INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2013; **17**:830–849
Published online 4 May 2012 in Wiley Online Library (wileyonlinelibrary.com/journal/nmf). DOI: 10.1002/fld.3686

SIMPLE-type preconditioners for cell-centered, colocated finite volume discretization of incompressible Reynolds-averaged Navier–Stokes equations

C. M. Klaij[1,*,†] and C. Vuik[2]

[1]*Maritime Research Institute Netherlands, P.O.Box 28, 6700AA Wageningen, The Netherlands*
[2]*Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O.Box 5058, 2600GB Delft, The Netherlands*

SUMMARY

This paper contains a comparison of four SIMPLE-type methods used as solver and as preconditioner for the iterative solution of the (Reynolds-averaged) Navier–Stokes equations, discretized with a finite volume method for cell-centered, colocated variables on unstructured grids. A matrix-free implementation is presented, and special attention is given to the treatment of the stabilization matrix to maintain a compact stencil suitable for unstructured grids. We find SIMPLER preconditioning to be robust and efficient for academic test cases and industrial test cases. Compared with the classical SIMPLE solver, SIMPLER preconditioning reduces the number of nonlinear iterations by a factor 5–20 and the CPU time by a factor 2–5 depending on the case. The flow around a ship hull at Reynolds number 2E9, for example, on a grid with cell aspect ratio up to 1:1E6, can be computed in 3 instead of 15 h. Copyright © 2012 John Wiley & Sons, Ltd.

Received 22 July 2011; Revised 2 December 2011; Accepted 9 April 2012

KEY WORDS: Navier–Stokes equations; Reynolds-averaged Navier–Stokes equations; incompressible flow; finite volume methods; iterative methods; SIMPLE-type preconditioning

1. INTRODUCTION

Starting point of this work is an existing finite volume RaNS code with SIMPLE-type solver and the desire to improve its iterative convergence.

The code, a MARIN in-house code named REFRESCO [1], solves flow problems encountered in the maritime industry, among others the flow around ship hulls. These RaNS simulations complement model testing by providing flow field details for diagnosing problems and improving designs. Besides, RaNS simulations at full scale give insight into scale effects and help to translate experimental results from model scale to full scale.

To be industrially applicable, the code must be sufficiently accurate, efficient, and robust. Accuracy is achieved by grid refinement and grid contraction to capture the thin boundary layers. This quickly leads to millions of cells and very high cell aspect ratio near the hull. In such cases, the efficiency and robustness of the iterative method are insufficient: careful tuning of the relaxation parameters is needed to obtain convergence, and the convergence rate is often disappointing. Our aim, therefore, is to improve the convergence rate of the iterative method and to reduce its sensitivity to relaxation parameters. This would not only reduce the computational time but also the amount of user input and thereby facilitate routine application of the code in an industrial design process.

*Correspondence to: C. M. Klaij, Maritime Research Institute Netherlands, P.O.Box 28, 6700AA Wageningen, The Netherlands.
†E-mail: c.klaij@marin.nl

The flow problems are characterized by high Reynolds numbers and complex geometries that challenge both discretization and iterative methods. The flow physics are modeled by the steady Reynolds-averaged Navier–Stokes equations combined with one-equation or two-equation turbulence models. The discretization method for these equations is based on a cell-centered, colocated finite volume method for unstructured grids of cells with arbitrary number of faces. The equations are linearized with Picard's method and solved in a segregated manner by using the classical SIMPLE method.

We aim at improving the iterative convergence by coupling the mass and momentum equations while keeping the turbulence model equations segregated. The coupled Navier–Stokes system is solved with a Krylov subspace method that requires a good preconditioner. Recent developments in the context of finite element methods have yielded a number of suitable preconditioners, see the overview in [2,3]. Among the possible choices are block preconditioners based on LDU decomposition [4–7], block preconditioners based on SIMPLE-type methods [2, 3, 8, 9], and saddle point ILU preconditioners [3, 8, 10]. Another possibility would be nonlinear multigrid with coupled Gauss–Seidel smoothing [11,12]. Our interest goes to SIMPLE-type preconditioners because these methods allow a matrix-free implementation and because large parts of the existing code can be reused.

SIMPLE(R) was first applied as preconditioner in the context of finite volume methods for structured grids with staggered variables in [9]; here, we present the extension to cell-centered, unstructured grids with colocated variables. This extension is nontrivial because of the additional stabilization term, needed to prevent spurious pressure oscillations. The stabilization term leads to a matrix with a wide stencil involving not only a cell's neighbors but also the neighbors of the neighbors. We do not wish to form this matrix as a compact stencil is crucial for simplicity and efficiency on unstructured grids. This constraint prevents a straightforward application of SIMPLE-type preconditioning. Therefore, we propose two modifications of SIMPLE(R) that avoid the construction of the stabilization matrix. The classical usage of SIMPLE-type methods as solver is then compared with the modern usage as preconditioner for a number of academic and industrial test cases. Four variants are considered: SIMPLE and SIMPLER [13, 14] and the recently proposed MSIMPLE and MSIMPLER [3, 8, 15]. The comparison focuses on how the solver choice affects the nonlinear convergence and the computional effort for a series of increasingly challenging test cases.

The layout of this paper is as follows. The discretization of the Navier–Stokes equations, including linearization and stabilization is treated in Section 2. We then discuss the usage of SIMPLE-type methods as solver in the first part of Section 3 and their usage as preconditioner in the second part, including the necessary modifications related to the stabilization term. We compare the performance of the different methods for academic test cases in Section 4 and for maritime test cases in Section 5 before drawing conclusions in Section 6.

2. DISCRETIZATION

Picard linearization of the steady, incompressible Navier–Stokes equations in two-dimensions and finite volume discretization with colocated variables leads to a linear system of the form:

$$\begin{bmatrix} Q_1 & 0 & G_1 \\ 0 & Q_2 & G_2 \\ D_1 & D_2 & C \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ g \end{bmatrix} \tag{1}$$

with $u = (u_1, u_2)$ the velocity and p the pressure. In this section, we specify the discretization choices that define the matrices Q, D, G, and C. The tensor notation of Wesseling [16] will be used whenever possible. The methods presented in this paper are designed for—and applied to—complex three-dimensional cases, but rectangular cells are assumed here to illustrate the main points of the discretization method.

2.1. Pressure-weighted interpolation method

The pressure-weighted interpolation method (PWIM) is used to avoid checkerboard oscillations on grids with colocated variables [16, 17]. The pressure-weighting term consists of a second-order

DOI: 10.1002/fld

derivative of the pressure gradient, scaled with a grid-dependent factor ϵh^2 with h as the cell size. This term is added to the velocity

$$\tilde{u}^\alpha \equiv u^\alpha + \epsilon h^2 (p_{,\alpha})_{,\beta\beta}, \tag{2}$$

and it is this pressure-weighted velocity that is used in Reynolds' transport theorem [18] to derive the conservation laws for momentum and mass

$$\int_\Omega \left(\rho u^\alpha \tilde{u}^\beta\right)_{,\beta} = \int_\Omega \left(\mu\left(u^\alpha_{,\beta} + u^\beta_{,\alpha}\right) - p\delta^{\alpha\beta}\right)_{,\beta}, \tag{3a}$$

$$\int_\Omega \tilde{u}^\alpha_{,\alpha} = 0 \tag{3b}$$

with ρ as the density, μ the dynamic viscosity coefficient, and δ the Kronecker delta. The comma denotes differentiation, and the summation convention applies to repeated indices. The PWIM thus adds a fourth-order pressure derivative to the left-hand side of the momentum equations and of the mass equation. Because $h \to 0$ upon grid refinement, the original equations are recovered. The factor ϵ is chosen such that Equation (2) is dimensionally correct and such that the pressure weighting does not interfere with the second-order accuracy of the finite volume discretization. The expression for ϵ will be given later on. We do not consider the time derivative as the equations will be solved directly in their stationary form without (pseudo) time-stepping techniques.

2.2. *Discretization of the momentum equations*

Partial integration of steady momentum equation (3a) over a cell Ω_i gives

$$\int_{\partial\Omega_i} \left(\rho u^\alpha \tilde{u}^\beta\right) n^\beta = \int_{\partial\Omega_i} \left(\mu\left(u^\alpha_{,\beta} + u^\beta_{,\alpha}\right) - p\delta^{\alpha\beta}\right) n^\beta \tag{4}$$

with n as the unit normal vector pointing outwards with respect to Ω_i. The cell boundary $\partial\Omega_i$ is divided into faces such that $\partial\Omega_i = \cup_j S_{ij}$ with $S_{ij} = \Omega_i \cap \Omega_j$. With the unknowns $(\cdot)_i$ and $(\cdot)_j$ colocated in the cell centers of Ω_i and Ω_j, the values $(\cdot)_{ij}$ at the faces S_{ij} must be obtained through interpolation. The discretization is uniquely defined once all the interpolation choices are stated.

