



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 | 4 |
| 2 | Stommel's box models | 5 |
| 2.1 | The Two-Box Model | 5 |
| 2.1.1 | Model construction | 5 |
| 2.2 | The Three-Box Model | 8 |
| 2.2.1 | Model construction | 8 |
| 2.3 | Equilibria | 9 |
| 2.3.1 | Introduction and problem statement | 9 |
| 2.3.2 | Modes of Equilibria | 10 |
| 3 | Finding Equilibria | 10 |
| 3.1 | Polynomial Reduction | 11 |
| 3.1.1 | Introduction | 11 |
| 3.1.2 | Implementation | 11 |
| 3.1.3 | Properties | 12 |
| 3.2 | Pseudo-arclength continuation | 13 |
| 3.2.1 | Introduction | 13 |
| 3.2.2 | Implementation | 14 |
| 3.2.3 | Properties | 14 |
| 3.3 | Comparison Polynomial Reduction and Pseudo-arclength Continuation | 14 |
| 4 | Conclusion and Further Research | 15 |
| 5 | Appendix: Matlab Code | 16 |
| 5.1 | Code for Pseudo-arclength Continuation | 16 |
| 5.2 | Code for Polynomial Reduction Root Finding | 21 |
| 5.3 | Code for calling the Polynomial Reduction Root Finding | 24 |
| 5.4 | Code for Absolute Value-approximation | 26 |

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, effects and mechanisms of changes in our ocean's characteristics [vA07].

This interest has given rise to a variety of different models in an attempt to better understand and predict the behaviour of the ocean and its influences. 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:

$$\frac{dT_1}{dt} = \nu_T(aT_1 - T_1), \quad (1a)$$

$$\frac{dS_1}{dt} = \nu_S(aS_1 - S_1). \quad (1b)$$

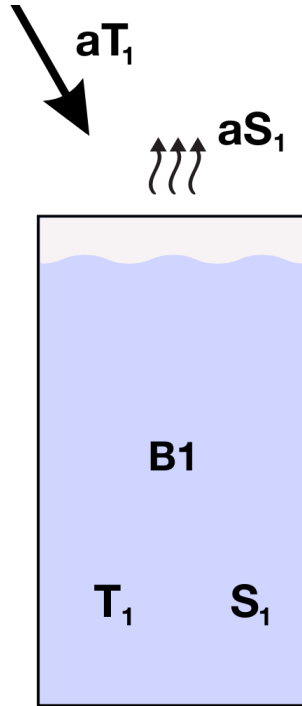


Figure 1: A Single Box B_1 with a temperature T_1 , a salinity S_1 an ambient temperature aT_1 , and ambient salinity aS_1 .

Here ν_T and ν_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 T_1 and salinity S_1 both converge to their respective ambient values (figure 2).

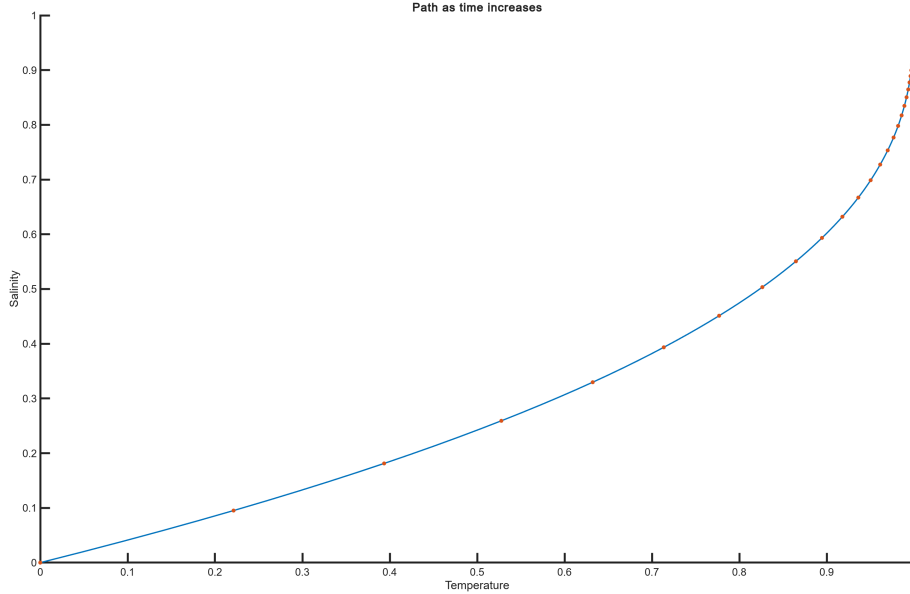


Figure 2: Values of T_1 and S_1 converging to aT_1 and aS_1 both set equal to 1 with equidistant points in time highlighted.

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

$$\frac{dT_2}{dt} = \nu_T(aT_2 - T_2), \quad (2a)$$

$$\frac{dS_2}{dt} = \nu_S(aS_2 - S_2). \quad (2b)$$

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

Indeed, flow ϕ will be encapsulated into the model after first defining a density:

$$\rho_x = \rho_0 + \rho_0 \epsilon_S (S_x - S_0) - \rho_0 \epsilon_T (T_x - T_0),$$

where ρ_0 , T_0 , S_0 are reference values for a density, temperature and salinity respectively. And c_T and c_S are waters' contraction coefficients for temperature and salinity.

Now it is possible to write:

$$\phi = \gamma \cdot \frac{\rho_1 - \rho_2}{\rho_0},$$

where γ is a constant describing the viscosity of water.

