*Dynamics of Atmospheres and Oceans*, 16 (1992) 379–403
Elsevier Science Publishers B.V., Amsterdam



# Computing the steady oceanic circulation using an optimization approach [1]

Eli Tziperman [a], William Carlisle Thacker [b] and Kirk Bryan [c]

[a] *Environmental Sciences and Energy Research, The Weizmann Institute of Science, Rehovot 76100, Israel*
[b] *Atlantic Oceanographic and Meteorological Laboratory, Miami, FL 33149, USA*
[c] *Geophysical Fluid Dynamics Laboratory, Princeton, NJ 08542, USA*

(Received 7 September 1990; revision accepted 2 August 1991)

ABSTRACT

Tziperman, E., Thacker, W.C. and Bryan, K., 1992. Computing the steady oceanic circulation using an optimization approach. Dyn. Atmos. Oceans, 16: 379–403.

The traditional method for computing the steady oceanic circulation has been by stepping an oceanic model forward in time until transients are damped by friction. An alternative method, which has the potential for being more economical is to minimize the sum of the squares of the residuals of the steady model equations. A variety of algorithms might be considered for computing the minimum; attention here is focused on preconditioned conjugate-gradient descent with the gradient computed using an adjoint model. The choice of variables, i.e. the preconditioning transformation used in the optimization process, is found to be critical to the efficiency of the method. An appropriate preconditioning transformation can be suggested by a heuristic analysis similar to that commonly used to test the stability of numerical models. The method is demonstrated within the context of the barotropic vorticity equation.

## 1. INTRODUCTION

Oceanic dynamics are characterized by a wide range of temporal scales. The high frequencies, which are associated with gravity waves, set the upper limit for the size of the time step for most numerical models; the low frequencies, which are associated with the slow mixing processes involved in the establishment of water-mass distributions, determine the number of time steps needed to spin the model up. Consequently, computing the steady oceanic response to steady forcing can be quite expensive. As the resolution of the model is increased, so is the computational expense.

[1] Contribution No. 22, Department of Environmental Sciences and Energy Research, The Weizmann Institute of Science, Rehovot, Israel.

In response to the need for accelerating the computation of the equilibrium circulation, Bryan (1984) suggested a two time-scale approach. When implemented, the value for the time step used in the momentum equations is essentially that set by the Courant condition for numerical stability, while the value used in the computation of salinity and temperature is much larger. Thus, his method is one of false transients, which should converge to a steady solution, if one does in fact exist. Data can also be used to accelerate the model spin-up (Sarmiento and Bryan, 1982). A restoring term forcing the temperature and salinity fields to be near the observed values is added to the model equations, thus guaranteeing that the density field is reasonable and allowing the momentum field to come quickly into adjustment. Both of these devices were used by Semtner and Chervin (1988) in spinning up a global, eddy-resolving model. The question we ask is whether there are other, more efficient methods that have not yet been considered. A variational counterpart to the nudging procedure of Sarmiento and Bryan has been presented in a recent paper (Tziperman and Thacker, 1989). Here we would like to point out that, in the absence of data, this variational approach can also yield the steady model circulation. The computational strategy is to define a function of the model unknowns that has a minimum when these unknowns exactly satisfy the steady model equations, and then to seek the minimum of this function (termed the objective function or cost function). Such a computation of the steady circulation would not be characterized by transients, true or false, as there would be no time-stepping; instead, there would be a converging sequence of approximations to the steady circulation produced by an iterative optimization algorithm. (It should be noted that some iterative methods are closely related to time stepping, but in our case there is no reason to expect the converging sequence of iterative solutions to behave like a time history of the corresponding time-dependent model that is stepped to equilibrium.)

Before proceeding with the discussion of this method, it is useful first to pause and consider whether such a computation would really be appropriate. A principal concern is that the steady state might not exist. For example, Cox (1987) and Semtner and Chervin (1988) find transient eddies in response to steady forcing. If an optimization procedure would produce a solution to the steady model equations, what would it mean? Would it be stable, i.e. when used as initial conditions for the steady model, would it evolve into something else? In practice the steady state would never be found exactly; if the solution is unstable, the small errors due to incomplete convergence would be amplified. When used as initial conditions for a time-dependent simulation, would an unstable steady state quickly produce eddies and resemble what is obtained by a time-stepping spin-up? Another possibility is that no steady state exists, stable or unstable. If this is the

case, what would the computations yield? Since the function to be minimized is defined as the sum of the squares of the residuals of the steady model equations, the minimum should be the best approximation to a steady state in a least-squares sense. Would such a solution look anything like that found by spinning the model up by time stepping? As the model is non-linear, another concern is whether the solution is unique. There may be several stable steady states, i.e. distinct minima of the cost function that might also be found by time stepping from different initial conditions; these different solutions might correspond to physically plausible steady circulation modes of the ocean that might be realized if circumstances could provide the necessary sequence of forcing.

Although these questions are interesting and worth pursuing, they will not be explored here. Still, the optimization approach should provide the means to answer them. Another way of looking at the steady circulation is as a long-term temporal average of the actual time-varying eddy-containing circulation. By properly parameterizing the Reynolds fluxes accounting for the transport by the ignored time-varying eddies, it should be possible to define a model with a statistically meaningful steady state. Perhaps studies with eddy-resolving models will be helpful in determining these parameterizations. Once such parameterizations are formulated, the optimization approach suggested here should be considered for computing the statistically steady circulation.

The variational approach to computing the steady circulation will be discussed here within the context of a simple barotropic vorticity-equation model. When compared with a fully three-dimensional, baroclinic, primitive-equation model, it is obvious that many challenging complications are ignored, so generalizing the results found here will not be trivial. Nevertheless, the transparency in understanding how the computational method works, which is obtained within the simpler context, more than offsets these disadvantages.

An efficient variational computation of the steady state requires the careful selection of the algorithm used to seek the minimum. Most optimization algorithms can be classified into three categories: (1) those using only the values of the objective function; (2) those using both values and gradients; and (3) those using values, gradients, and the Hessian matrix (the matrix of second partial derivatives of the objective function). Owing to the large number of unknows, algorithms in the first category can be eliminated as being too slow. Also due to the large number of unknowns, those in the third category might also be eliminated as requiring too much storage, although they might actually be feasible if sparse-matrix techniques can be used. This leaves the second category, which comprises conjugate-gradient methods. Each iteration of a conjugate-gradient method

starts with approximate values for the model inputs, (e.g. stream function); the model code is used to evaluate the objective function and an adjoint-model code to compute its gradient; next, a descent direction is determined from the gradient together with information obtained in previous iterations; finally, an approximation to the minimum along that direction is found, which provides a new set of inputs that reduce the value of the objective function.

For conjugate-gradient methods to be efficient (i.e. to require few iterations), it is important that the problem be well conditioned. In other words, it is important to take care in choosing variables in which the optimization problem is posed. For the barotropic vorticity-equation example, one could search for the optimal stream function or for the optimal vorticity field; having either, we can compute the other. The optimization variables need not have any particular physical significance; in fact, the physically meaningless variables for which the computations are most efficient, might be related to familiar physical variables by a complicated preconditioning transformation. In the examples discussed below, the preconditioning transformations are restricted to those defined by the $\nabla^2$ and $\nabla^4$ operators acting on the stream function $\psi$. The choise of optimization variables is crucial for the computation to be efficient. As preconditioning is an important aspect of any optimization problem, in particular the variational approach to data assimilation by fitting dynamical models to observations (Thacker and Long, 1988; Tziperman and Thacker, 1989), our purpose here is to gain some intuitive understanding of the way preconditioning works, at least within the context of computing the steady circulation.

