Mid-term report

Quantifying numerical diffusion and dispersion in D-Flow Flexible Mesh usinga lock-exchange experiment

by

D. Koetsenruijter

Student number:	4332601	
Project duration:	April 20, 2020 – June 12, 2020	
Thesis committee:	ir. L.M. Keyzer,	TU Delft, first supervisor
	Prof. dr. J. Pietrzak,	TU Delft, second supervisor

Contents

1 Introduction

Due to temperature and salinity differences density driven currents can occur. Because important water systems such as the drinking water supply and large coastal water systems in the Netherlands are strongly influenced by these currents it is important to understand these phenomena. With the use of D-Flow FM, a software package by Deltares, the mixture of salt- and fresh water can be numerically approximated. However, a side effect of these numerical flow models is the occurrence of numerical diffusion and dispersion caused by characteristics of the numerical discretization scheme that is used and the type of flow it is used to model.

Numerical diffusion is sometimes referred to as “numerical viscosity” since the associated approximation errors mimic the effect of an increase in viscosity, i.e. the solution is overdamped. Besides, numerical dispersion is related to unrealistic oscillations in an approximation of an advection-diffusion problem that may occur if stability of the solution is not ensured or if the observed phenomena are not smooth enough for the discretized solutions of the problem posed. Because avoiding such errors requires contrasting measures within the numerical model a quantification of their responses to certain modelling parameters is desirable. [O’Brien et al., 1950, Zijlema, 2015]

To be able to refer the sensitivity of numerical errors produced by D-Flow FM to specific settings and parameters of the model a lock-exchange experiment is set up. In a lock-exchange the initial situation consists of a rectangular tank with a free water equally leveled surface, for one half containing a fluid with a higher density than on the other half of the fluid tank, split vertically. In this research the density difference is caused by a salinity difference. At the start of the experiment (t=0) these fluids set in motion and from two wavefronts, a negative and a positive as a result of the gravitational and buoyant forces. The numerical errors produced during such an experiment have often been investigated and this research aims to use the results obtained from this experiment as a measure for the order of accuracy of the D-Flow FM numerical flow model.

Firstly, in modelling flow and transport phenomena, and numerical modelling in general, two variables have a major effect on the stability and accuracy of the solution; the spatial resolution and time step size. Additionally there are many different other numerical and flow related parameters in D-Flow FM that can be set and might have influence on the numerical errors produced. These will all be varied within relevant ranges during four stages of the simulation phase of this research; temporal variations, variations in spatial resolution, variations in the initial salinity difference and variations of other numerical parameters. Secondly, with respect to the lock-exchange experiment three characteristic variables are distinguished; the frontal wave speed, the thickness of the mixing layer and the water level. The goal of this research is thus to relate the numerical dispersion and -diffusion observed in the output of these characteristic lock-exchange variables to the change of the temporal and spatial resolution, the forcing and other numerical parameters of D-Flow FM.

To this end a sensitivity analysis per parameter is performed, where the sensitivity S of a parameter P is defined as the relative change of a state variable per change of this parameter: $S = (\delta x/x)/(\delta P/P)$. State variables are defined as the numerical dispersion and -diffusion observed in the output of the characteristic variables of the lock-exchange experiment. To quantify the errors of interest first the observed results will be normalized to the varying parameter and plotted against the observed errors. For example to quantify the errors observed while varying the resolution in x-, y- and z-direction the ratio of the cell-lenght, -width and -height will be considered to enable computation times to stay relatively similar. Hereafter they may be related to certain indicator parameters such as the Froude and Reynolds numbers at characteristic locations of the lock-exchange experiment. Finally, available and possibly relevant indicator parameters are the amount of times a cell was Courant limiting, the computed Richardson numbers and other numerical conditions. Depending on the parameters' relevance to each they are proposed to serve as an explanation for the observed errors. Subsequently the observed data is analyzed and compared to results found in different in order to verify and explain the computed results.

This document describes the research questions and hypotheses in the second chapter, subsequently it will provide some background information related to the lock-exchange, to modelling of flow and transport in general and to D-Flow Flexible Mesh in chapter three. Then, in chapter four the method used to attain answers to the research question(s) is presented. The results are presented in chapter five. Results with respect to obtaining a reference model which can be used as a constant basis throughout the simulations are discussed in chapter 5.1, the results of the interval analysis in chapter 5.2, the simulations of temporal, spatial, forcing and numerical parameters in chapter 5.3, numerical evaluation criteria in chapter 5.4, if it is performed the spectral analysis in chapter 5.5 en the sensitivity analysis in chapter 5.6. Finally the results discussed and compared to available literature in chapter six, after which also a conclusion is drawn concerning the research question(s).

2 Research Question & Hypothesis

2.1. Research questions

1. How much numerical diffusion and or dispersion does the D-Flow FM model produce when simulating a 3D lock-exchange experiment?

    1. What is the order of errors produced by D-Flow FM as a result of numerical diffusion and/or dispersion?
    2. How sensitive is the accuracy of the D-Flow FM model to time, space and numerically related parameters?
    3. What sort of errors are produced given different parameters?
    4. What parameters in the D-Flow FM model have the largest influence on errors related to numerical diffusion and dispersion?
    5. Are there indicators that predict the occurrence of numerical diffusion and dispersion in D-Flow FM?
    6. How do the physics around the mixing layer develop and what influence does this have on the accuracy numerical approximation in terms of numerical diffusion and -dispersion?
    7. How do internal waves develop at different model parameters and settings?
    8. What effect do internal waves have on the produced numerical errors?

3 Background

To be able to refer the sensitivity of numerical errors produced by D-Flow FM to specific settings and parameters of the model a lock-exchange experiment is set up. In a lock-exchange the initial situation consists of a rectangular tank with a free water equally leveled surface, for one half containing a fluid with a higher density than on the other half of the fluid tank, split vertically. In this research the density difference is caused by a salinity difference. At the start of the experiment (t=0) these fluids set in motion and from two wavefronts, a negative and a positive as a result of the gravitational and buoyant forces. The numerical errors produced during such an experiment have often been investigated and this research aims to use the results obtained from this experiment as a measure for the order of accuracy of the D-Flow FM numerical flow model.

In general with finite difference methods used to approximate convection problems a hydrostatic assumption gives errors related to vertical accelerations (i.e. velocity gradients). Because the continuity equation can only account for small variations in the vertical velocity gradient when larger accelerations occur numerical models show physically unrealistic phenomena such as water level elevations and density differences at unsuspected locations. For the case of the lock-exchange the same is valid: depending on the velocity variations seen near the mixing zone the hydrostatic assumption is valid, this often means the solution should be smoothly varying. For this reason near the negative and positive wave fronts (see 3.2) and near the interface between the two density driven currents numerical errors may be expected due to respectively steep salinity gradients causing large discontinuities and high shear stresses causing turbulence thus vertical velocity gradient.

For most flows in a lock exchange, depending mostly on the Reynolds number [Shin et al., 2004], resistance can be ignored. Also described in [Battjes, 2017], when damping of wave amplitude is as little as possible the energy conserving assumption is most valid and uniform flow behind the wavefront can be assumed. By examening the flow based on the above assumptions numerical errors may be be recognized at locations where the hydrostatic assumption is least valid. For example at t=0 because of

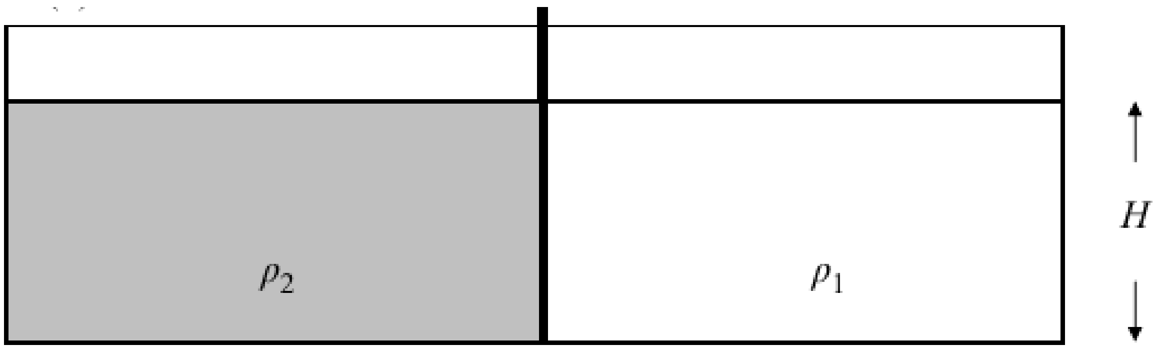


Figure 3.1: Basic lock-exchange setup - Source:Shin et al. [2004]

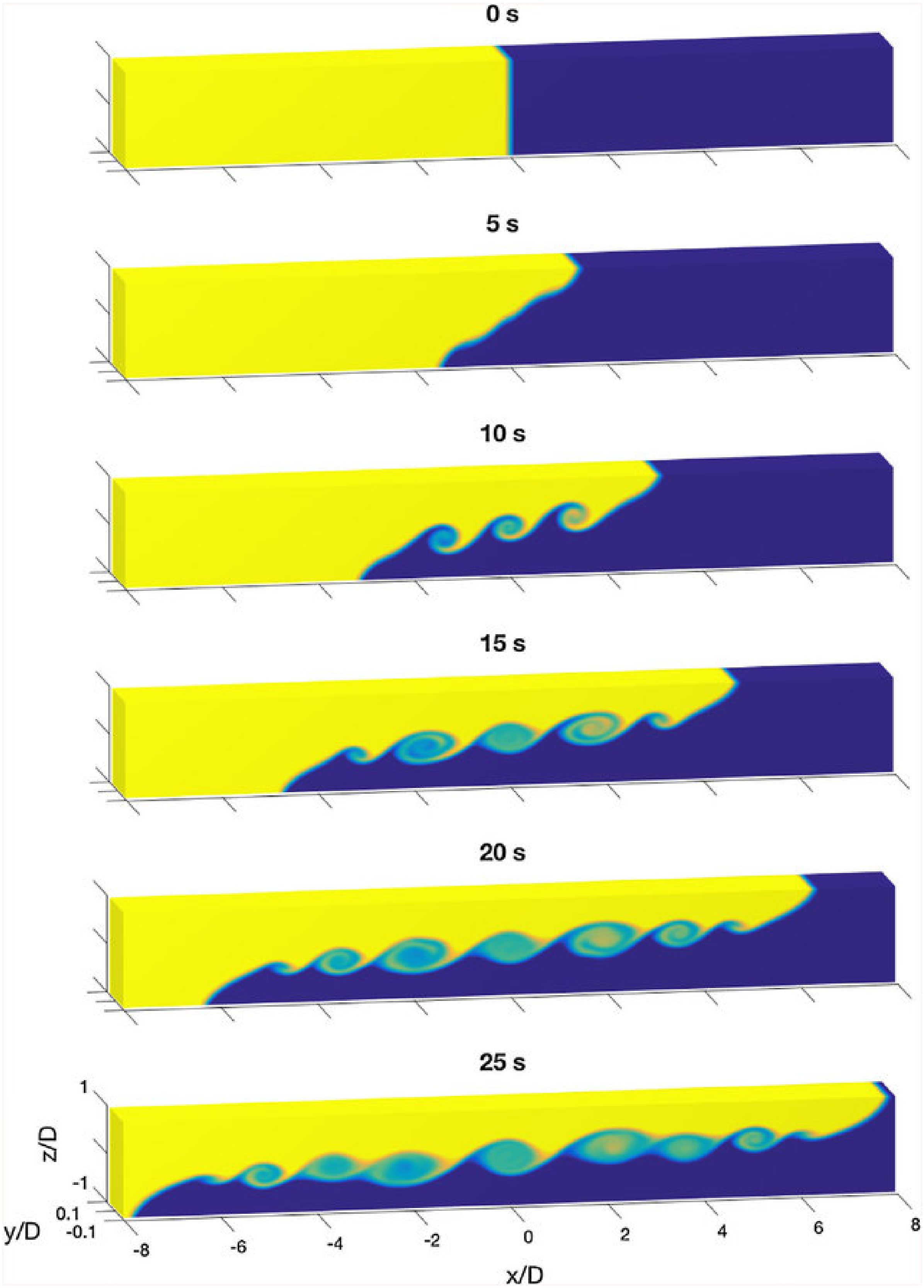


Figure 3.2: Lock-exchange experiment, flow over time - Source:Garcia et al. [2019]

