university of groningen / faculty of science and engineering / mathematics and app mathematics

Stommel's Box Models
A Numerical Analysis

Bachelor's Project Mathematics

July 2024

Student: Rick Ploeg (S3207269)

First supervisor: Dr. A.E. Sterk

Second assessor: Dr. H. Jardon Kojakhmetov

Abstract

This thesis is dedicated to analyze, implement and explain different methods of finding equilibria in dynamical systems. This is done through the lens of Stommel's Box Models of thermohaline circulation. These Stommel's Box Models are introduced, their relevancy is explained and the model is explicitly defined. Then, two different methods of equilibria computation are introduced: pseudo-arclength continuation and polynomial reduction after which a guidance is provided regarding the process of implementing these methods. The methods are implemented in a MATLAB framework and the developed code is provided along a recommendation of use cases, possible further development and further research.

Contents

1 Introduction

The currents of our oceans are one of the greatest forces of nature on our earth. They heavily impact many aspects of the planet's climate; from temperature and humidity to precipitation on all corners of all the continents. It is thus hard to imagine an ecosystem on our earth that is not acted on by the characteristics of the ocean currents. Indeed, many studies are dedicated to understanding causes, e ects and mechanisms of changes in our ocean's characteristics [vA07].

This interest has given rise to a variety of di erent models in an attempt to better understand and predict the behaviour of the ocean and it's in uences. These models vary greatly in purpose, complexity and resolution. High-resolution state of the art models are computationally expensive and typically reserved for large institutions. In contrast, this thesis is dedicated to a particular lightweight, low-resolution type of model: Stommel's box models. These models are implemented and its abilities and constraints are analysed in this thesis. Attention will be given to the challenge of numerically obtaining equilibria of the models for which multiple approaches will be implemented and compared.

2 Stommel's box models

2.1 The Two-Box Model

2.1.1 Model construction

Stommel introduced their model in 1961 in which they approached the modeling of ocean currents by imagining two connected volumes of water with different temperature and salinity [Sto61]. What later would be called Stommel's two-box model is a model that describes the flow of water between two volumes of water due to thermohaline forcing: A flow of water induced by a temperature and salinity gradient. Specially, it models changes over time of thermohaline properties between two volumes of water, each with their own forcing conditions.

Stommel's two-box model can be assembled by first introducing only a single box B_1 which is to be understood as a reservoir of water with a temperature T_1 and salinity S_1. In physical terms, these quantities of course have specific dimensions and values. However for the purposes of analysing the behaviour of the model they can be made to be dimensionless and for now have initial value 0.

Then, the box is expanded to include an ambient source of heat first set to an arbitrary value $aT_1 := 1$. Similarly, an ambient source of salination $aS_1 := 1$ is included. These represent the average influx over time of the sun's energy and the accumulation of salt into the water (see figure 1). To incorporate these into the model, the first differential equations are introduced:

$$\frac{dT_1}{dt} = {}_T(aT_1 \quad T_1); \tag{1a}$$

$$\frac{dS_1}{dt} = {}_S(aS_1 \quad S_1): \tag{1b}$$

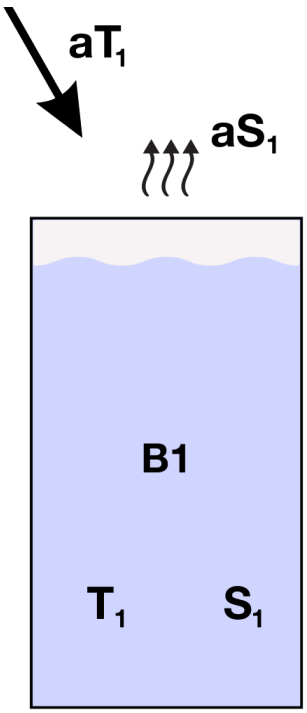

Figure 1: A Single Box B_1 with a temperature T_1, a salinity S_1 an ambient temperature aT_1, and ambient salinity aS_1.

Here ${}_T$ and ${}_S$ are constants characterizing the speed of heat and salinity conducting into the water. At this point, the differential equations already form a functioning model. The current model simulates a reservoir of water being subjected to an ambient source of heat and salination. Perhaps

unsurprisingly, the temperature $T1$ and salinity $S1$ both converge to their respective ambient values (gure 2).

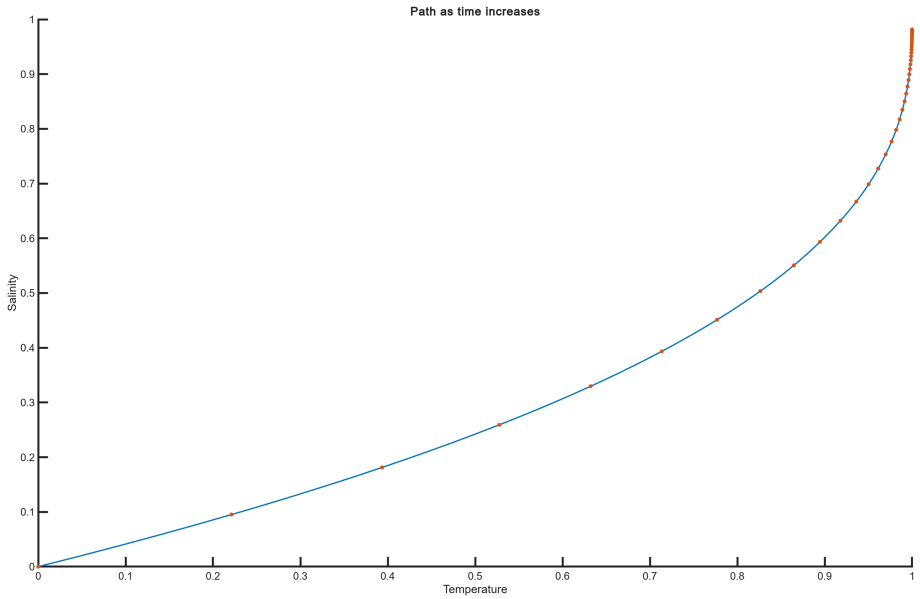

Figure 2: Values of T_1 and S_1 converging to aT_1 and aS_1 both set equal to 1 with equidistant points in time highlighted.

To continue the construction of the Stommel's models, a second box B_2 is de ned symmetrically to B_1 just di ering in the value of their ambient temperature and salinity (3). The set of di erential equations is thus expanded with:

$$\frac{dT_2}{dt} = {}_T(aT_2 \quad T_2); \tag{2a}$$

$$\frac{dS_2}{dt} = {}_S(aS_2 \quad S_2): \tag{2b}$$

At this point, the properties of each box both converge independently to their respective ambient values; Their values are not yet 'linked'. To nish the classic two-box model, the boxes are connected by a set of two tubes which allow water to ow in and out of the reservoirs to the other. One tube is classically imagined to allow a ow of deep water the second tube allows water to ow back at the surface. After all, the surface level of equatorial and polar waters is preserved when considering underwater ow patterns. Stommel's two-box model considers thermohaline induced ows implying that the ow is assumed to be solely dependant on the di erence of the density due to temperature and salinity.

Indeed, ow will be encapsuled into the model after rst de ning a density:

$$_x = {}_0 + {}_0 {}_S(S_x \quad S_0) \quad {}_0 {}_T(T_x \quad T_0);$$

where ${}_0$, T_0, S_0 are reference values for a density, temperature and salinity respectively. And c_T and c_S are waters' contraction coe cients for temperature and salinity.
Now it is possible to write:

$$= \quad \frac{1 \quad 2}{0};$$

where is a constant describing the viscosity of water.