In Section 2, using a simple model based on the barotropic vorticity equation, we formulate the calculation of the steady oceanic circulation as an optimization problem; in Section 3 we discuss preconditioning and its relationship to the temporal scales of the unsteady model and to the spectrum of eigenvalues of the Hessian matrix; specific computational examples with the simple model are given in Section 4; and we conclude in Section 5.

## 2. OPTIMIZATION FRAMEWORK FOR SOLVING THE STEADY VORTICITY EQUATION

The evolution of the model ocean is governed by the barotropic vorticity equation on a $\beta$-plane:

$$\frac{\partial \zeta}{\partial t} + \beta \frac{\partial \psi}{\partial x} + J(\psi, \zeta) + \kappa\zeta - \epsilon\nabla^2\zeta = \chi \tag{1a}$$

where the vorticity $\zeta$ is related to the stream function $\psi$ via a Poisson equation:

$$\nabla^2\psi = \zeta \tag{1b}$$

The circulation is forced by the curl of the wind stress, denoted here by $\chi$. The Jacobian determinant $J(\psi, \zeta) = \partial(\psi, \zeta)/\partial(x, y)$ accounts for advection of vorticity by the oceanic flow. Two types of friction are incorporated into the model, Newtonian bottom friction characterized by the coefficient $\kappa$ and horizontal eddy-viscous friction with coefficient $\epsilon$. Both coefficients are assumed to be constant, independent of time $t$ and of location $(x, y)$. For simplicity, the ocean basin is taken to be square and to have uniform depth. To satisfy the no-normal-flow and no-slip boundary conditions, both the stream function and the vorticity must vanish at the boundaries.

Given values for the initial vorticity field, the initial stream function can be found by solving eqn. (1b). Then, the vorticity field can be advanced in time using eqn. (1a). For computational purposes, partial derivatives are replaced by finite differences. A forward temporal difference is used along with Arakawa's (1966) conservative form of the Jacobian determinant, the usual five-point Laplacian, and centered differences for the $\beta$-term; details of the finite-difference equations are readily available (Tziperman and Thacker, 1989) and will not be repeated here.

For the circulation to be steady, the temporal derivative of the vorticity in eqn. (1a) must vanish; in the discrete model this is accomplished by requiring that there be no change from one time level to the next. The usual method for computing the steady state is simply to march the model forward under steady forcing until friction damps out transients. Alternatively, consider solving the system of equations obtained by omitting the temporal differences from the discrete model. For notational convenience they are presented in terms of partial derivatives, but should be interpreted as finite differences:

$$R = \beta\frac{\partial\psi}{\partial x} + J(\psi, \nabla^2\psi) + \kappa\nabla^2\psi - \epsilon\nabla^4\psi - \chi = 0 \tag{2}$$

This is a large system of coupled, non-linear, algebraic equations, and a variety of techniques for computing the solution might be tried; some useful references are the textbooks of Gill et al. (1981), Dennis and Schnabel (1983), Strang (1986) and Fletcher (1987).

One possibility that might be considered is Newton's method. In solving a non-linear system of equations, this method requires that a sequence of linear systems must be solved. These linear systems are obtained by linearizing the equations about a guess for the steady state, i.e. they

comprise equations for perturbations $\rho$ about the guess stream function $\psi_0$:

$$L(\psi_0)\rho = -R_0 \tag{3}$$

where the linear operator $L(\psi)$ (with coefficients depending on $\psi$) is:

$$L(\psi)\rho = \beta\frac{\partial\rho}{\partial x} + J(\psi, \nabla^2\rho) + J(\rho, \nabla^2\psi) + \kappa\nabla^2\rho - \epsilon\nabla^4\rho = 0 \tag{3a}$$

and where $R_0$ is the residual (eqn. (2)) evaluated for $\psi_0$. The solution $\rho$ of the linear system provides a correction to the guess $\psi_0$, and the improved guess can then be used to define a new linear system for the next correction. Unfortunately, the large linear system is not symmetric, so it is difficult to solve. Moro (1988) has used a variant of Newton's method to compute steady solutions for a similar quasi-geostrophic model, but he did not address the question of relative computational efficiency.

The approach considered here, is to formulate the computation of the steady state as an optimization problem. One way of doing this is to minimize the sums of the squares of the residuals of the steady model equations:

$$\mathcal{J} = \tfrac{1}{2}\int R^2 \, dx \, dy \tag{4}$$

where $R$ is defined in eqn. (2). A stream function for which $\mathcal{J}_{\min} = 0$ satisfies the steady model equations; if no steady solution exists, $\mathcal{J}_{\min} > 0$. The computational problem now becomes one of finding the minimum of $\mathcal{J}$, where the gradient of $\mathcal{J}$ must vanish:

$$g = \frac{\delta\mathcal{J}}{\delta\psi} = 0 \tag{5a}$$

The functional derivatives should be interpreted as partial derivatives with respect to the discrete variables representing the stream function at the interior grid points; boundary conditions are known, so they do not contribute to the gradient. It is not difficult to carry out the variations to obtain an expression for the gradient:

$$g = L^*(\psi)R \tag{5b}$$

where

$$L^*(\psi)R = -\beta\frac{\partial R}{\partial x} - \nabla^2 J(\psi, R) - J(R, \nabla^2\psi) + \kappa\nabla^2 R - \epsilon\nabla^4 R \tag{5c}$$

(Note that $L^*(\psi)$ is the adjoint of $L(\psi)$.) The finite-difference expressions for the partial derivatives in eqn. (5c) are determined by the finite-difference forms used in the model; it is not difficult to show that Arakawa's

Jacobian and centered differences in eqn. (2) require that Arakawa's Jacobian and centered differences also be used in eqn. (5c) (Tziperman and Thacker, 1989). The boundary conditions of the model constrain the variations of $\mathcal{J}$ at the boundaries, determining the form of the operators in eqn. (5c) at the boundaries, i.e. the boundary conditions for eqns. (5a) and (5b). Once the physical problem is formulated and the boundary conditions specified, however, the form of the operator eqn. (5c) is uniquely determined and quite easily derived. Now we are faced with solving a large, symmetric system of coupled, non-linear, algebraic equations (eqns. (5)) for $\psi$.

Newton's method might be considered for solving eqns. (5). It would require at each iteration an explicit representation of the Hessian matrix of the cost function $\mathcal{J}$:

$$G(\psi) = \frac{\delta^2 \mathcal{J}}{\delta \psi^2} \tag{6a}$$

At each Newton iteration, the following linearized system is solved to compute the correction $\rho_k$ to the previous approximation $\psi_{k-1}$ for the steady-state solution:

$$G(\psi_{k-1})\rho_k = -g_{k-1} \tag{6b}$$

where the subscript indexes the Newton iteration. (If the model were linear, the cost function would be quadratic, eqn. (5) would be a linear system, and only one Newton iteration would be needed.) Although determining the Hessian could be tedious for a three-dimensional, baroclinic ocean model, it is not too difficult in this case. The Hessian at $\psi$ acting on $\rho$ is (see Appendix):

$$G(\psi)\rho = L^*(\psi)L(\psi)\rho - J(R, \nabla^2\rho) - \nabla^2 J(\rho, R) \tag{6c}$$

where $R$ is the residual eqn. (2) evaluated for $\psi$, $L$ is the linearized model operator, $L^*$ is its adjoint, and $J$ is the Arakawa Jacobian. The Hessian of $\mathcal{J}$ is sparse, so it wouldn't require an overwhelming amount of storage; in fact, multiplication by the Hessian matrix could be coded like a finite-difference model involving higher derivatives.

If the terms involving the residual $R$ in eqn. (6c) are neglected, the Hessian is the product of the linearized model operator with its adjoint. These terms are due to the non-linearity of the model and should be small when the flow is nearly linear. Because they depend on the residual, they also vanish at the minimum of $\mathcal{J}$. When these terms are neglected, the Hessian is simply the linearized model operator times its adjoint, and its eigenvalues are, therefore, the squares of the moduli of the complex eigenvalues of the linearized model operator. Consequently, the condition

number of the optimization problem is the square of that of the original problem, and the equations to be solved in the optimization approach can be expected to be much more ill conditioned than the original equations.

