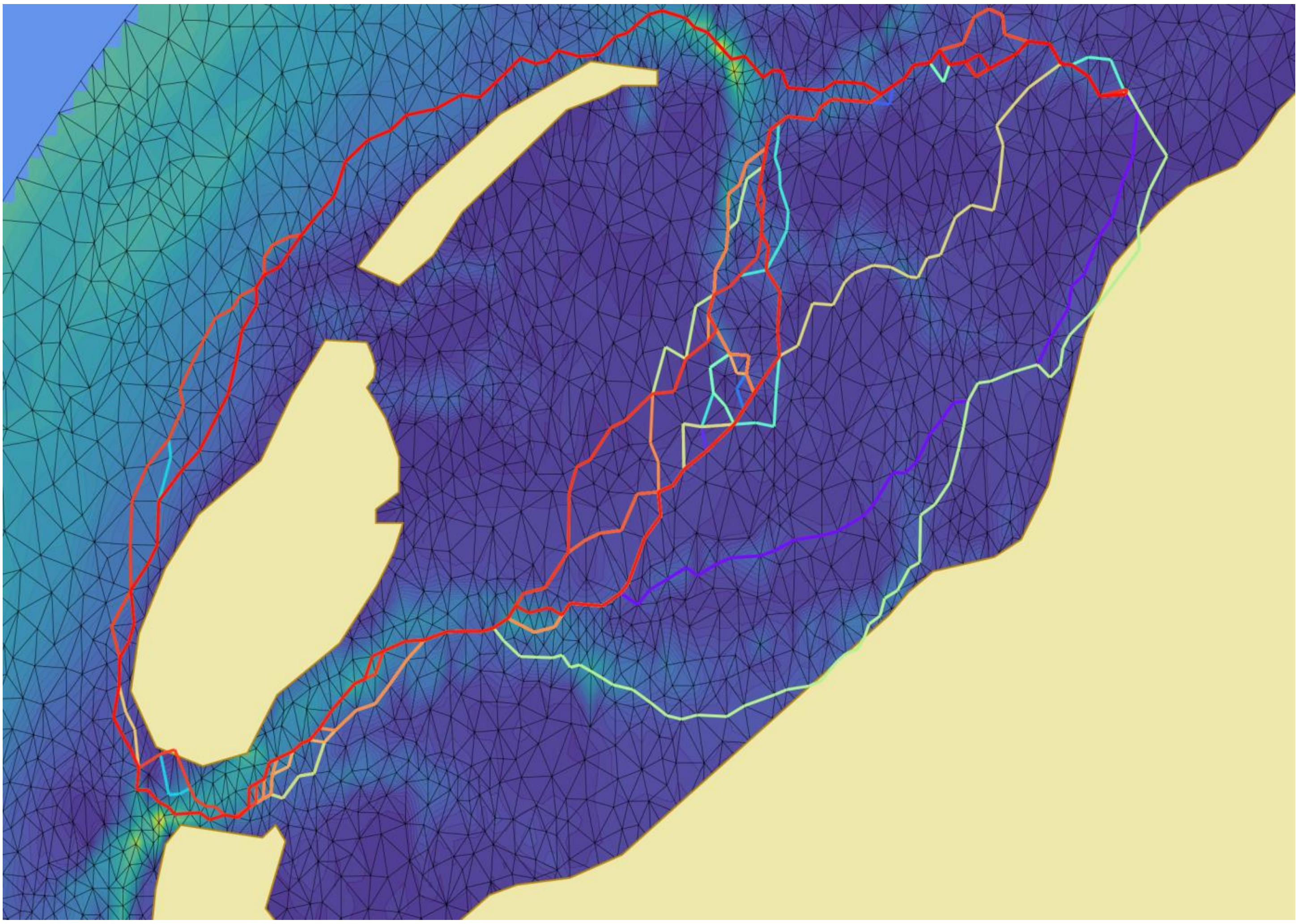
Route optimization in dynamic currents

Navigation system for the North Sea and Wadden Sea

J.P. van Halem
Delft University of Technology
September 25, 2019



Van Oord Marine ingenuity TU Delft Deltares Enabling Delta Life

The Zen of Python

"Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one– and preferably only one –obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those! "

Peters (2014)

Route optimization in dynamic flow fields

Navigation system for the North Sea and Wadden Sea

by

J.P. van Halem

To obtain the degree of Master of Science at the Delft University of Technology, to be defended publicly on
Thursday October 3, 2019 at 13:00.

Student number:	4597591	
e-mail:	PietervanHalem@hotmail.com	
Project duration:	February 11, 2019 – Oktober 3, 2019	
Chairman:	Prof. dr. ir. J.D. Pietrzak	TU Delft
Thesis committee:	Dr. ir. G. J. de Boer,	van Oord, Supervisor
	Prof. dr. ir. M. van Koningsveld,	TU Delft, van Oord
	Prof. dr. ir. M. Verlaan,	TU Delft, Deltares
	Dr. ir. F. Zijl,	Deltares
	Dr. F. Baart,	Deltares

An electronic version of this thesis is available at: `http://repository.tudelft.nl/`.

Faculty of Civil Engineering and Geosciences
Delft University of Technology

The MIT License (MIT)

Copyright (c) 2019 Pieter van Halem
Permission is hereby granted, free of charge, to any person obtaining a copy of the HALEM software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This study is preformed with HALEM version 0.3.0 and OpenCLSim version 1.0.0.

Acknowledgement

This Thesis describes the Graduation research that is done as the finalising part of the Master of Science program Hydraulic Engineering at the Faculty of Civil Engineering and Geosciences, Delft University of Technology, The Netherlands. This study was done in combination with an internship at van Oord, who also funded this research. The research is conducted at the office of Van Oord in Rotterdam. This research is done in close cooperation with Deltares. Deltares provided the hydrodynamic models that are used in this study. The last two months of this research Deltares provided a workplace to publish the results of this research in a paper. This unique cooperation between the Delft University of Technology, Van Oord and Deltares has resulted in a new route optimisation tool that contributes to the global research in route optimisation.

I want to thank my graduation-committee for their supervision during the project. My expressions of gratitude go especially to Gerben de Boer, Joris den Uijl, Fedor Baart, Julie Pietrzak, Firmijn Zijl, Martin Verlaan and Mark van Koningsveld. I want to thank them for their continuous support, feedback, expert judgement, and knowledge that made this research possible. I also would like to thank Van Oord for facilitating a place to work in a community with other students.

Finally, I would like to thank my parents an girlfriend for their continuous support and report reviews. Also special thanks to my friends and fellow Van Oord graduates for taking time to discuss my project and to relax from time to time.

J.P. van Halem
Delft, September 25, 2019

Abstract

This thesis introduces a new algorithm for optimising shipping routes within a dredging project. Highly dynamic and time-dependent hydrodynamic features influence shipping routes. Due to the complex interactions between the horizontal tide, vertical tide, stratifying forces, wind-driven forces, and limited water-depth, shipping routes were previously only optimised for large scale routes (order of 1000 km). This study presents an algorithm that can optimise shipping routes that are influenced by these small scales (order of 10 km) hydrodynamic features. This algorithm uses graph theory to solve for the time-dependent fastest path between start and destination. Graph theory searches for the optimal path through a set of nodes that are connected with edges. This study uses the time-dependent shortest path algorithm which accounts for the FIFO-criteria (Waiting criteria) and can solve the non-convex nature of the problem.

The input of this algorithm is a hydrodynamic model. These models are Computational Fluid Dynamic (CFD) models that calculate currents and water levels in a specific domain. The domain is discretised into cells and nodes to calculate these hydrodynamic features. This study uses the nodes of this hydrodynamic model as the vertices of the graph. However, for some cases, the hydrodynamic model has too many nodes for the shortest path algorithm. This study presents a method for reducing the number of nodes without reducing the spatial resolution. The nodes are reduced based on a combination of the vorticity and the magnitude of the flow.

This algorithm is implemented in a python software package named Hydrodynamic Algorithm for Logistic enhancement Module (HALEM). HALEM can determine the optimal shipping route for a given hydrodynamic model. Defining different cost functions results in different optimisation purposes. This thesis presents cost functions for the fastest route, shortest route, cheapest route and least polluting route. This software is then implemented in the OpenCLSim software so that this combination of software can optimise routes of entire projects. A case study simulates a beach-nourishment at Schouwen Westkop Noord to demonstrate the practical use of HALEM and OpenCLSim. For this project, 425,500 m^3 sand should be dredged offshore and pumped onto the beach. Due to the narrow gullies and tidal changes in hydrodynamic features, the routes were hard to predict. The simulation with HALEM and OpenCLSIM shows an increase in the production with 21 % compared to the simulation with just OpenCLSim.

Contents

Acknowledgement iii

Abstract v

1 Introduction 1

- 1.1 Developing a new model to optimise shipping routes . . . 2
- 1.2 Cooperation between OpenCLSim and HALEM . . . 3
- 1.3 Scope of the research . . . 4
- 1.4 Report overview . . . 4

2 Literature Study 5

- 2.1 Dredging processes . . . 5
- 2.2 Currents in the North,- and Wadden sea . . . 6
  - 2.2.1 Tidal currents . . . 6
  - 2.2.2 Wind driven currents . . . 7
  - 2.2.3 Density driven currents . . . 8
  - 2.2.4 Depth profiles of the currents . . . 10
- 2.3 Hydrodynamic models . . . 11
  - 2.3.1 Model properties . . . 11
  - 2.3.2 Models used in this study . . . 12
- 2.4 Ship movement study . . . 14
  - 2.4.1 Inertial models . . . 14
  - 2.4.2 Vector addition models . . . 14
  - 2.4.3 Squat . . . 15
- 2.5 Optimizing with graph theory . . . 17
  - 2.5.1 Variations of the shortest path problem . . . 17
  - 2.5.2 Distinctions between graphs . . . 17
  - 2.5.3 Single Source Shortest Path Problem (SSSPP) algorithms . . . 17
  - 2.5.4 Example of Dijkstra's algorithm . . . 18

3 Methods 21

- 3.1 Methods for model development . . . 21
  - 3.1.1 Graph strategy . . . 21
  - 3.1.2 Time dependent shortest path problem . . . 24
  - 3.1.3 Cost functions and variable shipping speed . . . 25
  - 3.1.4 Modular package implementation . . . 27
- 3.2 Methods for model application . . . 28
  - 3.2.1 Implementation of HALEM in OpenCLSim . . . 28
  - 3.2.2 Generalisation of the flow model: Tidal analysis . . . 28
  - 3.2.3 case study Schouwen Westkop Noord . . . 30

4 Model validation 33

- 4.1 Methods for model validation . . . 33
  - 4.1.1 Model validation with trivial test cases . . . 33
  - 4.1.2 Model validation with a real-life test case . . . 33
- 4.2 Results of the model validation . . . 35
  - 4.2.1 Model validation with trivial test cases . . . 35
  - 4.2.2 Model validation with a real test case . . . 36
- 4.3 Discussion of model validation . . . 37
  - 4.3.1 Discussion of model validation with trivial test cases . . . 37
  - 4.3.2 Discussion of model validation with real-life test case . . . 38

5 Results **41**
5.1 Results of the model development . . . 41
5.1.1 Roadmap generator layout . . . 42
5.1.2 Route optimizer layout . . . 42
5.1.3 Results of the node reduction . . . 42
5.2 Results of model application . . . 44

6 Discussion of the Results **49**
6.1 Features of HALEM . . . 49
6.2 Limitations of HALEM . . . 50
6.3 HALEM compared to the literature . . . 51

7 Conclusion and recommendations **53**
7.1 Recommendations . . . 54

A Nomenclature **55**
A.1 Latin Symbols . . . 55
A.2 Greek Symbols . . . 56
A.3 Units . . . 56
A.4 Abbreviations . . . 56

B Tidal analysis **57**

C Results of the validation case Texel for current compensation **59**

Bibliography **63**

List of Figures **68**

List of Tables **69**

1 | Introduction

Currently, the soaring fuel prices, environmental regulations, increasing computational power, and the drive to optimise projects leads to many researches that aim to optimise project logistics. These projects can vary from container transfer hub design to minimising the pollution of transatlantic shipping, or tracking the spill of dredging projects. For these projects, complexity can increase rapidly. What all these complex projects have in common is that they create a demand for optimisation tools that can handle the complex behaviour of the projects. A recent innovation is to simulate these complex projects with a digital twin. The digital twin first performs the project in a digital environment before the project is executed in real life (Boschert and Rosen, 2016). When the digital twin first preforms the project in a digital environment, the digital twin finds the opportunities and disadvantages of the working method more easily (Ensing and Knijff, 2019, Glaessgen and Stargel, 2012, Tuegel et al., 2011). Den Uijl (2017) present a new manner of finding the optimal working method. OpenCLSim contains the results of this study in the form of a python tool. OpenCLSim uses digital twins as an optimisation tool for complex logistic processes. Optimisation studies and further research in the behaviour of complex logistics projects can use OpenCLSim as a basis. For example, van der Bilt (2019) optimises dredging projects with OpenCLSim for emissions of vessels. Showing that dredging projects can use OpenCLSim as well. This tool produces more reliable estimations for projects and helps to find an optimum work method. The OpenCLSim tool improves the current estimation method by accounting for each event in a logistic process. This new method results in a more accurate estimation for logistic processes, and gives an excellent platform for optimsing these logistic processes. One of the optimisations that is possible with this model is shipping route optimisation. Shipping routes of dredging vessel are sensitive for route optimisation since they are influenced by dynamic currents and water depth.

Globally, shipping route optimisation is of significant importance for earning money, saving fuel and reducing emissions. Many research efforts have aimed to optimise shipping routes for both commercial purposes (Kobayashi et al., 2011, Mamanduru et al., 2016, Montes, 2005) and recreational purposes (Corno et al., 2016). Competitive races such as the Volvo Ocean Race and the American cup lead to the development of models that use the influence of winds, waves, and currents to gain an advantage over their competitors. For the Volvo-Ocean race, the AkzoNobel team used the output of a wind, current and wave model to optimise their routes (Deltares, 2017). In part thanks to this route optimisation the AkzoNobel team broke a ten-year record in the ninth stage, from the American city of Newport to Cardiff, the capital of Wales (Blake, 2018). This competitive drive for innovation is also present in commercial processes. For example, in Park and Kim (2015) the transatlantic routes were optimised for fuel consumption. Optimising these shipping routes leads to a decrease in costs, less pollution and an opportunity for companies to earn money.

The literature describes three methods for optimising the shipping routes. (1) the hill-climbing algorithm, (2) the annealing method, and (3) graph theory. These three methods have different assumptions and different uses. The first is the hill-climbing algorithm. The hill-climbing algorithm is a simple but very effective optimisation technique in numerical analysis. This algorithm belongs to the family of local search and is an iterative algorithm that begins with a random solution, and pursuits to improve the route from that initial solution. This technique is applied to many different problems and is also suitable for route optimisation. For route optimisation, the algorithm takes an initial route and a cost function as a function of position (x,y) as input. The algorithm then searches if there are cheaper neighbours than the current nodes. If there are cheaper neighbours than the current nodes, the route is updated. Iteratively, this results in an optimal path. The second methods is the Late Acceptance Hill Climbing (LAHC) which was first introduced by Burke and Bykov (2012) who implements this method. An advantage of this solution is that the continuous solution space does not need to be discretised into nodes. This feature is an advantage since it takes small scale properties (such as narrow gullies, or small islands) into account in the solution. A disadvantage of this method is that the hill-climbing algorithm finds optimal solutions for convex problems, which means that the algorithm considers only a solution when the solution does not have local minimums. When the solution has local minimums, the algorithm can get stuck in such a local minimum. Another disadvantage is that the answer depends on an initial solution and iterates based on this initial solution. This feature means that the method is a problem specific algorithm. This research aims to keep the method as generic as possible; the problem-specific property of these algorithms is, therefore, a substantial disadvantage.

A variation of the hill-climbing algorithm is the annealing method. The annealing method describes the optimal route between two points in a continuous space as a ’N-1’th order polynomial that can be characterised by N parameters. Finding the optimal value for these parameters resolves in the optimal route. Kobayashi et al. (2011), Kosmas and Vlachos (2012) show examples of the implementations of the annealing method for route optimisations. The annealing method is very similar to the hill-climbing algorithm. The main difference between the hill-climbing algorithm and the annealing algorithm is that the annealing assumes that the optimal route is an N-th order curve while for the hill-climbing algorithm the route can be anything. This similarity results in the same advantages for the annealing method as for the hill-climbing algorithm.

The last optimisation method is graph theory. Graph theory uses graphs that contains edges and vertices (nodes). The edges connect all the vertices. By doing so, the optimal route is a collection of these edges. Calculating the optimal route through nodes is a well-known problem in mathematics called the shortest path problem (or single-pair shortest path problem); this problem is a subsection of graph theory. This method has the advantage that it can solve non-convex problems, which means that the algorithm can not get stuck in a local minimum. Another advantage of graph theory is that the algorithm is not problem-specific (it does not need an initial route). For the hill-climbing algorithm and the annealing methods an initial route must be given. The fact that the shortest path algorithms are problem independent is a significant advantage since it makes the method a more general application. A disadvantage of graph theory is that it discretises the continuous solution space into nodes which results in a loss of spatial resolution. Another disadvantage of the graph theory is that the computational time can increase rapidly for ample solution space. The last disadvantage of graph theory compared to hill-climbing and the annealing method is the directional resolution. By discretising the continuous solution space into nodes and edges the possible directions of the route are limited. This disadvantages can result in a stair-like pattern in the solution.

The main objective of this research is captured in the main question. This main question is elaborated in four research questions. These follow in the next sections. The main question is:

> *How can the optimal shipping route be determined for given currents in the North Sea and Wadden sea?*

1.1. Developing a new model to optimise shipping routes

This study aims to find an algorithm that can work together with OpenCLSim to optimise the shipping parts of complex logistics projects. Developing a route optimisation tool and implement it in OpenCLSim expands the work of Den Uijl (2017), van der Bilt (2019), Van Koningsveld et al. (2019). This study investigates the influence of flow velocity and water depth on the shipping routes. Due to the impacts of these hydrodynamic features, the optimal shipping route changes with time and the optimal route is no longer the trivial route. This problem gives rise to a need for a tool that can predict the optimal shipping route for a given hydrodynamic model. The optimisation of the shipping routes is limited to the sailing within the dredging projects. This limitation means that the length of the route is typically less than 50 km. This study does not cover the optimisations of large scale routes. Other studies have already optimised these large scale route optimisations. Software packages are already available for these optimisations. This is, however, not available for shorter routes. For these short routes, the small scale details of the hydrodynamic features become essential. This problem gives rise to challenges in the optimisation process, such as computational power constraints and time-dependent solutions.

In this study graph theory is used to find the optimal route through a continuous solution space. The main reason for choosing this is that it cannot get stuck in local minima. Techniques such as the hill-climbing algorithm or the annealing method are known to depend on the initial conditions. For a different initial route, the algorithm can return a completely different solution. The algorithm returns a local minimum instead of a global minimum. This difference between the local minima and the global minima is crucial for highly dynamic problems such as flooding and drying of banks, or finding the route around islands. Another reason for choosing graph theory is that it does not need an initial solution. This property makes the resulting method more generally applicable. One of the reasons why graph theory would not be a good fit is because of the discretisation of the continuous solution space into nodes. Discretising the continuous solution space into nodes could neglect small scale properties (such as narrow gullies, or small islands). This problem is a real disadvantage of the model that Chapter 6 discusses. However, the model uses a hydrodynamic model which discretises the solution space into nodes and edges. When using the nodes of this hydrodynamic model, none of the information from the hydrodynamic model is lost. A significant advantage of this is that it creates no fake resolution by interpolating. For all the reasons explained above the shortest path algorithm is chosen over the annealing algorithms or the hill-climbing algorithms.

Graph theory discretises the continuous solution space in a graph. This graph contains nodes (vertices) and edges (arcs). With a cost function, the weight of each of the edges is determent. The shortest path is in graph theory defined as the collection of edges, that lead from start to destination, with the lowest sum of weights of the edges. Graph theory uses, for example, Dijkstra's shortest path algorithm to find the shortest path through the graph. If there are n times more vertices, the Dijkstra algorithm will get n^2 times slower (Orlin et al., 2010). The result of this is that computational power constraints limit the route optimisation. The first challenge of this research is to overcome these limitations. This research presents a new method for reducing the number of nodes of the graph without decreasing the spatial resolution of the graph. The first research question captures the research for reducing the number of nodes without reducing the spatial resolution. Research question 1 is:

What is the best graph strategy given the computational power constraints?

The second challenge of this research is the time-dependent nature of the optimal shipping route. This research aims to find the optimal shipping route for given currents. These currents are not constant in time. If a shipping route takes three hours, the currents can completely change their direction and magnitude in this time frame. These temporal changing currents need to be taken into account to create a correct prediction for the shipping routes. A significant issue that is induced by time-dependent route optimisation is waiting. When the hydrodynamic conditions change over time, it can be better to wait until the unfavourable conditions are passed and then to sail. This research presents an algorithm that can take these time-dependent currents and FIFO problem into account when solving for the shortest path. When currents change with time and waiting for tides is sometimes better than directly sailing, the optimal sailing velocity should not be constant. When the sailing velocity of the vessel cannot be constant, the definition of a cost function becomes non-trivial. By changing the cost function, the optimisation goal changes as well. This study presents a method for making calculations with dynamic vessel speed and presents four cost functions for the fastest route, shortest route, cheapest route, and least polluting route. The second research question captures the research for this time-dependent algorithm. Research question 2 is:

What is the best method for time-dependent route optimisations?

1.2. Cooperation between OpenCLSim and HALEM

The goal of this research is to optimise the shipping routes of complex logistics projects in OpenCLSim. The previous section introduces route optimisation. To make sure this research can be reproduced and used by third parties, this study uses the OpenEarth philosophy, after Van Koningsveld et al. (2010). This philosophy is used by many other scientists to ensure that their research is reproducible (Shen, 2014, Wilkinson et al., 2016). The ability to reproduce is done by developing a modular python tool. This tool should be able to optimise the route for a given start location, stop location, departure time, and hydrodynamic model. This python tool will be packaged and published. The third research question captures the study in publishing this package and making it modular. Research question 3 is:

How can the route planner be implemented in a modular package so that it can be used in different cases?

The python package needs to be implemented in OpenCLSim to optimise the shipping routes of complex logistics projects in OpenCLSim. This study presents a mix-in for the python package in OpenCLSim. The optimal route that is returned by the python package is dependent on the properties of the vessel. The most important parameters of which the python package depends are sailing velocity and draft. However, these parameters are not constant in a project. The difference in the sailing velocity of an empty ship and a fully loaded ship can vary with 2 knots. The difference in draft between a fully loaded ship and an empty ship can be up to 4 meters. The load factor of the ship can determine the draft and sailing velocity of the vessel ($r = \frac{W_{ship} - W_{empty}}{W_{full} - W_{empty}}$, in which W is the weight of the vessel, the subscripts ship, empty, and full mean respectively weight of the vessel, weight of the vessel when empty and weight of the vessel when full). This property means that the optimal route is dependent on the load factor of the vessel. In this study, a method to predict the optimal load factor per dredging cycle is presented. This method predicts the optimal load factor of the vessel in OpenCLSim combined with the python package. OpenCLSim and the python package optimises a project for the route and load factor. For this study, a reliable hydrodynamic model is of vital importance. Therefore, this study compares different hydrodynamic models, and accesses the influence of the hydrodynamic model. Research question 4 describes the research for the load factor optimisation and the optimisation of a complete project. Research question 4 is:

> *How can dredging projects be optimised for the given route optimisation package and OpenCLSim?*

1.3. Scope of the research

Some assumptions are made in this study to model the chaotic nature of reality. This section describes the main assumptions for this study. The first and foremost assumption is that inertial forces on the ship and inertial effects of the ship can be neglected. Inertial models are very accurate; however, they are also very computationally intensive. Since the inertial effects are only significant in the start and end phase of the route, the benefits of inertial models are minimal. In section 2.4, the model that is used to predict the ship movement is explained in more detail.

The second assumption is that currents only influence the ship movement. In reality, the ship movement is also influenced by wind and wave forcing. However, this study neglects these forces because it reduces the required memory for the calculations significantly. Including the wind and wave forcing should be a relatively simple step, it would, however, require a lot more memory (both RAM and storage memory). The third assumption in this study is that the hydrodynamic model is correct. In reality, the hydrodynamic models are not perfect (see chapter 6), and this can influence the outcome of the model. This assumption has particular substantial implications for the bathymetry. The bathymetry in tidal inlets in highly dynamic, which makes it hard to capture. This study assumes that the bathymetry is accurate.

The last influential assumption is that the route is a freely routable route. This assumption means that the route is allowed to deviate from a particular path. An example of a not free-routable route is sailing through a channel or being obliged to navigate through buoys. For these examples, the route optimisation tool is not valid and not useful.

1.4. Report overview

This report consists of 7 chapters. The next chapter describes the current literature and background of the problem. This chapter contains all the information for the development of the model. The third chapter presents the model. This chapter presents the methods for the model development and model application. The fourth chapter presents the model validation that of HALEM. This chapter shows both the methods and results of the model development. The fifth chapter presents the result that follows from the HALEM packages and the model applications. The sixth chapter discusses the results and shows the place of this study compared to the literature. The final chapter concludes this report. This section gives the conclusions and recommendations.

2 | Literature Study

A view of the existing literature is needed to answer the main and research question. This chapter explains the existing literature in five sections. The first section gives a broad overview of the dredging process. In the second section, a general overview of the currents in the North,- and Wadden sea is given. A numerical model is used to obtain data about these current. The third chapter describes the available models. The third section continues with how these currents influence the movement of the vessel. In this section, the model that is used to explain the ship movement is elaborated. This chapter finishes with the different route optimisation methods that are available. The last section elaborates on the method that is used in this study, which is graph theory. Each section can be read on its own; however, together, all the sections give a complete view of the existing literature on this subject.

2.1. Dredging processes

In this study route optimisation is applied for dredging projects. Dredging works are often schematised if four parts. These four parts are: 1. Dredging (or loading), 2. Sailing filled, 3. Placement (or unloading), 4. Sailing empty. fig. 2.1 shows these steps in a visual manner. In general, these four steps are repeated until the desired amount of material is moved.

This study tries to optimise dredging projects by optimising the shipping parts. This shipping parts are executed in step two and four of the dredging cycle. Optimising the shipping parts of dredging projects influences the loading and unloading parts of the dredging cycle as well. Loading a Trailing Suction Hopper Dredger (TSHD) is in general a non-linear process. This means that the hopper fills faster when it is empty than when it is almost full. filling the hopper for 100 % takes in general infinite time. Therefore, hoppers are in general not filled for 100 %. The filling of a hopper is indicated with the load factor ($r = \frac{W - W_{empty}}{W_{full} - W_{empty}}$, where r is the load factor and W is the weight of the vessel). Depending on the sailing time, loading curve, unloading cure, material properties and other parameters this load factor can be optimised. This means that when the sailing times are optimised the optimal load factor changes.

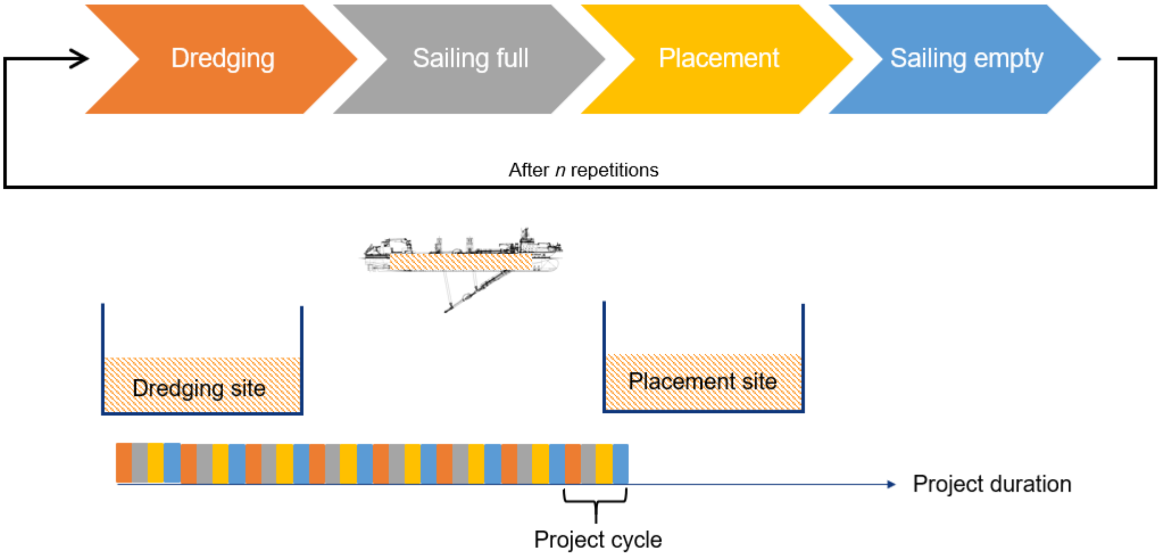


Figure 2.1: Schematisation of a dredging cycle

2.2. Currents in the North,- and Wadden sea

To asses, the influence of the currents on the optimal shipping route, knowledge about these currents is needed. This section presents a general overview of the currents in the North,- and the Wadden Sea. The currents that influence the shipping routes are large scale currents that often have a clear pattern. In this study, the currents are separated into three classes. The distinction between these classes is made based on the nature of the forcing of these currents. This results in the following classifications — first the tidal currents, second the wind-driven currents and lastly the density-driven currents. The last part of this chapter covers the three-dimensional nature of the currents. Here the distribution of the flow velocities over depth is elaborated. By distinguishing currents into three classes, a clear overview of the currents is provided, which provides a better understanding of the influence of the currents on the shipping route.

