INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **31**: 891–907 (1999)

FAST BOUNDARY-DOMAIN INTEGRAL ALGORITHM FOR THE COMPUTATION OF INCOMPRESSIBLE FLUID FLOW PROBLEMS

M. HRIBERŠEK* AND L. ŠKERGET

Faculty of Mechanical Engineering, University of Maribor, Smetanova ul 17, PO Box 224, SI-2000 Maribor, Slovenia

SUMMARY

The paper deals with the numerical solution of fluid dynamics using the boundary-domain integral method (BDIM). A velocity–vorticity formulation of the Navier–Stokes equations is adopted, where the kinematic equation is written in its parabolic form. Computational aspects of the numerical simulation of two-dimensional flows is described in detail. In order to lower the computational cost, the subdomain technique is applied. A preconditioned Krylov subspace method (PKSM) is used for the solution of systems of linear equations. Level-based fill-in incomplete lower upper decomposition (ILU) preconditioners are developed and their performance is examined. Scaling of stopping criteria is applied to minimize the number of iterations for the PKSM. The effectiveness of the proposed method is tested on several benchmark test problems. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: boundary element method; Navier–Stokes equations; velocity–vorticity formulation; iterative methods; preconditioning; stopping criterion

1. INTRODUCTION

Presently, boundary element methods and their variants are widely accepted for their effectiveness in the computation of potential problems (heat conduction, elasticity problems, inviscid fluid flow . . .), but severe problems are experienced when they are applied to strongly non-linear problems with prevailing domain-based effects (diffusion–convection problems). The main problems arise due to limitations concerning varying material properties, dense system and integral matrices and the computation of domain-based physical effects. Several techniques were developed to overcome these problems, including the subdomain technique, multiple reciprocity methods and others that were mainly successful in some special cases.

When dealing with the Navier–Stokes system of equations in velocity–vorticity formulation, the mentioned boundary element method (BEM) problems threaten to cancel out all the advantages of the boundary integral methods [1] for fluid flow problems. To overcome these problems a lot of work has been performed on the application of the subdomain technique and in the solution of the resulting discretized systems of equations [2], but problems are encountered in the computation of large fluid flow cases. One of the main causes is the computational scheme of flow kinematics, based on the vector potential formulation. Although

* Correspondence to: Faculty of Mechanical Engineering, University of Maribor, Smetanova ul 17, PO Box 224, SI-2000 Maribor, Slovenia.

Received September 1998
Revised November 1998

it offers an effective way of dealing with boundary conditions on the solid walls (unknown vorticity), it is very sensitive to the introduction of the subdomain technique [2] and offers very little memory saving potentials. On the other hand, vorticity and energy transport equations behave in a stable way no matter how many subdomains are used. It was therefore obvious to search for a new flow kinematics formulation, one that would capture the advantages in specifying boundary conditions and gain suitability for the use of the subdomain technique. The result is the development of a new boundary domain integral algorithm [Škerget *et al.*, 'Computational fluid dynamics by boundary-domain integral method', submitted to *Int. J. Numer. Methods Eng.* (1999)], which combines the ideas of boundary elements, the use of special fundamental solutions and the concept of a macro element. The algorithm is valid for both two- and three-dimensional flows. With the aim of making the method computationally effective, different solution strategies were developed for two-dimensional flows first, and are described in this paper. They can all be extended to three-dimensional flow computations, although the memory and computation cost is much higher. Despite this, there are several interesting applications of the three-dimensional version of the BDIM, ranging from coupling with vector potential flow kinematics formulation [1] in BDIM, where an effective treatment of external flows is possible, to conjugate heat transfer problems [Hriberšek and Kuhn, 'Conjugate heat transfer by boundary domain integral method', submitted to *Eng. Anal. Bound. Elem.* (1999)], where a combination with standard BEM approaches (boundary only discretization) is straightforward.

The present work continues the presentation of BDIM as developed in [Škerget *et al.* (1998)] and shows how the established velocity–vorticity formulation in BEM can further be developed in order to obtain a computationally efficient Navier–Stokes code.

2. NAVIER–STOKES EQUATIONS

In the present work, the fluid flow computation is based on an incompressible Newtonian fluid with a governing set of Navier–Stokes equations

$$\frac{\partial v_j}{\partial x_j}=0, \tag{2.1}$$

$$\frac{\mathrm{D}v_i}{\mathrm{D}t}=\nu\frac{\partial^2 v_i}{\partial x_j\,\partial x_j}-\frac{1}{\rho}\frac{\partial P}{\partial x_i}+Fg_i, \tag{2.2}$$

$$\frac{\mathrm{D}T}{\mathrm{D}t}=\kappa\frac{\partial^2 T}{\partial x_j\,\partial x_j}, \tag{2.3}$$

where v_i is the ith velocity component, x_i is ith co-ordinate, $\mathrm{D}/\mathrm{D}t$ represents the substantial derivative, $P=p-\rho g_j r_j$ is the modified pressure, p and g_j are the static pressure and the gravity respectively, while T stands for the temperature. The material properties, such as mass density ρ, specific isobaric heat c_p, kinematic viscosity ν and dynamic viscosity η, thermal diffusivity $\kappa=\lambda/\rho c_p$, where λ is the heat conductivity, are assumed to be constant parameters. The Boussinesq approximation is considered to model the buoyancy effect in the momentum equation (2.2) with function F as

$$F=\frac{\rho-\rho_0}{\rho_0}=-\beta_T(T-T_0), \tag{2.4}$$

where ρ_0 is the reference density at temperature T_0 and β_T is the thermal volume expansion coefficient.

In BDIM, the original set of Navier–Stokes equations is further transformed with the use of the velocity–vorticity variables formulation. As computational results in the present work are limited to the two-dimensional case, all the equations are written for this case.

With the vorticity vector ω_i representing the curl of the velocity field, the fluid motion computation scheme is partitioned into its kinematic and kinetic aspects [Škerget *et al.* (1998)]. The kinetics are governed by the vorticity transport equation

$$\frac{D\omega}{Dt} = \nu_0 \frac{\partial^2 \omega}{\partial x_j \partial x_j} + e_{ij} g_j \frac{\partial F}{\partial x_i}, \tag{2.5}$$

where e_{ij} $(i, j = 1, 2)$ is the permutation unit symbol $(e_{12} = -1,\ e_{21} = -1,\ e_{11} = e_{22} = 0)$.

