Ocean Dynamics
DOI 10.1007/s10236-008-0138-5

Modeling ocean circulation on unstructured meshes: comparison of two horizontal discretizations

Sergey Danilov · Qiang Wang · Martin Losch · Dmitry Sidorenko · Jens Schröter

Received: 19 February 2008 / Accepted: 18 June 2008

Abstract Finite-element models on unstructured meshes are frequently formulated in terms of continuous linear elements, which suffer from pressure modes and require stabilization. Alternatively, horizontal velocities may be represented with linear nonconforming elements. While the latter formulation uses three times more degrees of freedom for the velocity, it does not support pressure modes. The effects of stabilization are estimated by comparing the performance of continuous linear and nonconforming versions of the finite-element ocean circulation model (FEOM) in two simple configurations: a Munk gyre and baroclinic turbulence in a zonally reentrant channel. It is shown that, outside the free slip boundary layers, the presence of stabilization does not lead to noticeable effects if its strength is kept within certain limits. In order to evaluate the performance of FEOM, the baroclinic turbulence test is repeated with the MIT general circulation model (MITgcm), which serves as a benchmark, and reasonable agreement between different model codes is found. The two versions of FEOM have a similar computational cost, but both are significantly slower (per node) than the regular-mesh MITgcm. The paper also provides a brief description of the implementation of the nonconforming version of FEOM.

Keywords Unstructured mesh ocean modeling · Finite-element method

Responsible Editor: Eric Deleersnijder

S. Danilov (✉) · Q. Wang · M. Losch · D. Sidorenko · J. Schröter
Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany
e-mail: Sergey.Danilov@awi.de

1 Introduction

Unstructured meshes suggest a number of conceptual advantages for ocean modeling such as variable resolution and continuous representation of coastlines (or a recent review, see Pain et al. 2005). The former makes nesting unnecessary and is of potential interest for designing configurations with regional focus imbedded into a coarse resolution global ocean. However, current practical usage of unstructured meshes in oceanography is mostly limited to tidal and coastal applications where the propagation speed dependence of surface gravity and barotropic Kelvin waves on fluid depth is effectively taken into account by using variable spatial resolution. Applications of unstructured meshes to simulating large-scale ocean circulation are not common, which is partly associated with their relatively low computational efficiency as compared to their structured grid counterparts. The continuing search for computationally efficient and robust algorithms explains the recent emergence of several models working on unstructured meshes (see, e.g., Casulli and Walters 2000; Chen et al. 2003; Danilov et al. 2004; Ford et al. 2004; Zhang et al. 2004; Walters 2005; Stuhne and Peltier 2006; Fringer et al. 2006; Wang et al. 2008a; Zhang and Baptista 2008; White et al. 2008a). These models differ in the discretization method (finite elements or finite volumes), discretization type (representation of variables on meshes), solution algorithm, and area of application (coastal or large-scale). Because of strong vertical stratification of the real ocean, most models are formulated for vertically aligned meshes, implying that only the surface mesh is unstructured.

The numerical efficiency and accuracy of the models based on the finite-element (FE) method are defined

by functional spaces (polynomial order of functions) selected to represent variables. Although high-order elements allow achieving high spatial accuracy, most practical approaches use low-order polynomials. In this way, the degrees of freedom are used to represent the complex geometry of the computational domain. QUODDY (Lynch et al. 1996), ADCIRC (Westerink et al. 1992), and MOG2D (Carrère and Lyard 2003), as well as models formulated in Danilov et al. (2004) and Wang et al. (2008a) [finite-element ocean circulation model (FEOM)], use a linear continuous representation for velocity and elevation, while the nonhydrostatic ICOM (Ford et al. 2004) works on quadrilateral surface meshes and uses continuous linear (CL) velocities and elementwise constant pressure. The approach by Walters (2005) is formulated with the so-called RT0 element and is close to the finite-volume approach of Casulli and Walters (2000). It uses elementwise constant pressure (elevation) and associates the normal components of velocity with the edges of elements.

The CL representation for velocity and elevation (P_1–P_1 discretization) is an obvious choice because it requires minimum memory storage and results in a reasonable number of operations for assembling the right-hand sides (RHS) of model equations. Since this representation uses the full vector of horizontal velocities handling the Coriolis operator, which requires special care with finite-volume, C-grid-type discretization, poses no particular problem here. Its notorious difficulty is, however, the spurious pressure (elevation) modes. These are eliminated by different stabilization techniques, such as the generalized wave continuity equation method (used by QUODDY, ADCIRC, MOG2D), Galerkin least squares method (Codina and Soto 1997; Danilov et al. 2004), or pressure (elevation) split method (Zienkiewicz and Taylor 2000; Codina and Zienkiewicz 2002; Wang et al. 2008a). All methods share the drawback that the horizontal velocity field satisfies a modified, as opposed to exact, vertically integrated continuity equation. The bias of numerical solutions caused by stabilization is difficult to assess, except for simple cases.

Recently, it was shown that using the so-called linear nonconforming (NC) representation of velocity (to be explained in Section 2.1) suggests a number of advantages both in numerical accuracy and computational efficiency (see Hanert et al. 2005; Le Roux et al. 2005). In particular, such discretization is free of pressure modes and does not require stabilization. It serves as a basis to a model by White et al. (2008a) that further demonstrates the usability of this approach.

The aim of this paper is to assess the effects of stabilization by comparing the performance of two discretizations of horizontal velocity, the CL and NC, as implemented in FEOM. We show that the influence of stabilization on flow dynamics is felt as numerical viscosity, leading to a noticeable difference in dynamics of boundary flows subject to free slip boundary conditions. However, the effects of stabilization can be made reasonably small in other situations by controlling the strength of stabilization. We also show that, despite the fact that NC elements suggest a number of simplifications to the numerical algorithm, they do not make this method cheaper in terms of CPU time. Thus, both methods can be recommended, yet one is advised to check that the CL discretization is not overstabilized. Our implementation of the NC code is different from that suggested by White et al. (2008a) in that we apply a pressure correction algorithm instead of introducing a barotropic mode.

