Bergen Center for Computational Science

BCCS
TECHNICAL REPORT SERIES

Scalable linear solvers for the non-hydrostatic pressure calculation in the Bergen Ocean Model

Torsvik, T.

REPORT No. 23 **September 24, 2008**

UNIFOB
the University of Bergen research company

BERGEN, NORWAY

BCCS Technical Report Series is available at http://www.bccs.no/publications/

Requests for paper copies of this report can be sent to:

Bergen Center for Computational Science, Høyteknologisenteret,
Thormøhlensgate 55, N-5008 Bergen, Norway

Abstract

Calculation of the non-hydrostatic pressure term is usually associated with a sparse linear system of equations. Such systems can be solved efficiently by iterative linear solvers. Extra care should be taken in a parallel environment to ensure that the method scales well with the number of available CPUs. In this report we compare the performance of a classical linear solver with more elaborate methods that are available from an external library. We study the stability, accuracy, and execution time for the solvers using four test cases of different sizes and with different physical properties.

Contents

1 Introduction

The continuing increase in computer power allows ocean models to be run with increasingly higher resolution. With horizontal grid spacing of the order 1 km or less, models are starting to resolve processes that depend on non-hydrostatic pressure effects. The non-hydrostatic pressure term introduces an elliptic PDE which generally requires the solution of a sparse linear system of equations. Several iterative algorithms have been developed to solve this type of problem. The choice of iterative method is often a trade off between efficiency and stability. Methods that are guaranteed to converge usually have a slow rate of convergence, whereas more efficient methods may not converge for all problems. Algorithms designed for parallel computing should also have good scaling properties, i.e. the method should be able to take advantage of all the available CPUs assigned for the computation.

A non-hydrostatic version of BOM has already been developed [4, 8]. This version used the *successive over-relaxation* (SOR) method to calculate the non-hydrostatic pressure contribution. A slightly modified version of SOR is also used as the default solver in the parallelized version of BOM. Other solvers are available through the HYPRE library of scalable, high performance linear solvers. At the time of writing, the methods have been tested on three 2D test cases and one 3D test case.

The rest of this section contains an introduction of the mathematical and numerical models. Section 2 presents the different type of methods used for solving sparse matrix systems. A presentation of the test cases and results from model runs follows in section 3.

1.1 Governing equations for a 2D ocean model

Here we present the governing 2D equations in a Cartesian coordinate system (x, z), consisting of one horizontal and one vertical axis. The system can easily be extended to 3D by including a second horizontal dimension, and the Coriolis force. Assuming the fluid is incompressible, the continuity equation becomes

$$\frac{\partial U}{\partial x} + \frac{\partial W}{\partial z} = 0\,, \tag{1}$$

where U is the horizontal velocity and W is the vertical velocity. In the ocean, where density differences usually are small compared to the mean density, it is common to apply the Boussinesq approximation, which states that the density ρ can be replaced by the mean density ρ_0, except when it is related to the gravitational force. This gives us the Reynolds momentum equations

$$\frac{\partial U}{\partial t} + \frac{\partial U^2}{\partial x} + \frac{\partial UW}{\partial z} = \frac{1}{\rho_0}\frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left(A_M \frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial z}\left(K_M \frac{\partial U}{\partial z}\right)\,, \tag{2}$$

$$\frac{\partial W}{\partial t} + \frac{\partial UW}{\partial x} + \frac{\partial W^2}{\partial z} = \frac{1}{\rho_0}\frac{\partial P}{\partial z} + \frac{\rho g}{\rho_0} + \frac{\partial}{\partial x}\left(A_M \frac{\partial W}{\partial x}\right) + \frac{\partial}{\partial z}\left(K_M \frac{\partial W}{\partial z}\right)\,, \tag{3}$$

where P is the pressure, g is the acceleration of gravity, and A_M and K_M are the horizontal and vertical viscosity coefficients, respectively. If we also include the hydrostatic approximation, which states that the pressure at a certain depth depends only on the weight of water above it, eq. (3) becomes

$$\rho g = -\frac{\partial P}{\partial z}\,. \tag{4}$$

The general conservation equation for a state variable can be written as

$$\frac{\partial (\cdot)}{\partial t} + \frac{\partial U(\cdot)}{\partial x} + \frac{\partial W(\cdot)}{\partial z} = A_H \frac{\partial^2 (\cdot)}{\partial x^2} + K_H \frac{\partial^2 (\cdot)}{\partial z^2}\,, \tag{5}$$

where $(\cdot)$ represent a particular physical component in the model, and A_H and K_H are the horizontal and vertical diffusivity coefficients, respectively. In a homogeneous fluid where there are no temperature and salinity gradients, density is determined by setting $(\cdot) = (\rho)$ in eq. (5). In inhomogeneous fluid the density will usually depend on both temperature T and salinity S, determined by an equation of state

$$\rho = \rho(T, S, P)\,, \tag{6}$$

taken from the literature[6, 11], and temperature and salinity are determined by $(\cdot) = (T, S)$ in eq. (5). In order to obtain a closed set of equations, we need to specify the coefficients A_M, K_M, A_H, and K_H. This is far from a trivial task when designing a realistic ocean model, but for our purposes it is sufficient to use constant values for these coefficients.

At the free surface and the bottom we need to specify the kinematic boundary conditions, given by

$$W = \frac{\partial \eta}{\partial t} + U\frac{\partial \eta}{\partial x}\,, \qquad z = \eta(x,t)\,, \tag{7}$$

$$W = -U\frac{\partial H}{\partial x}\,, \qquad z = -H(x)\,, \tag{8}$$

and at lateral boundaries we apply the no-flow condition

$$U \cdot \mathbf{n} = 0\,, \tag{9}$$

where $\mathbf{n}$ is the outer normal unit vector.

1.2 Equations for a terrain-following model

The basic design of the terrain-following, or σ–coordinate, model was outlined by Blumberg and Mellor[5]. A transformation of the original equations in terms of Cartesian coordinates (x_C, z_C, t_C), into σ–coordinates (x, σ, t), is accomplished by

$$x = x_C\,, \qquad \sigma = \frac{z_C - \eta}{D}\,, \qquad t = t_C\,, \tag{10}$$

where the dynamic depth is defined as

$$D = H + \eta\,. \tag{11}$$

The derivatives of a dependent variable $A_C(x_C, z_C, t_C)$ are transformed into the σ–coordinate system $A(x, \sigma, t)$ by

$$\begin{aligned}
\frac{\partial A_C}{\partial x_C} &= \frac{\partial A}{\partial x} - \frac{1}{D}\left(\sigma\frac{\partial D}{\partial x_C} + \frac{\partial \eta}{\partial x}\right)\frac{\partial A}{\partial \sigma}\,,\\
\frac{\partial A_C}{\partial z_C} &= \frac{1}{D}\frac{\partial A}{\partial \sigma}\,,\\
\frac{\partial A_C}{\partial t_C} &= \frac{\partial A}{\partial t} - \frac{1+\sigma}{D}\frac{\partial \eta}{\partial t}\frac{\partial A}{\partial \sigma}\,.
\end{aligned} \tag{12}$$