To compute each Newton iteration to machine accuracy would be a time-consuming task. A better idea might be to solve for the correction only approximately and then move on to the next linearization. This is essentially what the conjugate-gradient algorithm does. Conjugate-gradient descent can also be thought of as an improvement over the method of steepest descent. For steepest descent, the new guess is obtained by moving in the steepest downhill direction from the previous guess:

$$\rho_k = -\alpha_{k-1} g_{k-1} \tag{7a}$$

where $\alpha_{k-1}$ is determined by searching for the minimum along the direction determined by the gradient. If the constant-cost contours are circles, then steepest descent converges in a single iteration, but if they are significantly non-circular, convergence will be extremely slow, since the steepest downhill direction is not in the direction of the minimum. The improvement offered by conjugate-gradient descent is in using a direction $d_{k-1}$ that takes into account previous descent directions:

$$\rho_k = -\alpha_{k-1} d_{k-1} \tag{7b}$$

If the cost function is quadratic, then, neglecting round-off errors, the conjugate-gradient descent algorithm is guaranteed to converge to the minimum in as many steps as there are unknowns to be found, with the best performance when cost contours in a large subspace are nearly circular. For a quadratic cost function the conjugate-gradient descent direction is:

$$d_k = -g_k + \frac{g_k^{\mathrm{T}} g_k}{g_{k-1}^{\mathrm{T}} g_{k-1}} d_{k-1} \tag{7c}$$

with $d_0 = -g_0$. If $\mathcal{J}$ is not quadratic, due to model non-linearities, then eqn. (7b) can still be used to improve the guess for the solution, but some generalization of eqn. (7c) might be preferred for computing the descent direction:

$$d_k = -B_k g_k \tag{7d}$$

where $B_k$ is an approximation to the inverse of the Hessian matrix of $\mathcal{J}$ built from the previous few directions and gradients. (If it were the inverse of the Hessian, it would be optimal in the sense that a single conjugate-gradient step would be as good as a Newton step). The advantage is that the Hessian is never needed, only the gradient vectors, which can be computed from eqn. (5b) or equivalently using an adjoint model (Thacker,

1989; Tziperman and Thacker, 1989). The improvement at each step is not expected to be as good as for Newton's method, but the amount of work per step is certainly much less. The performance of conjugate-gradient descent can be improved by posing the problem in terms of variables for which the cost contours are nearly circular, i.e. by preconditioning, which is closely related to the question of choosing the best descent direction. Computations presented below have been made using a conjugate-gradient routine from the NAG library (Numerical Algorithms Group (NAG), 1984).

## 3. PRECONDITIONING

If convergence is slow when time stepping to the steady state, then, owing to the condition-number considerations above, i.e. to the fact that the Hessian is approximately the square of the model operator, optimization with conjugate-gradient descent can be expected to converge slowly also. The reason for slow convergence of the conjugate-gradient descent is that the cost function is more sensitive to some variables than to others. Geometrically, this amounts to the cost surface being too flat in some directions of the parameter space. One way to get around this problem is by changing variables, so that in terms of the new variables the constant-cost contours are more nearly circular; this is referred to as preconditioning. If $M^{-1}\boldsymbol{\psi}$ is the vector of new variables defined by a transformation matrix $M$, then the transformed gradient vector is $M^{\mathrm{T}}g$ and the transformed Hessian matrix is $M^{\mathrm{T}}GM$. If the transformed Hessian is the identity matrix, then a single conjugate-gradient (steepest-descent) step is equivalent to a Newton step; but the amount of work is exactly the same as solving the original linearized problem, since $M$ would be the inverse of the linearized model operator. The idea behind preconditioning is to find some middle ground, so that the additional work required for accelerating convergence is more than offset by the savings in the number of iterations. This is a rather vague objective, and to accomplish it seems to be more of an art than a science. Some examples will be discussed below. Hopefully, the ingredients of the art will be made clear enough that they can be applied to other problems, in particular to that of computing the steady state for large oceanic circulation models.

The eccentricity of the constant-cost contours is indicated by the eigenvalues of the Hessian matrix (Thacker, 1989) so the eigenvalues can be helpful in suggesting a preconditioning transformation. Although the eigenvalue spectrum is unknown, it is possible to get some idea what it might be like. The real, positive eigenvalues of the Hessian are the squares of the moduli of the complex eigenvalues of the linearized model operator, which can be identified with the complex frequencies of the time-dependent

model. Thus the rate of convergence is tied to the disparity of the temporal scales that are described by the model. Focus attention first on the complex eigenvalues; they should already be familiar from their role in determining the size of the largest stable time step for the unsteady model (Vichnevetsky and Bowles, 1982). Using the same sort of reasoning involved in estimating the Courant condition, i.e. by assuming that all coefficients are constant even though they really are not approximate eigenvalues can be obtained, which correspond to approximate eigenvectors that are sinusoids. Taking the perturbations $\rho$ to be proportional to $\exp[i(k_x x + k_y y)]$, the eigenvalues of the linearized model operator defined in eqn. (3a) are given by:

$$\lambda(k_x, k_y) = i\beta k_x - (iUk_x + iVk_y)(k_x^2 + k_y^2) - \kappa(k_x^2 + k_y^2) - \epsilon(k_x^2 + k_y^2)^2 \tag{8a}$$

where for a square basin of length and width $L = N\Delta$, $\Delta$ being the grid spacing, $(k_x, k_y) = \pi(l, m)/L$, $l, m = 1, \ldots, N-1$. The spatially varying velocity components $-\partial\psi_0/\partial y$ and $\partial\psi_0/\partial x$ are estimated by the constants $U$ and $V$, which might not be a particularly good approximation, but it is the best that can be done without much more work. Similarly, from eqn. (6c), if the contributions from the non-linear-correction terms are neglected (they would vanish at the minimum of $\mathcal{J}$ or for a spatially constant residual), the eigenvalues of the Hessian might be estimated by:

$$\Lambda(k_x, k_y) = \left[\beta k_x - (Uk_x + Vk_y)(k_x^2 + k_y^2)\right]^2 + \left[\kappa(k_x^2 + k_y^2) + \epsilon(k_x^2 + k_y^2)^2\right]^2 \tag{8b}$$

In order to account for the effects of truncation errors on the eigenvalue spectrum, in the place of $k_x$ and $k_y$ there should actually be trigonometric functions of $k_x\Delta$ and $k_y\Delta$ resulting from the action of finite-difference operators (instead of partial-differential operators) on the complex exponential eigenfunctions:

$$\begin{aligned}\Lambda(k_x, k_y) = \big[&\beta \sin(k_x\Delta) - 2(U \sin k_x\Delta + V \sin k_y\Delta) \\ &(\cos k_x\Delta + \cos k_y\Delta - 2)\Delta^{-2}\big]^2\Delta^{-2} \\ &+\big[2\kappa(\cos k_x\Delta + \cos k_y\Delta - 2) \\ &+4\epsilon(\cos k_x\Delta + \cos k_y\Delta - 2)^2\Delta^{-2}\big]^2\Delta^{-4}\end{aligned} \tag{8c}$$

The $\beta$-term should dominate these expressions for the eigenvalues for large scales ($x \approx L$), but depending upon the model resolution and the

magnitude of the friction coefficients, this will probably not be the case for smaller scales ($x \approx \Delta$). Still, it is clear that the small scales will be responsible for the large eigenvalues and the large scales for the small eigenvalues. In other words, large-scale features are harder to compute than small-scale features on a high-resolution grid. This is exactly the same problem that plagues the Poisson equation. For the Poisson equation, the eigenvalues are proportional to $k_x^2 + k_y^2$, so the condition number is $k_{\min}^2/k_{\max}^2 = N^2$, where $N$ characterizes the number of points across the grid. For the optimization problem addressed here, depending on the values of the model's parameters, the condition number here might be more like $N^4$ or even as large as $N^8$. The problem for the Poisson equation is not as bad, and fast algorithms for solving it already exist, which suggests that the Poisson equation might provide an effective preconditioning transformation.