2.2.1. Tidal currents

The first class of currents are the currents that are driven by the tides. These tides are a well-known phenomenon that occurs on a very regular basis. The tides are so regular because the gravitational dance between celestial bodies drive them. The tides are generated by the forces of the moon and the sun. The difference in inertial forces of the orbital path and the gravitational forces of the celestial bodies results in a force exerted on the water called differential pull. This differential-pull result in a deformation of the water masses on the earth surface. The earth rotates underneath this deformation resulting in a tidal wave from the earth's reference frame.

The tidal signal that can be measured on earth can be decomposed in several components. According to the Fourier series, any periodic function can be decomposed into a summation of sinusoidal functions (see equation 2.1). When this is done for the tidal signal, the different components are called tidal constituents and are indicated with a letter and a number (for example M1, S2, 2MK5). The semi-diurnal components are indicated with a 2. The diurnal components are indicated with a 1. The long period components do not have a number. Each tidal constituent has its physical meaning. For example, the M stands for the moon, S for sun, and so on. This results in, for example, the M2 tide, which is semi-diurnal tidal constituent forced by the moon. The Fourier analysis of the tide results into several tidal constituents (n until N in equation 2.1) each with its own unique amplitude (a_n in equation 2.1) and phase(α_n in equation 2.1). An example of a tidal signal and the tidal constituents that follow from the tidal analysis is shown in appendix B. The most important tidal constituents and their physical meaning are explained below. Equation 2.1 shows the mathematical function for the tide. In this equation $\eta(t)$ is the measured water level, a_0 is the mean water level, a_n is the amplitude of the tidal component, ω_n is the angular velocity of the tide, and α_n is the phase difference of the tidal constituent.

$$\eta(t) = a_0 + \sum_{n=1}^{N} a_n cos(\omega_n t - \alpha_n) \tag{2.1}$$

An essential tidal constituent is a tidal wave that results from the forcing of the moon. This tidal constituent is called M2 tide. The higher harmonics of this wave are called M4, M6, et cetera. The forcing of the moon is, for an equilibrium tide, responsible for 69% of the tidal mechanism (Bosboom and Stive, 2009). This tidal constituent has a period of 12.42 hours. The forcing of the sun causes, for an equilibrium tide, the rest of the tidal mechanism. This tidal constituent is called S2 tide. The higher harmonics of this wave are called S4, S6, et cetera. This tidal constituent has a tidal period of 12.00 hours. The small difference between the M2 and the S2 tide result in a beating of the signal. This is known as the spring and neap tide variation.

Another tidal mechanism is the daily inequality (lunar-solar declination). This tidal constituent is called the K1 tide. In contrast with the M2 and the S2 tide, this component is not the result of a force. The sun and the moon rotate in (approximately) the same plane. However, the axis of the earth's rotation is not perpendicular to this plane. The angle of this difference is called the earth's declination. This angle reaches its minimum of -23.5 degrees on December 22 (Northern winter solstice), and its maximum of 23.5 degrees on June 21 (Northern summer solstice). When the earth rotates underneath the deformation of water with this declination, this results in a daily inequality. This tidal component has a period of 23.93 hours. Other diurnal components are O1 tide, which is the principal lunar tide, the P1 tide, which is the principal solar tide, and the Q1 tide, which is lunar elliptical tide. Other long period components are the Fortnightly components (Mf), the Monthly com-

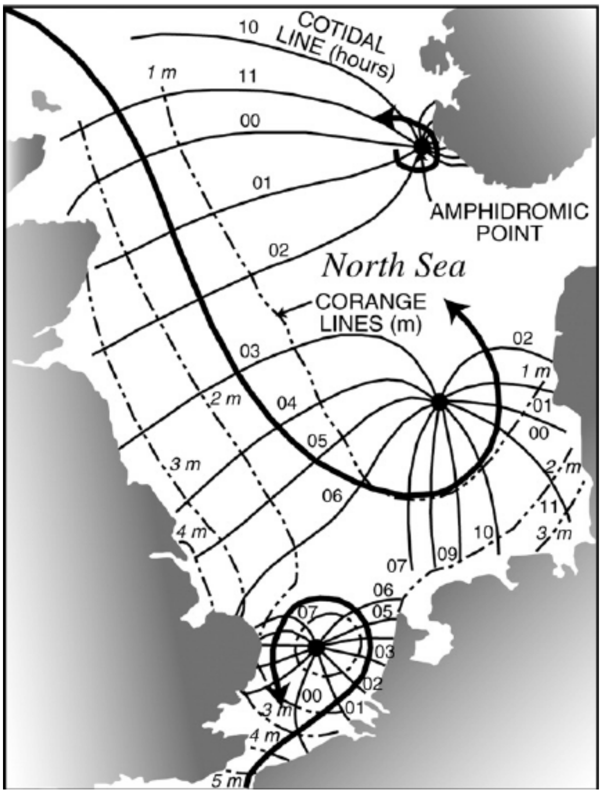


Figure 2.2: Overview of the North Sea amphidromic tidal system for the M2 tide. After Kvale (2006). Lines of equal tidal range are shown as “co-range lines”. The co-tidal lines indicate the time when high water arrives. Note the counterclockwise rotation of the tidal wave around each amphidromic point.

ponents (Nm), and the semiannual components (Ssa).

When there would be no landmasses, this would result in the equilibrium tide. Due to landmasses, this equilibrium tide is transformed into a tidal system with amphidromic points. The part of the system for the North sea is displayed in figure 2.2. The propagation of the tide can be explained as an oscillating wave around an amphidromic point. The propagation velocity and tidal deformation can be explained with a Kelvin wave (Bosboom and Stive, 2009). Figure 2.2 shows a schematisation of this kelvin wave, and it’s amphidromic points, of the North Sea. In this figure, the co-tidal lines indicate the time when high water arrives. Lines of the equal tidal range are shown as "co-range lines" (Kvale, 2006).

2.2.2. Wind driven currents

Winds and storms cause storm setups and wind-induced currents. For large storms, these currents cannot be neglected. Although these currents are highly irregular and seem on a small scale completely irregular, some large scale patterns and forces can be distinguished. The large scale patterns have a large influence on the wind climate of the North,- and the Wadden Sea. In this section, these large scale patterns are explained, and the influence on the North,- and Wadden sea climate is explained.

The primary source of earth’s heat is electromagnetic radiation from the sun. This radiation is most energetic at the equator and weakest at the poles. This results in a warm region at the equator and a cold region at the poles. Due to the rising of warm air, this creates a convection cell from the equator to the poles. If the earth were not rotating this single convection cell from the equator to the pole would be stable. However, due to the rotation of the earth, a Coriolis force exerted on the air, and the convection cell breaks up in three convection cells. Figure 2.3 shows a schematic representation of pressure belts and prevailing wind systems at the earth’s surface. These three convection cells cause the trade winds, the westerlies and the polar easterlies. These three different winds have different characteristics. The polar easterlies are moderate and blow most of the year only over land or ice. The westerlies are strong and variable winds. The trade winds are much

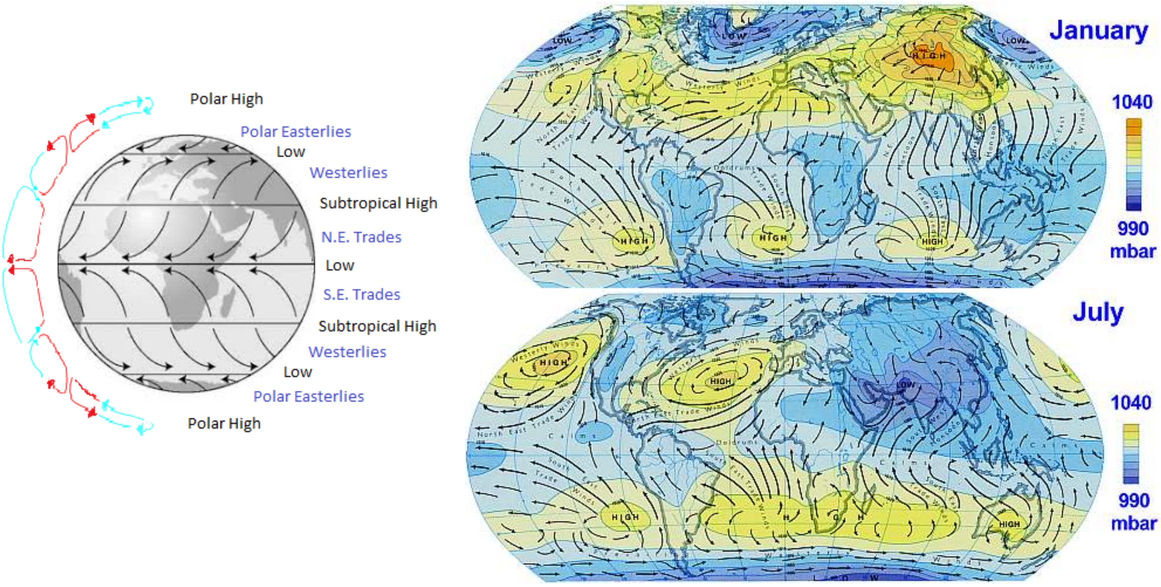


Figure 2.3: Global winds in January and July indicating the main wind systems (Anthhoni, 2000).

more moderate but persistent throughout the year. These global wind patterns describe the wind climate for a given part of the world. Just as for the tides, these winds would be stable if there were no landmasses. However, the landmasses cause instability in these convection cells. Since the land has much higher conductivity landmasses heat much faster in the summer than the oceans. Topographical features such as mountains play a significant role as well. This causes a different wind global wind pattern in the summer than in the winter (see figure 2.3).

In figure 2.3, the westerlies and the trade winds can be observed. The figure shows that especially the Asian landmass causes a considerable disruption in the pressure belts. It can also be noted that some tropical areas are dominated by trade winds (blowing in the same direction throughout the year), whereas seasonally reversing monsoons dominate other regions of the tropics and subtropics. For the North,- and Wadden sea, the influence of the westerlies is rather significant. However, the impact of the European landmass cooling down in the winter is visible, resulting in a landward wind. This describes the wind climate on the Dutch Continental shelf pretty good. In the summer, winds are more constant, and in the winter, large storms can cause large setups along the coastline.

2.2.3. Density driven currents

Due to differences in salinity or temperature of the water, a difference in density of the water can occur. A front is a boundary between two or more water masses. When there is a transition from one water mass to another, a front can arise. When the density of a water mass varies internally, stratification can occur. Stratification can result in internal waves or a shutdown of turbulence. The stratification and fronts induce currents, due to the 3D nature of these currents they cannot be predicted by 2DH modelling. Therefore the density-driven currents can be hard to predict and difficult to understand. These density-driven currents are explained in three categories: estuarine fronts, thermohaline fronts, and tidal fronts.

A good example of the importance of density-driven currents is described in Claessens (2016). Van Oord was building wind park Gemini, for workability conditions, currents were an important factor in the cost estimation of the wind park. The 2DH storm surge models predicted currents of only 0.7 $ms^{-}1$, however currents up to 1.2 $ms^{-}1$ were measured. In Claessens (2016), it is hypothesised that this difference in flow-velocity is due to density-driven currents. Understanding these currents is, therefore, of vital importance to use flow models.

The outflow of fresh river water into the salt ocean creates a region of tidal straining, stratification, and river plumes, these regions are generally called ROFIs (Region Of Freshwater Influence). Due to Coriolis these river plumes are deflected to the right (on the northern hemisphere) and create a coastal river. Figure 2.4 shows

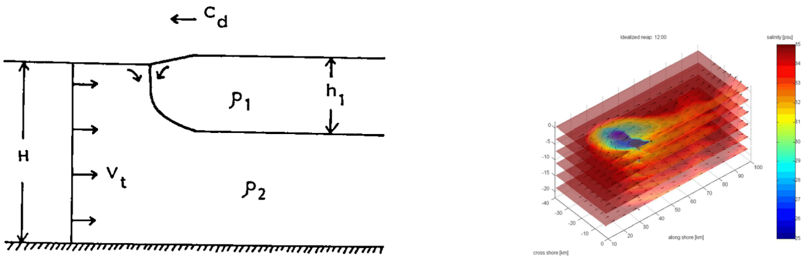


Figure 2.4: Schematization of a estuarine fronts (left) according to Yanagi (1987). Example of estuarine front (right) according to De Boer (2008). The example shows the river plume that is cause by the Rhine ROFI

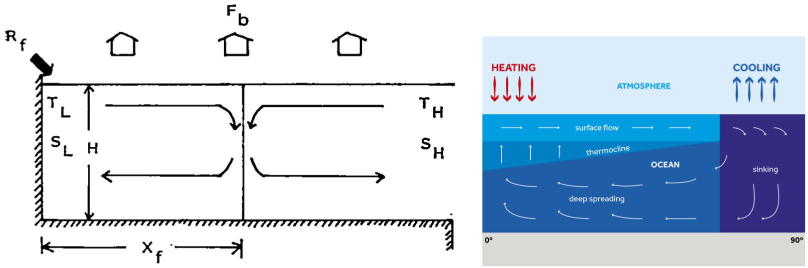


Figure 2.5: Schematization of a thermohaline front (left) according to Yanagi (1987). A example of a thermohaline front (right) according to University of Reading (2019). The example shows coastal upwelling due to temperature differences.

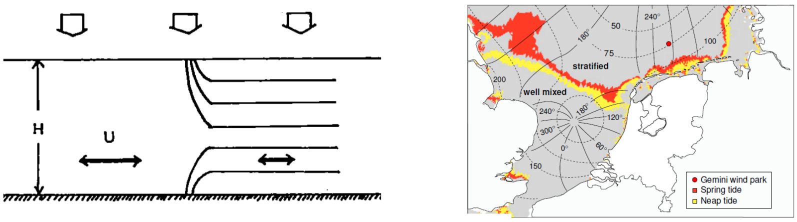


Figure 2.6: Schematization of a tidal front (left) according to Yanagi (1987). A example of a tidal front (right) according to Claessens (2016). The example shows the predicted position of frontal boundaries at spring tide (red) and neap tide (yellow). Based upon the Simpson-Hunter criteria for tidal fronts only taking into account tidal mixing and solar heating.

the schematisation of an estuarine front and an example. In De Boer (2008), the interaction between tides and stratification in the Rhine Region of Freshwater Influence is discussed.

When there is simultaneous heating and cooling of a water mass thermohaline circulation can occur. The warm water rise to the surface and the cold water sinks to the bottom resulting in a convection cell. The ocean conveyor belt and coastal upwelling are well-known examples of thermohaline circulation. Figure 2.5 shows a Schematization of a thermohaline front and an example.

The last forms of density-driven currents are tidal fronts. A tidal front is a unique structure in coastal waters where tidal mixing is dominant during the summer. The tidal front is the boundary where the mixing force of the tide is no longer large enough to create a well-mixed density distribution, and the depth profile becomes stratified. These tidal fronts are highly dynamic throughout the year and through the spring and neap tide cycle. In figure 2.6 the schematisation of a tidal front and the tidal fronts in the North,- and Wadden sea are shown.

These three types of fronts together form the basis of the classification of water masses in the North,- and the Wadden Sea. In figure 2.7 these water masses according to De Boer (2008) are displayed. In this figure, the Rhine (A4), and Themes (A2) ROFI are visible. Note that due to the current through the Channel the Themes

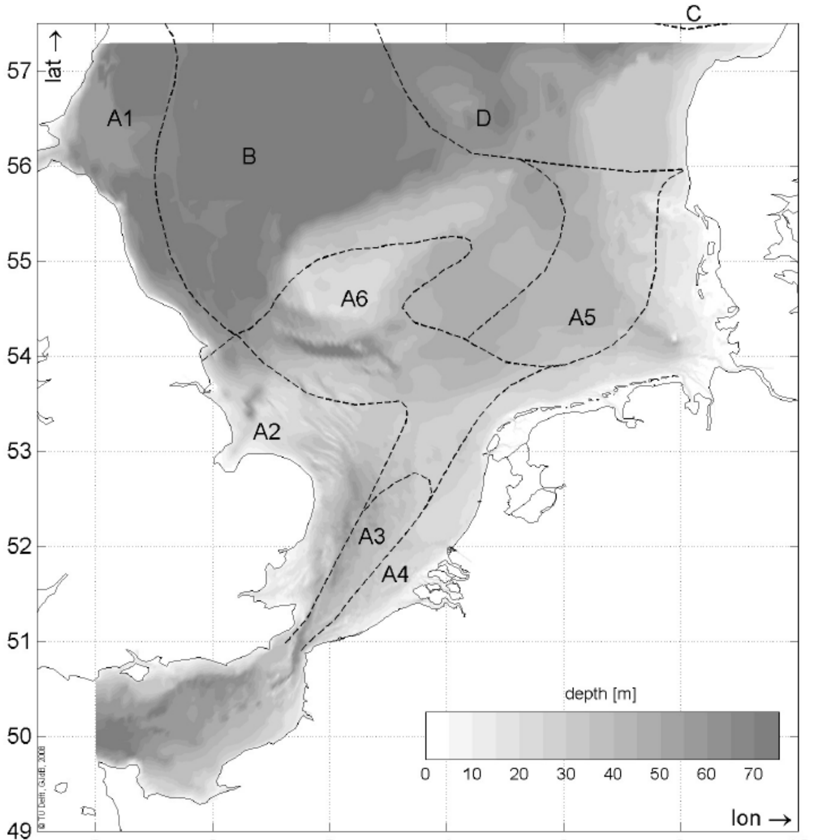


Figure 2.7: Classification of the different North Sea water masses. After De Boer (2008). The water types are A1 (Scottisch coastal water), A2 (English coastal water), A3 (Channel water), A4 (continental coastal water), A5 and A6 are transitional areas between A4 and B (Northern North Sea water), D is transitional water to C (Skagerrak and the Norwegian Rinne).

ROFI is deflected to the left instead of to the right. In the figure, the influence of the current through the channel is also visible. In the figure, the tidal fronts (A6 and A5) are indicated, and the stratified areas (B and D) are also visible.

2.2.4. Depth profiles of the currents

The combination of tidal currents, wind-driven currents, and density-driven currents result in a complex distribution of the total currents over the depth. However, most of the available hydrodynamics models are 2DH models. This is due to computation power restrictions and the applicability of 2D hydrostatic models for storm surge forecasting. These 2DH models are effective for predicting the water level, but the accuracy of flow velocity is barely investigated. This section describes the different velocity distributions over depth, and which is the most accurate for the North,- and Wadden sea.

For flow in a river, the depth profile of the currents can mathematically be determent. This results in a logarithmic profile over depth (Uijttewaal, 2018). This is the most widespread assumption for the profile of the currents over depth. Since the current profile of open water cannot mathematically be determined, a logarithmic profile is often assumed. The logarithmic profile is chosen out of a lack of alternatives. However, this is far beyond the assumptions that lead to the logarithmic profile (De Boer, 2008, Haren, 1990, Maas and van Haren, 1987) investigated the velocity profiles in the North Sea. From this research, it is found that the velocity profile is not linear in the North Sea (see Figure 2.8). It is found that the velocity vectors are ellipses, and rotate during the tidal cycle. In this article, it is found that the lower part of the profile forms a boundary layer and the upper part of the profile is linear. This research shows that the logarithmic profile should not be used at sea and that there is no suitable method for transforming the depth-averaged currents into surface currents.

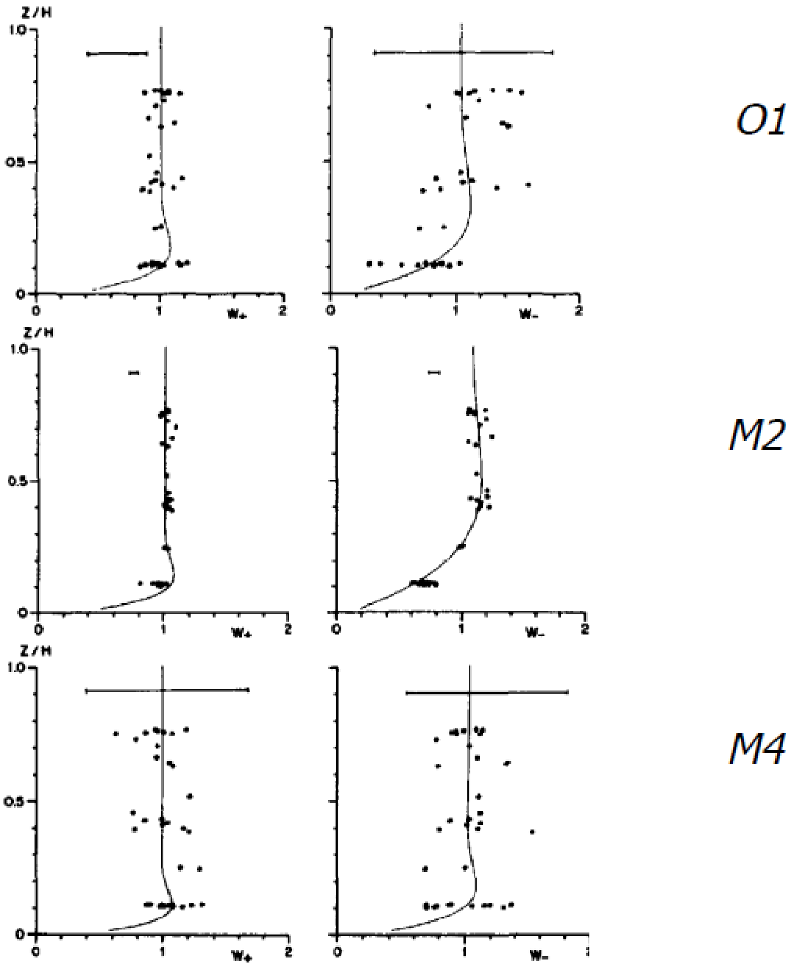


Figure 2.8: "Observations (denoted by dots) of amplitudes $W_{\pm}$ and phase angles $\theta_{\pm}$. of the M_2 frequency rotary current components as a function of normalized depth. Solid curves are best fit theoretical curves. Error bars are indicated at the top." (Maas and van Haren, 1987)

2.3. Hydrodynamic models

The objective of this study is to investigate the influence of the hydrodynamic conditions, described in Section 2.2, on the shipping route. To do so, data of the hydrodynamic conditions are needed; this data is gained from a model. The combination of the tidal, wind, and density-driven currents described in chapter 2.2 are so complex that they cannot (yet) be solved analytically. Therefore this study uses numerical solutions. These hydrodynamic models are a complicated subject in itself. In this study, the hydrodynamic models are only used as input of the route planning toolbox. This study does not go into detail about the specific properties and discretisations used to create the hydrodynamic models. However, some knowledge of the critical properties is needed to be able to use the hydrodynamic models for route optimisation. This section discusses the basic knowledge about hydrodynamic models required for the route planning toolbox.

For this study, shipping routes are limited to shipping routes on the North,- and the Wadden Sea. Therefore, the hydrodynamic models are limited to the North,- and the Wadden Sea. To make accurate computations for the North,- and Wadden sea, the DCSM (Dutch Continental Shelf Model) domain is used. Figure 2.9 shows the domain of the DCSM model.

2.3.1. Model properties

For different purposes, different models were developed. These models have many different properties. In this section, the most important properties are explained. There are many different properties, such as numerical schemes, stability criteria, boundary conditions, et cetera, that are not covered in this report.

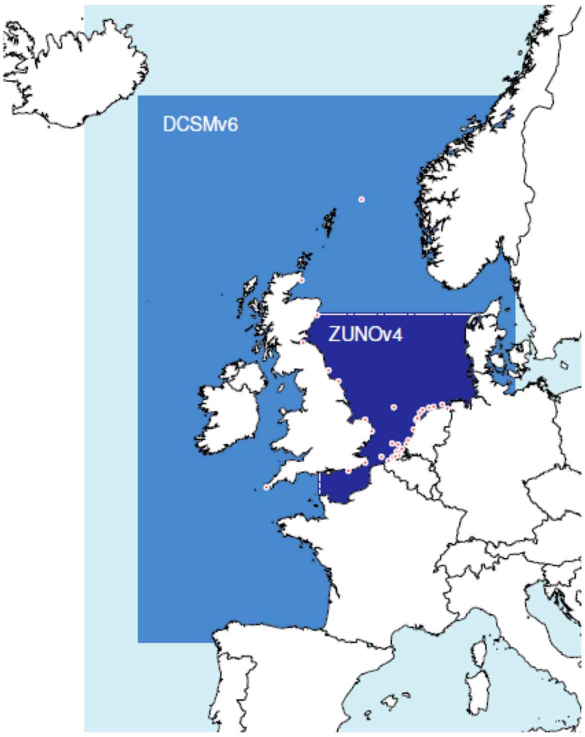


Figure 2.9: Model domain of DCSMV6-ZUNOV4-KF: The dotted white line indicates the border coupling between the DCSMV6 and ZUNOV4 models. The red dots with a white border are the locations of 32 water level measurement stations used for data assimilation (Sumihar, 2012).

In numerical models, a distinction is made between 2DH models and 3D models. The 3D models are detailed and accurate but computationally intensive, and the 2D models are faster but cannot model the density-driven currents and assume a logarithmic profile of the velocity over depth. For a long time, only the 2D models were used since 3D models were not yet feasible. However, nowadays the computational power has increased significantly, and 3D models are starting to take their place.

In numerical models, the continuous solution space can be discretised with a grid. For this grid, different choices can be made. In this report, three different types of grids are mentioned. The three different meshes are visualised in figure 2.10. For Cartesian grids, the resolution is constant in space. For curvilinear and irregular grids the resolution is not constant in space. The first and the most simple one and is a Cartesian grid. Each point on a Cartesian grid can be specified with an (m, n) coordinate and the neighbours follow directly from each point. The second grid is curvilinear. This is a more complicated variation of a Cartesian grid. Here the grid lines are curved, but each point can still be described with an (m, n) coordinate and the neighbours follow directly from the grid. The third grid type is irregular grids. Here the grid points do not have a specific order and cannot be determined from an (m, n) coordinate. The neighbours do not follow directly from the grid point but are defined in a separate list. When the grid points are connected with all triangles, this grid is called a Triangular Irregular Network (TIN). When the grid points are connected with triangles, squares, pentagons or hexagons, it is called a Flexible Mesh (FM). The right plot of Figure 2.10 shows an FM grid.

2.3.2. Models used in this study

The Netherlands is a country that is driven by the opportunities and threats of the ocean. This raises a need for information about the behaviour of the ocean that has resulted in sophisticated hydrodynamic models in many varieties. Since the output of these models is needed for many different purposes, a few of these models are publicly available. Rijkswaterstaat publishes these models on `http://noos.matroos.rws.nl/`. This report refers to these models as operational models.

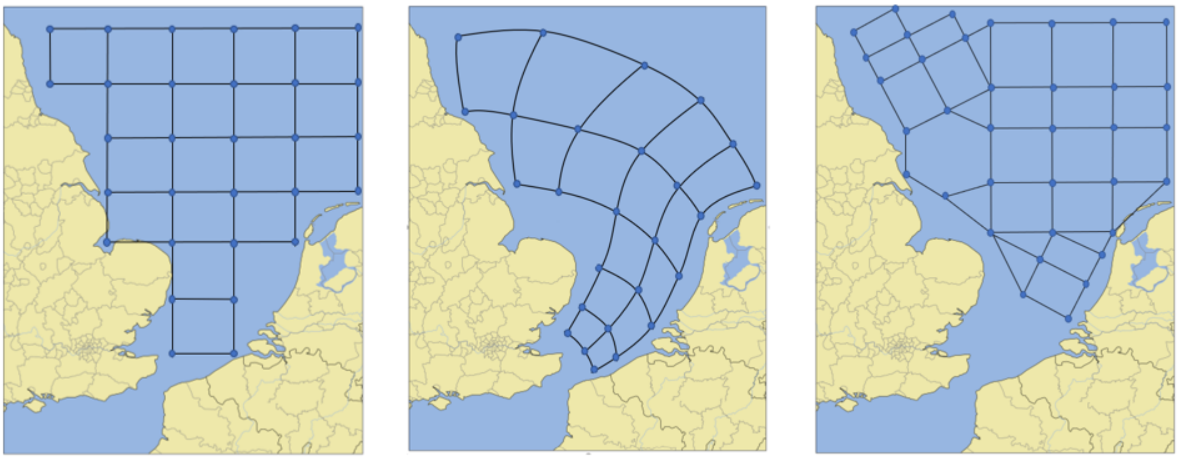


Figure 2.10: Three different types of grid discretisations. The grid on the left is a Cartesian grid, The grid in the center is a curvilinear grid and the grid on the right is a irregular grid (TIN).

