Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2010) 249–257
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

The Deflated Relaxed Incomplete Cholesky CG method for use in a real-time ship simulator

E. van 't Wout[a], M.B. van Gijzen[a], A. Ditzel[b], A. van der Ploeg[b], C. Vuik[a]

[a] *Delft University of Technology, Delft Institute of Applied Mathematics, Delft, The Netherlands*
[b] *MARIN, Maritime Research Institute Netherlands, Wageningen, The Netherlands*

Abstract

Ship simulators are used for training purposes and therefore have to calculate realistic wave patterns around the moving ship in real time. We consider a wave model that is based on the variational Boussinesq formulation, which results in a set of partial differential equations. Discretization of these equations gives a large system of linear equations, that has to be solved each time-step. The requirement of real-time simulations necessitates a fast linear solver. In this paper we study the combination of the Relaxed Incomplete Cholesky preconditioner and subdomain deflation to accelerate the Conjugate Gradient method. We show that the success of this approach depends on the relaxation parameter. For low values of the relaxation parameter, e.g. the standard IC preconditioner, the deflation method is quite successfull. This is not the case for large values of the relaxation parameter, such as the Modified IC preconditioner. We give a theoretical explanation for this difference by considering the spectrum of the preconditioned and deflated matrices. Computational results for the wave model illustrate the expected convergence behavior of the Deflated Relaxed Incomplete Cholesky CG method. We also present promising results for the combination of the deflation method and the inherently parallel block-RIC preconditioner.

Keywords: Conjugate Gradient method, Relaxed Incomplete Cholesky preconditioner, Subdomain deflation, real-time wave model

1. Introduction

The Maritime Research Institute Netherlands serves the maritime industry with innovative products. One of the products MARIN supplies is a ship manoeuvring simulator. These real-time navigation simulators are used for research, consultancy and training purposes.

The current wave model of the ship simulator is based on a predefined wave spectrum and only partially interacts with objects. A new wave model is under development, which depends on the motions of a ship and the bathymetry, resulting in realistic wave patterns. To fullfill the requirement of real-time calculations, the computational methods have to be very efficient.

A considerable amount of computation time is used by the linear solver in the wave model. Three linear solvers were implemented in the model: the Conjugate Gradient method combined with diagonal scaling, the Repeated

Email address: E.vantWout@tudelft.nl (E. van 't Wout)

1877-0509 © 2010 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.04.028

Red-Black preconditioner, and the Modified Incomplete Cholesky preconditioner [1]. The RRB preconditioner uses a recursive red-black ordering of nodes and by permitting some extra fill-in in the last block, the spectral condition number has a smaller order than for the MIC preconditioner [2, 3]. In this paper we study possible enhancements of the MIC preconditioner. We focus on the combination of the Relaxed Incomplete Cholesky preconditioner and subdomain deflation. The deflation method reduces the spectral condition number and can be parallelized efficiently [4].

2. Problem formulation

The description of the water waves, to be used in the ship simulator, is based on a variational Boussinesq formulation. This model has been developed by Klopman [5, 6] and is described in detail in [1].

The fluid motion is assumed to be inviscid and incompressible. Starting point of the wave model is therefore the Euler equations. By integrating these equations over the whole domain, one obtains an expression for the total pressure inside the fluid in terms of the water level and the velocity potential. Minimizing the pressure functional with respect to these two variables gives a variational description of the model. The resulting nonlinear model has been simplified by writing the velocity potential as a series expansion in terms of vertical shape functions of the fluid flow. This Boussinesq approach reduces the three spatial dimensions to two. A linearization process has been carried out, paying carefull attention to the properties of the wave model.

For given bathymetry and ship movements, the wave model calculates the water height. For example, the model is used to compute the wave height on a part of the Dutch river IJssel, resulting in the realistic wave pattern shown in Figure 1.

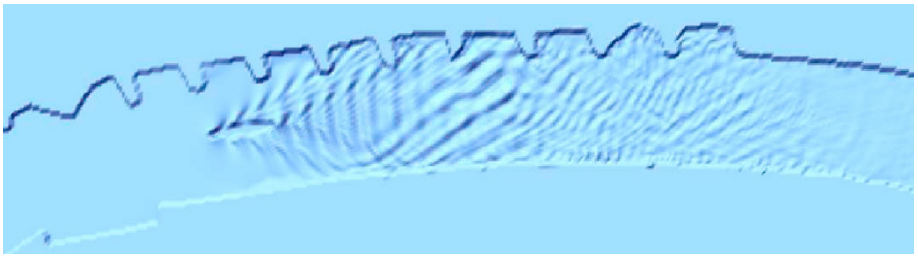


Figure 1: Wave pattern around a ship sailing through the river IJssel.