Suppose that the model had been formulated in terms of vorticity rather than stream function. Then, if $\sigma = \nabla^2\rho$ is the correction to the guess for the vorticity field, eqn. (3) would be:

$$\beta \frac{\partial \nabla^{-2}\sigma}{\partial x} + J(\psi_0, \sigma) + J\left(\nabla^{-2}\sigma, \nabla^2\psi_0\right) + \kappa\sigma - \epsilon\nabla^2\sigma = -R_0 \tag{9a}$$

and the transformed gradient would be $\nabla^{-2}g$ so the eigenvalues of the Hessian with respect to these variables would be:

$$\Lambda(k_x, k_y) = \left[\beta k_x\left(k_x^2 + k_y^2\right)^{-1} - \left(k_x U + k_y V\right)\right]^2 + \left[\kappa + \epsilon\left(k_x^2 + k_y^2\right)\right]^2 \tag{9b}$$

Similarly, if the optimization were cast in terms of $\nabla^2\zeta = \nabla^4\psi$, then the eigenvalues Hessian for that transformation would be:

$$\Lambda(k_x, k_y) = \left[\beta k_x\left(k_x^2 + k_y^2\right)^{-2} - \left(k_x U + k_y V\right)\left(k_x^2 + k_y^2\right)^{-1}\right]^2 + \left[\kappa\left(k_x^2 + k_y^2\right)^{-1} + \epsilon\right]^2 \tag{10}$$

Clearly, the net effect has been to divide each term by $(k_x^2 + k_y^2)^2$ in the first case and by $(k_x^2 + k_y^2)^4$ in the second. Consequently, to a different extent for the two cases, the domination by the small scales should be diminished and the influence of the large scales should be enhanced. Depending on the values of the model parameters, it could be possible that either or both of these transformations could result in a transformed Hessian with a smaller condition number.

To give some idea of the effect of these transformations the approximate spectra have been computed for a case in which the non-dimensional model parameters have the following values: $\kappa/L\beta = 0.05$, $\epsilon/L^3\beta^2 = 0.0001$

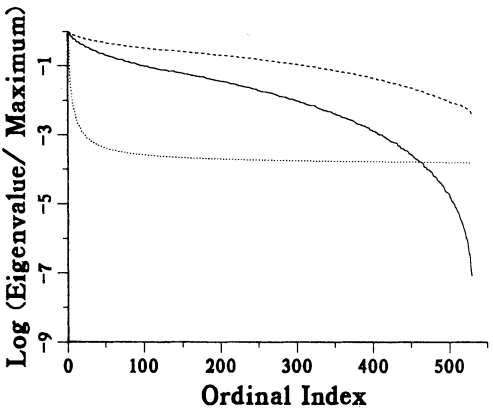


Fig. 1. Eigenvalue spectra estimated using eqns. (8b), (9b) and (10) with $\kappa/\beta L = 0.05$, $\epsilon/\beta L^3 = 0.0001$, and $U = V = 0$. Full line, dash and dot curves correspond, respectively, to $\psi$, $\zeta$, and $\nabla^2\zeta$ as the optimization variables and to the identity operator, $\nabla^{-2}$, and $\nabla^{-4}$ as the preconditioning operators. The eigenvalues are indexed in order of their magnitudes and normalized by dividing by the largest, so that all three curves can be conveniently displayed on a single plot.

and $U/L^2\beta = V/L^2\beta = 0.0001$. The non-dimensional wavenumbers have the values: $k_x L = l\pi$ and $k_y L = m\pi$, $l, m = 1 \ldots 23$, corresponding to a computational grid of $25 \times 25$ points. To show the three spectra conveniently on the same plot (Fig. 1) the eigenvalues have been normalized by dividing by the largest and arranged in descending order. The condition number is the reciprocal of the smallest normalized eigenvalue; the smallest condition number results from the preconditioning transformation from $\psi$ to $\zeta$ (eqn. (9b)) and the largest from transforming to $\nabla^2\zeta$ (egn. (10)). Figure 2 shows the corresponding results when the horizontal friction coefficient was set to be 100 times larger than in the standard case of Fig. 1. In this case the best conditioning number (flattest spectrum) is that obtained when using the Laplacian of the vorticity as the control variable. We shall return to this case in the numerical examples given below, confirm the conclusion concerning the choice of the best preconditioning for this parameter range, and explain this observation using the above analytic expressions for the eigenvalue spectrum (see end of Section 4).

More information about the rate of convergence is provided by the shape of the eigenvalue spectrum; the more clustered the eigenvalues, the better. However, the results of this approximate analysis on the clustering of approximate eigenvalues is probably less reliable than those concerning the condition numbers.

The price that must be paid for faster convergence associated with the preconditioning transformation from $\psi$ to $\zeta$ is that two Poisson equations

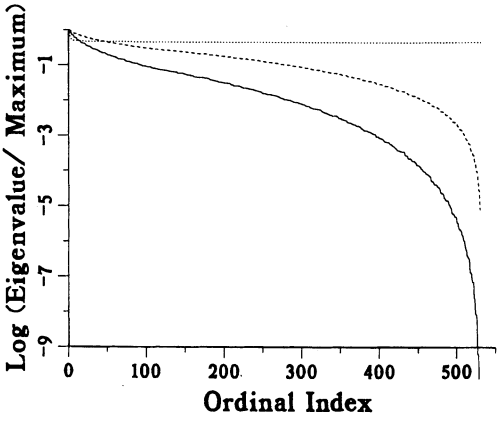


Fig. 2. As in Fig. 1, but using a horizontal friction coefficient that is 100 times larger: $\epsilon/\beta L^3 = 0.01$. Full line, dash and dot curves correspond to the same cases as in Fig. 1.

must be solved for each iteration, one to evaluate the stream function perturbation in order to compute the residual and the other to evaluate the transformed gradient. As long as the additional cost of the Laplacian inversions is not too high, this transformation is worthwhile. Fortunately, this is the case; we use a fast multigrid Poisson solver, which is described by Brandt (1984). (That same report suggests that multigrid methods might be more efficient than conjugate-gradient descent for computing the steady state).

## 4. COMPUTATIONAL EXAMPLES

The computational examples presented here illustrate the potential of the optimization approach for computing the steady state. In every case steady flow is computed for a square basin of depth $D$ and breadth $L$ represented by a $25 \times 25$ grid, and for the particular wind forcing $\chi = -A \sin(\pi x/L) \sin(\pi y/L)$. The examples are determined by the choice of values of the non-dimensional friction parameters $\kappa' = \kappa/\beta L$ and $\epsilon' = \epsilon/\beta L^3$ and by the choice of the preconditioning transformation. The value of the Rossby number $A/\beta^2 L^2$ is always taken as 0.1.

