Algorithmic Improvements to Finding Approximately Neutral Surfaces

Geoffrey J. Stanley, Trevor J. McDougall, and Paul M. Barker

1School of Mathematics and Statistics, University of New South Wales, Sydney, NSW, Australia

Abstract

Interior oceanic motions occur predominantly along, rather than across, the neutral tangent plane. These planes do not link together to form well-defined surfaces, so oceanographers use approximately neutral surfaces. To date, the most accurate such surface is the ω-surface, but its practical utility was limited because its numerical implementation was slow and sometimes unstable. This work upgrades the speed, robustness, and utility of ω-surfaces. First, we switch from solving an overdetermined matrix problem by minimal least squares, to solving an exactly determined matrix problem, obtained either by the normal equations (multiplication by the matrix’s transpose) or by discretizing Poisson’s equation derived from the original optimization problem by the calculus of variations. This reduces the computational complexity, roughly from $O(N^{3.5})$ to $O(N^{1.2})$, where $N$ is the number of grid points in the surface. Second, we update the surface’s vertical position by solving a nonlinear equation in each water column, rather than assuming the stratification is vertically uniform. This reduces the number of iterations required for convergence by an order of magnitude and eliminates the need for a damping factor that stabilized the original software. Additionally, we add “wetting” capacity, whereby incrops and outcrops are reincorporated into the surface should they become neutrally linked as iterations proceed. The new algorithm computes an ω-surface in a 1,440 by 720 gridded ocean in roughly 15 s, down from roughly 11 h for the original software. We also provide two simple methods to label an ω-surface with a (neutral) density value.

Plain Language Summary

In the deep ocean interior, the circulation is much simpler than a fully three-dimensional flow, because density places a strong constraint on the flow: water “wants” to be neutrally buoyant with its environment. As such, the deep ocean is well described as a stack of two-dimensional surfaces, flowing within (rather than across) these surfaces. Because seawater’s density is a nonlinear function of salinity, temperature, and pressure, these surfaces do not exist in reality: they can only be approximated. Oceanographers have been approximating these surfaces for nearly a century, but for half of that time it was not known exactly what surface was being approximated. Only in the past decade has an algorithm existed to numerically approximate these surfaces with great accuracy. However, that algorithm was computationally slow and occasionally unstable, greatly limiting the practical use of these approximate-yet-accurate surfaces. We have overhauled the algorithm, making it over 2,500 times faster on a typical modern oceanic data set (having one data point every quarter degree of latitude or longitude). Many oceanographic discoveries rely upon these types of approximate surfaces for their analyses. Our work enables such a pace of discovery to continue into the modern era of big ocean data.

1. Introduction

In most of the ocean, away from localized turbulent mixing events, the circulation distributes and mixes fluid largely along, rather than across, two-dimensional surfaces. These surfaces bend and undulate, stacked atop each other forming a continuum from the abyss to the base of the mixed-layer. Because of this along versus across anisotropy, the ocean interior can be effectively modeled as a set of shallow, immiscible layers, bounded above and below by these surfaces. Likewise, analyzing oceanic properties is often best done on such surfaces (Iselin, 1939), analogously to isentropic analysis in a dry atmosphere, for example.

The ideal of these surfaces is a neutral surface (McDougall, 1987), along which a fluid parcel, moving laterally and mixing with its environment as it goes, experiences no buoyant restoring forces: its density always matches the density of its environment. If the parcel were instead moving along a different surface,
its density would differ from that of its environment, and buoyant forces would tend to restore its vertical position toward the neutral surface.

However, neutral surfaces do not exist in the real ocean. This is a consequence of the fact that seawater is a binary fluid whose density is a nonlinear function of Absolute Salinity (henceforth, salinity) and Conservative Temperature (henceforth, temperature), as well as pressure. In such a fluid, neutral surfaces only exist if the “neutral helicity” is zero, which occurs when the gradients of salinity, temperature, and pressure are coplanar (McDougall & Jackett, 1988). (Zero neutral helicity is necessary but not sufficient for the existence of neutral surfaces, because islands can “hide” neutral helicity [Stanley, 2019a].) The real ocean does not, in general, manifest this condition of zero neutral helicity, but it is not far off (McDougall & Jackett, 2007). As such, spatially extensive surfaces in the ocean cannot exactly meet the ideal of a neutral surface, but they can do so approximately, with great accuracy. These are called “approximately neutral surfaces.”

To date, the most accurate class of approximately neutral surfaces is the $\omega$-surface (Klocker et al., 2009, hereafter KMJ). These surfaces are individually built to minimize errors from neutrality, without regards to being quasi-material or pycnotropic. This choice demands some discussion, since these functional forms provide beneficial oceanographic properties.

A three-dimensional density variable is quasi-material if it is a function only of salinity and temperature, while it is pycnotropic if it is a function only of specific volume (or in situ density) and pressure. With the quasi-material form, the only material sources of the density variable are directly caused by material sources of salinity or temperature. Such a density variable is unaffected by compressibility effects, which also has the benefit of filtering out vertical heave (Lang et al., 2020). With the pycnotropic form, the horizontal pressure gradient acceleration transforms perfectly when the density variable is used as the vertical coordinate, thereby producing an exact geostrophic streamfunction on isosurfaces of the density variable and producing an Ertel potential vorticity with no baroclinic production term (de Szoeke, 2000; de Szoeke et al., 2000). Now, a 3D density variable that is either quasi-material or pycnotropic can be constructed so that its isosurfaces are quite neutral, locally. Classic examples are to use potential density (a quasi-material variable) only near its reference pressure, or to use specific volume anomaly (a pycnotropic variable) only in a region near where its reference salinity and temperature values are selected. However, such variables cannot achieve good neutrality, globally. This is because water mass variations, such as between the northern and southern hemispheres, break the single-valued relationship imposed by the quasi-material or pycnotropic form (McDougall & Jackett, 2005). More specifically, Stanley (2019a) showed that there is a functional relationship between specific volume and pressure, or between salinity and temperature, along a (hypothetical) neutral surface, but that this relationship is multivalued and varies in space. Lang et al. (2020, Section 2.c) extended this fact of 2D neutral surfaces to the 3D case, showing that any pycnotropic or quasi-material density variable cannot maintain good neutrality, globally. We can now translate the quasi-material and pycnotropic concepts from 3D density variables to 2D surfaces.

We call an approximately neutral surface quasi-material if salinity is a function of temperature on that surface and call it pycnotropic if in situ density is a function of pressure on that surface. Recently, Stanley (2019a) created topobaric surfaces, which are locally pycnotropic: on the surface, in situ density is locally a function of pressure, but that function varies between regions of the surface. These regions were carefully determined using recent advances in computational topology, enabling topobaric surfaces to remain continuous across the region boundaries, unlike how discontinuities arose at latitudinal boundaries of extended orthobaric density (de Szoeke & Springer, 2009) or geopotential boundaries of patched potential density (Reid & Lynn, 1971). The locally pycnotropic form means topobaric surfaces maintain the good dynamic properties of pycnotropic variables and hence possess an exact geostrophic streamfunction (Stanley, 2019b), while also achieving a high degree of neutrality, globally. Alternatively, topobaric surfaces could have been constructed with the other form, to be locally quasi-material. Currently, the topological approach underlying topobaric surfaces is the only known way for a quasi-material or pycnotropic surface to be highly neutral, globally. It is therefore good to have a diversity of methods, with $\omega$-surfaces focusing solely on neutrality measures.

To minimize neutrality errors, $\omega$-surfaces begin from an initial approximately neutral surface and iterate over a two-step process. The first step is an optimization problem, solving for the density perturbation that
will minimize the neutrality error. The second step is to move the surface vertically by an amount determined by this density perturbation. We have overhauled both of these steps, as outlined next.

The first step, once discretized, is a globally coupled matrix problem for the density perturbation. This matrix problem is overdetermined, so KMJ sought a least squares solution. We turn this into an exactly determined system representing a discretized version of Poisson's equation, in two ways. First, we show how to transform the optimization problem into a Poisson equation by the calculus of variations. Second, we take the KMJ overdetermined matrix problem and apply the "normal equations," which simply multiplies both sides of the matrix equation by the matrix's transpose. These two approaches are intimately related, as follows. The KMJ matrix represented the discretized gradient operator, and we show that its negative transpose represents the discretized divergence operator. Thus, the product of these matrices represents the discretized Laplacian operator, and the normal equations have turned the KMJ matrix problem into a discretized Poisson equation. Numerically, the new matrix is sparse, square, symmetric, and full-rank—enabling use of a sparse Cholesky solver. Using a Cholesky solver, rather than the LSQR solver used by KMJ, reduces the time complexity of the ω-surface algorithm from roughly $O(N^{1.6})$ to roughly $O(N^{1.2})$, where $N$ is the number of grid points in the surface.

The second step is to update the vertical position of the surface. The original ω-surface algorithm divided the density perturbation field by the local density stratification, thereby obtaining a depth perturbation. This assumes, effectively, that the stratification is constant in the vertical direction and relies on subsequent iterations to refine the solution. Instead, the new algorithm searches vertically through each water column to find the appropriate depth for the updated surface. In practice, this new algorithm slightly increases the computational cost of each iteration but drastically reduces the number of iterations required for convergence. This new algorithm is akin to the way topobaric surfaces update their vertical position (Stanley, 2019a) and bears some similarity to level of neutral buoyancy calculations (Groeskamp et al., 2019; Jackett & McDougall, 1997; McDougall, 1987; Tailleux, 2013).

We have also added a new, minor step to each iteration of the ω-surface algorithm, called “wetting.” This enables “dry” water columns (where the surface has incropped or outcropped) to join the approximately neutral surface—that is, to “wet.” Such wetting is done when there is a neutral trajectory from the surface at a wet water column to a neighboring dry water column, ending above the sea-floor and below the sea-surface. This wetting algorithm allows the surface to expand laterally from its initial surface, as appropriate. For example, suppose the potential density surface that initializes the algorithm outcrops at some latitude in the Southern Ocean. The ω-surface algorithm may iteratively lessen the surface’s slope. With wetting, the final ω-surface will extend further poleward than the original potential density surface.

Another new feature is the provision of a physically motivated density value that labels a given ω-surface, once computed. Two options are provided. The first gives a density value that is only internally consistent with the oceanic data set in which the ω-surface lives, and thus cannot be used in comparison with other data sets. The second is a Jackett & McDougall (1997) $\gamma^d$ neutral density value, which enables comparison with published neutral density values. However, we stress that this comparison is not entirely straightforward, because the $\gamma^d$ label for the ω-surface will exactly equal $\gamma^d$ evaluated on the ω-surface only at a single point. That is to say, ω-surfaces and $\gamma^d$ isosurfaces differ.

Section 2 reviews and reformulates the theory of ω-surfaces. Section 3 details the numerical calculation of ω-surfaces, comparing the new algorithm to the KMJ algorithm where relevant. Numerical tests are reported in Section 4, mostly documenting the improved computational complexity of the new algorithm. Conclusions are provided in Section 5. Appendix A discusses how to assign density labels to ω-surfaces. Appendix B briefly describes Brent’s method for solving a nonlinear equation, and a minor extension to use an initial guess of the solution. Appendix C details the conditions under which the matrix problem possesses exact solutions. Appendix D examines the adjoint relationship between the discretized gradient and divergence operators.

Before proceeding, some housekeeping arises.

- The theory presented here is best cast with the TEOS-10 equation of state (IOC et al., 2010), a function of the quasi-conservative tracers Absolute Salinity $S$ and Conservative Temperature $\Theta$, and of pressure.
However, when analyzing model data, the model’s equation of state and its inputs should be adopted; this change passes through the theory seamlessly.

- We present the theory appropriate for a non-Boussinesq ocean and track a surface by its pressure at each lateral point. For a Boussinesq ocean, which uses depth instead of pressure in its equation of state (Young, 2010), we track the depth (positive and increasing downward, like pressure) of the surface, and the theory proceeds identically: simply replace pressure with depth.

- Software to compute $\omega$-surfaces is available at https://github.com/geoffstanley/neutral-surfaces in MATLAB. A Python version is under development.

## 2. Theory of $\omega$-Surfaces

### 2.1. Background

In an ocean of salinity $S$, temperature $\Theta$, and pressure $p$, the equation of state $R$ determines the in situ density $\rho = R(S, \Theta, p)$. The partial derivatives of $R$ with respect to $S$ and $\Theta$ define new scalar fields $\rho_S = \partial_R R(S, \Theta, p)$ and $\rho_\Theta = \partial_R R(S, \Theta, p)$ that are respectively akin to the haline and thermal expansion coefficients. Let $I$, $J$, and $K$ be the eastward, northward, and upward unit vectors, respectively.

