Electronic Transactions on Numerical Analysis.
Volume 26, pp. 330-349, 2007.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

EFFICIENT DEFLATION METHODS APPLIED TO 3-D BUBBLY FLOW PROBLEMS*

J. M. TANG† AND C. VUIK†

Abstract. For various applications, it is well-known that deflated ICCG is an efficient method to solve linear systems with an invertible coefficient matrix. Tang and Vuik [J. Comput. Appl. Math., 206 (2007), pp. 603–614] proposed two equivalent variants of this deflated method, which can also solve linear systems with singular coefficient matrices that arise from the discretization of the Poisson equation with Neumann boundary conditions and discontinuous coefficients. In this paper, we also consider the original variant of DICCG in Vuik, Segal, and Meijerink [J. Comput. Phys., 152 (1999), pp. 385–403], that already proved its efficiency for invertible coefficient matrices. This variant appears to be theoretically equivalent to the first two variants, so that they all have the same convergence properties. Moreover, we show that the associated coarse linear systems within these variants can be solved both directly and iteratively. In applications with large grid sizes, the method with the iterative coarse solver can be substantially more efficient than the one with the standard direct coarse solver.

Additionally, the results for stationary numerical experiments of Tang and Vuik [J. Comput. Appl. Math., 206 (2007), pp. 603–614] have only been given in terms of number of iterations. After discussing some implementation issues, we show in this paper that deflated ICCG is considerably faster than ICCG in the most test cases, by taking the computational time into account as well. Other 3-D time-dependent numerical experiments with falling droplets in air and rising air bubbles in water are performed, in order to show that deflated ICCG is also more efficient than ICCG in these cases, considering both the number of iterations and computational time.

Key words. deflation, conjugate gradient method, preconditioning, Poisson equation, symmetric positive semi-definite matrices, bubbly flow problems, inner-outer iterations

AMS subject classifications. 65F10, 65F50, 65N22

1. Introduction. Numerical simulations of bubbly flows are relevant to problems found in various disciplines, such as the oil, nuclear and chemical industries. These bubbly flows are governed by the incompressible Navier-Stokes equations. In many popular operator-splitting formulations of these equations, it is the Poisson problem, used to approximate the pressures, which is the most computationally challenging despite its elliptic origins, see also [22,23,27]. In our applications, the pressure problem has discontinuous coefficients, due to jumps in the density.

Efficient solution of the Poisson problem in complex domains depends upon the availability of fast solvers for sparse linear systems. Popular in use are Krylov subspace iterative solvers, such as the CG method with an incomplete Cholesky preconditioner, denoted by ICCG [17]. The results given in this paper for ICCG and its variants may be generalized to other SPD preconditioners, which have comparable spectral properties.

We seek to improve the ICCG method for the Poisson solve in order to overcome the slow convergence frequently observed in the presence of highly refined grids and flows with high density ratios or with many bubbles. The presence of small eigenvalues has a harmful influence on the convergence of ICCG. These slowly converging components are not cured by classical preconditioning. A significant improvement arises from the removal of the eigenmodes corresponding to these small eigenvalues from the system. This leads to the DICCG method, which is equivalent to ICCG complemented with a deflation technique [21]. The deflation technique has been further exploited by several authors, among them are [1,7,12,16,18,19,20,24,31,32].

*Received July 6, 2006. Accepted for publication June 8, 2007. Recommended by Y. Saad. Part of this research has been funded by the Dutch BSIK/BRICKS project.

†Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics, Mekelweg 4, 2628 CD Delft, The Netherlands ({J.M.Tang, C.Vuik}@tudelft.nl).

In many applications, DICCG has been proven to be an efficient method, such as in the applications of porous media flows [19] and ground water flows [32]. However, in these cases, the interfaces in the domain can be explicitly described and, so, applying the deflation technique is straightforward. Due to the possible appearance of complex geometries in our bubbly flow problems, the interfaces of the bubbles can only be described implicitly in general. Moreover, in the above applications, the coefficient matrix of the linear system is always nonsingular making the deflation technique straightforward to apply. However, in our applications with bubbly flows, we have singular coefficient matrices. Recently, we have investigated theoretical aspects with respect to the singularity of the linear systems for bubbly flow problems [28]. We proposed two equivalent variants of DICCG, where it has been demonstrated, both theoretically and numerically, that the resulting methods accelerate the convergence of the iterative process. Moreover, in practice, the singular coefficient matrix is often made invertible by modifying, for example, the last element, since this can be advantageous for the solver. The drawback of this approach is that the condition number becomes worse. We showed that this problem can be completely remedied by applying the deflation technique with one or more deflation vectors.

In this paper, we continue the research on deflation and the singularity of the linear system. We propose and investigate another variant of DICCG which can deal with the bubbly flow problems. This will be compared to the existing two variants, both theoretically and numerically. The new variant appears to be the most natural analogue of the existing DICCG method for invertible coefficient matrices as known in literature; see, e.g., [7, 19, 31, 32].

The main focus of [28] was to show theoretically that DICCG can be adapted so that it is applicable to singular systems, where 3-D numerical experiments have been carried out without taking efficiency issues into account. The ICCG and DICCG methods have been compared in terms of the number of iterations required for convergence to the solution. However, it is known that a reduction of the number of iterations does not guarantee a reduction of the required computational time, since the work per iteration may increase in the new method. Therefore, in this paper, we present some results investigating both the number of iterations and the required CPU time, in order to show that DICCG is efficient in use. To do so, DICCG has to be implemented efficiently. Hence, in this paper we will also consider some implementation issues in order to obtain an efficient program code.

Finally, in the 3-D numerical experiments done in [28], we used fixed density fields. However, in practice the density fields can change in time. Therefore, in this paper, we will perform some real-life time-dependent simulations with rising bubbles in water and falling droplets in air. We investigate whether DICCG is also effective for these kinds of applications.

Besides DICCG, well-known in the field of multigrid (MG) and domain decomposition methods (DDM) are the CG method in combination with additive coarse grid correction [5, 4] or balancing preconditioners [13, 14]. For elliptic problems with large jumps in the coefficients, successful multigrid solvers and preconditioners can be found in, e.g., [2, 3, 33], whereas appropriate domain decomposition methods and preconditioners have been considered in, e.g., [6, 15, 26, 30]. In this paper, we restrict ourselves to ICCG-type methods. This is because (complexity issues for) DDM and MG preconditioners are still the subject of current research for bubbly flow applications. In addition, ICCG-type methods have already proved to be extremely robust, even in cases with complex density fields containing relatively small bubbles and droplets.

This paper is organized as follows. First, the problem setting of bubbly flows is given in Section 2. Subsequently, in Section 3, we describe briefly the existing DICCG variants and define another DICCG variant for singular coefficient matrices. Moreover, some notes about the algorithm and the efficiency of deflation are made. After that, in Section 4, the DICCG

ETNA
Kent State University
etna@mcs.kent.edu

variants are treated and compared in more detail. In addition, we propose efficient methods to solve the coarse linear systems within these variants. Section 5 is devoted to some 3-D stationary and time-dependent numerical experiments, where the performance of both ICCG and DICCG are investigated, considering both the number of iterations and the computational cost. Finally, the conclusions are presented in Section 6. For more details we refer to [29].

2. Problem setting. We consider the singular symmetric and positive-semi-definite (SPSD) linear system,

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}. \tag{2.1}$$

The linear system (2.1) is derived from a second-order finite-difference discretization of the 3-D Poisson equation with Neumann boundary conditions:

$$\left\{ \begin{array}{rcll} -\nabla \cdot \left(\frac{1}{\rho(\mathbf{x})} \nabla p(\mathbf{x}) \right) & = & f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \frac{\partial}{\partial \mathbf{n}} p(\mathbf{x}) & = & g(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{array} \right. \tag{2.2}$$

where $p, \rho, \mathbf{x}$ and $\mathbf{n}$ denote the pressure, density, spatial coordinates, and the unit normal vector to the boundary, $\partial\Omega$, respectively. In the 3-D case, domain Ω is chosen to be a unit cube. We perform the computations on a uniform Cartesian grid, so that $n = n_x n_y n_z$, where n_x, n_y and n_z are the grid sizes in each spatial direction. Furthermore, we consider two-phase bubbly flows with, for example, air and water. In this case, ρ is piecewise constant with a relatively small contrast:

$$\rho = \left\{ \begin{array}{ll} \rho_0 = 1, & \mathbf{x} \in \Lambda_0, \\ \rho_1 = 10^{-3}, & \mathbf{x} \in \Lambda_1, \end{array} \right.$$

where Λ_0 is water, the main fluid of the flow around the air bubbles, and Λ_1 is the region inside the bubbles. We define the contrast of the problem as $\theta := \rho_1/\rho_0$.