With Picard linearization, the pressure-weighted mass flux at the faces

$$m_{ij} \equiv \left(\rho \tilde{u}^\beta n^\beta\right)_{ij}$$

is assumed to be known from the previous iteration. Many choices are possible for the advection term $m\,u^\alpha$, but only the first-order upwind difference scheme is relevant to the construction of Q_α because higher-order advection schemes are implemented in defect correction form [16]. Thus,

$$m_{ij} u^\alpha_{ij} = \frac{1}{2}(m_{ij} + |m_{ij}|)u^\alpha_i + \frac{1}{2}(m_{ij} - |m_{ij}|)u^\alpha_j.$$

For the diffusion term $u^\alpha_{,\beta}$, the first-order central difference is taken,

$$\left(u^\alpha_{,\beta}\right)_{ij} = \frac{u^\alpha_j - u^\alpha_i}{x^\beta_j - x^\beta_i}$$

with x^β_i the β component of the coordinates of the center of cell Ω_i. The term $u^\beta_{,\alpha}$ is assumed to be known from the previous iteration. Without this assumption or with Newton linearization instead of Picard linearization, system (1) would not have a block diagonal part. The construction of the matrices Q_α in system (1) follows by substitution of these interpolation choices in discrete momentum equation (4). Note that $Q_1 = Q_2$ because of Picard linearization and because of the explicit treatment of $u^\beta_{,\alpha}$.

The pressure at face S_{ij} is interpolated as

$$p_{ij} = \gamma p_i + (1-\gamma) p_j \tag{5}$$

with γ a grid-dependent interpolation factor ($\gamma = 1/2$ on uniform grids); the construction of the matrices G_α in system (1) follows.

DOI: 10.1002/fld

2.3. Discretization of the mass equation

Partial integration of mass equation (3b) over a cell Ω_i gives

$$\int_{\partial\Omega_i} \tilde{u}^\alpha n^\alpha = 0. \tag{6}$$

The interpolation of the velocity at face S_{ij} is the same as the interpolation of the pressure: $u^\alpha_{ij} = \gamma u^\alpha_i + (1-\gamma)u^\alpha_j$. The construction of the matrices D_α in system (1) follows from this choice.

The interpolation choice for the pressure-weighting term in Equation (2) is more involved. The second-order derivative of the pressure gradient is approximated with the second-order finite difference:

$$\epsilon h^2 (p_{ij,\alpha})_{,\beta\beta} = \epsilon_i p_{i,\alpha} - (\epsilon_i + \epsilon_j) p_{ij,\alpha} + \epsilon_j p_{j,\alpha}. \tag{7}$$

The term $p_{i,\alpha}$ can be obtained with Gauss' theorem

$$p_{i,\alpha} \equiv \frac{1}{|\Omega_i|}\int_{\Omega_i} p_{,\alpha} = \frac{1}{|\Omega_i|}\int_{\partial\Omega_i} p_{ij} n^\alpha \tag{8}$$

using the same interpolation (5) for p_{ij} as in the momentum equation. The central difference is used for the interpolation of the pressure gradient at the face S_{ij},

$$p_{ij,\alpha} = \frac{p_j - p_i}{x^\alpha_j - x^\alpha_i}.$$

The coefficients ϵ are chosen as

$$\epsilon_i = \gamma \frac{|\Omega_i|}{\mathrm{diag}(Q_\alpha)_i}, \qquad \epsilon_j = (1-\gamma)\frac{|\Omega_j|}{\mathrm{diag}(Q_\alpha)_j}. \tag{9}$$

Together, these choices lead to the matrix C in system (1). Note that the stencil of the matrix C is wide: it involves not only the neighbors of the cell but also the neighbors of the neighbors.

With these choices, we can now compute the pressure-weighted mass flux that was assumed to be known in the discrete momentum equations, thereby closing the discretization of the Navier–Stokes equations.

2.4. Discrete Laplacian

As we will see in the next section, SIMPLE-type preconditioners require the matrix R defined as

$$R \equiv -D\ \mathrm{diag}(Q)^{-1} G \qquad (= -\Sigma_i D_i\ \mathrm{diag}(Q_i)^{-1} G_i). \tag{10}$$

It is a Laplacian matrix obtained by applying the divergence operator to the gradient of p, scaled with the diagonal of Q. In practice, constructing R does not require matrix–matrix multiplications as Equation (10) suggests. An approximation that can be directly constructed on unstructured grids is given in [17]; this matrix arises from the integral

$$-\int_{\Omega_i} \epsilon p_{,\alpha\alpha}.$$

Partial integration of this term gives

$$-\int_{\partial\Omega_i} \epsilon p_{,\alpha} n^\alpha.$$

For the interpolation of $p_{,\alpha}$ at a face S_{ij}, we choose

$$\epsilon p_{ij,\alpha} = (\epsilon_i + \epsilon_j)\frac{p_j - p_i}{x^\alpha_j - x^\alpha_i}. \tag{11}$$

Note that this interpolation choice is the same as the one used for the middle term in the second-order difference on the right-hand side of Equation (7).

DOI: 10.1002/fld

2.5. Remarks on the pressure-weighted interpolation method

- The choice of the coefficient ϵ is directly related to the SIMPLE approximation $Q^{-1} \approx \text{diag}(Q)^{-1}$ presented in the next section. When this approximation is used in the discrete momentum equation $Q_\alpha u^\alpha = f - G_\alpha p$ and when f is disregarded, we obtain

$$u_i^\alpha = -\frac{|\Omega_i|}{\text{diag}(Q_\alpha)_i} p_{i,\alpha},$$

  which allows the velocity in a cell Ω_i to be 'interchanged' with the pressure gradient. This expression is only used here to illustrate the choice of ϵ, and it is obviously not used when solving the momentum equations. The interpolation of u at face S_{ij} is now (trivially) written as

$$\begin{aligned} u_{ij}^\alpha =& \gamma u_i^\alpha + (1-\gamma) u_j^\alpha \\ & - \gamma u_i^\alpha - (1-\gamma) u_j^\alpha + u_{ij}^\alpha. \end{aligned}$$

  The first part on the right-hand side is maintained, but the velocity u^α in the second part is interchanged for the pressure gradient

$$\begin{aligned} u_{ij}^\alpha =& \gamma u_i^\alpha + (1-\gamma) u_j^\alpha \\ & + \gamma \tfrac{|\Omega_i|}{\text{diag}(Q_\alpha)_i} p_{i,\alpha} + (1-\gamma) \tfrac{|\Omega_j|}{\text{diag}(Q_\alpha)_j} p_{j,\alpha} - \left(\gamma \tfrac{|\Omega_i|}{\text{diag}(Q_\alpha)_i} + (1-\gamma) \tfrac{|\Omega_j|}{\text{diag}(Q_\alpha)_j} \right) p_{ij,\alpha}. \end{aligned}$$

  We now recognize the second-order derivative of the pressure gradient from Equation (7) with the coefficients ϵ from Equation (9). This observation is inspired by [19]; it clearly shows the origin of the interpolation choices in the PWIM. It also shows that this choice of ϵ makes the PWIM dimensionally correct.
- The implementation of the advection scheme in defect correction form is essential to have the same $\text{diag}(Q)$ in the PWIM and hence the same matrix C, regardless of the user's choice for the advection scheme.
- Implicit time discretization and/or implicit relaxation (see Section 3.4) adds a diagonal term to Q that depends on the time-step and/or implicit relaxation factor. Because $\text{diag}(Q)$ affects the discretization of the mass and momentum equation through the PWIM, this leads to the unwanted situation that the converged steady-state solution depends on the user's choice of time-step and/or implicit relaxation factor (although the effect vanishes with grid refinement). Therefore, in the pressure-weighted interpolation only, we omit these contributions from $\text{diag}(Q)$.

3. ITERATIVE METHOD

In the first part of this section, the derivation of SIMPLE-type methods is discussed to prepare for their usage as preconditioner in the second part. Two modifications are proposed to avoid the construction of the stabilization matrix C. The velocity components in system (1) are grouped, so we can continue with the notation

$$\begin{bmatrix} Q & G \\ D & C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{12}$$

for sake of clarity and brevity.

3.1. SIMPLE-type methods

In this section, we derive the SIMPLE method, discuss the difference between prediction and correction form, and derive the additional pressure equation needed for SIMPLER.