Consider an approximately neutral surface in this ocean, on which the salinity, temperature, and pressure are $\tilde{S}$, $\tilde{\Theta}$, and $\tilde{p}$, respectively. Specifically, the overtilde denotes the restriction of a 3D scalar field to the 2D surface, followed by its projection onto the sphere of radius $R_{\text{Earth}} \approx 6.37 \times 10^6$ m. With this notation, $\nabla \tilde{S}$ provides the projected nonorthogonal gradient of the salinity on a surface, similar to what the neutral surface literature commonly denotes by $\nabla S = \left( \partial_x S, \partial_y S \right) I + \left( \partial_x S, \partial_y S \right) J$ where the partial derivatives sample $S$ within an approximately neutral surface (here, at constant $a$) but ignore variations in the surface’s height when measuring distance between sample points (Stanley, 2019a). Note that $\nabla \tilde{S}$ is a 2D vector field, whereas $\nabla S$ is a 3D vector field.

Let $A$ be a connected subset of the sphere where $\tilde{S}$, $\tilde{\Theta}$, and $\tilde{p}$ are defined, that is, where the surface has neither incropped nor outcropped.

The dianeutral vector, key to studying neutral surfaces, is defined as

$$ N = \rho_S \nabla \tilde{S} + \rho_\Theta \nabla \tilde{\Theta}, $$

and is commonly thought of as the gradient of locally referenced potential density (LRPD). The neutral tangent plane, defined at every point in the ocean, is the plane orthogonal to $N$. Taking the $S$ and $\Theta$ gradients within a surface, instead, gives the neutrality error,

$$ \varepsilon = \tilde{\rho}_S \nabla \tilde{S} + \tilde{\rho}_\Theta \nabla \tilde{\Theta}. $$

If $\varepsilon = 0$, then the surface is everywhere normal to the dianeutral vector: a perfectly neutral surface (McDougall, 1987). However, the nonlinear equation of state and the existence of nonzero neutral helicity precludes the existence of surfaces with $\varepsilon = 0$ everywhere (McDougall & Jackett, 1988). The goal is to construct a surface for which $\varepsilon$ is as small as possible, in some sense.

The way $\omega$-surfaces achieve this minimization is as follows. KMJ noted that the 2D curl of $\varepsilon$, namely $\partial_y (\varepsilon \cdot J) - \partial_x (\varepsilon \cdot I)$, is largely determined by the stratification and the neutral helicity, both of which are set by the ocean hydrography. Hence, they sought to adjust a surface’s neutrality error, from $\varepsilon$ to $\varepsilon^*$, without changing its 2D curl. Mathematically, they do this by adding a conservative vector field, $\nabla \Phi$, to $\varepsilon$:

$$ \varepsilon + \nabla \Phi = \varepsilon^*, $$

where $\Phi$ is a to-be-determined density perturbation, to be discussed in detail shortly (Section 2.3).

At this point, KMJ moved into the numerical implementation; we defer discussing their numerics until 21 in Section 3. Continuing theoretically for now, we frame the $\omega$-surface problem in two parts: a globally coupled linear problem and a decoupled, nonlinear problem in each water column.
This two-step procedure will be iterated until convergence, discovering an \( \omega \)-surface: a surface whose \( \epsilon \) field cannot be further reduced by adding a conservative vector field to it.

### 2.2. Step 1: The Global, Linear Problem

The global, linear problem is a constrained optimization problem for \( \Phi \):

\[
\min \epsilon[\Phi] , \quad \text{subject to } C[\Phi] = 0 ,
\]

where

\[
\epsilon[\Phi] = \int_A |\nabla \Phi + \epsilon|^2 \, dA
\]

is the \( \ell_2 \) norm of \( \nabla \Phi + \epsilon \), and \( C \) is a linear function of \( \Phi \). The constraint 4b should ensure the solution to 4a is unique, which is otherwise arbitrary up to an additive constant. KMJ ensured \( \Phi \) has zero mean, by setting

\[
\int_A \Phi \, dA = 0 , \quad \text{and } \Phi = \frac{1}{|A|} \int_A \Phi \, dA ,
\]

where \(|A| = \int_A dA\) is the area of the set \( A \) and \( c \neq 0 \) is an otherwise arbitrary constant.

### 2.3. Step 2: The Decoupled, Nonlinear Problem

After solving the global, linear problem 4 for \( \Phi \), the second step is a decoupled, nonlinear problem: use \( \Phi \) to update the pressure of the surface in each water column. KMJ linearized this problem: they divided the density perturbation \( \Phi \) by a factor proportional to the stratification to obtain a pressure perturbation, which is then used to adjust the pressure of the surface. To understand our nonlinear approach, we must first understand Step 1 better.

First consider how the \( \omega \)-surface algorithm can be modified to find an isosurface of, say, potential density referenced to 0 dbar, \( \sigma = \rho - \rho_0 \theta \). Simply replace \( \epsilon \) in 3 and 5 with the gradient of \( \sigma \) in the surface,

\[
\nabla \sigma = \frac{\partial \sigma}{\partial S} \frac{\partial S}{\partial \sigma} + \frac{\partial \sigma}{\partial \theta} \frac{\partial \theta}{\partial \sigma} ,
\]

where \( \sigma = \rho - \rho_0 \theta \). Then, Step 1 would solve 4 exactly (with \( \epsilon[\Phi] = 0 \) and \( \epsilon^* = 0 \)) by \( \Phi = -\nabla \sigma + \sigma_\epsilon \), where \( \sigma_\epsilon = |A|^{-1} \int_A \nabla \sigma \, dA \). Hence, Step 2 should move the surface vertically such that \( \sigma \) on the surface is increased by \( \Phi \)—making \( \sigma \) on the new surface a constant, \( \sigma_\epsilon \).

Next, consider a hypothetically perfect neutral density \( \gamma \) in an ocean with zero neutral helicity: isosurfaces of \( \gamma \) are perpendicular to the dianeutral vector, so \( \nabla \gamma = b \mathbf{N} \) for some scalar field \( b \) (Jackett & McDougall, 1997). Following the above logic, replace \( \epsilon \) in 3 and 5 with \( \epsilon \) in the surface, \( \nabla \epsilon = \frac{\partial \epsilon}{\partial S} \frac{\partial S}{\partial \epsilon} + \frac{\partial \epsilon}{\partial \theta} \frac{\partial \theta}{\partial \epsilon} \), where \( \sigma = \rho - \rho_0 \theta \). Then, Step 1 would find \( \Phi = -\nabla \sigma + \sigma_\epsilon \), where \( \sigma_\epsilon = |A|^{-1} \int_A \nabla \sigma \, dA \). Hence, Step 2 should move the surface vertically such that \( \gamma \) on the surface is increased by \( \Phi \), thereby finding the \( \gamma = \gamma_\epsilon \) isosurface.

Of course, \( \gamma \) and \( b \) are unknown to us, so \( \omega \)-surfaces must proceed without them and use \( \epsilon \) defined by 2. In the above two examples, the surface is moved vertically such that the relevant pressure-corrected density variable is increased by \( \Phi \). Since \( \epsilon \) defined by 2 is the gradient of LRPD within the surface in question, we argue that Step 2 should vertically move the surface in each water column so that the potential density, referenced to the pressure at which the surface intersects the local water column, is increased by \( \Phi \).

To understand Step 2 better, suppose the expansion coefficients \( \rho_S \) and \( \rho_\theta \) were constant. Let \( \tilde{S} \) and \( \tilde{\theta} \) denote the salinity and temperature on the updated surface, satisfying \( \Phi = \rho_S (\tilde{S} - S) + \rho_\theta (\tilde{\theta} - \theta) \) under this assumption. The neutrality error on the updated surface is \( \epsilon' = \rho_S \nabla \tilde{S} + \rho_\theta \nabla \tilde{\theta} \). Using 2 and the gradient of the above equation for \( \Phi \) yields \( \epsilon' = \epsilon + \nabla \Phi = \epsilon^* \). Now, \( \rho_S \) and \( \rho_\theta \) are not actually constant, and instead they vary slightly as the surface is adjusted vertically. As such, \( \epsilon^* \) is only approximately the neutrality error on the updated surface. Nonetheless, \( \omega \)-surfaces proceed iteratively by finding \( \Phi \) to minimize this approximate error.
neutrality error $\varepsilon^*$ on the updated surface. As the iterations proceed, $\Phi$ converges toward zero, and the approximate neutrality error $\varepsilon^*$ should converge to the actual neutrality error $\varepsilon$.

Even though $b$ is unknown to us, some benefit might come from using an approximate, precomputed $b$ field. Riha (2016) explored using $\tilde{b}e$ in place of $\varepsilon$ in 3, solving for $\Phi$, and updating the surface so that its LRPD is increased by $\Phi / \tilde{b} e$. This procedure slightly reduced the number of iterations required for convergence. Alternatively, one could accurately compute $b$ from a pair of nearby $\omega$-surfaces, but this precalculation of two $\omega$-surfaces means that such an algorithm is already slower than just computing an $\omega$-surface without $b$. Given these difficulties, we follow KMJ and proceed iteratively without $b$.

We note that weighting $\varepsilon$ by $\tilde{b}$ does change the optimization problem 4, and the resulting $\omega$-surface would differ from that without the $b$ weighting, but not enormously so, since $b$ does not vary greatly from unity (You & McDougall, 1990). This neglect of $b$ remains a limitation of the 2D neutral surface problem.

### 2.4. Reformulating Step 1 Into Poisson’s Equation

We now reformulate the global, linear part of the $\omega$-surface problem as Poisson’s equation, using the calculus of variations. Suppose $\Phi$ satisfies 4, let $\eta : A \to \mathbb{R}$ be a test function also satisfying $C[\eta] = 0$ and consider $\Phi + \delta \eta$ for any $\delta \in \mathbb{R}$. Linearity of $C$ implies $C[\Phi + \delta \eta] = 0$. Moreover, $\delta \Phi$ is a scalar function of $\delta$ with a minimum at $\delta = 0$, hence

$$0 = \frac{dC[\Phi + \delta \eta]}{d\delta} \bigg|_{\delta=0}$$

$$= 2 \int_A (e + \nabla \Phi) \cdot \nabla \eta \, dA$$

$$= 2 \int_A \eta \nabla \cdot \nabla \Phi \cdot \hat{n} \, d\ell - 2 \int_A \eta \nabla \cdot (\nabla \Phi + e) \, dA,$$

where $\hat{n}$ is the outward normal unit vector on the boundary $\partial A$ of $A$. Consider, temporarily, only those $\eta$ that are 0 on $\partial A$, so that the boundary integral on the last line of 7 vanishes. Then the area integral must be zero; since $\eta$ is otherwise arbitrary, this implies its integrand is zero, namely

$$\nabla^2 \Phi = -\nabla \cdot \varepsilon.$$

(8a)

Now relax the restriction on $\eta$, considering test functions that are nonzero on $\partial A$. Having already established that 8a holds, the area integral on the last line of 7 is zero. Hence the boundary integral on the last line of 7 must be zero, which implies its integrand is zero because $\eta$ is arbitrary:

$$\nabla \Phi \cdot \hat{n} = -\varepsilon \cdot \hat{n}.$$

(8b)

This is called the natural boundary condition. Of course, the additional condition

$$C[\Phi] = 0$$

(9)

remains necessary, to make the solution $\Phi$ unique.

The above mathematical mechanics proceed identically when applied to the 3D neutral density problem, where $\gamma$ is sought to minimize the $\ell^2$ norm of $\nabla \gamma - bN$, assuming $b$ were known. Indeed, the above derivation serves as proof of the statement by Eden and Willebrand (1999) that this minimization problem is equivalent to the Poisson equation $\nabla^2 \gamma = \nabla \cdot (bN)$ with the boundary condition $(\nabla \gamma - bN) \cdot \hat{n} = 0$, where $\hat{n}$ is normal to the ocean boundary. (Actually, Eden and Willebrand [1999] omitted $b$, which impacted their results but not this mathematical equivalence. Riha [2016] explored [in 2D] iteratively solving this Poisson equation with $b$ diagnosed from the previous solution, starting with $b = 1$.)

