Journal of Computational Physics 229 (2010) 8167–8179

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

The application of Jacobian-free Newton–Krylov methods to reduce the spin-up time of ocean general circulation models

Erik Bernsen [a,*], Henk A. Dijkstra [a], Jonas Thies [b], Fred W. Wubs [b]

[a] Institute for Marine and Atmospheric research Utrecht, Utrecht University, The Netherlands
[b] Johann Bernoulli Institute for Mathematics and Computer Science, Groningen University, The Netherlands

ARTICLE INFO

Article history:
Received 20 April 2010
Received in revised form 9 July 2010
Accepted 9 July 2010
Available online 16 July 2010

Keywords:
Ocean modelling
Jacobian-free Newton–Krylov methods
Preconditioning

ABSTRACT

In present-day forward time stepping ocean-climate models, capturing both the wind-driven and thermohaline components, a substantial amount of CPU time is needed in a so-called spin-up simulation to determine an equilibrium solution. In this paper, we present methodology based on Jacobian-Free Newton–Krylov methods to reduce the computational time for such a spin-up problem. We apply the method to an idealized configuration of a state-of-the-art ocean model, the Modular Ocean Model version 4 (MOM4). It is shown that a typical speed-up of a factor 10–25 with respect to the original MOM4 code can be achieved and that this speed-up increases with increasing horizontal resolution.

1. Introduction

Ocean modelling is an art. Certainly, we know the basic equations of fluid motion, but ironically we cannot use these equations directly because they apply to scales which cannot be resolved with the largest supercomputers for years to come. Consequently, we have to cope with the formidable problem of subgrid-scale representation to obtain the ocean model equations used to predict ocean flows.

In the late sixties, the first ocean model was developed at the Geophysical Fluid Dynamics Laboratory (GFDL). Descendants of this model still exist and the Modular Ocean Model version 4 (MOM4) is the latest version of this model [1]. Apart from this model, many other types of ocean models have been developed, such as isopycnal models (e.g., MICOM [2]) and hybrid coordinate models (e.g., HYCOM [3]). Recently, there also have been fruitful interactions between the engineering community and ocean modellers resulting in the development of unstructured mesh, finite element [4], and spectral element models [5].

Many of the ocean models mentioned above use explicit time stepping schemes. The advantage of these methods is that coding of all the relevant physical processes is relatively simple. The time step, however, is limited because of numerical amplification of truncation errors (through well-known stability criteria) rather than by the changes in the actual solution [6]. This limitation becomes even more restrictive as the spatial resolution increases.

In practice, initial conditions of an ocean model for a particular study are derived from so-called spin-up simulations. Such simulations, which are often started from a state of rest, provide an equilibrium solution of the ocean model. Given that the equilibration time scale of a three-dimensional ocean flow is about 5000 yr, the spin-up simulations are costly and form a barrier to extensive sensitivity studies of ocean models. Hence, the limitations of explicit schemes are extremely undesirable in the computation of equilibrium solutions.

* Corresponding author.
E-mail address: e.bernsen@uu.nl (E. Bernsen).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.07.015

In many of the new approaches in ocean modelling, implicit time-stepping techniques appear in some parts of the models. However, the potential of implicit techniques in ocean modelling has not been fully explored. Implicit time integration always leads to a problem where the solution of a system of nonlinear algebraic equations has to be determined. When applying some variant of the Newton–Raphson method, this leads to the problem of solving large linear systems of equations with some Jacobian matrix J. The structure of J depends on how the governing equations are discretized but, in general J is very ill-conditioned. Hence classical iterative methods will not work in solving these linear systems and preconditioning techniques are needed. The main barriers in applying implicit methods to realistic ocean models are (i) the difficulties in constructing the Jacobian matrices and (ii) the lack of efficient preconditioning techniques.

Over the last decade, several new preconditioning techniques have been applied to fully-implicit ocean models [7–9] with the aim to determine the bifurcation behavior of flows in these models. All these techniques were based on the explicit construction of the Jacobian matrix which one needs anyway for solving, for example, the linear stability problem of a particular steady flow. However, when using the Jacobian-Free (JFNK) method the Jacobian matrix is no longer explicitly needed for finding the spin-up solution. In addition, this method can directly use an explicit code, which is desirable as these codes are usually maintained by an user group or large center (for example, GFDL).

The JFNK method was already used to solve the spin-up problem of biogeochemical tracers [10–12] for periodic forcing and corresponding periodic equilibrium solutions. However, in these biogeochemical tracer models only passive tracers are considered and dynamical quantities such as, for instance, velocities are assumed to be given. The dynamical spin-up problem for periodic forcing was treated in Merlis and Khatiwala [13], but only for a relatively simple quasi-geostrophic model.

In this paper, we apply the JFNK method to speed-up the dynamical spin-up of a state-of-the-art ocean model. We use the simplifying assumption of steady forcing instead of periodic forcing resulting in steady equilibrium solutions rather than periodic equilibria. Although periodic forcing is the more realistic one, steady forcing remains important, especially if one is interested in longer time-scales and time-averaged solutions.

In Bernsen et al. [14] we started our development of the JFNK method by using a planetary geostrophic model, where the momentum equations are diagnostic, and only the temperature and salinity equations are prognostic. Next, we turned to MOM4 and considered only the wind-driven ocean-circulation by fixing the temperature and salinity field [15]. In the present paper, we synthesize both approaches to determine equilibrium solutions of MOM4 with all unknowns (velocities, free surface height, temperature and salinity) as prognostic variables.

In Section 2, we recapitulate basic features of MOM4 and the JFNK methodology with details in Appendix A. Specific problems in applying the JFNK method to the full MOM4 equations are addressed in Section 3 with particular details in Appendix B. Results for a representative test problem are presented in Section 4 and discussed in Section 5.

2. Application of the JFNK method to MOM4

The MOM4 ocean model is described in detail in Griffies et al. [1]. It is a primitive equation model that solves the hydrostatic momentum equations, the continuity equation and the equations for temperature and salinity on a so-called Arakawa B-grid. In addition, a free-surface formulation is applied where the sea-surface height is part of the solution.

In Bernsen et al. [15] it was shown that the discretized model equations of MOM4 can be cast into the form

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{F}(\vec{x}), \tag{1}$$

with $\vec{x}$ the state vector, $\vec{F}$ the residual and t the time. The state vector contains horizontal velocities, sea surface height and the tracer quantities (temperature and salinity) at grid points, whereas the residual $\vec{F}$ contains the discretized horizontal momentum equations, the vertically integrated continuity equation and the equations for temperature and salinity.

In this paper we are interested in finding equilibrium solutions of MOM4, hence we solve

$$\vec{F}(\vec{x}) = 0. \tag{2}$$

In the JFNK [16] method this system of non-linear equations is solved using a Newton–Raphson iteration. Starting from an initial guess $\vec{x}_0$, the iteration is given by

$$\vec{x}_{k+1} = \vec{x}_k + \Delta\vec{x}_{k+1}, \tag{3}$$

with $\Delta\vec{x}_{k+1}$ satisfying

$$J\Delta\vec{x}_{k+1} = -\vec{F}(\vec{x}_k), \tag{4}$$

where $J_{\vec{x}_k}$ is the Jacobian matrix of $\vec{F}(\vec{x}_k)$ defined by $(J_{\vec{x}_k})_{ij} = \partial F_i/\partial x_j$. In practice an inexact Newton method is applied where $\Delta\vec{x}_{k+1}$ satisfies (4) approximately

$$\|J_{\vec{x}_k}\Delta\vec{x}_{k+1} + \vec{F}(\vec{x}_k)\|_2 < \eta\|\vec{F}(\vec{x}_k)\|_2 \tag{5}$$

with $\eta < 1$ a specified accuracy.

Krylov methods, for instance GMRES [17], are used to solve the linear systems (4). In a Jacobian-Free method, a finite-difference approximation for the matrix–vector product is applied, exploiting the fact that Krylov methods only require the effect of applying $J_{\vec{x}_k}$ to a vector. An one-sided finite difference approximation

$$J_{\vec{x}_k}\vec{v} \approx \frac{\vec{F}(\vec{x}_k + \epsilon\vec{v}) - \vec{F}(\vec{x}_k)}{\epsilon}, \tag{6}$$

is used here, with ϵ a small parameter (more on the choice of ϵ follows in Section 3.2 below). The advantage of this approach is that now only the residual $\vec{F}$ is required and given an explicit timestepping code such as MOM4, this is much easier to obtain than an explicit representation of the full Jacobian $J_{\vec{x}_k}$. However, even the computation of the residual is non-trivial in a model such as MOM4 and in Bernsen et al. [15] it was shown how this is done. For convenience and completeness this is repeated in Appendix A.