The kinematics are derived for the solenoidal velocity field and is formulated in the form of a vector–elliptic Poisson's equation for the velocity vector

$$\frac{\partial^2 v_i}{\partial x_j \partial x_j} + e_{ij} \frac{\partial \omega}{\partial x_j} = 0. \tag{2.6}$$

In order to accelerate convergency and stability of the coupled velocity–vorticity iterative scheme, the false transient approach [3] is applied to the flow kinematic equation

$$\frac{\partial^2 v_i}{\partial x_j \partial x_j} - \frac{1}{\alpha} \frac{\partial v_i}{\partial t} + e_{ij} \frac{\partial \omega}{\partial x_j} = 0, \tag{2.7}$$

with α as a relaxation parameter.

Equations (2.3), (2.5) and (2.7) present the leading non-linear set of equations to which the weighted residuals technique of the BDIM has to be applied.

Integral representation of Equations (2.3), (2.5) and (2.7) can be derived by using integral representation of a parabolic diffusion–convective equation [Škerget *et al.* (1998)]. For all equations, the time derivative of a field function u (velocity, vorticity, temperature) is replaced by the finite difference approximation, namely

$$\frac{\partial u}{\partial t} \approx \frac{u_F - u_{F-1}}{\Delta t}, \tag{2.8}$$

with subscript F denoting the time step number. The following integral representations are obtained:

$$c(\xi) v_i(\xi) + \int_\Gamma v_i \frac{\partial u^*}{\partial n}\, d\Gamma = \int_\Gamma \left(\frac{\partial v_i}{\partial n} - e_{ij} \omega n_j \right) u^*\, d\Gamma - e_{ij} \int_\Omega \omega \frac{\partial u^*}{\partial x_j}\, d\Omega + \beta \int_\Omega v_{iF-1} u^*\, d\Omega, \tag{2.9}$$

$$\begin{aligned} &c(\xi)\omega(\xi) + \int_\Gamma \omega \frac{\partial U^*_\omega}{\partial n}\, d\Gamma \\ &= \frac{1}{\nu} \int_\Gamma \left(\nu \frac{\partial \omega}{\partial n} + e_{ij} n_i g_j F - \omega v_n \right) U^*_\omega\, d\Gamma + \frac{1}{\nu} \int_\Omega (\omega \hat{v}_j - e_{ij} g_j F) \frac{\partial U^*_\omega}{\partial x_j}\, d\Omega + \beta \int_\Omega \omega_{F-1} U^*_\omega\, d\Omega, \end{aligned} \tag{2.10}$$

$$c(\xi)T(\xi)+\int_\Gamma T\frac{\partial U_T^*}{\partial n}\,\mathrm{d}\Gamma$$
$$=\frac{1}{\kappa}\int_\Gamma\left(\kappa\frac{\partial T}{\partial n}-Tv_n\right)U_T^*\,\mathrm{d}\Gamma+\frac{1}{\kappa}\int_\Omega T\hat{v}_j\frac{\partial U_T^*}{\partial x_j}\,\mathrm{d}\Omega+\beta_t\int_\Omega T_{F-1}U_T^*\,\mathrm{d}\Omega. \tag{2.11}$$

In order to successfully apply the elliptic diffusion–convective fundamental solution,

$$U_\omega^*=\frac{1}{2\pi}K_0(\mu_\omega r)\exp\left(\frac{\bar{v}_jr_j}{2v}\right),$$
$$\frac{\partial U_\omega^*}{\partial x_j}n_j=\frac{n_j}{2\pi r^2}\left[\mu_\omega rK_1(\mu_\omega r)r_j-\frac{r^2}{2v}K_0(\mu_\omega r)\bar{v}_j\right]\exp\left(\frac{\bar{v}_jr_j}{2v}\right), \tag{2.12}$$
$$U_T^*=\frac{1}{2\pi}K_0(\mu_Tr)\exp\left(\frac{\bar{v}_jr_j}{2\kappa}\right),$$
$$\frac{\partial U_T^*}{\partial x_j}n_j=\frac{n_j}{2\pi r^2}\left[\mu_TrK_1(\mu_Tr)r_j-\frac{r^2}{2v}K_0(\mu_Tr)\bar{v}_j\right]\exp\left(\frac{\bar{v}_jr_j}{2\kappa}\right), \tag{2.13}$$

the decomposition of the velocity field $v_j=\bar{v}_j+\hat{v}_j$, where $\bar{v}_j$ is an average constant vector and $\hat{v}_j$, a perturbed part, is applied in Equations (2.10) and (211). For flow kinematics, the elliptic modified Helmholz fundamental solution, i.e.

$$u^*=\frac{1}{2\pi}K_0(\mu r),$$
$$\frac{\partial u^*}{\partial x_j}n_j=\frac{r_jn_j}{2\pi r^2}\mu rK_1(\mu r), \tag{2.14}$$

is applied. Notice, when $\bar{v}_j=0$, Equations (2.12) and (2.13) are equal to Equation (2.14). K_0 and K_1 are the modified Bessel functions of the second kind and $r_j(\xi,s)$ is the vector from the source point ξ to the reference field point s. The parameters μ, μ_ω, and μ_T are defined as

$$\mu^2=\frac{1}{v\Delta t}=\beta, \tag{2.15}$$
$$\mu_\omega^2=\left(\frac{\bar{v}}{2v}\right)^2+\beta, \tag{2.16}$$
$$\mu_T^2=\left(\frac{\bar{v}}{2\kappa}\right)^2+\beta_t, \tag{2.17}$$

with $\beta_t=1/\kappa\Delta t$. Searching for an approximate numerical solution, integral equations (2.9), (2.10) and (2.11) are written in a discretized manner in which the integrals over the boundary and domain are approximated by a sum of integrals over E individual boundary elements and C internal cells respectively. The variation of all field functions is approximated by the use of interpolation polynomials [Škerget *et al.* (1998)]. Interpolation functions, from constant interpolation [4] to quadratic interpolation [Škerget *et al.* (1998)], were applied. The combination of discontinuous boundary elements and continuous internal cells proved to be the most stable in obtaining solutions for higher *Re* value flows.

After applying the discretized integral equations to all subdomain boundary and internal (for vorticity and heat transport only) nodes, the following implicit matrix systems are obtained:

$$[H]\{v_i\} = [G]\left\{\frac{\partial v_i}{\partial n}\right\} + e_{ij}[G]\{\omega n_j\} - e_{ij}[D_j]\{\omega\} + \beta[D]\{v_i\}_{F-1}, \tag{2.18}$$

$$[H]\{\omega\} = \frac{1}{\nu}[G]\left\{\nu\frac{\partial \omega}{\partial n} - \omega n_n + e_{ij}n_i g_j F\right\} + \frac{1}{\nu}[D_i][\hat{v}_i]\{\omega\} - \frac{1}{\nu}e_{ij}[D_i]\{g_j F\} + \beta[D]\{\omega\}_{F-1}, \tag{2.19}$$

$$[H]\{T\} = \frac{1}{\kappa}[G]\left\{\kappa\frac{\partial T}{\partial n} - T n_n\right\} + \frac{1}{\kappa}[D_i][\hat{v}_i]\{T\} + \beta[D]\{T\}_{F-1}. \tag{2.20}$$

The set of equations (2.18), (2.19) and (2.20), together with selected boundary and initial conditions [Škerget *et al.* (1998)], has to be solved numerically to obtain the solution of a chosen fluid dynamics problem. To make the computations effective in terms of computation times and memory requirements, several modifications have to be made. The following sections present the most important steps in the construction of a fast BDIM numerical code.