Comparing 8a with the divergence of 3 reveals that $\varepsilon^*$ must be divergence free: $\varepsilon^* = \nabla \psi'$ for some scalar field $\psi$, where $\nabla$ denotes the gradient rotated 90° anticlockwise, that is, $\nabla \psi' = (-\partial_x \psi', \partial_y \psi')$ and satisfying $\nabla \cdot (\nabla \psi') = 0$. In other words, 3 is equivalently framed as a 2D Helmholtz decomposition, finding scalar fields $\Phi$ and $\psi$ such that
\[ \mathbf{e} = -\nabla \Phi + \nabla \psi. \]  

(10)

The boundary condition 8b makes this Helmholtz decomposition unique (without which \( \Phi + \eta \) also leads to a valid Helmholtz decomposition for any \( \eta \) satisfying \( C[\eta] = 0 \) and \( \nabla^2 \eta = 0 \) in the domain \( \Lambda \)). Loosely speaking, 8b puts as much of \( \mathbf{e} \) into \( \nabla \Phi \) rather than \( \nabla \psi \); mathematically speaking, it minimizes \( \int \mathbf{e} \cdot \mathbf{e} \). We can convert 8 into an equivalent problem with simpler, homogeneous boundary conditions. To do so, change \( \mathbf{e} \) in 8 to be \( \mathbf{e} \) which is identical to \( \mathbf{e} \) except that, on the boundary, the normal component of \( \mathbf{e} \) is set to zero. Then, 8 becomes

\[ \nabla^2 \Phi = -\nabla \cdot \mathbf{e}, \]  

(11a)

\[ \nabla \Phi \cdot \hat{n} = 0. \]  

(11b)

This manipulation of the source charge to yield homogeneous boundary conditions is not uncommon in geophysical fluid dynamics (Marshall, Hill, et al., 1997, and references therein). The distinction between \( \mathbf{e} \) and \( \mathbf{e} \) does not arise when the global, linear problem is framed as an optimization problem, as that only considers gradients appearing in the interior (looking ahead to 14 and 17a).

2.5. Connected Regions and Pinning

An isosurface of a 3D density variable, like potential density, is generally composed of multiple, nonoverlapping regions (connected components). Essentially, KMJ applied their algorithm separately to each such region and so obtained \( \omega \)-surfaces that also have multiple, nonoverlapping regions. However, since these regions are not connected to each other, they cannot be claimed as (approximately) neutrally connected and so should not be identified together as part of the same \( \omega \)-surface. Their only connection is vestigial, handed down from the common surface (a potential density isosurface, say) that initialized the KMJ software. But that initial surface is far less neutral than an \( \omega \)-surface, so the unconnected regions of the KMJ \( \omega \)-surface are not neutrally related with a level of accuracy worthy of \( \omega \)-surfaces. Therefore, we will work with just one connected region of the surface under consideration.

The connected component to work on is that which intersects a user-chosen reference location, with longitude \( \lambda_P \) and latitude \( \phi_P \). In part to ensure the surface does not incrop or outcrop at this reference cast, we replace 6, which KMJ used to make \( \Phi \) have zero mean, with

\[ C[\Phi] = \Phi(\lambda_P, \phi_P). \]  

(12)

which ensures the solution \( \Phi \) is zero at \( (\lambda_P, \phi_P) \). Using 12, the pressure at which the surface intersects the reference cast is fixed, unchanging between iterations. We say that the surface is “pinned” at \( (\lambda_P, \phi_P) \).

Pinning the solution at a reference cast has several other advantages. First, pinning is conceptually simple. In contrast, the requirement for \( \Phi \) to have zero mean does not translate into a simple concept about some mean (potential) density being constant, because the reference pressure associated with the LRPD perturbation \( \Phi \) differs for each water column. Moreover, when the surface is adjusted vertically according to the density perturbation \( \Phi \), it can incrop or outcrop in some places, so the mean \( \Phi \) in the remaining casts need not be 0. Pinning has no issues with incropping and outcropping: the surface cannot incrop or outcrop at the reference cast, it being pinned there. Finally, pinning facilitates comparison between different types of approximately neutral surfaces, as they can all be made to intersect a common point.

3. Computation of \( \omega \)-Surfaces

This section describes the numerical computation of \( \omega \)-surfaces. Each subsection corresponds to a subroutine of the algorithm, presented in the sequence that they are run. Further background on the KMJ algorithm is provided where relevant.
3.1. Numerical Setup and Initialization

Consider the oceanic data as a sequence of casts \( m \) indexed from 1 to \( N_D \). Let \( E_D \) be the number of pairs of adjacent casts. If \( D \) is the maximum number of casts adjacent to any cast (\( D = 4 \) on a typical rectilinear grid) then \( N_D = 1 \leq E_D \leq (D/2)N_D \).

Let \( A_m \) be the area associated with cast \( m \), typically thought of as the area of a tracer cell. Let \( \Delta_m \) be the lateral distance between the cell centers of adjacent casts \( m \) (for “me”) and \( n \) (for “neighbor”), and \( \Delta_{mn} \) the lateral distance of the face between these cells. Let \( A_{m,n} \) be the area associated with the region between adjacent casts \( m \) and \( n \). Figure 1 shows a schematic of this grid. We shall require \( A_{m,n} = \Delta_{m,n} \Delta_m \) (see Appendix D for details). Note that the sum of all \( A_{m,n} \) typically exceeds the sum of all \( A_m \) (by a factor of nearly 2 on a rectilinear grid), since the \( A_{m,n} \) areas overlap.

Let \( S_m(p) \) and \( \Theta_m(p) \) be the salinity and temperature in cast \( m \) as a function of pressure \( p \). These may be provided by linear or other interpolants of discrete data.

The iterative \( \omega \)-surface algorithm is initialized with any approximately neutral surface, such as an isosurface of potential density.

3.2. Surface Properties

Let \( p_m \) be the pressure on the approximately neutral surface in cast \( m \). The salinity and temperature on the surface are denoted by \( S_m \) and \( \Theta_m \), which without arguments are understood as shorthand for the functions \( S_m(p) \) and \( \Theta_m(p) \) evaluated at \( p_m \). A cast \( m \) that intersects the surface is called “wet.” The point of intersection between a wet cast \( m \) and the surface is called a “surface point,” also indexed by \( m \). At other, “dry” casts \( m \), the surface has incropped or outcropped, and \( p_m, S_m, \) and \( \Theta_m \) are NaN (not-a-number) here.

3.3. The Connected Region Containing \((\lambda_v, \varphi_v)\)

The first task is to determine the surface points that form the connected region containing the pinning location \((\lambda_v, \varphi_v)\). This is achieved by a breadth-first search (BFS) through the surface points. A (“first-in, first-out”) queue of surface points is maintained and initialized with the point at \((\lambda_v, \varphi_v)\). All surface points are initially marked as undiscovered. Each step of the BFS pops a surface point from the queue, marks it as discovered, then pushes all its undiscovered neighbors onto the tail of the queue. This is repeated until the queue is empty. Upon completion, all discovered surface points constitute the connected region containing \((\lambda_v, \varphi_v)\).

Let \( m_1, \ldots, m_N \) index these surface points. With \( N \) such surface points, the BFS runs in \( O(N) \) time. Let \( E \) be the number of adjacent pairs of surface points in this region, indexed by \((v_1, v_2), \ldots, (v_E, v_E)\). Because the surface can outcrop and incrop, and because the (initial) surface could have multiple connected components, \( N \leq N_D \) and \( E \leq E_D \). Again, \( N = 1 \leq E \leq (D/2)N \).

3.4. Wetting

The iterative \( \omega \)-surface algorithm naturally allows the surface to incrop or outcrop. This occurs at casts where the vertical update moves the surface above the sea-surface or below the sea-floor (Section 3.7.1) or fails to find a solution in the cast (Section 3.7.2). We say such a cast has become “dry.” In the KMJ algorithm, once a cast was dry, it remained so forevermore. However, subsequent iterations may adjust the surface so that a neutral trajectory exists from a neighboring surface point to this cast. When this happens, the dry cast should “wet.” We now achieve this as follows.

When the above BFS (Section 3.3) is adding neighboring surface points to the queue, it now tests for neutral trajectories to neighboring dry casts. Specifically, suppose the BFS has just popped the surface point \( m \), and
the neighboring cast \( n \) is dry. Two discrete points are considered linked by a neutral trajectory if their potential densities are equal when referenced to the average of the pressures at the two points. Hence, we solve

\[
R \left( S_m(p_m), \Theta_m(p_m), \frac{1}{2}(p_m + p_n) \right) = R \left( S_n(p_n^*), \Theta_n(p_n^*), \frac{1}{2}(p_m + p_n^*) \right)
\]

for \( p_n^* \) using Brent’s method in the interval \([p_n^{\text{top}}, p_n^{\text{bot}}]\) where \( p_n^{\text{top}} \) and \( p_n^{\text{bot}} \) are the top and bottom pressures in cast \( n \), and using an initial guess of \( p_n^* = p_n \) (see Appendix B). If a solution is found, then cast \( n \) becomes wet, so we set \( p_n \) to \( p_n^* \) and add the surface point \( n \) to the BFS queue. Otherwise, cast \( n \) remains dry. (Note, this procedure is equivalent to, but much faster than, a method that repeatedly attempts to wet the perimeter of the current surface, until no new casts are wet.)

If the surface point \( m \) were a different neighbor of \( n \), then the result \( p_n^* \) would differ, due to nonzero neutral helicity. This detail is ignored, because it is handled by the following steps (Sections 3.6 and 3.7).

It can happen that a dry cast wets, then incrops or outcrops at a later stage of the algorithm, cycling in this way indefinitely. A simple way to prevent this is to stop wetting after a user-defined number of iterations (default 5), once the large-scale readjustment is complete. A similar option is provided to only begin wetting at a certain user-defined iteration number (default 1).

### 3.5. Neutrality Error Discretization

The neutrality error \( \varepsilon \), defined by 2, is discretized from a surface point \( m \) to an adjacent surface point \( n \) as

\[
\varepsilon_{m,n} = \frac{(\rho_S)_{m,n}(S_n - S_m) + (\rho_\Theta)_{m,n}(\Theta_n - \Theta_m)}{\Delta_{m,n}},
\]

where

\[
(\rho_S)_{m,n} = \partial_\gamma R \left( \frac{1}{2}(S_m + S_n) + \frac{1}{2}(\Theta_m + \Theta_n), \frac{1}{2}(p_m + p_n) \right)
\]

and similarly for \( (\rho_\Theta)_{m,n} \). This discretization is motivated by the fact that, if the surface in question were perfectly neutral in continuous space between \( m \) and \( n \), then \( \varepsilon_{m,n} \) would be zero up to terms that are cubic in the differences \( S_n - S_m, \Theta_n - \Theta_m, \) and \( p_n - p_m \) (Jackett & McDougall, 1997; Appendix A).

In contrast, the KMJ software discretized \( (\rho_S)_{m,n} \) as

\[
\frac{1}{2} \left( \partial_\gamma R(S_m, \Theta_m, p_m) + \partial_\alpha R(S_n, \Theta_n, p_n) \right)
\]

and similarly for \( (\rho_\Theta)_{m,n} \). One can expand 16 in a Taylor series about \( \left( (S_m + S_n) / 2, (\Theta_m + \Theta_n) / 2, (p_m + p_n) / 2 \right) \) to show that 16 also leads to cubic order errors, but with larger coefficients than if 15 is used.

The form 15 requires \( E \) evaluations of \( \partial_\gamma R \) and \( \partial_\alpha R \), which is typically more than the \( N \) evaluations of each as required by 16. However, this is not the time-limiting step: its time complexity is \( O(E) \), which is \( O(N) \) since \( E \leq (D/2)N \).

### 3.6. Step 1: The Global, Linear Problem

The first step in the \( \omega \)-surface numerical algorithm is now discussed: how to determine the density perturbation \( \Phi_m \) at each surface point \( m \). Figure 2(a) shows an arrangement of casts and lists their adjacent pairs. This example grid will serve to illustrate the matrices in this section.

#### 3.6.1. Discretizing the Constrained Minimization Problem

The discrete form of 4 is
Figure 2. (a) An example ocean grid of uniform spacing ($\Delta_m, m = 1$) and areas ($\hat{\Delta}_m = 1, m = 1$). Land is shaded dark gray. There are $N_0 = 12$ enumerated casts, though the surface has outcropped at three casts (shaded light gray). The surface is to be pinned at cast 7. In the connected region containing this pinning cast, there are $N = 7$ surface points, and $E = 8$ pairs of adjacent casts, as listed. (b) The KMJ matrix problem involves an overdetermined matrix problem relating the gradient of $\Phi$ to $\varepsilon$, augmented by a row ($9 = E + 1$) that constrains the solution, here changed to pin $\Phi_7 = 0$. (c) The divergence matrix is the negative transpose of the gradient matrix. (d) The new formulation of $\omega$-surfaces involves an exactly determined Poisson equation (slightly modified in the seventh row to pin $\Phi_7 = 0$). The ranks of $G_C$, $D_C$, and $L_C$ are all $N = 7$.
where \( P \in \{1, \ldots, N\} \) so that \( m_P \) indexes the surface point at \((\lambda_P, \varphi_P)\).

