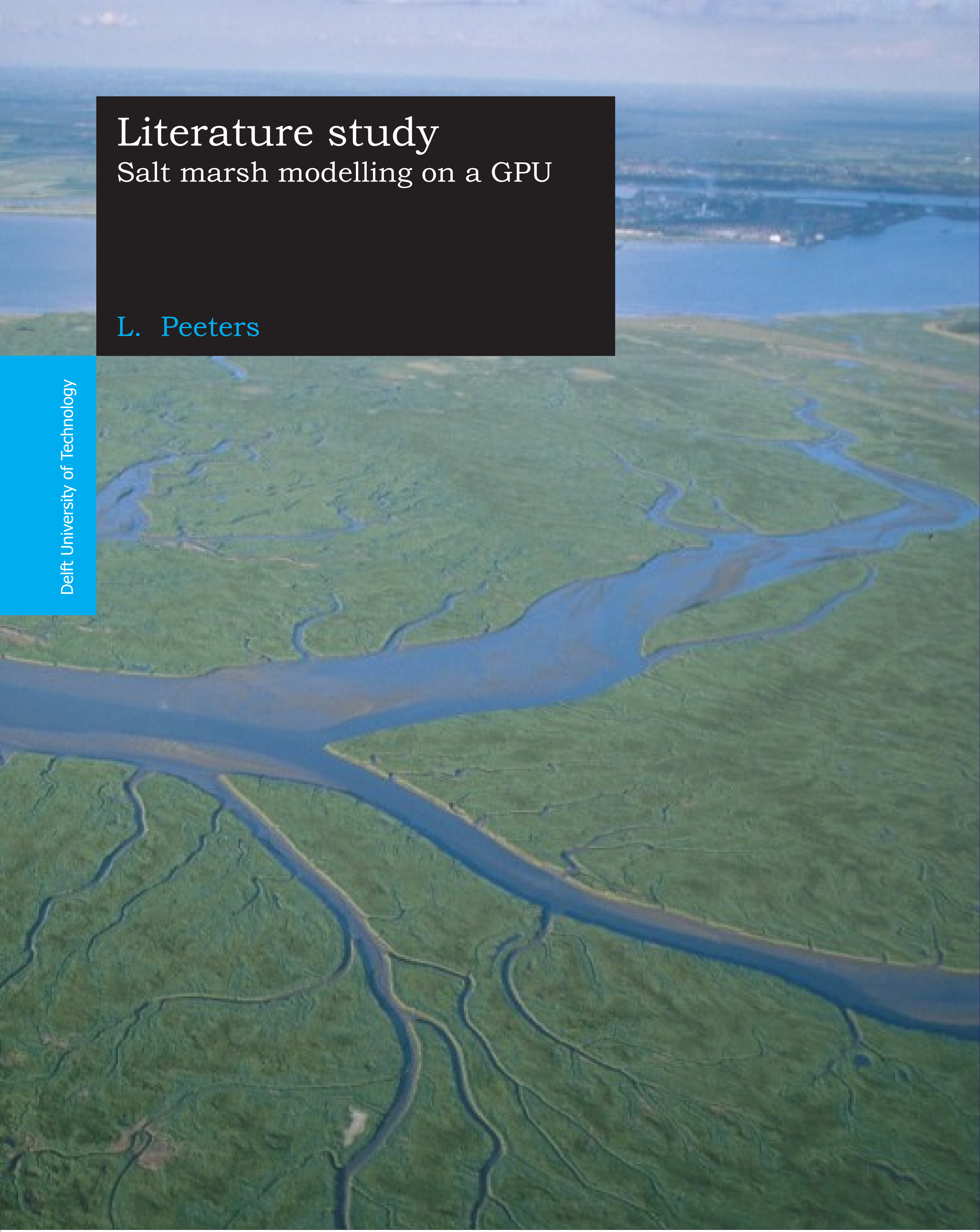


Literature study
Salt marsh modelling on a GPU

L. Peeters

Delft University of Technology

TUDelft Delft University of Technology

Challenge the future

Literature study

Salt marsh modelling on a GPU

by

L. Peeters

Student number:	4237064	
Project duration:	September 1, 2017 – June 1, 2018	
Supervisor:	Prof. dr. ir. C. Vuik,	TU Delft
Thesis committee:	Prof. dr. J. van de Koppel,	NIOZ

Delft University of Technology

Royal Netherlands Institute for Sea Research

List of Figures

Nomenclature

Physics constants

σ	Courant number
f	Coriolis parameter
Fr	Froude number
g	acceleration of gravity
Re	Reynolds number

Other symbols

$\bar{u}, \bar{v}$	averaged velocity components
Δt	time step
$\Delta x, \Delta y$	grid sizes
$\mathbf{n}$	normal coordinate
$\mathbf{s}$	tangential coordinate
μ	dynamic viscosity
ν_{3D}	part of eddy viscosity due to turbulence model in vertical direction
ρ	density
τ_b	bottom shear stress
τ_d	drag force exerted by plants
a	water depth
a_{eff}	effective water depth
C	Chézy roughness coefficient
C_b	bottom friction coefficient
C_d	vegetation friction coefficient
D	diffusion coefficient
f_{MOR}	morphological acceleration factor
h	vertical position of water surface
k	vegetation height
n	Manning coefficient
n_b	stem density
p	pressure
R	hydraulic radius

t	time
u, v, w	velocity components
v	kinematic viscosity
w_s	settling velocity
x, y, z	coordinate system
z_b	position of the bottom

Contents

1 Introduction

A salt marsh is a vegetated coastal ecosystem in the upper intertidal zone between land and open saltwater or brackish water that is regularly flooded by the tides [6]. Salt marshes are characterized by an intertidal surface elevated above the unvegetated tidal flat dissected by a dense network of unvegetated tidal creeks, which supply the salt marsh with sediment and nutrients. The intertidal zones is illustrated in Figure 1.1. During flood tide, water first flows in tidal channels and creeks, and then spreads over the marsh platform by overtopping channel boundaries or by entering through networks of minor channels. During ebb tide, water first drains from the platforms and then concentrates in the channels. Salt marshes have a large ecological value, as lots of animals inhabit these areas and highly adapted vegetation is found. They have an important value to human societies in that they protect the hind land from wave action and can dampen tidal waves, forming a first line of protection prior to human-build coastal defences such as dikes. For these reasons, salt marshes are conserved, protected and restored in many developed countries.

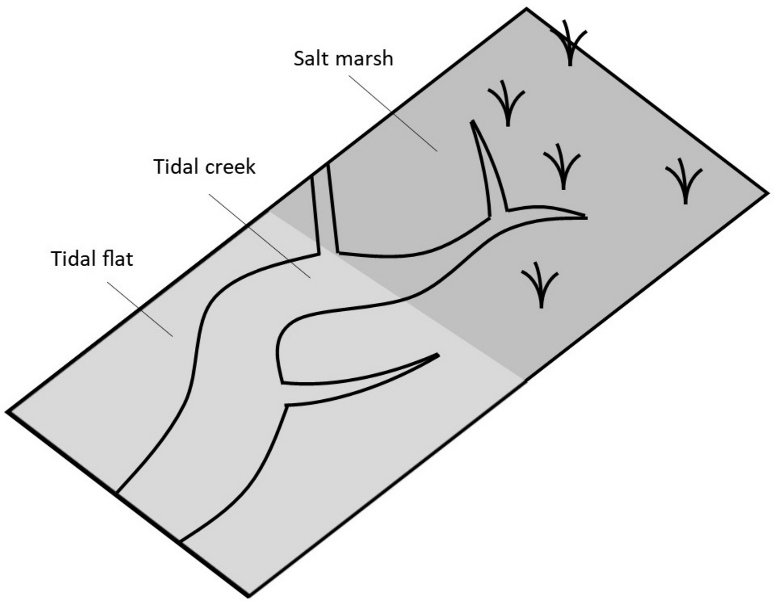


Figure 1.1: Intertidal area.

To help understanding the dynamics of salt marshes and aid in their management, models have been build that can describe salt marsh development. Central in these models are the interactions between vegetation growth, sedimentation-erosion processes and hydrodynamics. To build such a model, typically a vegetation growth model is coupled to a pre-existing hydrodynamic model, such as Delft3D or Telemac that provide a detailed description of water flow and sedimentation processes. The advantage of such an approach is that a relatively accurate and well-tested hydrodynamic and morphological model is implemented with little effort, to be linked to the biological component. There are a number of important drawbacks, however. First, models such as Delft3D are computationally heavy, limiting flexible study of biogeomorphological feedbacks in scientific contexts. Second, they are optimized for applied engineering contexts, to operate at the scales of estuaries, rivers and harbour, and are often ill-equipped to model small-scale biological-physical interactions occurring at small scales (< 1 m scale) on large grids covering extensive areas ($> km$ scale). These drawbacks limit the study of bio-physical processes on salt-marsh evolution.

In this thesis, I implement a coupled biological-physical model of the interaction between vegetation growth, sedimentation processes and hydrodynamics using graphics processors (GPUs) as a novel, emerging computing technology. GPUs are very efficient computing units specialized in floating point operations. For relatively low costs one can obtain supercomputer performance (1 Teraflop). The specialized nature of GPUs implies that hydrodynamic models need to be programmed from scratch, using a highly parallelized programming approach. Yet, their efficiency allows for effective implementation of the models on extensive spatial scales, allowing study of in consequences of small-scale biophysical interactions for salt-marsh development at ecosystem scale.

1.1. Outline

Chapter 2 will derive the shallow water equations and look at the boundary and initial conditions. Chapter 3 focuses on the different model components: hydrodynamics, morphodynamics and vegetation growth. For each component an overview of the most important processes and corresponding formulations will be given. Thereafter, in Chapter 4 we will look at the mathematical side of our model. We will discuss several spatial and time discretisations. Chapter 5 deals with the modelling on a graphics processing unit, several advantages and disadvantages will be listed. Chapter 6 will elaborate more on the specific problem we want to solve. In Chapter 7 the report concludes by outlining the research to be conducted and explaining the test problem.

2 Shallow Water Equations

To model salt marsh evolution three properties need to be modelled, being plant growth, sediment dynamics, and water flow. We will first derive the shallow water equations, which model water flow in areas where the water depth a is much smaller than the characteristic length of the water body. A salt marsh is an example of such area and by using this two-dimensional model the the main characteristics of this area will be represented. In Chapter 3 we will discuss the plant growth and sediment dynamics and elaborate more on the hydrodynamic model. To derive the shallow water equations, we need to look at the conservation laws from which we can subsequently derive the incompressible Navier-Stokes equations. The Navier-Stokes equations can be simplified under certain assumptions to the shallow water equations.

2.1. Model Equations

A conservation law is a statement that, for any attribute which can be neither created nor destroyed but which may merely move, the total rate of outflow from a certain region must equal the rate of decrease of that attribute located within that region. The equations of fluid flow are governed by conservation laws for mass, momentum and energy. We are primarily concerned with the first two, from which we can derive the incompressible Navier-Stokes equations. These are the equations of motion for a flow of a viscous fluid, which is a characteristic exhibited by all real fluids. For an inviscid fluid no shear stresses exist when it is in motion. The Euler equations are the equations of motion for a non-viscous fluid [7].

We denote $\mathbf{x} = (x, y, z)$ to be the position vector and $\mathbf{u} = (u, v, w)$ is the velocity field. The fluid is assumed to be a continuous medium (continuum hypothesis). Physical properties of the flow, such as water density ($\rho(t, \mathbf{x})$) and velocity ($\mathbf{u}(t, \mathbf{x})$), can then be described as time-dependent vector fields [3]. In the next two subsections we need the following two theorems.

Theorem 2.1.1 (Reynold's transport theorem) *For any material volume $V(t)$ and differentiable scalar field ϕ, we have*

$$\frac{d}{dt}\int_{V(t)} \phi dV = \int_{V(t)} \left(\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u})\right) dV \tag{2.1}$$

Theorem 2.1.2 (Divergence theorem) *Let Ω be a bounded domain in $\mathbb{R}^2$ with piecewise smooth boundary Γ. Let $\mathbf{n}$ be the unit outward normal and $\mathbf{v}$ a continuously differentiable vector field, then*

$$\int_\Omega \nabla \cdot \mathbf{v} = \int_\Gamma \mathbf{v} \cdot \mathbf{n} d\Gamma \tag{2.2}$$

2.1.1. Conservation of Mass

The mass conservation law, also known as the continuity equation, says the rate of change of mass in an arbitrary volume $V(t)$ equals the rate of mass production in $V(t)$

$$\frac{d}{dt}\int_{V(t)} \rho dV = \int_{V(t)} \sigma dV \tag{2.3}$$

where $\rho(t, \mathbf{x})$ is the density of the material particle at time t and position $\mathbf{x}$ and $\sigma(t, \mathbf{x})$ is the rate of mass production per volume. We assume σ is equal to zero (only for multiphase flows σ is non-zero, in which σ is zero holds for each flow separately) [3]. Using Equation 2.3 and Theorem 2.1 gives

$$\int_{V(t)} \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u})\right) dV = 0. \tag{2.4}$$

This holds for every volume $V(t)$ so the mass conservation equation is now given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{2.5}$$

An incompressible flow is a flow in which the density of each material particle remains the same during the motion

$$\rho(t, \mathbf{x}(t, \mathbf{y})) = \rho(0, \mathbf{x}(0, \mathbf{y})). \tag{2.6}$$

Incompressibility is a property of the flow and not of the fluid. It does not mean the fluid density is constant, but rather that it is independent of pressure p. Hence

$$\frac{D\rho}{Dt} \equiv \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho = 0. \tag{2.7}$$

We can now derive the following result for incompressible flows

$$\nabla \cdot \mathbf{u} = 0, \tag{2.8}$$

because $\nabla \cdot (\rho \mathbf{u}) = \rho \nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \rho$ and Equations 2.5 and 2.7 hold.

2.1.2. Conservation of Momentum

Momentum is the product of the mass and velocity of an object. Newton's law of conservation of momentum implies the rate of change of momentum and material volume equals the total force on that volume. The total force consist of the body forces ($\mathbf{f}^b$) and surface forces ($\mathbf{f}^s$).

$$\frac{d}{dt}\int_{V(t)} \rho \mathbf{u} dV = \int_{V(t)} \rho \mathbf{f}^b dV + \int_{S(t)} \mathbf{f}^s dS. \tag{2.9}$$

A body force acts on a material particle and is proportional to its mass (for example gravity, centrifugal and Coriolis forces). A surface force works on the surface of $V(t)$ and is proportional to area. Surface forces consist of forces normal to the surface (pressure) and forces tangential to the surface (shear stresses). Substituting $\phi = \rho \mathbf{u}$ into Theorem 2.1 gives

$$\int_{V(t)} \left[\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u})\right] dV = \int_{V(t)} \rho \mathbf{f}^b dV + \int_{S(t)} \mathbf{f}^s dS. \tag{2.10}$$

We can substitute the surface forces $\mathbf{f}^s$ as follow

$$\mathbf{f}^s = T \cdot \mathbf{n} \tag{2.11}$$

where T is given by 2.14 and $\mathbf{n}$ is the outward unit normal on dS [3]. By applying Theorem 2.2 this can be written as

$$\int_{V(t)} \left[\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u})\right] dV = \int_{V(t)} \rho \mathbf{f}^b + \nabla \cdot T dV. \tag{2.12}$$

Since this holds for every $V(t)$, the momentum conservation law is given by

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot T + \rho \mathbf{f}^b. \tag{2.13}$$

2.2. Assumptions

We will now explain several assumptions that are made when deriving the Navier-Stokes equations. First we will assume we have an incompressible flow under which Equation 2.8 holds. Also we will assume gravity is our only body force and thus we will have no Coriolis forces appearing in the momentum equations. These indicate the effect of the earths rotation. More information about Coriolis forces is given in Subsection 3.1.

We will now define a relation between the stress tensor and the motion of fluid. For readers which are not familiar with the definitions stress and tensor, additional information can be found in Appendix A. Such relation is needed to complete the system of equations, because otherwise we have fewer equations than dependent variables (underdetermined system [8]) [3]. Newtonian fluids are the simplest mathematical models of fluids that account for viscosity. While no real fluid fits the definition perfectly, many common liquids and gases, such as water and air, can be assumed to be Newtonian. From now on we will assume water is a Newtonian fluid and thus the stress tensor T can be written as [9]

$$T = -\left(p + \frac{2}{3}\mu\nabla\cdot\mathbf{u}\right)I + 2\mu D, \quad T_{ij} = -\left(p + \frac{2}{3}\mu\nabla\cdot u\right)\delta_{ij} + 2\mu D_{ij} \tag{2.14}$$

where μ is the dynamic viscosity, I is the unit tensor, p is the static pressure, δ_{ij} is Kronecker delta ($\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise) and D is the rate of strain given by

$$D = \frac{1}{2}[\nabla\mathbf{u} + (\nabla\mathbf{u})^T], \quad D_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.15}$$

The nine components (T_{ij}) fully describe the stress for each fluid element in motion [7]. Its diagonal components correspond to normal stresses, while the nondiagonal ones correspond to shear stresses [10]. Shear stresses arise in viscous fluids (all real fluids) as a result of the relative motion between the fluid and its boundaries or between adjacent layers of fluid. The viscous part of the stress tensor, τ, is obtained by leaving out the pressure p in Equation 2.14

$$\tau_{ij} = 2\mu D_{ij} - \frac{2}{3}\mu\delta_{ij}\text{div }\mathbf{v}. \tag{2.16}$$

Another assumption we make has to do with the density. For realistic temperature and salinity variations only small variations in density occur. Such small variations have no important consequences in most terms, so that we can just take a constant density: $\rho = \rho_0$. The only part where the density variations are important is in the gravity term ρg in the momentum equation in z-direction. In this term we will use the actual density. The approach of taking density variations into account only in the gravity term is termed the Boussinesq approximation and is commonly made in almost all kinds of geophysical flows [11].