Next, define $\mathbf{1}_p$ and $\mathbf{0}_p$ to be the all-one and all-zero vectors with p elements, respectively. Then, throughout this paper, the following assumption holds, which follows implicitly from the above problem setting.

ASSUMPTION 2.1. *We assume that the singular SPSD matrix, A, and the vector, b, satisfy $A\mathbf{1}_n = \mathbf{0}_n$, and $b^T\mathbf{1}_n = 0$. In addition, the algebraic multiplicity of the zero-eigenvalue of A is equal to one.*

Obviously, from Assumption 2.1, it follows that the system, $Ax = b$, is compatible. Hence, although A is singular, this linear system is always consistent and an infinite number of solutions, x, exist. Additionally, we stress that the results given in this paper cannot be generalized to singular matrices with an algebraic multiplicity of the zero-eigenvalue larger than one.

3. Implementation aspects of DICCG. In this section, we will give three variants of the DICCG method, after defining the deflation matrices. Moreover, the DICCG algorithm will be presented, and some implementation issues with respect to computations with the deflation matrices will be considered.

3.1. Definition of the deflation matrices. In ICCG, the resulting linear system to be solved is $M^{-1}Ax = M^{-1}b$, where M denotes the standard incomplete Cholesky (IC) preconditioner without fill-in; see, e.g., [8, Section 10.3.2]. Although A is singular, it can be shown that the IC preconditioner, M, is invertible [11, Theorem 3.2].

Next, let the open domain Ω be divided into subdomains Ω_j, $j = 1, 2, \ldots, k$, such that $\overline{\Omega} = \cup_{j=1}^{k} \overline{\Omega}_j$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$. The discretized domain and subdomains are denoted by Ω_h and Ω_{h_j}, respectively. Then, for each Ω_{h_j} with $j = 1, 2, \ldots, k$, we introduce a deflation vector, z_j, as follows:

$$(z_j)_i := \begin{cases} 0, & x_i \in \Omega_h \setminus \Omega_{h_j}; \\ 1, & x_i \in \Omega_{h_j}, \end{cases}$$

where x_i is a grid point in the discretized domain, Ω_h. Then, for $k > 1$, we define $Z_k := [z_1 \; z_2 \; \cdots \; z_k]$, which is called the deflation subspace matrix. Hence, Z_k consists of disjunct orthogonal piecewise-constant vectors. In the flop count analysis (Section 3.4) and in the numerical experiments (Section 5), we will take the subdomains, Ω_j, to be identical cubes. Subsequently, we define

$$P_{k-1} := I - AZ_{k-1}E_{k-1}^{-1}Z_{k-1}^T, \quad E_{k-1} := Z_{k-1}^T A Z_{k-1},$$

and

$$\widetilde{P}_k := I - \widetilde{A}Z_k\widetilde{E}_k^{-1}Z_k^T, \quad \widetilde{E}_k := Z_k^T \widetilde{A} Z_k,$$

where the invertible matrix, $\widetilde{A}$, is identical to A except for the last element, which is $\tilde{a}_{n,n} = (1+\sigma)a_{n,n}$ with $\sigma > 0$. It can be noticed that P_{k-1} results from one deflation vector less than $\widetilde{P}_k$. Moreover, note that both coarse matrices E_{k-1} and $\widetilde{E}_k$ are nonsingular, so that the corresponding coarse linear systems can be solved in a direct way.

3.2. Variants of DICCG. In [28], two variants of DICCG were proposed that can deal with singular coefficient matrices. In Variant (a), one solves

$$M^{-1}P_{k-1}A\tilde{x} = M^{-1}P_{k-1}b,$$

and, in Variant (b), one solves

$$\widetilde{M}^{-1}\widetilde{P}_k\widetilde{A}\tilde{x} = \widetilde{M}^{-1}\widetilde{P}_k b,$$

where $\widetilde{M}$ is the IC preconditioner based on $\widetilde{A}$. It has been proven that the equalities, $\widetilde{P}_k\widetilde{A} = P_{k-1}A$ and $\lim_{\sigma\to 0} \kappa_{\text{eff}}(\widetilde{M}^{-1}\widetilde{P}_k\widetilde{A}) = \kappa_{\text{eff}}(M^{-1}P_{k-1}A)$, hold. Both variants require approximately the same computational time for various test problems. Therefore, the two variants are equivalent for $\sigma \ll 1$.

It is common to solve the coarse linear systems, associated to E_{k-1} or $\widetilde{E}_k$, in a direct way; see [28]. However, in principle, these coarse sytems can also be solved iteratively, which can be more efficient if k is relatively large. In this case, Variant (a) is related to another variant of DICCG, called Variant (c), where we solve

$$M^{-1}P_k A\tilde{x} = M^{-1}P_k b,$$

where

$$P_k := I - AZ_kE_k^{+}Z_k^T, \quad E_k := Z_k^T A Z_k,$$

with E_k^+ equal to the Moore-Penrose inverse or pseudo-inverse. The standard inverse cannot be used, since E_k is singular due to

$$E_k\mathbf{1}_k = Z_k^T A Z_k \mathbf{1}_k = Z_k^T A \mathbf{1}_n = Z_k^T \mathbf{0}_n = \mathbf{0}_k, \tag{3.1}$$

using the fact that $Z_k\mathbf{1}_k = \mathbf{1}_n$. Although the inverse of E_k does not exist, it may be possible to use a direct solver for the corresponding coarse linear systems. Extra care should be needed by applying, e.g., Gaussian elimination or the band-Cholesky decomposition, to handle the singularity of E_k. One should generate a solution up to the null space of E_k. However, in this paper, we restrict ourselves to solve the coarse linear systems in Variant (c) iteratively. This does not cause problems, as long as these systems with E_k are consistent, see Section 4.1.2.

In addition, note that Variant (c) is basically identical to the original DICCG for invertible systems (see, e.g., [7, 19, 32]), since the original coefficient matrix and all k deflation vectors are used in this variant. Hence, it is the most natural generalization of DICCG for singular systems. Moreover, in Section 4.2, we will show that Variant (c) is mathematically equal to the other two variants. For the sake of completeness, the three variants are summarized in Table 3.1.

TABLE 3.1
Deflation and coarse matrices of the proposed deflation variants.

Variant	Deflation matrix	Coarse matrix
(a)	P_{k-1}	E_{k-1}
(b)	$\widetilde{P}_k$	$\widetilde{E}_k$
(c)	P_k	E_k

Subsequently, we will distinguish two main DICCG methods in this paper, where the difference is the solver of the coarse systems; see Definition 3.1.

DEFINITION 3.1.

- *DICCG1$-k$ is defined as DICCG corresponding to any deflation variant, where each coarse system is solved directly.*
- *DICCG2$-k$ is defined as DICCG corresponding to any deflation variant, where each coarse system is solved iteratively.*

Notice that any variant, as given in Table 3.1, can be used for both DICCG1$-k$ and DICCG2$-k$, since it appears that all variants are mathematically equivalent. However, for convenience, we restrict ourselves to Variant (a) and (b) for DICCG1$-k$ and Variant (a) and (c) for DICCG2$-k$ in this paper.

3.3. DICCG algorithm. From $\tilde{x}$, we can find solution x using the following expression (see, e.g., [19]), where P and Z are the corresponding deflation matrix and subspace deflation matrix, associated with DICCG1$-k$ or DICCG2$-k$:

$$x = ZE^{-1}Z^Tb + P^T\tilde{x}.$$

The DICCG algorithm is presented in Algorithm 1.
Operations with P in Algorithm 1 need a careful implementation, since the success of DICCG depends strongly on the way of implementing these operations. In the next subsection, we consider this topic in more detail.

Finally, we use the following termination criterion,

$$\frac{||M^{-1}P(b - A\tilde{x}_k)||_2}{||M^{-1}(b - Ax_0)||_2} < \epsilon, \tag{3.2}$$

in DICCG, which is equivalent to the standard termination criterion

$$\frac{||M^{-1}(b - Ax_k)||_2}{||M^{-1}(b - Ax_0)||_2} < \epsilon, \tag{3.3}$$

in ICCG, where $\epsilon > 0$. The left-hand sides of (3.2) and (3.3) are called the norm of the relative residual of DICCG and ICCG, respectively.