An additional task is to carry out an elementary comparison of the performance of FEOM with the structured-grid finite-volume MIT general circulation model (MITgcm) (Marshall et al. 1997) in terms of both CPU time and characteristics of the simulated fields. The mesh used for this comparison is regular. It is triangulated for FEOM and used as it is by the MITgcm. We show that, in the current implementation, FEOM is approximately a factor of 10 slower than the MITgcm in the particular configuration we have used. This comparison is computer-architecture dependent, but the slowness factor already appears to be close to the performance limit of FE codes using CL or NC elements, as these codes simply need many more operations to assemble RHSs of equations than regular grid codes, leaving alone mass matrix inversions.

The CL implementation of FEOM is given by Wang et al. (2008a), and here, we only mention the details that are different from the NC implementation, which is briefly described in Section 2. Section 3 presents experiments with the Munk gyre aimed at evaluating the dependence of numerical viscosity on the stabilization parameter. In Section 4, we present experiments with baroclinic turbulence in a zonally re-entrant channel and compare levels of eddy kinetic energy achieved in both P_1 and P_1^{NC} cases with those of MITgcm. Conclusions and discussion are presented in the last section.

2 Model details

2.1 NC setup

Both NC and CL implementations use CL functions to represent elevation and tracers. The difference between them is in the discretization of horizontal ve-

locity. In the NC case, the horizontal velocities are expanded in basis functions that are products of NC linear functions $P_1^{\mathrm{NC}}(x, y)$ with CL functions $P_1(z)$. The NC linear functions are associated with the edges of triangles. They are equal to 1 at their edge and go linearly to -1 at the opposing node. CL functions are associated with nodes. They are equal to 1 at their node and go linearly to 0 at neighboring nodes. The essential advantage of NC linear functions is their orthogonality on triangles. Since the number of edges is approximately three times that of nodes, there is no nullspace of the discrete gradient operator—no spurious pressure modes, and no stabilization is needed. These two properties, (1) the diagonal mass matrix and (2) the absence of stabilization, suggest that the NC representation of velocity is a promising choice for formulating a model. On the downside is the increased number of degrees of freedom and extra computational load in computing momentum advection due to the discontinuity of these functions.

The property of orthogonality is strictly maintained on z-level meshes for which the integration over vertical and horizontal coordinates on elements are independent. In the case of generalized vertical coordinates, elements are no longer rectangular prisms and Jacobians of transform from parent to physical domains are functions of horizontal coordinates. This circumstance destroys the orthogonality property. In order to maintain the efficiency of the code, horizontal lumping or special quadrature rules (see White et al. 2008a) have to be applied in such cases, which partly contaminates the mathematical elegance of the approach. If elements are only gently deformed, the deviations from orthogonality are small and horizontal lumping remains a good compromise. Although the functionality of generalized vertical coordinates is supported in the code, here, we will only discuss the case of z-level grids for brevity.

The momentum equation is discretized using the second-order Adams–Bashforth method for the Coriolis term and the momentum advection. Applying the Adams–Bashforth method to horizontal viscosity terms is not recommended and can even increase instability, but it is convenient and seldom leads to problems. The Coriolis term can be treated semi-implicitly, but this can be beneficial only on coarse meshes and is not discussed here. The contribution from vertical viscosity can be optionally treated implicitly (when viscosity is large and CFL limiting may occur). The contribution from the sea surface height (ssh) is implicit to suppress inertia-gravity waves. For this reason, the method remains only first-order accurate in time. One can easily make it second-order if required at almost no cost (by taking ssh semiimplicitly), but this is not recommended for large-scale ocean applications. The time-discretized momentum and vertically integrated continuity equations are

$$\delta(\mathbf{u}^{n+1} - \mathbf{u}^n) + g\nabla\eta^{n+1} - \partial_z A_v \partial_z \mathbf{u}^{n+1} = \mathbf{R}^{n+1/2}, \tag{1}$$

$$\delta(\eta^{n+1} - \eta^n) + \nabla \cdot \int_{-H(x,y)}^{0} \mathbf{u}^{n+1} dz = 0. \tag{2}$$

Here, $\delta = 1/\Delta t$, where Δt is the time step, A_v is the vertical viscosity coefficient, g the acceleration due to gravity, $\mathbf{u}$ is the horizontal velocity, η is the elevation, and n marks time steps. The RHS of the vertically integrated continuity equation is set to zero for simplicity.

The RHS of Eq. 1 contains all terms of the momentum equation other than time derivative, surface pressure, and vertical viscosity. The Coriolis, pressure, and viscous terms are computed as

$$\mathbf{R}^{n+1/2} = -\nabla p_h^{n+1/2}/\rho_0 + \nabla A_h \nabla \mathbf{u}^n -$$

$$(3/2 + \varepsilon)(\mathbf{f} \times \mathbf{u} + (\mathbf{v}\nabla)\mathbf{u})^n + (1/2 + \varepsilon)(\mathbf{f} \times \mathbf{u} + (\mathbf{v}\nabla)\mathbf{u})^{n-1}$$

Here, ε is a small constant chosen to stabilize the second-order Adams–Bashforth scheme, p_h is the pressure due to the weight of the fluid counted from $z = 0$, ρ_0 is the reference density, $\mathbf{v} = (\mathbf{u}, w)$ is the full velocity, A_h the horizontal viscosity coefficient, and $\mathbf{f}$ the Coriolis parameter.

The ideology of solving the pair of Eqs. 1 and 2 is standard (pressure correction method) and is similar to that used in the CL setup of FEOM (Wang et al. 2008a) and in other models working with implicit free surface (e.g., MITgcm).