3. COMPUTATIONAL PROCEDURE AND EFFICIENCY OF THE METHOD

3.1. Subdomain technique

Integrals in Equations (2.9), (2.10) and (2.11) are written for all the nodes on the boundary and in the domain, and when applied to large-scale computations they would cause a prohibitive computational and memory cost to the BDIM. Therefore, the application of the subdomain technique to all three leading equations is an obvious choice. The idea is to partition the entire solution domain into subdomains to which the described numerical procedure can be applied. The final systems of equations for the entire domain are obtained by adding the sets of equations for each subdomain together, considering compatibility and equilibrium conditions [Škerget *et al.* (1998)] between their interfaces, resulting in a more sparse system matrices suitable to be solved with iterative techniques [2].

Several versions of subdomains were developed and tested [4]. Although larger subdomains (more than one internal cell) are more stable when computing higher *Re* flows, they also result in densely populated matrices and problems in dealing with non-homogenous material properties. Therefore, the extreme approach of one cell = one subdomain has to be used in order to overcome these difficulties. In Table III, the positive influence of such subdomain division on the population of system matrices (equal to the sparse pattern of the ILU(0) preconditioner) is clearly recognizable.

3.2. Solution of a non-linear set of BDIM equations

The set of equations (2.18), (2.19) and (2.20) is coupled into a strong non-linear system of equations. The solution of a chosen problem has to be computed in a time loop, and within the time loop an iteration process (outer loop) is required to obtain the solution.

In the outer loop (denoted k), the attention must be focused on velocity and vorticity, as they represent the coupling of the flow kinematics with the flow kinetics. Since vorticity is always on the right-hand-side of Equation (2.18), velocity is dependent on vorticity distribution. To avoid problems with convergence it is necessary to relax the vorticity values,

$$^{k+1}\omega = \Theta\,^{k+1}\omega + (1-\Theta)\,^{k}\omega, \tag{3.21}$$

with the underrelaxation parameter Θ. Two approaches to this task were tested:

(a) underrelaxation of the entire vorticity field, Θ_a,
(b) underrelaxation of the pure domain vorticity values (except the values at solid wall, inflow and outflow boundaries), Θ_b.

In Table I, the comparison between both approaches is presented. Case 1 stands for a driven cavity problem ($Re = 100$, $\Delta t = 10^{12}$, 10×10 mesh), case 2 stands for a natural convection problem ($Ra = 10^4$, $\Delta t = 10^{12}$, 10×10 mesh). The comparison demonstrates that approach (b) is far more successful. The reason is that the boundary vorticity values from the flow kinematics are the only boundary conditions for the vorticity transport equation, from which the domain vorticity values are computed (Equation (2.19)).

3.3. Numerical integration

The evaluation of integrals in Equations (2.9), (2.10) and (2.11) is limited to local integration over one subdomain only. For the extreme subdomain technique, the numerical integration is performed locally for one cell and its corresponding boundary elements only, decreasing the required computational cost for BDIM integrals.

Since the kernels (2.12), (2.13) and (2.14) have steep magnitude variations in the vicinity of singular points, the polar transformation, together with the division of integrating intervals into subintervals, has to be applied. For the basic integration procedure, Gaussian quadrature is used.

The integrals for flow kinematics depend only on geometry and time step increment, therefore, their evaluation has to be performed only at the beginning of a computation. In contrast, all transport equations have kernels depending on geometry, material properties and average velocity values. Since at least velocity values change in each iteration during the computation, nominal integral values should change too. To avoid the updating of integrals in each iteration, some simplification has to be performed.

Freezing the value of $\bar{v}$, when the velocity within the subdomain does not change more than the predefined error $\epsilon_\int$ (N_s is number of subdomains), i.e.

$$\text{for } j = 1, \ldots, N_s\text{: if } \left| \frac{{}^{k+1}\bar{v}_i^j - {}^{k}\bar{v}_i^j}{{}^{k+1}\bar{v}_i^j} \right| < \epsilon_\int \text{ then freeze } \bar{v}^j, \tag{3.22}$$

is the key to an effective decrease in computational time. If in one subdomain $\bar{v}$ is frozen, the occurring change in velocity components v_i transfers to the perturbed part $\tilde{v}_i$. Freezing the velocity values does not greatly influence the overall convergence of the outer loop (Table II) except for large values of $\epsilon_\int$. The test cases are the same as in Table I, with $\Theta = 0.1$.

Table I. Comparison of underrelaxation techniques Θ_a and Θ_b

	$\Theta_a = 0.25$	$\Theta_a = 0.1$	$\Theta_b = 0.25$	$\Theta_b = 0.1$
Case 1	161	252	101	113
Case 2	15	36	9	10

Table II. Influence of $\epsilon_\int$ on the convergence of BDIM

	$\epsilon_\int = 0.0$	$\epsilon_\int = 0.01$	$\epsilon_\int = 0.1$	$\epsilon_\int = 0.25$
Case 1	113	116	137	148
Case 2	10	10	11	12

3.4. Preconditioned Krylov subspace methods

To increase the speed of the BDIM computations, iterative methods for the solution of all three linearized equations have to be used. One of the best possibilities is to use the preconditioned Krylov subspace methods, since they have low memory demands and can be quite easily incorporated into the code. Since conjugate gradients squared (CGS) [5] proved effective in previous BDIM formulations [2], it was chosen as a Krylov subspace solver.

In order to accelerate the convergence of KSM, preconditioning is used, i.e.

$$[Q]^{-1}[A]\{x\} = [Q]^{-1}\{b\}. \tag{3.23}$$

In many papers and reports it was found that the choice of preconditioner can be of crucial importance in decreasing the number of iterations (inner loop in BDIM) of KSM. ILU-based preconditioners proved to be very effective in the BDIM [2]. Apart from ILU(0), block ILU preconditioners can not be applied to the new scheme of the BDIM due to different system matrix structure. Since the ILU preconditioning gains on robustness if we allow a certain amount of fill-in during factorization, a level based fill-in ILU preconditioner was constructed for the use in BDIM.