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

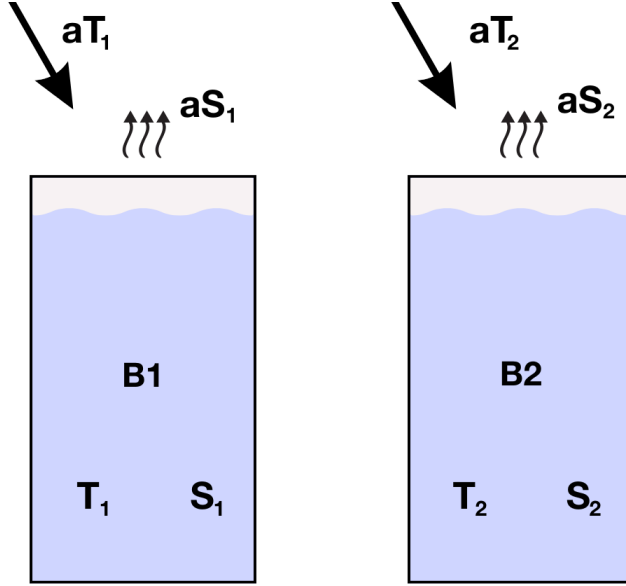


Figure 3: A Second Box B_2 with temperatures T_1 and T_2 , salinities S_1 and S_2 ambient temperatures aT_1 and aT_2 , and ambient salinities aS_1 and aS_2 .

is now possible to construct the following:

$$\frac{dT_1}{dt} = \nu_T(aT_1 - T_1) + (T_2 - T_1) \cdot |\phi|, \quad (3a)$$

$$\frac{dT_2}{dt} = \nu_T(aT_2 - T_2) + (T_1 - T_2) \cdot |\phi|, \quad (3b)$$

$$\frac{dS_1}{dt} = \nu_S(aS_1 - S_1) + (S_2 - S_1) \cdot |\phi|, \quad (3c)$$

$$\frac{dS_2}{dt} = \nu_S(aS_2 - S_2) + (S_1 - S_2) \cdot |\phi|. \quad (3d)$$

The model is then typically condensed by defining $T := T_1 - T_2$ and $S := S_1 - S_2$. 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 influence 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 $T_1 - T_2$ does. These were specifically prevalent when implementing periodic (seasonal) perturbations in the system's parameters.

Finally, the mentioned condensation results in the following:

$$\frac{dT}{dt} = \frac{dT_1}{dt} - \frac{dT_2}{dt} = \nu_T(aT_1 - aT_2 - T) - 2T \cdot |\phi|, \quad (4a)$$

$$\frac{dS}{dt} = \frac{dS_1}{dt} - \frac{dS_2}{dt} = \nu_S(aS_1 - aS_2 - S) - 2S \cdot |\phi|. \quad (4b)$$

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

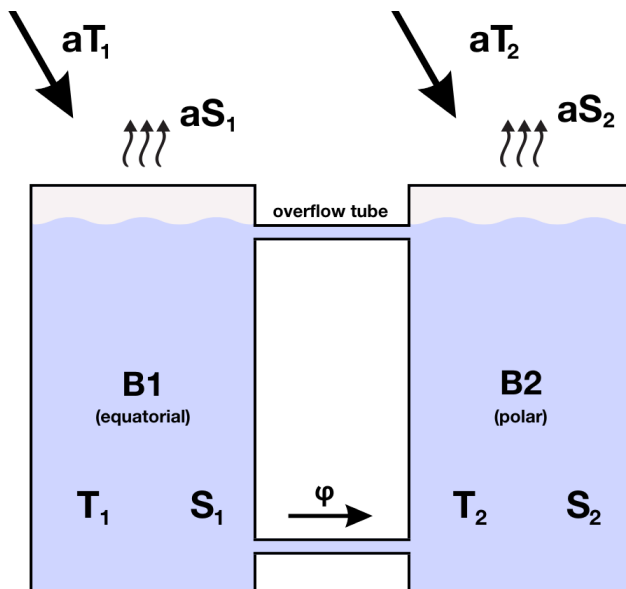


Figure 4: An illustration of Stommel's two-box model.

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

$$\frac{dT}{dt} = p_1 - T(1 + |T - S|), \quad (5a)$$

$$\frac{dS}{dt} = p_2 - S(p_3 + |T - S|), \quad (5b)$$

where ϕ has been reduced to $\phi = T - S$ 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 different 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 reflected in the model: The expanded model allows the north and south waters to adhere to their own ambient temperature- and salinity. And indeed, the flow from equatorial to northern waters need not be symmetric to the flow from southern to equatorial waters.

2.2.1 Model construction

A couple of clarifications to construct the differential 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 influenced equally under equal flow-conditions. The impact of an incoming flow 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 define the boundaries between the volumes of water. In the two-box model, this was implicitly chosen to be 1 : 1. However here, a ratio

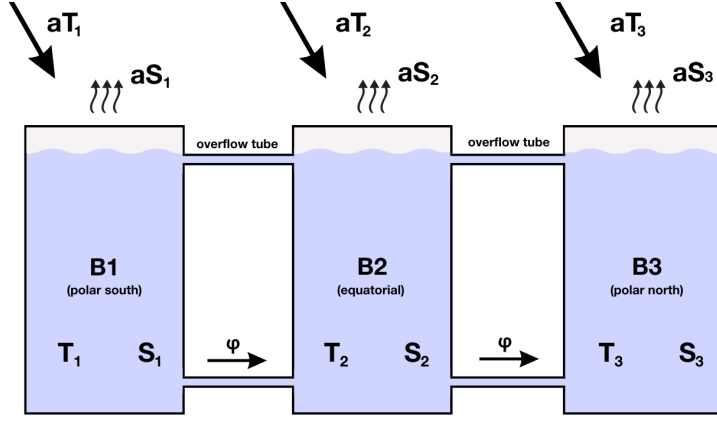


Figure 5: Illustration of Stommel's three-box model.

of $1 : 2 : 1$ is chosen to reflect 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 specific 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 differences of these values with boxes:

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

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 simplified resulting in the following:

$$\frac{dT_s}{dt} = p_{1,s} - T_s \left(1 + \frac{3}{4}|T_s - S_s|\right) - T_n \left(\frac{1}{4}|T_n - S_n|\right), \quad (7a)$$

$$\frac{dT_n}{dt} = p_{1,n} - T_n \left(1 + \frac{3}{4}|T_n - S_n|\right) - T_s \left(\frac{1}{4}|T_s - S_s|\right), \quad (7b)$$

$$\frac{dS_s}{dt} = p_{2,s} - S_s \left(p_{3,s} + \frac{3}{4}|T_s - S_s|\right) - S_n \left(\frac{1}{4}|T_n - S_n|\right), \quad (7c)$$

$$\frac{dS_n}{dt} = p_{2,n} - T_s \left(p_{3,n} + \frac{3}{4}|T_s - S_s|\right) - S_s \left(\frac{1}{4}|T_s - S_s|\right), \quad (7d)$$

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 reflect 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 $\frac{df}{dt} = f(\bar{x})$ it is of interest to find the zeros of f ; the points that satisfy $f(\bar{x}) = 0$. 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.

$$\frac{dT}{dt} = p_1 - T(1 + |T - S|) = 0, \quad (8a)$$

$$\frac{dS}{dt} = p_2 - S(p_3 + |T - S|) = 0. \quad (8b)$$

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 finding more generalizable insights.