First, during the prediction step, one solves for $\mathbf{u}^*$

$$\delta(\mathbf{u}^* - \mathbf{u}^n) + \partial_z A_v \partial_z \mathbf{u}^* + g\nabla\eta^n = \mathbf{R}^{n+1/2}. \tag{3}$$

The predicted velocity is then corrected by solving

$$\delta(\mathbf{u}^{n+1} - \mathbf{u}^*) + g\nabla(\eta^{n+1} - \eta^n) = 0. \tag{4}$$

Formally, combining Eqs. 3 and 4, one does not recover the original Eq. 1. There is a small difference due to the omission of the viscous contribution from Eq. 4. This difference, however, does not destroy the time accuracy of the method (cf. with discussion in Ford et al. 2004).

Discretizing Eqs. 3 and 4, one gets the following matrix equations:

$$(\delta M + D)\mathbf{u}^* = \delta M \mathbf{u}^n - g\mathbf{G}\eta^n + \mathbf{R}^{n+1/2}, \tag{5}$$

and

$$(M\mathbf{u}^{n+1} - M\mathbf{u}^*) + g\Delta t \mathbf{G}(\eta^{n+1} - \eta^n) = 0. \tag{6}$$

Here, the notation used for continuous fields is preserved for their discrete counterparts because the meaning is clear from the context. The matrices introduced above are the mass matrix

$$M_{ij} = \int N_i N_j d\Omega,$$

the matrix of vertical viscosity

$$D_{ij} = \int A_v \partial_z N_i \partial_z N_j d\Omega,$$

and of the gradient operator

$$\mathbf{G}_{ij} = \int \mathbf{N}_i \nabla \overline{N}_j d\Omega.$$

Here, N_i and $\overline{N}_i$ are the basis functions used to represent velocity and elevation. Let us look at the structure of M and D matrices. Due to the orthogonality of the horizontal basis functions, these contain only links between vertically aligned nodes. This implies that the problem of matrix inversion is split into E_{2D} (the number of 2D edges) subproblems, each of which can be inverted effectively by the sweep algorithm. Since the number of edges is three times that of the nodes, this inversion is relatively expensive, yet not as expensive as applying iterative solvers to invert the stiffness matrices in the CL case when implicit vertical viscosity is used.

In situations when the vertical viscosity does not introduce CFL limitations for the selected time step Δt, it can be included into the $\mathbf{R}$ term. If, additionally, one lumps the mass matrix in the vertical direction, there is no longer any need for a matrix inversion, and a very effective numerical algorithm follows.

Expressing velocity from Eq. 6, one gets

$$\mathbf{u}^{n+1} = \mathbf{u}^* - g\Delta t M^{-1}\mathbf{G}(\eta^{n+1} - \eta^n). \tag{7}$$

Now, we *first* discretize the vertically integrated continuity Eq. 2 and then substitute Eq. 7 to obtain

$$\delta M_\eta \Delta\eta + g\Delta t \mathbf{G}^T M^{-1}\mathbf{G}\Delta\eta = \mathbf{G}^T\mathbf{u}^*. \tag{8}$$

Here, $\Delta\eta = \eta^{n+1} - \eta^n$, and M_η is the mass matrix of the elevation problem. This step is essentially different from its analog in the CL case where we first rearrange the continuous equations and then apply the FE discretization (see Section 2.2). In the NC case, the discretized vertically integrated continuity equation will be satisfied by $\mathbf{u}^{n+1}$ on completing the time step (solving Eq. 8 and updating the horizontal velocity via Eq. 6).

Assembling matrix $\mathbf{G}^T M^{-1}\mathbf{G}$ is tricky but feasible and has to be done only once during the initialization phase of the model run. If vertical lumping is applied to the velocity mass matrix, the assembly task is substantially simplified.

In summary, the solution algorithm goes through the following steps:

- Compute $\mathbf{u}^*$ from Eq. 5 by inverting $\delta M + D$. This requires solving E_{2D} (the number of surface edges) subsystems of equations for vertically aligned edges. If the mass matrix is vertically lumped and vertical viscosity is taken explicitly, this step becomes elementary.
- Compute elevation from Eq. 8 by inverting the matrix $\delta M_\eta + g\Delta t\mathbf{G}^T M^{-1}\mathbf{G}$. This is the matrix of size N_{2D} (the number of surface nodes), which is assembled outside the time stepping loop.
- Update the velocity according to Eq. 7. This step is elementary if the mass matrix is vertically lumped, and it is associated with the sweep algorithm otherwise.

The computation of vertical velocity and solving the tracer advection-diffusions equation follow the same ideology as in the CL approach and are not discussed here. In particular, the Taylor–Galerkin and FCT advection schemes are available. All physical parameterization options of the CL code are also supported. The code is MPI parallelized and uses PETSc to solve for elevation.

2.2 Brief summary of the CL approach

In order to remove the spurious pressure modes, an analog of Eqs. 5 and 6 is written as

$$(\delta M + D)\mathbf{u}^* = \delta M\mathbf{u}^n - g\gamma\mathbf{G}\eta^n + \mathbf{R}^{n+1/2}, \tag{9}$$

$$(M\mathbf{u}^{n+1} - M\mathbf{u}^*) + g\Delta t\mathbf{G}(\eta^{n+1} - \gamma\eta^n) = 0. \tag{10}$$

Here, the difference to the NC case is in adding a multiplier γ to the η^n term. The strength of stabilization turns out to be proportional to $(1-\gamma)$. In practical applications, $\gamma = 0.95 - 0.97$ works well, and in some cases, even values closer to 1 lead to a stable algorithm. The major difficulty in solving Eq. 9 is the matrix inversion (horizontal basis functions are not orthogonal on elements), and for numerical efficiency, the vertical diffusion is delegated to the RHS whenever possible. Inverting the mass matrix is then done iteratively, as explained in Wang et al. (2008a), without calling PETSc.