One way to bring the \( A_{j,x,y} \) inside the square in 17a is to nondimensionalize the grid distances, with \( \hat{\Delta}_{m,n} = \Delta_{m,n} A_{m,n}^{-1/2} \) representing the nondimensional distance between \( m \) and \( n \). Also, let \( \hat{\varepsilon}_{m,n} = A_{m,n}^{1/2} \varepsilon_{m,n} \) having the same units as \( \varepsilon \); equivalently, \( \varepsilon_{m,n} \) is defined by 14 but with \( \Delta_{m,n} \) in the denominator. To top things off, let \( \hat{\Delta}_{m} = A_{m} N / A \) and \( \hat{c} = c / N \). Now, rewrite 17 with matrix–vector operations as

\[
\begin{align*}
\min_{\Phi} \quad & \| \mathbf{G} \hat{\Phi} + \bar{\varepsilon} \|_{2}^{2} \\
\text{subject to} \quad & \mathbf{C} \hat{\Phi} = 0,
\end{align*}
\]  

(20a)

where

- \( \Phi = \left[ \Phi_{m_{1}}, \ldots, \Phi_{m_{N}} \right]^T \),
- \( \bar{\varepsilon} = \left[ \hat{\varepsilon}_{j_{1},j_{2}}, \ldots, \hat{\varepsilon}_{j_{E},j_{F}} \right]^T \),
- \( \| \cdot \|_{2} \) denotes the \( \ell^2 \) norm, for example, \( \| \bar{\varepsilon} \|_{2} = \sqrt{\sum_{j=1}^{E} \hat{\varepsilon}_{j_{1},j_{2}}^2} \),
- \( \mathbf{G} \), representing the discretized gradient operator, is the \( E \times N \) matrix with row \( j \) having \( \hat{\Delta}_{m,j}^{-1} \) in column \( \bar{v}_{j} \) and \( \hat{\Delta}_{j,j}^{-1} \) in column \( u_{j} \),
- \( \mathbf{C} = \mathbf{M} \) to apply 18 or \( \mathbf{C} = \mathbf{P} \) to apply 19, where
- \( \mathbf{M} = \hat{c} \hat{\Delta}_{m_{1},j_{1}} \ldots \hat{\Delta}_{m_{N},j_{F}} \) is the \( 1 \times N \) matrix representation of the discretized area-averaging operator, and
- \( \mathbf{P} = [0, \ldots, 0, \hat{c}, 0, \ldots, 0] \) with the \( \hat{c} \) occupying entry \( P \), is the \( 1 \times N \) matrix for evaluation at the pinning cast.

Continuing the example from Figure 2(a), the top eight rows of Figure 2(b) show the matrices \( \mathbf{G} \) and the vectors \( \Phi \) and \( \bar{\varepsilon} \).

### 3.6.2. The KMJ Unconstrained Minimization Problem

KMJ defined the \( \omega \)-surface algorithm as an unconstrained minimization problem,

\[
\min_{\Phi} \| \mathbf{G}_{c} \hat{\Phi} + \bar{\varepsilon}_{0} \|_{2}^{2},
\]  

(21)

where \( \mathbf{G}_{c} = \left[ \mathbf{G} \quad \mathbf{C} \right] \) and \( \bar{\varepsilon}_{0} = \left[ \bar{\varepsilon} \quad 0 \right] \). Equation 21 is commonly expressed as an overdetermined (having more rows than columns, \( E + 1 > N \)) matrix problem, \( \mathbf{G}_{c} \hat{\Phi} = -\bar{\varepsilon}_{0} \). In general, this matrix problem is inconsistent (having no exact solution), so the least squares solution \( \hat{\Phi} \) is sought (i.e., solving 21), which KMJ found using the LSQR algorithm (Paige & Saunders, 1982).

Since \( \mathbf{G}_{c} \) is overdetermined, one might wonder whether 20 and 21 are equivalent. They are. The solution \( \hat{\Phi} \) to 21 satisfies \( \mathbf{C} \hat{\Phi} = 0 \) 20b exactly (if it did not, subtract the scalar \( \mathbf{C} \hat{\Phi} \) from \( \hat{\Phi} \), making \( \mathbf{C} \hat{\Phi} = 0 \) without changing \( \hat{\Phi} \)), thereby reducing \( \| \mathbf{G}_{c} \hat{\Phi} + \bar{\varepsilon}_{0} \|_{2} \). Hence, also \( \| \mathbf{G}_{c} \hat{\Phi} + \bar{\varepsilon}_{0} \|_{2} = \| \mathbf{G} \hat{\Phi} + \bar{\varepsilon} \|_{2} \) so 20a is met.

KMJ set \( \mathbf{C} = \mathbf{M} = [1, \ldots, 1] \), which in our parlance sets \( \hat{c} = 1 \) and all \( \hat{\Delta}_{m} = 1 \). Moreover, KMJ cancelled the factors \( \hat{\Delta}_{m,j}^{-1} \) in each row of \( \mathbf{G}_{c} \hat{\Phi} = -\bar{\varepsilon}_{0} \) effectively setting all grid distances and areas to unity. We note that neglecting the grid sizes reweights the equations and does alter the solution, since \( \mathbf{G}_{c} \hat{\Phi} = -\bar{\varepsilon}_{0} \) is over-determined and inconsistent. For example, two \( \omega \)-surfaces computed with OCCA data (see Section 4), one accounting for area weights and one not, differ in depth by a root-mean-square value of 0.60 m. For comparison, the depth difference between one of these \( \omega \)-surfaces and a topobaric surface has a root-mean-square value of 1.35 or 1.54 m.

Figure 2(b) shows the matrix problem \( \mathbf{G}_{c} \hat{\Phi} = -\bar{\varepsilon} \) for the running example, with \( \mathbf{C} \) pinning surface point \( P = 7 \).
3.6.3. The Normal Equations

The least squares solution to an overdetermined matrix problem is given by the so-called normal equations, which involve multiplication by the matrix’s transpose. In our case, $G_C\Phi = -\epsilon_0$ is left-multiplied by $D_C = -G_C^T$, yielding

$$L_C\Phi = -D_C\epsilon_0,$$

(22)

where $L_C = D_CG_C$ is a full-rank, square $(N\times N)$ matrix, so (22) offers an exact solution, namely $\Phi = -(L_C)^{-1}D_C\epsilon_0$. Figures 2(c) and 2(d) show $D_C$ and (22), respectively, for the running example.

Rather than inverting $L_C$ however, (22) is most efficiently solved by the Cholesky decomposition, which factors

$$L_C = TT^T,$$

(23)

where $T$ is a lower triangular matrix and $T^T$ its upper triangular transpose. Then, (22) is written $T\tilde{y} = -D_C\tilde{\epsilon}$ and rapidly solved for $\tilde{y}$ by forward substitution, upon which $T^T\Phi = \tilde{y}$ is rapidly solved for $\Phi$ by back substitution.

Another major advantage of constraining $\Phi$ by pinning now reveals itself. Note that $L_C = L - C^T C$, where $L = DG$ and $D = -G^T$, and that $L$ is sparse. When $C = P$, $L_C$ remains sparse because $P^T P$ is the $N$ by $N$ matrix of zeros except for $\epsilon^2$ in entry $(P, P)$. Thus, a sparse Cholesky solver can be used. In contrast, if $C = M$, then $L_C$ is fully dense because $M^T M = \epsilon^2[\bar{A}_{ij} \bar{A}_{ij}]_{i,j}$. If one wished to use $C = M$, then first solve for $\Phi$ using $C = P$ with $P = 1$ (pinning the first surface point) and subsequently update $\Phi$ by subtracting its mean, $M\Phi$.

A slight modification ensures $\Phi_{np} = 0$ identically rather than just up to machine precision. We replace the $P$th equation with $\Phi_{np} = 0$ by setting row $P$ of $L_C$ to 0 except with 1 in column $P$, and setting element $P$ of $D\epsilon$ to 0. (This is equivalent to adding all other rows to row $P$.) Since $\Phi_{np} = 0$ now, the off-diagonal values in column $P$ of $L_C$ are irrelevant, and thus also set to 0, in order to maintain symmetry of $L_C$. In the Figure 2(d) example, this manipulation would turn the seventh row and seventh column of $L_C$ to 0, except for a 1 placed in the $(7, 7)$ entry.

Section 4 will show that this change—solving the global, linear problem by Cholesky factorization rather than LSQR—dramatically reduces the computational complexity of the $\omega$-surface algorithm.

3.6.4. Connection to the Discretized Poisson Equation

We can arrive at the discretized Poisson equation (22) two ways. First, we can discretize the minimization problem 4 and then apply the normal equations, as above. Then, we will recognize that $D = -G^T$ is the matrix representation of the discretized divergence operator, which further implies that $L = DG$ is the matrix representation of the discretized Laplacian operator, and hence (22) is a discretized Poisson equation. The key insight here is that the gradient and negated divergence operators are, under certain boundary conditions, adjoints of one another—as are their discretized versions (Appendix D). Since the latter operate between finite-dimensional spaces, choosing bases for the spaces induces matrix representations of the operators. We chose bases such that $G$ represents the discretized gradient operator. Since the discretized negated divergence operator is the adjoint of the discretized gradient operator, the matrix representing the former ($-D$) is the (conjugate) transpose of the matrix representing the latter ($G$).

Second, we could convert the minimization problem 4 into the Poisson equation 11, then discretize the latter. The result would be the same matrix problem as obtained via the normal equations, above. Indeed, it is easily seen that 11 yields equations like the fourth row in the example problem (Figure 2(d)) for interior points and like the other rows for boundary points. To better understand the source charge manipulation that converts the inhomogeneous problem 8 into the homogeneous problem 11, one may consider introducing additional $\Phi$ and $\epsilon$ just past the boundary, and then discretizing 8a, noticing that the contribution of these boundary terms cancels identically, thanks to the natural boundary condition 8b.
3.7. Step 2: Vertically Updating the Surface

3.7.1. The Linearized KMJ Implementation

KMJ converted the density perturbation \( \Phi_m \), for each surface point \( m \), into a pressure perturbation

\[
\Delta p_m = \lambda \Phi_m / \frac{d\sigma_m}{dp}(p_m),
\]

(24)

where \( \sigma_m(p) = R(S_m(p), \Theta_m(p), p_m) \) is the LRPD in cast \( m \). To increase the algorithm’s robustness at the possible expense of speed, KMJ inserted the damping factor \( \lambda \) in (24), empirically chosen as \( \lambda = 0.2 \) and applied a minimum threshold of \( \Delta \sigma_m / \Delta p \geq 3 \times 10^{-4} \text{ kg m}^{-3} \text{ dbar}^{-1} \) (roughly corresponding to a squared buoyancy frequency greater than \( 3 \times 10^{-6} \text{ s}^{-2} \)) to avoid division by small numbers. Then, KMJ moved the surface from pressure \( p_m \) to \( p_m + \Delta p_m \). Essentially, the KMJ method is a linearization, like a single iteration of Newton’s method, of the nonlinear problem described next.

3.7.2. The Decoupled, Nonlinear Vertical Adjustment

Rather than moving the surface vertically by (24), which essentially assumes that the stratification is a constant function of pressure in each cast, we find the precise pressure \( p_m^* \) at which the LRPD differs from that on the surface by \( \Phi_m \), as described in Section 2.3. Numerically, we solve for \( p_m^* \) in

\[
R(S_m(p_m^*), \Theta_m(p_m^*), p_m^*) = R(S_m(p_m), \Theta_m(p_m), p_m) + \Phi_m.
\]

(25)

using Brent’s method with an initial guess of \( p_m^* = p_m \) and bounds of \([p_m^{\text{top}}, p_m^{\text{bot}}]\) (Appendix B). Upon solving 25, the surface point \( m \) is moved from pressure \( p_m \) to \( p_m^* \).

The reference pressure for potential density in 25 is the current pressure, \( p_m \). Using a reference pressure of \( \frac{1}{2}(p_m + p_m^*) \) is theoretically preferable and slightly more accurate, but both choices converge to the same solution (when \( \Phi_m = 0 \) for all \( m \)). In practice, using \( p_m \) is slightly faster because the LHS of 25 can be pre-computed, so each iteration of Brent’s method requires only one evaluation of \( R \).

3.8. Iteration

The entire procedure, from Sections 3.2 to 3.7, is iterated until convergence. The KMJ software iterated a fixed, user-chosen number of times. We now iterate for a user-specified minimum number of iterations (default 1), until the mean absolute value of \( \Phi \), namely \( \sqrt{N^{-1} \sum_m \Phi_m^2} \), drops below a user-specified threshold (default \( 10^{-7} \text{ kg m}^{-3} \), corresponding to about \( \pm 0.01 \text{ dbar} \) in relatively weakly stratified regions of buoyancy frequency \( N \approx 3 \times 10^{-4} \text{ s}^{-2} \)). We also provide an exit condition when the root-mean-square of the change in pressure between consecutive iterations drops below a user-specified threshold (default 0, i.e. deactivated by default). These exit conditions can be easily reconfigured as needed.

4. Numerical Tests and Comparisons of Approximately Neutral Surfaces

4.1. Data

We analyze realistic ocean data using two reanalysis products, namely the 2004–2006 OCCA climatology (Forget, 2010) and ECCO2 (Menemenlis et al., 2005) averaged over December 22–24, 2002. These data sets have \( 1^\circ \times 1^\circ \) and \( \frac{10^\circ}{4} \times \frac{15^\circ}{4} \) horizontal resolution, respectively, making for \( N_{OCC} = 36,792 \) and \( N_{ECC} = 670,039 \) casts.