2.1. Model equations

The Variational Boussinesq model is governed by a set of three coupled equations:

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot \left(\zeta \, \mathbf{U} + h \, \nabla \varphi - h \, \mathcal{D}_0 \, \nabla \psi \right) = 0, \tag{1}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U} \cdot \nabla \varphi + g \zeta = P_s, \tag{2}$$

$$\mathcal{M}_0 \, \psi + \nabla \cdot \left(h \, \mathcal{D}_0 \, \nabla \varphi - \mathcal{N}_0 \, \nabla \psi \right) = 0, \tag{3}$$

for the three basic variables water level ζ, surface velocity potential φ and structured vertical velocity potential ψ, and ∇ denoting the gradient operator $\begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$. The other symbols denote the known parameters gravitation g, water depth h, average current $\mathbf{U}$, imposed pressure P_s, and the positive parameters $\mathcal{D}_0, \mathcal{M}_0, \mathcal{N}_0$ that depend on the water depth. Observe that the first two equations (1) and (2) form a hyperbolic set of equations, similar to the shallow-water equations. The third equation (3) is elliptic and will result in the linear set of equations that has to be solved with a linear solver.

2.2. Discretization methods

The model equations contain both spatial and time derivatives. The Finite Volume method is used for the spatial discretization. To this end, the computational domain is partitioned into rectangular control cells. The normal derivatives on the edges of the cells are calculated using central finite difference approximations.

To avoid computationally intensive solutions of linear systems, use is made of an explicit time integration scheme. The solution of the discretized model equations (1), (2) at a new timestep can therefore be calculated from the previous timestep with a matrix-vector operation. Discretization of model equation (3) yields a linear system

$$A\mathbf{x} = \mathbf{b}, \tag{4}$$

with $A \in \mathbb{R}^{n\times n}$, and $\mathbf{x} \in \mathbb{R}^n$ denoting the variable ψ at the computational nodes, that are ordered lexicographically.

2.3. Matrix properties

Because the properties of the linear system (4) will determine the choice of the solver, we will list some properties of A. The five-point stencil used for the discretization yields a pentadiagonal matrix, i.e., A has nonzero elements on five diagonals only. The linear system is the discretized version of model equation (3), which is an elliptic partial differential equation. The matrix A is therefore strictly diagonally dominant and is symmetric positive definite (SPD).

3. Linear solver

At each time step, the wave model uses the solution of the linear system (4). Since the wave model is for use in a real-time ship simulator, the solution has to be calculated within the timestep of the model. A characteristic value of the timestep is 0.05 s.

The size of A is given by the number of computational nodes. Due to the requirement of real-time calculation and the limitations of current technology, typical values are 200×200 nodes, so a matrix of size $40\,000 \times 40\,000$. Future developments aim to increase these dimensions. Because of the properties of the Poisson-like equation (4), an iterative solver is most suitable.

3.1. Preconditioned Conjugate Gradient method

Since A is SPD, the Conjugate Gradient method is a natural choice as iterative solution method [7, 8]. The convergence of the CG method can be estimated with the well-known upper bound on the residual:

$$\|\mathbf{x}_n - \mathbf{x}\|_A \leq 2\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right)^n \|\mathbf{x}_0 - \mathbf{x}\|_A, \tag{5}$$

with $\lambda_{\min}$ and $\lambda_{\max}$ the minimal and maximal eigenvalue of A respectively, and $\|\cdot\|_A$ denoting the A-norm. This upper bound shows that the convergence of the CG method depends on the spectral condition number $\frac{\lambda_{\max}}{\lambda_{\min}}$.

By applying a preconditioner, the spectrum of the preconditioned matrix can be changed into a more favorable one, in particular one with a smaller spectral condition number. The symmetrically preconditioned system is given by

$$P^{-1}AP^{-T}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{6}$$

with $\tilde{\mathbf{x}} = P^T\mathbf{x}$, $\tilde{\mathbf{b}} = P^{-1}\mathbf{b}$, and $M = PP^T$ denoting the preconditioner. A suitable preconditioner can improve the convergence of the CG method considerably. The preconditioner should be chosen such that the system $P\mathbf{x} = \mathbf{y}$ is relatively easy to solve.