2.3. Navier-Stokes Equations

The Navier-Stokes equations are derived from the mass, momentum and energy conservation laws. We will focus on incompressible flows, for which effectively the energy equation is replaced by the condition of incompressibility 2.8. If the only body force we take into account is the gravitational force and substitute $T_{ij} = -p\delta_{ij} + \tau_{ij}$ Equation 2.13 becomes

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial x}(\rho u^2) + \frac{\partial}{\partial y}(\rho uv) + \frac{\partial}{\partial z}(\rho uw) + \frac{\partial p}{\partial x} - \frac{\partial\tau_{xx}}{\partial x} - \frac{\partial\tau_{xy}}{\partial y} - \frac{\partial\tau_{xz}}{\partial z} = 0 \tag{2.17a}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2) + \frac{\partial}{\partial z}(\rho vw) + \frac{\partial p}{\partial y} - \frac{\partial\tau_{xy}}{\partial x} - \frac{\partial\tau_{yy}}{\partial y} - \frac{\partial\tau_{yz}}{\partial z} = 0 \tag{2.17b}$$

$$\frac{\partial(\rho w)}{\partial t} + \frac{\partial}{\partial x}(\rho uw) + \frac{\partial}{\partial y}(\rho vw) + \frac{\partial}{\partial z}(\rho w^2) + \rho g + \frac{\partial p}{\partial z} - \frac{\partial\tau_{xz}}{\partial x} - \frac{\partial\tau_{yz}}{\partial y} - \frac{\partial\tau_{zz}}{\partial z} = 0 \tag{2.17c}$$

where g is the acceleration due to gravity and ρ the density. The viscous stresses τ_{ij} are given by Equation 2.16. For an incompressible flow, we can use Equation 2.7 to get the following momentum equations

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}+\frac{1}{\rho}\left(\frac{\partial p}{\partial x}-\frac{\partial \tau_{xx}}{\partial x}-\frac{\partial \tau_{xy}}{\partial y}-\frac{\partial \tau_{xz}}{\partial z}\right)=0 \tag{2.18a}$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}+\frac{\partial vw}{\partial z}+\frac{1}{\rho}\left(\frac{\partial p}{\partial y}-\frac{\partial \tau_{xy}}{\partial x}-\frac{\partial \tau_{yy}}{\partial y}-\frac{\partial \tau_{yz}}{\partial z}\right)=0 \tag{2.18b}$$

$$\frac{\partial w}{\partial t}+\frac{\partial uw}{\partial x}+\frac{\partial vw}{\partial y}+\frac{\partial w^2}{\partial z}+g+\frac{1}{\rho}\left(\frac{\partial p}{\partial z}-\frac{\partial \tau_{xz}}{\partial x}-\frac{\partial \tau_{yz}}{\partial y}-\frac{\partial \tau_{zz}}{\partial z}\right)=0 \tag{2.18c}$$

in which the viscous stresses τ_{ij}, using Equations 2.8 and 2.16, are given by

$$\tau_{ij}=\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right). \tag{2.19}$$

Thus for an incompressible flow (together with the assumptions we made in Section 2.2) the Navier-Stokes equations are given by Equations 2.18a-2.18c in combination with 2.19 and 2.8.

Reynolds Time-averaging Procedure

Although the Navier-Stokes equations are generally believed to describe turbulence, that is not particularly useful as our interest will be in the large-scale features only [11]. In order to isolate those, we will average the equations in some way. Here we suppose that each variable can be split into a slowly varying ”mean” value and ”random” value

$$u=\overline{u}+u'. \tag{2.20}$$

If we substitute this splitting into the Navier-Stokes equations and take the average, we obtain the Reynolds equations for the statistical average of a turbulent flow. These have the same form as the original equations; the difference is that additional stresses, referred to as Reynolds stresses appear (turbulent part) [11], [7]. If we combine them with the viscous stresses given in Equation 2.19 we get

$$\frac{\tau_{ij}}{\rho}=\nu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\overline{u_i'u_j'}. \tag{2.21}$$

2.4. Shallow Water Equations

In a lot of areas, for example in a tidal landscape or river, the water depth is much smaller than the horizontal length scales (includes physical dimensions such as width of estuary, horizontal scales of bottom topography, wave length). How small the ratio of the scales should be is not easy to say [11]. Under this shallowness assumption the Navier-Stokes equations can be simplified to the 3D shallow water equations (SWE). This derivation will be explained in Subsection 2.4.2 The 2DH SWE can be derived from the 3D SWE by integrating over the depth $a(x,y,t) = h(x,y,t) - z_b(x,y,t)$, where h represents the vertical position of the surface water and z_b the position of the bottom as shown in Figure 2.1. Which will be done in Subsection 2.4.3. For this derivation we need some necessary surface and bottom boundary conditions which we will state first.

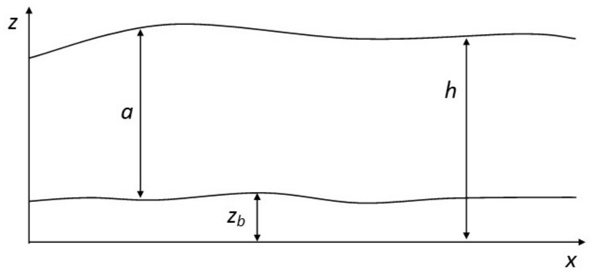


Figure 2.1: Sketch of the relevant variables.

2.4.1. Boundary Conditions

Surface and bottom conditions come in two kinds, we have kinematic, which say that normal water particles will not cross the boundary (impermeability), and dynamic conditions, which say something about the forces acting at the boundaries. The necessary boundary conditions are:

- We have a no-slip condition at the bottom
$$u_{z_b} = v_{z_b} = 0 \tag{2.22}$$
because we may assume the viscous fluid "sticks" to the bottom.
- Water particles will not cross the boundary at the bottom (no relative normal flow at the bottom)
$$\frac{\partial z_b}{\partial t} + u_{z_b}\frac{\partial z_b}{\partial x} + v_{z_b}\frac{\partial z_b}{\partial y} - w_{z_b} = 0. \tag{2.23}$$
Using the no-slip condition at the bottom (Equation 2.22) this gives
$$\frac{\partial z_b}{\partial t} - w_{z_b} = 0. \tag{2.24}$$
- Water particles will not cross the boundary at the surface (no relative normal flow at the surface)
$$\frac{\partial h}{\partial t} + u_h\frac{\partial h}{\partial x} + v_h\frac{\partial h}{\partial y} - w_h = 0. \tag{2.25}$$
- The pressure at the surface is equal to the atmospheric pressure p_a
$$p = p_a \text{ at } z = h. \tag{2.26}$$
The atmospheric pressure is the pressure within the atmosphere of the Earth.
- The surface shear stress (τ_{sx}, τ_{sy}) tangent to the water surface is defined as
$$\tau_{sx} = -\tau_{xx}\frac{\partial h}{\partial x} - \tau_{xy}\frac{\partial h}{\partial y} + \tau_{xz} \text{ at } z = h \tag{2.27}$$
and similarly for the y direction.
- The bottom shear stress (τ_{bx}, τ_{by}) is defined as
$$\tau_{bx} = -\tau_{xx}\frac{\partial z_b}{\partial x} - \tau_{xy}\frac{\partial z_b}{\partial y} + \tau_{xz} \text{ at } z = z_b \tag{2.28}$$
and similarly for the y direction.

2.4.2. Three-dimensional Shallow Water Equations

The central property in shallow water theory is that Equation 2.18c simplifies to the hydrostatic pressure distribution
$$\frac{\partial p}{\partial z} = -\rho g, \tag{2.29}$$
because all terms in Equation 2.18c are small relative to the gravitational acceleration and only the pressure gradient remains to balance it [11]. By using that at the free surface the pressure is equal to the atmospheric pressure, p_a, we find
$$p(z) = g\int_z^h \rho d + p_a. \tag{2.30}$$
We focus on depth-averaged SWE and thus we assume density to be constant over the depth, which gives
$$p = \rho g(h - z) + p_a. \tag{2.31}$$

in which $z = 0$ represents a certain reference plane. This equation implies the pressure is larger closer to the bottom. We can use it to determine $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$ and remove pressure from the momentum equations

$$\frac{\partial p}{\partial x} = \rho g \frac{\partial h}{\partial x} + g(h-z)\frac{\partial \rho}{\partial x} + \frac{\partial p_a}{\partial x} \tag{2.32a}$$
$$\frac{\partial p}{\partial y} = \rho g \frac{\partial h}{\partial y} + g(h-z)\frac{\partial \rho}{\partial y} + \frac{\partial p_a}{\partial y} \tag{2.32b}$$

The atmospheric pressure gradient $\frac{\partial p_a}{\partial x_i}$ may be important for the simulation of storm surges (tsunami-like phenomenon). This is not a goal of our model, so we will disregard this term. From now on we will use the Boussinesq assumption and take a constant density for all terms. Collecting the results we get the following 3D shallow water equations

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.33a}$$
$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} + g\frac{\partial h}{\partial x} + \frac{g(h-z)}{\rho_0}\frac{\partial \rho}{\partial x} - \frac{1}{\rho_0}\left(\frac{\partial \tau_{x,x}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z}\right) = 0 \tag{2.33b}$$
$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} + g\frac{\partial h}{\partial y} + \frac{g(h-z)}{\rho_0}\frac{\partial \rho}{\partial y} - \frac{1}{\rho_0}\left(\frac{\partial \tau_{y,x}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z}\right) = 0 \tag{2.33c}$$

2.4.3. Two-dimensional Depth-averaged Shallow Water Equations

We will now derive the two-dimensional depth-averaged (2DH) shallow water equations. Therefore we need to integrate the 3D SWE (Equations 2.33a-2.33c) over the water depth $a = h - z_b$. It is a logical step, because under the shallowness condition, conservation of mass implies that the vertical velocity of the fluid is small [11]. Vertical integrating allows the vertical velocity to be removed from the equations. To be able to perform this derivation we thus assume h is greater than z_b. The depth-averaged velocities are given by

$$\overline{u} = \frac{1}{a}\int_{z_b}^{h} u dz \tag{2.34a}$$
$$\overline{v} = \frac{1}{a}\int_{z_b}^{h} v dz \tag{2.34b}$$

These velocities are independent of z. We will first derive the averaged continuity equation and secondly the averaged momentum equations.

Averaged Continuity Equations

To integrate the continuity equation, we need to use Leibniz integration rule, which states

$$\frac{\partial}{\partial x}\int_{a(x)}^{b(x)} f(x,t)dt = f(x,b(x))\frac{d}{dx}b(x) - f(x,a(x))\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)dt. \tag{2.35}$$

It follows

$$\int_{z_b}^{h} \frac{\partial u}{\partial x}dz = \frac{\partial}{\partial x}\int_{z_b}^{h} udz - u_h\frac{dh}{dx} + u_{z_b}\frac{dz_b}{dx} \tag{2.36}$$

in which u_h, v_h denote the velocities at the surface (h) and u_{z_b}, v_{z_b} at the bottom (z_b). Thus

$$\begin{aligned} 0 &= \int_{z_b}^{h}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)dz \\ &= \frac{\partial}{\partial x}(a\overline{u}) + \frac{\partial}{\partial y}(a\overline{v}) + [w]_{z_b}^{h} - u_h\frac{\partial h}{\partial x} - v_h\frac{\partial h}{\partial y} + u_{z_b}\frac{\partial z_b}{\partial x} + v_{z_b}\frac{\partial z_b}{\partial y} \end{aligned} \tag{2.37}$$

We will impose the kinematic boundary conditions on the fluid surface and bottom which are given in Equations 2.23 and 2.25. The kinematic conditions say that water particles will not cross the boundary. Equation 2.37 now results in

$$\begin{aligned}&\frac{\partial h}{\partial t}-\frac{\partial z_b}{\partial t}+\frac{\partial}{\partial x}(a\overline{u})+\frac{\partial}{\partial y}(a\overline{v})\\&=\frac{\partial a}{\partial t}+\frac{\partial}{\partial x}(a\overline{u})+\frac{\partial}{\partial y}(a\overline{v})=0\end{aligned} \tag{2.38}$$

Averaged Momentum Equations

As a final step we need to integrate the horizontal momentum equations. By again using the Leibniz integration rule for the first three terms (Equation 2.35) we get

$$\begin{aligned}\int_{z_b}^{h}\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}dz&=\frac{\partial}{\partial t}\int_{z_b}^{h}udz+\frac{\partial}{\partial x}\int_{z_b}^{h}u^2dz+\frac{\partial}{\partial y}\int_{z_b}^{h}uvdz\\&+u_h\left(-\frac{\partial h}{\partial t}-u_h\frac{\partial h}{\partial x}-v_h\frac{\partial h}{\partial y}+w_h\right)\\&+u_{z_b}\left(\frac{\partial z_b}{\partial t}+u_{z_b}\frac{\partial z_b}{\partial x}+v_{z_b}\frac{\partial h}{\partial y}-w_{z_b}\right)\end{aligned} \tag{2.39}$$

The first term in brackets disappears due to Equation 2.25 and the second term in brackets due to Equation 2.23 we get

$$\int_{z_b}^{h}\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}dz=\frac{\partial}{\partial t}\int_{z_b}^{h}udz+\frac{\partial}{\partial x}\int_{z_b}^{h}u^2dz+\frac{\partial}{\partial y}\int_{z_b}^{h}uvdz \tag{2.40}$$

The first integral is by definition equal to $\frac{\partial}{\partial t}(a\overline{u})$. In the last two integrals we get multiple nonlinear terms, because

$$\begin{aligned}&\int_{z_b}^{h}(u-\overline{u})(v-\overline{v})dz\\&=\int_{z_b}^{h}uvdz-\overline{u}\int_{z_b}^{h}vdz-\overline{v}\int_{z_b}^{h}udz+\overline{uv}\int_{z_b}^{h}dz\\&=\int_{z_b}^{h}uvdz-a\overline{uv}.\end{aligned} \tag{2.41}$$

By rewriting we get

$$\int_{z_b}^{h}uvdz=a\overline{uv}+\int_{z_b}^{h}(u-\overline{u})(v-\overline{v})dz. \tag{2.42}$$

A similar operation can be applied to obtain

$$\int_{z_b}^{h}u^2dz=a\overline{u}^2+\int_{z_b}^{h}(u-\overline{u})(u-\overline{u})dz. \tag{2.43}$$

And thus Equation 2.40 reduces to

$$\frac{\partial}{\partial t}(a\overline{u})+\frac{\partial}{\partial x}(a\overline{u}^2)+\frac{\partial}{\partial y}(a\overline{uv})+\int_{z_b}^{h}(u-\overline{u})(u-\overline{u})dz+\int_{z_b}^{h}(u-\overline{u})(v-\overline{v})dz. \tag{2.44}$$

We will now move to the averaging of the terms which are the contribution of the gradient of the pressure

$$\int_{z_b}^{h}g\frac{\partial h}{\partial x}+\frac{g}{\rho_0}(h-z)\frac{\partial\rho}{\partial z}dz=ga\frac{\partial h}{\partial x}+\frac{ga^2}{2\rho_0}\frac{\partial\rho}{\partial x} \tag{2.45}$$

At last we will average the stresses. If we again use Leibniz integration rule we see

$$\begin{aligned}
&\int_{z_b}^{h}\left(\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{xy}}{\partial y}+\frac{\partial\tau_{xz}}{\partial z}\right)dz\\
&=\frac{\partial}{\partial x}\int_{z_b}^{h}\tau_{xx}dz+\frac{\partial}{\partial y}\int_{z_b}^{h}\tau_{xy}dz-\left[\tau_{xx}\frac{\partial h}{\partial x}+\tau_{xy}\frac{\partial h}{\partial y}-\tau_{xz}\right]_{z=h}+\left[\tau_{xx}\frac{\partial z_b}{\partial x}+\tau_{xy}\frac{\partial z_b}{\partial y}-\tau_{xz}\right]_{z=z_b}\\
&=\frac{\partial}{\partial x}\int_{z_b}^{h}\tau_{xx}dz+\frac{\partial}{\partial y}\int_{z_b}^{h}\tau_{xy}dz+\tau_{sx}-\tau_{bx} \qquad (2.46)
\end{aligned}$$

We substituted the last terms by τ_{sx} and τ_{bx} due to Equation 2.27 and 2.28. The 2DH SWE are now given by

$$\frac{\partial}{\partial t}(au)+\frac{\partial}{\partial x}(au^2)+\frac{\partial}{\partial y}(auv)+ga\frac{\partial h}{\partial x}+\frac{ga^2}{2\rho_0}\frac{\partial\rho}{\partial x}+\frac{1}{\rho_0}\tau_{bx}-\frac{1}{\rho_0}\tau_{sx}-\frac{\partial}{\partial x}(\overline{T}_{xx})-\frac{\partial}{\partial y}(\overline{T}_{xy})=0 \qquad (2.47a)$$

$$\frac{\partial}{\partial t}(av)+\frac{\partial}{\partial x}(auv)+\frac{\partial}{\partial y}(av^2)+ga\frac{\partial h}{\partial y}+\frac{ga^2}{2\rho_0}\frac{\partial\rho}{\partial y}+\frac{1}{\rho_0}\tau_{by}-\frac{1}{\rho_0}\tau_{sy}-\frac{\partial}{\partial x}(\overline{T}_{xy})-\frac{\partial}{\partial y}(\overline{T}_{yy})=0 \qquad (2.47b)$$

$$\frac{\partial a}{\partial t}+\frac{\partial}{\partial x}(au)+\frac{\partial}{\partial y}(av)=0 \qquad (2.47c)$$

We have omitted the overbars indicating the depth-averaged values and used $\bar{T}_{ij}$ to denote

$$\begin{aligned}
\overline{T}_{ij}&=\frac{1}{\rho_0}\int_{z_b}^{h}\tau_{ij}dz+\int_{z_b}^{h}(u_i-\overline{u}_i)(u_j-\overline{u}_j)\\
&=\int_{z_b}^{h}\nu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\overline{u_i'u_j'}+(u_i-\overline{u}_i)(u_j-\overline{u}_j)dz \qquad (2.48)
\end{aligned}$$

The terms $\frac{1}{\rho_0}\tau_{sx}$ and $\frac{1}{\rho_0}\tau_{sy}$ are contributions of the wind stress [11]. In this report we will not take into account the wind stress, so we disregard these terms. In the case of 2DH SWE the influence of the density gradient is usually small [11], so we will also disregard this term. We have now obtained

$$\frac{\partial}{\partial t}(au)+\frac{\partial}{\partial x}(au^2)+\frac{\partial}{\partial y}(auv)+ga\frac{\partial h}{\partial x}+\frac{1}{\rho_0}\tau_{bx}-\frac{\partial}{\partial x}(\overline{T}_{xx})-\frac{\partial}{\partial y}(\overline{T}_{xy})=0 \qquad (2.49a)$$

$$\frac{\partial}{\partial t}(av)+\frac{\partial}{\partial x}(auv)+\frac{\partial}{\partial y}(av^2)+ga\frac{\partial h}{\partial y}+\frac{1}{\rho_0}\tau_{by}-\frac{\partial}{\partial x}(\overline{T}_{xy})-\frac{\partial}{\partial y}(\overline{T}_{yy})=0 \qquad (2.49b)$$

$$\frac{\partial a}{\partial t}+\frac{\partial}{\partial x}(au)+\frac{\partial}{\partial y}(av)=0 \qquad (2.49c)$$