The OCCA and ECCO2 data sets employ the Boussinesq version of the Jackett & McDougall (1995) equation of state: a function of Practical Salinity, potential temperature, and depth. The latter is immediately converted, inside the Boussinesq equation of state function, to pressure by a factor \( -\epsilon g \rho_c \), where \( \epsilon = 10^{-4} \text{ dbar Pa}^{-1} \), \( g = 9.81 \text{ m s}^{-2} \) is the gravitational acceleration, and \( \rho_c = 1,027.5 \text{ kg m}^{-3} \) is the Boussinesq reference density. Henceforth, we reemploy \( R \) to denote this Boussinesq equation of state.
Furthermore, to probe the computational complexity of competing algorithms, we generate a synthetic aquaplanet (with nonperiodic boundaries and limited to 80°S to 80°N), with \( n \) oceanic grid points per horizontal dimension (making \( N_0 = n^2 \) casts) and 50 grid points in the vertical dimension, as follows. The depth of successive grid points increases cubically: for \( 1 \leq k \leq 50 \), grid point \( k \) has depth \(((k - 1)/49)^3(4,000 \, \text{m})\). We presume the model is also Boussinesq, with the same gravity and reference density as OCCA and ECCO2 above. The surface temperature is set to Gaussian noise with 2°C standard deviation, now with standard deviation of 0.1, and smoothed as above. The 3D salinity is again initialized as the surface temperature rescaled into the range [34, 36] and [36, 36.5], respectively. To make the neutral helicity nonzero, the surface salinity is augmented by another field increasing \( k \). Furthermore, to probe the computational complexity of competing algorithms, we generate a synthetic aquaplanet (with nonperiodic boundaries and limited to 80°S to 80°N), with \( n \) oceanic grid points per horizontal dimension (making \( N_0 = n^2 \) casts) and 50 grid points in the vertical dimension, as follows. The depth of successive grid points increases cubically: for \( 1 \leq k \leq 50 \), grid point \( k \) has depth \(((k - 1)/49)^3(4,000 \, \text{m})\). We presume the model is also Boussinesq, with the same gravity and reference density as OCCA and ECCO2 above. The surface temperature is set to Gaussian noise with 2°C standard deviation, now with standard deviation of 0.1, and smoothed as above. The 3D salinity is again initialized as the surface temperature rescaled into the range [34, 36] and [36, 36.5], respectively. To make the neutral helicity nonzero, the surface salinity is augmented by another field increasing \( k \).

### 4.2. Computing Surfaces

For each data set, we compute several types of approximately neutral surfaces. All surfaces are built to interpose surfaces of potential density (\( \sigma \)-surfaces), referenced to depth \( z_{\text{ref}} = 1,500 \, \text{m} \), are calculated as follows.

\[
\sigma_c = R(S_p(z_p), \Theta_p(z_p), z_{\text{ref}}).
\]

The depth of the isosurface is first approximated by linearly interpolating depth as a function of \( \sigma \) in each cast, then evaluating this interpolant at \( \sigma_c \). The resulting surface is then used as an initial guess to solve

\[
\mathcal{R}(S_m(z_m), \Theta_m(z_m), z_{\text{ref}}) = \sigma_c
\]

for \( z_m \) in each cast \( m \), using Brent’s method bounded in \([z_{m,\text{top}}, z_{m,\text{bottom}}]\) where \( z_{m,\text{top}} \) and \( z_{m,\text{bottom}} \) are the shallowest and deepest depths in cast \( m \). That is, we (linearly) interpolate \( S \) and \( \Theta \), not density.

b) Isosurfaces of in situ density anomaly (\( \delta \)-surfaces) with reference salinity and temperature values chosen as \( S_{\text{ref}} = S_p(z_p) \) and \( \Theta_{\text{ref}} = \Theta_p(z_p) \), respectively, are calculated similarly. The isosurface value is

\[
\delta_c = R(S_p(z_p), \Theta_p(z_p), z_{\text{ref}}) - R(S_{\text{ref}}, \Theta_{\text{ref}}, z_{\text{ref}}) = 0.
\]

The isosurface’s depth is first approximated by linearly interpolating depth as a function of \( \delta \) in each cast, then evaluating this interpolant at \( \delta_c \). The resulting surface serves as an initial guess to solve for \( z_m \) in

\[
\mathcal{R}(S_m(z_m), \Theta_m(z_m), z_{\text{ref}}) - \mathcal{R}(S_{\text{ref}}, \Theta_{\text{ref}}, z_{\text{ref}}) = \delta_c
\]

using Brent’s method as above.

c) Topobaric surfaces (\( \tau \)-surfaces) are computed with the software of Stanley (2019a). The topobaric code has been updated in tandem, most notably to have wetting capacity (Section 3.4), to use Brent’s method (rather than the bisection method as originally used) for its vertical update, and to have additional diagnostics.

d) Three algorithms for \( \omega \)-surfaces are used.

i) \( \omega_{\text{KMI}} \)-surfaces are calculated using the KMI software, only minimally changed to use the Boussinesq version of the Jackett & McDougall (1995) equation of state for consistency with all other results herein, and to add the necessary diagnostics. The KMI code was not updated to apply pinning, nor to use the correct discretization of \( \omega \) with 15.

ii) \( \omega_s \)-surfaces use the modern \( \omega \)-surface software of this paper, including solving the discretized Poisson equation 22 using Cholesky factorization.

iii) \( \omega_{\text{S}} \)-surfaces are a step between the preceding two, being like \( \omega_s \) but solving the gradient matrix problem 21 using LSQR.
Iterative algorithms (ωKSvD, ωV, ω+, and τ) are initialized from the best noniterative algorithm available, namely δ-surfaces, and are deemed “converged” when the root-mean-square of the depth difference between consecutive iterations drops below 1 mm. Brent’s method, used by all algorithms except ωKSvD, proceeds until a solution is found accurate to 0.1 mm. The LSQR algorithm (used by ωKSvD and ωV) uses a relative error tolerance of 10⁻⁶ (smaller values increase execution time at the expense of accuracy).

The computation time for each surface is recorded. (To mitigate aberrations apparent in the timings at low grid resolution, we repeat the calculation max(1, ceil(log₂(1,024²/N₀))) times and average the results.) All computations are performed on a single core of an Intel i7-8850H CPU clocking 2.60 GHz, typical of a modern personal laptop. To focus on the core algorithms, these computations are done without wetting (Section 3.4). Including wetting does not significantly change the timings, it being a fast algorithm that scales linearly, as O(N₀).

4.3. Veronis Density and ρψ Labels for the Surfaces

Here, we report the two density labels from Appendix A for the above surfaces. The reference location used to evaluate the two density labels is the pinning cast P. Since all our surfaces intersect this cast at a common depth of z₀ = 1,500 m, the same density labels apply to all our surfaces. (That said, the isosvalues of potential density and in situ density anomaly provide a more useful characterization of those surfaces.) The reference time for the first method, and in general the time at which data are sampled, is unambiguous, since we only use data from one time step of each model.

First, using the Boussinesq version of the method in Section A1, our surfaces have a Veronis density value of 1,027.7700 and 1,027.7864 kg m⁻³ in OCCA and ECCO2, respectively, using upper depth z₁00 = 5 m, reference depth z₀ = 0 m, and reference location (180.5°E, 0.5°N) for OCCA and (180.125°E, 0.125°N) for ECCO2.

Second, using the method of Section A2, our surfaces have a Jackett & McDougall (1997) ρψ neutral density value of 1,027.7827 and 1,027.7832 kg m⁻³ in OCCA and ECCO2, respectively. The former is ρψ evaluated with OCCA’s data at (188.5°E, 3.5°S, 1,500 m), with pressure evaluated by linearly interpolating OCCA’s output pressure variable at (188.5°E, 3.5°S) to 1,500 m. The latter is ρψ evaluated with ECCO2’s data at (188.125°E, 3.875°S, 1,500 m), with pressure (missing from ECCO2’s archived output) evaluated by integrating hydrostatic balance from 0 to 1,500 m, with zero atmospheric pressure to obtain the “sea pressure”: p₀ = g₀(z₀) ρ(188.125°E,3.875°S, z₀) dz₀, numerically computed by trapezoidal integration, consistent with ECCO2’s internals.

4.4. Results

Figure 3 shows the depth of the ω-surface, and of the other surfaces by way of comparison to the ω-surface’s depth, using ECCO2 data. All these surfaces intersect the cast at (180.125°E, 0.125°N) at 1,500 m depth on December 22–24, 2002, and hence have the same Veronis density label, yet clearly they differ from each other globally. The topobaric surface remains within about 10 m of the ω-surface, reflecting that these are both highly accurate approximately neutral surfaces. However, the potential density surface and in situ density anomaly surface deviate by 200 m or more vertically from the ω-surface in certain regions, mostly in the Southern Ocean and North Atlantic. The Jackett & McDougall (1997) neutral density surface deviates from the ω-surface by up to about 80 m, with notable deviations occurring globally. Differences in the depth of various surfaces effect similar differences in tracers—such as salinity, temperature, potential vorticity, or biogeochemical tracers—on the various surfaces.

The accuracy and computational speed of various neutral surface algorithms is summarized in two figures. Figure 4 shows neutrality errors (∥ε∥₂) versus CPU time for each algorithm as iterations proceed in (a) the 128 × 128 aquaplanet or (b) the OCCA data set. Figure 5 shows the CPU time required for convergence over a range of grid resolutions, for each algorithm.

Wereport in terms of the area-weighted ℓ² norm of ε, namely ∥ε∥₂ = \(\sqrt{\sum_{i,j} A_{ij} \frac{\tilde{Z}_{ij}^2}{\epsilon_{ij}^2}}\), where \(A_{ij} = \sum_{k=1}^{E} A_{ij,k}\), because that is what ω-surfaces (currently) minimize. However, a potentially more
relatable quantification is the fictitious dianeutral diffusivity, $D_f$, which is the dianeutral diffusivity generated by isopycnal mixing acting along the surface in question rather than along the neutral tangent plane (McDougall & Jackett, 2005). Mathematically, $D_f = K s \cdot s = K (g \rho^{-1} N^2)^2 \epsilon \cdot \epsilon$, where $s$ is the difference between the slope of the neutral tangent plane and the slope of the surface in question. Taking representative values for the isopycnal diffusivity $K = 1.000 \text{ m}^2 \text{s}^{-1}$, the gravitational acceleration $g = 10 \text{ m s}^{-2}$, the density $\rho = 10^3 \text{ kg m}^{-3}$, and the buoyancy frequency $N = 10^{-3} \text{ s}^{-1}$, then a neutrality error of $|\epsilon| = 10^{-8} \text{ kg m}^{-4}$ yields a fictitious dianeutral diffusivity of $10^{-5} \text{ m}^2 \text{s}^{-1}$. Keeping $D_f$ generally below $10^{-5} \text{ m}^2 \text{s}^{-1}$ is important to prevent modeling errors from overwhelming the real, physically supported dianeutral diffusion. Of course, this back-of-the-envelope scaling does not capture strong spatial variation of $\|\epsilon\|$ or $D_f$; for maps and histograms, see Figures 5 and 6 of Stanley (2019a).

The fastest algorithms are for $\sigma$- and $\delta$-surfaces. These are not iterative; they scale linearly with the number of surface points as they simply solve one nonlinear equation per cast. The former is slightly faster than the
latter, requiring one rather than two calls to the equation of state per step of Brent’s method. However, \(\sigma\) and \(\delta\)-surfaces are relatively far from neutral (Figure 4).

The KMJ code is by far the slowest method. The updated \(\omega\) algorithm is dramatically faster but achieves the same time complexity of \(O(N^{1.6})\) (Figure 5), as both algorithms share the same bottleneck: the LSQR solver. The acceleration of \(\omega_{\text{KMJ}}\) into \(\omega\) is due, in roughly equal measure, to (i) an order of magnitude reduction in the number of iterations required for convergence due to the new vertical update method (Section 3.7) and (ii) general coding improvements, particularly with regard to forming connected components, building the sparse matrix, and compiling time-sensitive subroutines.

In the OCCA test, the new vertical update method enables the \(\omega\) algorithm to converge robustly, whereas the \(\omega_{\text{KMJ}}\) algorithm does not converge after 50 iterations (Figure 4(b)). In fact, it does not converge even after 200 iterations, being repeatedly afflicted by iterations that increase \(\|\varepsilon\|_2\) (not shown), an example of which appears near minute 4 of Figure 4(b). Decreasing the damping parameter from \(\lambda = 0.2\) to \(\lambda = 0.1\) did not alleviate the issue.

The slightly less neutral result of \(\omega_{\text{KMJ}}\) relative to \(\omega\) or \(\omega_{+}\) in Figure 4(a) is because we measure \(\|\varepsilon\|_{\text{area}}\), using the correct expansion coefficients (Section 3.5) and with area-weighting (Section 3.6).