The other essential distinction from the NC case is that, in order to solve for the elevation, one does not use Eq. 10. Instead, one first expresses $\mathbf{u}^{n+1}$ from Eq. 4 modified by including γ and substitutes it into Eq. 2, and *then* discretizes the emerging equation. The difference to the NC algorithm is the replacement of operator $-\mathbf{G}^T M^{-1}\mathbf{G}$ by the Laplacian operator, which

does not support the pressure modes of operator $\mathbf{G}$. The price for this (necessary) modification is that $\mathbf{u}^{n+1}$ as found from Eq. 10 does not satisfy the vertically integrated continuity equation exactly. It is its unprojected counterpart from Eq. 4 that does. The latter is used to solve for vertical velocity and to advect tracers, while the former is used in the momentum equation. The difference between them is small (it is only due to reprojection on linear functions) but important for the consistency of the code. The need to keep two types of the horizontal velocity is the major conceptual disadvantage of the CL code. One of these velocities ($\mathbf{u}^{n+1}$) satisfies boundary conditions but does not exactly satisfy the discrete vertically integrated continuity equation. The other one, $\mathbf{u}^* - g\Delta t\nabla(\eta^{n+1} - \gamma\eta^n)$, on the contrary, does it, but it satisfies only weak impermeability boundary conditions. For this reason, one should expect main manifestations of effects from stabilization in boundary layers.

It should be remembered that all methods of stabilizations used in models employing $P_1 - P_1$ discretization have similar problems because the essence of stabilization is the regularization of the vertically integrated continuity equation. They all introduce the Laplacian operator (instead of or in addition to the true operator $-\mathbf{G}^T M^{-1}\mathbf{G}$) into the vertically integrated continuity equation, and in this way, have the same mathematical basis. The approach used by us has the advantage of explicitly providing the expression for the horizontal velocity, which ensures consistency with the vertical velocity equation.

The solution of the dynamical part follows the same three basic steps as in the NC case, but the prediction and correction steps require the inversion of the mass matrix or (mass + vertical viscosity) matrix. The inversion of the mass matrices can be done effectively and does not slow down the algorithm as much as the inversion of the (mass + vertical viscosity) matrix.

Applying the stabilization may, however, affect the dynamics. We address the question of the consequences of using stabilization by comparing the solutions from the code with stabilization (CL) and without it (NC).

2.3 Momentum advection in the NC code

Before describing the results of our numerical experiments, some remarks are necessary on the implementation of the momentum advection in the NC case. There are two points. The first one is on which form of the momentum advection term is preferable, and the second one is on how to properly treat discontinuities.

In order to guarantee momentum conservation, it seems reasonable to use the conservative form of the momentum advection $\nabla_3 \cdot (\mathbf{vu})$ where $\mathbf{v} = (\mathbf{u}, w)$ is full velocity and $\nabla_3 = (\nabla, \partial_z)$. Indeed, if it is projected on an appropriate (and for a while) differentiable function $\tilde{\mathbf{u}}$, and integrated by parts, the result is

$$\mathbf{A} = -\int \mathbf{v} \cdot (\mathbf{u} \cdot \nabla_3 \tilde{\mathbf{u}}) d\Omega,$$

with surface integrals over impermeable lateral walls set to zero. This form (with added flux penalties in the NC case because of discontinuity of basis functions) guarantees that the discretized momentum advection sums to zero. Indeed, the sum of test functions on an element is 1, which reduces the elemental part of the integral to zero. Also, the additional penalties always sum to zero in this case too.

The hidden inconsistency here is that, in writing the momentum advection as $\nabla_3(\mathbf{vu})$ instead of $(\mathbf{v} \cdot \nabla_3)\mathbf{u}$, one exploits the fact that the divergence of the full velocity is zero. This is true in the continuous case, but for the FE discretization, it holds only in the projection on a particular set of functions. This sense is not respected by $\mathbf{A}$, which implies that enforcing momentum conservation introduces noisy sinks and sources, and our experience is that the form $\mathbf{A}$ (with additional penalties) works only provided that the explicit viscosity is relatively high, and it remains prone to instabilities in flows with "rich" dynamics. It is worth mentioning that a similar difficulty with the tracer equations is avoided by respecting the consistency requirement on functional spaces used to represent tracers and vertical velocity (see White et al. 2008b; Wang et al. 2008a).

The nonconserving form $(\mathbf{v} \cdot \nabla_3)\mathbf{u}$ works stably and allows for an order of magnitude smaller explicit viscosity. We use this form in numerical experiments. One more stably working variant is to project the horizontal velocity on P_1 functional space to get $\mathbf{u}_c$, where the subscript "c" stands for "continuous," and write the momentum advection term as $(\mathbf{v} \cdot \nabla)\mathbf{u}_c$. The advantage of this form is that it is globally conservative ($\int (\mathbf{v} \cdot \nabla)\mathbf{u}_c d\Omega = 0$ because w is solved by projecting it on functions from the same space as $\mathbf{u}_c$) and that it does not require taking into account continuity penalties across vertical faces of elements. Its potential disadvantage is that a numerically efficient way of solving for $\mathbf{u}_c$ is achieved by lumping the mass matrix, which smoothes the momentum advection and thus introduces a mild scheme dissipation. We compare the performance of both advection schemes in Section 4.