Usually the terms belonging to $\overline{T}_{xx}, \overline{T}_{xy}, \overline{T}_{yx}$ and $\overline{T}_{yy}$ are related to local velocity gradients as follow

$$-\frac{\partial}{\partial x}(\overline{T}_{xx})-\frac{\partial}{\partial y}(\overline{T}_{xy})=-K_x\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \qquad (2.50)$$

$$-\frac{\partial}{\partial x}(\overline{T}_{xy})-\frac{\partial}{\partial y}(\overline{T}_{yy})=-K_y\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \qquad (2.51)$$

in which K_x, K_y represent some effective dispersion coefficients [7]. To simplify the above system, we could now assume the bottom friction term relates to a sink term and we could for instance take K_x, K_y

equal to aA

$$\frac{\partial}{\partial t}(au) + \frac{\partial}{\partial x}(au^2) + \frac{\partial}{\partial y}(auv) + ga\frac{\partial h}{\partial x} - Aa\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + Sau = 0 \tag{2.52a}$$

$$\frac{\partial}{\partial t}(av) + \frac{\partial}{\partial x}(auv) + \frac{\partial}{\partial y}(av^2) + ga\frac{\partial h}{\partial y} - Aa\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + Sav = 0 \tag{2.52b}$$

$$\frac{\partial a}{\partial t} + \frac{\partial}{\partial x}(au) + \frac{\partial}{\partial y}(av) = 0 \tag{2.52c}$$

in which A is a diffusion term and S denotes a friction term corresponding to the bottom friction terms τ_{bx} and τ_{by}. By using the averaged continuity equation given in Equation 2.52c we get

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial h}{\partial x} - A\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + Su = 0 \tag{2.53a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial h}{\partial y} - A\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + Sv = 0 \tag{2.53b}$$

$$\frac{\partial a}{\partial t} + \frac{\partial}{\partial x}(au) + \frac{\partial}{\partial y}(av) = 0 \tag{2.53c}$$

We will now inspect the terms in Equations 2.53a - 2.53c. The time-derivative is referred to as the local acceleration, while the second and third term are the convective accelerations. These three terms together are the inertia term. The fourth term is the water slope term, the fifth term is the viscosity term and the last term represents friction. To account for the influence of bottom friction we could for instance take

$$S = \frac{g}{C^2 a}\sqrt{u^2 + v^2} = \frac{g}{C^2 a}\|\mathbf{u}\| \tag{2.54}$$

in which C represents the Chézy coefficient, which depends on the bottom roughness [3].

Often the viscosity term (related to the lateral friction terms $(\overline{T}_{ij})$) is also disregarded. Then the SWE are of hyperbolic type (they have real characteristics). This implies that we can think of solutions as combinations of waves [11] and the number of boundary conditions is closely related to the behaviour of characteristics. We will discuss this in Subsection 2.5.2. If we do not disregard the viscous terms the SWE are parabolic [12] .

2.5. Boundary and Initial Conditions

To get a well-posed mathematical problem with a unique solution, a set of initial and boundary conditions need to be prescribed.

2.5.1. Initial Conditions

The plane at time t equal to zero is also a boundary of the region in the x, y, t space. For the shallow water equations the initial water levels (a), bottom elevations (z_b) and horizontal velocities $(u,\ v)$ must be specified on the entire domain. The problem is that we do not know precise initial conditions, so we have to make assumptions. Fortunately, the influence of wrong initial data gradually fades out due to wave damping by bottom friction. And if we have open boundaries that are not fully reflective we also have wave radiation into the ”outside world” [11].

2.5.2. Boundary Conditions

A set of differential equations does not mean anything unless appropriate boundary conditions are specified. The number of boundary conditions is closely related to the behaviour of characteristics if the system is hyperbolic. The SWE are hyperbolic if we do not take lateral friction into account and otherwise parabolic [12]. We will first discuss the difference between open and closed boundaries, subsequently we will discuss the boundary conditions for the hyperbolic and parabolic SWE.

Open and Closed Boundaries

The contour of the model domain consists of parts along ”land-water” lines (river banks, coast lines) which are termed closed boundaries and parts across the flow field which are termed open boundaries. Closed boundaries are natural boundaries and are situated at the transition between land and water. Open boundaries are always artificial ”water-water” boundaries. They are introduced to obtain a limited computational area and so to reduce the computational effort. In nature, waves can cross these boundaries unhampered and without reflection, but any boundary condition gives a certain amount of reflection, so an open boundary may not be as open as we would wish. We could try to define boundary conditions which do not suffer from reflection, but this is in general not possible [11]. To reduce the reflections at the open boundaries a so-called weakly reflecting boundary condition may be applied.

Hyperbolic

For a hyperbolic system the number of boundary conditions specified at any particular point of the boundary should be equal to the number of characteristics entering the region at that point [11]. The cases of zero or three boundary conditions are concerned with supercritical flow, which does not occur very often, while subcritical flow indicates one or two boundary conditions. Whether a fluid is subcritical or supercritical depends on whether the velocity is less or greater than the propagation velocity of an elementary surface wave. In other words whether the Froude number $Fr = \frac{|U|}{\sqrt{ga}}$ is smaller or larger than unity.

We will assume that flow at the boundaries is subcritical (also done in [12]). For subcritical flow we distinguish two situations, namely inflow and outflow. At inflow, we have to specify two boundary conditions and at outflow we have to specify one boundary condition [12]. In a tidal flow this situation will probably change in time between ebb and flood [11]. Summarizing we need exactly one condition for closed boundaries, which is that the normal velocity is zero [11]. For open boundaries [12] takes the first boundary condition to be an external forcing by the water level, the normal velocity, the discharge rate (depth times velocity) or for instance a weakly reflective boundary condition. In [12] the additional condition for inflow is that the velocity component along the open boundary is set to zero. But they mention it would be better to specify the tangential velocity component.

Parabolic

If we do not disregard the viscosity term the system becomes parabolic and the theory of characteristics does not apply any more. We should provide additional boundary conditions to those mentioned before [11]. For a closed boundary we should now also specify whether we have a no-slip boundary for which the tangential velocity is zero

$$u_s = 0 \tag{2.55}$$

here s represents the tangential coordinate. Or a free-slip boundary, where the shear stress is 0

$$\frac{\partial u_s}{\partial n} = 0 \tag{2.56}$$

Here n represents the normal coordinate. On an open boundary, the additional boundary condition is less clear. In [11] they propose

$$\frac{\partial u_n}{\partial n} = 0 \text{ on inflow and}$$
$$\frac{\partial u_s}{\partial n} = 0 \text{ on outflow}$$

3 Model Components

The processes involved in a salt marsh can be divided into three main domains: a hydrodynamic, morphodynamic and vegetation part. These parts will have to exchange information. The hydrodynamic model determines among other things the water depth a and the depth-averaged velocities u and v in the x and y direction. The morphodynamic part is concerned with calculating the bed elevation and the vegetation part with calculation the plant density. In this chapter we will explain multiple processes which could be used in each component. As illustrated in Figure 3.1 the hydrodynamics will be reviewed in Section 3.1, the morphodynamics in Section 3.2 and the plant growth model in Section 3.3. For each

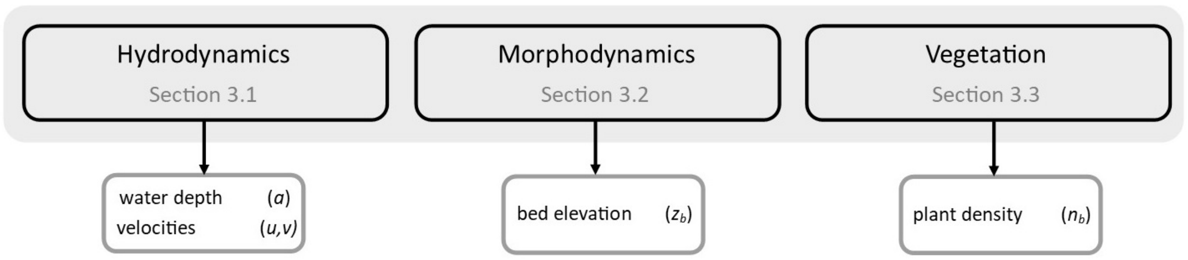


Figure 3.1: Model components.

process, first a small introduction shall be given, after which it is reviewed in more depth.

3.1. Hydrodynamic Model

The main component of the hydrodynamic model will be the shallow water equations (Subsection 2.4). These model water flow in areas where the water depth a is much smaller than the characteristic length of the water body. In the derivation of the SWE in Subsection 2.4, we have already decided to disregard Coriolis forces, wind stress and stratification. We will first explain these factors. The SWE cannot automatically deal with dry areas. Therefore we need wetting-drying methods. Additionally we could consider to add processes like tides, turbulence, meandering and tides to our model. These will be explained afterwards.

3.1.1. Coriolis Forces

The Coriolis effect is the effect of the Earth's rotation which induces a force known as the Coriolis force. This force is determined by the location of the model area on the Earth's globe (the angle of latitude).

The Coriolis force (f) is given by $2\Omega\sin(\phi)$ in which Ω is the angular rate of revolution and ϕ the geographic latitude. The importance of the Coriolis acceleration is measured by means of the Rossby number, defined as $Ro = \frac{V}{fL}$ where V is a measure of the magnitude of the velocity and L is a typical length scale. If $Ro << 1$, the Coriolis acceleration dominates [3]. We assume that we have an inertial frame of reference and thus the Coriolis forces are not taken into account in the momentum equations.

In applications with a rotating geometry (for instance with helicopter blades, in the atmosphere or oceans) it is necessary to use a rotating coordinate system and then the the Coriolis term should be added.

3.1.2. Mixing

Stratification is the formation of water layers based on salinity and temperature, which influence the density. These layers are normally arranged according to density, with the least dense water masses resting above the more dense layers.

The amount of stratification can divide water into the following categories: fully mixed, partly mixed or strongly stratified flow. Suppose that we have two or multiple layer flow with density ρ_i for layer i. We could do the integration over depth (Subsection 2.4.3) for each layer separately to derive $2i$ continuity equations and $2i$ momentum equations [11]. In the derivation of the SWE (Equations 2.53a - 2.53c) we already assumed that the water is well mixed and thus we will use one layer.

3.1.3. Wind Stress

The wind stress is the shear stress exerted by the wind on the surface water. It is affected by the wind speed, the shape of the wind waves and the atmospheric stratification.

For the sake of simplicity we disregarded the wind stress $\boldsymbol{\tau}_s$ in the derivation of the SWE, although wind stress could be an important driving force [11]. In [11] the following formula for the wind stress is given as

$$\frac{\boldsymbol{\tau}_s}{\rho_{\text{air}}} = c_f W^2 \tag{3.1}$$

in which W denotes the wind speed, ρ_{air} the density of the atmosphere and c_f a constant. The wind could also cause an increase in erosion in non-inundated areas.

Optionally we could take into account other weather influences like rainfall and evaporation, which influence the water depth. Evaporation mainly depends on the temperature and causes a decrease in the water depth, while rainfall causes an increase. Rainfall could also cause erosion. Splash erosion is the process in which individual raindrops break down the cohesive structure of the (consolidated) clay during low tide, resulting in a more easily erodible surface layer.

3.1.4. Bottom Friction

The terrain and vegetation exert shear stresses on the passing flow. The bottom stress is an unknown in the SWE and it has to be expressed in terms of the other variables.

In general it is assumed that the bottom stress depends quadratically on the depth-averaged velocities [11]. We could for example take the following formula for $\boldsymbol{\tau}_b$

$$\boldsymbol{\tau}_b = \rho_0 g \|\mathbf{u}\| \frac{1}{C_{2D}^2} \tag{3.2}$$

in which C_{2D} represents the 2D-Chézy coefficient. This formula indeed implies the bottom stress has the same direction as the depth-averaged velocities and depends quadratically on its magnitude [11]. To use the bottom stress for the sink term in the SWE (Equations 2.53a - 2.53c) given in Subsection 2.4.3 we have to divide by the water depth and density

$$S = \frac{\boldsymbol{\tau}_b}{\rho_0 a} \tag{3.3}$$

Different Chézy coefficients exists for different kinds of flows (for instance laminar or turbulent flow). The presence of vegetation also influences the bed shear stress. We will discuss what Chézy value to take in the case of no vegetation and in the presence of vegetation.

No Vegetation

Various expressions for the 2D-Chézy coefficient exist. The Chézy formulation just takes $C_{2D} = C$ [12], while the White–Colebrook formulation is given by

$$C_{2D} = 18 \log \left(\frac{12a}{k} \right) \tag{3.4}$$

in which k is the Nikuradse roughness length and a the total water depth [12]. Or we could use Manning's formulation

$$C_{2D} = \frac{a^{\frac{1}{6}}}{n} \tag{3.5}$$

in which n is the Gauckler-Manning coefficient (emperically derived parameter), which is dependent on many factors including the bed roughness [12]. Often the hydraulic radius R is used in these formulations instead of the water depth a, for instance in [2]. Note that for all these formulations the smaller the water depth, the higher the bottom stress in Equation 3.2.

Vegetation

In Subsection 3.3.2 we are going to explain paper [2] in which different Chézy coefficients are derived, depending on if the vegetation is emergent or submergent. So if vegetation is present we can use the Chézy values given in Equations 3.26 and 3.28 in Equation 3.2 to determine the bottom stress.

3.1.5. Wetting-drying

Salt marshes are tidal areas and so they are subject to alternated wetting and drying. A major issue for the shallow water equations in coastal modeling is their inability to deal with dry areas, where the water height is theoretically zero. The role of any wetting–drying method is to facilitate the appearance and disappearance of dry areas.

This section discusses many ways to deal with this. The crucial issues in a wetting-drying algorithm are

- the way in which the bottom depth is defined at a water level point
- the way in which the water level is defined at velocity points
- the criteria for setting a velocity and/or water level point wet or dry

If we have a collocated grid (Subsection 4.1.1) the first two issues do not apply, because the water depth and velocities are defined at the same grid points. Wetting-drying methods can be classified into two main categories: the deformed mesh (Lagrangian) methods and the fixed mesh (Eulerian) methods. The deformed mesh strategy is to let the nodes on the boundary between wet and dry zones move following the front. A drawback of this method is that an important part of the model is devoted to mesh adaption. The Eulerian methods can again be divided into two main approaches: flux-limiting and modified equation methods. With the flux-limiting strategy, only the discrete algebraic form of the hydrodynamic equations is modified, while with the modified equation strategy it is the original continuous form of the partial differential equations that is modified [13]. We will now give some examples of both methods, which shall again be divided into four categories according to the work of [1]:

- specifying a thin film of fluid over the entire domain
- checking to see if an element or node is wet, dry or potentially one of the two, and subsequently adding or removing elements from the computational domain
- linearly extrapolating the fluid depth onto a dry element and its nodes from nearby wet elements and computing the velocities
- allowing the water surface to extend below the topographic ground surface

These categories will now be discussed in more detail, in Figure 3.3 a representation of each category is shown.

Thin Film Algorithms

Thin film algorithms specify a viscous sublayer of fluid over the entire computational domain. This allows all nodes to be included in the computational domain at each time step. There is typically a minimum threshold depth that defines the categories of wet or dry in the model, even though there is some fluid present over the entire domain. After each time-step the maximum of a certain threshold, H_{dry}, and the water height is taken

$$a = \max(a, H_{dry}). \tag{3.6}$$

and thus the water height will never fall below this threshold.

There are now two possibilities. We could just update the velocities according to the SWE or we could check each cells water depth, and if it drops below H_{dry} we could set the velocity to zero and update the cell state to dry. Both methods require a nonzero depth in each cell.

Element Removing Algorithms

The idea now is to turn off/on mesh cells when the water thickness rises below/above a threshold value. We could also include partially wet elements besides dry and wet ones. The idea of having partially wet elements is shown in Figure 3.2 with a triangular element spatial discretization. The wet elements are included in the computational domain and the dry ones are not. However, in the case of partially wet elements, further consideration is necessary to determine whether the flow conditions at the wetting front are capable of fully wetting a partially wet element.

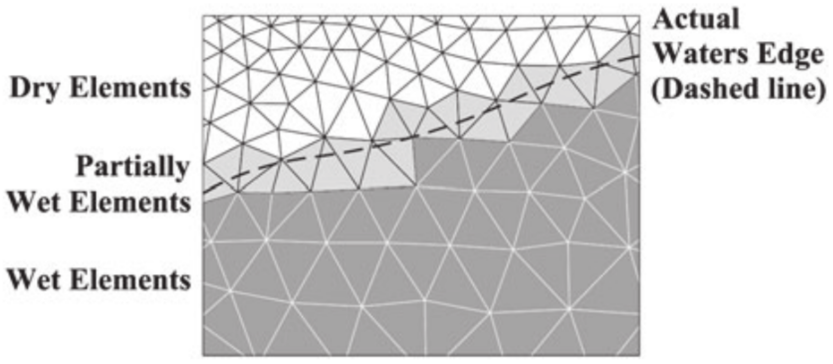


Figure 3.2: Unstructured triangular mesh illustrating wetting front in reality and as seen by numerical model. Image taken from [1].

Depth Extrapolating Algorithms

For this set of algorithms, the conditions at the wetting front are given special consideration and play a vital role in advancing the water's edge in the model. In most cases, the depth is extrapolated from wet cells onto dry cells if the conditions warrant that.

Negative Depth Algorithms

This method is also known as the porosity method in which the hydrodynamic equations are modified to allow water to flow in a porous layer below the bed and thus this approach allows the governing equations to be computed over the entire domain. The water depth can therefore be negative and areas with negative depths are considered to be dry. The wetting of dry cells is simulated, when the flow depth increases and eventually becomes positive. This method avoids handling dry or wet cells separately, but allows unphysical water fluxes through dry zones [13].

The first three methods are all flux-limiting methods. The porosity method follows the modified equation approach. Another method, which was not mentioned in [1] is an approach which allows the bed to move in time as water elevation drops, this leads to a similar formulation as the porous media methods, but without the need to introduce the concept of porosity. Generally, this leads to a simpler numerical formulation and fewer numbers of unknown parameters than in porous media wetting-drying methods [14].