the large initial disturbance a relatively large velocity gradient between the two sides of the front occurs. Because of this large discontinuity this is where numerical dispersion may be at its extreme. On the contrary, where internal waves are formed numerical diffusion may be significant if the model flattens out these small vertical velocity gradients. Lastly, if the frontal wave speed (the propagation velocity of the disturbances) does not comply with estimations as proposed by literature [Pietrzak, 1998] it could be an indication of numerical diffusion.

Because D-Flow FM implements many complex details of a numerical scheme only a few important aspects are shortly described here. For further elaboration the reader is referred to Appendix B - D-Flow Flexible Mesh or [Deltares, 2020]. D-Flow FM implements the shallow water equations that have as basic assumptions that the vertical velocity is small compared to horizontal scale, the vertical pressure is nearly hydrostatic and wave induced accelerations in the vertical cause negligible effect on the pressure distribution compared to the wave induced height difference. Further the shallow water equations have as property that the water density is directly proportional to bed- and wind friction terms. D-Flow FM uses an finite differences method with an upwind scheme (implicit) to solve the governing equations. Most importantly D-Flow is capable of implementing many sorts of meshgrid types, including triangles, quids, pentagons and hexagons in a curvilinear grid.

4 Method

To support answers to the research questions with data and reasonable arguments and thus finally obtain an indication or even quantification of the numerical dispersion and -diffusion produced by D-Flow FM the following research method is presented. It consists of four phases; modelling, running simulations over a range of values per parameter, analyzing the results using a sensitivity analysis and appropriate indicator parameters and finally discussing the obtained results. Based upon the result of these phases the report then aims to conclude by giving answers to the research questions.

4.1. Reference model

During the modelling phase one determines a reference model through modelling large ranges of desirably constant parameters. Based on a qualitative analysis of these results the model settings and parameters that will remain constant throughout the further research are reasonably defined.

First the terms and settings found in the basic D-Flow FM model are explored. In this reference model the observed numerical dispersion and -diffusion should be of an acceptable range and more importantly subject to little change. In the initial reference model a basic setup of the boundary conditions, the width, lenght and depth of the model and timescale of the simulation will be determined. This will take a few iterations and requires basic post processing of the results to asses the characteristic variables of the different simulations, in this case the temperature and number of layers in the vertical direction are considered. Through the results presented in this report it becomes clear which settings give minimal numerical diffusion and dispersion.

In order to provide a solid basis for the sensitivity analysis the reference model has to be sufficiently robust to enable all parameters to be explored. Besides after every parameter variation constant value for this parameter should be logically determined. To ensure a working model first most default settings will be used to set up the lock-exchange model. In this model the following parameters are thus defined; general settings such as working directory, model name etc. The basic geometry including lenght, width and depth. Basic physics such as water density, gravity acceleration etc. Boundary conditions being closed free-slip boundaries for either velocity or discharge. A computational grid with n- and m-number of cells in x- and y-direction respectively. And finally if desired location specific monitoring points with a higher frequency of output can be determined to reduce the memory load of the model.

With respect to post-processing two parameters are evaluated; the width averaged water level over time and the width averaged salinity over time. These state variables indicate numerical errors and enable classification of the observed error. For the salinity profiles points where they either have an extreme (physically unrealistic) value are highlighted and locations where characteristic values of the solution can be expected are inspected in more detail.

After reasonable accuracy and stability of the simulation is ensured the initial reference model is thus defined, although some final aspects of the model may be refined. Possibly more realistic physical properties, boundary conditions, a more relevant geometry or specifics of the flow model such as bed friction will be set based on the qualitative analysis of the produced results. However, most of these are expected to be left at their default values and are otherwise clearly substantiated with reason or result.

4.2. Simulations

In order to generate the data required for the sensitivity analysis a number of simulations have to be performed. By varying the different parameters the characteristic variables, defined as the frontal wavespeed, the thickness of the mixing layer and the water level, the associated numerical errors can be computed. Because this is done over a realistic and relevant range of values overall numerical diffusion and -dispersion produced by D-Flow FM in general models can be quantified and accordingly conclusions based on flow related- and numerical indicators may be drawn.

4.2.1. Interval analysis

At the start of the data generation the research strategy has almost been fully developed. But some important questions remain, This includes the variable ranges over which the chosen parameters will be defined. In order to obtain a reasonable range per parameter an interval analysis will be performed. This analysis will provide insight in what results come from different ranges of parameters and how applicable certain combinations of ranges are. Moreover it will provide insight in how many experiments have to be run.

4.2.2. Parameter variations

Based on variation in temperature and number of layers in the z-direction an initial reference model for the temporal variations is determined. After simulations with a varying time step size and Courant number and processing of these results a constant time step size for the next parameter variations will be chosen. Hereafter the spatial resolution is examined. Because of the third order factor of space it is expected that monitoring points are desirable in order to maintain realistic computation times. These monitoring points have to be set up at logically sound locations within the reference model and will give a high resolution insight in the development of characteristic variables at specific locations over time. Locations of interest will be an area through which a stable wave front passes, the boundary conditions at both ends of the model and near the mixing layer where internal waves are expected to form and thus high vertical velocities may occur.

For the parameter variations the sequential order matters as it is desirable to have a constant time step for all variations and therefore to have investigated the effect of the Courant number in combination with D-Flow FM's automatic time step setting before setting this constant time step. Because with respect to the spatial resolution the number of layers is prematurely examined in the reference model and considering a relatively equal computation time per variation is desired, the ratio between Δz and respectively Δx and Δy is varied for two different Δz values. This way while examening $\Delta x/\Delta z$ we can adjust Δy to maintain reasonable computation times and thus by varying Δz smaller resolutions in x- and y-direction can be investigated. This will be don first for the default cube shaped meshgrid and later verified for different types of meshgrid, including triangles, quids, pentagons and hexagons. Hereafter the forcing parameter, the initial salinity difference, will be investigated and lastly other parameters can be considered.

For the sake of clarity all variations are now repeated in the order they are investigated; For the temporal variation two aspects of the parameter are considered, being; the Courant number and the time step size. For the spatial variation five aspects of the resolution are considered, being; the ratios $\Delta x/\Delta z1$, $\Delta x/\Delta z2$, $\Delta y/\Delta z1$, $\Delta y/\Delta z2$ with similar computation times and the meshgrid type. The difference in salinity and finally possibly other parameters listed below:

- Viscosity
- Bed friction
- Entrainment (mixing terms)
- Advection scheme
- Solver parameters

For the experiments to be able to run the simulation procedure based on the reference model and the post-processing is automated as much as possible using Python. These scripts can be found in D.

4.3. Data analysis

The data analysis consists of a qualitative analysis of the results and accordingly a just quantification of the observed errors in order for the errors to be used in the sensitivity analysis. During the qualitative analysis the results of all the post processing of the different performed experiments are analyzed and a relation between the sensitivity of the observed numerical errors to the change in certain parameters are sought for, either physically or numerically. In order to correctly quantify the observed errors it will be determined what indicator parameters are of significance and what numerical conditions are applicable. Hereafter a sensitivity analysis is performed, where the sensitivity of a parameter S(P) is defined as the relative change of a state variable per change of this parameter: $S = (\delta x/x)/(\delta P/P)$. State variables are defined as the numerical dispersion and -diffusion observed in the output of the characteristic variables of the lock-exchange experiment. To quantify the errors of interest first the observed results are normalized to the varying parameter as prescribed by the conclusions of the qualitative analysis and similar methods used in literature [Adduce et al., 2012, Shin et al., 2004].

5 Results

5.1. Modelling

As a first exploration a few models were run with a single layer in the vertical direction. After some basic results were obtained the first results of experiments with 10 layers in the vertical direction were analyzed using Python based post-processing. Hereafter consistently two output parameters averaged over the width were compared over time: the water level and the salinity. During the modelling phase two sets of parameters were changed; Temperature and number of layers in the z-direction. The initial salinity difference was 10 ppt,the grid consisted of 100x3 cells and the simulation spanned 4 hours. The two performed variations are presented below along with general observations and conclusions regarding the reference model. Further settings and other results can be found in **??**.

5.1.1. Temperature variations

Salinity

As to the salinity some observations can be made.

- The frontal wave speeds at both fronts are almost linear over time. This indicates an energy conserving approximation may be valid.
- The frontal wave speed in the high density wave is lower than in the low density wave. This is probably because of the bed friction.
- At 6 degrees Celsius the model shows numerical diffusion characteristics for the low density wave. At higher temperatures it shows no direct numerical diffusion characteristics. However at 4 degrees Celsius on a different run it shows no such numerical diffusion characteristics. It is unclear why simulation 1205-3 shows this behaviour.
- At 4 degrees Celsius the frontal wave speeds are higher than at higher temperatures (10 and 15 degC). This can be explained through the formula that estimates the frontal wave speed based on the density difference and the water depth: $U = 0.5\cdot\sqrt{(g'\cdot d)}$. Where $g'=g(\rho2-\rho1)/\rho2$, given $\rho1 < \rho2$. [Pietrzak, 1998]. If the density of water decreases the factor $(\rho2-\rho1)/\rho2$ will decrease as well, thus increasing the effect of the salinity difference. Now, because water has it's highest density at 4 degrees Celsius, the effect of the density difference due to the salinity increases at higher or lower temperatures, meaning the reduced gravity will be smaller and thus the frontal wave speed will be smaller. This explains that wave frontal speed will be at its maximum at 4 degrees Celsius.
- At locations where internal waves may be formed the plot should be more detailed to draw conclusions. Also velocity profiles would be desirable.