3.2. Incomplete Cholesky preconditioner

Incomplete Cholesky preconditioners are among the best known preconditioners [9]. The symmetric matrix A can be approximated by a decomposition LL^T where L is a lower triangular matrix with a prescribed sparsity pattern. The sparsity pattern is chosen to be the same as of A, so $L + L^T$ can also be represented by a five-point stencil. For this choice the storage requirements of L and A are the same.

After scaling of the matrix, the outer diagonals of L equal the ones of A. Therefore, only the main diagonal of L has to be calculated, according to the incomplete Cholesky decomposition. An important difference between different versions of the incomplete decomposition is the handling of the fill-in elements. A common choice is to discard the fill-in elements, resulting in the Incomplete Cholesky decomposition [10]. The Modified Incomplete Cholesky decomposition on the other hand adds the fill-in elements to the diagonal elements [11]. These two decompositions can be combined by introducing a relaxation parameter ω, then ω times the fill-in element is added to the diagonal. This method is called the Relaxed Incomplete Cholesky decomposition [12]. For $\omega = 0$ one obtains the IC decomposition and for $\omega = 1$ the MIC decomposition.

3.3. Deflation

Preconditioners change the spectrum of the linear system to improve the convergence of the CG method. The RIC preconditioner is SPD by construction and therefore the preconditioned matrix is also SPD. The basic idea of the deflation method is to deflate some predefined vectors into the null-space [4, 13]. This is done by defining a projection matrix Q such that the deflated matrix QA has a nontrivial null-space. The CG method is then applied to the deflated matrix, which is singular and symmetric positive semi-definite. Since the convergence of the CG method does not depend on the zero eigenvalues, it is likely to be improved by applying deflation. The deflation method can be combined with preconditioners.

The vectors defining the projection in the deflation method are called deflation vectors, and have to be predefined. The choice of deflation vectors strongly influences the performance of the method. A natural choice is the set of eigenvectors corresponding to the smallest eigenvalues, thus reducing the effective spectral condition number. However, calculating eigenvectors is computationally expensive. Another choice is subdomain deflation [4], for which the deflation vectors are zero-valued outside a predefined subdomain and valued one inside. The easy structure of these deflation vectors results in a computationally efficient deflation method. For this method, the rectangular grid is partitioned into equally sized rectangular subdomains.

The null-space of the deflated matrix equals the span of the deflation vectors. This deflation subspace is in the case of eigenvector deflation given by the span of the smallest eigenvectors. For subdomain deflation, the null-space is given by vectors that are piecewise-constant on the subdomains. Since both deflation subspaces correspond to slowly varying components of the solution, both methods will result in similar convergence properties. Because of the real-time requirement of the linear solver, the computationally efficient subdomain deflation method is used.

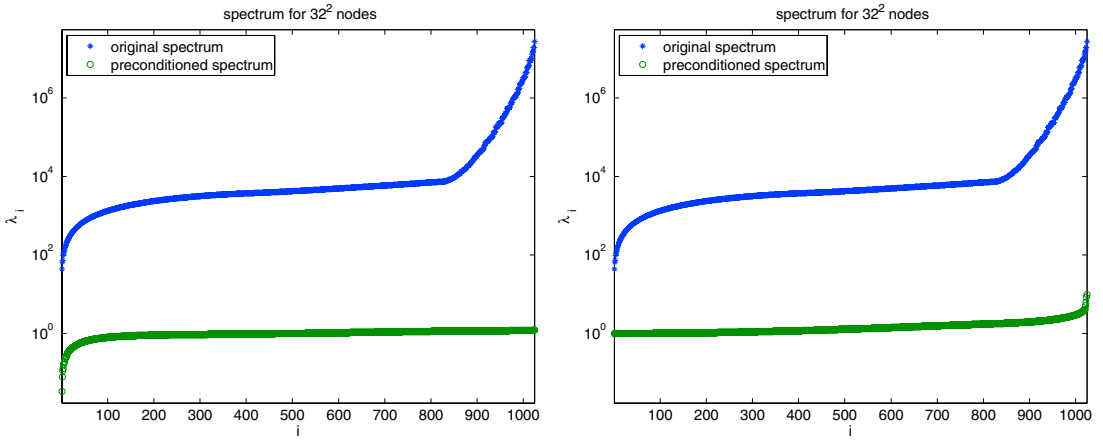
4. Deflated Relaxed Incomplete Cholesky CG method