The velocity perpendicular to iso–σ surfaces is defined by

$$\omega = W - U\left(\sigma\frac{\partial D}{\partial x} + \frac{\partial \eta}{\partial x}\right) - (1+\sigma)\frac{\partial \eta}{\partial t}\,. \tag{13}$$

The governing equations (1), (2), and (3) become

$$\frac{\partial UD}{\partial x} + \frac{\partial \omega}{\partial \sigma} + \frac{\partial \eta}{\partial t} = 0\,, \tag{14}$$

$$\frac{\partial UD}{\partial t} + \frac{\partial U^2 D}{\partial x} + \frac{\partial U\omega}{\partial \sigma} = -\frac{D}{\rho_0}\frac{\partial P}{\partial x} + \frac{1}{\rho_0}\frac{\partial P}{\partial \sigma}\left(\sigma\frac{\partial D}{\partial x} + \frac{\partial \eta}{\partial x}\right) + DF_x\,, \tag{15}$$

$$\frac{\partial WD}{\partial t} + \frac{\partial UWD}{\partial x} + \frac{\partial W\omega}{\partial \sigma} = -\frac{D}{\rho_0}\frac{\partial P}{\partial \sigma} + DF_\sigma\,, \tag{16}$$

where DF_x and DF_σ are the viscosity terms. The conservation equation (5), to be applied for (ρ) or (T, S), become

$$\frac{\partial (\cdot)D}{\partial t} + \frac{\partial U(\cdot)D}{\partial x} + \frac{\partial \omega(\cdot)}{\partial \sigma} = \frac{\partial}{\partial x}\left(DA_H\frac{\partial (\cdot)}{\partial x}\right) + \frac{\partial}{\partial \sigma}\left(\frac{K_H}{D}\frac{\partial (\cdot)}{\partial \sigma}\right)\,, \tag{17}$$

under the σ–coordinate transformation. The boundary conditions (7) and (8) simplify to

$$\omega(-1) = \omega(0) = 0\,. \tag{18}$$

1.3 The non-hydrostatic pressure term

Non-hydrostatic pressure effects can be included in the models in several different ways. We will follow Marshall et al.[10], and assume that the pressure can be decomposed into

$$P = P_{ATM} + P_\eta + P_{INT} + Q\,, \tag{19}$$

where P_{ATM} is the atmospheric pressure, $P_\eta = g\rho_0\eta$ is the pressure due to the surface elevation, and

$$P_{INT} = g\int_z^0 \rho(x, s, t)\, ds\,,$$

is the internal pressure. These three terms are included in hydrostatic models. The fourth term Q in (19) is the non-hydrostatic pressure.

BOM use a mode splitting strategy, where fast moving barotropic waves and slower baroclinic waves are calculated in separate steps. This is accomplished by calculating a depth averaged 2D part forward in time with a small time step, and thereafter do a single, larger time step for the 3D part. From the 2D calculation, we get provisional velocities $\tilde{U}$ and $\tilde{\omega}$. When mode splitting is used, an additional equation for Q is needed to close the set of equations. A full system of equations was derived by Kanarska and Maderich[9], and involves a large number of terms. Since the errors in the hydrostatic part of P usually dominate over the error in the non-hydrostatic part, we choose to use a simplified but less accurate set of equations, developed by Berntsen and Furnes[4], where Q is determined by

$$\frac{\partial}{\partial x}\left(D\frac{\partial Q}{\partial x}\right) + \frac{1}{D}\frac{\partial^2 Q}{\partial \sigma^2} = \frac{\rho_0}{\Delta t}\left(\frac{\partial \tilde{U}D}{\partial x} + \frac{\partial \tilde{\omega}}{\partial \sigma} + \frac{\partial \eta}{\partial t}\right)\,, \tag{20}$$

and UD and ω satisfy

$$UD = \tilde{U}D - \frac{\Delta t}{\rho_0}D\frac{\partial Q}{\partial x}\,, \qquad \omega = \tilde{\omega} - \frac{\Delta t}{\rho_0}\frac{1}{D}\frac{\partial Q}{\partial \sigma}\,.$$

Neumann boundary conditions are assumed at the surface, bottom, and lateral boundaries.

1.4 Parallelization of BOM

BOM is parallelized by a domain decomposition method. The global computational domain is split into sub-domains, depending on the number of processors available, and each processor works on its assigned sub-domain. The domain decomposition is only applied in the horizontal dimension, so all sub-domains consist of a block that extends from the free surface to the bottom. A sketch of the horizontal partitioning is shown in Fig. 1. Communication between processors working on different sub-domains is accomplished through the MPI protocol. Further information about the parallelization of BOM can be found in Avlesen[3].

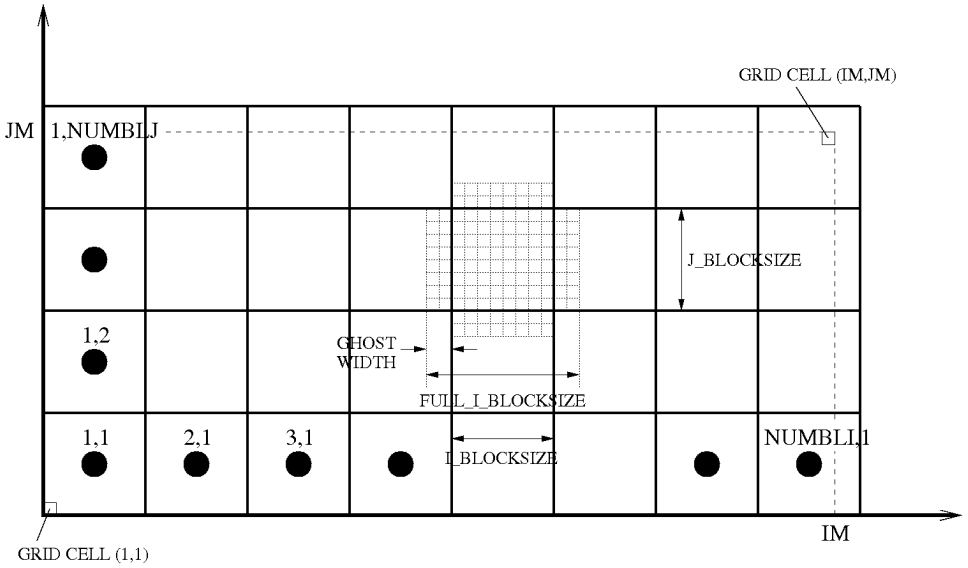


Figure 1: Block partitioning used in domain decomposition.

Hydrostatic ocean models can be parallelized efficiently by domain decomposition. Because most of the physical processes depend only on local conditions, at least within short time intervals, only data from the edges of the sub-domains need to be communicated between the compute nodes at each time step. This is achieved by allowing some overlap between the sub-domains, so that each sub-domain is surrounded by a halo of *ghost cells*. The structure of the parallelized code can be similar to the serial code, as long as the values stored in the ghost cells are updated before being used in calculations.