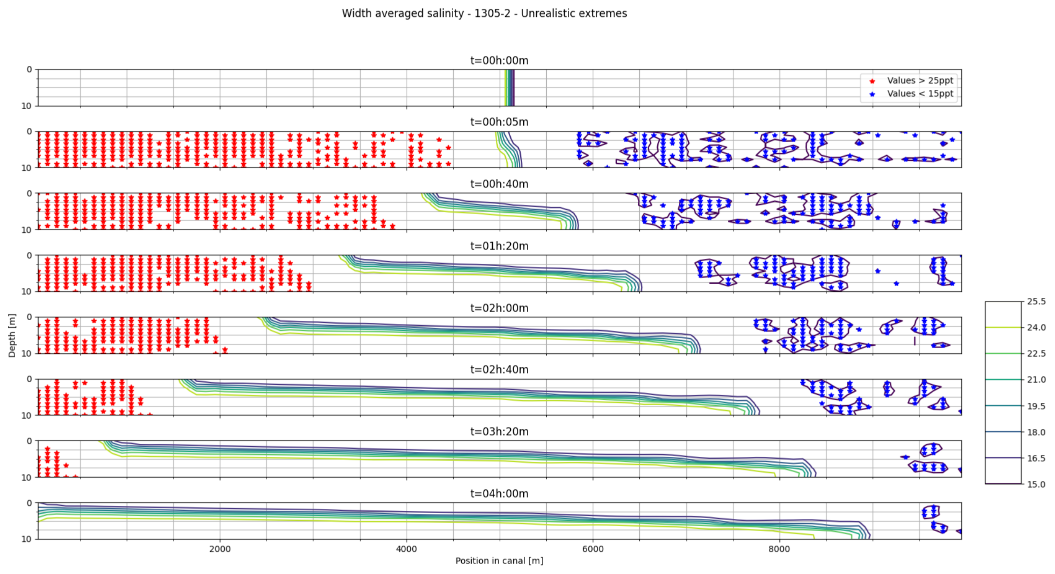


Figure 5.1: Contour plot - width averaged salinity for T=10 degrees Celsius

Water level

As to the water level the following can be said:

- When the temperature is higher, in the first time step, oscillations in the water profile can be seen in front of the disturbance, besides a large discontinuity around the center of the grid is observed. The fact that these oscillations are not observed at a temperature of 6 degC can be explained due to the relative effect the salinity difference has on the initial density difference, because this effect is smaller at lower temperatures due to a higher density of the water itself the disturbance is of a lower order than at higher temperature. These oscillations are could be a form of numerical dispersion and may be attributed to the hydrostatic assumption of the model, i.e. to compensate for the large vertical velocity gradient in the center of the grid at t=5min the model imposes a water level difference (the large discontinuity). Subsequently to deal with this water level difference in the middle of the grid, the approximation shows oscillations, i.e. in order to bridge the discontinuity in the water level and approximate the original water level at locations where the disturbance has not yet had any influence.
- Other oscillations are observed after the wave fronts (especially the high density wave) has passed. These oscillations could be a result of the artificial viscosity of the model, a process imposed to mimic energy loss due to heat transfer, this results in a transfer of kinetic energy in the form of waves. However, they could also be a result of the internal waves formed due to friction at the interface of the two density currents. Because in this mixing layer turbulence is likely to occur the model could show oscillations in the water level to compensate for the vertical velocity gradients associated with turbulence.

Because the temperature difference has an indirect effect on the forcing it is desirable to keep this at a constant level and try out a range of different salinity deltas. With this reasoning a default temperature of 4 degrees Celsius should be sustained. A final check in 1305-4 where a background temperature of 0 degrees Celsius, shows similar results as obtained from 1305-2 where a temperature of 10 degrees Celsius was set. Given that the density of water at 0 degrees (9998.7 kg/m^3) and 10 degrees (9997.5 kg/m^3) are similar, this substantiates this similarity. Finally the results obtained in a situation with 4 degree Celsius is presented.

5.1.2. Variations in number of layers in z-direction

Salinity

As to the salinity some observations can be made:

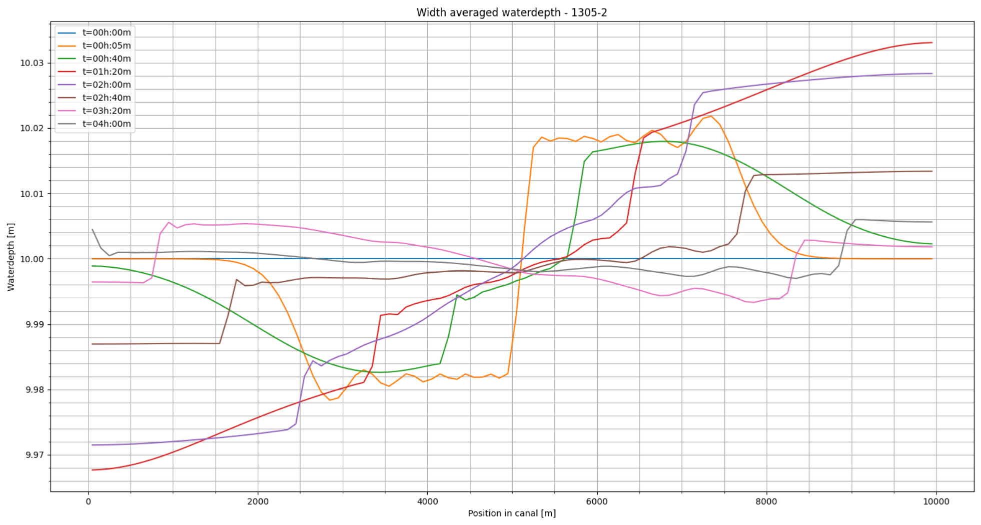


Figure 5.2: water level - Width averaged water level over time for T=10 degrees Celsius

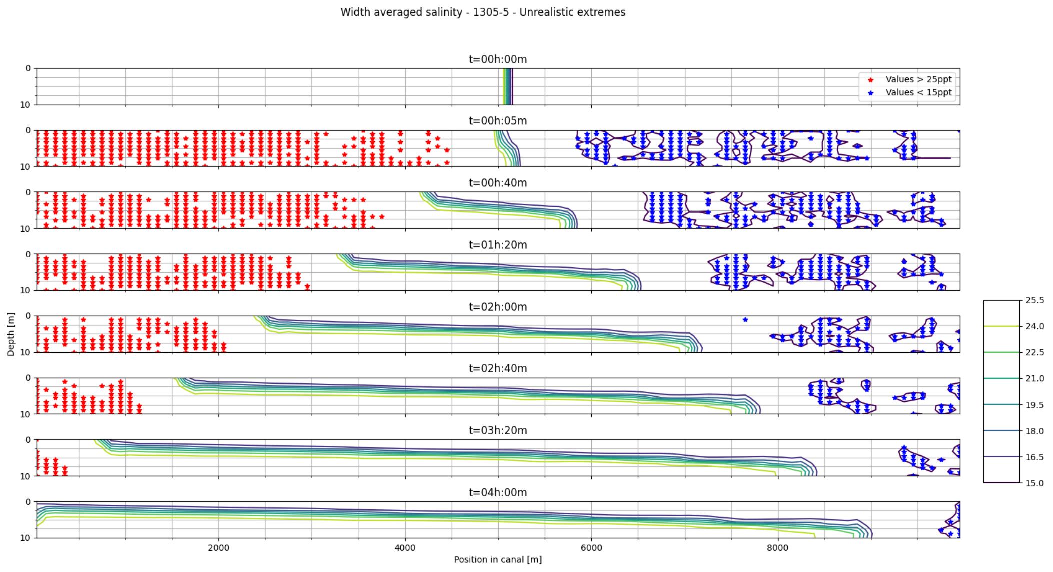


Figure 5.3: Contour plot - width averaged salinity for T=4 degrees Celsius

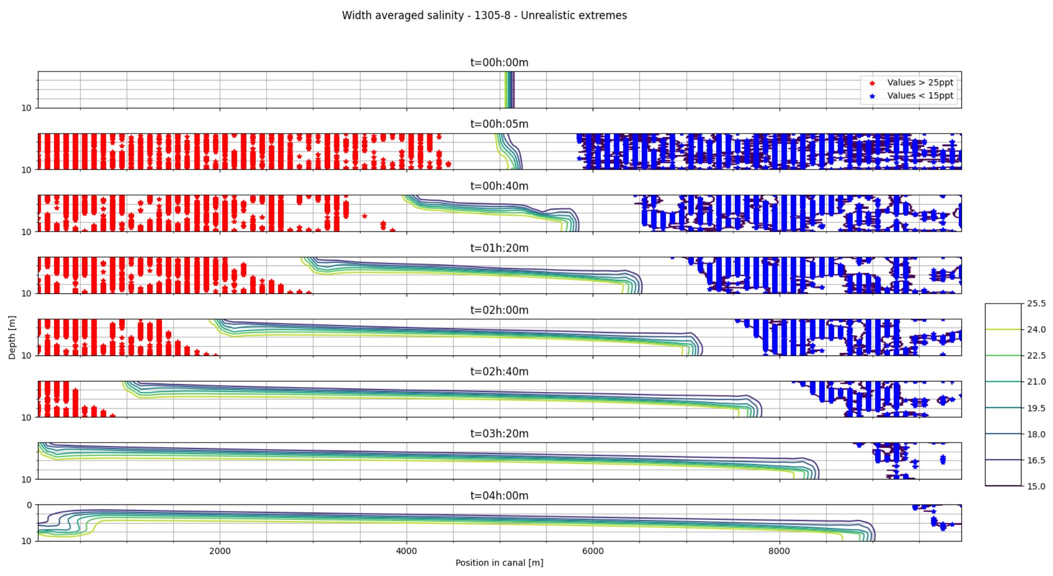


Figure 5.4: Contour plot - width averaged salinity for #z-layers=99

- The frontal wave speed of the lower density current seems to increase increasing layers. The frontal wave speed of the high density current remains almost the same. A frontal wave speed for the low density front of 0.38 m/s, 0.39 m/s and 0.42 respectively for simulations with 20, 50 and 99 depth layers. It seems as if it will approximate the estimated value of 0.49 m/s.

- Also fewer oscillations at the location where internal waves are expected are possibly observed.

Water level

As to the water level some observations can be made:

- The water level seems across all levels seems similar except for the oscillations at locations where the front has already passed, and ofcourse the difference in wave speed can be observed. In fact compared to the 10 layered model short oscillations seem to disappear with 20, 50 and 99 layers. However these shorter oscillations seem to be replaced by a single larger oscillation near the end of the simulation (between t=2h and t=4h), increasingly large as the number of layers increase. This could be attributed to the gradual compensation of the water level difference between the locations where the two wave fronts are situated, but it could also be a measure of the internal waves formed at the interface of the two density currents to compensate for the vertical velocity. This requires further investigation.

- Small oscillations around a constant water level difference across the first 3km of the grid, in each direction, at t=5min are similar across all simulations and indicate numerical dispersion to compensate for the discontinuity of the initial disturbance.

