NS) equation (Ethier and Steinman, 1994) via the Newton-Schwarz method, a nonlinear domain decomposition (DD) method.

Many research efforts have focused on the DD method (Glowinski *et al*., 1988, 1991, 1997; Chan *et al.*, 1989, 1990; Keyes *et al*., 1992; Quarteroni *et al*., 1994; Keyes and Xu, 1994; Bjørstad *et al*., 1997; Mandel, 1998; Lai *et al*., 1998). While these studies have mostly focused on linear theory, there has also been interest recently (Cai *et al*., 1996; Lee and Chen, 1998; Lee and Yu, 1999) in the application of DD in a nonlinear setting. A recent review on this topic is that of Gropp *et al*. (1998).

We propose in this paper a nonlinear approach to solving the (nonlinear) algebraic system resulting from finite-volume-difference discretization of partial differential equation. Our proposed setup combines the advantages of the overlapping and the nonoverlapping DD approach. The basic parallel Newton-Jacobi method and many preconditioning methods are investigated here. A comparative study on the accuracy and efficiency of these methods is presented. The model equations considered in this paper are the convection-diffusion and incompressible NS equations.

The governing equations are solved via a time marching procedure. To maintain use of only the compact scheme, the fourth-order scheme (Lee, 1999; Jea *et al*., 1999; Lee and Yu, 1997) is adopted to achieve higher order accuracy, therefore allowing for the possibility of a larger time step. At each time step, the associated pde is a boundary value problem, which yields a nonlinear algebraic system. Various methods have been developed to solve this problem. Among these are the SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) scheme (Patankar, 1980) and many of its variants. While these methods focus on solving iteratively linearized versions of the pde, we will solve the discrete algebraic system using a Newton-like method (Brown and Saad, 1990).

Parallel computation is performed here via the DD approach. The nonlinear algebraic system can be solved using the block Jacobi method, with or without overlap. Nonlinear Schur-complement type decomposition is also admissible, which tackles the interface problem and actually yields more accurate result (Lee and Chen, 1998), due to the capability of providing global information update. We note that the interface variables were excluded from all the other subproblems in the study of Lee and Chen (1998), as are usually adopted in similar linear DD setup. In the current paper, the subdomain variables form a (nonoverlapping) partition of the whole global system of equations. The interface problem is regarded as a preconditioner for the global discrete system. Since it is able to produce 'global update', the coarse grid type preconditioner has played an important role in the study of

linear DD theory and application. The nature of the coarse level preconditioner, in the current nonlinear framework, is quite different from that in the linear DD case. Another two-level preconditioner is investigated here by introducing a fine level interface problem, with or without nonlinear defect correction. A relaxation type interface preconditioner is also examined here. The accuracy and efficiency of all these methods and options are investigated in light of the model problems, in hopes of yielding useful guidance for more complicated applications.

We describe the problem formulation, finite volume discretization and numerical method in Section II. The non-linear DD approach is introduced in Section III, together with some details on the design of an interface preconditioner. Described in Section IV are the test cases and numerical experiments. A brief conclusion is given in Section V.

# II. Problem Formulation and Solution Procedure

## 1. The Test Problems

We consider the two-dimensional incompressible NS equation in primitive variables for a Cartesian coordinate system in non-dimensional form as follows (Tannehill *et al.*, 1997):

(1) Continuity equation:

$$u_x + v_y = 0; \tag{1}$$

(2) X-momentum equation:

$$u_t + uu_x + vu_y = -p_x + \frac{1}{Re}(u_{xx} + u_{yy}); \tag{2}$$

(3) Y-momentum equation:

$$v_t + uv_x + vv_y = -p_y + \frac{1}{Re}(v_{xx} + v_{yy}). \tag{3}$$

Here $Re$ represents the Reynolds number, $p$ is the pressure variable, and $u$ and $v$ are variables for the horizontal and vertical velocity, respectively.

A time marching procedure is assumed here. The idea in the PISO (Pressure-Implicit with Splitting of Operators) method (Issa, 1985) is to replace the continuity equation with a steady state pressure Poisson equation and to solve the following system:

$$\frac{u_x^n + v_y^n}{h_t} + (\frac{1}{Re}(u_{xx} + u_{yy}) - uu_x - vu_y)_x$$

$$+ (\frac{1}{Re}(v_{xx} + v_{yy}) - uv_x - vv_y)_y = p_{xx} + p_{yy}, \tag{4}$$

$$\frac{u^{n+1} - u^n}{h_t} + uu_x + vu_y = -p_x + \frac{1}{Re}(u_{xx} + u_{yy}), \tag{5}$$

$$\frac{v^{n+1} - v^n}{h_t} + uv_x + vv_y = -p_y + \frac{1}{Re}(v_{xx} + v_{yy}). \tag{6}$$

For simplicity, we have reserved the spatial derivative terms without expansion into differences. We note that in our work, a hybrid of upwind and central differences is used for the spatial derivatives to yield a nonlinear discrete algebraic system.

Note that the system consisting of Eqs. (1), (5) and (6), or of Eqs. (4) – (6), can be recast in a form of properly defined numerical fluxes:

$$U_t + F_x + G_y = Q. \tag{7}$$

To double-check our numerical observation, we solved both the coupled system, Eqs. (1), (5) and (6), and also solved in a decoupled manner (PISO) the system of Eqs. (4) – (6), and compared the results.

In addition to the NS equations, we tested the scalar nonlinear Burgers equation, here with $Q = 0$, and with

$$F = \frac{U^2}{2} - \frac{1}{Re}U_x,$$

$$G = \frac{U^2}{2} - \frac{1}{Re}U_y,$$

in the above equation.

## 2. The Finite-Volume-Difference Method

The finite volume approach is adopted via the integral conservation form. We therefore rewrite the above system in its associated integral form. Following our previous works (Lee and Chen, 1998; Lee and Yu, 1999), boundary-fitted cell-center type finite volumes with collocated grids are assumed for the purpose of geometry discretization. When resolving the temporal derivative term by means of the first order backward difference, we obtain

$$\iint_{\Omega_{i,j}} \frac{U^{n+1} - U^n}{h_t} dxdy + \iint_{\Omega_{i,j}} F_x dxdy$$

$$+ \iint_{\Omega_{i,j}} G_y dxdy = \iint_{\Omega_{i,j}} Q dxdy \tag{8}$$

with $\Omega_{i,j}$ being a local cell.

All the definite integrals are discretized as weighted averages involving the primitive (and the flux) variables at neighboring cells. The fourth-order scheme (Lee, 1999; Jea *et al.*, 1999; Lee and Yu, 1997) is adopted to achieve higher order accuracy, therefore allowing for the possibility of a larger time step.

We note that a time step is usually selected based on

the Courant-Friedrichs-Lewy (CFL) conditions. However, as we are using an implicit scheme, much larger time steps (Table 1) can be considered, thus providing great savings of resources.

We note that the discrete system of algebraic equations can be the one derived using either Eqs. (1), (5) and (6), or Eqs. (4) – (6).

FV discretization of the geometry and FV-FD (Finite-Volume-Finite-Difference) discretization of the differential equations can, thus, lead to serial computation, and also are useful for solving subproblems in subdomains through parallel computation as discussed in the next section.

# III. Domain-based Newton-Schwarz Method

The DD method is a technique for solving partial differential equations based on decomposition of the spatial domain into several subdomains. We adopt in this paper the nonlinear approach to solving the nonlinear algebraic system resulting from finite-volume-difference discretization of partial differential equations. We solve the steady state discrete nonlinear system by using an approximate matrix-free Newton's method, the Newton-GMRes (Newton Generalized Minimal Residual) (Brown and Saad, 1990). We note that overlapping nonlinear DD methods which follow this approach, together with some hybrid variants, were studied by the first author in Lee and Chen (1998), and that a defect correction approach was discussed by Cai *et al.* (1996).