Note how can take on both positive and negative values. A positive value for indicates a higher density in B_1 and would hence be interpreted as an underwater ow from B_1 to B_2. That being said, the model values are actually invariant to the sign of . This is due the over ow-tube ensuring any underwater ow is countered equally at the surface. With the ow having been properly de ned, it

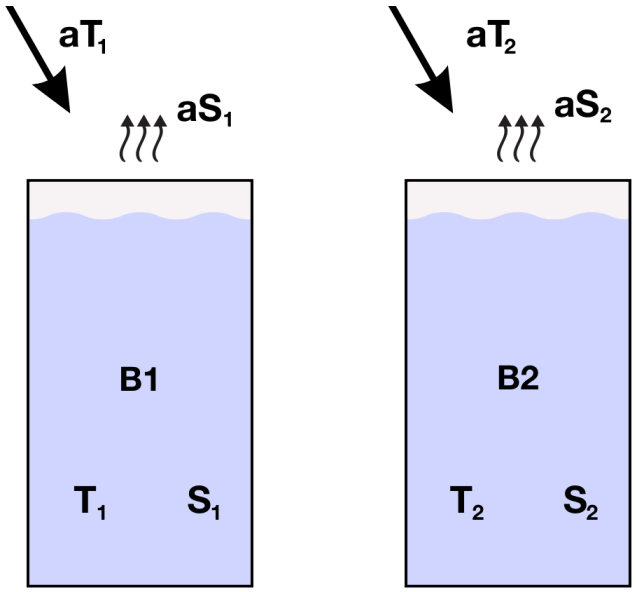

Figure 3: A Second Box B_2 with temperatures T_1 and T_2, salinities S_1 and S_2 ambient temperatures aT_1 and aT_2, and ambient salinities aS_1 and aS_2.

is now possible to construct the following:

$$\frac{dT_1}{dt} = {}_T(aT_1 \quad T_1) + (T_2 \quad T_1) \quad j \quad j; \tag{3a}$$

$$\frac{dT_2}{dt} = {}_T(aT_2 \quad T_2) + (T_1 \quad T_2) \quad j \quad j; \tag{3b}$$

$$\frac{dS_1}{dt} = {}_S(aS_1 \quad S_1) + (S_2 \quad S_1) \quad j \quad j; \tag{3c}$$

$$\frac{dS_2}{dt} = {}_S(aS_2 \quad S_2) + (S_1 \quad S_2) \quad j \quad j: \tag{3d}$$

The model is then typically condensed by de ning $T := T_1 \quad T_2$ and $S := S_1 \quad S_2$. This action results in a system with fewer degrees of freedom at the expense of losing information on the absolute values of the temperature and salinities. This does not in uence any relative behaviour of the system nor does it impact e.g. the amount of equilibria. It does deserve to be mentioned, however, that it is possible to construct conditions in which e.g. T_1 and T_2 might not converge or converge much slower than $T_1 \quad T_2$ does. These were speci cally prevalent when implementing periodic (seasonal) perturbations in the system's parameters.

Finally, the mentioned condensation results in the following:

$$\frac{dT}{dt} = \frac{dT_1}{dt} \quad \frac{dT_2}{dt} = {}_T(aT_1 \quad aT_2 \quad T) \quad 2T \quad j \quad j; \tag{4a}$$

$$\frac{dS}{dt} = \frac{dS_1}{dt} \quad \frac{dS_2}{dt} = {}_S(aS_1 \quad aS_2 \quad S) \quad 2S \quad j \quad j: \tag{4b}$$

While this model is fully functioning, the model does contain a handful parameters which e ectively only scale the model and not in uence inherent behavior of the system. The dimension of temperature

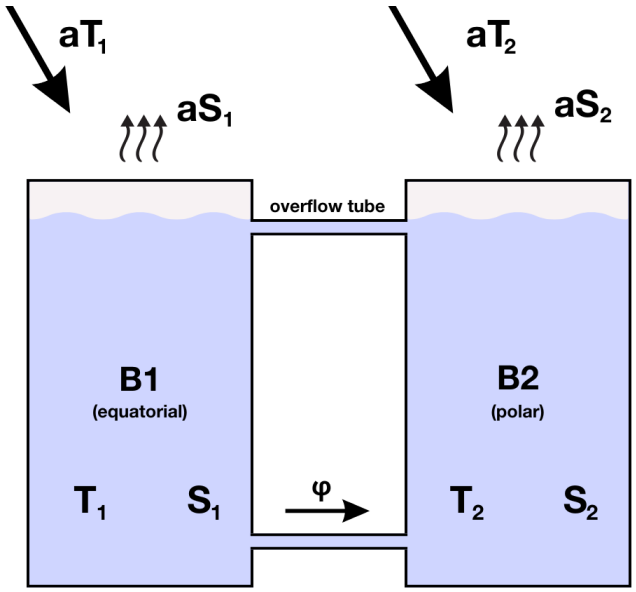

Figure 4: An illustration of Stommel's two-box model.

can e.g. be rescaled with a factor $\frac{1}{T}$. Similarly, S, t, and are rescaled to obtain the simpli ed:

$$\frac{dT}{dt} = p_1 \quad T(1 + jT \quad Sj); \tag{5a}$$

$$\frac{dS}{dt} = p_2 \quad S(p_3 + jT \quad Sj); \tag{5b}$$

where has been reduced to $= T \quad S$ and p_1, p_2 and p_3 are the remaining parameters in the system. Here p_1 and p_2 contain information on thermal and saline forcing while p_3 is a factor correcting for the originally di erent impacts of temperature and salinity on density.

2.2 The Three-Box Model

Stommel's two-box model models the interaction between two volumes of water: an equatorial and a polar volume. A natural expansion to the model would then be to include a third volume making it possible to separately model the north and south polar waters. This expansion enables a variety of intuitive properties of the real world to be re ected in the model: The expanded model allows the north and south waters to adhere to their own ambient temperature- and salinity. And indeed, the ow from equatorial to northern waters need not be symmetric to the ow from southern to equatorial waters.

2.2.1 Model construction

A couple of clari cations to construct the di erential equations of the three-box model need to be established. In the equations of the two-box model there exists an implicit assumption that the volume of water in each box is equal, or more precisely, the model values of both boxes are in uenced equally under equal ow-conditions. The impact of an incoming ow can, however, certainly be scaled to one's wishes. Indeed, the ratio of volumes of water in the boxes can to be chosen, analogous to the choice where on the globe to draw the imaginary lines that de ne the boundaries between the volumes of water. In the two-box model, this was implicitly chosen to be 1 : 1. However here, a ratio

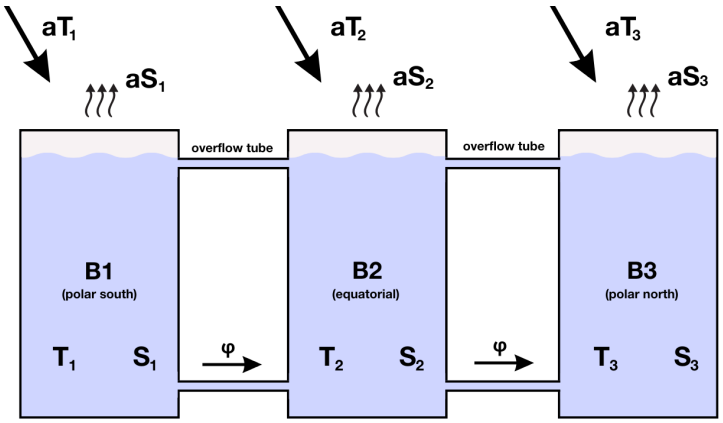

Figure 5: Illustration of Stommel's three-box model.

of 1 : 2 : 1 is chosen to re ect the natural idea that the sum of volumes of the polar regions is equal to the equatorial region. This choice is largely arbitrary as any positive ratios are equal up to scaling but the speci c choice is worth mentioning for clarity on the construction of the model's equations.

Indeed, the model now consists of three connected boxes: B_1, B_2 and B_3 (5 which, similarly to the two-box model each consist of a volume of water with temperatures and salinities. Once again, it is chosen to exclusively work with di erences of these values with boxes:

$$\begin{aligned} T_s &:= T_1 \quad T_2; \\ T_n &:= T_2 \quad T_3; \\ S_s &:= S_1 \quad S_2; \\ S_n &:= S_2 \quad S_3; \end{aligned}$$

where the subscripts s and n refer to north and south respectively.