- The water level differences at the locations of each wave front are again expected to be caused by the vertical velocity gradients near the wave fronts for which the model compensates with a water level rise, to fulfill the hydrostatic condition.

- At a larger number of layers the absolute water level difference across the first 3km of the grid, in each direction, seems to persist longer in the first 40min of the simulation.

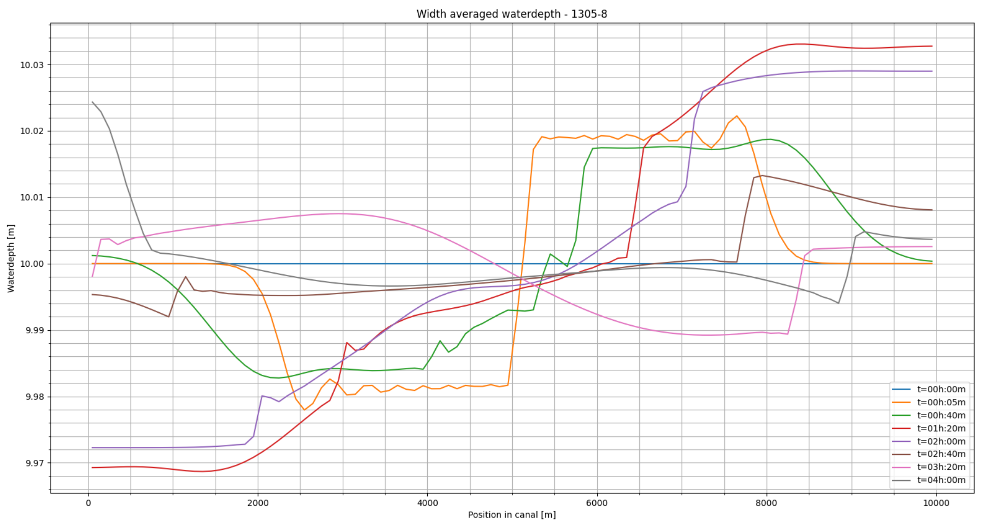


Figure 5.5: water level - Width averaged water level over time for #z-layers=99

5.1.3. General observations

Physical observations

The following physical process characteristic for the lock-exchange experiment are observed during both the temperature variations and the variations in number of z-layers.

- The frontal wave speeds at both fronts are almost linear over time. This indicates an energy conserving approximation may be valid.
- The frontal wave speed in the high density wave is lower than in the low density wave. This is probably because of the bed friction.
- For the temperature variations the results indicate a temperature of 4 degrees gives the highest water density thus the highest frontal wave speed. However simulating at a higher temperature could be more relevant, moreover the effect of the salinity difference is more noticeable, this is taken into account in the determination of the reference model.
- For the variation of the number of layers in the vertical direction the frontal wave speed increases a lot as the number of layers increase. It approaches a theoretical estimation as proposed by Pietrzak [1998].

Numerical observations

With respect to numerical phenomena the following is observed:

- Numerical dispersion is observed during all simulations at t=5min to deal with the initial disturbance in the water level. This initial disturbance is originates from the model's hydrostatic condition and the large vertical velocity gradients at the start of the simulation.
- Numerical diffusion is possibly observed near the interface of the two density currents. In this mixing layer where turbulence due to increased shear stresses is expected and clear oscillations are observed when the number of layers is 10. At a higher vertical resolution these short oscillations become a large oscillation with a single wave lenght stretching the grid. This transformation could be a result of the ratio between the horizontal and vertical resolution (1/1000 and 1/100 for 99 and 10 layers respectively) causing different modes in the approximation to be emphasized while trying to account for turbulence in the missing layer.

- The higher resolution, in simulations with more layers, cause more oscillations in the salinity contour lines to be visible near the front. These small oscillations are expected to be numerically diffused turbulence that may be expected at the progressing wave fronts.

5.1.4. Conclusions

The computation times were all within reasonable limits while the number of layers in the z-direction reached it's limit at 99. This is a positive result considering future variations in the spatial resolution. With respect to temperature also positive results are obtained that coincide with the expected physics, for this parameter a constant value of 4 degrees Celsius can be reasonably determined. However to obtain better insight in what causes the observed phenomena and how these relate to the numerical accuracy of the model more information is required. To this end other plots that are within reach and could be interesting are listed below:

- A table with the extreme values and there order of error
- Courant liming cells in the vertical
- Velocity magnitudes and vertical velocity
- 3D profiles instead of width averaged plots to get an idea of the numerical effects in the horizontal
- Detailed plots for the middle of the grid and at locations both fronts to get a detailed view on how the model deals with internal waves and large disturbances respectively.

5.2. Interval analysis

Here an interval analysis will be presented for parameters of the model that will be varied during the simulations. Based upon this a valid reasoning for the chosen parameter range is provided.

5.3. Simulations

Here the results per parameter variation will be presented.

5.3.1. Spatial variations

5.3.2. Temporal variations

5.3.3. Initial salinity

5.3.4. Numerical variations

5.4. Numerical evaluation criteria

Several numerical evaluation criteria, as explained in **??** and according to their relevance to a parameter, will be compared to the results.

5.5. Spectral analysis optional

If significant numerical dispersion is observed present a spectral analysis as proposed by Ruano et al. [2019].

5.6. Sensitivity analysis

Present sensitivity of model per parameter.

6 Discussion and conclusion

Final discussion of the results, their applicability and final conclusions are presented here.

Bibliography

C Adduce, G Sciortino, and S Proietti. Gravity currents produced by lock exchanges: experiments and simulations with a two-layer shallow-water model with entrainment. *Journal of Hydraulic Engineering*, 138(2):111–121, 2012.

Jurjen A. Battjes. *Unsteady Flow in Open Channels*. CAMBRIDGE UNIVERSITY PRESS, 2017. ISBN 1107150299. URL `https://www.ebook.de/de/product/27503143/jurjen_a_battjes_unsteady_flow_in_open_channels.html`.

Deltares. *Technical Reference Manual D-Flow FM*. Deltares systems, version 1.1.0 - svn revision 66571 edition, April 2020.

Mariangel Garcia, Paul F. Choboter, Ryan K. Walter, and Jose E. Castillo. Validation of the nonhydrostatic general curvilinear coastal ocean model (GCCOM) for stratified flows. *Journal of Computational Science*, 30:143–156, jan 2019. doi: 10.1016/j.jocs.2018.11.012.

George G. O'Brien, Morton A. Hyman, and Sidney Kaplan. A Study of the Numerical Solution of Partial Differential Equations. *Journal of Mathematics and Physics*, 29(1-4):223–251, 1950. ISSN 1467-9590. doi: 10.1002/sapm1950291223. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm1950291223`.

Julie Pietrzak. The use of TVD limiters for forward-in-time upstream-biased advection schemes in ocean modeling. *Monthly Weather Review*, 126(3):812–830, mar 1998. doi: 10.1175/1520-0493(1998)126<0812:tuotlf>2.0.co;2.

L. Prandtl. 7. bericht über untersuchungen zur ausgebildeten turbulenz. 5:136–139, 1925. ISSN 0044-2267. doi: 10.1002/zamm.19250050212.

J. Ruano, Aleix Báez Vidal, F.Xavier Trias, and Joaquim Rigola. A general method to compute numerical dispersion errors and its application to stretched meshes. 09 2019.

JO Shin, SB Dalziel, and PF Linden. Gravity currents produced by lock exchange. *Journal of Fluid Mechanics*, 521:1–34, 2004.

Cornelis Vuik, Fredericus Johannes Vermolen, Martin Bastiaan Gijzen, and MJ Vuik. *Numerical Methods for Ordinary differential equations*. VSSD, 2007.

M. Zijlema. Computational Modelling of Flow and Transport. *Collegedictaat CIE4340*, 2015. URL `https://repository.tudelft.nl/islandora/object/uuid%3A5e7dac47-0159-4af3-b1d8-32d37d2a8406`.

A

Turbulence modelling in D-Flow FM

A.1. Horizontal eddy viscosity

Modelling horizontal eddy viscosity has three seperate parameters that determine the total viscosity as follow: μ-H = μ-sgs + μ-v + μ-H-back.

These three parameters account for the following: 1. Horizontal turbulent viscosity may be underestimated because of the sub-grid scale turbulent motions, i.e. turbulence on a scale smaller than the meshgrid. This can be resolved by the sub-grid scale viscosity: μ-sgs 2. With Reynolds averaged shallow water equations horizontal eddy viscosity might not accounted for (enough) either thus D-Flow introduces the μ-v. 3. If extra constant or spatially dependant viscosity is desired the background viscosity μ-back may be added.

With respect to the 3D viscosity resulting from three-dimensional turbulence a closure model is used [Deltares, 2020, p.26]. For specific closure models one can even account for unresolved mixing through an ambient background mixing coefficient μ-V-back. Eventually the vertical eddy viscosity is thus calculated by a combination of the 3D viscosity μ-v and μ-mol, the latter being the kinematic viscosity of water, as follows: μ-v = μ-mol + max(μ-v, μ-v-back).

In D-Flow FM four turbulence closure models can be chosen, the first being user defined and the latter three based on models by Kolmogorov and Prandtl, all are explained in further detail in [Deltares, 2020, p.112-120];

1. Constant coefficient - resulting in a parabolic vertical velocity profile
2. Algebraic eddy viscosity closure model - based on the von Karman constant (κ), the bed friction (Cf), without including transport processes, computing mixing lenght (L), the shear velocity and the vertical turbulent viscosity μ-v.
3. K-ε turbulence model - involves solving a non-linear coupled system of equations describing turbulent kinetic energy (K) and energy loss (ε) including diffusivity coefficients (D), a turbulent kinetic energy production term (P), a Buoyancy flux (B) and a variation of kalibration terms (c1-3). Therafter the the vertical eddy viscosity μ-v is determined as proportianal to the ratio K^2/ε and the mixing lenght. Still, this coupled system has to be discretized in terms of advection and diffusion which is done explicitly by a first order upwind scheme and implicitly, respectively. Accordingly the production and buoyancy term are discretized while conserving the diagonally dominant matrix (ensuring positivity). Finally this leads to two tri-diagonal matrices for K and ε that can be solved using Thomas algorithm, which may be seen as the tri-diagonal LU-decomposition, by using specific boundary conditions.
4. K-τ turbulence - Where τ is a typical timescale of the turbulent eddies and the eddy viscosity is proportional to K·τ. Coupled by a system of convection diffusion equations including diffusivity, production and buoyance terms. The resulting advection equation is discretized with an first order upwind difference scheme and the vertical diffusion term is discretized implicitly by a temporal discretization scheme. Again this leads to two tri-diagonal matrices that can be solved by the Thomas algorithm using specific boundary conditions.

B Modelling of flow and transport

B.1. Molecular diffusion

As emphasized by [Battjes, 2017, p194]: “..molecular diffusion is irrelevant in civil engineering practice, where turbulent diffusion and dispersion are dominant..”.