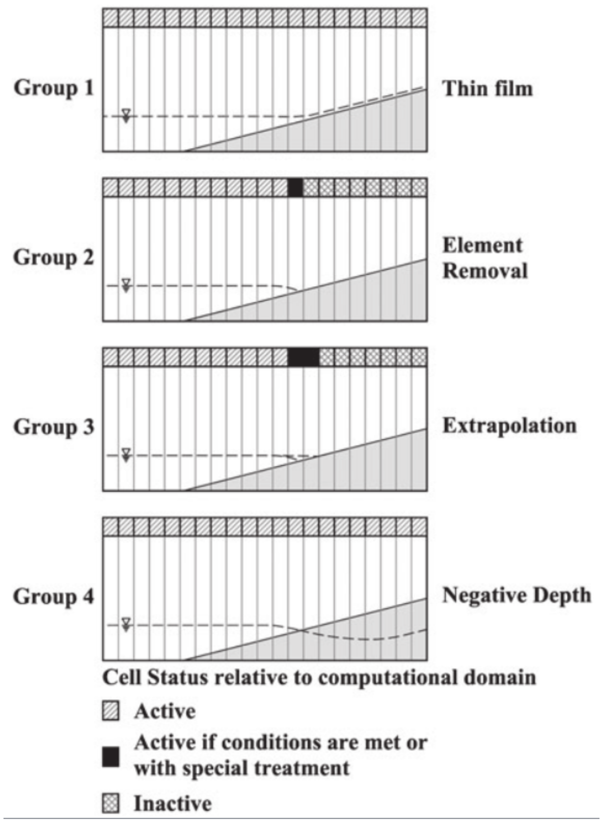

Group 1
Thin film
Group 2
Element Removal
Group 3
Extrapolation
Group 4
Negative Depth
Cell Status relative to computational domain
Active
Active if conditions are met or with special treatment
Inactive

Figure 3.3: Four categories of the wetting-drying algorithm. Image taken from [1].

3.1.6. Turbulence

A laminar flow is well ordered in such a way that each particle moves along a straight path parallel to the boundary. Usually most flows differ from laminar flows, because they exhibit a feature known as turbulence [7], which is characterized by chaotic changes in pressure and flow velocity.

For sufficiently large Reynolds numbers flows show rapid apparently random fluctuations. Such flows are referred to as turbulent. The Reynolds number is given by

$$Re = \frac{\rho U L}{\mu} = \frac{UL}{\nu} \tag{3.7}$$

in which U is a characteristic velocity of the flow, L a characteristic length scale, μ represents the dynamic viscosity, ν the kinematic viscosity and ρ the density. Turbulence still remains one of the great unsolved problems of physics. The difficulty is that turbulence is both nonlinear and stochastic [3]. Since turbulence is governed by the Navier-Stokes equations, one can model it by solving the Navier-Stokes equations. But in order to do this the grid should be sufficiently fine to resolve all flow scales in space and time. This is called direct numerical simulation (DNS) and comes at a corresponding computational price. It is shown that DNS is not feasible for general engineering applications [3]. Besides, we used the Reynolds time-averaging procedure in our derivation of the 2DH SWE, so DNS is not applicable anyway to Equations 2.53a - 2.53c. So if we want to model turbulent flow, we need to find an accurate turbulence model. We will now discuss the Reynolds-averaged Navier-Stokes and large-eddy simulation turbulence models.

The Reynolds-averaged Navier–Stokes (RaNS) equations model all turbulent phenomena. The RaNS equations themselves are open, since unknown turbulent velocities appear. A most commonly used approach is to establish a deterministic turbulence model to close the system of equations for the time-averaged flow [10]. Examples of popular RaNS models are the algebraic eddy viscosity closure model and the $k-\epsilon$ or the $k-L$ turbulence closure models. These models are all based on the so-called "eddy-viscosity" concept. The eddy viscosity, also commonly termed the turbulent viscosity, is related to a characteristic length and velocity scale

$$\nu_{3D} = c'_\mu L\sqrt{k} \tag{3.8}$$

in which c'_μ is a constant determined by calibration, L is the mixing length and k the turbulent kinetic energy [12]. Algebraic turbulence models (AEM), also known as zero equation turbulence models, are models that do not require the solution of any additional equations, but these models use analytical (algebraic) formulas to determine k and L [12]. The $k-\epsilon$ and the $k-L$ turbulence closure models are two equation models, because they include two extra transport equations to represent the turbulent properties of the flow. Here ϵ represents the turbulent dissipation [15].

For certain applications this may not be accurate enough. The alternative is to resolve the large-scale turbulent phenomena (eddies) with sufficiently fine meshes and sufficiently small time steps. This is referred to as large-eddy simulation (LES). The small eddies are modeled heuristically, to diminish the demands on computer resources. This is known as subgrid-scale modelling [3]. The required mesh widths are typically one order smaller than those for RaNS. Adding a corresponding smaller time step, a LES computation will be roughly four orders more expensive than a RaNS computation [16].

These turbulence models are usually defined for 3D flows. But in [11] it is mentioned that it is for instance possible to make the three-dimensional $k-\epsilon$ model well suited for 2D problems, by just averaging the equations in the same way as the momentum equations in Subsection 2.4.3.

3.1.7. Tides
Tides are the rise and fall of sea levels caused by the combined effects of the gravitational forces exerted by the Moon and the Sun and the rotation of the Earth. The influence of other celestial bodies is negligibly small.

The most important motions for the tide are the earth's rotation around its axis (1 day), the moon's orbit around the earth (27,32 days), and the earth's orbit around the sun (365,25 days) [17]. The tidal current (horizontal tide) and the water level (vertical tide) are two appearances of the same tidal phenomenon [17]. The Netherlands experiences approximately two high and low tides each day (semi-diurnal cycle). We could use the same tidal cycle every day, or try to take into account the spring and neap tides. Spring tides result in water levels that are higher than average and low water levels that are lower than average. Neap tides result in less extreme tidal conditions. There is about a seven-day interval between spring and neap tides [18]. In [19] tidal action was simulated by imposing a sinusoidal water-level fluctuation at the north and south open boundary of the grid. This fluctuation had an amplitude of 2.4 m, period of 745 min. The phase difference between both boundaries was 24 s. By using this approach we use the same tidal cycle everyday and disregard differences in the tide's range like spring and neap tides. Besides we only impose the water level without imposing the tidal current.

The ebb surge is commonly stronger than the flood surge as the outflow of the upstream drainage area will be concentrated in the channels whereas the inflow takes place through the channels and over the entire mudflat [20].

Of course the sea level rise also influences the sea level. Between 1890 and 2014 the sea level along the Dutch coast steadily rose by about 1.9 mm per year [21]. While for different places in the Netherlands the mean tidal difference approximately varies between 169 cm and 382 cm [22]. Thus it is probably not essential to take into account the sea level rise.

3.1.8. Meandering
Unvegetated intertidal mudflats and vegetated tidal salt marshes commonly have small meandering channels [23]. A meander is a bend in a sinuous water course. The inner bend bank builds out as a sharp mud ridge while the outer bend bank erodes to form a circular meander pool, the combination leads to bend sharpening.

First the tidal network quickly cuts down the intertidal areas giving them a permanent imprinting. Such a process is later possibly followed by a slower network elaboration, such as meandering [24]. An indicator of the sinuosity of the channels is the relative bend radius, $\frac{R}{W}$ for the bend apices, where R is the bend radius and W is the channel width at the top of the banks [20].

The morphology of several very sharp bends suggested that these bends sharpened over time. Apparently, a positive feedback emerges once a sharper bend is formed. The inner bend bank builds out as a sharp mud ridge, while the outer bend bank erodes to form a circular meander pool. The combination leads to bend sharpening, which explains very sharp meanders commonly observed on cohesive intertidal flats or in meandering rivers in cohesive sediments [20]. Gentle bends erode more slowly. Meander neck cutoff rarely occurs by downstream migration of such sharp bends. In general, meander dynamics are very slow because of the high thresholds [20]. If we compare estuarine meandering channels alluvial meandering rivers, we see they are less dynamic and the dynamics tend to be more localised [25].

3.2. Morphodynamic Model

The morphodynamic model will mainly be concerned with the simulation of erosion, sedimentation, bed elevation and sediment concentrations. First general information regarding sediment transport and sediment types will be given. Afterwards we will give several formulas to model erosion, sedimentation, bed elevation and transport.

3.2.1. Sediment Transport

Sediment transport is the movement of solid particles, typically due to a combination of gravity acting on the sediment, and/or the movement of the fluid in which the sediment is entrained.

Three layers can be distinguished: the bed load, suspended load and wash load. The bed load consists of the larger sediment which is transported by saltation, rolling, and dragging on the bed. The suspended load is the middle layer that consists of the smaller sediment that is suspended. The wash load is the uppermost layer which consist of the smallest sediment that can be seen with the naked eye; however, the wash load gets easily mixed with suspended load during transportation due to the very similar process [26]. We need to decide which layers we want to take into account. The bed load is hard to model, because the dependence on the velocity is small.

3.2.2. Sediment Types

We can also distinguish between cohesive and non-cohesive sediment. Sediment smaller than $63\ \mu m$ is generally defined as cohesive sediment.

Cohesive sediments in water form aggregated larger particles (often called flocs) through binding together (aggregation). The degree of flocculation depends on the salinity of the water [12]. These flocs are much larger than the individual sediment particles and settle at a faster rate [12]. Particles of non-cohesive sediment move individually. Mud belongs to cohesive transport, but sand and bedload belong to non-cohesive transport [12].

3.2.3. Transport Equations

A transport equation is used to determine the sediment concentration at each node. In Subsection 3.2.1 we already gave some information regarding sediment transport.

A well-known transport equation is the convection-diffusion equation. Here two transport mechanisms can be distinguished. Convection is the transport due to the motion of the medium and diffusion is the transport due to differences in concentration. The general convection-diffusion equation is given by

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) - \nabla \cdot (\mathbf{u} c) + R \qquad (3.9)$$

where c is the concentration of the transported quantity, D the diffusivity, and R describes the sources and sinks, which accounts for erosion and sedimentation. The first term at the right-hand side represents diffusion and the second term convection. Suspended sediments are transported across the shallow areas adjacent to channel networks mainly by advection, even though dispersion processes driven by concentration gradients become important as flow velocities decrease [24]. In this report we will assume R is given by the sedimentation rate (Sr) minus the erosion rate (Er). These will be explained in Subsection 3.2.4.

In [19] the morphodynamic model is based on the three-dimensional convection-diffusion equation for suspended sediment transport.

$$\frac{\partial c}{\partial t} + \frac{\partial (uc)}{\partial x} + \frac{\partial (vc)}{\partial x} + \frac{\partial (w - w_s)c}{\partial z} = \frac{\partial}{\partial x}\left(\epsilon_{s,x}\frac{\partial c}{\partial x}\right) + \frac{\partial}{\partial y}\left(\epsilon_{s,y}\frac{\partial c}{\partial y}\right) + \frac{\partial}{\partial z}\left(\epsilon_{s,z}\frac{\partial c}{\partial z}\right) \tag{3.10}$$

where c is the suspended sediment concentration, w_s is the settling velocity of suspended sediment, ϵ_{sx}, ϵ_{sy} and ϵ_{sz} are the eddy diffusivities in the x, y and z direction. We need to keep in mind that the settling velocities of mud and sand are really different. However, we are actually interested in a two-dimensional sediment transport model. Because we use the 2D SWE in our hydrodynamic model and therefore do not determine the velocity in the z direction. For mud transport we can use the relative simple depth-averaged approach

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} + v\frac{\partial c}{\partial y} - \frac{1}{a}\frac{\partial}{\partial x}\left(aD\frac{\partial c}{\partial x}\right) - \frac{1}{a}\frac{\partial}{\partial y}\left(aD\frac{\partial c}{\partial y}\right) - \frac{S}{a} = 0 \tag{3.11}$$

in which a represents the water depth, c the depth-averaged concentration, D the diffusion coefficient and S the source sink term [26], which accounts for erosion and sedimentation.

For the sake of simplicity we could also use a constant suspended sediment concentration at the entire domain. Then we do not need any transport equation.

3.2.4. Sedimentation and Erosion

The amount of erosion and sedimentation determine the bed elevation. Erosion is the action of surface processes (such as water flow or wind) that removes soil, rock, or dissolved material from one location, and then transport it away to another location. Sedimentation is the tendency for particles in suspension to settle out of the fluid in which they are entrained and come to rest against a barrier.

For cohesive sediments, generally the Partheniades-Krone formulation is used [26], in which the erosion rate and sedimentation rate are modeled as functions of bottom shear stress

$$Sr = w_s C_b\left(1 - \frac{\tau}{\tau_{cr,d}}\right) \quad \text{when } \tau < \tau_{cr,d} \tag{3.12}$$

$$Er = M\left(\frac{\tau}{\tau_{cr,e}} - 1\right) \quad \text{when } \tau > \tau_{cr,e} \tag{3.13}$$

where C_b is the suspended sediment concentration at the bottom, τ the bottom shear stress, $\tau_{cr,d}$ the critical shear stress for sedimentation, $\tau_{cr,e}$ the critical shear stress for erosion and M the erosion parameter [27]. At a vegetated marsh platform we often have the case that τ is smaller than $\tau_{cr,e}$, so that erosion may be neglected [27]. This is caused by an increase of shear strength of the sediment bed, due to the roots of salt marsh vegetation. The bed elevation change as a result of erosion and sedimentation is now given by

$$\Delta E = \frac{SR - ER}{\rho_s} \tag{3.14}$$

where ΔE is bottom elevation change (ms^{-1}) and ρ_s is the bulk density of bottom sediment.

Again there are other possibilities to model the erosion, sedimentation and bed elevation. For example without using thresholds. An example to model the erosion rate without thresholds is

$$Er = E_0\left(1 - p_E\frac{n_b}{k}\right)\|u\|z_b \tag{3.15}$$

in which n_b represents the plant density, E_0 the background erosion rate, k the carrying capacity, and p_E an erosion parameter. This formula takes into account that for a higher stem density and lower net speed the erosion rate is lower.

3.3. Vegetation

In this section we will look at the relationship between flow resistance and the presence of vegetation. The dominant vegetation species found in the tidal (brackish) marshes are: Spartina anglica, Scirpus maritimus, Aster tripolium and Phragmites. While in intertidal areas the most important vegetation types are: Vaucheria sp., Aster tripolium, Scirpus maritimus (Bolboschoenus maritimus) and Phragmites australis. We could decide to use one species in our model. But in reality, multiple species often compete for resources and differentially allocate biomass between above-ground and below-ground regions [28]. We will first look at how to model plant growth.

3.3.1. Plant Growth

In [19] multiple equations are given to model the plant growth of the Spartina anglica, a species of cordgrass which is frequently found in coastal salt marshes. The growth model is the sum of five factors: the initial plant establishment, the lateral expansion, the growth, mortality due to flow stress and mortality due to inundation height. We will explain each factor. Here we use the variable n_b to denote the stem density at the bottom.

- The initial plant establishment is given by

  $$P_{est} n_{b,est} \tag{3.16}$$

  where P_{est} is the chance of plant establishment and $n_{b,est}$ the stem density of a new established tussock.

- The lateral expansion of plants to neighbouring grid cells is given by

  $$D\left(\frac{\partial^2 n_b}{\partial x^2} + \frac{\partial^2 n_b}{\partial y^2}\right) \tag{3.17}$$

  where D is the plant diffusion coefficient.

- The growth of stem density within a cell up to its maximum carrying capacity K is given by

  $$r\left(1 - \frac{n_b}{K}\right) n_b \tag{3.18}$$

  where r is the intrinsic growth rate of stem density.

- The plant mortality by tidal flow stress is given by

  $$PE_\tau(\tau - \tau_{\sigma,p}) \text{ when } \tau > \tau_{cr,p} \tag{3.19}$$

  where PE_τ is the plant mortality coefficient related to flow stress, τ is the bottom shear stress and $\tau_{cr,p}$ is the critical shear stress for plant mortality.

- The plant mortality caused by tidal inundation stress is given by

  $$PE_H(H - H_{cr,p}) \text{ when } H > H_{cr,p} \tag{3.20}$$

  where PE_H is the plant mortality coefficient related to inundation stress, H the inundation height at high tide and $H_{cr,p}$ critical inundation height for plant mortality.

Of course there are other options to model the plant growth. We could for instance just include several of these five factors. Besides instead of taking into account plant establishment at each timestep, we could just start with an initial density and assume no new plants establish. The plant mortality (Equations 3.19 and 3.20) could also be determined without using a threshold with for instance the following formula depending on erosion

$$Er E_c n_b \sqrt{u^2 + v^2} \tag{3.21}$$

in which Er is the erosion rate, E_c a conversion factor from sediment erosion to plant loss.

3.3.2. Friction

The terrain and vegetation exert shear stresses on the passing flow. The magnitude of the shear stress of the bed is often characterised by means of roughness coefficient. Still many research is done to define the relationship between flow resistance and the presence and spatial distribution of vegetation. What we want is a relation between the vegetation characteristics, bed resistance, water depth and equivalent resistance coefficient.

Background

Early measurements (18th century) of flow velocities in channels revealed that the depth-averaged flow velocity $\bar{u}$ was a function of the water level slope i and the hydraulic radius R. The well-known Chézy formula is

$$\bar{u} = C\sqrt{Ri}. \tag{3.22}$$

In [26] the water depth a is used instead of the hydraulic radius R. Here C expresses the hydraulic roughness (Chézy roughness coefficient), a measure of the amount of frictional resistance water experiences, of bed and banks [2]. This parameter was first thought to be a constant. Note that the higher the Chézy value, the lower the roughness or resistance to flow is. This traditional approach of using a single resistance coefficient fails to correctly describe the physics of the phenomenon. One way of improving upon this description is to update the equivalent resistance coefficient based on the computed water depth. Further investigations revealed the following formulas for the Chézy value: those of Strickler, Manning or White-Colebrook. These are explained in Subsection 3.1.4.