3. Specific problems

In practice one encounters several problems when applying the JFNK method to a model such as MOM4. In the next three subsections, we address some of these specific problems.

3.1. Globalization

One of the problems associated with Newton's method is global convergence. Convergence is only guaranteed when starting from a point close enough to the equilibrium solution, which is not known a priori; to avoid this problem we use a continuation method with the forcing strength as a continuation parameter λ. The forcing of the model (wind stress and heat and fresh water fluxes at the surface) depends linearly on λ with $\lambda = 0$ corresponding to no forcing and $\lambda = 1$ corresponding to the desired forcing. We now increase the forcing from $\lambda = 0$ to $\lambda = 1$ in small steps $\Delta\lambda$. For each value of λ we use the JFNK method to solve (2) and we use as many Newton steps as needed to satisfy the stopping criterion

$$\|\vec{F}(\vec{x},\lambda)\|_2/N < \epsilon_N, \tag{7}$$

with N the dimension of the state vector and ϵ_N a fixed value. Note that here we write $\vec{F}(\vec{x},\lambda)$ instead of $\vec{F}(\vec{x})$ to indicate the dependence of the residual on the forcing strength λ. As an initial guess for the JFNK method we use the equilibrium solution for the previous value of λ If we use a continuation step $\Delta\lambda$ that is small enough, then the differences between equilibrium solutions for two successive values of λ are very small and hence Newton's method will converge without problems. Note that for $\lambda = 0$ we do not have to apply the JFNK method since then we know that the corresponding equilibrium solution is given by a state of no motion and uniform temperature and salinity fields.

In addition to this continuation method we use a linesearch method to improve the global convergence of Newton's method, resulting in the possibility to take larger continuation steps $\Delta\lambda$. The idea is to replace the Newton update (3) with an update of the form

$$\vec{x}_{k+1} = \vec{x}_k + \theta_{k+1}\Delta\vec{x}_{k+1}, \tag{8}$$

with the value of $0 < \theta_{k+1} < 1$ determined by the specific linesearch method that is used. Here we use a minimal reduction method [18] which requires that the residual improves at least with a factor $0 < \xi < 1$ between two successive Newton steps

$$\|\vec{F}(\vec{x}_{k+1},\lambda)\|_2 = \|\vec{F}(\vec{x}_k + \theta_{k+1}\Delta\vec{x}_{k+1},\lambda)\|_2 < \xi\|\vec{F}(\vec{x}_k,\lambda)\|_2. \tag{9}$$

Starting with $\theta_{k+1} = 1$, we check if (9) is satisfied and if this is not the case then we update θ_{k+1} according to

$$\theta_{k+1} \leftarrow \gamma\theta_{k+1}, \tag{10}$$

with $0 < \gamma < 1$. The update (10) is repeated until either (9) holds or the update (10) has been applied a specified maximum number of times. If this maximum number of updates is reached then the most recent value of θ_{k+1} is used even though (9) is not satisfied.

3.2. Convective adjustment

In ocean models the hydrostatic momentum equations do not prevent the occurrence of a statically unstable stratification ($\partial\rho/\partial z > 0$, where z is vertically upwards). To avoid these unphysical situations a convective adjustment scheme is required. In MOM4 this is implemented by using a variable vertical mixing coefficient κ_v for temperature and salinity given by

$$\kappa_v(\partial\rho/\partial z) = \kappa_{v,0} + f(\partial\rho/\partial z)(\kappa_{v,C} - \kappa_{v,0}), \tag{11}$$

with $\kappa_{v,C} \gg \kappa_{v,0}$ and f the step function

$$f(\partial\rho/\partial z) = \begin{cases} 0 & \text{if } \partial\rho/\partial z \leqslant 0, \\ 1 & \text{if } \partial\rho/\partial z > 0. \end{cases} \tag{12}$$

In this case we have that $\kappa_v = \kappa_{v,0}$ for a stable stratification whereas the vertical mixing coefficient switches to a value of $\kappa_{v,C}$ on grid points with an unstable stratification. This results for each grid point in a stratification that is either stable or almost stable, provided that the value of $\kappa_{v,C}$ is chosen large enough. Due to the discontinuity in the function f, Newton's method will have problems converging and it is required to use a smoothed function f. We choose

$$f(\partial\rho/\partial z) = \begin{cases} 0 & \text{if } \partial\rho/\partial z \leqslant 0, \\ 1 - \frac{1}{1+(\frac{\partial\rho}{\partial z})\frac{\Delta z_0}{\Delta\rho_0}\frac{\kappa_{v,C}-\kappa_{v,0}}{\kappa_{v,C}}} & \text{if } \partial\rho/\partial z > 0. \end{cases} \tag{13}$$

with the quotient $\Delta z_0/\Delta\rho_0$ determining the sensitivity of κ_v to $\partial\rho/\partial z$. Note that this new function f is now continuous but not continuously differentiable at $\partial\rho/\partial z = 0$ and hence convergence of Newton's method is not guaranteed. However, in practice it turns out that the lack of continuity in the derivative of f does not prevent convergence. Further note that as $\partial\rho/\partial z \to \infty$ we have that $f(\partial\rho/\partial z) \to 1$ and hence $\kappa_v \to \kappa_{v,C}$. Finally we note that if $\Delta z_0/\Delta\rho_0$ is chosen too small then unstable stratifications can occur whereas for values large enough the stratification is almost stable everywhere.

Although the convergence of Newton's method is improved by using (13) instead of (12) we still have some difficulties due to the fact that we use (6) to approximate the matrix vector product. In (6) the choice of ϵ is definitely non-trivial. On the one hand ϵ should not be too large because this introduces a truncation error, but on the other hand a value that is too small introduces cancellation errors. There is actually no guarantee that a suitable value of ϵ can be chosen such that both the cancellation and truncation errors are negligible. Since the value of $(\Delta z_0/\Delta\rho_0)$ has to be chosen quite large we have to choose very small values of ϵ to accurately compute derivatives of the vertical mixing terms involving (13). It turns out that in practice this value has to be so small that non-negligible cancellation errors are introduced.

This problem is solved by writing (and implementing) the residual as $\vec{F}(\vec{x}) = \vec{F}_{\text{vert}}(\vec{x}) + \vec{F}_{\text{other}}(\vec{x})$, with $\vec{F}_{\text{vert}}(\vec{x})$ containing the vertical mixing terms for the tracer equations and $\vec{F}_{\text{other}}$ all other terms. The action of the Jacobian is now given by

$$J_{\vec{x}_k}\vec{v} = J_{\vec{x}_k,\text{vert}}\vec{v} + J_{\vec{x}_k,\text{other}}\vec{v}, \tag{14}$$

with $J_{\vec{x}_k,\text{vert}}$ and $J_{\vec{x}_k,\text{other}}$ the Jacobian of $\vec{F}_{\text{vert}}(\vec{x}_k)$ and $\vec{F}_{\text{other}}(\vec{x}_k)$ respectively. For the part not including vertical mixing terms we use a finite difference approximation

$$J_{\vec{x}_k,\text{other}}\vec{v} = \frac{\vec{F}_{\text{other}}(\vec{x}_k + \epsilon\vec{v}) - \vec{F}_{\text{other}}(\vec{x}_k)}{\epsilon}, \tag{15}$$

with $\epsilon = 10^{-7}\cdot(1 + \|\vec{x}_k\|_2)/\|v\|_2$. To evaluate $J_{\vec{x}_k,\text{vert}}\vec{v}$ we use Coleman's method (see Appendix B) to compute and store J_{vert} explicitly in a sparse matrix format. The overhead of constructing this matrix is minimal, since it needs to be done only once each time a linear system (4) is solved and it requires only six evaluations of $\vec{F}_{\text{vert}}$ to construct $J_{\vec{x}_k,\text{vert}}$. The advantage over the finite difference approximation is that cancellation and truncation errors can be minimal because the values of $\epsilon_{k,j}$ in (B.1) can be chosen independently for each non-zero entry of $J_{\vec{x}_k}$. The values that we used are given by $\epsilon_{k,j} = 10^{-9}|x_j|$.