The equations of motion are then, analogously to the two-box model, derived and simpli ed resulting in the following:

$$\frac{dT_s}{dt} = p_{1;s} \quad T_s(1 + \frac{3}{4}jT_s \quad S_sj) \quad T_n(\frac{1}{4}jT_n \quad S_nj); \tag{7a}$$

$$\frac{dT_n}{dt} = p_{1;n} \quad T_s(1 + \frac{3}{4}jT_n \quad S_nj) \quad T_s(\frac{1}{4}jT_s \quad S_sj); \tag{7b}$$

$$\frac{dS_s}{dt} = p_{2;s} \quad S_s(p_{3;s} + \frac{3}{4}jT_s \quad S_sj) \quad S_n(\frac{1}{4}jT_n \quad S_nj); \tag{7c}$$

$$\frac{dS_n}{dt} = p_{2;n} \quad T_s(p_{3;n} + \frac{3}{4}jT_s \quad S_sj) \quad S_s(\frac{1}{4}jT_s \quad S_sj); \tag{7d}$$

where $p_{1;s}$, $p_{2;s}$ and $p_{3;s}$ are typically approximately equal to $p_{1;n}$, $p_{2;n}$ and $p_{3;n}$ respectively. A true equality would re ect a symmetry in ambient temperature and salinities in northern and southern waters. The symmetric case will later be used as a starting point to analyse the asymmetric case.

2.3 Equilibria

2.3.1 Introduction and problem statement

When analyzing a dynamical system, one classic topic of interest is regarding the existence of equilibria. That is, in a system $\frac{df}{dt} = f(x)$ it is of interest to nd the zeros of f; the points that satisfy $f(x) = 0$. These are the points at which the system experiences no further change. For example, in the case of

Stommel's two-box model the following equations would be derived from 5.

$$\frac{dT}{dt} = p_1 \quad T(1 + jT \quad Sj) = 0; \tag{8a}$$

$$\frac{dS}{dt} = p_2 \quad S(p_3 + jT \quad Sj) = 0: \tag{8b}$$

Finding the equilibria of the two-box model would equate to solving these equations for T and S. Occasionally, for some systems, this can be done analytically by hand or via symbolic programming. In this thesis however, the focus lies on numerical approaches in the spirit of nding more generalizable insights.

The equations as given could be fed into established numerical solvers like Matlab's root- nder. This will typically at least partially function as the algorithm often returns at least one root. It is however not guaranteed and typically unlikely that all roots are found. The solver rst tries (and in this case, fails) to solve the equations symbolically after which it then tries nd solutions based on di erent numerical root- nding methods. These methods su er from possibly failing to nd all roots due to e.g. a lack of a method to determine the amount of expected roots. It is key to realise that these issues are intimately connected to the existence of the $jT \quad Sj$ terms and the critical points where $T \quad S$ would otherwise changes sign. These absolute values introduce areas in the state-space where the model equations are not di erentiable. A property that prevents typical solving methods of converging e ciently to a solution.

Dijkstra addressed this problem in [Dij05] by introducing an approximation to the absolute-value function that is di erentiable and hence improves root- nding behaviour around points in the state-space where $T \quad S$. This approach is implemented and compared to a di erent approach. For this second approach, it is key to realise that the equations in Stommel's box models are 'quite close' to being a multivariate polynomial. Speci cally, it is possible to split the system in cases where $T > S$ and cases where $T < S$. The system then reduces to sub-systems each de ned by a relatively simple polynomial function. These polynomials can be solved with more e cient tools with the crucial insight that it is known from polynomials how many solutions one expects. The constructed subsystems then return their solutions which can then get checked for their validity.

These approaches are implemented, discussed further and compared in section 3.

2.3.2 Modes of Equilibria

Both the two- and three-box model contain a variety of equilibria. One natural way to classify the equilibria in the models is to recognize the di erent modes of ocean circulation they represent. This can be done for both the two- and three-box model of which here the three-box model is presented as this re ects the earth's oceans more accurately. When working with the three-box model, four modes of ocean ow can be recognized: The ows between the boxes can each be either positive or negative resulting in the total of four combinations of possible modes (6). For practical purposes, these di erent modes are of impressive importance; an overturning of ocean currents would have catastrophic consequences on ecosystems on the planet. For example, the up-welling of deep waters provide surface waters with nutrients with a regularity that many species rely on.

3 Finding Equilibria

Stommel's two-, and three-box models contain equations that are not typically well-behaved: The term $jT \quad S\,j$ is prevalent in the model and is not di erentiable in either T or S . Many numerical solving methods do not exhibit convergent behaviour when supplied with such terms. Hence, it is of great bene t to address this before attempting to solve the relevant equations. The challenge to transform these ill-behaved terms was addressed in two ways during this project.

The rst approach discussed involved reducing the model to several well-behaved sub-models of which the results could then be combined and re-interpreted to obtain the desired solutions.

The second approach involved addressing the absolute value-signs by implementing a di erentiable approximation of $jT \quad S\,j$. There exist many a potential approximation that could be used which was experimented with. In the results of this project it can be assumed that it was opted to use $jxj \quad \sqrt{x^2 + \ ^2} \quad$ for some small (7). This is a relatively simple approximation which comes with

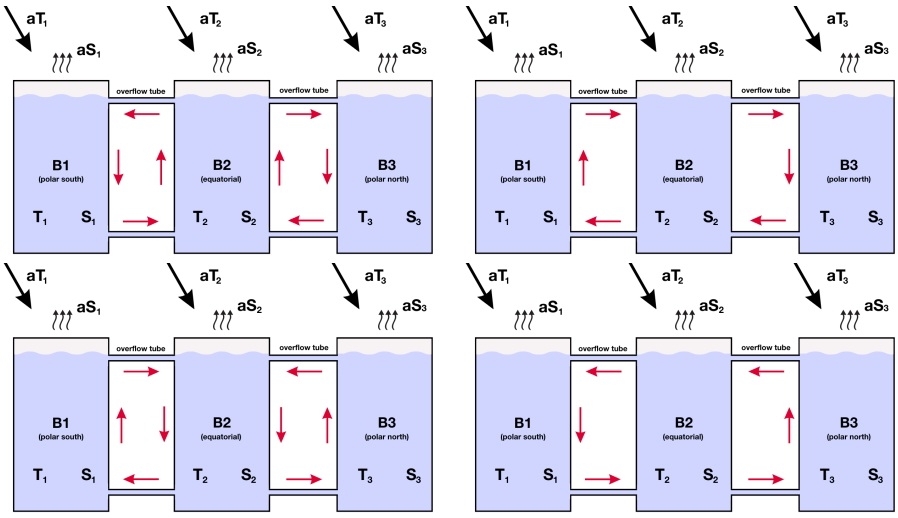

Figure 6: The four main modes of equilibria in Stommel's three-box model.

the advantage that it allows for a rather approachable method of controlling the 'sharpness' of its values near $x = 0$ by simply adjusting .

Both approaches are of use for both Stommel's two-, and three-box models. It was chosen to analyze both methods primarily through the lens of the three-box model as it is the more complex model of the two. All used methods are of course also applicable to the two-box model.

3.1 Polynomial Reduction

3.1.1 Introduction

The method of polynomial reduction rests on the fact that, if one were to ignore the absolute-value signs, all the equations in Stommel's box models are multivariate polynomials. This sparks an interest as polynomial functions are rather well documented and highly optimised solving methods are readily available. And indeed, it can therefore be bene cial to reduce the model to one that constitutes of only polynomials and using the obtained results to reconstruct solutions to the original equations.