But still in all these single roughness equations, vegetation is treated as large bed structures with a logarithmic flow profile above them [2]. While in reality there is also flow through the submerged vegetation. Actually four distinct zones can be identified as can be seen in Figure 3.4. In the first

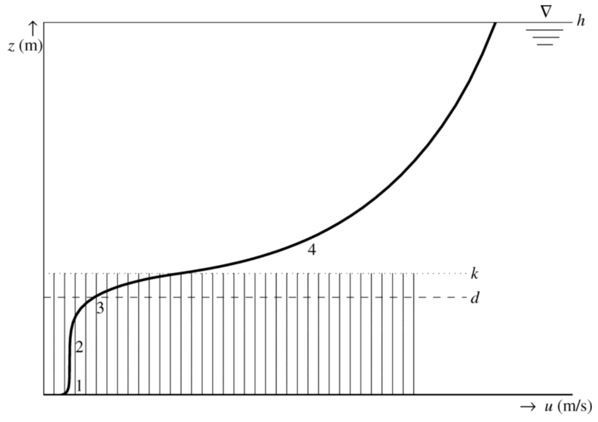


Figure 3.4: Four zones in the vertical profile for horizontal velocity $u(z)$ through and over vegetation. Image taken from [2].

zone the velocity is highly influenced by the bed and its vertical profile joins the logarithmic boundary layer profile. The second zone, in the vegetation, corresponds to a uniform velocity. While in the third zone there is a transitional profile between the zones two and four, where also a logarithmic profile is observed. The White-Colebrook, Strickler and Manning formulas are not correct and another type of resistance formula is needed. In paper [2] analytical expressions for C are found, which depend on the vegetation height k and vertical density. Two cases are considered: fully submerged vegetation or non-submerged (emergent) vegetation. In these expressions the solidity A_p (fraction of horizontal area taken by cylinders) which can be used to correct for the available volume, or available horizontal area in the calculation of fluid stress, is not take into account, because experimental evidence has shown it can be disregarded [2].

Emergent Vegetation

For emergent vegetation the total fluid shear stress ($\boldsymbol{\tau}_t$) is equal to the sum of the bottom shear stress ($\boldsymbol{\tau}_b$) and the drag force exerted by plants ($\boldsymbol{\tau}_d$) [2]

$$\boldsymbol{\tau}_t = \boldsymbol{\tau}_b + \boldsymbol{\tau}_d. \tag{3.23}$$

The total fluid shear stress is defined by

$$\boldsymbol{\tau}_t = \rho_0 g a i \tag{3.24}$$

in which i is the water level slope, g the gravitational acceleration, a the water depth, ρ_0 the fluid density ($\pm 1000\ kg/m^3$), $\mathbf{u}$ the depth averaged velocities and C the Chézy coefficient [2], [26]. The bottom shear stress $\boldsymbol{\tau}_b$ is given in Equation 3.2 and $\boldsymbol{\tau}_d$ is given by Baptist formula

$$\boldsymbol{\tau}_d = \frac{1}{2}\rho C_D m D a \|\mathbf{u}\| \tag{3.25}$$

here m represents the number of cylinders per m^2 horizontal area, D the cylinder diameter, a the water depth and C_D is the dimensionless bulk drag coefficient for flow through vegetation. According to [2], the bulk drag coefficient C_D can be seen as a function of, among other things, the spatial arrangement of the rigid cylinders. This formula considers plants as rigid cylinders with uniform properties and therefore mDa represents the surface of the cylinders. From Equation 3.23 the representative Chézy value for non-submerged vegetation can be deduced

$$C_k = \sqrt{\frac{1}{\frac{1}{C_b^2} + \frac{C_D m D a}{2g}}} \tag{3.26}$$

in which C_b represents the Chézy coefficient of the bed. Note that if we do not have any vegetation (m is equal to zero) the Chézy coefficient is equal to the Chézy coefficient of the bed. When the bed resistance is negligible with respect to the drag force of the vegetation (when we have tall and dense enough vegetation or a very small bed roughness), Equation 3.26 reduces to

$$C_k = \sqrt{\frac{2g}{C_D m D h}} \tag{3.27}$$

These formulas can be used in Equation 3.2 to determine the bottom friction when taking into account both the vegetation drag and bed shear stress.

Submergent Vegetation

For submerged conditions the derivation of an analytical equation for vegetation resistance proves to be more difficult. The four zones in the velocity profile can be reduced to a model of the two most important flow zones. This model, as illustrated in Figure 3.5, consists of a uniform flow velocity, u_c, inside the vegetation and a logarithmic flow profile, u_u, above the vegetation. The Chézy value for

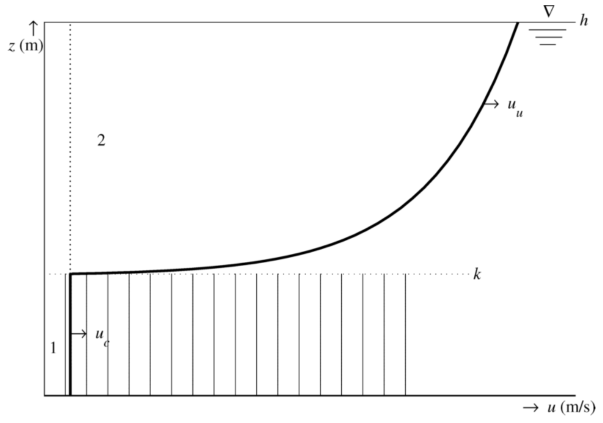


Figure 3.5: Two zones in the vertical profile for horizontal velocity $u(z)$ through and over vegetation. Image taken from [2].

submerged vegetation is then given by

$$C_r = \sqrt{\frac{1}{\frac{1}{C_b^2} + \frac{CmDk}{2g}}} + \sqrt{\frac{g}{k_0}} \ln\left(\frac{a}{k}\right) \tag{3.28}$$

in which k denotes the vegetation height and k_0 represents the Von Karman constant (0.4) [29]. Note that the first term on the right-hand side equals the representative roughness of emergent vegetation when the vegetation height is equal to the water depth (a is equal to k) and the second term goes to zero at the transition from submerged to emerged vegetation. These models only consider rigid vegetation elements, when in reality vegetation is flexible and thus gives rise to complex interactions between flow and vegetation structures [27]. Recent studies showed that the underlying assumption of Equation 3.28, the squared relationship between drag force and flow velocity derived for rigid stems, does not hold for flexible vegetation [29]. Furthermore, the complex vertical structure of real vegetation is neglected. In fact, marsh vegetation may be less dense at low height, where the main stem is located, and denser where the plant structure branches out [27].

How can we use the stem density at the bottom (n_b) in the derived friction coefficients? For rigid cylinders, the area occupied by stems in an area of one square meter, in other words the stem density, is given by $\frac{1}{4}mD^2\pi$. Thus we can use

$$mD = \frac{4n_b}{\pi D} \tag{3.29}$$

in Equations 3.26 and 3.28.

In Delft3D-flow [12] the user may choose to update the computed bed roughness and resistance coefficients less frequently than every time step to save computational time.

4
Numerical Integration

As most mathematical formulations for hydraulic phenomena cannot by solved by analytical methods, because the equations involved are too complicated, we will approximate the solution by using numerical methods. In this chapter we will discuss several space and time integration methods. In the end we will look at the possibility of using multi-scale methods.

4.1. Space Discretisation

Discretisation always entails subdivision of the domain into small cells. The resulting set of cells is called the computational grid. First various grids will be discussed and after that we will look at different space discretisations.

4.1.1. Grid

There are basically three types of grids: Cartesian, structured boundary fitted and unstructured. The different grids are illustrated in Figure 4.1. In a Cartesian grid the cells are rectangular. A grid is referred to as structured if all interior cell vertices belong to the same number of cells. Cartesian and structured boundary fitted grids are therefore structured. If the boundary consists of cell faces then the grid is boundary-fitted [3]. In Cartesian grids the boundary usually does not coincide with the cell faces. The advantage of structured grids is that data structures are easier, leading to smaller

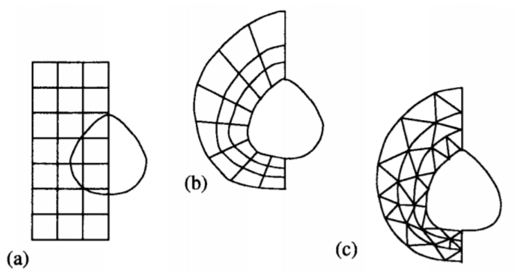


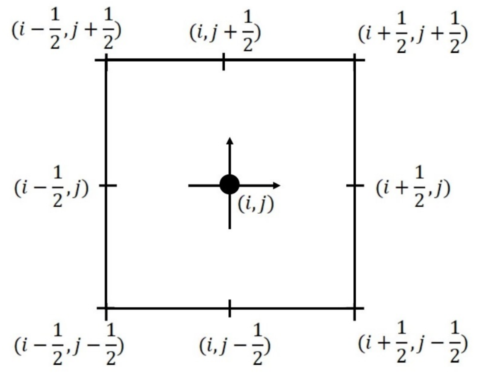
Figure 4.1: Three types of grids: (a) Cartesian; (b) structured boundary fitted; (c) unstructured. Image taken from [3].

computing times and suitability for GPUs. On the other hand in complicated domains the construction of unstructured grids is easier and requires far less computation time [3]. Another disadvantage of structured grids is that it may be difficult to control the distribution of the grid points: concentration of points in one region for reasons of accuracy produces unnecessarily small spacing in other parts of the solution domain and a waste of resources [9].

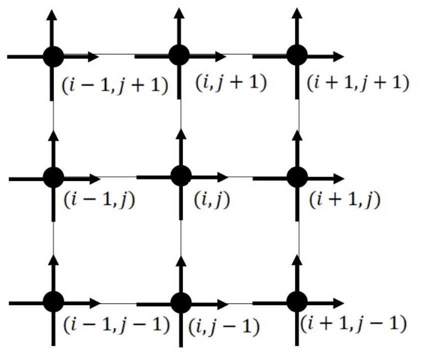
We can distinguish between a cell-centered, in which the grid points form the centers of the cells, or vertex-centered discretisation, in which the grid points are the vertices of the cells. For both discretisations the approach to the interior points will be the same for the finite difference and finite volume method (these methods will be discussed in Subsection 4.1.2), but the examination of the boundary conditions will be different [3].

Collocated and Staggered Grids

For the placement of dependent variables we can use a collocated or staggered approach. When all variables are stored in the same positions, we have a collocated grid arrangement. In a staggered grid not all quantities are defined at the same location in the numerical grid. If we look at the 2DH SWE given in Equations 2.53a, 2.53b and 2.53c the dependent variables are the depth-averaged velocities u, v, water depth a and bottom elevation z_b. Some well-known grids are the Arakawa A, B, C, D and E grids. A collocated grid (Arakawa A) is illustrated in Figure 4.2. Here i and j are the indices of the grid points in the x and y directions. All dependent variables are stored at the same grid points. The


$(i-\frac{1}{2},j+\frac{1}{2})$
$(i,j+\frac{1}{2})$
$(i+\frac{1}{2},j+\frac{1}{2})$
$(i-\frac{1}{2},j)$
(i,j)
$(i+\frac{1}{2},j)$
$(i-\frac{1}{2},j-\frac{1}{2})$
$(i,j-\frac{1}{2})$
$(i+\frac{1}{2},j-\frac{1}{2})$

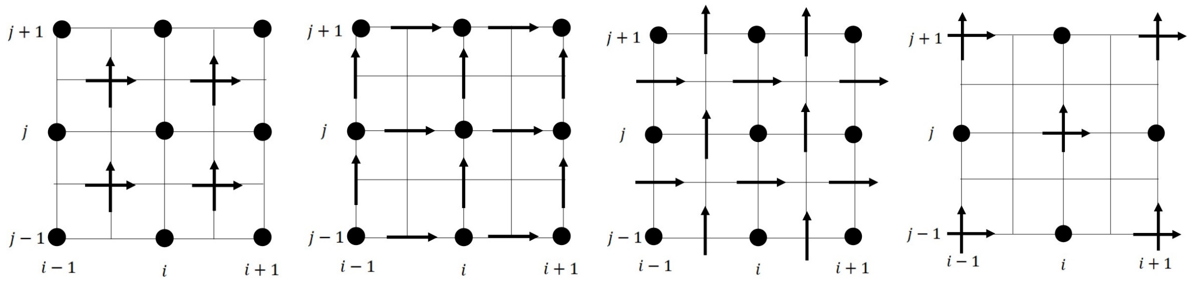
(a) Grid cell.



(b) Grid.

Figure 4.2: Collocated placement of unknowns; →, ↑: velocity components; •: water depth and bed elevation.

Arakawa B, C, D, E grids are illustrated in Figure 4.3. The Arakawa B grid is obtained by taking the flow velocities u and v in the corner points and the water depth a in the cell centres of the grid cell (i,j), the Arakawa C is obtained by taking the flow velocities u at grid points $(i+\frac{1}{2},j)$, v at $(i,j+\frac{1}{2})$ and water depth a at (i,j), etc. Note that with a staggered grid we always have a mixture of vertex-centered and cell-centered discretisations [3]. Here we explain the standard Arakawa grids, but there are more possibilities. For instance the bottom topography could be defined in other grid points than the water depth. For example when we use the Arakawa C grid and the bottom topography varies greatly, we could decide to define z_b in the four corner points instead of once in the center [10] (thus in the grid points $\left(i-\frac{1}{2},j-\frac{1}{2}\right),\left(i-\frac{1}{2},j+\frac{1}{2}\right),\left(i+\frac{1}{2},j-\frac{1}{2}\right)$ and $\left(i+\frac{1}{2},j+\frac{1}{2}\right)$ instead of (i,j)).



(a) Arakawa B (b) Arakawa C (c) Arakawa D (d) Arakawa E

Figure 4.3: Staggered placement of unknowns; →, ↑: velocity components; •: water depth and bed elevation.

An advantage of a staggered grid is that it is possible to use a smaller number of discrete state variables in comparison to discretisations on non-staggered grids, to obtain the same accuracy. This will result in a smaller computation time, although we probably need extra averaging operations to determine the variables in grid points where they are not defined [11]). A second advantage is that staggered grids for shallow water solvers prevent spatial oscillations in the water levels [12]. Odd-even decoupling is a discretization error that can occur on collocated grids and which leads to checkerboard patterns in the solutions. The odd-even ordering of grid points is also called the red-black ordering. A grid point $(x_i,\ y_j)$ is odd or even if $i+j$ is odd or even. Odd-even decoupling means that a variable in an odd point is only coupled with variables in even points and vice versa. Using a staggered grid is a simple way to avoid odd-even decoupling between the water depth and velocity. But staggered grids also have disadvantages. The boundary conditions could for example be hard. If a boundary condition prescribes the value of a variable located at the boundary, this value can be directly substituted. If the boundary condition involves the derivative of this variable or the value of an variable which is not located on the boundary, we have to do something extra. Another disadvantage of using staggered grids is that different variables are stored in different places and this makes it more difficult to handle different control volumes for different variables and to keep track of the metrics.

Ordering of Unknowns

We denote the approximation of the variable u at grid point (x_i, y_j) by $u_{i,j}$. This notation is used for all variables. We will now convert the double index (i, j) into a single index k. Then the approximation of the variable u at grid point x_k is denoted by by u_k. This converting can be done in a number of ways: horizontal numbering, vertical numbering and oblique numbering [30]. For the horizontal numbering the nodes are numbered sequentially in the horizontal direction, the vertical numbering numbers the nodes sequentially in the vertical direction and for oblique numbering the nodes are numbered sequentially along the lines $i+j=k$. We will use the horizontal numbering and thus

$$k = i + (j-1)n \tag{4.1}$$

in which $i \in \{0, \dots, n-1\}$, $j \in \{0, \dots, m-1\}$. Here n represents the number of grid points in the horizontal direction and m the number of grid points in the vertical direction. This numbering is shown in Figure 4.4. We will now denote the dependent variables for the SWE in the control volume k by: $u_k,\ v_k,\ a_k$

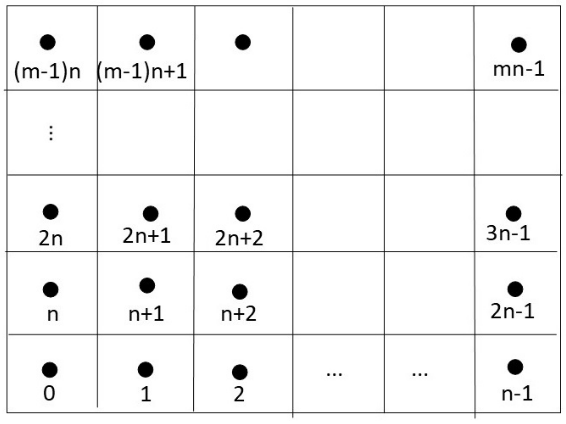


Figure 4.4: Horizontal numbering collocated grid.

and z_{b_k}. We will denote by $\mathbf{u}$ the vector composed of

$$\mathbf{u} = (u_0, v_0, a_0, z_{b_0}, u_1, v_1, a_1, z_{b_1}, \dots, u_k, v_k, a_k, z_{b_k}, \dots u_{mn-1}, v_{mn-1}, a_{mn-1}, z_{b_{mn-1}})^T \tag{4.2}$$

For the staggered grid we have to pay more attention to the ordering. In Figure 4.5 the staggered grid and the numbering is given, indicating which velocity components and depth have the same (array) number in x- and y-direction in the computational grid (grey box). In Figure 4.5 it is shown which velocity components and depth have the same (array) number in x- and y-direction in the computational grid (grey box). The variables corresponding to $\leftarrow, \uparrow$ and $\bullet$ have different control volumes (CV). They are illustrated in Figure 4.6. We will denote the first one by $V_1^{i,j}$, the second one by $V_2^{i,j}$ and the third one by $V_3^{i,j}$.

4.1.2. Discretisation

The most used mesh methods are the finite difference method (FDM), the finite volume method (FVM) and the finite element method (FEM). We will briefly explain them in this Subsection.

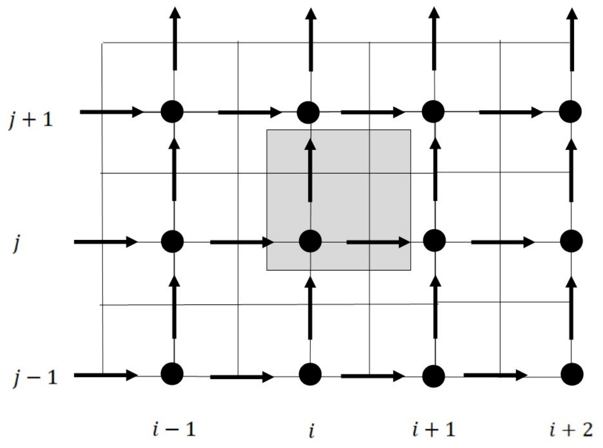


Figure 4.5: Cartesian staggered grid (Arakawa C) in which →, ↑: velocity components; •: water depth and bed elevation.