3.4.1. Level based fill-in ILU decomposition. Level based fill-in ILU (ILU(l), l is level of fill-in) is already known from other approximation methods, where it was used effectively as a preconditioner for KSM. The main idea is that the filling entries are classified by the level of the matrix entry that creates them. If we fix the number of levels to 1 and all the original entries have level 0, new entries in the preconditioner $[Q]$, obtained from $[A(0)]$ elements, have level 1. Limiting fill-in to only one level has a great advantage since there is no need to change the indirect addressing representation of $[Q]$ during the factorization process. The fill-in entries are stored in a new vector together with the known positions in the new $[Q]$ and the monitoring of the amount of fill-in is very convenient. Since the positions of the original entries of $[A]$ do not change during the outer iteration loop, it is only necessary to determine the sparse pattern of ILU(l) once, in the first outer iteration, which saves on CPU time.

Increasing the level of ILU(l) preconditioner increases the number of non-zero elements in the $[Q]$ matrix. To control the amount of new entries created, a threshold criterion ϵ_t is introduced. Here, only the new entries larger than a predefined ϵ_t fraction of the diagonal element of $[Q]$ are included ($acla(k)$ being the kth new entry):

$$\text{if } acla(k) > \epsilon_t\, acla(jdia(i)) \quad \text{then add } acla(k) \text{ into } [Q]. \tag{3.24}$$

Creating a fill-in in ILU(1) with a threshold criterion describes Algorithm 1.

Algorithm 1 ILU(1) decomposition with a threshold criterion

```
jnrow(1) = 0, ia = 0, k = 0
DO i = 2, n
  jdi = jdiA(i)
  jright = jbgA(i + 1) - 1
  coad = clA(jdi) * $\epsilon_t$
  DO j = jbgA(i) + 1,jright
    jpoint(jclA(j)) = j
  ENDDO
  DO j = jbgA(i),jdi - 1
    m = jclA(j)
    clA(j) = clA(j)/clA(jdiA(m))
```

```
      DO l = jdiA(m) + 1,jbgA(m + 1) − 1
        k = jpoint(jclA(l))
        IF (k.eq.0)THEN
          DO m = jnrow(i − l) + 1,ia
            if(jclA(l) .eq. jacla(m))goto 1
          ENDDO
          acla(ia) = − clA(j) * clA(l)
          IF(acla(ia) .lt. coad)goto 1
          ia = ia + 1
          jacla(ia) = jclA(l)
          goto 1
          ENDDO
        clA(k) = clA(k) − clA(j) * clA(l)
1       ENDDO
    ENDDO
    DO j = jbgA(i) + 1,jright
      jpoint(jclA(j)) = 0
    ENDDO
    jnrow(i) = ia
 ENDDO.
```

In Algorithm 1, the indirect addressing [6] for the representation of matrices was used. For this purpose four different vectors are needed as follows:

- **clA**: contains non-zero elements of the matrix $[A]$,
- **jclA**: contains coloumn numbers for each element of **clA**,
- **jbgA**: ith member gives position of the beginning of the ith row of matrix $[A]$ in vector **clA**,
- **jdiA**: ith member gives position of the diagonal element of ith row of matrix $[A]$ in vector **clA**.

For the representation of the preconditioner another set of the previous four vectors is needed and additionally one vector (**jpoint**) which serves as an interface between the original matrix representation for $[A]$ and a representation for $[Q]_{\mathrm{ILU}(l)}$.

In cases when problems with convergence occur, additional levels can be added. This is done simply by rewriting the original matrix $[A]$ into the representation vectors of $[Q]$ for ILU(l), followed by executing the same ILU routine (Algorithm 1) for adding one level of fill-in (level $l+1$) to a given preconditioner. The sparse pattern of $[A]$ and consequently of $[Q]$s does not change if the computational mesh and element connectivity remain the same. Therefore, at the beginning of the outer loop computation, several ILU(l) representations can be determined and stored for the later use. Algorithm 2 for this step reads as:

Algorithm 2 ILU(l) loop

```
1. Set number of ILU levels $NL$.
2. Transfer: $[A] \rightarrow [Q]_{\mathrm{ILU}(0)}$.
3. $l = 1$
   3.1. Execution of the basic ILU($l$) algorithm on $[Q]_{\mathrm{ILU}(l-1)}$
   3.2. Formulation of the new representation for ILU($l$).
   3.3. Transfer: $[A] \rightarrow [Q]_{\mathrm{ILU}(l)}$.
```

Table III. Timing and iteration information for ILU(l)

Solver	Mesh 50×50				Mesh 80×80			
Ra	10^3		10^4		10^3		10^4	
	I	T	I	T	I	T	I	T
CGS-ILU(0)	67	11.59	175	35.21	94	41.10	264	114.71
CGS-ILU(1)	23	4.99	78	19.47	30	16.69	157	87.79
CGS-ILU(2)	14	3.53	36	10.39	19	11.78	60	36.77
CGS-ILU(3)	10	3.04	18	6.05	13	9.63	22	16.26

I = number of iterations, T = CPU time (s).

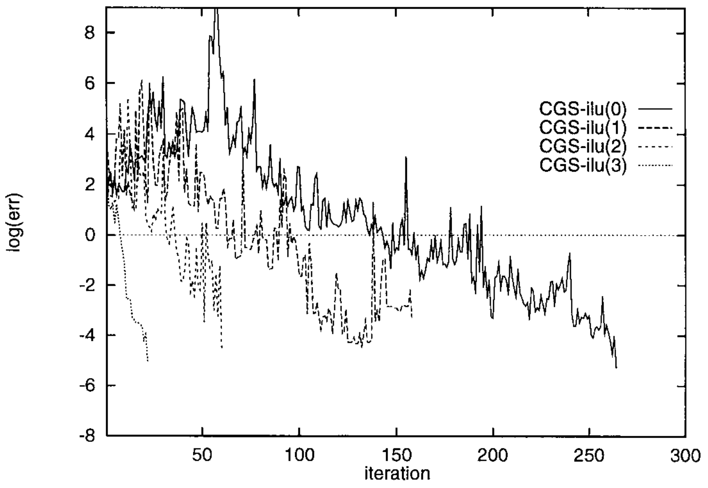

Figure 1. Effect of ILU(l) on convergence of CGS, mesh 80 × 80, $Ra = 10000$.

3.4. $l = l + 1$

3.5. IF ($l \leq NL$) go to 3.1.

4. Save the representation vectors for formed preconditioners (out of core).
5. End.

Another benefit of the looped creation of the ILU(l) preconditioners is present when ILU(l) preconditioned KSM exhibit problems with convergence. Then, the inner iteration loop can simply be restarted by switching to a higher level ILU preconditioner—ILU($l+1$).