The equations as given could be fed into established numerical solvers like Matlab’s root-finder. 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 first tries (and in this case, fails) to solve the equations symbolically after which it then tries find solutions based on different numerical root-finding methods. These methods suffer from possibly failing to find 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 $|T - S|$ terms and the critical points where $T - S$ would otherwise changes sign. These absolute values introduce areas in the state-space where the model equations are not differentiable. A property that prevents typical solving methods of converging efficiently to a solution.

Dijkstra addressed this problem in [Dij05] by introducing an approximation to the absolute-value function that is differentiable and hence improves root-finding behaviour around points in the state-space where $T \approx S$. This approach is implemented and compared to a different 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. Specifically, 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 defined by a relatively simple polynomial function. These polynomials can be solved with more efficient 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 different 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 reflects the earth’s oceans more accurately. When working with the three-box model, four modes of ocean flow can be recognized: The flows 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 different 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 $|T_* - S_*|$ is prevalent in the model and is not differentiable in either T_* or S_* . Many numerical solving methods do not exhibit convergent behaviour when supplied with such terms. Hence, it is of great benefit 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 first 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 differentiable approximation of $|T_* - S_*|$. 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 $|x| \approx \sqrt{x^2 + \epsilon^2} - \epsilon$ for some small ϵ (7). This is a relatively simple approximation which comes with

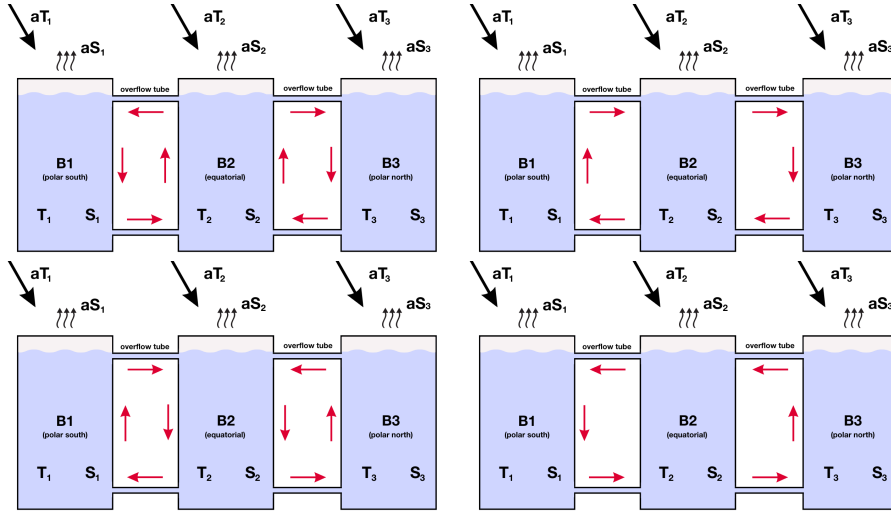


Figure 6: The four main modes of equilibria in Stommel's three-box model.

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

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 beneficial 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:

$$|x| = \begin{cases} x, & \text{for } x > 0 \\ -x, & \text{for } x \leq 0 \end{cases}$$

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:

$$\frac{dT_s}{dt} = \begin{cases} p_{1,s} - T_s(1 + \frac{3}{4}(T_s - S_s)) - T_n(\frac{1}{4}(T_n - S_n)), & \text{for } T_s > S_s \text{ and } T_n > S_n \\ p_{1,s} - T_s(1 - \frac{3}{4}(T_s - S_s)) - T_n(\frac{1}{4}(T_n - S_n)), & \text{for } T_s \leq S_s \text{ and } T_n > S_n \\ p_{1,s} - T_s(1 + \frac{3}{4}(T_s - S_s)) + T_n(\frac{1}{4}(T_n - S_n)), & \text{for } T_s > S_s \text{ and } T_n \leq S_n \\ p_{1,s} - T_s(1 - \frac{3}{4}(T_s - S_s)) + T_n(\frac{1}{4}(T_n - S_n)), & \text{for } T_s \leq S_s \text{ and } T_n \leq S_n \end{cases} \quad (9)$$

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

Define

$$\frac{dT_s}{dt} ++ := p_{1,s} - T_s(1 + \frac{3}{4}(T_s - S_s)) - T_n(\frac{1}{4}(T_n - S_n)),$$

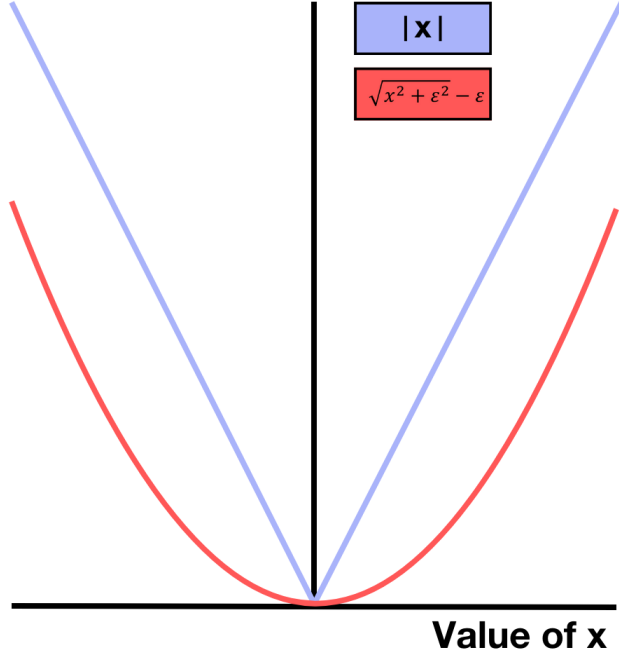


Figure 7: Figure illustrating the approximation of $|x|$ with $\sqrt{x^2 + \epsilon^2} - \epsilon$.