As mentioned earlier, the calculation of the non-hydrostatic pressure require the solution of an elliptic PDE, which in principle is a global problem that requires the inversion of a sparse matrix. In the iterative approach, data is only directly transferred between neighboring grid points, so several iterations may be needed to distribute data from a local event to the global domain. If the code is parallelized, each iteration requires data to be transferred between different compute nodes, adding to the total cost of the computation.

A diagram showing the time stepping procedure for BOM is shown in Fig. 2. The non-hydrostatic calculation has be implemented without making large changes in the original hydrostatic method.

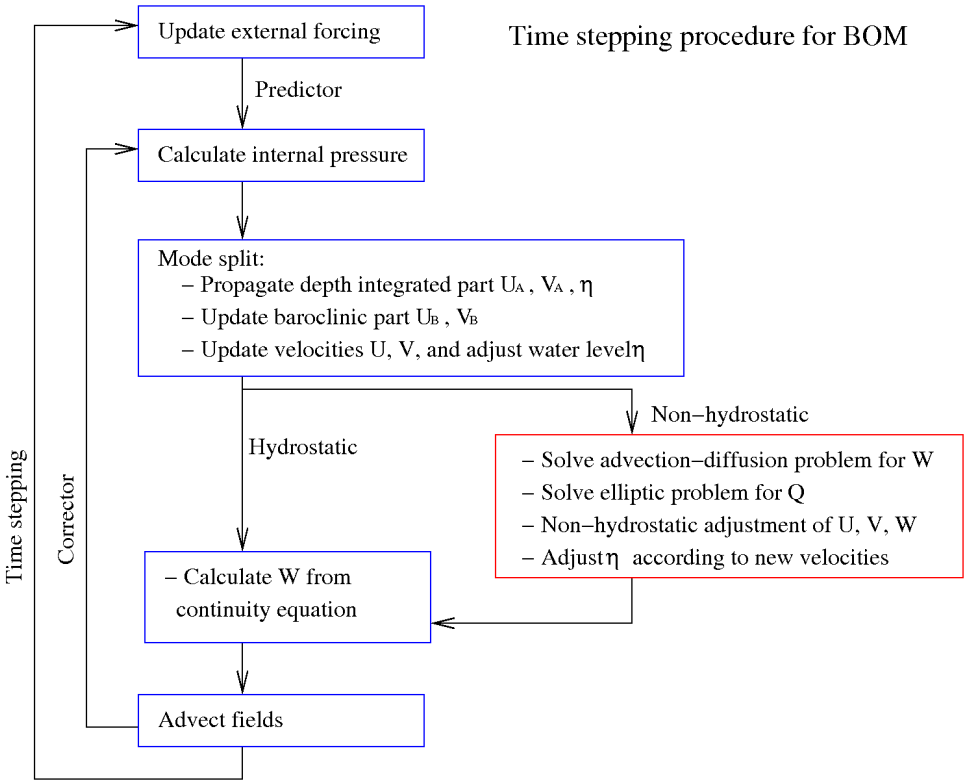


Figure 2: Diagram for time stepping in BOM with calculation of non-hydrostatic pressure.

2 Solvers for sparse matrix systems

Our task is to solve a linear system of equations

$$Ax = b\,, \qquad \text{or} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}x_j = \sum_{i=1}^{n} b_i\,, \tag{21}$$

where A is a sparse square matrix. Such a matrix can be decomposed as

$$A = L + D + U\,,$$

where D, L and U represent the diagonal, strictly lower-triangular, and strictly upper-triangular parts of A, respectively. We will briefly present iterative methods that are relevant for our study.

2.1 Stationary Iterative methods

An iterative method on the form

$$x^{(k)} = Bx^{(k-1)} + c\,, \tag{22}$$

is called a *stationary iterative method* if B and c are independent of the iteration count k. This class of methods include the *Jacobi method*, the *Gauss-Seidel (GS) method*, and the *Successive Over-relaxation (SOR) method*, among others. The Jacobi and Gauss-Seidel methods have similar structures

$$x^{(k)} = D^{-1}(L+U)x^{(k-1)} + D^{-1}b\,, \qquad \text{(Jacobi)} \tag{23}$$

$$x^{(k)} = (D-L)^{-1}(Ux^{(k-1)} + b)\,. \qquad \text{(GS)} \tag{24}$$

The Jacobi method is also known as the *method of simultaneous displacements*, because values are updated only based on values from the previous iteration, i.e. $x_i^{(k)}$ is independent of $x_j^{(k)}$ for $i \neq j$. Likewise, the Gauss-Seidel method is called the *method of successive displacements*, because the value of $x_i^{(k)}$ depends on all $x_j^{(k)}$ for $j < i$. The SOR method is based on Gauss-Seidel, but extrapolates the updated value based on the two last iterations, i.e. we first calculate an estimate $\bar{x}^{(k)}$ using Gauss-Seidel, and do the updating

$$x^{(k)} = \omega\bar{x}^{(k)} + (1-\omega)x^{(k-1)}\,,$$

where ω is the extrapolation factor. In matrix notation, the method can be written as

$$x^{(k)} = (D-\omega L)^{-1}(\omega U + (1-\omega)D)x^{(k-1)} + \omega(D-\omega L)^{-1}b\,. \qquad \text{(SOR)} \tag{25}$$

The value of ω is restricted to the interval $(0, 2)$, but proper *over-relaxation* is only achieved for $1 < \omega < 2$.

2.2 Parallelization of Stationary Iterative methods

Algorithms for parallelized solvers should aim at minimizing the idle time for each processor. This is achieved when each processor works on locally stored data, and do not rely on results from calculations done on a different processor. Algorithms that are efficient on single processor machines may easily perform very badly on a computer cluster if one processor needs to finish its work before the next one can start.

The Jacobi method can be parallelized fairly easily, because the update requires only data from the previous iteration. Although the Jacobi method is rarely used as the primary solver, it is often used as a *preconditioner* for other methods (more about preconditioners

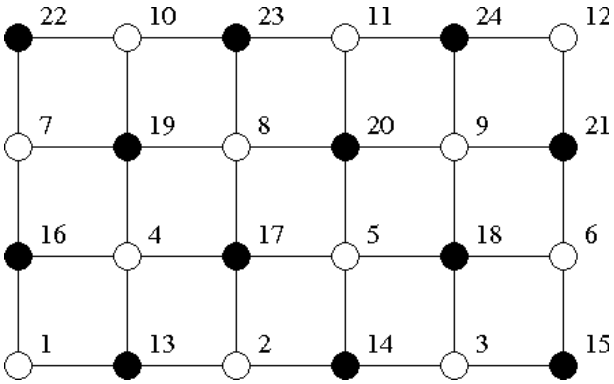


Figure 3: Red-black ordering of grid points

in section 2.4). The *block Jacobi method* applies the Jacobi method on a block diagonal matrix, which is a natural approach for models parallelized by domain decomposition.