We will now address the other question on how to correctly write additional penalties arising due to the discontinuous nature of NC elements. Writing the penalties is straightforward if equations are written in

the conservative form because fluxes are to be continuous across the boundaries of elements. This ideology cannot be applied to the nonconservative form of the momentum advection. A suitable method is to represent velocity as $\mathbf{u} = \sum_e \mathbf{u}_e \theta_e$ and introduce the same representation for test functions. Here, summation is over elements; $\mathbf{u}_e$ denotes velocity on element e and θ_e equals one inside element e and is zero outside it. With this representation, velocities and test functions are defined everywhere in the computational domain. On manipulating products of θ_e and delta-functions arising from the differentiation, one splits the form like **A** into a sum of integrals over interiors of elements and singular contributions from the vertical faces. For the nonconservative momentum advection form, the result is

$$\int \tilde{\mathbf{u}}(\mathbf{v}\cdot\nabla_3)\mathbf{u}d\Omega = \sum_e \int_e \tilde{\mathbf{u}}(\mathbf{v}\cdot\nabla_3)\mathbf{u}dS$$

$$- \sum_{\text{v.f.}} \int_{\text{v.f.}} <\tilde{\mathbf{u}}><\mathbf{un}>[\mathbf{u}]dS.$$

Here, subscript "v.f." stands for vertical faces between elements, **n** is the normal to the respective face, and the notations [] and <> of Hanert et al. (2005) for the difference and half sum are used. Such a form of singular terms is only valid for NC elements, and additional terms appear in the general case. It is easy to see that the momentum advection remains nonconservative because, on setting the test function to unity, all terms do not sum to zero. Yet, it produces a consistent approximation.

3 Influence of stabilization on representation of boundary layers

CL and NC setups are applied to simulate Munk gyre flow driven by the wind stress

$$\tau_x = -\tau_0 \cos(\pi y/L)$$

with $\tau_0 = 0.1$ N/m^2 in a rectangular box of 1,500 by 1,500 km on a β-plane $f = f_0 + \beta y$, where $f_0 = 10^{-4}$ s^{-1}, and $\beta = 2\times10^{-11}$ m^{-1} s^{-1} (L is the length of the box in the meridional direction and y is counted from the southern wall). The horizontal viscosity is set to A_h=540 m^2 s^{-1}. The thickness of the western boundary layer scales as $\delta_M = (A_h/\beta)^{1/3} = 30$ km. The real width of this layer (defined as the distance to zero crossing of the meridional velocity) is larger and depends on the boundary conditions. It is approximately $3\delta_M$ for the no-slip case. The Munk problem with its narrow boundary layers provides a strict test for the effect of (over) stabilization.

Two different horizontal resolutions of 10 and 5 km are used in the region of the western boundary. The resolution is reduced smoothly to 50 km in the interior of the domain. In the vertical, 10 unevenly spaced levels are used. There is no baroclinic forcing (no stratification), and momentum advection is off in order to compare model solutions with existing analytical solutions (see, for example, Pedlosky 1996). The experiments were run with time step of 1 h for 3 years. This time is still insufficient to reach a fully steady state, but deviations from equilibrium are already very small.

Two sets of experiments were carried out, one with no slip and the other with free slip boundary conditions. In both sets, CL runs were conducted with stabilization parameter $\gamma = 0.9$, 0.99, and 0.999. Upper panels of Fig. 1 show the comparison of model and analytical solutions for the no-slip case. It displays the meridional velocity profile drawn across the central latitude of the domain. There is no large difference between the CL and NC solutions in this case, and both discretizations lead to results that are very close to the analytical solution (the existing difference between model solutions and the analytical solution is due to the finite resolution). The influence of stabilization remains on a moderate level beginning from $\gamma = 0.9$, in agreement with our practice. The effects of stabilization are becoming noticeable below this value, and deviation of the velocity amplitude from the analytical solution can be in excess of 15% for $\gamma = 0.5$ (not shown).

However, the influence of stabilization can be much stronger in free-slip cases, as shown in the lower panels of Fig. 1. While the NC solution provides a very reasonable approach to the analytical solution, the CL solutions approach the NC results only at very low levels of stabilization. By comparing profiles of meridional velocity, it can easily be concluded that, except for the case of very low stabilization, there is extra friction against the western wall introduced by the presence of stabilization. Indeed, the part of the jet adjacent to the wall is retarded. This friction is small and does not distort the no-slip boundary layers (which are frictional by themselves). It broadens free-slip western boundary layers to accommodate for the gyre transport with reduced velocity amplitude. In free-slip cases, using $\gamma = 0.9$ produces results that are hardly acceptable.

The effect of stabilization becomes less pronounced as the mesh is refined but the horizontal viscosity coefficient is kept fixed. In such cases, boundary layers become better resolved and the same values of the stabilization parameter lead to smaller deviations from the analytical solutions. We observe this by comparing