3.3. Preconditioning

When solving the system (4) using GMRES or another Krylov method then in practice the method will not converge since the Jacobian is very ill conditioned. We use a right preconditioner $P_{\vec{x}_k}$ to improve the convergence behavior of the Krylov solver and hence solve a system equivalent to (4)

$$JP^{-1}_{\vec{x}_k}\vec{z}_{k+1} = -\vec{F}(\vec{x}_{k+1}), \tag{16a}$$

$$P^{-1}_{\vec{x}_k}\vec{z}_{k+1} = \Delta\vec{x}_{k+1}. \tag{16b}$$

The Krylov iteration that is used to solve the above system is referred to as the outer Krylov iteration. In order to explain the preconditioner we first write the Jacobian as a block matrix as follows

$$J_{\vec{x}_k} = \begin{bmatrix} A & G & B \\ D & K & 0 \\ C & E & T \end{bmatrix}, \tag{17}$$

where we dropped the subscript $\vec{x}_k$ of the sub-blocks for the sake of readability. Here the three rows of the matrix represent the momentum, vertically integrated continuity and tracer equations respectively. The columns represent the dependency on horizontal velocity, sea surface height and tracers, respectively. The A block contains advection and diffusion of momentum and the Coriolis parameter, the G and D blocks contain the gradient of sea surface height and the divergence of vertically integrated velocity. The K block contains a smoothing operator to suppress a null-mode existing on the B-grid used in MOM4 and the B block contains the buoyancy terms, i.e. the horizontal pressure gradient due to density differences (and not due to surface height elevation) given by $\int_{z'=z}^{0} g\nabla\rho dz'$. For the last row we have the C block representing the dependency of tracer advection on horizontal velocity and the T block containing the change in advection and diffusion due to changes in the tracers itself. Finally, the dependency on the thickness of the upper most layer in the discretized tracer equations is represented in block E.

We now use a block Gauss–Seidel preconditioner, similar to the one described in de Niet et al. [9] and hence $P_{\vec{x}_k}$ is defined by neglecting the block B containing buoyancy terms in the Jacobian J. To apply this preconditioner to a vector $\vec{b} = \begin{bmatrix} \vec{b}_{uv} & \vec{b}_{\eta} & \vec{b}_{ts} \end{bmatrix}^T$ with $\vec{b}_{uv}, \vec{b}_{\eta}$ and $\vec{b}_{ts}$ the parts corresponding to the horizontal velocities, sea surface height and tracers respectively, we need to find $\vec{y}$ satisfying

$$P_{\vec{x}_k}\vec{y} = \vec{b}. \tag{18}$$

Solving the above system is rather easy since by neglecting the B block the preconditioning matrix $P_{\vec{x}_k}$ becomes (block) triangular and hence we first solve the system

$$\begin{bmatrix} A & G \\ D & K \end{bmatrix} \begin{bmatrix} \vec{y}_{uv} \\ \vec{y}_{\eta} \end{bmatrix} = \begin{bmatrix} \vec{b}_{uv} \\ \vec{b}_{\eta} \end{bmatrix}, \tag{19}$$

using a GMRES [19,17] process. Because this GMRES process is applied each time that the preconditioner $P_{\vec{x}_k}$ needs to be applied we refer to it as an inner (GMRES) iteration. We use MRILU [20] as a left preconditioner to speed up the convergence of this inner iteration. Computing this preconditioner requires the matrix of the system (19) which can be obtained using Coleman's method (see Appendix B). Note that this requires the sparsity pattern of the matrix, which was already determined in Bernsen et al. [15]. Once we have the matrix we can use MRILU to compute a preconditioner for system (19). The computation of this matrix is actually a very costly step due to the large number of entries in the sparsity pattern and therefore we compute a new preconditioner for this inner (GMRES) iteration only if it fails to satisfy the stopping criterion

$$\left\| \begin{bmatrix} A & G \\ D & K \end{bmatrix} \begin{bmatrix} \vec{y}_{uv} \\ \vec{y}_{\eta} \end{bmatrix} - \begin{bmatrix} \vec{b}_{uv} \\ \vec{b}_{\eta} \end{bmatrix} \right\|_2 < \eta_{uv\eta} \left\| \begin{bmatrix} \vec{b}_{uv} \\ \vec{b}_{\eta} \end{bmatrix} \right\|_2, \tag{20}$$

within a specified maximum number of iterations. The next step is to compute the right hand side for the tracer equations as follows

$$\vec{b}_{ts} \leftarrow \vec{b}_{ts} - \begin{bmatrix} C & E \end{bmatrix} \begin{bmatrix} \vec{y}_{uv} \\ \vec{y}_{\eta} \end{bmatrix}. \tag{21}$$

Note that we do not explicitly need the matrix $[C\ E]$, but rather need the action of this matrix which can be computed using (14). Finally we need to solve the system

$$T\vec{y}_{ts} = \vec{b}_{ts}. \tag{22}$$

and this is done similarly to solving (19), using an inner GMRES iteration with MRILU as a left preconditioner. To compute this preconditioner we approximate T using Coleman's method (see Appendix B) and to avoid computing T at every Newton step we only update this preconditioner for system (22) if the inner (GMRES) iteration fails to satisfy the stopping criterion

$$\|T\vec{y}_{ts} - \vec{b}_{ts}\|_2 < \eta_{ts}\|\vec{b}_{ts}\|_2, \tag{23}$$

within a specified maximum number of iterations. Finally, the vector $\vec{y} = \begin{bmatrix} \vec{y}_{uv} & \vec{y}_{\eta} & \vec{y}_{ts} \end{bmatrix}$ satisfies (18) and is the result of applying the preconditioner $P_{\vec{x}_k}$ to a vector.

The above procedure requires the availability of the matrix vector product with A, $[C\ E]$ and the matrix of (19), which can be computed using (14). For instance, to compute $T\vec{y}_{ts}$ we use (14) to compute

$$\begin{bmatrix} B\vec{y}_{ts} \\ 0 \\ T\vec{y}_{ts} \end{bmatrix} = \begin{bmatrix} A & G & B \\ D & K & 0 \\ C & E & T \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vec{y}_{ts} \end{bmatrix} \tag{24}$$

and now $T\vec{y}_{ts}$ is obtained simply as the third block of the result of the above matrix–vector product. The advantage of this method is that it is very easy to program once a routine exists that computes the action $J\ v$, while the main disadvantage is that is not a very efficient implementation.

The preconditioner $P_{\vec{x}_k}$ described above is well suited for a JFNK approach and to see this we consider how expensive evaluating the blocks of the Jacobian using Coleman's method would be. The B block contains buoyancy terms, $\int_{z'=z}^{0} g\nabla\rho dz'$, and due to this vertical integral a lot of non-zero entries are introduced, resulting in the need for a lot of matrix–vector products when we would explicitly evaluate this block. The C block includes changes in vertical tracer advection due to changes in the vertical velocity. Since the vertical velocity is eliminated from the state vector (see Appendix A) any dependency on vertical velocity causes a lot of fill in in the sparsity pattern and hence this block would also require a lot of matrix vector products to compute. In the block Gauss–Seidel preconditioner that is used here the expensive computation of the B and C block is avoided since B is neglected and for C only the availability of the matrix–vector product is important. The matrix in(19) actually needs to be computed and this too is quite expensive for the same reasons that the C block is expensive to compute. However, this block needs not to be computed very often because it is only needed to compute a preconditioner for solving the system (19) and in practice the same preconditioner can be used over many Newton steps. Finally, the T block also needs

to be computed explicitly, and in practice it is computed more often than the matrix in (19) because of the presence of convective adjustment in this block. This is not really a problem, since evaluating the T block is relatively cheap due to the absence of a vertical integral as in the B block or the presence of a dependency on vertical velocity.

We note that applying the preconditioner $P_{\vec{x}_k}$ introduces some inaccuracies due to the fact that the systems (19) and (22) are not solved exactly, but only approximately up to the precision specified in (20) and (23). The result of these inaccuracies is that a slightly different preconditioner $P_{\vec{x}_k}$ is used in every outer Krylov iteration. We therefore use the F (lexible) GMRES [21] method that, contrary to ordinary GMRES, is able to deal with a different preconditioner in each iteration.