The literature about the deflation method concentrates on the combination of deflation and the standard IC preconditioner [14, 15, 16]. Here, we will study the deflation method combined with the RIC preconditioner. As will be shown, subdomain deflation complements the IC preconditioner, but not the MIC preconditioner. Because the convergence of the CG method mainly depends on the spectrum of the matrix, the spectrum of different preconditioned and deflated matrices will be presented in this section.

4.1. Spectrum of the RIC preconditioned matrix

From the Finite Volume discretization of Equation (3) and the Gershgorin circle theorem, it follows that the original matrix A has a minimal eigenvalue $\lambda_{\min} = O(h^2)$ and a maximal eigenvalue $\lambda_{\max} = O(1)$, with h denoting the characteristic mesh size. Considering the discrete Poisson equation, which is similar to the discrete model equation (4), the order of the extreme eigenvalues can be calculated for the IC and MIC preconditioned matrices [17]. For the IC preconditioner, we still have $\lambda_{\min} = O(h^2)$ and $\lambda_{\max} = O(1)$. Although the orders of magnitude are the same, in practice the spectral condition number reduces significantly due to the maximal eigenvalue being close to one. The MIC preconditioner results in $\lambda_{\min} = O(1)$ and $\lambda_{\max} = O(h^{-1})$, thus a smaller order of the spectral condition number.

Hence the spectra of IC and MIC preconditioned matrices differ in both the order of the condition number and the structure of the spectrum. Figure 2 shows the spectra of the preconditioned matrices for a typical test problem [1]. To calculate the spectrum explicitly, the size of this test problem is taken smaller than for usual applications.



(a) IC preconditioner (b) MIC preconditioner

Figure 2: The spectrum of the original matrix and the preconditioned matrices.

4.2. *Spectrum of the deflated matrix*

For the construction of the subdomain deflation vectors, the computational domain is partitioned into m subdomains. Because these deflation vectors are projected into the null-space, the spectrum of the deflated matrix has m zero eigenvalues. Although the spectrum of the subdomain deflated matrix is not exactly known in advance, one can expect (see Section 3.3) that the main differences between the nonzero parts of the spectra are in the smallest eigenvalues. The smallest nonzero eigenvalue will be larger when deflation is applied, and the effective spectral condition number will thus be smaller. Figure 3 shows the spectrum of a deflated matrix for the same test problem as in Figure 2.

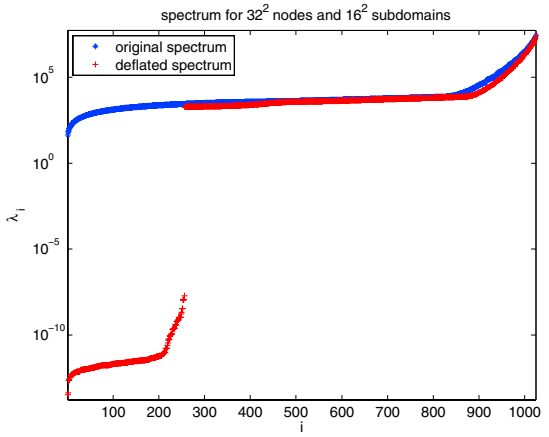
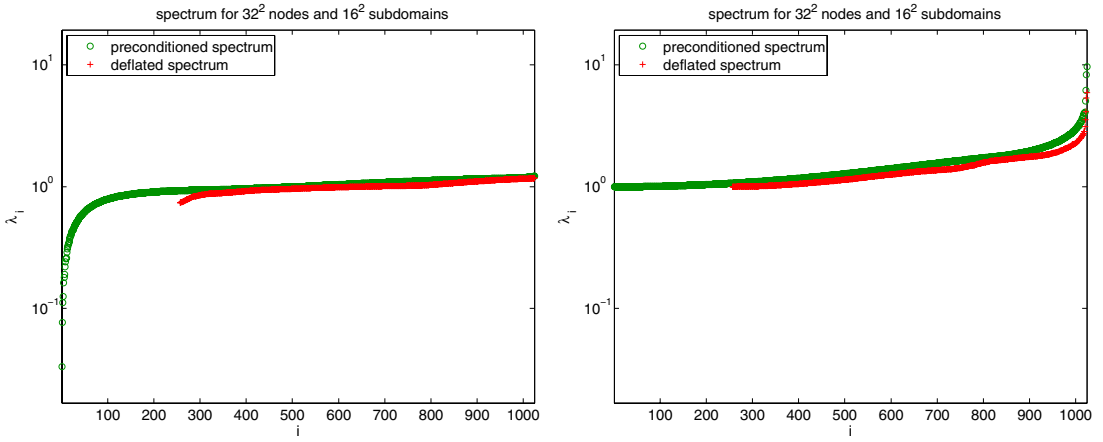