ETNA
Kent State University
etna@mcs.kent.edu

Algorithm 1 DICCG Algorithm solving $Ax = b$

choose a starting vector x_0,
choose DICCG1$-k$ or DICCG2$-k$ with a corresponding deflation variant
and matrices A, Z, E, P and M
compute $r_0 := b - Ax_0$, $\tilde{x}_0 := x_0$ and $\hat{r}_0 := Pr_0$,
solve $Mv_0 = \hat{r}_0$ and take $p_0 := v_0$
for $j := 0, \ldots,$ until convergence **do**
  $w_j := PAp_j$
  $\alpha_j := (\hat{r}_j, v_j)/(p_j, w_j)$
  $\tilde{x}_{j+1} := \tilde{x}_j + \alpha_j p_j$
  $\hat{r}_{j+1} := \hat{r}_j - \alpha_j w_j$
  solve $Mv_{j+1} = \hat{r}_{j+1}$
  $\beta_j := (\hat{r}_{j+1}, v_{j+1})/(\hat{r}_j, v_j)$
  $p_{j+1} := v_{j+1} + \beta_j p_j$
end for
$x := ZE^{-1}Z^T b + P^T \tilde{x}_{j+1}$

3.4. Treatment of operations with the deflation matrix. Some implementation analysis for DICCG will be treated in this subsection. Using this analysis, DICCG can be efficiently implemented in a program code, resulting in a fast solver; see also [29] for more details. Note that the main part of this analysis is only valid for 3-D regular grids, and that the flop counts and their analysis only hold for deflation subdomains, which are taken to be non-overlapping identical cubes.

3.4.1. Construction of AZ. The matrix-matrix product AZ can be computed by only determining the non-zero elements, which will be stored as a sparse matrix denoted by S_{AZ}. Denote the number of non-zero elements of the matrix, AZ, by γ. Then, S_{AZ} is a $\gamma \times 3$ matrix, where the first and second columns are filled with the row and column indices of the non-zero elements of AZ, respectively. The third column of S_{AZ} stores the corresponding values of these non-zero elements. The elements of S_{AZ} can be determined efficiently, since Z represents subdomains Ω_j which are non-overlapping cubes. Moreover, AZ has only contributions near the interfaces of these cubes and, hence, consists of relatively many zeros. So, the few nonzero elements of AZ may be known beforehand.

EXAMPLE 3.2. Let $A \in \mathbb{R}^{4\times 4}$ and $Z \in \mathbb{R}^{4\times 2}$ be matrices, obtained from the 1-D Poisson problem, given by

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Then, this leads immediately to

$$AZ = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}, \quad S_{AZ} = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 1 & -1 \\ 2 & 2 & -1 \\ 3 & 2 & 1 \end{bmatrix},$$

where $\gamma = 2$.

In addition, considering the number of floating point operations, it is not difficult to show that constructing S_{AZ} requires $\mathcal{O}(n^{2/3}k^{1/3})$ flops in the 3-D case.

3.4.2. Construction of E. The coarse matrix, $E := Z^T AZ$, can be easily formed during the construction of AZ. Each non-zero element of AZ makes exactly one contribution to E, by simply adding the value to the corresponding element of E.

It depends on the choice of technique for solving the linear system $Ey_2 = y_1$ in which sparse way E is best stored, resulting in S_E; see Section 4. For now, the number of floating point operations to solve such a linear system is denoted by θ.

3.4.3. Calculation of matrix-vector products Py and $P^T y$. In contrast to AZ and E, the deflation matrix, P, is not explicitly constructed. Instead, the matrix-vector product $Py := y - AZE^{-1}Z^T y$ is computed in a sequential way; see Algorithm 2. $P^T y$ can be treated in the same way. Both algorithms require $\mathcal{O}(n + \theta)$ flops.

Algorithm 2 Computing Py

1: $y_1 := Z^T y$
2: solve $Ey_2 = y_1$
3: $y_3 := AZy_2$
4: $Py := y - y_3$

Note that Z is not explicitly stored, since the matrix-vector products, $Z^T y$ and Zy_2, can be simply determined from y, requiring $\mathcal{O}(n)$ flops. Furthermore, since S_{AZ} is known, the products AZy_2 and $(AZ)^T y$ can also be easily computed using $\mathcal{O}(n^{2/3}k^{1/3})$ flops in 3-D.

4. Further analysis of the DICCG variants. In this section, we first show how the coarse systems, $Ey_2 = y_1$, can be solved for both DICCG1$-k$ and DICCG2$-k$. We then compare the deflation variants theoretically.

4.1. Solving the coarse linear system efficiently. Solving the coarse systems, $Ey_2 = y_1$, efficiently is crucial in both DICCG1$-k$ and DICCG2$-k$. Recall that we use a direct method to solve them in DICCG1$-k$, whereas an iterative method is adopted in DICCG2$-k$.

We define k_x to be the number of grid points in each spatial direction of a cube from Z, i.e., $k_x := \sqrt[3]{n/k}$, assuming that k is a divisor of n and that the number of grid points is equal in each spatial direction.

4.1.1. DICCG1$-k$. To solve $Ey_2 = y_1$ with a direct method, we can apply the band-Cholesky decomposition [8, Section 4.3.5] and, thereafter, band-back/forward substitution [8, Section 4.3.2]. In this case, the bandwidth of E is $k^{2/3} + k^{1/3}$ making the decomposition only efficient for relatively small k and/or n.

In the previous section, we noted that both $\widetilde{E}_k$ and E_{k-1} are invertible, so that their band-Cholesky decompositions exist. Considering floating point operations, constructing the Cholesky decomposition requires $\mathcal{O}(k^{7/3})$ flops, whereas the backward and forward substitutions take $\mathcal{O}(k^{5/3})$ flops.

4.1.2. DICCG2$-k$. To find a solution of $Ey_2 = y_1$ in DICCG2$-k$, we will apply the iterative solver ICCG (using the standard IC preconditioner without fill-in, i.e., IC(0)). This is possible and efficient, since E has the same properties as A. Obviously, E is SPSD and has a similar sparsity pattern to A, because Z is based on non-overlapping deflation subdomains. Moreover, E is better conditioned than A; see Theorem 4.1.

THEOREM 4.1. *Suppose E to be E_{k-1} or E_k. Let the eigenvalues of both A and E be sorted increasingly. Then,*

$$\kappa_{eff}(E) \leq \kappa_{eff}(A), \tag{4.1}$$

where κ_{eff} denotes the effective condition number.

Proof. Note first that both E_{k-1} and E_k have rank $k-1$, since Z has full rank and the algebraic multiplicity of the zero-eigenvalue of A is one. In addition, it is allowed to scale Z_k with $\sqrt{n/k}$ such that it satisfies $Z_k^T Z_k = I$. We do not lose generality, since the column space of Z_k remains the same.

In order to prove (4.1) for $E := E_k$, it suffices to show that

$$\lambda_2(A) \leq \mu_2(E_k) \quad \text{and} \quad \mu_k(E_k) \leq \lambda_n(A), \tag{4.2}$$

where $0 = \mu_1(E_k) < \mu_2(E_k) \leq \ldots \leq \mu_k(E_k)$ and $0 = \lambda_1(A) < \lambda_2(A) \leq \ldots \leq \lambda_n(A)$ are the eigenvalues of E_k and A, respectively.

The inequalities (4.2) can be derived from the Courant-Fischer Minimax Theorem; see, e.g., [10, Theorem 4.2.11]. From this theorem, we obtain in particular

$$\lambda_2(A) = \min_{x^T x = 1,\ x \perp u_1(A)} x^T A x, \quad \lambda_n(A) = \max_{x^T x = 1} x^T A x, \tag{4.3}$$

where $u_1(A)$ is the eigenvector corresponding to $\lambda_1(A)$; see [10, Section 4.2] for details.

Note that the identities, $u_1(A) = \mathbf{1}_n$ and $u_1(E) = \mathbf{1}_k$, hold due to Assumption 2.1 and (3.1). In addition, for $x = Z_k y$ we have $x^T A x = (Z_k y)^T A Z_k y = y^T E y$, $(Z_k y)^T (Z_k y) = y^T y$ and $(Z_k y)^T \mathbf{1}_n = y^T Z_k^T \mathbf{1}_n = y^T \mathbf{1}_k$, using the fact that $Z_k^T \mathbf{1}_n = \mathbf{1}_k$. Hence, this implies

$$\min_{(Z_k y)^T (Z_k y) = 1,\ Z_k y \perp \mathbf{1}_n} (Z_k y)^T A (Z_k y) = \min_{y^T y = 1,\ y \perp \mathbf{1}_k} y^T E y. \tag{4.4}$$

Now, combining (4.3) and (4.4) gives us

$$\lambda_2(A) = \min_{x^T x = 1,\ x \perp \mathbf{1}_n} x^T A x \leq \min_{y^T y = 1,\ y \perp \mathbf{1}_k} y^T E y = \mu_2(E),$$

which is the left inequality of (4.2). For the right inequality of (4.2), it follows in a similar way that

$$\mu_k(E) = \max_{y^T y = 1} y^T E y \leq \max_{x^T x = 1} x^T A x = \lambda_n(A),$$

where we have applied $\max_{(Z_k y)^T (Z_k y) = 1} (Z_k y)^T A (Z_k y) = \max_{y^T y = 1} y^T E y$.