The first model is the DCSMv6-ZUNOv4 kf model. This model is a domain decomposition model which are coupled through a horizontal domain decomposition. The ZUNOV4 model is a 2DH model which covers the whole southern North Sea including a part of the English Channel and has a finer resolution than the DCSMV6 model. The boundary conditions of the more detailed ZUNOV4 model are taken from, the larger continental shelf model DCSMV6. Figure 2.9 shows the domains for both models. The boundary between the DCSMv6 model and the zunov4 model is indicated with a white dotted line. The successor of the DCSMv6-ZUNOv4 kf model is the DCSM-FM models. These models are currently in development. When these models are finished, they will replace the DCSMv6-ZUNOv4 kf model.

In this study, four different numerical models were available. These four models are summed up below. The DCSMv6-ZUNOv4 is the current operational model and is publicly available at `http://noos.matroos.rws.nl/`. The FM models are currently in development by Deltares and will be the successor of the zuno model. All these models model the tidal and wind-driven currents.

- DCSMv6-ZUNOv4 (With Kalman-Filter)
- DCSM-FM 0.5 nm (nautical mile)
- DCSM-FM 100 m
- 3D DCSM-FM 0.5 nm

The first important distinction between the models is that the DCSMv6-ZUNOv4, DCSM-FM 0.5 nm, and the DCSM-FM 100 m models are 2DH models. These models can, therefore, not model density-driven currents (Zijl et al., 2015). The 3D DCSM-FM 0.5 nm model is, however, a fully 3D model and can model the density-driven flows (Zijl et al., 2018). The second important distinction is grid discretisation. The DCSMv6-ZUNOv4 uses a combination of a Cartesian grid and a curvilinear grid. The DCSM-FM 0.5 nm, DCSM-FM 100 m, and the 3D DCSM-FM 0.5 nm model use an FM grid. The third distinction is resolution. Since all of the models use either a curvilinear grid or an FM grid the grid size is nowhere constant. However, the grid size of the DCSM-FM 0.5 nm and the 3D DCSM-FM 0.5 nm model are identical and approximately 0.5 nm in Dutch coastal waters. The resolution of the DCSM-FM 100 m model is much finer and around 100 meters in Dutch coastal waters. Finally, the DCSMv6-ZUNOv4 model has the most course grid.

A critical remark can be placed with the DCSMv6-ZUNOv4 model. This model was initially developed for predicting water levels. When the model was developed, there was no accurate bathymetry available. This was compensated at some points with discrete jumps (Zijl et al., 2013). These sudden jumps can influence the modelled currents.

2.4. Ship movement study

A method is needed to predict ship movements to determine the optimal shipping route. This section presents the different methods to predict ship movements under the influence of forcing of currents, wind or waves.

2.4.1. Inertial models

The most accurate method for predicting ship movements is an inertial model. This is Newton's second law implemented on the ship. The result is a system of differential equations. Yasukawa and Yoshimura (2015) introduced a standardised method for applying these equations. The resulting equations are:

$$\begin{cases} (m+m_x)\dot{u}_t - (m+m_y)v_t R - x_G m R^2 = X_H + X_R + X_P \\ (m+m_y)\dot{v}_t + (m+m_x)u_t R + x_G m \dot{R} = Y_H + Y_R \\ (I_{zG} + x_G^2 m + J_z)\dot{R} + x_G m(\dot{v_m} + u_t R) = N_H + N_R \end{cases} \tag{2.2}$$

In this equation the variables of interest are v_m, u, R which are respectively lateral velocity and mid-ship, Surge velocity and yaw rate.x_G is the distance between the turning point of the ship and the centre of gravity. m, m_x and m_y are the mass of the ship and the added mass in x and y direction. I_{zG} is the moment of inertia of ship around the centre of gravity and J_z is the added moment of inertia. The forces on the ship are denoted in the right-hand side of the equations. The subscripts are respectively Hull forces, rudder forces and propeller forces. The hull force vector accounts for the wind, wave and current forces. The propeller force vector accounts for the amount of propulsion power In the rudder force vector, the rudder angle δ can be accounted for.

These standardised equations are called the Manoeuvring Modelling Group (MMG) equations. These equations are highly useful for determining the capabilities of the ship. Chen et al. (2015), Fitriadhy et al. (2012) use these equations to determine the minimum turning circle of a specific ship.

2.4.2. Vector addition models

These MMG equations are very accurate for predicting the limits of the ship but are also computationally intensive. In a steady-state condition, this system of differential equations results in a simple vector addition model. For this vector addition method, the velocity vector due to forcing (u_f, v_f) and the velocity vector of the ship compared to the water (u_s, v_s) are added to each other. In figure 2.11 the assumed parameters are displayed. For this calculation, the current location, goal destination, forcing velocity vector, and the length of the ship's velocity vector are known. The parameters of interest are the total velocity vector (u_t, v_t)

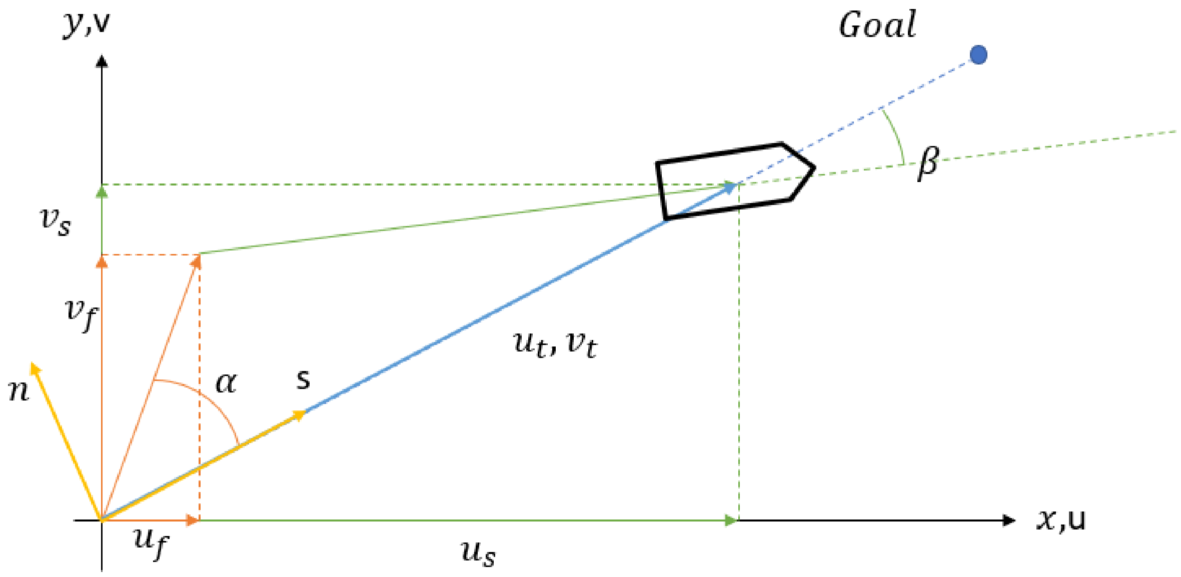


Figure 2.11: The parameters for the addition of the flow vector (orange) and the vessel vector (green). This gives the total velocity (Blue).

given that the direction of the total velocity vector is towards the goal. A new coordinate system is defined to calculate the total velocity vector. This system has its origin at the centre of the ship. The surge vector (s) along directed in the course of the ship, and the lateral vector (n) is directed perpendicular to the course of the ship (see figure 2.11). α is the angle between the course of the ship and the flow vector. This lateral component(n_f) and surge component (s_f) of the forcing vector can be calculated as follows:

$$\begin{cases} n_f = -n_s = \sqrt{u_f^2 + v_f^2}\sin\alpha \\ s_f = \sqrt{u_f^2 + v_f^2}\cos\alpha \end{cases} \tag{2.3}$$

Now the surge component of the total velocity vector ($s_t = s_f + \sqrt{v_{max}^2 - n_s^2}$) can be calculated given the lateral component of the ship ($n_s = -n_f$) and the length of the ship's velocity vector (V, reduced for squat). Now that the surge component of the total velocity is known a simple transformation can calculate the total velocity vector in the original coordinate system. The drift angle(β) is the angle between the ship's velocity vector (with respect to the water) and the course and is not equal to zero when currents are present.

$$\begin{cases} u_t = \left(s_f + \sqrt{V^2 - n_s^2}\right)\cos\alpha \\ v_t = \left(s_f + \sqrt{V^2 - n_s^2}\right)\sin\alpha \\ \beta = \tanh\dfrac{v_t - v_f}{u_t - u_f} \end{cases} \tag{2.4}$$

This method determines the total velocity of the ship for a given velocity of the ship and the velocity vector of the forcing. This velocity vector due to forcing is the substitution of three vectors. First, the velocity vector due to currents, second the velocity vector due to the wind, and third the velocity vector due to waves. These individual vectors can be calculated for the steady-state variation of the MMG equations. This results in a forcing vector due to the current that is equal to the currents itself. In this study, just the forcing due to the currents is used. However, the model can relatively only be updated by including a wind and wave vector.

2.4.3. Squat

When modelling ship movements, the squat is an important effect. The squat effect takes place when a ship travels through a confined space (such as a channel). Due to the decrease of the cross-sectional flow area that is caused by the presence of the ship, the flow velocity increases at the location of the ship. Bernoulli's law says that when the flow velocity increases the pressure decreases. This decrease in pressure causes a further sinkage of the ship called squat. This squat effect causes a bow wave in front of the ship witch increases the Resistance of the ship significantly. With empirical formulas, the sailing velocity in confined water can be calculated based on the dimensions of the channel and the sailing velocity in deep water. These equations can then be simplified to unconfined shallow water waterways (such as the Wadden Sea).

Many equations can predict squat effects. In this study, the equations that are used within van Oord are used as well. Equation 2.5 shows the equations for squat for confined spaces (van Oord, personal communication, 2019). In equation 2.5, the parameters are defined as: Froude number: Fr, deep water sailing velocity: V_{max}, gravitational constant:, g, water depth: h, total weight of the ship: W_{ship}, Length over the waterline of the ship: L, Width over the waterline of the ship:, B, draft of the ship:, T, density of water: ρ_w, width of the channel: b, under keel clearance: ukc, Hydraulic radius: R_H, block coefficient: C_b, blockage coefficient: S, critical sailing velocity, $V_{critical}$, maximal permissible squad max_squat, velocity at which squat is equal to max_squat: V_{max_squat}, and V the maximal sailing velocity for the given waterway.

$$
\begin{aligned}
Fr &= \frac{v_{max}}{\sqrt{gh}} \\
C_b &= \frac{W_{ship}}{LBT\rho_w} \\
S &= \frac{BT}{b+2h+2T+B} \\
R_H &= \frac{bh-BT}{b+2h+2T+B} \\
max\_squat &= h-T-ukc \\
V_{critical} &= \left(1.002+0.005\left(\frac{\sqrt{BT}}{R_H}\right)-0.1159\left(\frac{\sqrt{BT}}{R_H}\right)^2+0.019\left(\frac{\sqrt{BT}}{R_H}\right)^3\right)\tanh\left(\frac{1}{Fr^2}\right) \\
V_{max\_squat} &= min\left(0, max\_squat\frac{30}{C_bS^{2/3}}\right) \\
V &= min\left(V_{max\_squat}, V_{critical}\right)
\end{aligned}
\tag{2.5}
$$

Equation 2.5 is valid for shipping in confined waterways. When the waterway is unconfined, for example, on the open sea, another function should be used. For this study, it is assumed that the shipping route is freely routable; this means that the waterway is not confined. Equation 2.5 can be used for unconfined waterways by taking $b = 9WWL$.

2.5. Optimizing with graph theory

In Chapter 1, the different route optimisation methods that are available in the literature are summarised, and the choice for optimisation method that is used in this study is elaborated. The optimisation method that is used in this study is graph theory. This section elaborates the subject graph theory. Optimising a route through several vertices and edges is a classic problem. In the early stages of numerical computational mathematics, algorithms were developed to solve these problems. However, the real ground-breaking application of these algorithms developed much later due to the lack of computational power. Today we are living in a time and age where these numerical applications can solve significant problems. This study focuses on finding an innovative application for these decades-old algorithms. This section explains the shortest path algorithms that exist in the literature. This section starts with the variations of the shortest path algorithms that exist, continuous with the distinctions between graphs, to end up with a general description of the specific problem that is relevant for this study, namely the single-pair shortest path problem (SPSPP). Lastly, there is an example given of Dijkstra's algorithm; this is the most basic SPSPP algorithm and is used often in this study.

2.5.1. Variations of the shortest path problem

The shortest path problem consists out of multiple sub-problems. To find the right method for the correct problems, categorisation of these problems should be made. Four distinct categories can be defined for the shortest path problem (Cherkassky et al., 1996, Goldberg and Johnson, 2009, Turner, 2011). All four groups have different numerical methods to solve them. This section describes all four of them and explains which is relevant for the optimal route problem.

First, there is the all-pairs shortest path problem. This problem searches for the route with the lowest cost function that travels through all the vertices, for no given start or end. Second, there is the single-source shortest path problem. This problem searches for the route with the lowest cost function that travels through all the vertices, for a given start but no given end. Third, there is the single-destination shortest path problem. This problem searches for the route with the lowest cost function that travels through all the vertices, for no given start but a given end. This problem is the inverse of the single-source shortest path problem for undirected graphs; however, it is different for directed graphs. Lastly, there is the single-pair shortest path problem. This problem searches for the shortest path between a source vertex and a target vertex. The solution does not travel through all the points and depending on the cost function does not have to be the geometrical shortest path.

2.5.2. Distinctions between graphs

The shortest path algorithm seeks to find the cheapest solution of edges that are a solution for the specific problem. In graph theory, the route can travel from node to node when there is an edge specified. In graph theory, there is a database that contains all the nodes, edges, and weights of the specific problem. This database is called a graph. This section elaborates on the difference between the graphs.

A general distinction between graphs can be made based on directionality and weights Networkx (2014, 2015). Unweighted graphs are graphs that have no weights assigned to their edges; for these graphs, all the edges have the same weight. Weighted graphs have weights assigned to their edges. This is done with a so-called cost function (Nannicini and Liberti, 2008). The optimal route is the set of edges with the lowest total weight. This is called the optimisation of the cost function. Within the weighted graphs, there are directed, undirected and mixed graphs. For directed graphs, the output of the cost-function depends on the direction of the edge. for example: $C(V1,V2) \neq C(V2,V1)$ (C is the cost function, V1 and V2 are Vertex 1 and 2). A mixed graph means that some edges are directed, and some edges are undirected. This implies that a directional or mixed graph is per definition a weighted graph; otherwise, the cost function cannot be different.

2.5.3. Single Source Shortest Path Problem (SSSPP) algorithms

The problem that needs to be solved in this study is a single-pair shortest path problem for directed and weighted graphs. A commonly used method to solve the single-pair shortest path problem is by using a single-source shortest path problem method. The stop criteria are changed from *stop when all vertices are visited* to *Stop when target vertex is reached.* The single-destination and all-pair method cannot be used for the single-pair method.

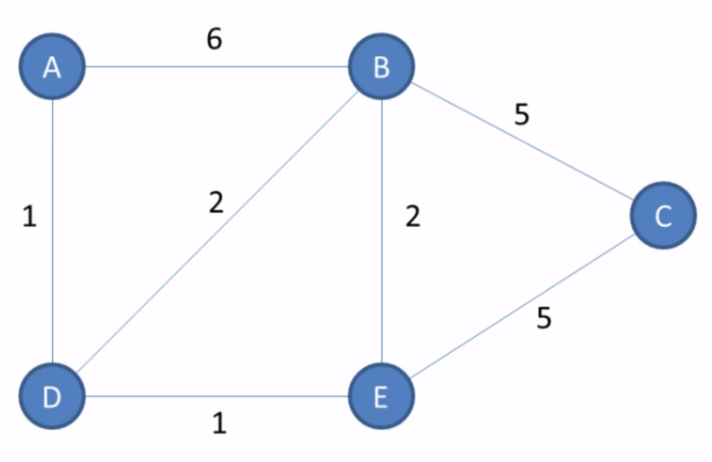


Figure 2.12: Graph of the simple example with a total of 5 vertices and 7 edges. The shortest pat from vertex A to B is: $A \rightarrow D \rightarrow B$

The Floyd-Warshall algorithm (Floyd and Warshall, 1962) specialises on finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles). The Johnson algorithm (Johnson, 1977) specialises for sparse graphs with positive or negative edge weights (but with no negative cycles). Both these algorithms calculate the shortest path or the all-pairs shortest path problem. These algorithms are therefore not useful in this study.

In Dijkstra (1959) the Dijkstra algorithm for the single-source shortest path problem is given. This method has a time complexity of $O(V^2)$ (where V is the number of vertices). The method is applicable for single-source shortest path problems with undirected weighted graphs.

Bellman and Odcira (1956) proposes the Bellman-Ford algorithm. The Bellman-Ford algorithm solves the single-source shortest path problem if edge weights may be negative. This method has a time complexity of $O(|VE|)$ (where V is the number of vertices and E the number of edges). The algorithm is slower than the Dijkstra algorithm but is more versatile in return. The method is applicable for single-source shortest path problems with directed weighted graphs.

Hart et al. (1968) the A* search algorithm is proposed for solving the single-pair shortest path problem. This algorithm uses heuristics to try to speed up the search. This method has a time complexity of $O(E)$. The method is applicable for single-pair shortest path problems with directed weighted graphs.

2.5.4. Example of Dijkstra's algorithm

This section shows a simple example of the steps that Dijkstra's algorithm makes to find the shortest route. For this example, the arcs are weighted and undirected. The simple example is to find the shortest path through the graph in figure 2.12. The following parameters are defined to start the computation:

- start vertex = A
- destination vertex = B
- current vertex = start vertex
- visited = []
- initial values of the weights see table 2.1

Table 2.1 show three columns. The first column is the column that indicates the vertex, and the second column indicates the distance from the initial vertex (A) to the vertex of that row, the last column shows the previous vertex in the shortest path to the vertex of that row.

Table 2.1: Simple example of Dijkstra's algorithm. Table with results at t = 0.

Vertex	Distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

In the enumeration below the while-loop of Dijkstra's algorithms is displayed. The loop is repeated until the while condition is no longer valid.

While current vertex is not destination:

1. Visit the unvisited vertex with the smallest know distance from the start vertex
2. For the current vertex, calculate the distance of each neighbour from the start vertex.
3. If the calculated distance is less than the known distance update the distance.
4. add the current vertex to the list of visited vertices.

Below the iterations for this example are listed. For this specific example, it takes four iterations. The shortest path for this example can easily be verified for this example. For Dijkstra's algorithm, it is proven that it always gives the correct solution for non-negative weights. From the fourth iteration in Table 2.2, the shortest route can be determent. The weight of vertex B is three and can be reached via D, D weights two and can be reached via A. The shortest route is therefore $A \rightarrow D \rightarrow B$ with a total weight of 3.

Table 2.2: Simple example of Dijkstra's algorithm. The table shows the detailed steps of the five iterations that it takes to find the shortest path

Iteration 1 Test if $None \neq B$

1. visit vertex A, current vertex = A.
2. $A \rightarrow D = 1$, $A \rightarrow B = 6$.
3. update the table below
4. visited = [A]

Vertex	Distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Iteration 2 Test if $A \neq B$

1. visit vertex D, current vertex = D.
2. $A \rightarrow B = 2$, $A \rightarrow E = 2$, $A \rightarrow A = 2$.
3. update the table below
4. visited = [A, D]

Vertex	Distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Iteration 3 Test if $D \neq B$

1. visit vertex E, current vertex = E.
2. $A \rightarrow B = 4$, $A \rightarrow C = 7$, $A \rightarrow D = 3$.
3. update the table below
4. visited = [A, D]

Vertex	Distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Iteration 4 Test if $E \neq B$

1. visit vertex B, current vertex = B.
2. $A \rightarrow A = 9$, $A \rightarrow D = 5$, $A \rightarrow E = 5$, $A \rightarrow C = 8$,
3. update the table below
4. visited = [A, D]

Vertex	Distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Iteration 5 Test if $B \neq B$. This is false so stop.

3 | Methods

In this chapter, the existing knowledge, given in chapter 2, is implemented for the specific problem of this study. The results of the methods is a python package that can solve for the optimal route given a hydrodynamic model. In this chapter, it is explained how the python tool is developed. The methods of model development are described in four parts. First, the graph strategy is discussed. In this section research question, one is answered. Second, the time-dependent shortest path problem is discussed. This section answers research question 2. Third, the cost functions and variable shipping speeds are discussed. Last, the modular package implementation is discussed. This section answers research question 3. After the methods for the model development are discussed the methods for the model application are discussed, this section answers research question 4.

3.1. Methods for model development

In this chapter, the computations behind the route optimisation toolbox are explained. With the development of a good route, optimisation toolbox, the first four research questions are answered. The section starts with the graph schematisation of the problem. This part answers the first research question. The section continues with the time-dependent route optimisation. This section answers the second research question. Lastly, the different cost functions are explained.

Many studies have aimed to optimise shipping route for many different purposes, on many different scales. Literature shows many route optimisation studies for transatlantic shipping routes (referred to as large scale shipping routes). On these large scales, currents become less important, and weather conditions dominate the routes. This results in a branch of route optimisation called weather routing (Böttner, 2007, Krata and Szlapczynska, 2018, Perera and Soares, 2017, Walther et al., 2016). This branch seeks to optimise routes for wind and wave fields. Other studies have searched to optimise shipping routes for comfort (Kosmas and Vlachos, 2012). These optimisations are mainly applied for passenger's vessels. Other studies tried to optimise routes from a more economical and environmental point of view. These studies have tried to minimise the fuel consumption of routes (Kuo, 2010, Lee et al., 2018b, Park and Kim, 2015). The last type of route optimisations is shipping speed optimisations (Li et al., 2018, Park and Kim, 2015). In these studies, the sailing velocity is varied in order to optimise for fuel consumption.

Many different algorithms are used in shipping route optimisations. Many studies use graph theory (Lee et al., 2018a, Mannarini et al., 2013, Montes, 2005). Other studies use hill-climbing algorithms (Dorer and Calisti, 2005, Tierney, 2013), a simulated annealing algorithm (Kosmas and Vlachos, 2012), the Pareto Evolutionary Algorithm (Vettor and Guedes Soares, 2016), or Neural networks (Zhang et al., 2019). Which algorithm is the best highly depends on the optimisation purpose.

In this study graph theory is used to find the optimal route through a continuous solution space. As discussed in chapter 1, there are several different methods for optimising a route through a continuous solution space. The main reason why this method is chosen is that it cannot get stuck in local minima. Methods as the hill-climbing algorithm or the annealing method are known for the outcome to depend on the initial conditions. For a different initial route, the algorithm can return a completely different solution. The algorithm returns a local minimum instead of a global minimum. This difference between the local minima and the global minima is crucial for highly dynamic problems such as flooding and drying of banks, or finding the route around islands. The main reason for choosing graph theory is the property that is that it cannot get stuck in local minima, but always returns the global minima.

3.1.1. Graph strategy

In order to calculate an optimal shipping route, The continuous solution space needs to be discretised into a graph with nodes and edges. For large graphs with many nodes and edges, computation time can increase very fast. However, for small graphs, the numerical error becomes large, and the resolution of the route becomes small. This section searches for an algorithm to keep the graph small but the resolution of the route as large as possible. This section answers research question 1, the first research question is:

What is the best graph strategy given computational power constraints?

For graphs with a high resolution and a relatively low number of nodes, the grid size cannot be uniform over the graph. For area's where the flow is uniform, and isotropic the grid size of the mesh can be much larger than for area's where the flow is non-uniform and non-isotropic. For creating such a grid, the nodes can be pruned on places where the resolution is not very important while maintaining the high resolution on important places. In order to create such a mesh with a varying grid-size per cell, a Triangular Irregular Network (TIN) is used. With a TIN grid, the nodes are no longer in a straight line or curved line (as done with straight or curvilinear meshes), but the points can be random. All the nodes are later connected with triangles. For meshing these random points into a TIN grid, the Delaunay algorithm is used. This algorithm maximises the minimum angle of all the angles of the triangles in the triangulation. This algorithm is named after Boris Delauney's work in 1934 (Delauney, 1934) this algorithm is implemented in the Scipy package.

A length scale is used to determine the spatial resolution of the grid. This length scale represents the spatial resolution of the grid. The length scale should be low for areas where the flow is non-uniform and non-isotropic, and the length scale should be large for areas where the flow is uniform and isotropic. For determining the grid size based on the length scale, a lower and upper bound of the grid size is determined. In the route optimisation package, the upper bound is set to the grid size of the flow model and the lower bound can be specified by the user. If the length scale in a certain point is equal to one, the grid size becomes the upper bound of the grid size, if the length scale is 0 the grid size in that point becomes the lower bound of the grid size. To determine the nodes of the TIN grid the route optimisation package loops over all the points (within the project domain) of the flow model and determines the distance to the closest node of the TIN grid. Only if this distance is larger than the length scale in that point, the node is added to the nodes of the TIN grid. When all the nodes are determined, the mesh is determined with Delauney's algorithm.

This length scale is based on the vorticity (curl) of the flow. The curl of the flow is a measure for the vorticity of the flow (see equation 3.1). When the absolute value of the curl is low, the flow is isotropic, and when the absolute value of the curl is high, the flow is non-isotropic. The curl of the flow can be transformed into a length scale. Figure 3.1 shows the relation between the curl and the length scale of the grid size. In order to calibrate the length scale, non-linearity parameters are introduced. This non-linearity parameter is a shape parameter of the length scale function. In equation 3.2 this non-linearity parameter is called β.

$$curl = \nabla \times \vec{u} = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \tag{3.1}$$

Another method to determine the length scale is by using the magnitude of the flow. For determining the length scale, it uses the same method as for the curl method only instead of the curl the magnitude of the flow velocity is used. In the python package, both methods are implemented in the same function. This is done with a blend factor. The length scale is calculated based on the curl and based on the magnitude of the flow. Then the length scale is averaged with a blend factor. In equation 3.2 the blend factor is called α. In this study, the length scale is chosen as a function of the flow.

When this function for the length scale is used, the length scale is a time-dependent parameter. However, the grid is constant in time. The length scale is set to the maximal value of the time series to transform the length scale into a non-time dependent value. Another method to do this is to take the time-averaged value of the length scale or the root-mean-square (RMS) value of the length scale. Since the length scale is an oscillating parameter, the mean value will approach 0, and the RMS value will give the standard deviation of the length scale. In this study, it is chosen to use the maximal value because this gives the normative length scale instead of the standard deviation of the length scale.

$$\frac{LS}{\Delta_{min}} = \alpha\left(1 + |\nabla \times \frac{\vec{u}}{u_r}|\right)^{-\beta_c} + (1-\alpha)\left(1 + |\frac{\vec{u}}{u_r}|\right)^{-\beta_m} \tag{3.2}$$

Equation 3.2 shows the function of the Length scale based on the properties of the flow. With: LS = resulting length scale, α = blend factor between the curl and the magnitude method, Δ_{min} = minimal length scale, β_c =

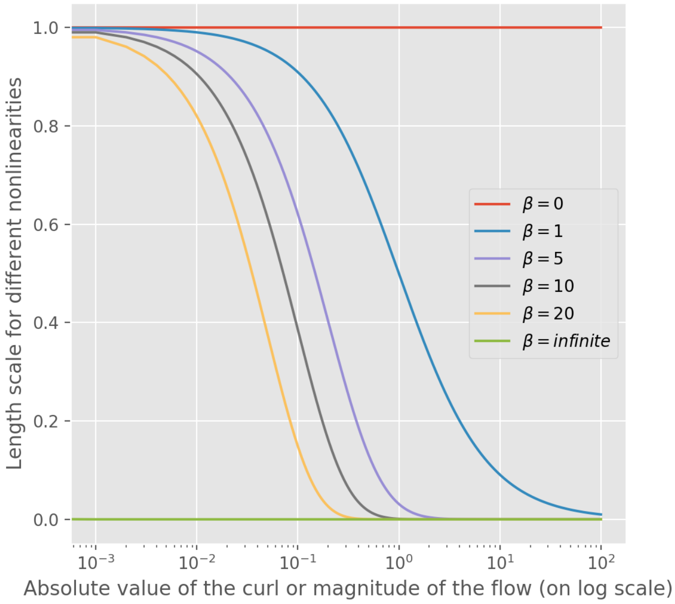


Figure 3.1: The Length scale as a function of the curl / magnitude of the flow for different non-linearity factors

non linearity parameter for the method with the curl of the flow, β_m = non linearity parameter for the method with the magnitude of the flow, $\vec{u}$ = the velocity vector of the flow, and u_r is the reference flow velocity. In order to correctly formulate the expression for the length scale all the parameters are made dimensionless. This results in a dimensionless length scale $\frac{LS}{\Delta_{min}}$, and a dimensionless flow vector $\frac{\vec{u}}{u_r}$. By doing so the parameters α, β_c, and β_m are dimensionless parameters. In this study the reference flow velocity (u_r) is set to 1.