The Gauss-Seidel and SOR methods can not be parallelized efficiently in such a straight forward way. Because the updating of $x_i^{(k)}$ depends on $x_j^{(k)}$ for $j < i$, the processors would have to work sequentially to solve the problem. This problem can be overcome by using a multicolor technique in the ordering of grid points. A two color ordering is sufficient for Gauss-Seidel and SOR, as shown in Fig. 3. Red points are not coupled with other red points, and likewise for black points, thereby allowing red and black grid points to be updated in alternating steps.

2.3 Multigrid methods

The stationary iterative methods mentioned in the previous section tend to dampen out the high frequency component of the error quickly, but may need considerably more iterations to smooth out the low frequency error. Multigrid methods exploits the fact that a low frequency error on a fine grid becomes a high frequency error on a coarse grid, as illustrated in Fig 4.

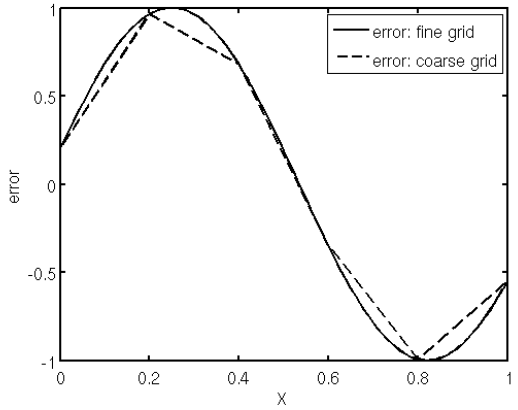


Figure 4: A low frequency error on a fine grid becomes a high frequency error on a coarse grid.

Assuming we want to solve the system on a fine grid, the multigrid method requires the construction of a hierarchy of successively coarser grids. At each level in the hierarchy we can obtain an approximate solution using a standard method (Jacobi, GS, SOR). If there

is a straight forward relation between grids of different sizes, which is usually the case for structured grids, it is also fairly simple to define inter-grid operations. Going from a coarse grid to a fine grid, *prolongation*, is done by interpolation, and going from a fine grid to a coarse grid, *restriction*, can be done by direct injection or averaging over some points in the fine grid. A simple multigrid cycle (V-cycle) is illustrated in Fig. 5.

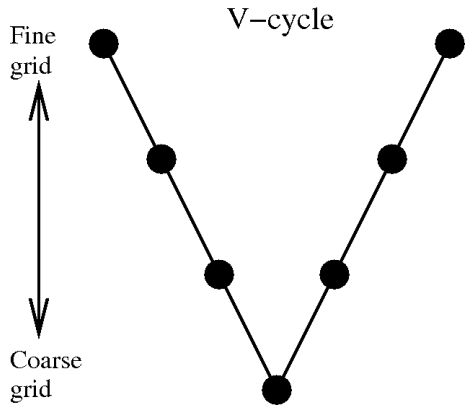


Figure 5: Multigrid V-cycle

In some cases, as for instance with unstructured grids, it is not so straight forward to define a hierarchy of grids and corresponding inter-grid operations. In such cases an algebraic multigrid (AMG) method may be more convenient. The AMG method does not require the explicit construction of coarser grids, but instead defines the "inter-grid operations" based only on the information contained in the structure of the coefficient matrix A.

2.4 Krylov methods

Krylov methods constitute an important branch of iterative methods for sparse matrix systems. Through the iterative process, a sequence $\{x_0, x_1, \ldots, x_n\}$ approximating $x_* = A^{-1}b$ is generated. Krylov methods search for solutions within the Krylov subspace

$$K_n = \text{span}\{b, Ab, A^2b, \ldots, A^{n-1}b\}\,.$$

Conjugate gradient (CG) iteration is the "classical" Krylov method. This method finds $x_n \in K_n$ that minimizes the norm of the error vector $||e_n||_A$, where

$$e_n = x_* - x_n\,,$$

and the matrix norm $||\cdot||_A$ is defined by

$$||\xi||_A = \sqrt{\xi^T A \xi}\,.$$

The CG method proceeds by a three-term recurrence relation, leading to a tridiagonal problem. Furthermore, the residuals

$$r_n = b - Ax_n\,,$$

generated by CG iterations are orthogonal. Unfortunately, this method mainly works for cases where the coefficient matrix A is symmetric positive definite. Krylov methods designed for more general coefficient matrices A either must give up the orthogonalization or the tridiagonal structure. The former case leads to biconjugate gradient (BiCG) methods, whereas the latter approach leads to the generalized minimal residuals (GMRES) method.

Lack of robustness is a drawback for iterative solvers. Both efficiency and robustness can be improved by using *preconditioners*. A preconditioner is a non-singular matrix M

with the properties that $Mx = b$ is inexpensive to solve, and that M in some sense is close to A. Once a preconditioner is available, we can transform the linear system (21) to

$$M^{-1}Ax = M^{-1}b\,. \tag{26}$$

A choice for M may be as simple as $M = diag(A)$. Other choices of M may be constructed by incomplete LU factorization, or by doing a single step with a classical iterative method.

2.5 The HYPRE library of high performance preconditioners

Red-black SOR is the only method that has been implemented directly in BOM. Other iterative methods are available through the HYPRE library[1]. The library is written in C, but includes an interface to FORTRAN. Table 1 includes a list of the solvers available for BOM through the HYPRE library.

solver name	solver type	preconditioned
SMG	geometric multigrid	NO
PFMG	geometric multigrid	NO
BoomerAMG	algebraic multigrid	NO
GMRES	Krylov method	YES
BiCGSTAB	Krylov method	YES

Table 1: List of HYPRE solvers available for BOM.

The first step when using a HYPRE solver is to provide the solver with information on the geometry of the problem and the structure of the discretization stencil. Then we define the coefficient matrix A and the right hand side vector b. These steps are more or less the same for all solvers. The last step before calling the solver is to define solver specific parameters. A list of the parameters used in our study is found in Appendix A.

3 Results from test case studies

The model has been tested on three 2D test cases and one 3D test case, which cover a wide range of problem sizes. We have used the two cluster computers `fimm`[1] and `hexagon`[2] at Bergen Center for Computational Science, UNIFOB AS, to run the simulations. Most of the results come from simulations on `hexagon`, using a maximum of 2 cores per computer node. A few longer simulations have been done on `fimm`.

PARAMETER	SYMBOL	VALUES
Number of processors	P	[1, 2, 4, 8, 16, 32, 64]
Residual tolerance	C	$[1.0 \times 10^{-2}, 1.0 \times 10^{-4}, 1.0 \times 10^{-7}]$

Table 2: Solver parameters

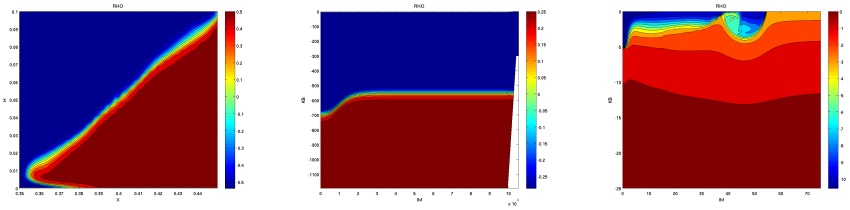
In these tests we focus on two parameters; the number of processors P, and the residual tolerance C for the iterative solvers. The values used in the tests are given in Table 2. The maximum number of iterations for SOR was 5.0×10^5 for cases (a), (b), and (c), and 5.0×10^3 for case (d). The other solvers used default values for maximum number of iterations, but these values were only attained in cases where the solvers did not converge.

