Chapter 2

Building blocks of the thermohaline ocean circulation model

2.1 Introduction

The stability analysis for the global ocean circulation is performed with a computer program called THCM, which stems from thermohaline circulation model. The code is a product of the cooperation of a number of people at different institutes in the last few years and involves physical oceanography, dynamical system theory, numerical analysis and computer science. The work in this thesis is mainly in the field of numerical analysis, but in this chapter we will describe the building blocks of the model, including the blocks from the other fields.

We start with the equations that govern the ocean flow in Section 2.2. This is the most physics based part of the thesis. We treat both the analytic and discrete equations. Section 2.3 describes numerical continuation, a technique from dynamical system theory, that allows us to compute the effect of changes in parameters on the solution and perform linear stability analysis. This technique requires the Jacobian matrix of the discrete ocean equations. The construction of this matrix, its structure and some crucial properties are treated in Section 2.4. We pay attention to a subproblem of our system, that is the saddle point problem, which occurs also in many other flow equations.

2.2 The global ocean circulation

We give in a few lines the partial differential equations that describe the thermohaline ocean circulation. The word thermohaline means that the flow is driven by heat and salinity fluxes at the ocean-atmosphere interface. There are six quantities that play an important role: the flow velocity in three directions (u, v, w), the pressure (p), salinity (S) and the temperature (T). All variables are functions of space and time in a spherical coordinate system (φ, θ, z), where 

\[ z = r - r_0 \]

the deviation from the reference earth radius \( r_0 \).
2.2.1 Model formulation

We consider flows in a spherical sector bounded by longitudes $\phi_e$ and $\phi_w$ ($-180 \leq \phi_e < \phi_w \leq 180$) and by latitudes $\theta_s$ and $\theta_n$ ($-90 < \theta_s < \theta_n < 90$). In case of the global ocean circulation we have $\phi_e = -180$ and $\phi_w = 180$ and the boundary conditions at the east and west boundary are periodic. Continents are allowed in the domain as well as a bottom topography (see Figure 2.1 for an example of the grid). So the ocean basin has a variable depth $D(\phi, \theta)$ and is bounded vertically by $z = -D(\phi, \theta)$ and a nondeformable ocean-atmosphere boundary $z = 0$. The flows in this domain are forced by a heat flux $Q_H$ (in Wm$^{-2}$), a zonal wind stress field $(\tau^\phi, \tau^\theta)$ (in Pa) and a virtual salt flux $Q_S$ (in ms$^{-1}$), which includes evaporation and fresh water fluxes due to melting ice, rivers and precipitation. The heat flux $Q_H$ is proportional to the temperature difference between the sea-surface temperature $T$ and a prescribed atmospheric temperature $T_S$, i.e.,

$$Q_H = -\lambda_T(T - T_S) \quad (2.1)$$

where $\lambda_T$ (in Wm$^{-2}$K$^{-1}$) is a constant exchange coefficient. The virtual salt flux $Q_S$ is similarly formulated as a restoring condition and is given by

$$Q_S = -\lambda_S(S - S_S) \quad (2.2)$$

where $\lambda_S$ (in ms$^{-1}$) is an exchange coefficient. Both wind and buoyancy forcing are distributed as a body forcing, implicitly defined by (2.1) and (2.2), over the first (upper) layer of the ocean having a thickness $H_m$.

Temperature and salinity differences in the ocean cause density differences according to

$$\rho = \rho_0(1 - \alpha_T(T - T_0) + \alpha_S(S - S_0)) \quad (2.3)$$

where $\alpha_T$ and $\alpha_S$ are the volumetric expansion coefficients and $T_0$, $S_0$ and $\rho_0$ are reference quantities. We use the Boussinesq approximation, which means that density differences have
2.2 The global ocean circulation

Effect only in combination with $g$, the acceleration due to gravity. Density variations in the horizontal momentum and continuity equations can be ignored. Under this approximation the density is equal to the constant reference density $\rho_0$, except in the vertical momentum equation where we use (2.3). Furthermore we use the hydrostatic approximation, i.e. accelerations in the vertical are negligible, which reduces the vertical momentum equation to $\frac{\partial \rho}{\partial z} = -\rho g$. With $r_0$ and $\Omega$ being the radius and angular velocity of the Earth, the governing equations for the zonal ($u$), meridional ($v$) and vertical velocity ($w$) and the dynamic pressure $p$ (the hydrostatic part has been subtracted) become

$$\frac{Du}{dt} - uv \tan \theta - 2\Omega v \sin \theta + \frac{1}{\rho_0 r_0 \cos \theta} \frac{\partial p}{\partial \phi} = A_v \frac{\partial^2 u}{\partial z^2} + A_H L_u(u,v) + \frac{\tau_0}{\rho_0 H_m} \tau^\theta G(z)$$  (2.4a)

$$\frac{Dv}{dt} + u^2 \tan \theta + 2\Omega u \sin \theta + \frac{1}{\rho_0 r_0} \frac{\partial p}{\partial \theta} = A_v \frac{\partial^2 v}{\partial z^2} + A_H L_v(u,v) + \frac{\tau_0}{\rho_0 H_m} \tau^\theta G(z)$$  (2.4b)

$$\frac{\partial w}{\partial z} + \frac{1}{r_0 \cos \theta} \left( \frac{\partial u}{\partial \phi} + \frac{\partial (v \cos \theta)}{\partial \theta} \right) = 0$$  (2.4c)

$$\frac{DT}{dt} - R(\rho)T = \frac{(T_S - T)}{\tau_T} G(z)$$  (2.4d)

$$\frac{DS}{dt} - R(\rho)S = \frac{(S_S - S)}{\tau_S} G(z)$$  (2.4f)

where $G(z) = \mathcal{H}(z/H_m + 1)$ and $\mathcal{H}$ is a continuous approximation of the Heaviside function. Furthermore, $C_p$ is the constant heat capacity and $\tau_T = \rho_0 C_p H_m/\lambda_T$ and $\tau_S = H_m/\lambda_S$ are the surface adjustment time scales of heat and salt, respectively. In addition,