Expression (4.1) for $E := E_{k-1}$ can be proven in a similar way as above. □

Next, there is no need to force invertibility of E_k, since ICCG can deal with singular matrices. We only have to ensure the consistency of all coarse linear systems during the process of DICCG2$-k$; see Theorem 4.2.

THEOREM 4.2. *All coarse linear systems within DICCG2$-k$ are consistent.*

Proof. Recall that E_{k-1} is invertible, so that we can restrict ourselves to $E := E_k$. The coarse linear systems $Ey_2 = y_1$ appear three times in Algorithm 1 (Lines 2, 4 and 12) which will be separately treated below.

In the matrix-vector product Pr_0 we have to solve the coarse linear system $Ey_2 = Z^T r_0$. This system is consistent, since it is compatible due to (3.1) and

$$(Z^T r_0)^T \mathbf{1}_k = r_0^T Z \mathbf{1}_k = r_0^T \mathbf{1}_n = b^T \mathbf{1}_n - x_0^T A \mathbf{1}_n = \mathbf{0}_n - x_0^T \mathbf{0}_n = \mathbf{0}_n.$$

Moreover, since

$$(Z^T A p_j)^T \mathbf{1}_k = p_j^T A Z \mathbf{1}_k = p_j^T A \mathbf{1}_n = \mathbf{0}_n,$$

the system $Ey_2 = Z^T Ap_j$ is also compatible. Hence, PAp_j is consistent. Finally, using the same argument as above, we conclude that $P^T\tilde{x}_{j+1}$ is also consistent, since $P^T\tilde{x}_{j+1} = \tilde{x}_{j+1} - ZE^{-1}Z^T A\tilde{x}_{j+1}$. □

From Theorem 4.2, we conclude that it is possible to solve the coarse systems, $Ey_2 = y_1$, iteratively. Each of the ICCG steps costs $\mathcal{O}(k)$ flops, and the efficiency of this method depends on the number of required inner ICCG iterations. Note that the solution of the coarse systems in Variant (c) is not unique, since it is determined up to a constant vector. If y_2 is a solution of $E_k y_2 = y_1$, then $y_2 + \alpha\mathbf{1}_k$ with $\alpha \in \mathbb{R}$ is also a solution. Fortunately, $y_3 := AZ_k(y_2+\alpha\mathbf{1}_k)$ is unique, due to the fact that $AZ_k\mathbf{1}_k = A\mathbf{1}_n = \mathbf{0}_n$. Hence, Algorithm 2 gives a unique Py for all deflation variants.

Next, in DICCG2$-k$ we have an inner-outer iterative process with DICCG and ICCG, so we need two different termination criteria. The inner and outer tolerances are called ϵ_{out} and ϵ_{in}, respectively. We will take

$$\epsilon_{in} = \omega\ \epsilon_{out}, \quad \omega > 0.$$

For large $\omega \geq 1$, DICCG2$-k$ does not converge, since the method appears to be sensitive to inaccurate solves of the coarse systems; see also [19, Section 3]. However, for small $\omega \ll 1$, the convergence of the inner iterations of DICCG2$-k$ is relatively slow or the inner solve may stagnate or even diverge due to a too severe tolerance. Therefore, ω should be chosen carefully to obtain an accurate and efficient method. From our numerical experiments with bubbly flows, it appears that

$$\omega = 10^{-2}$$

is an appropriate choice. We refer to [9, 25] for more information about inner-outer iterative processes and their termination criteria.

For large problems, it could be advantageous to solve the inner iterations with DICCG as well, instead of ICCG. The inner iterations could even be solved by a recursive application of DICCG. This is in analogy with multigrid; see also [7, Section 3].

4.2. Theoretical comparison of the DICCG variants. DICCG variants (a) and (b) have already been compared in [28]. Since they appear to be equivalent, we can restrict the theoretical comparison to Variant (a) and (c), in this subsection. We will show theoretically that both variants are the same, since it can be proven that the preconditioned deflated system $M^{-1}P_{k-1}A$ in Variant (a) is equal to $M^{-1}P_kA$ in Variant (c); see Theorem 4.3.

THEOREM 4.3. *The following identity holds:*

$$M^{-1}P_kA = M^{-1}P_{k-1}A. \tag{4.5}$$

Proof. Define first $\mathbf{0}_{p,q}$ and $\mathbf{1}_{p,q}$ as the all-zero and all-one $p \times q$ matrices, respectively. Moreover, suppose that $Z_k = [Z_{k-1}\ \mathbf{1}_n]$ instead of $Z_k = [Z_{k-1}\ z_k]$. Now, it suffices to show that $P_k - P_{k-1} = \mathbf{0}_{n,n}$. Note that

$$Z_k^T AZ_k = \begin{bmatrix} Z_{k-1}^T \\ \mathbf{1}_n^T \end{bmatrix} A \begin{bmatrix} Z_{k-1} & \mathbf{1}_n \end{bmatrix} = \begin{bmatrix} Z_{k-1}^T AZ_{k-1}^T & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{bmatrix},$$

leading to

$$E_k^+ = (Z_k^T AZ_k)^+ = \begin{bmatrix} (Z_{k-1}^T AZ_{k-1}^T)^{-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{bmatrix} = \begin{bmatrix} E_{k-1}^{-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{bmatrix}.$$

ETNA
Kent State University
etna@mcs.kent.edu

Hence, this yields

$$\begin{aligned} P_k - P_{k-1} &= AZ_{k-1}E_{k-1}^{-1}Z_{k-1}^T - AZ_kE_k^{+}Z_k^T \\ &= AZ_{k-1}E_{k-1}^{-1}Z_{k-1}^T - AZ_k \begin{bmatrix} E_{k-1}^{-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{bmatrix} Z_k^T \\ &= AZ_{k-1}E_{k-1}^{-1}Z_{k-1}^T - AZ_{k-1}E_{k-1}^{-1}Z_{k-1}^T \\ &= \mathbf{0}_{n,n}, \end{aligned}$$

which completes the proof for $Z_k = [Z_{k-1}\ \mathbf{1}_n]$.

From Theorem 2 of [28], we obtain the equality $P_{k-1}A = \widetilde{P}_k\widetilde{A}$. Therefore, [19, Lemma 2.9] can be applied, which states that both $Z_k = [Z_{k-1}\ \mathbf{1}_n]$ and $Z_k = [Z_{k-1}\ z_k]$ would lead to exactly the same matrix $\widetilde{P}_k\widetilde{A}$, since the column space of $[Z_{k-1}\ z_k]$ and $[Z_{k-1}\ \mathbf{1}_n]$ are equal. Hence, (4.5) also holds for the original $Z_k = [Z_{k-1}\ z_k]$. [1] □

According to Theorem 4.3, the preconditioned deflated systems for A with $k-1$ and k deflation vectors are the same. Hence, Variant (a) and Variant (c) give exactly the same convergence results in exact arithmetic. This conclusion is highly expected and seems quite natural.

We end this section with Corollary 4.4, which is a generalization of Theorem 2.12 of [19].

COROLLARY 4.4. *Let $s, t \in \mathbb{N}$ with $s < t \le k$. Then,*

$$\begin{aligned} \lambda_n(M^{-1}P_sA) &\geq \lambda_n(M^{-1}P_tA); \\ \lambda_{s+1}(M^{-1}P_sA) &\leq \lambda_{t+1}(M^{-1}P_tA). \end{aligned}$$

Moreover,

$$\kappa_{eff}(M^{-1}P_sA) \geq \kappa_{eff}(M^{-1}P_tA),$$

where κ_{eff} denotes the effective condition number.

Proof. Recall that $M^{-1}P_kA = M^{-1}P_{k-1}A = M^{-1}\widetilde{P}_k\widetilde{A}$, by combining Theorem 4.3 and [28, Theorem 2]. Therefore, the systems $M^{-1}P_sA$ and $M^{-1}P_tA$ can be transformed into systems with an invertible $\widetilde{A}$. Then the corollary follows immediately from Theorem 2.12 of [19]. □

Corollary 4.4 states that the effective condition number of the deflated preconditioned system corresponding to the singular matrix A decreases if we increase the number of deflation vectors. This means that, theoretically, the more deflation vectors, the faster the convergence of the deflation method in terms of the number of iterations, although more work is needed to solve the coarse systems.

5. Numerical experiments.

In this section, we present the results for some 3-D numerical experiments with both stationary and time-dependent problems, which will illustrate the theoretical results obtained in the previous sections. The computations are performed on a Pentium 4 (2.80 GHz) computer with a memory capacity of 1GB. Moreover, the code is compiled with FORTRAN g77 on LINUX.