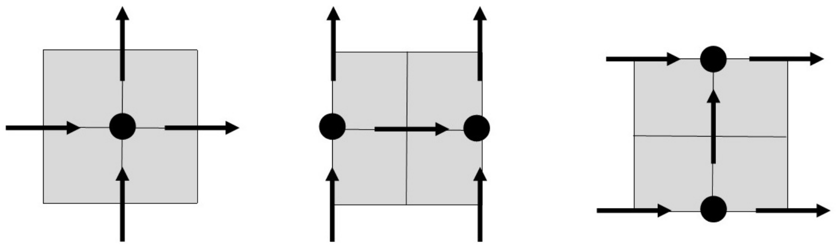


Figure 4.6: Control volume for • (left), → (middle) and ↑ (right) in which →, ↑: velocity components; •: water depth and bed elevation.

Finite Difference Method

Finite difference methods (FDM) approximate spatial derivatives by means of finite differences. Taylor series expansion is used to obtain these approximations. The result is one algebraic equation per grid node, in which the variable value at that grid point and a certain number of neighbor nodes appear as unknowns. Two major disadvantages of this method are that it is not clear how to proceed with non-equidistant grids and natural boundary conditions are hard to implement [30].

Finite Volume method

The finite volume method (FVM) does not posses the major disadvantages of the finite difference method. The FVM can accommodate any type of grid, so it is suitable for complex geometries [9]. The solution domain is subdivided into a finite number of contiguous control volumes (CVs), and it calculates the values of the conserved variables averaged across the volume. Therefore the volume integrals in a partial differential equation that contain a divergence term are converted to surface integrals, using the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. Because the flux entering a given volume is identical to that leaving the adjacent volume, these methods are conservative. Compared to the FDM and FEM only the FVM has an advantage of guaranteeing mass conservation.

Finite Element Method

The FEM is similar to the FVM in many ways. The domain is broken into a finite set of non-overlapping volumes or finite elements that are generally unstructured; in 2D, they are usually triangles or quadrilaterals, while in 3D tetrahedra or hexahedra are most often used [9]. The distinguishing feature of FEMs is that the equations are multiplied by a weight function before they are integrated over the entire domain. In the simplest FEMs, the solution is approximated by a linear shape function within each element in a way that guarantees continuity of the solution across element boundaries [9]. An important advantage of FEM is that it is well suited for unstructured grids. Besides all information in one element is used without considering neighbours. This makes the method very attractive for computer implementation. Another advantage of FEM is that the treatment of boundaries is almost always very natural [30]. A disadvantage is that this method is not conservative and not easy to make conservative. The FDM is also not conservative, but easier to adapt.

4.2. Time Integration

When using these spatial discretisation methods time is kept continuous, so we get a semi-discrete system of ordinary differential equations

$$M\frac{d\mathbf{u}}{dt} + A\mathbf{u} = 0 \tag{4.3}$$

where $\mathbf{u}$ contains all unknowns. Several standard methods exist to integrate these systems. In this section we will discuss some of these methods.

4.2.1. Explicit

When a direct computation of the dependent variables can be made in terms of known quantities, the computation is said to be explicit. In general, for these methods, the time step Δt and step size Δx can not be chosen independently. There is a criterion for the time step, called the Courant, Freidrichs and Lewy (CFL) condition, which represents a condition for stability. This criterion guarantees that the numerical solution is determined only by all the point sources that physically have an influence on the solution [30]. An explicit time integration of the shallow water equations on a rectangular grid is thus subject to the following time step condition based on the Courant number for wave propagation

$$CFL = 2\Delta t\sqrt{ga}\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} < 1 \tag{4.4}$$

where Δt is the time step, g acceleration of gravity, a is the total water depth and $\Delta x,\ \Delta y$ are the grid spaces [12].

4.2.2. Implicit

All explicit methods suffer from the CFL condition, which requires relatively small time-steps to be taken. To avoid such restrictions, implicit methods such as the Crank-Nicolson method can be used. When the dependent variables are defined by coupled sets of equations, and either a matrix or iterative technique is needed to obtain the solution, the numerical method is said to be implicit.

4.2.3. Semi-implicit

Examples of semi-implicit methods are the semi-implicit Euler and semi-implicit leap-frog method. These methods only treat some of the terms implicitly. In the semi-implicit Euler method, the method remains effectively explicit (we do not need to solve systems of equations at each time-step), but the values of the new time-level are used in subsequent equations.

4.2.4. Alternating Direction Implicit

Fully implicit methods have the disadvantage that a large system of algebraic equations has to be solved at each time-step. Another method to avoid this is the alternating direction implicit (ADI) method. In order to reduce computing time, the implicit scheme may be approximated by stages that are implicit in one direction only. The time step is divided in two time stages, where each stage consists of half a time step. In each part only tridiagonal systems have to be solved, along horizontal or vertical grid lines, which is considerably cheap. These methods were widely used for the approximation of shallow water equations [3]. Nowadays computers have more computing power and thus it is not as useful anymore to use ADI methods.

4.2.5. Basic Iterative Methods

To solve a coupled set of equations, obtained for instance after using an implicit method a basic iterative method (BIM) can be used. The idea of these methods is to iterate to solve the linear equations

$$A\mathbf{u} = \mathbf{f}. \tag{4.5}$$

We denote the sequence of iterands by

$$\{u^k\}_{k\geq 0} \text{ where } u^k \to u^* \text{ for } k \to \infty \tag{4.6}$$

here u^0 and u^* denote the initial guess and exact solution of the linear system in 4.5. The error vector of iterand k is given by

$$e^k = u^* - u^k. \tag{4.7}$$

The residual vector at iterand k is given by

$$r^k = f - Au^k. \tag{4.8}$$

The error vector is almost never known, so we will approximate the error by the residual. To construct an iterative scheme, we assume a non-singular matrix M exists and define the matrix N as $N = M - A$, we then write

$$A = M - N. \tag{4.9}$$

Instead of $A\mathbf{u} = \mathbf{f}$ we can now write $M\mathbf{u} = N\mathbf{u} + f$. By multiplying the left and right by M^{-1}, we can define the following iterative scheme

$$\begin{aligned}
\mathbf{u}^{k+1} &= M^{-1}N\mathbf{u}^k + M^{-1}\mathbf{f} \\
&= M^{-1}(M - A)\mathbf{u}^k + M^{-1}\mathbf{f} \\
&= \mathbf{u}^k + M^{-1}(f - A\mathbf{u}^k) \\
&= \mathbf{u}^k + M^{-1}\mathbf{r}^k
\end{aligned} \tag{4.10}$$

Iterative solution methods which can be completely characterized by one single matrix as above are called stationary iterative methods. Examples of stationary iterative methods are: Jacobi, Gauss-Seidel, Richardson, damped Jacobi and the successive-overrelaxation method. In general stationary iterative methods converge slowly. But these methods remain important, because they lend themselves for acceleration [3]. There are two classes of acceleration methods that when combined with a suitable stationary iterative method result in an incredibly efficient method. These are the Krylov subspace methods and the multigrid methods. The stationary iterative method is usually called preconditioner if a Krylov subspace is used for acceleration and is called smoother if multigrid is used [3]. We will now discuss both methods.

Multigrid Methods

Multigrid methods are considered to be among the most efficient solution techniques for systems of linear equations resulting from the discretization of partial differential equations. Their computational efficiency stems from a divide and conquer approach that decomposes the iteration error into frequency components and subsequently treats each component on its most appropriate scale of discretization [4]. Assume we want to solve the following linear system

$$A^h\mathbf{u}^h = f^h. \tag{4.11}$$

The eigenmodes of A^h can be divided into low and high frequency modes. The low frequency modes are slowly varying grid vectors that correspond to the small eigenvalues of $A^h \in \mathbb{R}^{n\times n}$. The low frequency modes have a damping factor close to 1 and are thus slow to converge while high frequency modes have a small damping factor and are thus fast to converge. The above reasoning motivates the use of coarser grids for solving the linear system of 4.11. We could have multiple grids, two or more, and also different cycles can be used: for example the V, W or F-cycle. In Figure 4.7 these different cycles are shown for a four-grid method (we have four vertical layers). To switch between grids of different sizes we need inter grid transfer operators. Restriction operators are used to transfer grid vectors from the fine to the coarse grid and interpolation operators to transfer grid vectors from the coarse to the fine grid.

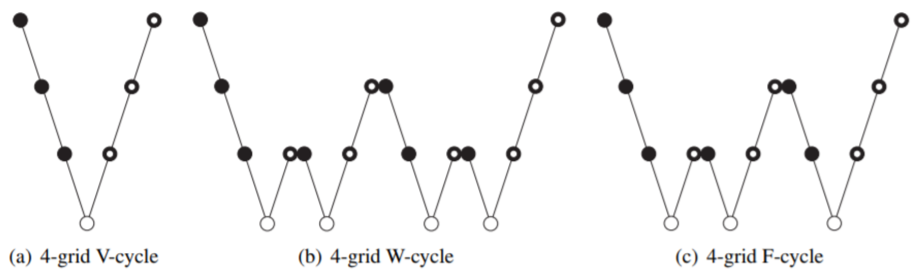


Figure 4.7: Different multigrid cycles for a 4-grid method. Image taken from [4].

Krylov Subspace Methods

We will now explain the idea behind krylov subspace methods. Basic iterative solution methods compute the iterates by the following recursion:

$$u^{i+1} = u^i + M^{-1}(f - Au^i) = u^i + M^{-1}r^i \tag{4.12}$$

where r^i denotes the residual. Writing out the first steps of such a process we obtain:

$$\begin{aligned} & u^0 \\ & u^1 = u^0 + (M^{-1}r^0) \\ & u^2 = u^1 + (M^{-1}r^1) = u^0 + M^{-1}r^0 + M^{-1}(f - Au^0 - AM^{-1}r^0) \\ & \quad = u^0 + 2M^{-1}r^0 - M^{-1}AM^{-1}r^0 \\ & \quad \vdots \end{aligned}$$

This implies that

$$u^i \in u^0 + \text{ span } \{M^{-1}r^0, M^{-1}A(M^{-1}r^0), \dots, (M^{-1}A)^{i-1}(M^{-1}r^0)\}. \tag{4.13}$$

The subspace $K^i(A; r^0) :=$ span$\{r^0, Ar^0, \dots, A^{i-1}r^0\}$ is called the Krylov-space of dimension i corresponding to matrix A and initial residual r^0. Thus a u^i calculated by a basic iterative method is an element of $u^0 + K^i(M^{-1}A; M^{-1}r^0)$.

The conjugate gradient (CG) method is an example of a Krylov subspace method. The CG method has three nice properties [4]:

- the approximation u^i is an element of $K^i(A, r^0)$
- optimality, the A-norm of the error is minimal
- short recurrences, only the results of one foregoing step are necessary; work and memory do not increase for an increasing number of iterations

Unfortunately this method can only be used in the special case that A is symmetric positive definite (SPD) ($A = A^T$ and $x^TAx > 0$ for $x \neq 0$), which will not often be the case. For general matrices it is impossible to obtain a Krylov method, which satisfies all these properties [4]. There are essentially three classes of methods to solve non-symmetric linear systems, while maintaining some kind of othogonality between the residuals

- Solve the normal equations with Conjugate Gradients

  CG applied to the normal equations, LSQR
- Construct a basis for the Krylov Subspace by a three-term bi-orthogonality relation

  Bi-CG type methods: Bi-CG, CGS, Bi-CGSTAB
- Make all residuals explicitly orthogonal in order to have an orthogonal basis for the Krylov subspace

  GMRES type methods: GMRES, GCR

The first class of problems does not satisfy the first property, the second class does not satisfy the second property and the third class does not satisfy the third property. For non-symmetric matrices it is difficult to decide which iterative method to use. The most popular methods are the Bi-CG type methods and the GMRES-type methods. In general CGS and Bi-CGSTAB are easy to implement and reasonably fast for a large class of problems. If break down or bad convergence occurs, GMRES like methods may be preferred. Finally, LSQR always converges, but can take a large number of iterations [4].

4.3. Scales

Solving all processes on the same time and spatial scales could lead to an incredibly large computation time. Therefore we could try to use different scales to solve certain processes. Afterwards we need to take into account how we couple those processes, because the exchange of information will probably be an issue.

4.3.1. Time

The morphological acceleration factor (morfac), f_{MOR}, is used to assist in dealing with the difference in time-scales between hydrodynamic and morphological developments. This approach allows to decouple both timescales by upscaling the bottom elevation changes by a constant factor f_{MOR}, thereby effectively extending the morphological time step

$$\Delta t_{morphology} = f_{MOR} \Delta t_{hydrodynamic}. \tag{4.14}$$

Using this technique, a hydrodynamic simulation of period T corresponds to geomorphic simulation of period $f_{MOR}T$, allowing for longer time geomorphic simulations with no extra computational cost. However, this approach assumes the bottom elevation changes are linear, which is only valid when the morfac is not too high. The selection of a suitable morphological acceleration factor remains a matter of judgement. For a fixed hydrodynamic timestep, larger morfac values reduce the simulation time, but at the same time large morfac may lead to unrealistic morphological change of the model. It is important to find a balance between the low computation costs (large morfac value) and acceptable error.

We will now explain how we can include this acceleration factor. If we run the hydrodynamic and morphological modules both with timestep Δt, we know we know the morphological and hydrodynamical variables at time Δt. Subsequently we can use the acceleration factor to determine the bed elevation at time $f_{MOR}\Delta t$. There are now several possibilities for the bed elevation between time Δt and $f_{MOR}\Delta t$: we could for instance use interpolation to determine the bed elevation at each time between Δt and $f_{MOR}\Delta t$. For the sake of simplicity we could also use the value known at time Δt until we have reached time $f_{MOR}T$ and then use the new value. The hydrodynamic model is run normally. At time $f_{MOR}T$ we have to use this procedure again.

If we want a less complex method, we could just upscale the bottom elevation changes each iteration, without extending the timestep. This is probably unrealistic and will not result in a reduction of computation time.

There is also a big difference in the time-scales between the hydrodynamic and vegetation models. The hydrodynamic processes have a much finer temporal resolution (typically seconds) than the vegetation dynamics (from months to years). Therefore, it could be beneficial to run the hydrodynamic, geomorphic and vegetation models computations all in separate time domains, and again exchange information at specific moments.

4.3.2. Spatial

The integration of processes operating at very different spatial scales (from m^2 to km^2) is a big challenge. Combining, on a single grid, the simulation of all hydrodynamic, sediment and vegetation processes would require tremendous computational capacities, even challenging for state-of-the-art supercomputers. So it would be beneficial to define the vegetation model on a fine grid, but the hydrodynamic and geomorphological parts on a coarser grid. Again the exchange of information will be a challenge.

5 Graphics Processing Units for Scientific Computing

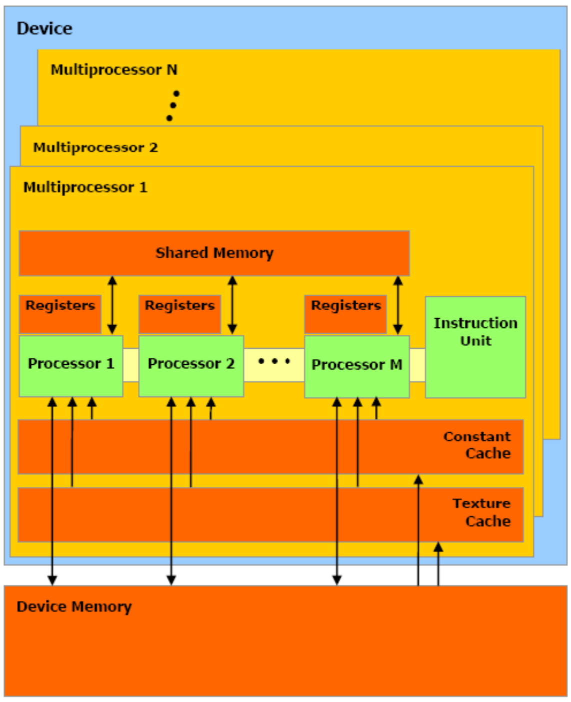
Graphics processing units (GPUs) initially were designed to support computer graphics. Their specific architecture provides high processing power and massive parallelism, which allows for efficient solving of typical graphics-related tasks. However, soon it was realised that such architectures are also suitable for many other computational tasks, such as scientific computing [31]. For relatively low costs one can obtain supercomputer performance (1 Teraflop). It appears however that some work has to be done to make an ordinary program suitable for use on the GPU. The development of the tools CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) helped to reduce this work [5]. But still algorithms must be specifically adapted (in a way which is possibly sub-optimal for a CPU (central processing unit)) in order to realize acceleration on the GPU. In this chapter background information about the GPU is given, such as information about threads, blocks and grids and different types of memory on the GPU. Afterwards we will focus on the advantages and disadvantages of a GPU.

5.1. GPU Architecture

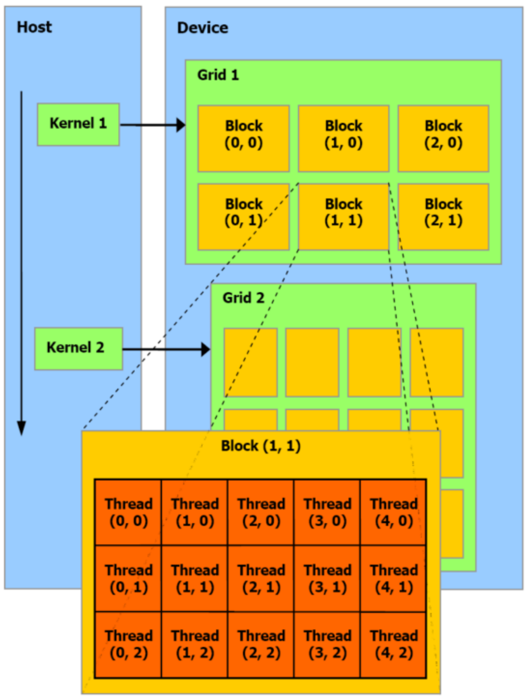
A GPU takes over tasks of a CPU. Calculations are performed much faster on a GPU, because of the fact that it consists of thousands of small, efficient cores, while a CPU consists only of a few cores. The most intensive functions can be computed on a GPU, while other calculations can still be performed on a CPU. The GPU is an SIMD (Single Instruction Multiple Data) processor. A single SIMD instruction encapsulates a request that the same operation be performed on multiple data elements in parallel. A GPU has a fixed number of processors within, these are called multiprocessors (MPs). Each multiprocessor further has eight scalar processors (SPs). A scalar processor is classified as a SISD processor (Single Instruction Single Data) [5]. A general GPU architecture is shown in Figure 5.1a (M is equal to eight). We will first explain what threads, blocks and grids are and subsequently explain the different types of memory on a GPU.

