Salt Intrusion Modelling via the Particle-tracking Framework

by

Sicong Lin

to obtain the degree of Master of Science,
in Applied Mathematics - Computer Simulations for Science and Engineering (COSSE),
at Delft University of Technology and Technical University of Berlin,
to be defended on Jan. 19th, 2024.

Student number:	5472725
Course code:	WI5000COSSE
Project Duration:	2023 Mar. - 2024 Jan.
Supervisors:	Dr.Ir. M. Verlaan, Delft University of Technology
	Dr.Ir. K. Vuik, Delft University of Technology
Thesis Committee:	Dr.Ir. M. Verlaan, Dr.Ir. K. Vuik,

Preface

The topic of this thesis project is to use the particle-tracking framework to study salt intrusion. This project was proposed by Deltares and is a small part of the SALTISolutions project, that attempts to build a digital twin model of the Rhine-Meuse Delta. From a mathematical perspective, the particle-tracking framework is inspired by the theoretical connection between partial differential equations that describes substance transport and stochastic differential equations (SDEs). The idea is to use replace the continuous substance distribution in the fluid by many particles. After tracking the particles, the consequent substance distribution can be approximated by the particles' distribution at certain time instant. There are two major achievement of this project. Firstly, different implementation of non-penetrating and flux boundary conditions for the particle-tracking framework are tested and compared. The recursion algorithm is preferred for the previous and the “Neumann-wise” is recommended for the latter. Secondly, the convergence order of SDE numerical schemes and their influences on the simulation of the vertical transport physics are examined by studying the one-dimensional pure diffusion test case. We see that most simple Euler scheme contaminates the stratification completely in a theoretical setting even using a small timestep, indicating that it might be worthwhile to switch to high-order schemes whenever possible. However, even if one can implement other higher order numerical scheme correctly and efficiently, which is not easy, it might be equivalently important to have enough particles and good interpolation of velocity and diffusion field in the computation domain, because they may introduces error comparable to the diffusion error. Since salt transport is already resolved by the Delft3D-FM model, it is of nonsense to use the particle-tracking framework to study transport. The one-dimensional pure diffusion test case is mainly used for verification of the method. Actually, the particle-tracking framework should be used in combination of the diagnostics timescales and tracer methods. This often requires calculation of the statistics of the particle ensemble. It will be interesting to conduct such analysis on a realistic three-dimensional test case, and this could be the field of future research.

This thesis work takes more or less nine months. It is hard to estimate because there were pauses. It is the first time that I have an opportunity to immerse myself in an independent research-oriented project. This is not easy to me, but thanks to support from people I know and my own persistence, I finally made it. I want to thank my supervisor Martin Verlaan for organizing this project and providing all the necessary support for this project to me. I want to thank Kees Vuik for helping me manage this project generally. I want to thank my parents for their consistent support mentally and financially. I want to thanks Avelon Gerritsma, Marlein Geraeds and Amey Vasulkar at Deltares for their innovative ideas on the particle-tracking framework. I also want to thank my teammates at DCF, such as Zelin Xu, Qingxin Liu, Yiting Li, Jiaxuan Zhang, Xin Tan, Xiaoyang Dong, Jinqiang Chen, Zhaoyu Yan, Qizhang Dong, and Junda Wu etc., for those unforgettable time on the soccer field and at the table. I would like to thank my COSSE colleagues at TU Berlin and AM colleagues at TU Delft for their support during my master studies. I want to express my sincerest gratitude to people who have accompanied in this long journey. Thanks you!

Sicong Lin
Delft, January 2024

Contents

Preface **i**

1 Introduction **1**
1.1 Research Background . . . 1
1.2 Organization of the report . . . 2

2 Modelling of salt intrusion **4**
2.1 Introduction . . . 4
2.2 Basic physics of salt intrusion . . . 4
2.3 Salt intrusion modelling by one-dimensional models . . . 5
2.4 Salt intrusion modelling using two- or three-dimensional models . . . 6
2.5 Salt intrusion modelling and climate change . . . 7
2.6 Salt intrusion modelling in the aquifer: a digression . . . 8

3 A brief description of the particle-tracking method **9**
3.1 Introduction . . . 9
3.2 Link the advection-diffusion equation to an SDE . . . 10
3.3 SDE numerical schemes . . . 11
3.4 Modelling the vertical transport physics . . . 13
3.5 Introduction of density estimation . . . 14
3.6 Introduction of test cases . . . 16
3.7 A digression into the smooth particle hydrodynamics . . . 17

4 Simple test cases **18**
4.1 sinusoidal vortex . . . 18
4.1.1 Problem description . . . 18
4.1.2 Results and discussion . . . 19
4.2 Channel with a bump . . . 19
4.2.1 Problem Description . . . 19
4.2.2 Add the diffusion effect . . . 21
4.2.3 How to compute the relative z-coordinate . . . 22
4.2.4 Results and discussion of the recursive particle update algorithm . . . 22
4.2.5 Implementation issues . . . 22

5 One-dimensional test cases **25**
5.1 Introduction . . . 25
5.2 Introduction and implementation of two simple test cases . . . 25
5.2.1 Pure diffusion problem . . . 25
5.2.2 Advection-diffusion problem . . . 27
5.3 Intro to the stratification case . . . 29

6 Simulation of random processes **32**
6.1 the GBM test case . . . 32
6.1.1 Summary of numerical schemes . . . 34
6.1.2 Modelling setting . . . 35
6.1.3 Results and discussion for the GBM case . . . 35
6.2 1D pure diffusion . . . 37
6.2.1 Problem description . . . 37
6.2.2 Equivalence of numerical schemes . . . 38
6.2.3 Implementation of the ”reflection method” in density estimation . . . 38
6.2.4 Error measure construction . . . 39
6.2.5 Parameter setting . . . 40

6.2.6 Results and Discussion 40
6.3 A extended discussion on the density estimation method 43

7 Implementation of the 2D tidal flume process 48
7.1 Problem description 48
7.2 Mathematical modelling of the problem 48
7.3 Discussion and analysis 50
7.3.1 Simulation parameter setting 50
7.3.2 Erroneous deterministic drift due to diffusion 50
7.3.3 Different implementation of boundary conditions 51
7.3.4 Sensitivity analysis 51
7.3.5 Uses of high-order scheme 51
7.4 Summary 52

8 Conclusion 60
8.1 Summary 60
8.2 Findings, limitations and future perspectives 60

References 62

A Appendix: Basics of stochastic process, SDEs and SDE numerical schemes 65
A.1 Basics of stochastic process and stochastic differential equations 65
A.1.1 Stochastic process basics 65
A.1.2 Introduction of stochastic differential equations 66
A.1.3 Stochastic integrals 67
A.1.4 Two equivalent definitions of SDEs 67
A.1.5 Stochastic differentiation 68

1 Introduction

1.1. Research Background

Salt intrusion, roughly speaking, is the process of denser seawater creeping along the seabed toward the land from the sea. This often happens in estuaries, which are the “mouths” that connects the river to the ocean. In the estuary, seawater mixes with freshwater from the river. When the mixing is strong, a gradual change from seawater to freshwater can be observed. When mixing is weak, a seawater wedge is formed because seawater is denser than freshwater and thus seawater can flow upstream along the river bed below the freshwater. Besides, seawater can penetrate through the river bed and the river bank and go into the aquifer, an underground layer that is porous and bears water. In short, salt intrusion can happen in surface flow and underground flow. Salt intrusion poses threats to the

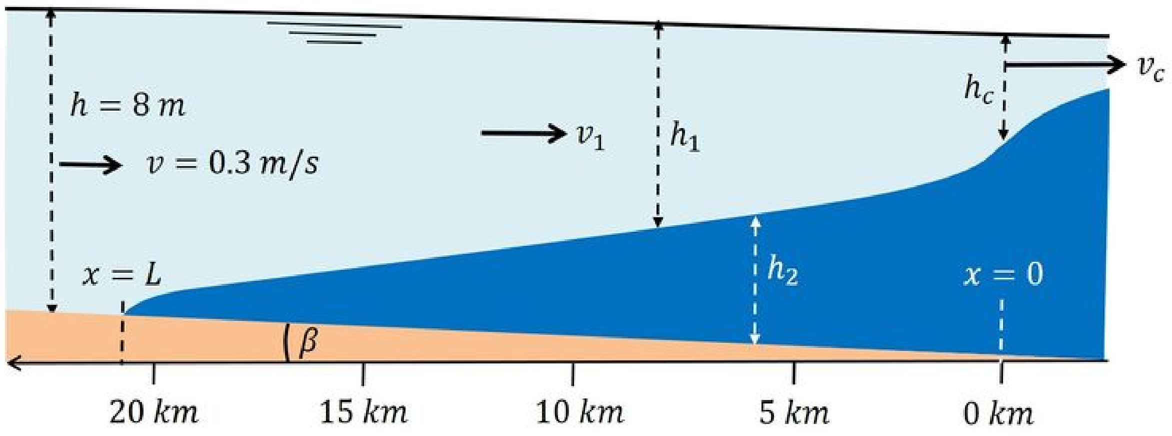


Figure 1.1: Longitudinal section of a salt wedge in a prismatic estuary retrieved from [37].

estuary and the freshwater ecosystem. Due to the effect of estuarine circulation, some organisms are used to living in the mixing region of seawater and freshwater with appropriate salinity and abundant nutrients from the upstream. The animals, plants and microbes there are often sensitive to the salinity of the water, thus the inflow of seawater changes their living environment and forces the estuary ecosystem to move further upstream. And without doubt the freshwater ecosystem further upstream will also be affected. Besides, salt intrusion also has a negative impact to people living upstream in the delta region. For example, the effect of salt intrusion may be so strong that the front of salt water travels far upstream to the freshwater intake point and contaminates the water there. This will affect water supply for drinking and agricultural purposes.

Given the ecological and societal impacts of salt intrusion, much research is devoted to this topic. The ongoing SALTISolutions project, funded by the Dutch Research Council (NWO), was set up with the purpose of setting up a digital-twin model for the Rhine-Meuse Delta, along with a toolbox for application purposes. The undergoing project is envisioned to monitor salt intrusion, evaluate its impacts, and

suggest possible measures to counter its negative impacts.
To contribute to this project, people at Deltares are implementing a particle-tracking method that makes use of the simulation results from the output of the Delft3D FM Suite [23], a hydrodynamic modeling package developed by Deltares. In the current modelling framework, the particle-tracking method is suited for studying salt intrusion mainly because it makes use of the (numerically) resolved flow field data to conduct pathway, connectivity and water age analysis in the Rhine-Meuse Delta. The current thesis project builds upon the current progress, where there is a two-dimensional particle-tracking module with interfaces written in Julia, a modern high-performance language. The major objective of this thesis project is to explore how to extend the currently 2D particle-tracking framework that can only track particles on the sea surface to a 3D one that includes vertical physics and to use the framework to study salt intrusion. Naturally, we can ask the major research question — " what are the major difficulties in using the particle-tracking framework to study salt intrusion"? Relevant research questions are proposed as well.

- How to implement initial and boundary conditions in the particle-tracking framework?
- How to determine the necessary number of particles and grid size?
- How will using high-order SDE numerical schemes affect the final results?
- Does the problem scale up well, i.e. a large ensemble of particles is used?
- What are the implications of the particle-tracking results?

This thesis report will attempt to address the research questions mentioned here.

1.2. Organization of the report

The organization of the paper is as follows. The second chapter discusses salt intrusion modelling. We briefly discussed the differences and uses of different 1D, 2D and 3D models, and corresponding examples are given. Salt intrusion into an aquifer is also mentioned for the sake of completeness, though it is not the scope of this thesis work. The third chapter first describes the major motivation about why people switch their attention from grid-based methods to the particle-tracking framework. It also discusses the mathematics of the particle-tracking framework. As for the minimally required theoretical background, the link between the transport equation and the stochastic differential equation (SDE) is explained and corresponding SDE numerical schemes to solve the equation are discussed. In terms of implementation, how to capture the vertical transport physics, introduction to kernel estimator and commonly uses test cases are discussed. Besides, the Smooth-Particle Hydrodynamics is briefly discussed to avoid confusion with the particle-tracking method.
Chapter four first uses the sinusoidal vortex case to highlight the importance of boundary condition treatment in the particle-tracking program. It then uses the channel case to examine the efficacy of the recursive particle location update algorithm to prevent particles from going across the boundary. It also compares the magnitude of the deterministic and random diffusion drift term. The fifth chapter mainly checks the viability to transform the diffusion equation to an SDE and solve the original equation by tracking the trajectories of particles and calculating their statistics. Furthermore, chapter six builds upon this result by calculating the convergence order of different SDE numerical schemes and comparing the magnitude of error from different sources. Lastly, chapter seven discusses a two-dimensional advection-dominant tidal flume case. It presents a reliable way to implement inward or outward flux boundary conditions. It also includes the vertical transport physics, by applying vertical background velocity and the random vertical velocity induced by the diffusion. Last but not least, we construct an error measure for sensitivity analysis to examine the stability of the results.
The roadmap of reading this thesis depends on readers' interests. For people who are mainly interested in the application perspectives of the particle-tracking, the fifth and the sixth chapter can be omitted, except the chapter six includes some recommendation on the choice of high-order numerical schemes. And for people who are interested in having a holistic of understanding the particle-tracking framework, they can focus their attention on chapter three as well as the appendix for the underlying mathematics. If they are interested in the implementation of initial and boundary conditions, they can find corresponding contents in chapter four and chapter seven. Last but not least, how to use the particle-tracking framework to study one-dimensional diffusion problem is summarized in Figure 1.2. Readers can refer to this "large picture" during reading from time to time to know where they are.

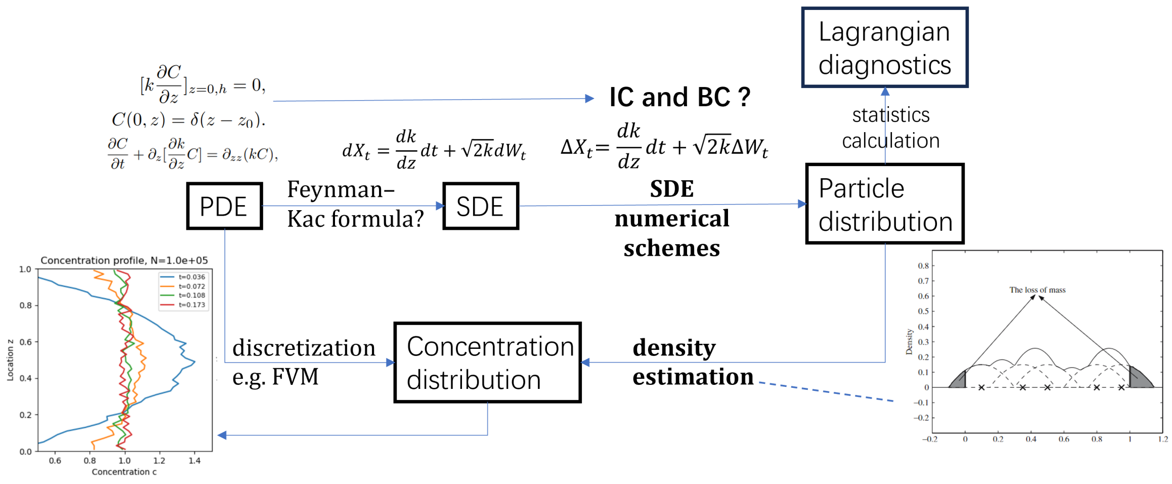

Lagrangian diagnostics
$[k\frac{\partial C}{\partial z}]_{z=0,h} = 0,$
$C(0,z) = \delta(z - z_0).$
$\frac{\partial C}{\partial t} + \partial_z[\frac{\partial k}{\partial z}C] = \partial_{zz}(kC),$
IC and BC ?
statistics calculation
$dX_t = \frac{dk}{dz}dt + \sqrt{2k}dW_t$
$\Delta X_t = \frac{dk}{dz}dt + \sqrt{2k}\Delta W_t$
PDE
Feynman–Kac formula?
SDE
SDE numerical schemes
Particle distribution
Concentration profile, N=1.0e+05
t=0.036
t=0.072
t=0.108
t=0.173
Location z
Concentration c
discretization e.g. FVM
Concentration distribution
density estimation
The loss of mass
Density

Figure 1.2: A explanation on how to use particle-tracking framework to solve transport problem using the 1D diffusion problem as an example.

2 Modelling of salt intrusion

2.1. Introduction

As discussed in the previous section, salt intrusion poses threats to the ecosystem and people's lives upstream along the river, like contaminating the drinking water there. Consequently, salt intrusion models are needed to monitor water quality for water management. On the other hand, sometimes numerical results indicate unobserved phenomena and inspire people to probe into the underlying dynamics of salt intrusion, as seen in [47].

Salt intrusion involves physical processes on different scales. On the largest scale, the seawater flows upstream toward the land along the river. People are often interested in the salt intrusion length, so a one-dimensional model with an appropriate modelling of the salt dispersion coefficient is needed. On a smaller scale, one may observe the mixing between the seawater and the fresh water in the estuary. Note that the mixing is a three-dimensional process that may have characteristic vertical and horizontal length scale, so a two- or three- dimensional model should be used for this purpose. And the seawater may flow through the porous water-bearing aquifer in an underground layer, which is governed by Darcy equation. In a nutshell, people set up various kinds of models for different purposes and have different treatments to capture physical process in surface flow and underground flow on different scales .

The following discussion is not supposed to be a holistic review of the current salt intrusion models. Instead, the governing equations, one of the most important ingredients in salt intrusion modelling, will be discussed. The organization of this chapter will be as follows. Before discussing the modelling details, some basic physics of salt intrusion are discussed. Additionally, two types of commonly used models for salt intrusion are discussed. Furthermore, how salt intrusion modelling connects to climate change is covered. Lastly, salt intrusion in the aquifer that is common in applications is briefly discussed, though it will not be the scope of the thesis project.

2.2. Basic physics of salt intrusion

The discussion below is a synthesis of the first chapter from [38] by Savenije. Salt intrusion is affected by morphology of an estuary, like its shape and bathymetry (the topography of the estuary floor). They are relevant to setting up correct boundary conditions in salt intrusion modelling. On the other hand, both the seawater inflow and freshwater outflow affect the morphology of the estuary. For one thing, tides can also erode the coastal bank and change the shape of the estuary. For the other thing, the freshwater outflow that carries sediments slow down during the mixing process in the estuary, and the sediments settle down in the estuary and may gradually change the morphology of the estuary.

Salt intrusion is driven by two forces, the seawater inflow and the freshwater outflow. The previous is governed by tidal effects, which are generated by the gravitation in the planetary system of the sun, Earth and the moon. The dominant frequencies of tides are semi-diurnal and diurnal, which are related to Earth's rotation. Other tidal effects of longer period exist and correspond to the moon revolving around the Earth and the Earth revolving around the sun. The major effect of tides is that it works as a harmonic oscillator that pumps seawater into the estuary and that pumps the water in the estuary back into the sea. The tidal effect is the major energy provider of salt intrusion. And the freshwater outflow from the river, the other driving force of the salt intrusion, has strong seasonal effects, like the

melting of snow further upstream, rainfall and evaporation. Besides, direct human inference, such as underground water pumping, discharge of runoff water and regulation of river discharge through a large dam, also affects salt intrusion, by adjusting freshwater discharge upstream.
In salt intrusion modelling, we are mainly interested in the mixing processes. The two major mixing processes are turbulent mixing and gravitational mixing. Bottom friction at the seafloor induces shear effect and eddies that lead to turbulent mixing. Gravitational mixing is driven by density difference of the flow. A schematic explanation is given here. Assume initially a salt wedge is observed. In the upper layer where freshwater is dominant, freshwater becomes denser after mixing with seawater and sinks, while seawater becomes lighter after mixing with freshwater and swells. To make up for the sinking and swelling effect, seaward flow in the upper layer and riverward flow in the lower layer should exist. This ends up with a schematic description of gravitational circulation. Gravitational circulation creates a region that is abundant with nutrients constantly updated from the upstream. This helps make the estuary a vital ecosystem. In addition to turbulent and gravitational mixing, tidal trapping and tidal pumping are important mixing processes that are related to the shape of the estuary, though they may not be the focus of the present thesis project. To characterize whether the mixing process is so strong that leads to stratification, recall the Richardson number Ri is often defined as the ratio of buoyancy effect and the inertia effect,

$$Ri = \frac{\Delta\rho g h}{\rho u^2}, \tag{2.1}$$

and similarly, the Estuarine Richardson number Ri_e is defined as the ratio of potential energy gain due to freshwater discharge and the mixing power of the tidal currents from [24] and [9] by Fischer

$$Ri_e = \frac{\Delta\rho}{\rho}\frac{gQ}{WU_t^3} \tag{2.2}$$

where Q is the freshwater discharge, W is estuary width and U_t is root-mean-square velocity of the tidal current. Note that the ratio in the Estuarine Richardson number is a ratio between two types of energy power the increase of potential energy prompts stratification and inhibits mixing, while the increase of tidal energy stimulates mixing. The estuary is regarded well-mixed if $Ri_e < 0.08$, partially stratified if $0.08 < Ri_e < 0.8$ and stratified if $Ri_e > 0.8$. In the previous discussion, it is assumed that the estuary is laterally homogeneous, which means the the Coriollis effect and flow in the lateral direction are not significant. If this not the case, the estuary might be laterally inhomogeneous. The four types of estuaries are plotted in Figure 2.1.

2.3. Salt intrusion modelling by one-dimensional models

In modelling saline intrusion, commonly used models and their application contexts are briefly summarized here

- 1D: working well upstream and mainly used for well-mixed estuaries; gravitational effect parametrization needed.
- 2DH: averaged in the vertical direction; suitable for detecting lateral effects.
- 2DV: contraction in the lateral direction or the assumption that the flow is uniform in the lateral direction; suitable for simulating vertical mixing.
- 3D: incorporating the whole hydrodynamics; suitable for simulating estuary with complex geometry.

The one-dimensional model can be used to determine how far upstream that the salt water goes. The one-dimensional model is simple enough for people to decompose the different driving forces of salt intrusion and analyze them separately.
The derivation of the one-dimensional model is explained here. Assume the horizontal scale is much larger than the vertical scale and that the water pressure is nearly hydrostatic, the shallow water equations (a.k.a. 2D Saint-Venant equation) can be derived from averaging the Navier-Stokes equation over depth. And the 1D Saint-Venant equations can be derived from the the shallow water equations by contraction in one direction. Besides, a salt transport equation is needed to ensure conservation of salt mass. The 1D Saint-Venant equations and the salt transport equation are the governing equations in the one-dimensional model. The coupling between the 1D Saint-Venant equations and the salt

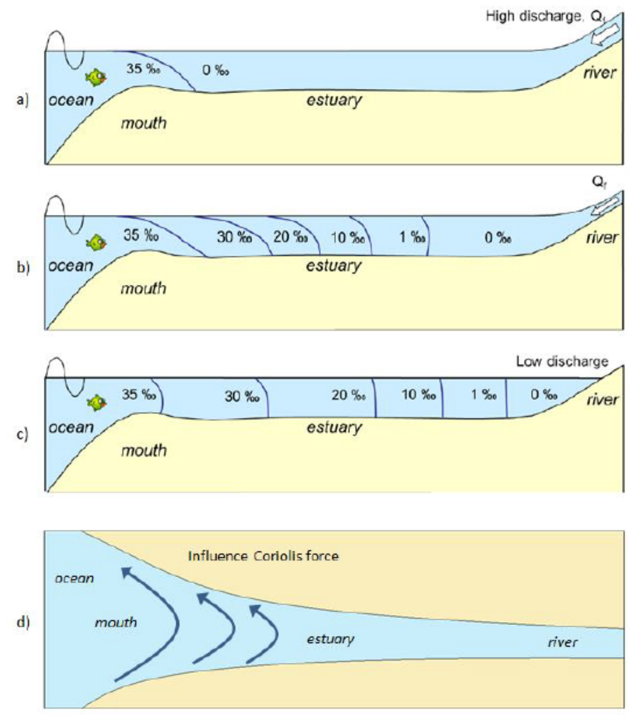


Figure 2.1: Longitudinal distribution of salinity for estuaries with different stratification degree: a) highly stratified (salt-wedge); b) partially mixed; c) well-mixed; d) laterally inhomogeneous, retrieved from [38].

transport equation could be implemented as follows in each time step, the hydrodynamic Saint-Venant equation is first used to updated the velocity field and the salt transport is then computed.
As mentioned before, Savenije [38] identified tide and freshwater discharge as the main driving forces of mixing in the estuary. Nevertheless, the attempts to split the driving forces of mixing into different components failed because people cannot measure or compute the parameters correctly, and more importantly, people don't know how different components interact with each other. To construct a usable model, people turn to empirical models that identify the major driving force in the model. Then, the link between in-situ observation and the magnitude of driving force is drawn through experiments. For verification, this model should be able to capture physical processes at smaller scales.
Daniels' master thesis [5] implemented the above idea. In particular, he set up and compared different dispersion coefficient models that tried to catch mixing mechanism in the estuary. He identified gravitational circulation and tidal pumping as primary driving mechanism compared to those mentioned above. He also used tidal flume experiments and a dataset of convergent esturaries to improve and fit the model. A direct application of the one-dimensional model on salt intrusion can be found in [2] and [10]. Notably in [10], a one-dimensional model was used with the tidal dispersion coefficient computed from a three-dimensional model in order to strike a balance between efficiency and accuracy.

2.4. Salt intrusion modelling using two- or three-dimensional models

Although the one-dimensional models are often computationally efficient, they cannot represent the physics in the other two directions. Notably, the stratification suppresses the mixing in the vertical direction, but its effect cannot be captured by a one-dimensional model.
The hydrodynamic equations in these models are often the two- or three- dimensional incompressible Reynolds-averaged Navier-Stokes (RANS) equations that satisfy the Boussinesq and hydrostatic assumptions. The Boussinesq assumption holds for natural convection problems with small temperature and thus density variation. The density variation is ignored in the temporal and the nonlinear advection term in the compressible Navier-Stokes equation, and it only plays a role in the external force term induced by the gravitation. The hydrostatic assumption says that gravitational force is balanced by the

pressure gradient force. In the vertical direction, a sigma coordinate is often used, which is plotted in Figure 2.2 to take the elevation of the free sea surface into consideration. These equations, accompanied by the salt transport equation as well as the state equation on salinity and temperature, constitute the governing equations in these models. The turbulence closure model is used to model the diffusivity in the horizontal and the vertical directions. And the boundary conditions in both directions are treated differently in different models. There are various ways in the discretization of the governing

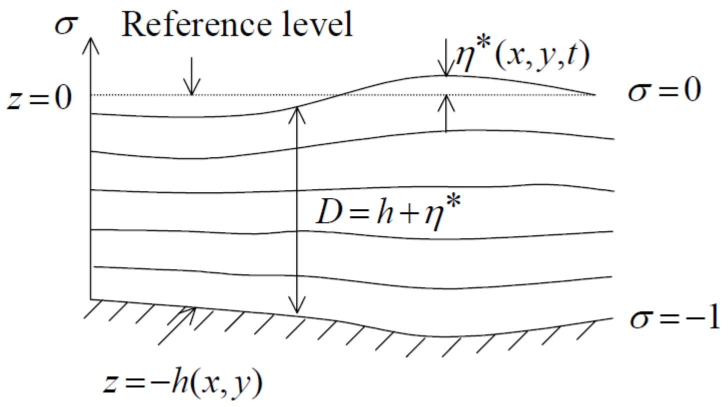


Figure 2.2: A sketch of the sigma coordinate system retrieved from [32] η^* is water surface elevation above static water level $z = 0$, D is the total water depth ($D = h + h^*$) and h is the static water depth. The main advantage of using the sigma coordinate is that one can use an array of surface and bottom elements to represent sea surface elevation and seabed topography.

equations mentioned above. For instance, a high-resolution unstructured-grid finite-volume model is used to study salt intrusion into Changjiang River (Yangtze River) in [47]. And an Euler-Lagrangian model was adopted for the Wu River Estuary in Taiwan in [26]. For more examples about salt intrusion modelling using two- or three- dimensional models, one may refer to [29], which gave a summary of commonly used numerical codes to model salt intrusion.

2.5. Salt intrusion modelling and climate change

It is expected that the salt intrusion will become more severe due to global warming at least in the following two aspects. Firstly, global warming will lead to sea-level rise and the seawater with a larger hydraulic head may flow further upstream. Secondly, the global warming is correlated to more frequent extreme events like coastal flooding and precipitation. Coastal flooding can also be seen as salt intrusion phenomenon that happens in a short period of time. Intense precipitation might have a positive effect on the period where there are frequent rains, but for other periods of time in a year salt intrusion might become more severe. Climate change may affect salt intrusion in other pathways other than sea-level rise and more frequent extreme events, but this will not be discussed here.

Salt intrusion modelling enables people to verify the above ideas about how climate change is linked to the salt intrusion and evaluate its impact. The review paper [29] by Mohammed et al. compared and discussed many case studies, and drew a conclusion that sea-level rise was positively correlated to salt intrusion in spite of different numerical experiment settings. And concretely speaking, according to the case study on the Chesapeake Bay in [18], both in wet and dry seasons the salt content and the salt intrusion length tend to increase, and an irregular temporal distribution of precipitation can also affect the volume of river discharge, which shapes the seasonal and inter-annual variation of saline intrusion. It was found out in [26] that the salt intrusion length would increase and that the flushing the water exchange effect will be suppressed due to sea-level rise. Other than direct modelling, data analysis also plays an important role in people's understanding of the link between climate change and salt intrusion. For example, a non-parametric model in [36] was constructed to capture the link between sea-level rise and increased salinity by analyzing a multi-decadal in-situ observation dataset. The study found out that the salinity would increase regardless of streamflow conditions. In another example, the output from a verified numerical model was used for data analysis to study the salinity intrusion variation due to changes in streamflow and tidal conditions in [33].

Note that the people's interventions like groundwater pumping, dredging and regulation of water flow through the dam can have impacts on salt intrusion. These are often discussed in case studies, and they are not discussed here.

2.6. Salt intrusion modelling in the aquifer: a digression

To make the discussion of salt intrusion complete, in addition to salt intrusion in surface flow, talk about salt intrusion in the aquifer, which is a solid layer under the ground near an estuary with many cavities that preserves water. The seawater often flows beneath the freshwater and has the possibility to penetrate through the river bed and the river bank, and enters into the aquifer. This may have a detrimental effect on the quality of the underground water in the coastal aquifer. The penetration process is governed by the Darcy law, which describes how fluid flows through a porous medium. Many studies on salt intrusion have been devoted to modelling this phenomenon. Some current advances can be found in [1], [28] , [4] and [12]. And a review paper [19] summarizes the available hydraulic and physical techniques to treat salt intrusion into aquifer. These will be outside the scope of the current work, because it is assumed that the riverbed is impermeable and only the surface current is considered.