4. Results

To test the JFNK method we consider an idealized configuration consisting of a spherical sector with a longitudinal range of $0° \leqslant \phi \leqslant 64°$ and a latitudinal range of $10°\text{N} \leqslant \theta \leqslant 74°\text{N}$. The ocean basin has a constant depth of $D = 5500$ m. We use 16 layers in the vertical, ranging from 25 m in the upper layer to 871 m in the bottom layer. At the surface we apply a wind stress profile given by

$$\tau^{\phi} = \lambda\tau_0 \cos\left(2\pi \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}\right), \tag{25a}$$

$$\tau^{\theta} = 0, \tag{25b}$$

with the amplitude $\tau_0 = 0.1$ Pa. Temperature and salinity are restored to

$$T = T_0 + \Delta T \cos\left(\pi \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}\right), \tag{26a}$$

$$S = S_0 + \Delta S \cos\left(\pi \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}\right), \tag{26b}$$

with restoring timescales $\tau_T = 30$ days and $\tau_S = 30$ days and amplitudes $\Delta T = 12.5$ °C and $\Delta S = 1$ psu for temperature and salinity respectively. Reference values for temperature and salinity are given by $T_0 = 15$ °C and $S_0 = 35$ psu. The density follows immediately from temperature and salinity using a linear equation of state

$$\rho = \rho_0 - \alpha(T - T_0) + \beta(S - S_0), \tag{27}$$

with the thermal expansion coefficient $\alpha = 10^{-1}$ kg m^{-3} K^{-1}, coefficient of saline contraction $\beta = 7.6{\cdot}10^{-1}$ kg m^{-3} psu^{-1} and reference density $\rho_0 = 1035$ kg m^{-3}. In the horizontal we use Laplacian friction and diffusion for tracers with coefficients given by $A_H = 2.5 \times 10^5$ m^2 s^{-1} and $K_H = 10^3$ m^2 s^{-1} respectively. We use a constant vertical friction coefficient of $A_V = 10^{-3}$ m^2 s^{-1} and in the parametrization for vertical mixing of tracers (11) we use $\kappa_{v,0} = 10^{-4}$ m^2 s^{-1} and $\kappa_{v,C} = 1$ m^2 s^{-1}. In the case of the new convective adjustment scheme (13) we used a value of $(\Delta z_0/\Delta\rho_0) = 10^4$ m^4 kg^{-1}.

We first consider the effect of the new convective adjustment scheme. We performed two 5000 yr simulations with the explicit timestepping version of MOM4, one with the new convective adjustment scheme (13) and one with the old scheme (12). We used a horizontal resolution of 16×16 grid points and the initial condition at $t = 0$ consists of a state of rest, with no sea surface elevation and a uniform temperature and salinity distribution.

In Fig. 1 we plot at each timestep $\|\vec{F}_t(\vec{x}(t))\|_2/\|\vec{F}_t(\vec{x}(0))\|_2$, with $\vec{F}_t(\vec{x}(t))$ the part of the residual corresponding to the temperature equation (i.e. tendency of temperature) and $\vec{x}(t)$ the state at time t obtained from the timestepper. For MOM4 with (12) we see spikes every now and then resulting from the value of κ_v switching from $\kappa_{v,0}$ to $\kappa_{v,C}$ or the other way around. For MOM4 with (13) the residual approaches an equilibrium solution much more smoothly, with only a few sharp peaks in the residual in the first 300 yr.

From Fig. 1 it is clear that for both the convective adjustment schemes the residual approaches zero and hence for both schemes an equilibrium solution is obtained. To demonstrate that this is the same equilibrium, we plot the meridional overturning streamfunction of the equilibrium solution for the new and old scheme in Fig. 2(a) and (b), respectively. Although there are quantitative differences, qualitatively the solutions are very similar. In Fig. 2 (c) and (d) the zonally averaged deviation from the reference density is plotted and again the new and old convective adjustment scheme give similar results. Note that in both cases the stratification appears to be almost stable everywhere. In Fig. 3, the depth averaged and zonally averaged vertical mixing coefficient is plotted for both equilibrium solutions. For the old scheme the value of κ_v is up to an order of magnitude larger at places where convective adjustment is active. However, the patterns in the vertical mixing field are very similar. For both the old and the new scheme convective adjustment mainly takes place in the northern regions and near the surface, although along the eastern boundary the region of convective adjustment extends more southward.

We now try to obtain the equilibrium solution, using the scheme (13) in MOM4, with the MOM4-JFNK model. Starting from a state of rest we increase the forcing λ in small steps of $\Delta\lambda = 0.025$ from $\lambda = 0$ to $\lambda = 1$ as described in Section 3.1. The stopping criterion of Newton's method is given by (7) with $\epsilon_N = 10^{-4}$. In the minimal reduction method we use values of $\xi = 0.95$ in (9) and $\gamma = 0.7$ in (10) with the maximum number of θ_k updates set to 50. The maximum number of (outer) iterations in the FGMRES method that is used to solve (4) is set to 200 and we do not restart the solver during these 200 iterations. The linear system is only solved up to a low accuracy of $\eta = 0.1$ because the convergence rate of Newton's method is very poor anyway due to convective adjustment. For solving the systems (19) and (22) we use a stopping criterion of

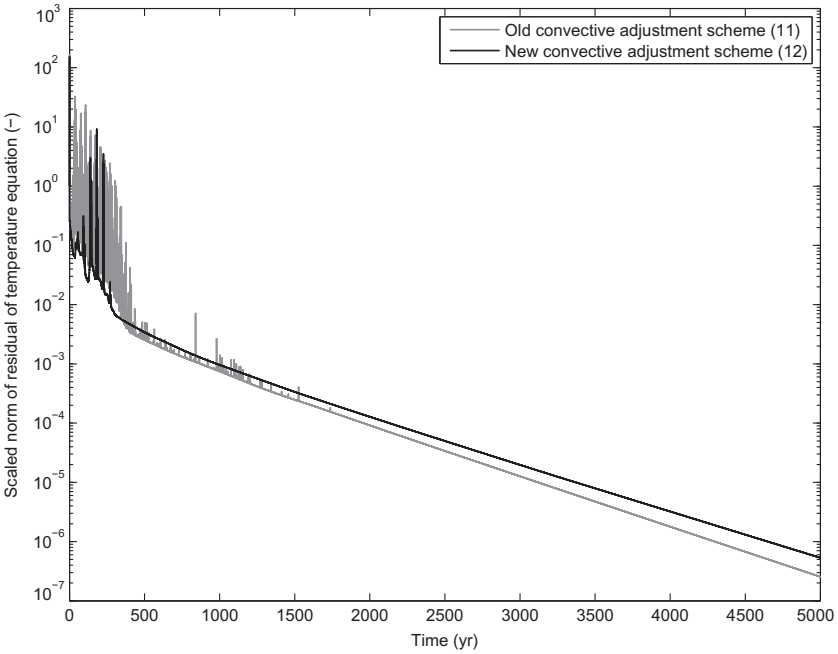

Fig. 1. The scaled norm of the residual $\|F_t(t)\|_2/\|F_t(0)\|_2$ for the temperature equation at each time step of a 5000 yr time-stepping run with MOM4. The grey line is the result when the old parametrization of convective adjustment (12) is used while the black line is the result of the new parametrization (13).

$\eta_{ts} = 10^{-4}$ in (23) and $\eta_{uvh} = 10^{-4}$ in (20), respectively. The maximum number of (inner) GMRES iterations in both cases is set to 30 with no restart during these 30 iterations.

Note that in the MOM4-JFNK model we work with a dimensionless state vector and residual. The dimensional and dimensionless state vector, denoted by $\vec{x}'$ and $\vec{x}$ respectively, are related by

$$\vec{x}'_{uv} = \mathcal{X}_{uv}\vec{x}_{uv}, \quad \vec{x}'_{\eta} = \mathcal{X}_{\eta}\vec{x}_{\eta}, \quad \vec{x}'_{t} = \mathcal{X}_{t}\vec{x}_{t}, \quad \vec{x}'_{s} = \mathcal{X}_{s}\vec{x}_{s},$$

where the uv, η, t and s subscripts refer to the part of the state vector corresponding to the momentum equations, sea surface height equation, temperature equation and salinity equation respectively. The scaling factors are given by $\mathcal{X}_{uv} = 10^{-2}$ m s^{-1}, $\mathcal{X}_{\eta} = 10^{-1}$ m, $\mathcal{X}_{t} = 10\,^{\circ}$C and $\mathcal{X}_{s} = 1$ psu. The residual is similarly scaled as follows

$$\vec{F}'_{uv} = \mathcal{F}_{uv}\vec{F}_{uv}, \quad \vec{F}'_{\eta} = \mathcal{F}_{\eta}\vec{F}_{\eta}, \quad \vec{F}'_{t} = \mathcal{F}_{t}\vec{F}_{t} \quad , \vec{F}'_{s} = \mathcal{F}_{s}\vec{F}_{s} \quad ,$$

with the scaling factors given by $\mathcal{F}_{uv} = 10^{-8}$ m s^{-2}, $\mathcal{F}_{\eta} = 10^{-4}$ s^{-1}, $\mathcal{F}_{t} = 10^{-6}\,^{\circ}$C s^{-1} and $\mathcal{F}_{s} = 10^{-7}$ psu s^{-1}.