The LSQR solver is considerably slower when the matrix’s nonzero entries vary in magnitude. On a latitude–longitude grid, \(G\) contains entries that are the square root of the ratio between the zonal and meridional

---

Figure 4. The area-weighted \(\ell^2\) norm of the neutrality error at each iteration, as a function of CPU time, in (a) a synthetic aquaplanet with 128 \(\times\) 128 casts or (b) the OCCA data set with 360 \(\times\) 160 casts, for the following neutral surface algorithms: potential density surfaces (\(\sigma\)), in situ density anomaly surfaces (\(\delta\)), neutral density surfaces (\(\gamma_n\), only for (b)), the KMJ algorithm (\(\omega_{\text{KMJ}}\)), the KMJ algorithm with all improvements except the Poisson formulation (\(\omega\)), the present algorithm (\(\omega_{+}\)), and topobaric surfaces (\(\tau\)). Iterative algorithms (\(\omega_{\text{KMJ}}, \omega, \omega_{+}, \tau\)) stop when the root-mean-square of the depth change of the surface between consecutive iterations drops below 1 mm, except for \(\omega_{\text{KMJ}}\) in (b) which is halted after 50 iterations. The neutrality error and CPU time for the final iteration of each algorithm are listed. Note that timings for \(\gamma_n\) and \(\omega_{\text{KMJ}}\) are in minutes, not seconds.
grid distances and others that are the inverse of this ratio. As such, the \( \omega \) algorithm stalled on ECCO2 data, which extends to the north pole, and so is not reported. It is possible to relax the error tolerance of the LSQR solver such that \( \omega \) can proceed on ECCO2 data, but doing so would degrade the resulting surface’s quality, and is not pursued further.

The \( \omega \) algorithm remains slower than the topobaric (\( \tau \)-surface) algorithm. The time-limiting step in the topobaric algorithm is the computation of the Reeb graph using ReCon, with time complexity \( O(N \log N + sN) \), where \( s \) is the number of saddles of pressure on the surface (Doraiswamy & Natarajan, 2013). Figure 5 shows that the \( \tau \)-surface computation time scales faster than \( N \log N \) (dashed green line), so evidently \( s \) scales somewhat faster than \( \log N \) in the synthetic ocean. If the real ocean exhibits \( s \sim \log N \), then topobaric surfaces would have time complexity \( O(N \log N) \), the time-limiting step being to sort \( N \) numbers. The supposition that \( s \sim \log N \) does not seem to be inconsistent with the OCCA and ECCO2 tests, though clearly more evidence is required.

Figure 5. The computation time for various algorithms, as in Figure 4, versus the number of surface points \( N \). Data are from a random synthetic aquaplanet (lines and circles), OCCA (triangles), or ECCO2 (stars). The numbers nearby each data point indicate the number of iterations required for each iterative algorithm to be deemed converged. However, for accurate scaling comparison, \( \omega_{\text{KMJ}} \), \( \omega \), and \( \tau \) data are plotted with the time these algorithms require to complete precisely five iterations. The \( \omega_{\text{KMJ}} \) algorithm was halted at 50 iterations without converging on OCCA data. The LSQR-based algorithms (\( \omega_{\text{KMJ}} \) and \( \omega \)) stalled on ECCO2 data, which covers the north pole where zonal grid distances go to zero. The \( \gamma \) algorithm was not run on the synthetic aquaplanet data. An asymptotic scaling for each algorithm’s CPU time with \( N \) is provided (right) and plotted (dashed lines). This is omitted for \( \sigma \), being almost indistinguishable from that for \( \delta \).
Finally, the new Poisson-based algorithm ($\omega_+$) achieves the highest accuracy of all algorithms (identical to the gradient-based $\omega_\gamma$ algorithm) and is faster even than the topobaric algorithm, with a similar if not slightly superior time complexity of roughly $O(N^{1.2})$. The Cholesky solver of $\omega_+$ is not slowed by nonuniform grid distances and provides exact solutions (so does not possess any error tolerance parameter).

We emphasize that the $O(N^{1.6})$ and $O(N^{1.2})$ scalings for $\omega_\gamma$ and $\omega_+$ are empirical and may not even have the correct form, but they roughly hold over the range of $N$ typically encountered in modern ocean models. In particular, the sparse Cholesky solver first reorders the matrix to reduce the number of nonzero entries in its triangular factors. This choice of reordering minimizes the solver’s memory footprint but does not necessarily minimize the number of floating point operations (Luce & Ng, 2014). Hence, analytic expressions for the time complexity of sparse Cholesky algorithms are rare. Nonetheless, detailed inspection of the $\omega_\gamma$ and $\omega_+$ algorithms’ matrix solver subroutines confirms these scalings work well over grid sizes ranging from $N = 64^2$ to $N = 2048^2$ (not shown).

5. Conclusion

A fluid parcel moving laterally also tends to move vertically so that its density matches that of its environment, where no buoyant forces disturb the parcel from its position in the water column. The set of all points reached by a fluid parcel moving in this way, and mixing with its environment as it goes, would ideally form a 2D surface, called a neutral surface. However, seawater’s nonlinear equation of state implies that neutral surfaces do not exist except in idealized scenarios and hence can only be approximated in the real ocean. Even still, interior oceanic motion is overwhelmingly directed along, rather than across, approximately neutral surfaces. However, many such surfaces are not sufficiently accurate: diffusion that is deemed to be along the surface actually contains a dianeutral component that can exceed the true and often miniscule, dianeutral diffusion (KMJ; Stanley, 2019a). The most accurate approximately neutral surface to date is the $\omega$-surface (KMJ), but its practical utility was hampered by the slow execution speed and occasional instability of the original algorithm. We have addressed both of these issues and added new features like wetting and labeling.

To accelerate the $\omega$-surface algorithm, we modified the algorithm’s first step, which solves a global, linear problem to find an optimal density perturbation. Instead of solving this overdetermined matrix problem by a least squares method (LSQR), we convert it into Poisson’s equation, the discretized form of which we solve by a sparse Cholesky factorization. This reduces the time complexity of the algorithm from roughly $O(N^{1.6})$ to roughly $O(N^{1.2})$, where $N$ is the number of grid points in the surface. Speed improvements are therefore greater for larger grids.

To stabilize the $\omega$-surface algorithm, we modified the algorithm’s second step, which updates the vertical position of the surface. The KMJ algorithm essentially treated the stratification as constant in the vertical direction, and weak stratification would lead to unrealistically large vertical adjustments of the surface. Now, we solve a nonlinear equation in each cast, determining the pressure at which the locally referenced potential density is increased by the optimal density perturbation found in the first step. This change also reduces the number of iterations required for the overall $\omega$-surface algorithm to converge by an order of magnitude, further accelerating the algorithm.

We also added a “wetting” capacity, which ensures the final surface extends as far toward its outcropping lines as it should. With wetting, the surface can be incropped or outcropped at a particular cast at one iteration, then rejoin the ocean interior at subsequent iterations if there is a neutral link to this cast from the surface at a neighboring cast.

Furthermore, we offer a method to label an $\omega$-surface (or any approximately neutral surface) with a density value, which may be a Veronis (1972) density constructed within the oceanic data set containing the surface, or may be a Jackett & McDougall (1997) $\gamma^n$ neutral density value.

On a $128 \times 128$ grid (tiny by modern standards), the original KMJ algorithm required over 2 min. For the $10^\circ \times 10^\circ$ ECCO2 data set, 50 iterations of the KMJ $\omega$-surface algorithm should require about 11 h, based on extrapolation of Figure 5—a generous estimate that ignores how nonuniform grid distances further slow
the KMJ algorithm. We have dramatically accelerated the $\omega$-surface algorithm, such that solutions for these cases are now found in about 0.3 and 15 s, respectively.

Our improvements render the $\omega$-surface algorithm faster than the topobaric surface algorithm of Stanley (2019a), which leveraged recent and fast tools from computational topology to handle the global nature of the neutral surface problem (whereas $\omega$-surfaces handle this aspect as a global matrix problem). Topobaric surfaces retain the advantage that they are locally pycnotropic and hence can be made to possess an exact geostrophic streamfunction (Stanley, 2019b).

With the global step of the $\omega$-surface algorithm reformulated as a Poisson equation, $\omega$-surfaces may now be suitable to employ in an online, parallel ocean general circulation model. Parallelized methods to solve Poisson’s equation, such as preconditioned conjugate gradient (PCG), have already been implemented in many ocean models to maintain zero divergence of the barotropic flow (Marshall, Adcroft, et al., 1997). On a single CPU, MATLAB’s Cholesky solver is significantly faster than MATLAB’s PCG solver for computing $\omega$-surfaces (not shown). However, PCG solvers may perform better in parallel. Moreover, PCG solvers, being iterative, can operate on very large grids when Cholesky solvers, being direct, may encounter memory limitations.

To run the new $\omega$-surface software, first arrange hydrographic data for salinity, temperature, and pressure (or depth, if Boussinesq) as multidimensional arrays with the vertical dimension contiguous in memory (first, in MATLAB). Then, calculate the pressure (or depth) of an initial approximately neutral surface, such as an isosurface of potential density. Choose a reference cast; the final $\omega$-surface will intersect this reference cast at the same pressure (or depth) that the initial surface did, and it will be a single connected region containing this reference cast. Adjust any additional parameters (such as what vertical interpolant to use, the grid distances, and the exit conditions) from their defaults, as desired. You are now ready to compute an $\omega$-surface.

Appendix A: Labeling an $\omega$-Surface With a Density Value

We present two simple methods to label an $\omega$-surface (or any approximately neutral surface) with a density value. In essence, both methods make a neutral trajectory from a point on the surface to a reference cast, where a density value is evaluated and assigned to the whole surface. The density chosen is what Veronis (1972) defined as the “local potential density,” but we henceforth call the “Veronis density.” It is the sea-surface in situ density plus the vertically integrated vertical gradient of LRPD, that is, the vertical component of $N$. As Veronis (1972) noted, this variable is only meaningful in the vertical direction, where it perfectly represents the stratification; its lateral variation, however, is not particularly meaningful, being remotely influenced by hydrographic variations shallower in the water column. Indeed, isosurfaces of the Veronis density are far from neutral. Hence, the Veronis density is used only to label a reference cast with density values, which are then laterally distributed along approximately neutral surfaces.

The first method takes the reference cast from the same oceanic data set in which one’s $\omega$-surfaces lives, and therefore the resulting Veronis density value is only internally consistent with this data set. This first method is mostly intended for studies in the vein of geophysical fluid dynamics, where the layer thickness or potential vorticity needs accurate assessment but there is no need to compare the precise numerical values of the Veronis density against other published values. If, instead, one needs to compare the surface’s density value with published Jackett & McDougall (1997) $\gamma_n$ values, use the second method, which essentially takes the reference cast from the Levitus (1982) atlas at (188°E, 4°S).

A1. Veronis Density on $\omega$-Surfaces

In the first method, the only data used are the oceanic data set in which one’s $\omega$-surfaces lives. This may be 3D spatial data, or 4D data with one time dimension. Select a reference longitude $\lambda_V$, reference latitude $\varphi_V$, and reference time $t_V$. Let $p_0^{\text{ref}}$ be the shallowest pressure value in the cast at $(\lambda_V, \varphi_V, t_V)$ and $p_{\text{ref}} = 0 \text{ dbar}$ (though other choices are possible for both). Now, consider an $\omega$-surface at time $t_0$, which need not equal $t_V$. Let $p_0$ be the pressure at which this surface intersects the cast at $(\lambda_V, \varphi_V, t_0)$. Figure A1 provides a schematic illustrating this procedure.
The first step is a temporal neutral trajectory. We usually think of a neutral trajectory across space at a fixed time, but now we consider a neutral trajectory across time at a fixed water column in space. The goal is to discover \( p_V \), the pressure at which a temporal neutral trajectory from the bottle at \((\lambda_V, \phi_V, p_0, t_0)\) intersects the reference cast \((\lambda_V, \phi_V, t_V)\). If \( t_V = t_0 \), then \( p_V = p_0 \), so jump to step two. Otherwise, let \( t_0, \ldots, t_N \) be the sequence of \( N + 1 \) time steps from \( t_0 \) to \( t_V = t_N \). Let \( S_n(p) \) and \( \Theta_n(p) \) be the salinity and temperature as functions of pressure in the cast at \((\lambda_V, \phi_V, t_n)\). Now, calculate a sequence of discretized temporal neutral trajectories by solving for \( p_n \) in

\[
R\left( S_{n-1}(p_{n-1}), \Theta_{n-1}(p_{n-1}), \frac{1}{2}(p_{n-1} + p_n) \right) = R\left( S_n(p_n), \Theta_n(p_n), \frac{1}{2}(p_{n-1} + p_n) \right),
\]

iteratively from \( n = 1 \) to \( n = N \). Again, we solve \( A1 \) numerically using Brent’s method, with \( p_{n-1} \) as an initial guess. Letting \( p_V = p_N \), a (neutral) density value for the final bottle at \((\lambda_V, \phi_V, p_V, t_V)\) is therefore appropriate for the bottle at \((\lambda_V, \phi_V, p_0, t_0)\) on the \( \omega \)-surface, and hence for the entire \( \omega \)-surface.