In this case however it could be relevant because of the specific experiment. Still with the goal of modelling the Rhine-Maas delta in mind emphasis may be put on diffusion and dispersion processes related to turbulence. Moreover it is assumed constant accross the experiment thus numerical errors can not be specifically dedicated to the diffusion.

B.2. Turbulent diffusion

To describe the turbulent diffusion the fluctuating quantaties are averaged over a certain time and lenght scale. Because of gravity the concentration gradient is positive in the downward z-direction. From the averaging of the turbulent motion and the gravity induced concentration gradient it follows, as quoted from [Vuik et al., 2007, p.197], that: “..turbulent fluctuations cause a mean transport in the direction of decreasing values of the mean concentration, as in a diffusion process.”

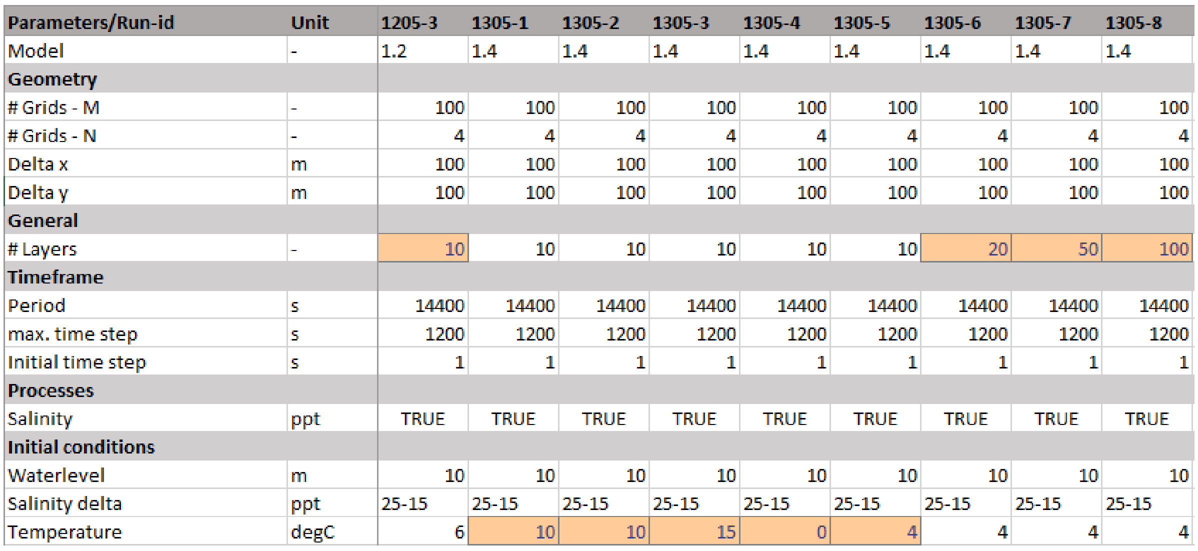
Because the fluctuating quantities cause such net transport processes, the resulting motion and turbulent properties need to be quantified. To this end Prandtl [1925] defined the concept of mixing lenght that defines the lenght over which the fluctiations cause a deviation from the average state and correspond to the average distance the turbulence eddies travel. For example, velocity fluctuations can be related to the velocity of the turbulent eddies. Further the turbulence induced transport processes are found to be proportional to the concentration gradient, also called the turbulence diffusivity. It has the order of magnitude of the eddie velocity times the mixing lenght.

For vertical diffusion in free surface flows the mixing lenght changes over the depth since it is induced by the bed friction expressed in the bed shear stress (τ-b). Using his an estimate of the particle velocity can be made by the so-called shear velocity which together with a parabolic variation of the mixing length over the depth gives a measure for the turbulence diffusivity: $\varepsilon t = K \cdot u \cdot L = K \cdot u \cdot z \cdot (1 - z/d)$. Where K is the Von Karman coefficient which was previously empirically determined.

In exactly this manner horizontal momentum can be vertically distributed through so-called Reynolds shear stress (τ-xz). Thus, in this case it is not a concentration (c) but a momentum per unit volume (ρu) that is diffused, an effect referred to as eddy viscosity. In a simple free surface flow [Vuik et al., 2007, p.199] shows how this leads to a logarithmic velocity profile.

C

Modelling results



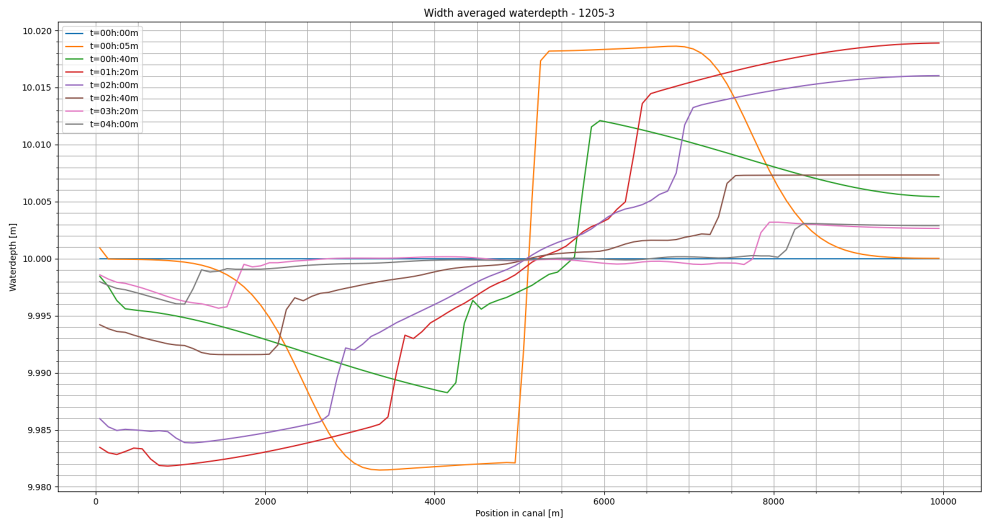
Parameters/Run-id	Unit	1205-3	1305-1	1305-2	1305-3	1305-4	1305-5	1305-6	1305-7	1305-8
Model	-	1.2	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
Geometry										
# Grids - M	-	100	100	100	100	100	100	100	100	100
# Grids - N	-	4	4	4	4	4	4	4	4	4
Delta x	m	100	100	100	100	100	100	100	100	100
Delta y	m	100	100	100	100	100	100	100	100	100
General										
# Layers	-	10	10	10	10	10	10	20	50	100
Timeframe										
Period	s	14400	14400	14400	14400	14400	14400	14400	14400	14400
max. time step	s	1200	1200	1200	1200	1200	1200	1200	1200	1200
Initial time step	s	1	1	1	1	1	1	1	1	1
Processes										
Salinity	ppt	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
Initial conditions										
Waterlevel	m	10	10	10	10	10	10	10	10	10
Salinity delta	ppt	25-15	25-15	25-15	25-15	25-15	25-15	25-15	25-15	25-15
Temperature	degC	6	10	10	15	0	4	4	4	4

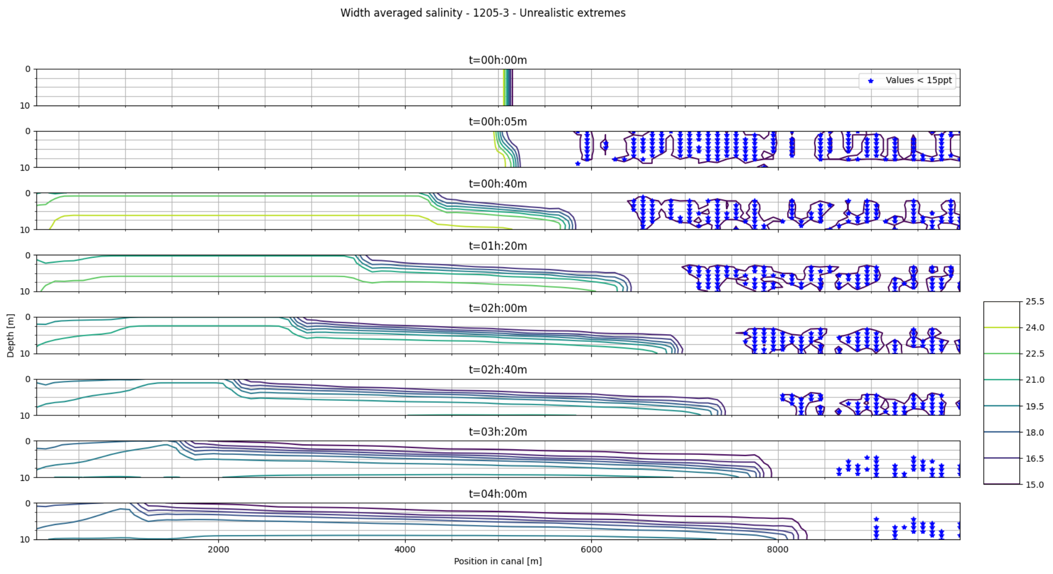
Figure C.1: Settings used for temperature and z-layer variation