## 1. The Global Discrete Problem

We will point out a key part of the setup here. In view of the discrete nonlinear algebraic system, we are concerned with the nonlinear (nonoverlapping) block Jacobi iteration and many preconditioned versions of this method. However, in the context of continuous geometry of this problem, this actually corresponds to overlapping subdomains because our setup assumes Dirichlet type interior boundary conditions. Therefore, in our setup, an interface-preconditioned Newton-Schwarz method actually corresponds to a nonlinear analogy of a combination of a Schwarz type and a Schur-complement type linear DD method.

## 2. Block Newton-Jacobi Method

The setup assumes that the discrete global nonlinear system is decomposed into partition of (nonoverlapping) blocks of equations. Parallel computation corresponding to the nonlinear block Jacobi method can thus be carried out. This is the basic Parallel Newton-Jacobi (PNJ) method studied and compared later with preconditioned methods.

We will next outline the PNJ procedure with convergence checking on the global domain, with an unknown vector $\boldsymbol{X}$ for a steady state problem:

**Procedure PNJ.**
Do while $\{|\boldsymbol{X}^{new} - \boldsymbol{X}^{old}| \geq global - tolerance\}$
  Step 1: Solve subproblem $F(x) = 0$ at each subdomain in parallel.
  Step 2: Carry out the Boundary Condition-Update procedure for all subproblems.
End Do

The above pseudo code, although sketchy, is quite general. For the current simple block Jacobi method, Step 2 actually becomes

  Step 2: *Each worker cpu receives the updated interior boundary variables from neighbors.*

We note that Step 2, as stated above, can be proceeded with processing of one or several preconditioners, as we will explain in the rest of this section. We will see that Step 2 can split up, as a generalization, into the solving of one (or several) preconditioner(s) and the Boundary-Condition-Update procedure for subsequent subproblems.

We note that from a purely algebraic point of view, the PNJ procedure is simply a nonoverlapping block Jacobi method applied to the global nonlinear discrete algebraic system.

## 3. Interface Preconditioner

We propose below an interface preconditioner for solving, on the interface $B$, the interface problem prior to each nonlinear block Jacobi iteration. We note that in our actual implementation, the interface preconditioner is squeezed into positions after and before two consecutive block Jacobi iterations. We will use the same notations for the set of continuous variables and for that of discrete variables. We describe below the Interface-Preconditioned Parallel Newton-Jacobi (IPPNJ) method:

**Procedure IPPNJ.**
Do while $\{|\boldsymbol{X}^{new} - \boldsymbol{X}^{old}| \geq global - tolerance\}$
  Step 1: Solve the discrete algebraic nonlinear system $F_i(x) = 0$, $x \in \Omega_i$ for all subdomain problems in parallel.
  Step 2a: Set up the interface problem, with information communicated among subdomains.
  Step 2b: Solve the nonlinear system for the interface problem

$$F_B(x) = 0,\ x \in B.$$

  Step 2c: Update via communication the interior boundary conditions for all the subproblems with the interface variables just solved.
End Do

We note that the interface problem is relatively small

in size and easier to solve using a Newton-like method, and that the solution should offer more accurate interior boundary conditions at the interface variables in a more efficient way. This is the goal of our preconditioned parallel nonlinear procedure. That is, acceleration on (only) the interface variables yields a preconditioner for subsequent DD iterations. We are, therefore, led naturally to the next topic.

## 4. Over-Relaxed Interface Preconditioner

Interface preconditioners are proposed here for solving the interface problem prior to each nonlinear block Jacobi operation. We can as well apply the idea of Successive Over-Relaxation (SOR) to the computed solution for these interface variables. Relaxation can be applied to all the subproblems and to the interface problem. Furthermore, the setup for the interface problem can be accelerated on the interior boundary conditions. Investigations on various relaxation parameters and combinations of mixing strategies are conducted here based on the basic Newton-SOR-Schwarz method and over-relaxed interface preconditioner. The observed numerics and dynamics indicate that the hybrid performs better than either parent method in terms of convergence and also accuracy. It is seen that this is true in both three-dimensional and two-dimensional vortex flows, and in three-dimensional traveling waves. The experimental results presented later will be useful in more general practical applications.

Among the various options mentioned above, we will describe in some detail only the case of an interface preconditioner set up by means of over-relaxation:

**Procedure PORIP (Parallel Over-Relaxed Interface Preconditioner).**
Do while $\{\left|\boldsymbol{X}^{new} - \boldsymbol{X}^{old}\right| \geq global - tolerance\}$
- Step 1: Solve discrete algebraic nonlinear system $F_i(x) = 0, x \in \Omega_i$ for all subdomain problems in parallel.
- Step 2a: Set up the interface problem, via information communicated from each subdomain, with relaxation of the interior boundary condition and optionally also of the interface variables.
- Step 2b: Solve the nonlinear system for the interface problem

$$F_B(x) = 0,\ x \in B.$$

- Step 2c: Update via communication the interior boundary conditions for all subproblems by the interface variables just solved, with an optional relaxation type acceleration.

End Do

## 5. Preconditioned Block Newton-Schwarz Procedure

For general application, we recast the discrelized nonlinear algebraic system of equations as

$$\Phi(u) = rhs,$$

where $\Phi = (\Phi_1, \ldots, \Phi_{n_d})^T$, $u = (u_1, \ldots, u_{n_d})^T$, with $u_i \in R^{d_s} \equiv \boldsymbol{X}_s$. Here, $s$ denotes a subdomain (and a subproblem), and $d_s$ denotes the dimension of the subproblem (on subdomains). In the case of our boundary-fitted cell-center finite-volume-difference setup, $d_s$ equals the product of the degrees of freedom at a node and the number of (internal) grids in each subdomain. We assume equal size subproblems for simplicity. The space $\boldsymbol{X}_s$ is, therefore, where a solution to the discrete subproblem resides. Consider the subproblems

$$\tilde{\Phi}_i(\tilde{u}_i) = \tilde{rhs}_i$$

and

$$J(\Phi_i, u_i) = \frac{\partial \Phi_i}{\partial u_i}.$$

We assume regularity of this portion of the global Jacobian.

With the notations introduced, we will describe in some detail the Preconditioned Parallel Newton-Jacobi (PPNJ) procedure, based on partitions of nonoverlapping subdomains and a specific order of the equations and variables:

**Procedure PPNJ.**
Do while (global convergence is achieved or the maximum number of DD iterations is exceeded).

(1) Do $i = 1, \ldots, n_d$ (in parallel)
- (i) set $\tilde{u}_i$ by means of $u$ and canonical projection;
- (ii) set $\tilde{rhs}_i$;
- (iii) obtain an approximated solution to

$$\tilde{\Phi}_i(\tilde{u}_i) = \tilde{rhs}_i;$$

- (iv) evaluate for local convergence the residual

$$\tilde{r}_i = \tilde{rhs}_i - \tilde{\Phi}_i(\tilde{u}_i);$$

- (v) evaluate for local convergence the difference

$$diff_i = \left\| \tilde{u}_i - \tilde{u}_{i\,sav} \right\|.$$

(2) Update global $u$ by communicating $\tilde{u}_i$ among relevant processors.