5.1.1. Threads, Blocks and Grids

We can distinguish between threads, blocks and grids. Threads calculate the same parallel computations that have to be performed, but on different parts of an array (so threads can run concurrently). A group of threads together is called a block and a group of blocks together is called a grid. The programmer has to specify how many of such threads exist in each block and how many blocks exist in each grid. One entire grid is handled by a single GPU chip. This chip is organized as a collection of MPs and those MPs are responsible for handling one or more blocks. It never happens that one block is divided over multiple MPs. Each MP is subdivided into multiple SPs and those are responsible for one or more threads. Each thread or block is identified by an index that has an x, y and z coordinate. An illustration of threads, blocks and grids is given in Figure 5.1b. In this figure the term kernel is used. Compute-intensive and data-intensive portions of a given application, called kernels, may be offloaded to the GPU, trying to achieve significant performance while the host CPU continues to execute non-kernel tasks [32].



(a) GPU architecture.



(b) Grids of Blocks with threads.

Figure 5.1: Overview of GPU architecture. Images taken from [5].

Execution of threads

Threads inside a block are grouped into warps. The scheduler that picks up threads for execution, does so in granularity of a warp. So, if the warp size is say, 32, it will pick 32 threads with consecutive thread Ids and schedule them for execution in the next cycle. Each thread executes on one of the SPs. The MPs are capable of executing a number of warps simultaneously. This number can vary from 512 up to 1024 on a GPU depending on the type of card. At the time of issue from the scheduler the MPs are handed over a number of blocks to execute. These can vary on the requirement that each thread imposes in terms of registers and shared memory since they are limited on each multiprocessor. For example suppose the maximum number of threads that can be scheduled on a multiprocessor is 768. Further if each warp is composed of 32 threads then we can have maximum of 24 warps. We could for example have the following division schemes: 256 threads per block * 3 blocks, 128 threads per block * 6 blocks and 16 threads per block * 48 blocks. Now each MP has a restriction on the number of blocks that can simultaneously run on it. So if the maximum number of blocks is say 8 then only the first and second schemes could form a valid execution configuration. By choosing the first two schemes instead of the third we also take into account that threads per block should be a multiple of warp size to avoid wasting computation on underpopulated warps and to facilitate coalescing. Coalescing is the act of merging two adjacent free blocks of memory [33].

5.1.2. Memory Model

Each multiprocessor has a set of memories associated with it. These memories have different access times. It must be noted that these memories are on the device (GPU) and are different from the Dynamic random-access memory (DRAM) available with the CPU. The different memories are

- Register Memory
  There are a fixed number of registers (per block) that must be divided amongst the number of threads (in a block) that are configured (by the previously discussed allocation of threads in blocks and grids). Registers are exclusive to each thread.
- Shared Memory

Shared memory is accessible to all the threads within a block. It is the next best thing after registers since accessing it is cheaper than the global memory.

- Texture Memory
  Texture memory is read-only and could be read by all the threads across blocks on a single multiprocessor. It also has a local cache on the SM.
- Global (device) Memory
  This memory is the biggest in size and is placed farthest from the threads executing on the multiprocessors. Its access times compared to the shared memory (access) latency might be up to 200 times more

The amount of time required to move n data items depends on the latency or start-up time α and the incremental time per data item moved: β, which is related to bandwidth. The bandwidth is the rate at which information can be transferred to or from the memory system. Therefore a simple formula for the time it takes to move n data items is: $\alpha + \beta n$. When a computer has a relatively high latency, it is useful to combine communications.

5.1.3. Single and Double Precision

Both single and double precision floating-point formats are used on GPUs. Single precision numbers occupy four bytes in computer memory, while double precision numbers occupy eight bytes. A byte consists of eight bits. The double precision format is preferred to have a lower aggregate error, but is also slower compared to the single precision format.

For iterative methods it is sometimes possible to use double precision calculations for a certain part of the calculations and single precision for others and still achieving a convergence as good as in the double precision case [5]. Such techniques are called mixed precision methods.

5.2. Application Programming Interfaces

The most popular software to use for a GPU are CUDA and OpenCL. CUDA (Compute Unified Device Architecture) is a software toolkit by Nvidia to ease the use of (Nvidia) graphics cards for scientific programming. OpenCL (Open Computing Language) is a restricted version of the C99 language with extensions appropriate for executing data-parallel code on a variety of heterogeneous devices. OpenCL can run on several types of GPUs (AMD, Nvidia) in contrast to CUDA which can only run on Nvidia graphics cards.

5.3. When to use GPUs

The GPU is specialized for compute-intensive, highly parallel computations and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control (the order in which instructions are executed). More specifically, the GPU is especially well suited to address problems that can be expressed as data-parallel computations (the same program is executed on many data elements in parallel) with high arithmetic intensity (ratio of arithmetic operations to memory operations). Because the same program is executed for each data element, there is lower requirement for sophisticated flow control. Besides big caches are not necessary, because the memory access latency can be hidden with calculations instead of big data caches. A GPU also has very high memory bandwidth compared to a CPU.

It could happen that algorithms only become slower when running them on GPUs. One of the possible causes that lead to longer computation times on a GPU compared to a CPU is the fact that sending information from the CPU to the GPU, and the other way around, is time-consuming. This could be the case if one iteration on the GPU is not fast enough compared to one iteration on the CPU, to catch up the time lost for sending information to the GPU. To achieve a faster calculation on a GPU compared to a CPU we need to perform more calculations on a GPU. And whenever possible we need to limit the amount of data transfers. Other downsides of GPUs we need to keep in mind are: limited memory is available, no error correcting memory (ECC), debugging can be complicated and fast double precision could, depending on the specific GPU, still be quite expensive.

5.3.1. Cluster

We will now explain the advantages of a GPU compared to a cluster. A computer cluster consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. They were designed to get more computing power, but communication is much slower than computation [5]. If we have an algorithm such that each processor can run almost independently of the other processors and little communication is necessary between them, it is beneficial to use a GPU instead of a cluster, because the connected computers in a cluster will send all memory back after each iteration. The cost at which supercomputer performance is available is a couple of hundred euros for a GPU. This makes it an attractive option already compared to setting up or sharing a cluster that might be hard to get access to or costlier to put up in the first place [5].

6 Problem

In the previous chapters we have discussed the various components which could be included in our model as well as several numerical discretisation and integration techniques. In this chapter we will focus on the current model used by the Royal Netherlands Institute for Sea Research (NIOZ) to simulate the dynamics of a salt marsh. At the end we will discuss the drawbacks of this model. The improvements I will focus on, will be stated in the research questions in Chapter 7.

6.1. Current Model

We will first review the domain, the processes involved, the boundary and initial conditions and the discretisations used in this model.

6.1.1. Domain

We will first define several variables. The lengths in the x and the y direction are represented by L_1 and L_2. The grid sizes in the x and y directions are denoted by Δx and Δy. The number of grid points in the x direction is given by n and in the y direction by m. Currently, a simple square, collocated grid is used of $100\ m \times 100\ m$, with the same number of unknowns and grid spacing in the x and the y direction. So we have that $n = m$, $\Delta x = \Delta y$ and $L_1 = L_2$. A Cartesian grid, as illustrated in Figure 6.1, is chosen, because the domain is a simple square. We have used a horizontal numbering for the unknowns.

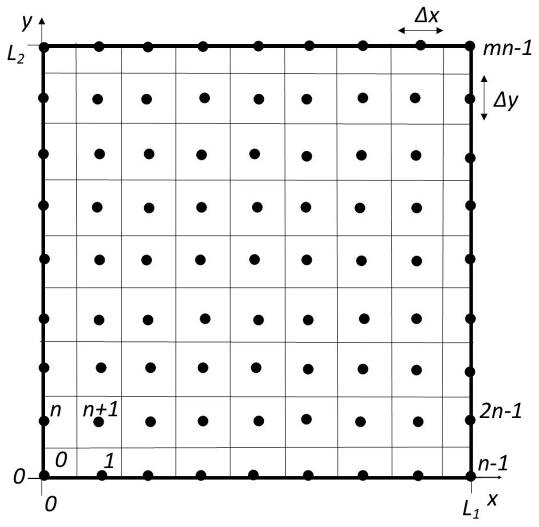


Figure 6.1: Cartesian collocated grid.

6.1.2. Included Processes

The model is based on a coupled morphodynamic, hydrodynamic and vegetation model. As shown in Figure 3.1 the hydrodynamic part is concerned with determining the depth-averaged velocities u, v and water depth a. The vegetation part calculates the vegetation density n_b and the morphodynamic part calculated the bed elevation z_b. We will explain what processes are included in each part and what differential equations are used to described these.

Hydrodynamics

The shallow water equations reflected in Equations 2.53a-2.53c were used. The thin film algorithm as explained in Subsection 3.1.5 is used. This method assumes that there is a thin layer of fluid over the whole computational domain. At the beginning of each iteration it takes the maximum between the water depth and a certain critical value H_{crit}

$$a = \max(a, H_{crit}).$$

By using this wetting-drying algorithm we can apply the 2DH SWE to each grid cell. No turbulence or meandering model is taken into account. The bottom friction is given by

$$\boldsymbol{\tau}_b = \rho g \mathbf{u}||\mathbf{u}||\frac{1}{C^2}$$

in which C represents the Chézy coefficient for submergent vegetation (Equation 3.28).

Morphodynamics

The morphodynamic model consists of three factors which contribute to the sediment loss and gain

$$\frac{\partial z_b}{\partial t} = S_{in} a_{eff} - E_0 \left(1 - p_E \frac{n_b}{K}\right)(u^2 + v^2) z_b + Dif(z_b, p).$$

The first factor is the sediment input. The constant S_{in} ensures that there is more sediment input for a larger effective water height. The effective water height is defined as

$$a_{eff} = a - H_{crit}.$$

Also there will be no gain of sediment at the grid cells in which the water depth was smaller or equal to the critical value before applying the wetting-drying method. The second term accounts for the sediment erosion. The parameter K is the carrying capacity, E_0 the background erosion rate and p_E is the fraction by which sediment erosion is reduced if the vegetation is at carrying capacity. This factor ensures that there is less erosion for a higher plant density, but more erosion for a larger net speed. The last term is a diffusion term which takes for granted less sediment mobility for a higher plant density. We will not discuss this formula in detail.

Vegetation

In this part, the vegetation density is calculated. The plant growth model is given by

$$\frac{\partial n_b}{\partial t} = D\left(\frac{\partial^2 n_b}{\partial x^2} + \frac{\partial^2 n_b}{\partial y^2}\right) + r\left(1 - \frac{n_b}{K}\right) n_b \left(\frac{K_p}{K_p + a}\right) - E_p n_b \sqrt{u^2 + v^2}.$$

The first term accounts for the dispersal of the vegetation, another for the growth and the last for the mortality. Here D is the plant diffusion coefficient, K_p is the value of the water level where plant growth is approximately half the maximum and E_p represents the plant loss due to flow. The dispersal factor is straightforward. The growth factor is logistic, namely the growth rate gets smaller as the vegetation density approaches the carrying capacity. The fraction involving K_p ensures the growth is less for a higher water depth. The last term assumes that there is a higher mortality rate for a higher density and net speed. Compared to the differential equation given in Subsection 3.3.1, the mortality due to inundation height and plant establishment are not taken into account. Only the initial plant establishment is prescribed in the initial conditions. These will be explained in Subsection 6.1.3.

6.1.3. Initial and Boundary Conditions

We will first discuss the boundary conditions. The right boundary in Figure 6.1 is chosen to be the sea side boundary. Thus this boundary is chosen to be open, while the lower, left and upper boundary are defined as closed boundaries. To prevent the tidal creeks from going through the upper, left and bottom boundaries, a vegetation density of one is taken at these boundaries. At the right boundary the normal derivative is set equal to zero (Neumann condition). At the left, top and bottom boundary reflective conditions are used for the velocities u and v. The second derivatives of u and v to x are set equal to zero at the outflow (right) boundary ($\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 v}{\partial x^2} = 0$). For the water depth the normal derivative is set equal to zero at all boundaries. At the outflow boundary it is prescribed that the sediment level is equal to zero. For the closed boundaries also the normal derivative is set equal to zero.

We will now discuss the initial conditions. Randomly 0.2 percent of the grid points are assigned an initial vegetation density of one. An initial slope of 0.002 m/m is assumed to apply in the bathymetry, so the sediment level ranges between zero at the right boundary (sea side), and 0.2 at the left boundary. The initial water height taken is equal to the critical value H_{crit}. The depth-averaged velocities in the x and the y direction are initially set equal to zero throughout the whole computational domain.

6.1.4. Discretisations

In the current model the Euler forward method is used in combination with the finite difference method (central differences). Applying these methods to the SWE given in Equations 2.53a-2.53c gives us the following system to solve for each grid point

$$u_C^{n+1} = u_C^n + \Delta t\left(-u_C\frac{u_E - u_W}{2\Delta x} - v_C\frac{u_N - u_S}{2\Delta y} - g\frac{h_E - h_W}{2\Delta x} + A\left(\frac{u_W - 2u_C + u_E}{\Delta x^2} + \frac{u_S - 2u_C + u_N}{\Delta y^2}\right) - S_C u_C\right)$$

$$v_C^{n+1} = v_C^n + \Delta t\left(-u_C\frac{v_E - v_W}{2\Delta x} - v_C\frac{v_N - v_S}{2\Delta y} - g\frac{h_N - h_S}{2\Delta y} + A\left(\frac{v_W - 2v_C + v_E}{\Delta x^2} + \frac{v_S - 2v_C + v_N}{\Delta y^2}\right) - S_C v_C\right)$$

$$a_C^{n+1} = a_C^n + \Delta t\left(-\frac{a_E u_E - a_W u_W}{2\Delta x} - \frac{a_N v_N - a_S v_S}{2\Delta y}\right)$$

As illustrated in Figure 6.2 C represents the grid point $(i,j) = k$, E the point $(i+1,j) = k+1$, W the grid point $(i-1,j) = k-1$, S the point $(i,j-1) = k-n$ and N the point $(i,j+1) = k+n$.

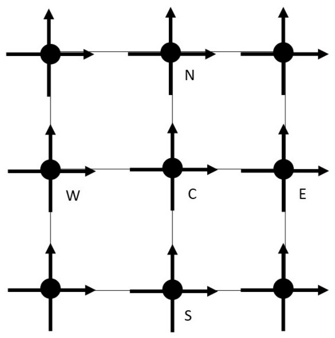


Figure 6.2: Grid.

6.2. Results

An example of a result of the current model is shown in Figure 6.3. In the first three images the right boundary represents the seaside boundary. The top left figure shows the vegetation density (n_b), the top right the sediment level (z_b), while the bottom left image shows the net speed ($\sqrt{u^2+v^2}$). The bottom right image is a cross-section through the y-axis showing the bottom elevation (brown line) and the water height (blue line). Although the model is actually a good representation of the meandering and bifurcations of tidal creeks, there are two main downsides, which we will now discuss. Please note that the meandering is a result of the initial vegetation patches.

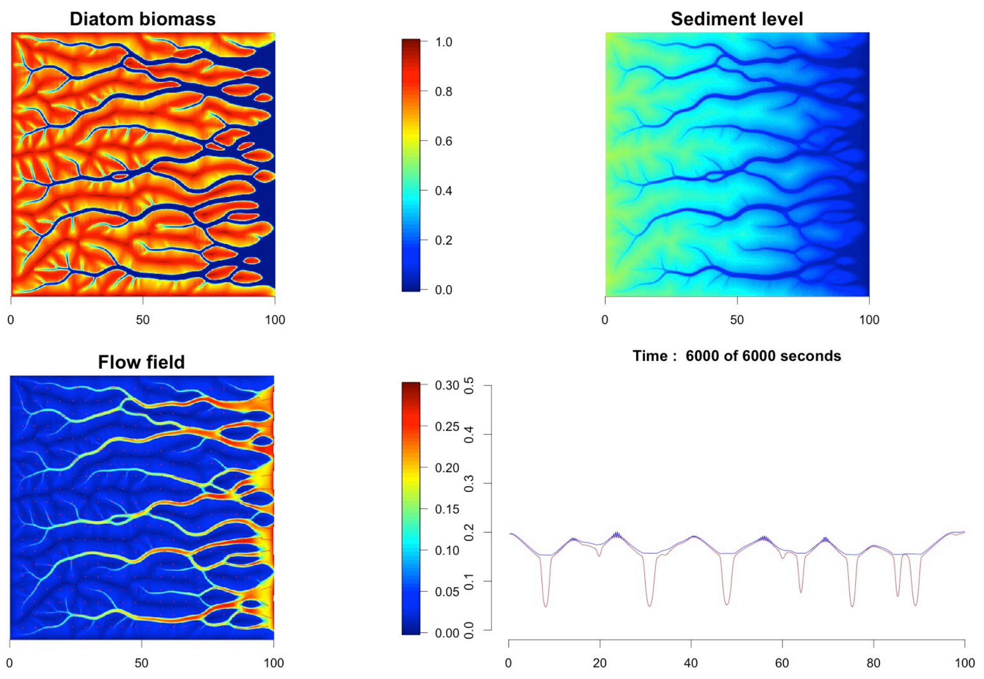


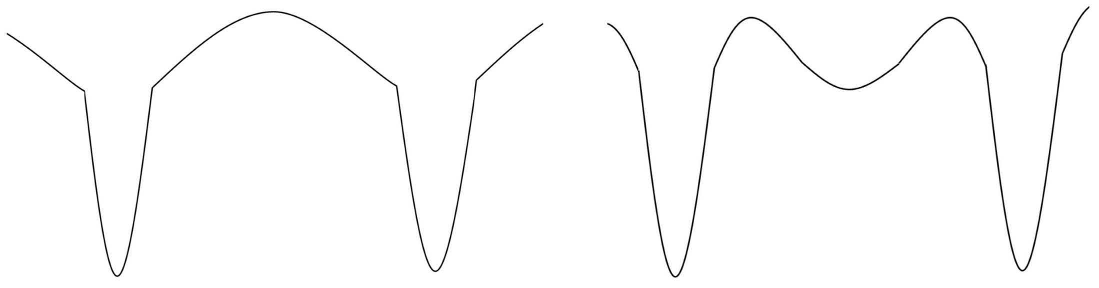
Figure 6.3: The resulting vegetation density, sediment level, net speed and cross section.