A number of additional parameters are available for the different HYPRE solvers (see Appendix A), but we have not done extensive tuning of these parameters. The tuning that has been done is restricted to the number of sweeps done at each grid level for the multigrid methods. It is our experience that SMG generally requires less parameter tuning than PFMG and AMG.

3.1 Description of test cases

	CASE	DIMENSIONS
(a)	A lock-release case.	402 x 101
(b)	An internal solitary wave.	4232 x 101
(c)	A river plume.	4802 x 201
(d)	A 3D lock-release case.	1600 x 16 x 201

Table 3: Test cases for mpibom.



(a) Lock-release (b) Solitary wave (c) River plume

Figure 6: Test cases for mpibom.

The test cases used in this study are listed in Table 3. Test case (a) is a lock-release case similar to Härtel et al.[7]. The model domain is a wave tank of length L = 0.8 m and depth H = 0.1 m. Initially the tank is divided in two, with heavy fluid of ρ = 1025.0 kg m^{-3} in the right half of the tank, and light fluid of ρ = 1023.955 kg m^{-3} in the left half of the tank.

[1] `fimm` home page: `http://www.parallab.uib.no/resources/cluster`
[2] `hexagon` home page: `http://www.parallab.uib.no/resources/hexagon`

SOLVERS	CASE (a)	CASE (b)	CASE (c)	CASE (d)
SOR	+	$+/-^{1}$	+	$-^{2}$
SMG	+	+	+	+
PFMG	+	+	+	+
AMG	$-^{3}$	$+/-^{4}$	+	+
GMRES + SMG	+	+	+	+
GMRES + PFMG	+	+	+	+
GMRES + AMG	$-^{3}$	+	+	+
BiCGSTAB + SMG	+	+	+	$+/-^{5}$
BiCGSTAB + PFMG	$-^{6}$	+	+	+
BiCGSTAB + AMG	$-^{3}$	+	+	+

+: OK; $-$: Failed; $+/-$: Failed for some parameters.
[1]SOR failed residual test for $C = 1.0 \times 10^{-7}$ in first time step.
[2]SOR failed residual test for all time steps.
[3]Solver failed to converge.
[4]Solver failed to converge for $P = [4, 8]$, $C = 1.0 \times 10^{-7}$.
[5]Solver failed to converge for $C = 1.0 \times 10^{-7}$.
[6]Solver failed to converge in first time step for $C = 1.0 \times 10^{-2}$, and for all time steps for $C = 1.0 \times 10^{-4}$ and $C = 1.0 \times 10^{-7}$.

Table 4: Convergence of solvers.

The interface between the two water masses is smoothed to avoid instabilities at the start of the simulation. Fig. 6(a) shows the density profile shortly after the initial state. This setup is also used for 3D test case (d), which has higher resolution, and includes the transverse dimension. The memory requirements for test case (d) exceeded the maximum capacity for a single computer node, so this case was only tested for processors in the range 2 to 64.

Test case (b) is an internal solitary wave breaking against a bottom slope. The model domain has length L = 105 km, and maximum depth H = 1200 m. The shelf slope is linear of degree 11.3°, and extends up to H = 300 m. The initial shape of the solitary wave is close to the KdV soliton solution, as seen in Fig. 6(b).

Test case (c) is a river plume propagating and gradually entraining into a stratified basin of denser fluid. The model domain has length L = 1200 m, and constant depth of H = 50 m. The river plume is introduced at the left boundary through a flow relaxation (FRS) zone.

3.2 Accuracy and solution time vs. cut off

Accuracy and stability are key properties of any solvers. Table 4 shows how the different solvers performed on the different test cases. In cases where SOR failed to obtain the required residual tolerance C, this could in principle be corrected by increasing the maximum number of iterations. However, this would require extremely long run-times for case (d) and problems with similar size, and this is not always possible in practice. Whenever some of the other solvers failed to converge, the non-hydrostatic pressure was set to zero, allowing the simulation to keep running. This allowed some of the simulations to recover even if the non hydrostatic pressure was not calculated at the startup of the simulation. This procedure did not work for the AMG solver, where the solver was not able to recover after a convergence failure, causing errors in all subsequent calls to the solver. Apart from SOR, we note that SMG, PFMG, and GMRES converges for all the test cases.It is possible that these results would be different if more time had been used to tune the solver parameters, but it is our opinion that a robust method should work with a minimum of such tuning.

Even if the residual is small, this does not guarantee that the error is small everywhere in the domain. Since the non-hydrostatic pressure results in correction terms for the horizontal and vertical velocities, we check our results by considering the maximum of these correc-

tion velocities. First of all we check that the different solvers give the same results. This has been done by running test cases (a), (b), and (c) for 2000 time steps with parameters $[P = 4, C = 1.0 \times 10^{-4}$. The results for kinetic energy (EKIN), and maximum horizontal (MAX UCORR) and vertical (MAX WCORR) velocity corrections are compared to the SOR result using the formula

$$\Delta\xi_{SOLVER} = \frac{\xi_{SOLVER} - \xi_{SOR}}{\xi_{SOR}}$$

where ξ represents EKIN, MAX UCORR, and MAX WCORR. The results are presented in Table 5. Since all the results were of the same order of magnitude, we only present this number in the table. The table shows that all solvers that converge give reasonable results in the long run.

CASE	EKIN	MAX UCORR	MAX WCORR
(a)	1.0×10^{-8}	1.0×10^{-6}	1.0×10^{-7}
(b)	1.0×10^{-11}	1.0×10^{-5}	1.0×10^{-6}
(c)	1.0×10^{-6}	1.0×10^{-6}	1.0×10^{-6}

Table 5: Relative errors for all solvers after 2000 time steps, with $P = 4$ and $C = 1.0 \times 10^{-1}$.

A similar test for all solver parameters has been performed, with runs over 200 time steps. Fig. 7 shows accuracy for the models. Results for $C = 1.0 \times 10^{-7}$ are taken as the "true solution", and used as reference for the $C = 1.0 \times 10^{-2}$ and $C = 1.0 \times 10^{-4}$ results. In cases where the solver did not converge for $C = 1.0 \times 10^{-7}$, the SMG result is used as reference.

The non-hydrostatic error will usually be small compared to errors from other sources, so we do not need to calculate UCORR and WCORR with very high precision. Using $C = 1.0 \times 10^{-4}$ seems to be a safe choice for all test cases, but we may be wasting computational resources to obtain an accuracy that is not needed. The $C = 1.0 \times 10^{-2}$ is reasonable for most cases, although the GMRES method does not give satisfactory results for the lock-release case.