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:

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

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

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

$$\frac{dS_n}{dt}_{++} = p_{2,n} - S_s \left(p_{3,n} + \frac{3}{4}(T_n - S_n)\right) - S_s \left(\frac{1}{4}(T_s - S_s)\right). \quad (10d)$$

These form a system of second-order polynomials of which efficient solving methods are readily available. These solutions are then verified 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 efficient 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 fixed p_1 and p_3 while varying p_2 .

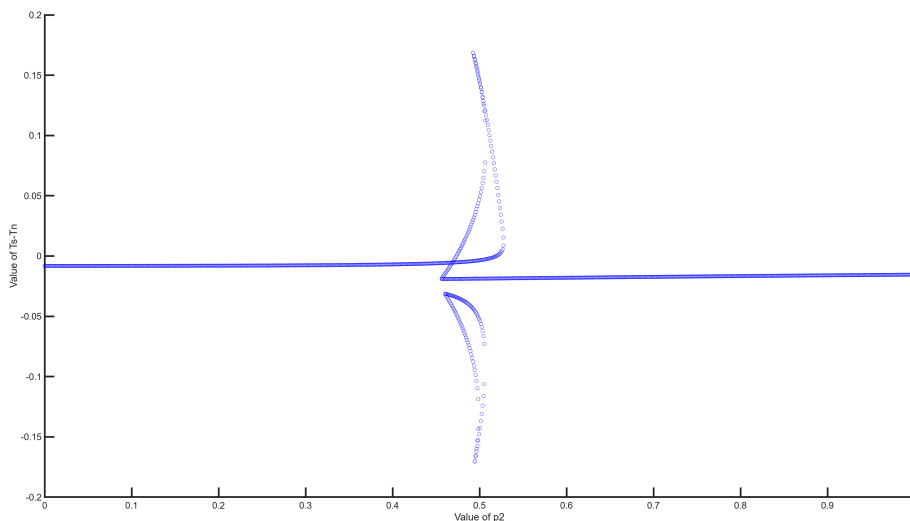


Figure 8: Scatter of values of $T_s - T_n$ over p_2 with fixed $p_{1,n} = 1.5 = 1.01p_{1,s}$, and $p_{3,n} = p_{3,s} = 0.3$

On the efficiency of the algorithm: it performs reliably on the given model as it relies on the established efficient 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 fine 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 significant gain in efficiency 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 different 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 differentiable function along a smooth branch [Dij05]. Given a function $f(x, \lambda)$ of x and λ arclength continuation seeks to solve for the values of x and λ such that $f(x, \lambda) = 0$. A classical struggle with numerical methods consists of dealing with and around bifurcations. Pseudo-arclength continuation offers a robust method capable of dealing with a plurality of types of bifurcations. This is approached by parameterizing solution values of x and λ i.e. $x(t)$ and $\lambda(t)$ with $f(x(t), \lambda(t)) = 0$. The introduction of this parameter t is done such that the arclength of the resulting curve is fixed, it is typically set to be equal to 1. This translates to the restriction

$$\dot{x}^T \dot{x} + \dot{\lambda}^2 = 1 \quad (11)$$

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

$$\left(\frac{x_i - x_{i-1}}{\Delta t} \right)^T (x_i - x_{i-1}) + \frac{\lambda_i - \lambda_{i-1}}{\Delta t} (\lambda_i - \lambda_{i-1}) = \Delta t \quad (12)$$

The value of Δt defines the step-size of the method and allows for control over accuracy of the computations. This accuracy of the method grows as Δt 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}, λ_{i-1}) to compute a next point (x_i, λ_i) . Indeed, it is possible to write:

$$f(x_i, \lambda_i) = 0 \quad (13a)$$

$$\left(\frac{x_i - x_{i-1}}{\Delta t}\right)^T (x_i - x_{i-1}) + \frac{\lambda_i - \lambda_{i-1}}{\Delta t} (\lambda_i - \lambda_{i-1}) = \Delta t. \quad (13b)$$

While this system of equations could be solved for x_i, λ_i as is, many methods of solving 13a benefit from being supplied a 'smart' initial condition. For this purpose, a linear extrapolation of previous solutions was used i.e.:

$$x_i \approx \bar{x}_i := x_{i-1} + (x_{i-1} - x_{i-2}), \quad (14)$$

$$\lambda_i \approx \bar{\lambda}_i := \lambda_{i-1} + (\lambda_{i-1} - \lambda_{i-2}), \quad (15)$$

where \bar{x} and $\bar{\lambda}$ are the initial conditions supplied to the solver. Since these extrapolations depend on previously found solutions, a different approach is needed for the first two solutions i.e. for $i = 1$ and $i = 2$. This addressed by exploiting the fact that, for this specific case, solutions to trivial cases are analytically available as are their derivatives. This makes it so that x_0 is analytically available and $\bar{x}_1 = x_0 + \dot{x}\Delta t$ can be used as an efficient initial condition.

3.2.3 Properties

Pseudo-arclength continuation performs well on the given model. More specifically, 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 confidently to a solution ($|\Delta x| \approx 10^{-6}$). Furthermore, the step-size Δt can be made dynamic to increase precision when encountering a solution path with high curvature. A different significant 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 λ will fail when the solution path encounters a turning point, introducing multiple solutions for a single combination of x and λ . The restriction that the method only finds 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 significantly higher efficiency; 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 effectively 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 require approximations of the model to be made.