The second step calculates this density value on the reference cast: the Veronis density, \( \rho_V \), at \((\lambda_V, \phi_V, p_V, t_V)\).

Mathematically, it is

\[
\text{Figure A1. Schematic illustration for the Veronis density labeling of an } \omega \text{-surface.}
\]

Three water columns are shown, each at reference longitude \( \lambda_V \) and latitude \( \phi_V \), but at different times, \( t_0, t_1, \) and \( t_2 \). The latter time is also the reference time, \( t_V \). An \( \omega \)-surface that intersects the \((\lambda_V, \phi_V, t_V)\) water column at pressure \( p_V \) is selected for labeling. A sequence of temporal neutral trajectories (dashed lines, left) link \((\lambda_V, \phi_V, p_0, t_0)\) to \((\lambda_V, \phi_V, p_1, t_1)\) and then to \((\lambda_V, \phi_V, p_2, t_2)\). These trajectories determine the pressure \( p_V = p_2 \) at which the Veronis density \( \rho_V \) is calculated on the reference cast \((\lambda_V, \phi_V, t_V)\). Because the temporal links are neutral trajectories, the Veronis density \( \rho_V \) serves as a neutral density label for each point along the temporal neutral trajectory, as well as any \( \omega \)-surface intersecting the temporal neutral trajectory. Note that the three \( \omega \)-surfaces shown at different times are neutrally linked only at \((\lambda_V, \phi_V)\); another temporal neutral trajectory (dashed line, right) from a different water column on the \( \omega \)-surface at time \( t_0 \) will, in general, reach the same water column at time \( t_1 \) above or below the \( \omega \)-surface at time \( t_1 \).
\[ \rho_V = \mathcal{R}(S_V(p_L^{\text{top}}), \Theta_V(p_L^{\text{top}}), p_{\text{ref}}) + \int_{p_L^{\text{top}}}^{p} \rho_{SV}(p) \frac{dS_V}{dp}(p) + \rho_{\Theta V}(p) \frac{d\Theta_V}{dp}(p) \, dp. \] (A2)

where \( S_V(p) \) and \( \Theta_V(p) \) provide the salinity and temperature as functions of pressure down the reference cast at \( (\lambda_V, \varphi_V, t_V) \), and where \( \rho_{SV}(p) = \partial_S \mathcal{R}(S_V(p), \Theta_V(p), p) \), and similarly for \( \rho_{\Theta V}(p) \). The integral in A2 is computed numerically by trapezoidal integration with a maximum interval size of 1 dbar, by default.

By assigning the Veronis density from A2 to the entire \( \omega \)-surface, as well as to every \( \omega \)-surface intersecting (or neutrally linked to) the cast at \( (\lambda_V, \varphi_V, t_V) \), one can create a 3D (or 4D) density field. The isosurfaces of this density field are \( \omega \)-surfaces, hence it is reasonable to write \( \omega = 1,027.0 \text{ kg m}^{-3} \), for example, to specify a certain \( \omega \)-surface—though the parameters \( \lambda_V, \varphi_V, t_V, p_L^{\text{top}}, \) and \( p_{\text{ref}} \) must also be specified. Had this procedure been applied using topobaric surfaces instead, for example, the resulting density field would be denoted by \( \tau \) and its isosurfaces would be topobaric surfaces.

The above method is somewhat rudimentary, since temporal neutral links are handled only at one location, \( (\lambda_V, \varphi_V) \). At other locations, \( \omega \)-surfaces sharing the same Veronis density label are not temporally neutrally linked (Figure A1, right dashed line). This is a spatiotemporal version of the problem with the early construction of an approximately neutral surface by McDougall (1987), which (essentially) began by calculating a zonal neutral trajectory, off of which meridional neutral trajectories were made (see also Figure 1 of You & McDougall [1990]). That surface had zero meridional neutrality error but had nonzero zonal neutrality error everywhere except the initial zonal trajectory. Methods to minimize the overall spatial and temporal neutrality errors are the subject of future work.

We emphasize that the Veronis density label is not to be interpreted as a value of the Jackett & McDougall (1997) neutral density variable \( \gamma^t \), because no connection is made to the reference data set underlying \( \gamma^t \).

### A2. Jackett & McDougall (1997) \( \gamma^t \) Labels for \( \omega \)-Surfaces

If the user desires a neutral density label that can be interpreted as a Jackett & McDougall (1997) \( \gamma^t \) label, the following method can be used. Simply evaluate the Jackett & McDougall (1997) \( \gamma^t \) neutral density variable at \( (S_L(p_L), \Theta_L(p_L), p_L, \lambda_L, \varphi_L) \). Here, \( (\lambda_L, \varphi_L) \) is a reference longitude and latitude that intersects the \( \omega \)-surface, and \( L \) indexes this cast in the user’s data set, so that \( (S_L(p_L), \Theta_L(p_L), p_L) \) is the salinity, temperature, and pressure where the surface intersects \( (\lambda_L, \varphi_L) \).

The time dimension is not explicitly mentioned in this method, because it is irrelevant: the neutral link is made directly to the Levitus (1982) reference cast, regardless of the surface’s timestamp. That said, this method should not be applied to paleoclimate or long-term climate change simulations, because the Jackett & McDougall (1997) \( \gamma^t \) variable loses accuracy as the ocean state drifts from the Levitus (1982) ocean atlas. Relatedly, an \( \omega \)-surface that exists only north of 64°N, where \( \gamma^t \) is not defined, cannot be labeled by this second method.

When reporting this neutral density label for one’s surface, one must specify the values \( \lambda_L \) and \( \varphi_L \) that were used. The choice of this position is not automated, but the user should select \( \lambda_L \) and \( \varphi_L \) to be as near to \( (188°E, 4°S) \) as possible.

To understand this method better, suppose \( (\lambda_L, \varphi_L) = (188°E, 4°S) \) exactly. With this choice, the \( \gamma^t \) software will calculate a discrete neutral trajectory from \( (S_L(p_L), \Theta_L(p_L), p_L) \) to the reference cast at \( (188°E, 4°S) \) in the Levitus (1982) data set, ending at pressure \( p^* \), say. This neutral trajectory from one data set to another can be thought of like a temporal neutral trajectory, as per Section A1. Then, the prelabeled \( \gamma^t \) field on the Levitus (1982) reference cast is interpolated to \( p^* \), providing a \( \gamma^t \) value for the user’s surface.

On a technical note, the prelabeled \( \gamma^t \) field was initialized by Jackett & McDougall (1997) as the Veronis density on the \( (188°E, 4°S) \) reference cast. That is, they initialized with the integrating factor \( b \) set to unity at \( (188°E, 4°S) \). However, they subsequently employed a relaxation method, which slightly adjusted \( b \). As such, the second method does not simply perform the first method using the Levitus (1982) reference cast, but rather simply calls the Jackett & McDougall (1997) software, which also ensures the same numerics and equation of state are used.
If \((\lambda_l, \varphi_l) \neq (188^\circ E, 4^\circ S)\) exactly, then the \(\gamma^m\) software calculates four neutral trajectories from \((S_l(p_r), \Theta_l(p_r), p_r)\) to each of the four nearest casts in the Levitus (1982) data set. Then, the prelabeled \(\gamma^m\) field on these four casts is interpolated vertically to the pressure at which they intersect the neutral trajectories. The final \(\gamma^m\) label for the user’s bottle \((S_l(p_r), \Theta_l(p_r), p_r)\), and hence the user’s surface, is taken by bilinearly interpolating the four \(\gamma^m\) values, according to where \((\lambda_l, \varphi_l)\) fits between the four casts.

With \((\lambda_l, \varphi_l) \neq (188^\circ E, 4^\circ S)\), the point to note is that the \(\gamma^m\) values on these four surrounding casts would ideally get their values by neutral trajectories back to the Levitus (1982) reference cast at \((188^\circ E, 4^\circ S)\), but because of nonzero neutral helicity in the Levitus (1982) data set, the pressure at which such neutral trajectories intersect the \((188^\circ E, 4^\circ S)\) reference cast depends on the path taken. Instead, the \(\gamma^m\) values on the four surrounding casts are determined by virtue of being on approximately neutral surfaces, and where these surfaces intersect the \((188^\circ E, 4^\circ S)\) reference cast at which \(\gamma^m\) was originally labeled. Since these approximately neutral surfaces were constructed by the Jackett & McDougall (1997) technology, the \(\gamma^m\) label for bottles far from the reference cast is less accurate than it would be if the present \(\omega\)-surface technology had (anachronistically) been used. Thus, \((\lambda_l, \varphi_l)\) should be chosen as \((188^\circ E, 4^\circ S)\) or as near to it as possible. Moreover, this is why we do not advocate for a method such as evaluating \(\gamma^m\) at all points on one’s \(\omega\)-surface and (somehow) averaging the results.

Appendix B: Brent’s Method With an Initial Guess and Bounds

To solve \(f(x) = 0\) for some continuous function \(f : [a, b] \rightarrow \mathbb{R}\), we employ Brent’s (1973) method. Like bisection, this is a bracketed root-finding method that is guaranteed to find a root inside an interval \([a', b']\) provided \(f(a') f(b') < 0\), that is, \(f\) changes sign somewhere in \([a', b']\). Brent’s method is faster than bisection, by a factor of roughly 2 for our purposes of solving equations like 13 and 25.

Given an initial guess \(x_0\) for the root, we follow MATLAB’s fzero function to construct an interval \([a', b']\) \(\subset [a, b]\) having \(f(a') f(b') < 0\) by geometrically expanding (by a factor of \(\sqrt{2}\)) a small interval containing \(x_0\) until a sign-change is found; if the limits \([a, b]\) are reached and no sign-change was found, Brent’s method is not executed and no root is returned.

Appendix C: Conditions for Exact Solutions to the Overdetermined Matrix Problem

We here consider the conditions under which an exact solution exists to the overdetermined matrix problem, \(G_C \Phi = -\hat{c}_p\). First consider the continuous case, supposing

\[
\int_{\mathcal{C}} \mathbf{e} \cdot d\mathbf{\ell} = 0
\]  
(C1)

for any closed loop \(\mathcal{C}\) in the surface. By the converse of the gradient theorem, \(\varepsilon = -\nabla \Phi\) for some scalar field \(\Phi\). Subtract \(C[\Phi]\) from \(\Phi\). Then \(\Phi\) solves 4, with \(\varepsilon(\Phi) = 0\), exactly.

Returning to the discrete case, exact solutions exist if (and only if)

\[
\sum_{i=1}^{f} \hat{c}_{c_i} = 0
\]  
(C2)

for every cycle of surface points \(c_0, \ldots, c_f\) with \(c_0 = c_f\) and \(c_{f+1} = c_1\). Condition C2 allows \(\Phi\) to be constructed by setting \(\Phi_{n_0} = 0\), say, and then initializing a BFS from node \(m_0\), setting \(\Phi_n = \Phi_m - \varepsilon_{m,n} \Delta_{m,n}\) where \(m\) is the parent of \(n\) in the BFS. Condition C2 ensures that the resulting \(\Phi\) is independent of the actual search tree used (and changes uniformly by a constant if a different root node is chosen). Again, \(C[\Phi]\) is then subtracted from \(\Phi\). The result is that the cost function in 17a is 0, exactly.

A special case of C2 occurs when there are no cycles. A meridional section is one such example, but more generally this case with no cycles occurs when the casts are arranged as a tree, in the language of graph theory. Having no cycles requires \(N = E + 1\), meaning \(G_C\) is square and full-rank and hence provides an
exact solution. Indeed, this case permits an exact neutral surface, built from neutral trajectories emanating from a particular cast.

One might wonder if condition C2 requires that the helicity on the surface be zero. It need not be. In an ocean with nonzero helicity everywhere, there exist surfaces that satisfy C2. To see this, consider three casts, with one pair linked by a neutral trajectory, and the other two pairs linked by nonneutral trajectories with equal and opposite neutrality errors, so C2 holds. By varying the nonneutrality of the latter two links, these three links can be made to constitute a neutral helix with either a positive or negative pitch. The intermediate value theorem ensures that there is some amount of nonneutrality of the latter two links such that this pitch is zero. We have thus created a valid (discretized) surface, satisfying C2, in an ocean of nonzero helicity.

**Appendix D: Gradients and Divergences as Adjoints**

Here, we first show that the gradient and negated divergence operators in continuous space are adjoints of one another. Then, we show that the same holds in our discretized space, which means that the matrix representing the negated divergence is the (conjugate) transpose of the matrix representing the gradient.