$$\frac{D}{dt} = \frac{\partial}{\partial t} + \frac{u}{r_0 \cos \theta} \frac{\partial}{\partial \phi} + \frac{v}{r_0 \cos \theta} \frac{\partial}{\partial \theta} + w \frac{\partial}{\partial z}$$

$$L_u(u,v) = \nabla_H^2 u + \frac{u}{r_0^2 \cos^2 \theta} - \frac{2 \sin \theta}{r_0^2 \cos^2 \theta} \frac{\partial v}{\partial \phi}$$

$$L_v(u,v) = \nabla_H^2 v + \frac{v}{r_0^2 \cos^2 \theta} + \frac{2 \sin \theta}{r_0^2 \cos^2 \theta} \frac{\partial u}{\partial \phi}$$

$$\nabla_H^2 = \frac{1}{r_0^2 \cos \theta} \left[ \frac{\partial}{\partial \phi} \left( \frac{1}{\cos \theta} \frac{\partial}{\partial \phi} \right) + \frac{\partial}{\partial \theta} \left( \cos \theta \frac{\partial}{\partial \theta} \right) \right]$$

In the equations (2.4a-b), $A_H$ and $A_V$ are the horizontal and vertical momentum (eddy) viscosity which we will take constant. The operator $R(\rho)$ in (2.4e-f) represent the mixing of heat and salt and will be discussed in the next subsection.
\[2\Omega = 1.4 \cdot 10^{-4} \quad [s^{-1}] \quad r_0 = 6.4 \cdot 10^6 \quad [m] \]
\[\tau_T = 7.5 \cdot 10^1 \quad [days] \quad \tau_S = 7.5 \cdot 10^1 \quad [days] \]
\[\alpha_T = 1.0 \cdot 10^{-4} \quad [K^{-1}] \quad \alpha_S = 7.6 \cdot 10^{-4} \quad [-] \]
\[C_p = 4.2 \cdot 10^3 \quad [J/(kgK)^{-1}] \quad \rho_0 = 1.0 \cdot 10^3 \quad [kgm^{-3}] \]
\[A_V = 1.0 \cdot 10^{-3} \quad [m^2 s^{-1}] \quad A_H = 1.0 \cdot 10^4 \quad [m^2 s^{-1}] \]

Table 2.1: Standard values of parameters used in the numerical calculations.

Slip conditions are assumed at the bottom boundary, while at all lateral boundaries no-slip conditions are applied. At all lateral boundaries and the bottom boundary, the heat flux is zero. As the forcing is represented as a body force over the first layer, slip and no-flux conditions apply at the ocean surface. Hence, the boundary conditions are

\[
z = -D, 0 : \quad \frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = w = \frac{\partial T}{\partial z} = \frac{\partial S}{\partial z} = 0, \tag{2.5a}
\]
\[
\phi = \phi_w, \phi_e : \quad u = v = w = \frac{\partial T}{\partial \phi} = \frac{\partial S}{\partial \phi} = 0, \tag{2.5b}
\]
\[
\theta = \theta_s, \theta_n : \quad u = v = w = \frac{\partial T}{\partial \theta} = \frac{\partial S}{\partial \theta} = 0. \tag{2.5c}
\]

In case of periodic boundary conditions

\[
\phi = \phi_w = 180, \phi_e = -180 : \quad u_w = u_e, v_w = v_e, w_w = w_e, T_w = T_e, S_w = S_e. \tag{2.6}
\]

Parameters that are fixed in the calculations are the same as in typical large-scale low-resolution ocean general circulation models and their values are listed in Table 2.1. Other parameter values will be specified in the sections where we present the results of numerical experiments.

### 2.2.2 Tracer mixing representation

To avoid spurious diapycnal mixing (i.e. non-physical mixing in the direction of the density gradient) in the model of the previous section, in Equations (2.4f-g) so-called neutral physics fluxes have to be specified. Background and details of these fluxes and of the so-called small slope approximation used can be found in [36]. Expressions are summarized in section 14.3 of [36] and in our notation the mixing operator becomes

\[
R(\rho) = \begin{bmatrix}
\nabla_H^T \frac{\partial}{\partial z} \\
\eta M(K_H + \eta G \kappa)S^T \\
\eta M(K_H - \eta G \kappa)S \\
\eta M K_H S^T S + K_V
\end{bmatrix}
\begin{bmatrix}
\nabla_H \\
\eta M(K_H - \eta G \kappa)S \\
\eta M K_H S^T S + K_V
\end{bmatrix}
\tag{2.7}
\]

where \(K_H\) is the neutral diffusivity and \(\kappa\) is the [31] skew diffusive mixing coefficient. In addition, the neutral slope vector \(S\) is given by

\[
S = \frac{\nabla_H \rho}{\partial \rho/\partial z} \tag{2.8}
\]
with \( \rho \) as in (2.3).

Note that we have introduced two homotopy parameters \( \eta_M \) and \( \eta_G \) to be able to continue smoothly between constant horizontal diffusivity (\( \eta_M = 0 \)) and neutral mixing (\( \eta_M = 1 \)) and from no meso-scale eddy representation (\( \eta_G = 0 \)) to the \([31]\) representation (\( \eta_G = 1 \)); the latter is referred below as GM-mixing.

We can only apply the representation above when the stratification is stable and when the slope of the isopycnals is small. To limit the slopes, we apply a similar procedure as in \([15]\) by defining \( \alpha_m \) as a maximum permissible slope angle and then multiplying \( \eta_M \) by the function \( f(\frac{\partial \rho}{\partial z}) \), where

\[
\begin{align*}
  f(x) &= 0 \quad , \quad x \leq -\xi \text{ or } x > 0 \quad (2.9a) \\
  f(x) &= 3\left(\frac{x + \xi}{\delta}\right)^2 - 2\left(\frac{x + \xi}{\delta}\right)^3 \quad , \quad -\xi \leq x \leq -\xi + \delta \quad (2.9b) \\
  f(x) &= 1 \quad , \quad -\xi + \delta \leq x \leq -\delta \quad (2.9c) \\
  f(x) &= 1 - 3\left(\frac{x + \delta}{\delta}\right)^2 + 2\left(\frac{x + \delta}{\delta}\right)^3 \quad , \quad -\delta \leq x \leq 0 \quad (2.9d)
\end{align*}
\]

where

\[
\xi = \frac{|\nabla H \rho|}{\tan \alpha_m} \quad (2.10)
\]

and \( \delta = 0.05 \tan \alpha_m \). In this way, the mixing coefficients are smoothly tapered to zero in regions where the slope becomes too large or where the stratification is physically unstable.