Figure 3: The spectrum of the original matrix and the subdomain deflated matrix.

4.3. Spectrum of the deflated RIC preconditioned matrix

Subdomain deflation is used in combination with the RIC preconditioner. The extreme choices of the relaxation parameter of the RIC preconditioner result in the IC and MIC preconditioner. We will consider these two preconditioners first.

The IC preconditioner maps the largest eigenvalues approximately to one while the smallest remain $O(h^2)$. Subdomain deflation maps the smallest eigenvalues to zero. Therefore, these two methods are complementary. This can be seen in Figure 4(a), presenting the nonzero part of the spectra, for the same test problem as before.



(a) IC preconditioner

(b) MIC preconditioner

Figure 4: The nonzero part of the spectrum of the preconditioned matrices and the deflated preconditioned matrices.

The MIC preconditioner results in a spectrum with the smallest eigenvalues approximately one. These eigenvalues will be mapped to zero by subdomain deflation. Because the number of deflation vectors is relatively small, the smallest nonzero deflated eigenvalues remain close to one. The MIC preconditioner and subdomain deflation thus do not complement each other. The spectra presented in Figure 4(b) show this observation, because the nonzero eigenvalues of the preconditioned and deflated matrices are approximately equal-valued.

These observations suggest the use of the DICCG method. However, in the undeflated case, the MIC preconditioner gives a spectral condition number that is significantly smaller than the IC preconditioner: $O(h^{-1})$ compared to $O(h^{-2})$. To use the advantage of the MIC preconditioner in the undeflated method and the advantage of the combination of subdomain deflation and the IC preconditioner, the deflation method is combined with the RIC preconditioner. A good choice of the relaxation parameter may lead to a deflated spectrum that is favorable than for both the IC and MIC preconditioner. The computational results in next section show that this is actually the case for some test problems.

5. Computational results

In previous sections, some properties of the Deflated Relaxed Incomplete Cholesky CG method have been given. In the undeflated version, the MIC preconditioner has a lower order of the condition number than the standard IC preconditioner. However, subdomain deflation complements the IC preconditioner, whereas it does not complement the MIC preconditioner.

In this section we will look at some computational results. Several experiments have been done with varying values of the relaxation parameter ω and the number of subdomains, which is the same as the number of deflation vectors. Characteristic values are taken for the parameters of the model equations [1]. Since we aim at real-time calculations on the present hardware configuration, the test problem uses a grid of 400×400 nodes, resulting in a set of 160 000 equations.

The algorithm has been implemented in the programming language `C++` and compiled with the `GNU C++ 4.2.4` compiler. The experiments have been performed on a computer with an `Intel Core 2 Duo E6850` processor that has a clock rate of 3 GHz, and 4 GB internal memory.

5.1. The deflated RICCG method

Remember we have a spectral condition number of $O(h^{-1})$ and $O(h^{-2})$ for the MIC and IC preconditioner, respectively. This suggests a better convergence of the MICCG method, so a smaller number of CG iterations. The results given in the top row of Table 1 show this behavior.

The results in Table 1 also show that the number of deflated CG iterations reduces when using more subdomains. For the deflated MICCG method it is only a small reduction while for the deflated ICCG method the number of CG iterations reduces sharply. This corresponds to the theory as given in Section 4.

# subdomains	$\omega = 0$ (IC)	$\omega = 0.5$	$\omega = 1$ (MIC)
no deflation	33	28	17
10×10	31	27	17
40×40	18	16	17
160×160	9	9	14

Table 1: Number of iterations of the deflated RICCG method.