5.1. Stationary experiments.

We apply the problem setting as given in Section 2. Four test problems are considered, with 0, 1, 8 and 27 air-bubbles, respectively, in the unit domain filled with water. These bubbles have the form of a sphere with a constant radius. Most of the results in this subsection are, however, based on the test problem with 27 bubbles. We will vary the contrast, θ, between phases. In addition, various numbers of deflation vectors, k, and various grid sizes, n_x, n_y and n_z with $n_x = n_y = n_z$, will be used in the experiments.

[1] An alternative proof of this theorem can be found in [29, Section 6.3].

ETNA
Kent State University
etna@mcs.kent.edu

First, ICCG and DICCG1$-k$ are used to solve the resulting linear systems. Then, in the last subsection, we will compare DICCG1$-k$ and DICCG2$-k$. Deflation variant (a) in DICCG1$-k$ and deflation variant (c) in DICCG2$-k$ are applied, throughout the experiments. Random starting vectors are used, and we choose a relative termination criterion with tolerance $\epsilon = 10^{-8}$. As a measure of the accuracy of the solutions, the exact relative residuals were also investigated in the experiments. However, since it appeared that they are comparable for ICCG and for both DICCG variants in all cases, these results are omitted.

5.1.1. Results with $n = 100^3$. The results for computational time and number of iterations for all test problems with grid size $n = 100^3$ and contrast $\theta = 10^{-3}$ are given in Table 5.1.

TABLE 5.1
Convergence results for ICCG and DICCG1$-k$ for all test problems with $n = 100^3$ and $\theta = 10^{-3}$. '# It' means the number of required iterations for convergence, and 'CPU' means the corresponding computational time in seconds.

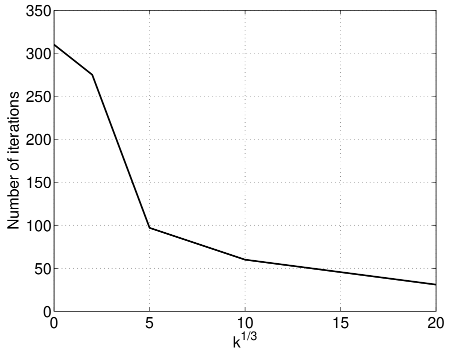
Test Problem	No Bubbles		One Bubble		Eight Bubbles		27 Bubbles	
Method	# It.	CPU	# It.	CPU	# It.	CPU	# It.	CPU
ICCG	170	25.2	211	31.1	291	43.0	310	46.0
DICCG1-2^3	109	20.2	206	37.5	160	29.1	275	50.4
DICCG1-5^3	56	11.3	58	11.5	72	14.2	97	19.0
DICCG1-10^3	35	8.0	36	8.5	36	8.2	60	13.0
DICCG1-20^3	22	26.5	25	27.6	22	27.2	31	29.3

Considering the results in Table 5.1, we see that DICCG1$-k$ always requires fewer iterations when compared to ICCG. It can be observed that, for larger k, DICCG1$-k$ requires fewer iterations than for smaller k, which is in agreement with Corollary 4.4. In addition, the optimal choice with respect to the CPU time is $k = 10^3$, i.e., in all test cases DICCG1-10^3 converges most rapidly. The improvement in the CPU time is relatively large compared to ICCG. Furthermore, one can notice that, in general, it is not always the case that more bubbles in the test problem require more iterations and, therefore, more CPU time for both ICCG and DICCG1$-k$ to converge. Namely, for DICCG1-2^3 and DICCG1-20^3, we observe that fewer iterations and CPU time are required for eight bubbles than for one single bubble. Finally, notice that for large k, DICCG1$-k$ requires significant CPU time due to the increase of the computational cost for solving systems with E. Hence, DICCG1$-k$ does converge with a small number of iterations for $k > 10^3$, but it requires a lot of CPU time for each iteration.

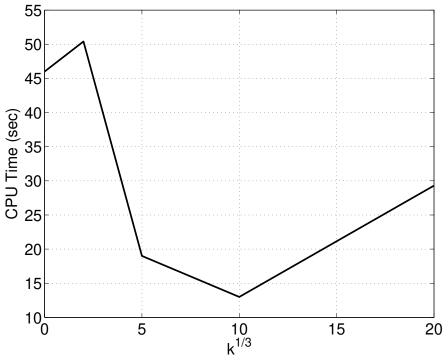
To visualize the results given in Table 5.1, we present those for the test problem with 27 bubbles in Figure 5.1. From Figure 5.1(a) it can be observed that as k is increased, the number of iterations of DICCG1$-k$ decreases significantly at the left of the curve. For large k, the benefit is smaller. Furthermore, after a peak at $k = 2^3$, the required CPU time for DICCG1$-k$ decreases until $k = 10^3$. Thereafter, the CPU time increases and DICCG1$-k$ is less efficient; see Figure 5.1(b). In Figure 5.1(c), the benefit factor for the number of iterations is depicted for each k. Obviously, the larger k, the larger the profit of DICCG1$-k$. For example, in the case of $k = 20^3$, DICCG1$-k$ requires almost 10 times fewer iterations than ICCG. Finally, in Figure 5.1(d) the benefit factor for the CPU time is depicted for each k. For $k = 10^3$, the maximum benefit factor is achieved. In this case, DICCG1-10^3 is around 3.5 times faster than ICCG.

Finally, for the same test problem with 27 bubbles, plots of the norms of the relative residuals of both ICCG and DICCG1$-k$, as defined in Section 3.3, can be found in Figure 5.2. Notice that the behavior of the relative residuals of ICCG are somewhat irregular, due to the

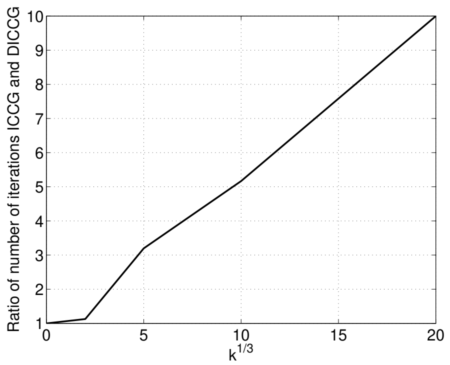
ETNA
Kent State University
etna@mcs.kent.edu



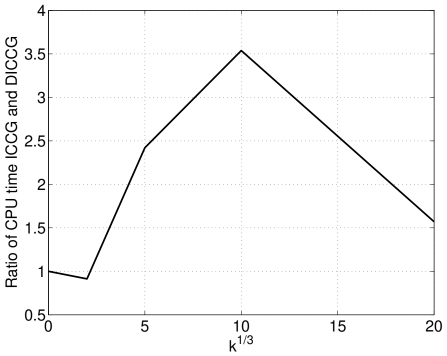
(a) Number of iterations versus $k^{1/3}$.



(b) CPU time versus $k^{1/3}$.



(c) Ratio of numbers of ICCG and DICCG1$-k$ iterations versus $k^{1/3}$.



(d) Ratio of CPU time required for ICCG and DICCG1$-k$ versus $k^{1/3}$.

FIG. 5.1. *Visualization of the results for the test problem with 27 bubbles.*

presence of the bubbles. For DICCG1$-k$ we can conclude that the larger k, the smoother the residual plot is and so, the faster the convergence of the iterative process.

5.1.2. Results for varying grid sizes and contrasts. The results for the test problem with 27 bubbles and with varying grid sizes are given in Figure 5.3. Here, we use $\psi := n_x/k_x$ as the ratio of the grid size and the number of deflation vectors, both in one spatial direction. We investigate whether DICCG1$-k$ is scalable, i.e., whether the number of iterations of DICCG1$-k$ is equal for all k and for a fixed ψ.

From the figure, one observes immediately that for larger grid sizes, the differences in performance between ICCG and DICCG1$-k$ become significantly larger. For instance, in the case of $n = 100^3$, ICCG converges in 275 iterations and 50.4 seconds, while DICCG1-10^3 finds the solution in 60 iterations and only in 13.0 seconds. Moreover, we notice that the number of ICCG iterations grows with the grid sizes, while the number of iterations for DICCG1$-k$ for both $\psi = 5$ and $\psi = 10$ remains approximately constant. It seems that in order to keep the number of iterations constant in DICCG1$-k$ as the grid size is increased, then the number of deflation vectors must also increase, proportionally to the grid sizes.