The coefficient \( K_V \) in (2.7) is the vertical mixing coefficient. Whereas in many ocean-climate models, usually \( K_V \) is taken constant (or spatially prescribed), too high values (in the order of \( 10^{-4} \) m\(^2\)s\(^{-1}\)) were needed to obtain a realistic strength of the meridional overturning. Mixing in a stratified ocean, however, requires a transfer from kinetic to potential energy. The available energy for mixing, say \( e \), is a topic of current research \([88]\) and it is supplied by the wind, the tides and buoyancy forcing. An overview of the processes responsible for the spatial pattern of \( e \) is, for example, given in \([44]\). According to a production-dissipation equilibrium of the turbulent kinetic energy \([77]\), the vertical mixing coefficient is defined as

\[
K_V = K_V^0 + \frac{\Gamma_e}{N_b^2} ; \quad N_b^2 = -\frac{g \partial \rho}{\rho_0 \partial z} \quad (2.11)
\]

where \( \Gamma_e = 0.2 \) is the mixing efficiency and \( N_b \) is the buoyancy frequency. A background value \( K_V^0 \) can be attributed to internal wave breaking and it is taken to be constant.

In case of an unstable stratification, \( N_b^2 < 0 \), additional mixing occurs through convective overturning. We can take this mixing into account through an additional mixing coefficient \( K_V^c \gg K_V^0 \) by formulating \( K_V \) as

\[
K_V = K_V^0 + \mathcal{F}(N_b^2)K_V^c + \eta_V \mathcal{F}(-N_b^2) \frac{\Gamma_e}{N_b^2} \quad (2.12)
\]
where $\eta_V$ is another homotopy parameter which can be used to study the situations with only constant mixing coefficient ($\eta_V = 0$) and variable mixing ($\eta_V = 1$). Furthermore, $\mathcal{F}$ is a mixing profile function which we take as

$$\mathcal{F}(x) = \max\{\tanh(-x^3), 0\} \quad (2.13)$$

such that additional convective mixing is generated smoothly as soon as $N^2 < 0$.

### 2.2.3 The discrete equations

The equations are discretized in space using a second order accurate control volume discretization method. The grid is stretched vertically, such that we have more cells close to the surface. The grid cells have indices $i = 1, \ldots, N$, $j = 1, \ldots, M$, $k = 1, \ldots, L$.

In the horizontal direction we use a B-grid and in the vertical direction a C-grid. In atmospheric and oceanographic sciences the vertical layout, where the vertical velocity $w$ is placed at the cell faces and the variables $u, v, p, T, S$ on the plane halfway the cell, is called a Lorenz grid. In previous versions of the code (see [86]) a C-grid was used for both the horizontal and the vertical discretization. The Figures 2.2 and 2.3, show the horizontal C- and B-grid respectively. In [87] we motivated a choice for a B-grid instead of a C-grid via detailed Fourier analysis on a simplified ocean model with Cartesian grid and constant Coriolis force. The main reason is that for small values of the horizontal viscosity $A_H$ the C-grid gives components with high frequencies (wiggles) in the solution of the latitudinal velocity field, which do not appear using a B-grid.

The high frequency components at the C-grid are caused by the fact that the lateral Ekman layer cannot be resolved on a coarse mesh. The Ekman layer is a boundary layer that occurs at the eastern boundary of a basin. The thickness of the layer is typically

$$\delta_E = (A_H/f)^{1/2},$$

where $f = 2\Omega \sin \theta$. Note that both $A_H$ and the Coriolis force are involved.

The Fourier analysis in [87] shows that in order to avoid wiggles in the solution we need a
longitudinal mesh size that satisfies
\[
\Delta \phi_{\text{max}} \ll \frac{2}{\pi^{1/3}} \left( \frac{\delta E}{L} \right)^{2/3}.
\]
Unfortunately in practice $\Delta \phi_{\text{max}}$ often does not satisfy this bound and wiggles will occur in the solution on the C-grid. There are two ways to improve the results: decrease the mesh size which is computationally expensive, or increase $A_H$, which is unattractive from a physical point of view, because we would like to take $A_H$ as small as possible. In case of the B-grid there is no such restriction.

In the equations (2.4a-2.4f) only derivatives of the pressure occur. The immediate consequence is that we can freely add a constant to the pressure field without changing the solution. The pressure is determined up to an additive constant. This holds as well for the discrete pressure field. On the C-grid we have a single degree of freedom, namely the constant vector for the pressure. However on the B-grid we have two degrees of freedom. There is a checker board-splitting of pressure nodes. Both the ”black” and the ”white” nodes form a vector that can be freely added to the solution of the pressure. In the next section we will see that we need the Jacobian of the discretized equations. Because of the degrees of freedom in the equations the Jacobian has two singular vectors in case of the B-grid and only one in case of the C-grid.

The spatially discretized model equations can be written in the form
\[
M \frac{du}{dt} = F(u) = L(u) + N(u, u)
\]
where the vector $u$ contains the unknowns $(u, v, w, p, T, S)$ at each grid point and hence has dimension $d = 6 \times N \times M \times L$. The operators $M$ and $L$ are linear and $N$ represents the nonlinear terms in the equations.