3.1.2 Implementation

The implementation was started by dropping the absolute value-signs in the model in exchange for an explicit case distinction. See that in general:

$$jxj = \begin{pmatrix} x; & \text{for } x > 0 \\ x; & \text{for } x \quad 0 \end{pmatrix}$$

This explicit case distinction can then be extended to the equations of the three-box model leading to the following from e.g. $\frac{dT_S}{dt}$ in 7:

$$\frac{dT_S}{dt} = \begin{cases} p_{1;S} & T_S(1 + \frac{3}{4}(T_S \quad S_S)) \quad T_n(\frac{1}{4}(T_n \quad S_n)); & \text{for } T_S > S_S \text{ and } T_n > S_n \\ p_{1;S} & T_S(1 \quad \frac{3}{4}(T_S \quad S_S)) \quad T_n(\frac{1}{4}(T_n \quad S_n)); & \text{for } T_S \quad S_S \text{ and } T_n > S_n \\ p_{1;S} & T_S(1 + \frac{3}{4}(T_S \quad S_S)) + T_n(\frac{1}{4}(T_n \quad S_n)); & \text{for } T_S > S_S \text{ and } T_n \quad S_n \\ p_{1;S} & T_S(1 \quad \frac{3}{4}(T_S \quad S_S)) + T_n(\frac{1}{4}(T_n \quad S_n)); & \text{for } T_S \quad S_S \text{ and } T_n \quad S_n \end{cases} \tag{9}$$

Notice that each individual case in 9 is indeed just a multivariate polynomial for which highly optimized solving methods exist.

De ne

$$\frac{dT_S}{dt}_{++} := p_{1;S} \quad T_S(1 + \frac{3}{4}(T_S \quad S_S)) \quad T_n(\frac{1}{4}(T_n \quad S_n));$$

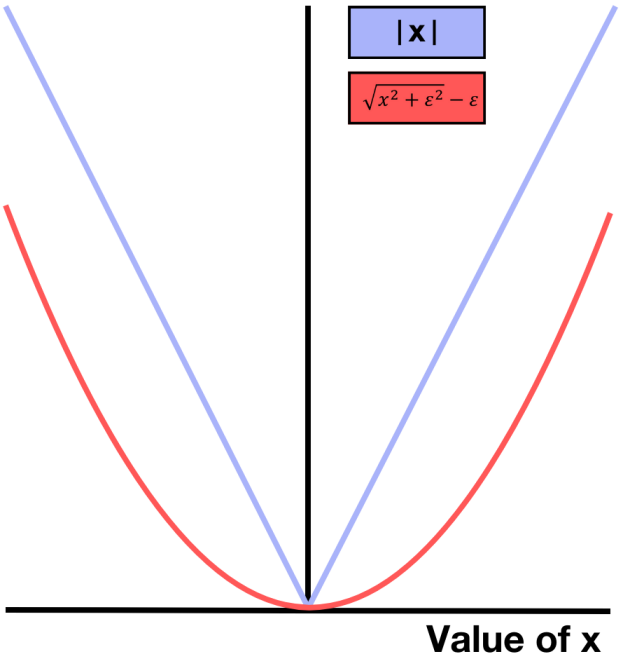

Figure 7: Figure illustrating the approximation of $|x|$ with $\sqrt{x^2 + \ ^2}$.

where the subscript ++ denotes the sign of T_s S_s and T_n S_n i.e. in this case both $T_s > S_s$ and $T_n > S_n$. This is then done with each of equation of 7 allowing for the construction of four new systems of equations: one for each combination of signs i.e. ++, + , + and . The equations relating to ++ are written out explicitly:

$$\frac{dT_s}{dt}_{++} = p_{1;s} \quad T_s(1 + \frac{3}{4}(T_s \quad S_s)) \quad T_n(\frac{1}{4}(T_n \quad S_n)); \tag{10a}$$

$$\frac{dT_n}{dt}_{++} = p_{1;n} \quad T_s(1 + \frac{3}{4}(T_n \quad S_n)) \quad T_s(\frac{1}{4}(T_s \quad S_s)); \tag{10b}$$

$$\frac{dS_s}{dt}_{++} = p_{2;s} \quad S_s(p_{3;s} + \frac{3}{4}(T_s \quad S_s)) \quad S_n(\frac{1}{4}(T_n \quad S_n)); \tag{10c}$$

$$\frac{dS_n}{dt}_{++} = p_{2;n} \quad S_s(p_{3;n} + \frac{3}{4}(T_n \quad S_n)) \quad S_s(\frac{1}{4}(T_s \quad S_s)): \tag{10d}$$

These form a system of second-order polynomials of which e cient solving methods are readily available. These solutions are then veri ed if they are also solution to the original problem. In this case, this can be done rather easily by simply checking validity of the conditions in 9. In a more general case, even with implicit conditions, it is typically computationally more e cient to check the validity of a given solution than it is to generate it.

This procedure of generating solutions is then repeated for the other cases resulting in a combined set of solutions which are the desired equilibria.

3.1.3 Properties

The method of polynomial reduction is a robust approach, one can be ensured that all possible equilibria will be found. This can be seen due to the fact that any solution of the equations in 7 is also a solution of at least one system of equations e.g. + . This fact allows these solutions to be seen as 'true' solutions up to numerical precision.

Here an example is presented of equilibria for a xed p_1 and p_3 while varying p_2.

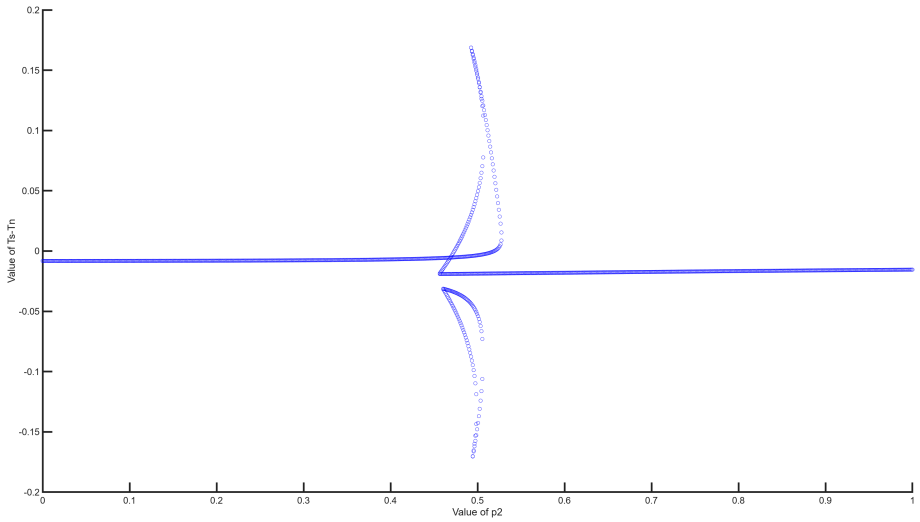

Figure 8: Scatter of values of T_s T_n over p_2 with xed $p_{1;n} = 1{:}5 = 1{:}01p_{1;s}$, and $p_{3;n} = p_{3;s} = 0{:}3$.

On the e ciency of the algorithm: it performs reliably on the given model as it relies on the established e cient solving methods for systems of polynomials. The method computes the equilibria for a given set of parameters. This also means that the method requires a ne step-size to accurately compute solutions near turning points of the branch.

Since the method computes equilibria without using any information about previous solutions, it is easily implemented to adopt parallel programming. This provides a signi cant gain in e ciency on systems that otherwise only allocate one core to the computations. This parallel computing has been implemented during the project and the method was allowed to use all available cores. In addition to the above, it is worth mentioning that this method scales poorly with the amount of boxes in the model. In particular, every connection between boxes adds an additional required case distinction doubling the amount of systems of equations. The three-box model contains two connections between boxes resulting in a total of four systems of equations but this rises exponentially with the amount of connections between boxes. If one were to expand on the model e.g. by including a box-network resembling the earth's di erent large oceans then this exponential growth becomes an issue.