Tides

In this model no tidal action is used. Instead a constant input of water is assumed everywhere. The consequence of this approach is that the whole computational domain is reached by the water at all times. In reality, though, it could be the case that the highest parts of the salt marsh are not reached by water during flood. Besides, for a real tidal cycle we also first have an inflow through the right boundary (flood) and subsequently an outflow (ebb). In this model the water always flows to the sea-side boundary. Thus we more or less have ebb tide all the time.

Sediment Concentration

In between tidal creeks we would like most of the sediment to fall down as soon as it floods the area between the tidal creek. This would result in the situation shown in Figure 6.4b. Here two tidal creeks are drawn with small hills of sediment at their boundaries. In our model we have the situation which is shown in Figure 6.4a. This could be related to how we define the sedimentation.



(a) Observed situation. (b) Wanted situation.

Figure 6.4: The area in between two tidal creeks.

7 Conclusion

In this chapter, the literature study is concluded with the future work of the thesis project. After stating the research questions, we propose a test problem area at which our future model can be tested.

7.1. Research

The three main aims of my thesis are stated below.

- How can we improve the current salt marsh model in terms of included processes?
- How can we improve the current salt marsh model in terms of the used grid, discretisation and solver?
- How to implement a multi-scale approach to limit the computation time and cost?

When working on these research questions, we need to take into account the features of the graphics processing unit. When utilizing a GPU, two things are very important: performing as many computations in parallel as possible (this makes the GPU perform as efficiently as possible) and sending as little data as possible back and forth (this negatively affects the computation time on the GPU). Not all methods will be be suited to be implemented efficiently on a GPU. We will now elaborate on each question.

7.1.1. Research Question 1

For this research question we have to find out if we can improve our model by adding processes which were not present, or by changing the formulations which were used. At the moment there is constant input of water at the whole computational domain. Instead we could incorporate tides in our model. Therefore, the water level at the sea side boundary should be prescribed. For instance by Equation 7.1 given in Section 7.2. Subsequently we would like our model to represent the situation in Figure 6.4b instead of Figure 6.4a. We could try to obtain this situation by redefining the sedimentation formula. At the moment it is assumed that there is more sedimentation for a larger effective water depth. Instead we could let it depend on the suspended sediment concentration. For example Equation 3.12 depends on the concentration, but it assumes there is no sedimentation above a certain threshold value. To determine the sediment concentration we can use a transport equation.. For instance the formula given in Equation 3.11. To simplify our formulations we could disregard the dependence of the sediment mobility on algae. Because this is not an essential element of our model.

7.1.2. Research Question 2

This research questions is concerned with the grid, the discretisation and the time-integration method. First we will look at the grid. At the moment a collocated grid is used. Instead we could implement a staggered grid in an attempt to reduce the computation time (staggered grid uses smaller amount of state variables) and prevent spatial oscillations. The Arakawa C grid (shown in Figure 7.1) is often used in computational fluid dynamics. In this image we located the velocity in the y direction on the top and bottom boundaries, the velocity in the x direction on the right boundary and the water depth, bottom elevation and velocity in the y-direction on the left boundary. Of course there are other options and it depends on the prescribed boundary conditions which one we should chose. We will

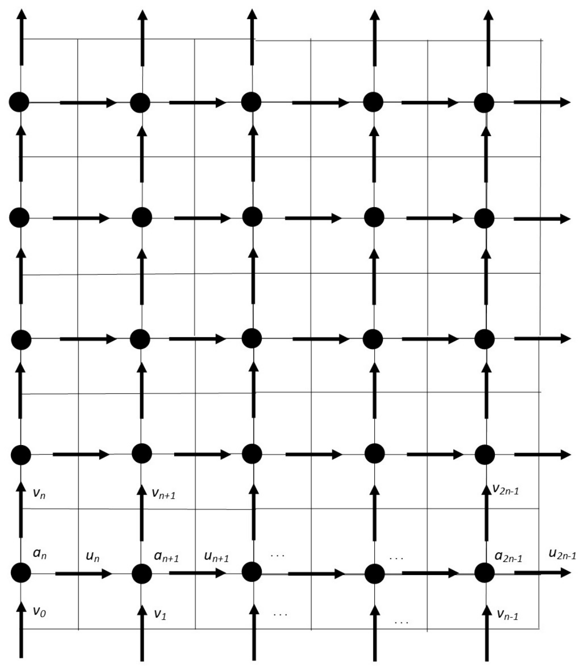


Figure 7.1: Staggered grid (Arakawa C).

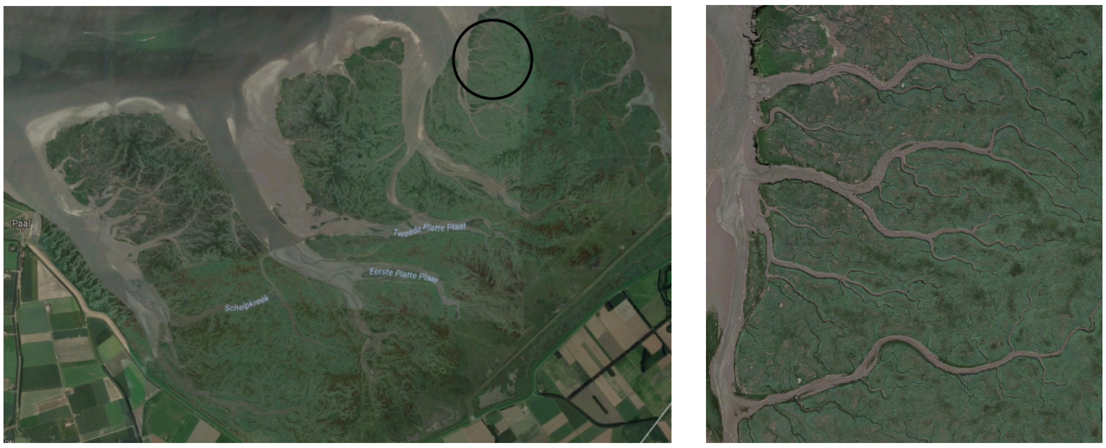
now discuss the spatial discretisation. Currently the finite difference method is used. Since the finite element method is especially well-suited for unstructured grids, and we have a structured grid we will not use this method. We shall implement the finite volume method, since it has the advantage of mass conservation. For time-integration the Euler forward method is used. If the CFL condition limits our time-step to much and time allows it, we could decide to use an implicit method like for instance Euler backward. In addition we could use a basic iterative method (Subsection 4.2.5) to solve the resulting system of equations. If we decide to use an implicit method, it is really important that we take into account the advantages and disadvantages of the implementation on a GPU.

7.1.3. Research Question 3

At the moment the same time and space discretisations are used for all modules. We could decide to use a certain acceleration factor in the morphological and vegetation equations to speed up these processes. First we could use the simple version of morphological acceleration factor, which upscales the bottom elevation each iteration. Afterwards we could use the more sophisticated approach which reduces the computation time. These method were both explained Subsection 4.3.1. The MORFAC is a multi-time-scale method. If time allows we could look at multi-space-scale methods and for instance decide to use a fine grid for the vegetation equations and a coarse grid for the morphological and hydrodynamic parts. This approach will result in a lower computation cost compared to solving all processes on a fine grid. A disadvantage however is that this multi-space-scale approach will increase the complexity of the implementation.

7.2. Test Problem

We could use the area shown in Figure 7.2b as a benchmark for our model. This area is situated in the northeast of the ”Verdronken Land van Saeftinghe”, which is shown in Figure 7.2a. The ”Verdronken Land van Saeftinghe” is located at the border between the Netherlands and Belgium at the left bank of the ”Westerschelde”. In total it contains 3580 hectare of salt marshes.



(a) Top view ”Verdronken Land van Saeftinghe”. (b) Zoomed in on test area.

Figure 7.2: Test area in ”Verdronken Land van Saeftinghe”.

Data

We have data for the M2 tide, the principal lunar semi-diurnal tide, at Bath. Bath is located at the right bank of the ”Westerschelde” (illustrated in Figure 7.3). The data is based on measurements between 1901 and 2008. The water surface elevation at Bath is given by

$$a(t) = a_0(t) + A_{M2}(t)\sin\left(\frac{2\pi t}{T_{M2}}\right) \tag{7.1}$$

in which the first term is given by

$$a_0(t) = 0.14\ m + a_1 t \tag{7.2}$$

and the sea level rise (a_1) is 2.5 $mm/year$. The tidal range is given by

$$A_{M2}(t) = 2.27\ m + a_{M2} t \tag{7.3}$$

and the increase of tidal range (a_{M2}) is 2.1 $mm/year$. So this formula takes into account the sea level rise and the increase of tidal range. If we do not want his, we can just set a_1 and a_{M2} equal to zero. But it does not take into account any spring and neap tides. The period, T_{M2}, is equal to 12 hours and 25 minutes. For the water elevation the NAP (Amsterdam Ordnance Datum) is used, which is a vertical datum in use in large parts of Western Europe.

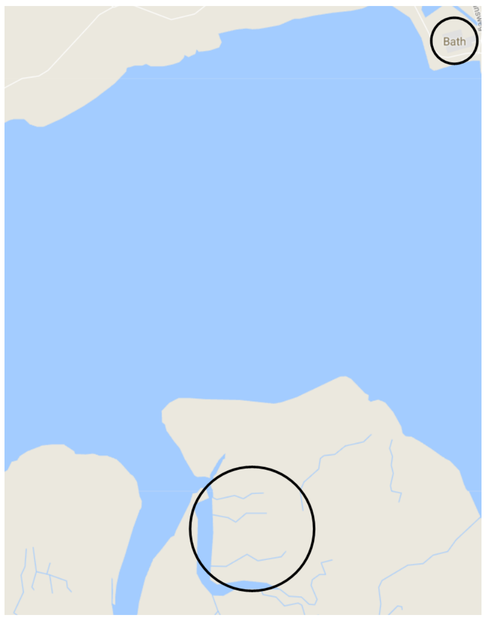


Figure 7.3: Location of Bath relative to our test area.

A
Important terms

Stress A physical quantity that expresses the internal forces that neighboring particles of a continuous material exert on each other. The normal stress is perpendicular to the surface, and the shear stress, often denoted τ, that is parallel to the surface.

Tensor Geometric objects that describe linear relations between geometric vectors, scalars, and other tensors.

Traction Force acting over any unit of area.

Stress tensor The Cauchy stress tensor consists of nine components σ_{ij} that completely define the state of stress at a point inside a material in the deformed state, placement, or configuration. The tensor relates a unit-length direction vector n to the stress vector $T(n)$ across an imaginary surface perpendicular to n (thus expressing a relationship between these two vectors):

$$\mathbf{T}^{(\mathbf{n})} = \mathbf{n} \cdot \sigma = \sigma \cdot \mathbf{n} \quad (\sigma_{ij} = \sigma_{ji}) \quad \text{or} \quad T_j^{(n)} = \sigma_{ij} n_i \tag{A.1}$$

where

$$\sigma = \left[T^{(e_1)} \quad T^{(e_2)} \quad T^{(e_3)} \right] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \equiv \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \tag{A.2}$$

where σ_{11}, σ_{22} and σ_{33} are normal stresses, and $\sigma_{32}, \sigma_{13}, \sigma_{21}, \sigma_{23}, \sigma_{13}$ and σ_{32} are the shear stresses.

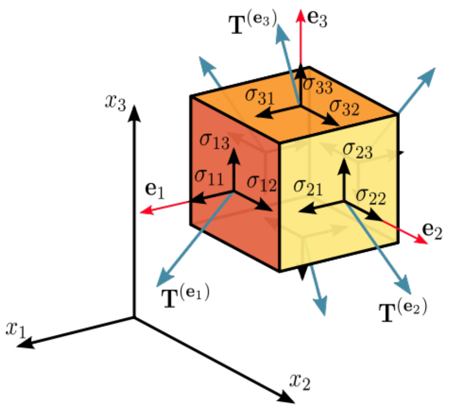


Figure A.1: Components of stress in three dimensions.

Strain rate The change in strain (deformation) of a material with respect to time.

Bibliography

[1] S. C. Medeiros and S. C. Hagen, *Review of wetting and drying algorithms for numerical tidal flow models,* International journal for numerical methods in fluids **71**, 473 (2013).

[2] M. Baptist, V. Babovic, J. R. Uthurburu, M. Keijzer, R. Uittenbogaard, A. Mynett, and A. Verwey, *On inducing equations for vegetation resistance,* Journal of Hydraulic Research **45**, 435 (2007).

[3] P. Wesseling, *Principles of computational fluid dynamics*, Vol. 29 (Springer Science & Business Media, 2009).

[4] C. Vuik and D. Lahaye, *Scientific computing (wi4201),* Lecture notes for wi4201 (2012).

[5] C. Vuik and C. Lemmens, *Programming on the GPU with CUDA* (Springer Science & Business Media, 2015).

[6] Salt marsh, (2017), https://en.wikipedia.org/wiki/Salt_marsh [Accessed: 24-10-2017].

[7] L. C. Van Rijn, *Principles of fluid flow and surface waves in rivers, estuaries, seas and oceans*, Vol. 11 (Aqua Publications Amsterdam, The Netherlands, 1990).

[8] Underdetermined system, (2017), https://en.wikipedia.org/wiki/Underdetermined_system [Accessed: 24-10-2017].

[9] J. H. Ferziger and M. Peric, *Computational methods for fluid dynamics* (Springer Science & Business Media, 2012).

[10] W.-Y. Tan, *Shallow water hydrodynamics: Mathematical theory and numerical solution for a two-dimensional system of shallow-water equations*, Vol. 55 (Elsevier, 1992).

[11] C. B. Vreugdenhil, *Numerical methods for shallow-water flow*, Vol. 13 (Springer Science & Business Media, 2013).

[12] D. Hydraulics, *Delft3d-flow user manual,* Delft, the Netherlands (2006).

[13] O. Gourgue, R. Comblen, J. Lambrechts, T. Kärnä, V. Legat, and E. Deleersnijder, *A flux-limiting wetting–drying method for finite-element shallow-water models, with application to the scheldt estuary,* Advances in Water Resources **32**, 1726 (2009).

[14] T. Kärnä, B. De Brye, O. Gourgue, J. Lambrechts, R. Comblen, V. Legat, and E. Deleersnijder, *A fully implicit wetting–drying method for dg-fem shallow water models, with an application to the scheldt estuary,* Computer Methods in Applied Mechanics and Engineering **200**, 509 (2011).

[15] RANS-based turbulence models, (2009), https://www.cfd-online.com/Wiki/RANS-based_turbulence_models [Accessed: 24-10-2017].

[16] A. Veldman, *Computational fluid dynamics,* Lecture Notes, University of Groningen, The Netherlands (2001).

[17] D. Hydraulics, *Delft3d-tide user manual,* Delft, the Netherlands (2014).

[18] Range variation: springs and neaps, (2017), https://en.wikipedia.org/wiki/Tide [Accessed: 6-11-2017].

[19] S. Temmerman, T. Bouma, J. Van de Koppel, D. Van der Wal, M. De Vries, and P. Herman, *Vegetation causes channel erosion in a tidal landscape,* Geology **35**, 631 (2007).

[20] S. Fagherazzi, M. Hannion, and P. D'Odorico, *Geomorphic structure of tidal hydrodynamics in salt marsh creeks,* Water resources research **44** (2008).

[21] Sea level: Dutch coast and worldwide, 1890-2014, (2016), http://www.clo.nl/en/indicators/en0229-sea-level-dutch-coast-and-worldwide [Accessed: 8-11-2017].

[22] Sea level: Dutch coast and worldwide, 1890-2014, (2017), https://nl.wikipedia.org/wiki/Getijde_(waterbeweging) [Accessed: 8-11-2017].

[23] M. G. Kleinhans, F. Schuurman, W. Bakx, and H. Markies, *Meandering channel dynamics in highly cohesive sediment on an intertidal mud flat in the westerschelde estuary, the netherlands,* Geomorphology **105**, 261 (2009).

[24] A. D'Alpaos, S. Lanzoni, M. Marani, and A. Rinaldo, *Landscape evolution in tidal embayments: modeling the interplay of erosion, sedimentation, and vegetation dynamics,* Journal of Geophysical Research: Earth Surface **112** (2007).

[25] M. G. Kleinhans, F. Schuurman, W. Bakx, and H. Markies, *Meandering channel dynamics in highly cohesive sediment on an intertidal mud flat in the westerschelde estuary, the netherlands,* Geomorphology **105**, 261 (2009).

[26] L. C. Van Rijn *et al.*, *Principles of sediment transport in rivers, estuaries and coastal seas*, Vol. 1006 (Aqua publications Amsterdam, 1993).

[27] S. Fagherazzi, M. L. Kirwan, S. M. Mudd, G. R. Guntenspergen, S. Temmerman, A. D'Alpaos, J. Koppel, J. M. Rybczyk, E. Reyes, C. Craft, *et al.*, *Numerical models of salt marsh evolution: Ecological, geomorphic, and climatic factors,* Reviews of Geophysics **50** (2012).

[28] M. L. Kirwan and A. B. Murray, *A coupled geomorphic and ecological model of tidal marsh evolution,* Proceedings of the National Academy of Sciences **104**, 6118 (2007).

[29] M. W. Straatsma and M. Baptist, *Floodplain roughness parameterization using airborne laser scanning and spectral remote sensing,* Remote Sensing of Environment **112**, 1062 (2008).

[30] J. van Kan, A. Segal, and F. Vermolen, *Numerical methods in scientific computing* (VSSD, 2005).

[31] Performance and Correctness of GPGPU Applications, (2017), http://fmt.ewi.utwente.nl/Workshops/NIRICT_GPGPU/ [Accessed: 24-10-2017].

[32] D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, *Heterogeneous Computing with OpenCL 2.0* (Morgan Kaufmann, 2015).

[33] Thread and block heuristics in cuda programming , http://cuda-programming.blogspot.nl/2013/01/thread-and-block-heuristics-in-cuda.html [Accessed: 24-10-2017].