The computational results of the test problem show clearly the difference between the IC and MIC preconditioner. Taking the relaxation parameter in between 0 and 1, one may expect that the number of CG iterations will also be in between the extreme choices. However, using intermediate values of ω combine the different advantages of the extreme methods and may therefore have a better convergence then both. Table 1 shows that for the case of 40×40 subdomains, so 1600 deflation vectors, the DRICCG method with $\omega = 0.5$ uses less CG iterations than both the DICCG and DMICCG method.

The number of CG iterations is a good measure for the performance of the method. However, the effort for one CG iteration depends on the preconditioners used. Therefore, the performance of the DRICCG method is also assessed on basis of the CPU time used. The CPU time listed in Table 2 is an average value of several time-steps of the wave model, for the same test problem as in Table 1.

# subdomains	$\omega = 0$ (IC)	$\omega = 0.5$	$\omega = 1$ (MIC)
no deflation	0.98	0.85	0.53
10×10	1.24	1.05	0.70
40×40	0.74	0.66	0.70
160×160	1.00	0.97	1.50

Table 2: The CPU time in seconds used by the deflated RICCG method.

The deflation method defines a relatively small system which has to be solved every CG iteration. It has the size of the number of deflation vectors. Using more deflation vectors will thus result in a more expensive deflation method. This can be seen in the results of the MIC preconditioner for which the number of CG iterations is almost constant, but with an increase in CPU time. In the case of the IC preconditioner, the number of CG iterations decreases for larger numbers of deflation vectors. The total CPU time for the smaller number of more expensive CG iterations can be seen in Table 2. In the case of 40×40 subdomains the CPU time drops below the undeflated method.

The optimal choice of the relaxation paramater is most of the time an intermediate one. Predicting the optimal combination of relaxation parameter and number of deflation vectors is an open question.

5.2. The block-RICCG method

For future development we also compose a block version of the RIC preconditioner. The block-RIC preconditioner is constructed by applying the RIC preconditioner per subdomain [12]. To this end, the computational domain is partitioned into rectangular subdomains.

Since the block-RICCG method discards coupling between subdomains, it is less effective than the RICCG method, but is inherently parallel. Therefore, we expect an increasing number of CG iterations for increasing numbers of blocks. In Tables 3 and 4 computational results are presented for the block-RICCG method on the same test problem as before. The actual parallelization is not done, i.e., the method is performed on one processor.

# blocks	$\omega = 0$	$\omega = 0.5$	$\omega = 1$
RIC	33	28	17
10×10	42	37	46
40×40	47	44	47
160×160	69	69	69

Table 3: Number of iterations of the block-RICCG method.

# blocks	$\omega = 0$	$\omega = 0.5$	$\omega = 1$
RIC	0.98	0.85	0.53
10×10	1.09	0.97	1.20
40×40	1.24	1.16	1.23
160×160	1.91	1.94	1.93

Table 4: The CPU time in seconds used by the block-RICCG method.

As expected, when increasing the number of blocks, the block-RICCG requires more iterations to converge. This results in a larger computation time.

5.3. The deflated block-RICCG method

As is done with the RIC preconditioner, the block-RIC preconditioner can be combined with the deflation method. The subdomain deflation method can be parallelized efficiently [4, 18]. The same subdomains are used for the block-RIC preconditioner and the deflation method, this is not a necessary requirement, however. Computational results of the deflated block-RICCG method are presented in Tables 5 and 6.

# subdomains	$\omega = 0$	$\omega = 0.5$	$\omega = 1$
RIC	33	28	17
10×10	40	35	46
40×40	27	25	29
160×160	14	14	14

Table 5: Number of iterations of the deflated block-RICCG method.

Comparing the results with the undeflated method of Tables 3 and 4 shows that for large numbers of blocks, the deflation method reduces the number of CG iterations considerably, resulting in smaller computation times.

# subdomains	$\omega = 0$	$\omega = 0.5$	$\omega = 1$
RIC	0.98	0.85	0.53
10×10	1.57	1.22	1.83
40×40	1.07	0.91	1.15
160×160	1.50	1.43	1.52

Table 6: The CPU time in seconds used by the deflated block-RICCG method.

In Tables 1 and 2 computational results of the deflated CG method with a full RIC preconditioner are shown. The computation times of the deflated block-RIC preconditioner are slightly larger, as can be seen in Table 6. The inherent parallelism of the block-RIC preconditioner makes these results quite promising for actual parallel computations.

