Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Steinar Brattøy Gundersen

Implementing and reviewing a published cellular automaton model for simulating erosion and deposition processes due to turbidity currents

Master's thesis in Applied Physics and Mathematics
Supervisor: Tor Nordam and Raymond Nepstad
June 2019

NTNU
Norwegian University of
Science and Technology

Steinar Brattøy Gundersen

Implementing and reviewing a published cellular automaton model for simulating erosion and deposition processes due to turbidity currents

Master's thesis in Applied Physics and Mathematics
Supervisor: Tor Nordam and Raymond Nepstad
June 2019

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Abstract

The mining industry has, for a long time, used the sea for tailings placement. In these disposal sites, turbidity flows commonly occur as a result of the tailings discharge, in which particles may be transported some distance away from the discharge point. As the discharge can contain environmentally harmful particles, it is desirable to be able to predict the interaction between the turbidity flow and the seabed. Motivation is also given by the ability to predict areas of erosion and deposition. The main objective in this thesis has been to review and implement a published cellular automata (CA) model. During the implementation, it became clear that the published description was not complete, and that it contained some errors. Thus the goal became to implement and test as much as possible of the model, creating a complete and correct description of the implementation.

To obtain an efficient implementation, selected parts of the code was written in Cython, to avoid the Python overhead associated with e.g., for-loops. Additionally, the code was parallelized using the Message Passing Interface (MPI) standard. The halo-exchange technique was combined with a self-composed algorithm (dubbed as the inverse-halo-exchange) in order to parallelize the CA. The program achieved superlinear speedup for tests conducted with up to about 64 CPUs, indicating that CAs are well suited for parallelization.

Some numerical simulations have been run, including a simplified case with Ranfjorden as bathymetry, where the results to some extent reflect the expected behavior of a turbidity current. Some simulations display signs of numerical instability. Potential causes and solutions are discussed.

Sammendrag

Gruveindustrien har i lengre tid benyttet seg av sjødeponi for lagring av “tailings”. I slike sjødeponi er det vanlig at turbiditetsstrømmer oppstår som følge av utslippet, hvilket medfører at partiklene fraktes en lengre distanse unna utslippspunktet. Blant annet på grunn av at utslippet kan inneholde miljøskadelige partikler, er det ønskelig å kunne predikere interaksjonen mellom turbiditetsstrømmen og havbunnen. Samtidig er det ønskelig å kunne forutsi hvilke områder som blir erodert og i hvilke områder sediment blir deponert. Hovedformålet med prosjektet har vært å gjennomgå og implementere en publisert cellular automata (CA) modell. Under implementasjonsprosessen, ble det klart at modellen ikke var fullstendig, og at den inneholdt noen feil. Slik ble målet å implementere så mye som mulig av denne modellen, og lage en fullstendig og korrekt beskrivelse av implementasjonen.

For å oppnå en effektiv implemetasjon, ble utvalgte deler av koden skrevet i Cython. Dette for å unngå “overhead” som Python induserer i for eksempel for-løkker. I tilegg, ble koden parallelisert ved å bruke standarden “Message Passing Interface” (MPI). En teknikk kalt “halo-exchange” ble benyttet i kombinasjon med en selvkomponert algoritme (som ble navngitt “inverse halo-exchange”) for å parallelisere CAen. Programmet oppnådde superlinær speedup i tester med opptil omtrent 64 CPUer, hvilket indikerer at en CA er velegnet til parallelisering.

Noen numeriske simuleringer har blitt kjørt, inkludert en forenklet simulering med Ranfjorden som batymetri. Resultatene reflekterer til en viss grad den forventede oppførselen til en turbiditetsstrøm. I noen tilfeller viser simuleringene tegn til numerisk ustabilitet. Potensielle årsaker og løsninger til dette blir diskutert.

Preface

This thesis is the result of a rather arbitrary choice I made last year. I had researched the available thesis proposals at the NTNU website, and I was determined I wanted to work within the field of numerical physics. One of the proposals that caught my eye was by Tor Nordam and was titled “Parallel simulations of transport and mixing in the ocean”. However, as I made an inquiry about the project, it became clear that this proposal was from a previous year, and had already been explored by another student. New ideas were spun. Tor mentioned he had a collegue at SINTEF Ocean that had been exploring the idea of simulating turbidity currents with a cellular automaton (CA). I already had some experience working with CAs from my classes in parallel computation, and so I was intrigued by the idea. Thus, I was introduced to Raymond Nepstad, and the three of us started outlining a thesis proposal. This random choice has led down a path with seemingly endless hours of programming and debugging, and a roller-coaster of ups- and downs; exactly what I had in mind. Yes, debugging in the late a.m. can be tough, but the view from the summit makes it worth it. I am grateful to Tor and Raymond for letting me experience this journey, and for always lending counsel when I’ve reached a “particularly steep hill”.

I would like to thank Tristan Salles, the author of the cellular automaton model upon which this thesis is based. The paper describing the model was written in 2007. Despite it being a long time ago, Salles responded to my emails, and answered my questions to the best of his abilities. For that I am grateful.

I want to thank my family and friends for supporting me, and keeping me sane while pursuing this degree. In particular, I would like to thank my brother and housemate Magnus Brattøy Gundersen for tolerating my lack of participation in house cleaning, during the final hours of this thesis. I would also like to thank my friend and fellow student Mauhing Yip, for reminding me to use the correct formatting for this thesis.

Steinar Brattøy Gundersen
Trondheim, Norway
June 2019

Contents

1 | Introduction

Ranfjorden [1] is a fjord of length 68 km located in northern Norway. The fjord runs narrowly, in a north-easterly direction into the country before it “bottoms out” near the town of Mo i Rana. Through the town runs the Rana River before connecting with the fjord. At its deepest, the fjord reaches 525 m. As is the case with several other Norwegian fjords, e.g., Bøkfjorden and Frænfjorden, this fjord is used as a means of depositing so-called mine tailings.

Tailings is a common term for the waste product of the mining industry. Commonly used ore refinement processes rely on chemical reactions for extracting the mineral. In such a process, the ore is milled into fine particles and chemicals are added to extract the desired mineral. The material left over from this process is a slurry of different sized particles, which is referred to as tailings. For the projected Nussir copper mine in Finnmark, northern Norway, the ore grade is estimated at 1.15 %, implying that the majority ($\sim 99\,\%$) of the mined material is considered as waste [2].

The disposal of mining tailings is a problem of major environmental and financial concern. On a world basis, the most commonly used method for tailings disposal involves land-based permanent storage. This is usually in the form of dedicated dams. There are several potential risks to using dams as storage, the most typical examples being dam failure, or contamination of the surrounding area from chemicals seeping through the dam. Consequences of such risks are severe, as the environmental impact may last anywhere from 100-1000 years [3]. As land-based storage methods require a large area, competition with agriculture, residential areas, and other space intensive activities may complicate matters on the financial side.

The potentially devastating risks associated with dams has led to a search for alternative disposal methods, and the discharge of tailings in the sea. This type of discharge may be divided into three categories: coastal shallow-water disposal (CTD), submarine tailing disposal (STD) and deep-sea tailing placement (DSTP). As the names imply, these categories are mainly differentiated by the depth at which the tailings end up. In CTD, tailings are disposed near the water surface. This category includes riverine discharge methods. In STD and DSTP, tailings are discharged by an underwater pipe at some depth beneath the surface. For STD the outlet depth is < 100 m, and for DSTP the depth is > 100 m [3]. Riverine discharge used to be a popular method, before environmental impacts were known.

In present Norway, the primary methods for tailings placement are STD and DSTP. These discharge methods induce a phenomenon known as turbidity currents. Turbidity currents are phenomena that also occur naturally, in which a current consisting of particles and water are propelled due to density differences. These currents may redistribute sediment from the sea floor through erosion and deposition. Put in context of STDs and DSTPs, turbidity currents are induced by the fine particles contained in the tailings.

As the chemicals contained within the tailings are distributed by the turbidity current, there is some interest in predicting the location of such deposits, and the evolution of the bathymetry in general.

The ambition for this thesis has been to review and implement a published cellular automaton (CA) model capable of simulating a turbidity current, and predict erosion and deposition. The model in question is due to T. Salles et al. [4]. In Ref. [4], the model is applied to various bathymetries, and several figures illustrate their time evolution due to the turbidity current. The provided examples indicate that the model is well suited for studying turbidity currents due to tailings discharges.

During the work on this thesis, it became clear that all details of the model are not completely described in Ref. [4], and that some of the equations contain errors. Hence, the goal has been to fill in the blanks, and implement and test as much as possible of the model, creating a complete and correct description of the implementation. To achieve good computational efficiency, the implementation has been parallelized using the message passing interface (MPI) standard, and Cython was used to gain additional speedup. Some tests have been carried out, revealing that the model may become numerically unstable in some cases, and still has some errors. What these errors may be, and how to handle the instabilities, is discussed.

This thesis is divided into chapters. In Chapters 2 and 3, some preliminary theory will be introduced, and in Chapter 4 information regarding implementation of the cellular automaton will be provided. This is followed by Chapter 5 in which the CA transition function is verified and discussed. Further testing and verification of the CA is performed and discussed in Chapter 6, followed by a numerical simulation performed on the case Ranfjorden in Chapter 7. The numerical performance of the implementation is presented and evaluated in Chapter 8, and a final discussion and conclusion is given in Chapters 9 and 10, respectively.

2 | Theory

The theory presented in this chapter provides some fundamental information about turbidity currents and cellular automata. This information will be referenced in later chapters, when discussing implementation and results. The reader may want to skip this chapter at first, and refer back to it as needed.

A turbidity current is a special case of the more general phenomena known as a gravity current. In Section 2.1, the characteristic features of a turbidity current will be explored. In Section 2.2 the cellular automaton will be defined, and some notation and terminology is presented. Lastly, in Section 2.3 some useful concepts and equations are defined, to be used for reference in subsequent chapters.

2.1 Gravity and turbidity currents

Gravity currents are mainly horizontal flows induced by differences in density, i.e., when a fluid of density ρ_c flows in a fluid of density $\rho_a < \rho_c$. Gravity currents are phenomena commonly experienced in everyday life. As an example, consider a "cube" of water being released on a horizontal plane. The water has a density much higher than the surrounding air. Thus, we have $\rho_{water} = \rho_c > \rho_{air} = \rho_a$ inducing a gravity current, causing the water to spread [5].

2.1.1 Sediment gravity flows

When the current fluid consists of a mix of (heavy) particles and a fluid identical to the ambient fluid, the gravity current is known as a particle-driven current, or a sediment gravity flow [5]. These flows are commonly encountered in the ocean, where gravity acts on the suspended particles, inducing the motion of the current. The particles can be kept from settling on the sea bed by some "support mechanisms", as will be elaborated below. Keeping the particles in suspension keeps the gravity current from terminating.

The particle support mechanisms, i.e., the phenomena keeping the particles in suspension, are used to distinguish between four types of sediment flow. The support mechanisms and corresponding type of sediment flow is illustrated in Figure 2.1. Turbidity currents are sediment flows, in which the particles are mainly supported by the upward velocity component of the turbulence [6].

The volumetric concentration of particles, is also commonly used to distinguish between different types of sediment flows. Following the nomenclature of Mulder and Alexander in Ref. [7], flows with volumetric concentration of sediments Q_{cj} are named according to Eq. (2.1).

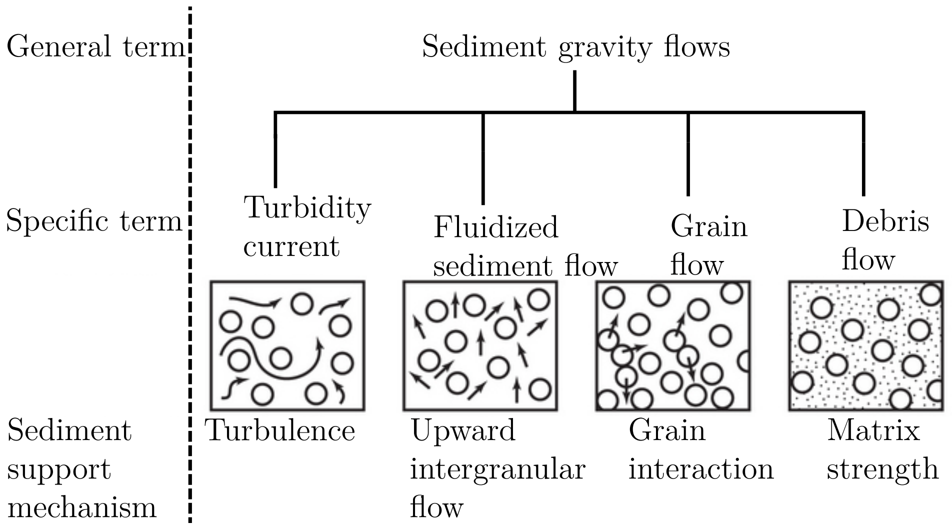


Figure 2.1: Diagram illustrating four types of sediment gravity flows and their support mechanisms. Adapted from [6].

$$\text{Sediment flow} \rightarrow \begin{cases} \text{Turbidity current} & \text{if } Q_{cj} \leq 0.09 \\ \text{Concentrated density flows} & \text{if } 0.09 < Q_{cj} < 0.4 \\ \text{Hyperconcentrated density flows} & \text{if } Q_{cj} \geq 0.4. \end{cases} \tag{2.1}$$

In this thesis, let a turbidity current at most consist of 9 % sediment per unit volume. At this concentration the interaction between sediment particles may be neglected [8].

Turbidity currents can be triggered by an avalanche or earthquake, in which case they are known as surge flows, or may be due to continuous sediment input from e.g., a river mouth or the discharge of mine tailings. The cause of the flow may determine its shape. A surge type turbidity current typically consists of a solitary “ball” of particles, while a more continuous flow typically consists of a head, a body and a tail. The head precedes body, and typically has a larger height than the succeeding parts. Particles deposited by a turbidity current form what is known as turbidites.

2.1.2 Interactions with the ambient

In the marine environment, a turbidity current interacts with the sea bed and the ambient fluid through several processes, two of which are entrainment and sediment deposition. These processes will be defined below.

Formally, entrainment refers to the process in which the volume and density of a turbidity current is affected by viscous effects [5]. These effects are acting on both the upper current interface and at the sea bed interface. At the upper interface, entrainment may be considered a process in which the current fluid is mixed with the ambient fluid. At the lower interface, entrainment refers to the process in which sediment is lifted from the sea bed into the current, i.e., erosion.

Considering first the upper interface, this effect is called water entrainment. Garcia and Parker found in 1986 [9] that the coefficient of water entrainment e_w could be approximated by,

$$e_w = \frac{0.075}{\sqrt{1 + 718\mathrm{Ri}^{2.4}}}, \tag{2.2}$$

where Ri is the Richardson number. Here $\mathrm{Ri} = g' Q_{th}/Q_v^2$, for a turbidity current with a reduced gravity g' (see Section 2.3.2), a thickness of the current Q_{th}, and speed Q_v [4].

Erosion and deposition are the processes describing mass exchange between the sea bed and the current flow. These processes are described by the Exner equation [10], which is the conservation equation of mass in the bed and the current. The Exner equation is given by

$$(1 - \gamma)\frac{\partial z}{\partial t} = -\nabla \cdot \mathbf{q_s}, \tag{2.3}$$

where z is the bed elevation, t is time, γ is the bed porosity, and $\nabla \cdot \mathbf{q_s}$ is the divergence of the sediment flux.

Erosion rate

In a turbidity current with speed Q_v and N_j different types of particles, each with a density ρ_j, the erosion rate can be approximated by $E_j v_{sj}$ [11]. Here, v_{sj} is the settling velocity for particle type j (see Section 2.3.5) and E_j is a sediment entrainment coefficient. Garcia and Parker found in 1993 [12] that the sediment entrainment coefficient E_j, could be approximated by

$$E_j = a\frac{Z_{eff}^5}{1 + \frac{a}{0.3}Z_{eff}^5}. \tag{2.4}$$

Here a is a constant equal to $1.3 \cdot 10^{-7}$, and

$$Z_{eff} = \lambda Z_u = \lambda \frac{u_*}{v_s} f(R_{pj}) \tag{2.5}$$

with $u_* = \sqrt{Q_v^2 c_D}$ being the shear-velocity of the density flow [11]. The shear-velocity u^* is here defined differently than in Ref. [4], where $\tilde{u}^* = Q_v c_D$ is used. c_D is a bed friction coefficient, and $\lambda = 1 - 0.228\sigma$, where

$$\sigma^2 = \sum_{j=1}^{N_j} (\varphi - \bar{\varphi})^2, \tag{2.6}$$

is the variance of the particle size D_{sj}, on the φ scale. The relation $\varphi = -\log_2 d_{sj}$ is used for conversion to and from the φ scale. Note that d_{sj} is the particle size in mm, while D_{sj} is given in m. The $\bar{\varphi}$ denotes the arithmetic average of the particle sizes on the φ scale.

Lastly the function f and the particle Reynolds number for particles of type j, R_{pj}, are given by

$$R_{pj} = \sqrt{\frac{g(\rho_j - \rho_a) D_{sj}}{\rho_a}} \frac{D_{sj}}{\nu}, \tag{2.7}$$

$$f(R_{pj}) = \begin{cases} R_{pj}^{0.6} & \text{if } R_{pj} \geq 3.5 \\ 0.586 R_{pj}^{1.23} & \text{if } 0 < R_{pj} < 3.5, \end{cases} \tag{2.8}$$

where ν is the kinematic viscosity of water. Note that the domain of f is expanded here in comparison with the definition given in Ref. [4] and [12], in which f is undefined for $R_{pj} < 1$.

2.1.3 Near-bed concentration

In 1993, Garcia found [13] a relation describing the near-bed concentration c_{nbj} for a turbidity current with poorly sorted sediment. Poorly sorted sediment implies that the variance of the particle sizes is large, while for well sorted sediment the variance is small [14]. The near-bed concentration is given by,

$$\frac{c_{nbj}}{Q_{cj}} = 0.40 \delta_j^{1.64} + 1.64, \qquad \text{with } \delta_j = \frac{D_{sj}}{D_{sg}}. \tag{2.9}$$

Here Q_{cj} is the layer-averaged volume concentration, and D_{sj} is the diameter of sediment type j in the turbidity current. The geometric mean size of the suspended sediment, D_{sg}, is defined as

$$D_{sg} = \left(\prod_{j=0}^{N_j} D_{sj}^{Q_{cj}} \right)^{1/\sum_{j=0}^{N_j} Q_{cj}}, \tag{2.10}$$

for a turbidity current with N_j particle types.

2.1.4 Numerical models and self-accelerating turbidity currents

The three-equation model (TEM) and the four equation model (FEM) denote two groups of models often used for modeling turbidity currents. These models were developed in the 1980s by Fukushima et al. [15], and have since then spawned several modified variants [16]. At its conception, the TEM was believed to be insufficient for adequately modeling turbidity currents, due to a violation of the turbulent kinetic energy balance, giving rise to the FEM. However, it has later been shown that the TEM does indeed give meaningful results, when the parameters of the current fulfill certain criteria called “ignition criteria”. When these criteria are met, TEM predicts the formation of “self-accelerating” turbidity currents. A turbidity current with velocity Q_v and sediment transport rate ψ, is said to be self-accelerating if these values increase downstream without limit after a finite downstream distance [16], i.e., $\lim_{x\to\infty} Q_v, \psi = \infty$, if x is the downstream coordinate.

The TEM scope is limited to the turbidity current itself, and does not include the evolution of the sea bed. The TEM may be used to resolve the evolution of the thickness Q_{th}, speed Q_v and sediment transport rate ψ of the current. Note that the volumetric sediment concentration of the current Q_{cj} is related to these variables through

$$Q_{cj} = \frac{\psi}{Q_{th} Q_v}. \tag{2.11}$$

2.2 Cellular Automata

A cellular automaton (CA) can be considered a time-evolving uniform grid G of cells. Each cell has a state S, and a set of neighbors N [17]. The number of possible states and the number of neighbors may be arbitrarily chosen. In a Cartesian grid, the cell neighborhood is typically the northern, southern, eastern and western neighboring cells, i.e., a von Neumann neighborhood. In Conway's "Game of Life" CA, the cells are either "dead" or "alive" [18], meaning that there are two possible cell states.

Characteristic for a CA is that the time-evolved state of a cell is determined locally, i.e., by considering the neighboring cells' states. In a CA, time-evolution is discretized. For each time step, a cell evolves by using a specified set of rules. Through these rules the neighbor states affect a cell. In a CA with binary states, one such rule could be "any cell with "state 1" with fewer than two "state 1" neighbors evolves to a "state 0" cell". This rule is an example from Conway's "Game of Life". These "time evolution rules" constitute a transition function σ.

The laws of a CA system are said to be local and uniform. Local because the state of a cell is determined by its neighboring states. And uniform because the laws are the same for all cells [17]. Thus a cellular automaton may be defined by:

$$< G, N, S, P, \sigma >,$$

with G defining the grid of cells, N defining the neighborhood of a cell, S defining the states of a cell, and σ defining the transition function of the CA. P is a set of parameters that may be input to the CA. These parameters can be physical constants, initial conditions, source terms, or other specified parameters. Cells that are on the edge of the grid G, are called border cells.

In using a CA to simulate a physical phenomenon, in particular the state S may be very complex. By complex, it is meant that the state of a cell may describe several physical quantities, e.g., a density, a height and an energy value. Thus, the state S is in fact a "combination" of several substates. For a set of possible density values ρ, and a set of possible height values h, the cell state S is the Cartesian product [19] of the substates: $S = \rho \times h$.

In light of this information on substates, one may make a distinction between the update rules in the transition function σ. A *local interaction* I is an update rule which involves a cell's neighbors. Local interactions are the most common update rules for CAs with non-complex states S. However, when the cell state of a CA consists of several substates, an update rule could use a substate A to update another substate B of the same cell. Update rules that do not involve neighboring cell states are called *internal transformations* T.

An update rule X, that uses a substate A to update a substate B is denoted

$$X : A \to B.$$

An update rule provides the *next iteration* of the substate. To differ between old and new iterations of a substate, superscripts are used, e.g., $A^{(n+1)}$ is the next iteration of $A^{(n)}$.

Perhaps somewhat confusingly, superscripts are also used in the notation $X : A \to B$, to indicate the number of substates which enter the update rule. For, e.g., a hexagonal grid, $Y : A^6 \to B$, may indicate that all six neighbors' substate A values are used to update the center cell B substate. Observe however, the use of parenthesis when used to indicate iterations.

2.2.1 Hexagonal grids

In this section some theory on hexagonal grids is presented. The motivation of this is the intention of using these as G in a cellular automaton. When working with hexagonal grids, one prominent issue is the choice of indices, i.e., how do you refer to a specific hexagon in the grid? There are several conventions for index labeling. What convention is best suited will depend on what operations are to be performed on the grid. One way to label the hexagons is illustrated in Figure 2.2A and B. With this convention, a grid of hexagonal cells can easily be stored using a 2-dimensional array.

It will be useful to be able to convert between the indices $[l, m]$, and the spatial coordinates (x, y) of a hexagon in the grid. The neighborhood of a hexagon in a hexagonal grid is isomorphous, i.e., we have a 6-fold symmetric neighborhood. This means that if the center-to-center distance between two neighboring hexagons is Δx, we can find the relation between indices and coordinates by using the blue triangle in Figure 2.2A. With $a = \Delta x/2$, and $c = \Delta x$, b is determined to be $\frac{\sqrt{3}}{2}\Delta x$. Thus, if $(l = 0, m = 0)$ corresponds to $(x = 0, y = 0)$, one finds

$$x = \Delta x(m + \frac{1}{2}l) \tag{2.12}$$

$$y = l\frac{\sqrt{3}}{2}\Delta x. \tag{2.13}$$

In Figure 2.2B a convenient way of referring to a cell's neighbors is shown. Using the ordering 1-6 as shown, the k'th neighbor of a cell is defined by this value. The k values are valid for cells away from the domain boundary. Cells at the grid boundary do not

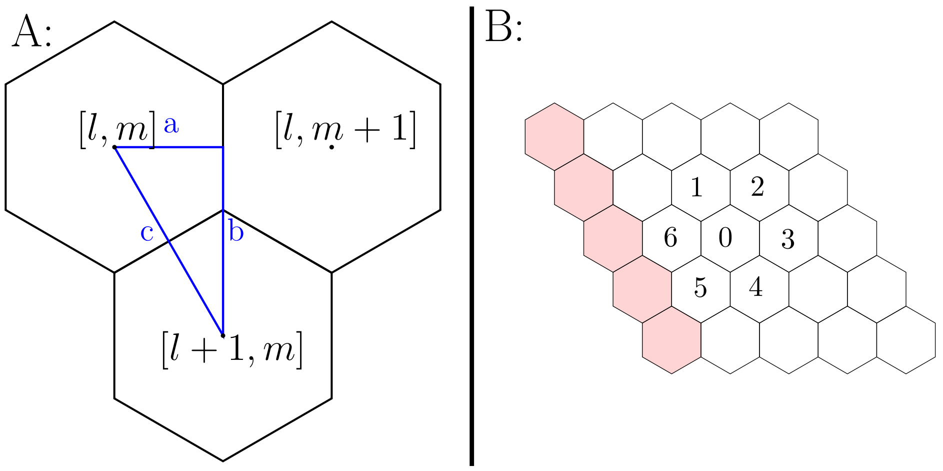


Figure 2.2: A: The figure illustrates one indexing convention for hexagonal grids. The blue triangle (edges a,b,c) illustrates geometrical length scales in the hexagonal grid. B: Using the indexing convention in subfigure A, the cells in a horizontal line share the first index, and the shaded hexagons share the second index. The k'th neighbor of a center cell 0 is defined according to the ordering 1-6 shown in the figure.

have certain neighbors, and will possess fewer valid k values. Table 2.1 indicates the relative array indices of a center cell and its neighbors.

Table 2.1: For some center cell 0 with indices $[l, m]$, the relative array indices for neighbor k is given by this table.

k	Relative indices of cell
1	$[l-1, m]$
2	$[l-1, m+1]$
3	$[l, m+1]$
4	$[l+1, m]$
5	$[l+1, m-1]$
6	$[l, m-1]$

2.3 Concepts and equations related to transition function σ

This section defines relevant equations and concepts which will be needed to define the CA update rules for modeling of turbidity currents.

2.3.1 Angle of repose

The angle of repose is a concept that enters when considering the geometry of a pile of e.g., sand. With the pile of sand in mind, the (critical) angle of repose θ_r, is the steepest angle the pile can make with a horizontal plane. Adding more sand to the top of the pile will result in an avalanche, in which mass is redeposited such that the radius of the pile increases [20]. See Figure 2.3.

The physical properties of the material being stacked, determines the value of θ_r. A typical example used to gain an intuition for the topic is how wet sand can be used to make sandcastles, while dry sand cannot. Thus, wet sand must have a larger θ_r, in comparison to the dry sand.

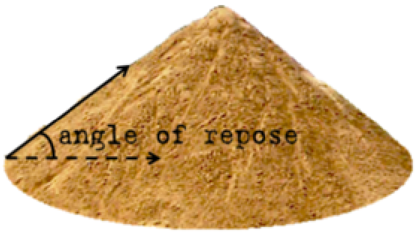


Figure 2.3: The figure illustrates the concept of the angle of repose θ_r.

2.3.2 Reduced gravity

The reduced gravity g', is a concept relevant to gravity currents. For a two-fluid system it can be defined [5] as,

$$g' = g\frac{\rho_c - \rho_a}{\rho_a}, \tag{2.14}$$

where g is the gravitational acceleration, and ρ_c and ρ_a, is the density of the current and the ambient fluid, respectively. For a particle flow consisting of N_j different types of particles, each with a corresponding density of ρ_j, and a volumetric concentration of Q_{cj}, the current density may be defined as

$$\rho_c = \sum_j^{N_j} Q_{cj}\rho_j + (1 - \sum_j^{N_j} Q_{cj})\rho_a. \tag{2.15}$$

Inserting Eq. (2.15) in Eq. (2.14) yields

$$g' = g\sum_{j=1}^{N_j} Q_{cj}\frac{\rho_j - \rho_a}{\rho_a}. \tag{2.16}$$

2.3.3 Numerical stability

A partial differential equation (PDE) can be solved numerically through spatial and temporal discretization. There are several ways to discretize the terms in a PDE, and there are varying advantages and drawbacks to most of them. The term numerical scheme refers to a method of discretizing a PDE.

The amplification factor G for a numerical scheme can be defined as $G \equiv \frac{\hat{\varphi}^{(k+1)}}{\hat{\varphi}^{(k)}}$, where $\hat{\varphi}^{(k)}$ is any Fourier mode of the numerical solution of the PDE at time step k. For this numerical scheme to be stable [21], G must fulfill

$$G \equiv \frac{\hat{\varphi}^{(k+1)}}{\hat{\varphi}^{(k)}} \leq 1 + K\Delta t \tag{2.17}$$

for a constant K. Δt is the time discretization. This condition on G will, for certain discretization schemes involving explicit time integration, lead to the CFL condition:

$$C \equiv \frac{u\Delta t}{\Delta x} \leq C_{max}, \tag{2.18}$$

where u is some characteristic speed in the PDE, and Δx is the spatial discretization. The value of C_{max} depends on the discretization scheme [22].

2.3.4 Run-up height

When a turbidity current of thickness Q_{th}, encounters an obstacle with a height $H > Q_{th}$, then by conversion of kinetic to potential energy, the turbidity current thickness will at most rise by an amount

$$h = \frac{1}{2}\frac{Q_v^2}{g'}, \tag{2.19}$$

where Q_v is the current speed, and g' is the reduced gravity of the flow [23, 4]. The run-up height r is thus defined as the maximum attainable height of the turbidity current:

$$r = Q_{th} + h. \tag{2.20}$$

2.3.5 Sphere settling velocity

The terminal velocity of a sphere v_s (also known as the sphere settling velocity) in laminar flow can be found by equating Stokes' law [24] for drag, and the net gravitational force. This yields an equation which can be solved for v_s:

$$v_s = \frac{1}{18}\frac{g'}{\nu}D_s^2. \tag{2.21}$$

Here ν is the kinematic viscosity of the ambient fluid, g' is the reduced gravitational acceleration and D_s is the diameter of the particle.

For larger particles, flow around the sphere is not laminar, and T. Salles suggests in [25] the following formula for calculating the terminal velocity of a sphere of diameter D_s in a turbidity current:

$$v_s = \begin{cases} \frac{1}{18}\frac{g'}{\nu}D_s^2. & \text{if } D_s \leq 100\,\mu\text{m} \\ \frac{10\nu}{D_s}\left[\sqrt{1+\frac{0.01g'D_s^3}{\nu^2}}-1\right] & \text{if } 100\,\mu\text{m} \leq D_s \leq 1000\,\mu\text{m} \\ 1.1\sqrt{g'D_s} & \text{if } D_s \geq 1000\,\mu\text{m} \end{cases} \tag{2.22}$$

Eq. (2.22) is derived from Van Rijn's formula for the settling speed for natural sand [26]. Inserting the settling velocity into the following equation by W. Dietrich [27] gives the dimensionless settling velocity $\widetilde{v}_s$,

$$\widetilde{v}_s = \frac{\rho_a v_s^3}{(\rho_j - \rho_a)g\nu}. \tag{2.23}$$

Figure 2.4 shows the sinking speed as calculated by Eqs. (2.22) and (2.23), for a range of particle diameters D_{sj}, using particle density $\rho_j = 2600\,\mathrm{kg/m^3}$, water density $\rho_a = 1000\,\mathrm{kg/m^3}$ and $\nu = 1 \cdot 10^{-6}\,\mathrm{m^2/s}$.

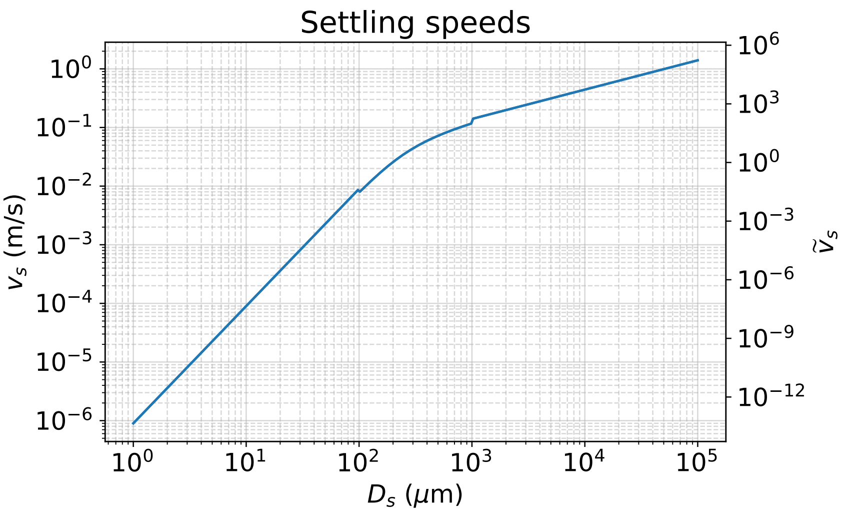


Figure 2.4: This figure displays the sinking velocity v_s for a sphere of various diameters. v_s is calculated by Van Rijn's formula, Eq. (2.22), and the dimensionless speed $\widetilde{v}_s$ is by Dietrich Eq. (2.23).