DOI: 10.1002/fld

3.1.1. Derivation of SIMPLE. Suppose that u and p are known at iteration k and define a prediction u^* and corrections u' and p' such that

$$u^{k+1} = u^* + u' \qquad p^{k+1} = p^k + p'. \tag{13}$$

The pressure at iteration k is used to obtain the velocity prediction u^* (which does not satisfy the mass equation) by solving

$$Qu^* = f - Gp^k. \tag{14}$$

At iteration $k+1$, we must have

$$Qu^{k+1} = f - Gp^{k+1}, \tag{15}$$

and we must satisfy the mass equation

$$Du^{k+1} + Cp^{k+1} = g. \tag{16}$$

Subtracting Equation (14) from Equation (15) and multiplying by Q^{-1} give the velocity correction equation

$$u' = -Q^{-1}Gp'. \tag{17}$$

The pressure correction equation follows from Equation (16),

$$\begin{aligned} Du^{k+1} + Cp^{k+1} = g &\Leftrightarrow D(u^* + u') + C(p^k + p') = g \\ &\Leftrightarrow (C - DQ^{-1}G)p' = g - Du^* - Cp^k. \end{aligned} \tag{18}$$

The term $C - DQ^{-1}G$ is the Schur complement. Because Q^{-1} is not available, the approximation used in the SIMPLE method is

$$Q^{-1} \approx \mathrm{diag}(Q)^{-1}, \tag{19}$$

where $\mathrm{diag}(Q)$ denotes the diagonal matrix where the diagonal entries are those of Q . The method thus becomes the following:

1. Solve Equation (14) for the velocity prediction u^*.
2. Solve Equation (18) for the pressure correction p'.
3. Compute the velocity correction u' by using Equation (17)
4. Update the pressure and velocity by using Equation (13).

Note that p' is zero, where p has a Dirichlet boundary condition, and that $\partial p'/\partial n = 0$, where p has a Neumann boundary condition.

Another derivation of SIMPLE is given in [2]: the matrix of Equation (12) is first factorized in block *LDU* form; the Schur complement then appears in the diagonal block of the factorization. A wide range of iterative methods can be derived by making different approximation choices for the Schur complement, from physics-based methods such as SIMPLE to purely algebraic methods.

3.1.2. Prediction versus correction. The SIMPLE method is traditionally presented with a velocity prediction and relaxation with parameter ω_u:

$$\begin{gathered} \text{Solve } Qu^* = f - Gp^k \text{ with initial condition } u^k, \\ \text{Relax } u^{k+1} = (1-\omega_u)u^k + \omega_u u^*. \end{gathered}$$

However, to keep in line with the stationary iteration discussed later on, we always use the (equivalent) velocity correction form

$$\begin{gathered} \text{Solve } Qu' = r_u^k \text{ with initial condition } 0, \\ \text{Relax } u^{k+1} = u^k + \omega_u u', \end{gathered}$$

where $r_u^k \equiv f - Qu^k - Gp^k$ is the residual of the momentum equation. The relation here is $u^* \equiv u^k + u'$, not to be confused with the velocity correction from Equation (17). The pressure equation is already in correction form and is usually relaxed with parameter ω_p.

DOI: 10.1002/fld

3.1.3. Derivation of pressure prediction in SIMPLER. To derive the equation for the pressure prediction p^* in SIMPLER, one assumes that u^k is known so that the momentum equation becomes

$$Gp^* = f - Qu^k \tag{20}$$

and the mass equation becomes

$$Cp^* = g - Du^k. \tag{21}$$

Multiplying Equation (20) by $-D\ \mathrm{diag}(Q)^{-1}$ and subtracting from (21) give

$$(C - D\ \mathrm{diag}(Q)^{-1}G)p^* = g - Du^k - D\ \mathrm{diag}(Q)^{-1}(f - Qu^k) \tag{22}$$

This is the prediction form used in [9]. The equivalent correction form reads

$$(C - D\ \mathrm{diag}(Q)^{-1}G)p' = r_p^k - D\ \mathrm{diag}(Q)^{-1}r_u^k \tag{23}$$

with residuals $r_u^k \equiv f - Qu^k - Gp^k$ and $r_p^k \equiv g - Du^k - Cp^k$. This is the form used in [8, 15].

Now that the derivation of SIMPLE-type methods is completed, we move on to the second part where these methods are used as preconditioners in a stationary iteration.

3.2. Nonlinear stationary iteration

The iterative method presented in this section solves the nonlinear system of discrete equations by a stationary iteration

$$x^{k+1} = x^k + \omega\tilde{A}_k^{-1}(b - A_k x^k), \tag{24}$$

where

$$A = \begin{bmatrix} Q & G \\ D & C \end{bmatrix}, \qquad x = \begin{bmatrix} u \\ p \end{bmatrix}, \qquad b = \begin{bmatrix} f \\ g \end{bmatrix}, \qquad \omega = \begin{bmatrix} \omega_u I & 0 \\ 0 & \omega_p I \end{bmatrix},$$

and $\tilde{A}^{-1}$ denotes an approximation of A^{-1}. The relaxation parameters ω_u and ω_p are applied to the corrections for u and p, respectively. Thus, at every nonlinear iteration k, a linear system of the form

$$Ax' = r$$

with residual $r \equiv b - Ax$ must be solved (approximately) for correction x'. We do so by solving the right preconditioned system

$$AP^{-1}y = r, \qquad x' = P^{-1}y$$

with a Krylov subspace method. We use the Flexible GMRES method [20] as implemented in PETSc [21] because it allows a variable preconditioner. This property is essential [9] as the inverse of P is computed approximately, making P^{-1} a different operator in every linear iteration.

Krylov subspace methods only require the action of A, not the matrix itself. Neither is the matrix required to construct the preconditioner because the preconditioner is already known. Therefore, we can follow a matrix-free approach: with the 'shell' matrix type in PETSc, it is sufficient to provide, for a given input vector y, a matrix multiplication routine with output vector Ay and a preconditioning routine with output vector $P^{-1}y$. The algorithms corresponding to SIMPLE-type preconditioning routines are given next, including the two modifications proposed to avoid the construction of the stabilization matrix C.

The matrix-free approach only requires the storage of Q and R. Because, for our discretization, Q is a block diagonal matrix with three equal blocks Q_i, we only have to store two n-by-n sparse matrices with n as the number of cells, instead of storing the nine matrices Q_i, D_i, and G_i and the matrix C. Not storing C is particularly advantageous because it is much less sparse than the other matrices because of its wide stencil.

DOI: 10.1002/fld

Algorithm 1 Action $x = P^{-1}y$ of SIMPLE preconditioner

Given y_u and y_p

1. Solve $Qx'_u = y_u$
2. Solve $Rx_p = y_p - Dx'_u$
3. Update $x_u = x'_u - \text{diag}(Q)^{-1}Gx_p$

3.3. SIMPLE-type preconditioners

The preconditioner P that corresponds to SIMPLE is given in Algorithm 1.

The derivation of SIMPLE (see Section 3.1.1) leads to the matrix $C + R$ on the left-hand side of the pressure correction equation in Step 2 of Algorithm 1. However, because C has a wide stencil, its construction is not straightforward on unstructured grids where the neighbors of a cell's neighbors are not readily available. One approach would be to take into account only the part of C that matches the compact stencil of R, but here, we choose to remove C altogether from the left-hand side of the pressure correction equation. Using R instead of $C + R$ in SIMPLE is our *first modification* to avoid the construction of C. This approximation is not as bad as it seems because Cp represents a fourth-order pressure derivative scaled with ϵh^2, which is likely to be small compared with the second-order derivative represented by Rp, at least for a smooth enough pressure field and a fine enough grid.

The action of P^{-1} implies the approximate solution of the velocity equation at Step 1 and of the pressure equation at Step 2. Therefore, only the construction (and storage) of Q and R is required; for D, G, and C, the action suffices. To solve the linear system associated with Q and R, we use the solvers and preconditioners from PETSc with a relative tolerance of 0.01. For Q, we use GMRES with Jacobi preconditioning, and for R, we use CG with IC (in parallel, the domain is decomposed into blocks, and we use CG with IC inside each block and Jacobi between the blocks).

The preconditioner P that corresponds to SIMPLER is given in Algorithm 2.

Instead of the usual pressure prediction at Step 1 of SIMPLER (see Section 3.1.3), we propose an alternative that does not include C on the left-hand side. Consider u^k to be known and write the pressure prediction as $p^k + p''$ so that

$$Qu^k + G(p^k + p'') = f.$$

Multiplying by $-D\,\text{diag}(Q)^{-1}$ then gives

$$-D\,\text{diag}(Q)^{-1}Gp'' = -D\,\text{diag}(Q)^{-1}(f - Qu^k - Gp^k) \quad \Leftrightarrow \quad Rp'' = -D\,\text{diag}(Q)^{-1}r_u^k.$$

The difference with the usual pressure prediction is that we do not involve the mass equation. Using this alternative in SIMPLER is our *second modification* to avoid the construction of C. Note that in SIMPLER, the relaxation factor ω_p in Equation (24) should only affect the pressure correction p' and not the pressure prediction $p^k + p''$, which explains the division by ω_p in Step 4 of Algorithm 2.