Computations were made using a slightly modified version of a code originally intended for fitting the time-dependent vorticity-equation model to model-generated observations (Tziperman and Thacker, 1989). That code sought the minimum of a cost function with two types of terms: those measuring the difference between the observations and their model counterparts, which have been omitted here, and those measuring the departure of the model from the steady state. The part that has been retained, i.e. the

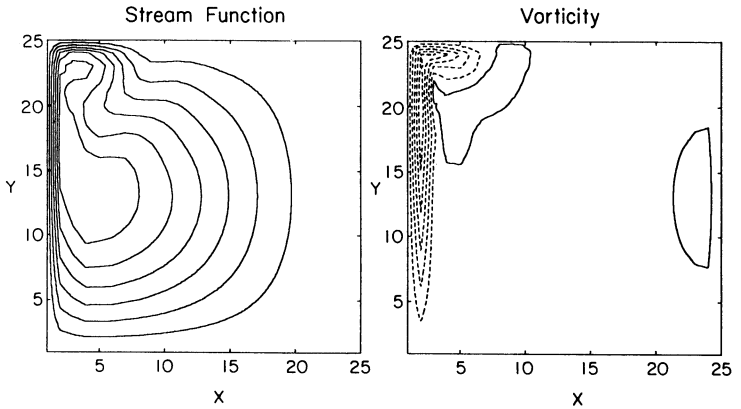


Fig. 3. Steady stream function and vorticity for $\kappa/\beta L = 0.05$, $\epsilon/\beta L^3 = 0.0001$, and $A/\beta^2 L^2 = 0.1$ (case 1 in Table 1).

sum of the squares of the rate of change of the vorticity over a single time step, is the same as the cost function defined in eqn. (4). The gradient is computed by taking a single time step forward with the model and then a step backward with the adjoint. Note that the present problem is not time dependent, and, therefore, time does not enter explicitly. Yet, the computation is done using time-dependent codes for the model and adjoint model, and the forward and backward steps are simply a device for computing first the residual (eqn. (2)) and then the gradient (eqn. (5b)), with the minimum of change to be existing time-dependent model-adjoint code. The only other modifications were associated with the implementation of the preconditioning transformations.

First, consider what we refer to as the standard case having values of $\kappa' = 0.05$ and $\epsilon' = 0.0001$, respectively, for the non-dimensional Newtonian- and Laplacian-friction parameters. The steady stream function and vorticity are shown in Fig. 3. This solution has been computed in four different ways: (1) using a traditional time-marching approach with the maximum stable time step and a single multigrid Poisson iteration per time step; (2) using the optimization approach with the stream function defining the space in which the mimimum is sought; (3) using the optimization approach with vorticity; and (4) using the optimization approach with the Laplacian of vorticity. These examples allow us to compare time-stepping with optimization and to demonstrate the importance of preconditioning. Figure 4 shows the rates at which the various methods converge.

For time-stepping, the logarithm of the ratio of the cost to the initial cost is plotted as a function of the number of steps used to spin the model up. In the first 20 or so time steps the cost increases, but thereafter it decreases

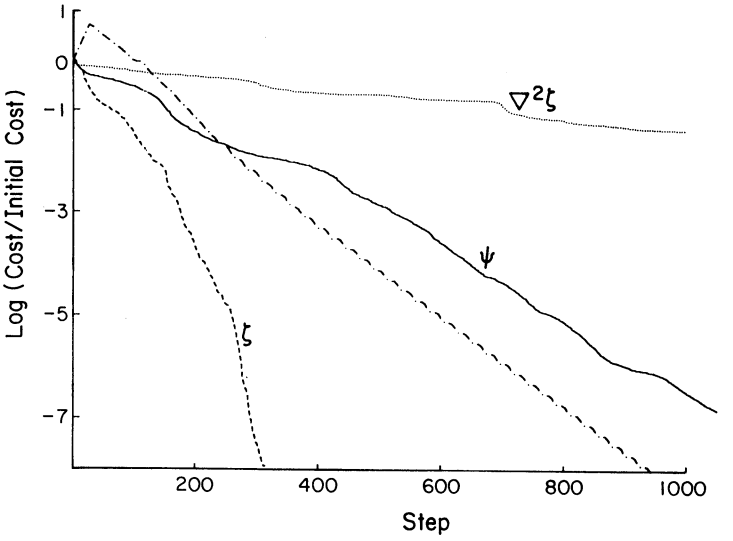


Fig. 4. Convergence rates for time-stepping (unmarked curve), for conjugate-gradient descent with no preconditioning (using stream function as optimization unknown, curve marked $\psi$), for conjugate-gradient descent preconditioned by $\nabla^{-2}$ (marked $\zeta$), and for conjugate-gradient descent preconditioned by $\nabla^{-4}$ (marked $\nabla^2\zeta$) for the solution shown in Fig. 3.

exponentially. Note that the stream function was obtained from the vorticity with only a single pass of the multigrid Poisson solver at each time step, so only an approximate solution was obtained. Increasing the number of multigrid iterations to two guaranteed that the Poisson solutions were highly accurate; although this reduced the number of iterations from 966 to 922 (see Table 1) it increased the overall CPU time from 6.420 to 8.242 s.

For the optimization examples, the logarithm of the cost ratio is plotted as a function of the number of evaluations of the cost function and its gradient by the NAG conjugate-gradient routine. (Spikes due to intermediate function evaluations needed by the line search algorithm have been suppressed. The curves simply connect successive values of the cost function at the completion of the descent step; horizontal spacing between plotted points is adjusted to account for the number of function evaluations per descent step.) When the search for the minimum is cast in terms of the stream function, there is no need to solve a Poisson equation, so each iteration is less expensive than a single step of the time-stepping method.

When vorticity is taken as the optimization variable, one Poisson equation must be solved in order to evaluate the residual and another to evaluate the gradient, so an iteration here is roughly equivalent to two time steps of the time-marching method; however, additional computations are

TABLE 1

Results of the three cases described in the text

| Case | Method | $\epsilon_b$ | $\epsilon_h$ | Variable | m-g cycles | Steps | CPU | Figures |
|---|---|---|---|---|---|---|---|---|
| 1 | c-g | 0.05 | 0.0001 | $\psi$ | – | 1236 | 12.365 | 3, 4, 5 |
| | c-g | 0.05 | 0.0001 | $\zeta$ | 2 | ♥340 | ♥4.997 | |
| | c-g | 0.05 | 0.0001 | $\nabla^2\zeta$ | 4 | $\gg$ | $\gg$ | |
| | Time-stepping | 0.05 | 0.0001 | – | 1 | 966 | 6.420 | |
| | Time-stepping | 0.05 | 0.0001 | – | 2 | 922 | 8.242 | |
| 2 | c-g | 0.15 | 0.0001 | $\psi$ | – | 1746 | 17.050 | 6, 7, 8 |
| | c-g | 0.15 | 0.0001 | $\zeta$ | 2 | ♥84 | 1.230 | |
| | c-g | 0.15 | 0.0001 | $\nabla^2\zeta$ | 4 | 2120 | 60.414 | |
| | Time-stepping | 0.15 | 0.0001 | – | 1 | 116 | ♥0.783 | |
| | Time-stepping | 0.15 | 0.0001 | – | 2 | 114 | 1.028 | |
| 3 | c-g | 0.05 | 0.01 | $\psi$ | – | $\gg$ | $\gg$ | 9, 10, 11 |
| | c-g | 0.05 | 0.01 | $\zeta$ | 3 | 636 | 10.809 | |
| | c-g | 0.05 | 0.01 | $\nabla^2\zeta$ | 4 | ♥236 | 6.673 | |
| | Time-stepping | 0.05 | 0.01 | – | 1 | 274 | ♥1.827 | |
| | Time-stepping | 0.05 | 0.01 | – | 4 | 270 | 3.649 | |

c-g, conjugate-gradient; multigrid, ●●●

Efficiency of calculation is measured in terms of both number of time steps and computer time (CPU seconds) required to reduce the cost function to below $10^{-6}$. For the conjugate gradient optimization, computing the gradient requires one forward step of the model and one backward step of the adjoint model, which together count as two steps in this table. The most efficient method of calculation for each case (tested separately in terms of CPU time and number of steps) is indicated by ♥. When $\gg$ appears instead of number of steps or CPU seconds it indicates that the conjugate gradient optimization did not converge after a very large number of iterations.

needed for the line search. With the Laplacian of the vorticity as the optimization variable, four Poisson solutions (two biharmonic inversions) were required to compute the cost and gradient, but to no avail (see Table 1); the computations failed to converge. Results were consistent with expectations based on the eigenvalues analysis, which indicated that vorticity would be the best of these three choices for optimization.