2.3.6 Chézy formula and speed of a turbulent current

Antoine Chézy is credited for finding a formula for the speed U of water flow in an open channel [10]. Let a portion of the channel have length L, height h and width ℓ. By equating the parallel component of the gravitational force with a frictional force, and solving for the speed U, Chézy found,

$$U^2 = \frac{2g}{C_f}\frac{V}{A}\sin\theta. \tag{2.24}$$

Here C_f is some frictional coefficient, g is the gravitational acceleration, $V = L \cdot h \cdot \ell$ is the volume of the fluid, $A = L(2h + \ell)$ is the area of the channel in contact with the fluid and θ is the angle of the channel with respect to the horizon [25]. The ratio V/A is known as the hydraulic channel radius, and for channels satisfying $h/\ell \to 0$, it can be approximated by $V/A \approx h$. Combining this with the small-angle approximation applied to $\sin\theta$, yields $U = C\sqrt{h\theta}$, where $C = 2g/C_f$. Using this, Middleton [28, 29] derived a formula suited for use in turbidity currents,

$$U = \sqrt{\frac{8g'}{f_0 + f_i}hs}, \tag{2.25}$$

where g' is the reduced gravity given by Eq. (2.16), $f_0 \approx 0.04$ is the Darcy-Weisbach constant, f_i is the upper interface friction coefficient, h and s are the thickness and slope of the channel, respectively. For sediment flows with N_j sediment types, each sediment having a concentration of Q_{cj}, Salles et al. suggests [4] using Eq. (2.25), with two alterations:

1. $(f_0 + f_i) \to f_0(1 + a)$, where $a \approx 0.43$ is an emperical constant.
2. $U \to U/\sqrt{\varphi}$, where $\varphi = \sum_{j=1}^{N_j} Q_{cj}$ is the concentration of all N_j sediment types.

Note that φ here is unrelated to the prveiously described φ scale for the particle size (Section 2.1.2.

3 Python programming

In this chapter some information regarding implementation in the programming language Python 3 is presented. The contents of this chapter mainly focuses on reducing the run time of Python code. Section 3.1 contains information about the Python library Cython, and Section 3.2 contains information about parallelizing Python programs.

3.1 Cython

This section contains some information about the Cython library for Python. Motivation for using Cython will be presented along with an example illustrating the superior performance of Cython loops.

In essence, Cython is a library for Python that can be used to speed up Python code. This is done by translating the Cython code into C/C++ and compiling it into Python extension modules [30]. The Cython language is a mix of Python and C/C++, and the source files usually have the extension `.pyx`. Pure Python can be translated and compiled, but the potential speedup gained would not necessarily be that good. The strength of the Cython language is the ability to specify explicit C type declarations to variables.

3.1.1 Why use Cython?

There are several factors motivating the use of Cython. One example is the Python implementation of low-level computational loops. The performance of for-loops in Python suffers due to the dynamic nature of the language [30]. Making use of the explicit type declarations, Cython provides the possibility of running for-loops at C speed. In Figure 3.1 two snippets of code are shown, each containing a function that adds all integers from 0 up to the input variable `x`. The function `test_py` is a pure Python function, and `test_cy` is a Cython function showcasing the use of type declarations.

```
def test_py(x):
    a = 0.0
    for i in range(x):
        a += i
    return a
```

```
def test_cy(int x):
    cdef int i = 0
    cdef int a = 0
    for i in range(x):
        a += i
    return a
```

Figure 3.1: The figure displays two functions performing the same operation to produce a result. The function test_py is a pure Python function, and the function `test_cy` is a Cython function.

The following example illustrates the performance increase due to adding type declarations to variables in Cython. Consider now that a matrix of size 1000×1000 is to be traversed by a for-loop. For simplicity, assume that this is equivalent to running either `test_py` or `test_cy` with $x = 10^6$. Then, `test_py(x)` and `test_cy(x)` have runtimes of approximately 31.2 ms and 56.4 µs, correspondingly. This means the Cython function is about 553 times faster than the Python function. Although both of the runtimes may seem small, consider running a simulation on a grid where cell data is stored in 7 matrices with size 1000×1000, and that those are traversed e.g., 10000 times during the simulation. The computational time then becomes 36 min for the Python version and 4 s for the Cython version.

As mentioned above, the Cython compiler accepts pure Python code. This means that during the development phase of a program, one can use Python as usual, then once functionality is in place, one can apply Cython to computationally intensive parts of the code. Using Cython to speed up a small (but frequently used) section of code can make a noticeable difference in runtime. As the Pareto Principle states: 80 percent of the runtime is spent in 20 percent of the code [30].

3.1.2 Compatibility

Cython is compatible with NumPy and message passing parallelism (see Section 3.2). Cython is able to directly manipulate the memory buffers of NumPy arrays by using “memoryviews”, removing the Python overhead from the runtime in Cython. This allows efficient use of NumPy arrays both in the Python part and in the Cython part of the code.

Note that for-loops can often be avoided by using vector operations when working with NumPy arrays. The NumPy vector operations are written in optimized C code, thus eliminating the need for Cython in most cases. In comparing runtimes, a NumPy operation and the corresponding Cython code should yield about the same result. However, there are situations where NumPy operations are less memory efficient than the corresponding Cython code [30], or an algorithm is difficult to write without using for-loops.

3.2 Parallel computing

In this section some basic concepts and relevant theory [31] on parallelization of computer programs will be introduced.

Modern computers are usually equipped with a multicore processor, i.e., a processor consisting of several CPUs or cores. The CPUs exchange information with the computer’s memory to do computations. In parallel computation, one distinguishes between shared- and distributed memory systems. The main difference between the two is the way the cores are connected to memory. In shared memory systems, the CPUs are all connected to a common memory trough an interconnect, and may access all data locations. In a distributed memory system, each core has its own private memory, and a core – memory pair can be referred to as a process. As each process has its own private memory, information from another process cannot be accessed directly.

A serial program is a program running on one core (or thread). Serial programs may be parallelized by splitting the work load between several processes. Doing so usually leads to reduced computation time and performance increase.

3.2.1 Performance

The speedup s of a program is a measure of how much faster a parallelized program is when compared to the serial program. Speedup may be defined as

$$s = \frac{t_{serial}}{t_{parallel}}. \tag{3.1}$$

Where t_{serial} and $t_{parallel}$ is the serial and parallel computation time, respectively. Further, these values may be expressed by,

$$t_{serial} = \sigma + \varphi \tag{3.2a}$$

$$t_{parallel} = \sigma + \varphi/p + \kappa, \tag{3.2b}$$

where σ denotes the inherently sequential part of the program, φ is the parallelizable part and p is the number of CPUs used. κ denotes the parallelization overhead, i.e., extra code introduced by the parallelization process. The code comprising the overhead κ usually depends on whether the system uses shared or distributed memory.

In essence, Eq. (3.2b) is the manifestation of Amdahls law [32] in its simplest form, the only difference being that Amdahl neglects the overhead κ. Amdahls law says that the if a parallel program contains a serial part σ, the speedup s will be bounded by the serial part. Define $f = \sigma/(\sigma + \varphi)$ as the fraction of time spent (by a serial program) in the inherently sequential part of the program. Then, Amdahls law says that the speedup of the program is bounded by

$$s \leq \frac{1}{f + (1 - f)/p}. \tag{3.3}$$

Hence, even if $p \to \infty$ the best possible speedup is $1/f$. Observe that if $f \to 0$ the speedup is p. This is known as linear speedup, and implies that the program perfectly divides the workload.

Gustafson's law states that for a particular program, the fraction f will decrease as the problem size increases. The size of a problem is usually measured in the number of floating point operations (FLOPs) required to complete it. The larger the problem, the more FLOPs must be performed. As f decreases the potential speedup increases. In other words, to achieve high speedup, the problem size should be large, to minimize the sequential fraction of the program [31]. For large problem sizes Gustafson predicts that the speedup is bounded by

$$s \leq p + (1 - p)f. \tag{3.4}$$

The experimental serial fraction of a program $e = (\sigma + \kappa)/(\sigma + \varphi)$, differs from f in that it includes the overhead κ. The Karp-Flatt metric [32] can be used for estimating e,

$$e_p = \frac{1/s_p - 1/p}{1 - 1/p}. \tag{3.5}$$

In the case where a program is not gaining the desired speedup, calculating e_p may bring insight into the reason for this. If the experimental serial fraction e_p is increasing as p is increasing, the culprit is probably parallel overhead. If e_p is constant for increasing p, the speedup is limited by the actual parallelization. That is, a large part of the program is still serial.

Efficiency E is a measure of how "well utilized" the cores are in a parallel program. It is defined as

$$E = \frac{s}{p} = \frac{t_{serial}}{p \cdot t_{parallel}}. \tag{3.6}$$

For a linear speedup $E = 1$. Note that super-linear speedups can in some cases be obtained. This implies $s > p$ (and $E > 1$), and can usually be attributed to e.g., efficient caching. In these situations a sub-problem fits into the CPU cache, and thus less time is spent accessing the computer's main memory than in the serial program.

3.2.2 High performance computing and MPI

Distributed memory systems are most commonly used in high performance computing (HPC) scenarios. In such systems the cause of overhead usually originates from communication between processes.

Message-passing interface (MPI) [31] is a standard that provides functions for handling generation of processes and communication between them, and libraries such as Open-MPI or MPICH make MPI accessible from C, C++ or FORTRAN. The Python package mpi4py [33], makes MPI accessible from Python. Note however that, mpi4py depends on having an MPI implementation already installed on the system.

In MPI, each CPU is given a "rank" as identification. The ranks are unique, non-negative integers starting at 0. MPI can also generate and handle CPU topologies. This means that ranks are distributed to some grid, and assigned coordinates. A Cartesian topology may be created by specifying the number of processors in each grid dimension. Then, with p_x and p_y denoting the number of CPUs in directions x and y, respectively, a 2-dimensional MPI topology with $p_x \times p_y$ CPUs may be created. Figure 3.2 illustrates how 4 processors may be assigned to coordinates in a MPI Cartesian grid. Further context for Figure 3.2 will be presented in the section below. For the 2D Cartesian topology, the CPU ranks follows the labeling scheme in the figure, with increasing rank values in a "row-major order" fashion. In addition to the rank identification, CPUs are assigned a column and a row in 2D Cartesian topologies. Thus, the CPU with coordinate $(row, col) = (0, 0)$ in Figure 3.2 is CPU 0.

The reduction function [31] is a function implemented in the MPI standard, that “reduces a set of numbers into a smaller set of numbers” by performing an operation on the set. Common operations are `MPI_MAX` which returns the maximum element, or `MPI_MIN` which returns the minimum element. These operations are useful in situations where each rank calculates some number, and e.g., the minimum value must be determined. The reduction function returns the result of the operation to one of the ranks. The `Allreduce` function performs a reduction operation and distributes the result to all ranks. Note that a variable is said to be local if it is accessible or readable by only one rank. A global variable is one that is shared by several CPUs. In MPI, global variables does not exist, as no ranks can manipulate the same memory location.

3.2.3 Parallelizing a cellular automaton using MPI

In order to parallelize a cellular automaton, recall that the problem consists of applying a transition function to a cell state, and that a cell evolution only depends on the state of its neighbors. Thus, a CA problem is in principle well suited for parallelization.

As a CA consists of a grid of cells, with each cell state interacting only with its neighbours, the grid may be divided into subgrids and iterated independently. Assigning each process to a subgrid, each CPU now only has to iterate part of the complete grid. Figure 3.2 illustrates how MPI may be used to divide a CA grid G among 4 CPUs. There is however an issue. To iterate a border cell the CPU requires information located in another subgrid, i.e., outside the local memory of the CPU.

Halo-exchange is a technique that may be used for solving this problem. In this technique, the subgrid is padded with an additional layer of cells, resembling a halo (see subgrid illustration in Figure 3.2). The “halo-cells” must be updated with current values from neighboring subgrids between each CA iteration. Let a halo-cell and the cell from which its value was copied, be referred to as “mirror cells”. This communication between two CPUs is illustrated in Figure 3.3. Equivalent halo-exchanges must be performed to all sides of a subgrid. Note that only the interior cells of a subgrid are part of the local computational domain, and that the halo-cells are “read-only” during the iteration.

Thus, a CA may be parallelized by using MPI to generate a 2D Cartesian grid of size $p_x \times p_y$, assigning each CPU to a subgrid, exchanging halos and iterating the subgrids independently. Note that halo-exchanges must be performed between each CA iteration. After the desired number of iterations, each subgrid may be “gathered” and assembled into the full-size grid, using MPI functions.

Let a subgrid be of size $A \times B$. Then the number of halo cells becomes $N_h = AB - (A - 2)(B - 2)$. Ideally, N_h should be much smaller than the total number of cells $N_t = AB$, i.e., N_h/N_t should be small. If the ratio N_h/N_t is large, a subgrid iteration will be dominated by operations related to MPI communication, and not actual computation. Regarding the relation between the subgrid dimensions A and B, it can be shown that the area (number of interior cells) is maximized when $A = B$, i.e., the subgrids should be square to minimize communication. Figure 3.4 shows the ratio N_h/N_t for a square subgrid, indicating for which subgrid size the communication overhead becomes acceptable.

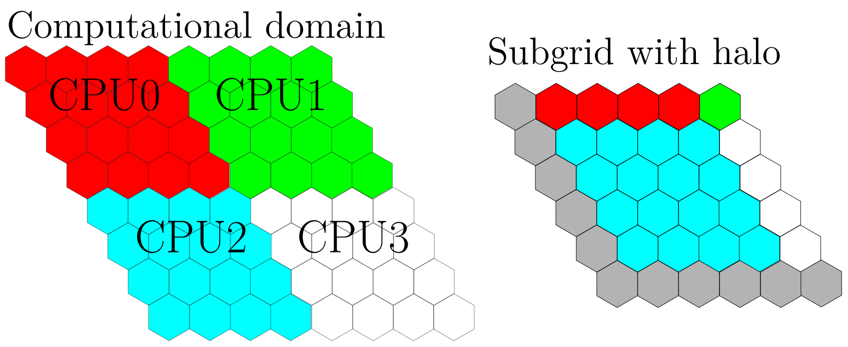


Figure 3.2: The left figure illustrates how a computational domain may be divided into subgrids, and given to separate CPUs, in order to reduce computation time. The color of the cell indicates subgrid association. The grey colored cells indicate the edge of the grid, and its values may be defined by some boundary condition. The right figure illustrates the subgrid of CPU 2 padded with an extra cell-layer, known as a halo. In halo-exchange, the halo cells are updated with values from subgrids of neighboring CPUs. Figure 3.3 illustrates how the halo cells are updated using the halo-exchange technique.

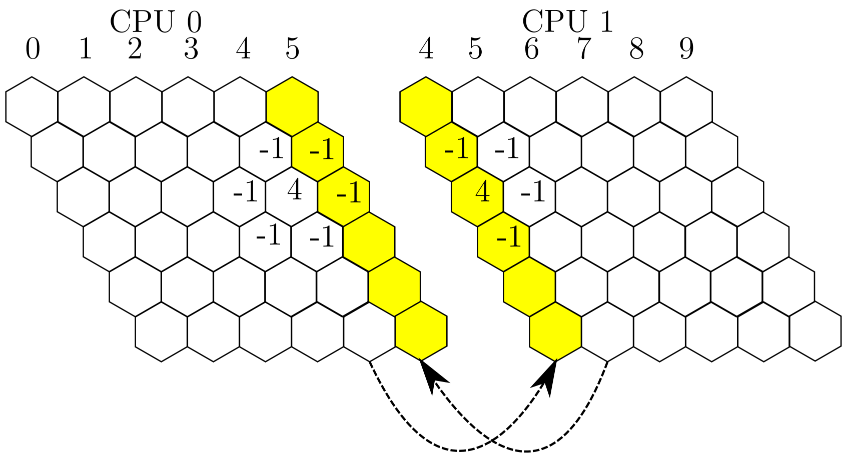


Figure 3.3: This figure illustrates communication between two CPUs during a halo exchange. Two columns are exchanged between CPU 0 and CPU 1, providing the required information to iterate the border cells. Each column is labeled according to the corresponding column index in the complete computational domain (see Figure 3.2). The arrows indicate communication between processes. Some dummy values in the cells help indicate the exchange. An equivalent operation is performed for all borders.

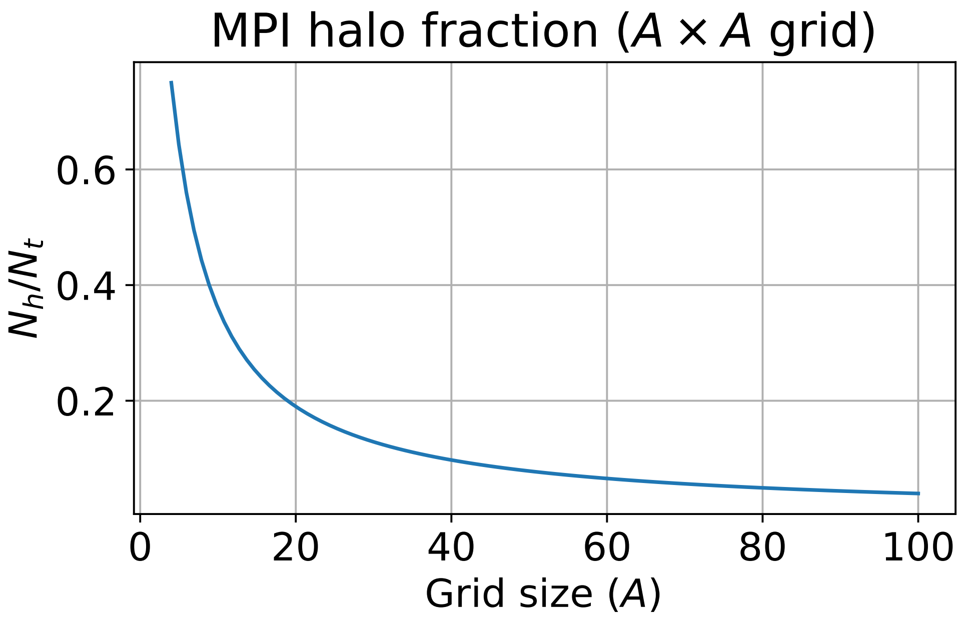


Figure 3.4: The plot indicates the ratio between the number of halo cells N_h and the total number of cells N_t in an MPI subgrid of size $A \times A$. For small subgrids, the amount of MPI communication may surpass the amount of actual computation performed in the grid.

4 | Implementation of the cellular automaton

In this thesis a cellular automaton (CA) has been used for simulating turbidity currents. This chapter is dedicated to discussing the implementation and functionality of the CA. This CA is mostly based on the implementation described by Salles et al. in Ref. [4]. Some modifications were made during implementation, as will be clearly stated in the relevant sections. The source-code for this thesis was written in the programming language Python 3, using the packages "NumPy" and "Cython", and can be found on Github [34]. The implementation was parallelized in MPI using the python package "mpi4py" [33].

Put in context of the definition of a CA, given in Section 2.2, the CA implemented for this thesis can be defined by:

$$< G, N, S, P, \sigma > .$$

Here G is a grid of hexagonal cells, with indexing as specified in Section 2.2.1. N is the set of neighborhood-cells adjacent to the hexagonal cell, illustrated in Figure 2.2B. The σ, and the sets S and P, will be elaborated in their own respective sections. S is the set of possible cell states, P is a set of parameters which are used in the CA computations and σ is the transition function. Section 4.5 contains some practical information about how to initialize and run the CA, and how the implementation was "cythonized" and parallelized.

4.1 Cell states S

The state S of a cell depends on seven substates. These substates are denoted by $Q_a, Q_{th}, Q_v, Q_{cj}, Q_{cbj}, Q_d, Q_o$. Thus,

$$S = Q_a \times Q_{th} \times Q_v \times Q_{cj} \times Q_{cbj} \times Q_d \times Q_o, \tag{4.1}$$

with each substate corresponding to a physical quantity specified in Table 4.1. Note that N_j is the number of different particle types included in the simulation, and that the index $j = 1, 2, ..., N_j$ denotes different sediment types. All substates (except Q_v) are illustrated in Figure 4.1. The substates Q_{th}, Q_v, Q_{cj} and Q_o describe the turbidity current, and may be denoted as "turbidity current states". The remaining substates Q_a, Q_d and Q_{cbj} describe the sea bed, and may be denoted "bed states".

4.1.1 Notation, substate indices and arguments

When specifying an update rule, a substate may be denoted with an argument, e.g., $Q_d(k)$. In this case k refers to neighbor number k, following the scheme presented in Section 2.2.1. The substate Q_o may be specified using two arguments, e.g., $Q_o(a, b)$. In this case the outflow from cell a to cell b is meant. If no argument is given or the argument 0 is used, the substate of the center cell is implied. The notation [ ] is used to specify a specific cell in the grid (see Figure 2.2), e.g., $Q_d\,[l, m]$ refers to substate Q_d of the cell with indices $[l, m]$. Iterations of substates are specified by a superscript, i.e., $Q^{(n+1)}$ refers to the iterated state of $Q^{(n)}$. Iteration is not specified however, if a substate is unaltered by that transformation (T) or interaction (I). If a substate is unaltered by that particular I or T, the most recently updated value is implied, as specified by the order in Section 4.4.

Table 4.1: This table explains the physical meaning for all substates in S, what unit they have, and how many values are stored per cell.

substate	Physical quantity	Unit	$\frac{\text{stored values}}{\text{cell}}$
Q_a	Cell altitude: bathymetry + Q_d	m	1
Q_d	Thickness of soft sediment	m	1
Q_v	Speed of turbidity current	m/s	1
Q_{cj}	Concentration of jth sediment type in turbidity current	1	N_j
Q_{cbj}	Concentration of jth sediment type in soft sediment	1	N_j
Q_{th}	Thickness of turbidity current	m	1
Q_o	Outflow of turbidity current	m	6

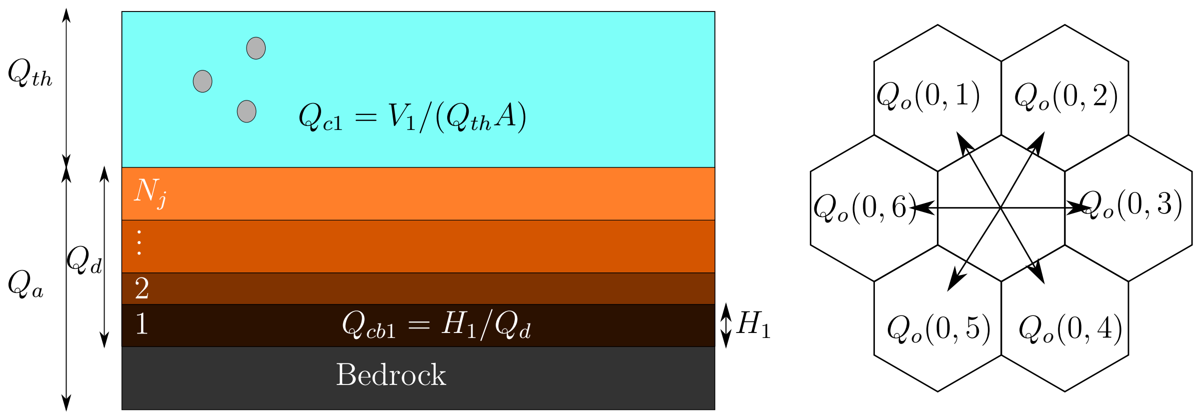


Figure 4.1: These figures illustrate some of the substates (see Table 4.1) constituting the cell state S. The left figure represents a cross section of a cell, the right figure is an overview of a cell and its six neighbors. N_j is the number of different sediment types in the simulation. V_1 is the total volume of all particles of type 1, and A is the base area of the hexagonal cell. $Q_o(0, 1)$ is the flow from cell 0 to neighbor 1. Note that in reality, the sediment types may be mixed, and that the well-defined segmentation shown here is just for illustration.

4.1.2 Initial and boundary conditions

In this section the initial values of the substates are described. Let X be the set of cells considered as sources, and B be some submarine terrain of non-erodible bedrock. The initial conditions may be specified by,

$$\left.\begin{aligned} Q_d^{(0)} &= Q_{d,0} \\ Q_a^{(0)} &= B[l,m] + Q_{d,0} \\ Q_{cbj}^{(0)} &= Q_{cbj,0} \\ Q_o^{(0)} &= 0 \end{aligned}\right\} \qquad \forall [l,m] \in G$$

$$Q_\gamma^{(0)}[l,m] = \begin{cases} 0 & \text{if } [l,m] \notin X \\ Q_{\gamma,0} & \text{if } [l,m] \in X \end{cases} \text{with } \gamma = \{th, v, cj\}.$$

Here $Q_{d,0}$ is some constant specifying the initial height of the erodible, soft sediment layer. $Q_{cbj,0}$ specifies the initial concentration of each sediment type on the sea bed. $Q_{th,0}, Q_{v,0}, Q_{cj,0}$ specifies the initial properties of the turbidity current. Respectively, they represent the initial thickness, the initial velocity, and initial concentration of sediment type j in the turbidity current. The substate Q_o is initialized to zero for all cells, and later calculated as part of the transition function σ.

The state S of a border cell, [border], is excluded from time evolution. The idea is that the substates of border cells are given appropriate values in order to not influence the cells in the simulation domain. In other words, the simulation should behave as if the grid G was infinitely large, and as there were no boundaries. An attempt to achieve this effect has been made by letting the boundary-cells receive values from interior-cells, and subsequently resetting the value of the boundary cells (back to their initial conditions), thus removing e.g., current thickness Q_{th} or sediment Q_d from the system.

4.1.3 Source cells

This section will explain how the source cells X of the grid are updated for each CA iteration. This is relevant because it is of interest to simulate a continuous sediment input. The source cells could correspond to the location of, e.g., a river mouth or some submarine tailing disposal process.

In order to simulate simulate this process, the current thickness Q_{th}, speed Q_v, and particle concentration Q_{cj} of the source cells must be updated between each iteration n. The new substate value is a weighted mean of the present value and the fixed input/source value. Using the notation specified above, let $Q_{\gamma,0}$ (with $\gamma = \{th, v, cj\}$) correspond to the source values. Then the updated substates become:

$$Q_v^{(n+1)} = \frac{Q_v^{(n)} + Q_{v,0} Q_{th,0} \Delta t}{Q_{th,0} \Delta t + Q_{th}^{(n)}}, \tag{4.2}$$

$$Q_{cj}^{(n+1)} = \frac{Q_{cj}^{(n)} + Q_{cj,0} Q_{th,0} \Delta t}{Q_{th,0}\Delta t + Q_{th}^{(n)}}, \tag{4.3}$$

$$Q_{th}^{(n+1)} = Q_{th}^{(n)} + Q_{th,0}\Delta t. \tag{4.4}$$

Δt is the time discretization used in the simulation and will be elaborated below. By accounting for the time discretization, these update rules ensure that mass is added to the system at a constant rate.

4.1.4 Concentration of deposited sediment

This section introduces the concentration C of the amount of sediment type j deposited, which is a concept relevant for analysis of results produced by the CA.

Consider a cell with an erodible soft sediment layer $Q_d = \sum_j^{N_j} H_j$, where H_j is the amount of sediment of type j in the cell (see Figure 4.1). During the course of a simulation, H_j may change through erosion and deposition. When considering only deposition, the amount of sediment of type j deposited in n iterations can be found by,

$$\Delta H_j^{(n)} = \begin{cases} H_j^{(n)} - H_j^{(0)} = Q_{cbj}^{(n)} Q_d^{(n)} - Q_{cbj}^{(0)} Q_d^{(0)} & \text{if } H_j^{(n)} \geq H_j^{(0)} \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Then the concentration of particle type j in the sediment *deposited* after n iterations can be found by,

$$C_j^{(n)} = \frac{\Delta H_j^{(n)}}{\sum_j \Delta H_j^{(n)}}. \tag{4.6}$$

Note that C_j is different from Q_{cbj}, which gives the concentration of particle type j for the total amount of sediment in that cell.

4.2 Parameters P

The global parameters P consist of twelve predefined constants which are input to the CA, and one run-time determined variable Δt.

The calculation of Δt is elaborated in Section 4.3. The rest of the global parameters are constants, defined in Table 4.2. These parameters are used in various calculations for the transition function σ, see Section 4.4. In this implementation the spatial discretization Δx is defined as the intercellular distance (see Section 2.2.1), while in Ref. [4], the apothem of the hexagon is used.

Table 4.2: This table describes the global parameters that must be specified prior to running the CA. The "Relevant calculation" column specifies whether the parameter is used in an update rule or another calculation. The update rules are elaborated in Section 4.4.

Parameter	Description	Relevant calculation	Unit
Δx	Spatial discretization	Δt, I_1	m
p_{adh}	Unmovable amount of density current	I_1	m
θ_f	Friction angle limit	I_1	$^\circ$
θ_r	Angle of repose	I_4	$^\circ$
g	Gravitational acceleration	g'	m/s^2
D_{sj}	Diameter of sediment type j	T_2	m
v_{sj}	Fall velocity of sediment type j	T_2	m/s
ρ_j	Density of sediment type j	T_2	kg/m^3
ρ_a	Density of ambient fluid	g'	kg/m^3
γ	Porosity of sea bed	T_2	1
c_D	Drag coefficient of sea bed	T_2	1
ν	Kinematic viscosity of water	T_2	m^2/s

4.3 Time step Δt

Components of the transition function σ update each substate in S using equations which will be defined in Section 4.4. Some of the update rules are derived from PDEs or ODEs in time, which are discretized using the explicit Euler method. Explicit Euler is a *conditionally stable* scheme [22], which means there will be restrictions on the parameters involved in the relevant PDEs or ODEs. In this implementation, the restriction is put on the time step Δt.

Let all cells with a non-zero current thickness Q_{th} be denoted as $[\ell]$. In other words $[\ell]$ is the set of cells in which there is a turbidity current. With $r\,[\ell]$ being the run-up height (defined in Eq. (2.20)) and $g'\,[\ell]$ the reduced gravity (defined in Eq. (2.16)) for cells in $[\ell]$, a characteristic speed can be defined as $u\,[\ell] = \sqrt{2r\,[\ell]\,g'\,[\ell]}$. Using $u\,[\ell]$ in the CFL condition (defined in Eq. (2.18)) and solving it for Δt yields:

$$\Delta t_{[\ell]} = \frac{\Delta x/2}{u\,[\ell]\,C_{max}} \tag{4.7}$$

where $\Delta t_{[\ell]}$ contains the Δt value for all cells in $[\ell]$. In this implementation $C_{max} \to 1$ is used. $\Delta t_{[\ell]}$ may be referred to as the maximum relaxation time [4]. The smallest value in $\Delta t_{[\ell]}$ is chosen as the global time step:

$$\Delta t = \min_{[\ell]} \Delta t_{[\ell]}. \tag{4.8}$$

This calculation of Δt is identical to the procedure described in Ref. [4]. However, some additional remarks on the parallelization of the Δt calculation are made in Section 4.5.2.

4.4 Transition function σ

As mentioned in Section 2.2, the transition function σ defines the time evolution of the CA. The transition function for this CA consists of two internal transformations and four local interactions. One CA time step constitutes running the six transition function components in the order specified by the following list:

1. Internal transformation (T_1): water entrainment
2. Internal transformation (T_2): erosion and deposition
3. Local interaction (I_1): turbidity current outflow update
4. Local interaction (I_2): turbidity current thickness and concentration update
5. Local interaction (I_3): turbidity current flow speed update
6. Local interaction (I_4): slope failure model/toppling rule

Each component of the transition function, contains one or several update rules, which will be elaborated in the following sections. As mentioned, the transformations and interactions are mostly based on the transition function in Salles et al. [4]. However, some rules are slightly modified, as will be explained.

Because the implementation uses a dynamic time step, Δt must be calculated prior to running the transition function. Denoting $\Delta t^{(n)}$ as the time step value at time iteration n, one time step in the CA constitutes:

1. Calculate new Δt: $\Delta t^{(n)}$
2. Run transition function $\sigma(\Delta t^{(n)}, S^{(n-1)}) \to S^{(n)}$

where $S^{(n)}$ are the states of the CA at time iteration n.

4.4.1 Internal transformation (T_1): water entrainment

This internal transformation accounts for the entrainment effect (see Section 2.1.2) of the turbidity current. The update rule is denoted by

$$T_1 : Q_{th} \times Q_{cj} \times Q_v \to Q_{cj} \times Q_{th}. \tag{4.9}$$

The water entrainment is expected to “dilute” the turbidity current with ambient water, increasing the current volume and decreasing the particle concentration. This effect is implemented through the water incorporation rate E_w,

$$E_w \equiv e_w Q_v = \frac{0.075 Q_v}{\sqrt{1 + 718\mathrm{Ri}^{2.4}}}, \text{ with } \mathrm{Ri} = \frac{g' Q_{th}}{Q_v^2}. \tag{4.10}$$

Ri is the Richardson number, as explained in Section 2.1.2. Thus, the update rules are the following [4],

$$Q_{th}^{(n+1)} = Q_{th}^{(n)} + E_w \Delta t, \tag{4.11}$$

$$Q_{cj}^{(n+1)} = \frac{Q_{cj}^{(n)} Q_{th}^{(n)}}{Q_{th}^{(n+1)}}. \tag{4.12}$$

4.4.2 Internal transformation (T_2): erosion and deposition

This internal transformation accounts for the erosion and deposition, i.e., the exchange of mass between the sea bed and the turbidity current. The update rule can be denoted by

$$T_2 : Q_a \times Q_{th} \times Q_{cj} \times Q_{cbj} \times Q_v \rightarrow Q_a \times Q_d \times Q_{cj} \times Q_{cbj}. \tag{4.13}$$

This rule is expected to either remove or add thickness to the soft sediment layer at the sea bed, Q_a and Q_d. The concentrations of particle types in the turbidity current and the sea bed, Q_{cj} and Q_{cbj}, are updated accordingly. Some changes are made to this implementation of T_2, with respect to the rule presented in Salles et al. [4].

The Exner equation (Eq. (2.3)) determines the amount of mass to be moved between the bed and the turbidity current. In our case, with several sediment types, the Exner equation can be expressed as

$$\frac{\partial z}{\partial t} = \frac{\partial Q_d}{\partial t} = \frac{\partial Q_a}{\partial t} = \frac{D - E}{1 - \gamma} = \frac{1}{1 - \gamma} \sum_{j=1}^{N_j} (D_j - Q_{cbj} E_j v_{sj}), \tag{4.14}$$

D is the deposition rate, i.e., amount of deposited mass per time unit. E is the erosion rate, i.e., amount of eroded mass per unit time. Correspondingly, D_j and $Q_{cbj} E_j v_{sj}$ are the deposition and erosion rates of sediment type j. The settling velocity of sediment type j, v_{sj}, is found by Eq. (2.22). Note that in Ref. [4], the dimensionless settling velocity, i.e., Eq. (2.23) is used instead of Eq. (2.22). The settling velocity is discussed when verifying T_2, in Section 5.2. Also note that the erosion term in Eq. (4.14) differs from the term as it is presented in Salles et. al [4], where the sinking velocity v_{sj} is omitted. However, as this leads to an inconsistency in the physical dimensions of the expression, it is assumed to be a typo. Furthermore, the corresponding bed-sediment conservation equation in e.g., Ref. [11] or Ref. [35] includes v_{sj}.