We will also consider the variation proposed in the context of finite element methods in [3, 8, 15] where the MSIMPLE approximation $Q^{-1} \approx M^{-1}$ replaces the SIMPLE approximation $Q^{-1} \approx$

Algorithm 2 Action $x = P^{-1}y$ of SIMPLER preconditioner

Given y_u and y_p

1. Solve $Rx''_p = -D\,\text{diag}(Q)^{-1}y_u$
2. Solve $Qx'_u = y_u - Gx''_p$
3. Solve $Rx'_p = y_p - Dx'_u - Cx''_p$
4. Update $x_u = x'_u - \text{diag}(Q)^{-1}Gx'_p$ and $x_p = x'_p + x''_p/\omega_p$

DOI: 10.1002/fld

$\text{diag}(Q)^{-1}$. Note that in finite volume methods, the mass-matrix M is the diagonal matrix of cell volumes. Because M does not depend on the nonlinear iteration, the main advantage of this approach is that $R = -DM^{-1}G$ does not need to be recomputed at every nonlinear iteration. It can be computed once and reused throughout the simulation. Obviously, we keep $\text{diag}(Q)$ in the PWIM; otherwise, we would be modifying the discretization as well as the iterative method.

3.4. Remarks on the iterative method

- As pointed out in Section 2, to construct Q at iteration k, the mass flux is assumed to be known from iteration $k-1$. Because of the PWIM, the mass flux depends on $\text{diag}(Q^{k-1})$, which, at iteration $k = 1$, is not available. Therefore, the simulation must start with a constant pressure field: the pressure gradient is then zero so that the pressure-weighting term drops out of Equation (2). To restart a simulation, it is important to have saved u, p, and $\text{diag}(Q)$ so that the mass flux can be exactly recomputed before constructing Q.
- Implicit relaxation [17] of the velocity is common when SIMPLE-type methods are applied to steady-state problems. If the velocity equation is written as $Qu^{k+1} = f$, then implicit relaxation with factor ω_i is given by

$$\left(Q + \frac{1-\omega_i}{\omega_i}\text{diag}(Q)\right)u^{k+1} = f + \frac{1-\omega_i}{\omega_i}\text{diag}(Q)u^k. \tag{25}$$

  Implicit relaxation is supplementary, it does not replace the explicit relaxation with parameters ω_u and ω_p in Equation (24). Thus, the velocity equation becomes $Q_{\omega_i}u = f_{\omega_i}$ and Q, and f are overwritten by Q_{ω_i} and f_{ω_i} in the implementation. Upon convergence, $u^{k+1} = u^k$, and the original equation is solved. Note that implicit relaxation does not affect the residual

$$r_u^k \equiv f - Qu^k - Gp^k = f_{\omega_i} - Q_{\omega_i}u^k - Gp^k$$

  so that Step 1 of Algorithm 1 simply becomes $Q_{\omega_i}x'_u = y_u$ (see also Section 3.1.2). However, when using SIMPLE as preconditioner, we also need the matrix–vector multiplication Ay that involves the matrix–vector multiplication Qy_u, not $Q_{\omega_i}y_u$. Therefore, Q should either not be overwritten or the contribution from ω_i should be removed from $Q_{\omega_i}y_u$.
- The success of SIMPLE hinges upon the existence of $\text{diag}(Q)^{-1}$ because this term is used in the pressure-weighted interpolation, in the implicit relaxation, Laplace operator, and velocity correction. We found that, at high Reynolds numbers, $\text{diag}(Q)$ can become very small in certain cells leading to sudden divergence. Therefore, the SIMPLE approximation $Q^{-1} \approx \text{diag}(Q)^{-1}$ is replaced by the more robust SIMPLEC approximation

$$Q^{-1} \approx \frac{1}{\frac{1}{2}\sum|Q|},$$

  where $\sum|Q|$ indicates the row sum of absolute values of Q. The factor $1/2$ is motivated by the observation that for a constant mass flux m, we have $\frac{1}{2}\sum|Q| = \text{diag}(Q)$. In the context of finite elements, the row sum of absolute values is also recommended in [2].

4. ACADEMIC TEST CASES

This section presents three popular test cases for Navier–Stokes solvers. These cases, being steady, two-dimensional laminar flows and restricted to Cartesian grids, can be reproduced with the discretization as presented in Section 2. The Reynolds number $\rho U_{\text{ref}}L/\mu$ varies from 10^2 to 10^5. In all cases, the unstructured QUICK scheme [22, 23] without limiter is used in defect correction form.

4.1. Flow over backward facing step ($Re = 100$)

This case simulates the flow over a backward facing step at very low Reynolds number. The domain is a rectangle of size $6L \times 2L$ from which a square of size L has been removed. The grid with $n \times m$ cells corresponds to a uniform distribution along the top and right boundary. The inflow

DOI: 10.1002/fld

condition is a parabolic velocity profile. The velocity is zero on the walls, and at the outflow, the pressure is zero.

4.2. *Lid-driven cavity flow (Re = 5000)*

This case simulates enclosed flow driven by a moving wall [24]. The domain is a square of size L. The grid with $n \times n$ cells is generated by applying the stretching function given by Equation (5.220) in [25] in both directions with parameters $a = 1/2$ and $b = 1.1$:

$$x = L\frac{(b+2a)c - b + 2a}{(2a+1)(1+c)}, \qquad c = \left(\frac{b+1}{b-1}\right)^{\left(\frac{\bar{x}-a}{1-a}\right)}, \qquad \bar{x} = 0, L/n, 2L/n, \ldots, L.$$

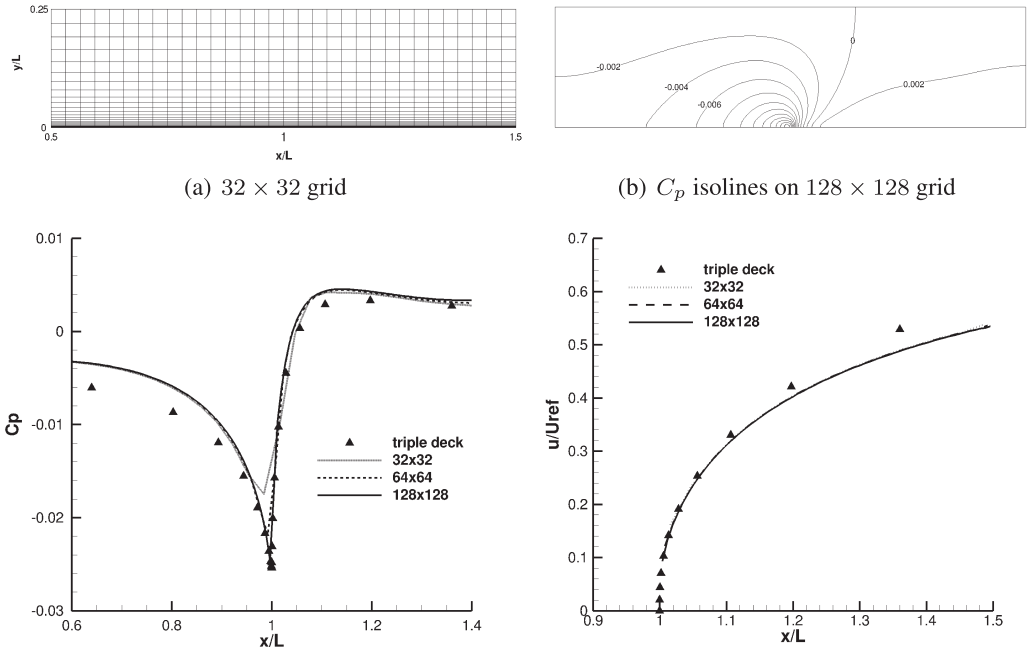
The velocity on the top boundary is $(U_{\text{ref}}, 0)$ and zero on the other boundaries.

4.3. *Flow over finite flat plate (Re = 100 000)*

This case is taken from [26]; it simulates the flow over the second half of a flat plate of finite length L and its wake. The domain is a rectangle of size $L \times 0.25L$. The grid with $n \times n$ cells is generated by applying the stretching function given by Equation (5.216) in [25] in y-direction with parameter $b = 1.01$ and domain height $H = 0.25L$:

$$x = H\frac{(b+1) - (b-1)c}{(c+1)}, \qquad c = \left(\frac{b+1}{b-1}\right)^{(1-\bar{x})}, \qquad \bar{x} = 0, H/n, 2H/n, \ldots, H.$$

At the inflow, (an approximation to) the Blasius velocity profile [27] is prescribed. The no-slip condition is imposed on the plate ($0.5 \leqslant x/L \leqslant 1$), and the free-slip condition is imposed in the wake ($1 \leqslant x/L \leqslant 1.5$). On the top boundary, the pressure is set to zero, and Neumann conditions are applied at the outflow boundary. Because this case is less common, an impression of the grid and flow field is given in Figure 1, including a comparison with triple deck theory from [28].



(a) 32×32 grid

(b) C_p isolines on 128×128 grid

(c) C_p profile at $y = 0$

(d) u profile at $y = 0$

Figure 1. Grid and flow field impression of the flow over flat plate. (a) 32×32 grid; (b) C_p isolines on 128×128 grid; (c) C_p profile at $y = 0$; (d) u profile at $y = 0$.