## 2.3 Numerical continuation

Steady state solutions lead to a set of nonlinear algebraic equations of the form
\[
F(u, \mu) = 0.
\]
Here the parameter dependence of the equations is made explicit through the parameter \( \mu \), which can be any of the parameters that we introduced in the previous section. Because \( \mu \) is one dimensional, \( F \) is a nonlinear mapping from \( \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d \).

Given a solution for equations (2.15) we can vary the parameter \( \mu \) and determine a branch of steady solutions. The computation of this branch is a numerical continuation problem. In this section we give a short introduction to the subject. For a more extended introduction we refer to [69, chapter 4] and [1].

By continuation we can find the relation between the parameter and the solution. It gives information about the effect of disturbances in the parameter, the stability of the solutions and whether multiple solutions exist for certain parameter values. On a branch we can encounter bifurcations, that is qualitative changes in the behavior of the solution, such as turning points, pitchfork and Hopf bifurcations.

Moreover continuation is useful as spin-up for complex problems with many problem parameters. If for some set of simple parameter values a solution is known, we can use this solution as a starting point for the continuation process. We follow the paths for all parameters until we reach the desired values.

In case of the ocean circulation problem, continuation of steady states is used for a combination of the reasons mentioned. The problem is huge and contains many problem parameters like earth rotation speed, the atmospheric temperature function, the precipitation, the wind field, the shape of the continents, the shape of the bottom of the ocean and many physical constants.

### 2.3.1 Pseudo-arclength parameterization

Because the equation \( F(\mathbf{u}, \mu) = 0 \) is underdetermined, we need an extra equation if we want to determine a particular solution on the branch. This equation is given by the parameterization of the curve, that defines a relation between \( \mathbf{u}, \mu \) and the parameterization variable \( s \). We use the pseudo-arclength method [47], because it is able to handle turning points.

The pseudo-arclength parameter \( s \) of the curve \((\mathbf{u}(s), \mu(s))\) is defined implicitly by

\[
\zeta \left\| \frac{d\mathbf{u}}{ds} \right\|_2^2 + \left( \frac{d\mu}{ds} \right)^2 = 1,
\]

where the number \( \zeta \) is used to place different emphasis on \( \mathbf{u} \) and \( \mu \). We choose \( \zeta = 1/d \), the inverse of the dimension of the vector \( \mathbf{u} \). Given a solution \((\mathbf{u}(s_0), \mu(s_0)) = (\mathbf{u}_0, \mu_0)\), which can be an analytically known starting solution or a previously computed point on the branch, we can replace the derivatives by a first order approximation. Multiplication with the square of the step length \( \Delta s = (s - s_0) \) gives the desired extra equation

\[
q(\mathbf{u}, \mu, s) = \zeta (\mathbf{u} - \mathbf{u}_0)^T (\mathbf{u} - \mathbf{u}_0) + (\mu - \mu_0)^2 - (s - s_0)^2 = 0.
\]

Fixing \( s \) we want to solve the following equation

\[
\begin{bmatrix}
F(\mathbf{u}, \mu) \\
q(\mathbf{u}, \mu, s)
\end{bmatrix} = 0.
\]

(2.18)
2.3.2 Predictor-Corrector method

Equation (2.18) is solved with a predictor-corrector method. The most commonly used predictor-corrector method is Euler-Newton continuation. The first-order accurate Euler method is used to predict the solution on the branch at $s$ and that prediction is the starting point for a Newton iteration that computes a more accurate solution. The Newton method requires the $(d+1) \times (d+1)$ Jacobian matrix $J_s(u, \mu)$ of (2.18) that is given by

$$J_s(u, \mu) = \begin{bmatrix} \Phi & F_\mu \\ q_u & q_\mu \end{bmatrix},$$  \hspace{1cm} (2.19)

where $\Phi$ is the matrix of derivatives of $F$ to $u$, $F_\mu$ the derivative to the parameter $\mu$, $q_u$ the derivative of $q$ with respect to $u$ and $q_\mu$ the derivative of $q$ with respect to $\mu$. Note that all the subblocks of the Jacobian matrix depend on $u$ and $\mu$.

The Euler predictor

$$(u^{(0)}, \mu^{(0)}) = (u_0, \mu_0) + (\dot{u}_0, \dot{\mu}_0)\Delta s,$$  \hspace{1cm} (2.20)

requires the tangent $(\dot{u}_0, \dot{\mu}_0)$, which is the derivative of $(u, \mu)$ with respect to $s$ at $s = s_0$. We can compute the tangent in the following way. Differentiation of $F(u, \mu) = 0$ with respect to $\mu$ gives

$$\Phi \frac{du}{d\mu} + F_\mu = 0.$$  \hspace{1cm} (2.21)

We can solve $du/d\mu$ from this equation. Furthermore we can substitute

$$\frac{du}{d\mu} = \frac{du}{ds} \frac{ds}{d\mu}$$  \hspace{1cm} (2.22)

in Equation (2.16). After some rearrangement we get

$$\frac{d\mu}{ds} = 1/ \sqrt{1 + \zeta \left\| \frac{du}{d\mu} \right\|^2}.$$  \hspace{1cm} (2.23)

To compute the tangent we solve Equation (2.21), compute $d\mu/ds$ with (2.23) and finally compute $du/ds$ with (2.22).

If we have two previously computed solutions, we can easily apply the secant method, which is favourable because it does not require the solution of a system with $\Phi$. Say we have a point $(u_{-1}, \mu_{-1})$ on the branch before $(u_0, \mu_0)$, then the prediction becomes

$$(u^{(0)}, \mu^{(0)}) = (u_0, \mu_0) + \frac{(u_0, \mu_0) - (u_{-1}, \mu_{-1})}{s_0 - s_{-1}}\Delta s.$$  \hspace{1cm} (2.24)

Both the secant and Euler predictor are first order accurate in $\Delta s$, therefore it does not make a big difference which one we choose. At the starting point of the continuation there is no previously computed point, so there we are forced to use the Euler predictor.
The Newton iteration requires solution of the updates in the linear equation

\[ J_s \left( u^{(k)}, \mu^{(k)} \right) \begin{bmatrix} \Delta u^{(k)} \\ \Delta \mu^{(k)} \end{bmatrix} = \begin{bmatrix} -F \\ -q \end{bmatrix}, \tag{2.25} \]

where the right hand side is a function of \((u^{(k)}, \mu^{(k)})\). The next approximation is

\[ (u^{(k+1)}, \mu^{(k+1)}) = (u^{(k)}, \mu^{(k)}) + (\Delta u^{(k)}, \Delta \mu^{(k)}). \tag{2.26} \]

The iteration stops as soon as \( \| (\Delta u^{(k)}, \Delta \mu^{(k)}) \|_\infty < \epsilon \), usually \( \epsilon = 10^{-3} \).