The deposition rate D_j is found by

$$D_j = v_{sj} c_{nbj}. \tag{4.15}$$

Here c_{nbj} is the near-bed concentration, which can be calculated using Eq. (2.9). The erosion rate E_j is found by using Eq. (2.4). Here, the definition of f (Eq. (2.8)) differs from the definition used in Ref. [4]. The definition of f was changed for this implementation in order to accommodate a larger range of particle sizes D_{sj}. Notice that the particle Reynolds number R_{pj} (Eq. (2.7)) may become much less than 1 for sufficiently small particles. For instance, a particle with diameter $D_{sj} \sim 1\,\mu\text{m}$ and density $\rho = 2650\,\text{kg/m}^3$ in water, has $R_{pj} \ll 1$.

Using Eqs. (4.15) and (2.4), the new values of Q_a and Q_d can be found by discretizing Eq. (4.14) using the explicit Euler scheme. This yields

$$z^{(n+1)} = z^{(n)} + \Delta t \frac{\sum_{j=1}^{N_j} \left(D_j - Q_{cbj}^{(n)} E_j v_{sj}\right)}{(1-\gamma)}, \tag{4.16}$$

where z represents substates Q_a and Q_d. The updated values of Q_{cj} and Q_{cbj} are found by

$$Q_{cj}^{(n+1)} = Q_{cj}^{(n)} - \Delta t \frac{D_j - Q_{cbj}^{(n)} E_j v_{sj}}{(1-\gamma) Q_{th}}, \tag{4.17}$$

$$Q_{cbj}^{(n+1)} = Q_{cbj}^{(n)} + \Delta t \frac{D_j - Q_{cbj}^{(n)} E_j v_{sj} - Q_{cbj}^{(n)} \sum_{j=1}^{N_j} \left(D_j - Q_{cbj}^{(n)} E_j v_{sj}\right)}{(1-\gamma) Q_d^{(n+1)}}. \tag{4.18}$$

In Ref. [4], in the equation corresponding to Eq. (4.17), the net-change in concentration, $\Delta t \frac{D_j - Q_{cbj}^{(n)} E_j v_{sj}}{(1-\gamma) Q_{th}}$, is *added* to the state Q_{cj}, instead of being subtracted as it is in Eq. (4.17). The turbidity current concentration Q_{cj} is expected to decrease if the current is depositing sediment, i.e., $D_j > Q_{cbj} E_j v_{sj}$. And conversely, Q_{cj} is expected to increase if the current is erosive. To achieve this, the sign must be negative, and thus this must be a typo in Ref. [4].

Additionally, Eq. (4.18) differs from the corresponding rule in Ref. [4], in that the new iteration value $Q_d^{(n+1)}$ is used in the denominator, instead of the old value $Q_d^{(n)}$. Using $Q_d^{(n)}$ leads to incorrect normalization, and is assumed to be a typo.

Note that these update rules may result in substates attaining unphysical values, i.e., concentration values Q_{cj}, Q_{cbj} that are outside the range $[0, 1]$, or negative sediment height Q_d. In this implementation, this is handled in the following way: if unphysical values occur for one of the cells to which this rule has been applied, then that cell state is reverted to its state prior to T_2 application. This correction, or a similar one, is a necessary addition to the rule T_2 as it is presented in Salles et al. [4].

4.4.3 Local interaction (I_1): turbidity current outflow update

This local interaction calculates the outflows Q_o for each cell. It can be denoted by

$$I_1 : Q_a^7 \times Q_{th}^7 \times Q_v \times Q_{cj} \to Q_o^6. \tag{4.19}$$

To find the outflow in each direction $Q_o(0, k)$, a minimization algorithm is used. This algorithm was composed by Di Gregorio et al. [36], and rewritten by D'Ambrosio et al. [37].

Initially, let A be the set $\{1, 2, 3, 4, 5, 6\}$ where each number represents a neighbor of the center cell (see Figure 2.2). In the algorithm, cells that shall not receive any outflow will be eliminated from A.

Step 1:

Define the angle β_k, specified by the differences in height (see Figure 4.2) between the central cell and the neighbor cell k,

$$\beta_k = \arctan\left(\frac{(Q_a + r) - (Q_{th}(k) + Q_a(k))}{\Delta x}\right) \tag{4.20}$$

Here r is the run-up height, as defined by Eq. (2.20). Comparing β_k with the friction angle limit θ_f, all neighbor cells whose $\beta_k < \theta_f$ are eliminated from the set A. Note that θ_f, and not the angle of repose θ_r, is used here (see Table 4.2).

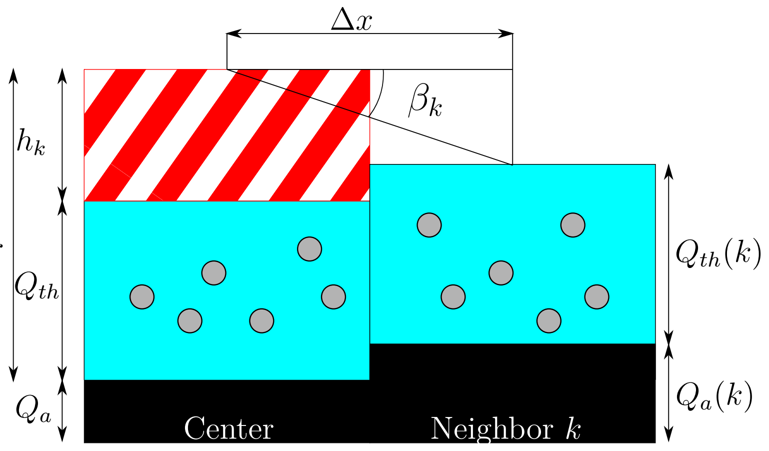


Figure 4.2: This figure illustrates how the angle β_k is defined in the local interaction I_1. h_k is the height reachable by a turbidity current due to its kinetic energy, see Section 2.3.4.

Step 2:

Let N_a =Card(A) be the number of cells in A. Define an average Γ:

$$\Gamma = \frac{1}{N_a}\Big(r - p_{adh} + \sum_{k\in A}(Q_a(k) + Q_{th}(k))\Big), \tag{4.21}$$

where p_{adh} is some unmovable amount of turbidity current thickness (see Table 4.2).

Step 3:

If $(Q_a(k) + Q_{th}(k)) \geq \Gamma$ remove cell k from A. If any cell is eliminated, go back to **Step 2**. The purpose is to have no outflow to any cells with an above average height among the neighbors.

Step 4:

Let $f(k), k \in [1,6]$ be the non-normalized outflows for a cell. Then $f(k)$ is found by

$$f(k) = \begin{cases} \Gamma - (Q_a(k) + Q_{th}(k)) & \text{if } k \in A \\ 0 & \text{if } k \notin A \end{cases}. \tag{4.22}$$

Finally, the normalized outflow $Q_o(0,k), k \in [1,6]$ is found by

$$Q_o^{(n+1)}(0,k) = \frac{Q_{th}}{r} p_r f(k). \tag{4.23}$$

Here $p_r = \sqrt{2rg'}\frac{\Delta t}{\Delta x/2}$ is a relaxation factor [4, 37] that is constant, and shared by all cells during a CA iteration n. The relaxation factor should satisfy $0 < p_r \leq 1$ according to Ref. [37]. In this implementation the constraint $0.2 \leq p_r \leq 1$ is used. This is enforced by comparing the calculated p_r to the predefined limits. The lower limit of p_r is determined through numerical experiment. While testing the CA, it was observed that without a lower boundary on p_r, the turbidity current did not distribute itself to the grid. In other words, it was confined to a small area about the source cells. Raising the lower boundary too high, led to excessive amounts of turbidity current flow, and a checkerboard pattern for the turbidity current height Q_{th} was observed.

4.4.4 Local interaction (I_2): turbidity current thickness and concentration update

This local interaction calculates the new turbidity current thickness, and corresponding sediment concentration, due to sediment flow between cells. The update rule can be denoted by

$$I_2 : (Q_{th} \times Q_{cj} \times Q_o^6)^7 \to Q_{th} \times Q_{cj}. \tag{4.24}$$

The expected behaviour is that when a cell has an outflow, its turbidity current thickness will decrease, and if there is an inflow, the thickness increases. The new concentration then becomes a weighted average of the old concentration Q_{cj}, and the neighbor cells' concentration $Q_{cj}(k)$.

The update rules are defined by

$$Q_{th}^{(n+1)} = Q_{th}^{(n)} + \sum_{k=1}^{6} (Q_o(k,0) - Q_o(0,k)), \tag{4.25}$$

$$Q_{cj}^{(n+1)} = \frac{\left(Q_{th}^{(n)} - \sum_{k=1}^{6} Q_o(0,k)\right) Q_{cj}^{(n)} + \sum_{k=1}^{6} Q_o(k,0) Q_{cj}^{(n)}(k)}{Q_{th}^{(n+1)}}. \tag{4.26}$$

4.4.5 Local interaction (I_3): turbidity current flow speed update

This local interaction updates the speed of the turbidity current. The update rule can be denoted by

$$I_3 : Q_a^7 \times Q_{th}^7 \times Q_o^7 \times Q_{cj} \to Q_v. \tag{4.27}$$

This update rule calculates the "open channel speed" for each direction k out of a cell using Eq. (2.25) with alteration 1, made by T. Salles (see Section 2.3.6). Alteration 2, i.e., the factor φ, is omitted from U in this implementation. The reason behind this is that the volumetric concentration Q_{cj} is already accounted for in the reduced gravity g'. This claim is supported by the corresponding equations in e.g., Refs. [29] and [38].

This rule considers the flow towards each of the 6 neighbor cells as an open channel flow, and so Eq. (2.25) is solved for each direction $k \in [1, 6]$. In Ref. [4], the slope s in each direction, s_k, is defined as, "the difference in height between the central cell $(Q_a + Q_{th})$ and the neighbor cell $(Q_a(k) + Q_{th}(k))$". This is interpreted as,

$$s_k = \frac{1}{\Delta x}\left[(Q_a + Q_{th}) - (Q_a(k) + Q_{th}(k))\right]. \tag{4.28}$$

Here, the factor $1/\Delta x$ is included in order to account for the distance between grid cells. Continuing the open channel flow analogue, the "height of the flow" h will in this case be the outflow $\widetilde{Q_o}(0, k)$ in direction k, i.e., in Eq. (2.24), $h \to \widetilde{Q_o}(0, k)$. Note that here, $\widetilde{Q_o}$ is the non-scaled outflow, i.e., $\widetilde{Q_o} = Q_o/p_r$, where p_r is the global relaxation constant (see Section 4.4.3). The model described in Ref. [4] uses the substate Q_o for this purpose. However, numerical experiments indicate that the non-scaled outflow $\widetilde{Q_o}$ provides the best results.

Using s_k and $\widetilde{Q_o}$ in Eq. (2.25), the speed in each neighbors' direction U_k can be calculated. Note that, in using the definition Eq. (4.28), s_k may become negative. As $U_k \propto \sqrt{s_k}$, the slope s_k must be set to zero if $s_k < 0$ before computing the speed in that direction, to avoid complex numbers. The speed of the cell Q_v is then found by the arithmetic mean of the nonzero U_k:

$$Q_v^{(n+1)} = \frac{1}{N}\sum_{k=1}^{6} U_k, \tag{4.29}$$

where $N \leq 6$ is the number of nonzero U_k.

Note that Eq. (4.28) is not explicitly defined in Ref. [4]. The slope s_k could also be defined using the absolute difference,

$$s_k = \frac{1}{\Delta x}\left|(Q_a + Q_{th}) - (Q_a(k) + Q_{th}(k))\right|. \tag{4.30}$$

The motivation behind using the absolute difference lies with the fact that a cell may have an outflow $Q_o(0,k) > 0$, even if the height difference in s_k is negative, due to the run-up effect (see Sections 2.3.4 and 4.4.3). However, as shall be seen in Section 6.2, the intention of Salles et al. may have been for s_k to be defined without absolute value, i.e., as Eq. (4.28). For either slope definition, all that is described above holds. Unless explicitly stated, the slope definition Eq. (4.28) will be used throughout this thesis.

4.4.6 Local interaction (I_4): slope failure model/toppling rule

This local interaction accounts for the effect of avalanches on the sea bed, in which an oversteepened hill collapses, and some soft sediment is transferred between cells. The corresponding bed concentrations are also updated. The interaction is denoted by

$$I_4 : Q_a^7 \times Q_{cbj}^7 \times Q_d^7 \to Q_a^7 \times Q_{cbj}^7 \times Q_d^7. \tag{4.31}$$

The implementation used in this thesis uses elements from Ref. [4] and from a model used in Ref. [20]. Extra motivation is given for the Q_d and Q_a update rules, as these were formed during the implementation of this project.

As grid cells attain different soft sediment height values Q_d, the angle to neighbor k, θ_k may be defined, as illustrated in Figure 4.3A. In the figure, $b \equiv Q_a - Q_d$ is the height of the bedrock. The angle θ_k is compared with the angle of repose θ_r: if $\theta_k > \theta_r$, some fraction of the soft sediment, $\Delta\widetilde{Q}_d$, is transferred from the center cell to the neighboring cell. The time evolution is illustrated in Figure 4.3B and 4.3C. The non-normalized fraction of mass $\Delta\widetilde{Q}_{d,k}$, to be moved from the center cell to its neighbor $k \in [1,6]$ is given by,

$$\Delta\widetilde{Q}_{d,k} = \begin{cases} 0 & \text{if } \theta_k \leq \theta_r \\ \min\left\{\xi_k, \frac{1}{2}\right\} & \text{if } \theta_k > \theta_r \text{ and } Q_d > 0 \end{cases} \quad \text{with } \xi_k = \frac{1}{2}\frac{dh_k - \Delta x \cdot \tan(\theta_r)}{Q_d}. \tag{4.32}$$

This implies that a cell may at the most transfer half of its soft sediment. From this the amount of mass to be transferred from a cell to a neighbor k is given by

$$\Delta Q_{d,k} = \frac{\Delta\widetilde{Q}_{d,k} \cdot Q_d}{N}, \tag{4.33}$$

where $\Delta\widetilde{Q}_{d,k}/N$ is the normalized fraction of mass, and N is the number of neighbors satisfying the condition in Eq. (4.32), i.e., $\theta_k > \theta_r$. Scaling with $1/N$ ensures that the amount of soft sediment to be distributed does not exceed the amount present in the cell.

The update rules become for the center cell,

$$z^{(n+1)}(0) = z^{(n)}(0) - \sum_k \Delta Q_{d,k}, \tag{4.34}$$

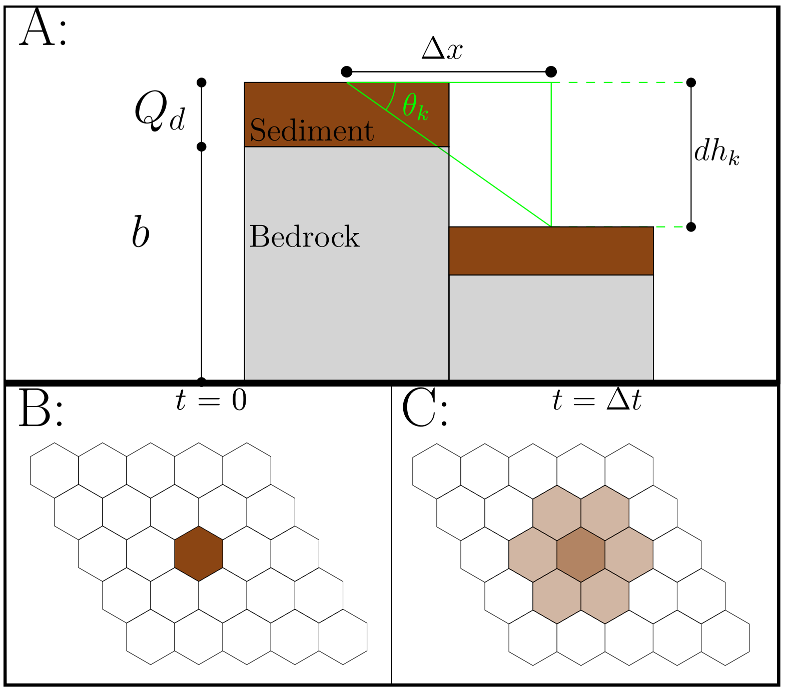


Figure 4.3: Slope failure model. **A**: Illustration of relevant variables for the slope failure model. $b \equiv Q_a - Q_d$ is the bedrock height, and dh_k is the height difference between a center cell and its neighbor $k \in [1, 6]$ (see Figure 2.2). **B+C:** Illustration of ideal behaviour of the slope failure model. The color intensity represents Q_d, i.e., the amount of mass in the cell. At $t = 0$ mass is concentrated in the center cell, while one time step Δt later, the mass is spread out to its neighbors.

with $z = \{Q_d, Q_a\}$, and

$$Q_{cbj}^{(n+1)}(0) = Q_{cbj}^{(n)}(0). \tag{4.35}$$

And for any neighbor cell k with $\theta_k > \theta_r$:

$$z^{(n+1)}(k) = z^{(n)}(k) + \Delta Q_{d,k}, \tag{4.36}$$

with $z = \{Q_d, Q_a\}$, and

$$Q_{cbj}^{(n+1)}(k) = \frac{1}{Q_d^{(n+1)}(k)} \left(Q_d^{(n)}(k) \cdot Q_{cbj}^{(n)}(k) + \Delta Q_{d,k} \cdot Q_{cbj}^{(n)}(0) \right). \tag{4.37}$$

This rule reaches convergence when no toppling occurs, i.e., when $\theta_k \leq \theta_r$ is satisfied for all cells and all directions. The rule also does not account for the size of the time step Δt. This means that the same amount of mass will be moved during a toppling event regardless the size of the time step.

The approach used here is slightly different to the approach used in T. Salles' model [4], in which an avalanche can only occur in one direction, thus exchanging sediment between only two cells at a time. Using the approach outlined above, sediment is exchanged between all eligible neighbor cells. Although, it is expected that both models produce similar results, the model described above was chosen for two reasons. Firstly, it would be reasonable to expect all neighbor cells to receive mass simultaneously during a slope failure (see demonstration in Section 5.6). Secondly, this model is expected to reach convergence in fewer CA iterations n than the alternative. Recall that I_4 is applied once per CA iteration (see Section 4.3). The number of iterations spent on a single toppling event, n_{I4}, could add up to a significant amount of simulated time $\sum_{n_{I4}} \Delta t^{(n)}$ (depending on the size of Δt). A faster converging model implies less elapsed simulation time, and thus the implemented model was the better choice.

Note that this local interaction is the only update rule that may alter the substate of another cell, and that this slightly complicated the parallelization process of the CA. Some remarks on this are made in Section 4.5.2.

4.5 Implementation specific considerations

This section aims to clarify the practicalities of using the CA implementation. Dependencies are listed, the configuration procedure is specified, and an implemented "failsafe" is presented. Sections 4.5.1 and 4.5.2 elaborates on how Cython and MPI has been applied for this CA.

Table 4.3 provides an overview of the Python package dependencies in this CA implementation. Additionally, mpi4py requires a working MPI implementation, and Cython requires a C/C++ compiler to be installed on the system.

Table 4.3: This table lists Python libraries used in the implementation of the CA, and briefly explains what they are used for.

Package	Use
NumPy	Various functions/operations ndarrays used for storing grid states S. One ndarray used for each substate (see Table 4.2).
Cython	Used to compile `.pyx` files
Configparser	Importing initial values and parameters.
xarray	Importing bathymetry data from netCDF.
Matplotlib	Producing figures/output
mpi4py	Running the CA in parallel

All simulation inputs, e.g., parameters in Table 4.2, initial conditions and bathymetry preferences are specified in a `.ini` file. This `.ini` file will from now on be called the "config file". At runtime this file will be read, and the settings are applied to the simulation. Note that source cells may be specified by using indices (x, y in config file) or UTM 33 coordinates (N, E in config file). UTM 33 coordinates are available for bathymetry loaded from NetCDF-files. Appendix A contains a sample/template config file.

The implementation allows for the toppling rule I_4 (Section 4.4.6) to be run during the CA initialization, i.e., prior to when the transition function σ is applied. A tolerance keyword may be specified in the config file, along with a tolerance *tol*. If *tol* is specified, the toppling rule will be applied to the initial sand layer $Q_d^{(0)}$, until Eq. (4.38) is satisfied.

$$\sqrt{\sum_G \left(Q_d^{(n+1)} - Q_d^{(n)}\right)^2} \leq tol \tag{4.38}$$

Here G is the CA grid.

During application of σ, the cell states S are checked for unphysical values. This implies that the substates must satisfy the following conditions after a transformation/interaction is applied: $0 \leq Q_{cj}, Q_{cbj} \leq 1 + \varepsilon$ and $0 \leq Q_{th}, Q_v, Q_d, Q_o$. If any substate attains an unphysical value, the program throws an exception. The ε is added to the limit, to compensate for floating-point errors. In this implementation ε is arbitrarily set to 10^{-4}.

4.5.1 Cythonizing the CA implementation

In this CA implementation, Cython has been applied to the most computationally intensive parts of the code. Disregarding any post-processing (e.g., plotting), the most computationally demanding part of running this simulation is the iteration of the CA, i.e., application of the transition function σ. For this reason, all transformations and interactions of the transition function were written and compiled using Cython. To ensure the best speedup, type declarations were applied, and “memoryviews” were used to manipulate the substate ndarrays, as described in Section 3.1.

4.5.2 Parallelizing the CA implementation

This CA implementation was parallelized using MPI. The parallelization procedure mostly consisted of dividing the computational domain into subgrids (as described in Section 3.2.3), and assigning them to a dedicated MPI rank. Let the coordinate of a rank be given by (row,col).

How the computational domain is divided, and thus what shape $\ell_x \times \ell_y$ the subgrids attain, is determined at runtime, based on the number of CPUs used p, and the size $L_x \times L_y$ of the complete CA grid. The dimensions $p_x \times p_y$ of the MPI 2D cartesian grid, i.e., “the number of processors in each direction” is $(p_x, p_y) = (\sqrt{p}, \sqrt{p})$ if p is a square number. If p is not a square number, the following method is used to determine (p_x, p_y). Let p_1 and $p_2 = p/p_1$ be the two factors dividing p, closest to $\sqrt{p}$, and let $p_1 > p_2$. Then, the dimensions $p_x \times p_y$ is determined by,

$$(p_x, p_y) = \begin{cases} (\sqrt{p}, \sqrt{p}) & \text{if } p \text{ is a square number} \\ (p/p_1, p_1) & \text{if } L_y \geq L_x \\ (p_1, p/p_1) & \text{if } L_x > L_y. \end{cases}$$

For instance, with $p = 512$, which is not a square number, the MPI grid becomes 32×16. Using this, the subgrid size $\ell_x \times \ell_y$ is determined by using the following rule. For ℓ_x, the rule becomes: if $L_x \bmod p_x = 0$, i.e., the number of processors in the x direction exactly divides the length L_x, then $\ell_x = L_x/p_x$. If that is not the case, ℓ_x is determined by

$$\ell_x = \begin{cases} \lfloor L_x/p_x \rfloor + L_x \bmod p_x & \text{if col} = 0 \\ \lfloor L_x/p_x \rfloor & \text{otherwise,} \end{cases} \tag{4.39}$$

where $\lfloor\ \rfloor$ denotes the floor operator. To determine ℓ_y, let $x \to y$ and col $\to$ row. Notice that if $L_z \bmod p_z \neq 0$, i.e., the number of cells in one direction cannot be divided equally among the number of processors in that direction, then, not every subgrid will have the same size.

Determining and distributing the global time discretization Δt

To calculate the global time step Δt, each rank calculates the time step value Δt_{rank} of its subgrid according to the rules in Section 4.3, then

$$\Delta t = \min_{rank} \Delta t_{rank}.$$

This minimum is found, and distributed to all ranks by calling the MPI function `Allreduce`. Note that if one of the subgrids contain no turbidity current, i.e., $Q_{th} = 0$ in all cells of that subgrid, the Δt_{rank} of that grid must be defined as some arbitrary large number (instead of zero), for the reduction operation to work properly. In this implementation Δt_{rank} defaults to 9999999 when a subgrid is empty.

Parallelizing the toppling rule

As mentioned in Section 4.4.6, the toppling rule not only updates the state of the center cell, but also alters the state of its neighbors. This implies that cells near the edge of a subgrid, may send mass to the halo-cell. In ordinary halo-exchange, no change made to a halo-cell will affect the subgrid from which it is mirrored. As such, any mass received by a halo cell would normally just be overwritten during the next halo exchange. To fix this, any mass received by a halo-cell has to be sent to the correct subgrid and added to its mirror cell. This resembles an “inverted halo-exchange”, whose communication can be illustrated by Figure 4.4. The cells receiving mass by the MPI communication are updated according to the “neighbor cell update rules”, i.e., Eqs. (4.36) and (4.37). A demonstration of the inverted halo-exchange can be seen in Section 5.6.

Parallelized CA time step

The iteration of the parallelized CA is about the same as presented in Section 4.3, with the only differences being the calculation of the time step, and that one or several halo-exchanges have to be applied after a transformation or interaction. For instance, transformation T_1 updates the substates Q_{th} and Q_{cj} (see Eq. (4.9)), and thus the halos of Q_{th} and Q_{cj} must be exchanged after its application.

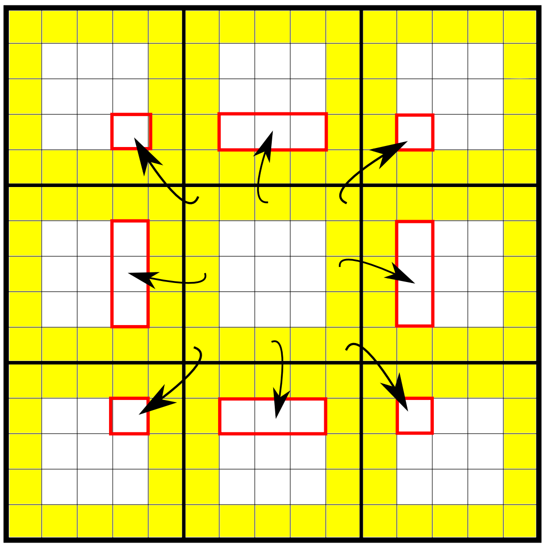


Figure 4.4: This figure illustrates the inverted halo-exchange for an MPI topology consisting of 9 ranks. Here, changes made to a halo-cell is sent back to the corresponding mirror cell. The arrows indicate communication, the yellow cells are halo-cells, and the red squared cells are cells receiving data. Only information sent from the center subgrid is shown here.

Prior to the *first* iteration, halo-exchanges are performed for all the substate grids. Note that this is only needed before the first iteration, and that for subsequent iterations only the halo exchanges listed below are needed. The global time step Δt is calculated using `Allreduce`, as described above. This is followed by a slightly modified transition function $\tilde{\sigma}$:

1. Internal transformation (T_1): water entrainment
2. Exchange halos of Q_{th}, Q_{cj}
3. Internal transformation (T_2): erosion and deposition
4. Exchange halos of Q_a, Q_d, Q_{cj}, Q_{cbj}
5. Local interaction (I_1): turbidity current outflow update
6. Exchange halo of Q_o
7. Local interaction (I_2): turbidity current thickness and concentration update
8. Exchange halos of Q_{th}, Q_{cj}
9. Local interaction (I_3): turbidity current flow speed update
10. Exchange halo of Q_v
11. Local interaction (I_4): slope failure model/toppling rule
12. Exchange halos of Q_a, Q_d, Q_{cbj}
13. Inverted halo-exchange of Q_a, Q_d, Q_{cbj} (toppling rule MPI modification)

Observe that $\widetilde{\sigma}$ effectively is equivalent to σ, with only the halo-exchanges and the inverse halo-exchange being the difference.

5 | Verifying the transition function σ

The purpose of this chapter is to discuss results for isolated update rules in σ, and determine whether the results display the expected behaviour. In this chapter, test-results for the components of σ are presented and discussed.

Some of the parameters (see Table 4.2) are kept constant during the verification. These are given by Table 5.1. Note that the parameters not specified in Table 5.1 assume different test values that will be specified in the relevant section. Also note that in all relevant update rules, only one sediment type is used, i.e., $N_j = 1$.

Table 5.1: Parameter values used for verifying the behaviour of σ.

Description	Parameter	Values
Spatial discretization	Δx	$1\,\mathrm{m}$
Gravitational acceleration	g	$9.81\,\mathrm{m/s^2}$
Sinking speed particle type j	v_{sj}	Given by Eq. (2.22)
Density particle type j	ρ_j	$2650\,\mathrm{kg/m^3}$
Density of water	ρ_a	$1000\,\mathrm{kg/m^3}$
Porosity	γ	0
Bed drag coefficient	c_D	0.003
Kinematic viscosity of water	ν	$1 \cdot 10^{-6}\,\mathrm{m^2/s}$

5.1 Internal transform T_1: water entrainment

In this section the internal transform T_1, defined in Section 4.4.1, is applied to an example grid cell. First, some example substate values are presented, and the transform is applied using these values. Second, the behaviour of the transform is illustrated by plotting the iterated substates for a range of turbidity current speeds Q_v. The expected behaviour of the water entrainment transform, is a dilution of the turbidity current, i.e., an increase in current thickness Q_{th} and decrease in concentration Q_{cj}.

By the update rule definition Eq. (4.9), the relevant input substates are turbidity current speed Q_v, thickness Q_{th} and concentration Q_{cj}. In this example, we define the substates of the cell to be $Q_v = 1\,\mathrm{m/s}$, $Q_{th} = 2\,\mathrm{m}$ and $Q_{cj} = 0.03$. For simplicity we also define a static time step $\Delta t = 1\,\mathrm{s}$. Using these values and Table 5.1, g' is calculated using Eq. (2.16), with the result being $g' \approx 0.5\,\mathrm{m/s^2}$. Inserting this into the definition of the Richardson number from Eq. (4.10) gives

$$\mathrm{Ri} = \frac{g' Q_{th}}{Q_v^2} = 1. \tag{5.1}$$

Applying the update rules given by Eqs. (4.11) and (4.12) gives,

$$Q_{th}^{(n+1)} = Q_{th}^{(n)} + \frac{0.075 Q_v \Delta t}{\sqrt{1 + 718\mathrm{Ri}^{2.4}}} \approx 2 + 2.8 \cdot 10^{-3}\,\mathrm{m}, \tag{5.2}$$

$$Q_{cj}^{(n+1)} = \frac{Q_{cj}^{(n)} Q_{th}^{(n)}}{Q_{th}^{(n+1)}} \approx 0.02996. \tag{5.3}$$

The result is a thickness $Q_{th}^{(n+1)}/Q_{th}^{(n)} > 1$ and concentration $Q_{cj}^{(n+1)}/Q_{cj}^{(n)} < 1$, which is characteristic for the expected dilution. These ratios are plotted in the Figure 5.1 for a range of current speeds Q_v.

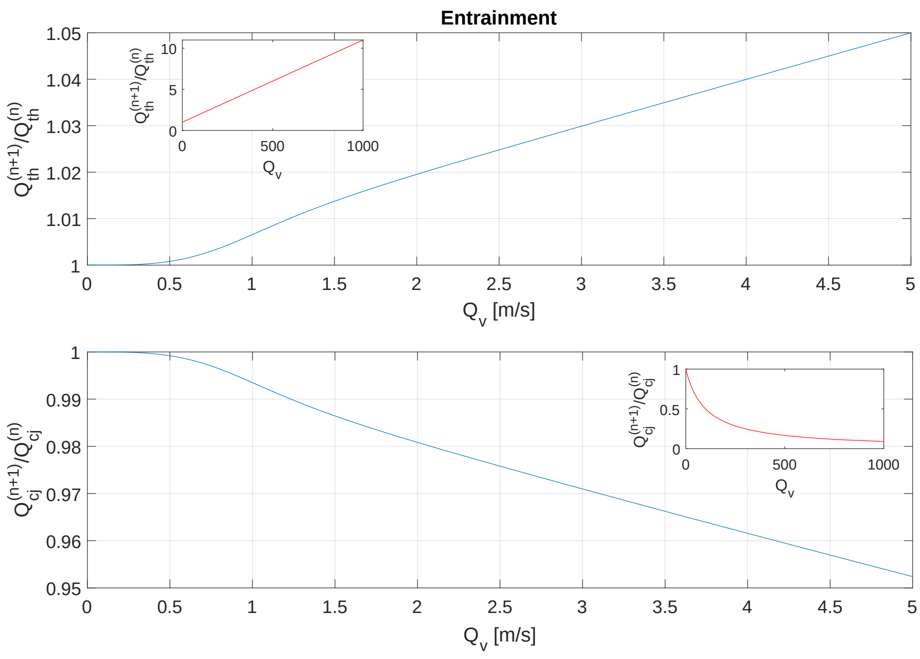
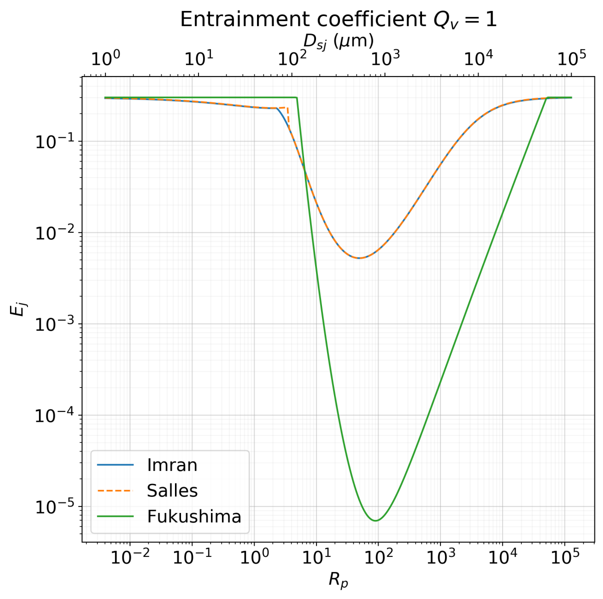


Figure 5.1: This figure displays the behaviour of the internal transformation T_1, responsible for the water entrainment of the turbidity current (see Section 4.4.1). The plots show how the turbidity current thickness Q_{th} grows, and the concentration Q_{cj} decreases, as the cell speed $Q_v \to \infty$. In the plot insets, the range of Q_v is increased to 1000 m/s to illustrate the asymptotic behaviour of the function.