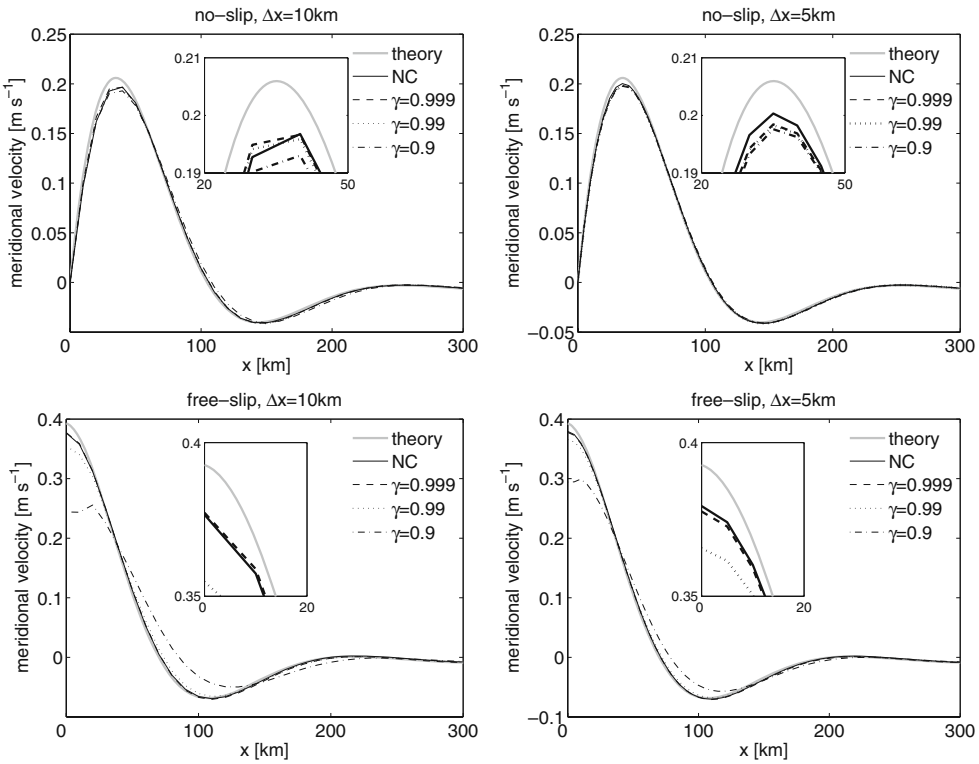

Fig. 1 The meridional velocity profile in the Munk gyre. The resolution is 10 km (*left panels*) and 5 km (*right panels*) in the western boundary layer. *Upper panels* correspond to the no-slip case and the *lower panels* correspond to the free-slip case. The influence of stabilization is small even for $\gamma = 0.9$ in the no-slip case but very strong in the free-slip case

results obtained on 10- and 5-km grids (left and right panels in Fig. 1). In the no-slip case, there is practically no difference between cases with three different γ, yet the NC solution is slightly more accurate and closer to the analytical solution. The agreement clearly improves in the free-slip case too, but $\gamma = 0.9$ still leads to significant errors. In the limit of very fine resolution, the effect of stabilization will be small. This can hardly lead to practical consequences, as in ocean modeling, one tries to reduce viscosity together with increasing resolution, thus making boundary layers thinner. So the practical recipe is rather to keep the stabilization as small as possible in order to minimize its effects.

4 Baroclinic turbulence in a channel

The question that still remains is what happens outside the boundary layers. In order to qualitatively answer, it we carry out a set of numerical experiments on baroclinic turbulence in a zonally reentrant channel. The turbulence statistics are rather sensitive to the explicit and scheme-implicit dissipation, and in this way, the effects of stabilization can also be diagnosed. This comparison is not so clean as in the Munk gyre case, as the properties of numerical baroclinic turbulence are affected by the momentum advection, which is different in CL and NC cases (tracer advection is implemented in the same way but can also be affected by different representations of the horizontal velocity). The experiments are conducted in a channel of 15 degrees in latitude and 40 degrees in zonal direction centered at 37.5° N at a resolution of about 16 km. In the vertical direction, the mesh contains 16 equally spaced levels going to the depth of 1,500 m. A linear equation of state is used with stratification being due to temperature only. The initial temperature distribution has horizontal and vertical gradients of 0.5×10^{-5} and 8.2×10^{-3} K/m, respectively. The stratification is maintained by relaxation to the initial profile in 1.5° northern and southern zones with relaxation coefficient dropping from 1/3 day^{-1} at the walls to zero outside the 1.5° zones.

Both (CL and NC) cases are run with the FCT advection scheme with explicit horizontal diffusivity of 50 m^2/s. For the NC case, runs with true NC and reprojected implementations of the momentum advection are carried out. The biharmonic viscosity of 2.5×10^{10} m^4/s is used, both versions are using (the same) explicit vertical viscosity and diffusion, and the mass matrix of the momentum equation is vertically lumped in the NC case for numerical efficiency.

In addition to comparing the performance of two versions of FEOM, similar computations were performed

with MITgcm (Marshall et al. 1997). The same mesh, initial temperature stratification, relaxation in southern and northern buffer zones, and physical parameters were used in these simulations. This serves two goals. The first one is to confront FEOM results with those of a more traditional model. The second one is the CPU time comparison. It is frequently mentioned that unstructured grid FE codes are slower (for the same amount of nodes) than structured grid codes, and here, we took the opportunity to measure the extent to which they are slower.

Each case is run for 3 years and eddy kinetic and eddy available potential energies are compared in the upper (kinetic) and lower (available potential) panels of Fig. 2. The eddy part of velocity and density fields are defined here as deviations from zonal mean values, and computation of the available potential energy follows the quasigeostrophic rules. As an illustration that the flow is in a turbulent regime, Fig. 3 shows a snapshot of the temperature field displaying a variety of eddy features characteristic of the well developed baroclinic turbulence. Since the channel is relatively wide, the characteristics of eddy flow are sensitive to the details of dissipation and, in particular, of the bottom drag. The latter is parameterized through the quadratic law with

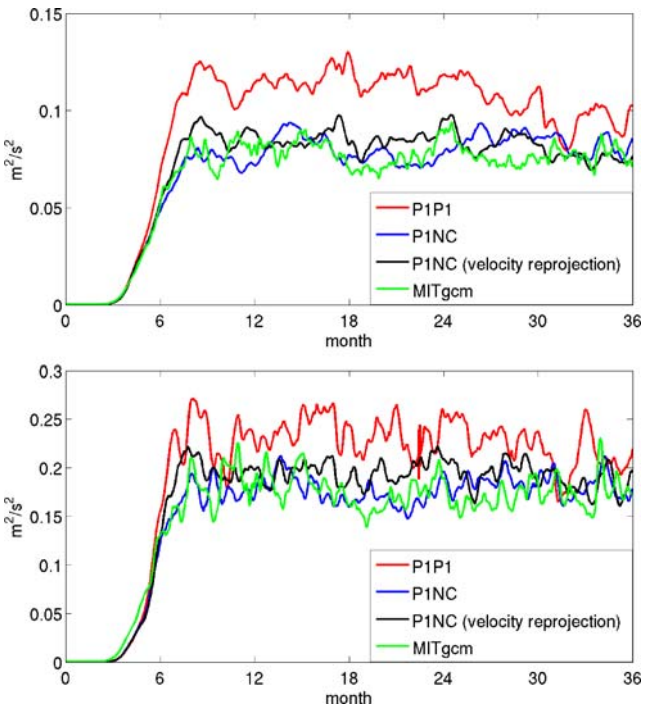

Fig. 2 Doubled eddy turbulent kinetic energy (*upper panel*) and eddy available potential energy (*lower panel*) in CL FEOM, NC FEOM (projected and true momentum advection), and MITgcm. The CL version simulates higher levels of energy compared to the NC version