3
A brief description of the particle-tracking method

3.1. Introduction

As mentioned in the introductory chapter, one motivation of this project is to make use of the numerically resolved flow field information from the Delft3D model. People are interested in releasing virtual particles in the flow field reconstructed by the available data and track where these particles go. Consequently, people often call such methods as Lagrangian diagnostics or particle-tracking methods.

Using the particle-tracking method to simulate salt intrusion belongs into the type of tracking the transport of passive scalars or tracers in the ocean. Passive scalars are diffusive substances often with low concentration in the fluid which have no impact on the dynamics of the fluid. Although this definition is vague, the salt dissolved in the saline ocean water can be treated as a passive scalar for particle-tracking method, because the density variation of ocean water induced by the dissolved salt is already considered by the ocean models. In applications, the particle-tracking method is not limited to ideal passive scalars. People also use the particle-tracking method to track floating plastics [39] or flow that carries sediments [3]. The idea is that as long as the space that the particles occupy is small and not comparable to the length scale that we are interested in, then the added particles almost have no impact on the flow dynamics and the particle tracking method can be applied.

As for salt intrusion modelling, the particle-tracking method has several advantages over grid-based methods, like the finite volume method. Firstly, the particle-tracking method calculates trajectories of particles, which enables people to conduct the pathway, connectivity and water age analysis much easier than using a grid-based method. This is the major advantage that we favour the particle-tracking method over other grid-based methods. For example, we may use the particle-tracking method to study the salt intrusion. We can reconstruct a “salt water front” by the particles that travel to the furthest upstream position. This is helpful in determining whether the salt water can travel so further upstream that it contaminates the water around the water intake point. Also, the trajectories of particles provide a “salt intrusion map” in the river-delta region where a river has many branches. Besides, the residence time of salt particles in the estuary can also be computed. The residence time is an important criterion for evaluating the flushing effect of fresh water. Secondly, the particle-tracking method may reflect the flow information at the subgrid scale, which will be inefficient to compute is one uses a grid-based method. Thirdly, the particle-tracking method has a better conservation property. For example, when calculating the evolution of the concentration field, one may end up having negative concentration values if a grid-based method is used. This often happens in the neighbourhood of a sink or source, where there are large concentration gradients. However, the particle-tracking method only gives positive concentration values.

In addition to using the particle-tracking method to simulate salt intrusion, this method can also be applied in biological connectivity, because the fish larvae often migrate along with currents. Furthermore, by providing insight into biological connectivity, the particle-tracking method can help people model the expansion of certain species in population and in spatial distribution. More information regarding this can be found in [41]. Besides, by applying the particle-tracking method, people can label water parcel

to answer the question about where this parcel comes and goes. This can help people understand the the water transit time between different regions and the ventilation time in the ocean circulation [43]. This type of application distinguishes itself in that the incompressibility of water enables people to bring up special methods to treat it. For more applications regarding the particle-tracking method, interested readers are referred to [41].
To apply the particle-tracking method, one needs to model the missing physics in the output of Delft3D FM model. This is mainly because of the grid size. The grid size in the ocean circulation model is often on the order of kilometers. A regional model might have a smaller grid size on the order of 10 meters, but it cannot resolve the eddies below this size. As for motion induced by eddies below the grid size, the advection-diffusion equation is introduced and a stochastic counterpart is formulated. This equation not only includes the molecular diffusion of the passive tracer but also accounts for the subgrid-scale transport by eddies smaller than the grid size.
The advection-diffusion equation (ADE) is transformed to the form of the Fokker-Plank equation and connected to a stochastic differential equation (SDE). This advection-diffusion equation characterize the movement of an ensemble of particles by the variable concentration, while the stochastic differential equation describes the motion of each single particle.
The organization of this chapter will be as follows. Firstly, the link between the ADE and the corresponding SDE is explained. Secondly, several SDE numerical schemes are discussed. For more theoretical background on stochastic process and SDEs, one may refer to the discussion in Appendix A. Thirdly, The kernel density estimation method to “translate the distribution of particles into that of concentration” will be discussed. Fourthly, several test cases with space-varying diffusivities in the vertical direction will be discussed. Lastly, the smooth particle hydrodynamics that is often confused with the particle-tracking method will be briefly discussed.

3.2. Link the advection-diffusion equation to an SDE

The subgrid unresolved physics are modelled by the advection-diffusion equation. Follow the discussion in [41], the transport equation of the passive tracer is given by

$$(\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\nabla)\boldsymbol{C} = \nabla\cdot(\boldsymbol{J}\cdot\nabla\boldsymbol{C}), \tag{3.1}$$

where C is the concentration of the passive tracer and $\boldsymbol{J}$ is the transport tensor that represents the subgrid transport. It’s a common practice to decompose the transport tensor into a symmetric part $\boldsymbol{K}$ and an anti-symmetric part $\boldsymbol{A}$.

$$\boldsymbol{J} = \boldsymbol{K} + \boldsymbol{A}. \tag{3.2}$$

The symmetric part will be positive definite if it corresponds to diffusion, while the anti-symmetric part is often linked to advection. With this decomposition, the transport equation becomes

$$(\frac{\partial}{\partial t} + \boldsymbol{v}^{\dagger}\cdot\nabla)\boldsymbol{C} = \nabla\cdot(\boldsymbol{K}\cdot\nabla\boldsymbol{C}) \tag{3.3}$$

where

$$\boldsymbol{v}^{\dagger} = \boldsymbol{v} + \boldsymbol{v}^{*} \tag{3.4}$$

defines the residual-mean velocity and

$$v_j^{*} = -\partial_i A_{ij} \tag{3.5}$$

is called the eddy-induced velocity. Note that the eddy-induced velocity is nondivergent, so is the residual-mean velocity. This results in the tracer equation with a advective flux term on the LHS

$$\frac{\partial \boldsymbol{C}}{\partial t} + \nabla\cdot(\boldsymbol{v}^{\dagger}\boldsymbol{C}) = \nabla\cdot(\boldsymbol{K}\cdot\nabla\boldsymbol{C}). \tag{3.6}$$

Note that

$$\partial_i\partial_j(K_{ij}C) = \partial_i(K_{ij}\partial_j C + C\partial_j K_{ij}).$$

Transform the advection-diffusion equation into the form of a Fokker-Plank equation

$$\frac{\partial \boldsymbol{C}}{\partial t} + \nabla\cdot(\boldsymbol{v}^{\textbf{drift}}\boldsymbol{C}) = \nabla\nabla : (\boldsymbol{K}\boldsymbol{C}), \tag{3.7}$$

where

$$\boldsymbol{v}^{\text{drift}} = \boldsymbol{v}^{\dagger} + \nabla \cdot \boldsymbol{K} \tag{3.8}$$

defines the drift velocity. To characterize the motion of particles in a diffusive fluid, one can track the trajectory of each particle, which is described by Equation (A.4). Now rewrite it with the definition of Ito integral,

$$d\mathbf{X_t} = \mathbf{f}(\mathbf{X_t}, t)dt + \boldsymbol{\sigma}(\mathbf{X_t}, t)d\mathbf{W}_t, \mathbf{X}_{t_0} = \mathbf{X}_0, \tag{3.9}$$

where $\mathbf{f}$ is the deterministic drift and $\boldsymbol{\sigma}$ is a tracer diffusion tensor. And W_t is a standard Wiener process to represent the uncertainty in the motion. One can also characterize the motion of particles by studying the evolution of the particle cloud. For this purpose, the concept of the transitional probability is introduced.
The transition probability $p(t, x_t|t_0, x_{t_0})$ is defined to be the conditional probability of a particle that starts at location x_{t_0} and ends up at location x_t. To characterize the motion of the particles in a fluid, one studies the Fokker-Plank equation that describes the evolution of the transitional probability

$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} + \nabla \cdot (\boldsymbol{f}\boldsymbol{p}) = \nabla\nabla : (\boldsymbol{B}\boldsymbol{p}), \quad \boldsymbol{p}(\boldsymbol{t}_0, \boldsymbol{x}) = \boldsymbol{p}_0(\boldsymbol{x}), \tag{3.10}$$

where f is the drift term and

$$\boldsymbol{B} = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^{\boldsymbol{T}} \tag{3.11}$$

is the diffusive term. To align with the Equation (3.7), one takes

$$\boldsymbol{B} = \boldsymbol{K}, \quad \boldsymbol{f} = \boldsymbol{v}^{\text{drift}} = \boldsymbol{v}^{\dagger} + \nabla \cdot \boldsymbol{K}, \quad \boldsymbol{p} = \boldsymbol{C}. \tag{3.12}$$

The transition probability and concentration are linked in the sense that higher concentration implies a higher chance for a particle to occur. Note that if the concentration is calculated by the number of particles over an infinitesimal region, both transition probability and concentration have the same dimension. The concentration and the probability density of particles are differed only by a constant. Additionally, the SDE (3.9) and the Fokker-Planck equation (3.7) not only represent two ways of characterizing the motion of the particles, but also have a formal mathematical link. Interested readers are referred to [48] and [35].
Lastly, some intuition about how concentration is linked to probability is given. Assume the whole computation domain is divided into an infinite number of infinitesimal regions. The concentration at a certain position is the (limit) ratio of the number of particles in the infinitesimal region over its volume, and the integration of concentration over the whole domain gives a constant, which is the number of all particles. Similarly, the probability distribution at a certain position is the (limit) ratio of the probability that the particle occurs there over the volume of the infinitesimal region, and the integration of probability density over the whole domain gives the unity. Consequently, the concentration and the probability density are only differed by a constant, the number of all particles in the whole domain.

3.3. SDE numerical schemes

In the first place, discuss the notion of the order of convergence. In this section, only the one-dimensional SDE is discussed for the purpose of a clear illustration. Take the following ordinary differential equation as an example

$$\frac{dx}{dt} = f(x,t), x(t_0) = x_0. \tag{3.13}$$

A numerical scheme of solving the ODE above is

$$x_{t_{k+1}} = x_{t_k} + f(x_{t_k}, t_k)\Delta t, \tag{3.14}$$

where Δt is the step size. Take $T = t_0 + N\Delta t$ and define $T_N = \{t_k = t_0 + k\Delta t \,|k = 0, 1, \dots, N\}$. Write the analytical solution of the ODE as $x(t)$. If

$$|x_{t_N} - x(T)| \le M(\Delta t)^{\gamma}, \tag{3.15}$$

where M is a positive number, the numerical scheme of the ODE is said to be convergent in order γ. In the appendix, the Euler scheme (A.17) is introduced to solve the stochastic differential Equation (A.20) in the Ito sense to find an integral curve of the trajectory of a particle

$$x_{t_{k+1}} = x_{t_k} + f(x_{t_k}, t)\Delta t + \sigma(x_{t_k}, t)\Delta W_{t_k}, \tag{3.16}$$

where $\{x_{t_k},\ t_k \in T_N\}$ is a realization of the random process $\{X_t, t \in T\}$. $\{\Delta W_{t_k},\ t_k \in T_N\}$ is a realization of a sequence of normally distributed random variable with mean 0 and variance Δt. To account for the effect of different realizations, we need to "add" in the Equation (3.15) an expectation operator. This is indeed the strong convergence of a stochastic process

$$E\{|X_{t_N} - X(T)|\} \leq M(\Delta t)^\gamma, \tag{3.17}$$

where again M is a positive number and the Euler scheme for the random process $\{X_t, t \in T\}$ is said to converge strongly with order γ. This notion of convergence is also known as trackwise convergence, because any deviation from the exact track of each particle contributes to the total error. To model passive tracer transport like salt or contaminants, one might be interested in the statistics of the particles rather than the position of any single particles. This leads to the definition of the weak convergence, with a smooth function h,

$$|E\{h(X_{t_N})\} - E\{h(X(T))\}| \leq M(\Delta t)^\gamma, \tag{3.18}$$

where again M is a positive number and the Euler scheme for the random process $\{X_t, t \in T\}$ is said to converge in the weak sense with order γ. If we take $h(x) = x$, we see that the weak convergence can reflect the the deviation of the mean position of different numerical realizations from the true mean positions. The weak convergence might can have a larger convergence order than the strong convergence because the errors of different realizations might cancel out. For example, the Euler scheme converges in the strong sense with order $O(\Delta t)^{1/2}$ and converges in the weak sense with order $O(\Delta t)$.

People came up with ways to numerically solve ODEs with that have a higher order of convergence than Euler methods. These methods can be found with Taylor approximation or belong to the type of predictor-correction methods, like the Runge-Kutta methods. One can also set up high-order methods for SDEs from these two perspectives. However, in light of the Ito lemma, one can anticipate that the numerical schemes found with Taylor expansion for SDEs will include more terms than ODEs. The Milstein scheme to solve Equation (A.20) is a good example

$$x_{t_{k+1}} = x_{t_k} + f(x_{t_k}, t_k)\Delta t + \sigma(x_{t_k}, t_k)\Delta W_{t_k} + \frac{1}{2}\sigma(x_{t_k}, t_k)\frac{\partial \sigma}{\partial x}(x_{t_k}, t_k)(\Delta W_{t_k}^2 - \Delta t), \tag{3.19}$$

which converges in the strong sense with order $O(\Delta t)$. Higher order schemes than the Milstein scheme are also possible. For example, there is one scheme that converges in the strong sense with order $O(\Delta t)^{3/2}$ and in the weak sense with order $O(\Delta t)^2$. Interested readers are referred to [16]. After the discussion of numerical methods that are obtained with Taylor expansion, two examples of the predictor-corrector methods are introduced. As for Equation (A.20) under the Ito sense, the Heun scheme that mimics the trapezoidal method in numerical integration is introduced,

$$x^p_{t_{k+1}} = x_{t_k} + f(x_{t_k}, t_k)\Delta t + \sigma(x_{t_k}, t)\Delta W_{t_k} \tag{3.20}$$

$$x_{t_{k+1}} = x_{t_k} + \frac{1}{2}(f(x_{t_k}, t_k) + f(x^p_{t_{k+1}}, t_k))\Delta t + \sigma(x_{t_k}, t_k)\Delta W_{t_k}. \tag{3.21}$$

The Heun scheme for the SDE in the Ito sense converges in the strong sense with order $O(\Delta t)^{1/2}$ and in the weak sense with order $O(\Delta t)$. To achieve higher order convergence in the strong sense, we want to include the stochastic terms in the correction step. It is now stated without proof that this can be achieved by considering the SDE in the Stratonovich sense (A.22) and ends up with the following scheme

$$x^p_{t_{k+1}} = x_{t_k} + f(x_{t_k}, t_k)\Delta t + \sigma(x_{t_k}, t_k)\Delta W_{t_k} \tag{3.22}$$

$$x_{t_{k+1}} = x_{t_k} + \frac{1}{2}(f(x_{t_k}, t_k) + f(x^p_{t_{k+1}}, t_k))\Delta t + \frac{1}{2}(\sigma(x_{t_k}, t_k) + \sigma(x^p_{t_{k+1}}, t_k))\Delta W_{t_k}. \tag{3.23}$$

Such scheme converges both in the strong sense and in the weak sense with order $O(\Delta t)$.

3.4. Modelling the vertical transport physics

Since the vertical length scale is often several orders smaller than the horizontal length scale, it is essential to capture missing physics with the advection-diffusion equation in the vertical direction. For the same reason, the one-dimensional test case is widely applied and will be discussed in the next section. A major goal of this thesis work is to add the vertical transport to the already existent two-dimensional particle tracking model. The two-dimensional model is suitable for tracking the floaters on the sea surface, as seen in [7], while the three-dimensional model is supposed to take the vertical transport into the consideration. This is important in modelling the salt intrusion in the Rhine-Meuse Delta, because the stratification in the vertical direction suppresses mixing of the sea water and fresh water in the vertical direction. Consequently, the dynamics of salt intrusion are changed, for instance, the salt sea water is more likely to travel upstream along the river bed.
To capture the physics in the estuary and model the salt intrusion appropriately, a high order scheme might be necessary for capturing the vertical transport. In [15], Graewe et al. argues that the simple Euler scheme is not enough for the particle-tracking method in the current ocean models. High order numerical schemes helps capture the transport physics, especially the stratification in the vertical direction as revealed in [45]. The stratification suppresses mixing, which means that in the simulation with the particle-tracking method the pycnocline (the surface that separates two layers of water with different densities and thus is the location with the largest density gradient) will act as an impermeable barrier for particles. Low-order schemes like the Euler method have large numerical diffusion that causes large leakage across the pycnocline surface the and contaminates the stratification phenomenon, while high-order schemes with small numerical diffusion makes the pycnocline more impermeable. The comparison is shown in Figure 3.1. In addition to a proper SDE numerical schemes, a good representation

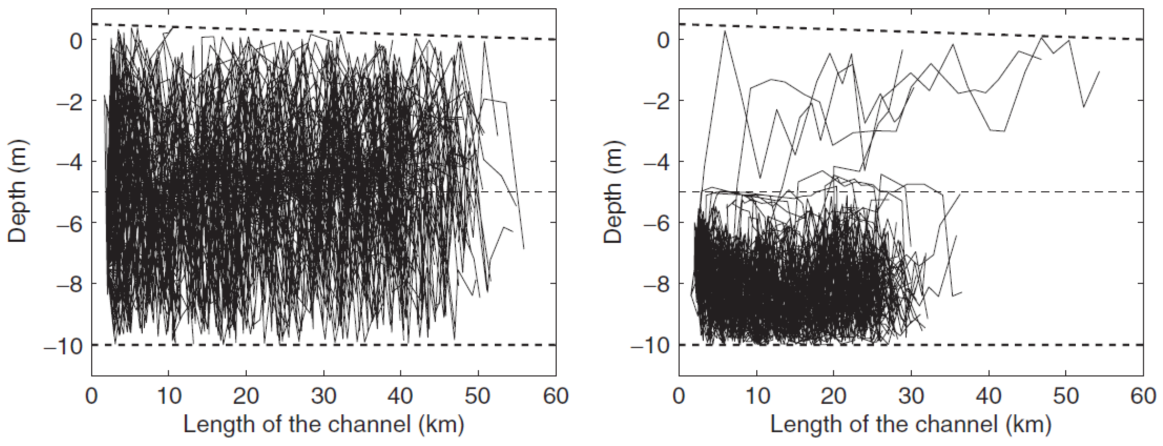


Figure 3.1: The release of 100 particles in stratified flow in a channel. A vertical cross section of the channel is shown. On the left side a numerical simulation with the Euler scheme, and on the right side the same simulation with the higher order stochastic Runge–Kutta scheme. The simulation time was equal to 36 h, with a time step of one hour.

of the diffusion profile is needed to simulate the vertical transport appropriately. It is stated in [14] that the parabola diffusion profile is generally accepted if the estuary is well mixed and thus no significant stratification occurs. The intuition is that at the sea bed (bottom) and at the air-water interface (top) are solid boundary without any diffusion, and the diffusion will be most intensive in the middle because boundary effect becomes weak. The parabola diffusion profile is also favoured because an analytical solution to the pure diffusion problem with parabolic profile exists.
In the case of strong stratification, a so-called "double-parabola" diffusion profile (Figure 3.2) was first suggested by [45] and formally established in [14] along with observation data. Later, it is discussed in [44] and [16]. Exact formula of this along with a shape parameter a

$$k(z) = \begin{cases} \frac{2(1+a)(1+2a)}{a^2 H^{1+\frac{1}{a}}} z(H-2z)^{\frac{1}{a}}, \ 0 \le z \le \frac{H}{2}, \\ \frac{2(1+a)(1+2a)}{a^2 H^{1+\frac{1}{a}}} (H-z)(2z-1)^{\frac{1}{a}}, \ \frac{H}{2} \le z \le H, \end{cases} \tag{3.24}$$

where the profile will consist of two parabolas if $a = 1$, and a large value of a implies a large gradient of the diffusion profile at the pycnocline as seen in Figure 3.2. Due to the stratification, the turbulent

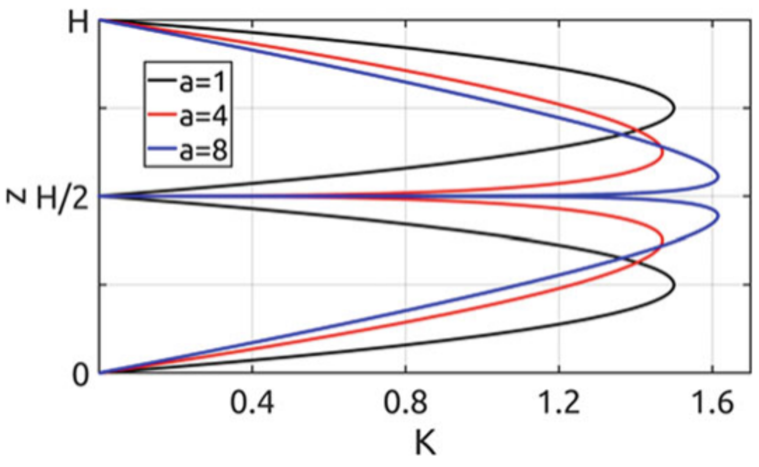


Figure 3.2: Diffusivity profile $K = k/\overline{k}$ for different values of the parameter α, where $\overline{k}$ represents the average value of the diffusion strength.

mixing is suppressed, and the molecular mixing is often several orders smaller than that of turbulent mixing. Thus, this forces the diffusion profile at the interface of the stratification to be near zero.
Last but not least, Stijnen [45] proposed an intuitive way to implement an pycnocline (stratification boundary) or other boundary surfaces as solid boundary that prevents passive tracers crossing the boundary. This is summarized in Algorithm 1. The underlying assumption is that in the neighbourhood of the boundary, the diffusion becomes so small that no particle can go across the boundary due to the boundary effect. This methods turns out to be effective in preventing particles from going across the isopycnal line as shown in Figure3.3, though its mathematical validity is under doubt.

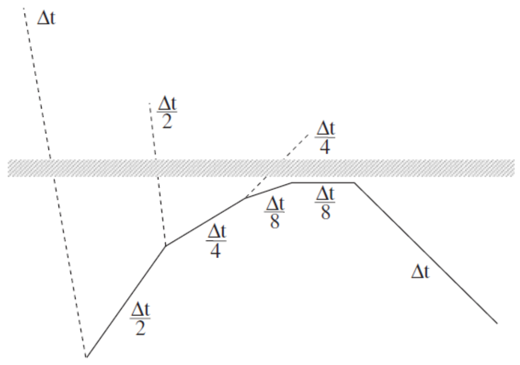


Figure 3.3: Example of the movement of a particle that bends along the coast retrieved from [45].

3.5. Introduction of density estimation

After calculating the trajectories of particles, it is sometimes necessary to transform the spatial distribution of particles to the concentration distribution in the spatial domain. In other words, we need to estimate a density. Remember that the density and the concentration are related by the fact that implies a high probability for the particle to occur implies high concentration in this neighbourhood, as seen in Equation (3.12). Naturally, one can divide the spatial domain into many small boxes, count the number of particles in each box and divide the number by the volume of the box. The simple estimation method has some disadvantages. It requires sufficiently many particles and corresponding boxes to achieve a good estimation of the concentration. Specifically, the concentration estimation is subjected to the selection of boxes, like the size or the center of those boxes. Also, the number of the boxes used in the estimation cannot be too large if a smooth concentration distribution is wanted. As for computation aspects, using too many boxes is also a large computational burden.

Algorithm 1 Particle location update algorithm by Stijnen et al. [45]

```
1: procedure Locupdate($x^{(k)}$,$y^{(k)}$, $u(x,y)$, $v(x,y)$, $\Delta t$, U)    ▷ $U$ is the finite computation domain.
2:     $x^{(k+1)} = x^{(k)} + u(x^{(k)}, y^{(k)})\Delta t$
3:     $y^{(k+1)} = y^{(k)} + v(x^{(k)}, y^{(k)})\Delta t$
4:     if $(x^{(k+1)}, y^{(k+1)}) \in U$ then
5:         return $(x^{(k+1)}, y^{(k+1)})$
6:     else
7:         $(\tilde{x}^{(k+1)}, \tilde{y}^{(k+1)}) \leftarrow$ Locupdate($x^{(k)}$,$y^{(k)}$, $u(x,y)$, $v(x,y)$, $\frac{\Delta t}{2}$, $U$)
8:         $(x^{(k+1)}, y^{(k+1)}) \leftarrow$ Locupdate($\tilde{x}^{(k+1)}$,$\tilde{y}^{(k+1)}$, $u(x,y)$, $v(x,y)$, $\frac{\Delta t}{2}$, $U$)
9:         return $(x^{(k+1)}, y^{(k+1)})$
```

The kernel estimator allows people to reduce the number of particles in the density estimation by an order of magnitude. The following will only be a brief discussion about the kernel estimator method that focus on the application. For more information about it, interested readers are referred to [44] and [43]. Intuitively speaking, each particle "induces" nontrivial concentration in a small domain around it. The kernel estimator takes the form

$$\hat{p}(t, \mathbf{x}) = \frac{1}{N\lambda^d} \sum_{n=1}^{N} K(\frac{\mathbf{X}^{(n)} - \mathbf{x}}{\lambda}), \tag{3.25}$$

where N is the number of realizations of stochastic process $\overline{X}$, d is the dimension of the spatial domain, $K(\mathbf{u})$ is a kernel function with $\int K(\mathbf{u})d\mathbf{u} = 1$ and $\mathbf{X}^{(n)}$, $n = 1, \ldots, N$ are samples from the stochastic process $\mathbf{X}^{(n)}$. Two kernel functions which are density function symmetric around the origin are often used

$$K(\mathbf{u}) = (2\pi)^{-d/2} \exp{(-\frac{1}{2}\mathbf{u}^\mathsf{T}\mathbf{u})} \tag{3.26}$$

is the Gaussian kernel and

$$K(\mathbf{u}) = \frac{1}{2}\nu_d^{-1}(d+2)(1 - \mathbf{u}^\mathsf{T}\mathbf{u})|_{(\mathbf{u}^\mathsf{T}\mathbf{u} \le 1)} \tag{3.27}$$

is the Epanechnikov kernel, where $\nu_d = 2\pi^{d/2}/(d\Gamma(d/2))$ is the volume of the unit d-dimensional sphere and $\Gamma(x)$ is the Gamma function. Choosing the kernel function to be density function ensures that the kernel estimator is also a density. In Figure 3.4, a density estimation using three points and the Gaussian kernel is demonstrated. It can be shown that the kernel estimator is consistent if the one-

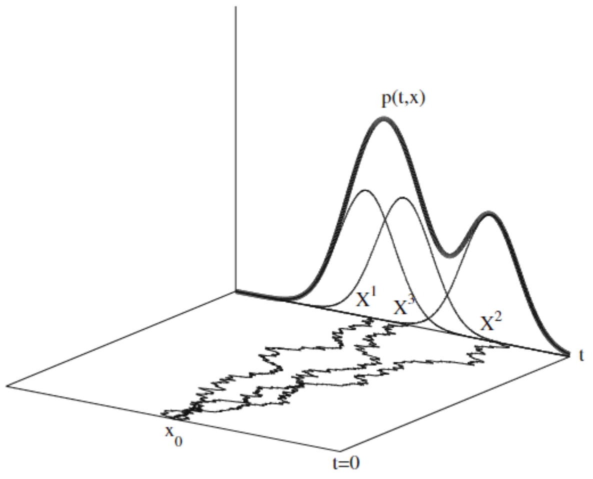


Figure 3.4: A density estimator using three points with the Gaussian kernel retrieved from [44]

dimensional density $\hat{p}(t, x)$ is twice continuously differentiable in $\mathcal{R}$. However, the computational domain of interest is often finite. "Leakage of the induced probability" (seen in Figure 3.5) happens if a particle is situated close enough to the boundary such that a proportion of the domain with induced concentration is outside the computational domain. To construct a consistent kernel estimator, a simple way is to reflect the the induced "probability density" back and add to the density function within the computation domain. According to the Chapter 2.4 in [43], the reflection method can ensure a consistent estimator, but still results in a large bias near the boundary, which is proportional to the used bandwidth (i.e. $O(\lambda)$). To improve this situation, the boundary kernel method was also discussed in [44]. In [30] where this method was first proposed, the major idea can be summarized as using certain polynomials as kernel for density estimation near the endpoints. However, this will be beyond the scope of this thesis for two reasons. Firstly, the author admits that he does not have enough knowledge to grasp how to implement this method and more importantly analyze the results. Secondly, the program will be run using the GPU, so too many conditional statements will be detrimental to the efficiency of the program. For the above mentioned, only the reflection method will be used. For more detailed discussion of the kernel density method and error analysis, readers may refer to [43] and [44]. And for an overview of the density estimation methods, readers are strongly recommended to [42].

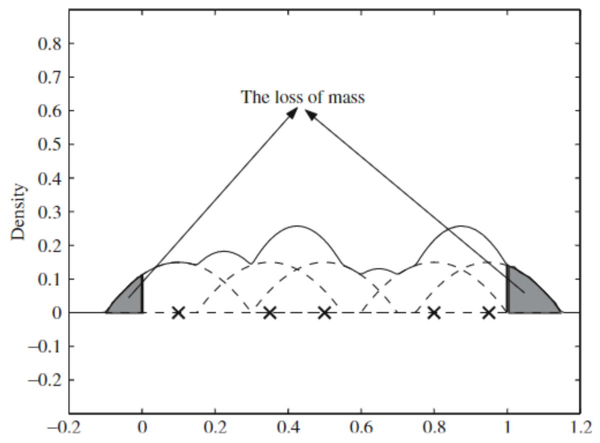


Figure 3.5: The loss of mass due to kernel estimator retrieved from [44]

3.6. Introduction of test cases

Before the stochastic numerical schemes in the implementation of the particle-tracking method are used in applications, they are tested with several commonly accepted test cases. The test case can be a one-dimensional problem or it can be a two- or three-dimensional one. Analytical solutions of the test cases are often available for verification purposes.

In the one-dimensional case, people are interested in the pure diffusion problem and the advection-diffusion problem. The diffusion problem is often set with all particles staying at the same place initially. If the vertical diffusion profile is a parabola, an analytical solution of the concentration distribution is available for comparison with the concentration distribution estimation constructed by density estimation method. And if the vertical diffusion profile is a "double parabola", one is interested in the effect of the barrier effect of the isopycnal surface (the surface where the flow is of same density) and thus calculate the proportion of particles that cross the isopycnal surface. As for the advection-diffusion problem, one is interested in the averaged residence time of particles in the domain with unity length. These test cases are summarized in [15] and also discussed in [44] and [16]. The one-dimensional pure problem will also be discussed in Chapter 5 and Chapter 6.