DOI: 10.1002/fld

4.4. Numerical results

SIMPLE-type methods are more often used as solver than as preconditioner; in which case, P^{-1} replaces $\tilde{A}^{-1}$ in Equation (24), and a single SIMPLE step is used per nonlinear iteration. Here, we compare both approaches for the four variants (M)SIMPLE(R). We call these variants KRYLOV-(M)SIMPLE(R) when used as preconditioner.

The influence of the relaxation parameters is considered first. The equations are solved in stationary form; no (pseudo) time derivative is used. The implicit relaxation parameter is fixed ($\omega_i = 0.9$), and the explicit relaxation parameters for the velocity (ω_u) and for the pressure (ω_p) are varied manually (see [29] for an automatic variation of ω_p). The relative tolerance for solving the linear subsystems in P^{-1} is set to 0.01 in accordance with [3]. The results for (M)SIMPLE(R) as solver are shown in Table I and for (M)SIMPLE(R) as preconditioner in Table II. We compare the number of nonlinear iterations needed to converge the test cases to machine precision, starting from uniform flow. In the KRYLOV-(M)SIMPLE(R) cases, we also show the average number of linear iterations per nonlinear iteration needed to solve the linear system $Ax' = r$ up to a relative tolerance of 0.1.

Table I shows that, as expected, SIMPLE is highly sensitive to the relaxation parameters, notably ω_p. The only choice that gives convergence is $\omega_p = 0.2$. SIMPLER is much less sensitive to the relaxation parameter: it does not diverge for any of the parameter choices. The choice of ω_p is almost irrelevant, but the choice of ω_u still influences the convergence. However, SIMPLER does not reduce the number of nonlinear iterations, except for the lid-driven cavity. MSIMPLE and MSIMPLER are clearly poor solvers. Table II shows that the use of (M)SIMPLE(R) as preconditioner greatly reduces the number of nonlinear iterations compared with their use as solver. The reduction in nonlinear iterations should outweigh the additional cost of using the SIMPLE-type methods as preconditioner instead of solver; this issue will be addressed for the maritime test cases in Section 5 because the academic test cases are too small to give meaningful CPU time measurements. Table II also shows that the choice of relaxation parameters still has a significant effect; the optimal choice seems to be $\omega_u = 1.0$ and $\omega_p = 0.5$. We also notice that the MSIMPLE(R) variants perform remarkably well as preconditioner given their poor performance as solvers. The (M)SIMPLER variants require much less linear iterations per nonlinear iteration than the (M)SIMPLE variants. Clearly, the most promising variants are KRYLOV-(M)SIMPLER, so we continue by studying these in more detail.

The effect of grid refinement is shown in Table III. The trend is that the number of linear iterations per nonlinear iteration increases with grid refinement. The largest increase is observed between the

Table I. Number of nonlinear iterations needed to converge the academic cases to machine precision, starting from uniform flow.

Case	ω_u	ω_p	SIMPLE	SIMPLER	MSIMPLE	MSIMPLER
BFS (96×48)	1.0	1.0	—	470	796	508
	1.0	0.5	—	468	1608	948
	1.0	0.2	475	471	4040	2170
	0.7	1.0	—	672	789	701
	0.7	0.5	—	670	1602	958
	0.7	0.2	666	670	4033	2194
LDCF (128×128)	1.0	1.0	—	2516	>10000	4158
	1.0	0.5	—	3394	>10000	4079
	1.0	0.2	5241	3374	>10000	4769
	0.7	1.0	—	5433	>10000	5208
	0.7	0.5	—	5105	>10000	5296
	0.7	0.2	7011	4862	>10000	5393
FP (128×128)	1.0	1.0	—	690	2089	741
	1.0	0.5	—	690	4662	1011
	1.0	0.2	687	688	>10000	1590
	0.7	1.0	—	990	2203	1065
	0.7	0.5	—	990	4536	1577
	0.7	0.2	987	988	>10000	8392

DOI: 10.1002/fld

Table II. Number of nonlinear iterations needed to converge the academic cases to machine precision, starting from uniform flow. Between brackets is the average number of linear iterations per nonlinear iteration.

Case	ω_u	ω_p	KRYLOV SIMPLE	KRYLOV SIMPLER	KRYLOV MSIMPLE	KRYLOV MSIMPLER
BFS (96×48)	1.0	1.0	60 (20.3)	63 (5.5)	41 (13.0)	52 (10.6)
	1.0	0.5	51 (25.4)	50 (9.7)	42 (14.0)	42 (10.7)
	1.0	0.2	108 (14.3)	111 (6.9)	110 (13.2)	109 (16.2)
	0.7	1.0	66 (28.1)	61 (10.0)	62 (12.5)	64 (12.2)
	0.7	0.5	68 (28.1)	67 (11.3)	60 (14.7)	59 (14.4)
	0.7	0.2	111 (16.3)	114 (7.5)	109 (13.7)	112 (15.2)
LDCF (128×128)	1.0	1.0	371 (28.8)	176 (16.4)	134 (21.9)	155 (16.3)
	1.0	0.5	462 (22.1)	381 (8.4)	121 (28.7)	347 (9.0)
	1.0	0.2	1052 (11.3)	1026 (5.2)	233 (28.8)	311 (15.8)
	0.7	1.0	374 (28.8)	193 (22.5)	137 (26.0)	151 (16.2)
	0.7	0.5	456 (28.7)	365 (11.1)	169 (23.2)	312 (11.9)
	0.7	0.2	1012 (14.3)	914 (5.7)	168 (28.6)	712 (7.9)
FP (128×128)	1.0	1.0	150 (15.6)	202 (3.5)	278 (9.0)	144 (4.1)
	1.0	0.5	53 (26.8)	46 (13.1)	47 (24.3)	62 (8.6)
	1.0	0.2	113 (19.5)	106 (9.9)	99 (24.9)	—
	0.7	1.0	120 (18.7)	106 (8.3)	66 (14.1)	82 (8.2)
	0.7	0.5	59 (28.2)	50 (16.4)	52 (24.0)	50 (14.1)
	0.7	0.2	104 (26.2)	104 (10.7)	99 (24.9)	182 (9.2)

Table III. Number of nonlinear iterations needed to converge the academic cases to machine precision, starting from uniform flow. Between brackets is the average number of linear iterations per nonlinear iteration.

Case	grid	KRYLOV-SIMPLER	KRYLOV-MSIMPLER
BFS	24×12	73 (3.1)	54 (4.0)
	48×24	56 (4.7)	48 (5.9)
	96×48	50 (9.7)	42 (10.7)
LDCF	16×16	140 (3.9)	152 (5.2)
	32×32	344 (4.2)	346 (5.7)
	64×64	383 (5.0)	323 (6.3)
	128×128	381 (8.4)	347 (9.0)
FP	32×32	45 (3.3)	80 (2.5)
	64×64	46 (6.2)	88 (3.9)
	128×128	46 (13.1)	62 (8.6)

two finest grids: quadrupling the number of cells roughly doubles the number of linear iterations per nonlinear iteration.

The effect of the relative tolerance for the solution of the coupled linear system is shown in Table IV. The number of nonlinear iterations drops as the tolerance is tightened, notably for the lid-driven cavity case, but the decrease in nonlinear iterations is modest while the number of linear iterations per nonlinear iterations roughly doubles each time the relative tolerance is decreased by a factor 10. Therefore, it may not pay-off to solve the linear system very accurately.

The overall behavior is as expected: the SIMPLE-type methods are less sensitive to relaxation parameters when used as preconditioner and converge ten times faster in terms of nonlinear iterations. One effectively 'trades' nonlinear iterations for linear iterations. Our alternative pressure prediction further reduces the number of linear iterations without increasing the number of nonlinear iterations. In the next section, we will see whether such trade-offs lead to significant gain in CPU time. For the time being, the main conclusion is that the two modifications needed to avoid the construction of the stabilization matrix are not detrimental to the performance.

DOI: 10.1002/fld

Table IV. Number of nonlinear iterations needed to converge the academic cases to machine precision, starting from uniform flow. Between brackets is the average number of linear iterations per nonlinear iteration.

Case	tol. system	KRYLOV-SIMPLER	KRYLOV-MSIMPLER
BFS (96×48)	0.100	50 (9.7)	42 (10.7)
	0.010	44 (16.8)	44 (24.6)
	0.001	44 (23.0)	44 (36.3)
LDCF (128×128)	0.100	381 (8.4)	347 (9.0)
	0.010	136 (39.2)	129 (37.2)
	0.001	105 (74.8)	103 (72.0)
FP (128×128)	0.100	46 (13.1)	62 (8.6)
	0.010	41 (22.5)	42 (21.2)
	0.001	44 (50.6)	45 (42.8)

5. MARITIME TEST CASES

In this section, we consider the prediction of the viscous resistance of ship hulls. An impression of the flow field around a ship hull in the vicinity of the stern (aft end) is given in Figure 2. It shows streamlines around the stern and the axial velocity field in the wake.