3.3 Solution time vs. number of processors

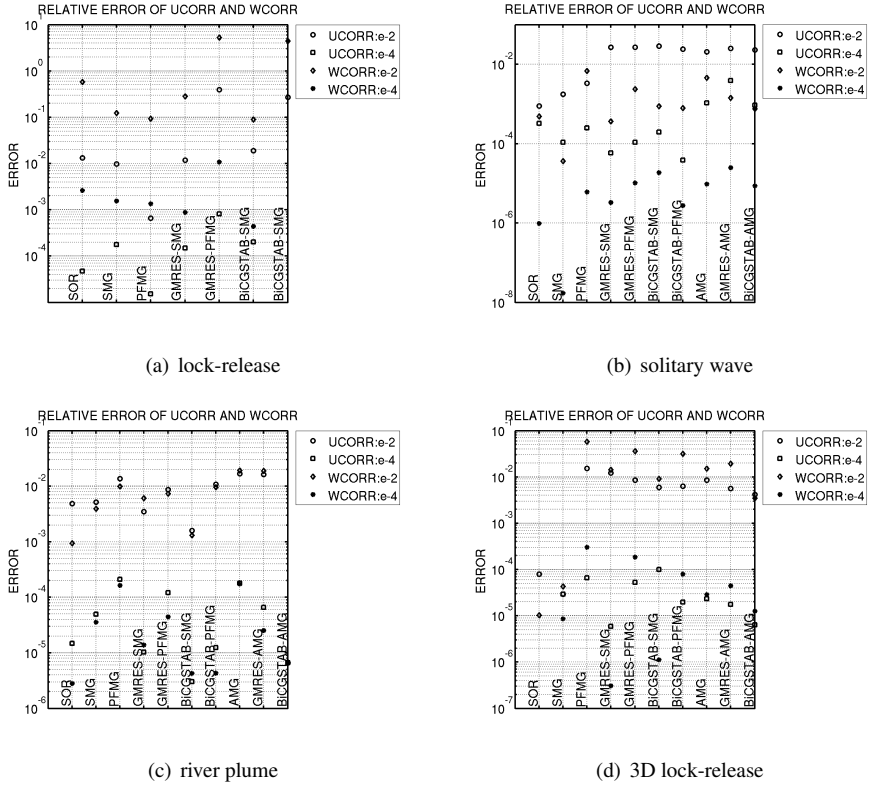
Test case runs with 200 time steps have been used to look at the performance of the solvers for different numbers of CPUs. The results from these runs are presented in Figs. 8 and 9. The top row shows the total execution time, and the second row shows the shortest solver time for a single time step, averaged over the 20 best recorded values. The two last rows shows speedup and parallel efficiency, defined by

$$\text{speedup:} \quad S(P) = \frac{\text{execution time for} P \text{ processors}}{\text{execution time for} \min(P) \text{ processors}},$$

$$\text{parallel efficiency:} \quad PE(P) = \frac{S(P) \times \min(P)}{P},$$

respectively. Results are shown for $C = 1.0 \times 10^{-4}$. The total execution time of a hydrostatic simulation is also included for comparison.

The SOR is the fastest solver for the smallest case (a), and generally scales well with increasing number of CPUs. Super-linear speedup is attained for test cases (b) and (c) (see Figs. 8(h) and 9(g)), which means that the speedup is larger than N for N processors. This is probably a cache effect due to the memory hierarchy in the computer. As the cache size (local CPU memory) increases with increasing number of CPUs, less data is needed to be transmitted between the main memory and the CPU, thereby decreasing the execution time



(a) lock-release

(b) solitary wave

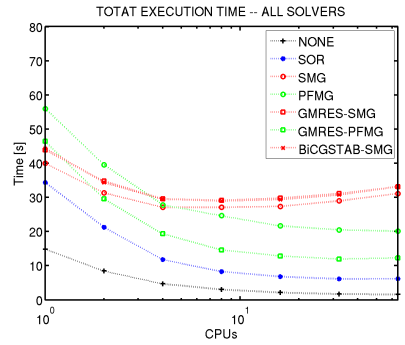
(c) river plume

(d) 3D lock-release

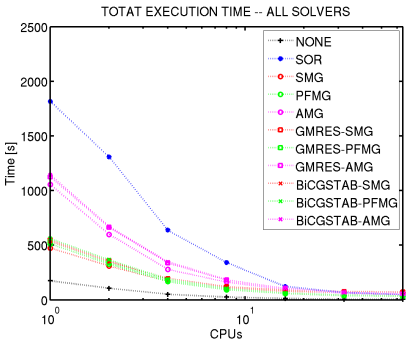
Figure 7: Accuracy of the solvers. U and W correction terms after 200 time steps, for $C = 1.0 \times 10^{-2}$ and $C = 1.0 \times 10^{-2}$.

of the calculation. In this case the simple data structure of SOR has an advantage over the more elaborate solvers. However, SOR does not scale well with the problem size, and is clearly the slowest solver for larger problems.

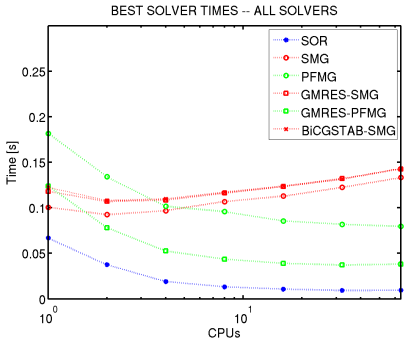
The fastest solver for larger problems is PFMG, or a Krylov method with PFMG preconditioner. The SMG solver performs well for intermediate size problems, but is clearly slower for case (d). The AMG method is consistently slower than PFMG and SMG. In most cases the execution times for the Krylov methods are close to the respective preconditioners, indicating that the preconditioning dominates in the calculation. However, these methods seem to perform better as the problem size increases.