In the following, a detailed description of the efficiency of developed ILU(l) preconditioners in the case of the energy transport equation for natural convection in a closed cavity is presented. Only the first outer iteration solution (for $\Delta t = 10^{12}$) was computed for the test problem as it represents the toughest test for the preconditioner (no initial guess available). Constant elements and cells were used. All computations were performed on a HP 712 workstation.

From the data in Table III, it can be seen that the rate of convergence increases with the increasing level number in the preconditioner. This is clearly presented in Figure 1, where the iteration errors, except the last converging one, are plotted. The decrease in the number of iterations is accompanied by the increase of non-zero entries in $[Q]$, Table IV. This increases the CPU time needed for the ILU decomposition, Table V. Imposing a threshold criterion brings some CPU time savings with little or no additional iterations, however, the threshold should not be set too tight, Table VI.

3.4.2. Stopping criterion. One of the most important factors in the successful implementation of iterative methods is the choice of an appropriate stopping criterion. When only one solution of the system of equations is needed (a common case in potential problems by BEM) and no reasonable initial guess is available (zero vector taken as initial guess), the standard stopping criterion for Krylov subspace methods is used, i.e.

$$\frac{\|\{r\}_{\mathrm{KSM}}\|}{\|[Q]^{-1}\{b\}\|} \leq \epsilon, \tag{3.25}$$

where r_{KSM} denotes the residual of a KSM, a product of one Krylov iteration step.

In case of non-linear computations in the BDIM, the resulting systems of linearized equations have to be solved several hundred or even thousand times before the solution of a problem is achieved. In the BDIM iteration process there is a possibility to use the solution of the previous outer iteration as the initial guess for CGS. However, in order to preserve the advantage of a good initial guess, a combination of the standard stopping criterion, Equation (3.25), with the stopping criterion based on true equation residual norm should be used. In the following, error plots will correspond to these definitions:

Table IV. Number of non-zero entries in preconditioners

	Mesh			
	10×10	25×25	50×50	80×80
ILU(0)	4300(1.72)	27 625(0.28)	111 500(0.07)	286 400(0.03)
ILU(1)	7361(2.94)	48 338(0.49)	196 760(0.13)	506 810(0.05)
ILU(2)	9296(3.71)	62 546(0.64)	256 295(0.16)	662 045(0.06)
ILU(3)	11 966(4.78)	84 041(0.86)	349 165(0.22)	906 565(0.09)

Table V. CPU time for ILU decomposition

	Mesh			
	10×10	25×25	50×50	80×80
ILU(0)	0.001	0.02	0.08	0.26
ILU(1)	0.01	0.09	0.28	0.64
ILU(2)	0.03	0.16	0.38	1.02
ILU(3)	0.08	0.53	0.65	1.94

Table VI. Influence of threshold criterion, ILU(3), mesh 80×80

	CGS	CGS-$\epsilon_t=10^{-6}$	CGS-$\epsilon_t=10^{-3}$	CGS-$\epsilon_t=10^{-1}$
Iterations	22	23	24	55
CPU time (s)	16.26	15.53	15.97	33.12
CPU time (ILU) (s)	1.94	1.39	1.18	1.05
Non-zero elements	906 565	784 821	730 267	664 021

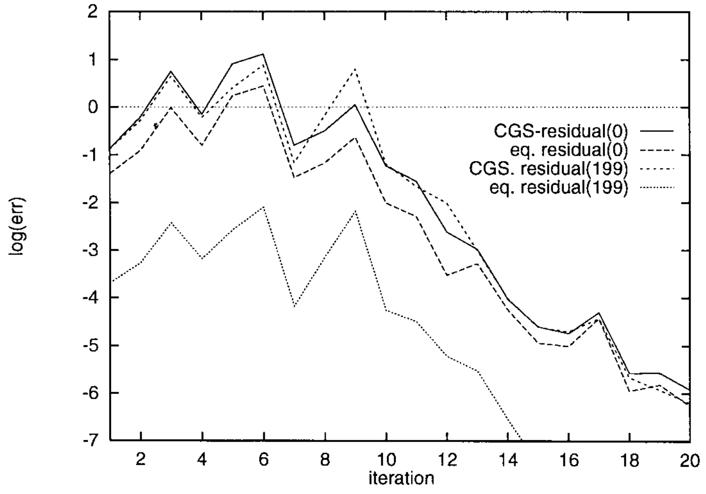

Figure 2. Flow kinematics.

- CGS:

$$\text{err} = \frac{\|\{r\}_{\text{CGS}}\|}{\|[Q]^{-1}([A]^0\{x\} - \{b\})\|}, \tag{3.26}$$

- Equation residual:

$$\text{err} = \frac{\|[A]\{x\} - \{b\}\|}{\|\{b\}\|}. \tag{3.27}$$

For the stopping criterion the value $\epsilon = 10^{-6}$ was imposed. The natural convection test problem ($\Delta t = 10^{12}$, steady state, 10×10 subdomains mesh) was used as a model problem for testing the behaviour of iterative methods.

If we compare the histories of first outer iteration with convergence histories for the 200th outer iteration (Figures 2 and 3), we can see that CGS errors are quite similar and lead to roughly the same number of iterations until convergence is achieved. This shows that using only a good initial guess does not automatically lead to faster convergence of CGS. On the other hand, Figures 2 and 3 show that when a good initial guess (solution from the outer iteration number 199) is used, the equation residual norm (3.27) decreases much faster than the CGS error. To use this fact and to decrease the number of iterations an economical way of monitoring the residual norms of the original equations is needed.

Monitoring the convergence of the true residual demands an additional matrix–vector product evaluation, which costs more CPU time. Since the ratio between $\|\{r\}_{\text{CGS}}\|$ and $\|\{r\}\|$ does not change much through the iterations (Figures 2 and 3) an obvious choice is to add an extra evaluation of $\|r\|$ only when the $\|\{r\}_{\text{CGS}}\|$ has reached a certain level. In the present case, the following strategy was used:

Algorithm 3 Scaling of the stopping criterion

1. Let $\|^m\{r\}\| = (\|[A]^m\{x\} - \{b\}\|)/\|\{b\}\|$ and $\|^m\{r_{\text{CGS}}\}\| = \|f_{\text{CGS}}\|/(\|[Q]^{-1}([A]^m\{x\} - \{b\})\|)$, f_{CGS} being the quasi-residual form of a CGS [5], and m the inner iteration number.
2. Set initial guess: $\{x\} = {}^0\{x\}$, ${}^0\{x\}$ is the solution from the previous outer iteration.