In order to create a graph, both nodes and arcs are needed. In the previous subsection, the nodes are reduced into a TIN mesh. The arcs of the graph need to be determined to create a useful graph. The arcs of the graph are based on the neighbours of each node. The arcs are determined from a start node to a neighbouring node. By doing the possible directions of the shipping, the route is schematized from a continuous 360-degree solution space to a discrete number of directions. In Figure 3.2, the neighbouring nodes of a central node are shown. In blue, the direct neighbours of the central node are shown. For these edges, the continuous 360-degree solution space is discretised in 6 directions. This results in a directional resolution of 360/6 = 60 degrees. However, the directional resolution can be increased when the edges are not only defined between direct neighbours, but also between neighbours of neighbours. The orange nodes of Figure 3.2 are not direct neighbours of the central node but are neighbours of the blue nodes. In this report, the orange nodes are referred to as second layer neighbours (or nb = 2). With these second layer neighbours, the number of arcs is increased from 6 to 12, and the directional resolution is increased to 360/12 = 30 degrees. The green nodes of Figure 3.2 are not second layer neighbours of the central node, but they are neighbours of the orange nodes. In this report, the green nodes are referred to as third layer neighbours (or nb = 3). With these third layer neighbours, the number of arcs is increased from 12 to 24, and the directional resolution is increased to 360/24 = 15 degrees. This process is generalised for any layer of neighbouring nodes. This parameter for the directional resolution (nb) is set as an input parameter for creating the graph.

When the edges are defined for only direct neighbours, the hydrodynamic features of an arc are the average of the start and stop nodes. When this method is applied for multiple layers of neighbouring nodes, the algorithm is allowed to "jump over" obstacles. In other words, the spatial resolution of the graph decreases. Therefore, a new algorithm is presented to increase the directional resolution without decreasing the spatial resolution. The nodes between the start and stop node are also taken into account to prevent the loss of spatial resolution. The right plot of Figure 3.2 shows how these in-between nodes are determined. In this figure, the nodes of influence for the arc between the two blue nodes are determined. The two blue nodes are third layer neighbours of each-other (nb = 3). For this example, the nodes of influence are the blue nodes plus the green nodes. The green nodes are defined as the set of nodes that are contained in both the sets of

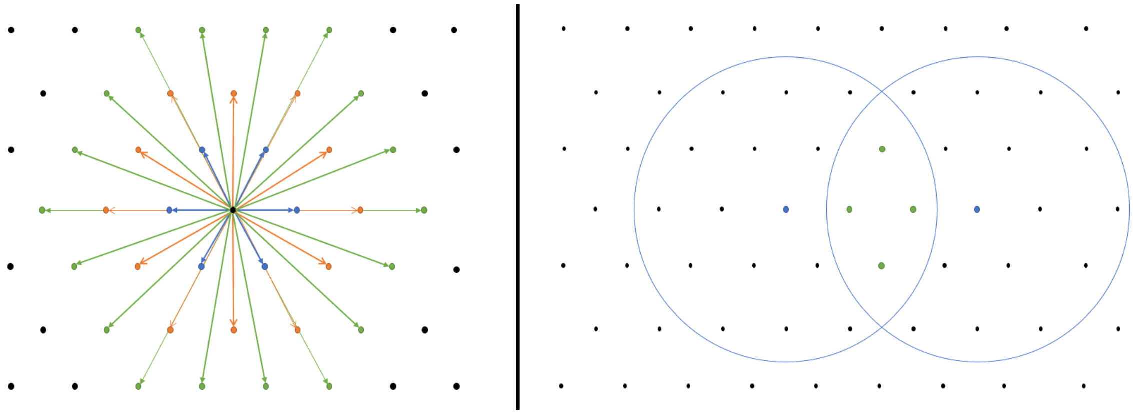


Figure 3.2: The left figure shows the branches for 3 neighbouring layers in a perfect Delauney mesh, the right figure shows the nodes that determine the weight of the branch.

second layer neighbour nodes of the start and stop nodes. The nodes of influence, of a (nb)'th order arc, are determined in this study as the start and stop node plus the set of nodes that are contained in both the sets of the (nb-1) layer neighbour nodes of the start and stop nodes (in which nb is the number of neighbouring layers).

3.1.2. Time dependent shortest path problem

In Chapter 2.5 it is found that the single pair shortest path problem (SPSPP) can be solved with Dijkstra's algorithm or with the A* algorithm. For the SPSPP the weights of the arcs between the vertices are constant in time. However, for the problem that is considered in this study, the weights of the arcs are highly dynamic in time. This distinction gives rise to the Time-Dependent Shortest Path Problem (TDSPP). By finding a solution for the TDSPP, the second research question is answered. Research question 2 is:

What is the best method for time-dependent route optimisations?

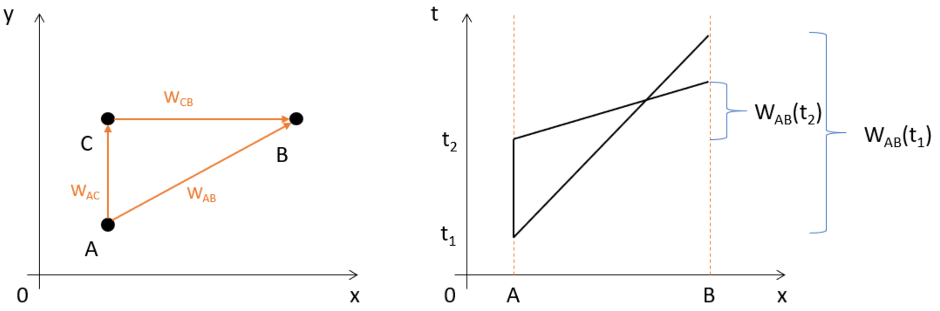
The TDSPP can be considered by changing the weights of the arcs from a constant value to a time series (Nannicini et al., 2009). In the algorithm, the weight of the arc can be determined by taking the correct weight for the departure time of that vertex from the time series. An important note is that for the TDSPP 3D graphs do not have to be considered. A logical thought would be that the TDSPP can be solved by adding a time dimension to a 2D graph resulting in a 3D graph. However, this method gives rise to several complications with discretising the velocity and makes the TDSPP algorithms much slower. Therefore in order to keep the computation time low, a 2D graph is considered.

When the weights of the arcs are defined as a time series instead of as a single value, the time-dependent shortest path problem is considered. The time-dependent shortest path problem (TDSPP) can be solved with a variant on the Dijkstra (Yu and Qin, 2008) and A* (El-Sherbeny, 2014) algorithms. These algorithms are very generic and can be used in most cases. However, there are a few conditions of correctness for the algorithms. When these conditions of correctness are satisfied, the algorithm is proven to give the correct solution. The time-dependent A* algorithm has the following conditions of correctness:

- Non negative arc weights;
- FIFO property: For all vertex $v \in V$ and $t_1 \leq t_2$, $t_1 + W_{AB}(t_1) \leq t_2 + W_{AB}(t_2)$
- Triangle inequality: For all edges $e(A,C) \in E$, $W_{AB}(t) \leq W_{AC}(t) + W_{CB}(t)$
- Time window condition: For all vertex $v \in V$ and $t_1 \leq t_2$, $[a_{v1}, b_{v2}] \leq [a_{v2}, b_{v2}]$

The time-dependent Dijkstra algorithm has the following conditions of correctness:

- Non negative arc weights;
- FIFO property: For all vertex $v \in V$ and $t_1 \leq t_2$, $t_1 + W_e(t_1) \leq t_2 + W_e(t_2)$



Triangle inequality:
$W_{AB} \leq W_{AC} + W_{CB}$

FIFO property:
$t_1 + W_{AB}(t_1) \leq t_2 + W_{AB}(t_2)$

Figure 3.3: Triangle inequality and the FIFO property explained with an simple example. The triangle inequality is on a length scale while the FIFO property is on a time scale.

The FIFO condition implies that waiting for tides is never faster than sailing away directly (Yu and Qin, 2008). The triangle inequality implies that travelling around something does not result in a lower cost than travelling in a straight line (see figure 3.3). The graphs for which the FIFO and the triangle inequality hold are so-called FIFO graphs. When Dijkstra or A* are used for non-FIFO graphs, the solution can get stuck in a local minimum.

The FIFO property of a graph can be specified per edge. The graph is called a FIFO graph if all the edges have the FIFO property. When a edge does not comply with the FIFO property it can be transformed to a FIFO edge (Yong-liu and Aiguang-yang, 2007). Suppose that W_{ij} is the weight function of the arc $(v_i,\ v_j)$. According to the definition of the FIFO property an arc is FIFO if $t_1 + W_{ij}(t_1) < t_2 + W_{ij}(t_2)$. For the $\lim_{t \to 0} \frac{W_{ij}(t_2) - W_{ij}(t_1)}{t_2 - t_2}$ This can be rewritten to $\frac{dW}{dt} < -1$. For non-FIFO edges this condition will be exceeded for at least one time interval. The time interval Ω is the waiting interval of extreme points (t_s, t_m), see figure 3.4. Note that this is not the same as the interval for witch $\frac{dW}{dt} < -1$ holds.

A non-FIFO edge can be transformed into a FIFO edge with equation 3.3. In figure 3.4, an example of a non-FIFO edge that is converted to a FIFO edge is shown. In the left figure, the travel time is plotted against the departure time and on the right, the arrival time is plotted against the departure time. It is visible that in the time interval Ω it is better to wait until t_m. A horizontal line represents this in the right plot and the left plot as a slope of -1.

$$W' = \begin{cases} W_{ij}(t) & \text{if } t \notin \Omega \\ t_m + W_{ij}(t_m) - t & \text{if} t \in \Omega \end{cases} \tag{3.3}$$

Now that the graph satisfies the FIFO condition and the non-negative arc weights condition, the graph complies with all the conditions of correctness for the TDSPP Dijkstra's algorithm. Note that the conditions of correctness for the A* algorithm are not satisfied. Unfortunately, the triangle inequality is complicated to satisfy for the considered problem of this study. For this reason, the A* algorithm is not used in this study. All the shortest path computations are executed with the time-dependent Dijkstra algorithm.

3.1.3. Cost functions and variable shipping speed

This section describes the different cost functions and restrictions that transform the mathematical solution for graph theory into a practical solution for hydrodynamic route optimisation. In order to optimise for the cheapest and least pollutant path, the shipping speed should be considered as a variable instead of as a constant. This is done by adding an extra dimension to the nodes. Meshing the nodes to a TIN grid with Delauney edges results in a 1D list of nodes and a list with for each node a list of edges. Each node can be specified with a single index that refers to the list of nodes from which the (Lat, Lon) values can be read. With this index, the edges of the node can be requested from the list of edges.

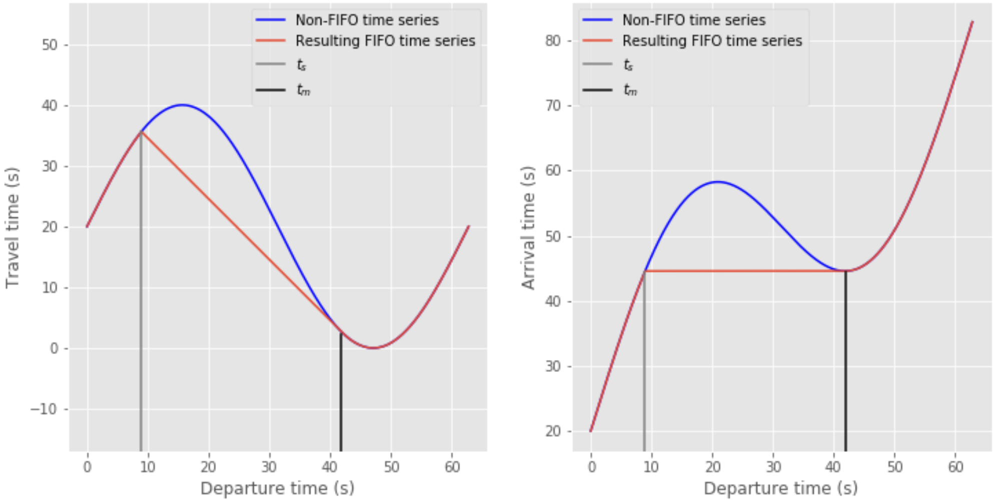


Figure 3.4: Transforming a non-FIFO Time series of an edge into a FIFO time series. In the left figure, the travel time is plotted against the departure time and on the right, the Arrival time is plotted against the departure time. It is visible that in the time interval Ω it is better to wait until t_m. A horizontal line represents this in the right plot and the left plot as a slope of -1.

In order to make the shipping speed variable, the nodes are now described by two indices. The first index refers to the list of (Lat, Lon) values and the second index refers to a list of shipping velocities. The shipping speed is determined by the second index of the destination node. For example with $v_{ship} = [3,5,7]$ the shipping speed on the edge $([2305,2],[110,1])$ is 5 m/s and the shipping speed on the edge $([421,2],[1120,0])$ is 3 m/s (Note that indices start with 0 in python). A critical note is placed by the number of steps in the discretisation of the shipping velocity. The computation time scales with the square of the number of steps in the shipping velocities. A large number of different possible shipping velocities result in considerable computational time and a sizeable preprocessing file.

For a node with a certain location index and shipping speed index, the node can reach all the all other shipping velocities for the destination node. For example, for the edge from (spatial) node 120 to 140, and $v_{ship} = [3,5,7]$, all the edges between these nodes are:

- $([120,0],[140,0])$, $([120,0],[140,1])$, $([120,0],[140,2])$
- $([120,1],[140,0])$, $([120,1],[140,1])$, $([120,1],[140,2])$
- $([120,2],[140,0])$, $([120,2],[140,1])$, $([120,2],[140,2])$

In this study, there are four cost functions defined. These four cost functions result in optimisation for the shortest path, the fastest path, the cheapest path, and the least pollutant path. For the optimisation for the cheapest path and the least pollutant path, the number of steps in the discretisation of the shipping velocity must be at least two. With these four cost functions, most optimisations can be performed. Below the details of each cost function are described.

The fastest route can be calculated when the weights of the arcs are the travel time between the nodes. The travel time is equal to the velocity divided by the distance. The distance between two nodes can be calculated with the haversine equation (de Mendoza, 1795), and the velocity can be calculated with equation 2.4. The cost function returns an infinite travel time when one of two conditions are met. These checks are done for all the cost functions. The first condition is that both the start and destination node cannot be masked. When the start or destination nodes are masked, there is not sufficient water depth. The second condition is that the ship has enough installed power to sail against the current. This gives $n_s > v_{max}$, where v_{max} is the ship's velocity over land and n_s is the of the flow velocity perpendicular to the course of the ship (see equation 2.4).

Finally, the time series is transformed into FIFO edges with the method described in subsection 3.1.2.

The shortest route can be calculated when the weights of the arcs are the distance between the nodes. The distance between two nodes can be calculated with the haversine equation. This cost function is the most straightforward cost function. The cost of an arc is only dependent on the distance between the start and destination point of the arc. The distance between two points in a WGS84 coordinate system can be calculated with the haversine function. The time series is transformed into FIFO edges with the method described in subsection 3.1.2.

The cheapest route can be calculated when the weights of the arcs are the costs in euros between the nodes. The function that determines the cost of sailing with a ship can vary between different ship and different companies. This function of the cost is most of the time considered as corporate secret. Therefore this function is not hard-coded in the route optimisation package. In the preprocessing calculation, this function needs to be specified. The part that is hard-coded in the route optimisation package are the dependencies of the cost. In this study, it is assumed that the cost is dependent on time and fuel consumption.

Equation 3.5 calculates the cost in order to travel from a start node to a destination node. This function consists of two parts. The first part is non-variable cost. This represents the cost per unit of time for the ship. The second part is the variable cost. When the ship has a larger power output, it burns more fuel and is, therefore, more expensive. In Barrass (2004), Holtrop; and Mennen (1982) it is found that the fuel consumption is related to the sailing velocity to the third power. For this use, all the calibration parameters are summarised in λ_f and λ_t. When this equation for the sailing cost is used, there is an optimal sailing velocity. This optimal sailing velocity is a function of λ_f and λ_t.

In equation 3.5 W is the weight of the arc in euros, t_T is the travel time in seconds, v_{max} is the velocity of the ship compared to water, λ_t is the cost of the ship per second in euros, and λ_f is the fuel rate of the ship.

$$W = f(t_T, v_{max}) = \lambda_t t_T + \lambda_f t_T v_{max}^3 \tag{3.4}$$

The least pollutant route can be calculated when the weights of the arcs are the emitted pollution of that edge. This pollution can be CO2. However, vessels emit other particulate matters as well. The function to calculate the emission per edge is shown in Equation (3.5). For this function, the emission is related to the third power of the shipping velocity (Górski et al., 2013, Molland et al., 2011). In this equation, W is the weight of the arc in emission terms, t_T is the travel time in seconds, and v_{max} is the velocity of the ship compared to water.

$$W = f(t_T, v_{max}) = t_T v_{max}^3 \tag{3.5}$$

3.1.4. Modular package implementation

An important objective of this study is to make the results general applicable. In this study, it is chosen to do this by developing an open-source python tool that contains the route optimisation method. Since OpenCLSim was written in python, it was chosen to write HALEM in python as well to make the mix in more fluently. With the development of this modular tool, the third research question is answered. The third research question is:

> *How can the route planner be implemented in a modular package so that it can be used in different cases?*

In order to make the python code of this study in a modular package, a few features were included in the python code. The first is the Git version management (Chacon and Straub, 2014). The python code is gathered in a repository, and the repository is managed in GitHub [1]. GitHub offers the possibility to make backups of the software. It also is a great platform for corporations between multiple authors. GitHub offers the possibility for a third party branch of the master. In this branch, new features can be added, and the code can be changed. When the third party is done, it can create a pull request so that the changes are implemented in the master branch. To make this repository accessible for everyone the package is published on the Python Package Index [2] (PyPI). The releases of the package are given a Digital Object Identifier (DOI) (Freeman and Freeman, 2013, Vernon, 2010). This is done via Zenodo [3]. To make the package more accessible for other user

---

[1] https://github.com/TUDelft-CITG/Route_optimization_in_dynamic_currents

[2] https://pypi.org/project/halem/

[3] https://zenodo.org/record/3363005.XUwXEegza70

documentation is written. This documentation is published on ReadTheDocs [4]. To guarantee the quality of HALEM each commit on GitHub is automatically tested for bugs with CirclCI [5]. In the repository, test functions are defined (Bader, 2018). These test functions are designed in such a way that it tests for bugs. In CircleCI a docker is build that runs the test for each commit on Github.

3.2. Methods for model application

After the model is developed, the usability of the model is demonstrated in a case study. For this case study, the routes of a dredging project are optimised with the HALEM package. The goal of this case study is to see how much influence the currents have on the route and how much the optimal route deviates from the standard route.

3.2.1. Implementation of HALEM in OpenCLSim

For this research, the practical case study is simulated with OpenCLSim. In order to make OpenCLSim compatible with HALEM, a mix-in has to be created. This mix-in is realised in 3 steps. These steps are explained below.

First, OpenCLSim needs to know that the user wants to optimise the route. This needs to be specified in the user interface. For now, the user interface of the OpenCLSim project is a notebook. In the user interface the roadmap is first loaded from a file and then added to the environment with $my\_env.Roadmap = Roadmap$. Besides adding the roadmap to the environment, the movable object needs to have the instance Routable. In this instance the value for $optimize\_route$ needs to be *True* (Default value *False*). In this instance, the optimisation type needs to be specified with the options: time, space, cost and co2 (default time). With these two actions, the OpenCLSim software knows that it must optimise the route. Secondly, in the function $get\_distance$ the route between the start and destination location is determined. This is done based on the one-line function in HALEM and the given optimisation type. It is first checked if the route must be optimised. If the route must be optimised the one-line function is used, otherwise the route is a straight line between start and end. Third, the object is moved from the start location to the destination. If the object has the class movable the object is moved according to the optimised path.

Due to the restriction in water depth, the optimal sailing route that results from HALEM is dependent on the draft of the vessel. Since the draft of the vessel is related to the load factor ($r = \frac{W_{ship} - W_{empty}}{W_{full} - W_{empty}}$) it is safe to say that the optimal route is dependent on the load factor. Sailing with a higher load factor can result in a longer route than sailing with a lower load factor. However, sailing with a low load factor can decrease production since less volume is displaced. This trade-off between sailing time and load factor can be optimised. This section explains the method that is used for optimising the load factor in OpenCLSim in combination with the HALEM packages. This method is then implemented in the mix in that is created for OpenCLSim. When an array of load factors is specified in the user interface, this load optimisation function becomes active.

The optimisation of the load factor is done with a simple brute force calculation. For each load factor in the list of load factors, the duration of the dredging cycle is calculated. Then the production of the dredging cycle is calculated by multiplying the load factor by the hopper capacity and dividing it by the duration of the dredging cycle. The load factor for which the product is the highest is chosen as the optimal load factor.

3.2.2. Generalisation of the flow model: Tidal analysis

In order to optimise the shipping routes based on the hydrodynamic conditions, a flow model is needed. However, most projects are planned far in advance. Hydrodynamic models typically give a prediction time of up to 10 days (for example the DCSMv6-ZUNOv4 model has a prediction time of 3 days). This is much less than is needed in order to plan ahead of time. In order to be able to use the model for planning purposes, the flow model is generalised. With a generalised flow model simulations can be done further in advance, and the model is no longer dependent on the prediction time of the hydrodynamic models by a generalisation of the hydrodynamic model. Since the tidal components of the hydrodynamic models are highly regular, an easy to predict the generalisation is based on the tide. The wind-driven behaviour and the density-driven behaviour

---

[4] https://halem.readthedocs.io/en/latest/

[5] https://circleci.com/gh/TUDelft-CITG/Route_optimization_in_dynamic_currents/tree/master

of the hydrodynamic models are neglected for the generalisation, and the tidal signal is filtered from the total signal. This is done with a tidal analysis that applies a Fourier analysis on the signal and filters the tidal constituents from the total signal. In this research, the python package Utides is used, since it is available (open source), complete, validated, and a reliable package (Bowman, 2019). For each node of the hydrodynamic model, a tidal analysis is performed for the water level, u, and v flow velocities, this generalised signal is then saved in a new file. The results of this tidal analysis and the used tidal constituents can be found in Appendix B.

With the results of a tidal analysis, the water level and flow velocity can be reproduced for every moment in time. This means that for each moment in the future, no matter how far away, an estimation of the water levels and flow velocities can be determined. However, it is not possible to create a roadmap that is always valid. Since the transformation from flow field to weights of the edge is such a computation-intensive calculation, this must be preprocessed. Therefore the time period of the roadmap is chosen in such a way that indefinitely repeating the roadmap gives a good approximation for the hydrodynamic conditions.

The M2 tide has a period of 12 hours and 25.2 minutes (12.42 hours) And the S2 tide has a period of 12 hours exactly. The combination of these two signals can be rewritten as a carrier wave and a signal wave, see equation 3.6. In order to be able to repeat the roadmap, the roadmap must have the exact length of the period of the carrier wave. For the M2 S2 interdependency, this results in a period of the carrier wave of 355 hours (14.8 days), see equation 3.7. In which, T_{M2} is the period of the M2 tide and T_{S2} is the period of the S2 tide and T_c is the period of the carrier wave. It should be noted that indefinitely repeating the roadmap is not an exact solution of the tidal flows. Due to other tidal constituents than just the S2 and M2 tide, the spring tide neap tide variations will vary from month to month. This is not accounted for when the roadmap is repeated indefinitely. However, the errors that result from this are small in comparison with the signal and are therefore accepted.

$$sin(ax) + sin(bx) = 2\, sin\left(\frac{1}{2}(a+b)\, x\right)\, cos\left(\frac{1}{2}(a-b)\, x\right) \tag{3.6}$$

$$\begin{aligned} \frac{1}{T_c} &= \frac{1}{T_{M2}} - \frac{1}{T_{S2}} \\ T_C &= \frac{1}{\frac{1}{12.42} - \frac{1}{12}} = 355 hours \end{aligned} \tag{3.7}$$

The duration of a tidal observation — generally called the observation "length" — will vary from case to case. This means that the resolvability of independent constituents, each having its fixed frequency, varies from situation to situation as well. For how many constituents must be solved can be determined with the Rayleigh criteria. The definition of the Rayleigh criteria is given in equation 3.8, Foreman and Henry (1989). In which T is the length of the observed period, Rey is the Rayleigh parameter, T_{M2} is the period of the M2 tide and T_{S2} is the period of the S2 tide. For this analysis, the length of the observed time series is 840 hours (35 days), and the period of the carrier wave is 355 hours. This means that $R \le \frac{T}{T_c} \le 2.36 = 2.3$ .For this analysis, the length of the observed time series is 840 hours (35 days), and the period of the carrier wave is 355 hours. this means that $Rey \le \frac{T}{T_c} \le 2.36 = 2.3$.

$$\left(\frac{1}{T_{M2}} - \frac{1}{T_{S2}}\right) T \ge Rey \tag{3.8}$$

For the tidal decomposition, a python package called Utides is used. This packages can decompose an h, u, v time series into tidal constituents and rebuild the astronomical tide from these constituents. In table 3.1 the settings that were used in the tidal analysis calculations are displayed.

Table 3.1: Used parameters in the tidal analysis calculation. The tidal analysis calculations are performed with the Utides package in python.

Start of original time series	08 – 05 – 2018 00 : 00 : 00
End of original time series	10 – 06 – 2018 23 : 50 : 00
Time step of the original time series	10 min
Start of tidal analysis time series	09 – 05 – 2018 23 : 00 : 00
End of tidal analysis time series	24 – 05 – 2018 23 : 00 : 00
Time step of the tidal analysis time series	60 min
Latitude	53 degrees;
nodal	$False$
trend	$False$
method	ols
conf_int	$linear$
Rayleigh_min	2.3

3.2.3. case study Schouwen Westkop Noord

The case study that is done in this graduation research is the beach nourishment at Schouwen Westkop Noord (near the Oosterscheldekering in Zeeland). This project is chosen out of several available case studies such as Windpark Fryland, beach nourishment Ameland, Project Afsluitdijk or Project Fhermanbelt. This project was chosen over the other projects because of two reasons. First, the OpenCLSim modelling of this project is straightforward. It is just one vessel, one source location and one dump location. A project such as Afsluitdijk, Fhermanbelt of Windpark Fryland included much more vessels, more source locations and more dump locations. This would be much more difficult to simulate in Openclsim and would make the results of the case study more difficult to interpret. Second, the route was not a straight line between the start and destinations. Due to the bathymetry of this location, it is not possible to sail in a straight line between the start and destination. Since the vessel must sail around a shallow part, there is much more to optimise, than compared to the other projects. By working out this case study research question, 4 is answered. Research question 4 is:

> *How can dredging projects be optimised for the given route optimisation package and OpenCLSim?*

For all simulations holds that they are not an exact copy of reality. Firstly, the parameter settings of the project and the equipment are realistic, but not of real equipment. Secondly, the simulations do not cover every part of the project. These simulations do not take into account events such as crew change, fuelling, delays (scheduled and not-scheduled), mobilisation, and demobilisation. Therefore these simulations give realistic indications for the influence of route optimisation, but cannot be used as a project estimation.

In Figure 3.5, the bathymetry of the case study is displayed. In this figure water depth at low water is displayed. From this figure, it becomes clear that the pump location is hard to reach for the given tide. The route that maximises the minimal water depth is much longer than the shortest route. This route goes through a gully along the coast of Schouwen-Duiveland. This is a natural gully caused by the large tidal flow induced by the Oosterscheldekering.

In order to simulate the project, the OpenCLSim python package is used. This software package was chosen since van Oord is part of the development of this package and because it is an excellent platform to use the route optimisation function. What this package does and how it does it is further explained is Chapter 1. In order to use the HALEM package in the Openclsim package, a mix-in needs to be created. With this mix-in, the user can specify whether it wants to optimise the route, for which cost function it wants to optimise, and so on. Creating this mix-in gives partly answer to research question 3. This mix also optimises the load factor width the route optimisation. With this optimisation, the load factor can dynamically change in time. For this case study, the optimisation of the load factor is very important since the project is so limited by bathymetry.