ETNA
Kent State University
etna@mcs.kent.edu

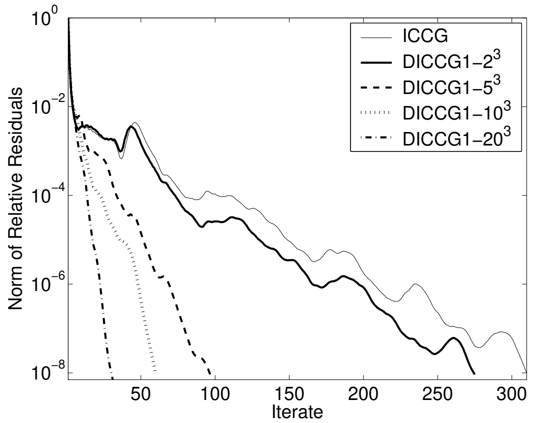
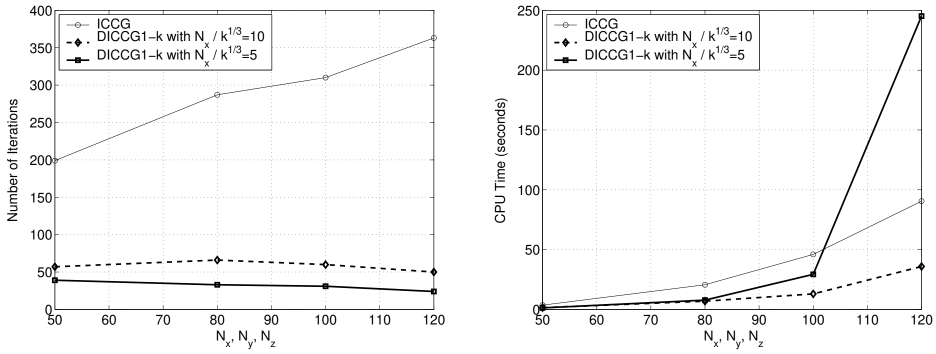


FIG. 5.2. *Relative residual plots for ICCG and DICCG1−k for the test problem with 27 bubbles and various k.*



(a) Number of iterations versus grid size per direction.

(b) CPU time versus grid size per direction.

FIG. 5.3. *Results for the test problem with 27 bubbles for varying grid sizes. ICCG and DICCG1−k with both $\psi = 5$ and $\psi = 10$ are given.*

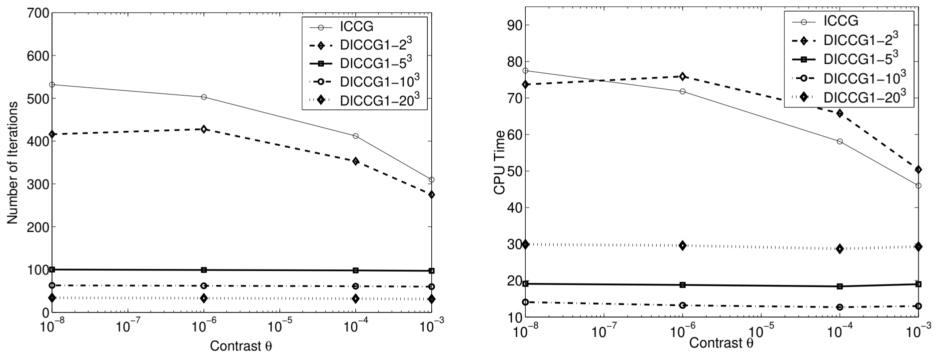
Moreover, the CPU time required for DICCG1−k increases more or less quadratically with grid size, which is a consequence of the expensive direct solve of the coarse linear systems. This is in agreement with the theory; cf. Section 4.1.1. In the next subsection, this will be remedied by using DICCG2−k instead of DICCG1−k.

Next, after experiments with varying grid sizes, we fix the grid size as $n = 100^3$ and vary the contrast, θ, between the phases. The results are given in Figure 5.4.

From the figure, we see that DICCG1−k for $k > 2^3$ hardly depends on the contrast, θ, while ICCG becomes obviously worse if we choose a smaller θ. In other words, DICCG1−k with $k > 2^3$ is insensitive to the contrast, θ, in terms of both the number of iterations and the CPU time.

5.1.3. Comparison of DICCG1−k and DICCG2−k. In the previous subsections, we have seen that DICCG1−k is very efficient as long as $k < 20^3$. From $k = 20^3$, DICCG1−k requires too large of the computational cost per iterate, although only a relatively low number

ETNA
Kent State University
etna@mcs.kent.edu



(a) Number of iterations versus contrast between the phases.

(b) CPU time versus contrast between the phases.

FIG. 5.4. *Results for the test problem with 27 bubbles for varying contrast θ.*

of iterations is needed. The bottleneck is the expensive construction of the banded Cholesky decomposition of E. Direct computations with E can be avoided by using DICCG2$-k$, hopefully resulting in a fast solver for large k. In this subsection, a numerical comparison between DICCG1$-k$ and DICCG2$-k$ will be made. Note that since we fix $\epsilon_{out} = 10^{-8}$ for all test cases, the stopping tolerance, $\epsilon_{in} = 10^{-10}$, should be adopted for the inner iterations in DICCG2$-k$, as mentioned in Section 4.1.2.

Some results for the test problem with 27 bubbles and varying grid sizes can be found in Figure 5.5. Similar results have been found for the other test problems. It appears that the number of iterations required for both DICCG1$-k$ and DICCG2$-k$ is more or less equal in all test cases, in agreement with Theorem 4.3, so these results are omitted.

We notice that for a relatively small number of deflation vectors, DICCG1$-k$ and DICCG2$-k$ perform approximately the same. However, for problems with relatively large k, DICCG2$-k$ is clearly more efficient. The differences between the two DICCG methods becomes significant at $k = 20^3$. In addition, observe that in all cases DICCG1$-k$ achieves its optimum at $k = 10^3$, whereas the optimum of DICCG2$-k$ is achieved for $k > 10^3$. Hence, we conclude that DICCG2$-k$ is the most efficient method for $k > 10^3$. This conclusion is rather natural, but cannot be drawn beforehand. For example, unforeseen problems solving the coarse systems may occur, since the precise efficiency of solving these systems is not known exactly, and the consistency of these systems may be lost due to rounding errors.

5.2. Time-dependent experiments. In the previous subsection we have considered test problems with fixed geometries, i.e., the bubbles were fixed in the computational domain and did not evolve in time. In this subsection, we apply two realistic 3-D simulations of 250 time steps. In the first simulation, an air bubble is rising in water. In the second simulation, a water droplet is falling in the air.

We adopt the mass-conserving level-set method [22] for the simulations, but it could be replaced by any operator-splitting method in general. At each time step, a Poisson problem (2.2) has to be solved, which is the most time-consuming part of the whole simulation. Therefore, in this section we concentrate on this part of each time step. We investigate whether both DICCG methods are still efficient for these time steps. Again, deflation variant (a) in DICCG1$-k$ and deflation variant (c) in DICCG2$-k$ are applied.

ETNA
Kent State University
etna@mcs.kent.edu

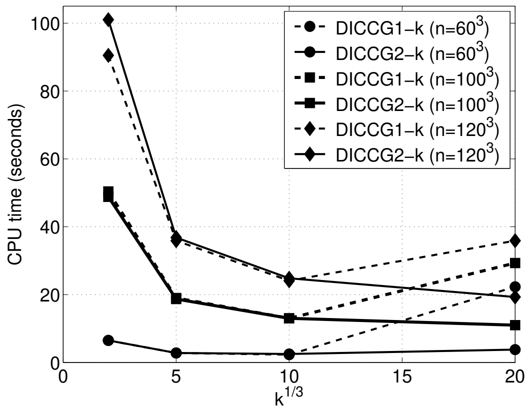
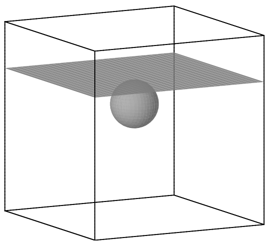
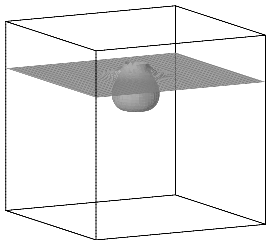


FIG. 5.5. *CPU Time of DICCG1−k and DICCG2−k for the test problem with 27 bubbles with various grid sizes.*

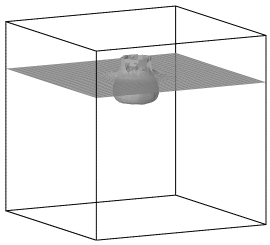
5.2.1. Rising air bubble in water. We consider a test problem with a rising air bubble in water without surface tension. The exact material constants and other relevant conditions can be found in [22, Section 8.3.2]. The starting position of the bubble in the domain and the evolution of its movement during the 250 time steps can be seen in Figure 5.6.



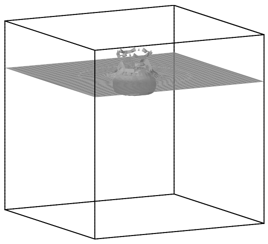
(a) $t = 0$.