In order to compare with the eigenvalue spectra of Figs. 1 and 2, we have computed the actual eigenvalues of the unpreconditioned Hessians and both preconditioned Hessians for the standard case. The Hessians are obtained by finite differences from the gradient vectors computed using the adjoint model, and the eigenvalues are calculated using a NAG routine for symmetric matrices. The resulting spectra, which are shown in Fig. 5, differ from those based on the Fourier modes because of the effects of the

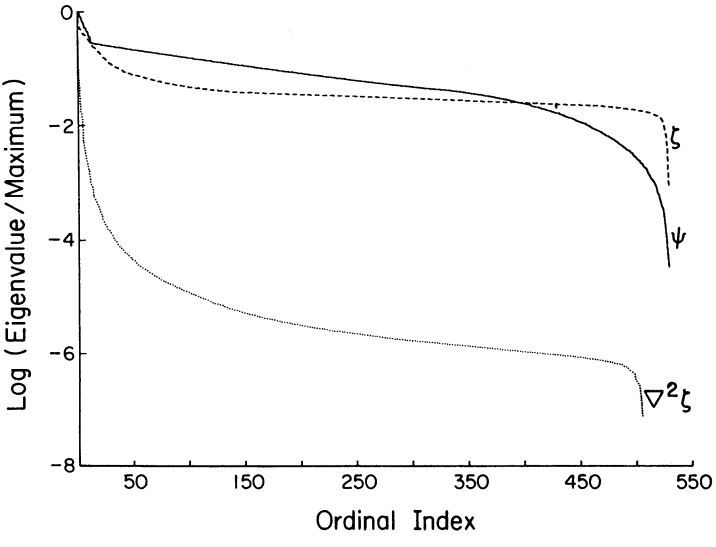


Fig. 5. Eigenvalue spectra of the Hessian matrices for the solution shown in Fig. 3. Curves marked $\psi$, $\zeta$, $\nabla^2\zeta$ correspond to the same choices for the optimization variables as in Fig. 2.

non-linearities, but there is still qualitative agreement, indicating that the heuristic analysis has some merit.

To understand better the way preconditioning works, we now consider some other choices for the friction parameters. Increasing the value of the non-dimensional bottom friction to 0.15 from 0.05 yields the steady flow shown in Fig. 6. The convergence rate for both choices of preconditioning (Fig. 7) is much faster than for the previous case, as might be expected

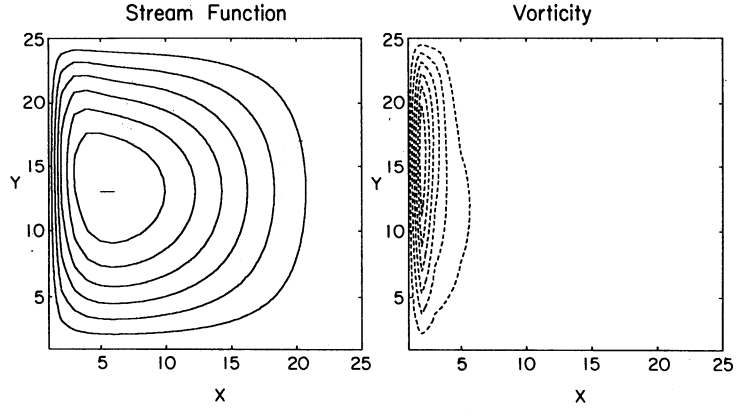


Fig. 6. Steady stream function and vorticity for $\kappa/\beta L = 0.15$, $\epsilon/\beta L^3 = 0.0001$, and $A/\beta^2 L^2 = 0.1$ (case 2 in Table 1).

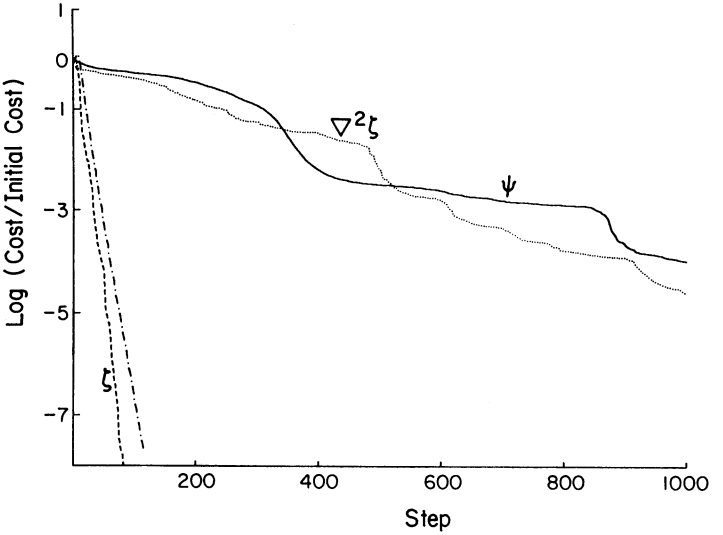


Fig. 7. Convergence rates as in Fig. 4, but for the solution shown in Fig. 6.

from the larger unpreconditioned eigenvalues with smaller scales. Their eigenvalue spectra (Fig. 8) are characterized by more degeneracy and smaller condition numbers. Convergence rates with vorticity as the optimization variable are quite good, but not as good as for time-stepping. Although fewer steps were required for the vorticity-optimization ap-

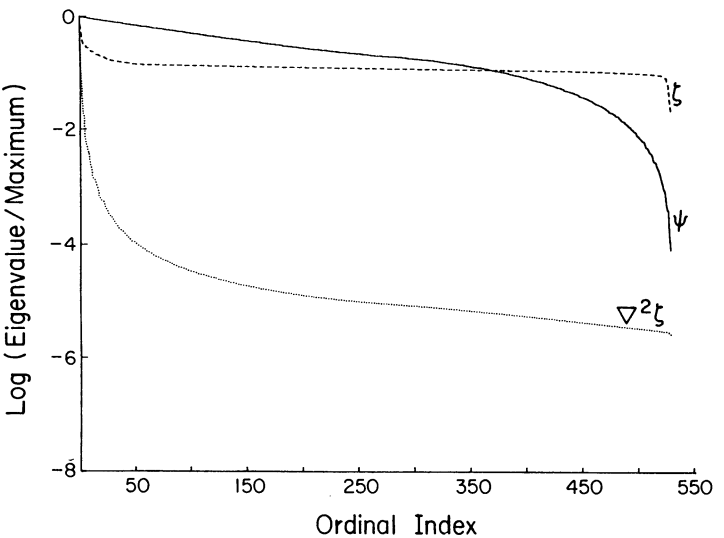


Fig. 8. Eigenvalue spectra of the Hessian matrices for the solution shown in Fig. 6.