In two- or three-dimensional case, people often use test cases of the pure diffusion problem along the isopycnal surface. Even though this will be beyond the scope of this thesis, a summary of that is given here. If the isopycnal surface is flat, the diffusivity tensor is a diagonal matrix in the principal axes that includes an isopycnal component and a diapycnal component (the transport component along or across the isopycnal surface). To relate the principal axes and the geoscience coordinates, one can transform the diffusivity tensor with a rotation matrix. Release particles at the origin initially, the analytical solution of the concentration distribution to the pure diffusion problem with flat isopycnal surface exists for comparison. And if the isopycnal surface is non-flat, one is often interested in the pure diffusion process without the diapycnal component, where the particles released on the isopycnal surface will remain on

the surface in the simulation. However, the numerical error for the particle-tracking method tends to draw the particle away from the isopycnal surface. Thus, an error measure can be constructed that calculates the average deviation of all particles from the isopycnal surface. These cases are discussed extensively in [46]. They are also discussed in [44] and [16].
After applying the particle-tracking framework to test cases, we can draw our attention to a relatively realistic 2D case, the tidal flume case, as we will discuss in Chapter 7.

3.7. A digression into the smooth particle hydrodynamics

Speaking of particle method, people from the community of fluid mechanics immediately have smooth particle hydrodynamics (SPH) in mind, which was developed in [13] by astrophysical experts Monoghan and Gingold in 1977. The idea of SPH is simple: it replaces fluid parcels, where a single fluid parcel contains a large number of water molecules and is regarded small macroscopically in space at the same time, with an ensemble of particles. These particles often have a certain weight and a circular or ball-shaped geometry. The particles can collide with each other as well as with the boundary wall. One can imagine, if a large number of particles is used, their motions can reflect the motion of the fluid. This method has the advantage over methods with meshes like the finite element method, in that it can deal with large deformation that happens rapidly. This is because the particles that “comprise of the boundary” can move more freely than the boundary of a cell.
The SPH is widely used in different applications, such as fluid-structure interaction in [25] that involves large deformation of free surface like wave impact on solid structure under water and parachute landing, the flow modelling in granular media like the machining vibratory finishing process in [11], and geotechnical problems in [20]. Besides, the SPH also makes its way into saline intrusion modelling, which is discussed in the review paper [34]. However, the SPH will not be the scope of this work.

4 Simple test cases

Before getting into detailed discussion of particle-tracking method, it will be interesting to observe some simple test cases that inspires us to think about its potential problems, because the addition of diffusion mechanism into the system can make the particles go outside the boundary.

4.1. sinusoidal vortex

4.1.1. Problem description

To have some intuition about what particle-tracking method is, it would be nice to start with a really simple test case – a sinusoidal vortex. This problem setting follows from Nelleke's bachelor thesis in [39]. The computational domain is a rectangular region, with following velocity field

$$u = -sin(dx)cos(dz), \tag{4.1}$$

$$w = cos(cx)sin(cz). \tag{4.2}$$

where the height of the domain $H = 50$, the length of the domain $L = 2H$ and the parameter $c = d = \pi/H$, as indicated in Figure 4.1. Consider the advection-diffusion equation, i.e. Equation 3.10) in Chapter three. We are now mainly interested in the equation in a two-dimensional rectangular domain with $\boldsymbol{B} = \text{diag}(\epsilon_x, \epsilon_z) = (1, 10^{-5})$ and $\boldsymbol{v} = [uv]^T$.

$$\frac{\partial c}{\partial t} + \frac{\partial}{\partial x}\left[(u + \epsilon_x)\frac{\partial c}{\partial x}\right] + \frac{\partial}{\partial z}\left[(w + \epsilon_z)\frac{\partial c}{\partial z}\right] = \frac{\partial^2}{\partial^2 x}(\epsilon_x c) + \frac{\partial^2}{\partial^2 z}(\epsilon_z c). \tag{4.3}$$

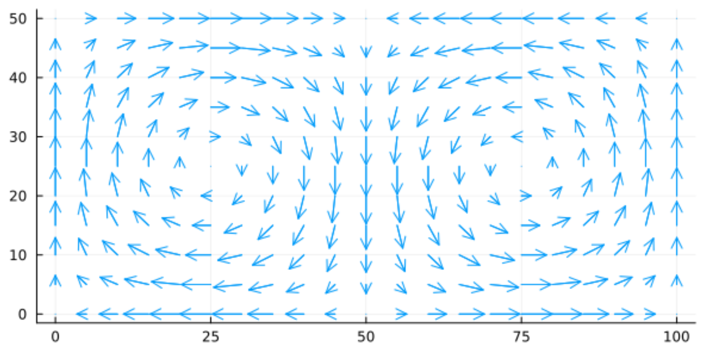


Figure 4.1: Sinusoidal vortex induces a rotational velocity field - the horizonal direction is x-direction and the vertical direction is z-direction.

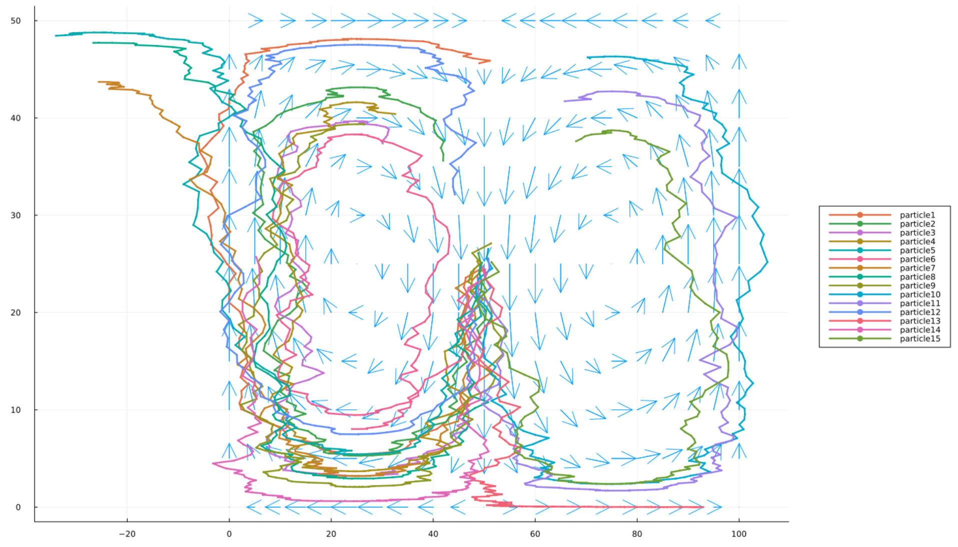


Figure 4.2: Particle tracking simulation in a sinusoidal vortex.

By the relation introduced between Equation (3.10-3.12) (or the Feynman–Kac formula), the following control equation of particle motion is derived.

$$dX_t = (u + \frac{d\epsilon_x}{dx})dt + \sqrt{2\epsilon_x}dW_t, \tag{4.4}$$

$$dZ_t = (u + \frac{d\epsilon_z}{dz})dt + \sqrt{2\epsilon_z}dW_t, \tag{4.5}$$

where dW_t is the increment of the corresponding Wiener process, which obeys a normal distribution centered at zero with variance dt.

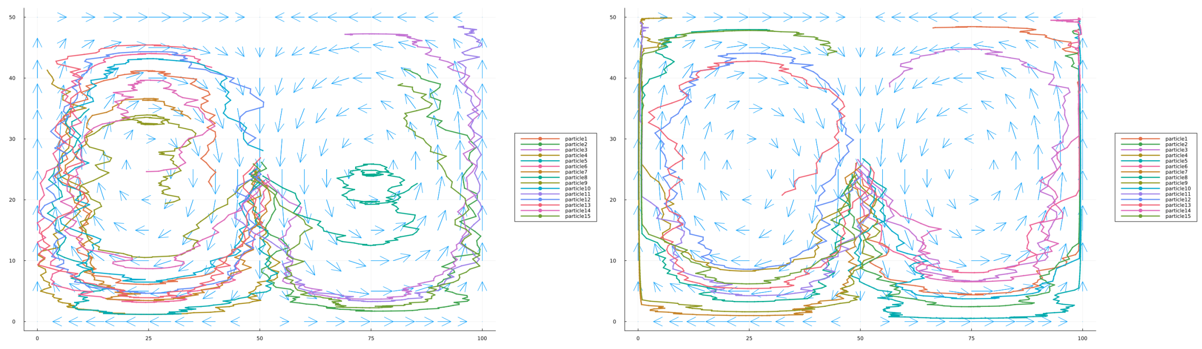
4.1.2. Results and discussion

The problem setting seems to be easy, but as seen in Figure 4.2, it is immediately noticed that the particle may go out of the domain. Of course, this is unsatisfactory, so methods to tackle this are necessary. For practical purposes, two simple solutions can be used. Firstly, one can take a reflection method. Once the particle is going to cross the boundary, then reflect it back. Secondly, one can take a diffusion suppression method. The reason for the particle to travel out of domain is diffusion. Thus, once the particle is near the boundary, i.e. the distance is less than three standard values away from the expected displacement, then no displacement updated is carried out in that direction. Both methods work for this simple problem, as seen in Figure 4.3. The difference between these two methods is that the suppression of diffusion will enable the boundary region to be an “attractor” that attracts particles.
If we consider the reason why particles go out of the domain carefully, we immediately recognize that this is a problem of setting a proper boundary condition. The reflection method and the diffusion suppression method are both ways to approximately implement the impenetrable boundary conditions. This problem is expected to be difficult if the boundary is not flat. Also, one may already realize that the suppression diffusion method is not realistic for more complicated cases, because it would not be easy to determine the boundary region that may evolve with time. Last but not least, the modified version of the reflection method will be adopted in more complicated cases as we will see in the next section.

4.2. Channel with a bump

4.2.1. Problem Description

In the flow field Figre 4.4, we observe advection flow (from left to right) through a channel with a bump. We will first discuss the setting, including geometry, a given analytical stream function and the induced velocity field. We will then add some diffusion effect to explore the particle tracking method. The height of the bump $h_b(x)$ and its derivative $h_b'(x)$ are



(a) Reflection method. **(b)** Diffusion suppression method.

Figure 4.3: Particle tracking simulation in a sinusoidal vortex.

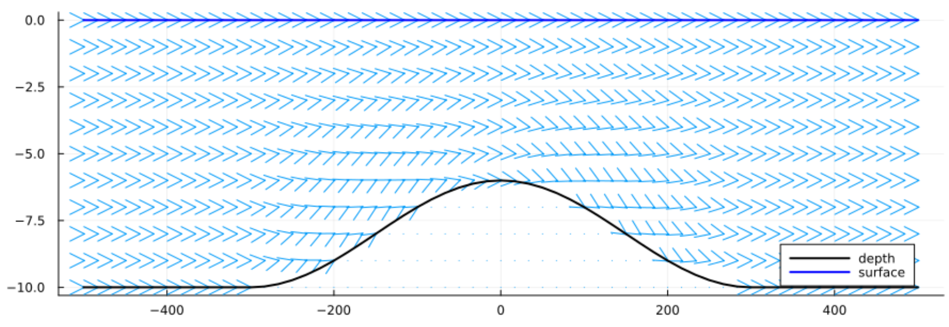


Figure 4.4: Channel with a bump. The vector field indicates the streamlines.

$$h_b(x) = \begin{cases} \frac{1+\cos\frac{\pi x}{l_b}}{2}, & -l_b < x < l_b, \\ 0, & \text{otherwise.} \end{cases}$$

$$h_b'(x) = \begin{cases} \frac{-\pi \sin\frac{\pi x}{l_b}}{2l_b}, & -l_b < x < l_b, \\ 0, & \text{otherwise} \end{cases}$$

The local depth of the water $d(x) = d_m - d_b \cdot h_b(x)$ and its derivative are

$$d(x) = \begin{cases} d_m - d_b \cdot \frac{1+\cos\frac{\pi x}{l_b}}{2}, & -l_b < x < l_b, \\ d_m, & \text{otherwise.} \end{cases}$$

$$d'(x) = \begin{cases} \frac{d_b \pi \sin\frac{\pi x}{l_b}}{2l_b}, & -l_b < x < l_b, \\ 0, & \text{otherwise.} \end{cases}$$

The planar stream function $\Psi(x, z)$ is

$$\Psi(x, z) = \begin{cases} 0, & z \geq 0, \\ -u_m \cdot d_m, & z < -d(x), \\ -u_m \cdot d_m \cdot z/d(x), & \text{otherwise.} \end{cases}$$

The horizontal velocity and vertical velocity function of the incompressible flow are

$$u(x, z) = -\frac{\partial \Psi}{\partial z} = \begin{cases} 0, & z \geq 0, \\ 0, & z < -d(x), \\ u_m \cdot d_m \,/d(x), & \text{otherwise.} \end{cases}$$

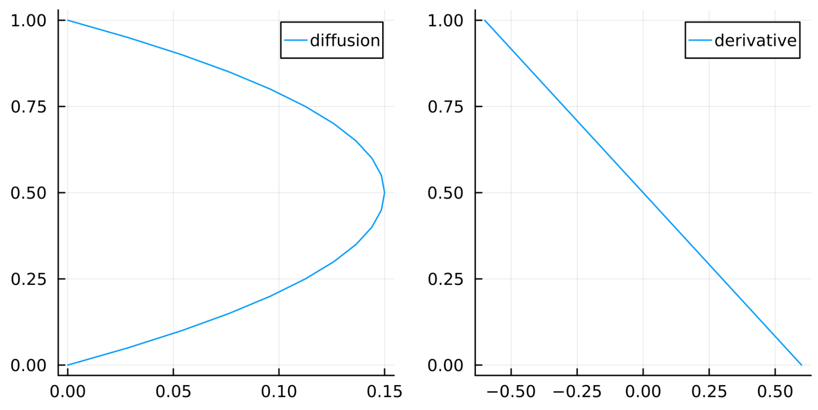


Figure 4.5: The profile of parabolic diffusion and its derivative.

$$w(x,z) = \frac{\partial \Psi}{\partial x} = \begin{cases} 0, & z \geq 0, \\ 0, & z < -d(x), \\ u_m \cdot d_m \cdot z/d^2(x) \cdot d'(x), & \text{otherwise.} \end{cases}$$

The computation domain is $x \in [-L, L], z \in [-d_m, 0]$. The parameters are $d_b = 4,\ l_b = 300,\ d_m = 10,\ u_m = 1,\ L = 500$. Then the actual horizontal and vertical velocity function are

$$u(x,z) = \begin{cases} 0, & z \geq 0, \\ 0, & z < -d(x), \\ \frac{10}{10-2[1+\cos(\pi x/300)]}, & \{-d(x) \leq z < 0\} \cap \{-300 < x < 300\}, \\ 1, & \text{otherwise.} \end{cases}$$

$$w(x,z) = \begin{cases} 0, & z \geq 0, \\ 0, & z < -d(x), \\ \frac{\pi}{60} \cdot \frac{z}{[4-\cos(\pi x/300)]^2} \cdot \sin(\frac{\pi x}{300}), & \{-d(x) \leq z < 0\} \cap \{-300 < x < 300\}, \\ 0, & \text{otherwise.} \end{cases}$$

There are two observations. As for the horizontal velocity, we see it is unity in the region where local depth is equal to the constant d_m (or “outside the bump region”). The vertical velocity is nontrivial only in the bump region, and its sign is positive for $x < 0$, the center of the channel in the horizontal direction, and vice versa. And, the maximal value of the vertical velocity is less than 0.06, which is really small.

4.2.2. Add the diffusion effect

It is interesting to add the diffusion effect into this test case, a case of laminar flow. In Section 3.4, the double parabola is introduced to represent the stratification of water body. A much simpler diffusion to represent the account for the diffusive transport (probably driven by turbulence in water) is used, with a parabola profile as shown in Figure 4.5. If particles are released without diffusion, it is very likely that there will be no problems because particles will follow the streamlines, which do not intersect with the boundary. To tackle this problem, one may use the recursion algorithm (reflection method). In reality, in the near-wall region there is strong mixing that will decelerate the particles there. Compared to the reflection method which does not involve any deceleration, the recursion algorithm will slow down the particles in the near-wall region in the vertical direction. We will use the channel with a bump case to test the efficacy of the recursion algorithm. To add the deterministic and random drift induced by turbulent diffusion, we consider the following parabola diffusion profile and its derivative is

$$K(z_{rel}) = K_{MAG} \cdot 6 z_{rel}(1 - z_{rel}), \quad \frac{dK}{dz} = K_{MAG} \cdot 6(1 - 2z_{rel}),$$

where z_{rel} is the relative z-coordinate ranging in [0,1] from the local riverbed to the river surface, C_{MAG} is the parameter to control the diffusion magnitude, and the induced displacement in dt is

$$w_{ind}dt = \frac{dK}{dz}dt + \sqrt{2K}dW_t. \tag{4.6}$$

To help compare the magnitude of the deterministic and random drift, set $dt = 1$, because this makes the the standard value of dW_t equal to dt.

4.2.3. How to compute the relative z-coordinate

To compute z_{rel}, one may use

$$z_{rel} = (z + d_m)/d_m,$$

where d_m is a constant. The '+' sign is used in the formula because $z < 0$. Although this result also gives rise to $z \in [0, 1]$, where $z = 0$ at the lower side of the rectangle domain and z=1 at the upper side of the rectangle domain, this does not make sense because we are now using a constant d_m instead of the local depth d. This construction is not completely meaningful in the bump region because it lets the riverbed cut off the diffusion profile at the middle. Thus, to compute z_{rel}, we adopt the following construction

$$z_{rel} = (z + d)/d,$$

where d is the local water depth. This construction is more meaningful in the sense that the diffusion effect is limited in the computation domain and plays the role of ”diffusion” as expected.

4.2.4. Results and discussion of the recursive particle update algorithm

As for the strategies to prevent particles from going out, there are three options. One can use the reflection method or the diffusion method mentioned in the previous section. And one can implement Algorithm 1, the recursive particle location update algorithm by Stijnen: the algorithm that divides the timestep by half, takes a new random number and thus weakens the strength of diffusion strength and ”controls” the direction of turbulent velocity. One problem of this algorithm is the degenerated computation efficiency when the recursion becomes too deep. It is suggested to limit the recursion depth manually. The efficacy of the recursive algorithm is demonstrated in Figure 4.6. Using this algorithm, the particles cannot cross the boundary. To probe into the question about what leads to particle leaving the boundary, Figure 4.7 and Table 4.1 were made. Clearly, the background vertical velocity is too small to have any substantial effect. We first pay our attention to the time between $t = 77$ to $t = 82$. When one looks at Figure 4.7, it is easy to conclude that particles leaving the boundary is because the derivative term $\frac{dk}{dz}dt$ is too large, as indicated by the circle in the plot. Nevertheless, this is misleading. When one takes a serious look at Table 4.1, one immediately finds out that the derivative term is small. The particle leaves the boundary there because the diffusion term $\sqrt{2k}$ multiplied to a relative large number sampled from $N\ (0, 1)$. The derivative term may also be too large to lead to overshoot, as demonstrated in the second time interval. In short, under the current setting where the derivative term $\frac{dk}{dz}dt$ and the diffusion term $\sqrt{2k}$ often have similar magnitude and thus their interaction leads to particles leaving the boundary occasionally.
To study other numerical setting, the values of K_{MAG} is modified. And the result is demonstrated in Figure 4.8. With increasing value of K_{MAG}, there are two observations. Firstly, the particle gets closer to the orange line, which is mainly driven by the derivative term. On the contrary, without any diffusion, the particle will follow the streamlines. Secondly, the motion of the particle becomes more violent, which can be explained by the combined effect of the two terms as mentioned above.

4.2.5. Implementation issues

Major modifications to the original code are briefly demonstrated here. Hopefully, this also clarifies some of the concepts mentioned before.

```
function run_simulation(d)
    # code blocks omitted
    t_targets = ...
    for t in t_targets
        s = @view p[:, i] # take a reference to the particle location
        #simulate!(f!,...) # old code
```

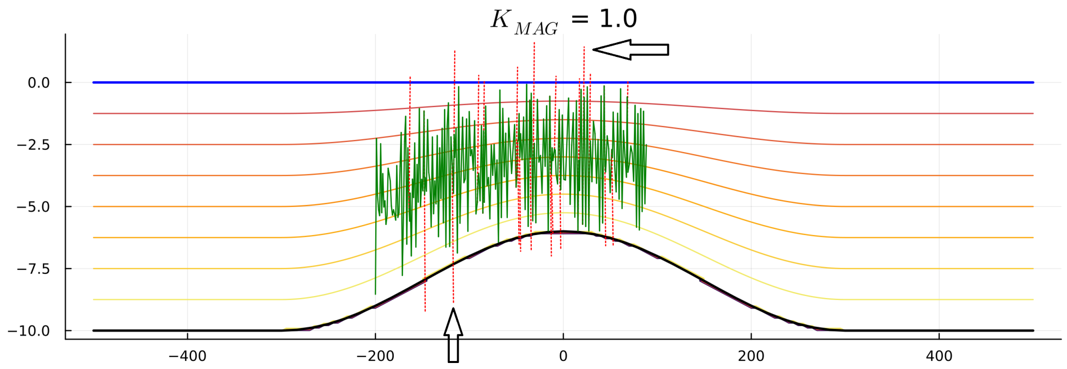


Figure 4.6: Single particle test - $K_{MAG} = 1.0$. The red dashed line indicates the rejected displacement. The two arrows indicate two intervals specified in Table 4.1.

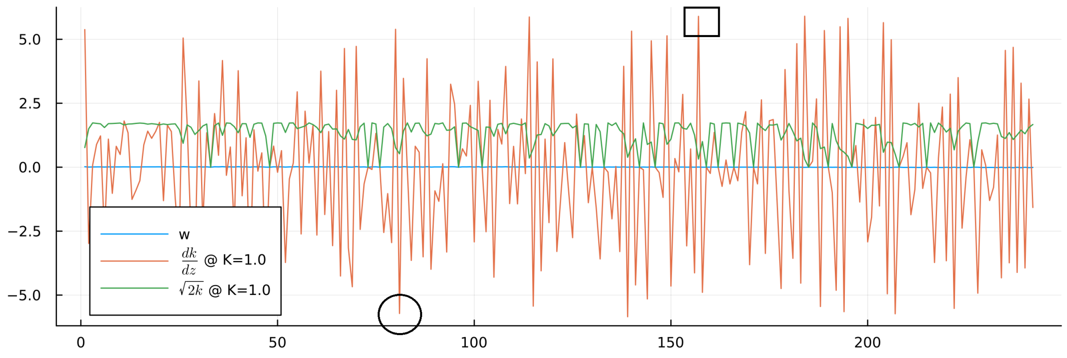


Figure 4.7: Velocity information of the single particle test - $K_{MAG} = 1.0$. The circle and the rectangle indicate two intervals specified in Table 4.1.

Table 4.1: Displacement calculation at two specified time intervals.

t	**x**	**z**	wdt	$\frac{dk}{dz}dt$	$\sqrt{2k}$
77.00	-118.280	-3.273	0.012	-0.654	1.722
78.00	-116.919	-8.935	0.000	0.000	0.000
77.50	-117.599	-3.613	0.013	-0.089	1.732
78.00	-116.917	-4.467	0.016	1.322	1.690
79.00	-115.551	1.277	0.000	0.000	0.000
78.50	-116.234	-2.100	0.008	-2.552	1.568
79.00	-115.550	-3.019	0.011	-1.032	1.706
80.00	-114.179	-1.848	0.007	-2.948	1.509
81.00	-112.803	-6.873	0.025	5.391	0.760
82.00	-111.422	-0.170	0.001	-5.718	0.525
165.00	18.858	-0.279	-0.000	-5.446	0.727
166.00	20.514	-5.715	-0.007	5.343	0.788
167.00	22.168	1.430	0.000	0.000	0.000
166.50	21.341	-2.526	-0.003	-0.990	1.708
167.00	22.168	-0.599	-0.001	-4.812	1.034

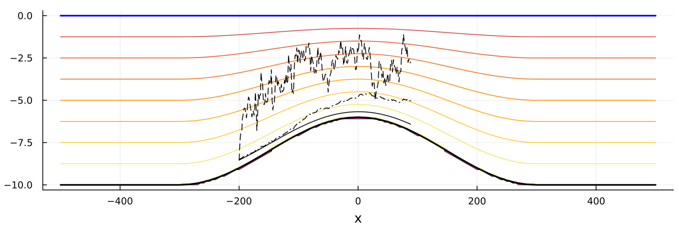


Figure 4.8: Comparison of values the single particle test @ $K_{MAG} = 0.1, 0.001, 0.0$ with dashed, dashdot and solid lines respectively.

```
            simulate!(f!,g!,...) # a new and parallel code
    end
end
function simulate!(f!,g!, ...)
    # code blocks omitted
    forward!(ds_determinitic, f!, ...)
    forward!(ds_random, g!, ...)

    # code blocks omitted to compute virtual displacement

    # check if the particle satisfy the following conditions
    # Five cases here for each particle
    # Case 1: after virtual displacement, the particle is still inside the
       domain
    ...
    # Case 2: after virtual displacement, the particle is outside the domain, so
        halve dt is needed
    ...
    # Case 3: the dt is too small to be halved, so let the particle cross the
       boundary
    ...
    # Case 4,5: the particle is already outside or the particle has not yet been
        released into the domain, so nothing needs updating
    ...
end
```

5 One-dimensional test cases

5.1. Introduction

The major motivation for this chapter is from the literature, where the author argues that it is viable to use the particle-tracking framework to simulate transport problems, but it would not be sufficient to use the Euler scheme to capture the stratification effect in the sea water due to large numerical diffusion of the numerical schemes. Now we would like to test this argument. The discussion here will follow that from Chapter 3.4 to Chapter 3.6.

Recall that we obtain the advection-diffusion equation in the Fokker-Plank form. We have found its stochastic counterpart and attempt to solve the SDE by the random walk algorithm through certain numerical schemes. The plan of this chapter is as follows. The two test cases, the pure diffusion problem and the advection-diffusion problem, are introduced. To verify my implementation, the numerical solutions of two simple test cases are compared with analytical solutions. After that, different numerical schemes are applied to the stratification problem.

5.2. Introduction and implementation of two simple test cases

Two simple test cases are described and implemented. The results are compared to potential analytical solutions to verify the stochastic methods to solve the transport equation. Note that the kernel density estimation techniques are also implemented here. These two test cases will be the solid basis for more useful test case.

5.2.1. Pure diffusion problem

Recall the advection-diffusion equation in the Fokker-Plank form

$$\frac{\partial C}{\partial t} + \nabla \cdot (v^{\text{drift}} C) = \nabla\nabla : (KC),$$

where

$$\boldsymbol{v}^{\textbf{drift}} = \boldsymbol{v}^{\dagger} + \boldsymbol{\nabla} \cdot \boldsymbol{K} \tag{5.1}$$

is the drift velocity. **In the literature, people often make no distinction between the true velocity v and the residual-mean velocity $v^{\dagger}$ in most cases,**

$$\boldsymbol{v}^{\dagger} = \boldsymbol{v} + \boldsymbol{v}^{*} = \boldsymbol{v} - \boldsymbol{\partial_i A_{ij}}, \tag{5.2}$$

we will also adopt the assumption here, so the one-dimensional transport problem can be written as

$$\frac{\partial C}{\partial t} + \partial_z((w + \frac{\partial k}{\partial z})C) = \partial_{zz}(kC),\ t \geq 0,\ 0 \leq z \leq h, \tag{5.3}$$

where the concentration distribution $C = C(z,t)$, the one-dimensional diffusion profile $k = k(z)$ and the velocity in the vertical direction is w. Firstly, now consider pure diffusion problem ($w = 0$) with all particles releasing at position $z = z_0$

$$\begin{aligned} \frac{\partial C}{\partial t} + \partial_z[\frac{\partial k}{\partial z}C] = \partial_{zz}(kC),\ t \geq 0,\ 0 \leq z \leq h, \\ [k\frac{\partial C}{\partial z}]_{z=0,h} = 0, \\ C(0,z) = \delta(z - z_0). \end{aligned} \tag{5.4}$$

And perform the following non-dimensionalizing operations

$$z' = \frac{z}{h}, \quad k' = \frac{k}{k_0}, \quad t' = \frac{t}{t_0}, \quad t_0 = \frac{h^2}{k_0}, \quad k_0 = \overline{k} = \frac{1}{h}\int_0^h k(z)dz. \tag{5.5}$$

and this gives rise to the following 1D non-dimensional pure diffusion problem

$$\begin{aligned} \frac{\partial C}{\partial t'} + \partial_{z'}(\frac{\partial k'}{\partial z'}C) = \partial_{z'z'}(k'C),\ t' \geq 0,\ 0 \leq z' \leq 1, \\ [k'\frac{\partial C}{\partial z'}]_{z'=0,1} = 0, \\ C(0,z') = \delta(z'h - z_0). \end{aligned} \tag{5.6}$$

First note that the the transport variable concentration C is left intact. Secondly, note that the timescale t_0 here can be really large in the application, because the water depth in the Rhine-Meuse Delta region is on the order of 10 meters, and the numerically computed k_0 is on the order of $10^{-3}m^2/s$.
Assume then nondimensional vertical diffusion profile is a parabola

$$k'(z') = 6z'(1 - z'), \quad \int_0^1 k'(z')dz' = 1, 0 \leq z' \leq 1. \tag{5.7}$$

According to [44], the pure diffusion problem has the analytical solution expressed as an infinite sum of the product of the Legendre polynomial $P_n(z)$ and the exponential function

$$C(t,z) = 1 + \sum_{n=1}^{\infty}(2n+1)P_n(2z-1)P_n(2z_0-1)e^{-6n(n+1)t}. \tag{5.8}$$

And if we further choose $z_0 = 0.5$, only the even-order terms are left in the infinite sum. And one can plot the approximate analytical solution of the evolution of the concentration distribution by the first serveral terms in the infinite series (Figure 5.1) Lastly, it is worthwhile to mention now that the stratification problem can be studied if we change the parabola diffusion profile to a "double-parabola" one. More details about this will be discussed later. **Assume temporarily the variables are nondimesional without the prime notation.** Follow the notation in Equation (3.12),

$$B(z) = k(z), \quad \sigma(z) = \sqrt{2k(z)}, \quad f(z) = \frac{dk}{dz}, \tag{5.9}$$

the governing equation of the moving particles

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t, X_{t_0} = X_0. \tag{5.10}$$

The random walk algorithm for this problem with Euler scheme is

$$\begin{aligned} \Delta X_t = \frac{dk}{dz}\Delta t + \sqrt{2k(z)}\Delta W_t, \\ X_0 = z_0/h. \end{aligned} \tag{5.11}$$

In the "implementation", the setting of time step, the number of used particles and the type of kernel method is

$$\Delta t = 3 \times 10^{-5}, \quad N = 10^3, 10^4, 10^5, \quad \text{kernel method } = \text{"box"}, \text{"Epanechnikov"}.$$

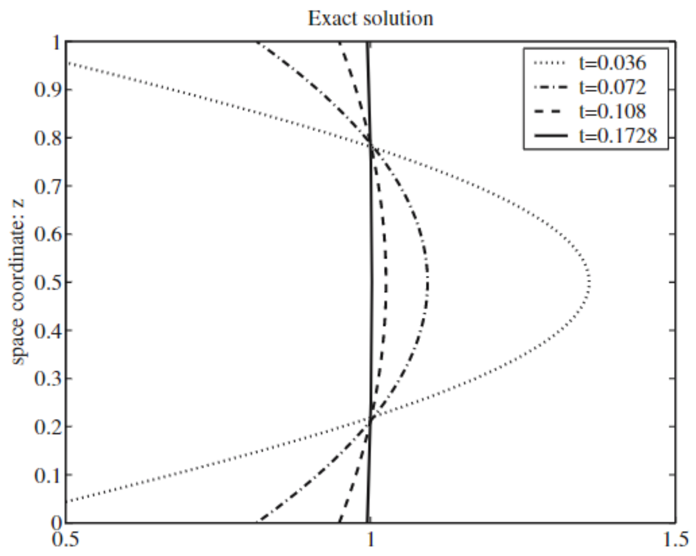


Figure 5.1: The concentration distribution for different moments of the simulation, retrieved from [44].