Let \( f : A \rightarrow \mathbb{R} \) and \( g : A \rightarrow \mathbb{R}^2 \) be continuously differentiable scalar and 2D vector fields, respectively. Consider integrating \( \nabla \cdot (\mathbf{g}) \) over the domain \( A \). Manipulating this either with the divergence theorem or the product rule yields

\[
\oint_{\partial A} f \, \mathbf{g} \cdot \hat{n} \, d\mathbf{l} = \iint_{A} \nabla \cdot \mathbf{g} \, dA + \iint_{A} f \nabla \cdot \mathbf{g} \, dA. \tag{D1}
\]

Suppose the LHS is zero, say because \( \mathbf{g} \cdot \hat{n} = 0 \) on \( \partial A \). Then, D1 becomes

\[
\langle \nabla f, \mathbf{g} \rangle_{H^2} = \langle f, -\nabla \cdot \mathbf{g} \rangle_{H^2}, \tag{D2}
\]

where \( \langle f_a, f_b \rangle_{H^1} = \iint_{A} f_a(x) f_b(x) \, dx \) and \( \langle g_a, g_b \rangle_{H^2} = \iint_{A} g_a(x) \cdot g_b(x) \, dx \) are inner products, which make the complete vector spaces \( H_1 = \{ f : A \rightarrow \mathbb{R} \mid f \text{ is } C^1 \} \) and \( H_2 = \{ g : A \rightarrow \mathbb{R}^2 \mid g \text{ is } C^1, g(x) \cdot \hat{n}(x) = 0 \ \forall x \in \partial A \} \) into Hilbert spaces. The gradient is a linear operator \( \nabla : H_1 \rightarrow H_2 \), while the divergence is a linear operator \( \nabla \cdot : H_2 \rightarrow H_1 \). Equation D2 is the definition for two operators—here, the negated divergence and the gradient—to be adjoints of each other.

We now show this adjoint relationship holds in our discretized system. First, let us finish nondimensionalizing the grid. As a reminder, the nondimensional distance between cells \( m \) and \( n \) is \( \Delta_{m,n} = A_{m,n} \Delta_{m,n}^{-1/2} \). Similarly, let \( \Delta_{m,n}^\perp = A_{m,n} \Delta_{m,n}^{-1/2} \) be the nondimensional lateral distance of the face between adjacent cells \( m \) and \( n \). The adjoint relationship will require that \( A_{m,n} = \Delta_{m,n} \Delta_{m,n}^{-1} \) exactly. This nondimensionalizes to define \( \hat{A}_{m,n} = \Delta_{m,n} \hat{A}_{m,n}^\perp = 1 \). Recall, the tracer cell areas \( A_m \) were nondimensionalized to \( \hat{A}_{m,n} \) which need not be unity.

Now, the discretized divergence of a 2D gridded vector field is defined as the net flux out of the grid cell, divided by the area of that cell; the outward flux across each grid face is the vector component normal to the face multiplied by the face length. Mathematically,

\[
D \hat{\mathbf{e}} = \left[ \frac{1}{A_{m,n}} \sum_{j=1}^{E} (\hat{\mathbf{e}}_{i,j} - \hat{\mathbf{e}}_{i,j}) \hat{\mathbf{e}}_{i,j} \right]_{j=1,...,N} \tag{D3}
\]

where \( \delta \) is the Kronecker delta (\( \delta_{ij} = 1 \) if \( i = j \), otherwise \( \delta_{ij} = 0 \)). Boundary terms do not appear in D3 because the component of \( e \) normal to the boundary was set to zero (Section 3.6.4). In similar notation, the action of the discretized gradient operator \( \mathbf{G} \) on \( \Phi \) is

\[
\mathbf{G} \Phi = \left[ \frac{\Phi_{i,j} - \Phi_{i,j}}{\hat{A}_{i,j}} \right]_{j=1,...,E}. \tag{D4}
\]
The vectors $\hat{\Phi}$ and $\hat{c}$ live in Hilbert spaces $H_1 = \mathbb{R}^N$ and $H_2 = \mathbb{R}^F$, respectively, equipped with inner products

$$\langle f, g \rangle_{H_1} = \sum_{i=1}^{N} \hat{A}_{m,i} f_i g_i,$$

$$\langle f, g \rangle_{H_2} = \sum_{j=1}^{E} \hat{A}_{j,v} f_j g_j.$$  \hfill (D5a) \hfill (D5b)

We can now directly verify that

$$\langle (G \hat{\Phi}, \hat{c}) \rangle_{H_2} = \sum_{j=1}^{E} \hat{A}_{j,v} \left[ \Phi_{v} - \Phi_{u} \right]_{\Delta_{u,j,v}}$$

$$= \sum_{j=1}^{E} \hat{A}_{j,v} \left[ \sum_{i=1}^{N} (\delta_{m,j,v} - \delta_{m,j,uv}) \Phi_{m,i} \right]$$

$$= \sum_{j=1}^{E} \hat{A}_{j,v} \left[ \sum_{i=1}^{N} \delta_{m,j,v} \Phi_{m,i} \right]$$

$$= \sum_{i=1}^{N} \hat{A}_{m,i} \Phi_{m,i} \left[ 1 - \hat{A}_{m,i} \sum_{j=1}^{E} (\delta_{m,j,v} - \delta_{m,j,uv}) \Delta_{u,j,v} \right]$$

$$= \langle (\hat{\Phi}, -D\hat{c}) \rangle_{H_1}. $$  \hfill (D6)

The third equality required $\hat{A}_{m,n} = \Delta_{m,n}$. We have thus shown that the operators represented by $-D$ and $G$ are (conjugate) transposes of one another, and hence the matrices $-D$ and $G$ are (conjugate) transposes of one another.

Removing hats from the variables yields the dimensional equivalent. This does change the inner product (since $A_{m,n}$ is nonuniform, whereas $\hat{A}_{m,n} = 1$), but also changes the gradient and divergence operators, such that the adjoint relationship holds.

Data Availability Statement

OCCA and ECCO2 data are available from https://www.ecco-group.org/products.htm. Code to reproduce the results herein is available at http://doi.org/10.5281/zenodo.4615747.

Acknowledgments

We thank Sjoerd Groeskamp, Andreas Klocker, a third anonymous reviewer, and Stephen Griffies for many helpful comments. All authors acknowledge support from the Australian Research Council through grant FL150100090.

References

Brent, R. P. (1973). Algorithms for minimization without derivatives. Englewood Cliffs, NJ: Prentice-Hall.

de Szoeker, R. A. (2000). Equations of motion using thermodynamic coordinates. Journal of Physical Oceanography, 30(11), 2814–2829. https://doi.org/10.1175/1520-0485(2000)030<2814:EOMUTO>2.0.CO;2

de Szoeker, R. A., & Springer, S. R. (2009). The materiality and neutrality of neutral density and orthobaric density. Journal of Physical Oceanography, 39(8), 1779–1799. https://doi.org/10.1175/2009JPO4042.1

de Szoeker, R. A., Springer, S. R., & Oxilia, D. M. (2000). Orthobaric density: A thermodynamic variable for ocean circulation studies. Journal of Physical Oceanography, 30(11), 2830–2852.

Doraiswamy, H., & Natarajan, V. (2013). Computing Reeb graphs as a union of contour trees. IEEE Transactions on Visualization and Computer Graphics, 19(2), 249–262. https://doi.org/10.1109/TVCG.2012.115

Eden, C., & Willebrand, J. (1999). Neutral density revisited. Deep Sea Research Part II: Topical Studies in Oceanography, 46(1), 33–54. https://doi.org/10.1016/S0967-0645(98)00013-1

Forget, G. (2010). Mapping ocean observations in a dynamical framework: A 2004–06 ocean atlas. Journal of Physical Oceanography, 40(6), 1201–1221. https://doi.org/10.1175/2010JPO4043.1

Groeskamp, S., Barker, P. M., McDougall, T. J., Abernathey, R. P., & Griffies, S. M. (2019). VENM: An algorithm to accurately calculate neutral slopes and gradients. Journal of Advances in Modeling Earth Systems, 11, 1917–1939. https://doi.org/10.1029/2019MS001613

IOC, SCOR, & IAPSO. (2010). The international thermodynamic equation of seawater—2010: Calculation and use of thermodynamic properties. UNESCO Intergovernmental Oceanographic Commission, Manuals and Guides, 56, 196 pp.

Iselin, C. O. D. (1939). The influence of vertical and lateral turbulence on the characteristics of the waters at mid-depths. Eos, Transactions American Geophysical Union, 20(3), 414. https://doi.org/10.1029/TR020i003p00414

Jackett, D. R., & McDougall, T. J. (1995). Minimal adjustment of hydrographic profiles to achieve static stability. Journal of Atmospheric and Oceanic Technology, 12(2), 381–389. https://doi.org/10.1175/1520-0426(1995)012<0381:MAHPT>2.0.CO;2

Jackett, D. R., & McDougall, T. J. (1997). A neutral density variable for the world's oceans. Journal of Physical Oceanography, 27(2), 237–263. https://doi.org/10.1175/1520-0485(1997)027<0237:ANDVFT>2.0.CO;2

Klocker, A., McDougall, T. J., & Jackett, D. R. (2009). A new method for forming approximately neutral surfaces. Ocean Science, 5(2), 155–172. https://doi.org/10.5194/os-5-155-2009
Lang, Y., Stanley, G. J., McDougall, T. J., & Barker, P. M. (2020). A pressure-invariant neutral density variable for the world’s oceans. *Journal of Physical Oceanography, 50*, 3585–3604. https://doi.org/10.1175/JPO-D-19-0321.1

Levitus, S. (1982). Climatological atlas of the world ocean. NOAA Professional Paper, 13, 1–173.

Luce, R., & Ng, E. G. (2014). On the minimum FLOPs problem in the sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications, 35*(1), 1–21.

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. *Journal of Geophysical Research, 102*(C3), 5753. https://doi.org/10.1029/96JC02775

Marshall, J., Hill, C., Perelman, L., & Adcroft, A. (1997). Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *Journal of Geophysical Research, 102*(C3), 5733–5752. https://doi.org/10.1029/96JC02776

McDougall, T. J. (1987). Neutral surfaces. *Journal of Physical Oceanography, 17*(11), 1950–1964. https://doi.org/10.1175/1520-0485(1987)017<1950:NS>2.0.CO;2

McDougall, T. J., & Jackett, D. R. (1988). On the helical nature of neutral trajectories in the ocean. *Progress in Oceanography, 20*(3), 153–183. https://doi.org/10.1016/0079-6611(88)90001-8

McDougall, T. J., & Jackett, D. R. (2005). An assessment of orthobaric density in the global ocean. *Journal of Physical Oceanography, 35*(11), 2054–2075. https://doi.org/10.1175/JPO2796.1

McDougall, T. J., & Jackett, D. R. (2007). The thinness of the ocean in S–Θ–p space and the implications for mean diapycnal advection. *Journal of Physical Oceanography, 37*(6), 1714–1732. https://doi.org/10.1175/JPO3114.1

Menemenlis, D., Hill, C., Adcroft, A., Campin, J.-M., Cheng, B., Ciotti, B., et al. (2005). NASA supercomputer improves prospects for ocean climate research. *Eos, Transactions American Geophysical Union, 86*(9), 89. https://doi.org/10.1029/2005EO090002

Paige, C. C., & Saunders, M. A. (1982). *LSQR: An algorithm for sparse linear equations and sparse least squares*. ACM Transactions on Mathematical Software, 8(1), 43–71. https://doi.org/10.1145/355984.355989

Reid, J. L., & Lynn, R. J. (1971). On the influence of the Norwegian–Greenland and Weddell seas upon the bottom waters of the Indian and Pacific oceans. *Deep Sea Research and Oceanographic Abstracts, 18*(11), 1063–1088. https://doi.org/10.1016/0011-7471(71)90094-5

Riha, S. (2016). *Methods for forming approximately neutral surfaces in the ocean* (Doctoral dissertation). Sydney, NSW, Australia: University of New South Wales.

Stanley, G. J. (2019a). Neutral surface topology. *Ocean Modelling, 138*, 88–106. https://doi.org/10.1016/j.ocemod.2019.01.008

Stanley, G. J. (2019b). The exact geostrophic streamfunction for neutral surfaces. *Ocean Modelling, 138*, 107–121. https://doi.org/10.1016/j.ocemod.2019.04.002

Tailleux, R. (2013). Available potential energy density for a multicomponent Boussinesq fluid with arbitrary nonlinear equation of state. *Journal of Fluid Mechanics, 735*, 499–518. https://doi.org/10.1017/jfm.2013.509

Veronis, G. (1972). On properties of seawater defined by temperature, salinity, and pressure. *Journal of Marine Research, 30*(2), 227.

Young, W. R. (2010). Dynamic enthalpy, conservative temperature, and the seawater Boussinesq approximation. *Journal of Physical Oceanography, 40*(2), 394–400. https://doi.org/10.1175/2009JPO4294.1