5.2 Internal transform T_2: erosion and deposition

In this section the erosion and deposition rule T_2, presented in Section 4.4.2 is demonstrated. The transform is applied to a solitary cell, considering first, the erosion and deposition rates individually, before combining them (Eq. (4.14)) in order to better understand the characteristics of the transform. The expected behaviour is a balance between deposition and erosion that depends on the turbidity current states, and the



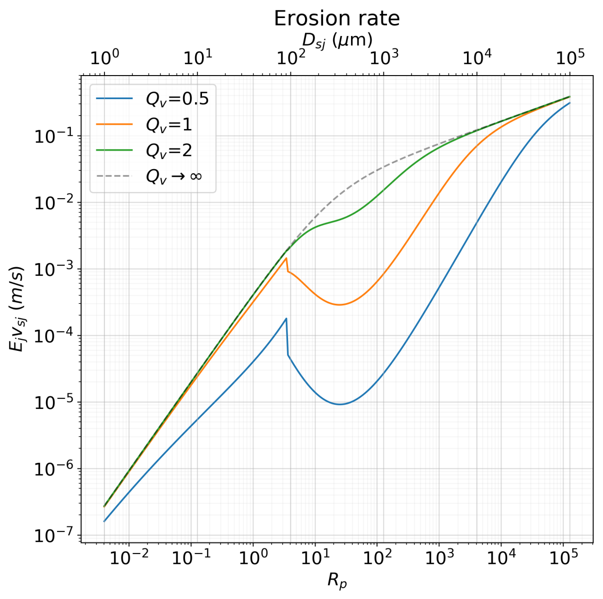


Figure 5.2: The left figure illustrates the entrainment coefficient E_j as a function of the particle Reynolds number R_{pj} (and particle diameter D_{sj}), for the implemented model (Salles), and two other commonly used entrainment models by Imran [35] and Fukushima [15] (see Appendix B). The right figure illustrates the erosion rate's dependence on D_{sj} and R_{pj}. The legend indicates at what turbidity current speed Q_v the plot was captured.

particles it contains. From the definition Eq. (4.13), the relevant substates for this rule are Q_a, Q_{th}, Q_{cj}, Q_{cbj} and Q_v.

First, consider the entrainment coefficient E_j, as given by Eq. (2.4). Since $N_j = 1$, the bed-concentration is $Q_{cbj} = 1$. Then, by specifying a current speed Q_v and a particle diameter D_{sj}, and using the values specified in Table 5.1, E_j can be calculated. The left plot in Figure 5.2 illustrates the dependence of E_j on the particle Reynolds number R_{pj} and the particle diameter D_{sj}, for the current speed $Q_v = 1\,\text{m/s}$. Here, the implemented entrainment model (Salles) is plotted alongside two alternative entrainment models, by Imran [35] and Fukushima [15] (see Appendix B). Observe that the only significant difference between the Salles and Imran models is the sharpness of the Salles model at $D_{sj} \approx 1 \cdot 10^2\,\mu\text{m}$.

The erosion rate is obtained by $E_j v_{sj}$, and is shown in the right plot of Figure 5.2. The plot was made using the same initial conditions as described above, with the current speeds $Q_v = \{0.5, 1, 2\}\text{m/s}$. The dashed line indicates $E_j v_{sj}$ for $Q_v \to \infty$. Observe that the erosion rate is bounded: $E_j v_{sj} \leq 0.3 v_{sj}$. This is because $\lim_{Z_{eff}\to\infty} E_j = 0.3$ (see Eq. (2.4)). Notice that the erosion rate "dips" significantly near $D_{sj} \approx 5 \cdot 10^2\,\mu\text{m}$ for speeds $Q_v \lesssim 2\,\text{m/s}$. This is because when Q_v is sufficiently large, it can dominate Z_{eff}, but when this is not the case, the factor $f(R_{pj})/v_{sj}$, which carries the particle size dependence, has more influence. The factor $f(R_{pj})/v_{sj}$ has a global minimum close to $D_{sj} \approx 5 \cdot 10^2\,\mu\text{m}$, and is strictly increasing for both smaller and greater D_{sj}. This explains why the erosion rates seem to converge to the dashed line for $D_{sj} \lesssim 1 \cdot 10^2\,\mu\text{m}$ and $D_{sj} \gtrsim 1 \cdot 10^4\,\mu\text{m}$.

The deposition rate D_j (see Eq. (4.15)) depends on the current concentration Q_{cj} and particle size D_{sj}. For currents with one particle type $N_j = 1$, the deposition rate becomes $D_j = 2.04 Q_{cj} v_{sj}$. Recall that a turbidity current implies a concentration $Q_{cj} \in [0, 0.09]$. This gives an upper boundary on the deposition rate: $D_j \lesssim 0.2 v_{sj}$. As there is no other particle dependence in the deposition rate for $N_j = 1$, D_j will behave similarly to the sinking velocity v_{sj} for changes in particle size. v_{sj} is illustrated in Figure 2.4. Note that for turbidity currents with several particle types $N_j > 1$, the near-bed concentration c_{nb} depends on particle diameter D_{sj} and the mean particle diameter D_{sg}.

Now consider the rate of change in bed sediment height $\partial z/\partial t$. Using $\gamma = 0$, $N_j = 1$, and $Q_{cbj} = 1$, the Exner equation (Eq. (4.14)), becomes $\partial z/\partial t = D_j - E_j v_{sj}$. For $\partial z/\partial t > 0$, there is net deposition, while for $\partial z/\partial t < 0$ there is net erosion. From the above estimated boundaries of the deposition and erosion rates, it is evident that the two can be of similar order, indicating that $\partial z/\partial t$ can both be positive and negative.

The boundary of the deposition rate is lower than the erosion rate boundary, meaning that erosion should dominate in dense ($Q_{cj} \approx 0.1$), fast ($Q_v \to \infty$) turbidity currents. As the current speed decreases, one expects the erosion rate to decrease, and the deposition term may dominate.

Combining the erosion model by Imran and a deposition model (see Appendix B) by the same author, the Imran erosion/deposition model [35] can be implemented. In Figure 5.3, $\partial z/\partial t$ is plotted as a function of D_{sj}, for some values of Q_{cj} and $Q_v = \{0.5, 2, 3\}$m/s, both for the model described in Section 4.4.2 and the Imran model. Notice that the

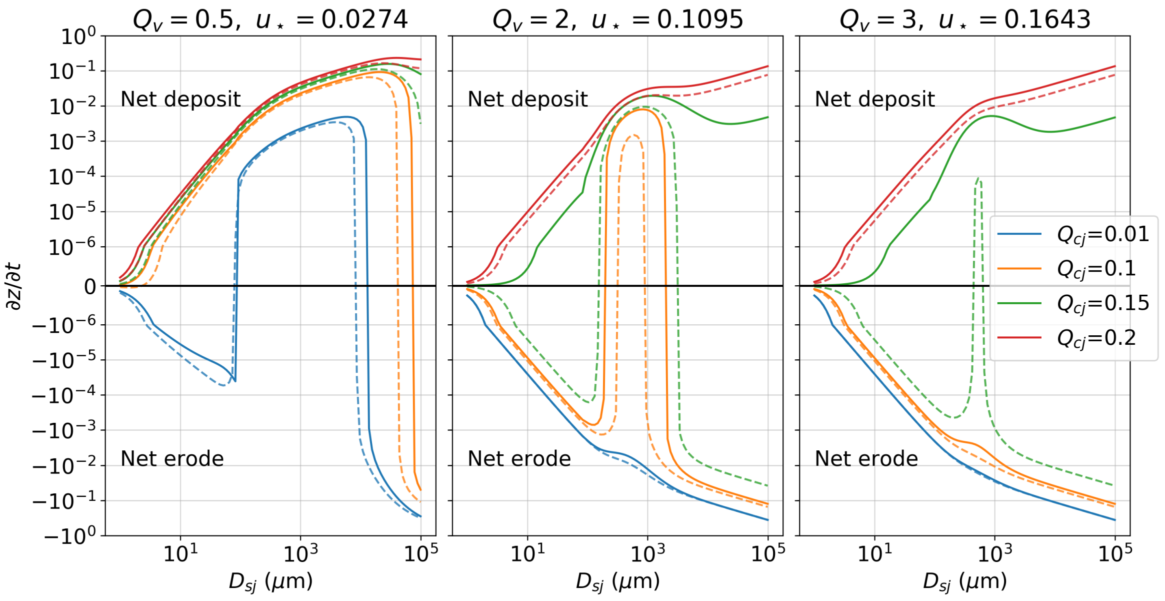


Figure 5.3: These plots serve the purpose of illustrating the behaviour of transition rule T_2, and as a comparison of the erosion/deposition model used in this implementation against a model due to Imran et al. [35]. The change in bed sediment height $\partial z/\partial t$ is plotted as a function of the particle diameter D_{sj}. The solid lines represent the implemented model, while the dashed lines indicates the behaviour of the model by Imran. The figure title indicates the turbidity current speed Q_v and shear-velocity $u_\star$. The legend indicates for which turbidity current concentration Q_{cj} the curves were captured.

models display similar behaviour for all concentrations except $Q_{cj} = 0.15$. However, as for the purpose of this implementation, the turbidity currents concentration will be bounded by 0.09 (see Section 2.1.1), so any discrepancy for concentrations $Q_{cj} > 0.09$ is of little relevance in this thesis.

The sinking speed v_{sj} used in this implementation, and the one used by Salles et al. [4] differs. In Ref. [4] it is stated that the dimensionless sinking speed (see Section 2.3.5) is used. When using the dimensionful v_{sj}, Eq. (2.22) the models by Imran and Salles seem to agree, producing the results shown in the figures above, while using the dimensionless speed $\tilde{v}_{sj}$, yields large deviations between the models. Furthermore, upon inquiry, the corresponding author of Ref. [4], states that it is indeed the dimensionful sinking speed, i.e., Eq. (2.22), that should have been used in Ref. [4]. These arguments constitute the reasons for using the dimensionful sinking speed over the dimensionless speed.

5.3 Local interaction I_1: turbidity current outflow update

This section demonstrates the internal transform I_1, presented in Section 4.4.3. The expected result is an outflow that depends on the value of $Q_a(k) + Q_{th}(k)$, for the neighboring cells, and the height and energy of the center cell. The relevant substate is Q_o^6.

In order to verify the implementation of I_1, the Q_o value is compared with an example provided in D'Ambrosio [37]. The example is adapted to fit this implementation, and the result can be seen in Figure 5.4.

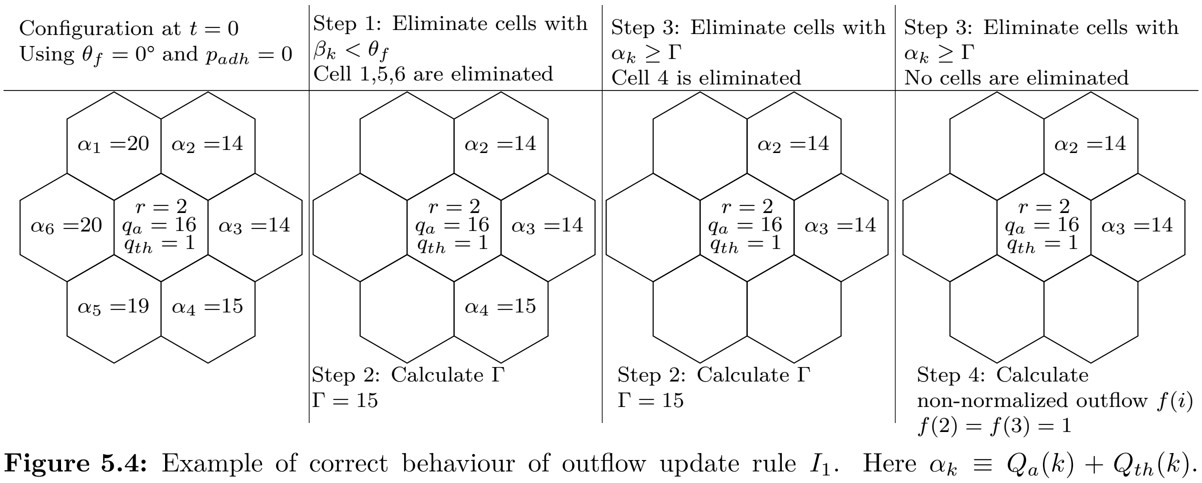

Configuration at $t = 0$
Using $\theta_f = 0°$ and $p_{adh} = 0$
Step 1: Eliminate cells with $\beta_k < \theta_f$
Cell 1,5,6 are eliminated
Step 3: Eliminate cells with $\alpha_k \geq \Gamma$
Cell 4 is eliminated
Step 3: Eliminate cells with $\alpha_k \geq \Gamma$
No cells are eliminated
$\alpha_1 = 20$
$\alpha_2 = 14$
$\alpha_6 = 20$
$r = 2$
$q_a = 16$
$q_{th} = 1$
$\alpha_3 = 14$
$\alpha_5 = 19$
$\alpha_4 = 15$
Step 2: Calculate Γ
$\Gamma = 15$
Step 2: Calculate Γ
$\Gamma = 15$
Step 4: Calculate non-normalized outflow $f(i)$
$f(2) = f(3) = 1$

Figure 5.4: Example of correct behaviour of outflow update rule I_1. Here $\alpha_k \equiv Q_a(k) + Q_{th}(k)$. Adapted from Ref. [37].

5.4 Local interaction I_2: turbidity current thickness and concentration update

In this section the local interaction I_2 is applied to the cell neighborhood illustrated in Figure 5.4. Recall that the local interaction I_2 is responsible for updating the thickness Q_{th} and concentration Q_{cj} of the turbidity current, based on cell outflows Q_o.

The normalized outflows of the center cell, $Q_o(0,2), Q_o(0,3)$, are assumed to be $\frac{1}{4}f(k) = 0.25$, calculated in the previous section. The center cell has cell bathymetry height $Q_a = 16$, and turbidity current thickness $Q_{th} = 1$, and the current concentration is assumed to be $Q_{cj} = 0.03$, before application of I_2. Assume that the neighbor cells have no turbidity current, i.e., $Q_{th}(k) = 0$, so that $\alpha_k = Q_a(k)$ (see Figure 5.4). With these assumptions, the update rules Eqs. (4.25) and (4.26) gives for the center cell:

$$Q_{th}^{(n+1)} = \underbrace{Q_{th}^{(n)}}_{1} + \underbrace{\sum_{k=1}^{6} (Q_o(k,0) - Q_o(0,k))}_{-0.5} = 0.5,$$

and,

$$Q_{cj}^{(n+1)} = \frac{\overbrace{\left(Q_{th}^{(n)} - \sum_{k=1}^{6} Q_o(0,k)\right) Q_{cj}^{(n)}}^{0.5\cdot 0.03} + \overbrace{\sum_{k=1}^{6} Q_o(k,0) Q_{cj}^{(n)}(k)}^{0}}{\underbrace{Q_{th}^{(n+1)}}_{0.5}} = 0.03.$$

Assuming the neighbor cells 2 and 3 (in Figure 5.4) only receive flow from the center cell, their new substate values are given by:

$$Q_{th}^{(n+1)} = \underbrace{Q_{th}^{(n)}}_{0} + \underbrace{\sum_{k=1}^{6} (Q_o(k,0) - Q_o(0,k))}_{0.25} = 0.25,$$

and,

$$Q_{cj}^{(n+1)} = \frac{\overbrace{\left(Q_{th}^{(n)} - \sum_{k=1}^{6} Q_o(0,k)\right) Q_{cj}^{(n)}}^{0} + \overbrace{\sum_{k=1}^{6} Q_o(k,0) Q_{cj}^{(n)}(k)}^{0.25\cdot 0.03}}{\underbrace{Q_{th}^{(n+1)}}_{0.25}} = 0.03.$$

Consider the turbidity current thickness Q_{th} update rule, i.e., Eq. (4.25). This equation alters the previous iteration of Q_{th}, based on the net-flux of the cell. This update rule is then effectively a continuity equation for turbidity current flow. Next, consider the turbidity current sediment concentration Q_{cj} update rule, i.e., Eq. (4.26). The first term in the denominator, $\left(Q_{th}^{(n)} - \sum_{k=1}^{6} Q_o(0,k)\right) Q_{cj}^{(n)}$, gives the amount of sediment type j left (in the current) in the cell after the outflow Q_o is removed. The second term,

$\sum_{k=1}^{6} Q_o(k,0)Q_{cj}^{(n)}(k)$, calculates the amount of sediment type j flowing into the considered cell from the neighbor cells. Thus, the denominator calculates the net-amount of sediment type j in the cell after the turbidity current flow Q_o is accounted for. This is then normalized by the updated turbidity current height, to give the volumetric concentration Q_{cj}.

5.5 Local interaction I_3: turbidity current flow speed update

In this section the local interaction I_3, which updates the turbidity current speed Q_v, is applied to a cell. Let the cell neighborhood be similar to Figure 5.4, except that the alterations made by interactions I_1 and I_2, in the previous sections, are applied. Thus, the relevant substates of the center cell are $Q_{th} = 0.5\,\mathrm{m}$, $Q_a = 16\,\mathrm{m}$ and $Q_{cj} = 0.03$. Using these values, with values from Table 5.1, the reduced gravity becomes $g' \approx 0.5\,\mathrm{m/s^2}$.

The slope s given by Eq. (4.28), becomes

$$s = (16 + 0.5) - (14 + 0.25) = 2.25.$$

The speed of the center cell $Q_v(0)$ can be calculated by using Eq. (2.25) in Eq. (4.29), as specified in Section 4.4.5. For simplicity, assume that the relaxation factor p_r (from local interaction I_1) is 1. Since $U_k \propto \sqrt{Q_o(0,k)}$, only cells that are receiving an outflow from the center cell will have a nonzero U_k. Thus,

$$U_k = \begin{cases} 0 & \text{for } k = 1,4,5,6 \\ \sqrt{\frac{8g'}{f_0(1+a)} Q_o(0,k) s_k} \approx 6.27\,\mathrm{m/s} & \text{for } k = 2,3. \end{cases} \tag{5.4}$$

As there are only two nonzero U_k, both of which are equal, the turbidity current speed becomes

$$Q_v \approx 6.27\,\mathrm{m/s}. \tag{5.5}$$

Note that the slope s_k is calculated using the new turbidity current thickness values Q_{th}, i.e., the current thickness after the flow $Q_o(0,k)$ calculated in I_1 has already been "moved" between cells. Applying this local interaction before the interaction I_2, would result in a slope s_k where the flow Q_o has not yet reached the cells, and could thus yield a larger speed U_k. This insight could prove useful when running the complete simulation, as shall be seen in Chapter 6.

Recall that in Section 4.4.5, there was some ambiguity as to the implementation of the local interaction. Specifically, to how the height difference between cells should be defined, and thus, the definition of the slope s. Two possible definitions of s were presented in Section 4.4.5. Considering this local interaction in isolation provides limited insight into how the two will behave in the context of the full transition function σ. For this reason, results are presented in Section 6.2 illustrating the different outcomes for the two slope definitions.

An important remark regarding the substate Q_v, is that the substate is merely used as an indicator of the turbidity current energy. The speed does have some influence on the *amount* of outflow between cells through the run-up effect, and which cells receive outflow, but the actual speed at which the current moves through cells in the CA grid, is unaffected by Q_v.

5.6 Local interaction I_4: slope failure model/toppling rule

This section is dedicated to demonstrating the results of the slope failure model presented in Section 4.4.6. This update rule is expected to spread the soft sediment, as if there was an avalanche. The effect may be somewhat comparable to diffusion in that it tends to decrease gradients. The relevant substates are Q_d, Q_a and Q_{cbj}, as seen by Eq. (4.31). The results of applying I_4 to selected scenarios are presented, as well as a demonstration of the MPI toppling rule implementation elaborated in Section 4.5.1.

Let the source of sediment be a single cell $[m_0, l_0]$, such that the relevant initial conditions (ICs) are

$$z^{(n=0)}\,[m,l] = \begin{cases} z_0 & \text{if } [m,l] = [m_0, l_0] \\ 0 & \text{else} \end{cases} \tag{5.6}$$

with $z = \{Q_d, Q_{cbj}\}$, and z_0 corresponds to an initial sediment height $Q_{d,0}$ and concentration $Q_{cbj,0}$. In this case the boundaries are set to be impenetrable barriers, i.e., Q_d, $Q_a \to \infty$. Two test scenarios are used: one on a horizontal plane $B = 0$, where B is the height of the non-erodible bedrock, on a hexagonal grid G of 200×200 cells, and one with some submarine terrain $B \neq 0$ using a hexagonal grid of 200×150 cells.

5.6.1 Test scenario with no terrain

Let the cell altitude be equal to the sediment height, $Q_a^{(n)} \equiv Q_d^{(n)}$, i.e., the bedrock height B is 0. Figure 5.5 illustrates the result of applying the toppling rule for $n = 10000$ iterations. The plots use the same simulation parameters, except for the repose angle which takes on the values $\theta_r = 0°$ and $\theta_r = 10°$ for the left and right plot, respectively. In reference to Eq. (5.6), the following initial conditions are used $l_0 = m_0 = 100$, $Q_{d,0} = 5000$, and, $Q_{cbj,0} = 1$. The large plots show the erodible sand height normalized by its max height $Q_d/\max(Q_d)$, and the inset plots show the concentration Q_{cbj}. This figure was produced using MPI with 4 CPUs.

Now consider the result of $\theta_r = 0°$. Calculating $\sum_{[l,m]} Q_d^{(n)}[l,m]$ at $n = 10000$, yields 5000.0 m, implying that the system mass is conserved. The average cell Q_d value, and the standard deviation are correspondingly $\bar{Q}_d \approx 12.5\,\text{cm}$ and $std(Q_d) \approx 3.40\,\text{cm}$. The minimum and maximum grid Q_d values are respectively, 4.17 cm and 16.5 cm. Calculating the system mass after $n = 10000$ iterations, for $\theta_r = 10°$, also yields 5000.0 m.

In Figure 5.6, a cross section of $Q_d^{(n)}$, through the center of the sediment pile is plotted for some selected time iterations n. The cross section corresponds to the horizontal dotted lines shown in each of the plots in Figure 5.5. In the $\theta_r = 10°$ plot, the sediment is observed to form a "hill", with peak at $m = 100$ and base at $m = 70$ and $m = 130$. At $n = 10000$, the peak of the hill has a height 5.32 m. Using the calculated values $Q_d^{(10000)}[l_0, 69]$ and $Q_d^{(10000)}[l_0, 100]$, the average incline is calculated to be $\bar{\theta} \approx 10.1°$.

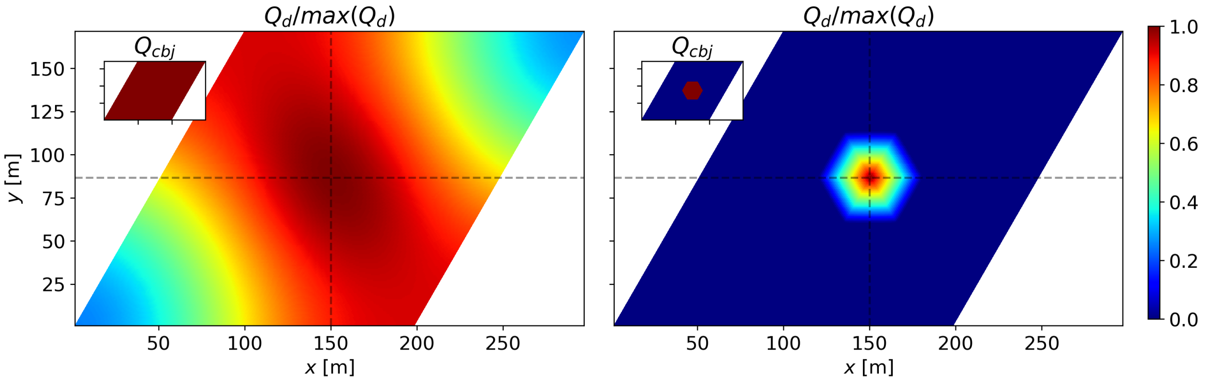


Figure 5.5: The plots illustrate the results of the slope failure model on a 200 × 200 cell grid, with initial conditions as described in Section 5.6.1, and $\theta_r = 0°$ (left figure) and $\theta_r = 10°$ (right figure). The colorbar indicates the value of Q_d and Q_{cbj}. These figures were captured after 10000 time steps. The horizontal and vertical lines indicate the position of $[l_0, m_0]$.

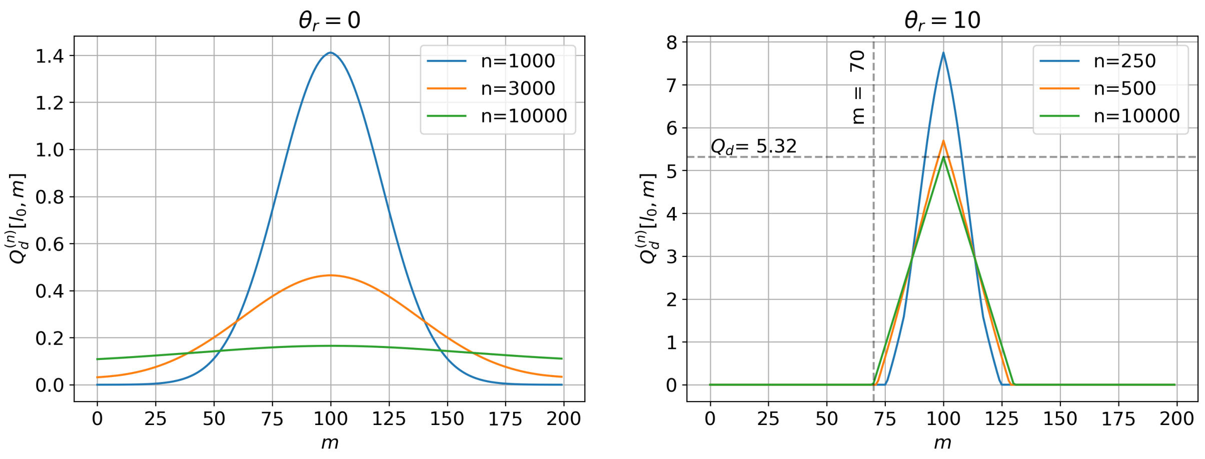


Figure 5.6: The figures show $Q_d^{(n)}[l_0, m]$, i.e., a cross section of Q_d through the center of the sediment pile (corresponding to the horizontal lines in the plots of Figure 5.5) for an angle of repose $\theta_r = 0°$ (left plot) and $\theta_r = 10°$ (right plot). The legend indicates the time iteration n for which the plot was captured. In the right figure the dotted lines indicate the height of the pile and index where the hill begins for $n = 10000$.

5.6.2 Test scenario in Ranfjorden

Let the cell altitude be $Q_a^{(n)} \equiv Q_d^{(n)} + B$, with $B \neq 0$. Define $B = B(x, y)$, so that B resembles a portion of the Norwegian fjord Ranfjorden, as shown in Figure 5.7. Note that in this case $\Delta x = 50\,\mathrm{m}$. This bathymetry can be selected in the simulation by using the configuration `bathymetry=ranfjorden` in the config file.

Figure 5.8 shows the sediment distribution $Q_d^{(n)}$, and concentration $Q_{cbj}^{(n)}$ (inset plots) after applying the slope failure model with the Q_a specified, for $n = \{500, 750, 1000, 10000\}$ iterations. In this case $\theta_r = 0°$, and the initial conditions are given by Eq. (5.6), with $Q_{d,0} = 5000$, and, $Q_{cbj,0} = 1$. The source area is selected by specifying the UTM 33 coordinates $E = 461022$, $N = 7356967$, which for $\Delta x = 50\,\mathrm{m}$ corresponds to indices $[l_0, m_0] = [161, 100]$, and is indicated by the red dot in the Figure 5.7. The data used in Figure 5.8 was collected by running I_4 with MPI, using 4 CPUs.

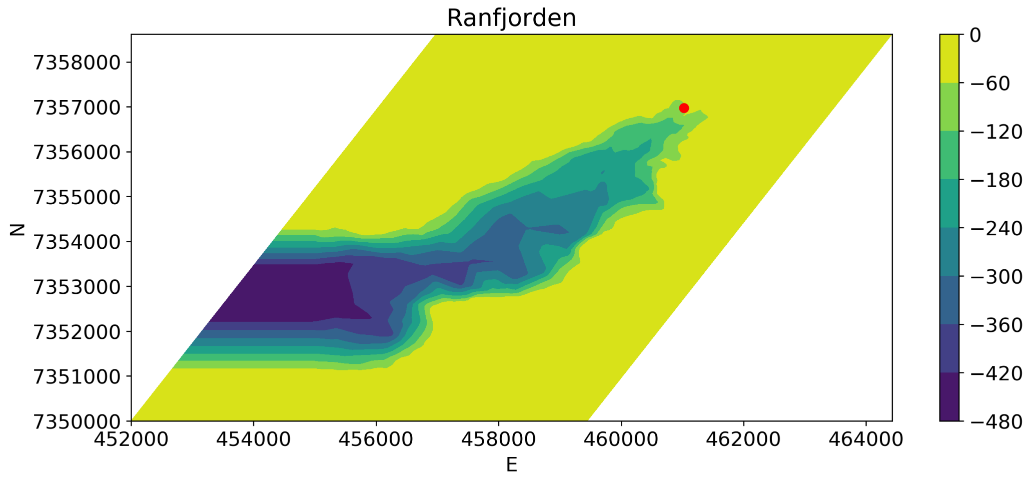


Figure 5.7: This figure illustrates the submarine terrain (bedrock) B used in Section 5.6.2. The colorbar indicates the height of the bedrock relative to the sea surface. Zero height indicates either the approximate shoreline location, or areas not included in the bathymetry data. The red dot indicates the source area.

5.6.3 Demonstration of inverse halo-exchange

The Figure 5.9 illustrates how the toppling rule I_4 would behave without employing the inverse halo-exchange (see Section 4.5.2), while using MPI. Both figures use the initial condition described in Section 5.6.1, with $\theta_r = 0°$. In the left figure the sediment Q_d is localized to just one of the MPI ranks, and any sediment sent to a border/halo cell will be overwritten by the "normal" halo-exchange (see Section 3.2.3), and lost to the system. In the right figure the sand spreads to the entire grid, implying that mass is transferred across ranks.

5.6.4 Discussion of the slope failure model

Consider the case with no bathymetry, and angle of repose $\theta_r = 0°$. In the left plot of Figure 5.5, both Q_d and Q_{cbj} indicate that sediment Q_d has been propagated from the initial cell $[l_0, m_0]$ to all cells in the grid.

The expected result in this case, is a steady state in which the sediment resembles a completely level, horizontal surface. With a constant amount of sediment in the system,

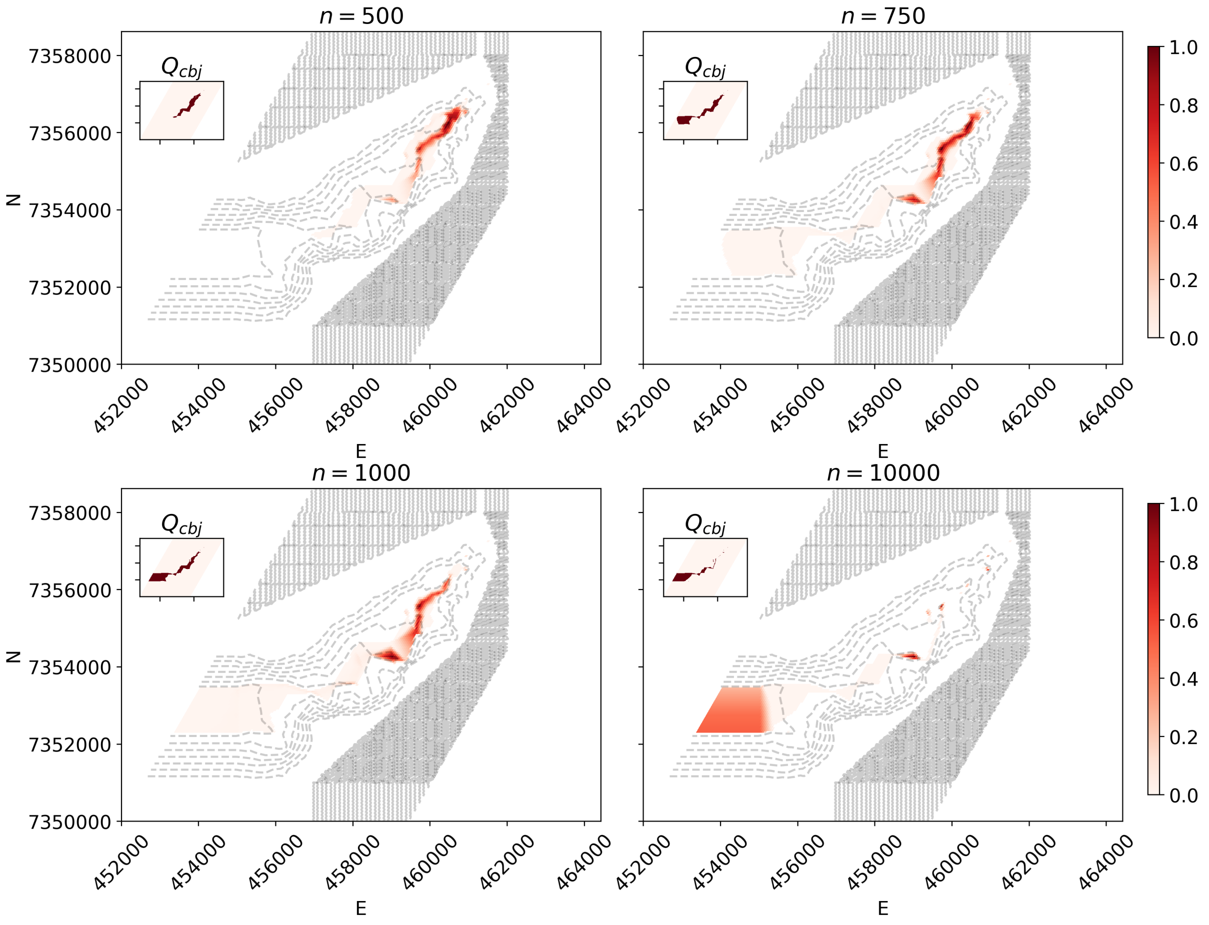


Figure 5.8: The plots illustrate the results of the slope failure model on a 200 × 150 cell grid, with initial conditions as described in Section 5.6.2. The large plots show $Q_d/\max(Q_d)$, and the insets show Q_{cbj}. The angle of repose θ_r is 0°, and the colorbar indicates the value of $Q_d/\max(Q_d)$ and Q_{cbj}. The title indicates at which iteration n the figure was captured.