(3) Check global convergence by evaluating

$$max_i diff_i \text{ or } \|\Phi(u)\| \text{ on } \boldsymbol{X}^{n_d}_{n_s}.$$

(4) If global convergence is satisfied, then break. Otherwise
- (i) Do an optional accelerator or preconditioner, such

**Table 1.** Parameters of Test Runs with Supnorm

| case | eq. | Re | DD_iter | ht | ht_CFL | nx, ny, nz | nsub | PISO | method |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2D Burgers | 1 | 100 | 1e-3 | 9.00e-06 | 60, 60 | 9 | 0 | various |
| 2 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 4 | 0, 1 | PNJ |
| 3 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 4 | 0, 1 | IPPNJ |
| 4 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 9 | 0, 1 | PNJ |
| 5 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 9 | 0, 1 | IPPNJ |
| 6 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 4 | 0 | various |
| 7 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 9 | 0 | various |
| 8 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 4 | 1 | various |
| 9 | 2D NS | 10 | 20 | 1e-3 | 5.62e-06 | 240, 240 | 9 | 1 | various |
| 10 | 3D NS | 10 | 20 | 1e-3 | 2.31e-04 | 30, 30, 30 | 8 | 0 | PORIP |
| 11 | 3D NS | 10 | 50 | 1e-3 | 2.31e-04 | 30, 30, 30 | 8 | 0 | PORIP |
| 12 | 3D Burgers | 10 | 20 | 1e-3 | 2.31e-04 | 30, 30, 30 | 8 | 0 | PORIP |
| 13 | 2D NS | 10 | 30 | 1e-3 | 1.13e-05 | 120, 120 | 9 | 0 | PORIP |

as an interface preconditioner or global coarse level preconditioner.

(ii) Update the interface variables through communication and approximation schemes in the case of a two level preconditioner.

(5) Update the DD iteration counter.

End Do

This specializes to a more detailed version of our previous method provided that an interface problem is solved as an interface preconditioner in Step 4(i) above.

### 6. Other Preconditioners

We note that the interface problem can be solved optionally with a fine-level interface preconditioner. This is usually affordable since the two level setup for the interface problem is still comparatively cheaper than the other subproblems, in terms of both storage and computation. It is, therefore, a natural iterative refinement procedure based on consideration of the load balance in some applications.

We note that a global coarse level preconditioner is also a good choice because it hopefully produces global information update, as motivated by the linear DD theory. However, we will not discuss these details here but will examine the test results in next section.

## IV. Numerical Results and Discussions

All the cases were tested on a PC cluster with 16 Intel Pentium II 400 MHz cpus, 256 MB RAM and the Linux operating system with MPI parallel environment at the National Center for High-Performance Computing (NCHC). The parameters and methods for all the test cases are listed in Table 1. The physical domain of $x$, $y$ and $z$ was (0.0, 1.0) in cases 1, 10, 11 and 12; the physical domain of $x$ and $y$ was (1.0, 1.66) in cases 2, 3, 6 and 8; (1.0, 2.0) in cases 4, 5, 7, 9 and 13. Some statistics concerning the cpu time are shown in Table 2. Accuracy and convergence results are shown in Figs. 1–30. We explain these results in some detail below.

**Case 1.** A very severe stopping criterion was applied here for solving Burgers' equation. Convergence up to nine digits was enforced. The maximum number of DD-iterations, whether needed or not, was set to be as large as 100, just to examine the stability in a long run. However, in practical time marching, this may be as small as five or twenty.

Four DD methods were tested here: PNJ, IPPNJ, the Coarse-level-Preconditioned Parallel Newton-Jacobi (CPPNJ), and the Fine-level Interface-Preconditioned Parallel Newton-Jacobi (FIPPNJ). Shown in Figs. 1 and 2 are the accuracy and convergence results in relative sense. These numerics indicate that all four DD methods yielded very stable discrete dynamics. CPPNJ converged fastest while the basic PNJ was the slowest. The other two, IPPNJ and FIPPNJ, converged at about the same rate, with the latter slightly faster as can be seen from the produced output.

As far as accuracy is concerned, the coarse preconditioner was the least accurate, in contrast to its fastest convergence. The basic interface preconditioner IPPNJ could achieve the same level of accuracy, both in absolute and relative terms, and took only about half the number of iterations as did the basic PNJ method. This is similar to the comparison between the classical point Gauss-Seidel and the point Jacobi iterations for solving a linear system.

We note that the precision achieved with the two-level preconditioners, CPPNJ and FIPPNJ, clearly depended on local interpolation or approximation and, therefore, was restricted by the spatial grid resolution. Noted here is the fact that the FIPPNJ preconditioner saturated at an earlier stage in absolute convergence and, therefore, was forced, unnecessarily, to iterate up to the required 100 maximum DD iterations. This hurt in every way the cpu-time result (Table 2). However,

**Table 2.** CPU Time in Cases 1 – 9

| | case | iter | average commu. | total commu. | averate sub. solver | total sub. solver | other overhead | total cpu time |
|---|---|---|---|---|---|---|---|---|
| 1 | PNJ | 81 | 5.00e+00 | 4.05e+02 | 5.58e+00 | 4.52e+02 | 9.00e+00 | 8.66e+02 |
| | IPPNJ | 38 | 6.39e+00 | 2.43e+02 | 5.45e+00 | 2.07e+02 | 3.00e+00 | 4.53e+02 |
| | CPPNJ | 11 | 9.90e+00 | 1.01e+02 | 6.33e+00 | 6.96e+01 | 2.40e+00 | 1.81e+02 |
| | FIPPNJ | 100 | 1.88e+01 | 1.88e+03 | 4.34e+00 | 4.34e+02 | 6.00e+00 | 2.32e+03 |
| 2 | PISO 0 | 20 | 6.40e+01 | 1.28e+03 | 7.05e+01 | 1.41e+03 | 3.00e+01 | 2.72e+03 |
| | PISO 1 | 20 | 1.00e+02 | 2.00e+03 | 2.95e+02 | 5.90e+03 | 7.00e+01 | 7.96e+03 |
| 3 | PISO 0 | 20 | 5.50e+01 | 1.10e+03 | 6.35e+01 | 1.27e+03 | 5.00e+01 | 2.42e+03 |
| | PISO 1 | 20 | 1.64e+02 | 3.28e+03 | 5.40e+02 | 1.08e+04 | 3.00e+01 | 1.41e+04 |
| 4 | PISO 0 | 20 | 1.45e+02 | 2.90e+03 | 1.64e+02 | 3.28e+03 | 8.00e+01 | 6.26e+03 |
| | PISO 1 | 20 | 1.71e+02 | 3.42e+03 | 6.70e+02 | 1.34e+04 | 8.00e+01 | 1.69e+04 |
| 5 | PISO 0 | 20 | 1.25e+02 | 2.50e+03 | 1.39e+02 | 2.78e+03 | 8.00e+01 | 5.36e+03 |
| | PISO 1 | 20 | 2.28e+02 | 4.56e+03 | 1.19e+03 | 1.38e+04 | 1.50e+02 | 2.84e+04 |
| 6 | PNJ | 20 | 6.21e+01 | 1.24e+03 | 6.80e+01 | 1.36e+03 | 4.00e+01 | 2.64e+03 |
| | IPPNJ | 20 | 5.40e+01 | 1.08e+03 | 6.15e+01 | 1.23e+03 | 4.00e+01 | 2.35e+03 |
| | CPPNJ | 20 | 1.75e+02 | 3.50e+03 | 7.35e+01 | 1.47e+03 | 5.00e+01 | 5.02e+03 |
| 7 | PNJ | 20 | 1.43e+02 | 2.86e+03 | 1.61e+02 | 3.22e+03 | 9.00e+01 | 6.15e+03 |
| | IPPNJ | 20 | 1.22e+02 | 2.44e+03 | 1.37e+02 | 2.74e+03 | 9.00e+01 | 5.25e+03 |
| | CPPNJ | 20 | 6.15e+02 | 1.23e+04 | 1.74e+02 | 3.48e+03 | 1.20e+02 | 1.59e+04 |
| 8 | PNJ | 20 | 1.00e+02 | 2.00e+03 | 2.95e+02 | 5.90e+03 | 7.00e+01 | 7.97e+03 |
| | IPPNJ | 20 | 9.55e+01 | 1.91e+03 | 2.87e+02 | 5.74e+03 | 7.00e+01 | 7.71e+03 |
| 9 | PNJ | 20 | 1.67e+02 | 3.34e+03 | 6.70e+02 | 1.34e+04 | 1.10e+02 | 1.69e+04 |
| | IPPNJ | 20 | 7.60e+02 | 1.52e+04 | 7.20e+02 | 1.44e+04 | 1.00e+02 | 2.97e+04 |