The project is first worked out with the current methodology to quantify the improvement over the old method. The modelling for this base case is performed with the OpenCLSim software without the HALEM package. This base case gives a good view of the challenges of these projects, and the solutions that are provided by the HALEM package. The modelling of this project is done with a specified route for the hopper. The route that is

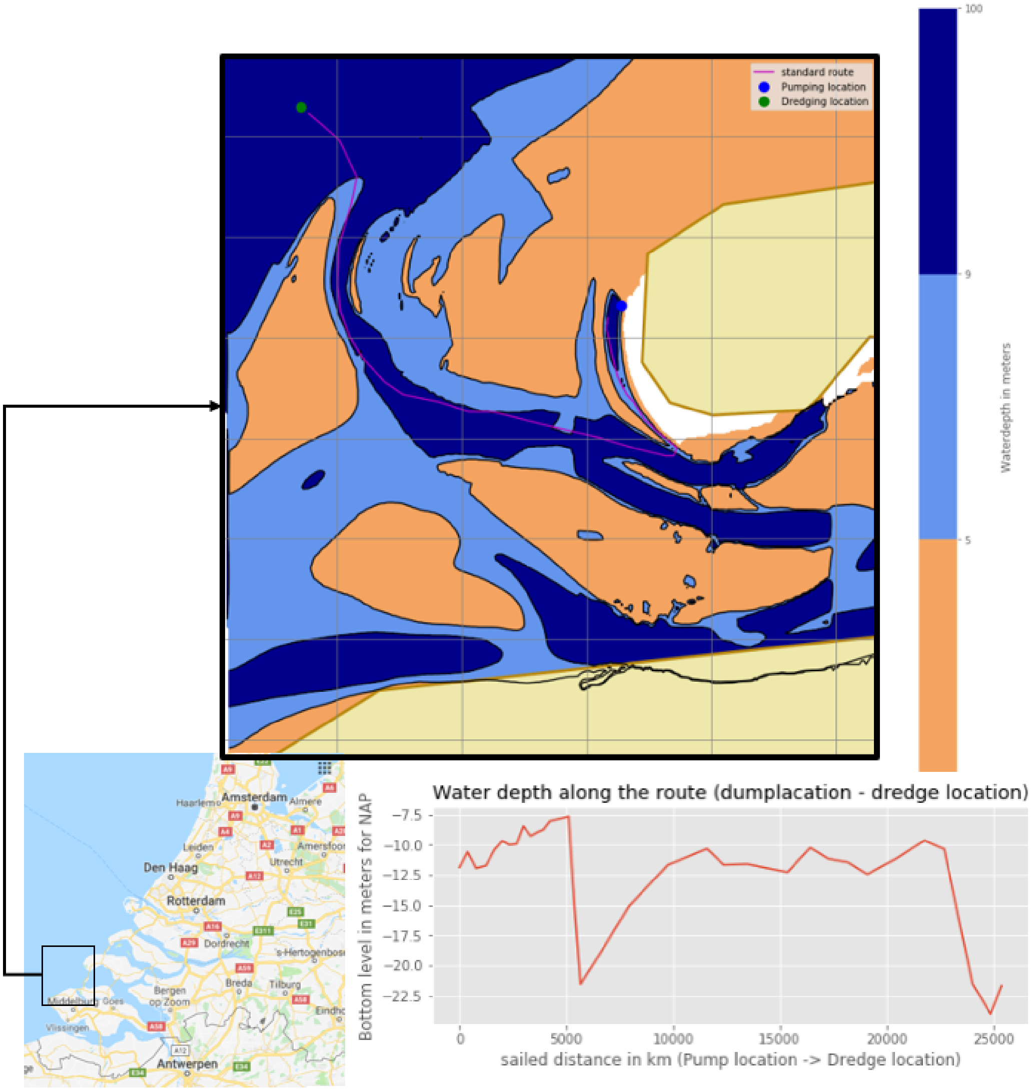


Figure 3.5: In this figure, the bathymetry of the case study is displayed. In the upper figure shows the water depth of the area for a water level at mean low water (NAP -1.3 m). The dark blue areas the water depth is larger than the draft full + ukc of the vessel. The light blue areas are only accessible when the ship is empty, and the orange areas are never accessible for the vessel. The pump and dredge locations are indicated with blue and green dots. The standard route is indicated with a magenta line. The lower figure displays the bathymetry of the standard route from pump location to dredge location.

specified is the route the maximises the minimal water depth of the route. In the right figure of Figure 5.5 the specified route and the bathymetry of the area are displayed. In the lower-left figure of Figure 5.5 the water depth during the route is given (from pump location to dredge location). This route utilised the narrow gully along the coast of Shouwen.

The vessel has a depth restriction to simulate the influence of the dynamically changing bathymetry on the project. The lowest point of the bathymetry is at the start of the gully along the coast of Shouwen (The most right point of the route), the bed level is here -7.86 below NAP. At this point, the water level (based on the tidal analysis of the Zuno model) is given to the model with the OpenCLSim mix-in *core.HasWeather*. The minimal needed water depth of the vessel is assigned to the simulation with the mix-in *core.HasDepthrestriction* and is based on the load factor of the vessel. When a depth restriction and hydrodynamic conditions are specified in OpenCLSim, the software optimises the load factor of each dredging cycle by maximising the production of that cycle. The maximal production is a trade-off between waiting for the tide and reducing the load factor.

For the route optimisations of the case study, the HALEM package is used instead of *core.HasDepthrestriction* and *core.HasWeather*. Since the route is optimised by the HALEM package the route is no longer given as an input value, therefore only the start and stop location is given. For the case study the routes are optimised for the path, sailing velocity, and the load factor, these optimisations are based on a time-dependent flow, and water level field instead of a single point.

To asses the influence of the different hydrodynamic models on the route, five different road maps are made. All the are the same except for the source of the hydrodynamic data. The first four road maps are generated for the four different hydrodynamic models that are described in Section 2.3. The last roadmap is generated for the hydrodynamic data from the tidal analysis. The tidal analysis that is done is for the DCSM-Zuno model. The DCSM-Zuno model is chosen for the tidal analysis since it is the most accessible model for practical use.

4 | Model validation

This chapter describes the model validation of HALEM. This chapter contributes to the overall credibility of HALEM but is not necessary for the story of this thesis. This chapter consists of a method of validation, results of the validation and the discussion of the validation.

4.1. Methods for model validation

Chapter 3.1 describes the theory behind the python functions. However, to know that the method works, the method needs to be validated. By validating the model, it is found whether the model is implemented correctly and whether it works correctly. The validation of the model is split up in two parts; the first part checks if the model is correctly implemented, and the second part checks if the model gives correct results. To check if the model is correctly implemented the model is tested for some trivial test cases. With these test cases, for which the outcome is known beforehand, it is checked if the model works according to the expectations. The outcome of the model is compared to an executed project to check if the model gives correct results.

4.1.1. Model validation with trivial test cases

To test if the route optimisation function works according to the expectations the toolbox is tested with trivial test cases. These test cases are not a real situation that can occur in the real world but are cases that test the limits of the route optimisation toolbox. For these test cases, the exact solution or the expected result is known, with this, the toolbox can be tested if it behaves according to the expectations.

The first and most simple case is to find the fastest route for a potential flow field. In Equation (4.1), the equations that describe the flow are displayed. In this equation, x_0 and y_0 are the coordinates for the lower-left corner, and L and B are the sizes of the domain. These equations result in a simple rotating flow around the centre of the domain. For this flow common sense dictates that the fastest shipping route should be a sinus between the start and endpoint in its first Fourier mode.

$$\begin{cases} u = cos(\frac{\pi(x-x_0)}{L}) \\ v = -cos(\frac{\pi(y-y_0)}{B}) \end{cases} \tag{4.1}$$

The second test case is to find the fastest route through a rotation flow around the centre of the domain with a tidal component in the time. In Equation (4.2) the equations that describe the flow are displayed. In this equation, x_0, y_0, t_0 are the coordinates for the lower-left corner and L, B and T is the size of the domain. The expected result should be a sinus between the start and endpoint in a higher Fourier mode, dependent on the tidal period of the flow.

$$\begin{cases} u = cos(\frac{\pi(x-x_0)}{L}) \times sin(2\pi\frac{t-t_0}{T}) \\ v = -cos(\frac{\pi(y-y_0)}{B}) \times sin(2\pi\frac{t-t_0}{T}) \end{cases} \tag{4.2}$$

Another test case for the route optimisation is waiting for an obstacle. With this test case, the solution to the FIFO problem is tested. In this test case, this is simulated with a case without currents and a uniform water depth of 20 meters. Except for the first 5 hours, then the water depth in the area for $x \in [3,4]$ and $y \in [50.4,50.6]$ is 0 meters. The fastest route is to wait for the obstacle to disappear (after 5 hours) and then to sail. Sailing around the obstacle takes in this example more time. When the solution for the FIFO problem works, the solution waits and travels in a straight line when the solutions for the FIFO property fails; the solution will travel around the obstacle.

4.1.2. Model validation with a real-life test case

Section 3.1 discusses the mathematical correctness of the model. Chapter 4.1 discusses the correctness of the implementation of the model. In order to make the validation of the model complete, the model should as well be tested for applicability. This section describes the validation of the applicability of the model. To do so, it is tested how good of a fit the model is compared to reality. If the model has a good fit with reality, the model is applicable.

In order to compare the model with reality, the measurements of a project that is already completed are compared to the simulation of that project. For the measurements of the project AIS (Automatic Identification System) data is used. The location, sailing velocity and other parameters of a vessel are always recorded with AIS. This AIS data is saved as a time series. AIS data is open-source data. However, van Oord buys the filtered AIS database from a third party. For this study, the van Oord AIS is used because it is available.

From May the 9th 2018 to June the 12th 2018 van Oord executed beach nourishment at Texel. For this project van Oord used the Geopotes 15. This is a Trailing Suction Hopper Dredger with a hopper capacity of 9,930 m^3 and a length overall of 133.54 m. This ship made 184 dredging cycles to complete the project. The project has a dredging location relatively close to the project site. The beach nourishment is performed by pumping the liquidised sand in a pipe to the shore. At the project site, there are two connection points. After a section of the beach, nourishment is completed for the specific beach section an extra pipe is coupled to the existing pipe to extend the reach of the hopper and to nourish another part of the beach. By adding an extra pipe, the resistance increased, and the pumping takes longer.

The dredging cycle is discretised in four steps: dredging, sailing full, pumping, and sailing empty. With a python script, the time per activity and the sailing velocities can be extracted from the AIS data. In the script, two polygons are defined. The first polygon is a polygon around the dredging location and the second polygon is a polygon around the pumping location. If the vessel is in the dredging polygon it is dredging, if the vessel is in the pumping polygon, it is pumping, and if it is not in a polygon, it is either sailing full or sailing empty. If the previous activity was dredging it is sailing full otherwise it is sailing empty. With these definitions, the time spent on each activity and the sailing full and sailing empty velocities can be determined. The project is simulated by simulating each sailing activity separately. This is done with the HALEM package and a simple for loop. A specific start point (Lon, Lat) endpoint (Lon, Lat), and time of departure is extracted from the AIS data. This results in 184 start points for sailing full, 184 start points for sailing empty, 184 departure times, 184 sailing full velocities, and 184 sailing empty velocities. For each of these 368 routes, the optimal shipping route is calculated using HALEM. The input shipping velocities that are used in HALEM are the averaged sailing speed full over the entire project, and the averaged sailing empty full over the entire project. This results in $184 \times 2 = 368$ calculated sailing velocities.

The simulated project and the actual project are compared with a spread plot. On the x-axis, the measured velocities are plotted, and on the y-axis, the calculated velocities are plotted. For the perfect model, the fitted line should have a slope of 1 an r^2 value of 1 and an intercept of 0. The slope is the result of the fit of the points to a linear regression (minimising for r^2), and the r^2 value is how good the line the points describes. An important note is that the slope does not have to be 1 for the r^2 to become 1.

The hydrodynamic model that is used for this validation case is the DCSMv6-ZUNOv4 kf model. This is done because it is the most available model. Other models were not available for the specific domain and time period of this specific project. This model is a 2DH model that does take wind forcing into account. A disadvantage of this model is that it returns a depth-averaged current instead of surface currents. The model that is used is a 2DH model and does not account for variations of the flow velocity over depth. The output of this model is, therefore, a depth-averaged result. Since the flow velocity at the bed must be zero due to the no-slip condition. The surface flow velocity is, in most cases, larger than the depth-averaged velocity. There is, however, no proper way to convert the depth-averaged flow to the surface flow. Most of the time this conversion is done with a logarithmic profile, which results in a factor of about 1.3, this is, however, most of the time a choice for the lesser of two evils. Chapter 2.2.4 explains the challenges of transforming the depth-averaged currents into surface currents. In this study, there are three different simulations done for three different surface compensation factors. The three surface compensation factors are 1, 1.3 and one where the slope of the spread plot is equal to 1. The value for the surface compensation factor for which the slope of the spread plot is equal to one is calculated iteratively.

4.2. Results of the model validation

The validation of the model is separated into two parts. The first part of the model validation is the trivial test cases, and the second part is the comparison to an executed project. In chapter 4.1, the method of the first part of the model validation is explained. This section describes the results of the first part of the model validation. Chapter 4.1 describes three different test cases for which the outcome is known. With these three test cases, the model is validated whether it does what is expected. In chapter 3.2 the mathematical correctness of the model is elaborated. These trivial test cases are therefore not intended to check the mathematical correctness of the model, but these cases test if the model is implemented correctly.

4.2.1. Model validation with trivial test cases

Chapter 4.1 names three test cases. The first is a simple rotating flow around the centre of the domain. The second is a time-dependent rotation flow around the centre of the domain with a tidal component. And the third is a test case for flooding and drying, in which the vessel should wait for a bank to flood again. These three test cases test the implementation of the model.

In figure 4.1, the results of the three simulations are displayed. The top three plots of figure 4.1 show the top-view in latitude and longitude. These figures show the optimised route, the start and endpoints, the nodes of the graph, and the hydrodynamic conditions at $t = t_0$. The hydrodynamic conditions are displayed in two parts. The quiver plot shows the direction of the flow (Note that it does not represent the magnitude correctly due to scaling), And the contour plot represents the water depth. The lower three graphs represent the s/t diagram of the route. In these figures s is the sailed distance since the start of the route and t is the time in

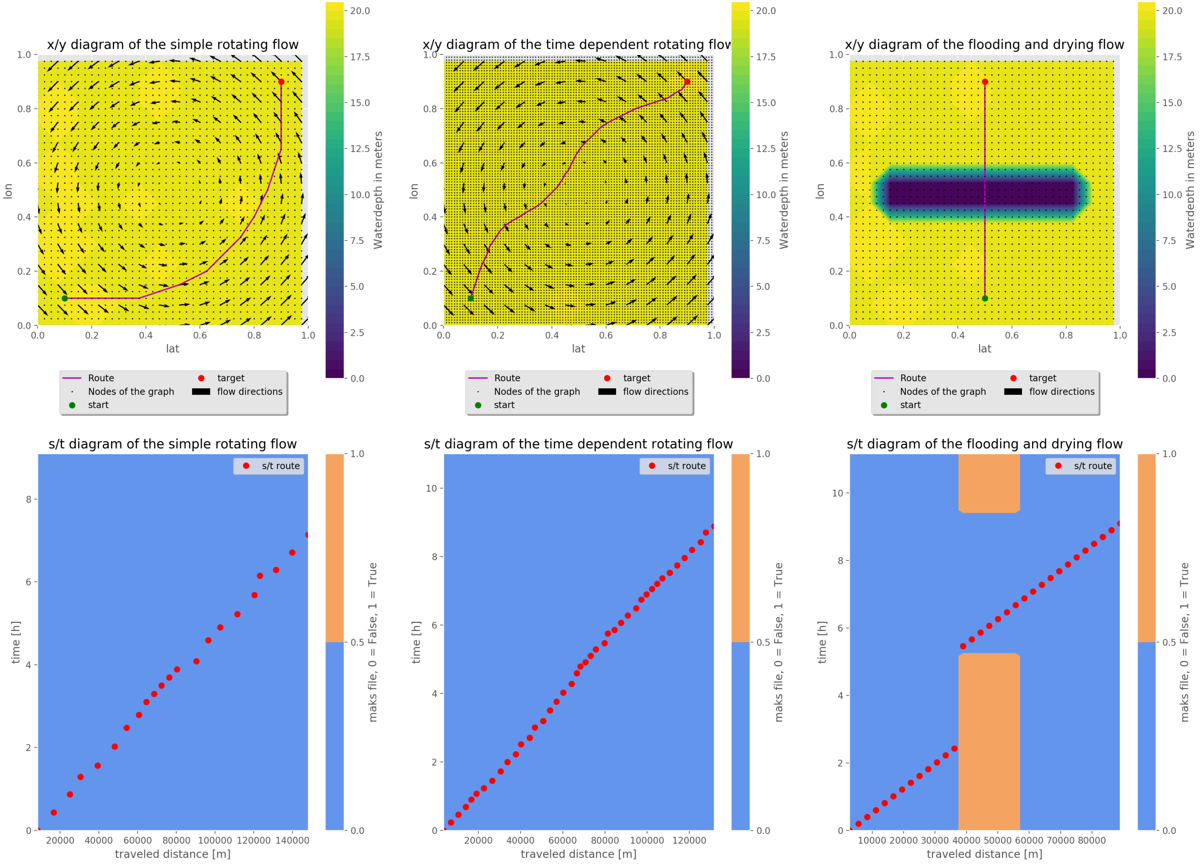


Figure 4.1: This figure displays the three test cases that are described in chapter 4.1. The top three plots show the top-view in latitude and longitude. The lower three graphs represent the s/t diagram of the route. The most left plots display the results for the simple rotating flow, the centre plots show the results time-dependent rotating flow, and the right plots show the results for the flooding and drying test case.

hours since the start of the route. In these figures, the red dots represent the nodes of the route. The contour plot represents the mask file of the roadmap. The mask file indicates if the vessel can sail over that edge. If it is possible the mask array for that edge and time is equal to *False*, if it is not possible to sail over that edge the mask array for that edge and time is equal to *True*. The contour plot of this masked array shows, therefore, the obstacles that the route faces in its path. The contour plot displays the barriers that it has to wait for as well as the deadlines for the route to be possible. For the example, for the flooding and drying case (lower right plot of figure 4.1) the vessel has to wait for the bank to flood. In the contour plot, this is displayed as a column starting at the bottom ($t = t_0$) of the plot. However, the vessel has a deadline as well. It has to sail over the bank before it dries again. This is displayed in the contour plot with a column starting at the top ($t = t_e$) of the plot.

The results of the test cases show the expected route. In chapter 4.1 the expected results where defined as a sinus in its first Fourier mode for the simple rotational flow, sinus in its higher Fourier mode, dependent on the period of the tide, for the time-dependent rotational flow, and a route that waits for the bank to flood for the flooding and drying case. An important detail is that the simulations are run with several neighbouring layers parameter (see chapter 3.1) of 3,3,1 for respectively the simple rotating flow, the time-dependent rotating flow and the flooding and drying flow. From figure 4.1 it is visible that this parameter influences the number of possible directions (displayed in figure 3.2). The number of neighbouring layers parameter for the flooding and drying test case is set to 1 since the route is a straight line. The calculation with the time-dependent rotating flow is done with a grid size that is 2.5 times smaller than the grid size of the simple rotating flow and the flooding and drying flow. For this particular test case, the results did not show the expected behaviour for a larges grid size. This is due to the more detailed route of this test case. For the other test cases increasing the grid size does not result in a better result.

4.2.2. Model validation with a real test case

By comparing the measurements of an executed project to a simulated project with the HALEM package, the applicability of the model is validated. Chapter 4.1.2 the methods for this validation is described. This section describes the results that follow from this validation. First, the results of the AIS data interpretation are presented, and from this AIS data the comparison between the measurements and the simulations are presented.

The dredging cycle is discretised into four parts. Dredging, sailing full, unloading (for this project this is pumping), and sailing empty. With a python script, these parts of the project are separated. For each dredging cycle, the time spent on the activity and the sailing velocity per sailing activity is determined. Figure 4.2 shows three plots of this AIS data. The most right plots show the route sailed by the Geopotes 15. In this figure, we can see the dredging location and the pumping location. The figure shows as well that the Geopotus went twice to Den Helder for fueling or a crew change or fueling or maintenance. This figure also shows two polygons. The first polygon is a rectangle around the dredging location and the second polygon is a rectangle around the pumping location.

In Appendix C The results of the model validation with current compensation are shown. Figure 4.3 shows the results of the comparison between the measurements and the simulation without compensation for surface currents. The results of the fitted line are an r^2 value of 0.742, a slope of 0.56, and an intercept of 2.787. Another way of interpreting the r^2 value is by comparing the standard deviation of the time series of the measurement with the standard deviation of the time series of the simulation. Figure 4.3 displays a lower standard deviation for the simulated data and a higher standard deviation for the measured data. This shows that the model accounts for some of the variability but not for all the variability; in other words, the r^2 value is lower than one.

Figure C.3 shows the results of the comparison between the measurements and the simulation for compensation for surface currents factor of 1.3. This simulation shows, however, not a significant increase in the slope. Figure C.4 shows the results of the comparison between the measurements and the simulation for compensation for surface currents factor of 2.6. This is the simulation for which the compensation factor for the surface is chosen in such a way that the slope of the spread plot is 1. The value of 2.6 is found iteratively.

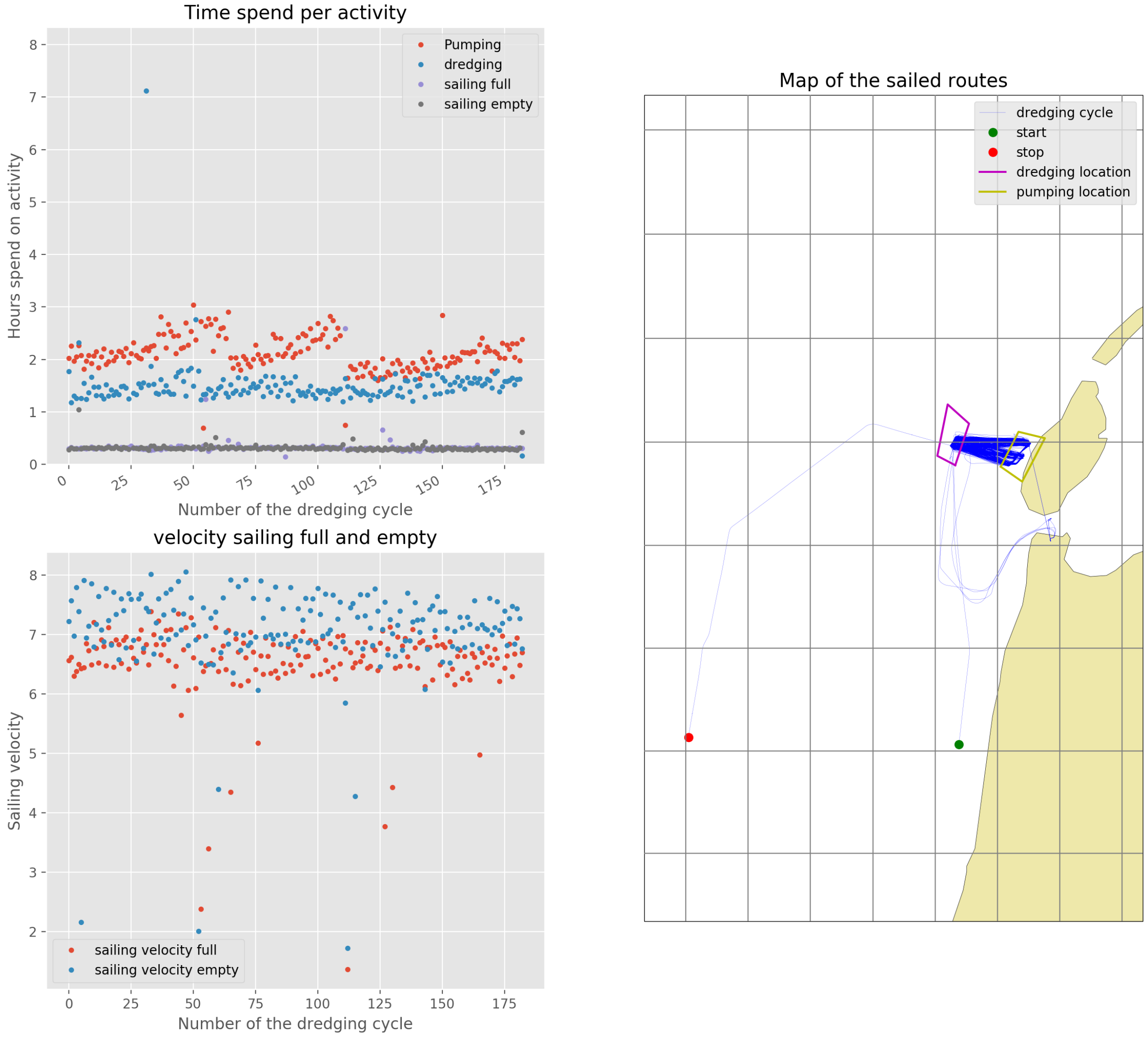


Figure 4.2: This figure displays the AIS-data analysis of the Texel beach-nourishment project. The right figure shows the route sailed by the Geopotes 15 and the left figures show the time spend per activity and the sailing velocities.

4.3. Discussion of model validation

The validation of the model is separated into two parts. The first part of the model validation is the trivial test cases, and the second part is the comparison to an executed project. By executing trivial test cases, the model implementation is validated. For each of the conditions of the correctness of Dijkstra's algorithm, a test case is constructed for which the outcome is known in advance. By performing these trivial test cases, it is checked if there are no errors in the package. In the second part, a simulation of a project is compared to the measured reality. By doing so, the fit of the model with reality is assessed.

4.3.1. Discussion of model validation with trivial test cases

In this study, three trivial test cases are executed. In chapter 4.1, the method of the first part of the model validation is explained. Chapter 4.2.1 the results of these trivial test cases are presented. With these test cases, it is tested if the model is implemented correctly in python. The first test case is the test case for a rotating flow around the centre of the domain. The second test case is the test case for a time-dependent rotating flow with a tidal component. The last test case is a test case for flooding and drying, where the FIFO condition is tested. The result displayed in figure 4.1 show that the results are the same as the expected results. From this, it is concluded that the model is implemented correctly and that there are no mistakes made in the implementation of the model in python.

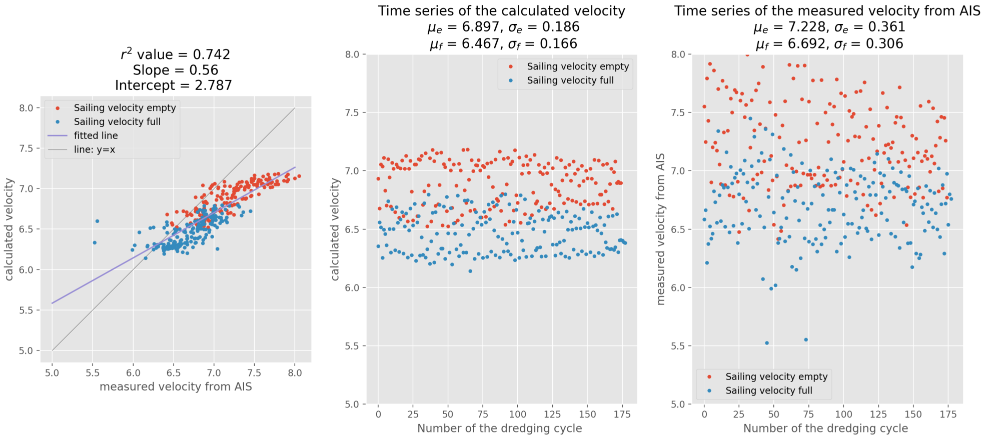


Figure 4.3: These are the results for the validation case Texel with a compensation factor based on a logarithmic profile of 1.0. The centre plot and the right plot show respectively the distribution of the calculated sailing velocity and the distribution of the measured sailing velocity. In the title of these plots, the mean values and standard deviations are given. The left figure shows the correlation between the measured and calculated velocity. The parameters that describe the correlation are displayed in the title.

More reliable validation of the model could be done with a physical experiment. The steady-state rotating flow experiment and the time-dependent rotating flow experiment could be performed in a testing facility to validate the outcome of the model. With these physical experiments, the ship movement model can better be validated. These experiments can result in better confidence intervals for the model.

4.3.2. Discussion of model validation with real-life test case

A comparison is made between the measurements of an executed project to a simulated project with the HALEM package to test the model's applicability. By comparing the measurements and the simulation, the ship movement model can be validated. In chapter 4.1.2, the method for this validation is described. Chapter 4.2.2 shows the results of this validation. This section explains the meaning of these results. In this section, the reasons why the model is not perfect are given, and it is explained when this model is a good fit or applicable.