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

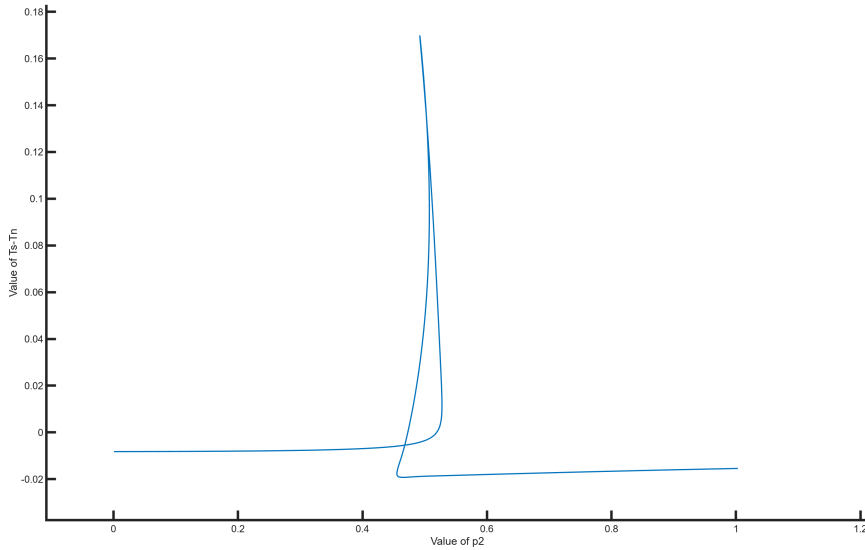


Figure 9: Values of $T_s - T_n$ over p_2 with fixed $p_{1,n} = 1.5 = 1.01p_{1,s}$, and $p_{3,n} = p_{3,s} = 0.3$.

4 Conclusion and Further Research

This thesis considered different methods of finding 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 differ 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 efficiency 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 specifically oceanographic models, the methods can find a wide variety of use wherever the computation of equilibria is desired.

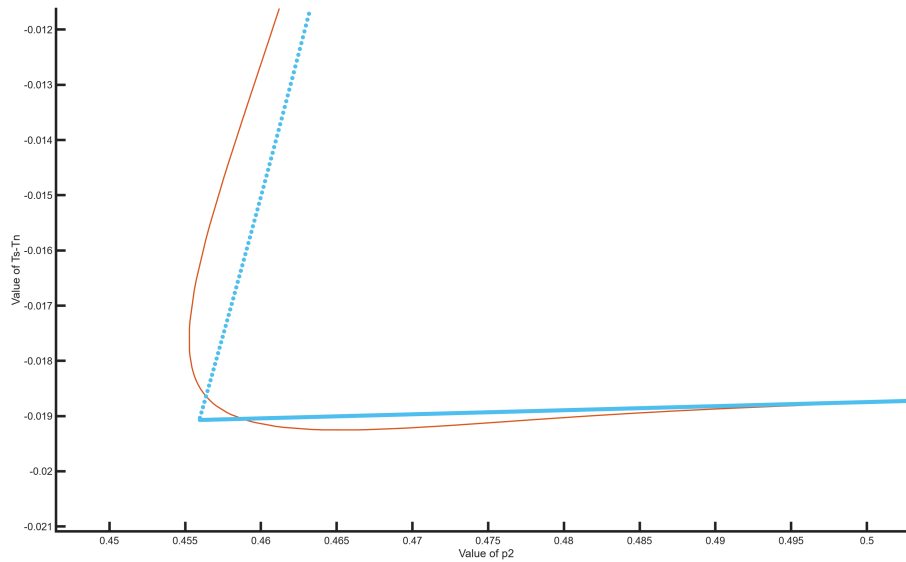


Figure 10: A zoomed in view of a corner comparing arclength continuation ($\epsilon = 0.01$) in orange and the scatter of polynomial reduction in blue.

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 figure 10 and 9.

```

1  syms t tprev yprev dp Tz Tn Sz Sn y dy dTz dTn dSz dSn p pprev Tzprev
    Tnprev Szprev Snprev mod1 mod2 real
2  %p=p2, p1,p3 fixed
3  tic
4  N=20000; %amount of steps
5  dt=1/400; %stepsize
6  %preallocate
7  Tzsol=zeros(N,1); %y=[Tz,Tn,Sz,Sn];
8  Tnsol=zeros(N,1);
9  Szsol=zeros(N,1);
10 Snsol=zeros(N,1);
11 psol=zeros(N,1);
12 tsol=zeros(N,1);
13
14 dTzsol=zeros(N,1); %y=[Tz,Tn,Sz,Sn];
15 dTnsol=zeros(N,1);
16 dSzsol=zeros(N,1);
17 dSnsol=zeros(N,1);
18 dpsol=zeros(N,1);
19
20
21 %fix p1, p3 if iterate over p2 (aSs, aSn).
22 p1=1.5;