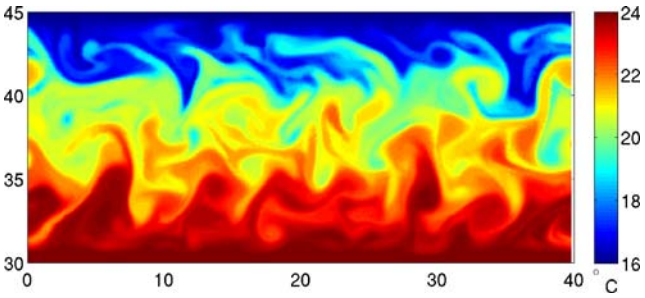

Fig. 3 A snapshot of temperature field at 100-m depth at the end of integration in the CL FEOM

the drag coefficient $C_d = 0.0025$. Removing the bottom drag leads to the appearance of large eddies occupying the entire channel in width.

The levels of eddy kinetic and eddy available potential energies are similar in CL and NC setups. In the NC case, both momentum advection schemes show similar results, which is counterintuitive, because the reprojected variant of momentum advection is expected to introduce some dissipation. The other counterintuitive fact is that CL levels of the eddy kinetic and available potential energies are slightly higher than the levels of the NC setup. They, however, show a tendency toward the NC results. The conclusion is that the stabilization present in the CL version does not damage the performance and does not lead to additional dissipation inside the domain.

The MITgcm results match very closely the results of both NC versions. Given the sensitivity of the eddy energy levels to the total dissipation (explicit and scheme-implicit) and recognizing the differences in discretization and implementation, the agreement between all simulations is surprisingly good.

5 Numerical efficiency

Despite the fact that no true matrix inversion is needed in the predictor and corrector steps of the NC setup in our experiments (explicit vertical viscosity and vertical lumping are used), the NC code is only marginally faster (less than 10%) than the CL code, which always does mass matrix inversions to solve for $\mathbf{u}^*$ and full velocity. One model year of the channel experiment (mesh of 0.4 million nodes) takes approximately 8 h on a single node (8 Power4 1.7-GHz processors) of IBM p655 in both implementations with $\Delta t = 15$ min.

Although this performance is already reasonably fast, it is still much slower than the performance of structured grid codes. The performance of the MITgcm on the same mesh (but without triangulation) and on

the same architecture is considerably faster. On a single processor, CL FEOM CPU time per time step compares to that of MITgcm as 6.5/0.35 s≈18, while on the full node (eight processors), the ratio is 0.85/0.14 s≈6. Note that the MITgcm code does not scale optimally because of the details of the architecture and the size of the mesh. This gives a slowness factor of about 10, which can perhaps be only slightly improved for FE $P_1 - P_1$ or $P_1^{nc} - P_1$ codes on unstructured triangular grids. The reason is, as mentioned in Section 1, a much higher number of operations needed to assemble the RHSs of equations on unstructured meshes, the need to invert mass matrices, and the relatively high cost of the FCT algorithm on unstructured meshes.

The essential part of the CPU load in our implementation of NC code is the assembly of the RHS of the momentum equation. The edge penalty part of the momentum advection term is not so expensive by itself, but it becomes more so in parallel implementation where the cycle over edges is to be repeated twice, first to collect combinations of velocities on edges (which are to be communicated between neighboring processors) and then to compute the contributions to the RHS. This splitting also requires additional storage. It turns out that the predictor/corrector steps together require comparable time in NC (the NC version is slightly faster) and CL setups, although in the latter case, mass matrix inversions are needed.

The analysis of CPU time per model time step shows that the horizontal velocity part takes about 40% of the total time, and 10% more are required to solve for elevation. This share depends on the tracer advection scheme and on how many tracers are advected. The FCT scheme is relatively expensive, and in the channel experiments, only temperature is advected (it takes 33%). Adding salinity would increase the weight of the tracer part to 50%, and it will dominate if more tracers are used.

The relationship between the CPU time of CL and NC cases can change in some special cases. For example, the CL code slows down in the case of free-slip boundary conditions (rotations are performed to local normal and tangential directions). In this case, one deals with matrices of doubled size (because velocity components become coupled) and uses a more sophisticated algorithm for matrix inversion. This leads to about 30% overhead. In the same vein, using local coordinate frames on elements (instead of spherical coordinates) in future setups will lead to more overhead in the CL case than in the NC case. Finally, the CL code is more expensive in the implicit vertical viscosity option (unless horizontal lumping is applied). These are the reasons why the NC setup is of further potential interest. It should, however, be mentioned that the CL setup is more robust with respect to the choice of viscosity and time step, so it remains a good choice too.

6 Conclusions

In this paper, we tried to estimate effects of stabilization of CL of FEOM by comparing model results to analytical solutions and to results of the NC setup, which does not require stabilization. The artifacts introduced by stabilization depend on the problem under consideration and can be strong close to free-slip boundaries. Except for this, there is no strong influence provided the stabilization strength is sufficiently small ($\gamma \geq 0.95$). The broadening of the western boundary current in the Munk gyre experiment remains small, and both eddy TKE and APE in the turbulent flow in a zonally reentrant channel are not affected by stabilization and reach approximately the same levels in both CL and NC cases. The overall conclusion from this comparison is that stabilization does not damage solutions in many cases but should be kept as small as possible in order to minimize its influence in boundary layers. Surely, the simple cases considered here do not exclude other possible effects that may emerge in longer simulations and under other circumstances, but they simply give an idea of consequences of using the stabilization. The potential of the CL code is illustrated by a study of overflows in Wang et al. (2008b), and other examples with both setups will be presented elsewhere.