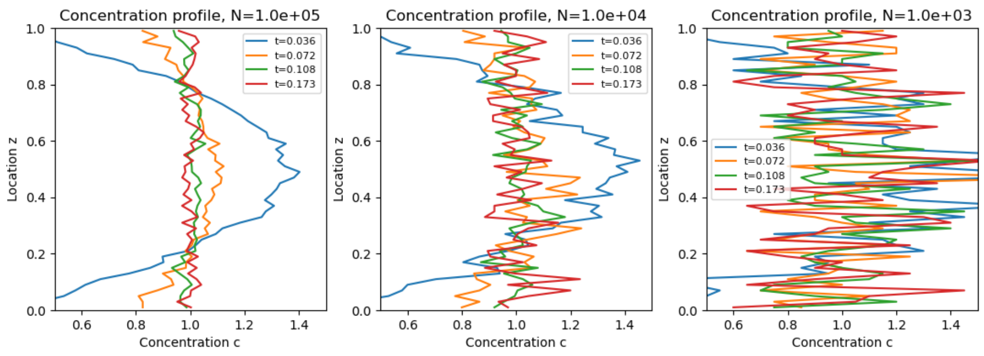


Figure 5.2: Numerical approximation of concentration distribution by the box kernel.

Recall from Chapter 3.5, we also need to transform the location of particles to a distribution function using the density estimation method. For this purpose, we will use a bandwidth value provided from [44]

$$\lambda_{opt} = K_{std} N^{-1/5}, \tag{5.12}$$

where K_{std} is the sample standard deviation of the location of particles. Only the numerical approximation of concentration distribution profiles are shown. In spite of available analytical solutions, more complicated error measure are not used because the error order of the stochastic methods is not our major focus and this test case is only used for verification of our implementation. From Figure 5.2 and Figure 5.3, both kernels obtain a numerical solution close to the analytical one if one uses enough number of particles. And more particles are needed if one uses a box kernel. Besides, if the reflection operation is not used at the grid points near the boundary, a sharp decrease of the concentration will occur there. As for the efficiency of applying box kernel and Epanechnikov or Gaussian kernel, the previous one only needs to sweep through all particles once while the latter two need it for each grid points. Note that 101 grid points are used in Figure 5.3 and the $100\times$ computation effort already becomes a heavy burden to a personal laptop.

5.2.2. Advection-diffusion problem

The following discussion follows [6] and [44]. Deleersnijder et al. [6] considered the transport in the upper mixed layer of the ocean across the pycnocline, as shown in Figure 5.4, where particles experience both diffusive and advective (sinking) effects. The upper boundary is the air-sea interface and the

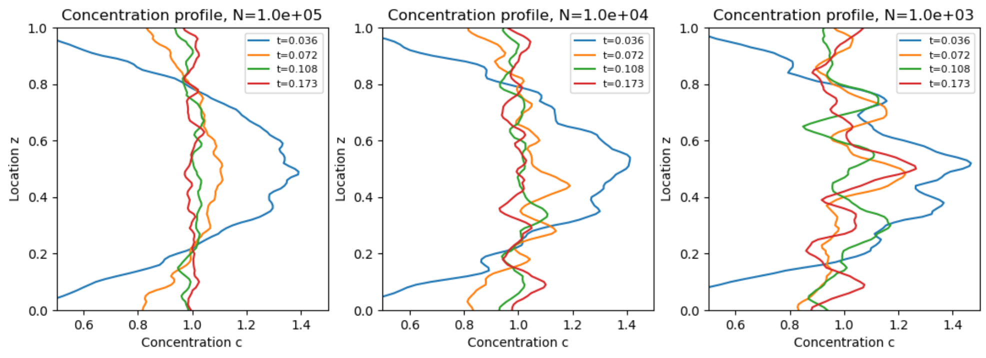


Figure 5.3: Numerical approximation of concentration distribution by the Epanechnikov kernel. Reflection is applied for the approximation at grid points near the boundary.

lower boundary is the pycnocline that is beneath the upper mixed layer in the ocean. Thus, different boundary conditions are applied. Besides, one can easily identify from Figure 5.4 that the advective effect is the major force that transport particles across the pycnocline. Assume that the sinking velocity w is a finite constant and construct the 1D non-dimensional advection-diffusion problem

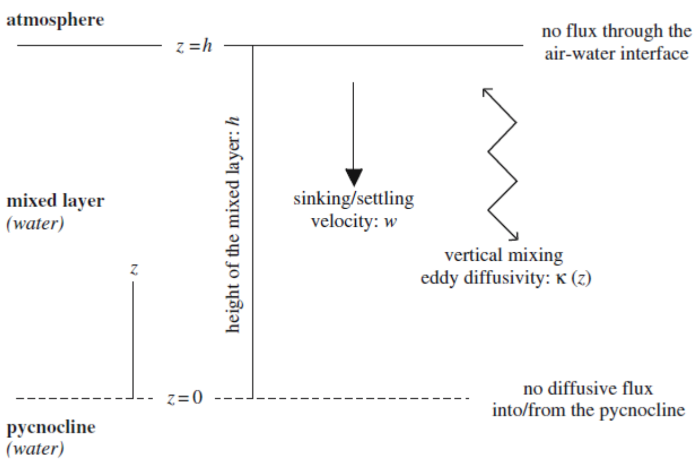

atmosphere
$z = h$
no flux through the air-water interface
height of the mixed layer: h
sinking/settling velocity: w
mixed layer (water)
vertical mixing eddy diffusivity: $\kappa(z)$
z
$z = 0$
no diffusive flux into/from the pycnocline
pycnocline (water)

Figure 5.4: Sinking-diffusion model: illustration of its geometry, parameters and boundary conditions, retrieved from [6].

$$\begin{aligned} \frac{\partial C}{\partial t} + \partial_z[(w + \frac{\partial k}{\partial z})C] = \partial_{zz}(kC),\ t \geq 0,\ 0 \leq z \leq h, \\ [k\frac{\partial C}{\partial z}]_{z=0} = 0, \\ [wC + k\frac{\partial C}{\partial z}]_{z=h} = 0, \\ C(0, z) = \delta(z - z_0). \end{aligned} \tag{5.13}$$

To see how the boundary conditions are defined, one can transform the advection-diffusion Equation (5.3) in Fokker-Plank form into its original form,

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial z}(wC + k\frac{\partial C}{\partial z}). \tag{5.14}$$

We immediately see that at the air-sea interface ($z = h$), no particle can penetrate through it, while at the pycnocline ($z = 0$) the particle can cross the boundary through advection but not through diffusion. By the following operation

$$z' = \frac{z}{h}, \quad k' = \frac{k}{k_0}, \quad t' = \frac{t}{t_0}, \quad Pe = \frac{wh}{k_0}, \quad t_0 = \frac{h}{w}, \quad k_0 = \overline{k} = \frac{1}{h}\int_0^h k(z)dz. \tag{5.15}$$

the problem becomes

$$\begin{aligned} \frac{\partial C}{\partial t'} + \partial_{z'}[(1+\frac{1}{\mathsf{Pe}}\frac{\partial k'}{\partial z'})C] = \partial_{z'z'}(\frac{k'}{\mathsf{Pe}}C),\ t' \geq 0,\ 0 \leq z' \leq 1. \\ [k'\frac{\partial C}{\partial z'}]_{z'=0} = 0, \\ [C + \frac{1}{\mathsf{Pe}}\frac{\partial C}{\partial z'}]_{z'=1} = 0, \\ C(0,z) = \delta(z-z_0). \end{aligned} \tag{5.16}$$

The Peclet number Pe is a non-dimensional parameter that computes the ratio of advective transport rate versus diffusive transport rate. If Peclet number is large, the problem is advection-dominant, while the problem is diffusion-dominant if the Peclet number is small. And note that the timescale corresponding to sinking velocity (advection) is different from the timescale in the pure diffusion problem. Although it is difficult to look for the analytical solution to this problem, it is possible to compute the residence time of particles in the adjoint from according to [6]. Specifically, for particles initially located at $z = z_0$, the nondimensional "average" time of a particle kept inside the upper mixed layer is called the residence time. **Assume that the following variables are nondimensional without the prime notation.** The analytical solution of the nondimensional residence time is given by generalized incomplete beta function

$$\theta(z) = z + (\frac{z}{1-z})^{\mu} B_{1-z}(1+\mu, 1-\mu), 0 \leq z \leq 1, \tag{5.17}$$

where $B_{1-z}(1+\mu, 1-\mu)$ is a generalized incomplete beta function,

$$B_{1-z}(1+\mu,1-\mu) = \int_0^{1-z} \sigma^{\mu}(1-\sigma)^{-\mu} d\sigma, \quad \mu = \mathsf{Pe}/6. \tag{5.18}$$

We again find out the equivalent stochastic differential equation. Follow the notation in Equation (3.12),

$$B(z) = \frac{k(z)}{\mathsf{Pe}}, \quad \sigma(z) = \sqrt{2\frac{k(z)}{\mathsf{Pe}}}, \quad f(z) = 1 + \frac{1}{\mathsf{Pe}}\frac{dk}{dz}, \tag{5.19}$$

the governing equation of the moving particles

$$dX_t = f(X_t,t)dt + \sigma(X_t,t)dW_t, X_{t_0} = X_0. \tag{5.20}$$

The random walk algorithm for this problem with Euler scheme is

$$\begin{aligned} \Delta X_t = 1 + \frac{1}{\mathsf{Pe}}\frac{dk}{dz}\Delta t + \sqrt{2\frac{k(z)}{\mathsf{Pe}}}\Delta W_t, \\ X_0 = z_0/h. \end{aligned} \tag{5.21}$$

Additionally, it is worth mentioning that the random walk algorithm for the advection-diffusion problem in [44] is distinct from Equation (5.21) by a factor of w because the algorithm in the paper is intended for the questions without nondimensional operation. To compute the nondimensional residence time using the result from that algorith, one needs to divide the true residence time by a factor of the time parameter $t_0 = h/w$. This will also give similar results as those in the nondimensionalized setting.

5.3. Intro to the stratification case

The discussion before proves that it is feasible to use the particle-tracking framework to study mass transport. Furthermore, in [14] Graewe argued that the most prevalent Euler scheme in the implementation of the particle-tracking method is not enough mainly because of its large numerical diffusion. This happens When the upstream river flow is weak, the estuary is often weakly mixed and thus stratification between the light riverwater and the dense seawater is established.

To convince their audience, Graewe et al. continued their discussion of the one-dimensional pure diffusion problem in Equation (5.4) with a socalled "double-parabola" diffusion profile in Figure 3.2. The particles are released at the location that is away from the upper boundary with the distance of one-fourth of the height of the water column.

In this setting with Euler scheme, even if the time step is small as 3×10^{-5}, the pycnocline does not act as a barrier and particles simply cross the pycnocline. However, with the Milstein scheme that is only strongly accurate in $O(\Delta t)$, the particles hardly cross the pycnocline. This phenomenon is clearly demonstrated in Figure 5.5. And actually, with the first-order Milstein scheme the pycnocline continues to act as a barrier till at least one non-dimensional timescale, as shown in Figure 5.7, while with the Euler scheme, the water column is almost mixed mixed after one-fifth of the non-dimensional timescale.

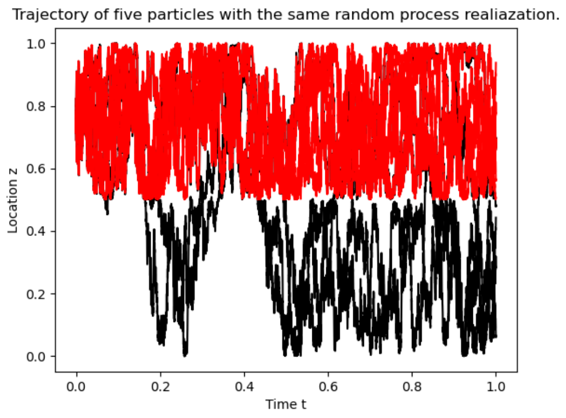


Figure 5.5: Comparison on using Euler and the 1st-order Milstein scheme to compute trajectories of five particles with the same random process realiazation.

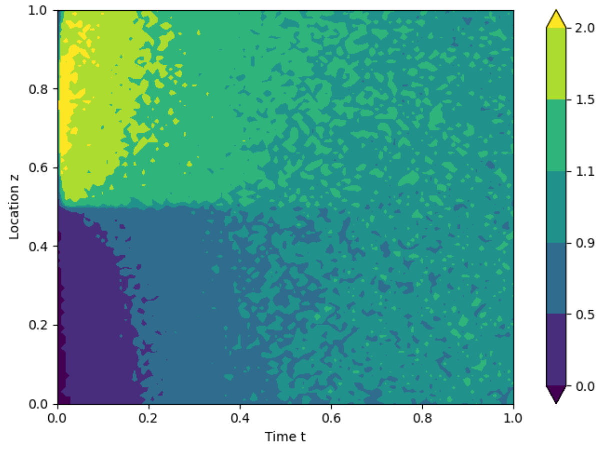


Figure 5.6: Evolution of concentration field via the Euler scheme.

In [15] Graewe concluded that the stratification phenomenon will be contaminated by the Euler scheme in a theoretical setting. Admittedly, the result supports the argument by Graewe about using high-order numerical schemes in a theoretical test case. However, there are two distinction between the theoretical test case and reality. Firstly, the used diffusion profile in our model seems different from the one obtained from the experiment (Figure 5.8). The diffusion in the upper and lower layer of the water column is asymmetric, which may lead to different physics. Secondly, it is necessary to calculate the diffusion timescale t_0 by grouping up the following variables and compare it with true simulation time,

$$\text{height } h, \text{ diffusion strength } \bar{k}, \text{ timestep } dt. \tag{5.22}$$

The diffusion timescale can be constructed as

$$t_0 = \frac{h^2}{\bar{k}}. \tag{5.23}$$

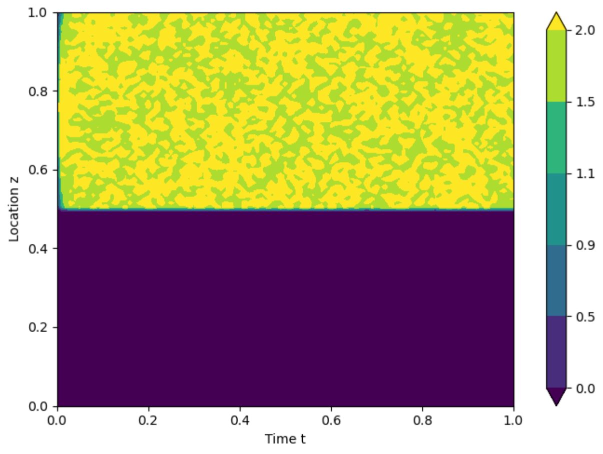


Figure 5.7: Evolution of concentration field via the 1st-order Milstein scheme.

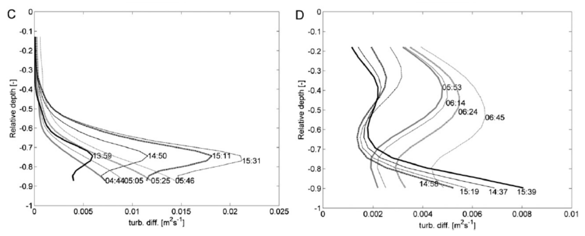


Figure 5.8: Turbulent diffusivity profiles at selected times during ebb and flood at two stations estimated from Reynolds shear stress profiles (de Nijs et al. 2010) and with the Munk and Anderson (1948) relationship for the turbulent Prandtl–Schmidt number, retrieved from [31]

If the true simulation time is much smaller than the diffusion time scale, we may guess other physic mechanism than diffusion is dominating the current transport process.

6 Simulation of random processes

We see in the last chapter that the most simple Euler scheme may fail to represent the stratification phenomenon. Though this is a theoretical case to large extent and realistic diffusion profile has a less strict requirement on the diffusion error of the numerical scheme, it will be beneficial for us to look at some higher-order schemes.
This chapter will introduce the stochastic modelling by exploring the geometric Brownian motion (GBM). The modelling of such a simple case as GBM will shed light on some common difficulties of stochastic modelling. Additionally, to apply the particle-tracking method based on the stochastic modelling, the one-dimensional diffusion test case will be re-examined for studying the convergence order of different SDE schemes. We have witnessed nice convergence results in [14] by Gräwe, but when we switch to a more reasonable error measure, the convergence order of SDE falls far from theoretical values. This result indicates the existence of other limiting factors of the particle-tracking framework, even in such a simple test case. For example, it is not easy to reconstruct the concentration scalar field at the boundary accurately.

6.1. the GBM test case

To do stochastic simulation, it would be beneficial to start from a simple case, the GBM. The stochastic process X_t that is GBM satisfies the following stochastic differential equation (SDE),

$$dX_t = \mu X_t dt + \sigma X_t dW_t, \tag{6.1}$$

where W_t is a Wiener process whose increments are independent identical Gaussian distribution with mean 0 and variance dt, μ is the drift constant and the σ is the volatility constant. The stochastic process has an analytical solution

$$X_t = X_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right). \tag{6.2}$$

where X_0 is the initial condition of the random process. The logarithm of the solution is a superposition of a deterministic drift along and a Brownian motion. Additionally, GBM has a nice property in that it is possible to calculate the analytical expression of its expectation

$$E[X_t] = X_0 e^{\mu t}, \tag{6.3}$$

which is independent of σ. The explicit expression for X_t and $E[X_t]$ will be useful when considering the order of convergence. To calculate convergence order, it will be required to define appropriate error measure.
From probability theory, weak convergence have several equivalent definitions. Equation 3.18 gives one definition. Other equivalent formulations include the convergence of mean as well as the expectation. To make use of the explicit expression of the expectation of the Brownian motion, we would like to

formulate the error as difference between empirical expectation and the true expectation. The strong error $E_s(t)$ and weak error $E_w(t)$ of the GBM modelling at time t will be defined as

$$E_s(t) = \frac{1}{N_p}\sum_{i=1}^{N_p} |\hat{X}_t^i - X_0 e^{\mu t}|, \tag{6.4}$$

$$E_w(t) = |\frac{1}{N_p}\sum_{i=1}^{N_p} \hat{X}_t^i \quad - X_0 e^{\mu t}|, \tag{6.5}$$

where $\hat{X}_t^i$ represents the location of the i-th particle at time t and N_p represents the number of particles. When comparing the numerical solution to the analytical solution of an SDE and calculating the strong error of numerical schemes, it's crucial to ensure that the generated trajectories match. This involves making sure that the random elements in the SDE numerical schemes are consistent. Specifically, the only random component in these schemes is the sequence of Brownian increments. To achieve that, the same sequence of random numbers is needed, each of which is generated from a normally distributed variable.
There is one problem that hinders people from drawing consistent trajectories. Often in stochastic simulation, the sampling frequency of drawing Brownian increments (and calculating the analytical solution) is often orders of magnitude larger than that of calculating the numerical solution. To secure the same realization in calculating a numerical solution on a coarser time grid, it will be required to add up all the occurring Brownian increments on the fine time grid before the next time instant drawing from the coarse time grid. The algorithm to calculate the trajectory by the Euler method on an evenly-spaced coarse time grid is illustrated in Algorithm 2. Literature [17] by Highham and Desmond gived a detailed

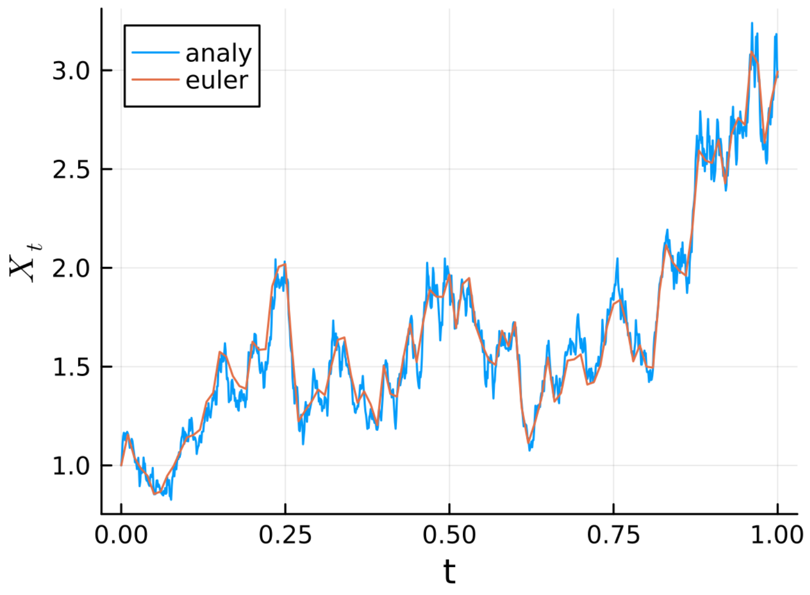


Figure 6.1: Comparison between the analytical solution and a consistent trajectory. The size of the time step in calculating the trajectory is ten times as large as the sampling frequency of the analytical solution.

discussion on how to simulate some simple stochastic processes, for instance, the GBM. In particular, they also discussed how to obtain a consistent trajectory given a sequence of Brownian increments. Furthermore, they also listed the error sources in the stochastic simulation. Namely

- sampling error: due to approximating the expectation by a sampled mean.
- random number bias: dominated by the structure of the random number generator.
- round-off error: inevitable and relevant to the floating-point number type used.

These are innate errors in stochastic simulation, so their orders of magnitude needs to be limited to have a reliable simulation result.

Algorithm 2 Euler SDE numerical solver on a coarser time grid adapted from [17]

```
1: procedure Euler method($X_0, dW_{seq}, gridratio$)    ▷ $dW_{seq}$ is the given random number sequence, and the Euler scheme is $X_{k+1} = X_{t_k} + f(t_k, X_k)dt + \sigma(t_k, X_k)dW_k$
2:     $N = Length(dW_{seq})$    ▷ The number of steps of the analytical solution.
3:     $Neuler = Int(N/gridratio)$    ▷ The number of steps of the numerical solution by Euler method. Often $N$ and $gridratio$ are chosen such that their division results in an integer.
4:     for $k = 0 : Neuler - 1$ do
5:         $startindex = (k-1) * gridratio$
6:         $endindex = startindex + gridratio$
7:         $dW \leftarrow 0$
8:         while $startindex \leq i \leq endindex$ do
9:             $dW \leftarrow dW + dW_{seq}[i]$
10:        $X_{k+1} = X_k + f(t_k, X_k) * dt + \sigma(t_k, X_k) * dW$
       return $X_0, X_1, \ldots, X_N$
```

6.1.1. Summary of numerical schemes

Consider a one-dimensional SDE

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t, \tag{6.6}$$

where f and g are smooth functions and ΔW_t is the increment of the Wiener process. To solve the above the SDE numerically, we introduce four numerical schemes, which are demonstrated by updating X_{t_n} to $X_{t_{n+1}}$, along with ΔW_{t_n} as the increment of the Wiener process in $[t_n, t_n+\Delta t]$. The Euler scheme is

$$X_{t_{n+1}} = X_{t_n} + f(X_{t_n}, t)\Delta t + \sigma(X_{t_n}, t)\Delta W_{t_n}. \tag{6.7}$$

The 1st-order Milstein scheme (M1) is

$$X_{t_{n+1}} = X_{t_n} + f(X_{t_n}, t_n)\Delta t + \sigma(X_{t_n}, t_n)\Delta W_{t_n} + \frac{1}{2}\sigma(X_{t_n}, t_n)\frac{\partial \sigma}{\partial x}(X_{t_n}, t_n)(\Delta W_{t_n}^2 - \Delta t). \tag{6.8}$$

The Heun shceme is

$$\begin{aligned} X_{t_{n+1}}^p &= X_{t_n} + f(X_{t_n}, t_n)\Delta t + \sigma(X_{t_n}, t_n)\Delta W_{t_n} \\ X_{t_{n+1}} &= X_{t_n} + \frac{1}{2}[f(X_{t_n}, t_n) + f(X_{t_{n+1}}^p, t_n)]\Delta t + \frac{1}{2}[\sigma(X_{t_n}, t_n) + \sigma(X_{t_{n+1}}^p, t_n)]\Delta W_{t_n}. \end{aligned} \tag{6.9}$$

The fourth-order Runge-Kutta scheme (RK4) is

$$\begin{aligned} K_0 &= f(X_{t_n}, t_n), & G_0 &= \sigma(X_{t_n}, t_n), \\ X_{t_n}^{(0)} &= X_{t_n} + \frac{1}{2}K_0\Delta t + \frac{1}{2}G_0\Delta W_n, & & \\ K_1 &= f(X_{t_n}^{(0)}, t_n + \frac{1}{2}\Delta t), & G_1 &= \sigma(X_{t_n}^{(0)}, t_n + \frac{1}{2}\Delta t), \\ X_{t_n}^{(1)} &= X_{t_n} + \frac{1}{2}K_1\Delta t + \frac{1}{2}G_1\Delta W_n, & & \\ K_2 &= f(X_{t_n}^{(1)}, t_n + \frac{1}{2}\Delta t), & G_2 &= \sigma(X_{t_n}^{(1)}, t_n + \frac{1}{2}\Delta t), \\ X_{t_n}^{(2)} &= X_{t_n} + \frac{1}{2}K_2\Delta t + \frac{1}{2}G_2\Delta W_n, & & \\ K_3 &= f(X_{t_n}^{(2)}, t_n + \Delta t), & G_3 &= \sigma(X_{t_n}^{(2)}, t_n + \Delta t), \end{aligned}$$

$$X_{t_{n+1}} = X_{t_n} + \frac{1}{6}(K_0 + 2K_1 + 2K_2 + K_3)\Delta t + \frac{1}{6}(G_0 + 2G_1 + 2G_2 + G_3)\Delta W_n. \tag{6.10}$$

The properties of the numerical schemes to are summarized in the Table 6.1. To compare numerical schemes that can only be used for Ito or Stratonovich SDE, the following trick is applied. From Equation (A.23), we can transform the original Stratonovich SDE to an Ito one where the old parameter and

numerical scheme	SDE type	strong error	weak error
Euler	Ito	0.5	1
Milstein1	Ito	1	1
Heun	Stratonovich	1	1
RK4	Statonovich	2	4

Table 6.1: A summary of numerical schemes and their order of convergence, adapted from Chapter 3.3 in [21] by Stijnen.

convergence type	strong	weak
T_{end}	1	1
number of realization, N_p	10^6	5×10^7
number of timestep, $N_T = T_{end}/\Delta t$	2^3-2^9	2^3-2^9
gridratio	1	1

Table 6.2: Parameters used to calculate the convergence order of GBM.

new parameter (with a tilde) are related by

$$\tilde{\mu} = \mu + 0.5\sigma^2, \tag{6.11}$$
$$\tilde{\sigma} = \sigma. \tag{6.12}$$

This enables us to find an expression for the analytical solution of the Stratonovich SDE using the Equation (6.2) and Equation (6.3).
Lastly, the stability of SDE numerical schemes is beyond the scope of this work.

6.1.2. Modelling setting

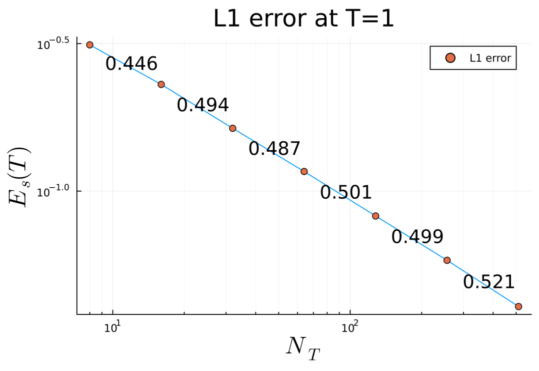
The parameters of the Brownian motion are

$$\mu = 0.5, \quad \sigma = 1. \tag{6.13}$$

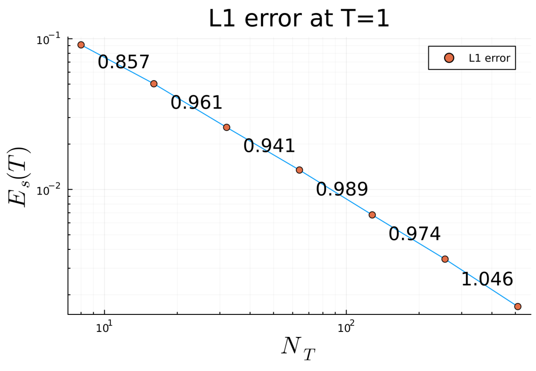
The parameters for calculating strong and weak convergence results are summarized in Table 6.2. Grid ratio is chosen to be one for simplicity.