In each iteration of the Newton process we have to solve Equation (2.25) where \( J_s \) is given by (2.19). If we assume \( \Phi \) is invertible we can multiply the first rows with \( \Phi^{-1} \) which results in the following system

\[ \begin{bmatrix} I & \Phi^{-1} F \mu \\ q_u & q_\mu \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta \mu \end{bmatrix} = \begin{bmatrix} -\Phi^{-1} F \\ -q \end{bmatrix}. \tag{2.27} \]

If we now introduce \( v \) and \( w \), the solutions of the equations

\[ \Phi v = -F, \text{ and } \Phi w = F\mu, \tag{2.28} \]

we can easily compute the updates via

\[ \begin{align*}
\Delta \mu &= -(q + q_u v)/(q_\mu - q_u w), \quad \tag{2.29} \\
\Delta u &= v - \Delta \mu w. \quad \tag{2.30}
\end{align*} \]

Summarizing, each step in the Newton iteration starts with the computation of \( v \) and \( w \) by solving (2.28). Then \( \Delta \mu \) and \( \Delta u \) can be computed with equations (2.29-2.30). In this process the solution of the two systems with \( \Phi \) is by far the most expensive.

Note that the second equation in (2.28) is equal to (2.21), so \( w = -du/d\mu \). In fact we get the Euler predictor for the next step for free. So we can use both the Euler-predictor and the secant predictor.

### 2.3.3 The adaptive Shamanskii corrector

The Newton method requires at each step the solution of system (2.25). To be able to solve the system we first of all have to construct it. We have to build the Jacobian \( J_s(u, \mu) \), or equivalently \( \Phi(u, \mu) \) and the vector \( F_{\mu}(u, \mu) \). Because of the nonlinear terms in the equations and the complex mixing scheme, the computation of \( \Phi \) is quite expensive. Given \( \Phi \) and \( F_{\mu} \), we solve the systems (2.28) via a Krylov subspace iteration that demands the construction of a preconditioner for \( \Phi \). In Section 3.2 of the next chapter we will explain what we mean by Krylov subspace iteration and preconditioning. For now it is enough to know that the construction of a good preconditioner is the most expensive step in the procedure.
Instead of the Newton method we can use an alternative fixed point method. There is a wide range of such methods available, for an extended overview see [11]. We will shortly mention a few fixed point methods that are variants of Newton’s method.

If we simply fix the Jacobian after the first step we get the Newton-chord method. The convergence rate will be linear instead of quadratic. If the first guess is relative close to the solution the convergence is acceptable. In general we will have to take smaller steps in the continuation method. The advantage of the Newton-chord method is that we avoid a lot of computations. The Jacobian $\Phi$, the vector $F_\mu$ and the preconditioner need to be computed only once. Because $F_\mu$ is fixed, we have to solve only the first equation in (2.28) at each step of the iteration.

The Shamanskii method [70] combines the Newton-chord with the Newton method. The Jacobian is fixed for an a priori chosen number of steps. For example each third step will be an expensive Newton step and the two steps in between are much cheaper Newton-chord steps.

The last method can be improved to the adaptive Shamanskii method. Instead of choosing the number of Newton-chord steps on beforehand, one can decide to switch from Newton to Newton-chord based on the convergence of the fixed point iteration. The idea is simple: if the Newton-chord method is converging not fast enough, a Newton step is performed. However we have to determine when it is advantageous to take a single expensive Newton step instead of a number of cheap Newton-chord steps. Therefore we introduce $T_n$, the time needed for a Newton step, and $T_{nc}$, the time needed for an Newton-chord step. Further let $\kappa_n$ be the convergence constant of the Newton method and $\kappa_{nc}$ the convergence constant of the Newton-chord method. Hence if $\varepsilon_k$ is the error of the $k$-th approximation of $(u, \mu)$ (in the Euclidean norm), then with the Newton method the next approximation will have an error close to $\varepsilon_{k+1} = \kappa_n \varepsilon_k^2$ and with the Newton-chord method $\varepsilon_{k+1} = \kappa_{nc} \varepsilon_k$. After $l$ Newton-chord iterations the estimate of the error is $\varepsilon_{k+l} = (\kappa_{nc})^l \varepsilon_k$. From these expressions we can easily compute the number of Newton-chord iterations, that we need, to obtain the same accuracy as obtained by one step of the Newton method or, if $\kappa_n \varepsilon_k^2 < \varepsilon$, the target accuracy $\varepsilon$:

$$l = \log(\max(\kappa_n \varepsilon_k, \varepsilon/\varepsilon_k)) / \log(\kappa_{nc}).$$

The amount of time needed by $l$ inner iterations is $l \cdot T_{nc}$. As long as $l \cdot T_{nc} < T_n$, the time for the Newton-chord steps is less than the time for one Newton step, so it is beneficial to perform the Newton-chord steps. The relation $l \cdot T_{nc} < T_n$ can be rewritten such that we get

$$\kappa_{nc} < \exp\left[ \log\left( \max\left( \kappa_n \varepsilon_k, \frac{\varepsilon}{\varepsilon_k} \right) \right) \cdot \frac{T_{nc}}{T_n} \right].$$

(2.31)

After each inner iteration we can calculate new estimates for $\kappa_{nc}$ and $T_{nc}$ and check if this relation still holds. If it doesn’t we perform a Newton step that gives new estimates for $\kappa_n$ and $T_n$.