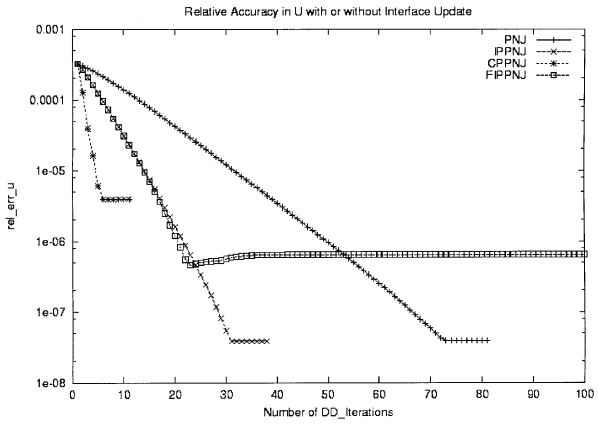


**Fig. 1.** Relative accuracy in solving Burgers' equation in Case 1.

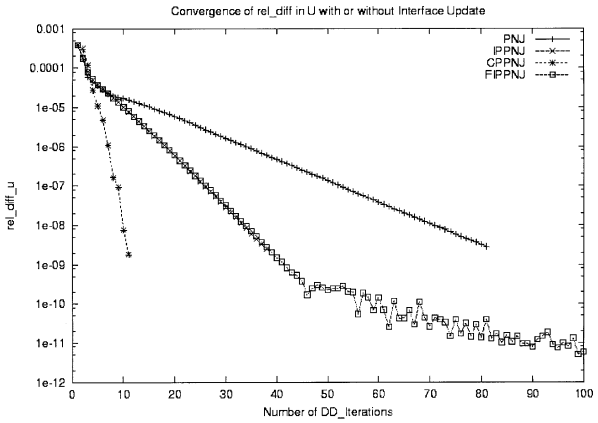


**Fig. 2.** Relative convergence in solving Burgers' equation in Case 1.

the relative convergence history (Fig. 2) does verify the stability in computation. In any case, the FIPPNJ result seems to be poorer than that of IPPNJ. Therefore FIPPNJ was excluded in test Cases 2 – 9 described below, which are concerned with the NS equations.

**Case 2.** We validated the basic Newton-Jacobi method (PNJ) applied to both the decoupled and the coupled approach, i.e., with or without PISO. Then we tested in Cases 6 – 9 our proposed preconditioners using these two different approaches.

The absolute accuracy of the variables $p$ and $v$ is shown in Figs. 3 and 4, respectively. Validation of our algorithms and implementation, with or without PISO, is provided by the numerics here. The accuracy for the case without PISO is only slightly inferior to the case with PISO.

**Case 3.** The basic interface-preconditioner (IPPNJ) was applied here, similar to Case 2, using the two approaches with

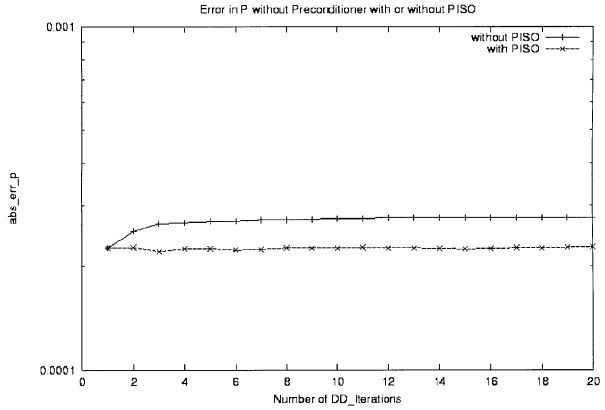


**Fig. 3.** Accuracy in $p$ with or without PISO in Case 2.

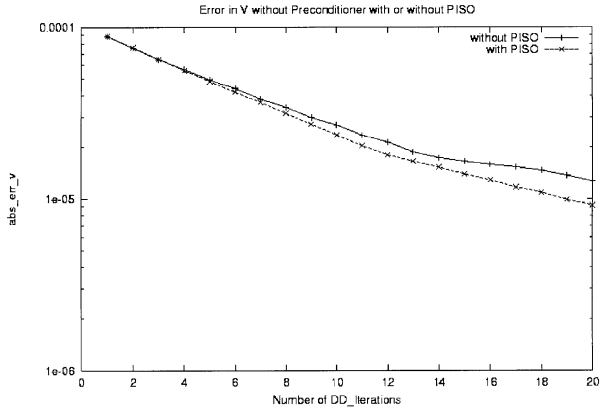


**Fig. 4.** Accuracy in $v$ with or without PISO in Case 2.

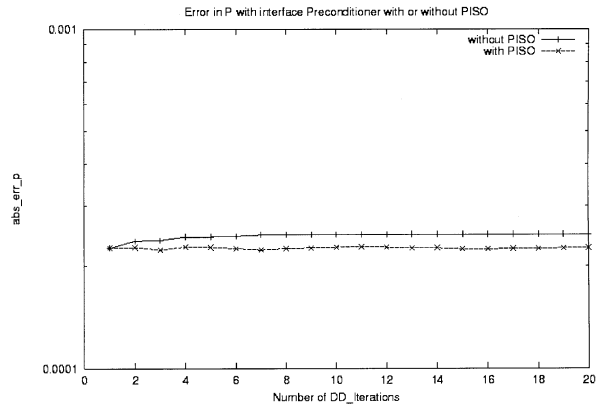


**Fig. 5.** Accuracy in $p$ with or without PISO in Case 3.

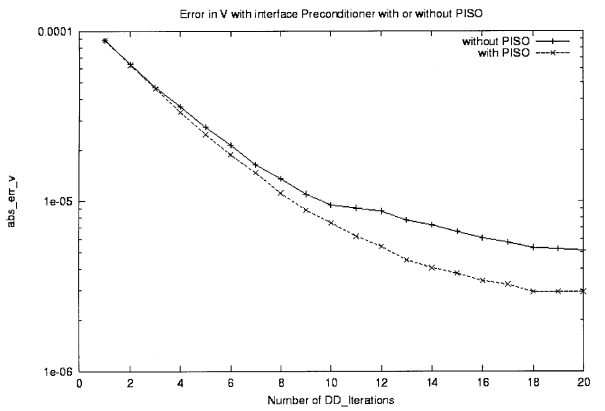


**Fig. 6.** Accuracy in $v$ with or without PISO in Case 3.

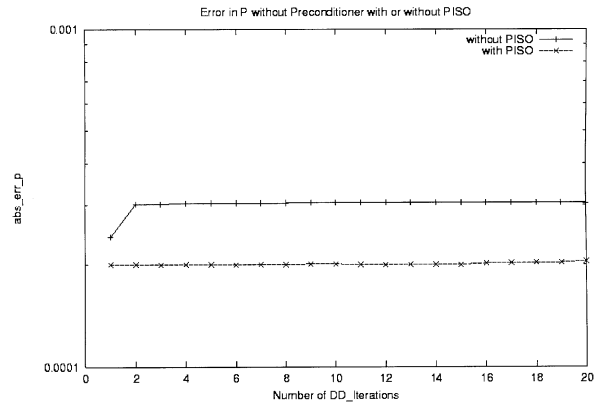


**Fig. 7.** Accuracy in $p$ with or without PISO in Case 4.

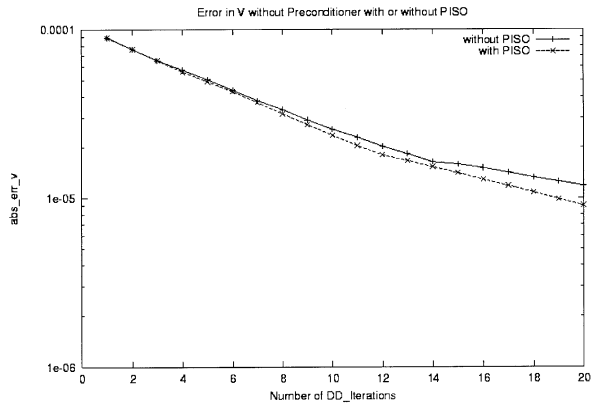


**Fig. 8.** Accuracy in $v$ with or without PISO in Case 4.

or without PISO. The findings, shown in Figs. 5 and 6, are similar to those for Case 2.