In Fig. 4(a) we plot $\|\psi_{\mathrm{M}}(t) - \psi_{\mathrm{M,JFNK}}\|_{\max}$ with $\psi_{\mathrm{M}}(t)$ and $\psi_{\mathrm{M,JFNK}}$ the meridional overturning streamfunction of the MOM4 timestepping model at time t and of the steady-state solution of the MOM4-JFNK model, respectively. Clearly the timestepper and the JFNK method approach the same equilibrium solution. In Fig. 4(b) we plot $\log|\psi_{\mathrm{M}}(t) - \psi_{\mathrm{M,JFNK}}|$ at $t = 5000$ yr showing only very small differences between the solution of the timestepper and of the JFNK method.

We now apply the JFNK method for higher resolutions: $16 \times 16 \times 16$, $32 \times 32 \times 16$ and $64 \times 64 \times 16$. For all resolutions the same physical and numerical parameters, which are given above, are used. For all resolutions the JFNK method converges and an equilibrium solution is found. In Fig. 5 the average number of (outer) FGMRES iterations per Newton step is reported as a function of the forcing strength λ. For all resolutions convergence is very quickly when the forcing is almost zero but it becomes slower at stronger forcing; the number of iterations also increases with increasing resolution. However, except for the very last continuation step, the difference in iterations between the resolutions $32 \times 32 \times 16$ and $64 \times 64 \times 16$ is not very large. In Table 1 we see that the number of inner iterations for solving the systems (19) and (22) increases for higher resolutions. The total number of Newton steps even decreases and this is due to the fact that the stopping criterion (7) depends scales with the number of grid points.

Most important is of course the CPU time that was needed. For each resolution we make a comparison to the CPU time that a 5000 yr timestepping run at the same resolution would cost. To compute this CPU time a much shorter run was performed to estimate the amount of CPU time that is required per physical year of timestepping. Note that these timestepping runs use an almost identical configuration as the MOM4-JFNK model, with slight differences in parameters and the old convective adjustment scheme (13) is used. However, these differences are not expected to change the timing of the model. At the $16 \times 16 \times 16$ resolution a timestep of 1 day for the tracers and surface height equation, 0.5 days for the momentum equations and 90 barotropic sub-timesteps are used. For the $32 \times 32 \times 16$ and $64 \times 64 \times 16$ resolutions these timesteps are divided by two and four, respectively.

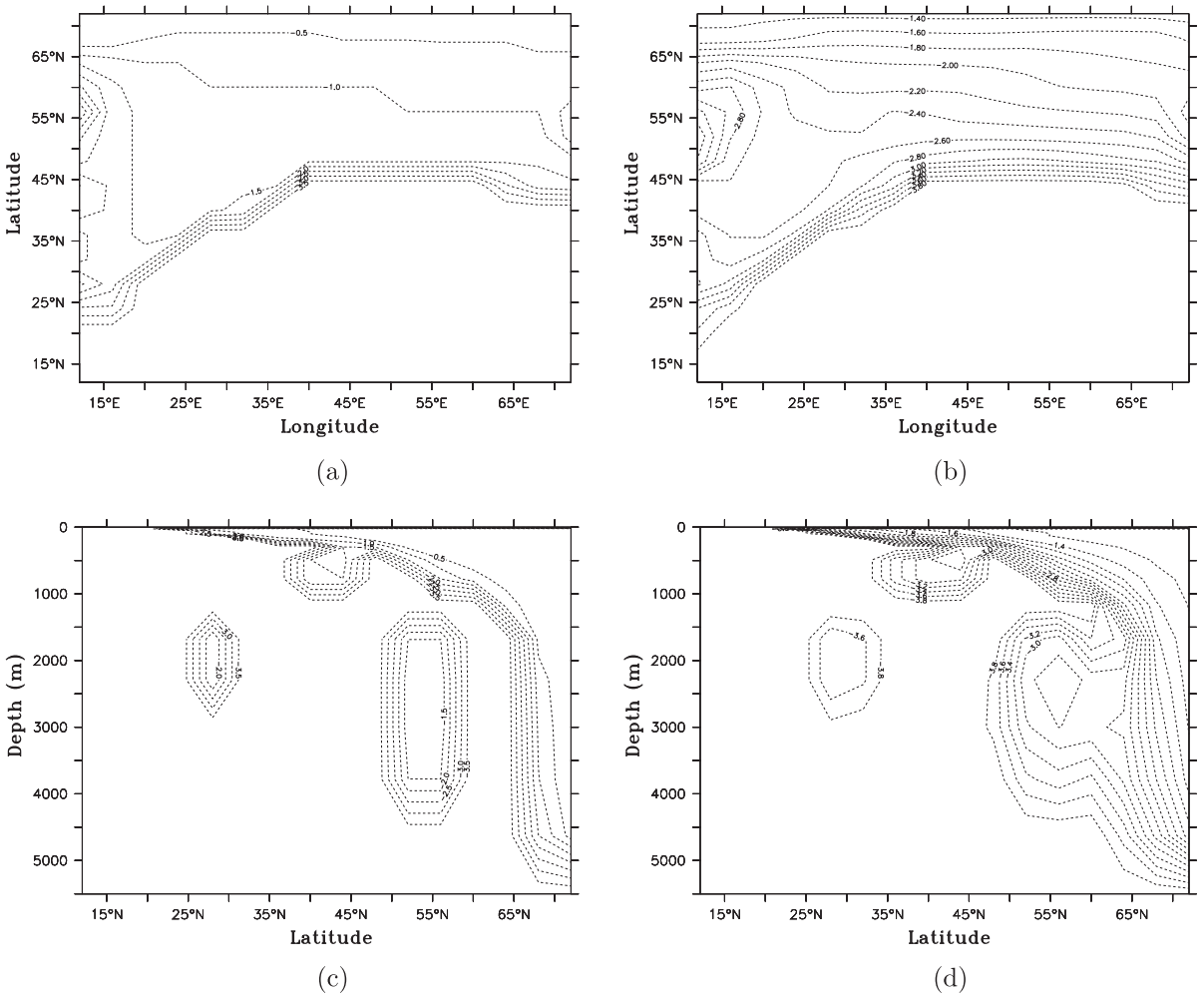

Fig. 3. (a) The vertically averaged vertical mixing coefficient ($m^2\ s^{-1}$) on a log scale of the equilibrium solution found after 5000 yr for the test problem with the old convective adjustment scheme (12). (b) Same, but now with scheme (13). (c) The zonally averaged vertical mixing coefficient ($m^2\ s^{-1}$) on a log scale of the equilibrium solution found after 5000 yr for the test problem with the old convective adjustment scheme (12). (d) Same, but now with scheme (13).

For all resolutions the JFNK method was clearly faster than a timestepping run of 5000 yr at the same resolution. The speed-up is in the order of a factor 12–24 depending on resolution. In this example a higher resolution seems to favor the JFNK method. For the timestepper, doubling the horizontal resolution leads to an increase of approximately a factor of eight in CPU time, due to four times as many grid points and the halving of the timestep. The JFNK method suffers less from the higher resolutions with a doubling of horizontal resolution leading to an increase of CPU time with a factor of 4.4 or 6.4 depending on resolution.

5. Summary and discussion

In this paper the JFNK method has been applied to the state-of-the-art model MOM4 to shorten the CPU time of the computation of an equilibrium state of the model. As a typical example, we have computed the wind- and thermohaline driven flows in a northern hemispheric spherical sector. The implementation of this method is far from trivial because of the preconditioning and because of the implementation of convective adjustment. A slight adaptation of the original convective adjustment scheme in MOM4 was therefore required.