6. Conclusions

For use in a ship simulator, a wave model has been developed. At every time step a linear system has to be solved in real time. Because of the properties of the linear system, the CG method with the Relaxed Incomplete Cholesky preconditioner is used.

To improve the performance of the RICCG method, we have combined it with the deflation method. The computational efficient choice of subdomain deflation is used. The spectrum of the preconditioned matrices has a different structure for different values of the relaxation parameter. This difference is of great importance for the convergence of the RICCG method. It also explains the influence of the deflation method on the RICCG method, i.e., subdomain deflation complements the IC preconditioner, but not the MIC preconditioner.

Computational results of the Deflated RICCG method show the expected convergence behavior. That is, for small numbers of deflation vectors the number of CG iterations is minimal for large relaxation parameters while for large numbers of deflation vectors it is minimal for small relaxation parameters. In general, it is an open question how to make an optimal a priori choice for the relaxation parameter and number of deflation vectors.

Using a block version of the RIC preconditioner makes the linear solver more suitable for parallelization. Subdomain deflation can naturally be combined with the block-RIC preconditioner. We have shown that deflation improves the convergence of the block-RICCG method considerably, and reduces the required computation time. The deflated block-RICCG method is thus a promising method for use in the wave model, especially on a parallel environment.

Acknowledgement

The authors gratefully acknowledge the contributions of Gert Klopman and Anneke Sicherer-Roetman to the research project.

References

[1] E. van 't Wout, Improving the linear solver used in the interactive wave model of a real-time ship simulator, Master's thesis, TU Delft, http://ta.twi.tudelft.nl/nw/users/vuik/numanal/wout_eng.html (2009).
[2] A. van der Ploeg, Preconditioning for sparse matrices with applications, Ph.D. thesis, Rijksuniversiteit Groningen (1994).
[3] C. W. Brand, An incomplete-factorization preconditioning using repeated red-black ordering, Numerische Mathematik 61 (1992) 433–454.
[4] J. M. Tang, Two-level preconditioned Conjugate Gradient methods with applications to bubbly flow problems, Ph.D. thesis, TU Delft (2008).
[5] G. Klopman, M. W. Dingemans, B. van Groesen, A variational model for fully non-linear water waves of Boussinesq type, in: Proceedings IWWWFB, 2005.
[6] G. Klopman, B. van Groesen, M. W. Dingemans, A variational approach to Boussinesq modelling of fully non-linear water waves, accepted for publication in Journal of Fluid Mechanics.
[7] H. A. van der Vorst, Iterative Krylov methods for large linear systems, Cambridge University Press, Cambridge, 2003.
[8] M. R. Hestenes, E. Stiefel, Methods of Conjugate Gradients for solving linear systems, Journal of Research of the National Bureau of Standards 49 (6) (1952) 409–436.
[9] Y. Saad, Iterative methods for sparse linear systems, 2nd Edition, SIAM, Philadelphia, 2000.
[10] J. A. Meijerink, H. A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Mathematics of Computation 31 (137) (1977) 148–162.
[11] I. Gustafsson, A class of first order factorization methods, BIT Numerical Mathematics 18 (2) (1978) 142–156.
[12] O. Axelsson, G. Lindskog, On the eigenvalue distribution of a class of preconditioned methods, Numerische Mathematik 48 (1986) 479–498.
[13] R. A. Nicolaides, Deflation of Conjugate Gradients with applications to boundary value problems, SIAM Journal on Numerical Analysis 24 (2) (1987) 355–365.
[14] R. Nabben, C. Vuik, A comparison of deflation and coarse grid correction applied to porous media flow, SIAM Journal on Numerical Analysis 42 (4) (2004) 1631–1647.
[15] J. M. Tang, C. Vuik, Efficient deflation methods applied to 3-D bubbly flow problems, Electronic Transactions on Numerical Analysis 26 (2007) 330–349.
[16] J. M. Tang, C. Vuik, New variants of deflation techniques for pressure correction in bubbly flow problems, Journal of Numerical Analysis, Industrial and Applied Mathematics 2 (3-4) (2007) 227–249.
[17] H. A. van der Vorst, The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors, Lecture Notes in Mathematics 1457 (1990) 127–136.
[18] J. Frank, C. Vuik, On the construction of deflation-based preconditioners, SIAM Journal on Scientific Computing 23 (2) (2001) 442–462.