**Cases 4** and **5.** These are similar to Cases 2 and 3. Nine subdomains were taken here, instead of four subdomains as in Cases 2 and 3. We note that the spatial grid resolutions are the same as those for Cases $2-5$. The results, shown in Figs. $7-10$, are similar to those for Case 2.

**Cases 6** and **7.** The coupled system, without PISO, was solved

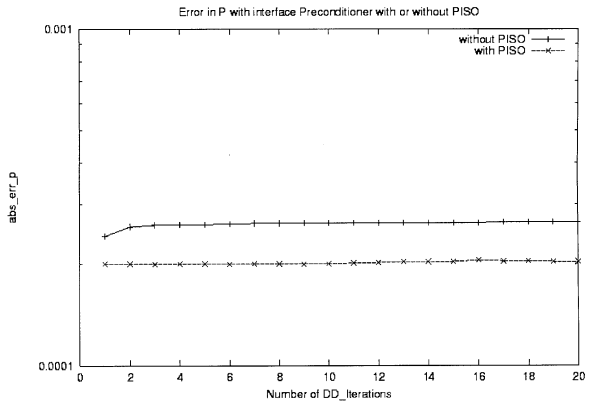


**Fig. 9.** Accuracy in $p$ with or without PISO in Case 5.

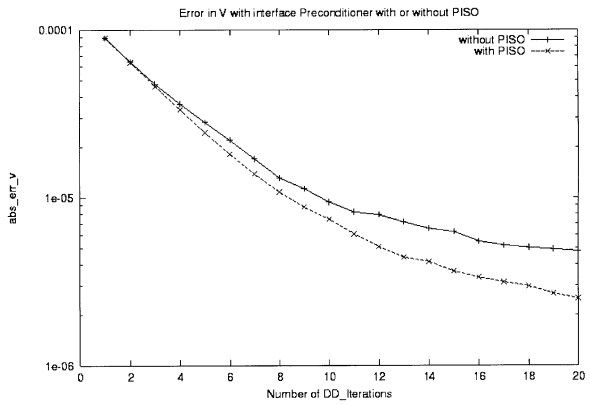


**Fig. 10.** Accuracy in $v$ with or without PISO in Case 5.

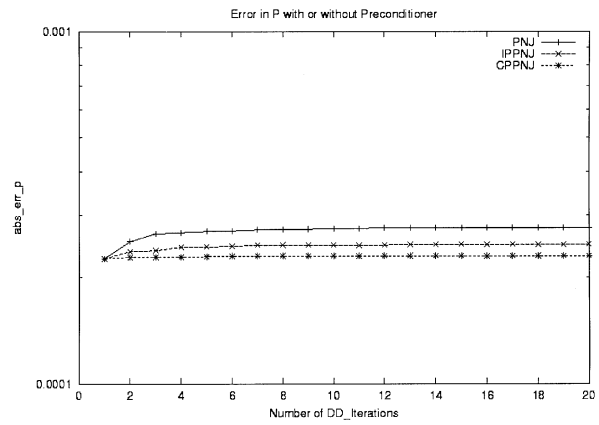


**Fig. 11.** Accuracy in $p$ with or without preconditioner in Case 6.

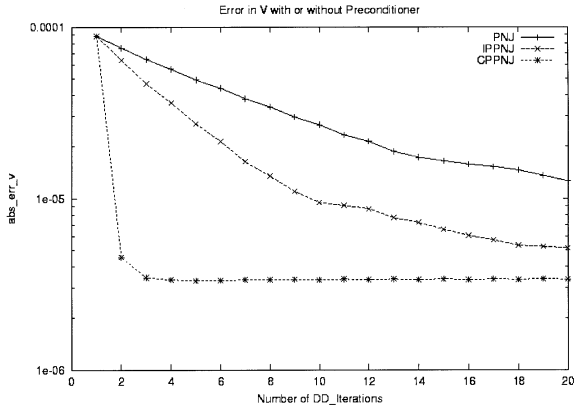


**Fig. 12.** Accuracy in $v$ with or without preconditioner in Case 6.

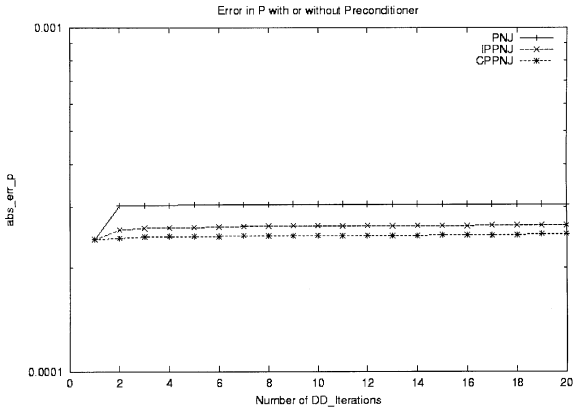


**Fig. 13.** Accuracy in $p$ with or without preconditioner in Case 7.

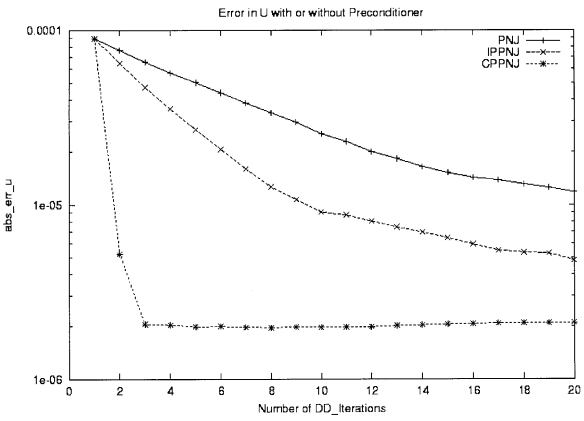


**Fig. 14.** Accuracy in $u$ with or without preconditioner in Case 7.

here, with an interface preconditioner (IPPNJ), with a coarse level preconditioner (CPPNJ), or without a preconditioner (PNJ). The partition of the global region consists of four subdomains in Case 6 and nine in Case 7. It is clearly seen, in Figs. 11 and 12, that CPPNJ was most accurate, and that IPPNJ was also more accurate than PNJ. Although not as smooth as in the case of the normalized two-norm, the numerics for the maximum norm shown here do indicate the relative spirits in these DD methods. Results for the case of nine subdomains are similar, as shown in Figs. 13 – 15.