Figure 4.3 shows the spread plot of the calculated sailing velocity versus the measured AIS velocity. For the perfect model, the fitted line should have a slope of 1 an r^2 value of 1 and an intercept of 0. The spread plot has an actual r^2 value of 0.742 and an actual slope and intercept of 0.56 and 2.787. Respectively. The slope is the result of the fit of the points to a linear regression (minimising for r^2), and the r^2 value is how good the line the points describes. An important note is that the slope does not have to be 1 for the r^2 to become 1. The reason why the r^2 value and the slope deviate from 1 is explained below.

The most strange result is that the slope of the spread is only half of what is expected. A reason for this misfit of the slope could be the difference between depth-averaged currents and surface currents. In chapter 4.2.2, the result of a simulation with a compensation factor of 1.3 and a compensation factor of 2.6 is shown. These values are chosen with the reason that 1.3 is the most commonly used factor for this transformation, and 2.6 is that value for which the resulting slope is 1. For the simulation with the compensation factor of 1.3 this results in a r^2 value of 0.719, a slope of 0.543, and a intercept of 3.125 (see figure C.3). For the simulation with the compensation factor of 2.6 this results in a r^2 value of 0.65, a slope of 0.978, and a intercept of -0.15 (see figure C.4). In the results, it is visible that the simulation with a surface current compensation factor of 1.3 does improve the slope of the spread plot. Figure C.4 shows the result for the surface current compensation factor for which the slope of the spread plot is equal to 1. Here it is found that the depth average current should be multiplied with a factor of 2.6 in order to represent the surface currents. However, this is a too big factor to be realistic. In order to get an accurate description of the surface currents, a 3D model is needed. From this, we can conclude that the 2DH model cannot be improved with a surface current compensation factor and that the misfit in the slope of the spread plot is not due to the difference between surface currents and depth-averaged currents.

Another likely reason for the misfit of the slope of the spread plot could be the shortcomings of the model. Chapter 6 explains the shortcomings of the model, and these shortcomings explain why the slope is not equal to 1. These assumptions are neglecting of wind and wave forces, the neglecting of inertial effects of the ship, and the truncation errors made in the roadmap. With these assumptions and errors, this model is not perfect. This explains why the model only can predict a part of the variability and not all of the variability. From this, it is concluded that the currents account for about 56% of the variability in the shipping velocity, but not for 100% of the variability. Another factor that is not taken into account in the simulation are the decisions of the captain. In the simulation, it is assumed that the ship always sails the fastest path to the destination with full power. However, other factors can make the captain decide not to sail with full power or not with the fastest route. This can influence the result of the slope as well as the result of the r^2 value.

The r^2 value would be considered a low r^2 value for a replica of the project; however, for this case, it is considered a good correlation. The reason why this value is low is that there are factors that are not taken into consideration for this validation case. The first factor that can account for a lower r^2 value is the chosen polygon. In order to make sure all the cycles were contained in the polygon, the polygon is quite large. This results in substantial variability in the start and stops velocities. The second reason that can account for a lower r^2 value is inertial effects. Since the sailing time is short, the inertial effects (accelerating and decelerating) are substantial. However, one of the core assumptions of the route optimisation package was that inertial effects could be neglected. In combination with the large polygon, this leads to a decrease of the r^2 value. The third reason that can account for a lower r^2 value is wind and wave forcing. These forces on the ship were not included in the model and can cause an extra variability in the shipping velocity. This will result in a lower r^2 value. The last reason that can account for a lower r^2 value are decisions of the captain and non-variable shipping speeds. For this validation case, the shipping velocity with respect to water is assumed constant; however, in reality, the captain does not sail constant with full throttle and changes its shipping velocity with time. This result in a lower r^2 value. All factors considered an r^2 value of 0.742 is large enough to consider this a good model.

5 | Results

The results of this thesis consist out of two parts. The software package HALEM is the first and most important result of this study. CrefCH:Results$_{o}f_{t}he_{m}odel_{d}evelopmentdescribesthe$HALEM$package.Secondly, therearetheresultsof th$

5.1. Results of the model development

This section explains the result of model development. The results of the model development led to a package. This package is entirely modular so that it is usable without needing other software or data. Making the package modular answers the fourth research question. The route optimisation toolbox is published as an open-source repository on Git-Hub Halem (2019). The link in figure 5.1 provides access to the repository. The name of this package is HALEM (Hydrodynamic Algorithm for Logistical Enhancement. Module). From this chapter onward the route optimisation toolbox will also be referred to as HALEM. Listing 5.1 (Halem, 2019) sows the code to install the software with python. Another way to use the package is to fork the branch and install it in editor mode.

Figure 5.1: `https://github.com/TUDelft-CITG/Route_optimization_in_dynamic_currents`

```
pip install halem
```

Listing 5.1: installation manual for installing the route optimisation package HALEM

This paragraph describes the toolbox that results from the model development. All the features the literature study and chapter three describe are processed in two python functions. The first function schematizes the continuous solution space into a graph. The second functions calculate the optimal route using this graph. This section contains three subsections. The first two subsections explain the steps of the two functions, fig. 5.2 shows a schematisation of these steps. The last section shows the results of the node reduction. The documentation website of HALEM [1] shows an example of route optimisation. This example optimises the route for a rotating flow around the centre of the domain. For this example is a fictive hydrodynamic model used so that the model runs without any other data.

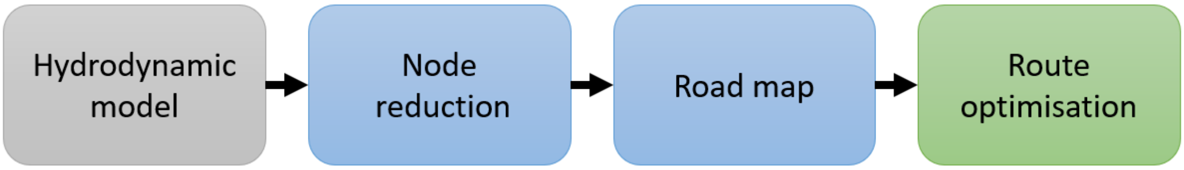


Figure 5.2: flow chard of route optimisation steps

---

[1] https://halem.readthedocs.io/en/latest/examples.html

5.1.1. Roadmap generator layout

The first function of the route optimisation package is the Road map generator. This function takes the flow model and other parameters and generates the road map. This function is also called the preprocessing function.

The first step is to load the hydrodynamic model. The road map generator function takes as input a class $Load\_flow$. For each type of flow model (DCSMv6-ZUNOv4 or DCSM-FM 0.5 nm or another flow model), a custom $Load\_flow$ class is defined. The second step is the node reduction. 3.1 describes the method for the node reduction. First, a length scale based on the curl of the flow and the magnitude of the flow is determined. Then the nodes are added if the nearest neighbour is further away than the length scale.

A large schematisation is the minimal water depth and the shipping velocity. These parameters determine for a large part if a time series satisfies the FIFO condition or not. Therefore these parameters have to be known in the preprocessing part. If these parameters are determined after the preprocessing the computation of the optimal route can take much longer (in worse cases up to a factor 100). To make a distinction between sailing full and sailing empty, multiple arrays for the shipping velocity can be specified in the preprocessing phase. A example for the input of the shipping speed is $v_{ship} = [[1,3,5],[2,4,6],[3,5,7]]$. The last step of the preprocessing phase is to calculate the time series of the weights of the edges. The algorithm generates a list of weights for each of the four cost functions (time, space, cost, co2). These lists contain several graphs that are equal to the number of arrays in the shipping velocity. For the previous example, this results in 4 lists with for each list three graphs. In the optimisation phase, the right graph can be selected based on the correct maximal shipping velocity (for this example 5,6, or 7 m/s) for that case.

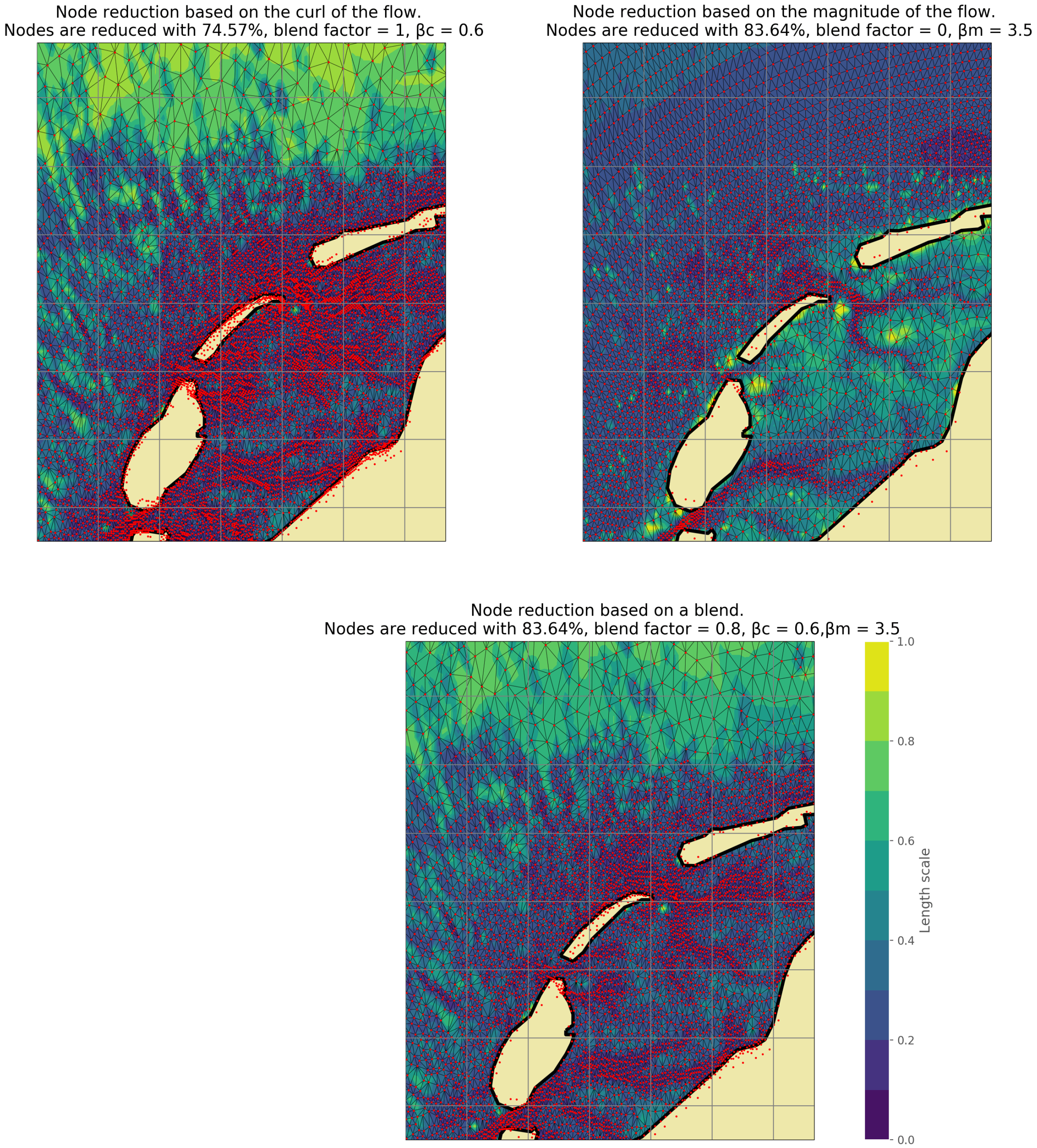
5.1.2. Route optimizer layout

The second function of the route optimisation package is the route optimiser function itself. The initialisation function of HALEM defines four one-line functions. For each cost function (time, space, cost, and co2) a function is made that calculates the optimal route for that cost function. A critical note can be placed to the start and stop node of the algorithm. Since the continuous solution space is discretised into discrete nodes, the input coordinates of the start and stop location are not the same as the coordinates of the start and stop vertex of the graph. The function uses the nearest node to the exact start location as the start node. This approximation results in a truncation error — the same holds for the end node. When the start and endpoint are determined, Dijkstra's algorithm calculates the optimal route. The Dijkstra function returns a large matrix of numbers. The one-line functions extract the optimal route (x, y, t) from this matrix.

5.1.3. Results of the node reduction

This chapter 3.1.1 answers the first research question. the first research question is *"What is the best graph strategy given computational power constraints?"* In chapter 3.1.1 it is found that the number of nodes can be reduced based on the vorticity of the flow and the magnitude of the flow, while still keeping the resolution of the graph high. This section shows the results of the node reduction. This step removes resolution in the places that do not matter. However, if the wrong parameters are used, this can result in a graph with a low resolution.

In figure 5.3 the results of the node reductions are displayed. The first figure shows the node reduction based on the vorticity of the flow ($\alpha = 1$, $\Delta_{min} = 0.04$, $\beta_c = 0.6$, $\beta_m = 3.5$), the second figure shows the node reduction based on the magnitude of the flow ($\alpha = 0$, $\Delta_{min} = 0.04$, $\beta_c = 0.6$, $\beta_m = 3.5$), and the last figure shows a node reduction based on a blend between the vorticity and the magnitude ($\alpha = 0.8$, $\Delta_{min} = 0.04$, $\beta_c = 0.6$, $\beta_m = 3.5$).



Node reduction based on the curl of the flow.
Nodes are reduced with 74.57%, blend factor = 1, βc = 0.6

Node reduction based on the magnitude of the flow.
Nodes are reduced with 83.64%, blend factor = 0, βm = 3.5

Node reduction based on a blend.
Nodes are reduced with 83.64%, blend factor = 0.8, βc = 0.6,βm = 3.5

Figure 5.3: This figure shows the results of the node reduction. The red dots are the nodes after the node reduction step, the lines between the nodes are the arcs of the graph, and the contour plot is the length scale of the node reduction step (LS in equation 3.2)

5.2. Results of model application

This section describes the results of the route optimisation for the project Schouwen Westkop Noord. In Section 3.2.3, the methods for the optimisation of the project are described. Project Schouwen Westkop Noord is a beach-nourishment in Zeeland, near the Oosterscheldekering, due to gullies, banks and tides the sailing is hard to predict. Therefore, the HALEM package is implemented in OpenCLSim to see what the influence of route optimisation on projects is. In this study shows seven simulations of the project, the first simulation is the base case, and the other six simulations are optimisations with different hydrodynamic models (see Section 3.2.3). The base case only optimises the load factor. The other simulations optimise the route, sailing velocity and load factor with HALEM.

The input parameters of the seven simulations are all kept the same. Listing 5.2 shows the input parameters of the simulations, their units, and their physical meaning. These values are fictional. The real values of the hoppers of van Oord cannot be published due to corporate secret-keeping. To show the behaviour of OpenCLSim and HALEM without breaking corporate secret-keeping fiction values are used. The scaling for each parameter is different. The real value of the parameter can deviate up to 30 % compared to the real value.

```
load_factor = np.linspace(0,1,11)             # load factors                                 (-)

start = [3.6740, 51.7096]                     # Location of the pumping area                 (lon, lat)
stop = [3.5226,51.7688]                       # Location of the dredging area                (lon, lat)
Volume = 425_500                              # Total volume to be dredged                   (m^3)

unloading_rate = 1.5
loading_rate = 1.5
ukc = 1.0                                     # Under Keel clearance                         (m)
WWL = 20                                      # Width on Water Line                          (m)
LWL = 80                                      # Length on Water Line                         (m)
hopper_capacity = 4000                        # Maximal capacity of the hopper               (m^3)
V_full = 10*0.514444444                       # Velocity in deep water when maximal loaded   (m/s)
V_emp = 12*0.514444444                        # Maximal sailing velocity empty in deep water (m/s)
T_full = 6.5                                  # Draft when maximal Loaded                    (m)
T_emp = 3.5                                   # Draft When empty                             (m)
WVPI_full = 10000                             # Weight when maximal loaded                   (tf)
WVPI_empt = 4000                              # Weight empty                                 (tf)

func_cost = compute_cost(700_000, 0.008)      # Cost function for route optimiser            ($)
func_co2 = compute_co2(1)                     # Cost function for route optimiser            (g CO2)
func_velo = compute_v_provider(V_emp, V_full) # Vessel velocity is dependent on load factor  (m/s)
```

Listing 5.2: Input parameters of the seven simulations in OpenCLSim

The first simulation for this case study is the base case. This base case represents the state of OpenCLSim without the HALEM package. This base case does not optimise the routes. The vessel follows a predetermined path. Figure 5.5 shows a schematic representation of the results of the base case. In the lower-left figure, the percentage of the time spent per project activity are displayed. From this figure, it becomes clear that sailing is a large part of the project, which takes 68 % of the total project duration. The base case is used as a reference case for how much the HALEM package improves the project. To do so, some core values are used. In table section 5.2 the core value of the base case are displayed. In this table values for project duration, the number of project dredging cycles, production, and averaged time and volume per cycle, can be found.

Section 2.3 introduces four different hydrodynamic models. The case study uses these different models to access the influence of the hydrodynamic models. In Figure 5.4, the three different grids of these models are shown. From these figures, it is visible that the 3D DCSM 0.5 nm model is the most coarse. In plot 2 of Figure 5.4, the nodes of the DCSMv6 zunv4 kf model are shown. It is visible that this model has a medium-fine grid. From the figure, it is visible that this is a curvilinear grid. Plot 3 of Figure 5.4 shows the nodes of DCSM FM 100 m model. This model has a fine grid. From the figure it is visible that the model has different resolutions in different places, this is a clear feature of a Flexible Mesh (FM) grid.

The simulations for the 3D DCSM-FM 0.5 nm and DCSM-FM 0.5 nm model show that the resolution of this model is too coarse to capture the essence of the problem. The fifth plot of Figure 5.5 shows the grid of these two models (the models have the same grid points). This figure shows that the grid is so coarse that it does

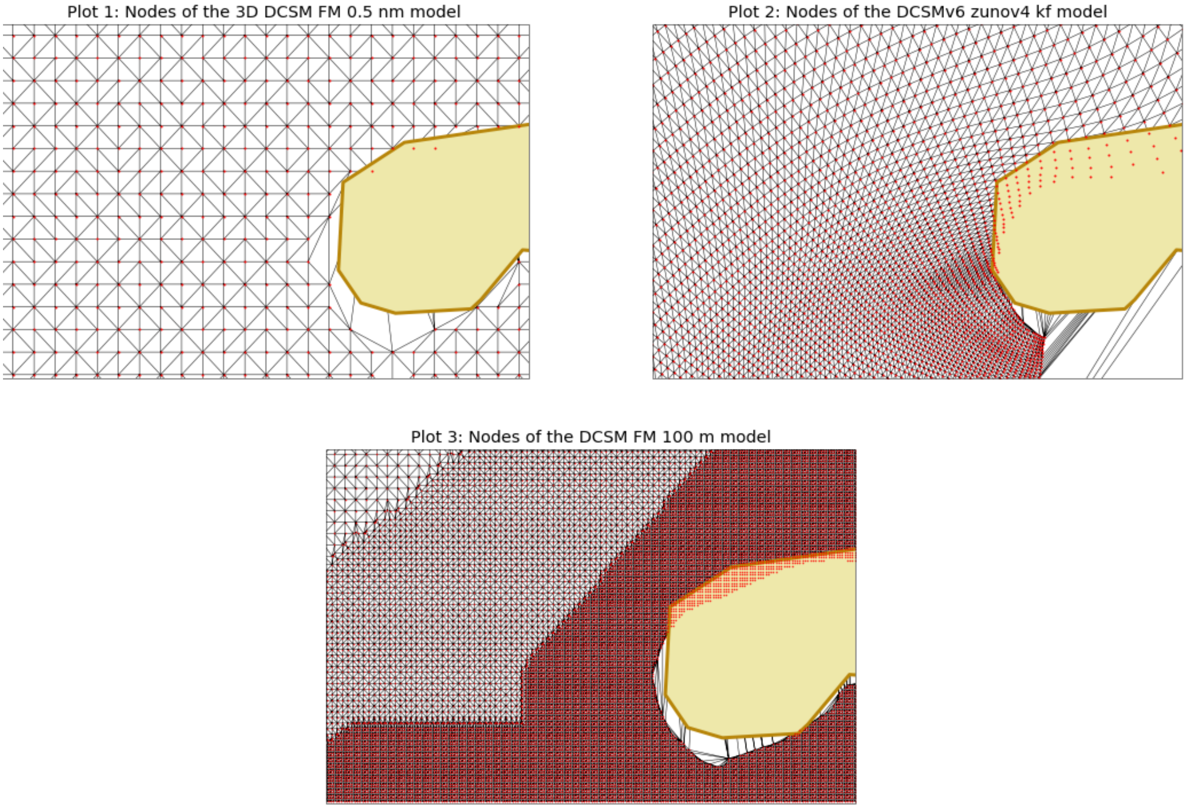

Plot 1: Nodes of the 3D DCSM FM 0.5 nm model
Plot 2: Nodes of the DCSMv6 zunov4 kf model
Plot 3: Nodes of the DCSM FM 100 m model

Figure 5.4: This figure shows the different girds of the used model for the case study. Note that the black lines are is the TIN mesh of the HALEM software, and not the edges of the hydrodynamic model.

not show the gully, in which the pumping location is located. This coarse resolution means that the vessel never has to sail through the gully because the water level is not sufficient. These were unrealistic results. Therefore the DCSM-FM 0.5 nm and 3D DCSM-FM 0.5 nm are not included in the results.

The case study uses three different hydrodynamic models. The first is the DCSMv4-zunov6 kf model (referred to as zuno model), second the DCSM-FM 100m model (referred to as the FM model), and last the tidal analysis model based on the DCSMv4-zunov6 kf (referred to as the zuno model tidal analysis). Each model is run twice, once with the lowest directional resolution possible (1 layer of neighbouring nodes), and once with the highest directional resolution possible (2 or 3 layers of neighbouring nodes, the directional resolution is limited by the computational power).

The results of the simulations are shown in Section 5.2 and Figure 5.5. It is visible that the hydrodynamic effects have a significant influence on the outcome of the projects. The results show that the DCSM FM model with two neighbouring layers (referred to as the FM model with nb = 2) captures the route optimisation for this specific project the best. This simulation shows improvement in the production of 21 per cent, an increase of the total sailed distance of 8 per cent and a decrease of the project duration of 17 per cent over the base case. This improvement is considered a fairly significant improvement for dredging projects. In the paragraphs below the specifics of the results are explained, and explains why the FM model with nb = 2 captures the route optimisation the best.

From the second plot of Figure 5.5, it is visible that the total sailed distance decreases when the directional resolution increases. This effect is due to the Manhattan distance error (Black, 2019). The limited directional resolution causes this error. This is visible in Figure 5.5. For a larger number of neighbouring layers, the Manhattan distance error decreases and the improvements increase. The seventh plot of Figure 5.5 shows these smooth tracks.

From Section 5.2 and Figure 5.5 it can be seen that the difference between the Tidal analysis zuno model and the real data zuno model are minimal. For the highest directional resolution (zuno model nb = 3 and TA zuno model nb = 3) the difference in production is 2 %, the difference in project duration is 1 %, and the difference in total sailed distance is 1 %. These differences are comparable to the truncation errors in the model. The reason that these differences are so small is that water depth dominates the route. The water depth is mainly a tidal feature. Tidal analysis models accurately describe these tidal features.

On further examination of the FM model, we can see that there is some strange behaviour. For the simulations with the FM model, the reduction in production is also approximately half. Moreover, the simulations show an increase in the total sailed distance. However, from plot 8 in Figure 5.5, we can see that the vessel takes shorter routes than the base case. This contradiction is due to the increase in the number of dredging cycles. In the distribution of the load factors, it is visible that the load factor for the FM simulations is on average lower than for the other models (shown in plot 3 of Figure 5.5). This difference is because the grid of the zuno model barely captures the problem. However, the FM has such a fine grid that all the small details are fully captured by the hydrodynamic model. This results in a more accurate description of the project, which turns out to be less optimistic than the results for the zuno model.

In the second plot of Figure 5.5 it is visible that the FM model and the base case do not always sail away directly, but sometimes wait for the tide to change. For the base case, these are logical and expected results. However, for the FM model, this result in a smaller slope of plot 1 of Figure 5.5. The physical meaning of this decrease in the slope is that the production decreases (for this case, it is only a small decrease). This error is due to the lack of resolution in the discretisation of the load factor. If the load factor were discretised in steps of 0.01 instead of steps of 0.1, this error would be prevented. The resolution of the load factor is in this study not increased since it is limited by computational power.

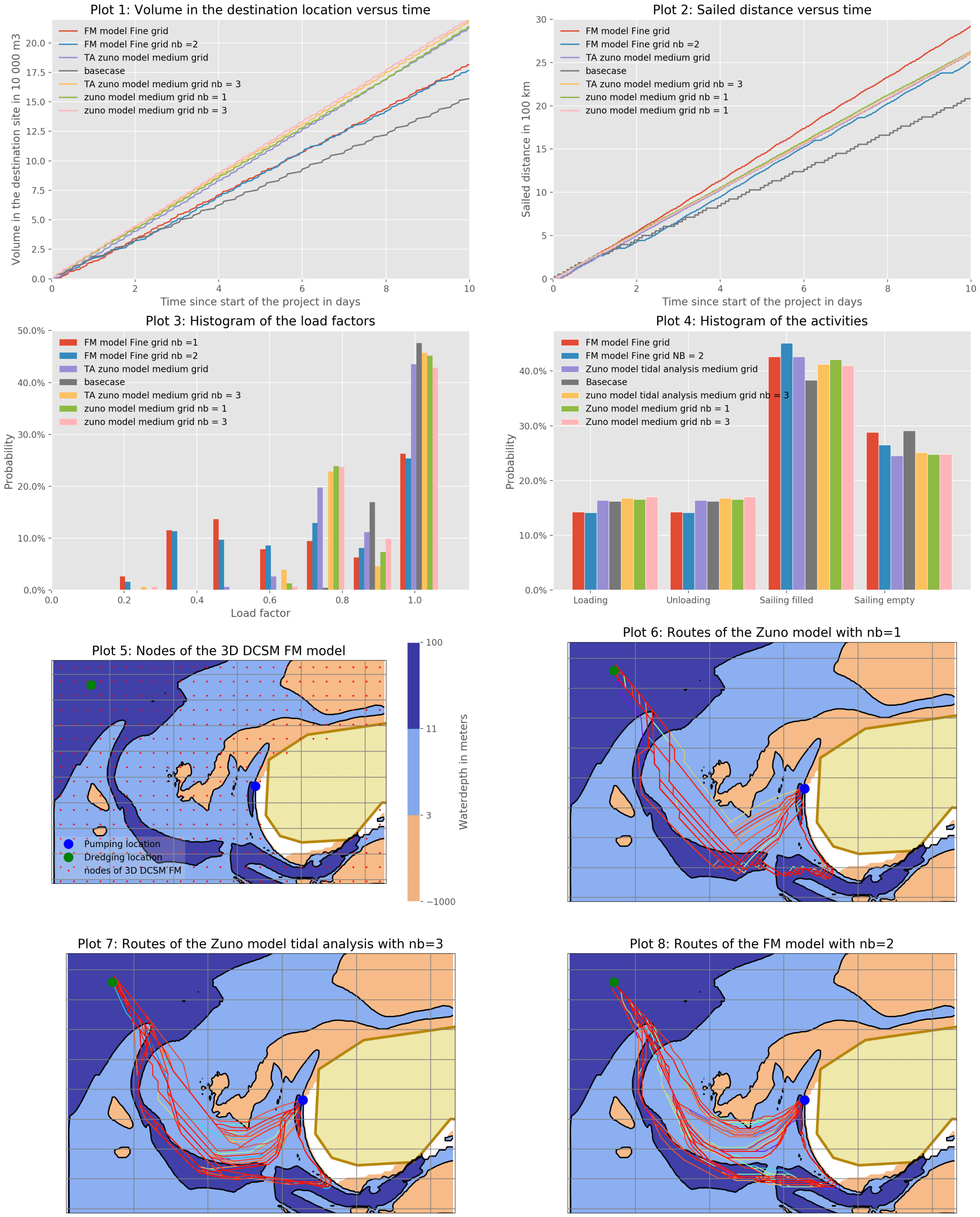


Figure 5.5: This figure shows the results of the simulations visually. Plot 1 shows the volume in the destination location. Plot 2 shows the total sailed distance since the start of the project. Note that for plot 1 and 2 shows only the first ten days of the project. This is done to make details more visible. Plot 3 shows the histogram of the load factors of the different simulations. Plot 4 shows the activity distribution of the simulations. Plot 5 shows the nodes of the 3D DCSM FM 0.5 nm and DCSM FM 0.5 nm model. Plot 6 shows the optimised routes for the simulation with the zuno model and nb = 1. Plot 7 shows the optimised routes for the simulation with the zuno model tidal analysis and nb = 3. Plot 8 shows the optimised routes for the simulation with the FM model and nb = 2. The contour-plot of plot 5 - 8 represents the water depth. The orange area never has sufficient water depth. The dark blue always has enough water depth, independent on the tide and load factor. For the light blue area, it depends on the tide and the load factor whether the water depth is sufficient. Note that for plot 6 - 8, the colour of the route changes with time. The route is blue at the start of the project and red at the end of the project. The colour changes with the colours of the rainbow gradually.