C.1. Settings

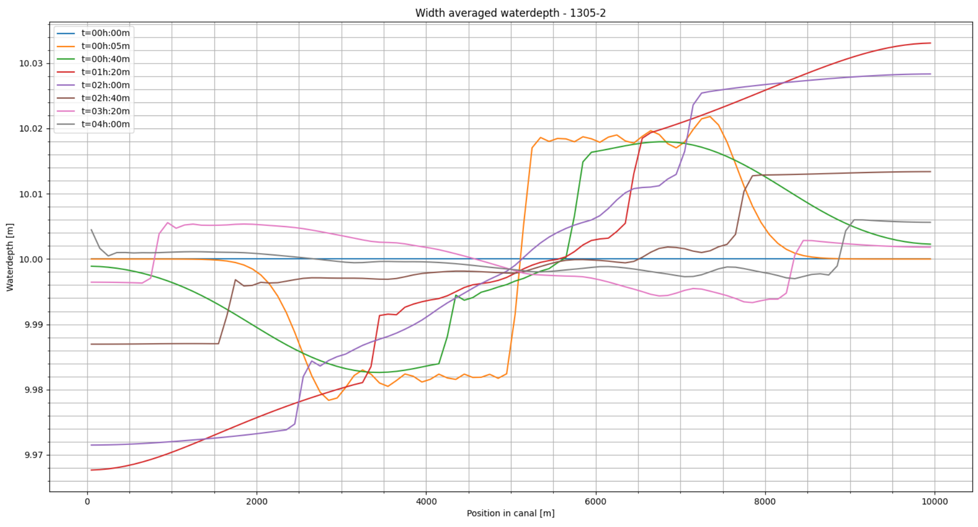
C.2. Temperature variations

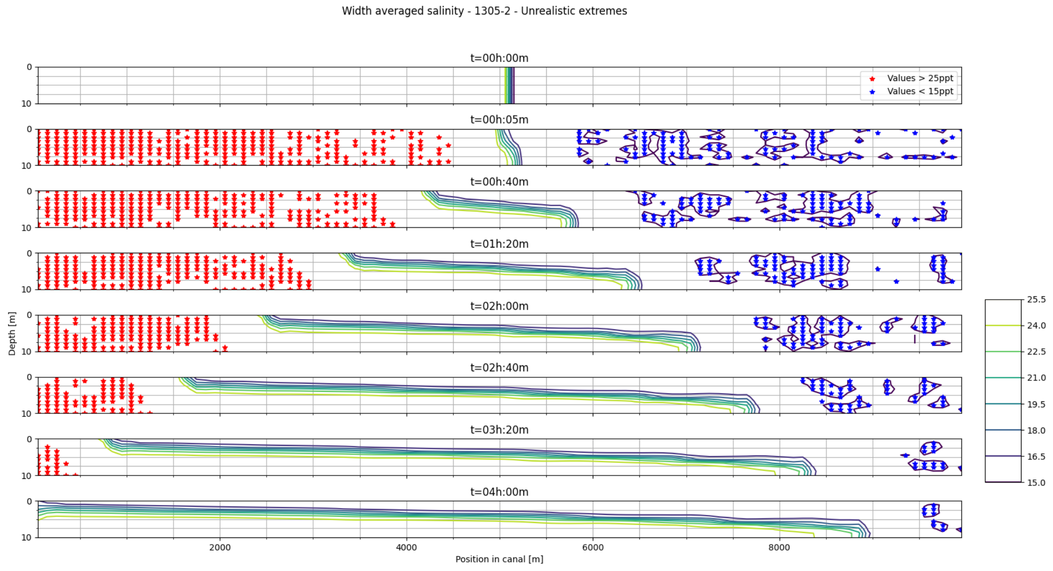
C.2.1. Simulation 1205-3



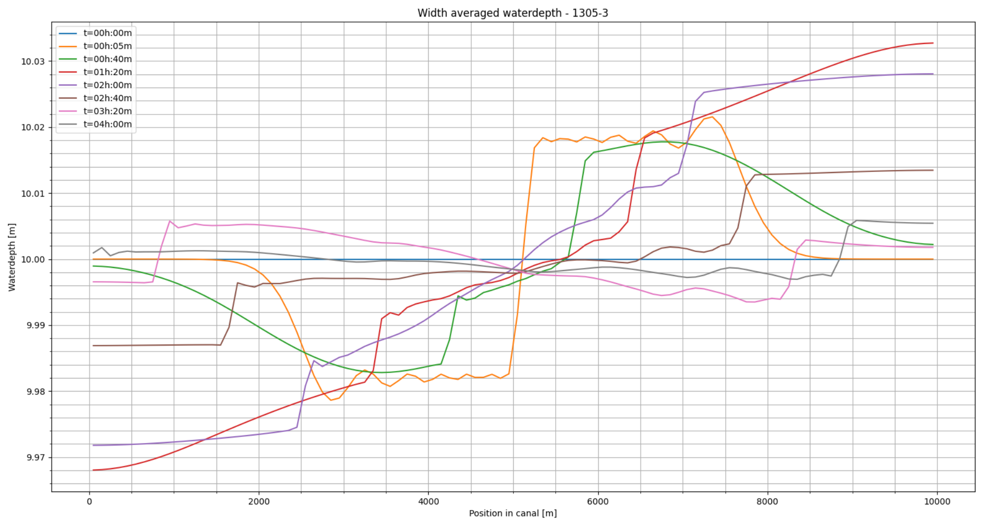


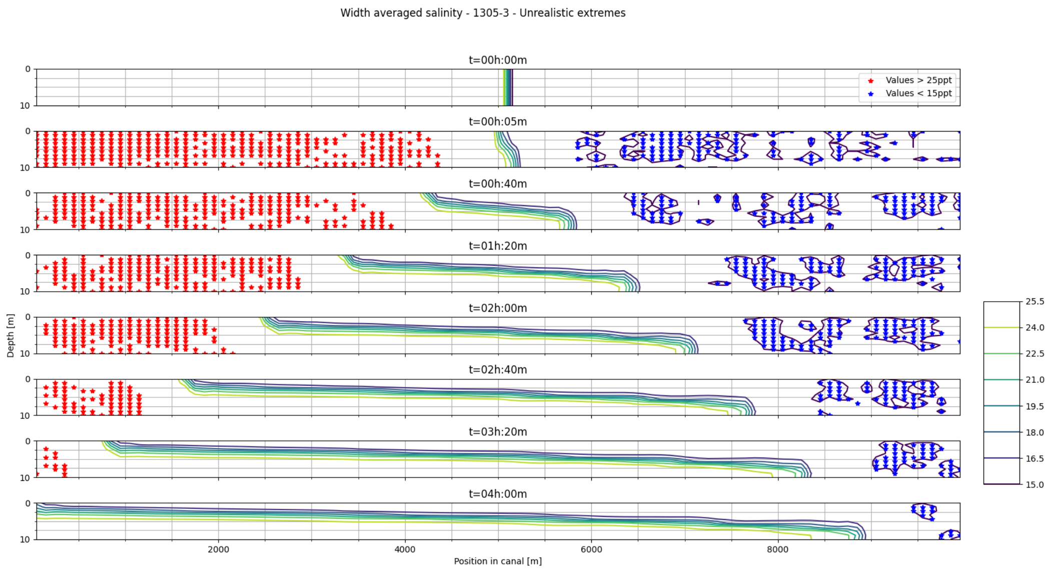
C.2.2. Simulation 1305-2



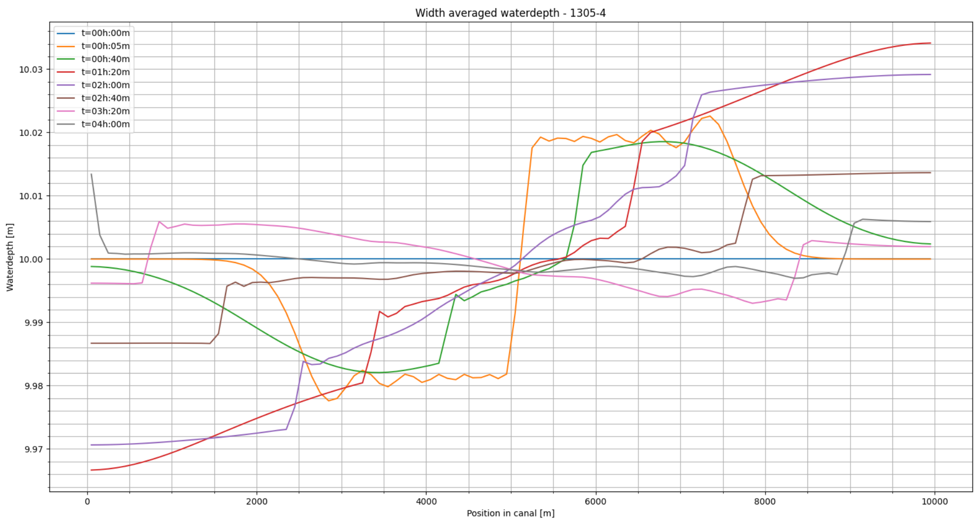


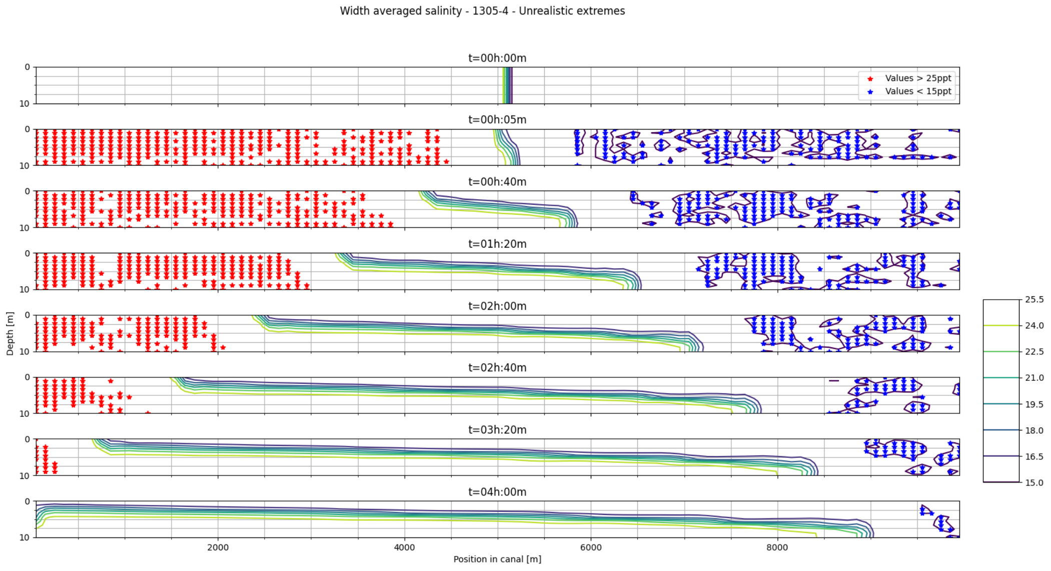