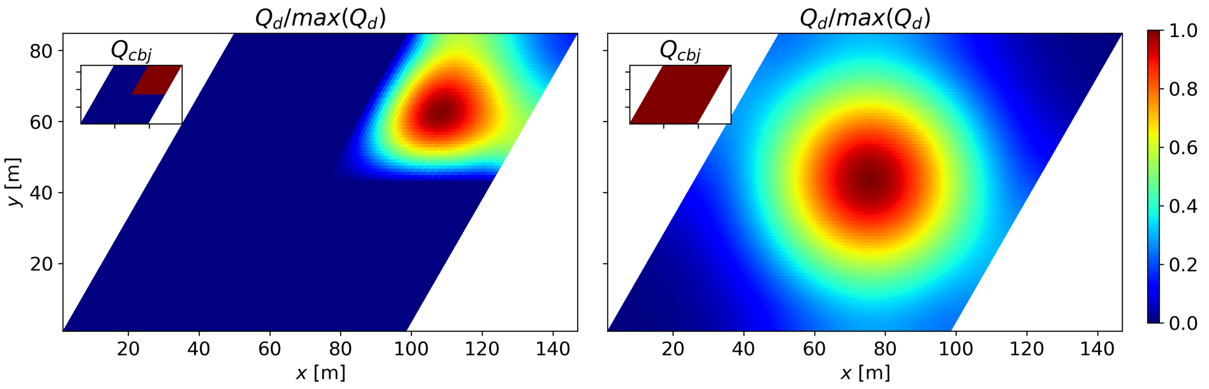


Figure 5.9: The figures illustrates the result of applying the toppling rule, without (left figure) and with (right figure) the inverse halo-exchange, described in Section 4.5.2. The large figures display the normalized sand height $Q_d/\max(Q_d)$, at time iteration $n = 1000$, and the inset figures display the concentration Q_{cbj}. The colorbar is valid for both the large and inset figures.

the initial sediment height of 5000 m is expected to be equally distributed among the 200×200 interior cells. Thus, the steady state of the system is achieved when every cell has a sediment height $\frac{5000\,\mathrm{m}}{200^2} \approx 12.5\,\mathrm{cm}$. For $n = 10000$, the cell average was computed to 12.5 cm, with a standard deviation of $std(Q_d) \approx 3.40\,\mathrm{cm}$, indicating that most of the cells have a value close to the steady state value. Further iteration of the CA is expected to decrease the standard deviation, i.e., $std(Q_d) \to 0$ as $n \to \infty$. However, due to the angle between cell heights θ (see Section 4.4.6) decreasing as steady state is approaching, the fraction of mass to be transferred in each iteration (cf. Eq. (4.32)) is also decreased. This implies that the majority of sediment redistribution occurs for low n, since the height differences, and thus θ is greatest for those iterations. The evolution of the cross section (left plot in Figure 5.6), confirms this. The pile of sediment shrinks significantly during the first 3000 iterations, compared to the change between 3000 and 10000 iterations. In the evolution of the cross section, the diffusive behaviour of the toppling rule is easily seen.

In Figure 5.5, Q_d does not appear to be isotropic when viewed from the initial cell, contrary to ones expectations. This is a consequence of the fact that the initial cell is closer to the top-left and bottom-right corners than the top-right and bottom-left corners. As the walls are impenetrable ($Q_d, Q_a \to \infty$ at boundary), sediment will build up near the closest walls more quickly.

Considering now the case with no bathymetry and $\theta_r = 10°$. With the angle of repose $\theta_r = 10°$, the steady state ($n \to \infty$) should be a 10° hill of sediment Q_d, according to the theory in Section 2.3.1. This is confirmed both by visual comparison with Figure 5.5, and by calculating the average incline of the hill, which was found to be 10.1°. The concentration Q_{cbj} plot confirms that all sediment is confined to a "hexagonal shape" about the initial cell $[l_0, m_0]$.

The hexagonal shape of the sediment pile, Q_{cjb} and Q_d in Figure 5.5, is probably due to the hexagonally shaped cells used in the implementation, and the fact that height differences are only compared in 6 directions.

For the results using Ranfjorden (Figure 5.7) as bedrock, the expected behaviour is for most of the sediment Q_d to move in the direction of greatest descent. The results (Figure 5.8) indicate that the sediment follows a path that is reasonable, when considering the indicated bathymetry contour lines. The concentration Q_{cbj} plots indicate that as n increases, most cells at high altitude have lost their sediment.

6 | Testing the complete cellular automaton

In this chapter the implemented cellular automaton model for turbidity currents will be tested. The tests consists of several simplified scenarios, for which there exist other models or results to compare with.

6.1 Test case 1: 1D channel

This section contains information about the configuration that was used to test the implemented CA against the behaviour of the three equation model (TEM), briefly described in Section 2.1.4. The test was inspired by the paper by Hu et al. [16], and consists of running two simulations with similar initial conditions. TEM predicts that one of the ICs will "ignite" a self-accelerating turbidity current, while the other will not. The TEM predicts the steady state ($n \to \infty$) of the turbidity current speed Q_v and mass transport rate ψ, and is shown in Figure 4 Ref. [16]. Note that the results in this section were acquired using 1 CPU, i.e., the non-parallelized CA.

6.1.1 Configuration, initial and boundary conditions

In this case, the grid G is a hexagonal grid with 1500×3 cells, with boundary conditions specified such that the turbidity flow is constricted between the two grid boundaries in the downstream direction, i.e., the sides of length 1500. Thus, $Q_d^{(n)}[\text{border}], Q_a^{(n)}[\text{border}] \to \infty$ is used for these boundaries, and absorbing boundaries are used for the short edges, following the description in Section 4.1.2. This effectively provides a one-dimensional domain in which the turbidity flow can move. One particle type is used, i.e., $N_j = 1$, and the parameters are specified in Table 6.1. Note that of the parameters listed in Table 6.1, D_{sj}, v_{sj}, ν, ρ_j, ρ_a, and c_D were specified in Ref. [16], and the rest were chosen through numerical experimentation. Simulation config files are given in Appendices A.2 and A.3.

These simulations were run with an upper limit constraint on the time step Δt. That is, the dynamically calculated time step defined in Section 4.3 is compared against a predefined maximum value τ, such that $\Delta t \leq \tau$ always is satisfied. Here τ is experimentally determined to 1 ms. This is discussed in Section 6.1.3.

In this case, a plane with slope $s = 0.05$ is used as the layer of non-erodible bedrock $B = B(l)$. By the cell geometry shown in Figure 2.2, observe that $B(l)$ can be defined by,

$$B(l) = \frac{\sqrt{3}}{2}\Delta x \tan(s) l, \tag{6.1}$$

Table 6.1: This table shows parameter values used by the simulations in the 1D channel case.

Description	Parameter	Values
Spatial discretization	Δx	1 m
Unmovable amount of density current	p_{adh}	0 m
Friction angle limit	θ_f	0°
Angle of repose	θ_r	30°
Gravitational acceleration	g	9.81 m/s^2
Particle diameter	D_{sj}	0.1 mm
Particle sinking speed	v_{sj}	0.0084 m/s
Particle density	ρ_j	2650 kg/m^3
Water density	ρ_a	1000 kg/m^3
Sea bed porosity	γ	0
Sea bed drag coefficient	c_D	0.004
Water kinematic viscosity	ν	$1 \cdot 10^{-6}$ m^2/s

where l is the index increasing in the downstream direction.

As mentioned, two sets of initial conditions, acquired from Ref. [16], are used in this simulation. Let $IC1$ and $IC2$ denote the set of ICs generating a non-ignitive and ignitive turbidity current, respectively. Some of the IC values are common for the two sets. Initially, the entire grid interior is covered by an erodible sand cover $Q_{d,0}$ of height 1 m. As $N_j = 1$, there is only one particle type in the sea bed, and so $Q_{cbj,0} = 1$. Let the source area S be defined as a single cell with indices $\left[\tilde{l}, \tilde{m}\right] = [50, 1]$, that is approximately 50 m away from the edge of the grid. Then, the remaining substate values are given by,

$$Q_\xi^{(0)}[l,m] = \begin{cases} 0 & \text{if } [l,m] \notin S \\ Q_{\xi,0} & \text{if } [l,m] \in S \end{cases} \quad \text{with } \xi = \{th, v, cj\}, \tag{6.2}$$

where $Q_{\xi,0}$ are constants specified for each member of ξ. Specific ξ values for $IC1$ and $IC2$ are given in Table 6.2. Note the sediment transport rate ψ provided in Ref. [16] is replaced by an initial volumetric concentration Q_{cj} through the relation Eq. (2.11). Both ψ and the calculated Q_{cj} is shown in Table 6.2.

Table 6.2: This table shows the numerical values used for the sets of initial conditions $IC1$ (non-ignitive) and $IC2$ (ignitive) in test case 1.

$\xi \backslash \mathrm{Q}_{\xi,0}$	$IC1$	$IC2$
th	2 m	1 m
v	0.652 m/s	0.699 m/s
$c0$	0.00291	0.00672
ψ	0.0038 m^2/s	0.0047 m^2/s

The TEM results (figure 4 Ref. [16]), display the turbidity current speed Q_v, and the mass transport rate ψ in its steady state, i.e., at time $t \to \infty$. As an indicator of steady state, the two-norm of the change in turbidity current thickness $L_2 = \sqrt{\sum_{[l,m]}(Q_{th}^{(n)} - Q_{th}^{(n-1)})^2}$, was calculated. The steady state is reached at $L_2 \leq tol$, where tol is a predefined tolerance.

6.1.2 1D channel results

For both sets of initial conditions, the amount of simulated time was 100.0 s, in $n = 100000$ CA iterations. The source cell was active for all n. At this point the calculated two-norm was 0.029 m, and 0.085 m, for $IC1$ and $IC2$, respectively.

Figure 6.1 shows the dependence on the downstream coordinate l, for the change in bed soft sediment $\Delta Q_d^{(n)} = Q_d^{(n)} - Q_d^{(0)}$, the turbidity current speed $Q_v^{(n)}$, and the sediment transport rate $\psi^{(n)}$, at $n = 100000$ CA iterations, using the $IC1$ configuration. Figure 6.2, shows the corresponding results for the $IC2$ configuration.

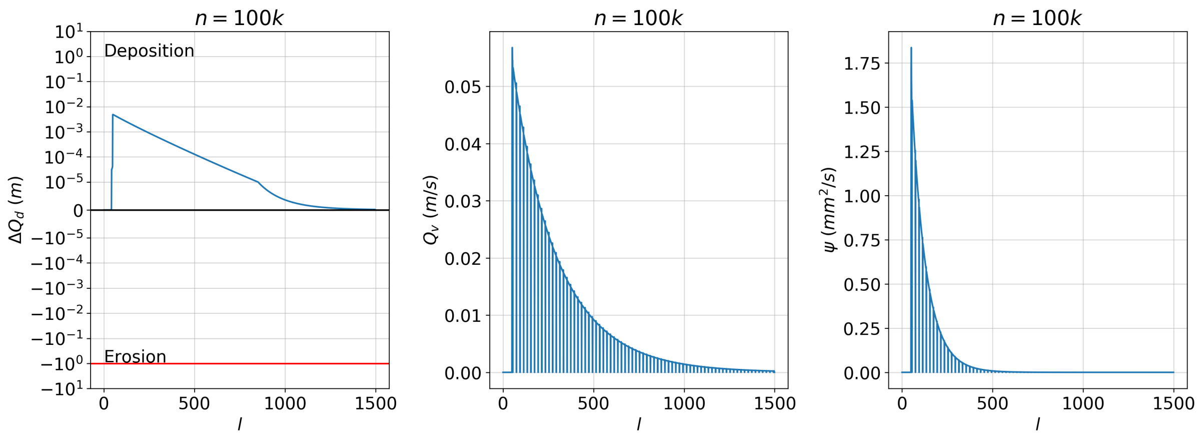


Figure 6.1: This figure shows the result of simulating an 1D channel using the implemented CA with the initial conditions $IC1$. TEM (see Section 2.1.4) predicts that these ICs produce a turbidity current that terminates after < 500 m. From left to right the figures show respectively, the bed evolution $\Delta Q_d^{(n)} = Q_d^{(n)} - Q_d^{(0)}$, the turbidity current speed $Q_v^{(n)}$, and the sediment transport rate $\psi^{(n)}$ at $n = 100000$ CA iterations. The red line in the ΔQ_d plot indicates $Q_d = 0$.

6.1.3 Discussion

Recall that a self-accelerating turbidity current with speed Q_v, and a mass transport rate ψ, is characterized by $Q_v, \psi \to \infty$ as the current propagates downstream (see Section 2.1.4). By visual inspection of the turbidity current speed Q_v, and mass transport rate ψ, it is evident that neither the initial condition $IC1$, nor $IC2$, produce a self-accelerating turbidity current. Both ICs produce results in which $Q_v, \psi \to 0$. Furthermore, the speed and mass transport rate is observed to oscillate violently between a finite value and zero. The speed is influenced by the flow between a cell and its k'th neighbor $Q_o(0,k)$, and the height difference between them $(Q_a(0) + Q_{th}(0)) - (Q_a(k) + Q_{th}(k))$ (see Section 4.4.5). For the speed Q_v to become zero, either of these terms have to become zero. The outflow is also influenced by the difference in Q_a, and Q_{th}. Thus, zero speed Q_v, may be due to $(Q_a(0) + Q_{th}(0)) \approx (Q_a(k) + Q_{th}(k))$. For small slopes, $\Delta Q_a(k) = Q_a(0) - Q_a(k) \approx 0$, this implies that the turbidity current thickness of the

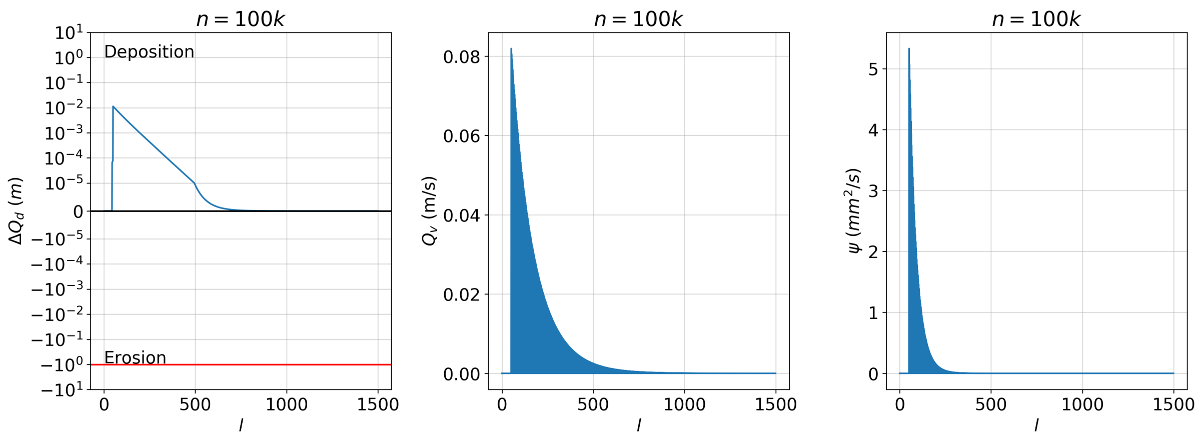


Figure 6.2: This figure shows the result of simulating an 1D channel using the implemented CA with the initial conditions $IC2$. TEM (see Section 2.1.4) predicts that these ICs produce a self-accelerating turbidity current. From left to right the figures show respectively, the bed evolution $\Delta Q_d^{(n)} = Q_d^{(n)} - Q_d^{(0)}$, the turbidity current speed $Q_v^{(n)}$, and the sediment transport rate $\psi^{(n)}$ at $n = 100000$ CA iterations. The red line in the ΔQ_d plot indicates $Q_d = 0$.

two cells is similar, $\Delta Q_{th}(k) = Q_{th}(0) - Q_{th}(k) \approx 0$. This is a known weakness of the Chézy formula: at zero slope, the speed becomes zero.

In order to achieve $Q_v \to \infty$ (and ψ, through Eq. (2.11)), either of the factors above has to grow to a very large value. Which for small bathymetry slopes, $\Delta Q_a(k) \approx 0$, implies that $\Delta Q_{th}(k)$ has to be large. As Q_{th} tends to grow downstream due to water entrainment, $\Delta Q_{th}(k)$ is expected to become negative in the downstream direction. However, for negative $\Delta Q_{th}(k)$ and $|\Delta Q_{th}(k)| > |\Delta Q_a(k)|$, the speed Q_v becomes zero, due to the slope definition Eq. (4.28) (see Section 4.4.5). This is an argument for using the mentioned alternative slope definition Eq. (4.30), as the speed Q_v can be greater than zero for negative $\Delta Q_{th}(k)$. The implemented slope definition was selected for reasons that will be elaborated in the subsequent Section 6.2.

An alternative to the Chézy formula, is to integrate the acceleration of the turbidity current in time. This way, the $(n+1)$'th iteration speed depends on the (n)'th iteration speed, and Q_v can be nonzero at zero slopes. The alternative model is derived by Mulder et al. in Ref. [38]. Replacing the present speed model I_3, with the model by Mulder et al., could be a solution for the oscillating speeds.

For these simulations, the dynamically calculated time step Δt was constrained by $\Delta t \le \tau$. Unphysical behaviour of the turbidity current was observed when using the unconstrained Δt, in that the cellular automaton was not able to propagate the current at a rate faster than what was added by the source term (see Section 4.1.3). This resulted in the current thickness Q_{th} assuming a "bell-shaped" curve about the source, that grew in magnitude for each iteration, i.e., $\max Q_{th}^{(n)} > \max Q_{th}^{(n-1)}$. This seemingly numerically unstable behaviour propagated to other substates, and for instance resulted in speeds Q_v oscillating between $\sim 80\,\text{m/s}$ and $0\,\text{m/s}$. With the restriction on Δt, the current thickness does not accumulate in one cell, but is distributed to the grid.

The values assumed by Q_v and ψ, is not on the scale of the specified initial conditions/source values. One reason for this is the way source cells S are updated (see Section 4.1.3). The amount of turbidity current added, is proportional on the time step Δt. Attempts at running this simulation with constant source cell values, i.e., replacing the expressions in Section 4.1.3 with $Q_{th}[S] = Q_{th,0}$, $Q_{cj}[S] = Q_{cj,0}$, and $Q_v[S] = Q_{v,0}$, yielded a similar problem as described above, where the thickness of the turbidity current increased faster than what the CA was able to move, i.e., the outflow calculated outflow Q_o becomes too small. Attempts at forcing greater cell outflow by modifying the lower boundary of the relaxation factor p_r (see Section 4.4.3) has been made, but yielded no noticeable difference to the result.

Contrary to what is expected [16] of the two ICs, the speed of $IC2$ drops off faster than the speed of $IC1$. This is simply due to the difference in initial turbidity current thickness $Q_{th,0}$ between the ICs. As the source cell thickness for $IC1$ is larger than it is for $IC2$, this presumably leads to greater outflow Q_o for the $IC1$, and thus a larger Q_v.

Both the bed evolution plots ΔQ_d, reflect the low speed of the turbidity current, as only deposition (and no erosion) occurs. In both cases, the deposited sediment forms a "wedge shape" that reaches a height of approximately 2 cm. The figure may give the impression that such a shape would collapse due to slope failure. But this is not the case, as the ΔQ_d axis is displayed in the log-scale. Assuming that the neighbor cell has sediment height 0 m, a 2 cm sediment peak merely results in a $\arctan 0.02 = 1°$ angle, which is much smaller than the angle of repose $\theta_r = 30°$.

In the CA, each grid cell contains a finite amount of sediment Q_d, while the TEM does not account for changes in the bed (see Section 2.1.4). In the TEM, the bed must thus be an infinite source of sediment. This difference may explain different behaviours of the models in cases where the CA results contain erosion.

The results gathered in this test case, indicate that the CA implementation is not well suited for for predicting the properties of a turbidity current in its steady state. In order for the CA to reliably show the steady state as it is predicted by the three equation model, the implemented rules would probably have to be reformulated. Part of the problem, is believed to lie with the fact that the speed Q_v is merely used as an indicator of the turbidity current energy (see Section 5.5), and not as an actual parameter determining the speed at which the turbidity current moves through cells, i.e., distance.

6.2 Test case 2: Rupert Inlet

Here, the implemented CA model is compared against the results of Salles et al. in Ref. [4]. A simulation configuration resembling the setup "Second conceptual case" in Ref. [4] is used. Recall that the slope s used in local interaction I_3 (see Section 4.4.5) was ambiguously defined in Ref. [4]. In this section, results using the alternate slope definition (Eq. (4.30)) are compared to the implemented definition (Eq. (4.28)).

The bathymetry used in this section is inspired by the "lower reach" of the Rupert Inlet channel, in British Columbia, Canada, as it is described by A. Hay [39]. In this test, the turbidity current is expected to follow a bathymetry resembling a sinusoidal channel. By the figures in Ref. [4], the current is expected to be mostly erosive in the channel thalweg, while depositing sediment at the banks. Three different sized sediment types D_{sj} are used. The distribution of these sediment types are expected to be such that smaller particles D_{sj} are propagated and deposited farther onto the banks, while the coarser particles remain in the thalweg. The results in this section were gathered using the parallelized implementation with 4 CPUs.

6.2.1 Configuration, initial and boundary conditions

Let the CA grid G be a hexagonal grid with 320×120 cells. All grid boundaries are set to absorbing, as described in Section 4.1.2. Note that some of the details about the case were omitted in Ref. [4], and that some parameters used here deviate from the values provided. Any deviation or modifications made to the parameters will be clearly stated, and are discussed in Section 6.2.3.

Three sediment particle types are used, i.e., $N_j = 3$. These particle types are given properties similar to clay, silt, and fine sand, which are listed in Table 6.3. All particle types are assumed to have the density $\rho_j = 2600\,\mathrm{kg/m^3}$. In Table 6.3, the simulation parameters used here are listed alongside the corresponding values used in Ref. [4]. The config file can be found in Appendix A.4.

The bedrock B used in this case is shown in Figure 6.3, and can be selected in the config file by specifying `terrain=salles2`, with a slope of 0.7° (see Appendix A.4). The channel depth and width are 2.75 m and 6 m, respectively, and the channel cross section has a parabolic shape as can be seen in Figure 6.6. More context on Figure 6.6 will be given when the simulation results are presented below. The meanders of the channel are in the shape of a sinousoid with amplitude 3 m and wavelength 80 m. At the end of the channel, is a planar area.

In this case the source area S is defined to be two cells in the center of the channel. These cells have indices $[\tilde{l}_1, \tilde{m}_1] = [5, 64]$ and $[\tilde{l}_2, \tilde{m}_2] = [6, 64]$, and are indicated in Figure 6.3. The source is active for the first $n = 10000$ CA iterations. Using the notation in Section 4.1.2, the initial substate values are specified in Table 6.4. Note that the initial value $Q_{cj,0}$ was not provided in Ref. [4]. An assumption was made that the simulated turbidity current had a volumetric sediment concentration of 9 %, and a relative sediment concentration equal to the provided bed sediment concentration Q_{cbj}.

6.2.2 Results

Figure 6.4 shows the evolution of the soft sediment layer $\Delta Q_d^{(n)} = Q_d^{(n)} - Q_d^{(0)}$, for $n = 2000$, $n = 5000$, $n = 10000$ and $n = 20000$ CA iterations.

The simulated time, i.e., $\sum_n \Delta t^{(n)}$ was 5909 s (about 1 hour and 40 minutes) at $n = 20000$ iterations. The source was active for $n = 10000$ iterations, corresponding to 444 s, and the amount of sediment discharged by the source was about 225.5 tonnes. In Figure 6.5 the concentration C (see Section 4.1.4) of clay, silt, and fine sand deposited after

Table 6.3: This table shows parameter values used in this simulation, as well as the corresponding parameters used by Salles et al. in ref [4]. Values omitted in Ref. [4] are tabulated as "Omitted", and blank table cells indicate that the parameter value is shared in both configurations. All particle types are assumed to have the density ρ_j. Recall that the spatial discretization Δx is defined differently in Ref. [4] (see Section 4.2). The value listed in Ref. [4] (cell apothem) has been converted to the corresponding intercellular distance Δx.

Description	Parameter	This simulation	Ref. [4]
CA grid	G	320×120	160×60
Spatial discretization	Δx	0.5 m	2 m
Unmovable amount of density current	p_{adh}	0.1 m	Omitted
Friction angle limit	θ_f	1°	Omitted
Angle of repose	θ_r	50°	30°
Gravitational acceleration	g	$9.81\,\mathrm{m/s^2}$	
Clay particle diameter	$D_{s,\text{clay}}$	5 µm	
Silt particle diameter	$D_{s,\text{silt}}$	60 µm	
Fine sand particle diameter	$D_{s,\text{fine sand}}$	135 µm	
Particle sinking speed	v_{sj}	By Eq. (2.22)	By Eq. (2.23)[a]
Particle density	ρ_j	$2600\,\mathrm{kg/m^3}$	
Water density	ρ_a	$1000\,\mathrm{kg/m^3}$	
Sea bed porosity	γ	0.3	
Sea bed drag coefficient	c_D	0.003	
Water kinematic viscosity	ν	$1 \cdot 10^{-6}\,\mathrm{m^2/s}$	

[a] Eq. (2.23) is stated used in Ref. [4]. In correspondence with the author, Eq. (2.22) is said to be used.

Table 6.4: This table shows the numerical values used as initial conditions in test case 2.

Description	Substate	Value
Initial erodible sand cover	$Q_{d,0}$	0.5 m
Source TC height	$Q_{th,0}$	2.5 m
Source TC speed	$Q_{v,0}$	0.2 m/s
Initial sand cover concentration	$Q_{cbj,0}$	clay: 0.8 silt: 0.15 fine sand: 0.05
Source TC concentration	$Q_{cj,0}$	$0.09 \cdot Q_{cbj,0}$

$n = 20000$ iterations is illustrated. Figure 6.6 shows the evolution of the channel cross section at indices $l = 5$ and $l = 40$, for various CA iterations n.

Ambiguous slope definition in local interaction I_3

To differentiate between the results gathered using the alternative slope definition, i.e., Eq. (4.30), and the "regular" definition, i.e., Eq. (4.28), let the results be denoted $\widetilde{I}_3$ and I_3, respectively. In Figure 6.7, the bed evolution $\Delta Q_d^{(n)}$, is illustrated at $n = 20000$ iterations, for $\widetilde{I}_3$. The amount of time simulated was 4821 s (about 1 hour and 20 minutes)

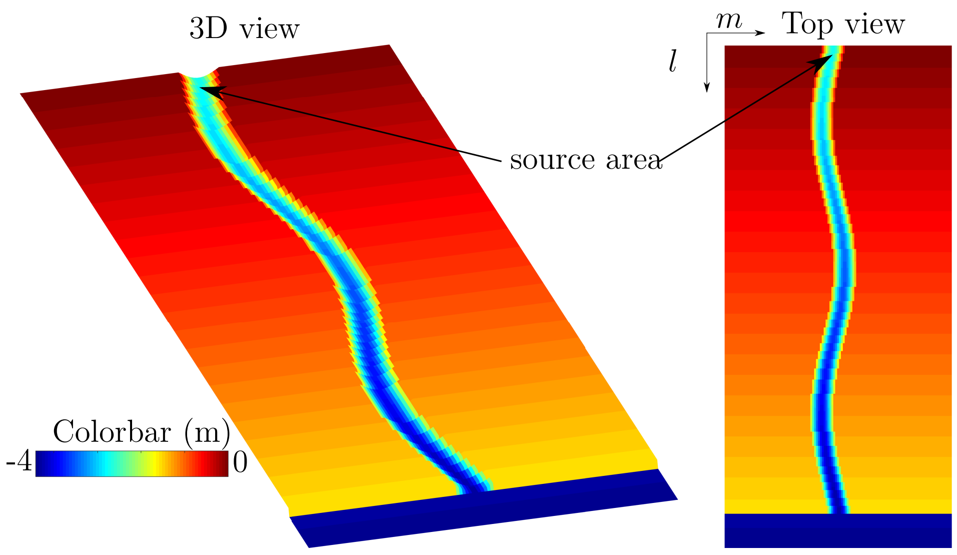


Figure 6.3: The figure shows the bedrock B used in test case 2: Rupert Inlet. The colorbar indicates the height (in meter) of the bedrock. Also indicated is the source area S, and the direction of positive array indices $[l, m]$.

at $n = 20000$ iterations. The source was active for 10000 iterations, corresponding to 606 s and about 307.5 tonnes of sediment was discharged.

The time-averaged cell max speed in cells sharing the m index,

$$\bar{Q}_v[l] = \max_m \frac{1}{N} \sum_{n'} Q_v^{(n')}[l, m],$$

is shown for $\widetilde{I}_3$ and I_3, in Figure 6.8. The value $\bar{Q}_v[l]$, was calculated using $N = 10$ samples of Q_v between $n = 1000$ and $n = 10000$, with a sampling rate of 1000 iterations.

6.2.3 Discussion

Parameter alterations

In this test case, the grid G size of the simulation was doubled, and the spatial discretization Δx was decreased in order to achieve a better resolution output. Initial tests using the original grid size and Δx, yielded results in which the bathymetry channel width consisted of 3 cells. This motivated a higher resolution.

The value of the parameters p_{adh} and θ_f were experimentally determined. Increasing p_{adh} had the effect of slowing the advancement of the turbidity current, i.e., more iterations are needed for the current to move across the grid. Adjusting θ_f seems to influence the "climbing" ability of the turbidity current, as a higher value led to more current spilling out of the channel. This makes sense as θ_f determines how big the height difference between two cells needs to be in order to get an outflow $Q_o(0, k)$ (see Section 4.4.3).

The angle of repose θ_r had to be increased, in order to prevent the channel sides from collapsing entirely, which was the case when $\theta_r = 30$. It is possible that the intent of

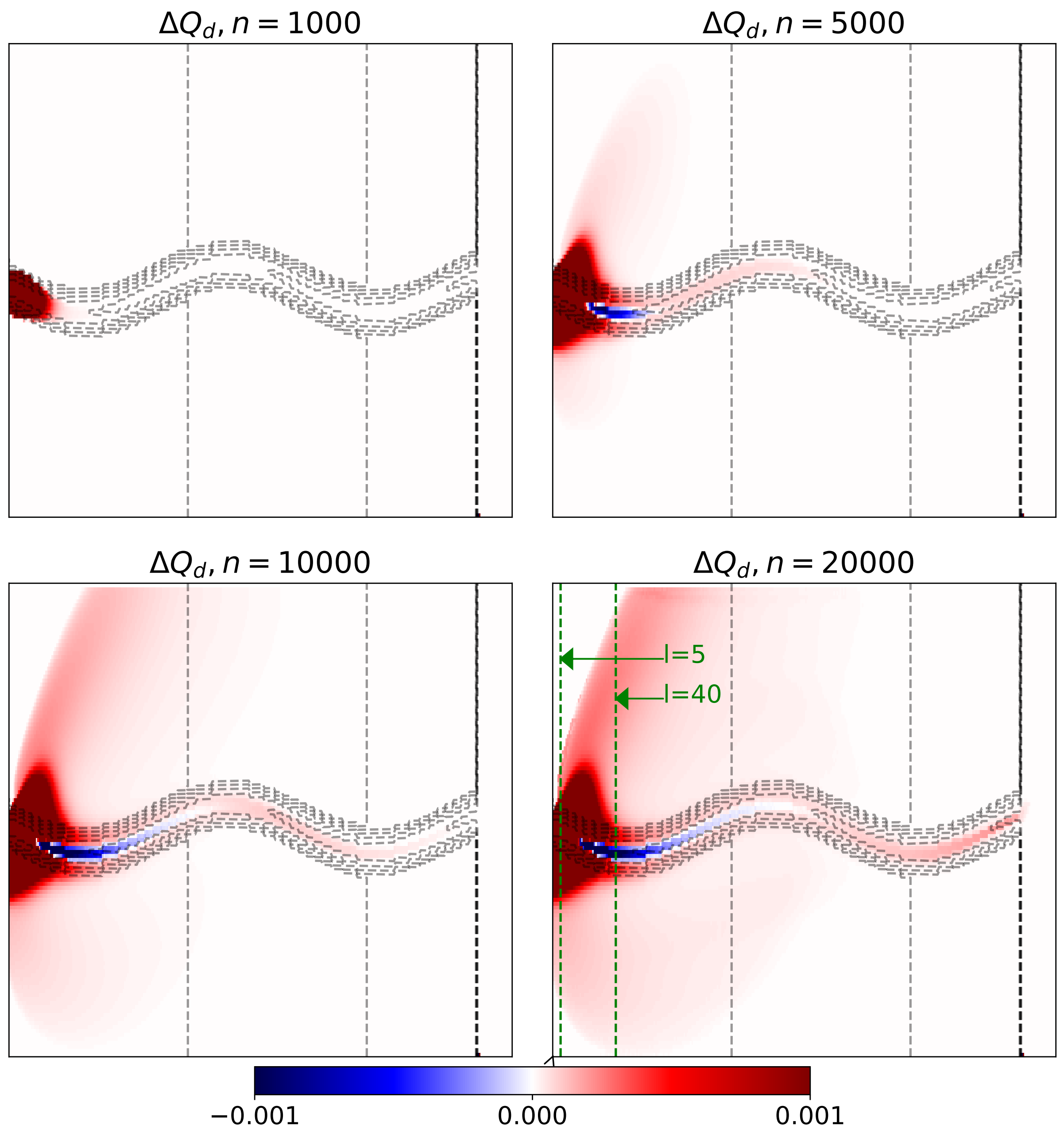


Figure 6.4: This figure illustrates the sea bed evolution ΔQ_d, after $n = 2000$, $n = 5000$, $n = 10000$ and $n = 20000$ iterations, using the CA configuration specified in Section 6.2. The colorbar indicates the eroded (blue) and deposited (red) sediment amount in meter. The locations of cells with array indices $l = 5$ and $l = 40$ are indicated in the $n = 20000$ plot. Note that the range of the colorbar is *not* specified as $[\min \Delta Q_d, \max \Delta Q_d]$, due to the difference in the amount of deposited and eroded sediment.

the toppling rule I_4 (in Ref. [4]), was for it to only be applied to the sediment that is deposited. That is, the initial soft sediment layer $Q_{d,0}$ is assumed to be stable, and thus excluded from I_4. This may involve separating the substate Q_d as it is presently defined, into substates Q_{d1} and Q_{d2}, representing the initial erodible sediment cover, and the deposited sediment, respectively.