3. CGS loop: $m = 1$, maxit.Additional steps at the end of the main CGS loop:
   - if $(m = 1)$ $R = R = \epsilon(\|^m\{r_{\text{CGS}}\}\|/\|^m\{r\}\|)$
   - if $(\|^m\{r_{\text{CGS}}\}\| \le R)$ then
     compute $\|^m\{r\}\|$, if $(\|^m\{r\}\| \le \epsilon)$ stop iterating,
     endif.

This convergence monitoring reveals in practice as very effective since it reduces the number of iterations of Krylov solvers and preserves the accuracy of the results throughout the outer iteration loop of BDIM, Figure 4. In Figure 4, curves A and C are results with initial guesses from the previous outer iteration, and curves B an D are results of using zero initial vector. This test case was performed on 92×92 subdomain mesh with quadratic macro elements, with 101 325 unknowns in system (3.23).

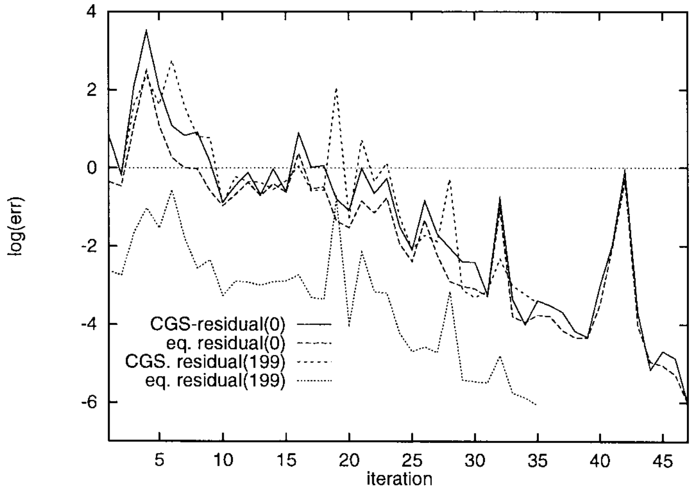

Figure 3. Vorticity transport, CGS-ILU(0), $Ra = 1000$.

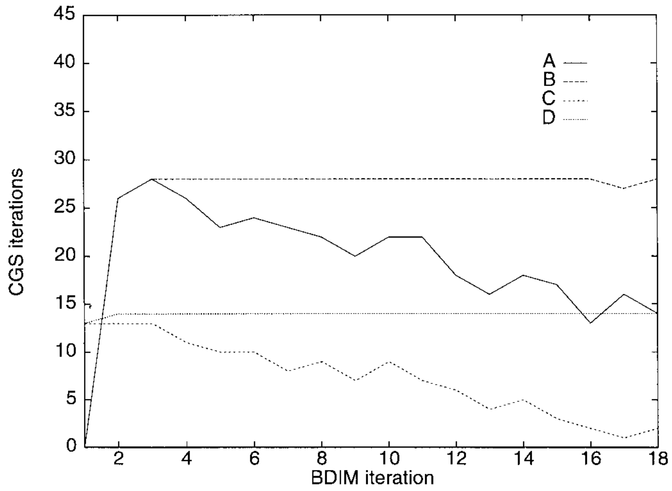

Figure 4. Reduction of CGS iterations through scaling of stopping criterion: curves A and B, CGS-ILU(1) for flow kinematics, curves C and D, CGS-ILU(4) for heat transport.

Table VII. Natural convection in closed cavity: average *Nu* numbers

	Grashof number				
Nusselt number	10^3	10^4	10^5	10^6	10^7
BDIM	1.06	2.01	4.08	7.99	14.18
Kaminski [7]	1.06	—	4.08	7.99	15.09

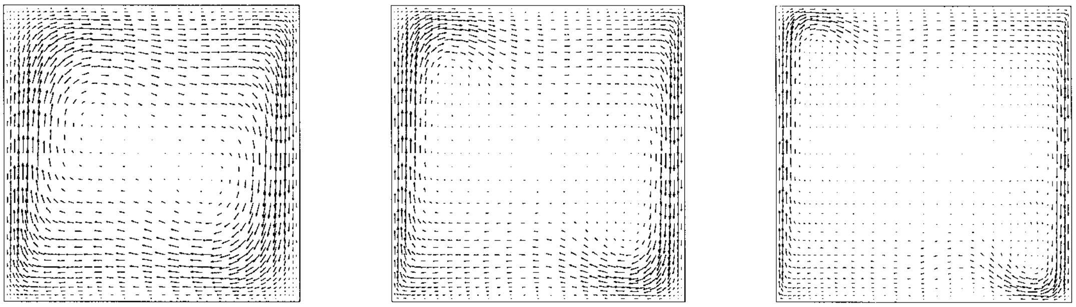

Figure 5. Natural convection, velocity field: $Gr = 10^5$ (left), $Gr = 10^6$ (middle), $Gr = 10^7$ (right).

4. TEST EXAMPLES

A series of common computational fluid dynamics (CFD) benchmark tests were computed to test the abilities of the BDIM. The first results for natural convection in a closed cavity are presented in [Škerget *et al.* (1998)] for *Ra* up to 10^6. The driven cavity problem for *Re* up to 1000 and backward-facing step flow for *Re* up to 400 are also presented in [Škerget *et al.* (1998)] together with the results on mesh size sensitivity. Here, results for these test cases for higher values of *Re* and *Gr* numbers are presented. In all tests, the quadratic interpolation for boundary elements and internal cells was used.

4.1. Natural convection in a closed cavity

The natural convection in a closed square cavity was computed for *Gr* number values in the range of 10^3–10^7, and air ($Pr = 0.71$) was selected for the fluid. The results in the form of average *Nu* values for the vertical mid-plane are compared with the results of the finite volume method [7] and given in Table VII. The computational mesh consisted of 20×20 non-uniform subdomains. Table VII presents a comparison of average Nusselt values for the vertical mid-plane with the results of [7] for an infinite side wall conductivity. The agreement of BDIM results is very good as a mesh with only 400 subdomains was used. Figures 5 and 6 show velocity and temperature plots respectively for *Gr* values in the range 10^5–10^7.

4.2. Driven cavity

In [Škerget *et al.* (1998)] the driven cavity problem was computed with a 20×20 subdomain mesh, which allowed accurate flow computation up to $Re = 1000$. Here, a 30×30 non-uniform subdomain mesh was used and flows up to $Re = 3200$ were computed. Figures 7 and 8 show the computation results for the selected *Re* number values. The comparison with benchmark results [8] shows good agreement, Figures 9–11. Additionally, to the flow pattern of

$Re = 1000$, for $Re = 3200$ a new large recirculation region is formed near the upper left singular point, as well as an additional small eddy in the lower right recirculation region.