3.2 Pseudo-arclength continuation

3.2.1 Introduction

In the general sense, arclength continuation is a method of computing zeros of a di erentiable function along a smooth branch [Dij05]. Given a function $f(x; \ )$ of x and arclength continuation seeks to solve for the values of x and such that $f(x; \ ) = 0$. A classical struggle with numerical methods consists of dealing with and around bifurcations. Pseudo-arclength continuation o ers a robust method capable of dealing with a plurality of types of bifurcations. This is approached by parameterizing solution values of x and i.e. $x(t)$ and (t) with $f(x(t); \ (t)) = 0$. The introduction of this parameter t is done such that the arclength of the resulting curve is xed, it is typically set to be equal to 1. This translates to the restriction

$$\dot{x}^T \dot{x} + \dot{-}^2 = 1 \tag{11}$$

where $\dot{x}$ denotes di erentiation with respect to t. This restriction does, however, introduces a hurdle as the derivatives $\dot{x}$ and $\dot{-}$ are not generally numerically available and hence a di erent approach is required in numerical settings. Indeed, given a solution-pair $(x_{i\ 1}; \ _{i\ 1})$ a suitable approximation to 11 can be made which does not require $\dot{x}$ nor $\dot{-}$:

$$\frac{x_i \quad x_{i\ 1}}{t)}^T (x_i \quad x_{i\ 1}) + \frac{_i \quad _{i\ 1}}{t}(\ _i \quad \ _{i\ 1}) = \quad t \tag{12}$$

The value of t de nes the step-size of the method and allows for control over accuracy of the computations. This accuracy of the method grows as t shrinks to 0 and this estimation to arclength is where pseudo-arclength continuation gets its name.

3.2.2 Implementation

As is not uncommon in numerical mathematics, pseudo-arclength continuation uses information about a previous point $(x_{i\ 1};\ _{i\ 1})$ to compute a next point $(x_i;\ _i)$. Indeed, it is possible to write:

$$f(x_i;\ _i) = 0 \tag{13a}$$

$$\frac{x_i \quad x_{i\ 1}}{t)}^T (x_i \quad x_{i\ 1}) + \frac{i \quad i\ 1}{t}(\ _i \quad\ _{i\ 1}) = \quad t: \tag{13b}$$

While this system of equations could be solved for $x_i;\ _i$ as is, many methods of solving 13a bene t from being supplied a 'smart' initial condition. For this purpose, a linear extrapolation of previous solutions was used i.e.:

$$x_i \quad \overline{x_i} := x_{i\ 1} + (x_{i\ 1} \quad x_{i\ 2}); \tag{14}$$

$$_i \quad \overline{\ _i} :=\ _{i\ 1} + (\ _{i\ 1} \quad\ _{i\ 2}); \tag{15}$$

where $\overline{x}$ and $\overline{\ }$ are the initial conditions supplied to the solver. Since these extrapolations depend on previously found solutions, a di erent approach is needed for the rst two solutions i.e. for $i = 1$ and $i = 2$. This addressed by exploiting the fact that, for this speci c case, solutions to trivial cases are analytically available as are their derivatives. This makes it so that x_0 is analytically available and $\overline{x_1} = x_0 + \underline{x}\ \ t$ can be used as an e cient initial condition.

3.2.3 Properties

Pseudo-arclength continuation performs well on the given model. More speci cally, it performs well on any given branch of the model. The extrapolation of previous solutions provides a strong initial condition which typically requires less than four iterations of the solver to converge con dently to a solution ($j \ xj \quad 10^{\ 6}$). Furthermore, the step-size t can made dynamic to increase precision when encountering a solution path with high curvature. A di erent signi cant feature of the method is the fact that it can avoid certain problems arising around bifurcation with classical methods. A method simply tracing the equilibria as a function of x and will fail when the solution path encounters a turning point, introducing multiple solutions for a single combination of x and . The restriction that the method only nds a single branch is a notable one; many systems exhibit equilibria on disconnected branches in the state-space. In addition to this, the method does not inherently detect the existence of disconnected branches. If one were to implement a method of detecting disconnected branches then this would allow pseudo-arclength continuation to provide a more complete set of solutions.

3.3 Comparison Polynomial Reduction and Pseudo-arclength Continuation

When comparing the methods of polynomial reduction and pseudo-arclength continuation it becomes apparent that they both provide satisfactory results in the context of small projects. The more one seeks to scale up the given model the more pseudo-arclength continuation increases in appeal. It computes equilibria with a signi cantly higher e ciency; any single iteration is completed approximately 5 times quicker. Do however note that polynomial reduction is easier adapted to perform computations in parallel to optimize any allocated computing power.

On the accuracy of the methods, polynomial reduction e ectively has near-perfect accuracy as no approximations in the model are required and all operations can be done up to machine precision. This is in contrast to the provided implementation of pseudo-arclength continuation as that does inherently requires approximations of the model to be made.

And nally, polynomial reduction is argued to be more approachable to be implemented which de nitely is an aspect to be considered depending on the use case.

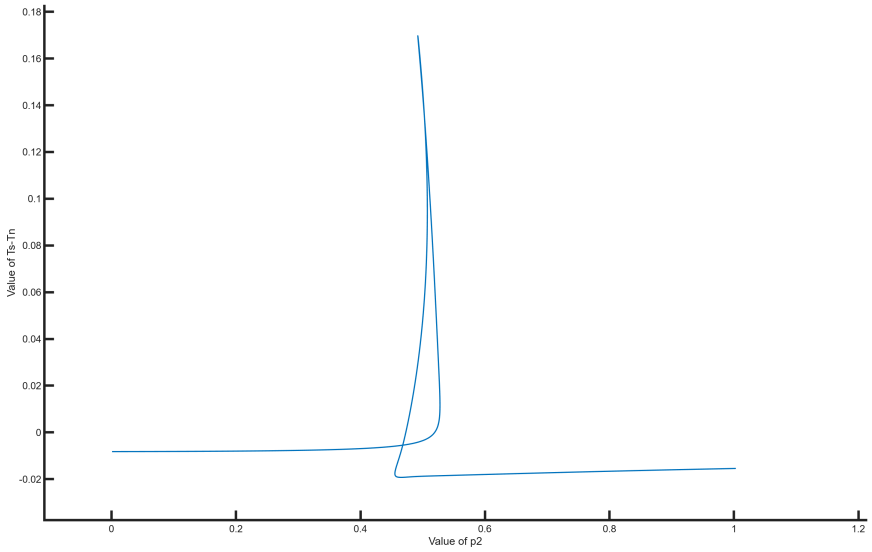

Figure 9: Values of T_s T_n over p_2 with xed $p_{1;n} = 1{:}5 = 1{:}01p_{1;s}$, and $p_{3;n} = p_{3;s} = 0{:}3$.

4 Conclusion and Further Research

This thesis considered di erent methods of nding equilibria through the lens of Stommel's box models. It can be concluded that both polynomial reduction and pseudo-arclength continuation are valid approaches to be implemented to compute equilibria of the models. Their use cases di er when one's project becomes more complex favoring arclength continuation. One feature that was not implemented in a general-use script is to combine the methods in a single algorithm. The overall e ciency of pseudo-arclength continuation can be utilized on single branches in the solution space. Polynomial reduction can then be used in parallel to aid detecting separate branches and computing solutions to seed the pseudo-arclength continuation method with.

This project was limited to implementing the described models and methods and explaining them to an undergraduate level audience. In particular, this thesis was written to complement Dijkstra's book [Dij05] on oceanography and hopes to provide some insight on which methods one could use when working with these or more complex and realistic models. That being said, the discussed methods are of course not bound to speci cally oceanographic models, the methods can nd a wide variety of use wherever the computation of equilibria is desired.