Table 5.1: This table shows the results of the different simulations with the digital twin of the use-case Schouwen Westkop Noord. The different core values describe the results of the simulation. The first column is the base case, and in the six other columns the simulations for the six different hydrodynamic models are found.

Core value		Base case	Zuno model		FM model		Zuno model Tidal analysis	
Layers of Neighbours	:	-	1	3	1	2	1	3
Production (10^3 m3 / week)	:	107	150	154	130	129	149	153
Project duration	:	27 d 22 h	19 d 20 h	19 d 6 h	22 d 23 h	23 d 4 h	20 d 0 h	19 d 12 h
Number of cycles	:	109	117	118	148	144	118	119
Total sailed distance [10^6 m]	:	5.72	5.22	5.03	6.75	6.16	5.26	5.09
Percentage of time spend on loading	:	16 %	17 %	17 %	14 %	14 %	16 %	17 %
Percentage of time spend on unloading	:	16 %	17 %	17 %	14 %	14 %	16 %	17 %
Percentage of time spend on sailing full	:	38 %	41 %	41 %	43 %	46 %	43 %	41 %
Percentage of times pend on sailing empty	:	30 %	25 %	25 %	29 %	26 %	25 %	25 %
Improvement in production	:	0 %	41 %	45 %	21 %	21 %	40 %	43 %
Reduction of project time	:	0 %	29 %	31 %	18 %	17 %	28 %	30 %
Reduction in sailing distance	:	0 %	9 %	12 %	-18 %	-8 %	8 %	11 %

6 | Discussion of the Results

The previous chapters described the model development, model validation, and model application. This chapter discusses the meaning of these results and describes the features, schematisation, and pitfalls of the model. The discussion exists out of four parts. The first section describes the strong points of the model. The second section describes the limitations of the model. The third section describes how the model should be used. The last section describes how the results compare to other route optimisation studies.

6.1. Features of HALEM

The computational time of both the preprocessing step and the route calculation step significantly increase when the model uses the nodes of the hydrodynamic model. Therefore the first improvement is the node reduction. For longer routes or more detailed hydrodynamic models, the number of nodes in the hydrodynamic model become so large that the computational times can become unreasonably long. This study presents a new method to reduce the number of nodes without reducing the resolution of the route. This method uses the vorticity and the magnitude of the flow to prune the nodes in irrelevant areas. The result is that the method removes points in isotropic and homogeneous areas, and keeps the nodes in relevant areas. This step is the most crucial discretisation of this study. This step aims to remove resolution in the places where it is not needed. This so-called node reduction results in a road map with a low number of nodes but a high spatial resolution. However, poorly chosen parameters (Δ_{min}, α, β_c, and β_m) could result in a bad roadmap. This roadmap might miss important features such as currents, bathymetry, or other hydrodynamic properties. This error could eventually lead to a physically impossible route.

The second improvement this study makes is the addition of time-depended hydrodynamic features. Nannicini et al. (2009) presents a method that includes time dependencies by changing the weights of the graphs to time series. This transformation results in a TDSP (Time Depended Shortest Path) algorithm with a slight modification of Dijkstra's algorithm. For this algorithm, the FIFO criteria need to be satisfied. The method presented in Yong-liu and Aiguang-yang (2007) transforms the edges in such a way that the FIFO criterium is guaranteed. With route optimisation for cost or emission reduction, the optimal sailing velocity is no longer constant. In HALEM a new method is implemented for optimising the sailing velocity during the route. In addition in HALEM the sailing velocity is discretised in N different velocities (where N is an input value for the preprocessing). For each point in space, there are then N nodes defined. In the graph these nodes have the same latitude and longitude coordinates but have different sailing velocities. For each connection between two adjacent points in space, N^2 edges are generated to connect all the different sailing velocities. This discretisation results in an excellent way to include the optimal sailing velocity in the optimal shipping route.

This study embeds the python tool HALEM in a framework to optimise not only single routes but the routes of entire projects. This tool is OpenCLSim. OpenCLSim implements the digital twin methodology. This software can be used as a platform to optimise the routes for entire projects. This study implements the HALEM software in OpenCLSim with a so-called python mix-in. With this mix-in, OpenCLSim can optimise the routes of the projects with a given hydrodynamic model. The combination of the HALEM tool and the OpenCLSim tool provide new insights into the influence of hydrodynamic features on the optimal shipping route. The mix-in of HALEM in OpenCLSim gives the option to optimise the load factor of the vessel. The preprocessing phase discretises the load factors of the vessel in a limited number of steps. OpenCLSim determines the resulting production of the dredging cycle for each of these load factors. The method then selects the load factor with the highest production.

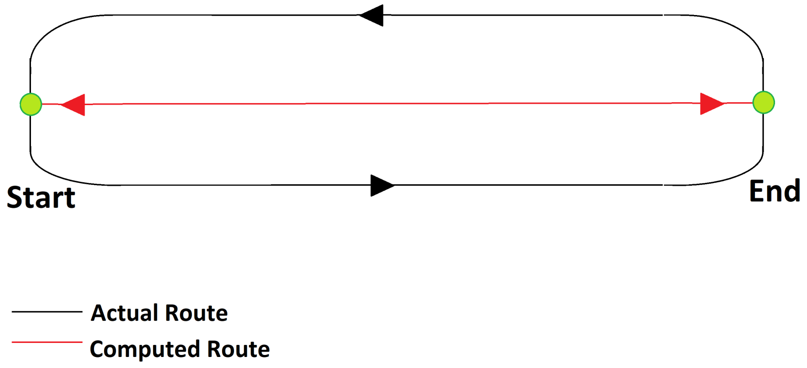


Figure 6.1: This figure shows the limitations that follow from neglecting the inertial forces. The black route is the optimal route, without currents or limitations, with the minimal turning circle. The red line shows the optimal route that neglects the minimal turn circle.

6.2. Limitations of HALEM

Chapter 1 introduces the assumptions of HALEM. These assumptions were neglecting the inertial forces, neglecting wind and wave forces, and assuming that the path of the vessel is freely routable. These assumptions have a considerable influence on the behaviour of HALEM. The implemented features of HALEM find their origin in these assumptions. Since these assumptions are essential to the model, the section below discusses these influences.

The first assumption is neglecting the inertial forces. Neglecting the inertial forces results in two errors within the route. The first error is that acceleration and deceleration are not taken into account. This error is relatively small since sailing velocities in a dredging project are relatively low, and acceleration and deceleration times are short in comparison to the total sailing time.

The second error caused by this assumption is neglecting the minimum turning circle of the vessel. Neglecting the inertial forces assumes that the ship can physical take any bend, no matter how sharp, in reality, this is not the case. The second error that results from neglecting inertial forces is not taking in account the minimal turning circle. Fitriadhy et al. (2012), Yasukawa and Yoshimura (2015), Zhang et al. (2017) discuss the effects of inertial forces on the minimal turning circle. When the distance between the start end location of a project are close to each other, the vessel can not sail in a straight line between start and end because of the minimum turning circle. Since the ship also has to sail back, the route becomes an oval between the start and finish. Figure 6.1 Shows this effect. The AIS data of the validation case shows this behaviour as well (see Figure 4.2). The optimal shipping route that is calculated by HALEM is for this case not possible. How large the effect of this turning circle highly depends on the distance between the start and end of the project, bathymetry and currents and is therefore difficult to quantify.

The second assumption is that the shipping of the project is free navigation. This assumption is reasonable since the contractor gets, for most cases, a permit to sail outside the buoys for dredging purposes. This only results in a small error with the squat calculations. The squat calculation assumes that the waterway is unrestricted. However, when the route sails through a narrow channel (such as for case study Schouwen-Duiveland) the waterway is restricted, and this assumption is not valid. This effect leads to an overestimation of the sailing velocity.

The last assumption is the neglecting of the wind and wave forces. In reality, the wind and wave forces also influence the ship movement. By neglecting these forces the required memory for the calculations reduces significantly. Including the wind and wave forcing should be a relatively simple step, it would, however, require a lot more memory (both RAM and storage memory). The case study Schouwen was, among other reasons, chosen because wind and wave influences were not dominant. The client chose the period of the case study based on the lowest amount of waves and winds present. The results show that water depth dominates the shipping routes. Currents, waves, and wind have far less impact on the shipping routes. Another reason why water depth dominates the project is that the bathymetry dictates the route. Therefore it is safe to model this specific project without wind and waves.

In the simulations of section 5.2 the model calculates the needed water depth for the route as the draft of the vessel plus the ukc. The ukc can be seen as a safety factor to prevent grounding of the vessel. The use case uses a minimal ukc of 1.0 meter. The actual ukc is, in reality, not a constant but a probability distribution. Due to dynamic water depths while sailing, the ukc is not constant with time (however, always greater than the minimal ukc). When the mean ukc is lower, the probability of grounding is more substantial than when the average ukc is larger. Therefore, it can be argued that when the shipping routes are optimised the averaged ukc becomes lower and therefore the risk of grounding becomes larger.

What the distribution of the ukc actual is and how it changes is highly dependent on the use case. The mean ukc does not necessarily decrease when optimising shipping routes since sailing velocities decreases in shallow waters (Due to the squat effect, see Section 2.4.3). In the base case, the vessel waits at low water for the tide to rise in order to leave the shallow channel. This limited water depth means that the ukc for the base case is already low. The effect of optimising the routes on the distribution of the uck is in this study not investigated.
The discussion that optimising the routes may lead to an increase in the probability of grounding is an example of an argument of why HALEM or OpenCLSim should not be used as an autopilot. Other arguments regard safety, ship collision, or other topics. OpenCLSim or HALEM should never be used as the absolute truth but it should function as an advisory role. On vessels the captain should make the decision. Although OpenCLSim or HALEM can advise the captain, it can not and should not replace said captain.

6.3. HALEM compared to the literature

This study shows that hydrodynamic features influence shipping routes. The case study showed that optimising shipping routes results in a significant improvement of the project production. This optimisation results in the question of why HALEM should be used instead of other available models. For example, Kosmas and Vlachos (2012), Lee et al. (2018b), Park and Kim (2015), Wang et al. (2019), Zhang et al. (2019) show good alternative models for ship route optimisation. Which model is better highly depends on the application. Different physical processes, such as currents, water depth, wind, and waves, influence route planning. The application determines which process dominates the route optimisation. Route planning models (including this study), in general, do not take all physical processes into account. This section explains why this study neglects certain physical processes and elaborates which physical processes are taken into account in other models and why.

As described in Section 3 HALEM uses the flow velocity and water depth output of a hydrodynamic model as the input of the route optimisation. This study does not take wind and wave influences into account. The ship movement model reduced the sailing velocity for squat and compensated for the currents. The ship movement model neglects inertial forces to determine the sailing velocities. These assumptions result in a model that can optimise the route when hydrodynamic features are the dominant process. This model can optimise the route when for routes limited by water depth and currents. The case study reflects this. The client chose the period of the case study such that the least amount of waves and winds are present. This period results in a case study where wind and wave forces are not dominant. Due to the limiting bathymetry of the case study, the water depth mainly dominates the shipping routes. Due to its high resolution, this physical process of available water depth is relatively good captured in HALEM. These properties make HALEM a good model for this specific case study.

While HALEM functions as a good model for specific projects, other project might have other dominating physical processes. For these other projects, Table 6.1, shows six route optimisation studies for four different purposes. This table shows which studies take which processes into account. Study 1 (Wang et al., 2019), study 2 (Lee et al., 2018b), and study 3 (Park and Kim, 2015) optimise shipping routes for transatlantic shipping. For this case, wind, waves and variable engine power are the dominant processes. Study 5 (Zhang et al., 2019) optimises the shipping routes in arctic waters. In this study arctic water ice forming was the dominant process, showing that the dominant processes determine the model that must be used to optimise the routes. Lastly, study 6 (Kosmas and Vlachos, 2012) optimised shipping routes for passenger-vessels. Comfort was the main optimisation purpose in this study; for this purpose, winds and wave are the dominant physical processes. Since HALEM does not capture winds, waves, variable engine power, or ice forming, it cannot be used for transatlantic shipping, arctic shipping or passenger shipping. However, since Kosmas and Vlachos (2012), Lee et al. (2018b), Park and Kim (2015), Wang et al. (2019), Zhang et al. (2019) do not take dynamic water depths into account and have a courser resolution, they cannot be used for route optimisation of dredging projects.

Table 6.1: Comparison of different dominant physical processes in different route optimisation studies.

	Variable engine power	Currents	Dynamic Water Depth	Wind	Waves
Kosmas and Vlachos (2012)				X	X
Park and Kim (2015)	X			X	X
Lee et al. (2018b)	X	X		X	X
Wang et al. (2019)	X	X		X	X
Zhang et al. (2019)				X	X
HALEM		X	X		

The importance of taken the correct dominant processes into account also holds for the input of HALEM. The case study compares the simulations with the DCSM and the simulation with the tidal analysis of the DCSM model. For this specific use case, the results did not show significant differences. This similarity lies in the fact that the tide mainly determines the water level. Other features such as wind forces, waves, and stratification have much less influence than the tide, which means that tidal features dominate the hydrodynamic conditions. However, hydrodynamic conditions are not always dominated by tidal features. Other features may dominate in other cases. For example, the stratified flows of the Frisian front dominated the shipping for the realisation of Gemini Wind park (Claessens, 2016). In order to make an accurate simulation for these routes, a hydrodynamic model that accurately captures the three-dimensional flows is of vital essence. When the model output does not represent the dominating hydrodynamic conditions, the results of the route optimisation are not accurate. Therefore knowledge about the dominating hydrodynamic conditions of the project site is of vital importance for the use of HALEM. This study shows the potential use of hydrodynamic models. Therefore, it is recommended to keep investing in the development of high resolution 3D hydrodynamic models. Without the correct hydrodynamic model, these route optimisations are not possible.

7 | Conclusion and recommendations

The main question of this study is: *How can the optimal shipping route be determined for given currents in the North,- and Wadden sea?* To elaborate the main question, four research questions were formulated. These research questions should be answered first to form a conclusion to this study. This sections answers the main questions by first answering the research questions.

The first research question is: *What is the best graph strategy given the computational power constrains?* As discussed in Section 3.1.1 this study uses the nodes of the hydrodynamic model as the vertices of the graph. This approach introduces the problem that there are too many nodes. Therefore the nodes of the graph are reduced. This nodes reduction is made based on the vorticity and the magnitude of the flow. This method results in a grid with a low number of nodes, but a high resolution in the places where it is relevant. Once the nodes of the graph are determined, the nodes of the graph are connected with edges to complete the graph. Due to the node reduction, the nodes do not have a structure anymore. Therefore, Delaunay's algorithm generates a TIN mesh to find neighbours of each node. To increase the directional resolution of the graph edges are not only defined between neighbouring points, but also between neighbours of neighbours. The algorithm uses this parameter for the directional resolution (nb) as an input parameter to create the graph.

The second research question and its answer is: *What is the best method for time-dependent route optimisations?* This question is answered in Section 3.1.2. For the time-dependent shortest path problem (TDSPP), this study uses the method presented in Nannicini et al. (2009). This study uses a time-dependent Dijkstra algorithm that takes temporal variations into account by changing the weight into time series. A transformation of the edges meets the FIFO-criteria of this algorithm. Yong-liu and Aiguang-yang (2007) presents a method for this transformation. The weights of the edges are determined based on the vector addition model. This study defines weights for four different optimisation purposes. These four purposes are the shortest route, the fastest route, the cheapest route and the least polluting route.

The third research question and its answer is: *How can the route planner be implemented in a modular package so that it can be used in different cases?* This question is answered in Section 3.1.4. This research implements the algorithm in an open-source modular python package called HALEM. By developing this package, many different projects can use the algorithm. This package is distributed and published via GitHub, PyPI, Zenodo, and ReadTheDocs (van Halem, 2019). The OpenCLSim software is used to make the influence of route optimisation on dredging projects visible (Van Koningsveld et al., 2019). This study presents a mix-in for the HALEM software into the OpenCLSim software.

Research question 4 and its answer is: *How can dredging projects be optimised for the given route optimisation package and OpenCLSim?* A case study simulates a beach-nourishment at Schouwen Westkop Noord to demonstrate the practical use of HALEM and OpenCLSim, see Section 5.2. For this project, 425,500 m^3 sand should be dredged offshore and pumped onto the beach. Due to the narrow gullies and tidal changes in hydrodynamic features, the routes were hard to predict. The simulation with HALEM and OpenCLSIM shows an increase in the production with 21 % compared to the simulation with just OpenCLSim.

The main question and its answer is: *How can the optimal shipping route be determined for given currents in the North Sea and Wadden Sea?*. This thesis introduces a new algorithm for optimising shipping routes within a dredging project; HALEM. This tool uses graph theory to find the time-dependent fastest path between start and destination. The case study discussed in this thesis demonstrates that route optimisation can increase project production by 21 %. This study concludes that HALEM is an innovative tool that can help access the influence of small scale hydrodynamic features on shipping routes. The same case study showed that HALEM, in combination with OpenCLSim, can reduce the cost and emission of a project significantly. Since the model is generally applicable for projects dominated by small scale hydrodynamic features at any location, the model contributes to the global search for optimisation and reduction of emissions.

7.1. Recommendations

This study presents an algorithm that works and meets the requirements that the research questions demanded. However, it still can be improved on a few points. This section provides the point of improvements to increase the accuracy of the output.

Neglecting the wind and wave forces is the first and most crucial error that this study makes. These wind and wave forces can play an essential role in small scale route optimisation. Implementing these wind and wave forces is relatively easy. The wind force changes the forcing vector in Equation (2.4). The waves do not change the direction of the course but change the acceptable ukc (under keel clearance). With small waves, the ukc can be small, and with big waves, the ukc must be higher. With extreme waves the work-ability conditions of the vessel can be exceeded, an infinite large ukc solves this.

Neglecting the inertial forces, acceleration, and turning circle is the second invalid assumption that this study makes. In Chapter 6 the specific errors of this assumption are discussed. A post-processed hill-climbing refinement can solve these errors. The main disadvantages of the hill-climbing algorithm is that it depends on the initial solution, making it a convex algorithm. However, after graph theory finds the optimal route, the refinement step is no longer a non-convex problem, and an initial solution is known. This problem is suitable for the hill-climbing algorithm. This refinement step can take the inertial forces of the vessel into account. This step will also solve the Manhattan path error that found in Section 5.2 and allow the algorithm to run on a lower directional resolution.

This study introduces four cost-functions for four different optimisation purposes. However, the case study only uses the fastest route and does not use the other three cost-functions. Although Section 3.1.3 introduces a method for including the variable shipping speeds, the case study does not take this variable into account. It is therefore recommended to conduct additional research in the effect of these variable shipping speed and the influence of different cost-functions.

Computational power limits the computations of this research. This research is performed on a computer which has 24.0 GB RAM (random access memory), and an Intel(R) Core(TM) i5-8350U CPU @1.70 GHz 1.90 GHz processor. Although Van Oord generously gave a relatively large amount of RAM, the memory error still formed a limitation in this study. Another limitation was the computation time. By performing most of the calculations in the pre-processing step, the computation times were managed to be kept relatively low. However, computing the entire Schouwen project with the high-resolution FM model, and a high directional resolution resulted in computation times in the order of hours. These limitations in RAM and computational time for large scale projects can be solved in by implementing the following improvements:

1. **A professional clean-up of the software.** Before the lack of capabilities of HALEM can be attributed to external factors such as programming language, and hardware, it must be considered that the code of HALEM itself can be inefficient in some points. The current software of HALEM is developed with little programming experience. This package is the first real package developed by the author. This lack of experience means that the code can be inefficient at some points. A clean-up of the code by someone with more coding experience will fix this problem.

2. **Rewriting the HALEM package in a faster programming language.** Python is a relatively slow programming language. Therefore there is still room improving the efficiency by implementing HALEM in a faster language.

3. **Multi-threading, the software.** HALEM currently uses only a single core of the processor. This limitation is because python does not support multi-threading. Writing the software in a language that support multi-threading will solve this issue.

4. **Running the software on a server.** When a professional clean-up of the code and other software related measures to improve the capabilities of HALEM do not work, it can be considered to improve the hardware on which HALEM is run. An example of a better platform is a server. Running HALEM on a server can provide a faster CPU, with more cores, and more RAM than a laptop or desktop can provide.

A Nomenclature

A.1. Latin Symbols

Symbol	Meaning
b	Width of the waterway
C_b	Block coefficient
E	Edge of a graph, connection between two vertices
FR	Froude number
g	Gravitational constant
h	Water depth
I_{zG}, J_z	Moments of inertia of the vessel
L, B, T	Dimensions of the vessel (Length, Width, Draft)
LS	Length scale for the pruning of the nodes
m	Inertial mass of the vessel
m_x, m_y	Added inertial mass in x and y direction
n_s, s_s	Velocity vector of the vessel rotated to the course of the vessel
n_f, s_f	Forcing vector rotated to the course of the vessel
r	Load factor
r^2	Coefficient of determination
R	Yaw rate of the vessel
Ray	Rayleigh parameter
R_H	Hydraulic radius
S	Blockage coefficient
T_{M2}	Tidal period of the M2 tide
T_{S2}	Tidal period of the s2 tide
T_C	Tidal period of the carrier-wave
$\overrightarrow{u} = [u, v]$	Flow velocity in x,- and y-direction
u_f, v_f	Forcing vector on the vessel
u_t, v_t	Velocity of the vessel
ukc	Under keel clearance
V_{max}	Velocity of the vessel in deep water
V	Velocity of the vessel reduced for squat
W_{empty}	Total weight of the vessel when fully empty
W_{full}	Total weight of the vessel when fully loaded
W_{ship}	Total Weight of the vessel
X_H, Y_H, N_h	Hull forces on the vessel
X_P	Propeller forces on the vessel
X_R, Y_R, N_R	Rudder forces on the vessel

A.2. Greek Symbols

Symbol	Meaning
α	Blend factor between the vorticity and the magnitude of the flow
α_1	Angle between course and horizontal
α_2	Angle between course and flow velocity
β	Drift angle
β_m	Non-linearity factor for the magnitude of the flow
β_c	Non-linearity factor for the vorticity of the flow
δ	Rudder angle
Δ_{min}	Minimal resolution of the grid
ρ_w	Density of water

A.3. Units

This report has been based upon the metric system and adopts the SI system and its derivative units:

Symbol	Meaning
h	hours
km	kilometers
kg	kilograms
m	meters
s	seconds

A.4. Abbreviations

Abbreviation	Meaning
AIS	Automatic Identification System
CFD	Computational Fluid Dynamcis
DCSM	Dutch Continental Shelf Model
DOI	Digital Object Identifier
FIFO	First In First Out
FM	Flexible Mesh
GUI	Graphical User Interface
HALEM	Hydrodynamic Algorithm for Logistic Enhancement, Module.
MMG	Maneuvering Modeling Group
OpenCLSim	Opensource Complex Logistic Simulator
PyPI	Python Package Index or Cheese Shop
RAM	Random access Memory
RMS	Root Mean Square
ROFI	Region of fresh water influence
SSSPP	single-source shortest path problem
SPSPP	single-pair shortest path problem
TDSP	Time Dependent Shortest Path problem
TSHD	Trailing Suction Hopper Dredger
TIN	Triangular Irregular Network
ukc	Under keel clearance

B | Tidal analysis

This appendix shows the results for the tidal analysis for one point. In the figure and table of this appendix are the resulting constituents and time series displayed.

Table B.1: Result of the tidal analysis in one point. This table shows all the tidal constituents for h, u and v.

	Amplitude h	Phase h	Amplitude u	Phase u	Amplitude v	Phase v
M2	1.35	86.21	0.36	109.98	0.57	43.35
S2	0.3	19.04	0.08	41.53	0.13	340.16
M4	0.14	173.52	0.03	125.64	0.04	198.1
M6	0.1	155.61	0.03	103.01	0.03	233.55
O1	0.09	250.04	0.02	49.02	0.02	127.77
K1	0.08	298.02	0.01	63.94	0.02	155.61
MS4	0.08	112.31	0.01	190.12	0.02	160.57
2MS6	0.08	89.23	0.01	145.95	0.01	206.99
MSF	0.05	270.89	0.01	228.7	0.01	167.12
M8	0.03	219.37	0.0	69.92	0.01	209.96
2SM6	0.01	36.76	0.0	251.59	0.0	48.67
S4	0.01	89.04	0.0	27.25	0.0	163.63
M3	0.01	59.46	0.0	167.97	0.0	140.65
SK3	0.01	209.17	0.0	203.73	0.0	57.25
2MK5	0.0	251.54	0.0	54.33	0.0	168.68
3MK7	0.0	261.21	0.0	278.97	0.0	271.29
2SK5	0.0	271.67	0.0	309.37	0.0	199.08

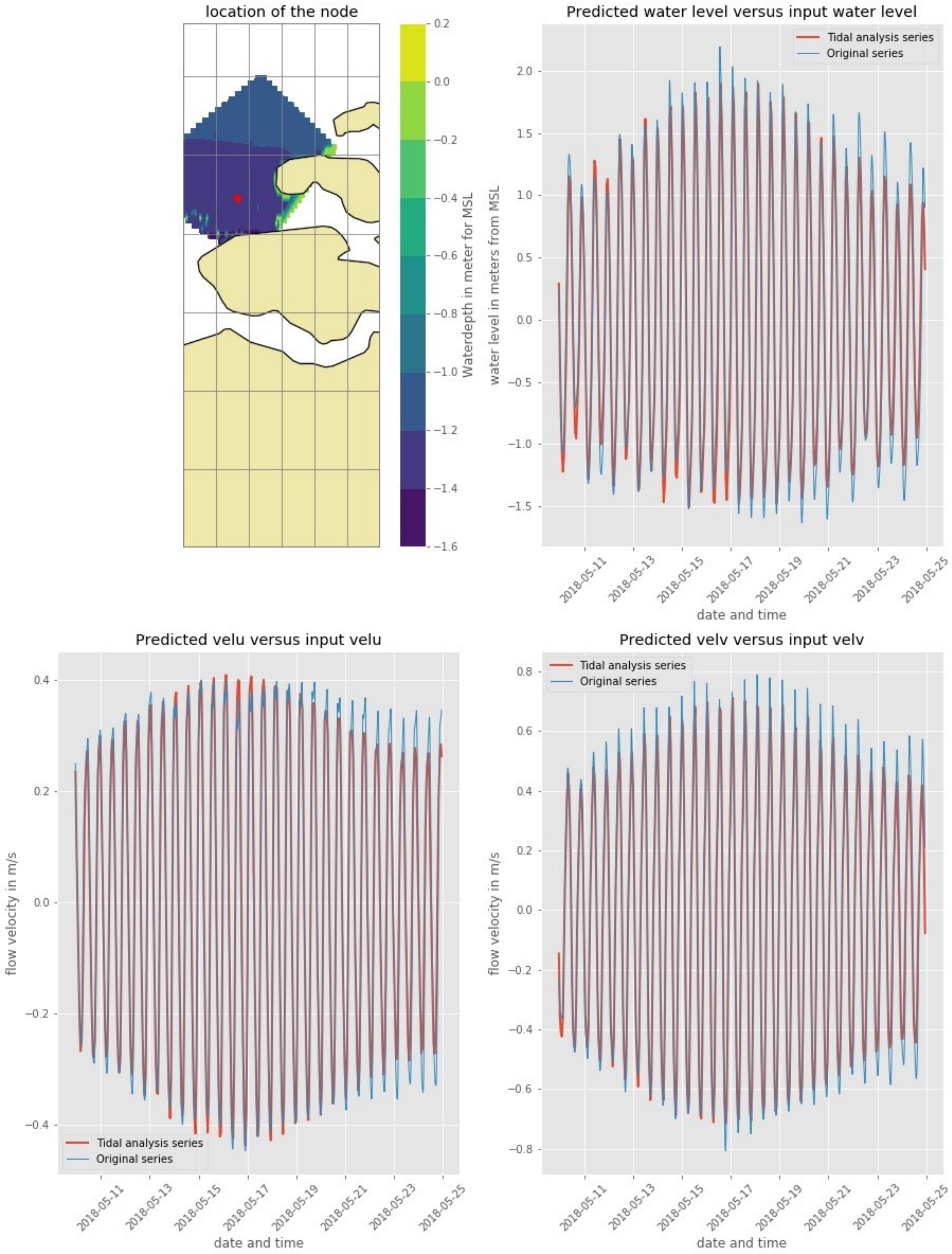


Figure B.1: Results of the tidal analysis for a single node. A list of all the used tidal constituents is displayed in table B.1

C | Results of the validation case Texel for current compensation

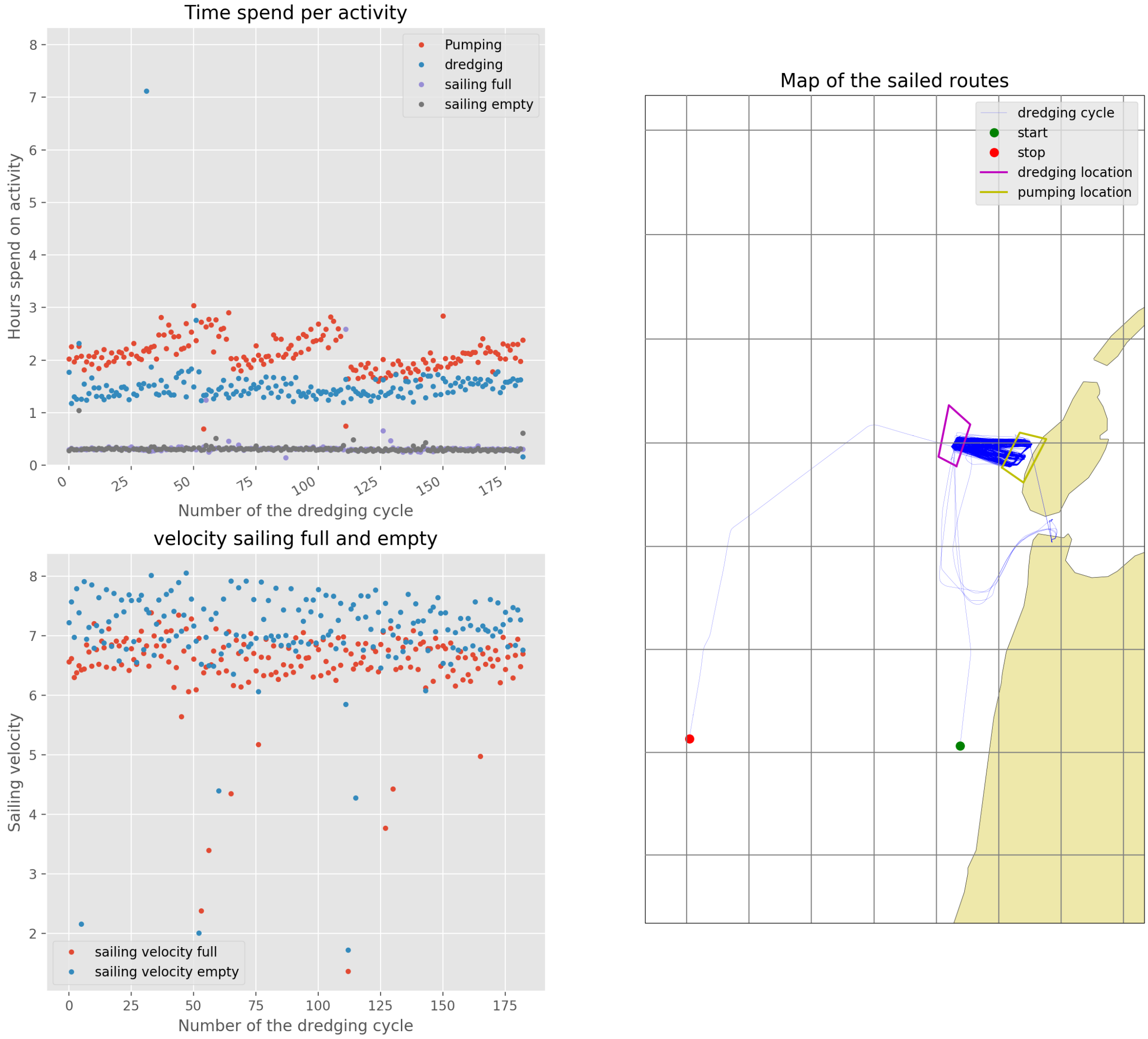


Figure C.1: This figure displays the AIS-data analysis of the Texel beach-nourishment project. the right figure shows the route sailed by the Geopotes 15 and the left figures show the time spend per activity and the sailing velocities.