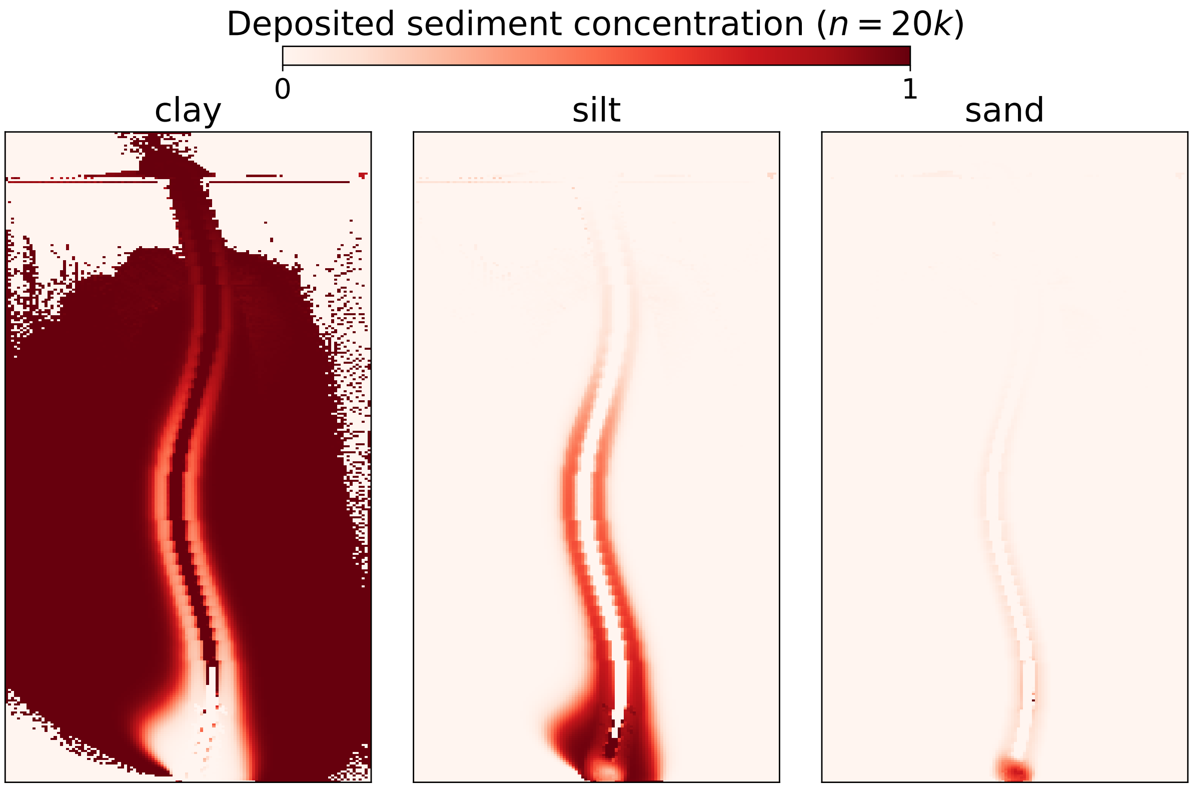


Figure 6.5: This figure illustrates the relative concentration of deposited sediment at $n = 20000$ CA iterations, using the initial conditions described in Section 6.2. The source cell S is near the bottom of the plots, such that the index l is increasing in the upward direction.

The altered sinking velocity is discussed in Section 5.5.

Simulation results

As can be seen in Figure 6.4, the turbidity current displays some erosive behaviour in the channel thalweg, while sediment is deposited at the banks. Close to the source S, there are significant amounts of deposited sediment, both in the thalweg and at the banks. Downstream, near the end of the channel, sediment is deposited inside the channel, implying that the turbidity current does not possess enough speed Q_v to support the particles it contains (see Section 5.2). The area of erosion is, as expected by the results in Ref. [4], located in the thalweg of the bathymetry, and can be seen expanding downstream in the plots $n = 5000$ through $n = 20000$.

The colorbar in the figure displays erosion and deposition in the range $\pm 1\,\text{mm}$. This range was selected in order for the erosive activity of the turbidity current to be visible in the plots. That is, had the range been higher, the eroded area would be displayed as white in the plots. In the corresponding ΔQ_d plot in Ref. [4], erosion is clearly visible on the range $\pm 30\,\text{cm}$. This indicates that the amount of sediment eroded in these results are minuscule in comparison with the results shown in Ref. [4]. In the channel cross-section evolution at index $l = 40$, all sampled $Q_d^{(n)}$ curves overlap, indicating that the channel is approximately unchanged. Index $l = 40$ is indicated in Figure 6.4, and cuts the channel

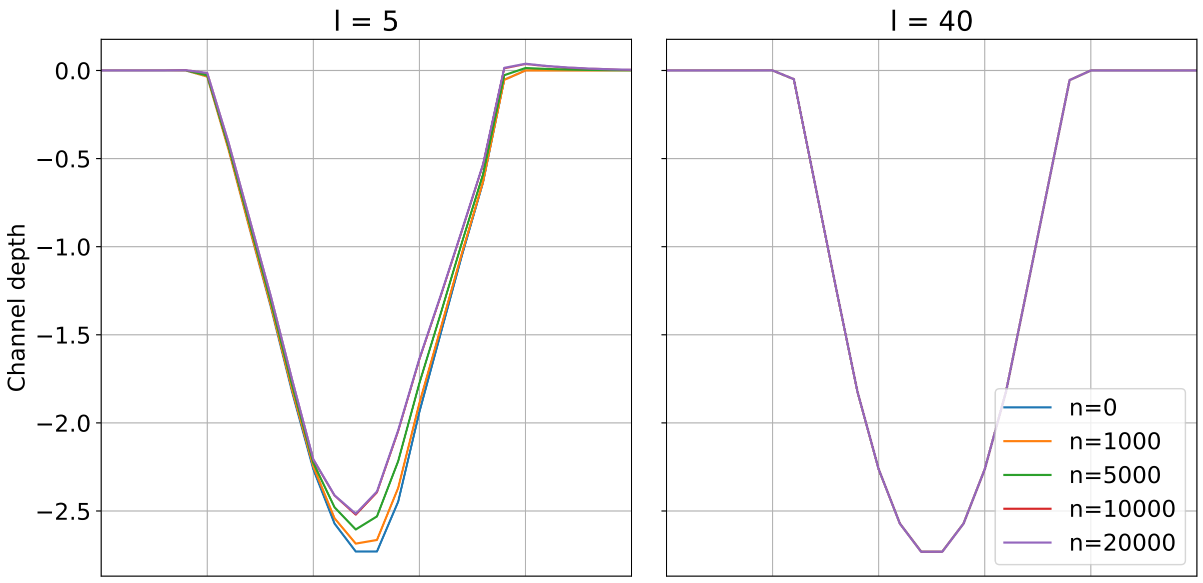


Figure 6.6: This figure shows the evolution of the cross section in the channel at indices $l = 5$ and $l = 40$, for CA iterations n indicated by the figure legend. The location of $l = 5$ and $l = 40$ is shown in Figure 6.4.

where the erosion is strongest.

As was revealed in the Section 5.2, the main contribution to increased erosional activity is the turbidity current speed Q_v. Thus, evidence is strongly suggesting that Q_v is too low for any noticeable erosion. Figure 6.8 shows that the time-averaged cell max speed of the current $\bar{Q}_v[l]$ rapidly drops off, and asymptotically approaches zero as the downstream distance increases. Note that this implies that the current does not fulfill the criterion of a self-accelerating turbidity current, i.e., $Q_v[l] \xrightarrow{l\to\infty} \infty$. From Figure 5.3 it is deduced that a speed Q_v of about 1 m/s to 2 m/s is needed for a current with concentration $Q_{cj} \sim 0.09$, to become erosive. In Figure 6.8, such speeds seems to be barely achieved for cells very close to the source S.

Why does the speed Q_v approach zero? The speed Q_v is computed by averaging speed contributions in each direction (see Section 4.4.5). Should most of the contributions to the average speed in a cell be low, the calculated average becomes low. In the implemented version all the contributions U_k are weighted equally (cf. Eq. (4.29)). It could be that averaging all U_k using the outflow $Q_o(0, k)$ as weights would give a more accurate result. This way, the direction in which most of the current is flowing out of the cell, has a larger influence on the calculated speed Q_v.

As was mentioned in the discussion of the 1D channel, the speed is influenced by the differences $\Delta Q_a(k) = Q_a(0) - Q_a(k)$, and $\Delta Q_{th}(k) = Q_{th}(0) - Q_{th}(k)$. If either ΔQ_a or ΔQ_{th} is small, the speed becomes small. As the slope of the bedrock is 0.7°, the ΔQ_a is small. By the (top) Figure 6.9, the current thickness is observed to be relatively evenly distributed, implying that for neighbor cells ΔQ_{th} is small.

The speed shown in Figure 6.8, does not seem to be in correspondence with the specified initial condition $Q_{v,0} = 0.8$ m/s. This is to be expected, as the source value $Q_{v,0}$ is

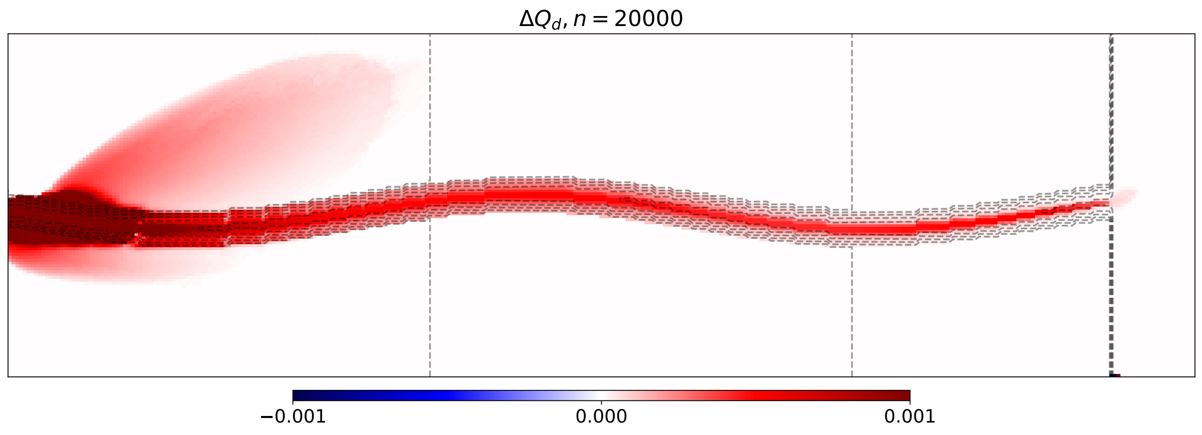


Figure 6.7: This figure illustrates the sea bed evolution as computed after $n = 20000$ CA iterations, using the same CA configuration and initial conditions as in Figure 6.4, except that the alternate definition $\widetilde{I}_3$ (see Section 6.2.2) is used when calculating the turbidity current speed. The colorbar indicates the eroded (blue) and deposited (red) sediment amount in meter.

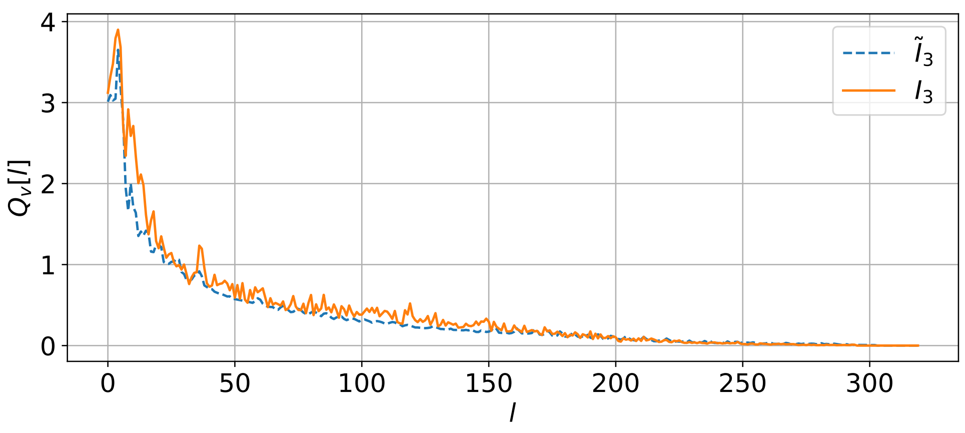


Figure 6.8: This figure shows the time-averaged max speed $\bar{Q}_v[l]$ (see Section 6.2.2) as a function of the downstream cell indices l, for CA implementations using I_3 and $\widetilde{I}_3$.

scaled by the time step Δt (see Section 4.1.3), and as $Q_v^{(n+1)}$ is independent of $Q_v^{(n)}$. As mentioned in Section 6.1.3, a weak point of the speed model implemented in this thesis is that for slopes $s = 0$, the current achieves no speed. A model derived by T. Mulder et al. in Ref. [38], computes the speed Q_v by adding the change due to acceleration. The model is arguably more computationally intensive than the implemented model, but for near horizontal bathymetries, using the acceleration based model could give a more accurate result.

Another possible reason for the low amount of erosion, is the low amount of large particles in the turbidity current. In Figure 5.3, particles with a large diameter D_{sj}, have a much higher erosion rate than small particles. For instance, in a current with speed $Q_v = 0.5\,\mathrm{m/s}$, particles with diameter $\sim 1\,\mu\mathrm{m}$ have a net erosion rate $< 10^{-6}\,\mathrm{m/s}$, while particles with diameter $D_{sj} \gtrsim 10\,\mu\mathrm{m}$ have a net erosion rate of $> 10^{-5}\,\mathrm{m/s}$. In the CA configuration used here, the relative concentration of particles (see Table 6.4) in the turbidity current was assumed to be identical to the concentration in the bed. This

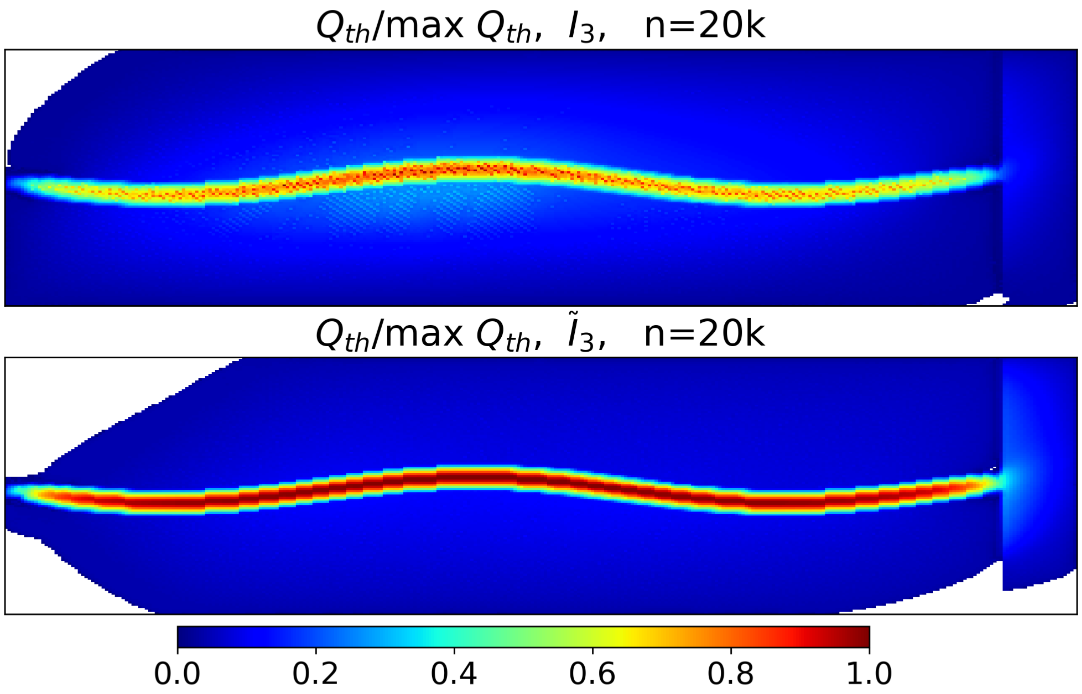


Figure 6.9: This figure illustrates the turbidity normalized turbidity current thickness $Q_{th}^{(n)} / \max Q_{th}^{(n)}$ for I_3 (top) and $\widetilde{I}_3$ (bottom) at $n = 20000$ iterations.

resulted in 20 % of the discharged particles having a diameter > 10 µm, which could yield a low net erosion rate, and thus a low amount of erosion. This means that increasing the number of large particles in the turbidity current, could increase the amount of erosion.

In the area close to the source cell, a relatively large amount of sediment is deposited, as can be seen in Figure 6.4. In the cross section plot at array index $l = 5$, the amount of deposited sediment can be seen by the evolution of the channel bed. In the thalweg, the deposited sediment has decreased the channel depth from 2.75 m, to approximately 2.5 m. And at one side of the channel, an increase in bed height indicates the formation of a levee. The amount of sediment accumulated in the levee is small in comparison to the results achieved in Ref. [4].

The concentration of the deposited sediment C (Figure 6.5), indicates that the particle type with the smallest diameter, i.e., the clay, is the most easily distributed. Approximately 100 % of the sediment deposited far from the channel axis, is clay. When moving closer to the channel axis, the larger particle types starts dominating C, i.e., these particles constitute a larger part of the sediment being deposited. All of this is in compliance with the corresponding plots in Ref. [4]. Notice that farther downstream, clay is deposited inside the thalweg. As mentioned, the main reason for the turbidity current depositing sediment is insufficient speed Q_v to keep the particles in suspension. The low speed in Figure 6.8 explains why sediment is deposited. The reason for the high clay concentration in the deposited sediment, is believed to simply be a consequence of the relatively high concentration of clay in the source concentration $Q_{cj,0}$.

In Figure 6.10, the concentration of the turbidity current concentration Q_{cj} drops signif-

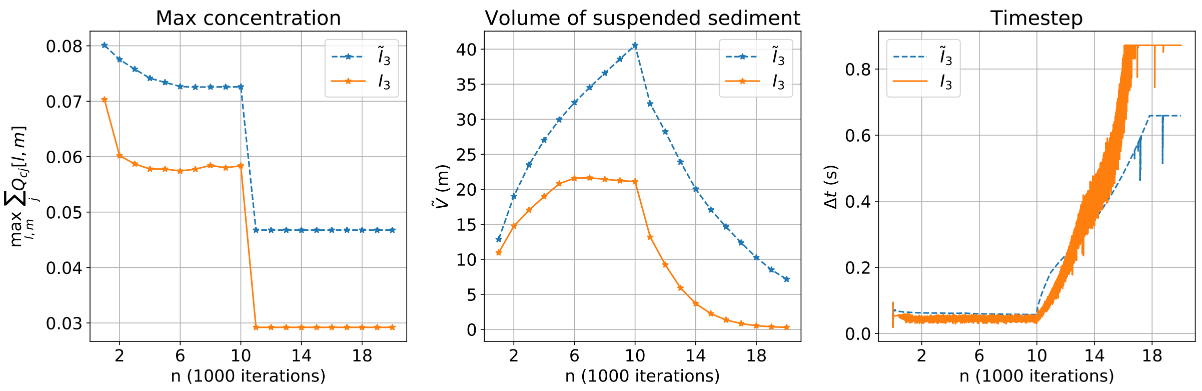


Figure 6.10: The figure illustrates the dependence on n for selected values in the CA. From left to right, the plots show the maximum turbidity current sediment concentration $\max_G Q_{cj}$, the volume $\widetilde{V}$ of suspended sediment in the turbidity current, and the size of the time step Δt as a function of the CA iteration n.

icantly at $n = 10000$ CA iterations, i.e., as the source becomes inactive. This drop can also be seen in the volume of suspended sediment $\widetilde{V}$. Both of these indicate a decrease in the amount of sediment in the turbidity current, in correspondence with the deposition dominated behaviour discussed above.

In the bed evolution plot ΔQ_d, the deposited sediment is observed to have a slight preference to landing in the direction of “increasing l, increasing m” array indices. This must be a consequence of the combination of the way this bathymetry was generated, and the way that the CA hexgrid is stored. The CA grid is stored by using a regular 2D array, using the technique explained in Section 2.2.1. The bathymetry seen in Figure 6.3, is then applied to the 2D array. This induces a skewness to the bathymetry. A solution to this, is to generate the bathymetry in such a way that the induced skewness is accounted for. In other words, the bathymetry would have to be generated such that “south” does not follow the shaded column in Figure 2.2, but rather follows the cells intersected in a straight vertical line. The effect is limited to this test case.

In its present state, the CA simulation displays to some extent the intended behaviour. The particle type distribution C appears to be corresponding well with the results shown in Ref. [4]. However, given that the CA configuration and initial conditions used here and in Ref. [4] are as close to identical as possible, there seems to be a problem concerning the amount of eroded and deposited sediment. This is assumed to be an issue originating from either the speed interaction I_3, or the outflow interaction I_1. The amount of outflow Q_o directly influences the speed in a cell.

Comparison with the alternate speed interaction $\widetilde{I}_3$

When the speed of the turbidity current is calculated using $\widetilde{I}_3$, the bed evolution $\Delta Q_d^{(20000)}$ plot (Figure 6.7) reveals that no erosion is occurring. The reason for this, as is mentioned above, must be due to the low speed Q_v obtained by the turbidity current. When comparing the curves of I_3 and $\widetilde{I}_3$ in the time averaged max speed plot (Figure 6.8), they appear to be of equal shape and magnitude. However, the I_3 curve appears less smooth, with some peaks appearing. These peaks in Q_v is assumed to be the reason that erosion occurs for I_3, but not for $\widetilde{I}_3$.

Also observed in Figure 6.7 is that more of the deposited sediment lands nearby or inside the channel, compared to the results of I_3. The reason for this is that the turbidity current is that the thickness Q_{th} is more intense/concentrated about the channel axis in the $\widetilde{I}_3$ result, while being more dispersed for I_3. This can be seen in Figure 6.9.

Fluctuating values in numerical simulations is generally a bad sign, that can indicate numerical instability. In general, the plots gathered with I_3 are observed to feature more fluctuations than the plots using $\widetilde{I}_3$. This is true for the figures showing Q_v, Q_{th}, and the time step Δt. In the end, comparison of the bed evolution ΔQ_d, and the concentration C was what led to the selection of I_3 over $\widetilde{I}_3$.

7 | Simulating turbidity currents in Ranfjorden

In this chapter the implemented cellular automaton is tested on bathymetry data from Ranfjorden. The bathymetry data was acquired from Kartverket [40], and has a resolution of 50 m between each data point. Figure 7.1 indicates the simulation domain, in relation to the surrounding geography.

To confirm the validity of the simulation, the results are compared to Figure 19 in Ref. [41], in which the bathymetry evolution in Ranfjorden for the period 2012-2016 is depicted. Additional description of the bathymetry is provided in the report, "Tiltaksorientert overvåking av Ranfjorden i 2018" by the Norwegian Institute for Water Research [42], and corresponding reports from previous years [43, 44]. In Ref. [42], sediment samples are collected from several locations in the fjord, and analyzed. Mine tailings are found in several of these locations, the farthest out being near Bustneset (location 4 in Figure 7.1). The fine fraction (FF), i.e., the fraction of particles with diameter $< 63\,\mu\text{m}$, is measured at several locations in the fjord, and is found to be $\geq 80\,\%$ for all locations. The samples were collected from the top 5 cm of the bed. The authors claim the high FF is mostly due to the mining discharge. The lowest fine fractions, were sampled near locations 1, 3 and 4 in Figure 7.1. The authors also claim that the area close to location 3 is less prone to fine grained sedimentation.

7.1 CA configuration, initial and boundary conditions

For this simulation, let the CA grid G be a 700×530 cell grid, with spatial discretization 14 m. The spatial discretization was arbitrarily selected, such that the provided bathymetry fit the grid size. Selecting Δx smaller than the resolution of the provided bathymetry, results in an upscaling. The primary intention behind this, is to shrink the cell area, such that the substate values of the source cells are of an appropriate size. This will be elaborated below. The simulation results were gathered using 25 CPUs.

Similarly to the configuration used in the Rupert Inlet test case (see Section 6.2), three sediment particle types are used ($N_j = 3$), that all have the density $\rho_j = 2650\,\text{kg/m}^3$. The particle types are categorized as fine, medium and coarse grained. All parameters values (and the grid size) are listed in Table 7.1, and the simulation config file is given in Appendix A.5.

The information given in Ref. [42] and [44] forms the basis for some of the initial conditions used in this simulation. Recall that the bathymetry data for Ranfjorden was illustrated in Section 5.6.2 (see Figure 5.7), and that the plot axes corresponds to the eastern (E) and northern (N) UTM 33 coordinates.

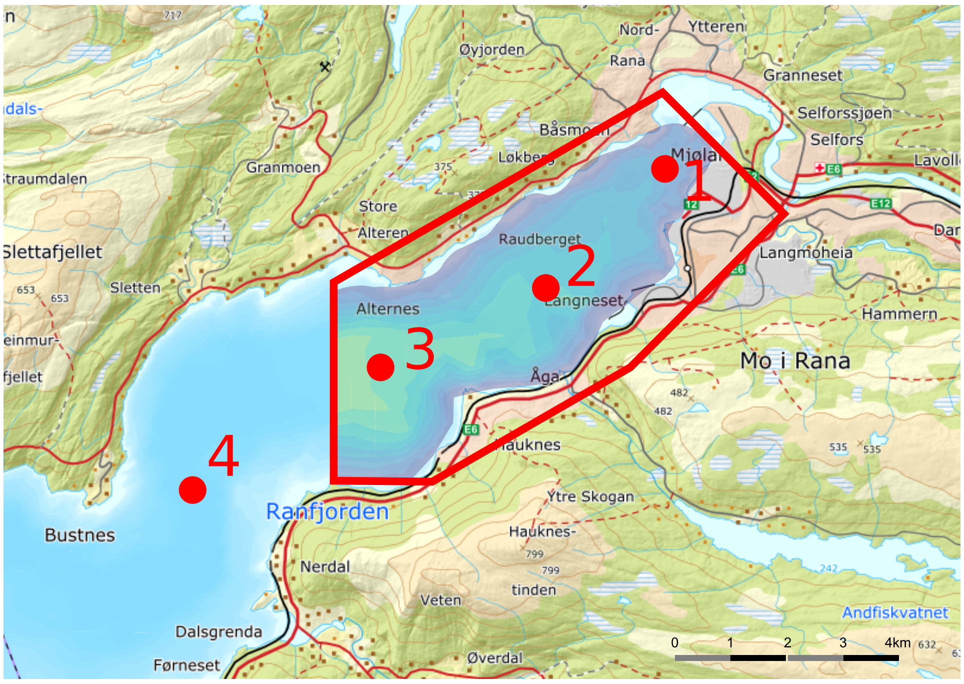


Figure 7.1: The outlined area indicates the area included in the simulation. The red dots indicate points of interest (POIs), that correspond to the approximate location where the fine fraction (FF) was measured in Ref. [42]. The approximate depth for each POI is given. POI 1 (88 m): approximate location of the tailings discharge point, FF = 86 %. POI 2 (341 m): FF = 93 %, POI 3 (427 m): FF = 80 %, POI 4 (490 m): FF = 83 %. Note that the POIs only depict the *approximate* location of the sample areas.

Table 7.1: This table shows CA parameter values (and grid size) used in the Ranfjorden simulation. All particle types are assumed to have the density ρ_j.

Description	Parameter	Value
CA grid	G	700×530
Spatial discretization	Δx	$14\,\mathrm{m}$
Unmovable amount of density current	p_{adh}	$0\,\mathrm{m}$
Friction angle limit	θ_f	$0°$
Angle of repose	θ_r	$50°$
Gravitational acceleration	g	$9.81\,\mathrm{m/s^2}$
Fine particle diameter	$D_{s,\mathrm{fine}}$	$5\,\mathrm{\mu m}$
Medium particle diameter	$D_{s,\mathrm{medium}}$	$80\,\mathrm{\mu m}$
Coarse particle diameter	$D_{s,\mathrm{coarse}}$	$135\,\mathrm{\mu m}$
Particle sinking speed	v_{sj}	By Eq. (2.22)
Particle density	ρ_j	$2600\,\mathrm{kg/m^3}$
Water density	ρ_a	$1000\,\mathrm{kg/m^3}$
Sea bed porosity	γ	0.3
Sea bed drag coefficient	c_D	0.003
Water kinematic viscosity	ν	$1 \cdot 10^{-6}\,\mathrm{m^2/s}$

Let the source S be the cell with indices corresponding to coordinates $E = 461022$, $N = 7356967$, which for $\Delta x = 14\,\text{m}$ is the cell with indices $[l_0, m_0] = [575, 357]$. The source is active during the first 10000 CA iterations, and the red dot in the Figure 5.7 shows the location of the source cell.

For this scenario, the substate values of the source cells is calculated using the emission permits from 2014 for the mining company Rana Gruber AS, provided in Ref. [44]. As a simplification, the discharged sediment is assumed to consist of a mix of the coarse, medium and fine material, in the same proportion as given in the permit. The discharged material, and the relative amount is given by Table 7.2.

Table 7.2: This table shows the 2014 permit for fine, medium and coarse sediment types in Ranfjorden by Rana Gruber AS [44], and the calculated relative particle concentration.

Particle type	Discharge permission (kilo tonnes)	Relative amount (%)
Fine	60	2
Medium	290	9.6
Coarse	2650	88.4
Total	3000	100

The source values are calculated using the following assumptions. The amount of discharged material D per time unit equals the permitted amount, i.e., 3 Mt/yr, which becomes 95 kg/s on average, and the volumetric concentration of the emitted material is $Q_{cj,\Sigma} = \sum_j Q_{cj} = 0.01$. Then, if the density of the sediment $\rho_j = 2650\,\text{kg/m}^3$, the thickness of the turbidity current source becomes,

$$Q_{th,0}\Delta t = \frac{D}{A_{hex} Q_{cj,\Sigma} \rho_j} \Delta t \approx 2.1\,\text{cm} \cdot \Delta t.$$

Here, $A_{hex} = \frac{\sqrt{3}\Delta x^2}{2}$ is the area of a cell in the grid. Notice that had Δx been set to the "native" resolution of the bathymetry data, i.e., 50 m, the source value would be about 0.16 cm.

Thus, using the notation in Section 4.1.2, the substate initial, and source values are given by Table 7.3. In Ref. [42] the relative amount of fine grained (particle diameter $\leq 64\,\mu\text{m}$) found on the sea bed was $\geq 80\,\%$. For this reason, the sea bed concentration Q_{cbj} has been chosen to reflect this. The values $Q_{d,0}$ and $Q_{v,0}$ were arbitrarily chosen.

7.2 Results

The amount of time simulated $\sum_n \Delta t^{(n)}$ was 486 979 s (about 135 hours) in 20000 iterations, and about 8.3 kt (kilo tonne) of suspended sediment was discharged during the first 24 hours (10000 iterations) of simulation time.

Figure 7.2 shows the evolution of the soft sediment layer $\Delta Q_d^{(n)} = Q_d^{(n)} - Q_d^{(0)}$, for $n = \{5000, 10000, 15000, 20000\}$ CA iterations, and in Figure 7.3, the volume (in meter)

Table 7.3: This table shows the numerical values used as initial conditions for the Ranfjorden simulation.