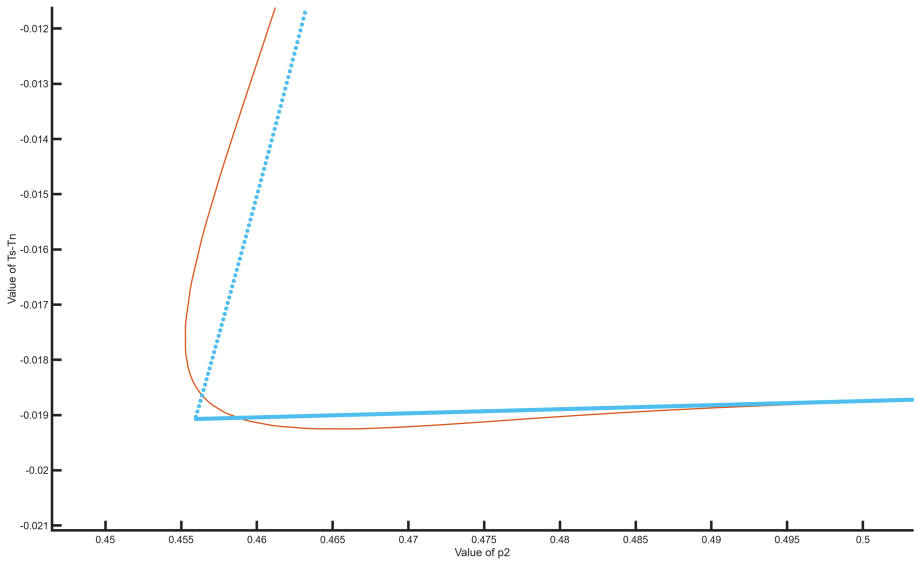

Figure 10: A zoomed in view of a corner comparing arclength continuation (= 0:01) in orange and the scatter of polynomial reduction in blue.

5 Appendix: Matlab Code

A variety of scripts and functions were developed during this project. A selection of core scripts was selected and is presented here. A variety of additional scripts concerning edge-cases and niche analyses are available upon request.

5.1 Code for Pseudo-arclength Continuation

Arclength1.m:
Main script for pseudo-arclength continuation over p_2 in Stommel's three-box model. This was used to generate gure 10 and 9.

```
syms t tprev yprev dp Tz Tn Sz Sn y dy dTz dTn dSz dSn p pprev Tzprev
    Tnprev Szprev Snprev mod1 mod2 real
%p=p2, p1,p3 fixed
tic
N=20000; %amount of steps
dt=1/400; %stepsize
%preallocate
Tzsol=zeros(N,1); %y=[Tz,Tn,Sz,Sn];
Tnsol=zeros(N,1);
Szsol=zeros(N,1);
Snsol=zeros(N,1);
psol=zeros(N,1);
tsol=zeros(N,1);

dTzsol=zeros(N,1); %y=[Tz,Tn,Sz,Sn];
dTnsol=zeros(N,1);
dSzsol=zeros(N,1);
dSnsol=zeros(N,1);
dpsol=zeros(N,1);


%fix p1, p3 if iterate over p2 (aSs, aSn).
p1=1.5;
```

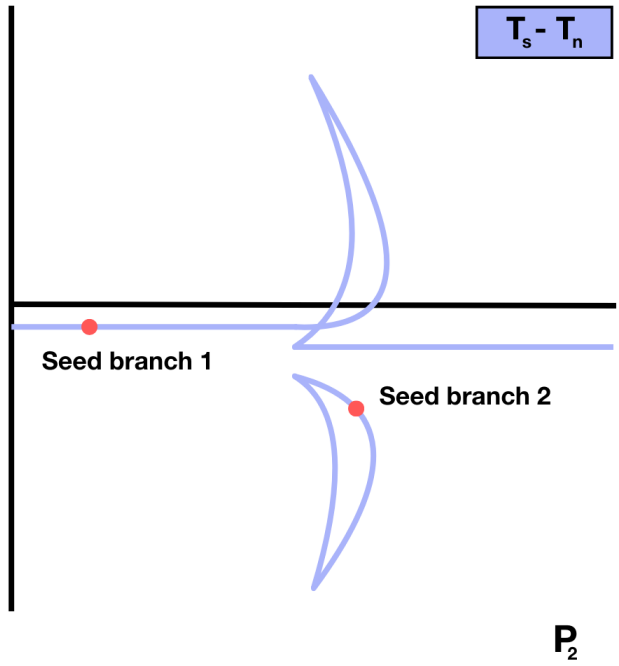

Figure 11: An illustration on how one could combine both methods to compute multiple branches with pseudo-arclength continuation.

```
p3=0.3;
aTs=p1;
aTn=p1*1.01;
% (aSz,aSn=p2)



%Determine tangent (first analytically available)
dpsol(1)=1;
dTzsol(1)=0.3;
dTnsol(1)=0.3;
dSzsol(1)=1;
dSnsol(1)=1;

%setup equations
eq11=aTs-Tz.*(1+3/4*cabs(Tz-Sz))-Tn.*(1/4*cabs(Tn-Sn))==0;
eq12=aTn-Tn.*(1+3/4*cabs(Tn-Sn))-Tz.*(1/4*cabs(Tz-Sz))==0;
eq13=p-Sz.*(p3+3/4*cabs(Tz-Sz))-Sn.*(1/4*cabs(Tn-Sn))==0;
eq14=p-Sn.*(p3+3/4*cabs(Tn-Sn))-Sz.*(1/4*cabs(Tz-Sz))==0;
eq1test=p==psol(1);
testsolve=vpasolve([eq11,eq12,eq13,eq14,eq1test]);

eq2=[dTz,dTn,dSz,dSn]*([Tz,Tn,Sz,Sn]-[Tzprev,Tnprev,Szprev,Snprev])'+
    dp*(p-pprev)-(t-tprev)==0;
%eq2=[dTz,dTn,dSz,dSn]*([Tz,Tn,Sz,Sn]-[Tzprev,Tnprev,Szprev,Snprev])
    '+(p-pprev)/(t-tprev)*(p-pprev)-(t-tprev)==0;
eq3=t-tprev==dt;