We note that very heavy communication was observed

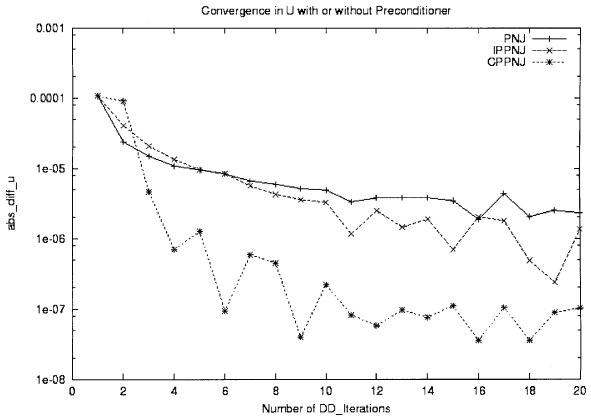


**Fig. 15.** Convergence in $u$ with or without preconditioner in Case 7.

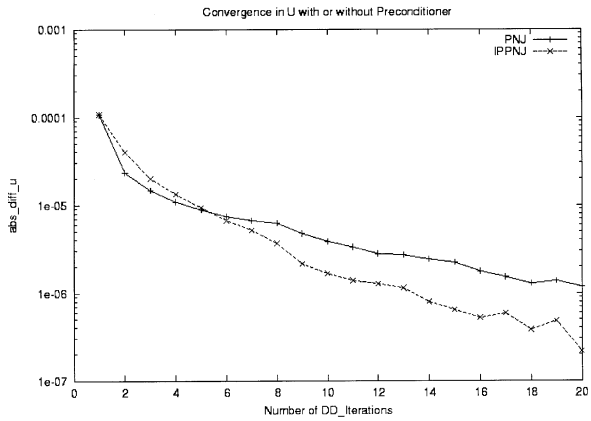


**Fig. 18.** Convergence in $u$ with or without preconditioner in Case 8.

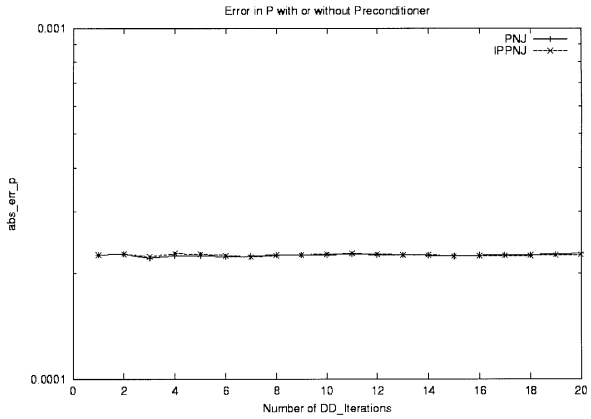


**Fig. 16.** Accuracy in $p$ with or without preconditioner in Case 8.

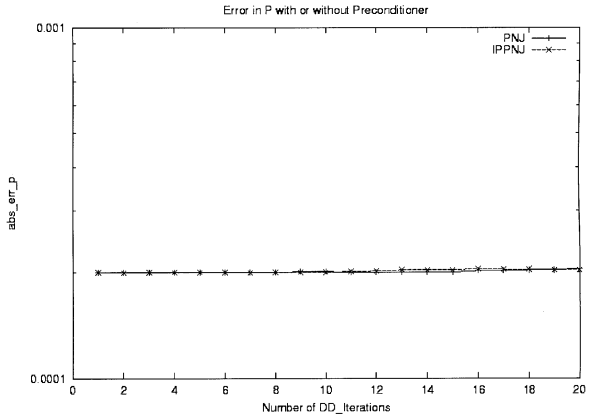


**Fig. 19.** Accuracy in $p$ with or without preconditioner in Case 9.

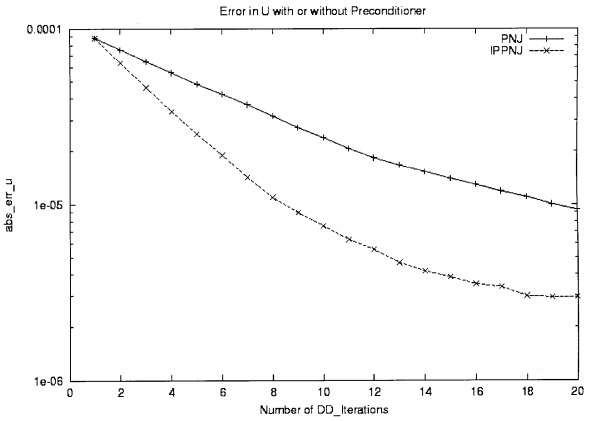


**Fig. 17.** Accuracy in $u$ with or without preconditioner in Case 8.

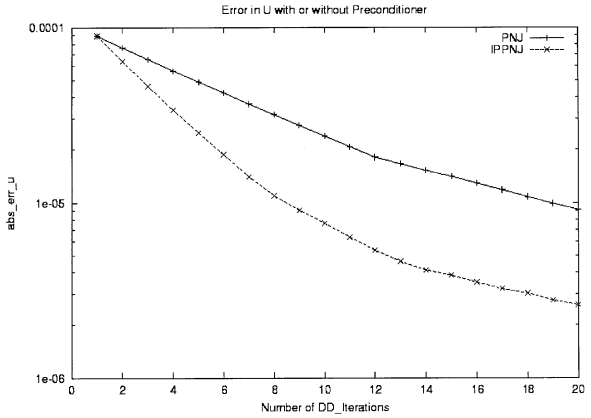


**Fig. 20.** Accuracy in $u$ with or without preconditioner in Case 9.

in the coarse preconditioner in the pre- and post-data processing of solving the global coarse problem. Therefore, we omitted CPPNJ from the subsequent test cases.

**Cases 8** and **9.** Here we solved the discrete system, via PISO, and compared the PNJ and IPPNJ methods. There were four subdomains for Case 8 and nine for Case 9. The interface preconditioner is seen (Figs. 16 – 21) to converge faster and to be more accurate.

The remaining Cases, 10 – 13, are related to relaxation technology with the parameters shown in Table 1. Detailed