As demonstrated, the speed-up ranges from a factor of 12 (for the lowest resolution case) to 24 (for the highest resolution case). The JFNK method scales better with horizontal resolution than the timestepper. The timestepper requires in principle a factor of 8 more CPU time due to doubling of horizontal resolution, while for the JFNK method this turns out to be significantly lower. Improvements in these speed-ups are likely possible. For instance it is possible to use a less strict stopping criterion for Newton's method (higher value of ϵ_N in (7)) for forcing straights $\lambda < 1$. It is after all only the equilibrium solution at

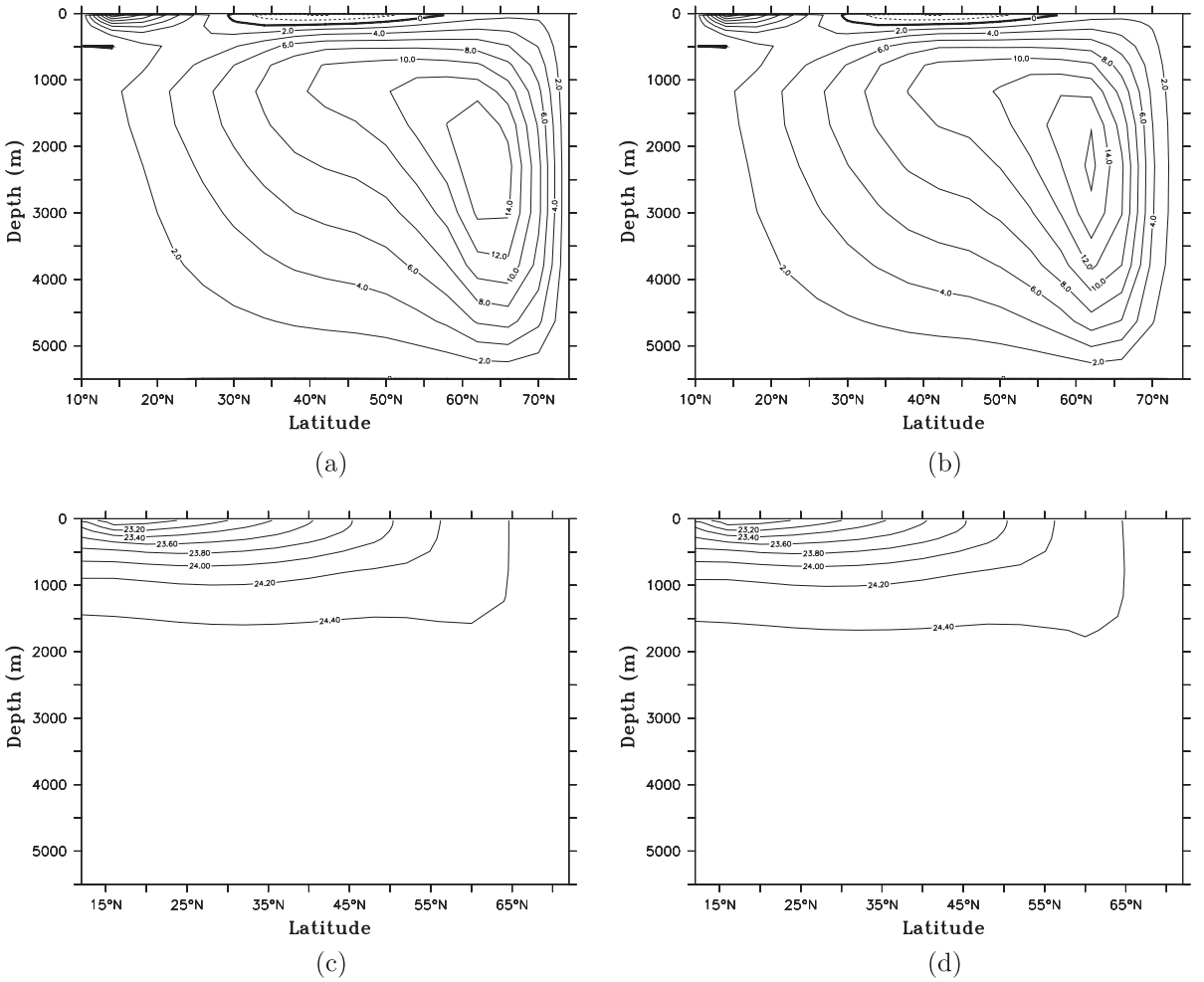

Fig. 2. (a) The meridional overturning streamfunction (Sv) of the equilibrium solution found in MOM4 after 5000 yr for the test problem with the old convective adjustment scheme (12). (b) Same, but now with scheme (13). (c) The zonally averaged deviation from the reference density (kg m^{-3}) of the equilibrium solution found after 5000 yr with (12). (d) Same, but now with scheme (13).

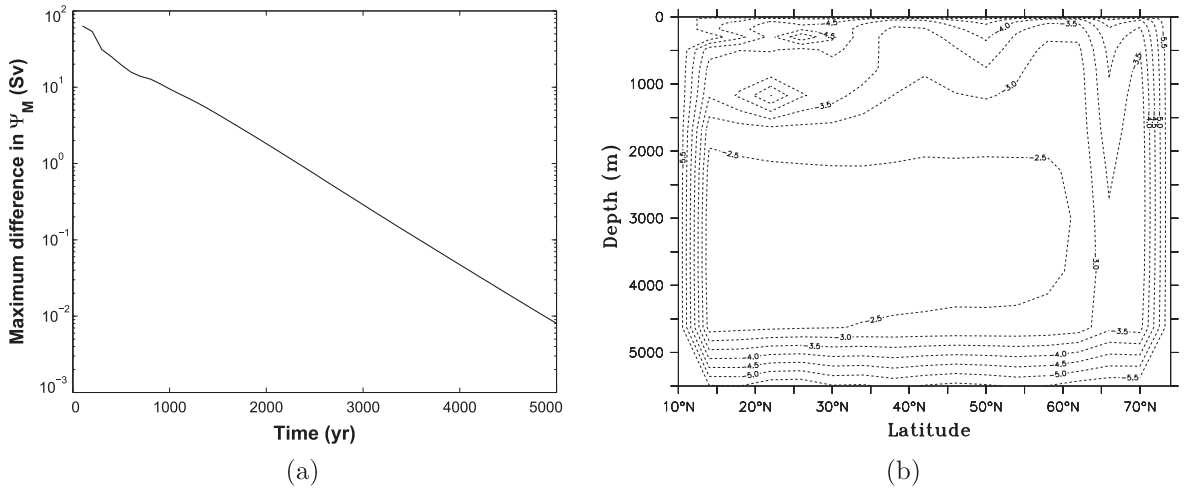

Fig. 4. (a) Maximum difference between the meridional overturning streamfunction (Sv) obtained by the timestepper ($\Psi_M(t)$) and by the JFNK method ($\Psi_{M,JFNK}$) versus time in years. In (b) this difference field is plotted at the final time step of the timestepper at $t = 5000$ yr on a log scale.

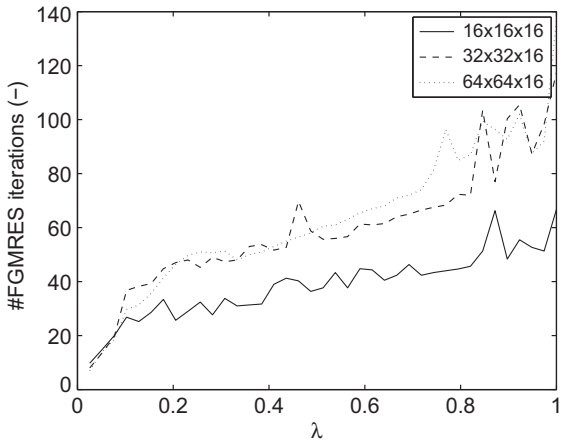

Fig. 5. The average number of (outer) FGMRES iterations per Newton iteration at a resolution of $16 \times 16 \times 16$, $32 \times 32 \times 16$ and $64 \times 64 \times 16$ as a function of the forcing strength λ.

Table 1
Number of Newton iterations, FGMRES outer iterations, (F)GMRES inner iterations and F evaluations for several resolutions. Also the total CPU time for the MOM4-JFNK model and the speedup compared with 5000 yr of timestepping with MOM4 is reported.

Resolution	$16 \times 16 \times 16$	$32 \times 32 \times 16$	$64 \times 64 \times 16$
Number of unknowns	15,648	64,544	262,176
Total number of Newton iterations	145	125	116
Total number of F evaluations ($\times 10^5$)	1.4	2.5	2.9
Average number of iterations for (19)	4.4	6.4	8.3
Average number of iterations for (22)	12.3	15.8	18.4
CPU time of MOM4-JFNK (h)	1.1	7.04	31.8
CPU time of MOM4 timestepping (h)	14.0	96.6	763.1
Speedup	12.7	13.1	24.0

$\lambda = 1$ that we are really interested in. Another improvement could be to replace the function f in (13) with a function that is continuous and continuously differentiable everywhere possibly leading to a higher convergence rate of Newton's method.

In this article we used the JFNK method to compute steady states only, but it can also be applied to transient runs using a fully implicit time-stepping scheme. Although we expect lower speed-ups than for the spin-up problem, there is still a possible reduction in CPU time due to the much larger timesteps that can then be used. We expect that the systems resulting from these implicit time-stepping schemes will be easier to solve than the full steady system, because it is generally better conditioned due to an increase of the diagonal values of the Jacobian matrix. Furthermore, implicit timestepping could be used to obtain periodic equilibria under periodic boundary conditions. We consider periodic forcing as a steady forcing plus a periodic perturbation. Hence we can use a two-step approach: First a spin-up with a steady forcing and next starting from the resulting equilibrium solution a transient computation with periodic forcing.