Description	Substate	Value
Initial erodible sand cover	$Q_{d,0}$	1 m
Source TC thickness	$Q_{th,0}$	2.1 cm
Source TC speed	$Q_{v,0}$	2 m/s
Initial sand cover concentration	$Q_{cbj,0}$	$\begin{cases} \text{fine: } 0.8 \\ \text{medium: } 0.10 \\ \text{coarse: } 0.10 \end{cases}$
Source TC concentration	$Q_{cj,0}$	$\begin{cases} \text{fine: } 0.01 \cdot 0.02 \\ \text{medium: } 0.01 \cdot 0.096 \\ \text{coarse: } 0.01 \cdot 0.884 \end{cases}$

of the deposited, and eroded sediment at iteration n is plotted. Note that the volume is given in meter, as the factor A_{hex} is omitted. The maximum volume of sediment deposited is 15.8 m, which corresponds to $15.8\,\text{m} \cdot A_{hex}\rho_j \approx 7.1\,\text{kt}$ of deposited sediment.

The particle type distribution of the deposited sediment C (see Section 4.1.4) is shown in Figure 7.5. Figure 7.6 shows the n dependence of the maximum volumetric concentration in the grid, $\max_{[l,m]} Q_{cj}$, the "volume" of the suspended sediment $\tilde{V} = \sum_{[i,j]} Q_{th}Q_{cj}$, and time step value $\Delta t^{(n)}$. As above, the plotted $\tilde{V}$ is the volume of the sediment measured in meter, as the factor A_{hex} is omitted. At $n = 20000$, the volume $\tilde{V}$ is 0.33 m, which corresponds to about 148 t of suspended mass.

7.3 Discussion

From the bed evolution plot (Figure 7.2), it is observed that the simulated turbidity current (TC) is mostly depositing sediment, with the exception of some erosional activity appearing at $n = 20000$ iterations. It appears that most of the sediment being deposited, ends up tracing out a somewhat continuous path, stretching from the source cell S, and into the deeper parts of the fjord. Considering the contour lines of the bathymetry, the path follows what seems to be the fastest descending curve. This much is to be expected, according to the results of Ref. [41]. Additionally, the area experiencing erosion at $n = 20000$ iterations, does seem to coincide with a larger area where erosion has occurred during the period 2012-2016, according to Ref. [41].

Running alongside the path of deposited sediment, however, there is expected to be a trench, where the turbidity current has eroded the sediment. This is not observed in the obtained results. In order for the current to be erosive, the current speed Q_v should be above a certain threshold, which for currents of volumetric concentration $Q_{cj} \sim 0.01$, is about 1 m/s to 2 m/s (see Figure 5.3). This indicates that the turbidity current does not achieve sufficient speed Q_v for it to be erosive. As can be seen in Figure 7.4, the speed Q_v has a maximum value of about 0.9 m/s, which it achieves at $n = 10000$ iterations.

However, as can be seen by comparing Figure 7.3 (right) and Figure 7.4, some erosion does occur even when Q_v is $< 1\,\mathrm{m/s}$. This is probably due to very diluted parts of the turbidity current with concentration $Q_{cj} < 0.01$. As can be seen in Figure 5.3, as the concentration Q_{cj} decreases, the amount of speed Q_v needed to become erosive decreases.

The colorbar in Figure 7.2 is on the order of µm, giving the impression that the amount of deposited sediment is small. An increase in cell height by 5 µm corresponds to about 2.25 kg worth of sediment deposited in that cell. As the calculation presented above revealed, the amount of deposited sediment in all cells adds up to about 7.1 kilo tonnes, i.e., 85.5 % of the discharged mass. At $n = 20000$ iterations, the mass of the sediment suspended in the turbidity current amounts to 148 t, or about 2 % of the discharged mass. This leaves about 12.5 % of the mass unaccounted for. This may be explained by the transport of mass over the absorbing boundary. Note that the simulated domain does not cover POI 4 in Figure 7.1, which was found to contain elements from the discharged tailings in Ref. [42]. It is therefore probable that the 12.5 % of discharged mass is transported out of the simulation domain. In Ref. [43] it is stated that as much as 50 % of the discharged mass is transported past POI 4 in Figure 7.1.

The concentration plot indicates that the fine particles are easily dispersed and deposited to the distal parts of the grid, while the coarser particles (medium and coarse) are deposited more concentrated, in areas closer to the source. This fits with the measured fine fractions (FF) in Ref. [42], which found that the fine fractions were low close to the source (POI 1), high near the middle (POI 2), and low near the end of the simulation domain (POI 3). It is somewhat suspicious that no sediment at all is deposited at the bottom of the fjord (POI 3). The reason could be that when the current reaches this position, it has already deposited much of its sediment due to the low speed Q_v. This means that the concentration of the current Q_{cj} is probably very small. Then if Δt is too big, the rule T_2 cannot be applied without Q_{cj} turning negative (see Eq. (4.17)), and thus being reverted to its previous state (see Section 4.4.2). This can be solved by either, ensuring that Δt is sufficiently small, or by modifying the "safety rules" in T_2 such that the current terminates if e.g., $Q_{cj} < 0$. This is elaborated further in Chapter 9.

The time step $\Delta t^{(n)}$ features some oscillations in the first ~ 2500 iterations, then for as long as the source is active, no fluctuations occur. For $n > 10000$, it grows by a factor of approximately 4-5, and starts oscillating between iterations. This change in magnitude makes sense because, the characteristic speed $u = \sqrt{2rg'}$ of the system (see Section 4.3) becomes very small when Q_{cj} becomes small, due to the reduced gravity g', thus yielding a large Δt. When the source becomes inactive, there is a large drop in the maximum concentration Q_{cj}. The fluctuating Δt value must also be due to variations in speed u. The reduced gravity g' depends on Q_{cj}, which in general is assumed to have a much slower variation than what is observed here. The source of the fluctuations is assumed to originate from the run-up height r, which is dependent on Q_v. Although, looking relatively smooth in Figure 7.4 (except at $n = 10000$), the speed substate has proven to be somewhat unpredictable, due to the Chézy equation. By implementing, and selecting an appropriate an upper limit τ on Δt, as was done in Section 6.1, the large jump in time step size and possibly the fluctuations may be avoided.

It was observed during the simulations of this bathymetry, that certain combinations of initial conditions led to numerical instabilities. For instance, an attempt to simulate a continuous emission (source always active) over an extended period of time ($n = 100k$ CA iterations), led to unphysical substate values. In that case, the current thickness Q_{th} and speed Q_v grew to 200 m and 100 m/s, respectively, in certain areas of the grid. This area was visible in the ΔQ_d plots as an area with very high erosion. As the CA was iterated, this area of erosion grew in size until it covered an unreasonably large part of the grid. It could be that the reason for this, is, as was observed in Section 6.1, when no restriction on Δt was used, an accumulation of Q_{th} due to large source value. If this is the case, it could be that the solution mentioned in the previous paragraph also could be the solution to this problem, i.e., limit Δt.

Another possible explanation for the area of erosion is that the amount of large particles in the current is now very big (88.4 %). In Section 6.2.3 an explanation for the low amounts of erosion was argued to be the high concentration of fine particles. Particles with a large diameter D_{sj} have a higher erosion rate than small particles (see Figure 5.3). In the area showing erosion in the ΔQ_d plot, the majority of deposited particles are coarse particles (see Figure 7.5).

Altogether, the results fit to some extent with the description by the reference material [41, 42, 43, 44]. Sediment is deposited, along what appears to be the curve of steepest descent into the bottom of the fjord. The smaller particle types are more easily distributed to the distal parts of the grid. The current simulation considered a temporary discharge of suspended sediment lasting 10000 iterations. Erosion and deposition are processes that have been acting on the fjord bathymetry for decades. To see large changes in the bathymetry ΔQ_d, a continuous tailings discharge would probably have to be simulated for at least several years. In its present state, the implemented CA displays signs of numerical instability, that would have to be solved before such a simulation can be run. Further discussion and commentary to the implemented CA is featured in Chapter 9.

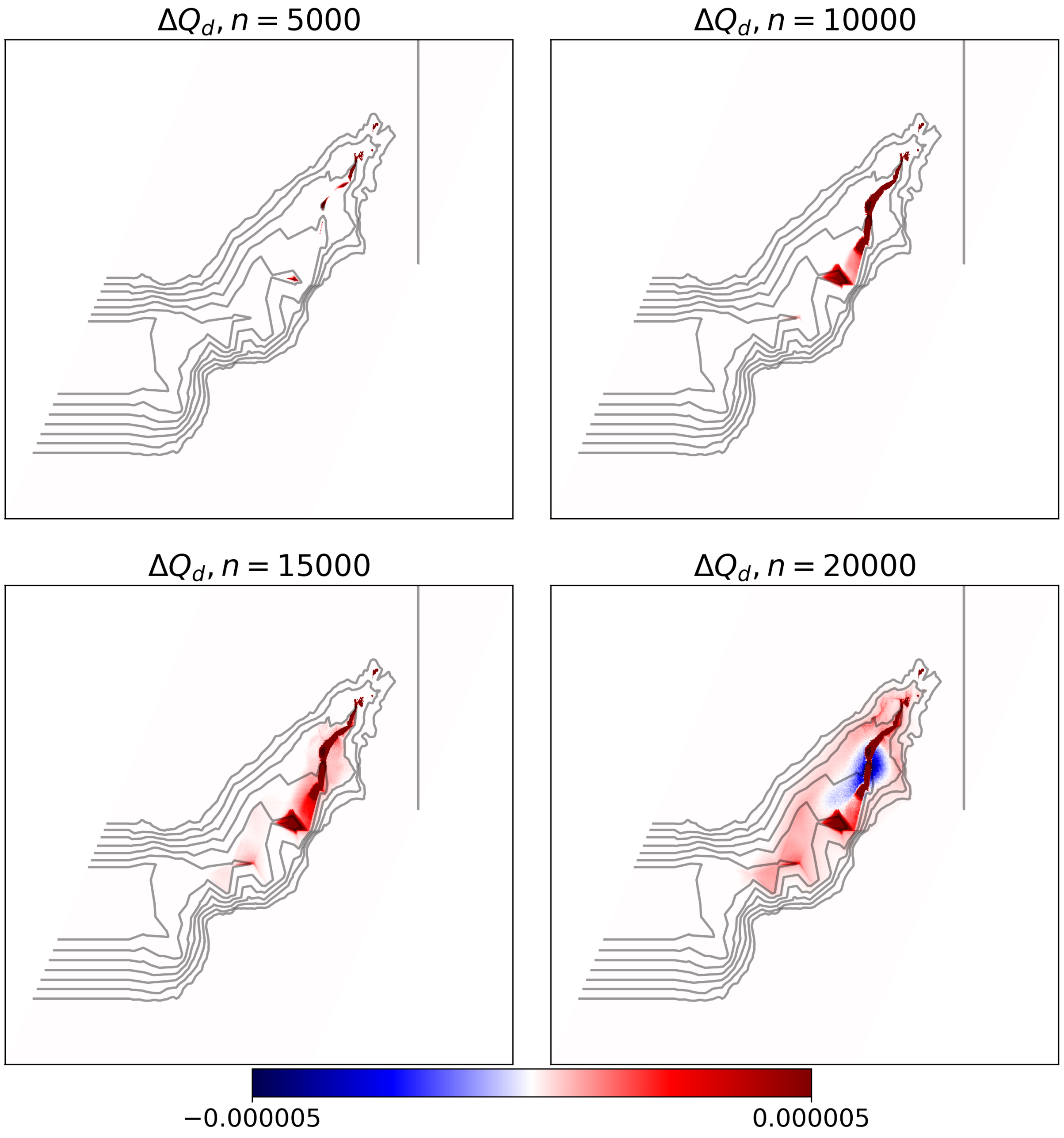


Figure 7.2: This figure illustrates the sea bed evolution ΔQ_d, after $n = 2000$, $n = 5000$, $n = 10000$ and $n = 20000$ iterations, using the CA configuration specified in Section 7.1. The colorbar indicate areas of net erosion (blue) or net deposition (red) of sediment in meter.

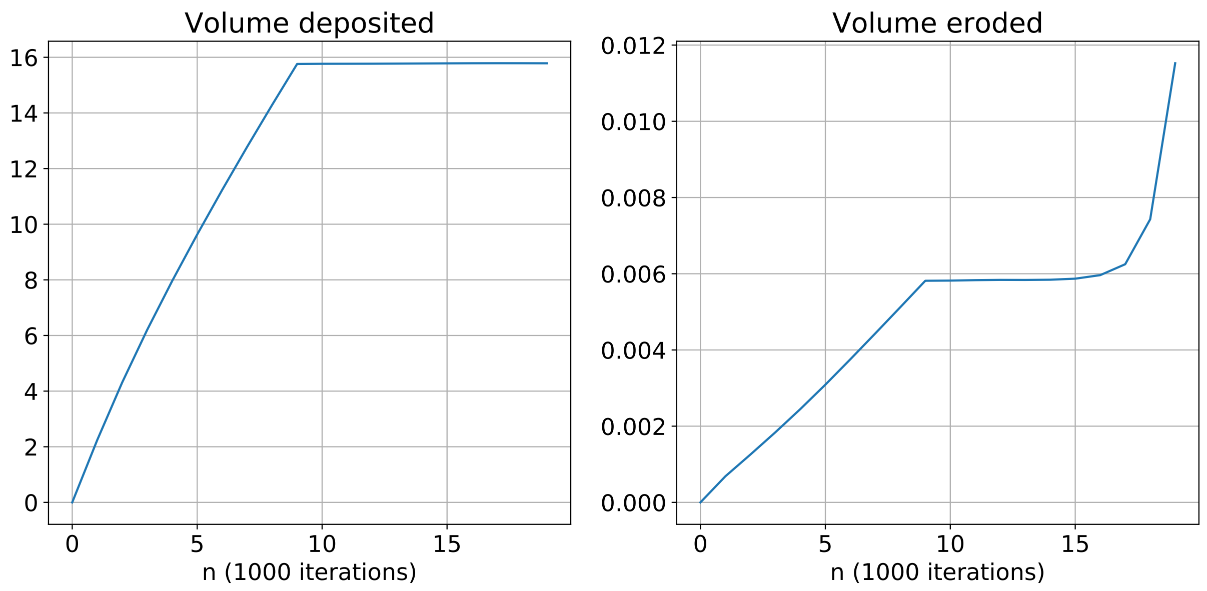


Figure 7.3: The figure illustrates the time evolution of the sea bed. The left plot displays the amount of volume (in meter) gained by the bed, and the right plot displays the amount of volume (in meter) lost by the sea bed.

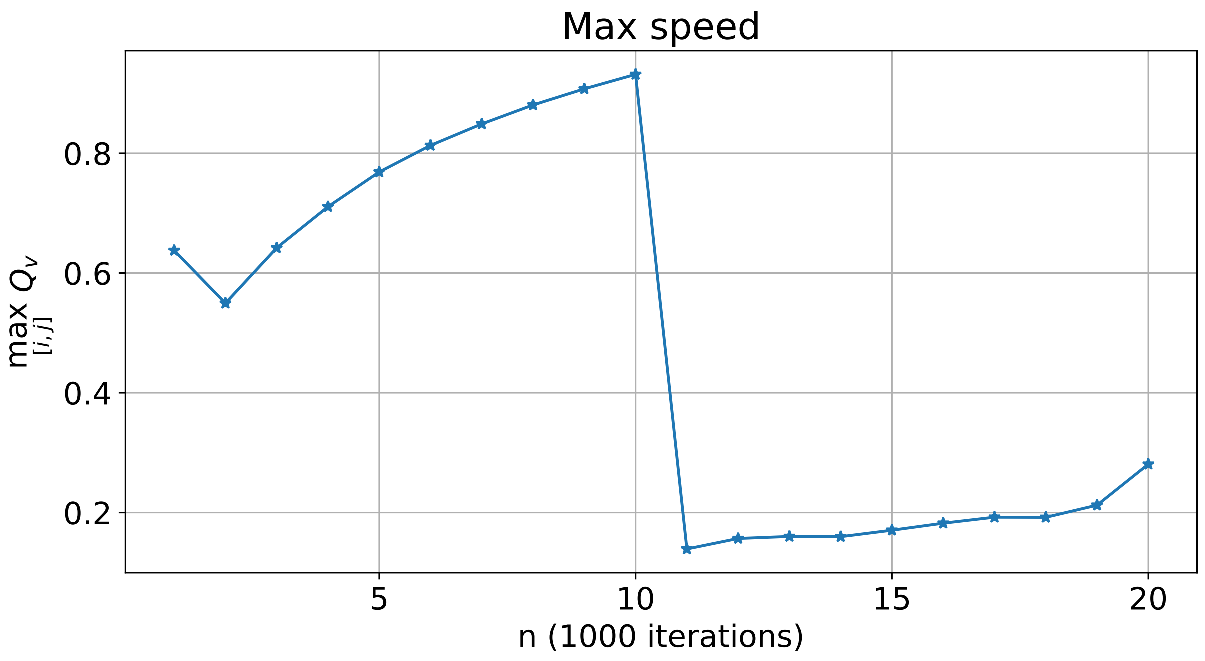


Figure 7.4: This figure displays the maximum speed Q_v attained by the turbidity current at a given iteration n.

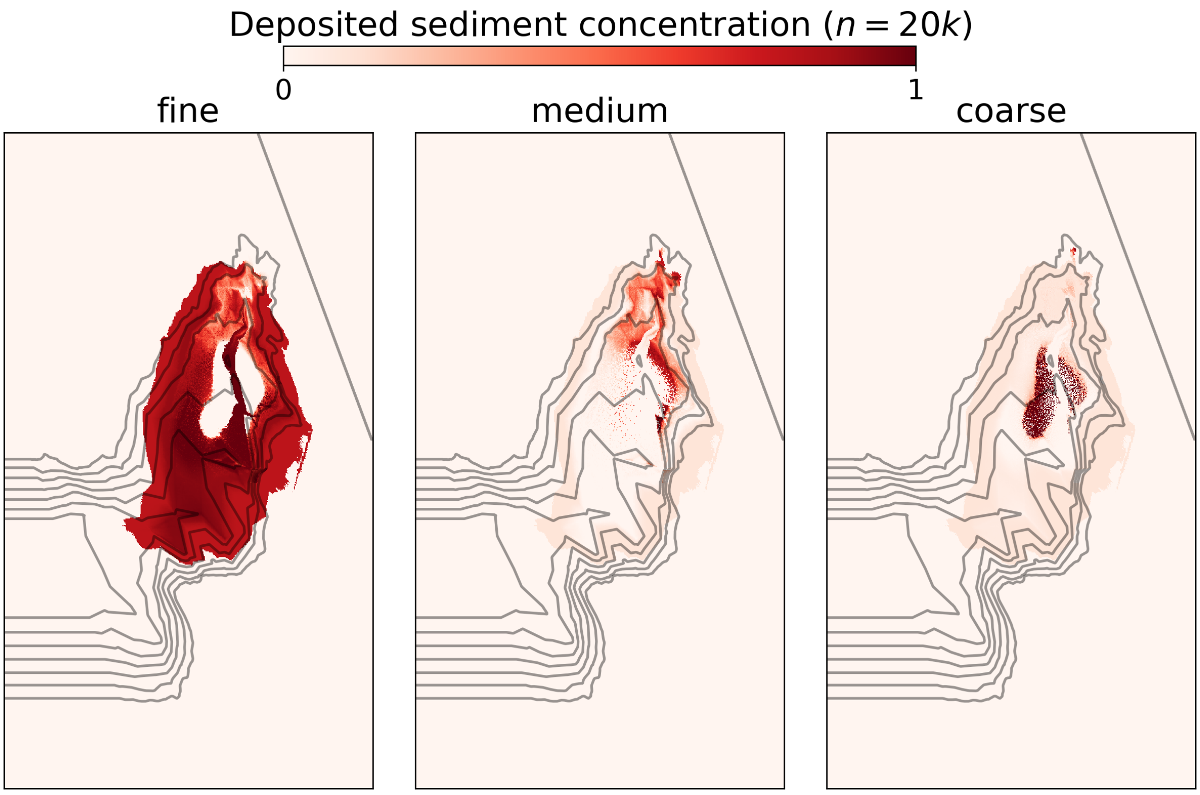


Figure 7.5: This figure illustrates the relative concentration of deposited sediment at $n = 20000$ CA iterations, using the initial conditions described in Section 7.1.

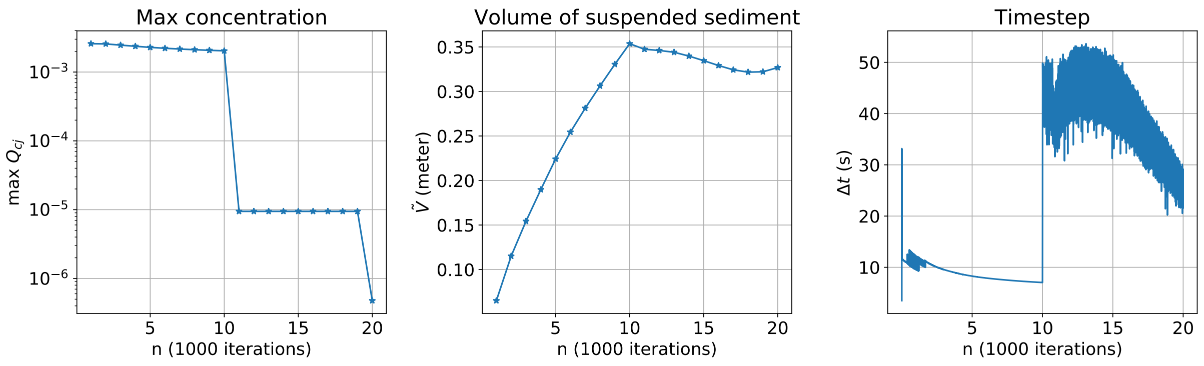


Figure 7.6: The figure shows properties of the turbidity current (maximum volumetric density and mass) and the simulation (size of time step Δt) as functions of the CA iteration n.

8 | Numerical performance

This chapter discusses the numerical performance of the implemented cellular automaton. For benchmarking the performance of the parallelization, the measures presented in Section 3.2 are used. Note that all benchmarking was performed using the implementation with commit `484c0f980c758881ea87d83a9a0ad3e00d41e3fb` at [34].

8.1 MPI benchmarking

Tests were run at the Linux cluster "Fram", at the University of Tromsø. Fram consists of 1006 nodes. Most nodes have 32 CPU cores and 64 GB memory, and are connected by an InfiniBand network [45].

The bechmarking performed for this program consists of running the same simulation, with identical initial conditions, for several problem sizes and a varying number of CPUs p. The CA parameters, bathymetry, and initial conditions are identical to the ones presented in Section 6.2, except that the size of the grid G, and the spatial discretization is scaled, and the indices of the source cells are adjusted. Results are gathered for three different problem sizes.

Recall that the problem size of a program, can be measured by the number of performed floating-point operations (FLOPs). Here, the problem size is varied by altering the grid G size of the cellular automaton, while keeping the number of CA iterations n fixed. Let G_1, G_2 and G_3 refer to grid sizes 1000×1000, 2000×2000 and 4000×4000, respectively. The spatial discretization Δx for G_1, G_2 and G_3, is respectively, $\frac{60\,\mathrm{m}}{1000}$, $\frac{60\,\mathrm{m}}{2000}$ and $\frac{60\,\mathrm{m}}{4000}$. The config file used in the benchmark is provided in Appendix A.6. Note that for these benchmarks, the simulation produces no graphical output, and only `npy` files of the substates are saved at the sample iterations. The intent of this is to minimize the amount of time spent by the program running code from external libraries.

Figure 8.1 shows the time T used by p CPUs, $T(p)$, to perform $n = 100$ iterations on the grids G_1, G_2 and G_3. The singlecore runtimes $T(1)$ are 234.6 s, 1028.3 s and 4319.3 s for G_1, G_2 and G_3, respectively. The number of processors p for which T was sampled follows the sequence 2^k with $k = [0, 9]$. In Table 8.1 the speedup s (Eq. (3.4)) and the experimental serial fraction e (Eq. (3.5)) has been calculated for all sampled p values.

The average values of the positive experimental serial fractions e are 0.82 %, 0.35 %, and 0.21 %, for G_1, G_2, and G_3, respectively. Using these values as to approximate the serial fraction f, the theoretical speedup boundaries can be calculated by Amdahls law and Gustafsons law (see Section 3.2). The values are given in Table 8.2.

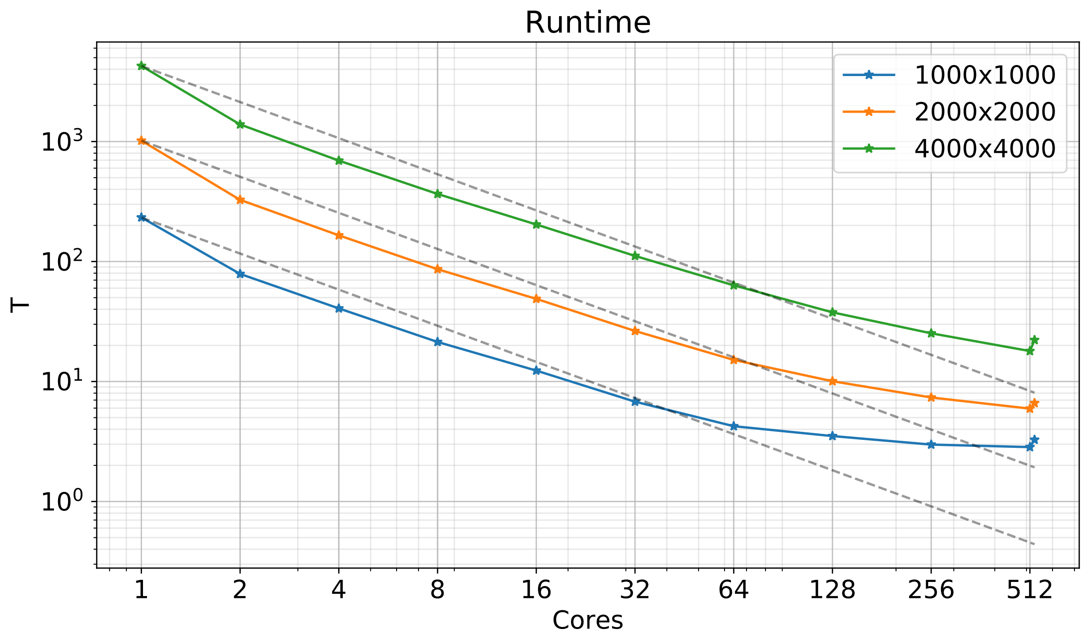


Figure 8.1: This figure shows the runtime $T(p)$ as a function of the number of used CPUs p, for $n = 100$ CA iterations. The grid size of the CA is specified in the legend. The config used for this benchmark resembles an "up-scaled" version of the Ruperts Inlet test case, described in Section 6.2. The dashed lines indicate the runtime with ideal speedup, i.e., $T(p) = T(1)/p$.

8.2 Performance evaluation

Visual inspection of Figure 8.1 indicates that all problem sizes get very good speedup when $p \lesssim 32$. This is supported by the calculated s values in Table 8.1. As anticipated by the theory in Section 3.2, the larger problem sizes are able to achieve higher speedups than programs with smaller problems.

As a consequence of the superlinear speedup $s > p$ for certain p values, the experimental serial fraction e of the program is negative, as can be seen in Table 8.1. As the p values increase, and sublinear speedups are achieved, the e becomes positive and increasing. This indicates that the parallelization overhead is causing a speedup bottleneck. Recall

Table 8.1: Performance data for running a various grid sizes, for $n = 100$ iterations using bathymetry B resembling the lower reach of Ruperts Inlet (see Figure 6.3). G_1, G_2 and G_3 refer to grids with sizes 1000×1000, 2000×2000 and 4000×4000, respectively.

Problem size\p		2	4	8	16	32	64	128	256	512	529
Nodes		1	1	1	1	2	4	8	16	32	32
G_3	s	3.1	6.2	11.7	21.0	38.3	67.4	113.2	169.3	238.0	192.5
	$e(\%)$	-35.0	-11.7	-4.5	-1.6	-0.5	-0.1	0.1	0.2	0.2	0.3
G_2	s	3.1	6.2	11.8	20.8	38.6	67.3	101.3	138.3	171.3	154.0
	$e(\%)$	-35.79	-11.68	-4.60	-1.55	-0.55	-0.08	0.21	0.33	0.39	0.46
G_1	s	3.0	5.7	10.9	18.9	34.3	55.0	66.4	78.2	82.1	71.3
	$e(\%)$	-32.36	-10.08	-3.81	-1.03	-0.22	0.26	0.73	0.89	1.02	1.22

Table 8.2: Theoretical possible speedups for the implemented CA, as calculated by Amdahls law and Gustafsons law. Here Gustafsons law is used to predict possible speedup with $p = 128$, and f is the average of the positive e values in Table 8.1.

Grid	Approximate f (%)	Amdahl	Gustafson (p=128)
G_1	0.82	121.3	127.0
G_2	0.35	287.3	127.6
G_3	0.21	465.1	127.7

that process startup, communication, and synchronization are examples of parallel overhead. This makes sense as when the number of CPUs grows, the size of a subgrid is decreased, and the ratio N_h/N_t grows, as can be seen in Figure 3.4. As mentioned in Section 3.2, a high N_h/N_t ratio, implies a relatively large amount of MPI communication, compared to the amount of computation performed on the subgrid. This leads to an increase in runtime T.

Another potential bottleneck is the network communication between nodes. As p increases, no single compute node is able to handle the computation, and thus CPUs on different nodes must to communicate through the network. Network communication is considerably slower than communication within the same node. Thus, when p increases, not only does the ratio N_h/N_t increase (for a fixed grid size), but also the amount of network communication increases. As mentioned, a single node on Fram can handle up to $p = 32$.

In this case the $p = 32$ sample was benchmarked using 2 nodes. The reason is for using 2 nodes instead of 1, is due to memory. Each node has 64 GB of memory to share between its processes. By running $p = 32$ on 2 nodes, and assuming 16 cores per node, yields 4 GB memory per core. If $p = 32$ is run on 1 node, each core has 2 GB memory per core. The number of nodes specified in Table 8.1, was determined by assuming 1 node per 16 CPUs.

The theoretical speedups calculated by Gustafsons law, reflects that the serial fraction of the program is low. It predicts that for sufficiently large problems, achievable speedup at $p = 128$ is approximately 128. That is, a very small amount of time is spent on inherently serial computation (see Section 3.2.1). Observe that the speedup at $p = 128$ approaches 128 as the problem size increases, just as predicted by Gustafson (see Table 8.1).

Now consider the speedups of $p = 512$ and $p = 529$ for G_3 (see Table 8.1 and Figure 8.1). For $p = 512$, the MPI grid $p_x \times p_y$ becomes 16×32. For a CA grid size of 4000×4000, each subgrid becomes size 250×125. For $p = 529$, the MPI grid becomes $(p_x, p_y) = (23, 23)$. But, as 4000 cells cannot be divided equally among 23 processors, some processors must compute a larger problem, e.g., the subgrid of rank 0 has dimensions 194×194, while other ranks in MPI column 0 (and row 0) has dimensions 173×194 (194×173). Notice that rank 0 has to compute about 7000 more cells than "interior ranks" with subgrid dimension 173×173, and about 6000 more cells than each rank in the $p = 512$ case. This is an example of a load imbalance, and it explains the loss of speedup from $p = 512$ to $p = 529$.

Overall, the implemented cellular automaton displays good speedup, and a low serial fraction. Two factors are identified to inhibit the speedup: communication, and load imbalance. As long the ratio N_h/N_t is low, the communication is low, and the achievable speedup should be high. And, as the previous paragraph exemplifies, load imbalance can be an issue when the dimensions of the MPI grid does not divide the dimensions of the computational domain. When these issues are accounted for, the cellular automaton should be able to achieve very good speedup, for high p values.

9 | Miscellaneous discussion

9.1 Known issues, suggestions, and potential improvements to the cellular automaton

In this section comments are made on the transition function σ and its components. The behaviour of the entrainment transformation T_1 and the turbidity current update local interaction I_2 were discussed in some detail in Chapter 5. These components of σ are relatively trivial, which in a sense is good as there is not a lot that can go wrong. Because the substates of the CA influence each other directly and indirectly (e.g., the speed Q_v influences the outflow Q_o, which influences the current thickness Q_{th}), detecting the origin of an issue may be difficult. In the following, some known issues related to selected components of σ are discussed.

9.1.1 Turbidity current outflow update I_1

Recall that the outflow Q_o between two cells should be scaled by a relaxation factor $p_r = \sqrt{2rg'}\frac{\Delta t}{\Delta x/2}$ (see Section 4.4.3). In D'Ambrosio et al. [37] the factor is said to satisfy $0 < p_r \leq 1$. In this implementation the factor was restricted by an experimentally determined lower boundary of 0.2 and an upper boundary of 1. Setting the the lower boundary to 0, led in most simulated cases to very small cell outflows Q_o, and thus to a turbidity current restricted to a small area about the source. It was thus decided that some lower boundary > 0 had to be implemented. However, when this boundary was set to high (e.g., $p_r \geq 0.4$), the turbidity current thickness Q_{th} began oscillating between cells, in a checkerboard pattern. It would seem that selecting $p_r \geq 0.2$ was sufficiently low to prevent the checkerboard oscillation, yet high enough to keep the current moving. However, as numerical instabilities were observed during some of the Ranfjorden simulations (even with $p_r \geq 0.2$), it could be that the factor p_r indeed has to be less than 0.2 in order for the simulation to be stable in certain situations.

The relaxation factor p_r depends on the thickness Q_{th}, speed Q_v and density Q_{cj} of the turbidity current. It becomes small when either of these substate values are small. Recall that the smallest $p_r \geq 0.2$ is used by all cells. In the results shown in previous sections, both Q_v and Q_{cj} usually tend to zero, which may explain why p_r became very small when no lower limit was set. It is possible that if the cause of the unreasonably low values of Q_v and Q_{cj} are resolved in some other way, there will be no need for a lower boundary on p_r (besides 0), which in turn may remove the instabilities.

9.1.2 Turbidity current flow speed update I_3

A commonly observed behaviour of the CA is that the speed Q_v asymptotically approaches zero, away from the source cells. This is observed for all of the simulated cases.

In the Ranfjorden case, the speed was in general observed to be very low (about 0.9 m/s). In the following, some potential causes of this behaviour and solutions are suggested.