The intercomparison of CL and NC discreizations as implemented in FEOM supports the currently forming opinion (Hanert et al. 2005; Le Roux et al. 2005) that NC discretization of horizontal velocities is a good alternative to CL horizontal velocities mostly used thus far in FE codes. The obvious advantage of the NC representation of the horizontal velocity is the absence of pressure modes, which leads to a cleaner pressure correction algorithm and consistent vertically integrated continuity equation. Orthogonality of NC basis functions in horizontal directions is another advantage, but taking it rigorously into account requires z-levels in the vertical direction. On generalized vertical grids with deformed elements, the orthogonality is destroyed and horizontal lumping (or special quadrature rules) is needed to get a numerically efficient algorithm. The robustness of different implementations in such situations remains to be explored.

In practice, the advantages of the NC elements do not lead to a faster code, as the number of operations needed to assemble RHSs is as high as or greater

than with the CL discretization. In perspective, working toward codes that do not need a longitude–latitude basis may change the situation in favor of the NC implementation.

The progress in the acceptance of unstructured grids for modeling large-scale ocean circulation is heavily dependent on the numerical efficiency of unstructured grid models, and the important task for future research is to establish the optimal approach. The current work is a step in this direction.

References

Carrère L, Lyard F (2003) Modeling the barotropic response of the global ocean to atmospheric wind and pressure forcing—comparisons with observations. Geophys Res Lett 30:1275

Cassuli V, Walters RA (2000) An unstructured grid, three-dimensional model based on the shallow water equations. Int J Numer Methods Fluids 32:331–348

Chen C, Liu H, Beardsley RC (2003) An unstructured grid, finite-volume, three-dimensional, primitive equations ocean model: applications to coastal ocean and estuaries. J Atmos Ocean Technol 20:159–186

Codina R, Soto O (1997) Finite element solution of the Stokes problem with dominating Coriolis force. Comput Methods Appl Mech Eng 142:215–234

Codina R, Zienkiewicz OC (2002) CBS versus GLS stabilization of the incompressible Navier-Stockes equations and the role of the time step as the stabilization parameter. Commun Numer Methods Eng 18:99–112

Danilov S, Kivman G, Schröter J (2004) A finite element ocean model: principles and evaluation. Ocean Model 6:125–150

Ford R, Pain CC, Piggott MD, Goddard AJH, de Oliveira CRE, Uplemby AP (2004) A non-hydrostatic finite-element model for three-fimensional stratified flows. Part I: model formulation. Mon Weather Rev 132:2832–2844

Fringer OB, Gerritsen M, Street RL (2006) An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. Ocean Model 14:139–173

Hanert E, Le Roux DY, Legat V, Deleersnijder E (2005) An efficient Eulerian finite element method for the shallow water equations. Ocean Model 10:115–136

Le Roux DY, Sène A, Rostand V, Hanert E (2005) On some spurious mode issues in shallow-water models using a linear algebra approach. Ocean Model 10:83–94

Lynch DR, Ip JTC, Naimie, CE, Werner FE (1996) Comprehensive coastal circulation model with application to the Gulf of Maine. Cont Shelf Res 16:875–906

Marshall J, Adcroft A, Hill C, Perelman L, Heisey C (1997) A finite-volume, incompressible Navier-Stokes model for studies of the ocean on parallel computers. J Geophys Res 102:5753–5766

Pain CC, Piggott MD, Goddard AJH, Fang F, Gorman GJ, Marshall DP, Eaton MD, Power PW, de Oliveira CRE (2005) Three-dimensional unstructured mesh ocean modelling. Ocean Model 10:5–33

Pedlosky J (1996) Ocean circulation theory. Springer, Berlin Heidelberg New York

Stuhne GR, Peltier WR (2006) A robust unstructured grid discretization for 3-dimensional hydrostatic flows in spherical geometry: a new numerical structure for ocean general circulation modeling. J Comput Phys 213:704–729

Walters RA (2005) Coastal ocean models: two useful finite-element methods. Cont Shelf Res 25:775–793

Wang Q, Danilov S, Schröter J (2008a) Finite element ocean circulation model based on triangular prismatic elements, with application in studying the effect of topography representation. J Geophys Res 113:C05015

Wang Q, Danilov S, Schröter J (2008b) Comparison of overflow simulations on different vertical grids using the finite element ocean circulation model. Ocean Model 20:313–335. doi:10.1016/j.ocemod.2007.10.005

Westerink JJ, Luettich RA, Blain CA, Scheffner NW (1992) ADCIRC: an advanced three-dimensional circulation model for shelves, coasts and estuaries; report 2: users manual for ADCIRC-2DDI. Contractors report to the US Army Corps of Engineers. US Army Corps of Engineers, Washington D.C.

White L, Deleersnijder E, Legat V (2008a) A three-dimensional unstructured mesh shallow-water model, with application to the flows around an island and in a wind driven, elongated basin. Ocean Model 22:26–47

White L, Legat V, Deleersnijder E (2008b) Tracer conservation for three-dimensional, finite element, free-surface, ocean modeling on moving prismatic meshes. Mon Weather Rev 136:420–442

Zhang Y, Baptista AM (2008) SELFE: A semi-implicit Eulerian-Lagrangian finite-element model for cross-scale ocean circulation. Ocean Model 21:71–96

Zhang YL, Baptista AM, Myers EP (2004) A cross-scale model for 3D baroclinic circulation in estuary-plume-shelf systems: I. Formulation and skill assessment. Cont Shelf Res 24: 2187–2214

Zienkiewicz OC, Taylor RL (2000) The finite element method. V. 3. Fluid dynamics. Butterworth–Heinemann, Oxford