6.1.3. Results and discussion for the GBM case

Note that the analytical solution is calculated by Equation 6.2 and Equation 6.3. Under the current numerical setting, the results from Figure 6.2 to Figure 6.5 are nice because most of the convergence order are almost as expected. The strong convergence order is slightly impaired for large Δt and has a nice fit to theoretical value for small Δt. This may be attributed to unstability of the numerical algorithm using a large time step and the interaction of a large timestep and other factors. As for weak convergence results, although the RK4 scheme has the smallest error, the error stagnates at the order of 10^{-4}, thus contaminating the convergence order. As for the other three schemes, the convergence order remains closes to theoretical values except between $\Delta t = 1/2^7$ and $\Delta t = 1/2^8$, which might be due to the roundoff error.
To try as much as possible to reduce the effect of the sampling error — the difference between the true expectation and the empirical expectation, a relatively large sample is used after many attempts. From the results, it is observed that strong convergence is less sensitive to the change of samples. To achieve convergence in the same range of N_T, much more realizations are needed, indicating that the weak convergence is more easily affected by sampling error. There is a necessity to use a large sample to obtain a weak convergence result, which is also mentioned in [17].
In spite of using a large sample, the convergence order may deteriorate as seen in the RK4 case. Consequently, there may be some limiting factors other than the parameters we tried to change here. A possible candidate of the limiting factor may still be the lack of enough particles. Other limiting factors include the round-off error and the random number bias inherent in the adopted Mersenne Twister random number generator. However, to probe into this issue further will be beyond the scope of this thesis. And the convergence orders are as expected. In short, in the GBM test case, from the perspective of implementation, we not only see the delicacy to draw a consistent trajectory or

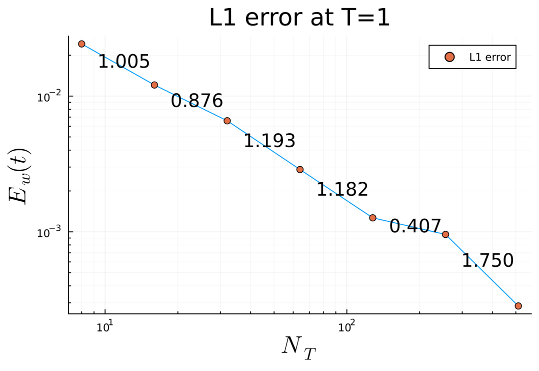


(a) Strong convergence of EM scheme

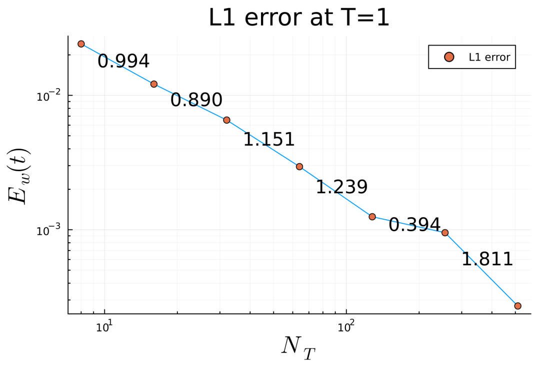


(b) Strong convergence of M1 scheme

Figure 6.2: Strong convergence of two Ito schemes - averaged over 100000 realizations: the horizontal axis represents the number of spacing in the time grid, $\Delta t = 1/N_T$.

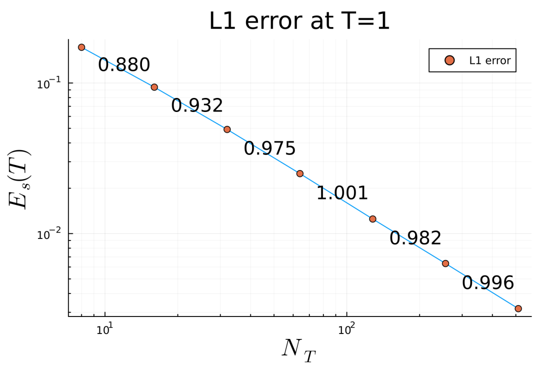


(a) Weak convergence of EM scheme

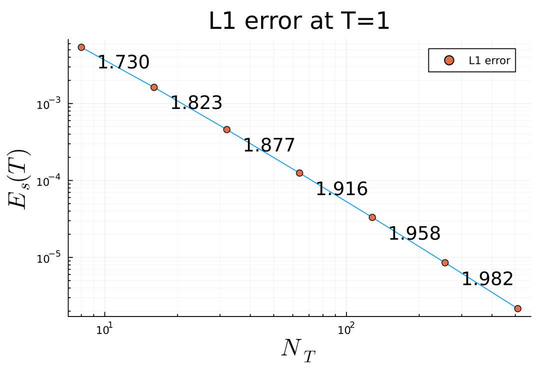


(b) Weak convergence of M1 scheme

Figure 6.3: Weak convergence of two Ito schemes - averaged over 50000000 realizations: the horizontal axis represents the number of spacing in the time grid, $\Delta t = 1/N_T$.

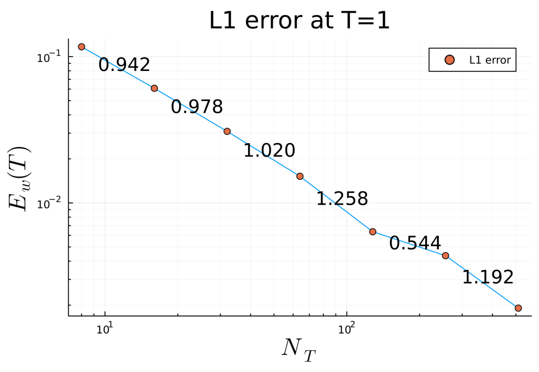


(a) Strong convergence of Heun scheme with Stratonovich-type interpretation

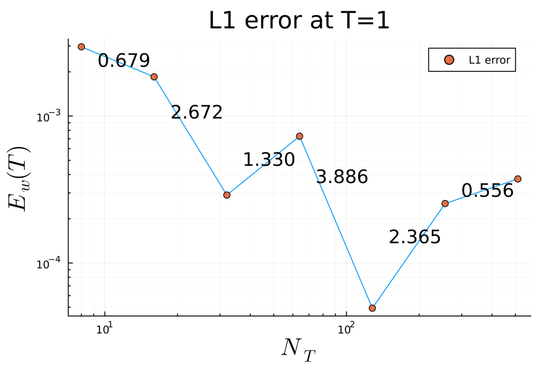


(b) Strong convergence of RK4 scheme with Stratonovich-type interpretation

Figure 6.4: Strong convergence of two Stratonovich schemes - averaged over 100000 realizations: the horizontal axis represents the number of spacings in the time grid, $\Delta t = 1/N_T$.



(a) Weak convergence of Heun scheme



(b) Weak convergence of RK4 scheme

Figure 6.5: Weak convergence of two Stratonovich schemes - averaged over 50000000 realizations: the horizontal axis represents the number of spacings in the time grid, $\Delta t = 1/N_T$.

realization but we also see in our numerical experiments the existence of some limiting factors that make the numerical solution from converging to the analytical solution slower than expected or even destroy the convergence property. These problems prevail in the stochastic simulation. As we will see in the 1D diffusion test case in the next section, as the model becomes more complicated, there will be more elements adding into the "limiting factor" part.

6.2. 1D pure diffusion

6.2.1. Problem description

To model the vertical transport physics, it is suggested to start from a simple case. In reality, we are not given an explicit expression of the diffusion profile, but the diffusion value at grid points. Thus, it is natural to approximate the derivatives with appropriate numerical methods. Moreover, how to interpolate a diffusion function will also affect the final result. The number of sample points and the interpolation splines that connect sample points will influence the final result. In [14] Graewe discussed the influence of these two factors. However, in our implementation, only an interpolated diffusion function will be available from the data of a Delft3D FM model, so those will be beyond the scope in my discussion. To mimic the setting of a given interpolated diffusion function, we assume that we have a continuous diffusion function while the derivatives will be approximated by a second-order numerical scheme in space.

The only derivative to be approximated is the first-order one. We will use the 2nd order central difference method and a second-order one-sided difference method if the central difference method doesn't work near the boundary. The schemes and their leading error terms are listed

$$f'(z) = \frac{f(z+h) - f(z-h)}{2h} - \frac{h^2}{6} f'''(z), \tag{6.14}$$

and

$$f'(z) = \frac{-3/2 f(z) + 2f(z+h) - 1/2 f(z+2h)}{h} + \frac{h^2}{3} f'''(z), \tag{6.15}$$

and

$$f'(z) = \frac{3/2 f(z) - 2f(z-h) + 1/2 f(z-2h)}{h} + \frac{h^2}{3} f'''(z). \tag{6.16}$$

6.2.2. Equivalence of numerical schemes

The Ito SDE for the diffusion equation is

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t = \frac{dk(X_t)}{dx} dt + \sqrt{2k(X_t)} dW_t. \tag{6.17}$$

According to the literature, the corresponding Stratonovich SDE is

$$dX_t = f(X_t, t)dt - \frac{1}{2}\sigma(X_t, t)\frac{\partial \sigma}{\partial x}(X_t, t)dt + \sigma(X_t, t)dW_t = \frac{1}{2}\frac{dk(X_t)}{dx} dt + \sqrt{2k(X_t)} dW_t. \tag{6.18}$$

After the transformation, the coefficient of the f function is changed while the σ function remains unchanged. This means the implementation for the two types of SDEs will not vary too much. Last but not least, the numerical schemes used are summarized in Chapter 6.1.1.

6.2.3. Implementation of the "reflection method" in density estimation

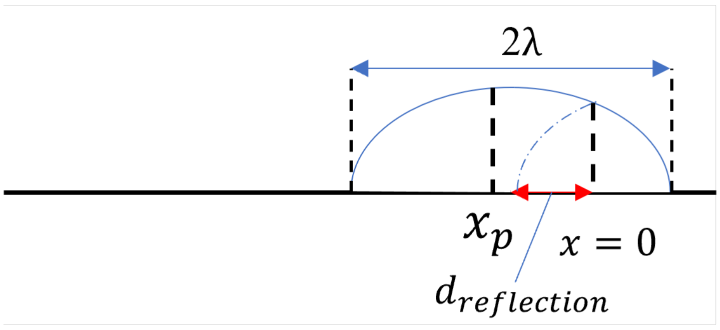


Figure 6.6: Explanation of the implementation for the "reflection method" in kernel density estimation - the demo plots the situation in the computation near the right boundary $x = 0$. A particle at x_p induces a density with span two times the kernel bandwidth. When reflection method is applied, the density induced right to the boundary will be reflected and affects the density calculation in $[0, d_{reflection}]$.

As explained in Chapter 3.5, it is required to be careful about particles near the boundary, because part of their induced densities can lay outside the computation domain, as demonstrated in Figure 6.6. The reflection method can be implemented using the following algorithm. Assume now we are interested in estimating the density at x near $x = 0$ induced by a particle located at x_p.

- Add the density at x directly induced by the particle located at x_p.
- Check if $x < \lambda$. If so, then continue the following computation.
- Compute $d_{reflection} = \lambda - (x_p - 0)$.
- Add the reflection part if $x < d_{relfection}$.

The above idea is realized in the following algorithm. It would necessary to discuss the situation near $x = 1$ separately, though the idea is similar. For example, one needs to replace $x < \lambda$ by $1 - x < \lambda$. The implementation for non-flat boundary and higher-dimensional problems will be more difficult. Thus, the reflection method may not be practical in applications.

Algorithm 3 Reflection method near $x = 0$ in one-dimensional density estimation

```
1: procedure Reflection method($c, x_p, \lambda,$ Epa_Kernel)
2:     $p \leftarrow 0$
3:     $p =$ Epa_Kernel$((x - x_p)/\lambda)$
4:     $d_{reflection} \leftarrow \lambda - (x_p - 0)$
5:     if then$x < d_reflection$
6:         $u_{reflection} = (x = x_p)/\lambda$
7:         $p =$ Epa_Kernel$(u_{reflection})$
       return p
```

6.2.4. Error measure construction

Remember that the pure diffusion has an explicit analytical solution expressed in Equation (6.23), which we will denote as $C_A(z,t)$ now. The numerical solution is denoted as $C_P(z,t)$. In [14], the error of the numerical solution with respect to the analytical solution is expressed as

$$\epsilon_{lit} = \sqrt{\frac{1}{N_T}\sum_{n=1}^{N_T}\frac{1}{N_z}\sum_{n=1}^{N_z}[C_A(z_i,t_n) - C_P(z_i,t_n)]^2}, \tag{6.19}$$

where N_T is the number of observation time instants $t_1, t_2, \ldots, t_{N_T}$ and N_z is the number of grid points in the one-dimensional domain. This construction seems to lead to some nice convergence results that matches to the numerical schemes used in the computation, as shown in the Fig.2 in [14]. However, from my perspective, this averaging is an over-simplification of the original diffusion problem, because it does not consider the variation of modelling error in space and time. Use to kernel method in Equation (3.25), the numerical solution can be written as

$$C_P(t,z) = \frac{1}{N\lambda}\sum_{n=1}^{N} K\left(\frac{X_t^{(n)} - z}{\lambda}\right). \tag{6.20}$$

To calculate the convergence order, we need to fit the numerical solution into the form of Equation (3.18), thus, we can take

$$h(X_t^{(n)}, z) = \frac{1}{\lambda}K\left(\frac{X_t^{(n)} - z}{\lambda}\right), \hat{E}h(X_t^{(n)}, z) = C_P(t,z), \tag{6.21}$$

and the expectation operator E is approximated by the sample mean $\hat{E}$. Naturally, L1-error instead of L2-error is adopted and the convergence order at different location might need to be considered separately. Thus, the error should be defined as the following,

$$\epsilon(z_i,t_j) = |C_A(z_i,t_j) - C_P(z_i,t_j)| = \left|C_A(z_i,t_j) - \frac{1}{N\lambda}\sum_{n=1}^{N} K\left(\frac{X_{t_j}^{(n)} - z_i}{\lambda}\right)\right|, \tag{6.22}$$

where $X_{t_j}^{(n)}$ represents location of the n-particle in ensemble at time t_j, i is the spacial grid index ranging from 1 to N_z and n is the temporal grid index ranging from 1 to N_T. Compared to ϵ, ϵ_{lit} performs spatial and temporal averaging as well as adopts the L2 error. In the following, we will account for the influences of these factors. The analytical solution of 1D diffusion problem is written again here

$$C(t,z) = 1 + \sum_{n=1}^{\infty}(2n+1)P_n(2z-1)P_n(2z_0-1)e^{-6n(n+1)t}. \tag{6.23}$$

The involved Legendre polynomials can be calculated by scientific computing mathematical library. Note the terms in this infinite series decayed exponentially. The infinite series is truncated until the next term is comparable to machine precision.

parameter	values
numerical schemes	euler, (M1, Heun, RK4)
number of particles N_p	10^8
bandwidth λ	$\lambda = K_{std} N_p^{-1/5}$
timestep Δt	$3 \times 10^{-3}, 10^{-3}, 3 \times 10^{-4}, 10^{-4}, 3 \times 10^{-5}, 10^{-5}, 3 \times 10^{-6}$
observation times T_{obs}	$0.036 : 0.036 : 0.216$
kernel	Epanechnikov
z-grid, number of grid N_z	$0.0 : 0.02 : 1.0,\ N_z = 51$
spatial gridsize for approximating spatial derivative h	2.0×10^{-5}

Table 6.3: A list of parameters used in the 1D diffusion example - K_{std} is the sample standard deviation of particles' location at a specific observation time.

6.2.5. Parameter setting

According to [44] and [42], the optimal bandwidth is

$$\lambda \sim O(N_p^{-\frac{1}{d+4}}),\ \text{where } d \text{ is the dimension of the problem, here } d = 1, \tag{6.24}$$

and the error due to the use of the kernel estimator (density estimation technique)

$$\epsilon_{de} \sim O(\lambda),\ \text{near the boundary,} \quad \epsilon_{de} \sim O(\lambda^2),\ \text{in the interior of domain.} \tag{6.25}$$

As for $d = 1$, both the required bandwidth and the resulting density estimation error decreases slowly with increasing particle number N_p. Since 10^8 particles is already a large burden for a personal computer with an RTX3050 GPU, this implies that we may not be able to observe convergence because we are not able to use enough particles. Other parameters are summarized in Table 6.3. As for numerical schemes, all four schemes were first used to plot the convergence order plot. We will soon see that the convergence order deteriorates when the error becomes small. For the sake of this, only the result of the euler scheme is used to examine the effect of spatial averaging.

6.2.6. Results and Discussion

Error averaging in space and at different times

In [14], it shows that the four numerical schemes mentioned above have almost the same weak convergence order as the theoretical value using error measure ϵ_{lit}. Inspired by the nice convergence result and the error measure ϵ we constructed, we first paid attention to the effect of different numerical schemes. In view of the error measure construction, we decided to calculate **error averaging in space and at different times**. For this purpose, Figure 6.7 is plotted. At different time instants, the orders of magnitude of errors are similar. And the convergence orders also exhibit similar trends — the convergence order is near one when the error (or the time step) is relatively large. However, when the error is small, the convergence order is quickly contaminated. The result that when Δt is relatively large, Euler, Heun and M1 scheme having a weak convergence of order one matches theoretical values. As for the RK4 scheme, in spite of the best accuracy among all schemes, its convergence order is far from the theoretical value four even at relatively large Δt. There is a general trend in Fig 6.7 that the error stagnates when the error drops to scale of 0.01. The fact of error stagnating implies the existence limiting factors that hinder the decrease of the error. This may be attributed to the sampling error (the number of particles is not enough), because the sampling error decreases really slow with N_p (proportional to $N_p^{-0.2}$). However, a very large sample (10^8) is already used for this application with the help of GPU acceleration. A larger sample is not realistic on a personal laptop. Though what the limiting factor is in the example remains inconclusive, Table 6.4 lists the potential errors in this example as well as their orders of magnitude.

Error at different locations and times

The convergence results using a L1 error by us does not look nice when comparing to the convergence result in [14] using a L2 error. By comparing the formulas of two different error measures, we guessed that it would be interesting to **examine the effect of spatial averaging**. For this purpose, Figure 6.8 and Figure 6.9 were plotted. In general, we observe that as the size of timestep gets smaller, the

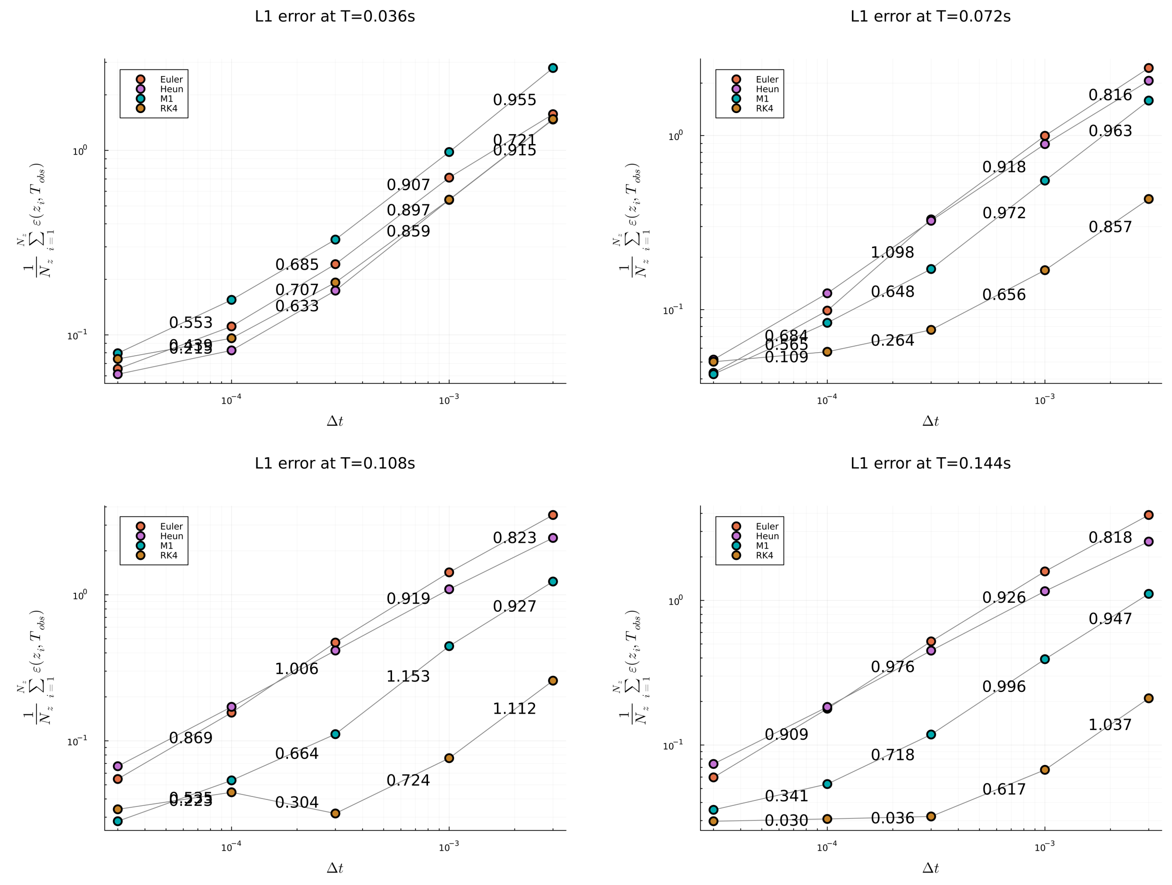


Figure 6.7: Weak convergence of 10^8 particles 1D diffusion demo with L1 averaging in space - the tangent indicates the convergence order, the tangent of the Heun method is not plotted for clarity.

estimated concentration approaches the analytical solution and the error becomes smaller. However, when looking at the distribution of the error, we found out that the boundary point has a significant contribution to the global error. To explore the magnitude of error at boundary points, Figure 6.10 is plotted. Each of the two boundary grid points accounts for 10-20% of the global error. There are 51 points in total, which implies the remaining 49 grid points account for 60-80% of the total error. Simple calculation indicates the magnitude of the local error at the boundary points is almost an order large than that at inner points. This fact probably contributes to the nice convergence result by Graewe in [14], because when L2 error is adopted, the boundary error is likely to dominate the global error. The above analysis weakens the strength of arguments made in by Graewe in [14]. Graewe argued that using high-order SDE numerical schemes is crucial so as to prevent the flow physics from destroying by the strong numerical error in the Euler scheme. Even though this argument is reasonable in certain setting, like the 1D stratified water column, the global error in 1D diffusion example cannot provide much support to this argument, because the global error is dominated by the local error at the boundary. The boundary error is strongly related to the choice of kernel and the technique in the boundary region to construct a consistent density estimation. Speak of practical side, the framework in this report to calculate substance transport using the particle-tracking method may have difficulties to perform density estimation at the boundary, either because there is a large error there or because too complicated methods are needed. Note that the reflection method discussed in Chapter 3.5 is already not easy too implement if the surface of the boundary is not flat. Last but not least, it may be interesting to see how the local error varied at different time instants. And Figure 6.11 was plotted for this purpose. There are several observations though the plots look messy from the first sight. Firstly, the convergence order seems to be unity for error at $z = 0$ except for the first time instant. This again supports above explanation of the nice convergence results by Graewe in [14], and the conclusion that taking spatial and temporal averaging at the same time to compute a convergence order might be a over-simplification. Secondly,

Error term	Order of magnitude
approximating the derivative term dk/dx in the SDE	$O((\Delta x)^k)$, k dependent on the numerical scheme
numerically solving the SDE	$O((\Delta t)^k)$, k dependent on error type and schemes
sampling error, according to [44]	proportional to $N_p^{-1/5}$
concentration estimation, according to [44]	$O(\lambda^k)$, k dependent on if it is close to the boundary
bias of random number generator	dependent on the generator
round-off error	machine precision, 1×10^{-16}

Table 6.4: A summary of error terms in stochastic simulation of the 1D diffusion test case - Δx is the spatial grid size, N_p is the number of particles, λ is the bandwidth of the kernel. Here are some additional explanation. The error order of the spatial derivative term depends on the choice of the finite-difference scheme. The sampling error refers to the error in using a finite number of realization to calculate the expectation. Concentration estimation applies the technique of density estimation, which is a research domain of mathematicians. Interested readers are recommended to the introductory text [42]. As for its error order, this depends on the choice of kernel, size of bandwidth and other factors. For example, in this example, the reflection method is used at the boundary to obtain a consistent estimation. However, the price is that the order of the local density estimation error drops from $O(\lambda^2)$ to $O(\lambda)$ at the boundary.

the error at different location can fluctuate as the size of timestep becomes smaller. This observation may be attributed to the innate randomness of this simulation that interacts with the discretization operations made before. Last but not least, in general, we see the local convergence has a trend that converges spatially with order one, if the fluctuation parts are ignored.

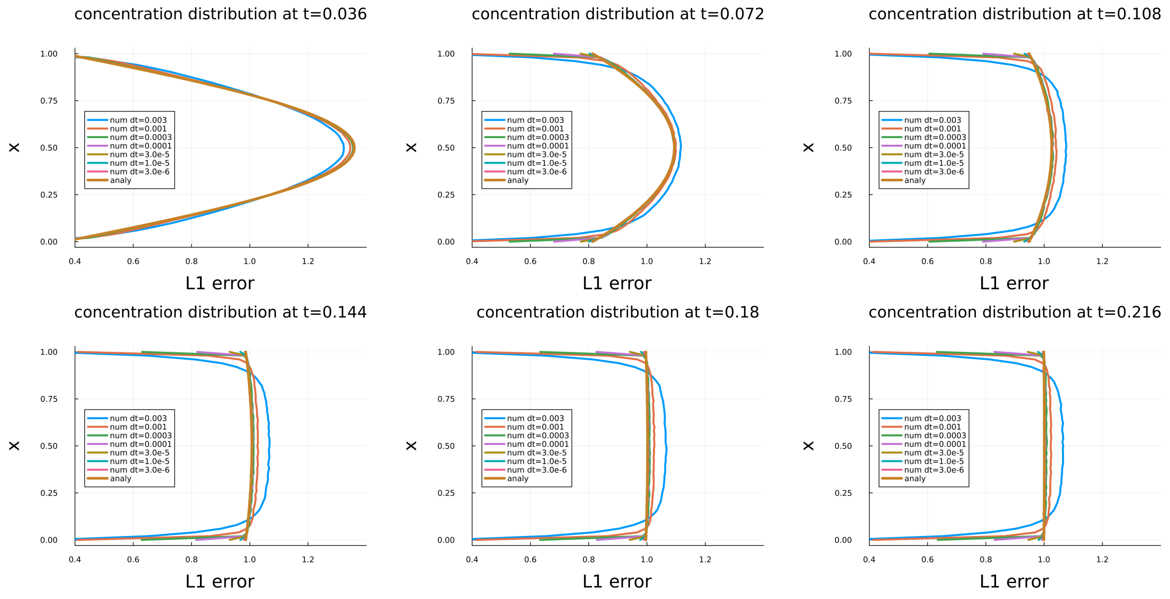


Figure 6.8: Estimated concentration of the 1D diffusion case using the Euler scheme.

Conclusions

We observe that with the newly defined L1 error measure, the convergence results deteriorate when Δt becomes smaller. This implies the existence of certain limiting factor. It is guessed that the number of used particles is the limiting factor, because the relevant sampling error decreases very slowly with increased number of particles. So the **first take-home message** will be a large particles sample is needed to have accurate results in the particle-tracking framework, especially if one is using the framework to model substance transport. Furthermore, we observe that the local L1 error is really large near the boundary for relatively large Δt. The local error decreases with decreased Δt. This observation reflects the complicated interaction between timestep size and numerical schemes on representing the vertical physics. In this case, a smaller timestep does represent the physics near the boundary well, unlike in the stratification case, only the use of a higher order SDE scheme can represent the stratification well, so the **second take-home message** is that when people use the particle-tracking framework, they need to conduct enough parameter analysis on the choice of numerical schemes and timestep

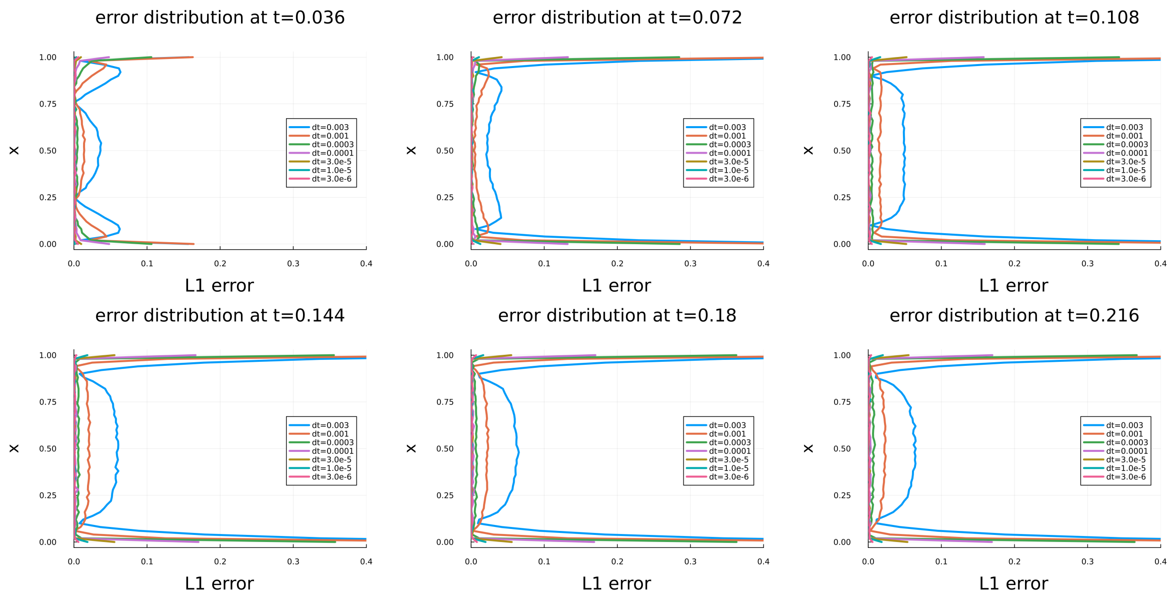


Figure 6.9: Estimated concentration of the 1D diffusion case using the Euler scheme.

size, in order to check whether the vertical transport physics are reflected reasonably.
Besides, there are other factors that affect the results of the particle-tracking simulation and the estimated concentration from that. It would be nice to make a summary of all involving factors for future reference for other test cases using the particle-trackinig framework.

1. Choice of the SDE numerical scheme.
2. Number of particles used.
3. Data of diffusion (the resolution of the data will affect the accuracy of calculating the derivatives).
4. Numerical schemes in approximating derivatives.
5. Density estimation method(including the choice of kernel and bandwidth).
6. Inherent error of a scientific computing program.
    - Inherent error in the random number generator.
    - Floating-point error.

Given the fact the theoretically simple 1D diffusion test case already gives out much trouble when it concerns the analytical solution and convergence order, it will not be beneficial to pay attention to analytical solution in the application. Thus, instead of constructing an estimated concentration that may already be available from a finite-volume method simulation, it is suggested to pay more attention to the statistics of particles that account for their qualitative behaviours. For instance, what is the proportion of particles that go across the pycnocline? This will be a good indicator of the strength of mixing. Furthermore, this provides another perspective of an existing simulation.