The computation of Q_v involves averaging the speed contribution U_k for all six sides out of a cell (see Section 4.4.5). Consider for the sake of argument, the case where a cell has outflows $Q_o = \{1, 99\}$, towards two of its neighbor cells, and the corresponding speed contributions U_k are 1 m/s and 99 m/s. Using the implemented calculation, the computed cell speed Q_v becomes 50 m/s, even though 99 % of the outflow has speed 99 m/s. Computing the average, using the outflow as weights, the speed becomes about 98 m/s, which seems more reasonable. Implementing this may lead to an overall increase in the calculated speeds Q_v.

It is possible, that a similar effect to the weighted mean calculation suggested above, was achieved by using the slope definition s without absolute value (see Section 4.4.5), by favouring contributions U_k where the outflow was large. In the comparison of the results from I_3 and $\widetilde{I}_3$, the results using I_3 must have achieved a higher speed Q_v by excluding the directions in which the slope s was negative, in order for erosion to occur. It is assumed that only small amounts of outflow is sent to "higher" cells due to the run-up effect, and that the majority of the flow is in the downstream direction (with positive s), i.e., $Q_{o,run-up} \ll Q_{o,downstream}$. This means that the directions with positive slope s, and larger outflow, in general has larger speed contributions U_k. Thus, it is possible that by using the weighted mean calculation suggested above, the slope calculation may be redefined permanently to using the absolute value, i.e., Eq. (4.30), without losing speed Q_v.

A consequence of the specified order of application of the transition function components (see Section 4.4), is that the slopes s used in the Chézy formula, i.e., the height difference of $Q_a + Q_{th}$ between neighbor cells, differs from the cell heights used to calculate the outflow $Q_o(0, k)$ between those cells. The outflow is calculated in local interaction I_1, while the speed is calculated in local interaction I_3. In local interaction I_2, the thickness Q_{th} and concentration Q_{cj} of the turbidity current is updated due to the outflows. Thus, as the slope s is calculated in the speed interaction (I_3), the cell heights are not equal to what was used in the Q_o calculation. Changing the order of the interactions I_2 and I_3 in σ would possibly yield larger slopes s in the calculation of U_k, and thus larger speeds Q_v.

It is possible that the low speed of the turbidity current is simply a consequence of using the Chézy formula (see Section 2.3.6). The speed Q_v becomes zero if there is no slope, i.e., if the sum $Q_a + Q_{th}$ is equal for two neighbor cells, or if the outflow $Q_o(0, k)$ is zero (see Section 4.4.5). As mentioned in several of the previous discussions, implementing an alternative model by T. Mulder [38] could be a solution. The model computes the next iteration of the speed as $Q_v^{(n+1)} = Q_v^{(n)} + a\Delta t$, where a is the acceleration of the particles due to the various forces acting. By using this model, it is possible that the current speed will act as if it has some inertia, and thus retain some speed when the slope is zero, as compared to the present model.

An alternative explanation for the decreasing Q_v is that the balance between erosion and deposition is somehow offset when using three sediment types. The behaviour of

T_2 was only examined for the case $N_j = 1$ (see Section 5.2). It would be an advantage to thoroughly verify the behaviour of $\partial z/\partial t$ for several particle types $N_j > 1$. The simulation results in previous sections are all dominated by sediment deposition. If the deposition rate is much greater than the erosion rate, the current concentration Q_{cj} is decreased. When the source cell is turned off at $n = 10000$ in figures 6.10 and 7.6, the maximum concentration immediately decreases. This could imply that, away from the source, the concentration Q_{cj} is very low for all iterations, and that the max concentration shown in the figures is high for the sole reason that the source is active. If the concentration Q_{cj} is low away from the source, it affects the reduced gravity g', which in turn affects the speed Q_v. Thus, ensuring that there is balance between the erosion rate and deposition rate could be the key for preventing low speeds Q_v.

9.1.3 Erosion and deposition T_2

The transformation T_2 has a large influence on the behaviour of the simulation. It is the only connection between the substates describing the bed, and the ones describing the turbidity current. Thus, as was mentioned in the previous paragraph, ensuring that there exists a “balance” between the calculated erosion rate and deposition rate is important. The results from the previous chapters suggest that the deposition rate may dominate the bed conservation equation, leading to unreasonably low concentrations Q_{cj}, that may further affect the substates. It is therefore suggested that further study of the T_2 transformation is performed, to ensure that the erosion and deposition rates are of reasonable relative sizes.

In the transformation T_2, probably the most important modification was the implemented “correction”, used for ensuring that the substates assume physical values (see Section 4.4.2). However, the implemented solution is not ideal. The update rules are observed to be very unstable, often yielding unphysical values, e.g., concentrations $\{Q_{cbj}, Q_{cj}\} < 0$ or $\{Q_{cbj}, Q_{cj}\} > 1$, when no correction like this is implemented. But, ignoring the unphysical values, the update rules does indicate that either erosion or deposition should have occurred in the cell, while with the correction, nothing happens. As mentioned in the discussion of the Ranfjorden results, perhaps a better way of dealing with these unphysical values would be to implement ways for the turbidity current to terminate. For instance, if $Q_{cj} < 0$ is detected, there probably is a high deposition rate in the cell, and thus the current should terminate. Conversely, if $Q_{cbj} < 0$ is detected, then there is probably a high erosion rate in that cell, so the concentration Q_{cbj} could be set to zero, and the sediment of type j “moved” from the sea bed into the turbidity current.

The update rules of T_2 originate from integrating the Exner equation in time, through the forward Euler scheme. While the Euler scheme is easy to implement, it is merely *conditionally* stable, and may yield imprecise results for repeated iterations due to accumulated error. The method is known to be inaccurate even with very small time steps Δt, when used for a large number of iterations. Ideally, a higher-order method could be applied to the PDEs. An example of this can be found in Ref. [11], where a second-order method is applied to the bed-continuity equation.

9.1.4 Toppling rule I_4

In the current implementation, the toppling rule I_4 is applied to the soft sediment layer Q_d, and the initial sand cover layer $Q_{d,0}$ is specified as a constant over the grid. For certain bathymetries, with steep inclines, this may lead to a large redistribution of sediment during the first CA iterations. Recall that, for this reason the toppling rule I_4 could be specified to run until convergence, prior to the rest of the transition function σ is applied (see Section 4.5). Another option would be to assume that the initial sand cover $Q_{d,0}$ is in its steady state, and that the toppling rule only applies to sediment that is deposited by the turbidity current. In order to do this, the initial layer of sediment and the deposited sediment would probably have to be divided into separate substates, e.g., Q_{d1} and Q_{d2}, respectively. Then the toppling rule could then be applied to only Q_{d2}. The steepness of the initial bathymetry in examples described in Ref. [4] indicates that an approach like this may have been used, but if so, it is not described.

9.1.5 General implementation

The application order of the components in σ, listed in Section 4.4, implies the following. Consider an empty cell, next to a cell with a non-zero turbidity current thickness. The cell receives a turbidity current thickness Q_{th} when the local interaction I_2 is applied. Then, the speed is calculated by applying I_3. However, since this cell has no outflow yet (the outflow is calculated in I_1, which has already been run for this CA iteration), the speed Q_v becomes zero. Now, the next iteration starts. In transformation T_2, the cell substate Q_v is still zero, which means that the shear velocity $u^* = 0$ (and the erosion rate $E = 0$), and thus the current may only deposit sediment. If the deposition rate is very high, or Δt is high, this may cause the turbidity current to deposit all the sediment it contains. This may, or may not be an issue depending on the implemented "correction" against unphysical values in the transformation T_2 (see Section 4.4.2). Either way, by adding an additional outflow calculation I_1 prior to the speed calculation I_3, cells that received a turbidity current thickness in this iteration may also get a non-zero speed, and thus an erosion rate $E > 0$.

In order to save computation time, a variable grid size could be implemented. As of now, every grid cell in the CA implementation has the same size. For certain bathymetries, the interesting events could be isolated to a smaller section of the grid. For instance, the object of interest in the Rupert Inlet case, is the channel (see Figure 6.3). In order to better resolve the channel, the grid size would have to be increased and the spatial discretization would have to be decreased. This leads to a higher density of cells, both in the periphery, and near the channel. Thus, a large amount of computation time is spent on areas that would be adequately resolved using a lower density grid. If several grid densities could be combined, such that a higher density is used near areas of interest, while a lower density is used elsewhere, this could lead to saved computation time. This is a technique commonly used in computational fluid dynamics.

9.2 Parallelization of the cellular automaton

Benchmarking the parallel performance of the cellular automaton indicates that this is a problem well suited for parallelization. Besides ensuring that the relative amount of MPI communication is low, load imbalance was identified as a potential issue. The issue arises when the MPI grid dimensions does not equally divide the dimensions of the CA grid. The speedups at $p = 512$ and $p = 529$ CPUs demonstrate this behaviour (see Chapter 8). This can be solved by dividing the remainder, $L_x \bmod p_x$, among as many processors as possible $\widetilde{p}_x < p_x$, in that direction (see Section 4.5.2). Consider the scenario that occurred for $p = 529$ as an example. The computational grid has size 4000×4000, and the MPI grid is 23×23. The remainder becomes $4000 \bmod 23 = 21$ cells. Thus, by expanding the subgrid of 21 of the CPUs by one, the load imbalance can be minimized.

The simulation cases presented in this thesis are on a relatively small scale. Relatively small grid sizes, and small simulation durations were used. Thus, the need for high computational parallelism was not really justified for these cases. However, for simulations using higher resolution bathymetries with large grid sizes, running over extended periods of time, parallelism can be very beneficial. As was seen in Chapter 8, the CA has been benchmarked with up to 529 CPUs, using the Linux cluster Fram. This implementation was parallelized with the intent of running on CPU clusters. By using a graphics processor unit (GPU), it is possible that the same performance may be achieved without the need for a cluster.

10 | Conclusions and future work

In this thesis a cellular automaton (CA) used to simulate turbidity currents has been implemented. The CA simulates the evolution of the sea bed due to erosion and deposition, and was based on the work done by T. Salles et al. in Ref. [4]. The published description of the model has been found not to be entirely complete, and to contain some errors. Hence, a large portion of the work in this thesis has been to interpret the published model, and try to arrive at an implementation and a description that is as complete as possible. Tests reveal some remaining issues, and these have been discussed, along with potential strategies to resolve them.

The implementation of the CA was done using the programming language Python. The CA was parallelized by using the Message Passing Interface (MPI) standard, and by combining the halo exchange technique with a self-composed algorithm that has been named the inverse halo exchange. Cython was used on computationally intensive parts of the code.

All of the implemented CA transition function components have been examined individually. The water entrainment component was applied to a cell, and the resulting turbidity current thickness and concentration was plotted for a range of turbidity current speeds Q_v. The water entrainment effect was found to increase the thickness, while decreasing the concentration of the turbidity current, as expected. Considering scenarios with one particle type, the erosion and deposition rules were examined separately, before plotting the net rate of change of the sea bed (the Exner equation) for different particle sizes D_{sj}. The implemented model was compared with alternative models by Imran [35] and Fukushima [15], and was found to correspond well with the Imran model.

The outflow implementation was compared with an existing example of the outflow algorithm, created by D'Ambriosio [37], and was found to give the correct results. A dynamically calculated relaxation factor p_r, that is used to scale the outflow calculation, was found to restrict the propagation of the turbidity current, and a lower limit to the factor was implemented due to this. Two simple examples were used to demonstrate the calculation of received turbidity current thickness in the local interaction I_2, and calculation of the cell speed I_3. An important remark was made, that in this implementation, the speed Q_v is merely used as a measure of energy, and it does not influence the speed at which the current propagates through the grid.

The slope failure model was verified by application on a level bedrock, and by using the Ranfjorden bathymetry as bedrock. For the level bedrock, the cross section of the grid was plotted, and the inclination angle of the curve θ was compared to the given angle of repose θ_r. With $\theta_r = 0°$, and $\theta_r = 10°$ the hill inclination was found to be a close approximation of θ_r. A distortion of the displaced sediment was observed for the case with $\theta_r = 0°$, and is believed to be caused by the skewness of the domain combined with no-flux boundary conditions. The toppling rule was found to need a lot more iterations

n before θ approached θ_r, when θ_r was small. The implemented inverse halo exchange was demonstrated, and confirmed to give the correct results.

The complete CA implementation was tested against steady state values as predicted by the three equation model (TEM). In this case the CA model did not perform well. Despite the TEM predicting that one of the simulated scenarios would result in a self-accelerating turbidity current, and that another scenario would not, the CA did not produce results where a self-accelerating turbidity current was observed in either scenario. A large part of the reason for this is believed to be that the speed of the turbidity current Q_v is not directly linked to the current propagation speed, and that in this implementation the amount of sediment in the system is finite, while in TEM it is not.

A comparison between results gathered from this implementation, and the results in Ref. [4] was conducted. The results were found to be somewhat in agreement, especially concerning the distribution of deposited particle sizes. The observed amounts of deposition and erosion did not agree with Ref. [4]. Some assumptions were made concerning the initial conditions of this simulation, which could be the reason for the discrepancy. Other potential reasons include too low turbidity current speed Q_v. Possible explanations and solutions for the low speed Q_v have been discussed.

For the final simulation, bathymetry from the inner part of Ranfjorden was used. The results displayed areas of deposition that seemed to agree with the described state of the bed in Refs. [41, 42, 43, 44]. Small amounts of erosion were observed in areas that corresponded to areas where erosion had been observed on site [41]. In certain simulations, numerical instabilities were observed. These are believed to be due to the relaxation factor p_r. A possible solution is discussed. An issue regarding the speed of the turbidity current has been discussed, along with possible solutions.

The parallel performance of the implemented CA has been tested by using the Linux cluster Fram. Speedups, and the estimated serial fraction were measured for several combinations of problem sizes and number of CPUs. The program achieved super-linear speedups for all the problem sizes, up to a certain number of CPUs. Larger grids/problems achieved best speedup. The highest speedup achieved was 238, when using 512 CPUs. Higher speedup is believed to be achievable by ensuring that the relative amount of computation is low, and that the work is equally distributed between the MPI ranks. For instance, when the dimensions of the MPI grid does not divide the dimensions of the computational grid, load imbalance between the MPI ranks may occur, reducing the speedup.

The implemented CA shows promise in predicting the evolution of sea beds, under the influence of turbidity currents. More work is, however, required in order to achieve a tool that can be used with reliable precision.

Future work

The immediate priority for any future work would be to solve the remaining numerical instability issue. Potential causes and solutions have been discussed.

When any remaining numerical issues are solved, the cellular automaton model may be used to study the bathymetry evolution of several real tailings deposit locations. The implementation is “HPC-ready” as has been demonstrated, and so high resolution bathymetries may potentially be used to predict deposition and erosion activity due to sediment discharge, in some detail. Candidates for such simulations are e.g., Ranfjorden and Bøkfjorden.

Bibliography

[1] Geir Thorsnæs. “Ranfjorden”. In: *Store Norske Leksikon* (2016). Accessed on 30.05.19. URL: `https://snl.no/Ranfjorden`.

[2] Nussir. *Quantities and Estimates.* Accessed on 25.06.19. 2014. URL: `http://www.nussir.no/en_projec_nussir.php`.

[3] Eva Ramirez-Llodra et al. “Submarine and deep-sea mine tailing placements: a review of current practices, environmental issues, natural analogs and knowledge gaps in Norway and internationally”. In: *Marine Pollution Bulletin* 97.1-2 (2015), pp. 13–35.

[4] T. Salles et al. “Cellular automata model of density currents”. In: *Geomorphology* 88.1-2 (2007), pp. 1–20.

[5] Marius Ungarish. *An Introduction to Gravity Currents and Intrusions.* 1st ed. CRC Press, 2009.

[6] Deborah Anne Edwards. *Turbidity currents : dynamics, deposits and reversals.* 1991.

[7] Thierry Mulder and Jan Alexander. “The physical character of subaqueous sedimentary density flows and their deposits”. In: *Sedimentology* 48.2 (2001), pp. 269–299.

[8] Vanessa Teles et al. “CATS – A process-based model for turbulent turbidite systems at the reservoir scale”. In: *Comptes Rendus - Geoscience* 348.7 (2016), pp. 489–498.

[9] G Parker et al. “Experiments on turbidity currents over an erodible bed”. In: *Journal of Hydraulic Research* 25.1 (Jan. 1987), pp. 123–147.

[10] M. Hanif Chaudhry. *Open-channel flow.* 2nd ed. Springer, 2008.

[11] Scott F. Bradford and Nikolaos D. Katopodes. “Hydrodynamics of Turbid Underflows. I: Formulation and Numerical Analysis”. In: *Journal of Hydraulic Engineering* 125.10 (1999), pp. 1006–1015.

[12] Marcelo Garcia and Gary Parker. “Experiments on the entrainment of sediment into suspension by a dense bottom current”. In: *Journal of Geophysical Research: Oceans* 98.C3 (1993), pp. 4793–4807.

[13] Marcelo H Garcia. “Depositional Turbidity Currents Laden with Poorly Sorted Sediment”. eng. In: *Journal of Hydraulic Engineering* 120.11 (1994), pp. 1240–1263.

[14] Maurice E. Tucker. *Sedimentary rocks in the field.* Wiley, 2003.

[15] Yusuke Fukushima, Gary Parker, and H. M. Pantin. “Prediction of ignitive turbidity currents in Scripps Submarine Canyon”. In: *Marine Geology* 67.1-2 (1985), pp. 55–81.

[16] Peng Hu, Thomas Pähtz, and Zhiguo He. “Is it appropriate to model turbidity currents with the three-equation model?” In: *Journal of Geophysical Research F: Earth Surface* 120.7 (2015), pp. 1153–1170.

[17] Tommaso Toffoli. *Cellular automata machines : a new environment for modeling.* eng. 1987.

[18] Martin Gardner. “MATHEMATICAL GAMES The fantastic combinations of John Conway’s new solitaire game “life””. In: *Scientific American* 223 (1970).

[19] S. Warner. *Modern Algebra*. Dover Books on Mathematics. Dover Publications, 1990.

[20] Raymond Nepstad. “DREAM for mining releases: Changing bathymetry by deposition”. Memo. 2012.

[21] Dale Anderson, John C Tannehill, and Richard H Pletcher. *Computational fluid mechanics and heat transfer*. CRC Press, 2016.

[22] Bernhard Müller. “Introduction to Computational Fluid Dynamics”. Lecture notes for the course Computational Heat and Fluid Flow. 2018.

[23] James W. Rottman et al. “Unsteady gravity current flows over obstacles: Some observations and analysis related to the phase II trials”. In: *Journal of Hazardous Materials* 11 (1985), pp. 325–340.

[24] Daniel R Lynch et al. *Particles in the coastal ocean: Theory and applications*. Cambridge University Press, 2014.

[25] Trisdan Salles. *Evolution sedimentaire des canyons et chenaux sous-marins: Modélisation numérique du remplissage sédimentaire des canyons et chenaux sous-marins par approche génétique*. Éditions universitaires européennes, 2006.

[26] R. Soulsby. *Dynamics of Marine Sands*. Thomas Telford Ltd, 1998.

[27] William E Dietrich. “Settling velocity of natural particles”. In: *Water resources research* 18.6 (1982), pp. 1615–1626.

[28] G V Middleton. “Small-Scale Models of Turbidity Currents and the Criterion for Auto-Suspension”. In: *Journal of sedimentary research.* 36.1 (1966).

[29] G V Middleton. “Sediment Deposition from Turbidity Currents”. In: *Annual Review of Earth and Planetary Sciences* 21.1 (1993), pp. 89–114.

[30] Stefan Behnel et al. “Cython: The best of both worlds”. In: *Computing in Science and Engineering* 13.2 (2011), pp. 31–39.

[31] Peter Pacheco. *An Introduction to Parallel Programming*. eng. Elsevier Science, 2011.

[32] Alan H Karp and Horace P Flatt. “Measuring parallel processor performance”. In: *Communications of the ACM* 33.5 (1990), pp. 539–543.

[33] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. “MPI for Python”. In: *Journal of Parallel and Distributed Computing* 65.9 (2005), pp. 1108–1115.

[34] Steinar Brattøy Gundersen. *ProjectCA on Github*. Accessed on 30.05.19. URL: `https://github.com/steinabg/ProjectCA/tree/using_cython`.

[35] Jasim Imran, Gary Parker, and Nikolaos Katopodes. “A numerical model of channel inception on submarine fans”. In: *Journal of Geophysical Research: Oceans* 103.C1 (Jan. 1998), pp. 1219–1238.

[36] S. Di Gregorio et al. “Mount ontake landslide simulation by the Cellular Automata model SCIDDICA-3”. In: *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* 24.2 (1999), pp. 131–137.

[37] D. D’Ambrosio, S. Di Gregorio, and G. Iovine. “Simulating debris flows through a hexagonal cellular automata model: SCIDDICA S 3-hex”. In: *Natural Hazards and Earth System Science* 3.6 (2003), pp. 545–559.

[38] T. Mulder, J. P. M. Syvitski, and K. I. Skene. “Modeling of erosion and deposition by turbidity currents generated at river mouths”. In: *Journal of Sedimentary Research* 68.1 (1998), pp. 124–137.

[39] Alex E. Hay. “Turbidity currents and submarine channel formation in Rupert Inlet, British Columbia: 2. The roles of continuous and surge-type flow”. In: *Journal of Geophysical Research* 92.C3 (1987), p. 2883.

[40] Kartverket. *Dybdedata og terrengmodeller av havbunn.* Accessed on 17.06.19. URL: `https://www.kartverket.no/data/dybdedata-og-terrengmodeller-av-havbunn/`.

[41] Anders E. Haugen. “Distribution, deposition and impact of tailing disposal on the seafloor in Ranfjorden, northern Norway”. MA thesis. The Arctic University of Norway, 2018.

[42] Sigurd Øxnevad et al. *Tiltaksorientert overvåking av Ranfjorden i 2018. Overvåking for Mo Industripark AS, Celsa Armeringsstål AS, Elkem Rana AS, Ferroglobe Mangan Norge AS, Rana Gruber AS, Miljøteknikk Terrateam AS og Rana kommune.* Norsk Insitutt for Vannforskning, 2018.

[43] Torbjørn M. Johnsen et al. *Miljøundersøkelser i Ranfjorden 1994-96.* Norsk Insitutt for Vannforskning, 2004.

[44] Sigurd Øxnevad et al. *Tiltaksrettet overvåking av Ranfjorden i henhold til vannforskriften. Overvåking for Mo Industripark, Celsa Armeringsstål, Fesil Rana Metall, Glencore Manganese Norway og Rana Gruber.* Norsk Insitutt for Vannforskning, 2016.

[45] UNINET. *UNINET Documentation Sigma2, Fram.* Accessed on 16.06.19. URL: `https://documentation.sigma2.no/quick/fram.html`.

A Configuration files

A.1 Template config file

In this section an example CA config file is shown. All variables are commented, indicating the purpose or corresponding variable in the description given in the thesis.

```
[simulation_parameters]
ny = 100 # Number of cells in y-direction
nx = 100 # Number of cells in x-direction
nj = 3  # Number of sediment types
dx = 1  # Spatial discretization
terrain = rupert # See comment below
converged_toppling = [1, 1e-7] # [boolean, tolerance]
sphere_settling_velocity = salles # See below
slope=np.deg2rad(20) # Slope of terrain (rupert,salles2,sloped_plane)
num_iterations = 10000 # Number of iterations to perform
sample_rate = 1000 # How often should results be saved
output = 2d # Specify which output you want
x = np.ix_(np.arange(50,51)) # source area x-coordinate (m-index)
y = np.ix_(np.arange(4,5)) # source area y-coordinate (l-index)
theta_r = 10 # Angle of repose
q_th[y,x] = 1.5 # Initial turbidity current thickness
q_v[y,x] = 10 # Initial turbidity current speed
q_cj[y,x,0] = 0.03*0.2 # Initial turbidity current concentration (sediment 0)
q_cj[y,x,1] = 0.03*0.4 # Initial turbidity current concentration (sediment 1)
q_cj[y,x,2] = 0.03*0.4 # Initial turbidity current concentration (sediment 2)
q_cbj[y,x,0] = 1 # Initial source area bed concentration (sediment 0)
q_cbj[y,x,1] = 0 # Initial source area bed concentration (sediment 1)
q_cbj[y,x,2] = 0 # Initial source area bed concentration (sediment 2)
q_d[y,x] = 1 # Initial source area soft sediment height
q_cbj[interior,0] = 0.2 # Initial grid bed concentration (sediment 0)
q_cbj[interior,1] = 0.4 # Initial grid bed concentration (sediment 1)
q_cbj[interior,2] = 0.4 # Initial grid bed concentration (sediment 2)
q_d[interior] = 1 # Initial grid sediment height
g = 9.81 # gravitational acceleration
f = 0.04 # darcy-weisbach constant
a = 0.43 # empirical constant (Chzy equation)
rho_a = 1000 # density of ambient fluid (kg/m^3)
c_d = 0.003 # bed drag coefficient
nu = 1.5182e-06 # kinematic viscosity of ambient fluid
porosity = 0.3 # bed porosity
p_f = np.deg2rad(0) # friction angle limit (corresponds to theta_f in
    documentation)
p_adh = 0.1 # unmovable amount of sediment
```

```
rho_j = np.array([2650, 2650, 2650], dtype=np.dtype("i"), order='C') # array
    of sediment type densities (kg/m^3)
d_sj = np.array([50e-6, 100e-6, 300e-6]) # array of sediment type diameters
    (m)
```

- Terrain may either be the name of a .netCDF file, or one of the keywords {ranfjorden, rupert, salles2, sloped_plane}.
- sphere_settling_velocity may either be set to 'salles' in which case the speed is calculated at runtime, or can be explicitly set by an array of size nj with settling speeds with the respective settling speeds of the particle types.

A.2 Test case 1: 1D channel config ($IC1$)

```
[simulation_parameters]
ny = 1500
nx = 3
nj = 1
dx = 1
terrain = sloped_plane
slope = 0.05
sphere_settling_velocity = np.array([0.0084], dtype=np.double)
num_iterations = 100000
sample_rate = 1000
output=npy,bathymetry
x = np.ix_(np.arange(1,2))
y = np.ix_(np.arange(50,51))
theta_r = np.deg2rad(30)
q_th[y,x] = 2.0
q_v[y,x] = 0.652
q_cj[y,x,0] = 0.00291
q_cbj[y,x,0] = 1
q_d[y,x] = 1
q_cbj[interior,0] = 1
q_d[interior] = 1
g = 9.81
f = 0.04
a = 0.43
rho_a = 1000
c_d = 0.004
nu = 1e-06
porosity = 0
p_f = np.deg2rad(0)
p_adh = 0
rho_j = np.array([2650], dtype=np.dtype("i"), order='C')
d_sj = np.array([0.1e-3])
```

A.3 Test case 1: 1D channel config ($IC2$)

```
[simulation_parameters]
ny = 1500
nx = 3
nj = 1
dx = 1
terrain = sloped_plane
slope = 0.05
sphere_settling_velocity = np.array([0.0084], dtype=np.double)
num_iterations = 100000
sample_rate = 1000
output= npy,bathymetry
x = np.ix_(np.arange(1,2))
y = np.ix_(np.arange(50,51))
theta_r = np.deg2rad(30)
q_th[y,x] = 1
q_v[y,x] = 0.699
q_cj[y,x,0] = 0.00672
q_cbj[y,x,0] = 1
q_d[y,x] =1
q_cbj[interior,0] = 1
q_d[interior] = 1
g = 9.81
f = 0.04
a = 0.43
rho_a = 1000
c_d = 0.004
nu = 1e-06
porosity = 0
p_f = np.deg2rad(0)
p_adh = 0
rho_j = np.array([2650], dtype=np.dtype("i"), order='C')
d_sj = np.array([0.1e-3])
```

A.4 Test case 2: Rupert Inlet lower reach config

```
[simulation_parameters]
ny = 160*2
nx = 60*2
nj = 3
dx = 1/2
terrain = salles2
converged_toppling = [1, 1e-7]
sphere_settling_velocity = salles
slope=np.deg2rad(0.7)
num_iterations = 20000
sample_rate = 1000
output = 2d,bathymetry,stability
x = np.ix_(np.arange(64,65))
y = np.ix_(np.arange(4,6))
theta_r = np.deg2rad(50)
q_th[y,x] = 2.5
q_v[y,x] = 0.2
q_cj[y,x,0] = 0.09*0.8
q_cj[y,x,1] = 0.09*0.15
q_cj[y,x,2] = 0.09*0.05
q_cbj[y,x,0] = 0.8
q_cbj[y,x,1] = 0.15
q_cbj[y,x,2] = 0.05
q_d[y,x] = 0.5
q_cbj[interior,0] = 0.8
q_cbj[interior,1] = 0.15
q_cbj[interior,2] = 0.05
q_d[interior] = 0.5
g = 9.81
f = 0.04
a = 0.43
rho_a = 1000
c_d = 0.003
nu = 1e-06
porosity = 0.3
p_f = np.deg2rad(1)
p_adh = .1
rho_j = np.array([2600, 2600, 2600], dtype=np.dtype("i"), order='C')
d_sj = np.array([5e-6, 60e-6, 135e-6])
```

A.5 Ranfjorden config file

```
[simulation_parameters]
ny = 700
nx = 530
nj = 3
dx = 14
terrain = rf
converged_toppling = [1, 1e-7]
sphere_settling_velocity = salles
num_iterations = 20000
sample_rate = 500
output=stability
x=357
y=575
theta_r = np.deg2rad(50)
q_th[y,x] = 0.021
q_v[y,x] = 2
q_cj[y,x,0] = 0.01*0.02
q_cj[y,x,1] = 0.01*0.096
q_cj[y,x,2] = 0.01*0.884
q_cbj[y,x,0] = 0.8
q_cbj[y,x,1] = 0.1
q_cbj[y,x,2] = 0.1
q_d[y,x] = 1
q_cbj[interior,0] = 0.8
q_cbj[interior,1] = 0.1
q_cbj[interior,2] = 0.1
q_d[interior] = 1
g = 9.81
f = 0.04
a = 0.43
rho_a = 1000
c_d = 0.003
nu = 1e-06
porosity = 0.3
p_f = np.deg2rad(0)
p_adh = 0
rho_j = np.array([2650, 2650, 2650], dtype=np.dtype("i"), order='C')
d_sj = np.array([5e-6, 80e-6, 135e-6])
```

A.6 Benchmarking config file

Parameters `ny,nx,dx,x` are adjusted for each grid size $G_1 = 1000 \times 1000$, $G_2 = 2000 \times 2000$, and $G_3 = 4000 \times 4000$. G_1: `ny=1000`, `nx=1000`, `dx=60/1000`, `x=np.ix_(np.arange(533,534))`, G_2: `ny=2000`, `nx=2000`, `dx=60/2000`, `x=np.ix_(np.arange(1066,1067))`, and G_3 is given in completeness below.

```
[simulation_parameters]
ny = 4000
nx = 4000
nj = 3
dx = 60/4000
terrain = salles2
sphere_settling_velocity = salles
slope=np.deg2rad(0.7)
num_iterations = 100
sample_rate = 99
output = npy
x = np.ix_(np.arange(2133,2134))
y = np.ix_(np.arange(4,6))
theta_r = np.deg2rad(50)
q_th[y,x] = 2.5
q_v[y,x] = 0.2
q_cj[y,x,0] = 0.09*0.8
q_cj[y,x,1] = 0.09*0.15
q_cj[y,x,2] = 0.09*0.05
q_cbj[y,x,0] = 0.8
q_cbj[y,x,1] = 0.15
q_cbj[y,x,2] = 0.05
q_d[y,x] = 0.5
q_cbj[interior,0] = 0.8
q_cbj[interior,1] = 0.15
q_cbj[interior,2] = 0.05
q_d[interior] = 0.5
g = 9.81
f = 0.04
a = 0.43
rho_a = 1000
c_d = 0.003
nu = 1e-06
porosity = 0.3
p_f = np.deg2rad(1)
p_adh = .1
rho_j = np.array([2600, 2600, 2600], dtype=np.dtype("i"), order='C')
d_sj = np.array([5e-6, 60e-6, 135e-6])
```

B | Additional models

B.1 Imran sediment entrainment model

The sediment entrainment coefficient E_j can be modelled [35] by

$$E_j = a\frac{Z^5}{1 + \frac{a}{0.3}Z^5}, \tag{B.1}$$

with $a = 1.3 \cdot 10^{-7}$, and

$$Z = \alpha_1 \frac{u_*}{v_{sj}} R_{pj}^{\alpha_2}. \tag{B.2}$$

Where u_* is the shear-velocity, v_{sj} is the settling velocity of a particle, and R_{pj} is the particle Reynolds number. The parameters (α_1, α_2) take respective values $(1, 0.6)$ for $R_{pj} > 2.36$ and $(0.586, 1.23)$ for $R_{pj} \leq 2.36$.

B.2 Imran deposition rate model

The deposition rate D can be modelled [35] by

$$D = v_{sj} r_0 C, \tag{B.3}$$

where v_{sj} is the sediment settling velocity, $r_0 = 1.8$ is a constant, and C is the layer-averaged concentration.

B.3 Fukushima sediment entrainment model

The sediment entrainment coefficient E_j is modelled by Fukushima [15] as,

$$E_j = \begin{cases} 0 & \text{for } Z < Z_c \\ 3 \cdot 10^{-12} Z^{10} (1 - \frac{Z_c}{Z}) & \text{for } Z_c < Z < Z_m \\ 0.3 & \text{for } Z > Z_m, \end{cases} \tag{B.4}$$

where $Z = R_{pj}^{0.5}\mu$ and $\mu = \frac{u_*}{v_{sj}}$. R_{pj} is the particle Reynolds number, u_* is the shear-velocity, and v_{sj} is the settling velocity of a particle. $Z_c \approx 5$ and $Z_m = 13.2$.

Steinar Brattøy Gundersen

Master's thesis in Applied Physics and Mathematics

NTNU
Norwegian University of
Science and Technology