4.3. Backward-facing step flow

As reported in [1], the computational mesh with 60×12 non-uniform subdomains proved to be best for the $Re = 400$ computation, therefore, it was also used for the computation of the flow with $Re = 800$. At this Re value an additional recirculation region is formed at the upper wall. From the experimental results [9] it is also known that three-dimensional effects influence the flow at that Re value, therefore, the results of two-dimensional numerical computations

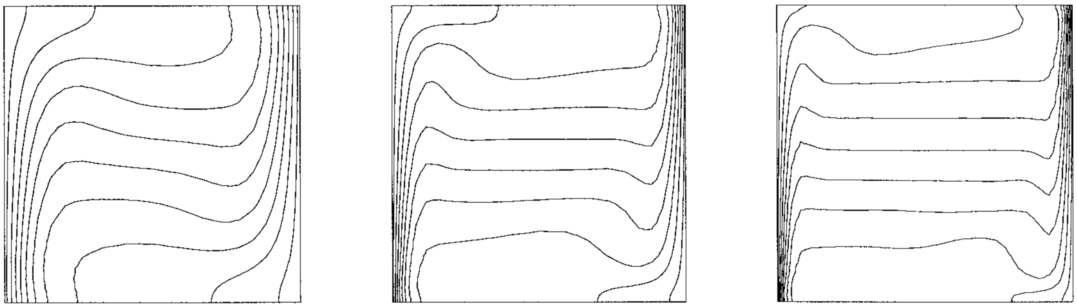

Figure 6. Natural convection, temperature isolines: $Gr = 10^5$ (left), $Gr = 10^6$ (middle), $Gr = 10^7$ (right).

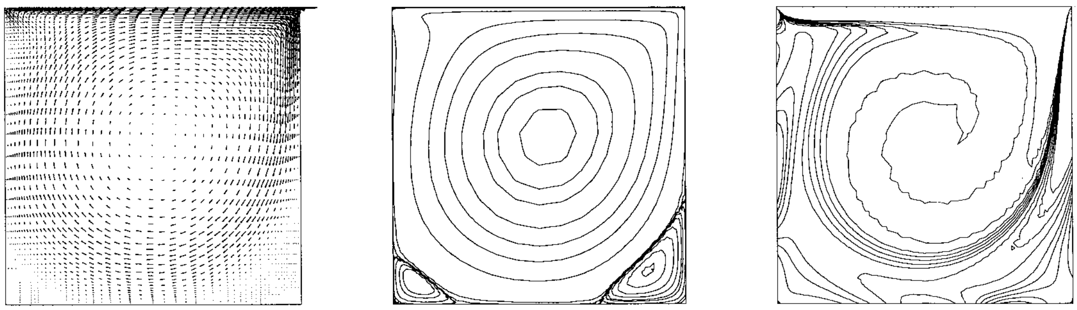

Figure 7. Driven cavity, $Re = 1000$: velocity (left), streamlines (middle), vorticity (right).

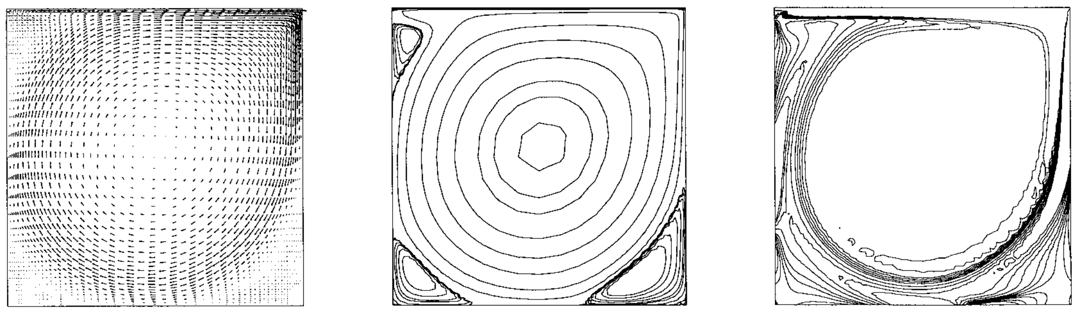

Figure 8. Driven cavity, $Re = 3200$: velocity (left), streamlines (middle), vorticity (right).

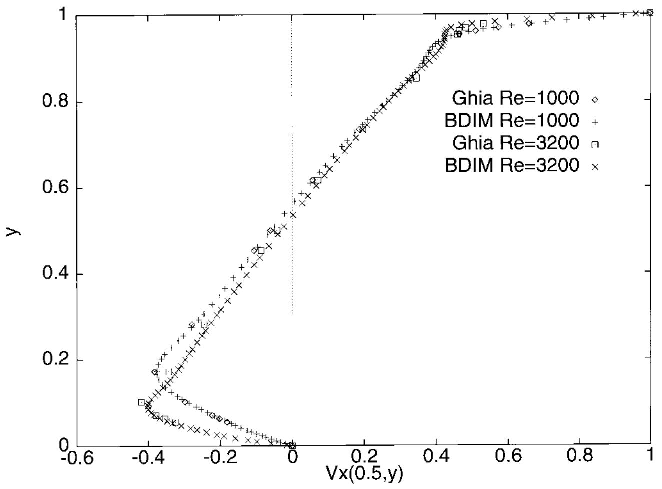

Figure 9. Driven cavity, x component velocity profiles.

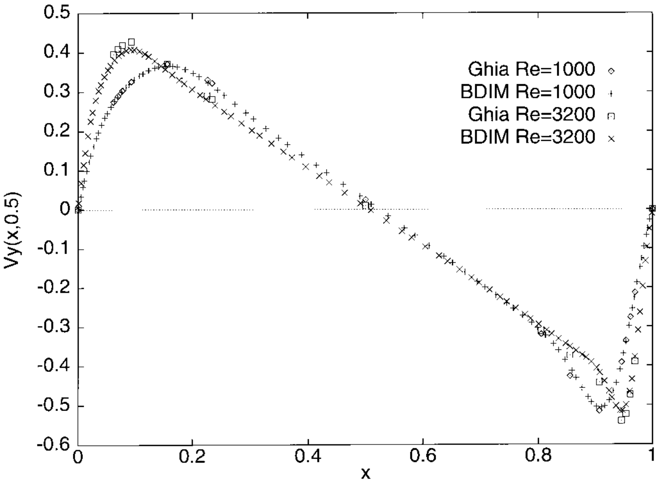

Figure 10. Driven cavity, y component velocity profiles.

could not be directly compared. Nevertheless, due to the complex flow structure and the interaction of eddies, the results of the two-dimensional computations still give a good insight into capabilities of an approximation method.