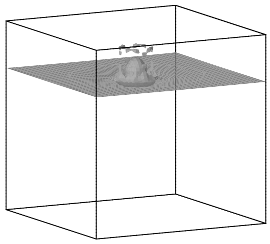
(b) $t = 50$.



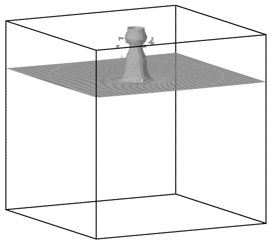
(c) $t = 100$.



(d) $t = 150$.



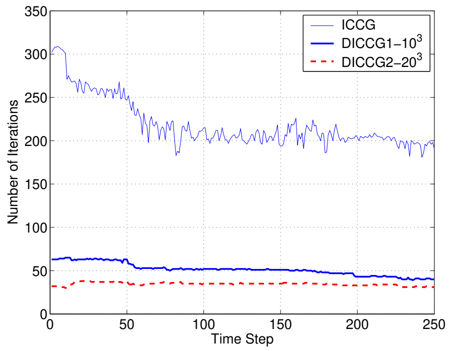
(e) $t = 200$.



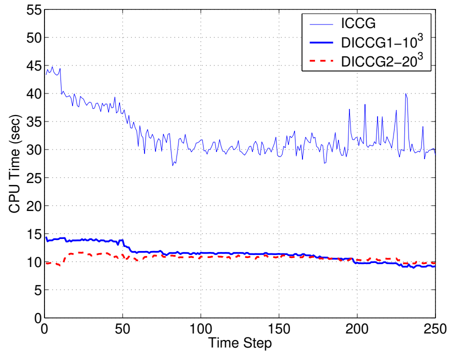
(f) $t = 250$.

FIG. 5.6. *Evolution of the rising bubble in water without surface tension in the first 250 time steps.*

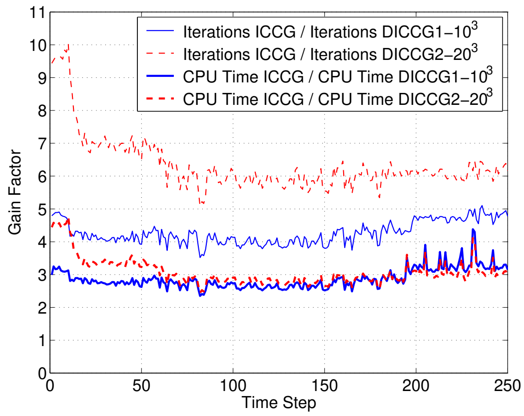
In [22], the Poisson solver is based on ICCG. Here, we will compare this method to both DICCG1$-k$ and DICCG2$-k$ for $n = 100^3$. It turns out that DICCG1-10^3 and DICCG2-20^3 are the best methods in the sense that they need the lowest computational time to perform the simulation (cf. Figure 5.5). The results are presented in Figure 5.7.



(a) Number of iterations versus time step.



(b) CPU time versus time step.



(c) Gain factors of DICCG1 -10^3 and DICCG2 -20^3 with respect to ICCG.

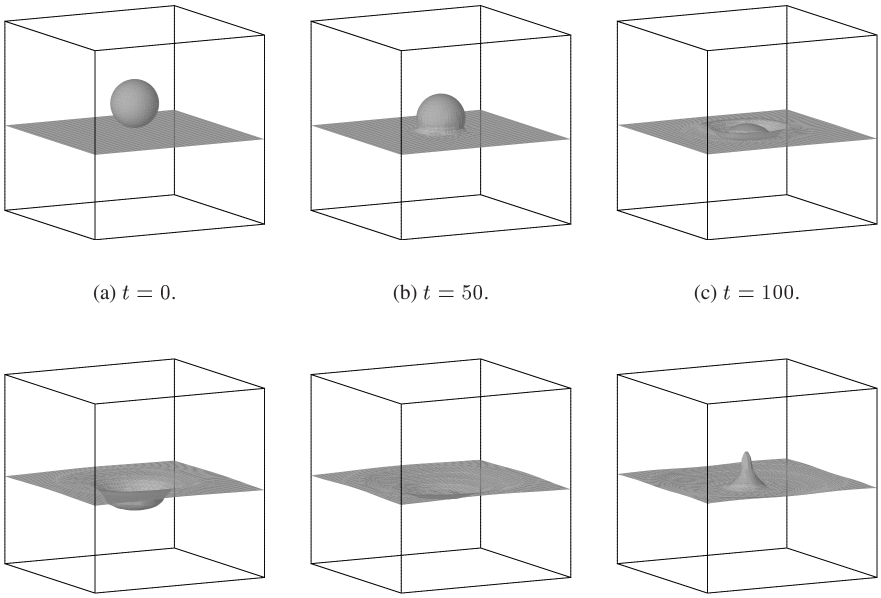
FIG. 5.7. *Results for ICCG, DICCG1* -10^3 *and DICCG2* -20^3 *for the simulation with a rising air bubble in water.*

From Figure 5.7(a), we notice that the number of iterations is strongly reduced by the deflation method. DICCG1 -10^3 and DICCG2 -20^3 require at most 60 iterations, while ICCG converges in between 200 and 300 iterations for most time steps. For each time step, DICCG2 -20^3 requires fewer iterations than DICCG1 -10^3, which is in agreement with Corollary 4.4. Moreover, we observe the erratic behavior of ICCG, whereas the deflation methods seem to be less sensitive to the geometries of the bubbles, during the evolution of the simulation. Considering CPU time, DICCG1 -10^3 and DICCG2 -20^3 also show very good performance, see Figure 5.7(b). For most time steps, ICCG requires 25–45 seconds to converge, whereas both deflation methods are comparable and only need around 9–14 seconds. Moreover, in Figure 5.7(c), one can find the gain factors for both the ratios of the iterations and the CPU time between ICCG and the two deflation methods, respectively. From this figure, we conclude that DICCG1 -10^3 or DICCG2 -20^3 need approximately 4–8 times fewer iterations, depending on the time step. More importantly, at all time steps both deflation methods converge approximately 2–4 times faster than ICCG.

ETNA
Kent State University
etna@mcs.kent.edu

We end this subsection with the remark that similar results can be found for other choices of grid sizes. It appears that for problems with larger grid sizes, the deflation method becomes more favorable, when compared to ICCG.

5.2.2. Falling water droplet in air. Similar to the previous subsection, we consider a test problem with a falling water droplet in air without surface tension; see [22, Section 8.3.4]. Again, DICCG1-10^3 and DICCG2-20^3 are the fastest methods for the simulation. The results are presented in Figure 5.9.



(a) $t = 0$. (b) $t = 50$. (c) $t = 100$.

(d) $t = 150$. (e) $t = 200$. (f) $t = 250$.

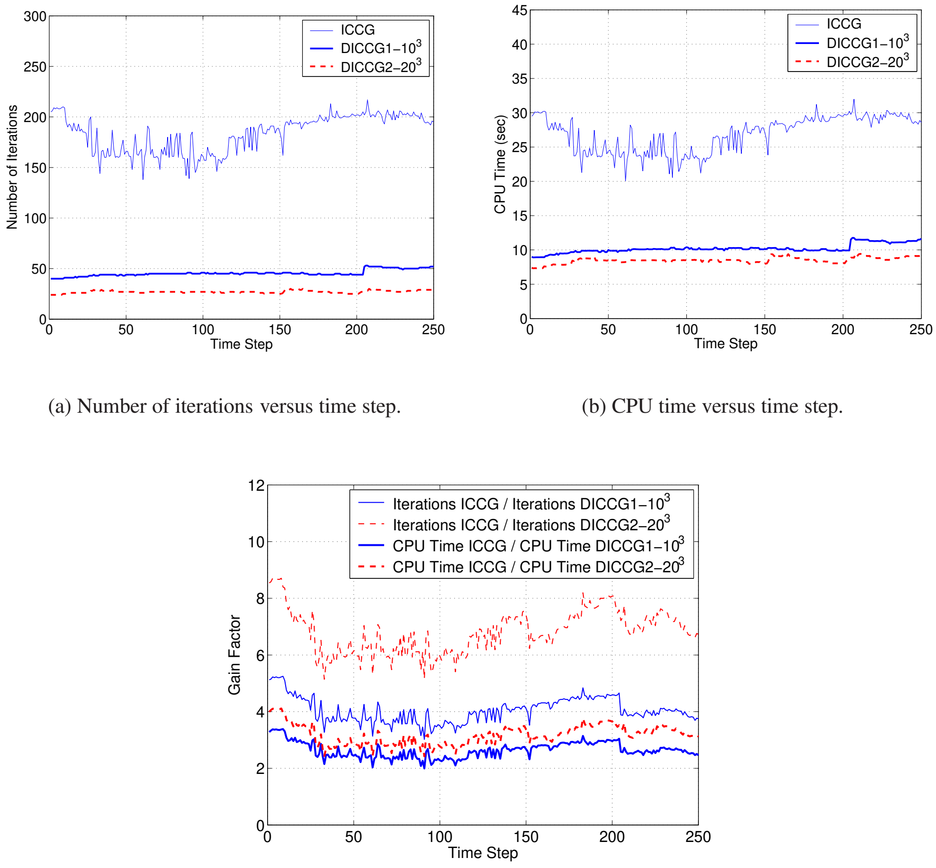
FIG. 5.8. *Evolution of the falling droplet in air without surface tension in the first 250 time steps.*

Similar observations as those in the previous subsection can be drawn from Figure 5.9. Obviously, the deflation method is a more efficient method, when compared with ICCG, in terms of both number of iterations and required CPU time. We observe that both DICCG1-10^3 and DICCG2-20^3 need approximately 3–5 times fewer iterations and they converge more or less 2–4 times faster than ICCG. In this test problem, it can be observed that DICCG2-20^2 performs somewhat better than DICCG1-10^3.