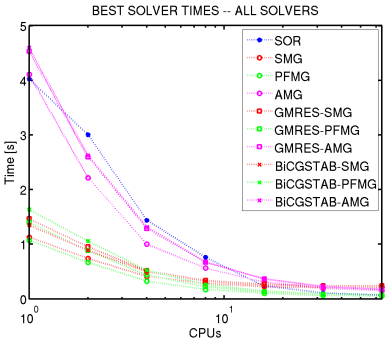
(a) lock-release



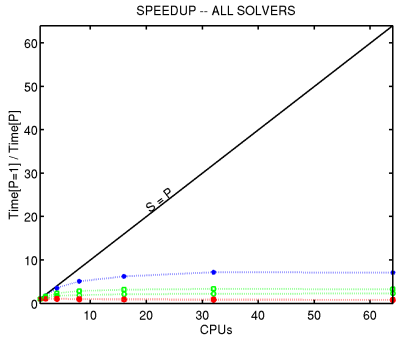
(b) solitary wave



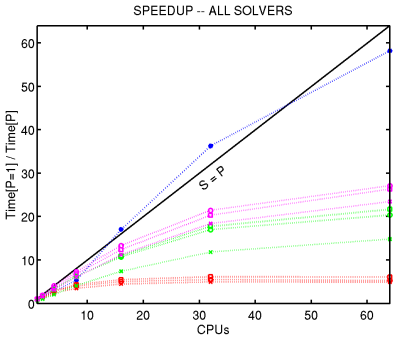
(c) lock-release



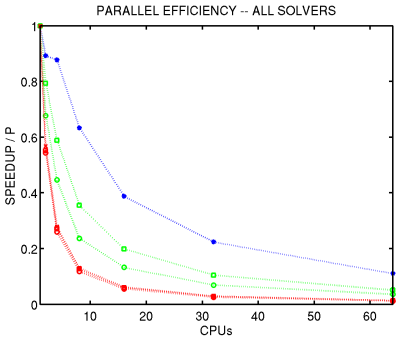
(d) solitary wave



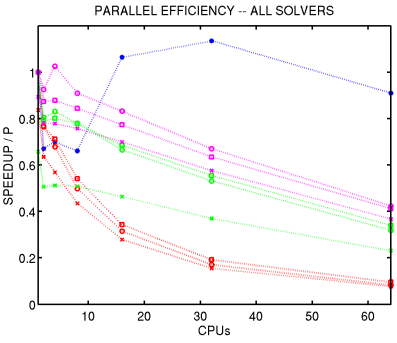
(e) lock-release



(f) solitary wave

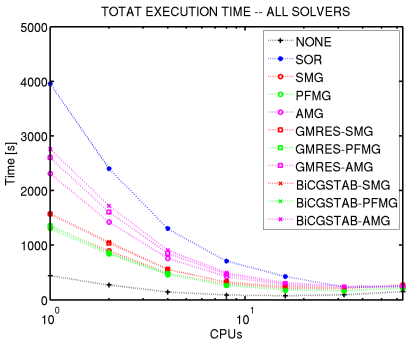


(g) lock-release

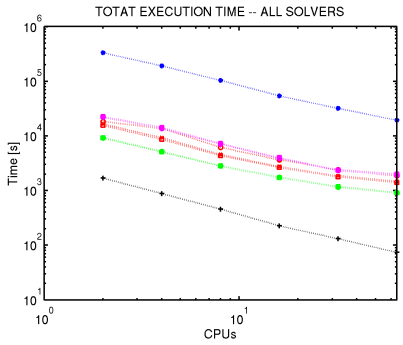


(h) solitary wave

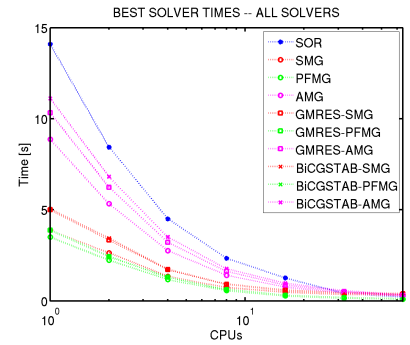
Figure 8: Total execution time, best solver time, speedup, and parallel efficiency for cases (a) and (b). $C = 1.0 \times 10^{-4}$.



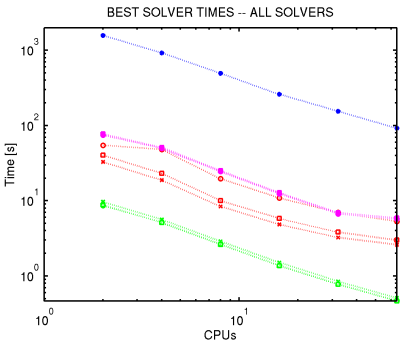
(a) river plume



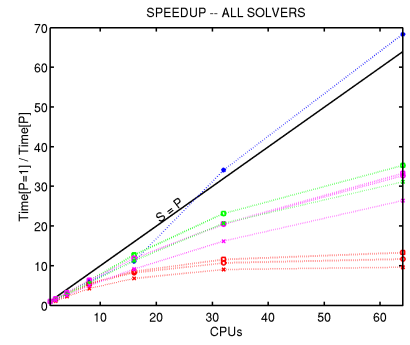
(b) 3D lock-release



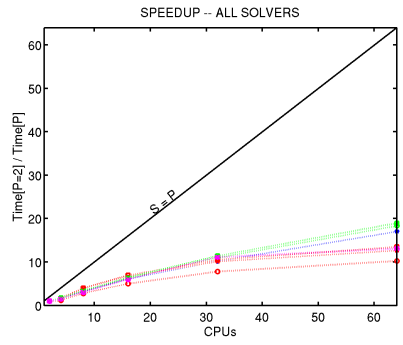
(c) river plume



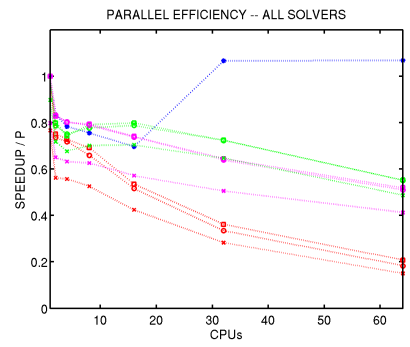
(d) 3D lock-release



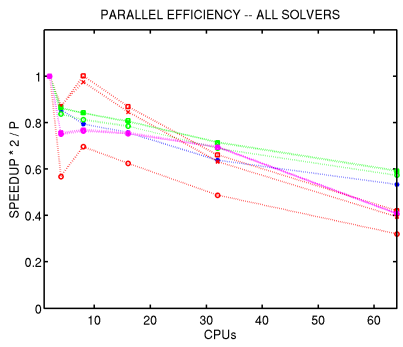
(e) river plume



(f) 3D lock-release



(g) river plume



(h) 3D lock-release

Figure 9: Total execution time, best solver time, speedup, and parallel efficiency for cases (c) and (d). $C = 1.0 \times 10^{-4}$.

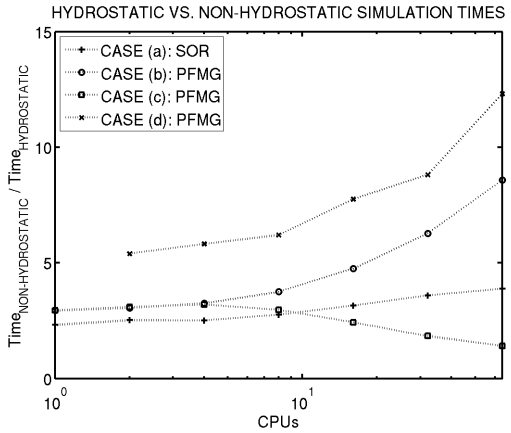


Figure 10: Comparison between execution times for hydrostatic and non-hydrostatic simulations. $C = 1.0 \times 10^{-4}$.