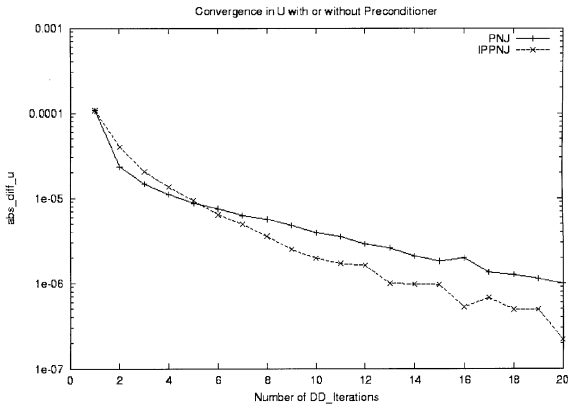


**Fig. 21.** Convergence in $u$ with or without preconditioner in Case 9.

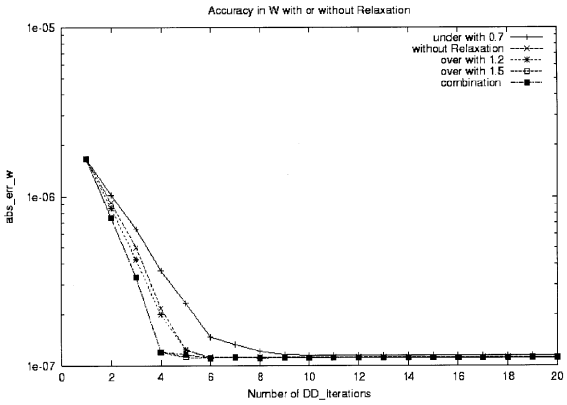


**Fig. 24.** Accuracy in $w$ with or without relaxation in Case 10.

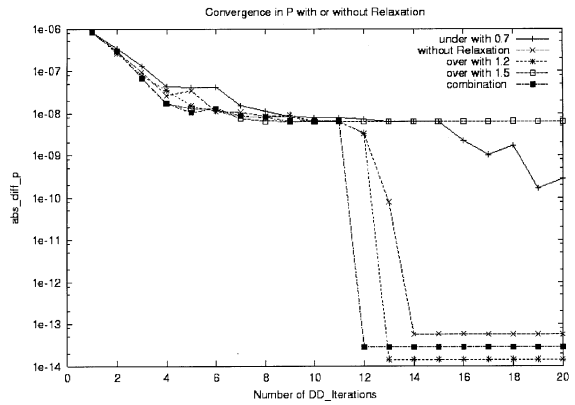


**Fig. 22.** Convergence in $p$ with or without relaxation in Case 10.

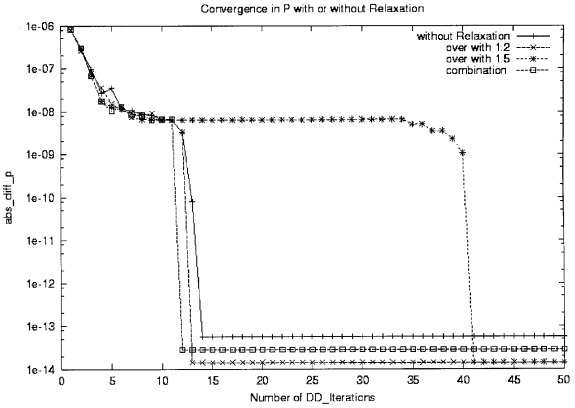


**Fig. 25.** Convergence in $p$ with or without relaxation in Case 11.

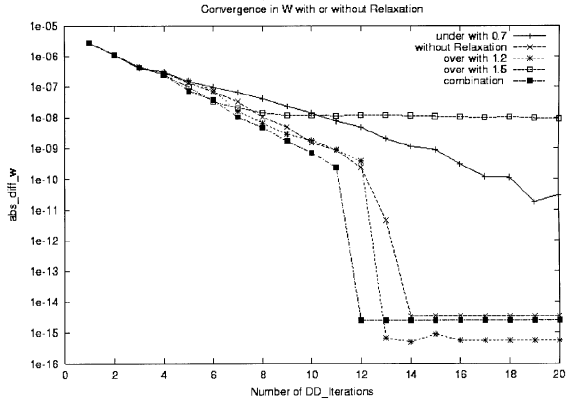


**Fig. 23.** Convergence in $w$ with or without relaxation in Case 10.

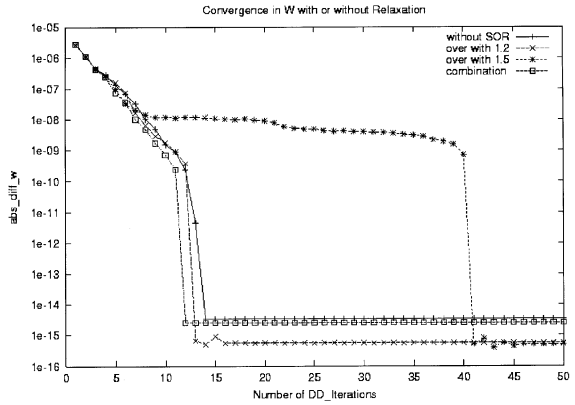


**Fig. 26.** Convergence in $w$ with or without relaxation in Case 11.

description follows.

**Case 10.** We conducted tests here with under-, over- and no relaxation. The observation then lead us naturally to a hybrid. Precisely, the relaxation parameter took the values 0.7, 1.0, 1.2 and 1.5. Convergence of the pressure variables and $z$-component velocity $W$ is shown in Figs. 22 and 23, respectively. The exact accuracy achieved is shown in Fig. 24 for $W$. These three figures show the results up to 20 subdomain level iterations. The numerical results indicate a monotonic relationship between the accuracy and the relaxation parameter $\alpha$. The $\alpha = 1.5$ test run was the most accurate one, and under-relaxation ($\alpha = 0.7$) was the worst one. As far as convergence is concerned, the test runs with no relaxation ($\alpha = 1.0$) and

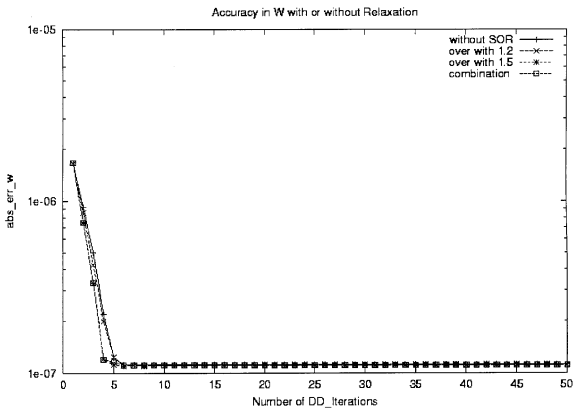


**Fig. 27.** Accuracy in $w$ with or without relaxation in Case 11.

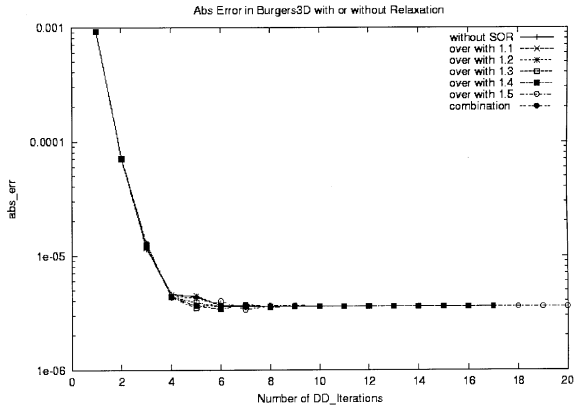


**Fig. 29.** Absolute error in solving the 3D Burgers' equation with or without relaxation in Case 12.

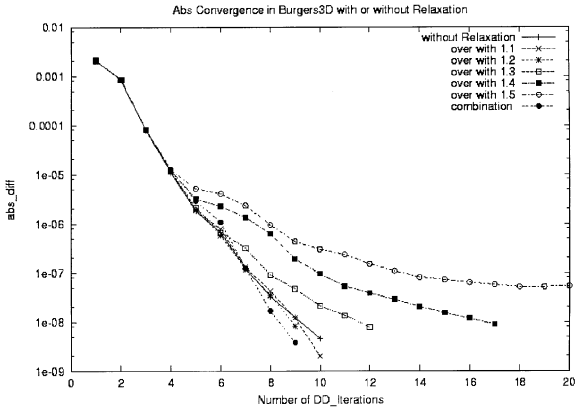


**Fig. 28.** Absolute convergence in solving the 3D Burgers' equation with or without relaxation in Case 12.

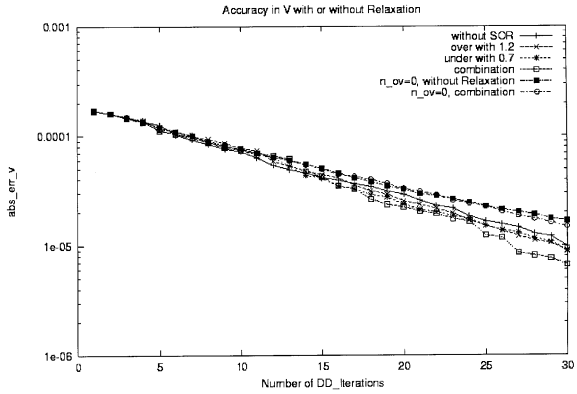


**Fig. 30.** Absolute error in $v$ in solving the 2D NS equation with or without relaxation in Case 13.

low relaxation ($\alpha = 1.2$) were the two winners. However, notice that the $\alpha = 1.5$ case converged fastest at the second DD iteration, right after the over-relaxation applied once.

This suggests a combination of relaxation with different relaxation parameters. We are, thus, led to a combination strategy, which includes three times of over-relaxation with $\alpha = 1.5$ and then no relaxation ($\alpha = 1.0$) afterwards. It is amazing to see that the hybrid beated both of the parent methods ($\alpha = 1.5$ and $\alpha = 1.0$), not only in convergence, but also in accuracy. Actually, the combination strategy won the accuracy and 6-digit convergence contest at DD-iteration 11.