```

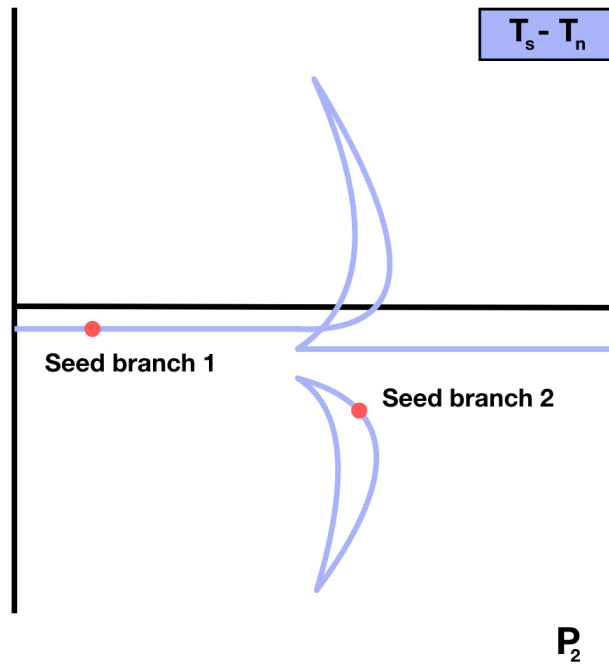



Figure 11: An illustration on how one could combine both methods to compute multiple branches with pseudo-arclength continuation.

```

23 p3=0.3;
24 aTs=p1;
25 aTn=p1*1.01;
26 % (aSsz, aSn=p2)
27
28
29
30 %Determine tangent (first analytically available)
31 dpsol(1)=1;
32 dTzsol(1)=0.3;
33 dTnsol(1)=0.3;
34 dSzsol(1)=1;
35 dSnsol(1)=1;
36
37 %setup equations
38 eq11=aTs-Tz.*(1+3/4*cabs(Tz-Sz))-Tn.*(1/4*cabs(Tn-Sn))==0;
39 eq12=aTn-Tn.*(1+3/4*cabs(Tn-Sn))-Tz.*(1/4*cabs(Tz-Sz))==0;
40 eq13=p-Sz.*(p3+3/4*cabs(Tz-Sz))-Sn.*(1/4*cabs(Tn-Sn))==0;
41 eq14=p-Sn.*(p3+3/4*cabs(Tn-Sn))-Sz.*(1/4*cabs(Tz-Sz))==0;
42 eq1test=p==psol(1);
43 testsolve=vpasolve([eq11,eq12,eq13,eq14,eq1test]);
44
45 eq2=[dTz,dTn,dSz,dSn]*([Tz,Tn,Sz,Sn]-[Tzprev,Tnprev,Szprev,Snprev])'+
    dp*(p-pprev)-(t-tprev)==0;
46 %eq2=[dTz,dTn,dSz,dSn]*([Tz,Tn,Sz,Sn]-[Tzprev,Tnprev,Szprev,Snprev])
    '+ (p-pprev)/(t-tprev)*(p-pprev)-(t-tprev)==0;
47 eq3=t-tprev==dt;
48

```

```

49 %initial conditions (analatically/numerically available)
50 psol(1)=testsolve.p;
51 tsol(1)=0;
52 Tzsol(1)=testsolve.Tz; %Tz
53 Tnsol(1)=testsolve.Tn; %Tn
54 Snsol(1)=testsolve.Sn; %Sz
55 Szsol(1)=testsolve.Sz; %Sn
56 tsol(1)=0;
57
58 eq11mod=aTs-Tz.*(1+3/4*mod1*(Tz-Sz))-Tn.*(1/4*mod2*(Tn-Sn))==0;
59 eq12mod=aTn-Tn.*(1+3/4*mod2*(Tn-Sn))-Tz.*(1/4*mod1*(Tz-Sz))==0;
60 eq13mod=p-Sz.*(p3+3/4*mod1*(Tz-Sz))-Sn.*(1/4*mod2*(Tn-Sn))==0;
61 eq14mod=p-Sn.*(p3+3/4*mod2*(Tn-Sn))-Sz.*(1/4*mod1*(Tz-Sz))==0;
62
63
64 ttest1=0;
65 ttest2=0;
66 for i=1:N
67     %prepare equations by subbing in previous values
68     eq11s=eq11;
69     eq12s=eq12;
70     eq13s=eq13;
71     eq14s=eq14;
72     %eq13s=subs(eq13,p,psol(i));
73     %eq14s=subs(eq14,p,psol(i));
74     eq2s=subs(eq2,[pprev,Tzprev,Tnprev,Szprev,Snprev,tprev,dp,dTz,dTn,
75         dSz,dSn],[psol(i),Tzsol(i),Tnsol(i),Szsol(i),Snsol(i),tsol(i),
76         dpsol(i),dTzsol(i,:),dTnsol(i,:),dSzsol(i,:),dSnsol(i,:)]);
77     %eq3s=subs(eq3,tprev,tsol(i));
78     %precision increase attempt
79     eq3s=subs(eq3,[tprev,dt],[tsol(i),min(dt,max(min(abs(Tzsol(i)-
80         Szsol(i))/3,abs(Tnsol(i)-Snsol(i))/3),1e-6))]);
81     ttest1=ttest1+min(dt,max(min(abs(Tzsol(i)-Szsol(i)),abs(Tnsol(i)-
82         Snsol(i))),1e-6));
83     %solve for next values
84     [ptemp,Tztemp,Tntemp,Sztemp,Sntemp,ttemp]=vpasolve([eq11s,eq12s,
85         eq13s,eq14s,eq2s,eq3s],[p,Tz,Tn,Sz,Sn,t],[psol(i),Tzsol(i),
86         Tnsol(i),Szsol(i),Snsol(i),tsol(i)]); %temporary storing of
87         values. Also, maybe different solver
88     if isempty(Tztemp)
89         %If solutionfinding fails, try with loosened conditions and
90         find
91         %nearest
92         i
93         break
94         sign1=1;
95         sign2=1;
96         if Tzsol(i)-Szsol(i)<0
97             sign1=-1;
98         end
99         if Tnsol(i)-Szsol(i)<0
100             sign2=-1;
101         end
102         eq11smod=subs(eq11mod,[mod1,mod2],[sign1,sign2]);
103         eq12smod=subs(eq12mod,[mod1,mod2],[sign1,sign2]);
104         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

```

```

144     end
145 end
146 % funcheck1=@(Tz,Tn,Sz,Sn) aTs-Tz.*(1+3/4*abs(Tz-Sz))-Tn.*(1/4*abs(Tn-
147 % funcheck2=@(Tz,Tn,Sz,Sn) aTn-Tn.*(1+3/4*abs(Tn-Sn))-Tz.*(1/4*abs(Tz-
148 % funcheck3=@(Tz,Tn,Sz,Sn,aSs) aSs-Sz.*(p3+3/4*abs(Tz-Sz))-Sn.*(1/4*
149 % funcheck4=@(Tz,Tn,Sz,Sn,aSn) aSn-Sn.*(p3+3/4*abs(Tn-Sn))-Sz.*(1/4*
150 %
151 % funcheck1(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14))
152 % funcheck2(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14))
153 % funcheck3(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14),psol(14))
154 % funcheck4(Tzsol(14),Tnsol(14),Szsol(14),Snsol(14),psol(14))
155
156 K=i;
157 hold on
158 plot(psol(2:K),Tzsol(2:K)-Tnsol(2:K))
159 %plot(psol(2:K),Tzsol(2:K)-Szsol(2:K))
160 %plot(psol(2:K),Tnsol(2:K)-Snsol(2:K))
161
162 hold off
163 toc
164
165 % hold on
166 % plot(psol(2:i),Tzsol(2:i)-Tnsol(2:i))
167 % hold off
168 % toc