C.2.3. Simulation 1305-3



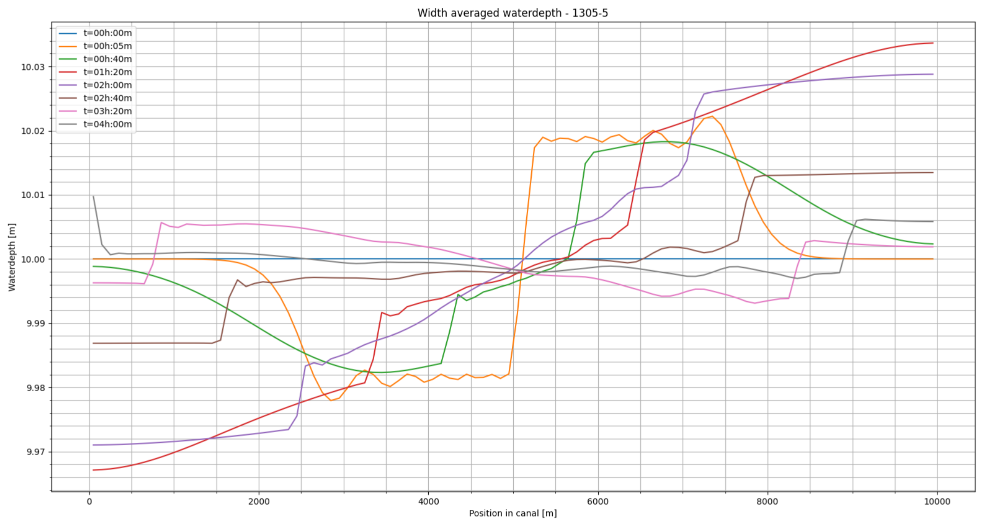


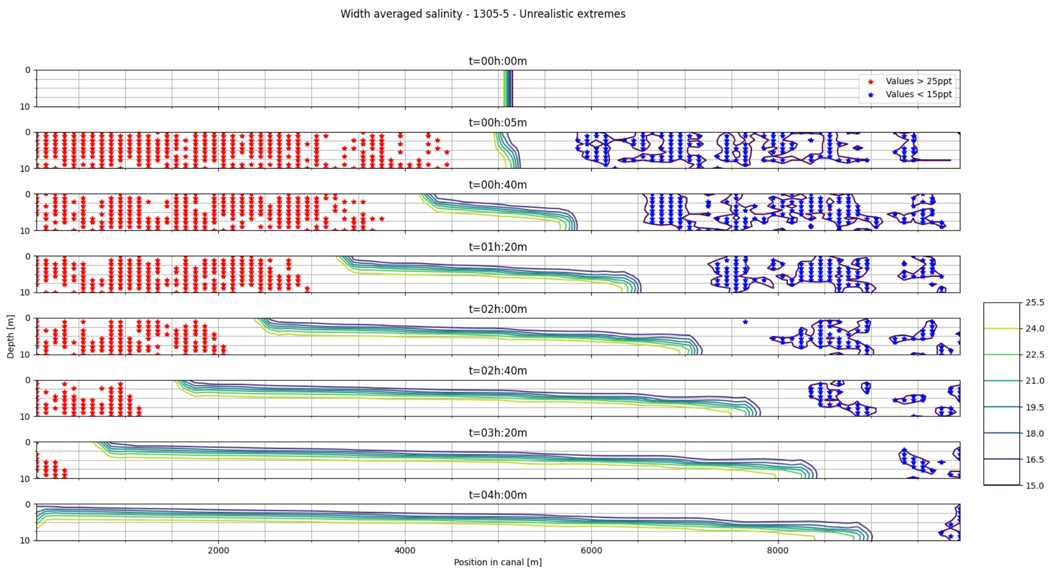
C.2.4. Simulation 1305-4





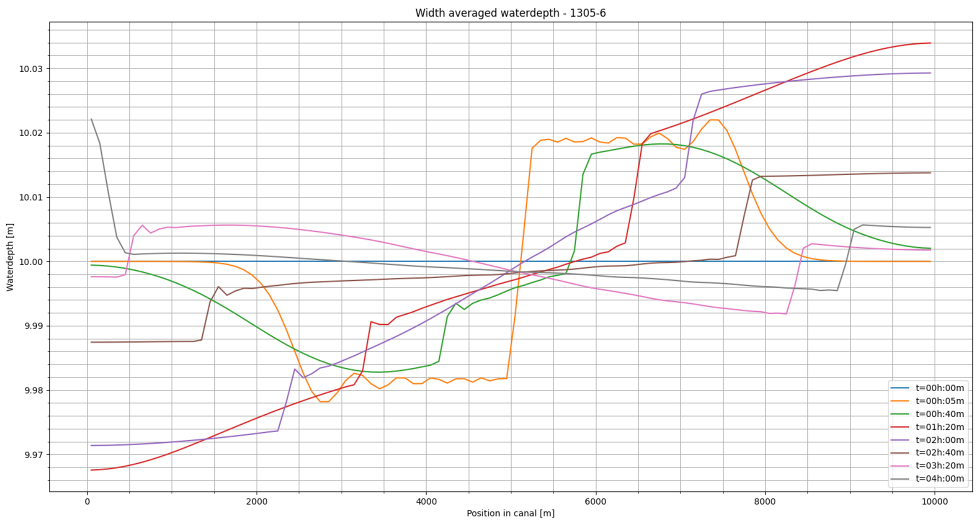
C.2.5. Simulation 1305-5

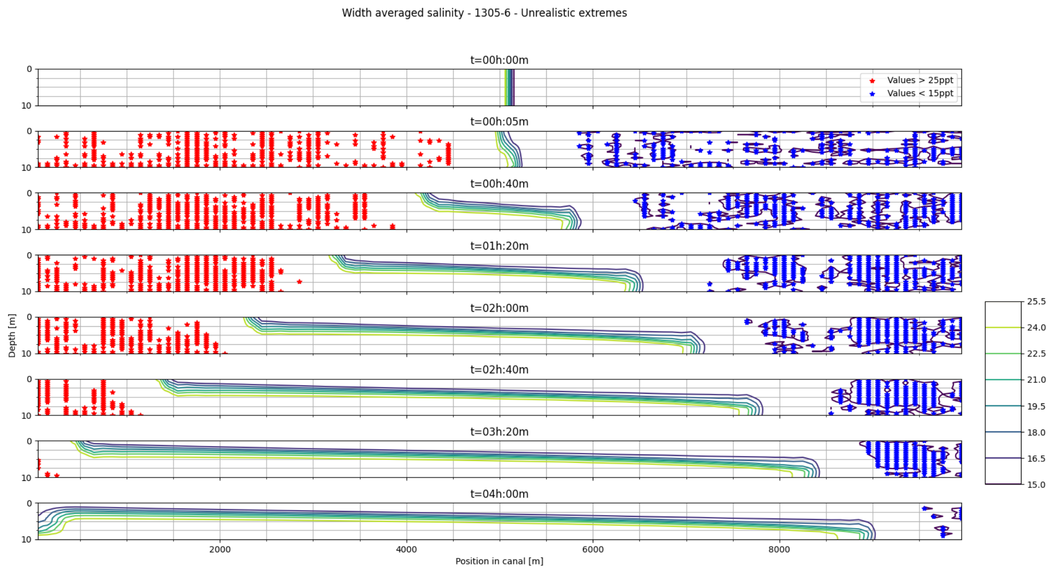




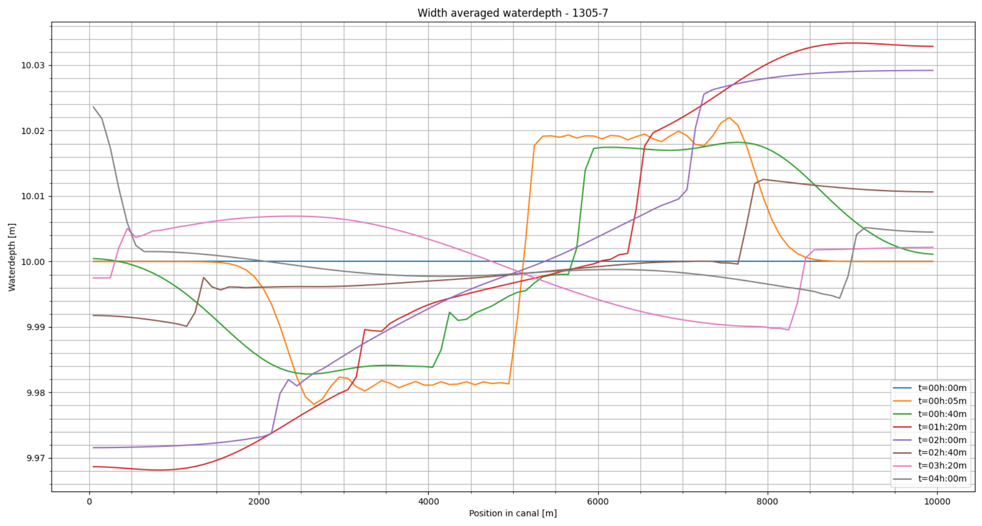
C.3. Number of layers in vertical direction variations