```

```
%initial conditions (analatically/numerically available)
psol(1)=testsolve.p;
tsol(1)=0;
Tzsol(1)=testsolve.Tz; %Tz
Tnsol(1)=testsolve.Tn; %Tn
Snsol(1)=testsolve.Sn; %Sz
Szsol(1)=testsolve.Sz; %Sn
tsol(1)=0;

eq11mod=aTs-Tz.*(1+3/4*mod1*(Tz-Sz))-Tn.*(1/4*mod2*(Tn-Sn))==0;
eq12mod=aTn-Tn.*(1+3/4*mod2*(Tn-Sn))-Tz.*(1/4*mod1*(Tz-Sz))==0;
eq13mod=p-Sz.*(p3+3/4*mod1*(Tz-Sz))-Sn.*(1/4*mod2*(Tn-Sn))==0;
eq14mod=p-Sn.*(p3+3/4*mod2*(Tn-Sn))-Sz.*(1/4*mod1*(Tz-Sz))==0;


ttest1=0;
ttest2=0;
for i=1:N
    %prepare equations by subbing in previous values
    eq11s=eq11;
    eq12s=eq12;
    eq13s=eq13;
    eq14s=eq14;
    %eq13s=subs(eq13,p,psol(i));
    %eq14s=subs(eq14,p,psol(i));
    eq2s=subs(eq2,[pprev,Tzprev,Tnprev,Szprev,Snprev,tprev,dp,dTz,dTn,
       dSz,dSn],[psol(i),Tzsol(i),Tnsol(i),Szsol(i),Snsol(i),tsol(i),
       dpsol(i),dTzsol(i,:),dTnsol(i,:),dSzsol(i,:),dSnsol(i,:)]);
    %eq3s=subs(eq3,tprev,tsol(i));
    %precision increase attempt
     eq3s=subs(eq3,[tprev,dt],[tsol(i),min(dt,max(min(abs(Tzsol(i)-
        Szsol(i))/3,abs(Tnsol(i)-Snsol(i))/3),1e-6))]);
    ttest1=ttest1+min(dt, max(min(abs(Tzsol(i)-Szsol(i)), abs(Tnsol(i)
       -Snsol(i))) , 1e-6));
    %solve for next values
    [ptemp,Tztemp,Tntemp,Sztemp,Sntemp,ttemp]=vpasolve([eq11s,eq12s,
       eq13s,eq14s,eq2s,eq3s],[p, Tz, Tn, Sz, Sn,t],[psol(i),Tzsol(i),
       Tnsol(i),Szsol(i),Snsol(i),tsol(i)]); %temporary storing of
       values. Also, maybe different solver
    if isempty(Tztemp)
        %If solutionfinding fails, try with loosened conditions and
           find
        %nearest
        i
        break
        sign1=1;
        sign2=1;
        if Tzsol(i)-Szsol(i)<0
            sign1=-1;
        end
        if Tnsol(i)-Szsol(i)<0
            sign2=-1;
        end
        eq11smod=subs(eq11mod,[mod1,mod2],[sign1,sign2]);
        eq12smod=subs(eq12mod,[mod1,mod2],[sign1,sign2]);
        eq13smod=subs(eq13mod,[mod1,mod2],[sign1,sign2]);
```

```
97          eq14smod=subs(eq14mod,[mod1,mod2],[sign1,sign2]);
98          [ptemp,Tztemp,Tntemp,Sztemp,Sntemp,ttemp]=vpasolve([eq11smod,
                eq12smod,eq13smod,eq14smod,eq2s,eq3s],[p, Tz, Tn, Sz, Sn,t
                ])
99          %[ptemp,Tztemp,Tntemp,Sztemp,Sntemp,ttemp]=vpasolve([eq11s,
                eq12s,eq13s,eq14s,eq2s,eq3s],[p, Tz, Tn, Sz, Sn,t],'random
                ',true);
100     end
101         id=1;
102     if length(Tztemp)>1
103         idtemp=1;
104         dist=100;
105         %linear estimate:
106         est=[psol(i),Tzsol(i),Tnsol(i),Szsol(i),Snsol(i),tsol(i)]+[
                dpsol(i),dTzsol(i),dTnsol(i),dSzsol(i),dSnsol(i),1]*(tsol(i
                )-tsol(i-1));
107         %for i2=1:length(Tztemp)
108             [idtemp,disttemp]=dsearchn([ptemp,Tztemp,Tntemp,Sztemp,
                    Sntemp,ttemp],est);
109             %[idtemp,disttemp]=dsearchn([ptemp(i2),Tztemp(i2),Tntemp(
                    i2),Sztemp(i2),Sntemp(i2),ttemp(i2)],[psol(i),Tzsol(i),
                    Tnsol(i),Szsol(i),Snsol(i),tsol(i)]);
110             % if disttemp<dist
111             %     dist=disttemp;
112             %     id=idtemp;
113             % end
114         %end
115     end
116 if mod(i,100)==0
117     i
118     ttest1
119 end
120 
121     psol(i+1)=ptemp(id);
122     Tzsol(i+1)=Tztemp(id);
123     Tnsol(i+1)=Tntemp(id);
124     Szsol(i+1)=Sztemp(id);
125     Snsol(i+1)=Sntemp(id);
126     tsol(i+1)=ttemp(id);
127     ttest2=ttest2+tsol(i+1);
128     psol(i+1)-psol(i);
129     dpsol(i+1)=(psol(i+1)-psol(i))/(tsol(i+1)-tsol(i));
130     dTzsol(i+1)=(Tzsol(i+1)-Tzsol(i))/(tsol(i+1)-tsol(i));
131     dTnsol(i+1)=(Tnsol(i+1)-Tnsol(i))/(tsol(i+1)-tsol(i));
132     dSzsol(i+1)=(Szsol(i+1)-Szsol(i))/(tsol(i+1)-tsol(i));
133     dSnsol(i+1)=(Snsol(i+1)-Snsol(i))/(tsol(i+1)-tsol(i));
134     % pold=pnew;
135     % yold=ynew;
136     % told=tnew;
137 
138     %equations lol again
139     % eq1=x^2+y^2-1==0;
140     % eq2=dy*(y-yold)+dx*(x-xold)-(t-told)==0;
141     % eq3=t-told==dt; %gonna need derivatives?
142     if psol(i)>1
143         break
```

```
    end
end
% funcheck1=@(Tz,Tn,Sz,Sn) aTs-Tz.*(1+3/4*abs(Tz-Sz))-Tn.*(1/4*abs(Tn-Sn));
% funcheck2=@(Tz,Tn,Sz,Sn) aTn-Tn.*(1+3/4*abs(Tn-Sn))-Tz.*(1/4*abs(Tz-Sz));
% funcheck3=@(Tz,Tn,Sz,Sn,aSs) aSs-Sz.*(p3+3/4*abs(Tz-Sz))-Sn.*(1/4*abs(Tn-Sn));
% funcheck4=@(Tz,Tn,Sz,Sn,aSn) aSn-Sn.*(p3+3/4*abs(Tn-Sn))-Sz.*(1/4*abs(Tz-Sz));
%
% funcheck1(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14))
% funcheck2(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14))
% funcheck3(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14),psol(14))
% funcheck4(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14),psol(14))

K=i;
hold on
plot(psol(2:K),Tzsol(2:K)-Tnsol(2:K))
%plot(psol(2:K),Tzsol(2:K)-Szsol(2:K))
%plot(psol(2:K),Tnsol(2:K)-Snsol(2:K))

hold off
toc

% hold on
% plot(psol(2:i),Tzsol(2:i)-Tnsol(2:i))
% hold off
% toc
```