Fig. 10 shows a comparison between execution times for the hydrostatic model vs. the fastest non-hydrostatic model, calculated by

$$\Delta T(P) = \frac{\text{Time}_{NON-HYDROSTATIC}}{\text{Time}_{HYDROSTATIC}} .$$

The non-hydrostatic usually require from 2 to 10 times longer run times than the hydrostatic model. In most cases the difference increases with increasing number of CPUs, which is probably due to more efficient memory use in the hydrostatic than in the non-hydrostatic simulation. This is not the trend in test case (c), where the difference between the hydrostatic and non-hydrostatic decreases with increasing number of CPUs. This is the only case with FRS zones at the boundaries, and it is conceivable that this process can be a bottleneck of the same magnitude as the non-hydrostatic pressure calculation in the program.

4 Concluding remarks

A number of different linear solvers have been tested on four test cases with different physical properties and model sizes. The tests have shown that we may benefit from using scalable linear solvers from an external library such as HYPRE, but no single solver performed best for all cases. The SOR method has an advantage for small model problems, and is least likely to suffer from memory bottlenecks. The SMG method performs well over-all, and requires the least parameter tuning to work. The PFMG method is fast for both intermediate and large problems, and is generally reliable if enough sweeps are done at each grid level. The GMRES with PFMG preconditioner is slightly faster than PFMG for the largest problem, but there may be issues accuracy. The AMG and BiCGSTAB methods do not seem to perform as well as the other methods in our tests.

The choice of linear solver for the non-hydrostatic pressure is therefore problem dependent. For the application at hand, the calculation of the non-hydrostatic pressure component, the total pressure error is likely to be dominated by the hydrostatic part of the calculation. It can therefore be argued that we do not need to calculate this term with a very high degree of accuracy. Even so, it is useful to have some control on the error of the calculation to be sure that the error is not of the same magnitude as the value we calculate. For this purpose we find it is advisable to use a maximum residual tolerance of approximately $C = 1.0 \times 10^{-2}$.

Acknowledgments

This work has been supported by NFR grant No. 800274, "Non-hydrostatic Ocean General Circulation Models". I am greatly indebted to Dr. Helge Avlesen, who wrote most of the MPI code for BOM, and initiated the work that has lead to this report. I am also grateful for the contribution of Prof. Jarle Berntsen, in particular with respect to the 3D version of the code.

References

[1] Hypre: Library of high performance preconditioners. https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html. 2.5

[2] Hypre reference manual, version 2.0.0. https://computation.llnl.gov/casc/hypre/download/hypre-2.0.0_ref_manual.pdf, 2006. UCRL-CODE-222953. 4

[3] H. Avlesen. Advanced user support: On the parallelization of a non hydrostatic, sigma co-ordinate ocean model, 2004. 1.4

[4] J. Berntsen and G. Furnes. Internal pressure errors in sigma-coordinate ocean models-sensitivity of the growth of the flow to the time stepping method and possible non-hydrostatic effects. *Continental Shelf Research*, 25:829–848, 2005. 1, 1.3

[5] A.F. Blumberg and G.L. Mellor. A description of a three-dimensional coastal ocean circulation model. In N.S. Heaps, editor, *Three-Dimensional Coastal Ocean Models*, volume 4 of *Coastal and Estuarine Series*, page 208. American Geophysical Union, 1987. 1.2

[6] A.E. Gill. *Atmosphere-Ocean Dynamics*. Academic Press, 1982. ISBN-0-12-283520-4. 1.1

[7] C. Härtel, E. Meiburg, and F. Necker. Analysis and direct numerical simulation of the flow at a gravity-current head. part 1. flow topology and front speed for slip and no-slip boundaries. *Journal of Fluid Mechanics*, 418:213–229, 2000. 3.1

[8] Y. Heggelund, F. Vikebø, J. Berntsen, and G. Furnes. Hydrostatic and non-hydrostatic studies of gravitational adjustment over a slope. *Continental Shelf Research*, 24:2133–2148, 2004. 1

[9] Y. Kanarska and V. Maderich. A non-hydrostatic numerical model for calculating free-surface stratified flows. *Ocean Dynamics*, 53:176–185, DOI 10.1007/s10236–003–0039–6, 2003. 1.3

[10] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey. A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research*, 102(C3):5753–5766, 1997. 1.3

[11] D.-P. Wang. Mutual intrusion of a gravity current and density front formation. *J. Phys. Oceanogr.*, 14:1191–1199, 1984. 1.1

Appendix A: Parameters for the linear solvers

The following tables shows the parameters we have used for the solvers, and the values used for each of the the test cases. The list is not exhaustive, and the values chosen for the parameters may not be optimal. See the reference manual for HYPRE[2] for further description about the parameters.

PARAMETER	CASE (a)	CASE (b)	CASE (c)	CASE (d)
SOR				
Maximum iterations	5.0×10^5	5.0×10^5	5.0×10^5	5.0×10^3
SMG				
SetNumPreRelax	1	1	1	2
SetNumPostRelax	1	1	1	2
PFMG				
SetNumPreRelax	3	3	2	3
SetNumPostRelax	3	3	2	3
SetRelaxType	1	1	–	1
AMG				
SetCoarsenType	6	6	6	6
SetStrongThrshld	0.25	0.25	0.25	0.25
SetNumSweeps	2	2	2	2
SetInterpType	0	0	0	0

Table 6: Parameters for linear solvers.

PARAMETER	CASE (a)	CASE (b)	CASE (c)	CASE (d)
GMRES + SMG				
SMGSetTol	0.0	0.0	0.0	0.0
SMGSetMaxIter	1	1	1	1
SMGSetZeroGuess	+	−	+	−
SMGSetNumPreRelax	1	1	1	1
SMGSetNumPostRelax	1	1	1	1
GMRES + PFMG				
PFMGSetTol	0.0	0.0	0.0	0.0
PFMGSetMaxIter	1	1	1	1
PFMGSetZeroGuess	+	−	+	−
PFMGSetNumPreRelax	3	3	3	3
PFMGSetNumPostRelax	3	3	3	3
GMRES + AMG				
AMGSetTol	0.0	0.0	0.0	0.0
AMGSetMaxIter	1	1	1	1
AMGSetCoarsenType	6	6	6	6
AMGSetStrongThrshld	0.25	0.25	0.25	0.25
AMGSetNumSweeps	2	2	2	2
BiCGSTAB + SMG				
SMGSetTol	0.0	0.0	0.0	0.0
SMGSetMaxIter	1	1	1	1
SMGSetZeroGuess	+	−	+	−
SMGSetNumPreRelax	1	1	1	1
SMGSetNumPostRelax	1	1	1	1
BiCGSTAB + PFMG				
PFMGSetTol	0.0	0.0	0.0	0.0
PFMGSetMaxIter	1	2	1	2
PFMGSetZeroGuess	+	+	+	+
PFMGSetNumPreRelax	3	3	3	3
PFMGSetNumPostRelax	3	3	3	3
BiCGSTAB + AMG				
AMGSetTol	0.0	0.0	0.0	0.0
AMGSetMaxIter	1	1	1	1
AMGSetCoarsenType	6	6	6	6
AMGSetStrongThrshld	0.25	0.25	0.25	0.25
AMGSetNumSweeps	2	2	2	2

Table 7: Parameters for linear solvers.