The adaptive Shamanskii method can be implemented easily and gives asymptotically the fastest convergence in time possible, therefore it is the corrector method of choice. In [86] we showed that one can save much time using the adaptive Shamanskii method instead of Newton’s method.
2.3.4 Step size control

The last issue we want to address about continuation is the choice of $\Delta s = s - s_0$ in (2.17), i.e. the size of the step along the branch. We can simply fix the step size over a number of steps, but that is not very efficient. In practice we are mainly interested in the point(s) on the curve where the parameter equals some target value $\hat{\mu}$, for example a bifurcation point or the point where the parameter takes its physical values. We would like to step as fast as possible through the branches until we reach the desired value. We don’t want to take unnecessarily small steps, but if we choose the step size too large we risk slow convergence in the correction process or even divergence from the branch. The step size has to be chosen such that on one hand the step size in the predictor is as large as possible, while on the other hand the number of iterations in the corrector is as small as possible. These demands are conflicting, so we have to find the balance between both. In this paragraph we describe a method that searches this balance.

Seydel [69, paragraph 4.6] suggests that there exists something like an optimal number of iterations, call it $N_{opt}$ for the Newton iteration. Let $N$ be the number of iterations in the last Newton process, if $N > N_{opt}$ we could better take a smaller step size and if $N < N_{opt}$ we may be able to take a larger one. This leads to the following definition for $\xi$, the multiplication factor for the step size,

$$
\xi = \begin{cases} 
0.5 & \text{if } \frac{N_{opt}}{N} < 0.5 \\
\frac{N_{opt}}{N} & \text{if } 0.5 \leq \frac{N_{opt}}{N} \leq 2 \\
2 & \text{if } 2 < \frac{N_{opt}}{N},
\end{cases}
$$

where the next step is $\xi \Delta s$. The increase or decrease is limited by a factor 2 in order to prevent too rapid changes in the size of the step. The method is not very sensitive to the precise value of $N_{opt}$, in case of the Newton method 5 seems a good value, in case of the adaptive Shamanskii the value 7 works fine.

The adaptive step size is combined with a trial and error method: if the corrector diverges or does not converge within $2N_{opt}$ iterations, the step is rejected and a new step is taken from the last solution with half the step size. The adaptive step size was successfully used in the computations in [86].

2.4 The Jacobian matrix

The solution of the two equations involving the Jacobian matrix $\Phi$ in (2.28) is the bottle neck in the computations. Hence, if we want to speed up THCM, we have to find a better way to solve these equations. Before we pay attention to the solution of large linear systems, we have to know more about construction and structure of the Jacobian.
2.4 The Jacobian matrix

2.4.1 Construction

In previous formulations of THCM [86], the entries in the Jacobian matrix were computed analytically and directly from the discretized equations. As the mixing formulations (2.7) and (2.12) are quite complicated, it turned out to be advantageous to use an additional numerical evaluation of the Jacobian matrix. In [14], the problem of how to estimate the Jacobian matrix $J$ of a general mapping $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ using the least number of function evaluations is discussed.

The j-th column of $J$ is approximated by

$$J_j = \frac{F(\cdots, x_j + \varepsilon, \cdots) - F(\cdots, x_j, \cdots)}{\varepsilon}.$$  

(2.32)

for small $\varepsilon$. In general, we need to calculate $m$ of such differences to approximate the Jacobian.

For sparse Jacobian matrices, a group of columns can be determined such that no two columns in this group have a nonzero element in the same row position. This is because, for large-scale problems, the elements of $F$ are dependent on a limited number of variables $x_j$. Let $C$ be such a group of columns and let $d \in \mathbb{R}^m$ be a vector with $d_j = \varepsilon$ if $J_j \in C$ and $d_j = 0$ otherwise. Then the difference

$$J_C = \frac{F(x + d) - F(x)}{\varepsilon}.$$  

(2.33)

contains the non-zero elements of the columns belonging to $C$. By partitioning the columns of the Jacobian matrix in this way, we can more efficiently evaluate the Jacobian since the number of differences we need to calculate are strongly reduced. For the implementation of this approach, we used subroutines provided by [14]. These subroutines consists of (i) routines that determine the groups $C$, such that the number of groups is minimal; and (ii) routines that determine for a given group $C$ the vector $d$ and extract the columns $J_j$ from $J_C$. These routines require the sparsity pattern of the Jacobian matrix, i.e., the variables on which $F$ depends, and a routine that evaluates $F(x)$.

This method is applied only to the mixing formulations in (2.7) and (2.12). The rest of the entries in the Jacobian is computed analytically from the discrete equations.

The continuous formulation of convective adjustment in (2.12) combined with the numerical evaluation of the Jacobian, can cause large off-diagonal couplings between $T$ and $S$ in the Jacobian, which makes the solution of the equations more difficult.

2.4.2 Structure

In each step of the Newton process we have to solve one or two linear systems of equations with the matrix $\Phi$ in (2.19), which is by far the most expensive part of the computation.

The equation $\Phi \mathbf{u} = \mathbf{b}$ has the following structure

$$
\begin{bmatrix}
A_{uv} & E_{uv} & G_{uv} & 0 \\
0 & 0 & G_w & B_{TS} \\
D_{uv} & D_w & 0 & 0 \\
B_{uv} & B_w & 0 & A_{TS}
\end{bmatrix}
\begin{bmatrix}
\mathbf{u}_{uv} \\
\mathbf{u}_w \\
\mathbf{u}_p \\
\mathbf{u}_{TS}
\end{bmatrix}
= 
\begin{bmatrix}
\mathbf{b}_{uv} \\
\mathbf{b}_w \\
\mathbf{b}_p \\
\mathbf{b}_{TS}
\end{bmatrix},
$$

(2.34)
Chapter 2. Building blocks of the thermohaline ocean circulation model

The four rows in (2.34) represent the discrete version of the equations for conservation of momentum in longitudinal and latitudinal direction (equations (2.4a,2.4b)), hydrostatic equilibrium (equation (2.4c)), conservation of mass (equation (2.4d)) and conservation of heat and salt (equations (2.4e,2.4f)), respectively.