5.2 Code for Polynomial Reduction Root Finding

Vparoot01.m:
Basic root finder for polynomial reduction method. This was used to generate figure 8 and 10.

```
function rootlist=vparoot01(aTs,aTn,aSs,aSn,p3)
syms Tz Tn Sz Sn %initiate symbol temperature and zinity (differences)
    for north and south
assume(Tz,'real')
assume(Tn,'real')
assume(Sz,'real')
assume(Sn,'real')
funcheck1=@(Tz,Tn,Sz,Sn) aTs-Tz.*(1+3/4*abs(Tz-Sz))-Tn.*(1/4*abs(Tn-Sn
   ));
funcheck2=@(Tz,Tn,Sz,Sn) aTn-Tn.*(1+3/4*abs(Tn-Sn))-Tz.*(1/4*abs(Tz-Sz
   ));
funcheck3=@(Tz,Tn,Sz,Sn) aSs-Sz.*(p3+3/4*abs(Tz-Sz))-Sn.*(1/4*abs(Tn-
   Sn));
funcheck4=@(Tz,Tn,Sz,Sn) aSn-Sn.*(p3+3/4*abs(Tn-Sn))-Sz.*(1/4*abs(Tz-
   Sz));

    %Seperate absolute value piecewise
    %constants taken from Dijkstra
    dTspospos= aTs-Tz.*(1+(3/4)*(Tz-Sz))-Tn.*((1/4)*(Tn-Sn))==0;
    dTnpospos= aTn-Tn.*(1+(3/4)*(Tn-Sn))-Tz.*((1/4)*(Tz-Sz))==0;
    dSspospos= aSs-Sz.*(p3+(3/4)*(Tz-Sz))-Sn.*((1/4)*(Tn-Sn))==0;
    dSnpospos= aSn-Sn.*(p3+(3/4)*(Tn-Sn))-Sz.*((1/4)*(Tz-Sz))==0;
    dTsnegpos= aTs-Tz.*(1-(3/4)*(Tz-Sz))-Tn.*((1/4)*(Tn-Sn))==0;
    dTnnegpos= aTn-Tn.*(1+(3/4)*(Tn-Sn))-Tz.*(-(1/4)*(Tz-Sz))==0;
    dSsnegpos= aSs-Sz.*(p3-(3/4)*(Tz-Sz))-Sn.*((1/4)*(Tn-Sn))==0;
    dSnnegpos= aSn-Sn.*(p3+(3/4)*(Tn-Sn))-Sz.*(-(1/4)*(Tz-Sz))==0;

    dTsposneg= aTs-Tz.*(1+(3/4)*(Tz-Sz))-Tn.*(-(1/4)*(Tn-Sn))==0;
    dTnposneg= aTn-Tn.*(1-(3/4)*(Tn-Sn))-Tz.*((1/4)*(Tz-Sz))==0;
    dSsposneg= aSs-Sz.*(p3+(3/4)*(Tz-Sz))-Sn.*(-(1/4)*(Tn-Sn))==0;
    dSnposneg= aSn-Sn.*(p3-(3/4)*(Tn-Sn))-Sz.*((1/4)*(Tz-Sz))==0;
    dTsnegneg= aTs-Tz.*(1-(3/4)*(Tz-Sz))-Tn.*(-(1/4)*(Tn-Sn))==0;
    dTnnegneg= aTn-Tn.*(1-(3/4)*(Tn-Sn))-Tz.*(-(1/4)*(Tz-Sz))==0;
    dSsnegneg= aSs-Sz.*(p3-(3/4)*(Tz-Sz))-Sn.*(-(1/4)*(Tn-Sn))==0;
    dSnnegneg= aSn-Sn.*(p3-(3/4)*(Tn-Sn))-Sz.*(-(1/4)*(Tz-Sz))==0;
rootspospos=vpasolve(dTspospos,dTnpospos, dSspospos,dSnpospos);
rootsnegpos=vpasolve(dTsnegpos,dTnnegpos,dSsnegpos,dSnnegpos);

rootsposneg=vpasolve(dTsposneg,dTnposneg, dSsposneg,dSnposneg);
rootsnegneg=vpasolve(dTsnegneg,dTnnegneg,dSsnegneg,dSnnegneg);




for i=length(rootspospos.Tz):-1:1
    if (rootspospos.Tz(i,1)<rootspospos.Sz(i,1) || rootspospos.Tn(i,1)
       <=rootspospos.Sn(i,1))
        rootspospos.Tz(i)=[];
        rootspospos.Sz(i)=[];
        rootspospos.Tn(i)=[];
        rootspospos.Sn(i)=[];
    end
```

```
end

for i=length(rootsposneg.Tz):-1:1
    if (rootsposneg.Tz(i,1)<rootsposneg.Sz(i,1) || rootsposneg.Tn(i,1)
       >=rootsposneg.Sn(i,1))
        rootsposneg.Tz(i)=[];
        rootsposneg.Sz(i)=[];
        rootsposneg.Tn(i)=[];
        rootsposneg.Sn(i)=[];
    end
end

for i=size(rootsnegpos.Tz,1):-1:1
    if (rootsnegpos.Tz(i,1)>rootsnegpos.Sz(i,1) || rootsnegpos.Tn(i,1)
       <=rootsnegpos.Sn(i,1) )
        rootsnegpos.Tz(i)=[];
        rootsnegpos.Sz(i)=[];
        rootsnegpos.Tn(i)=[];
        rootsnegpos.Sn(i)=[];
    end
end

for i=size(rootsnegneg.Tz,1):-1:1
    if (rootsnegneg.Tz(i,1)>rootsnegneg.Sz(i,1) || rootsnegneg.Tn(i,1)
       >=rootsnegneg.Sn(i,1) )
        rootsnegneg.Tz(i)=[];
        rootsnegneg.Sz(i)=[];
        rootsnegneg.Tn(i)=[];
        rootsnegneg.Sn(i)=[];
    end
end

 %f1=funcheck1(rootspospos.Tz(1),rootspospos.Tn(1),rootspospos.Sz(1),
    rootspospos.Sn(1))
 %f2=funcheck2(rootsposneg.Tz(1),rootsposneg.Tn(1),rootsposneg.Sz(1),
    rootsposneg.Sn(1))
 %f3=funcheck2(rootsnegneg.Tz(1),rootsnegneg.Tn(1),rootsnegneg.Sz(1),
    rootsnegneg.Sn(1))
 %f4=funcheck2(rootsnegpos.Tz(1),rootsnegpos.Tn(1),rootsnegpos.Sz(1),
    rootsnegpos.Sn(1))

% rootslist=zeros(length(rootsneg.Ts)+length(rootspos.Ts),4);
rootlist(:,1)=[rootspospos.Tz;rootsnegpos.Tz;rootsposneg.Tz;
   rootsnegneg.Tz];
rootlist(:,2)=[rootspospos.Tn;rootsnegpos.Tn;rootsposneg.Tn;
   rootsnegneg.Tn];
rootlist(:,3)=[rootspospos.Sz;rootsnegpos.Sz;rootsposneg.Sz;
   rootsnegneg.Sz];
rootlist(:,4)=[rootspospos.Sn;rootsnegpos.Sn;rootsposneg.Sn;
   rootsnegneg.Sn];
for i=1:size((rootlist),1)
    rootlist(i,5)=aTs;
    rootlist(i,6)=aTn;
    rootlist(i,7)=aSs;
    rootlist(i,8)=aSn;
    rootlist(i,9)=p3;
```

```
92  end
93
94
95  % for i=1:size(rootlist,1) %rootcheck debug
96  %     funcheck1(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
        ,4))
97  %     funcheck2(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
        ,4))
98  %     funcheck3(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
        ,4))
99  %     funcheck4(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
        ,4))
100 % end
```

5.3 Code for calling the Polynomial Reduction Root Finding

Main script calling basic root- nder for polynomial reduction method. This was used to gerenate gure 8 and 10.

```
n=501; %two hours for 50.000
N=n+1;
pmin=0.4;
pmax=0.6;
rootcell=cell(n+1,1);
rootbranch=cell(1);
sortdim=2; %Optional to make nicer plots to know in advance which
    value needs to be plotted for sorting
sortdim2=1;

tic
aTn=1.5;
aTs=1.5*1.01;
%This script will loop over values for aSn and aSs; namely p2;
p3=0.3;
parfor i=1:N
    p2=(pmax-pmin)/n*(i-1)+pmin;
    rootcell{i}=[vparoot01(aTs,aTn,p2,p2,p3)];
    rootcell{i}=sortrows(rootcell{i},sortdim);
end

rootcount=0; %amount of found roots
[lastcount,~]=size(rootcell{1}); %amount of roots found by i'th
    parameter value.
branchcount=0; %amount of finished branches
for i=1:length(rootcell)
    [jmax,~]=size(rootcell{i}); %amount of roots by i'th parameter
        value.
    if jmax==lastcount
            rootcount=rootcount+1;
        for j=1:jmax
            rootbranch{branchcount+j}(rootcount,1:9)=rootcell{i}(j,:);
        end


    else
        rootcount=1;
        branchcount=branchcount+lastcount;
        [lastcount,~]=size(rootcell{i});
        jmax=lastcount;
        for j=1:jmax
            rootbranch{branchcount+j}(rootcount,1:9)=rootcell{i}(j,:);
        end

    end
end
% rootarray(rootcount+1:end,:)=[];
toc
hold on
for i=1:size(rootbranch,2)
    scatter(rootbranch{i}(:,7),rootbranch{i}(:,sortdim)-rootbranch{i
        }(:,sortdim2))
```

```
end
hold off
%sort data for human eyes maybe
% scatter(rootline1(:,3),rootline1(:,1))
% hold on
% scatter(rootline2(:,3),rootline2(:,1))
% scatter(rootline3(:,3),rootline3(:,1))
% scatter(rootline4(:,3),rootline4(:,1))
```

```
% rootarray=rootline1;
% if exist("rootline2","var")
%     rootarray=[rootline1;rootline2];
%     if exist("rootline3","var")
%         rootarray=[rootline1;flip(rootline2,1);rootline3];
%         if exist("rootline4","var")
%             rootarray=[rootline1;flip(rootline2,1);rootline3;
   rootline4];
%         end
%     end
% end
% plot(rootarray(:,3),rootarray(:,2))
% hold on
% plot(rootarray(:,3),rootarray(:,1))
```

5.4 Code for Absolute Value-approximation

Small script enabling a choice for absolute value-approximation. This was used to generate gure 10 and 9.

```
function cabs=cabs(x)
%choices for e.g. differentiable absolute values
epsilon=0.01;
cabs=sqrt(x^2+epsilon)-sqrt(epsilon);
%cabs=x*tanh(1/epsilon*x);
%cabs=abs(x);
%multiple choices for testing
end
```

References

[Dij05] H. Dijkstra. *Nonlinear Physical Oceanography*. Springer Dordrecht, 2005.

[Sto61] Henry Stommel. Thermohaline convection with two stable regimes of ow. *Tellus*, 13(2):224{230, 1961.

[vA07] Hendrik M. van Aken. *The Oceanic Thermohaline Circulation: An Introduction*. Springer New York, NY, 2007.