Figures 12 and 13 show velocity vector and vorticity plots respectively, for $0 < x < 10$ and $0 < x < 15$ ($x = 15$ marks the outflow boundary and $0 < y < 1$).

The lengths of the recirculation regions can be depicted from Figure 14, where the vorticity values at the upper and lower walls are plotted for the entire channel length. Additionally, Table VIII compares the data with data from [9].

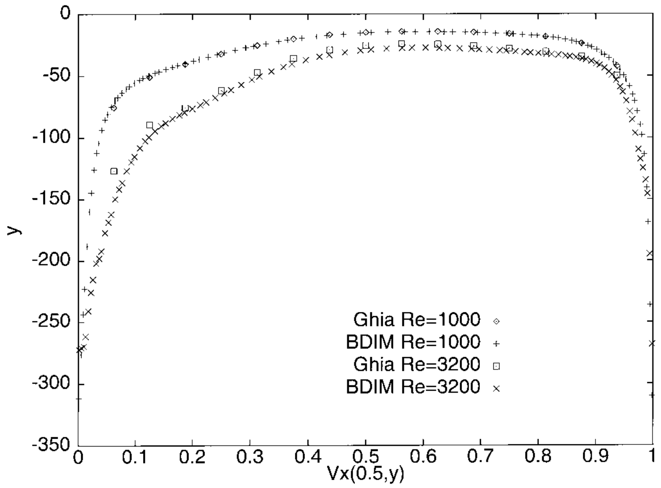

Figure 11. Driven cavity, vorticity profiles.

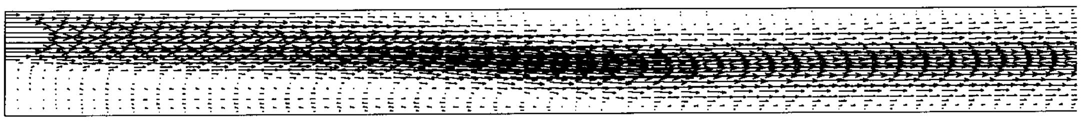

Figure 12. Backward-facing step, velocity field at $Re = 800$.

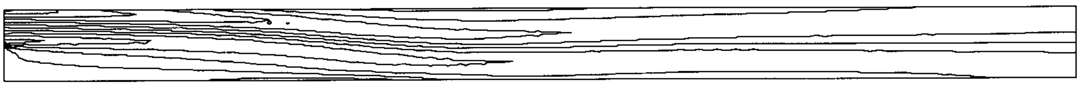

Figure 13. Backward-facing step, vorticity field at $Re = 800$.

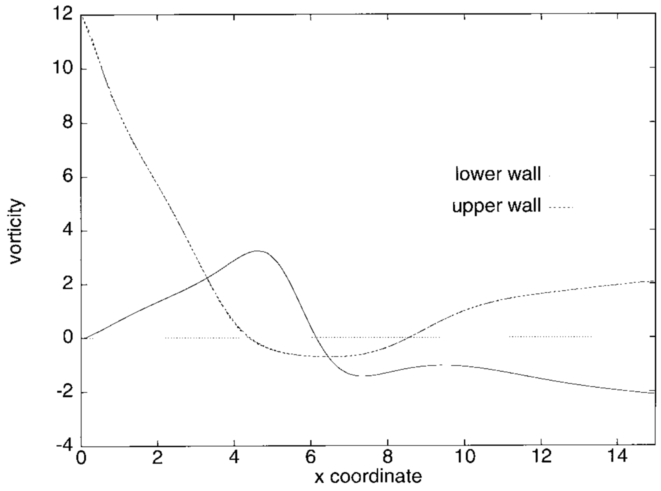

Figure 14. Backward-facing step, vorticity at the lower and upper wall.

Table VIII. BFS, comparison of recirculation data: P_1, lower reattachment point; P_2, upper separation point; P_3, upper reattachment point

	P_1	P_2	P_3
BDIM	6.2	4.3	8.6
Armaly [9]	7.2	5.3	9.4

5. CONCLUSION

Computational techniques for the fast computation of CFD problems by the BDIM are presented. Freezing average velocity values in the computation of integrals, scaling of the stopping criterion for Krylov subspace methods and the level based ILU preconditioning are the main tools for decreasing the computational cost of the BDIM. Several computational results show that the accuracy of the method is preserved even in cases of higher values of *Re* and *Ra* numbers.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support received from the Ministry of Science and Technology, Republic of Slovenia. Further thanks go to Alexander von Humboldt Foundation, FRG, for accepting the project ‘Fast boundary domain integral method for transport problems in fluid dynamics’, of Dr M. Hriberšek, as part of the project’s results are included in the present work.

REFERENCES

1. A. Alujevič, G. Kuhn and L. Škerget, ‘Boundary elements for the solution of Navier–Stokes equations’, *Comput. Methods Appl. Mech. Eng.*, **91**, 1187–1201 (1991).
2. M. Hriberšek and L. Škerget, ‘Iterative methods in solving Navier–Stokes equations by the boundary element method’, *Int. J. Numer. Methods Eng.*, **39**, 115–139 (1996).
3. G. Guj and F. Stella, ‘A vorticity–velocity method for the numerical solution of 3D incompressible flows’, *J. Comput. Phys.*, **106**, 286–298 (1993).
4. L. Škerget and Z. Rek, ‘Boundary-domain integral method using a velocity–vorticity formulation’, *Eng. Anal. Bound. Elem.*, **15**, 359–370 (1995).
5. P. Sonneveld and P. Wesseling, ‘Multigrid and conjugate gradient methods as convergence acceleration techniques’, in *Multigrid Methods for Integral and Differential Equations*, Clarendon Press, Oxford, 1985, pp. 117–167.
6. A. Van der Ploeg, ‘Preconditioning techniques for non-symmetric matrices with application to temperature calculations of cooled concrete’, *Int. J. Numer. Methods Eng.*, **35**, 1311–1328 (1992).
7. D.A. Kaminski and C. Prakash, ‘Conjugate natural convection in a square enclosure: effect of conduction in one of the vertical walls’, *Int. J. Heat Mass Transf.*, **29**, 1979–1988 (1986).
8. U. Ghia, K.N. Ghia and C.T. Shin, ‘High-*Re* solutions for incompressible flow using the Navier–Stokes equations and a multigrid method’, *J. Comput. Phys.*, **48**, 387–411 (1982).
9. B.F. Armaly, F. Durst, J.C.F. Pereira and B. Schonung, ‘Experimental and theoretical investigation of backward-facing step flow’, *J. Fluid Mech.*, **172**, 473–496 (1983).