```

5.2 Code for Polynomial Reduction Root Finding

Vparoot01.m:

Basic rootfinder for polynomial reduction method. This was used to generate figure 8 and 10.

```

1 function rootlist=vparoot01(aTs,aTn,aSs,aSn,p3)
2 syms Tz Tn Sz Sn %initiate symbol temperature and zinity (differences)
   %for north and south
3 assume(Tz,'real')
4 assume(Tn,'real')
5 assume(Sz,'real')
6 assume(Sn,'real')
7 funcheck1=@(Tz,Tn,Sz,Sn) aTs-Tz.*(1+3/4*abs(Tz-Sz))-Tn.*(1/4*abs(Tn-Sn)
   );
8 funcheck2=@(Tz,Tn,Sz,Sn) aTn-Tn.*(1+3/4*abs(Tn-Sn))-Tz.*(1/4*abs(Tz-Sz)
   );
9 funcheck3=@(Tz,Tn,Sz,Sn) aSs-Sz.*(p3+3/4*abs(Tz-Sz))-Sn.*(1/4*abs(Tn-
   Sn));
10 funcheck4=@(Tz,Tn,Sz,Sn) aSn-Sn.*(p3+3/4*abs(Tn-Sn))-Sz.*(1/4*abs(Tz-
   Sz));
11
12 %Seperate absolute value piecewise
13 %constants taken from Dijkstra
14 dTspospos= aTs-Tz.*(1+(3/4)*(Tz-Sz))-Tn.*((1/4)*(Tn-Sn))==0;
15 dTnpospos= aTn-Tn.*(1+(3/4)*(Tn-Sn))-Tz.*((1/4)*(Tz-Sz))==0;
16 dSspospos= aSs-Sz.*(p3+(3/4)*(Tz-Sz))-Sn.*((1/4)*(Tn-Sn))==0;
17 dSnpospos= aSn-Sn.*(p3+(3/4)*(Tn-Sn))-Sz.*((1/4)*(Tz-Sz))==0;
18 dTsnegpos= aTs-Tz.*(1-(3/4)*(Tz-Sz))-Tn.*((1/4)*(Tn-Sn))==0;
19 dTnnegpos= aTn-Tn.*(1+(3/4)*(Tn-Sn))-Tz.*(-(1/4)*(Tz-Sz))==0;
20 dSsnegpos= aSs-Sz.*(p3-(3/4)*(Tz-Sz))-Sn.*((1/4)*(Tn-Sn))==0;
21 dSnnegpos= aSn-Sn.*(p3+(3/4)*(Tn-Sn))-Sz.*(-(1/4)*(Tz-Sz))==0;
22
23 dTsposneg= aTs-Tz.*(1+(3/4)*(Tz-Sz))-Tn.*(-(1/4)*(Tn-Sn))==0;
24 dTnposneg= aTn-Tn.*(1-(3/4)*(Tn-Sn))-Tz.*((1/4)*(Tz-Sz))==0;
25 dSsposneg= aSs-Sz.*(p3+(3/4)*(Tz-Sz))-Sn.*(-(1/4)*(Tn-Sn))==0;
26 dSnposneg= aSn-Sn.*(p3-(3/4)*(Tn-Sn))-Sz.*((1/4)*(Tz-Sz))==0;
27 dTsnegneg= aTs-Tz.*(1-(3/4)*(Tz-Sz))-Tn.*(-(1/4)*(Tn-Sn))==0;
28 dTnnegneg= aTn-Tn.*(1-(3/4)*(Tn-Sn))-Tz.*(-(1/4)*(Tz-Sz))==0;
29 dSsnegneg= aSs-Sz.*(p3-(3/4)*(Tz-Sz))-Sn.*(-(1/4)*(Tn-Sn))==0;
30 dSnnegneg= aSn-Sn.*(p3-(3/4)*(Tn-Sn))-Sz.*(-(1/4)*(Tz-Sz))==0;
31 rootspospos=vpasolve(dTspospos,dTnpospos,dSspospos,dSnpospos);
32 rootsnegpos=vpasolve(dTsnegpos,dTnnegpos,dSsnegpos,dSnnegpos);
33
34 rootsposneg=vpasolve(dTsposneg,dTnposneg,dSsposneg,dSnposneg);
35 rootsnegneg=vpasolve(dTsnegneg,dTnnegneg,dSsnegneg,dSnnegneg);
36
37
38
39
40 for i=length(rootspospos.Tz):-1:1
41     if (rootspospos.Tz(i,1)<rootspospos.Sz(i,1) || rootspospos.Tn(i,1)
42         <=rootspospos.Sn(i,1))
43         rootspospos.Tz(i)=[];
44         rootspospos.Sz(i)=[];
45         rootspospos.Tn(i)=[];
46         rootspospos.Sn(i)=[];
47     end