The block-structured or unstructured hexahedral grids used in this section require eccentricity and nonorthogonality corrections for which we refer to [17]. Furthermore, the k-omega SST turbulence model [30] is added. The turbulence model equations are solved in a segregated manner: the eddy viscosity is assumed to be known when solving the mass and momentum equations, whereas the velocity and pressure are assumed to be known when solving the turbulence kinetic energy and turbulence dissipation equations.

While experimenting with maritime test cases, we encountered two problems:

1. The MSIMPLE-type methods did not converge; the reason being that the pressure equation could not be solved at all, not even when using the converged solution as initial condition instead of uniform flow.
2. The relative tolerance of 0.1 for the solution of the coupled linear system could not be reached. Figure 3 shows a typical example of the linear convergence using a maximum number of 30 linear iterations (without restart) per nonlinear iteration. Most of the reduction is reached in the first 5–10 iterations; the linear convergence stagnates after 15 linear iterations. Decreasing the relative tolerance for the linear susbsystems from 0.1 to 0.01 did not improve the convergence of the coupled linear system.

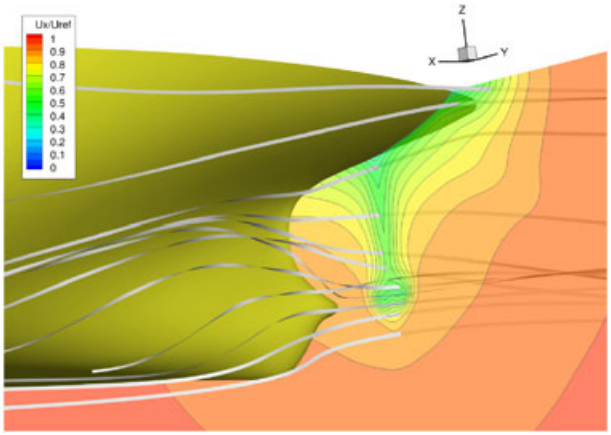


Figure 2. Impression of the flow field at the stern of the full-scale tanker.

DOI: 10.1002/fld

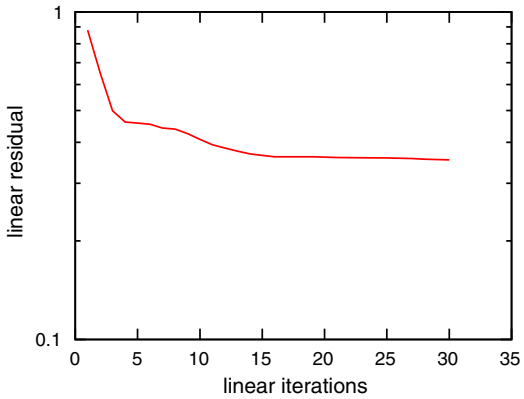
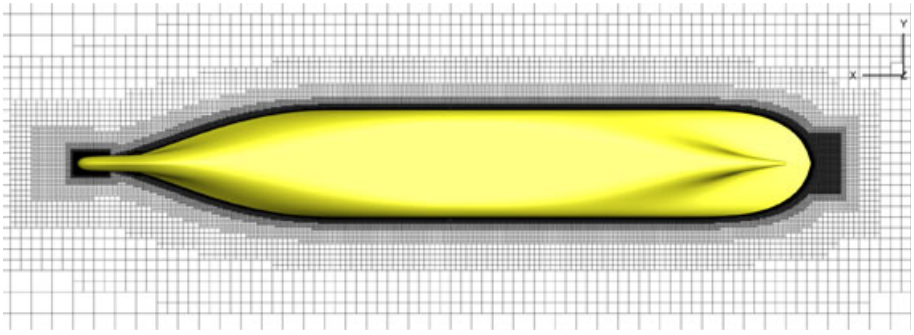


Figure 3. Linear convergence with FGMRES of the coupled system at nonlinear iteration 100 for the model-scale tanker on the medium grid.

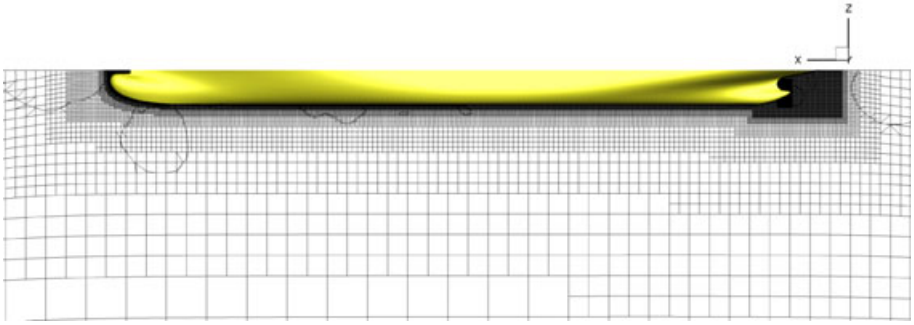
Both problems are probably linked to the grid stretching near the hull that gives very small cells with very large aspect ratio; this issue remains to be investigated. Because of these two problems, we only compare SIMPLE to KRYLOV-SIMPLER with a *fixed number* of 5 linear iterations per nonlinear iteration. As we will see, the nonlinear iterations do converge with this fixed number of linear iterations instead of a relative tolerance.

5.1. Model-scale container vessel ($Re = 1.35 \times 10^7$)

This case simulates the flow around a container vessel hull at towing tank scale 1/50 on an unstructured hexahedral grid, generated with Hexpress [31]. The grid has 13.3 m cells. The clustering



(a) water plane



(b) symmetry plane

Figure 4. Impression of the grid (13.3 m cells) around the container vessel. (a) water plane; (b) symmetry plane.

DOI: 10.1002/fld

towards the wall is such that the maximum cell aspect ratio is 1:1600 and the maximum y^+ is 0.95, so wall-functions are not needed. An impression of the grid is given in Figure 4.

For both solvers, the implicit relaxation parameter is $\omega_i = 0.9$, and the relative tolerance for the solution of the linear systems is 0.01. For SIMPLE as solver, the explicit relaxation parameters are $\omega_u = 0.2$ for momentum and turbulence and $\omega_p = 0.1$ for the pressure. For KRYLOV-SIMPLER, the explicit relaxation parameters are $\omega_u = 0.8$ for momentum and turbulence and $\omega_p = 0.3$ for the pressure. The chosen relaxation parameters are sharp: slightly higher values lead to divergence for SIMPLE and stagnation for KRYLOV-SIMPLER.

The convergence results are shown in Table V and in Figure 5. In this case, KRYLOV-SIMPLER requires seven times fewer nonlinear iterations to convergence than SIMPLE. Such a reduction agrees with the results obtained for the academic cases; it gives a modest saving in CPU time.

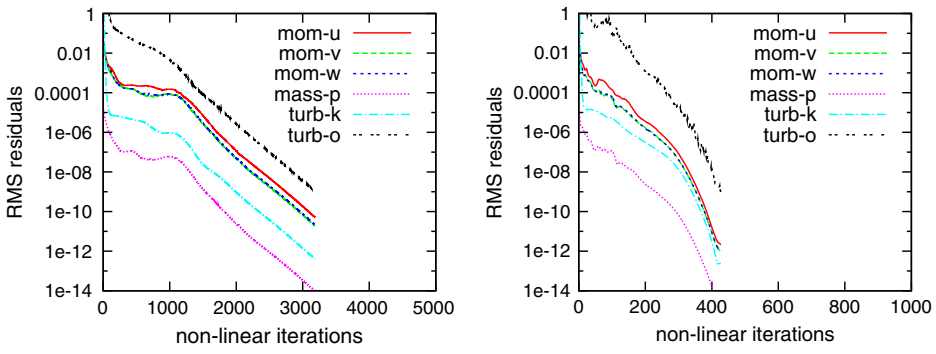
5.2. *Model-scale tanker ($Re = 4.6 \times 10^6$)*

This case simulates the flow around a tanker hull at towing tank scale 1/50 on a series of grids. The block-structured grids were generated with GridPro [32]. Four grids are considered: a very coarse grid with 250 k cells, a coarse grid with 500 k cells, a medium grid with 1 m cells and a fine grid with 2 m cells. The maximum y^+ is 1.8 on the very coarse grid and goes down to 0.9 on the fine grid. The maximum cell aspect ratio is 1 : 7 000. An impression of the fine grid is given in Figure 6.

The relaxation parameters are the same as for the container vessel. Usually, one would take the coarse grid solution as initial condition on the finer grid, but here, we choose to start from uniform flow on all grids to illustrate the robustness of the solvers. The results are given in Table VI and Figure 7. Again, we find a significant reduction of the number of nonlinear iterations when using KRYLOV-SIMPLER. We also see that the convergence rate is not grid independent (as expected), but the increase in nonlinear iterations is modest: for KRYLOV-SIMPLER, the first three grids give similar counts, and only the finest grid requires significantly more iterations. We also find that KRYLOV-SIMPLER reduces the required CPU time roughly by half on the finest grids.

Table V. Number of nonlinear iterations and wall clock time needed to converge the container vessel case, starting from uniform flow.