Each $G^*$ represents the discrete gradient operator and each $D^*$ the discrete divergence operator. Due to the spherical coordinate system these discrete operators are not necessarily each others transpose. However there exist diagonal matrices $\Lambda_1$ and $\Lambda_2$ such that $G^T = \Lambda_1 D^* \Lambda_2$, which is important for numerical stability. The matrices $A_{TS}$ and $A_{uv}$ are of convection-diffusion type, and $A_{uv}$ includes the Coriolis terms. The matrices $B^*$ represent the couplings between the dynamic and thermodynamic variables. The matrix $E_{uv}$ represents the coupling to $w$ in the material derivative in equations (2.4a,2.4b). This coupling is very weak hence we will ignore the block in this section. It is treated as another zero block which simplifies the calculations. As one can see, next to $E_{uv}$ there are six zero blocks, which means that severable variables are not coupled. All the submatrices are structured and sparse and there are only a few nonzeros per row in each matrix.

We use clustering for $u, v$, because the equations (2.4a) and (2.4b) have the same structure. As the same holds for (2.4e) and (2.4f), also the variables $T, S$ are clustered.

Because of the B-grid (see Figure 2.3), in principle there are not equally many unknowns of each kind. We added dummy variables to make sure there are precisely $d = 6 \times N \times M \times L$ unknowns. These dummy variables are located outside the domain and are described by trivial equations. In general, the dummy equations for $w$ and $p$ will generate nonzero diagonal entries in the second and third diagonal block of (2.34). To obtain the given structure, we have to remove these equations in advance. Immediate consequence is that the dimensions of the vectors $u^*$ in (2.34) are no longer the same. Let $d_u$ be the dimension of each $u^*$, then it holds that $d_{uv} = d_{TS} = 2 \times N \times M \times L$ and $d_w < d_p \leq N \times M \times L$. The inequality $d_w < d_p$ holds because we have a Lorenz grid in the vertical direction (see Figure 2.3). Per water column, there is one internal pressure node more than there are vertical velocity nodes.

The matrix in (2.34) is huge, far from diagonal dominant and indefinite, i.e. it has eigenvalues with positive as well as negative real part. This makes the system hard to solve for both the direct and iterative methods, that we will describe in Chapter 3.

2.5 The saddle point problem

In the previous chapter we already mentioned the bottle neck in the computation of ocean flows: the solution of systems with the Jacobian matrix. This is by far the most expensive part. A step to models with a higher resolution is only possible if we improve the way the systems are solved. The problems in solving the Jacobian matrix of the ocean equations are largely caused by the fact that it stems from a set of coupled partial differential equations, which gives a complex matrix structure as in Equation (2.34). Only in the next chapter we will introduce
2.5 The saddle point problem
the reader to the solution of large systems of equations. First we want to describe a somewhat easier problem that partially resembles the structure of ocean matrix: the saddle point problem.

A saddle point problem occurs in several flow problems and is, alike the ocean system, hard to solve. In our quest for better solvers for the Jacobian in ocean flows, we studied the saddle point problem, because better insight in the solvers for the saddle point problems could help us in the design of a new solver for the ocean systems.

A saddle point problem is given by an equation

\[ Kx = b, \]

where \( K \in \mathbb{R}^{(n+m) \times (n+m)} \) \((n \geq m)\) is a matrix of the special form

\[ K = \begin{pmatrix} A & B^T \\ B & C \end{pmatrix}, \]

where \( C \in \mathbb{R}^{m \times m} \) is symmetric and negative semidefinite, \( A \in \mathbb{R}^{n \times n} \), and \( B \in \mathbb{R}^{n \times m} \). In this thesis we only consider \( C = 0 \). Often the matrix \( A \) will be symmetric positive definite. Then the matrix \( K \) itself is symmetric indefinite. Like \( \Phi \) in (2.34), \( K \) is indefinite, structured and sparse. A saddle point problem occurs in the numerical solution of partial differential equations with constraints, like for example the (Navier)-Stokes or Oseen equations.

2.5.1 The Navier-Stokes equations

The incompressible Navier-Stokes equations on an open bounded domain are

\[ -\nu \Delta u + (u \cdot \nabla)u + \nabla p = 0, \]
\[ \nabla \cdot u = 0. \]

(2.37)

The linearized version of this equation is

\[ -\nu \Delta u + (\bar{u} \cdot \nabla)u + \nabla p = 0 \]
\[ \nabla \cdot u = 0, \]

(2.38)

where the "wind" \( \bar{u} \) is such that \( \nabla \cdot \bar{u} = 0 \). These Oseen equations occur if we solve the Navier-Stokes equations via a Picard iteration, where we take \( u = u^{(m)} \) and \( \bar{u} = u^{(m-1)} \), the previous guess for the solution. Discretization of the Oseen equations gives a saddle point problem with non-symmetric positive definite \( A \).

If we further simplify the equations and drop the convective term \( (\bar{w} \cdot \nabla)u \) we get the Stokes equation

\[ -\nu \Delta u + \nabla p = 0, \]
\[ \nabla \cdot u = 0. \]

(2.39)

After discretization we get a saddle point problem with symmetric and positive definite \( A \).
2.5.2 Saddle point problems and ocean flows

There is an important connection between saddle point problems and the ocean problem. The Jacobian matrix $\Phi$ from (2.34) can be written as two coupled saddle point problems. This is easily shown if we apply the permutation matrix $Q$, which corresponds to the block-permutation $(1,3,4,2)$, to $\Phi$:

$$Q\Phi Q^T = \begin{bmatrix}
A_{uv} & G_{uv} & 0 & E_{uv} \\
D_{uv} & 0 & 0 & D_w \\
B_{uv} & 0 & A_{TS} & B_w \\
0 & G_w & B_{TS} & 0 
\end{bmatrix}.$$  \hspace{1cm} (2.40)

One immediately recognizes the two saddle point problems: the first is a Navier-Stokes system including Coriolis for $u, v$ and $p$ (the dynamical part) and the second is a saddle point problem for $T, S$ and $w$ (the thermodynamical part). The last system is certainly not symmetric, because $B_{TS}$ is the discretization of the term $\rho_0 g (\alpha_T T - \alpha_S S)$ in (2.4c), so it is a constant matrix, whereas $B_w$ is the discretization of $w \frac{\partial T}{\partial z}$ and $w \frac{\partial S}{\partial z}$ in (2.4e) and (2.4f) respectively, so it depends on $T$ and $S$, and consequently we have $B_{TS} \neq B_w$.