Finally, a small jump in the DICCG1-10^3 performance can be noticed around the 205th timestep in Figure 5.9. This might be the result of the appearance of an rising droplet, which can also be observed in Figures 5.8(e) and 5.8(f). This jump is not significant in DICCG2-20^3. Apparently, a larger set of deflation vectors k gets rid of that droplet.

6. Conclusions. In the literature, the deflation method DICCG has already been proven to be efficient for invertible linear systems. In [28], it has been shown that DICCG can be easily adapted so that it is also applicable for singular linear systems. Additionally, the method appeared to be efficient for bubbly flow problems when measured by the required number of iterations. In this paper, we have shown that computational time is also gained by applying DICCG instead of ICCG, if an efficient implementation has been used.

ETNA
Kent State University
etna@mcs.kent.edu



(a) Number of iterations versus time step.

(b) CPU time versus time step.

(c) Gain factors of DICCG1 -10^3 and DICCG2 -20^3 with respect to ICC G.

FIG. 5.9. *Results for ICCG, DICCG1* -10^3 *and DICCG2* -20^3 *for the simulation with a falling water droplet in air.*

Three variants of DICCG are proposed, where we have proven that all of them give the same convergence results in exact arithmetic, since their preconditioned deflated systems are identical. Furthermore, the involved coarse linear systems can be solved both directly and iteratively. The resulting DICCG methods are denoted by DICCG1$-k$ and DICCG2$-k$, which only differ in the implementation of the coarse solvers. A direct coarse solver is adopted in DICCG1$-k$, whereas an iterative coarse solver is applied in DICCG2$-k$. Some theoretical properties of these coarse systems have been derived, which are of importance to DICCG2$-k$.

Several 3-D numerical experiments for bubbly flow problems have been performed to test the efficiency of both DICCG methods. For relatively small numbers of deflation vectors, DICCG1$-k$ performs very well, but DICCG2$-k$ is more efficient for larger grid sizes and/or number of deflation vectors. Compared with ICCG, both methods significantly reduce the computational cost in all test cases, especially for large problems. Additionally, DICCG is insensitive to the contrasts between the phases, while ICCG has difficulties for large contrasts. Furthermore, we have shown that DICCG is scalable in terms of iterations and CPU time, as long as the number of deflation vectors is chosen proportional to the grid size.

Finally, the success of the deflation method has been emphasized in realistic simulations with a falling droplet in air and a rising bubble in water. Compared to ICCG, the benefit of the deflation method is obviously observed, in terms of both the number of iterations and the CPU time.

Acknowledgments. We would like to thank the anonymous referees for their valuable remarks and comments that enabled us to substantially improve this paper. Moreover, we also thank Kees Oosterlee and especially Scott MacLachlan for the helpful discussions and comments on the paper and thorough proof-reading of this manuscript.

REFERENCES

[1] A. CHAPMAN AND Y. SAAD, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[2] A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, G. N. MIRANDA, AND J. W. RUGE, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1064–8275.

[3] J. E. DENDY, JR., *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[4] M. DRYJA, *An additive Schwarz algorithm for two and three dimensional finite element elliptic problems*, in Domain Decomposition Methods, T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., SIAM, Philadelphia, PA, 1989, pp. 168–172.

[5] M. DRYJA AND O. B. WIDLUND, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., SIAM, Philadelphia, PA, 1990, pp. 3–21.

[6] M. DRYJA AND O. B. WIDLUND, *Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems*, Comm. Pure Appl. Math., 48 (1995), pp. 121–155.

[7] J. FRANK AND C. VUIK, *On the construction of deflation-based preconditioners*, SIAM J. Sci. Comput., 23 (2001), pp. 442–462.

[8] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Third ed., The John Hopkins University Press, Baltimore, 1996.

[9] G. H. GOLUB AND Q. YE, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput. 21 (1999), pp. 1305–1320.

[10] R. HORN AND C. JOHNSON, *Matrix Analysis*, Cambridge University Press, USA Edition, 1990.

[11] E. F. KAASSCHIETER, *Preconditioned conjugate gradients for solving singular systems*, J. Comput. Appl. Math., 24 (1988), pp. 265–275.

[12] L. YU. KOLOTILINA, *Preconditioning of systems of linear algebraic equations by means of twofold deflation, I. Theory*, J. Math. Sci., 89 (1998), pp. 1652–1689.

[13] J. MANDEL, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[14] J. MANDEL, *Hybrid domain decomposition with unstructured subdomains*, in Domain Decomposition Methods in Science and Engineering, Sixth International Conference on Domain Decomposition, A. Quarteroni, J. Periaux, Y. A. Kuznetsov, and O. B. Widlund, eds., Contemp. Math., vol. 157, Amer. Math. Soc., Providence, RI, 1994, pp. 103–112.

[15] J. MANDEL AND M. BREZINA, *Balancing domain decomposition for problems with large jumps in coefficients*, Math. Comp., 216 (1996), pp. 1387–1401.

[16] L. MANSFIELD, *Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1314–1323.

[17] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[18] R. B. MORGAN, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[19] R. NABBEN AND C. VUIK, *A comparison of deflation and coarse grid correction applied to porous media flow*, SIAM J. Numer. Anal., 42 (2004), pp. 1631–1647.

[20] R. NABBEN AND C. VUIK, *A comparison of deflation and the balancing preconditioner*, SIAM J. Sci. Comput., 27 (2006), pp. 1742–1759.

[21] R. A. NICOLAIDES, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM J. Matrix Anal. Appl., 24 (1987), pp. 355–365.

[22] S. P. VAN DER PIJL, *Computation of bubbly flows with a mass-conserving level-set method*, PhD thesis, Delft University of Technology, 2005.

[23] S. P. VAN DER PIJL, A. SEGAL, C. VUIK, AND P. WESSELING, *A mass-conserving Level-Set method for modelling of multi-phase flows*, Int. J. Numer. Meth. Fluids, 47 (2005), pp. 339–361.
[24] Y. SAAD, M. YEUNG, J. ERHEL, AND F. GUYOMARC'H, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926.
[25] V. SIMONCINI AND D. B. SZYLD, *On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods*, SIAM Rev., 25 (2005), pp. 247–272.
[26] B. SMITH, P. BJØRSTAD, AND W. GROPP, *Domain Decomposition*, Cambridge University Press, Cambridge, 1996.
[27] F. S. SOUSA, N. MANGIAVACCHI, L. G. NONATO, A. CASTELO, M. F. TOME, V. G. FERREIRA, J. A. CUMINATO AND S. MCKEE, *A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces*, J. Comput. Phys., 198 (2004), pp. 469–499.
[28] J. M. TANG AND C. VUIK, *On deflation and singular symmetric positive semi-definite matrices*, J. Comput. Appl. Math., 206 (2007), pp. 603–614.
[29] J. M. TANG AND C. VUIK, *An efficient deflation method applied on 2-D and 3-D bubbly flow problems*, DUT Report 06-01, Delft University of Technology, 2006.
[30] A. TOSELLI AND O. B. WIDLUND. *Domain Decomposition: Algorithms and Theory*, Computational Mathematics, 34, Springer, Berlin, 2005.
[31] C. VUIK, A. SEGAL, AND J. A. MEIJERINK, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, J. Comput. Phys., 152 (1999), pp. 385–403.
[32] C. VUIK, A. SEGAL, J. A. MEIJERINK, AND G. T. WIJMA, *The construction of projection vectors for a deflated ICCG method applied to problems with extreme contrasts in the coefficients*, J. Comput. Phys., 172 (2001), pp. 426–450.
[33] P. M. DE ZEEUW, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.