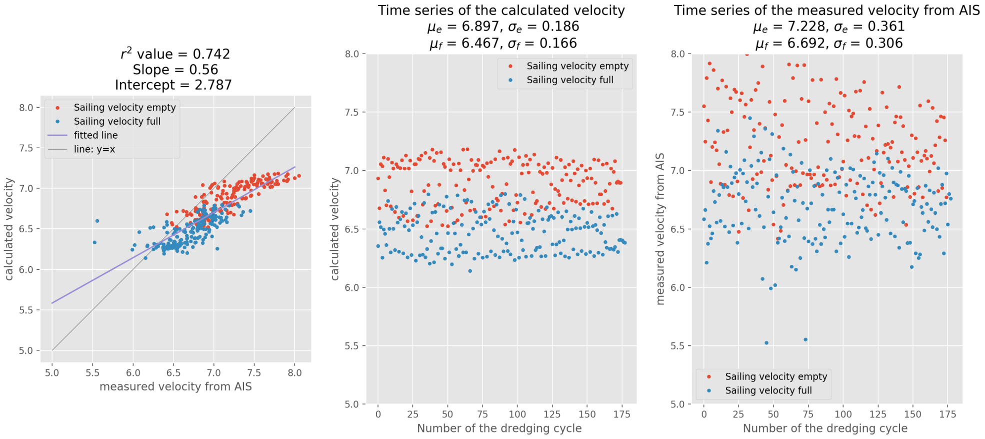


Figure C.2: These are the results for the validation case Texel with a compensation factor based on a logarithmic profile of 1.0. The centre plot and the right plot show respectively the distribution of the calculated sailing velocity and the distribution of the measured sailing velocity. In the title of these plots, the mean values and standard deviations are given. The left figure shows the correlation between the measured and calculated velocity. The parameters that describe the correlation are displayed in the title.

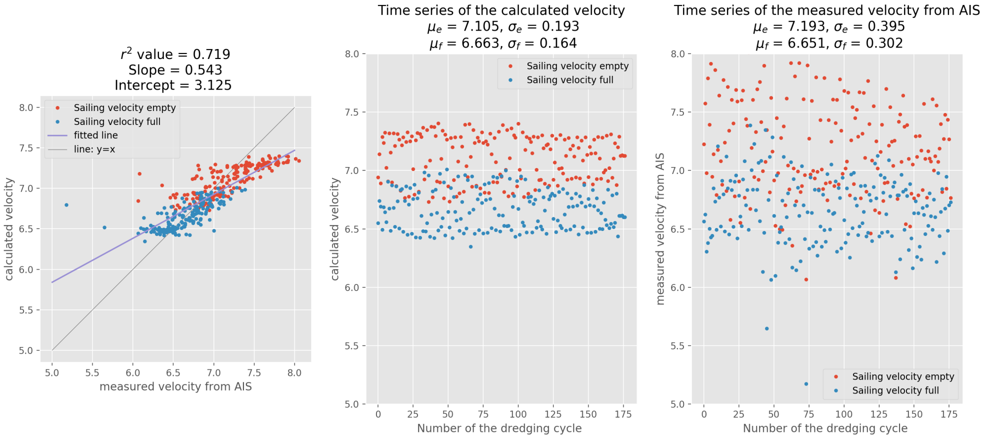


Figure C.3: These are the results for the validation case Texel with a compensation factor based on a logarithmic profile of 1.3. The centre plot and the right plot show respectively the distribution of the calculated sailing velocity and the distribution of the measured sailing velocity. In the title of these plots, the mean values and standard deviations are given. The left figure shows the correlation between the measured and calculated velocity. The parameters that describe the correlation are displayed in the title.

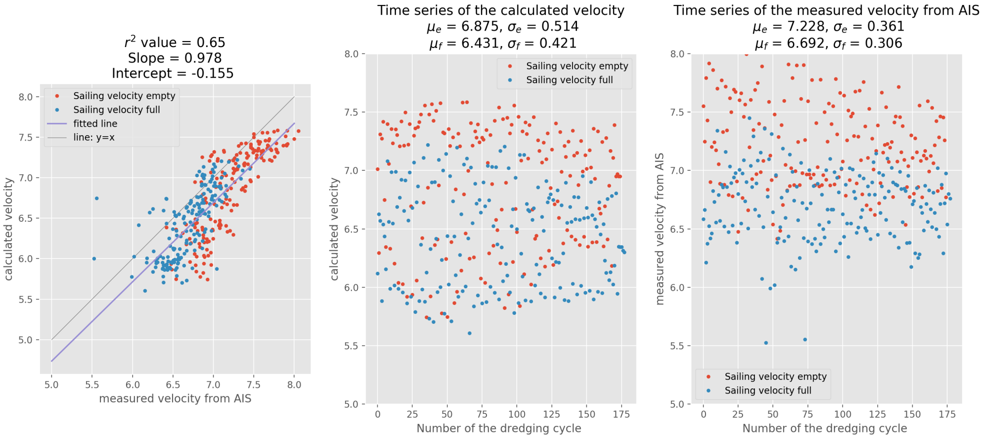


Figure C.4: These are the results for the validation case Texel with a compensation factor based on a logarithmic profile of 2.6. The centre plot and the right plot show respectively the distribution of the calculated sailing velocity and the distribution of the measured sailing velocity. In the title of these plots, the mean values and standard deviations are given. The left figure shows the correlation between the measured and calculated velocity. The parameters that describe the correlation are displayed in the title.

Bibliography

Floor Anthhoni. Currents and circulation Part 1, 2000. URL `http://www.seafriends.org.nz/oceano/currents.htm`.

Dan Bader. *Python Tricks The Book*. 2018. ISBN 9781775093305.

C B Barrass. Chapter 2 - Preliminary estimates for group weights for a new ship. In C B Barrass, editor, *Ship Design and Performance for Masters and Mates*, pages 17–33. Butterworth-Heinemann, Oxford, 2004. ISBN 978-0-7506-6000-6. doi: https://doi.org/10.1016/B978-075066000-6/50003-9. URL `http://www.sciencedirect.com/science/article/pii/B9780750660006500039`.

Richard Bellman and C Odcira. on a Rooting Problem. 1956. URL `https://apps.dtic.mil/dtic/tr/fulltext/u2/606258.pdf`.

Paul E Black. Manhattan distance, 2019. URL `https://www.nist.gov/dads/HTML/manhattanDistance.html`.

James Blake. AkzoNobel vestigt record in Volvo Ocean Race | Andere sporten | AD.nl. *Algemeen Dagblad*, 2018. URL `https://www.ad.nl/andere-sporten/akzonobel-vestigt-record-in-volvo-ocean-race~a9d71d30/`.

Judith Bosboom and Marcel Stive. *Coastal Dynamics I*. 2009. ISBN ISBN 9789065623720.

Stefan Boschert and Roland Rosen. Digital Twin—The Simulation Aspect. In Peter Hehenberger and David Bradley, editors, *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*, pages 59–74. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32156-1. doi: 10.1007/978-3-319-32156-1{_}5. URL `https://doi.org/10.1007/978-3-319-32156-1_5`.

Cu Böttner. Weather routing for ships in degraded conditions. *International Symposium on Safety, Security and ...*, (April):1–7, 2007. URL `http://www.martrans.org/symposium/papers/TrackC/C54boettner.pdf`.

Wesley Bowman. 0.2.4, UTide, 2019. URL `https://pypi.org/project/UTide/`.

Edmund K. Burke and Yuri Bykov. The late acceptance Hill-Climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2012. ISSN 03772217. doi: 10.1016/j.ejor.2016.07.012.

Scott Chacon and Ben Straub. Pro Git. *Pro Git*, 2014. doi: 10.1007/978-1-4842-0076-6.

Chen Chen, Shigeaki Shiotani, and Kenji Sasa. Effect of ocean currents on ship navigation in the east China sea. *Ocean Engineering*, 104:283–293, 2015. doi: 10.1016/j.oceaneng.2015.04.062. URL `http://dx.doi.org/10.1016/j.oceaneng.2015.04.062`.

Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming, Series B*, 73(2):129–174, 1996. ISSN 00255610. doi: 10.1007/BF02592101.

Christiaan (TU Delft) Claessens. Episodic density-induced current velocities at the Gemini offshore wind park. 2016.

Matteo Corno, Simone Formentin, and Sergio M. Savaresi. Data-driven online speed optimization in autonomous sailboats. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):762–771, 2016. ISSN 15249050. doi: 10.1109/TITS.2015.2483022.

Gerben J De Boer. *(32)On the interaction between tides and stratification in the Rhine Region of Freshwater Influence*. 2008. ISBN 9789090238487.

J de Mendoza. Rios. *Memoria sobre algunos metodos nuevos de calcular la longitud por las distancias lunares y explicaciones prácticas de una teor{\'\i}a para la solución de otros problemas de navegación. Imp. Real,* 1795.

Boris Delauney. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS*, (6):793–800, 1934.

Deltares. Nederlands onderzoek en topsport komen samen tijdens Volvo Ocean Race - Deltares, 2017. URL `https://www.deltares.nl/nl/nieuws/nederlands-onderzoek-en-topsport-komen-samen-tijdens-volvo-ocean-race/`.

Joris Den Uijl. Integrating engineering knowledge in logistical optimisation. 2017.

E. W. Dijkstra. Color Imaging 108==A Note on Two Probles in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0029-599X. doi: 10.1007/BF01386390.

Klaus Dorer and Monique Calisti. An Adaptive Approach to Dynamic Transport Optimization. *Applications of Agent Technology in Traffic and Transportation*, pages 33–49, 2005. doi: 10.1007/3-7643-7363-6{_}3.

Nasser A. El-Sherbeny. The Algorithm of the Time-Dependent Shortest Path Problem with Time Windows. *Applied Mathematics*, 05(17):2764–2770, 2014. ISSN 2152-7385. doi: 10.4236/am.2014.517264.

Sophie Ensing and Tom Knijff. Agent-based modeling and simulation of public transport to identify effects of network changes on passenger flows. 2019.

A Fitriadhy, Wan Nik W B, S Khalid, S Rahimuddin, M F Ahmad, Che Wan M N, and M Ferry. Maneuvering Prediction of Research Vessel " Discovery " in Calm Water. (October):1–7, 2012.

S Floyd and R Warshall. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. ISSN 00010782. doi: 10.1145/367766.368168.

M G G Foreman and R F Henry. The harmonic analysis of tidal model time series. 12:109–120, 1989.

Erick Freeman and Elisabeth Freeman. *Head First Design Patterns*, volume 34. 2013. ISBN 0596007124.

Edward Glaessgen and David Stargel. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. (April), 2012. doi: 10.2514/6.2012-1818.

D Goldberg and S Johnson. *The Shortest Path Problem.* 2009. URL `https://books.google.nl/books?hl=nl&lr=&id=Vz0Syt3VqAcC&oi=fnd&pg=PR13&dq=Variants+of+the+shortest+path+problem&ots=dbN0LdAgmN&sig=tW9ipL80AebY7Egxh-piL7KDwWY#v=onepage&q&f=false`.

W. Górski, T. Abramowicz-Gerigk, and Z. Burciu. The influence of ship operational parameters on fuel consumption. *Zeszyty Naukowe / Akademia Morska w Szczecinie*, nr 36 (108(108), 2013. ISSN 1733-8670.

Pieter Van Halem. Source code Halem, 2019. URL `https://github.com/TUDelft-CITG/Route_optimization_in_dynamic_currents`.

J J M V A N Haren. Observations on the horizontal and vertical structure of currents at sub - tidal frequencies in the central north sea. 27(1):1–23, 1990.

Peter E. Hart, Nils J. Nilsson, and Bertram Rapheal. A formal basis for the heuristic determination of minimal cost path, 1968.

J. Holtrop; and G.G.J. Mennen. AN APPROXIMATE POWER PREDICTION METHOD. 1982. doi: doi:10.4271/971010.

Donald B. Johnson. Efficient Algorithms for Shortest Paths in Sparse Networks. *Journal of the ACM*, 24(1):1–13, 1977. ISSN 00045411. doi: 10.1145/321992.321993.

Eiichi Kobayashi, Takashi Asajima, and Nobuo Sueyoshi. Advanced Navigation Route Optimization for an Oceangoing Vessel. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 5(3):377–383, 2011. ISSN 2083-6473. URL `http://www.transnav.eu/Article_Advanced_Navigation_Route_Optimization_,19,307.html`.

O. T. Kosmas and D. S. Vlachos. Simulated annealing for optimal ship routing. *Computers and Operations Research*, 39(3):576–581, 2012. ISSN 03050548. doi: 10.1016/j.cor.2011.05.010. URL `http://dx.doi.org/10.1016/j.cor.2011.05.010`.

Przemyslaw Krata and Joanna Szlapczynska. Ship weather routing optimization with dynamic constraints based on reliable synchronous roll prediction. *Ocean Engineering*, 150(December 2017):124–137, 2018. ISSN 00298018. doi: 10.1016/j.oceaneng.2017.12.049. URL `https://doi.org/10.1016/j.oceaneng.2017.12.049`.

Yiyo Kuo. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers and Industrial Engineering*, 59(1):157–165, 2010. ISSN 03608352. doi: 10.1016/j.cie.2010.03.012. URL `http://dx.doi.org/10.1016/j.cie.2010.03.012`.

Erik P. Kvale. The origin of neap-spring tidal cycles. *Marine Geology*, 235(1-4 SPEC. ISS.):5–18, 2006. ISSN 00253227. doi: 10.1016/j.margeo.2006.10.001.

Sung Min Lee, Myung Il Roh, Ki Su Kim, Hoeryong Jung, and Jong Jin Park. Method for a simultaneous determination of the path and the speed for ship route planning problems. *Ocean Engineering*, 157 (November 2017):301–312, 2018a. ISSN 00298018. doi: 10.1016/j.oceaneng.2018.03.068. URL `https://doi.org/10.1016/j.oceaneng.2018.03.068`.

Sung Min Lee, Myung Il Roh, Ki Su Kim, Hoeryong Jung, and Jong Jin Park. *Ocean Engineering*, (November 2017):301–312, 2018b. ISSN 00298018. doi: 10.1016/j.oceaneng.2018.03.068.

Xiaohe Li, Baozhi Sun, Qianbo Zhao, Yanjun Li, Zhenyu Shen, Wei Du, and Nan Xu. Model of speed optimization of oil tanker with irregular winds and waves for given route. *Ocean Engineering*, 164(November 2017): 628–639, 2018. ISSN 00298018. doi: 10.1016/j.oceaneng.2018.07.009. URL `https://doi.org/10.1016/j.oceaneng.2018.07.009`.

L. R. M. Maas and J. J. M. van Haren. Observations on the vertical structure of tidal and inertial currents in the central North Sea. *Journal of Marine Research*, 45(2):293–318, 1987. ISSN 00222402. doi: 10.1357/002224087788401106.

Arijit Mamanduru, De, Vamsee Krishna Reddy, Angappa Gunasekaran, Nachiappan Subramanian, and Manoj Kumar Tiwari. Composite particle algorithm for sustainable integrated dynamic ship routing and scheduling optimization. *Computers and Industrial Engineering*, 96:201–215, 2016. ISSN 03608352. doi: 10.1016/j.cie.2016.04.002. URL `http://dx.doi.org/10.1016/j.cie.2016.04.002`.

Gianandrea Mannarini, Giovanni Coppini, Paolo Oddo, and Nadia Pinardi. A Prototype of Ship Routing Decision Support System for an Operational Oceanographic Service. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 7(2):53–59, 2013. ISSN 2083-6473. doi: 10.12716/1001.07.01.06.

Anthony F Molland, Stephen R Turnock, and Dominic A Hudson. *Ship Resistance and Propulsion: Practical Estimation of Propulsive Power*. Cambridge University Press, 2011. doi: 10.1017/CBO9780511974113.

Anel A. Montes. Naval Postgraduate Thesis. *Flying*, (December), 2005.

Giacomo Nannicini and Leo Liberti. Shortest paths on dynamic graphs. *International Transactions in Operational Research*, 15(5):551–563, 2008. ISSN 14753995. doi: 10.1111/j.1475-3995.2008.00649.x.

Giacomo Nannicini, Giacomo Nannicini, Dorothea Wagner, Roberto Wolfler-calvo, and Frank Nielsen. Point-to-point shortest paths on dynamic time-dependent road networks To cite this version : Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks par e de :. 2009.

Networkx. Graph generators — NetworkX 1.9 documentation, 2014. URL `https://networkx.github.io/documentation/networkx-1.9/reference/generators.html`.

Networkx. Shortest Paths — NetworkX 1.10 documentation, 2015. URL `https://networkx.github.io/documentation/networkx-1.10/reference/algorithms.shortest_paths.html`.

James B. Orlin, Kamesh Madduri, K. Subramani, and M. Williamson. A faster algorithm for the single source shortest path problem with few distinct positive lengths. *Journal of Discrete Algorithms*, 8(2):189–198, 2010. ISSN 15708667. doi: 10.1016/j.jda.2009.03.001. URL `http://dx.doi.org/10.1016/j.jda.2009.03.001`.

Jinmo Park and Nakwan Kim. Two-phase approach to optimal weather routing using geometric programming. *Schmerz*, 29(6):679–688, 2015. ISSN 14322129. doi: 10.1007/s00773-015-0321-6.

Lokukaluge P. Perera and C. Guedes Soares. Weather routing and safe ship handling in the future of shipping. *Ocean Engineering*, 130(September 2016):684–695, 2017. ISSN 00298018. doi: 10.1016/j.oceaneng.2016.09.007. URL `http://dx.doi.org/10.1016/j.oceaneng.2016.09.007`.

Tim Peters. Zen of Python. 2014.

Helen Shen. Interactive Notebooks. *Nature*, 515:151–152, 2014.

J. Sumihar. Development of a Kalman filter for DCSMv6e. 2012.

K Tierney. Late Acceptance Hill Climbing for The Liner Shipping Fleet Repositioning Problem. *Proceedings of the 14th EUME Workshop*, pages 1–7, 2013. URL `http://www.itu.dk/people/kevt/papers/lahc-lsfrp-eume2013.pdf`.

Eric J. Tuegel, Anthony R. Ingraffea, Thomas G. Eason, and S. Michael Spottswood. Reengineering Aircraft Structural Life Prediction Using a Digital Twin. *International Journal of Aerospace Engineering*, 2011:1–14, 2011. ISSN 1687-5966. doi: 10.1155/2011/154798.

Lara Turner. Variants of shortest path problems. *Algorithmic Oper. Res.*, 6(2):91–104, 2011. ISSN 1718-3235. URL `https://journals.lib.unb.ca/index.php/AOR/article/view/18312`.

Willem Uijttewaal. *Turbulence in Hydraulics*. 2018. ISBN 9783540773405.

University of Reading. Global atmospheric and oceanic circulation, 2019. URL `https://www.futurelearn.com/courses/come-rain-or-shine/0/steps/15237`.

V van der Bilt. Assessing emission performance of dredging projects. Technical report, TU Delft, van Oord, 2019.

Pieter van Halem. Halem, August 2019. URL `https://doi.org/10.5281/zenodo.3363005`.

M. Van Koningsveld, G. J. De Boer, F. Baart, T. Damsma, C. Den Heijer, P. Van Geer, and B. De Sonnevile. OPENEARTH - inter-company management of: Data, Models, Tools and Knowledge. *WODCON XIX: Dredging Makes the World a Better Place*, page 14 pp., 2010. URL `http://repository.tudelft.nl/view/ir/uuid:87e0b19a-b6c9-4761-a17c-91f525577499/`.

Mark Van Koningsveld, Joris Den Uijl, Fedor Baart, and Anne Hommelberg. OpenCLSim. 7 2019. doi: 10.5281/ZENODO.3304278. URL `https://zenodo.org/record/3304278#.XSxdk-gzaUk`.

L. Ceder Vernon. *The Quick Python Book*. Manning, Greenwitch, second edi edition, 2010. ISBN 89781935182207.

Roberto Vettor and C. Guedes Soares. Development of a ship weather routing system. *Ocean Engineering*, 123:1–14, 2016. ISSN 00298018. doi: 10.1016/j.oceaneng.2016.06.035. URL `http://dx.doi.org/10.1016/j.oceaneng.2016.06.035`.

Laura Walther, Anisa Rizvanolli, Mareike Wendebourg, and Carlos Jahn. Modeling and Optimization Algorithms in Ship Weather Routing. *International Journal of e-Navigation and Maritime Economy*, 4:31–45, 2016. ISSN 24055352. doi: 10.1016/j.enavi.2016.06.004. URL `https://linkinghub.elsevier.com/retrieve/pii/S2405535216300043`.

Helong Wang, Wengang Mao, and Leif Eriksson. A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization. *Ocean Engineering*, 186(June):106131, 2019. ISSN 00298018. doi: 10.1016/j.oceaneng.2019.106131. URL `https://doi.org/10.1016/j.oceaneng.2019.106131`.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C. t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan Van Der Lei, Erik Van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:1–9, 2016. ISSN 20524463. doi: 10.1038/sdata.2016.18.

Tetsuo Yanagi. Classification of "siome", streaks and fronts. *Journal of the Oceanographical Society of Japan*, 43(3):149–158, 1987. ISSN 00298131. doi: 10.1007/BF02109215.

H. Yasukawa and Y. Yoshimura. Introduction of MMG standard method for ship maneuvering predictions. *Journal of Marine Science and Technology (Japan)*, 20(1):37–52, 2015. ISSN 09484280. doi: 10.1007/s00773-014-0293-y.

Yong-liu and Aiguang-yang. Study on Non-FIFO Arc in Time-Dependent Networks. *Proceedings - SNPD 2007: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 3:588–593, 2007. doi: 10.1109/SNPD.2007.445.

Jeffrey Xu Yu and Lu Qin. Finding Time-Dependent Shortest Paths over Large Graphs. pages 205–216, 2008.

Chi Zhang, Di Zhang, Mingyang Zhang, and Wengang Mao. Data-driven ship energy efficiency analysis and optimization model for route planning in ice-covered Arctic waters. *Ocean Engineering*, 186(June): 106071, 2019. ISSN 00298018. doi: 10.1016/j.oceaneng.2019.05.053. URL `https://doi.org/10.1016/j.oceaneng.2019.05.053`.

Wei Zhang, Zao Jian Zou, and De Heng Deng. A study on prediction of ship maneuvering in regular waves. *Ocean Engineering*, 137(November 2015):367–381, 2017. ISSN 00298018. doi: 10.1016/j.oceaneng.2017.03.046. URL `http://dx.doi.org/10.1016/j.oceaneng.2017.03.046`.

Firmijn Zijl, Martin Verlaan, and Herman Gerritsen. Improved water-level forecasting for the Northwest European Shelf and North Sea through direct modelling of tide, surge and non-linear interaction. *Ocean Dynamics*, 63(7):823–847, 2013. ISSN 1616-7341. doi: 10.1007/s10236-013-0624-2.

Firmijn Zijl, Julius Sumihar, and Martin Verlaan. Application of data assimilation for improved operational water level forecasting on the northwest European shelf and North Sea. *Ocean Dynamics*, 65(12):1699–1716, 2015. ISSN 16167228. doi: 10.1007/s10236-015-0898-7.

Firmijn Zijl, Jelmer Veenstra, and Julien Groenenboom. The 3D Dutch Continental Shelf Model - Flexible Mesh (3D DCSM-FM). 2018.

List of Figures

2.1 Schematisation of a dredging cycle 5
2.2 Amphidromic points in the North sea 7
2.3 Global Wind paterns 8
2.4 Estuarine fronts 9
2.5 Thermohaline front 9
2.6 Tidal fronts 9
2.7 Classification of the different North Sea water masses 10
2.8 Depth profile of currents 11
2.9 Model domain of DCSMV6-ZUNOV4-KF 12
2.10 Grid discretisations 13
2.11 Vector addition model 14
2.12 Example Graph 18

3.1 The Length scale as a function of the curl 23
3.2 Arcs of the Graph 24
3.3 Triangle inequality and the FIFO property 25
3.4 FIFO transformation 26
3.5 bathymetry of the case study 31

4.1 Trivial validation results 35
4.2 AIS data 37
4.3 Validation results 38

5.1 QR code GitHub 41
5.2 flow chard of route optimisation steps 41
5.3 Node reduction results 43
5.4 Different grids for the case study 45
5.5 Use case results 47

6.1 TTurning circle limitations 50

B.1 Results of the tidal analysis for a single node 58

C.1 AIS data 59
C.2 Validation results 1 60
C.3 Validation results 2 60
C.4 Validation results 3 61

List of Tables

2.1 Simple example of Dijkstra's algorithm 19
2.2 Simple example of Dijkstra's algorithm worked out 20

3.1 Used parameters in the tidal analysis calculation 30

5.1 Use case results 48

6.1 Comparison of different dominant physical processes in different route optimisation studies. . . 52

B.1 Result of the tidal analysis in one point 57

Listings

5.1 installation manual for installing the route optimisation package HALEM 41
5.2 Input parameters of the seven simulations in OpenCLSim 44