Because $\Phi$ and $K$ have similar properties we study the numerical solution of system (2.35) hoping to be able to apply similar techniques to $\Phi$. Therefore in Chapters 4 and 6 the focus is on the saddle point problem and in Chapter 5 it plays an important role as well. The knowledge on the saddle point problem is relevant for the ocean system, as we will see in Chapter 7, where saddle point problems occur in the solution of equations with $\Phi$.

2.6 Time stepping, data assimilation and stability analysis

So far we assumed that we solve steady states of the equations (2.4a-f), hence all the time derivatives vanished. However THCM can do time stepping as well. In fact, the algorithms for continuation of steady states and time integration have a lot in common. In this section we shortly pay attention to the solution of time dependent ocean problems, because the data assimilation algorithm uses time integration.

To solve the time dependent equation (2.14) we apply the theta-method

$$-M \frac{U^{(n+1)} - U^{(n)}}{\Delta t} = \Theta F \left(U^{(n+1)}\right) + (1 - \Theta) F \left(U^{(n)}\right) \hspace{1cm} (2.41)$$

with initial condition $U^{(0)}$. The matrix $M$ is given by

$$M = \begin{bmatrix}
I_{uv} & 0 & 0 & 0 \\
0 & I_{uv} & 0 & 0 \\
0 & 0 & I_{TS} & 0 \\
0 & 0 & 0 & I_{TS} 
\end{bmatrix}. \hspace{1cm} (2.42)$$
which is a diagonal matrix with ones on the diagonal for the discrete momentum equations (2.4a-b) and the discrete tracer equations (2.4e-f) that contain a time derivative. Note that the hydrostatic pressure equation (2.4c) and the equation of conservation of mass (2.4d) do not have a time derivative, so the corresponding diagonal entries in $M$ are zero.

The choice of $\Theta \in [0, 1]$ defines the properties of the scheme (2.41).

If $\Theta = 0$ we obtain the explicit Forward Euler (FE) method. With FE, it is easy to compute the next states, because there are no system solves involved. However FE requires a very small time step to guarantee stability and convergence. In a typical ocean problem the maximum time step is in the order of an hour, while the timescales in the ocean can be in the order of a thousand years. Hence a model run would require an enormous amount of time steps.

If $\Theta \neq 0$ we have an implicit method. If $\Theta = 1$, we get the Backward Euler method, which is unconditionally stable, so the time step size $\Delta t$ can be arbitrary large. For the effect of other choices for $\Theta$ (like $\Theta = 1/2$) see for example [78].

The solution $u^{(n+1)}$ of the implicit equation is obtained by a predictor-corrector method, similar to the one we described in Section 2.3.2 for the case of numerical continuation. The Jacobian matrix involved in the corrector iterations has the form

$$\Phi_t = \frac{1}{\Delta t} M + \Phi.$$  \hfill (2.43)

The Jacobian has the same structure as $\Phi$ in Equation (2.34). The difference between the two is that the matrices $A_{uv}$ and $A_{TS}$ in $\Phi_t$ are more diagonally dominant as $1/\Delta t$ is added to the diagonals. In general diagonal dominance is a pleasant property for the solvers.

For more background information on time integration see [39].

Next to straightforward implicit time stepping the data assimilation algorithm uses backward integration, which we will shortly clarify. The aim of data assimilation is to combine observations (in our case satellite data of SSH and geoid) with the numerical model (here THCM). We want to optimize the initial value for time integration such that measurements are approximated over a specified time interval as good as possible in least squares sense, leading to an error functional. For the minimization of this error functional one uses the steepest descent. Hence the gradient of the error functional has to be computed. This gradient is the integral of initial values that should be specified to an adjoint linearized model, such that at time $t$, somewhere in the specified interval, precisely the difference between measurement and the solution of our flow model based on the current initial is met. This means that first we have to solve our model on the whole interval and then from every $t$ in the specified interval we have to integrate backwards with the adjoint linearized model to find the according initial value of that adjoint model. For details we refer to [76]. In practice it means that we have to solve systems with the transpose of the matrix $\Phi_t$. Hence the relevance for this thesis is, that we have to take in account that the solver that we design for a Jacobian with structure (2.34) should be able to
The question whether a steady solution $u$ of Equations (2.4) is stable, is related to time integration. A stable solution is characterized by the fact that small distortions do not grow, but damp out as time progresses. If these distortions grow the initial solution is unstable. The growth of distortions is related to the eigenvalues and eigenvectors of the generalized eigenvalue problem

$$\Phi(u)v_i = \lambda_i M v_i,$$

were the diagonal matrix $M$ is given by (2.42). If all eigenvalues lie in the left half plain, i.e. $\Re(\lambda_i) < 0$, the solution is stable for all initial solutions close enough to $u$. If at least one eigenvalue, say $\lambda_1$, has a positive real part the solution is unstable, i.e. starting with $u + \alpha v_1$, where $\alpha > 0$ may be arbitrary small, will always diverge from $u$.

Qualitative changes in the eigenvalues of $\Phi$ along the branch are related to qualitative changes of the solution: bifurcation points. For example a Hopf bifurcation is equivalent to a complex eigenpair with a real part that changes of sign. At a Hopf bifurcation the solution changes from stationary to periodic. See [69, Ch.2] and [17, Ch.3] for more details.

We compute the eigenvalues and eigenvectors with the Jacobi-Davidson iteration method, presented in [72].