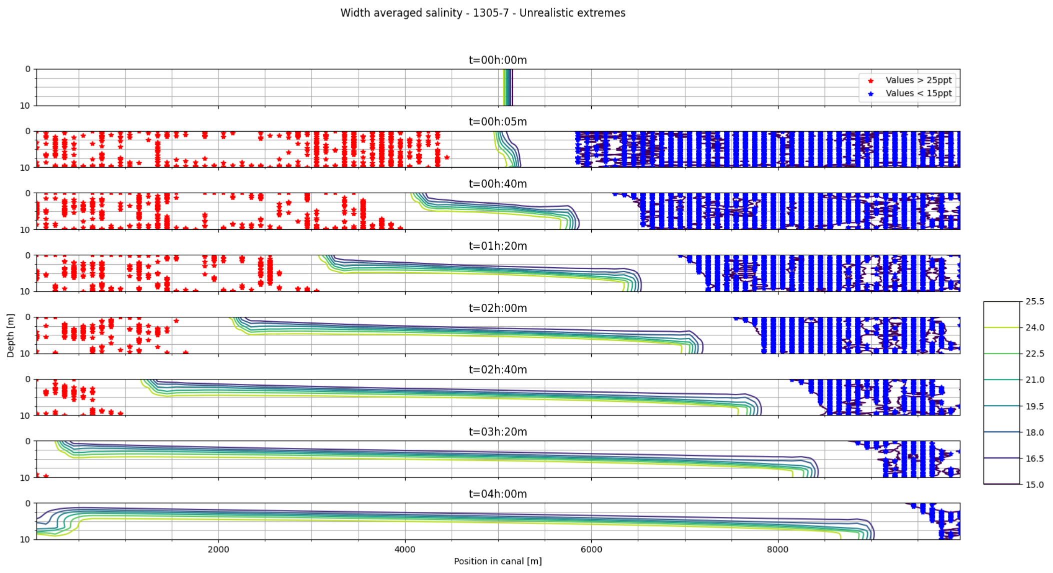
C.3.1. Simulation 1305-6



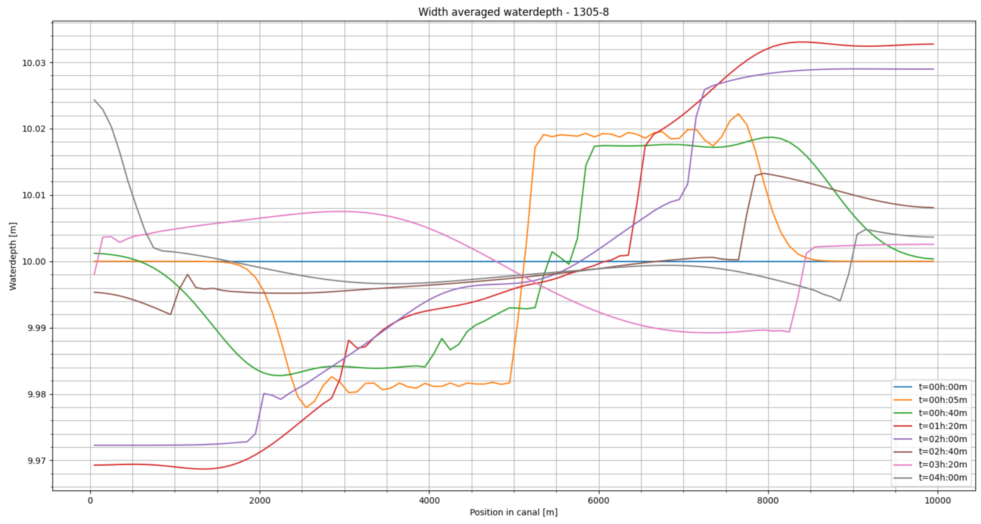


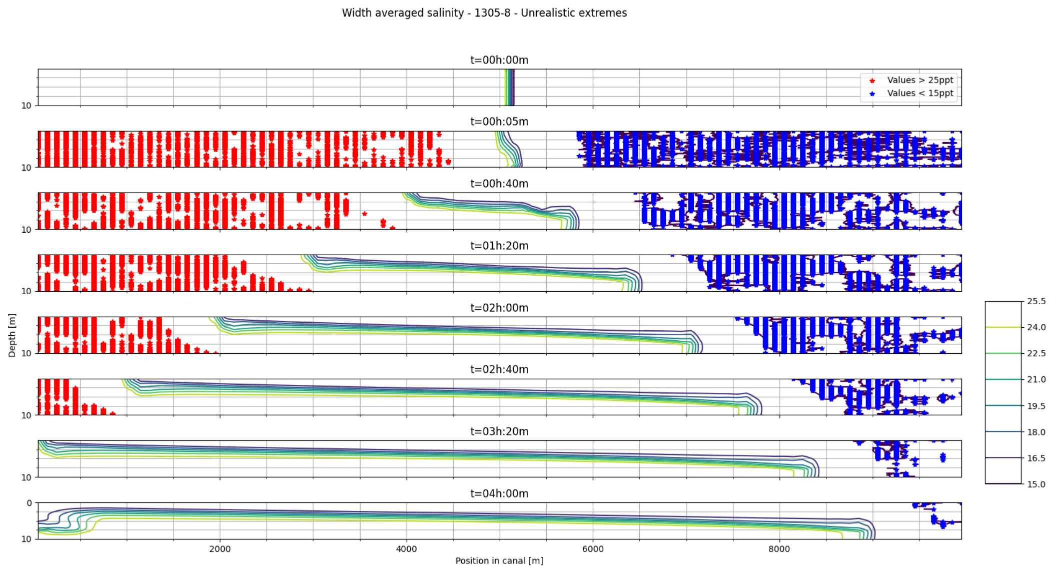
C.3.2. Simulation 1305-7





C.3.3. Simulation 1305-8





D
Python post-processing code

```
import xarray as xr import numpy as np import pandas as pd import
matplotlib.pyplot as plt from mpl\_toolkits.mplot3d import Axes3D import
seaborn as sns

class simulations(object): \#\# Simulation run 1205-3 sim1205\_3 = dict(
netcdf\_file =
'./data/initial-model-1.2-1205-3/initial-model-1.2\_map.nc', run\_id =
'1205-3', sal\_min = 15, sal\_max = 25, depth = 10, layers=20 )

## Simulation run 1305-1
## First run :%s/mesh2d_nLayers/mesh2d_nLayerss/g
sim1305_1 = dict(
    netcdf_file = './data/initial-model-1.4-1305-1/InitialModel_map.nc',
    run_id = '1305-1',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=10
)

## Simulation run 1305-2
sim1305_2 = dict(
    netcdf_file = './data/initial-model-1.4-1305-2/InitialModel_map.nc',
    run_id = '1305-2',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=10
)

## Simulation run 1305-3
sim1305_3 = dict(
    netcdf_file = './data/initial-model-1.4-1305-3/InitialModel_map.nc',
    run_id = '1305-3',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=10
)
```

```
## Simulation run 1305-4
sim1305_4 = dict(
    netcdf_file = './data/initial-model-1.4-1305-4/InitialModel_map.nc',
    run_id = '1305-4',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=10
)

## Simulation run 1305-5
sim1305_5 = dict(
    netcdf_file = './data/initial-model-1.4-1305-5/InitialModel_map.nc',
    run_id = '1305-5',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=10
)

## Simulation run 1305-6
sim1305_6 = dict(
    netcdf_file = './data/initial-model-1.4-1305-6/InitialModel_map.nc',
    run_id = '1305-6',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=20
)

## Simulation run 1305-7
sim1305_7 = dict(
    netcdf_file = './data/initial-model-1.4-1305-7/InitialModel_map.nc',
    run_id = '1305-7',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=50
)


## Simulation run 1305-8
sim1305_8 = dict(
    netcdf_file = './data/initial-model-1.4-1305-8/InitialModel_map.nc',
    run_id = '1305-8',
    sal_min = 15,
    sal_max = 25,
    depth = 10,
    layers=100
)
```

class dataset(): def **init**(self, netcdf_file, run_id, sal_min, sal_max, depth,layers): self.dataset_location = netcdf_file self.dataset = xr.open_dataset(self.dataset_location) self.df_waterdepth = self.dataset.mesh2d_waterde self.df_salinity = self.dataset.mesh2d_sa1.to_dataframe() self.run_id = run_id self.sal_min = sal_min self.sal_max = sal_max self.depth = depth self.df_courant = self.dataset.mesh2d_Numlimdt.to_dataframe() self.layers=layers

```
def theoretical_frontal_wavespeed(self):
    p1 = 1000+self.sal_min
    p2 = 1000+self.sal_max
    Y = min(p1/p2, p2/p1)
    g = 9.81
    gr = g*(1-Y)
    return 0.5*np.sqrt(gr*self.depth)

def plot_waterdepth_time(self, n=1, save=False, fig_name='width_averaged_waterdepth'):
    '''
   Plots n number of equal time differenced plots of the waterdepth averaged over the width. Th
    '''
    time = self.df_waterdepth.index.unique(level='time')

    # plot initial situation
   t_delta = self.df_waterdepth.loc[time[0]].groupby('mesh2d_face_x').mean()
    name = time[0].strftime('t=%Hh:%Mm')
    t_delta.mesh2d_waterdepth.plot(label=name)

    for t_idx in range(0, len(time), int(len(time)/n)):
        if t_idx == 0:
            t_idx=1
        name = time[t_idx].strftime('t=%Hh:%Mm')
     t_delta = self.df_waterdepth.loc[time[t_idx]].groupby('mesh2d_face_x').mean()
        t_delta.mesh2d_waterdepth.plot(label=name)


    plt.title(f'Width averaged waterdepth - {self.run_id}')
    plt.xlabel('Position in canal [m]')
    plt.ylabel('Waterdepth [m]')
    plt.legend()
    plt.minorticks_on()
    plt.grid(True, which='both')

def plot_salinity_time(self, n=1, save=False, fig_name='width_averaged_salinity', heatmap=Tr
    # Get data
    time = self.df_salinity.index.unique(level='time')
    df = self.df_salinity.reset_index(level='mesh2d_nLayers')

    # Initiate subplots
    fig, axes = plt.subplots(n+2, 1, sharex=True, sharey=True)
    plt.subplots_adjust(hspace=0.7)

    # plot initial situation
   avg_sal = df.loc[time[0]].groupby(['mesh2d_face_x', 'mesh2d_nLayers']).mean()['mesh2d_sa
    avg_sal = avg_sal.reset_index()
   piv = avg_sal.pivot_table(values='mesh2d_sa1', index='mesh2d_nLayers', columns='mesh2d_fa

    if heatmap:
     hm = sns.heatmap(piv, ax=axes[0], cmap='plasma', cbar_kws={'label':'Salinity [ppt]'}, c
        hm.invert_yaxis()
        plottype = 'Heatmap'

    if contour:
        x,y=np.meshgrid(piv.columns,piv.index)
        axes[0].grid()
        cf = axes[0].contour(x,y, piv.values, 6, vmin=15, vmax=25)
        cp = axes[0].contour(x,y, piv.values, 6, vmin=15, vmax=25)
        #axes[0].clabel(cp, cp.levels, colors='black', fontsize=6)
        fig.colorbar(cf, cax=fig.add_axes([.92,.2,.03,.3]))
        plottype = 'Contourlines'

    if extremes:
        rdots=None
        bdots=None
        df_lows = avg_sal[avg_sal.mesh2d_sa1 < 15].reset_index()
        if not df_lows.empty:
        bdots = axes[0].plot(df_lows.mesh2d_face_x, df_lows.mesh2d_nLayers, 'b*')
        df_highs = avg_sal[avg_sal.mesh2d_sa1 > 25].reset_index()
        if not df_highs.empty:
        rdots = axes[0].plot(df_highs.mesh2d_face_x, df_highs.mesh2d_nLayers, 'r*')

        plottype = 'Unrealistic extremes'

    name = time[0].strftime('t=%Hh:%Mm')
    axes[0].set_title(name)
    axes[0].get_yaxis().get_label().set_visible(False)
    axes[0].get_xaxis().get_label().set_visible(False)
    axes[0].set_yticks([0,self.layers-1])
    axes[0].set_yticklabels([str(self.depth), '0'])
    axes[0].minorticks_on()
    axes[0].grid(True, which='both')

    i = 0
    for t_idx in range(0, len(time), int(len(time)/n)):
        i+=1
        if t_idx == 0:
            t_idx=1

     avg_sal = df.loc[time[t_idx]].groupby(['mesh2d_face_x', 'mesh2d_nLayers']).mean(
        avg_sal = avg_sal.reset_index()
     piv = avg_sal.pivot_table(values='mesh2d_sa1', index='mesh2d_nLayers', columns='m

        if heatmap:
            hm = sns.heatmap(piv, cmap='plasma', ax=axes[i], cbar=False)
            hm.invert_yaxis()

        if contour:
            x,y=np.meshgrid(piv.columns,piv.index)
           cp = axes[i].contour(x,y, piv.values, levels=6, vmin=15, vmax=25)
            #axes[i].clabel(cp, cp.levels, colors='black', fontsize=6)

        if extremes:
            df_lows = avg_sal[avg_sal.mesh2d_sa1 < 15].reset_index()
            if not df_lows.empty:
           bdots = axes[i].plot(df_lows.mesh2d_face_x, df_lows.mesh2d_nLayers, 'b*')

            df_highs = avg_sal[avg_sal.mesh2d_sa1 > 25].reset_index()
            if not df_highs.empty:
           rdots = axes[i].plot(df_highs.mesh2d_face_x, df_highs.mesh2d_nLayers, 'r*')

        name = time[t_idx].strftime('t=%Hh:%Mm')
        axes[i].set_title(name)
        axes[i].set_yticks([0,self.layers-1])
        axes[i].set_yticklabels([str(self.depth), '0'])
        axes[i].minorticks_on()
        axes[i].grid(True, which='both')

        if i != n+1:
            axes[i].get_xaxis().get_label().set_visible(False)
        if i != round(len(axes)/2):
            axes[i].get_yaxis().get_label().set_visible(False)


    # Plot legend if applicable
    if extremes:
        handles=[]
        labels=[]
        if rdots:
            handles.append(rdots[0])
            labels.append(f'Values > {self.sal_max}ppt')
        if bdots:
            handles.append(bdots[0])
            labels.append(f'Values < {self.sal_min}ppt')

        axes[0].legend(handles=handles, labels=labels, loc='best')

    plt.suptitle(f'Width averaged salinity - {self.run_id} - {plottype}')
    axes[-1].set_xlabel('Position in canal [m]')
    axes[round(len(axes)/2)].set_ylabel('Depth [m]')
    plt.tight_layout(rect=[.1,.1,.8,.8])
```