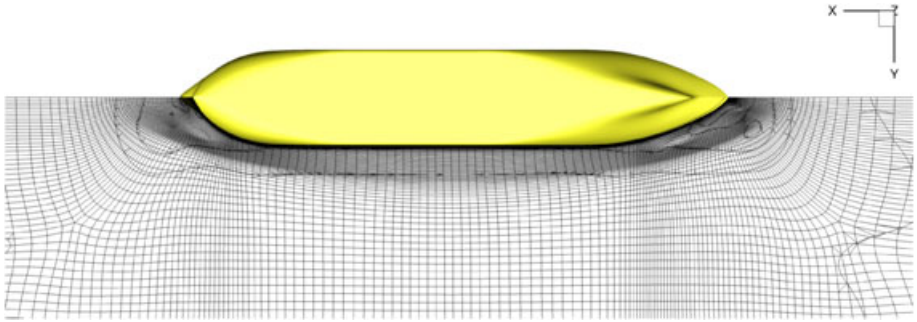
		SIMPLE		KRYLOV-SIMPLER	
Grid	CPU cores	# its	Wall clock	# its	Wall clock
13.3 m	128	3187	5 h 26 min	427	3 h 27 min



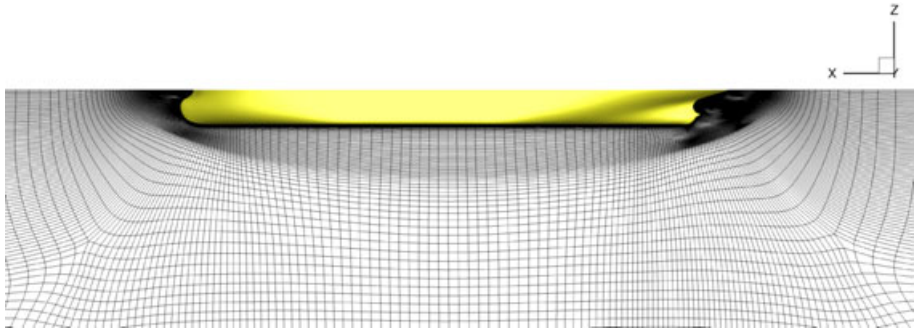
bare hull (13.3 m cells)

Figure 5. Convergence of the container vessel case with SIMPLE (left column) and KRYLOV-SIMPLER (right column). Notice the factor 5 in the scaling of the x-axis. (a) bare hull (13.3 m cells).

DOI: 10.1002/fld



(a) water plane



(b) symmetry plane

Figure 6. Impression of the fine grid (2 m cells) around the tanker hull. (a) water plane; (b) symmetry plane.

Table VI. Number of nonlinear iterations and wall clock time needed to convergence the model-scale tanker case to machine precision, each case starting from uniform flow.

		SIMPLE		KRYLOV-SIMPLER	
Grid	CPU cores	# its	Wall clock	# its	Wall clock
0.25 m	8	1379	25 min	316	29 min
0.5 m	16	1690	37 min	271	25 min
1 m	32	2442	57 min	303	35 min
2 m	64	3534	1 h 29 min	519	51 min

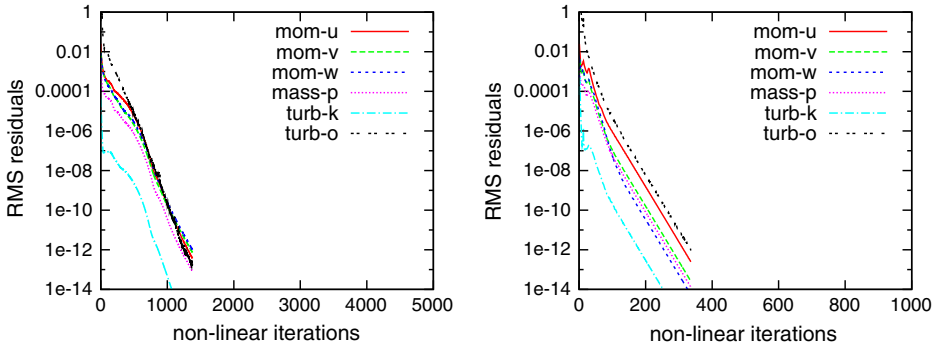
5.3. Full-scale tanker ($Re = 2 \times 10^9$)

Here, we consider the same tanker hull but now at full scale. The fine grid from the model-scale case was adjusted to full scale. This gives a grid of 2.7 m cells with a maximum y^+ of 1.2 and a maximum cell aspect ratio of 1 : 930 000.

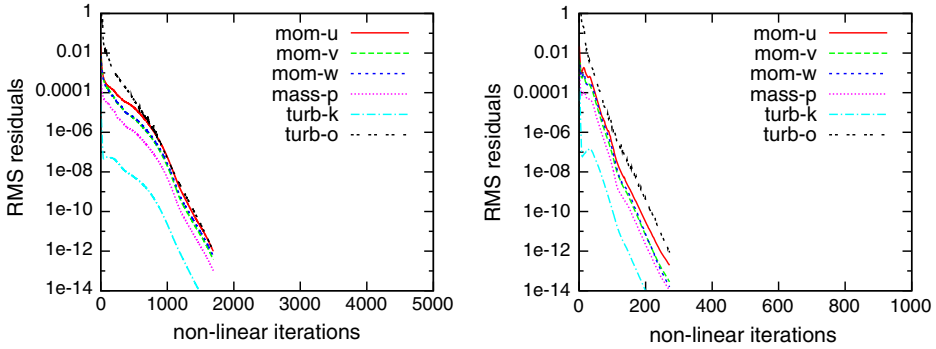
Unfortunately, we could not obtain good convergence with the QUICK scheme: the QUICK scheme without limiter that was used in all previous calculations diverges, with limiter it stagnates. Therefore, in this case only, advection is treated with a blending between the upwind scheme (25%) and the central scheme (75%). This is also the only case where we cannot start from uniform flow; instead, the upwind scheme was used to generate an initial condition.

For both solvers, the implicit relaxation parameter is $\omega = 0.8$, and the relative tolerance for the solution of the linear systems is 0.01. For SIMPLE as solver, the explicit relaxation parameters are $\omega_u = 0.1$ for momentum and turbulence and $\omega_p = 0.05$ for the pressure. For KRYLOV-SIMPLER, the explicit relaxation parameters are $\omega_u = 0.7$ for momentum and turbulence and $\omega_p = 0.3$ for the pressure.

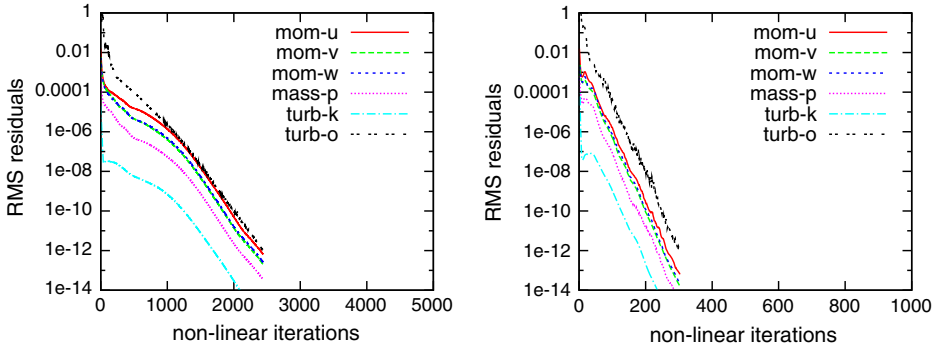
DOI: 10.1002/fld



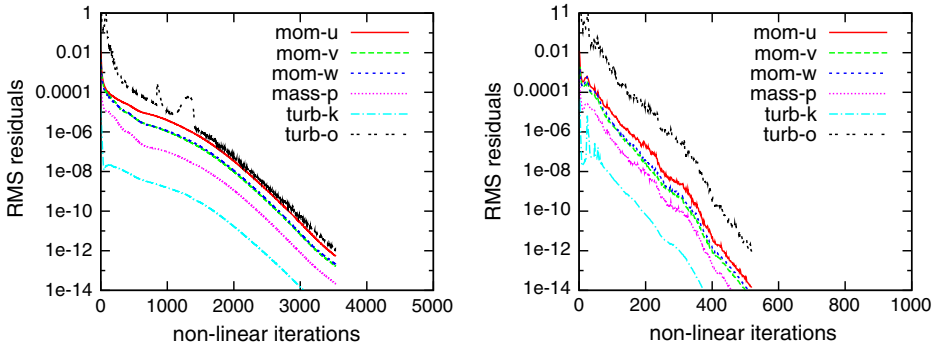
(a) very coarse grid (0.25 m cells)



(b) coarse grid (0.5 m cells)



(c) medium grid (1 m cells)



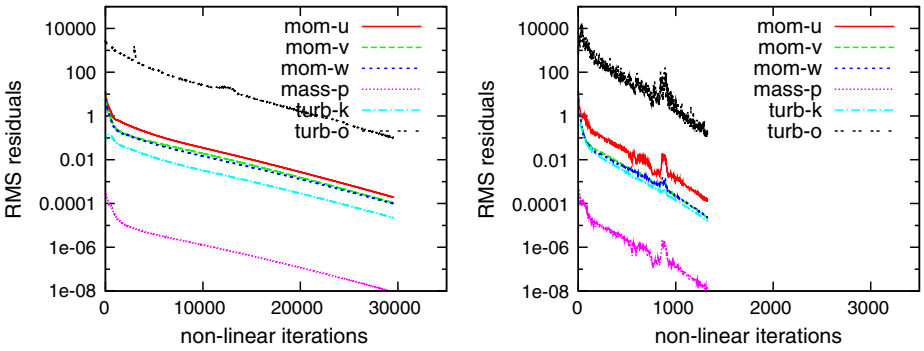
(d) fine grid (2 m cells)

Figure 7. Convergence of the model-scale tanker case with SIMPLE (left column) and KRYLOV-SIMPLER (right column). Notice the factor 5 in the scaling of the x-axis. (a) very coarse grid (0.25 m cells); (b) coarse grid (0.5 m cells); (c) medium grid (1 m cells); (d) fine grid (2 m cells).