**Case 11.** The test runs for Case 11 extended up to DD-iteration 50. The results for convergence (Figs. 25 and 26) and accuracy (Fig. 27) indicate the amazing stability of the algorithms, the software implementation, and the observed dynamics.

**Case 12.** With the three dimensional nonlinear Burgers equation, seven test runs were conducted with relaxation parameters $\alpha$ = 1.0, 1.1, 1.2, 1.3, 1.4 and 1.5, and with the hybrid specified in Case 10. Results for convergence are shown in Fig. 28, and in Fig. 29 for accuracy. Again, we observe that both the convergence and the accuracy results exhibit monotonicity with respect to the relaxation parameter, and, a pleasant surprise, that the hybrid outperformed all the cases in terms of both convergence and accuracy. The physics of the convection-diffusion type flow are certainly different from those in the previous case of a vortex. It is amazing, therefore, to observe the very similar phenomenon in both.

**Case 13.** The effect of a relaxed interface preconditioner was examined here. Two dimensional NS equations were taken. Both the interface-preconditioned Newton-Jacobi method, relaxed or not, and the Newton-SOR-Schwarz method, as in Cases 10 – 12, were tested. The results show advantages of the preconditioned class and also advantages of the subclass of a combined mixed relaxation strategy. Only accuracy of $V$ is presented as Fig. 30.

One point should be noted about the performance. Over-relaxation and our hybrid trials required no extra communi-

cation at all and cost only one additional vector update in terms of computation.

## V. Conclusion

We have designed several natural candidates for preconditioners of the interface (sub)problem in the Newton-Schwarz approach. Compared with the basic parallel nonlinear block Jacobi procedure, these preconditioners are more accurate for the Burgers and NS equations. The choice of preconditioner certainly depends on, among others factors, the spatial resolution and required precision of computed results. Based on our experiments, the IPPNJ method converges faster while achieving the same or better accuracy and requiring less computation time. The other preconditioners, CPPNJ and FIPPNJ, may achieve moderate precision and converge faster in terms of the number of DD iterations, but both apparently require heavier communication. We believe that further technology improvement in system architecture will resolve this communication inefficiency to a large extent. We also investigated the SOR type domain decomposition preconditioner, which exhibits very monotonic behavior both in convergence and in accuracy. A combination strategy showed very good results, better than those of both parent methods. Our implementation yielded very stable numerics and dynamics in terms of long time DD-iterations. These are observed on flow of either a wave type or a vortex type.

### Acknowledgment

This research work was funded by the National Science Council, R.O.C., under contract NSC 88-2113-E321-003. The idea was developed on a SUN workstation cluster provided by CERFACS, France, and finalized and tested on a dedicated PC cluster at the National Center for High-Performance Computing, R.O.C. The authors received very helpful suggestions and comments in valuable discussions with Professor Iain S. Duff and Mr. Luc Giraud at CERFACS. Very sincere gratitude is expressed here.

### References

Bjørstad, P. E., M. S. Espedal, and D. E. Keyes, Eds. (1997) *Proceedings of the 9th International Conference on Domain Decomposition Methods*, Ullensvang, Norway.

Brown, P. N. and Y. Saad (1990) Hybrid krylov methods for nonlinear system of equations. *SIAM J. Sci. Stat. Comput.*, **11**, 450-481.

Cai, X. C., W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young (1996) Newton-Krylov-Schwarz algorithms for the 2D full potential equation. *Proceeding of the Seventh Copper Mountain Conference on Iterative Methods*, Colorado, CO, U.S.A.

Chan, T., R. Glowinski, J. Périaux, and O. Widlund, Eds. (1989) *Proceedings of the Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, PA, U.S.A.

Chan, T., R. Glowinski, J. Périaux, and O. Widlund, Eds. (1990) *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, U.S.A.

Ethier, C. R. and D. A. Steinman (1994) Exact fully 3D Navier-Stokes solutions for benchmarking. *International Journal for Numerical Methods in Fluids*, **19**, 369-375.

Glowinski, R., G. H. Golub, G. A. Meurant, and J. Périaux, Eds. (1988) *Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, U.S.A.

Glowinski, R., Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, Eds. (1991) *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, U.S.A.

Glowinski, R., J. Périaux, and Z. Shi, Eds. (1997) *Proceedings of Eighth International Conference on Domain Decomposition Methods*, Beijing, P.R.C.

Gropp, W. D., D. E. Keyes, L. C. McInnes, and M. D. Tidriri (1998) *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel Implicit CFD*. NASA/CR-1998-208435, ICASE, Hampton, VA, U.S.A.

Issa, R. I. (1985) Solution of the implicitly discretized fluid flow equations by operator-splitting. *Journal of Computational Physics*, **62**, 40-65.

Jea, K. C., M. Yu, and D. Lee (1999) A study on finite volume methods. In: *Iterative Methods on Scientific Computation II*. IMACS, Austin, TX, U.S.A.

Keyes, D. E. and J. Wu, Eds. (1994) *Domain Decomposition Methods in Science and Engineering*. Comtemporary Mathematics, **180**, AMS, Providence, RI, U.S.A.

Keyes, D. E., T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, Eds. (1992) *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, U.S.A.

Lai, C. H., P. E. Bjørstad, M. Cross, and O. Widlund, Eds. (1998) *Proceedings of the 11th International Conference on Domain Decomposition Methods*, Greenwich, U.K.

Lee, D. (1999) A new approach to finite volume-finite difference methods. *Chinese Journal of Numerical Mathematics and Applications*, **21**(2), 96-102.

Lee, D. and C. H. Chen (1998) A study on domain based parallel flow computation. *Proceeding of the Fifth National Conference in Computational Fluid Dynamics*, Lungtan, Taoyuan, Taiwan, R.O.C.

Lee, D. and M. Yu (1997) A functional approach to finite volume-finite difference method. *Fu Jen Studies-Science and Engineering*, **31**, 89-101.

Lee, D. and M. Yu (1999) A study on parallel flow computation by Newton-Schwarz method. *Proceeding of the Sixth National Conference in Computational Fluid Dynamics*, Taitung, Taiwan, R.O.C.

Mandel, J. L., Ed. (1998) *Proceedings of the tenth International Conference on Domain Decomposition Methods*, AMS, Providence, RI, U.S.A.

Morton, K. W. (1996) *Numerical Solution of Convection-Diffusion Problems*. Chapman & Hall, London, U.K.

Patankar, S. V. (1980) *Numerical Heat Transfer and Fluid Flow*. Hemisphere, Washington, D.C., U.S.A.

Quarteroni, A., Y. A. Kuznetsov, J. Périaux, and O. B. Widlund, Eds. (1994). *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*. Contemporary Mathematics, **157**, AMS, Providence, RI, U.S.A.

Tannehill, J. C., D. A. Anderson, and R. H. Pletcher (1997) *Computational Fluid Mechanics and Heat Transfer*, 2nd Ed. Taylor & Francis, Bristol, PA, U.S.A.

# 平行流力計算區域分割方法之界面預優化研究

李天佑　游輝宏

行政院國家科學委員會國家高速電腦中心計算數學小組

## 摘　要

本文設計並探討非線性區塊迭代式區域分割方法以處理界面變數問題（interface problem）。並利用所設計之各種不同的預優化子（preconditioner）針對Navier-Stokes（NS）方程組進行非結構化（unstructured）非均勻網格（non-uniform grid）流體數值模擬（numerical fluids simulation）平行計算方法之研究與探討。此外，將預優化子本身經過鬆弛(relaxation)之前處理以加速收斂，可啓發實用之混合式方法，獲致更佳效果。