6.3. A extended discussion on the density estimation method

This section attempts to explore deeper into the construction of density estimation by calculating the error of this method “analytically”. The error in this simulation will be examined much more carefully. The concentration will be treated as probability density. For this purpose, the following notation is introduced. $p(x)$ represents the analytical solution. For particle of index $n,\ n = 1, 2, \ldots, N$ located at $X^{(n)}$, it induces an unit-impulse distribution

$$\hat{p}^{(n)}(x) = \delta(x - X^{(n)}). \tag{6.26}$$

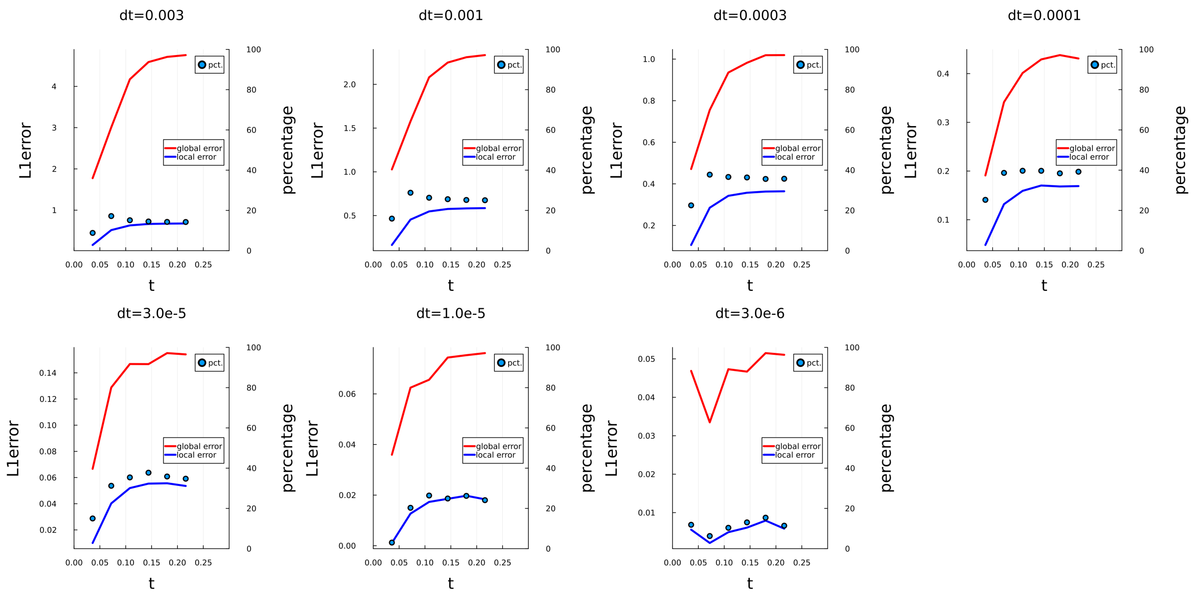


Figure 6.10: Local error at a boundary point versus global error and their ratio at different time instants

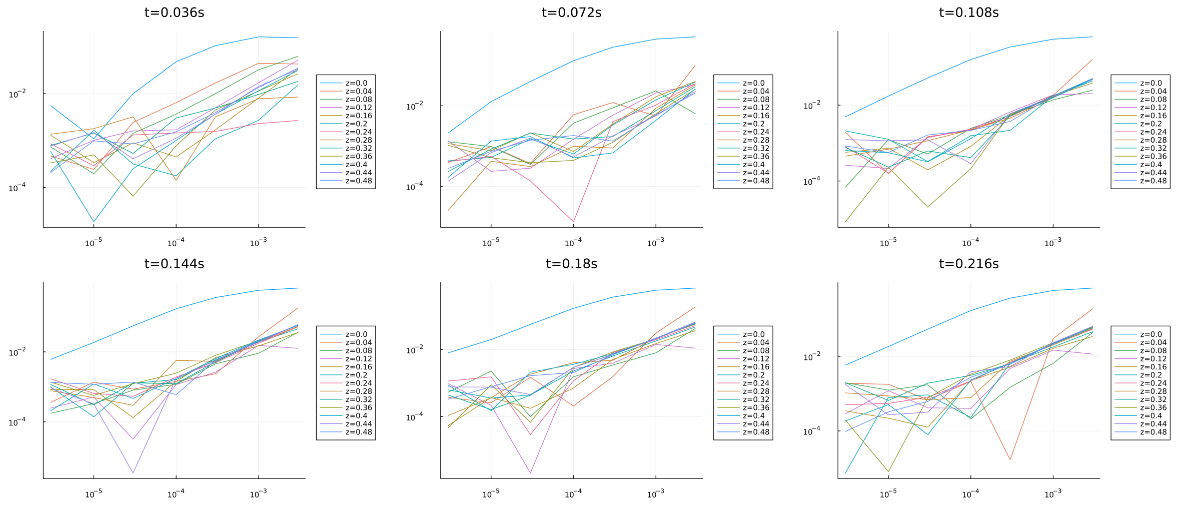


Figure 6.11: Local error at different time instants.

Note that both x and $X^{(n)}$ range in $[0,1]$. Intuitively speaking, one may apply one may guess when the number of particles approaches infinity, the expectation of $\hat{p}^{(n)}(x)$ converges to the desired density.

$$E[\hat{p}^{(n)}(x)] \xrightarrow{?} p(x), \quad \text{where } E[\hat{p}^{(n)}(x)] = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \delta(x - X^{(n)}). \tag{6.27}$$

In fact, the dirac function cannot be evaluated at the location of impulse, which makes the above expectation not converge pointwise. Thus, seek the weak convergence in the integral sense (cumulated density function)

$$\tilde{F}(u) = E[\int_{\Omega} \hat{p}^{(n)}(x)g(u-x)dx] = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \int_0^1 \delta(x - X^{(n)})g(u-x)dx = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{X^{(n)}\le u}. \tag{6.28}$$

where $g(x) = \mathbb{1}_{x\in[0,1]}$. The last term is a natural estimation of the sample distribution. And the sample distribution should converge the original distribution when the sample size approaches infinity. It needs to be argued about why the first equality holds. We guess that the proof of the weak convergence of this distribution requires us to construct an infinite sequence of distribution estimation with different particle number n and prove its convergence by certain theorem, for example, the Levy's continuity theorem along with the help of the characteristic function. However, the proof will be beyond the scope of the report. In conclusion, although $E[\hat{p}^{(n)}(x)]$ does not converge pointwise to the analytical solution, $\tilde{F}(u)$ **is likely to** converge to $F(u) = \int_0^u p(x)dx$. Indeed, take arbitrary infinitesimal $\Delta > 0$,

$$\tilde{F}(u+\Delta) - \tilde{F}(u) = p(u)\Delta = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{u \le X^{(n)} \le u+\Delta}. \tag{6.29}$$

The last equality holds because the last terms also represents the percentage of particles occurring in $[u, u+\Delta]$. Given that only a limited number of particles are available for practical purposes and one is still interested in calculating a density distribution, one can use the density estimation method, where each particle induces a density function around it of mass $1/N$. The shape of each induced density function is determined by the choice of the kernel function. In other words, the density estimation method, which involves the operation of calculating convolutions, can be regarded as a smoothing technique in the case of finite number of particles. To perform the convolution correctly, it is essential to know the support of the kernel. The Epanechnikov kernel is defined to have a compact support on $[-1, 1]$. In the current setting with bandwidth λ, it is expected that the kernel function has a compact support on $[-\lambda, \lambda]$. This transformation of variable gives out a constant in the following formula

$$\int_{-1}^{1} K(u)du = 1 = \int_{-\lambda}^{\lambda} K(w)\frac{dw}{\lambda}, \quad \text{where } w = \lambda u. \tag{6.30}$$

Remember that the convolution formula of function f and g

$$(f \circ g)(u) = \int_{-\infty}^{\infty} f(x)g(u-x)dx. \tag{6.31}$$

In the current setting, each particle (not near the boundary) located at $\hat{X}^{(n)}$ induced at density in $[\hat{X}^{(n)} - \lambda, \hat{X}^{(n)} + \lambda]$. Then limiting the influence of each particle in a 2λ-width region naturally gives the following convolution result

$$\tilde{p}(u) = \int_{\Omega} \frac{1}{N} \sum_{n=1}^{N} \delta(x - X^{(n)}) \frac{1}{\lambda} K(\frac{u-x}{\lambda}) dx = \frac{1}{N\lambda} \sum_{n=1}^{N} K(\frac{X^{(n)} - u}{\lambda}), \tag{6.32}$$

where Ω represents the integration region (in this case $[0, 1]$), K is the kernel functions such as box, Gaussian and Epanechnikov, and λ is the bandwidth of the kernel. The emerging constant $1/\lambda$ is due to the transformation of variable. The importance of this factor can be immediately seen if one integrates the result of Equation (6.32) and finds the integral with value one. And the last equality holds because of the property of the Dirac function. Now $\tilde{p}(u)$ is used to approximate the analytical solution $p(u)$. And denote

$$\tilde{p}(u) = K \circ E[\hat{p}^{(n)}(x)]. \tag{6.33}$$

Now, let us take a closer look at the difference between the analytical density p and the estimated density $\tilde{p}$

$$p - \tilde{p} = (p - K \circ p) + (K \circ p - K \circ E\hat{p}^{(n)}). \tag{6.34}$$

The first term represents the error solely induced by the density estimation method, which we now denote as the filtering error, while the second term can be interpreted as the error from the stochastic particle-tracking simulation, which will be denoted as nonfiltering error. To calculate these two terms, it is necessary to calculate $K \circ p$

$$(K \circ p)(u,t) = \int_0^1 C_A(z,t) \frac{1}{\lambda} K\left(\frac{u-z}{\lambda}\right) dz, \tag{6.35}$$

which is always positive and depends on location and time, so as bandwidth λ. It is possible to calculate it with the help of symbol math calculator *Sympy* and the numerical quadrature algorithm in the same package. It is possible to calculate the numerical integral with high accuracy because the integrand is

only a polynomial, where each monomial is the product of a Legendre polynomial and an exponentially decaying factor. It takes around around 50 monomials to obtain machine precision thanks to the decaying factor. And the one-dimensional Epanechnikov kernel $K(x) = \frac{3}{4}(1-x^2)|_{x^2 \leq 1}$.
To compute Equation (6.35), one needs to determine the bandwidth λ at each observation time, calculated using different time steps dt. Notably, the bandwidth is proportional to $N^{-0.2}$, where N is number of particles. Given that the largest time step is already quite small, the variation in bandwidth calculated using different dt is not significant. This also passes to the filtered solution. Looking at the filtered solution, it's evident that, for most grid points, the solution calculated with different dt only varies minimally. This can be observed in Figure 6.12 and Figure 6.13.
Here are several other observations on the filtering results. Firstly, it is observed that the filtering error is significant at the two boundary points. The is because filtering operation (Equation (6.35)) is performed in a finite domain, so there are some boundary effects, making the filtering operation there not consistent, i.e., the filtering solution will not approach the original solution when the bandwidth approaches zeros. And due to the current choice of bandwidth and the grid size, no other grid point is affected. This result also confirms the necessity to use the reflection method at the boundary to make up for the current density estimation method. The second observation comes from the Figure 6.14. The nonfiltering error not only includes the error from solving the SDEs numerically, but also includes the error at the boundary brought by nonconsistent filtering operation. We observe that the non-filterint error decreases with timestep dt quickly, while the filtering error at the boundary is nondecreasing. These two factors act together in forming the weird deflection at the boundary in Figure 6.14. Besides, it can be seen that when the error in solving SDEs is large, there is a large deficiency in calculating the solution near the boundary when using the particle-tracking framework. In other words, when dt is large, the particles seem to gather in the middle. This is probably because the deterministic term in the SDE is dominant compared to the random term. Furthermore, it is also observed that the nonfiltering error brought by solving SDEs numerically is way larger than the filtering error, which is barely noticeable in the previous figures. This suggests high order SDE numerical schemes and small timestep size are very useful in the particle-tracking framework, though to some extent requires more detailed discussion.

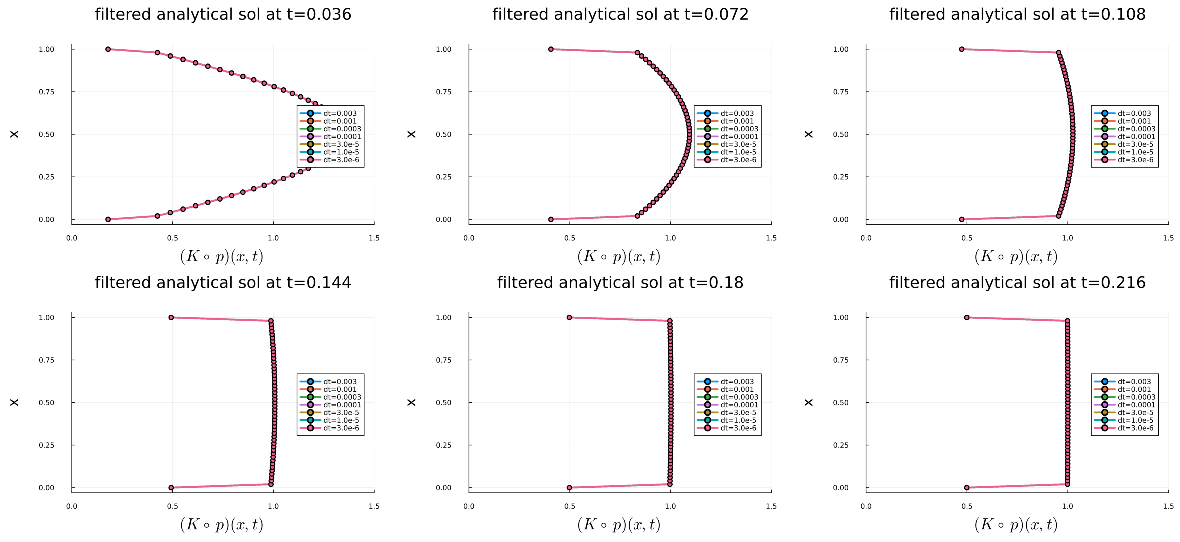


Figure 6.12: Filtered analytical solution distribution.

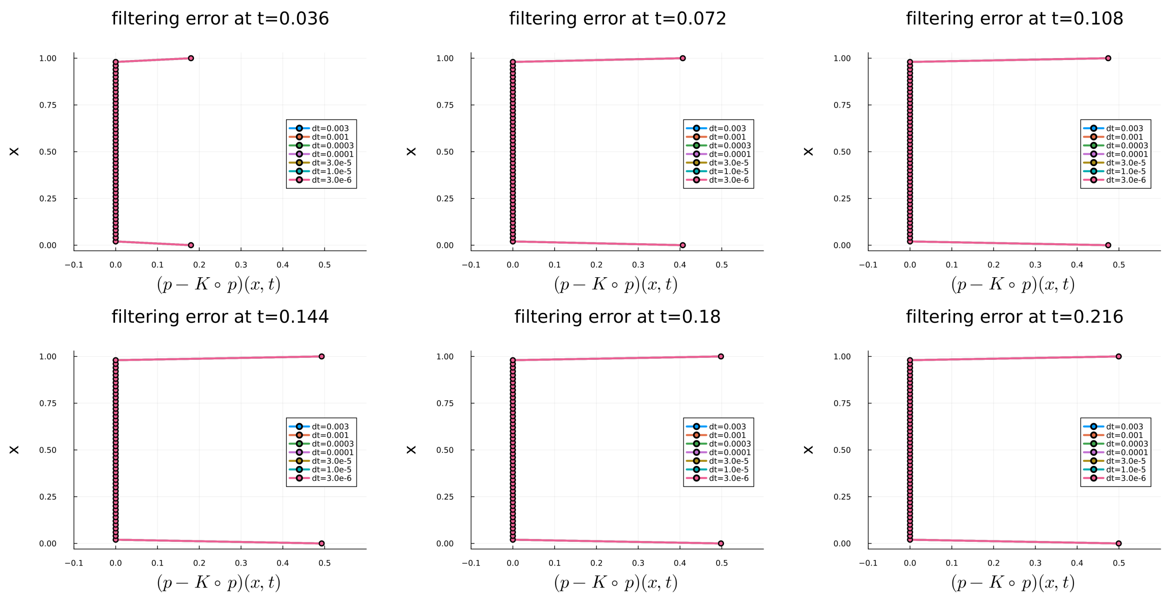


Figure 6.13: Filtering Error distribution.

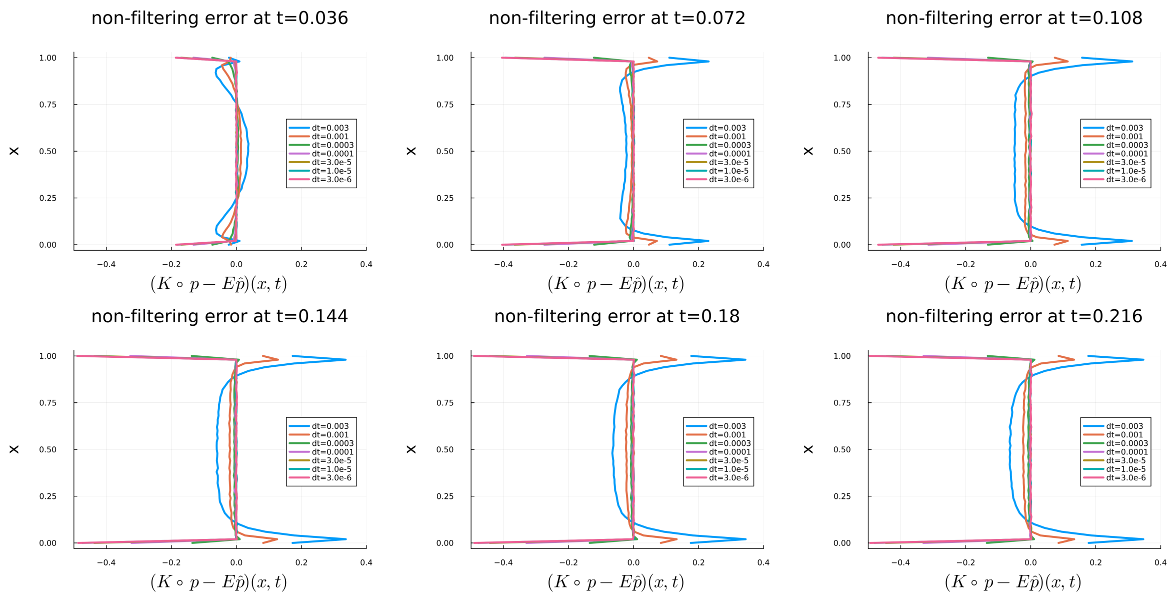


Figure 6.14: Nonfiltering error distribution: this error includes both the error coming from solving SDEs numerically and the error at the boundary due to the in consistent filtering operation.

7 Implementation of the 2D tidal flume process

The 2D tidal flume test case can be considered to be a prototype for realistic salinity intrusion, because this includes most significant elements, tidal periods, stratification and a damped vertical diffusion. Compared to early chapters, this chapter will pay special attention to the implementation of the flux boundary conditions.

7.1. Problem description

After conducting particle-tracking simulation on simple test cases and exploring the numerical scheme that may play a role there, a more interesting test case awaits exploration. This test case mimics the periodic tides intruding into a water channel, where the model is a 2DV one, one that has been averaged across the cross-section of the river.
This data of the test case is probably drawn from a thin and long water tank that is used for experiments about tidal effect. Thus, the water tank can be seemed as a semi-enclosed one with a schematic plot shown in Figure 7.2. In the water tank, periodic forcing condition is placed on the left that drives salt water come in and go out of the domain periodically like tides. In the experiment of the first few cycles, salt water is advected from the open boundary on the left. After several cycles, more saline water stays inside the rectangular domain and gradually reaches a dynamic balance of inward and outward salt flux. The parameters for this model can be found in Table 7.1. Note that the simulation time is about 10 cycles of tidal periods. A dynamic simulation of this transport phenomena can be seen in Figure 7.1. Note that one can see the salt wedge not only from the estimated concentration, but also from the scatter plot of particles. And We see particles behind the salt wedge goes upward when the tides retreats from the domain.
In the studies of salt intrusion, salt intrusion length is often interesting. It can be loosely defined as the distance between the mouth of an estuary and the location where salinity is above a certain value. To study the intrusion length of the lab example, psu values $2.0, 6.0, 10.0$ are chosen to represent low, medium and high salinity condition. The result is plotted in Figure 7.3. There are some observations. Note that we scale the intrusion length by the total length of the channel. Firstly, the salt intrusion length is larger in the lower part of the water column. This indicates the occurrence of salt wedge and stratification. Secondly, the salt water approaches almost half of the water channel in low salinity condition. As for medium and high salinity condition, the maximum intrusion length (considering all z locations) do not vary much, which again indicates the strong advection property of the current case.

7.2. Mathematical modelling of the problem

Remember in Section 5.2.1, a 1D pure diffusion problem was discussed. It is interesting to use the particle-tracking method to study the transport of solute or substance in the domain. The methodology is as follows. Firstly, the particles are sampled according to the initial density distribution. And then, the particles moves under some physical processes like diffusion or advection (and radioaction as well

Parameter	Value
Channel length (m), $L_{channel}$	130
Channel width (m)	1
Channel depth (m), H	0.2
Water level (m), z_{surf}	-0.003
Location of forcing layer($x = ?m$), $x_{forcing}$	9
Salinity of inflow water (psu)	12.5
Tidal period (s)	600
Tidal amplitude (m)	0.0075
Simulation time (s)	6000

Table 7.1: Model parameters for the tidal flume case.

as other chemical or biological processes). The effects of these processes are represented as terms in the advection-diffusion(-reaction) equation. Lastly, the distribution of a solute is estimated from the empirical distribution of particles.
In this model, the same methodology is used. The physical processes under consideration will be advection and vertical diffusion. So the control equation of particles' motion using Euler scheme is

$$\begin{aligned} dX_t &= udt, \\ dZ_t &= (w + \frac{dk}{dz})dt + \sqrt{2k}(e\sqrt{dt}), \end{aligned} \tag{7.1}$$

where u, w are the horizontal and the vertical velocity that varies with time and location (x, z), k is the vertical diffusion that varies with z, and the random number $e \sim N(0, 1)$ is drawn during the calculation. More complicated numerical scheme like the Milstein first-order scheme can be used if necessary. In the implementation, we will notice that $\frac{dk}{dz}$ is dominant compared to background velocity w and the random part. Even worse, the introduction of it leads to unphysical results as we will see, so it will not be included in the simulation later.
Appropriate enforcing the initial condition(IC) and the boundary condition(BC) is also necessary. To ease implementing the IC and estimating the concentration from empirical distribution of particles, a fixed grid with certain dx, dz is used, as shown in Figure 7.2. Then, to implement IC, we can simply sweep through all grid in the computation domain, calculate the concentration at the center of each grid, and sample correct number of particles in each cell accordingly. In comparison, the implementation of BC is more troublesome in this test case because of the questionable boundary at $x = 0$. It is supposed to be a driving force of the experiment with salt water inflow, but the horizontal velocity there remains zero (a zero Dirichlet boundary condition) in the whole simulation time span. To cope with this, it is proposed to move the left boundary from $x = 0$ to $x = 9$. Then, there are two ways of boundary conditions implementation — "Dirichlet-wise" and "Neumann-wise" implementation. The previous implementation treats the layer of cells at the left of the domain (the cells with blue cross in the figure) as a forcing condition. Then one can release or extract corresponding number of particles in the forcing layer so that the number of particles inside the cells matches the required concentration. We will see that the latter implementation outperforms the previous one. The "Neumann-wise" implementation calculates the salt flux into the layer of cells at the left of the domain. This implementation borrows its idea from the finite-volume method. One assumes that salt is distributed uniformly in the water that enters the computation domain in $[t, t + \Delta t]$. To calculate the inward salt flux, one can easily calculate the total inward water mass and then the inward salt mass, as explained in Figure 7.4. Besides, Nothing needs to be done at the right boundary because flow can penetrate through that boundary freely. Actually, the particle in this case can never reach the right boundary of the computation domain. Figure 7.5 is plotted to demonstrate the the inlet condition at $x = 9$. Observe the variation of total water flux, the system becomes stable approximately after two cycles and has a peak flux of 0.06 m^3/s. The horizontal velocity and salinity fluctuating at $x = 9$ also gives people some understanding of how the lab experiment looks like. Besides the BC at the left and right boundary, the non-penetrating boundary conditions at the top and bottom boundary can be implemented using the "recursive algorithm" discussed previously.

7.3. Discussion and analysis

7.3.1. Simulation parameter setting

In Section 7.2, as for the implementation of IC and BC, it is only saying that after a grid is set, corresponding number of particles are sampled inside the cell. In fact, the implementation is more complex than this simple sentence. Suppose each grid inside the computation domain has a size (dx, dy, dz). And suppose the salinity at the center of the cell is s. Furthermore, we release in each cell of same volume c particles per unit salinity, i.e. the mass of salt in grams dissolved in one kilogram of salty water, then the number of particles sampled inside the corresponding cell is $c \cdot s$. Then each particle represents salt of mass

$$m_p = \frac{dV \rho_w s}{cs} = \frac{\rho_w dV}{c}, \tag{7.2}$$

where ρ_w is the salinity of the water and the volume of a cell $dV = dxdydz$. Note that here it is assumed that the density of seawater remains constant, $1020Kg/m^3$, approximately the density of the surface seawater. This assumption is reasonable because seawater density is insensitive to salinity change in the current setting. According to [40], at three typical temperatures 5°C, 20°C, 35°C, the density difference between the water with average salinity 35 psu in the ocean and freshwater is less than $0.02Kg/m^3$. Besides, the density difference between the water with salinity 12.5 psu (the salinity of incoming seawater in this toy problem) and freshwater is less than $0.01Kg/m^3$. As a result, the variation in seawater density is less than 2%, thus it can be assumed that it is constant. From the formula, we know that if the cell size and c are fixed, then the salt mas represented by each particle m_p will be fixed as well.

The particle-tracking method discretizes the continuous salinity scalar field with particles, where each particle carries salt of mass m_p. Then to study the evolution of the salinity, one can also track the motion of salt particles and estimate the salinity scalar field by the distribution of particles. In order to improve the accuracy of the solution, it is intuitive to use more particles, which increases the computation time. To strike a balance between accuracy and trade-off, the parameters in Table 7.2 are used. Note that fixing c, dx, dy, N_z means fixing m_p. Lastly, there are two things to be noted in this simulation. Fixing

Parameter	Value
Number of particles for each unit salinity inside a cell, c	10
Number of cells in the vertical direction, N_z	10
Cell span, or dy (m)	1
Cell width, or dx (m)	4
Salinity of inflow water (psu), s_{inlet}	12.5
time step size dt (s)	1
Starting time t_{start} (s)	1200
Particle simulation duration (s)	600

Table 7.2: Simulation parameters for the tidal flume case.

dx may end up with a vertical column of boundary grid at the right boundary of the computation domain. Since no particle will ever reach that place, this will not violate the uniform grid assumption. And the vertical uniform grid is generated between $z = z_{surf}$ and $z = -H$. When considering the grid for estimating salinity, the change in the waterlevel will be ignored for simplification purpose.

As for the choice of parameters, to control numerical error due to the use of numerical schemes to calculate the trajectory of particles, dt cannot be too large. Otherwise, the unnatural banded structure in the scatter plot of particles can be observed, as in Figure 7.6. After trial-and-error, $dt = 1s$ was chosen as indicated in Table 7.2. The vertical cell size is fixed according to the height of water column.

7.3.2. Erroneous deterministic drift due to diffusion

From the Figure 7.7, we see there is a unnatural cavity in the scatter plot of particles. This is because of the introduction of the deterministic drift, which dominates the vertical movement of particles near the sea bed. To check this, we would like to see different components of vertical velocity near the seabed. For this purpose, we plotted Figure 7.8 that describes the demonstrates the distribution of vertical diffusion, its derivative (the deterministic induced vertical velocity) and background vertical velocity.

While the background velocity and the diffusion near the seabed is near zero, the deterministic part of the induced vertical velocity is relatively large pointing upward, thus drives the particle near the seabed away. Furthermore, we notice that the diffusion profile at certain x location is of a bowl-shape like a theoretical parabolic diffusion profile, this also drives the particle in the upper layer to the center of the water column. Having large vertical velocity components pointing to the middle of the water column, this implies the deterministic part of the vertical diffusion adds a very strong mixing to the system, making the particle gather at the center of the water column. However, this is far too strong. The reason for this is still unclear. The error here may be due to the interpolation error of the vertical diffusion. Or only the random part of the vertical diffusion should be accounted for the effects of the turbulent mixing. No matter which case it is, the effect of $\frac{dk}{dz}$ will be thrown away in the discussion later. This results prompts us to carefully use the vertical diffusion data, otherwise it may end up with erroneous results.

7.3.3. Different implementation of boundary conditions

By comparing Figure 7.9 and Figure 7.10 with the error measure defined in the following subsection, the “Dirichlet-wise” implementation even fails to capture the total salt mass flux into the computation domain, while the “Neumann-wise” implementation not only succeeds in capturing the total salt mass flux, but also matches the salinity distribution at different times to a satisfactory extent, which can also be seen in the gif Figure 7.1. Actually, whether particle-tracking framework can match the incoming or outgoing salt flux is the major difference between the two implementation. The reason why the “Dirichlet-wise” implementation fails in representing the flux is because the socalled “forcing layer” we use is not thin. When the incoming flow is of small velocity compared to the dx of the forcing layer, this may inhibits sampling new particles into the domain. And if the incoming velocity is of large velocity, this may leads to sampling more particles in the computation domain than needed. Similar issue also happens when there is an outgoing flux at the boundary.

7.3.4. Sensitivity analysis

We are now interested in the sensitivity of our simulation results with respect to m_p or c and dx, as well as the choice of the time-stepping scheme like Equation 7.1. We will now use the test case with parameters shown in Table 7.2 as our reference. To make cases under different parameters comparable, an unified error measure is needed. Assume the observation time $\{t_i\}_{i=1}^{N_T}$ and a fixed rectangular grid $\{(x_i, z_j)\}$ where i ranges from 1 to $\lfloor \frac{L_{channel}-x_{forcing}}{dx} \rfloor + 1$ and j ranges from 1 to N_z. Then construct the different salt mass evolution with time as the error measure

$$\begin{aligned} E(t_i) &= \sum_{(x_i,z_j)} |(N_p(x_i,z_j)/c - s_{true}(x_i,z_j))| \cdot dV \cdot 1020 \text{ g}, \\ &= \sum_{(x_i,z_j)} |(N_p(x_i,z_j)/c - s_{true}(x_i,z_j))| \cdot (dx \cdot 1 \cdot \frac{H - z_{surf}}{N_{z,grid}}) \cdot 1020 \text{ g}, \end{aligned} \tag{7.3}$$

where $N_p(x_i, z_j)$ is the number of particles inside the cell centered at (x_i, z_j) and $s_{true}(x_i, z_j)$ at (x_i, z_j) is the salinity recorded in the data. This error measure calculates the difference of salt mass in each cell and add them up. This error measure can also been as a L1 absolute error of salinity, but this error is constructed by salinity in mass, so it can contrasted to the overall incoming salinity mass.