Applying the method to other ocean models is certainly possible, but some effort is required. As a first step the residual should be made available and it really depends on the implementation of the existing explicit timestepping code how difficult this is. This is made more complicated by the fact that we actually need two residual functions ($\vec{F}_{\text{vert}}$ and $\vec{F}_{\text{other}}$) and also because most ocean models use some kind of barotropic-baroclinic mode-splitting.

For ocean models that use a rigid-lid approach instead, we have to realize that it is no longer possible to express the model as (1) because the continuity equation is an algebraic constraint without time derivatives. In this case (1) should then be replaced with $M\mathrm{d}\vec{x}/\mathrm{d}t = \vec{F}(\vec{x})$ where M is a diagonal matrix having the value of one at the diagonal elements for the prognostic equations and a value of zero for algebraic constraints.

Changing the parametrization for convective adjustment is usually easily done in most ocean models provided that convective adjustment is already implemented using a variable vertical mixing coefficient. Finally, in the preconditioner we use the sparsity pattern of several parts of the Jacobian matrix to compute blocks of the Jacobian matrix with Coleman's method. There is an ongoing effort to apply JFNK methods to the Parallel Ocean Program (POP [22]) ocean model as well and it turns out that the sparsity patterns for MOM4 and POP are in fact very similar.

In this paper we used restoring boundary conditions for both temperature and salinity. A more common choice of boundary conditions are mixed boundary conditions, where temperature is restored to a prescribed profile and for salinity a freshwater flux is prescribed. When such a flux boundary condition is used a null space is introduced and the salinity field of the equilibrium solution is only determined up to a constant. In principle, to remove this null space we can replace one of the equations of the residual with a constraint on total salinity

$$\vec{w} \cdot \vec{x}_s = S_{\text{tot}},$$

with S_{tot} the total amount of salt in the ocean, $\vec{x}_s$ the part of the state vector corresponding to the salinity field and $\vec{w}$ containing the corresponding volumes of grid boxes. Of course, these changes in the residual require that the preconditioner is adjusted as well.

In summary, the methodology in this paper may lead to a more efficient computation of spin-up solutions in a large class of state-of-the-art ocean models.

Appendix A. Computation of the residual in MOM4

In this appendix we show how the residual in MOM4 is computed making use of the existing timestepping code as much as possible. MOM4 uses The Arakawa B-grid which is a staggered grid with two types of cells: *U*-cells for horizontal velocities and *T*-cells for sea-surface height and tracers. The corners of each *T*-cell consist of the centers of four *U*-cells and both horizontal velocity components are defined on the same grid points. MOM4 uses a two level time-stepping scheme for which the grid is also staggered in time. At time-level $\tau - 1/2$, that is at time $t = \Delta t(\tau - 1/2)$, the variables on *T*-cells are defined, such as temperature, salinity and pressure. At the time-level τ, variables on the *U*-cells are defined, such as the horizontal velocities. The only variable defined at both time-levels is the sea-surface height.

A simplified description of the two-level time-stepping scheme is given below. At time-level $\tau - 1/2$, the sea-surface height ($\eta_T^{\tau-1/2}$), temperature ($T^{\tau-1/2}$), salinity ($S^{\tau-1/2}$) and vertical grid spacing ($\Delta z_T^{\tau-1/2}$) are given at *T*-cells. Furthermore, the horizontal velocities ($\vec{u}^{\tau}$), the vertically integrated velocities ($\vec{U}^{\tau}$) and the vertical grid spacing (Δz_u^{τ}) are available at *U*-cells and at time-level τ. At time-level τ the sea-surface height is given at *U*-cells (η_u^{τ}) as well as *T*-cells (η_T^{τ}). To update the variables from $\tau - 1/2$ to $\tau + 1/2$ and from τ to $\tau + 1$, the following procedure is applied:

1. Update variables defined on time-level $\tau - 1/2$ to time-level $\tau + 1/2$.
   (a) Compute the tendency (i.e. an approximation of the time derivative) $\delta\eta_T$ of the sea-surface height at time-level τ using $\vec{U}^{\tau}$ and calculate the thickness weighted tendencies, δT and δS, for temperature and salinity. This weighting is needed for the conservation of heat and salinity. Note that for the computation of δT and δS we need not only horizontal velocities, which are given, but also vertical velocities, which follow immediately from the horizontal velocities using the continuity equation. The vertical diffusion terms are usually treated implicitly, but optionally it is also possible to treat them explicitly.
   (b) Update the sea-surface height to time-level $\tau + 1/2$:

   $$\eta_T^{\tau+1/2} \leftarrow \eta_T^{\tau-1/2} + \delta\eta_T$$

   and update the vertical grid spacing $\Delta z_T^{\tau+1/2}$ at *T*-cells.
   (c) Now update *T* and *S* from $\tau - 1/2$ to $\tau + 1/2$:

   $$(T,S)^{\tau+1/2} = \frac{\Delta z_T^{\tau-1/2}(T,S)^{\tau-1/2} + \Delta t(\delta T, \delta S)}{\Delta z_T^{\tau+1/2}}.$$

2. Update variables defined on time-level τ to time-level $\tau + 1$.
   (a) Update the barotropic variables, using sub-time stepping, to obtain $\vec{U}^{\tau+1}$, $\eta_u^{\tau+1}$ and $\eta_T^{\tau+1}$.
   (b) Update the vertical grid spacing $\Delta z_u^{\tau+1}$ at *U*-cells using the sea-surface height obtained in the previous step.
   (c) Compute the thickness weighted baroclinic velocity tendencies, say $\delta\vec{u}$. These are obtained from the momentum equations, where the term $\Delta z_u g \nabla\eta$ is omitted from the pressure gradient term.
   (d) Compute the velocity field at $\tau + 1$ as follows

   $$\vec{u}^{\tau+1} = \frac{\Delta z_u^{\tau}\vec{u}^{\tau} + \Delta t\delta\vec{u}}{\Delta z_u^{\tau+1}}$$

   and correct these velocities such that the vertically integrated velocity matches the vertically integrated velocity obtained from the barotropic sub-time stepping

   $$\vec{u}^{\tau+1} \leftarrow \vec{u}^{\tau+1} + \frac{\vec{U}^{\tau+1} - \sum \Delta z_u^{\tau+1}\vec{u}^{\tau+1}}{\sum \Delta z_u^{\tau+1}},$$

   where the summation is over all grid points in the same fluid column.

To calculate the residual function $\vec{F}(\vec{x})$ of MOM4, we use a state vector $\vec{x}$ defined as

$$\vec{x} = [\vec{x}_{uv}, \vec{x}_\eta, \vec{x}_t, \vec{x}_s], \tag{A.1}$$

consisting of the temperature ($\vec{x}_t$), salinity ($\vec{x}_s$) and sea-surface height field ($\vec{x}_\eta$) at *T*-cells and the horizontal velocity field ($\vec{x}_{uv}$) at *U*-cells. Assuming that these fields are available we use the following algorithm to compute the residual $\vec{F}$:

1. Compute variables which depend directly on the state vector, such as the sea-surface height at U-cells, η_u, vertical grid spacing at T and U-cells, Δz_T and Δz_U and vertically integrated velocity field $\vec{U}$. The vertical grid spacing depends directly on the sea-surface height fields while the vertically integrated horizontal velocity depends directly on the vertical grid spacing and the 3D velocity field $\vec{u}$. The sea-surface height at U-cells is obtained by linear interpolation of the sea-surface height at T-cells.
2. Compute the tendency $\delta\eta_T$ of η_T using $\vec{U}$ and calculate the thickness weighted tendencies, δT and δS, for temperature and salinity. Note that here we use the option to treat vertical diffusion explicitly rather than implicitly.
3. Compute tendencies for horizontal velocities.
   (a) Compute the thickness weighted baroclinic velocity tendencies, $\delta\vec{u}$. These are obtained from the momentum equations, where the term $\Delta z_u g\nabla\eta$ is omitted from the pressure gradient term.
   (b) Correct for the omitting part in the pressure gradient term by adjusting the thickness weighted velocity tendency as follows

$$\delta\vec{u} \leftarrow \delta\vec{u} - \Delta z_u g\nabla\eta.$$

The residual is now given by the vector $\vec{F}(\vec{x}) = \left[\vec{F}_{uv}(\vec{x}), \vec{F}_\eta(\vec{x}), \vec{F}_t(\vec{x}), \vec{F}_s(\vec{x})\right]$ with $\vec{F}_{uv}(\vec{x}) = \delta\vec{u}/\Delta z_U$, $\vec{F}_\eta(\vec{x}) = \delta\eta_T$, $\vec{F}_t(\vec{x}) = \delta T/\Delta z_T$ and $\vec{F}_s(\vec{x}) = \delta S/\Delta z_T$. Note that $\delta\vec{u}$, δT and δS contain the thickness weighted tendencies and these are converted to normal tendencies by dividing by Δz_T or Δz_U.