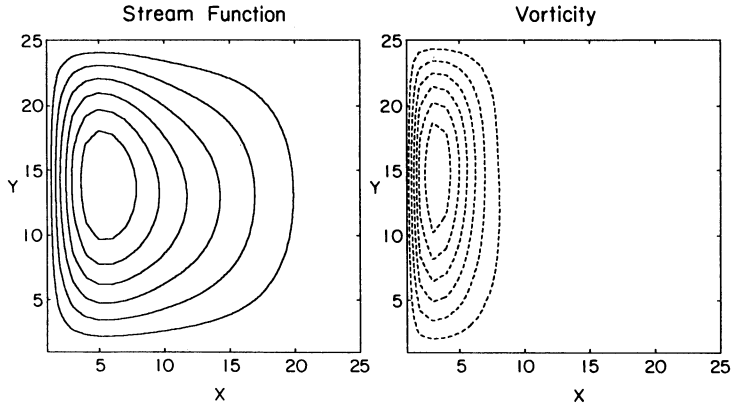


Fig. 9. Steady stream function and vorticity for $\kappa/\beta L = 0.05$, $\epsilon/\beta L^3 = 0.01$, and $A/\beta^2 L^2 = 0.1$ (case 3 in Table 1).

proach, time-stepping required less CPU time because we could get away with using only one multigrid Poisson pass. Unfortunately, a similar trick does not seem to work for the optimization approach, probably because the error in the Poisson solution does not give the required accuracy in the cost function and its gradient. This does not mean that the optimization approach could not be made more efficient. What is needed is a better preconditioning transformation. The two that we have considered are quite simple and are intended only to demonstrate how the method works.

Increasing the value of the non-dimensional horizontal viscosity to 0.01 from 0.00010, with the body friction taken at the standard value of 0.05 results in the solution shown in Fig. 9 with convergence rates that are shown in Fig. 10. The eigenvalues for this case are shown in Fig. 11. Biharmonic preconditioning requires the fewest iterations in this case, but time-stepping again takes the least CPU time. Perhaps by examining different possibilities, using the eigenvalue analysis described in Section 3, a better preconditioning transformation can be found, which would result in faster convergence.

The results for the different choices of the optimization variables are consistent with the ideas discussed in Section 3. Slow convergence is a result of the breadth of the eigenvalue spectrum, which in turn is determined by the spatial scales resolved by the model. In every case the $\beta$-term dominates at large scales, but the friction terms can dominate at small scales, depending on the values of the coefficients. For the standard values this is the case, with both types of friction contributing about the same. A factor of 3 increase in body friction increases the eigenvalues associated with the small scales and thus makes the Poisson preconditioning more necessary. A hundredfold increase in horizontal friction causes the small-

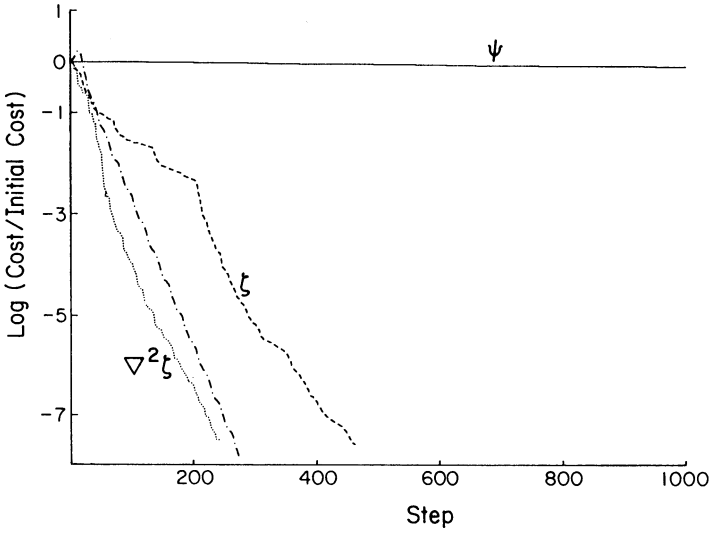


Fig. 10. Convergence rates as in Fig. 4, but for the solution shown in Fig. 9.

scale part of the spectrum to be spread out much more, so double Poisson preconditioning by using the Laplacian of the vorticity is now more effective. This sort of analysis may be very useful in more complicated situations where preconditioning is crucial for optimization to be computationally feasible.

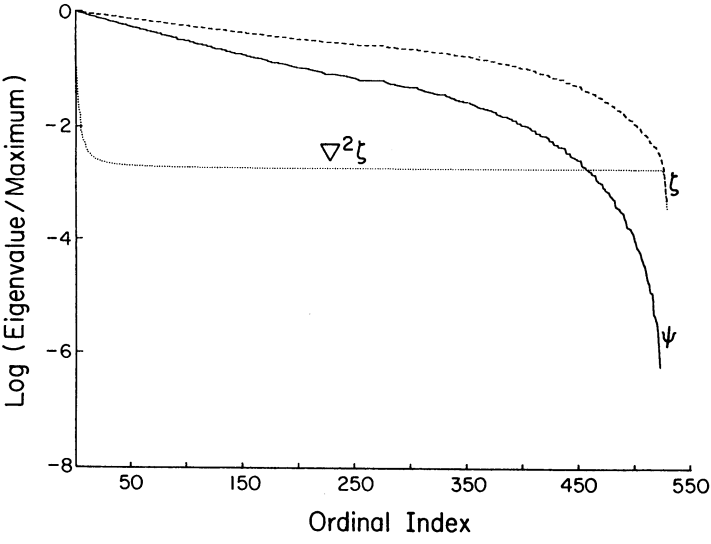


Fig. 11. Eigenvalue spectra of the Hessian matrices for the solution shown in Fig. 9.

It is useful to speculate on the use of an optimization approach to computing the steady state for a more realistic model. The important point is that the condition number is associated with the extreme temporal scales of the model. Whereas the spatial scales of the vorticity model completely determine its temporal scales, the richer variety of wave and mixing processes in more sophisticated models would play an important role, and intelligent preconditioning would have to take these physical processes into account. The tricks mentioned in the introduction for accelerating convergence of time-stepping to steady state (the two-time-scale approach and diagnostic spin-up), which exploit knowledge of the temporal scales, should have counterparts that improve the conditioning of the more general steady-state optimization problem. Still, preconditioning will have to address contribution to temporal extremes due to the range of spatial scales.

It is interesting to note that the best preconditioning transformation in two out of three of the cases we have tried was related to which term dominates the small scales in the model equation. When bottom friction dominates the small scales (case 2 in Table 1), the vorticity is the most efficient control variable for the optimization. When the horizontal friction is large and dominates the small scales (case 3), the Laplacian of the vorticity is best. This is an important observation, as it gives us some intuitive feeling on the way preconditioning may work in more complex situations. Let us try and explain this, using case 3 as an example. We then comment on the standard case (case 1), for which a more general form of the same explanation is needed. Begin by considering expressions (8b), (9b) and (10) for the eigenvalue spectrum for the different preconditioning transformations, noting again that the largest eigenvalues are determined by the small scales (i.e. by large wavenumbers $k_x$, $k_y$). Let $(k_{x-\min}, k_{y-\min}) \approx \pi/L$ be the smallest wavenumbers in the $(x, y)$ directions, corresponding to scale of the basin, $L$. Let also $(k_{x-\max}, k_{y-\max}) \approx \pi/\Delta$ be the largest wavenumbers, corresponding to the grid scale. When the horizontal friction coefficient is as large as in case 3, the term $\epsilon(k_{x-\max}^2 + k_{y-\max}^2)^2$ in eqn. (8b) dominates the largest eigenvalue, while the term $\beta k_{x-\min}$ dominates the smallest (in fact, for the very large value of the horizontal friction used in case 3, the $\beta$-term and the horizontal friction term are both of the same order of magnitude for the large scales, but this does not invalidate the explanation below). The condition number is, therefore

$$\frac{\epsilon\left(k_{x-\max}^2 + k_{y-\max}^2\right)^2}{\beta k_{x-\min}} \tag{11a}$$

When the Laplacian of the vorticity is used as the control variable, the largest eigenvalue is simply dominated by $\epsilon$, and the smallest is dominated

by $\beta k_{x-\min}(k^2_{x-\min} + k^2_{y-\min})^{-2}$. In this case the condition number is smaller:

$$\frac{\epsilon\left(k^2_{x-\min} + k^2_{y-\min}\right)^2}{\beta k_{x-\min}} \tag{11b}$$

explaining the fact that the Laplacian of vorticity is a better control variable for this parameter range.