We first increases the number of used particles by increasing the number of particles per salinity in each cell c from 10 to 100. Note that this reduces the salt mass represented by each particle m_p to each one-tenth. We do not see significant variation in the evolution of salinity mass difference and total salinity mass in Figure 7.11 with respect to Figure 7.10. We then fix m_p. Change dx, and c will change accordingly. The results are in Figure 7.12 and Figure 7.13. Both figures are quite similar to Figure 7.10. In conclusion, these results indicates that enough particles are used in the computation. Furthermore, if we change the cell size dx while fixing the salt mass represented by each particle, the results does not change much. In other words, the results are insensitive to the change of cell size dx, **if we use enough particles**. Besides, we also tried to use a fourth-order Runge-Kutta scheme to solve the ODEs, and it also gives similar results and thus the plots are not provided here.

7.3.5. Uses of high-order scheme

From Figure 7.14, it was found out that the uses of the Milstein first-order scheme does not make significant changes in representing the overall results, and in particular, the vertical mixing.

7.4. Summary

The tidal flume test cases is collected from a lab experiment that conducts tidal research in a rectangular tank of simple geometry. Several major observations are made here.

Firstly, we tried to represent realistic vertical physics by including the effects of background vertical velocity and vertical diffusion. Even though we failed to include the effect of the deterministic $\frac{dk}{dz}$ term, after excluding it, the movement of particles in the vertical direction seems reasonable. We see that switching to a high-order numerical scheme like the Milstein first-order scheme from the Euler scheme does not have much influence on the simulation results. This is probably linked to the relatively small timestep $dt = 1$ in the simulation. However, there are still much to explore in how to represent the vertical transport physics. Firstly, if the flow changes violently, whether the assumption of one-way coupling here hold or not? As we see in our results, all salt particles near the seabed actually move upward. If this would happen, then there should be a large density difference between the water near the seabed and the water above it, thus creating a downward flow. Other factors that will play a role in representing the vertical physics are the interpolation error, the numerical scheme and the choice of time stepsize. Even though the latter does not play a significant role in the tidal flume case, it is suggested to test higher-order scheme in more realistic cases to check if the vertical transport physics is well represented.

Secondly, this tidal flume case may inspire people on the implementation of complex boundary conditions. It seems that the “Neumann-wise” implementation is preferred to account for the inward or outward mass flux conditions. This implementation will be particularly useful when one wants to focus on the computation constrained in a specific domain.

To conclude this chapter, it would nice to try the timescale method here. We plot the distribution of the scaled residence time in the domain in Figure 7.15. Note that the residence time distribution reflects in each cell the residence time of salt water in the domain. The more particles in the cell and the longer time particles stay in the computation domain, the residence time of the salt water in that cell will be larger. Note that the residence time of salt water will drop after mixing with freshwater.

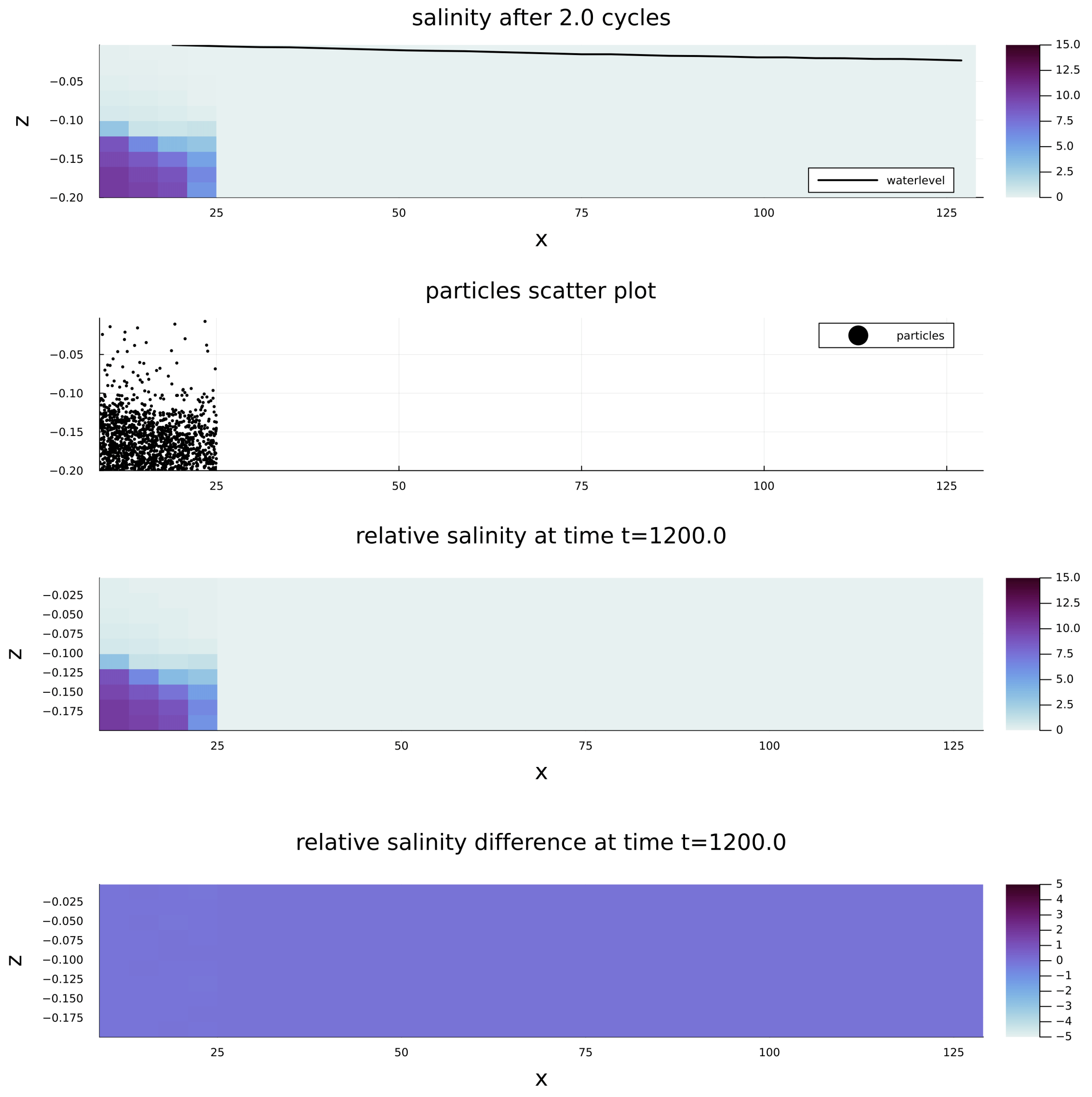


Figure 7.1: A demo of tidal flume case: a lab experiment about salt intrusion into a water channel. Relative salinity difference is obtained by salinity in data minus salinity calculated from particles.

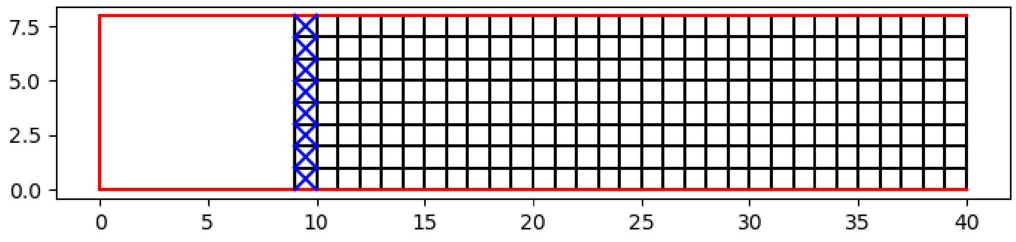


Figure 7.2: Schematic plot of tidal flume device. The domain with square grid is the computation region, and the blank domain is the “dead region”. The upper and lower boundary in red represents water-air interface and seabed respectively, thus non-penetrating boundary condition needs to be implemented. The left boundary in red represents a questionable boundary. The right boundary in black represents a free boundary where fluid can flow across it freely. The grid with blue cross in the computation domain represents the actual forcing layer of the numerical experiment. **The size of the domain and the grid are not drawn to scale.**

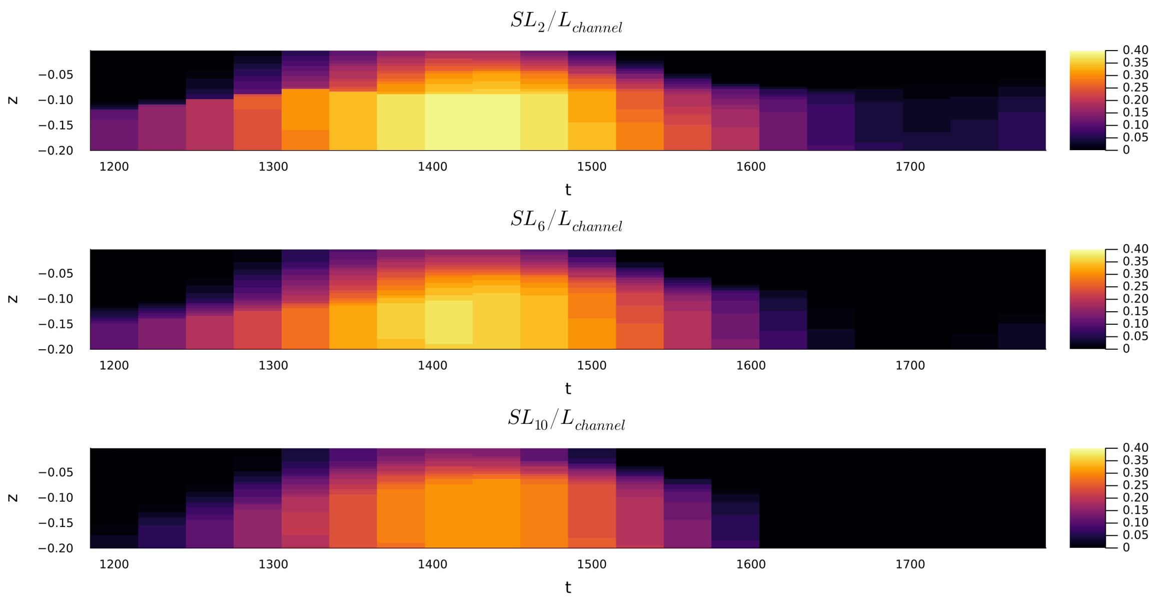


Figure 7.3: Scaled salinity intrusion length that varies with time and space. Three Salinity intrusion conditions are plotted, representing low, medium and high salinity conditions respectively.

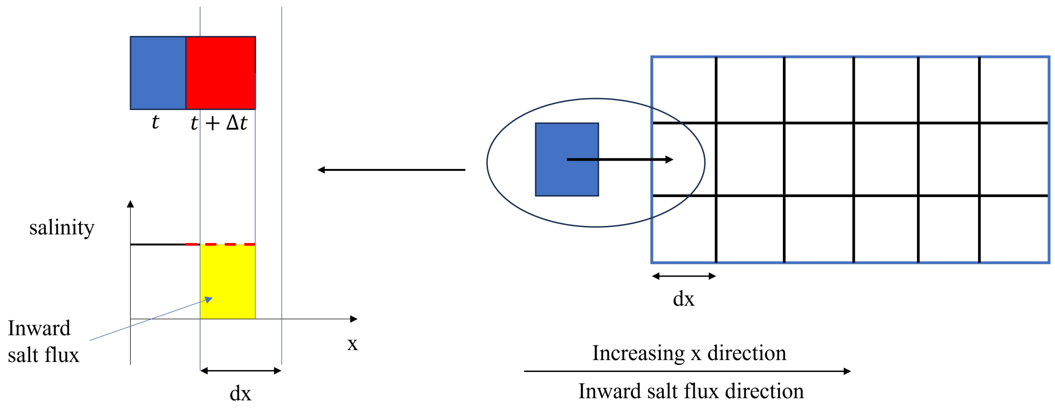


Figure 7.4: An explanation about "Neumann-wise" boundary condition implementation. This implementation assumes a uniform density distribution in the water flow into the computation domain. This implementation is inspired by contents in [8], especially Figure 4.1 in Dr. Dunsbergen's PhD thesis, where it assumes a 1D Gaussian-like distribution and tracks its evolution in space before sampling the particles. If we assume a uniform salinity distribution in the incoming water parcel and the shape of the water parcel does not change much in each time-stepping, this implementation can be easily carried to 3D cases. One follows similar procedure as this case by tracking the center of water parcel, calculating the inward flux and specifying the sampling region after the update of existing particles in the computation domain. Of course, one can throw away the two assumptions, but this will require careful interpolation work.

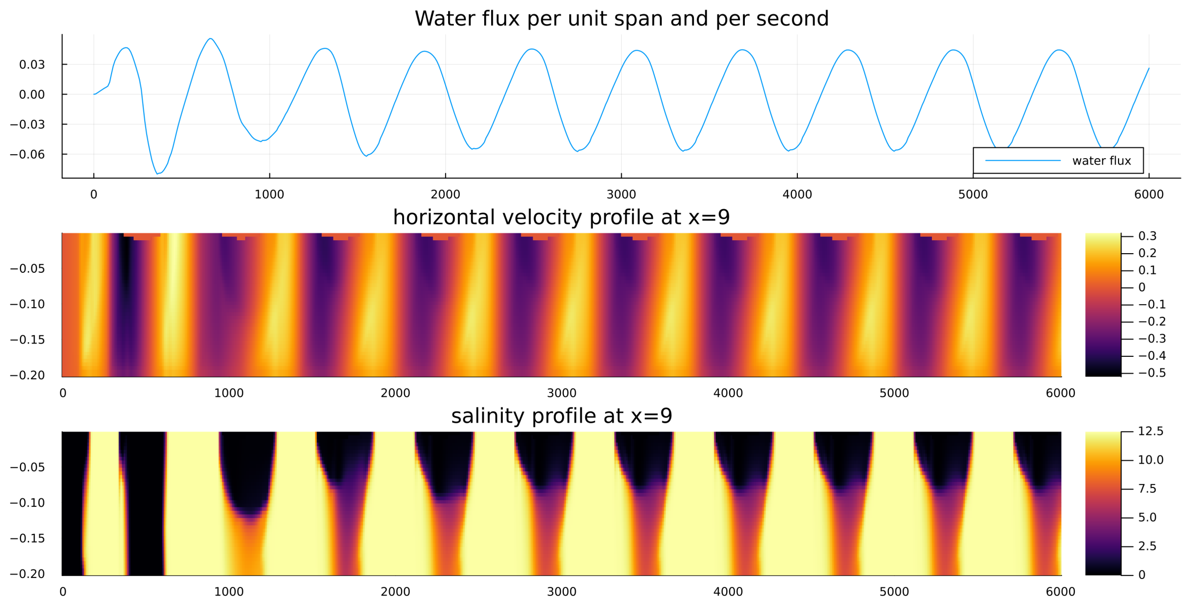


Figure 7.5: Total water flux, horizontal and salinity profile at $x = 9m$ vary with time: 40 cells in the vertical direction to estimate the water flux and plot the profiles. The horizontal axis of the heatmap is time t in second, and the vertical axis is the location z in meter.

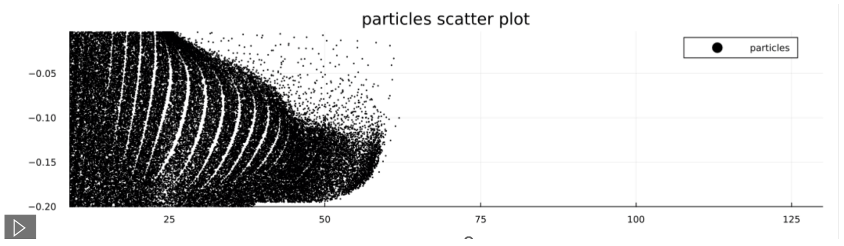


Figure 7.6: A screenshot of particles at some time calculated at $dt = 10s$. The horizontal axis is x-axis, the vertical axis z-axis.

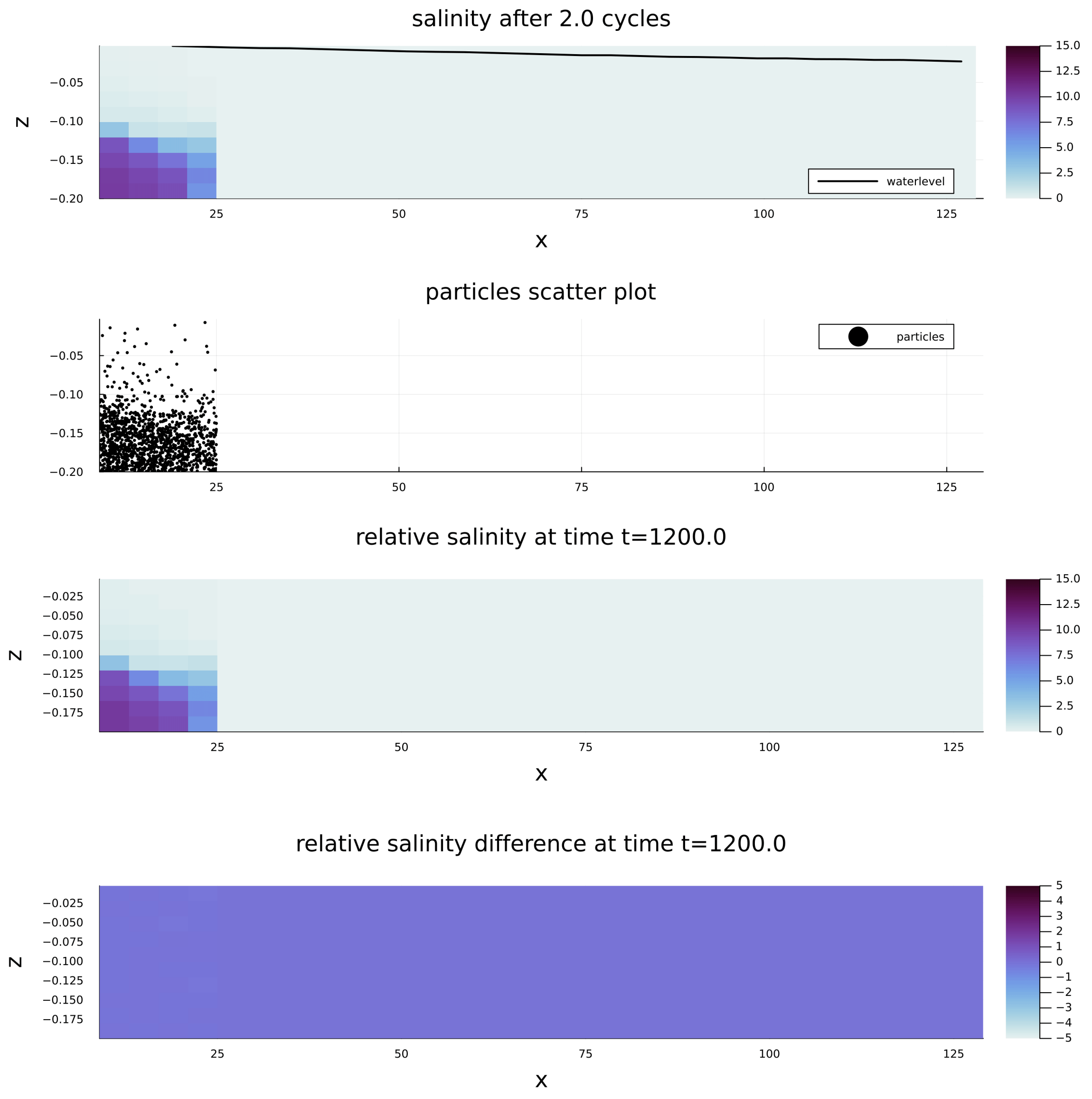


Figure 7.7: A demo of tidal flume case: a lab experiment about salt intrusion into a water channel. Relative salinity difference is obtained by salinity in data minus salinity calculated from particles. In this simulation, **the vertical velocity includes the dk/dz component, which adds too strong mixing to the system**.

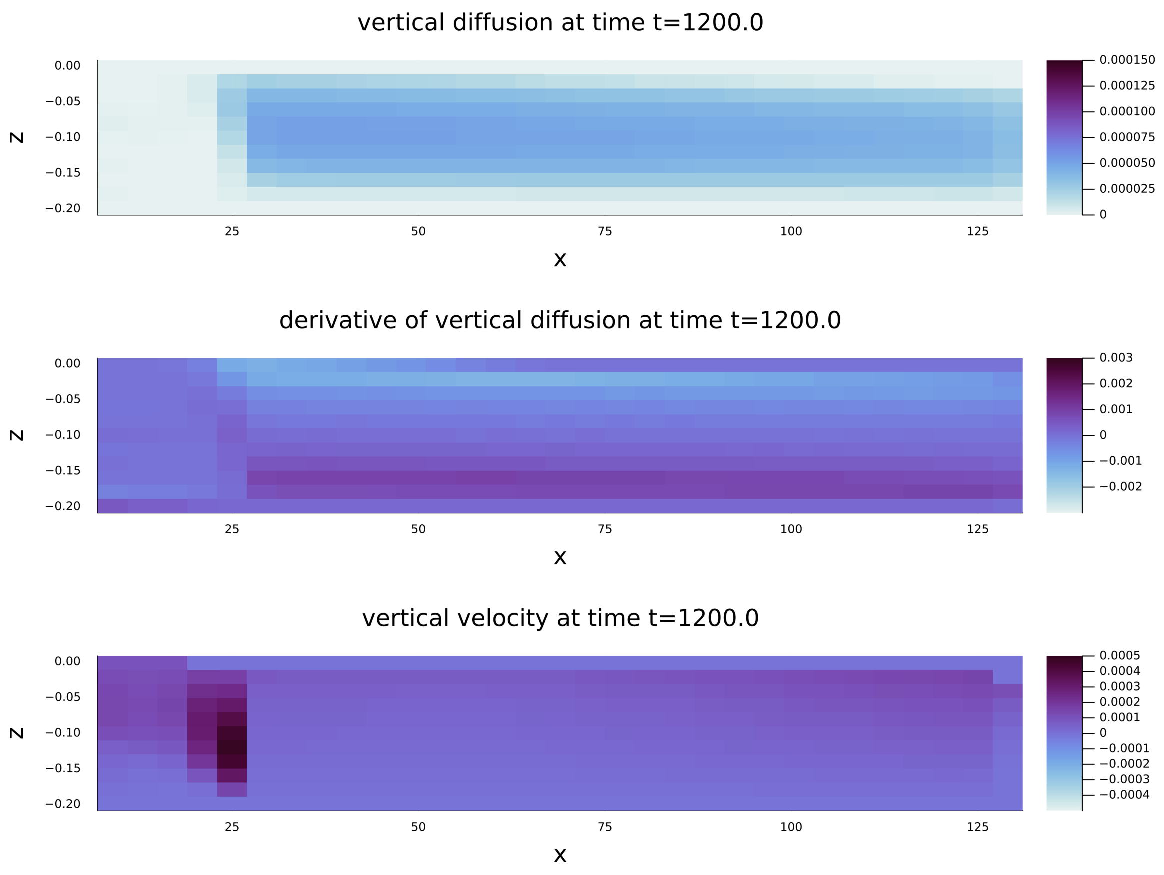


Figure 7.8: The distribution of vertical diffusion, its derivative (the deterministic induced vertical velocity) and background vertical velocity.

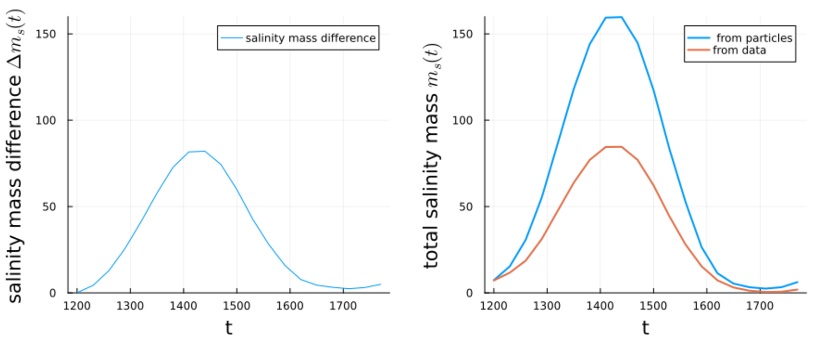


Figure 7.9: Salinity mass difference $E(t)$ and total salinity mass in the computation domain - using “Dirichlet-wise” implementation.

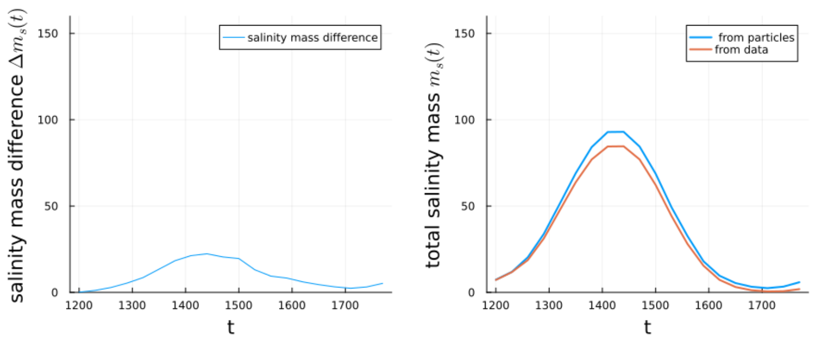


Figure 7.10: Salinity mass difference $E(t)$ and total salinity mass in the computation domain - using “Neumann-wise” implementation.

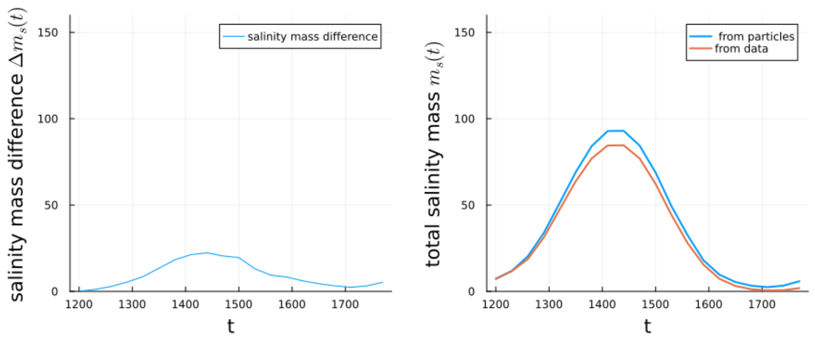


Figure 7.11: Sensitivity analysis: $c = 100$.

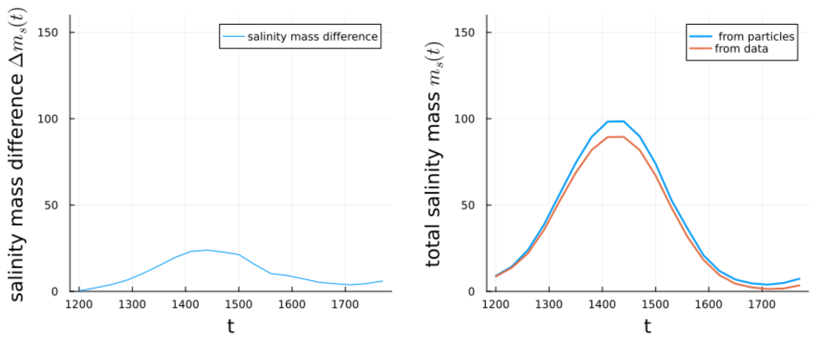


Figure 7.12: Sensitivity analysis: $c = 5$, $dx = 2$.

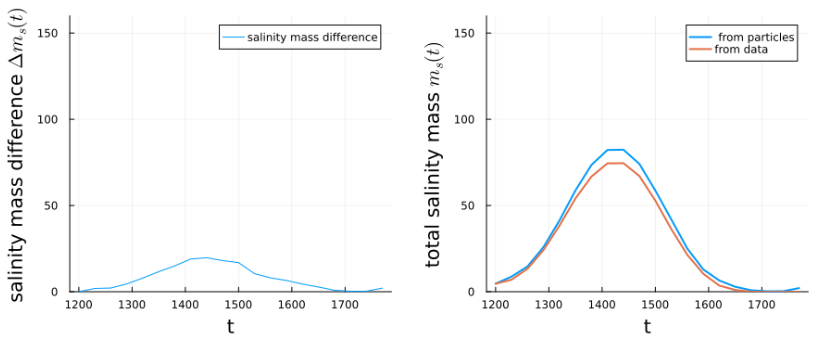


Figure 7.13: Sensitivity analysis: $c = 20$, $dx = 8$.

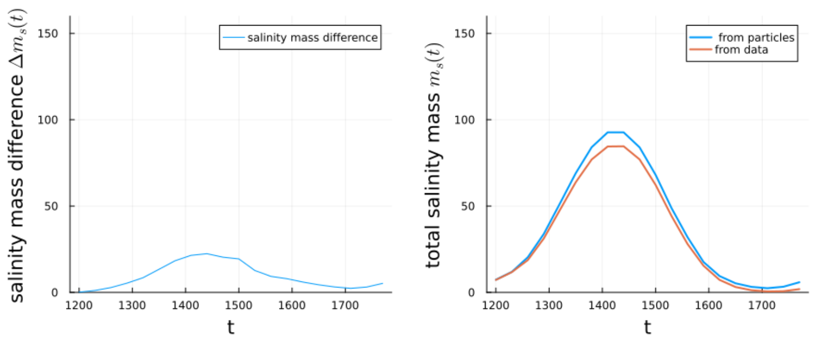


Figure 7.14: Sensitivity analysis: Calculation the particle-tracking results using the M1 scheme.