Steps (2) and (3a) of the residual calculation require most of the coding, but these are the steps for which we can directly reuse code from the time-stepping algorithm. Step (1) is relatively easily implemented because of the modular setup of MOM4. Subroutines for calculating the vertical grid spacing at U and T-cells from the sea-surface height are directly available and so are subroutines for interpolating fields from T-cells to U-cells. Only the vertically integrated velocity field needs to be computed by hand in step (1), but this takes only a few lines of code. For step (3) a subroutine for calculating the omitted term was already available.

There is one issue with the free-surface formulation. Due to volume conservation of the computational domain it holds that the integral of the time derivative of the sea-surface height over the domain (it is only defined on a 2D domain) is zero. By construction this property is retained in the numerical scheme and hence $\eta_T \cdot \vec{v} = 0$, where $\vec{v}$ contains the area of the grid cell on which the corresponding element in η_T is defined. Since the time derivative of η_T is a part of the residual vector $\vec{F}(\vec{x})$ we have that $\vec{F}(\vec{x}) \cdot \vec{w} = 0$, where $\vec{w} = [0, \vec{v}, 0, 0]$ is just $\vec{v}$ extended by zeros for the other components. The resulting Jacobian satisfies

$$J^T\vec{w} = 0$$

and with $\vec{w} \neq; 0$ this means that the Jacobian is singular and the JFNK method will break down. To solve this issue we remove one of equations for the sea-surface height from the residual $\vec{F}(\vec{x})$ and replace it with the condition

$$\vec{x} \cdot \vec{w} = 0. \tag{A.2}$$

The resulting Jacobian matrix will generically be non-singular and we can apply the JFNK method.

Hence, using this algorithm we can calculate the residual of MOM4 efficiently and reuse most of the original MOM4 code. The same subroutines that are required for time stepping are also useful for the calculation of the residual and only the order in which the subroutines are called slightly changes.

Appendix B. Method of Coleman

In Coleman et al. [23] and Coleman [24] a method is described that can be used to approximate the Jacobian matrix from a residual $\vec{F}(\vec{x})$ assuming that the sparsity pattern of the Jacobian is known. First we use the sparsity pattern of J to find a partition C_1, C_2, C_3,…, C_q of it's columns such that no two columns in $C_k (k = 1,\ldots,q)$ share a non-zero entry on the same row. Given a non-zero entry (i,j) of J we have a unique k such that $j \in C_k$ and we can use the finite difference approximation

$$J_{ij} \approx \frac{F_i(\vec{x} + \epsilon_{kj}\vec{e}_j) - F_i(\vec{x})}{\epsilon_{kj}}, \tag{B.1}$$

with $\vec{e}_j$ the j-th unit vector and for small values of ϵ_{kj}. Since no two columns in C_k share a non-zero entry on the same row we have that $J_{ij} = 0$ for each $\bar{j} \in C_k$ other than $\bar{j} = j$ and hence we can write

$$J_{ij} \approx \frac{F_i(\vec{x} + \sum_{\bar{j}\in C_k}\epsilon_{kj}\vec{e}_j) - F_i(\vec{x})}{\epsilon_{kj}} = \frac{F_i(\vec{x} + \vec{v}) - F_i(\vec{x})}{\epsilon_{kj}},$$

with $\vec{v}_k = \sum_{j\in C_k}\epsilon_{kj}\vec{e}_j$. Hence we can approximate J using the residual evaluations $\vec{F}(\vec{x})$ and $\vec{F}(\vec{x} + \vec{v}_k)(k = 1, 2, \ldots, q)$. The efficiency of the method is determined by the number of parts q of the column partitioning. In practice this number q is much smaller than the dimension of $\vec{x}$ and the number is determined by the stencil used in the discretization (and hence usually independent of the resolution).

References

[1] S. Griffies, M. Harrison, R. Pacanowski, A. Rosati, A Technical Guide to MOM4, NOAA/Geophysical Fluid Dynamics Laboratory, 2004. <http://www.gfdl.noaa.gov/fms>.
[2] R. Bleck, C. Rooth, D. Hu, L. Smith, Salinity-driven thermocline transients in a wind- and thermohaline-forced isopycnic coordinate model of the north atlantic, J. Phys. Oceanogr. 22 (1992) 1486–1505.
[3] R. Bleck, An oceanic general circulation model framed in hybrid-cartesian coordinates, Ocean Model. 4 (2001) 55–88.
[4] S. Danilov, G. Kivman, J. Schröter, A finite-element ocean model: principles and evaluation, Ocean Model. 6 (2004) 125–150.
[5] M. Iskandarani, D. Haidvogel, J. Levin, A three-dimensional spectral element model for the solution of the hydrostatic primitive equations, J. Comp. Phys. 186 (2003) 397–425.
[6] P. Roache, Computational Fluid Dynamics, Hermosa Publishing, Albequerque, NM, USA, 1976.
[7] H.A. Dijkstra, H. Öksüzŏglu, F.W. Wubs, E.F.F. Botta, A fully implicit model of the three-dimensional thermohaline ocean circulation, J. Comp. Phys. 173 (2001) 685–715.
[8] W. Weijer, H.A. Dijkstra, H. Oksuzoglu, F.W. Wubs, A.C. De Niet, A fully-implicit model of the global ocean circulation, J. Comp. Phys. 192 (2003) 452–470.
[9] A.C. de Niet, F.W. Wubs, A.D. Terwisscha van Scheltinga, H.A. Dijkstra, A tailored solver for the bifurcation analysis of ocean-climate models, J. Comput. Phys. 227 (2007) 654–679.
[10] S. Khatiwala, Fast spin up of ocean biogeochemical models using matrix-free Newton-Krylov, Ocean Model. 23 (2008) 121–129.
[11] X. Li, F. Primeau, A fast Newton-Krylov solver for seasonally varying global ocean biogeochemistry models, Ocean Model. 23 (2008) 13–20.
[12] E.Y. Kwon, F. Primeau, Optimization and sensitivity study of a biogeochemistry ocean model using an implicit solver and in situ phosphate data, Global Biogeochem. Cy. 20 (2006).
[13] T.M. Merlis, S. Khatiwala, Fast dynamical spin-up of ocean general circulation models using Newton-Krylov methods, Ocean Model. 21 (2008) 97–105.
[14] E. Bernsen, H.A. Dijkstra, F.W. Wubs, A method to reduce the spin-up time of ocean models, Ocean Model. 20 (2008) 380–392.
[15] E. Bernsen, H. Dijkstra, F. Wubs, Bifurcation analysis of the wind-driven ocean circulation with MOM4, Ocean Model. 30 (2009) 95–105.
[16] D. Knoll, D. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[17] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, 1996.
[18] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optimization 4 (1994) 393–422.
[19] V. Frayssé, L. Giraud, S. Gratton, J. Langou, A Set of GMRES Routines for Real and Complex Arithmetics on High Performance Computers, CERFACS, 2003. <http://www.cerfacs.fr/algor/reports/2003/TR_PA_03_03.pdf>.
[20] E.F.F. Botta, F.W. Wubs, MRILU: An effective algebraic multi-level ILU-preconditioner for sparse matrices, SIAM J. Matrix Anal. Appl. 20 (1999) 1007–1026.
[21] V. Frayssé, L. Giraud, S. Gratton, A set of flexible-GMRES routines for real and complex arithmetics, in: CERFACS, 1998.
[22] R. Smith, P. Gent, Reference Manual for the Parallel Ocean Program (POP), 2002. <http://climate.lanl.gov/Models/POP/POP_Reference.ps>.
[23] T.F. Coleman, B.S. Garbow, J.J. Moré, Software for estimating sparse jacobian matrices, ACM T. Math. Software 10 (1984) 329–345.
[24] T.F. Coleman, B.S. Garbow, J.J. Moré, Algorithm 618: Fortran subroutines for estimating sparse jacobian matrices, ACM T. Math. Software 10 (1984) 346–347.