The standard case (3 in Table 1) offers insight into more complex (and realistic) situations, where no single term clearly dominates the model equation. In this case the vorticity is predicted by the analytic expressions for the eigenvalues to be the best preconditioning transformation, and indeed it turns out to be the best choice in the numerical experiments (Figs. 3–5). But an examination of the different terms in the model equation shows that the horizontal friction is the dominant term for the smallest length scales and not the vorticity. In general, the preconditioning operator should approximate the Hessian (preconditioning by the Hessian should give a condition number equal to 1). When vorticity is the best choice, this means that the 'shape' of the spectrum for bottom friction alone is more similar to the shape of the full spectrum (in terms of stream function) than is the shape of the spectrum for $\beta$ alone or the shape for horizontal friction alone. When Laplacian of vorticity is the best choice, the shape of the spectrum for horizontal friction alone is closest to the shape of the full spectrum. The spectrum in terms of the best choice will be the flattest. When a single term dominates many scales in the model equations (as in cases 2 and 3), its contribution to the eigenvalue spectrum will be similar to the spectrum of the full model, and it will, therefore, be the best preconditioning. In more complicated cases (such as our case 1) the global shape of the eigenvalue spectrum must be considered, and our heuristic derivation of the eigenvalues seems capable of taking into account this factor, as evident in the successful prediction of vorticity as the best preconditioning in case 1.

## 5. CONCLUSIONS

The computational examples have demonstrated that the optimization approach using a conjugate-gradient algorithm is a viable alternative to the traditional time-marching method for computing steady circulation, at least when an efficient preconditioning transformation is available. But would it work for a more complicated oceanic general circulation model (OGCM)? To properly answer that question is beyond the scope of this paper, but a few comments here might be appropriate.

First, defining a steady-state cost function for a GCM is fairly straightforward. Sums of squares of residuals for each type of equation (temperature, salinity, velocity components, etc.) would represent different types of costs. The sum of these individual costs, with coefficients to insure dimensional homogeneity, would provide a cost function whose minimum would determine the steady flow. One question would be how to determine the magnitudes of the coefficients, i.e. the cost associated with one type of residual relative to the others; this is essentially the question of the appropriate scales for non-dimensionalization, and such scaling issues are intimately tied to the problem of preconditioning.

Once the cost function is defined, the next question is how to compute its gradient without worrying about preconditioning. Again, it is possible to advance the model one step in time in order to compute the residuals of the steady equations. The residuals can then be multiplied by the dimensional coefficients used in defining the cost. The results are then used in the adjoint model to get the gradient vector. This would follow exactly the same pattern as described above, except for the minor effort needed to accommodate the coefficients.

The difficult aspect will be the choice of an appropriate preconditioning transformation. A GCM will have a larger variety of wave types than the vorticity-equation model considered here; there will be fast gravity waves as well as slow Rossby waves. Consequently, the spectrum can be expected to be more spread out. Although the preconditioning problem will still involve the different spatial scales, it will be further complicated by the different wave types. Rather than speculating further, it is best to leave the details to a future paper supported by computations.

The results presented here have not made a strong case for the optimization approach as an alternative to time-stepping. More mathematical research is still needed. The most important point of the paper is the analysis of numerical conditioning in terms of the model's temporal scales. This provides the physical guidance for finding the preconditioning transformations that might make the optimization approach a practical alternative.

Results of this paper have implications for other adjoint-related oceanographic modeling efforts. For example, one problem of current interest is that of fitting a steady, general circulation model to data by varying initial conditions (and boundary fluxes, etc.) until changes over a single time step and deviations from the data are both small. If the model uses the rigid-lid approximation, the barotropic flow is governed by a vorticity equation, so, in the limit that steadiness is much more important than data, the analysis presented here applies directly. Results presented here suggest that, for a physically important range of mixing parameters, vorticity should be a

much better control variable than stream function for those problems. It also suggests that failure to converge might be due to ill conditioning of the optimization problem. If a successful preconditioning transformation cannot be found, it might be a good idea to reformulate these inverse problems so that the steady problem must be solved exactly, adjusting the boundary fluxes, etc., but 'not' initial conditions. At each iteration it would be necessary to compute the steady model flow, given the fluxes, and to solve a steady adjoint problem, hoping to exploit speed of time-stepping in computing this sequence of steady solutions.

## REFERENCES

Arakawa, A., 1966. Computational design for long term numerical integration of the equations of fluid motion: Two dimensional incompressible flow. Part 1. J. Comput. Phys., 1: 119–143.

Brandt, A., 1984. Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, GMD-Studien Nr. 85. Gesselschaft für Mathematik und Datenverarbeitung mbH Bonn, 183pp.

Bryan, K., 1984. Accelerating the convergence to equilibrium of ocean-climate models. J. Phys. Oceanogr., 14: 666–673.

Cox, M.D., 1987. An eddy-resolving numerical model of the ventilated thermocline: Time dependence. J. Phys. Oceanogr., 17: 1044–1056.

Dennis, J.E. and Schnabel, R.B., 1983. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Engelwood Cliffs, NJ, 378pp.

Fletcher, R., 1987. Practical Methods of Optimization. Wiley, New York, 436pp.

Gill, P.E., Murray, W. and Wright, M.H., 1981. Practical Optimization. Academic Press, San Diego, CA, 401pp.

Moro, B., 1988. A model of barotropic circulation with unsymmetric forcing. I. Steady flows. Dyn. Atmos. Oceans, 12: 259–287.

Numerical Algorithms Group (NAG), 1984. Fortran Library Manual, Mark II, 6 Vols, NAG, Oxford, UK.

Sarmiento, J.L. and Bryan, K., 1982. An ocean transport model for the North Atlantic. J. Geophys. Res., 87: 394–408.

Semtner, A.J. and Chervin, R.M., 1988. A simulation of the global ocean circulation with resolved eddies. J. Geophys. Res., 93: 15 502-15 522.

Strang, G., 1986. Introduction to Applied Mathematics. Wellesley–Cambridge, MA, 758pp.

Thacker W.C., 1989. The role of the Hessian matrix in fitting models to measurements. J. Geophys. Res., 94: 6177–6196.

Thacker, W.C. and Long, R.B., 1988. Fitting dynamics to data. J. Geophys. Res., 93: 1227–1240.

Tziperman, E. and Thacker, W.C., 1989. An optimal-control/adjoint-equations approach to studying the oceanic general circulation. J. Phys. Oceanogr., 19: 1471–1485.

Vichnevetsky, R. and Bowles, J.B., 1982. Fourier Analysis of Numerical Approximations of Hyperbolic Equations. SIAM, Philadelphia, PA, 140pp.

## APPENDIX: DERIVATION OF THE HESSIAN OPERATOR

To compute the Hessian at a point $\psi_0$, it is simplest to start by expressing the cost function in terms of perturbations $\rho$ about $\psi_0$:

$$J(\psi_0+\rho)=\int\left[R_0(\psi_0)+L(\psi_0)\rho+J(\rho,\nabla^2\rho)\right]^2\,\mathrm{d}x\,\mathrm{d}y \tag{A1}$$

The Hessian is the second functional derivative of $J$ with respect to $\rho$ at $\rho=0$, so contributions to the Hessian come from the terms of the integrand that are quadratic in $\rho$. Thus, the first functional derivative of

$$\int\left\{\left[L(\psi_0)\rho\right]^2+2R_0J(\rho,\nabla^2\rho)\right\}\,\mathrm{d}x\,\mathrm{d}y \tag{A2}$$

yields the Hessian operator acting on $\rho$:

$$G(\psi_0)\rho=L^*(\psi_0)L(\psi_0)\rho-J(R_0,\nabla^2\rho)-\nabla^2J(R_0,\rho) \tag{A3}$$