DOI: 10.1002/fld

Table VII. Number of nonlinear iterations and wall clock time needed to converge the full-scale tanker case, starting from an initial flow field obtained with the upwind scheme.

		SIMPLE		KRYLOV-SIMPLER	
Grid	CPU cores	# its	Wall clock	# its	Wall clock
2.7 m	64	29 578	16 h 37 min	1330	3 h 05 min



bare hull (2.7 m cells)

Figure 8. Convergence of the full-scale tanker with SIMPLE (left column) and KRYLOV-SIMPLER (right column). Notice the factor 10 in the scaling of the x-axis. (a) bare hull (2.7 m cells).

The results are given in Table VII and Figure 8. The benefit of KRYLOV-SIMPLER is most pronounced in this case: 20 times less nonlinear iterations and 5 times less CPU time. Clearly, for high Reynolds number and cell aspect ratio, the advantage of KRYLOV-SIMPLER is beyond question.

6. CONCLUSION

In this paper, we compared the convergence of four SIMPLE-type methods, applied as solver and as preconditioner to a number of academic and industrial test cases. Using the methods as preconditioner is not as straightforward, but the additional complexity pays off: the methods are less sensitive to relaxation parameters and converge faster, both in terms of nonlinear iterations and in terms of CPU time.

To apply these methods in the context of cell-centered, colocated variables, we proposed two modifications to avoid the construction of the stabilization matrix. We thereby maintain the compact stencil needed for applications on unstructured grids.

For the academic test cases, we found that the additional pressure prediction in SIMPLER-type preconditioners is worthwhile. It does not reduce the number of nonlinear iterations, but it significantly reduces the average number of linear iterations per nonlinear iteration. Furthermore, we found MSIMPLER to be a poor solver but a good preconditioner; its performance is comparable with SIMPLER but with the benefit that the Schur complement approximation is built only once instead of every nonlinear iteration.

There is a large gap between the well-behaved academic test cases and the maritime test cases. For the latter, we did not obtain any convergence with MSIMPLER preconditioners. With SIMPLER preconditioners, we could not solve the linear system up to the desired relative tolerance of 0.1. These problems are probably related to the very small cells with very high aspect ratio found in maritime applications. We did not address this issue but instead compared the classical SIMPLE

DOI: 10.1002/fld

solver with SIMPLER as preconditioner with a fixed number of 5 linear iterations per nonlinear iteration. Depending on the case, we found that SIMPLER as preconditioner can reduce the nonlinear iterations by a factor 5–20 and the CPU time by a factor 2–5. Most importantly, the speedup increases with the complexity. The factor five reduction in CPU time is found for the most challenging test case—the flow around a ship hull at Reynolds number 2×10^9 on a grid with maximum cell aspect ratio of 1 : 930 000.

ACKNOWLEDGEMENTS

The authors would like to thank Guus Segal and Auke van der Ploeg for their valuable comments.

REFERENCES

1. Vaz G, Jaouen F, Hoekstra M. Free-surface viscous flow computations. Validation of URANS code FreSCo. *Proceedings of ASME 28th International Conference on Ocean, Offshore and Arctic Engineering*, Honolulu, Hawaii, May 31–June 5, 2009; 425–437.
2. Elman H, Howle V, Shadid J, Shuttleworth R, Tuminaro R. A taxonomy and comparison of parallel block multilevel preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics* 2008; **227**:1790–1808.
3. Segal A, ur Rehman M, Vuik C. Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics: Theory, Methods and Applications* 2010; **3**:245–275.
4. Kay D, Loghin D, Wathen A. A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal on Scientific Computing* 2002; **24**(1):237–256.
5. Elman H, Howle V, Shadid J, Shuttleworth R, Tuminaro R. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing* 2006; **27**(5):1651–1668.
6. Elman H, Howle V, Shadid J, Silvester D, Tuminaro R. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. *SIAM Journal on Scientific Computing* 2007; **30**(1):290–311.
7. Benzi M, Olshanskii M. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing 2006* 2006; **28**(6):2095–2113.
8. ur Rehman M, Vuik C, Segal G. SIMPLE-type preconditioners for the Oseen problem. *International Journal for Numerical Methods in Fluids* 2009; **61**:432–452.
9. Vuik C, Saghir A, Boerstoel G. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *International Journal for Numerical Methods in Fluids* 2000; **33**:1027–1040.
10. de Niet A, Wubs F. Two preconditioners for saddle point problems in fluid flows. *International Journal for Numerical Methods in Fluids* 2007; **54**(4):355–377.
11. Vanka S. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**(1):29–48.
12. Volker J, Lutz T. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids* 2000; **33**(4):453–473.
13. Patankar S, Spalding D. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal Of Heat And Mass Transfer* 1972; **15**:1787–1806.
14. Patankar S. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill: McGraw-Hill New York, 1980.
15. ur Rehman M. Fast iterative methods for the incompressible Navier-Stokes equations. *Ph.D. Thesis*, Technische Universiteit Delft, Delft, The Netherlands, 2010.
16. Wesseling P. *Principles of Computational Fluid Dynamics*. Springer: Springer-Verlag Berlin Heidelberg New York, 2001.
17. Ferziger J, Perić M. *Computational Methods for Fluid Dynamics*, 3rd ed. Springer: Springer-Verlag Berlin Heidelberg New York, 2002.
18. Aris R. *Vectors, Tensors and the Basic Equations of Fluid Mechanics*. Dover: Dover New York, 1989.
19. Miller T, Schmidt F. Use of a pressure-weighted interpolation method for the solution of the incompressible Navier-Stokes equations on a nonstaggered grid system. *Numerical Heat Transfer* 1988; **14**:213–233.
20. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal On Scientific Computing* 1993; **14**:461–469.
21. Balay S, Buschelman K, Gropp W, Kaushik D, Knepley M, Curfman-McInnes L, Smith B, Zhang H. PETSc Web page, 2011. http://www.mcs.anl.gov/petsc [access date July 1, 2011].
22. Barth T, Jespersen D. The design and application of upwind schemes on unstructured meshes. *AIAA Paper 89-0366* 1989. Aerospace Sciences Meeting, 27th, Reno, NV, Jan 9–12, 1989.
23. Waterson N, Deconinck H. Design principles for bounded higher-order convection schemes – a unified approach. *Journal of Computational Physics* 2007; **224**:182–207.
24. Ghia U, Ghia K, Shin C. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics* 1982; **48**:387–411.

DOI: 10.1002/fld

25. Tannehill J, Anderson D, Pletcher R. *Computational Fluid Mechanics and Heat Transfer*, 2nd ed. Taylor & Francis: Taylor & Francis Levittown, 1997.
26. Hoekstra M. Numerical simulation of ship stern flows with a space-marching Navier-Stokes method. *Ph.D. Thesis*, Technische Universiteit Delft, Delft, The Netherlands, 1999.
27. Schlichting H. *Boundary-layer Theory*, 6th ed. McGraw-Hill, Inc: McGraw-Hill New York, 1968.
28. Veldman A. Boundary layer flow past a finite flat plate. *Ph.D. Thesis*, Rijksuniversiteit Groningen, Groningen, The Netherlands, 1976.
29. Houzeaux G, Aubry R, Vázquez M. Extension of fractional step techniques for incompressible flows: the preconditioned Orthomin(1) for the pressure Schur complement. *Computers & Fluids* 2011; **44**(1):297–313.
30. Menter F. Performance of popular turbulence models for attached and separated adverse pressure gradient flows. *AIAA Journal* 1992; **30**(8):2066–2072.
31. Numeca webpage, 2011. http://www.numeca.com/.
32. GridPro webpage, 2011. http://www.gridpro.com/ [access date July 1, 2011].

DOI: 10.1002/fld