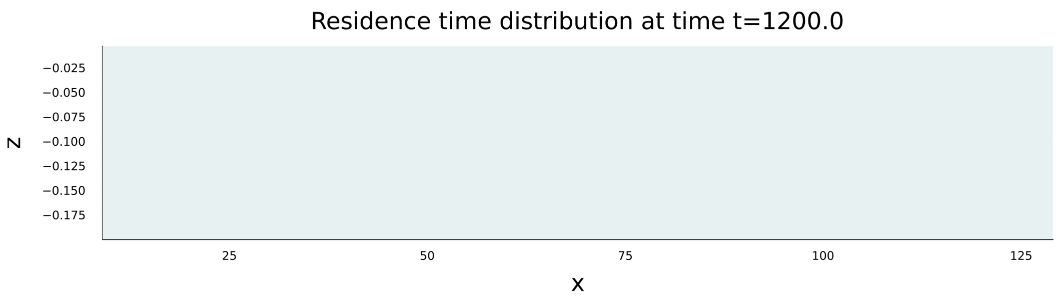


Figure 7.15: The distribution of scaled residence time in the region. The residence time is calculated by summing up the the residence time of all particles and dividing $c \times s_{inlet}$.

8 Conclusion

8.1. Summary

In this thesis, a particle-tracking framework for salinity intrusion is developed. It would be nice to summarize the major conclusions, findings or observations in each chapter here. Chapter two introduces the basic physics of salt intrusion, especially the occurrence of salt wedge and the commonly used models for that. The third chapter mainly discusses the motivation for developing the particle-tracking framework. The mathematics behind the it is also discussed where much is devoted to SDE numerical methods and density estimation. This establishes a sound mathematical basis for the discussion later. Chapter four signifies the importance to implement the non-penetrating boundary conditions reasonably when the diffusion is present using two simple examples. The fifth and the sixth chapter carries out extensive discussion to the one-dimensional (advection-)diffusion equation. By following existing literature in chapter five we confirm that the particle-tracking framework can be used to tackle transport problems with suitable parameter setting. Then in chapter six, we carefully examine the error measure of the problem, study the influences of different error sources. We see how numerical diffusion strongly affects the results near the boundary. This may be regarded as another example of large numerical error contaminating the transport physics. This may provide people with some new inspiration about the importance of SDE numerical schemes and the choice of timestep size. Then in chapter seven by studying the tidal flume case, we came up with ways to implement inward and outward boundary conditions. We also tried representing the effect of vertical diffusion. Furthermore, after carefully selecting different kinds of parameters, a sensitivity analysis proves that if we control the salt mass represented by each particle, the result is stable with respect to changes in grid size and the number of particles. Besides, note the sensitivity analysis partially addresses the research problem about results stability with respect to the number of particles and grid size.

8.2. Findings, limitations and future perspectives

Now it's the time to answer our major research questions. The major difficulties in using the particle-tracking framework to study salt intrusion are

- how to represents the vertical physics correctly
- implementation of initial and boundary condition

Representing the vertical transport physics reasonably requires smart choice on the numerical schemes and appropriate timestep size as well as a good interpolated vertical diffusion function. All three factors are important and worth attention and requires extensive parameter studies in numerical experiments. In particular, one can label out the significant vertical physic in a test case and check how well the particle-tracking framework represents it. This can be seen as verification technique.
Furthermore, in chapter four and seven, we implemented certain initial conditions and boundary conditions in the particle-tracking framework. The implementation of initial conditions requires appropriate sampling. This often assumes a uniformly distributed rectangular grid and involves special treatment near the boundary when the grid is trimmed off. The implementation of non-penetrating boundary conditions is done by a recursion algorithm. This algorithm is better than the reflection algorithm because

it keeps the particles near the boundary instead of driving them away, which models the mixing effect by the wall-bounded turbulence better. And as for the implementation of the inward or outward flux conditions, the "Neumann-wise" way is suggested. The implementation is demonstrated in different test cases, which can be useful reference for future work. This partially addresses the first research question about the implementation of initial and boundary conditions. Similar ideas can be extended to three-dimensional cases easily if one assumes the concentration is uniform in the incoming seawater parcel, but to achieve high-order accuracy one needs to take care about the interpolation. More importantly, in realistic 3D cases, we often want to trim down the data in a large domain to limit our discussion in sub-domain to decrease the computation cost, which often leads to inward or outward boundary conditions from a practical perspective.

Even though we are unable to answer research questions like "what are the implications of 3D particle-tracking simulation"? Still, by having the results we have, we can conclude that since salt transport is already resolved in the common ocean FVM-based model, it is neither necessary nor useful in using the particle-tracking algorithm to resolve salt transport. Note that calculating the salinity concentration is merely a way of verification and the density estimation operation is the computation bottleneck in our experiment. The particle-tracking framework for solving transport problems can be useful in tracking the dispersion of pollutants in the ocean. Then, here's the problem: what is the use of the particle-tracking algorithm for salt intrusion? The power of particle-tracking algorithm lies in its use in tracer and timescale methods, as elaborated in [27] and the special volume that includes this article. For example, in salt intrusion, people are often interesting in the residence time of salt mass into the estuary freshwater system, which is essential to the ecosystem there. Using the particle-tracking algorithm allows us to study the salt mass residence time distributed in a large domain. Moreover, by calculating the statistics of particles, a unified timescale can also be computed, which is more flexible compared to normal grid-based method. These interesting timescales that can be calculated with the help of the particle-tracking algorithm include flushing time, the time needed to drive a certain proportion of salt out of estuary system or the time needed to reduce the salinity down to a certain level, and the transit time, the time needed for the bulk of salt water flux to reach certain location in the river upstream. In short, the tracer and timescale methods is still a new field that awaits people's exploration. People are expected to have insights for coastal management from the particle-tracking results.

Furthermore, it will be nice to comment on several research questions that are still open and discuss some possibilities of future work. Firstly, how to reduce the error in the tracking the particles trajectories? Using high-order SDE numerical schemes is a frequently mentioned option, and well interpolated functions of the velocity and the diffusion in the computation domain should not be overlooked. As for the first option, one can embed existing SDE packages to our program to test their efficacy. However, the difficulties can be how to select an appropriate Ito schemes or interpret the SDE problem in a Stratonovich way consistently. Even if one takes care of the aforementioned issues and has a correct implementation of SDE schemes, there are still problems. Namely, high-order derivatives terms are emerging, which can lead to numerical stability issues. The efficiencies of those numerical schemes are also worrying, because they can include much more terms than simple schemes. For the interpolation of velocity and diffusion, it is suggested to examine both the original data and the interpolation method carefully, otherwise the interpolation error can contaminate the particle-tracking results. The second research question discussed here is the scaling of this problem. This problem requires people's attention because many particles are needed in a more realistic 3D case. Obviously, the GPU parallelization for this particle-tracking framework is the suitable choice. The advantage of using GPU parallelization lies in the independence of the movement of each particle. Nevertheless, the difficulty is mainly in the implementation under the current framework, because CUDA.jl in the Julia environment is still under development and people can expect weird bugs when they conduct memory management and atomic operations.

References

[1] Sudip Basack et al. “Saltwater intrusion into coastal aquifers and associated risk management: Critical review and research directives”. In: *Journal of Coastal Research* 38.3 (2022), pp. 654–672.

[2] Md. Jabed Abdul Naser Bhuiyan and Dushmanta Dutta. “Assessing impacts of sea level rise on river salinity in the Gorai river network, Bangladesh”. In: *Estuarine, Coastal and Shelf Science* 96 (2012), pp. 219–227. issn: 0272-7714. doi: 10.1016/j.ecss.2011.11.005.

[3] Wilson Mahera Charles. “Transport Modelling in Coastal Waters using Stochastic Differential Equations”. Delft University of Technology, 2007. url: http://resolver.tudelft.nl/uuid:43dae97c-431c-4ed3-8064-35686940a32a.

[4] Elena Crestani et al. “Large-Scale Physical Modeling of Salt-Water Intrusion”. In: *Water* 14.8 (2022), p. 1183. doi: 10.3390/w14081183.

[5] Jeroen Daniëls. “Dispersion and dynamically one-dimensional modeling of salt transport in estuaries”. Delft University of Technology, 2016. url: http://resolver.tudelft.nl/uuid:0043be29-6a88-419e-991d-2618f4f21c37.

[6] Eric Deleersnijder, Jean-Marie Beckers, and Eric JM Delhez. “The residence time of settling particles in the surface mixed layer”. In: *Environmental Fluid Mechanics* 6 (2006), pp. 25–42.

[7] *Deltares team secures second place in forecasting challenge Atlantic Ocean with new development in particle tracking*. Deltares, 2021. url: https://www.deltares.nl/en/news/deltares-usa-secures-second-place-in-forecasting-challenge-atlantic-ocean-with-new-development-in-particle-tracking.

[8] Daniël Willem Dunsbergen. “Particle models for transport in three-dimensional shallow water flow.” In: (1996).

[9] Hugob. Fischer. “Mass transport mechanisms in partially stratified estuaries”. In: *Journal of Fluid Mechanics* 53.4 (1972), pp. 671–687. doi: 10.1017/S0022112072000412.

[10] William E Fleenor and Fabián A Bombardelli. “Simplified 1-D Hydrodynamic and Salinity Transport Modeling of the Sacramento–San Joaquin Delta: Sea Level Rise and Water Diversion Effects”. In: *San Francisco Estuary and Watershed Science* 11.4 (2013). doi: 10.15447/sfews.2013v11iss4art2.

[11] Saarang Gaggar et al. “Application of smoothed particle hydrodynamics for the simulation and analysis of vibratory finishing process”. In: *The International Journal of Advanced Manufacturing Technology* 108.1-2 (2020), pp. 183–190. issn: 0268-3768. doi: 10.1007/s00170-020-05307-9.

[12] Markus Giese and Roland Barthel. “Saltwater intrusion in fractured crystalline bedrock.” In: *Hydrogeology Journal* 29.7 (2021).

[13] Robert A. Gingold and Joseph J. Monaghan. “Smoothed particle hydrodynamics: theory and application to non-spherical stars”. In: *Monthly Notices of the Royal Astronomical Society* 181.3 (1977), pp. 375–389. issn: 0035-8711. doi: 10.1093/mnras/181.3.375.

[14] Ulf Gräwe. “Implementation of high-order particle-tracking schemes in a water column model”. In: *Ocean Modelling* 36.1-2 (2011), pp. 80–89. issn: 1463-5003. doi: 10.1016/j.ocemod.2010.10.002.

[15] Ulf Gräwe et al. “Why the Euler scheme in particle tracking is not enough: the shallow-sea pycnocline test case”. In: *Ocean Dynamics* 62.4 (2012), pp. 501–514. issn: 1616-7341. doi: 10.1007/s10236-012-0523-y.

[16] Arnold Heemink et al. “Lagrangian Modelling of Transport Phenomena Using Stochastic Differential Equations”. In: *The Mathematics of Marine Modelling: Water, Solute and Particle Dynamics in Estuaries and Shallow Seas*. Springer, 2022, pp. 213–242.

[17] Desmond J Higham. “An algorithmic introduction to numerical simulation of stochastic differential equations”. In: *SIAM review* 43.3 (2001), pp. 525–546.

[18] Bo Hong and Jian Shen. “Responses of estuarine salinity and transport processes to potential future sea-level rise in the Chesapeake Bay”. In: *Estuarine, Coastal and Shelf Science* 104 (2012), pp. 33–45. issn: 0272-7714. doi: 10.1016/j.ecss.2012.03.014.

[19] Mohammed S Hussain et al. “Management of seawater intrusion in coastal aquifers: a review”. In: *Water* 11.12 (2019), p. 2467.

[20] Mohammed Russedul Islam, Md. Aftabur Rahman, and Kimitoshi Hayano. “Application of smoothed particle hydrodynamics (SPH) for simulating various geotechnical problems”. In: *SN Applied Sciences* 2.4 (2020), p. 687. issn: 2523-3963. doi: 10.1007/s42452-020-2379-y.

[21] J.W.Stijnen. “Numerical methods for stochstic environmental models”. Delft University of Technology, 2002. url: http://resolver.tudelft.nl/uuid:fc7196f7-2405-44b1-80a2-3d47bf175eca.

[22] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[23] Herman WJ Kernkamp et al. “Efficient scheme for the shallow water equations on unstructured grids with application to the Continental Shelf”. In: *Ocean Dynamics* 61 (2011), pp. 1175–1188.

[24] Yichun Li and Jingui Liu. “A numerical study on salinity stratification at the Oujiang River Estuary, China”. In: *Acta Oceanologica Sinica* 38.11 (2019), pp. 40–50.

[25] Moubin Liu and Zhilang Zhang. “Smoothed particle hydrodynamics (SPH) for modeling fluid-structure interactions”. In: *Science China Physics, Mechanics & Astronomy* 62.8 (2019), p. 984701. issn: 1674-7348. doi: 10.1007/s11433-018-9357-0.

[26] Wen-Cheng Liu and Hong-Ming Liu. “Assessing the Impacts of Sea Level Rise on Salinity Intrusion and Transport Time Scales in a Tidal Estuary, Taiwan”. In: *Water* 6.2 (2014), pp. 324–344. doi: 10.3390/w6020324.

[27] Lisa V. Lucas and Eric Deleersnijder. “Timescale Methods for Simplifying, Understanding and Modeling Biophysical and Water Quality Processes in Coastal Aquatic Ecosystems: A Review”. In: *Water* 12.10 (2020). issn: 2073-4441. doi: 10.3390/w12102717. url: https://www.mdpi.com/2073-4441/12/10/2717.

[28] Rena Meyer, Peter Engesgaard, and Torben O Sonnenborg. “Origin and dynamics of saltwater intrusion in a regional aquifer: Combining 3-D saltwater modeling with geophysical and geochemical data”. In: *Water Resources Research* 55.3 (2019), pp. 1792–1813.

[29] Ruqayah Mohammed and Miklas Scholz. “Critical review of salinity intrusion in rivers and estuaries”. In: *Journal of Water and Climate Change* 9.1 (2018), pp. 1–16. issn: 2040-2244. doi: 10.2166/wcc.2017.334.

[30] Hans-Georg Müller. “Smooth optimum kernel estimators near endpoints”. In: *Biometrika* 78.3 (1991), pp. 521–530.

[31] Michel AJ de Nijs, Julie D Pietrzak, and Johan C Winterwerp. “Advection of the salt wedge and evolution of the internal flow structure in the Rotterdam Waterway”. In: *Journal of Physical Oceanography* 41.1 (2011), pp. 3–27.

[32] Hisamichi Nobuoka and Nobuo Mimura. “Precise nearshore currents model using sigma coordinate system”. In: *Asian And Pacific Coasts 2003: (With CD-ROM)*. World Scientific, 2004, pp. 1–11.

[33] Marta Payo-Payo et al. “Multiscale temporal response of salt intrusion to transient river and ocean forcing”. In: *Journal of Geophysical Research: Oceans* 127.3 (2022), e2021JC017523.

[34] Subhrangshu Purkayastha and Mohammad Saud Afzal. “Review of Smooth Particle Hydrodynamics and its Applications for Environmental Flows”. In: *Journal of The Institution of Engineers (India): Series A* 103.3 (2022), pp. 921–941. issn: 2250-2149. doi: 10.1007/s40030-022-00650-4.

[35] Hannes Risken. “The Fokker-Planck Equation, Methods of Solution and Applications”. In: *Springer Series in Synergetics* (1996). issn: 0172-7389. doi: 10.1007/978-3-642-61544-3.

[36] Andrew C. Ross et al. “Sea-level rise and other influences on decadal-scale salinity variability in a coastal plain estuary”. In: *Estuarine, Coastal and Shelf Science* 157 (2015), pp. 79–92. issn: 0272-7714. doi: 10.1016/j.ecss.2015.01.022.

[37] *Salt Wedge estuaries*. url: http://www.coastalwiki.org/wiki/Salt_wedge_estuaries.

[38] Hubert HG Savenije. *Salinity and tides in alluvial estuaries*. Gulf Professional Publishing, 2005.

[39] Nelleke Scheijen. “Modeling of the vertical transport of micro plastics in the ocean”. Delft University of Technology, 2017. url: http://resolver.tudelft.nl/uuid:b17ae877-d2db-4ea8-8280-394741395224.

[40] H. Schmidt et al. “The density–salinity relation of standard seawater”. In: *Ocean Science* 14.1 (2018), pp. 15–40. doi: 10.5194/os-14-15-2018. url: https://os.copernicus.org/articles/14/15/2018/.

[41] Erik van Sebille et al. “Lagrangian ocean analysis: Fundamentals and practices”. In: *Ocean Modelling* 121 (2018), pp. 49–75. issn: 1463-5003. doi: 10.1016/j.ocemod.2017.11.008.

[42] Bernard W Silverman. *Density estimation for statistics and data analysis*. Vol. 26. CRC press, 1986.

[43] Darýa Spivakovsk̀a. “Reverse-time diffusion in environmental models”. Delft University of Technology, 2007. url: http://resolver.tudelft.nl/uuid:43dae97c-431c-4ed3-8064-35686940a32a.

[44] Darya Spivakovskaya, Arnold W Heemink, and Eric Deleersnijder. “Lagrangian modelling of multi-dimensional advection-diffusion with space-varying diffusivities: theory and idealized test cases”. In: *Ocean Dynamics* 57 (2007), pp. 189–203.

[45] J. W. Stijnen, A. W. Heemink, and H. X. Lin. “An efficient 3D particle transport model for use in stratified flow”. In: *International Journal for Numerical Methods in Fluids* 51.3 (2006), pp. 331–350. issn: 0271-2091. doi: 10.1002/fld.1132.

[46] Hyder Ali Muttaqi Shah Syed. “Lagrangian Modelling Of Transport Processes In The Ocean”. Delft University of Technology, 2015. url: https://doi.org/10.4233/uuid:e968a8b7-1a0d-437c-9cdd-5d8af2c6dc4a.

[47] Pengfei Xue et al. “Saltwater intrusion into the Changjiang River: A modelguided mechanism study”. In: *Journal of Geophysical Research: Oceans* 114.C2 (2009). issn: 0148-0227. doi: 10.1029/2008jc004831.

[48] Hidekazu Yoshioka, Koichi Unami, and Toshihiko Kawachi. “Stochastic process model for solute transport and the associated transport equation”. In: *Applied Mathematical Modelling* 36.4 (2012), pp. 1796–1805.

A

Appendix: Basics of stochastic process, SDEs and SDE numerical schemes

A.1. Basics of stochastic process and stochastic differential equations

The introduction about stochastic process and SDE mainly follow [16] and [41], while using [22] by Jazwinski for knowledge about the basics of stochastic process. Interested readers are suggested to these references for more information.

A.1.1. Stochastic process basics

A probability space is defined as a triplet of (Ω, B, P). Ω is called sample space, where each element ω in the space is called a sample. And the sets of samples form an event space B, which is a borel field and closed under operations of taking complements, infinite union, and intersection. Probability function P is a mapping that maps elements in the sample space to elements on the close interval $[0, 1]$. Random variables are defined from the sample space to a measurable space, often to numbers. Furthermore, consider a family of random variables $\{X_t, t \in T\}$, where T is a parameter set that consists of numbers. T is an interval or consists of discrete numbers. The family of random variables $\{X_t, t \in T\}$ is a random process, though someone thinks name of a process is justified only if the measurable space consisting of numbers to which a random variables is mapped is an interval.
Given a finite set $T_k = \{t_1, t_2, \dots, t_k\} \subseteq T$, we have a set of random variables $\{X_{t_1}, X_{t_2}, \dots, X_{t_k}\}$. We can define a finite-dimensional distribution $F(X_{t_1}, \dots, X_{t_k})$ or a joint probability density function $p(X_{t_1}, \dots, X_{t_k})$. It is stated without proof that a stochastic process is completely fixed if the finite-dimensional distribution or the joint probability density function for any finite set T_k is determined.

According to [16], a standard Brownian motion (also known as Wiener process) W_t is defined to be a stochastic process with identical independent increments

$$W_t - W_s \sim N(0, t-s), t > s \text{ and } W_0 = 0. \tag{A.1}$$

The Browinian motion is also Markovian, which means that the statistics of the future states are determined by the present state only. The previous states do not give any additional information. It will be beneficial to draw a link between the Browinian motion and the particle model. First, the Brownian motion is used to capture the randomness of the diffusion of particles: the strength of diffusion is related to the maximum distance of a particle induced by the diffusion in a fixed time step. Second, the Markovian property plays a role because the position of a particle at the next time step will be determined by the position of the particle at the current time step. In other words, no information at the previous time steps is needed. The Markovian property will be our assumption, otherwise the model will become more complicated and may be hard to implement because the information about the previous states needs to be stored.

The Gaussian process is a stochastic process where finite-dimensional distribution of the stochastic process for any parameter set T is normal. A white Gaussian process $\{X_t, t \in T\}$ is a Gaussian process with covariance

$$E\{(X_t - EX_t)(X_\tau - EX_\tau)\} = Q(t)\delta(t-\tau), \quad \tau \in T, \tag{A.2}$$

where $Q(t)$ is a positive semidefinite covariance matrix and $\delta(t-\tau)$ is a Dirac function.

A.1.2. Introduction of stochastic differential equations

Follow the discussion in [16] by Heemink et al. if not stated otherwise. The trajectory of a moving particle in a nondiffusive fluid can be described by the following ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t), \mathbf{x}(t_0) = \mathbf{x_0}, \tag{A.3}$$

where the position of the particle at time t is written as $\mathbf{x}(t) = (x(t), y(t), z(t))$. In the case when the influence of the diffusion cannot be neglected, a Gaussian white-noise process is used to capture the randomness in the movement of the particle. Consequently, we discuss an SDE that models the moving of particle in a diffusive fluid,

$$\boldsymbol{\frac{d\mathbf{X_t}}{dt} = \mathbf{f}(\mathbf{X_t}, t) + \sigma(\mathbf{X_t}, t)\mathbf{N}_t, \mathbf{X}_{t_0} = \mathbf{X_0},} \tag{A.4}$$

where $\mathbf{f}$ may represent the velocity of the background flow, $\boldsymbol{\sigma}$ may represent the strength of the diffusion and N_t is a random process to represent the uncertainty in the motion. To characterize N_t more precisely and introduce the concepts that are relevant to SDEs, the following one-dimensional SDE is considered. Those concepts can be extended to a system of SDE later.

$$\frac{dX_t}{dt} = f(X_t, t) + \sigma(X_t, t)N_t, X_{t_0} = X_0. \tag{A.5}$$

Again, the model is supposed to be Markovian because it will be simple and easy to implement. It can be proved that the model will be Markovian if the random process N_t is a Gaussian white noise process with the following statistics,

$$E\{N_t\} = 0, \quad E\{N_t N_s\} = \delta(t-s), t \geq s.$$

where E is the expectation function and δ is the Dirac (impulse) function. In order to integrate Equation A.5 to find the trajectory of a particle, it will be attempting to move the dt to the RHS of the equation. It can be proved that the mean-square generalized derivative of a Wiener process is a Gaussian continuous white noise process

$$\frac{dW_t}{dt} = N_t, \quad \text{or} \quad dW_t = N_t dt.$$

Then Equation (A.5) can be written as

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t. \tag{A.6}$$

or in an integral form

$$X_t = X_{t_0} + \int_{t_0}^{t} f(X_t, t)dt + \int_{t_0}^{t} \sigma(X_t, t)dW_t, \tag{A.7}$$

where on the RHS of equation the first integral is a deterministic integral and the second integral is a stochastic one. The stochastic integral is discussed in the next subsection.

A.1.3. Stochastic integrals

For the sake of simplicity, it will be beneficial to discuss a simple stochastic integral $\int_{t_0}^{t} W_s dW_s$, where W_s is a Wiener process and its deterministic counterpart $\int_{t_0}^{t} sds$. The Riemann definition of the deterministic integral is

$$\int_{t_0}^{t} sds = \lim_{\Delta t \to 0} \sum t_i^*(t_{i+1} - t_i) = \frac{t^2 - t_0^2}{2}, \tag{A.8}$$

where the $t_0 < t_1 < \cdots < t_{n-1} < t_n = t$ is a partition on the interval $[t_0, t]$ and t_i^* is a point on the closed interval $[t_i, t_{i+1}]$. It can be shown that the limit of the infinite sum exists and thus the integral is well defined. The limit result can easily be verified by taking $t_i^* = (t_i + t_{i+1})/2$. Thus, it is natural to define a stochastic counterpart of a deterministic Riemann integral

$$\int_{t_0}^{t} W_s dW_s = \text{l.i.m.}_{\Delta t \to 0} \sum W_{t_i^*}(W_{t_{i+1}} - W_{t_i}), \tag{A.9}$$

where the l.i.m. represents the limit in the mean square sense. Consider a sequence of random variables $\{X_i, i = 1, 2, \dots\}$, the sequence converges to X in the mean square sense if

$$\lim_{n \to \infty} E\{(X_n - X)^2\} = 0,$$

which can be expressed as

$$\text{l.i.m.} X_n = X.$$

The mean square convergence implies that as n approaches infinity, the probability of the event that the variance between X_n and X is not zero approaches zero. Note that the mean square convergence is stronger than the convergence in probability, which can be defined as

$$lim_{n \to \infty} P(|X - X_n| > \epsilon) = 0, \quad \forall \epsilon > 0,$$

and denoted as

$$\text{plim} X_n = X.$$

The stochastic limit of Equation (A.9) can be derived as

$$\int_{t_0}^{t} W_s dW_s = \frac{W_t^2 - W_{t_0}^2}{2} - \frac{t - t_0}{2} + \sum_i (t_i^* - t_i). \tag{A.10}$$

The stochastic nature of the integral is reflected in different choices of t_i^*. Japanese mathematician Ito gave the first definition of stochastic integral

$$\int_{t_0}^{t} \sigma_s dW_s = \text{l.i.m.}_{\Delta t \to 0} \sum \sigma_{t_i}(W_{t_{i+1}} - W_{t_i}). \tag{A.11}$$

Note that the Ito integral is computed by taking the value of the integrand at the left boundary point of the interval $[t_i, t_{i+1}]$. Thus, Equation (A.10) evaluated in the Ito sense gives

$$\int_{t_0}^{t} W_s dW_s = \frac{W_t^2 - W_{t_0}^2}{2} - \frac{t - t_0}{2}. \tag{A.12}$$

The integral of a Wiener process includes an additional term compared to its deterministic counterpart.

A.1.4. Two equivalent definitions of SDEs

With the Ito integral defined in Equation (A.11), the integrals in Equation (A.7) is well defined,

$$X_t = X_{t_0} + \int_{t_0}^{t} f(X_t, t)dt + \int_{t_0}^{t} \sigma(X_t, t)dW_t. \tag{A.13}$$

It is easy to derive a simple numerical scheme to solve the equation above.

$$X_{t+\Delta t} = X_t + \int_t^{t+\Delta t} f(X_s, s)ds + \int_t^{t+\Delta t} \sigma(X_s, s)dW_s \tag{A.14}$$

$$\approx X_t + \int_t^{t+\Delta t} f(X_s, s)ds + \int_t^{t+\Delta t} \sigma(X_t, t)dW_s \tag{A.15}$$

$$= X_t + f(X_t, t)\Delta t + \sigma(X_t, t)(W_{t+\Delta t} - W_t), \tag{A.16}$$

or

$$X_{t+\Delta t} = X_t + f(X_t, t)\Delta t + \sigma(X_t, t)\Delta W, \tag{A.17}$$

where ΔW is the Wiener increment with mean 0 and variance Δt. The derived numerical scheme is called the the Euler-Maruyama scheme. This scheme for solving the SDE is easy to implement because the Wiener increment can be generated with a random number generator that obeys a Normal distribution and other components of this scheme are deterministic.
There are other ways to define a stochastic integral other than an Ito way given by Equation (A.11). For example, the Stratonovich definition of a stochastic integral gives

$$\int_{t_0}^{t} \sigma_s dW_s = \text{l.i.m.}_{\Delta t \to 0} \sum \frac{\sigma_{t_i} + \sigma_{t_{i+1}}}{2}(W_{t_{i+1}} - W_{t_i}). \tag{A.18}$$

Under this definition, Equation (A.9) gives

$$\int_{t_0}^{t} W_s dW_s = \frac{W_t^2 - W_{t_0}^2}{2}, \tag{A.19}$$

which gets rid of the left and right boundary point in contrast to the Ito integral result.
It is stated without proof that the SDE under Ito and Stratonovich definition have the following relationship. If a stochastic process X_t can be described by the SDE in the Ito sense as

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t, \tag{A.20}$$

then the same stochastic process can be described by the SDE in the Stratonovich sense as

$$dX_t = f(X_t, t)dt - \frac{1}{2}\sigma(X_t, t)\frac{\partial \sigma}{\partial x}(X_t, t)dt + \sigma(X_t, t)dW_t. \tag{A.21}$$

In the other way around, if a stochastic process X_t can be described by the SDE in the Stratonovich sense as

$$dX_t = f(X_t, t)dt + \sigma(X_t, t)dW_t, \tag{A.22}$$

then the same stochastic process can be described by the SDE in the Ito sense as

$$dX_t = f(X_t, t)dt + \frac{1}{2}\sigma(X_t, t)\frac{\partial \sigma}{\partial x}(X_t, t)dt + \sigma(X_t, t)dW_t. \tag{A.23}$$

A.1.5. Stochastic differentiation

With mean square convergence defined in the previous section, we can have a formal definition of derivative of a stochastic process by replacing the normal limit with the difference quotient convergent to zero in the mean square sense. This will not be the scope of the present discussion. Now introduce the Ito lemma that gives the formula to compute the the derivative of a function of a stochastic process. Suppose $\{X_t, t \in T\}$ is a random process that is governed by Equation (A.6). Let $g(x,t)$ be a sufficiently smooth function. The derivative of $g(X_t, t)$ is the following SDE

$$dg_t = \frac{\partial g}{\partial t}dt + \frac{\partial g}{\partial x}dX_t + \frac{1}{2}\sigma^2\frac{\partial^2 g}{\partial x^2}dt. \tag{A.24}$$

The Ito lemma enables us to derive the governing SDE of the derivative of a function of stochastic process or verify the analytical solution of an SDE. For example, consider a exponential function $g(t) =$

e^{bt} with a constant b. The differentiation of the function g, along with an appropriate initial condition, forms an ordinary differential equation

$$\frac{dg}{dt} = bg, \quad g(0) = 1. \tag{A.25}$$

Now we are interested in the SDE that governs the temporal derivative of $g(W_t, t) = e^{bW_t}$. To apply Ito lemma, we compute

$$\frac{\partial g}{\partial t} = 0, \frac{\partial g}{\partial x} = bg, \frac{\partial^2 g}{\partial x^2} = b^2 g.$$

The Ito lemma gives

$$dg_t = bg_t dW_t + \frac{b^2}{2} g_t dt. \tag{A.26}$$

Note that an additional dt pops up. Furthermore, to transform the Ito SDE to a Stratonovich one, by following Equation (A.21) one derives

$$dg_t = bg_t dW_t. \tag{A.27}$$

Note that the Stratonovich SDE (A.21) mimics a deterministic differential equation, so as a Stratonovich integral mimics a deterministic integral in Equation (A.19).