```

```

47 end
48
49 for i=length(rootsposneg.Tz):-1:1
50     if (rootsposneg.Tz(i,1)<rootsposneg.Sz(i,1) || rootsposneg.Tn(i,1)
51         >=rootsposneg.Sn(i,1))
52         rootsposneg.Tz(i)=[];
53         rootsposneg.Sz(i)=[];
54         rootsposneg.Tn(i)=[];
55         rootsposneg.Sn(i)=[];
56     end
57 end
58 for i=size(rootsnegpos.Tz,1):-1:1
59     if (rootsnegpos.Tz(i,1)>rootsnegpos.Sz(i,1) || rootsnegpos.Tn(i,1)
60         <=rootsnegpos.Sn(i,1) )
61         rootsnegpos.Tz(i)=[];
62         rootsnegpos.Sz(i)=[];
63         rootsnegpos.Tn(i)=[];
64         rootsnegpos.Sn(i)=[];
65     end
66 end
67 for i=size(rootsnegneg.Tz,1):-1:1
68     if (rootsnegneg.Tz(i,1)>rootsnegneg.Sz(i,1) || rootsnegneg.Tn(i,1)
69         >=rootsnegneg.Sn(i,1) )
70         rootsnegneg.Tz(i)=[];
71         rootsnegneg.Sz(i)=[];
72         rootsnegneg.Tn(i)=[];
73         rootsnegneg.Sn(i)=[];
74     end
75 end
76 %f1=funcheck1(rootspospos.Tz(1),rootspospos.Tn(1),rootspospos.Sz(1),
77     rootspospos.Sn(1))
78 %f2=funcheck2(rootsposneg.Tz(1),rootsposneg.Tn(1),rootsposneg.Sz(1),
79     rootsposneg.Sn(1))
80 %f3=funcheck2(rootsnegneg.Tz(1),rootsnegneg.Tn(1),rootsnegneg.Sz(1),
81     rootsnegneg.Sn(1))
82 %f4=funcheck2(rootsnegpos.Tz(1),rootsnegpos.Tn(1),rootsnegpos.Sz(1),
83     rootsnegpos.Sn(1))
84
85 % rootslist=zeros(length(rootsneg.Ts)+length(rootspos.Ts),4);
86 rootlist(:,1)=[rootspospos.Tz;rootsnegpos.Tz;rootsposneg.Tz;
87     rootsnegneg.Tz];
88 rootlist(:,2)=[rootspospos.Tn;rootsnegpos.Tn;rootsposneg.Tn;
89     rootsnegneg.Tn];
90 rootlist(:,3)=[rootspospos.Sz;rootsnegpos.Sz;rootsposneg.Sz;
91     rootsnegneg.Sz];
92 rootlist(:,4)=[rootspospos.Sn;rootsnegpos.Sn;rootsposneg.Sn;
93     rootsnegneg.Sn];
94 for i=1:size((rootlist),1)
95     rootlist(i,5)=aTs;
96     rootlist(i,6)=aTn;
97     rootlist(i,7)=aSs;
98     rootlist(i,8)=aSn;
99     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
97 %     ,4))
97 %     funcheck2(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
98 %     ,4))
98 %     funcheck3(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
99 %     ,4))
99 %     funcheck4(rootlist(i,1),rootlist(i,2),rootlist(i,3),rootlist(i
100 % end
```

5.3 Code for calling the Polynomial Reduction Root Finding

Main script calling basic root-finder for polynomial reduction method. This was used to generate figure 8 and 10.

```
1 n=501; %two hours for 50.000
2 N=n+1;
3 pmin=0.4;
4 pmax=0.6;
5 rootcell=cell(n+1,1);
6 rootbranch=cell(1);
7 sortdim=2; %Optional to make nicer plots to know in advance which
   value needs to be plotted for sorting
8 sortdim2=1;
9
10 tic
11 aTn=1.5;
12 aTs=1.5*1.01;
13 %This script will loop over values for aSn and aSs; namely p2;
14 p3=0.3;
15 parfor i=1:N
16     p2=(pmax-pmin)/n*(i-1)+pmin;
17     rootcell{i}=[vparoot01(aTs,aTn,p2,p2,p3)];
18     rootcell{i}=sortrows(rootcell{i},sortdim);
19 end
20
21 rootcount=0; %amount of found roots
22 [lastcount,~]=size(rootcell{1}); %amount of roots found by i'th
   parameter value.
23 branchcount=0; %amount of finished branches
24 for i=1:length(rootcell)
25     [jmax,~]=size(rootcell{i}); %amount of roots by i'th parameter
   value.
26     if jmax==lastcount
27         rootcount=rootcount+1;
28         for j=1:jmax
29             rootbranch{branchcount+j}(rootcount,1:9)=rootcell{i}(j,:);
30         end
31
32     else
33         rootcount=1;
34         branchcount=branchcount+lastcount;
35         [lastcount,~]=size(rootcell{i});
36         jmax=lastcount;
37         for j=1:jmax
38             rootbranch{branchcount+j}(rootcount,1:9)=rootcell{i}(j,:);
39         end
40     end
41 end
42 end
43 % rootarray(rootcount+1:end,:)=[];
44 toc
45 hold on
46 for i=1:size(rootbranch,2)
47     scatter(rootbranch{i}(:,7),rootbranch{i}(:,sortdim)-rootbranch{i}
   }(:,sortdim2))
```



```

49 end
50 hold off
51 %sort data for human eyes maybe
52 % scatter(rootline1(:,3),rootline1(:,1))
53 % hold on
54 % scatter(rootline2(:,3),rootline2(:,1))
55 % scatter(rootline3(:,3),rootline3(:,1))
56 % scatter(rootline4(:,3),rootline4(:,1))
57
58
59
60
61 % rootarray=rootline1;
62 % if exist("rootline2","var")
63 %     rootarray=[rootline1;rootline2];
64 %     if exist("rootline3","var")
65 %         rootarray=[rootline1;flip(rootline2,1);rootline3];
66 %         if exist("rootline4","var")
67 %             rootarray=[rootline1;flip(rootline2,1);rootline3;
        rootline4];
68 %         end
69 %     end
70 % end
71 % plot(rootarray(:,3),rootarray(:,2))
72 % hold on
73 % 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 figure 10 and 9.

```
1 function cabs=cabs(x)
2 %choices for e.g. differentiable absolute values
3 epsilon=0.01;
4 cabs=sqrt(x^2+epsilon)-sqrt(epsilon);
5 %cabs=x*tanh(1/epsilon*x);
6 %cabs=abs(x);
7 %multiple choices for testing
8 end
```

References

- [Dij05] H. Dijkstra. *Nonlinear Physical Oceanography*. Springer Dordrecht, 2005.
- [Sto61] Henry Stommel. Thermohaline convection with two stable regimes of flow. *Tellus*, 13(2):224–230, 1961.
- [vA07] Hendrik M. van Aken. *The Oceanic Thermohaline Circulation: An Introduction*. Springer New York, NY, 2007.