Quantification of coastal morphological characteristics for transect profiles along the North Sea coast

Institute:
Name: Rijkswaterstaat
Address: Zuiderwagenplein 2
8224 AD Lelystad
Country: The Netherlands

Internship student:
Name: I.M. (Ivo) Naus
Student number: 3947203

Supervisors Rijkswaterstaat:
Ir. R.J.A. (Rinse) Wilmink
Drs. Q.J. (Quirijn) Lodder

Internal UU supervisor:
Dr. T.D. (Timothy) Price

Table of contents

1. Introduction

The effects of climate change, primarily sea level rise, might cause an increase in coastal maintenance in the future. Nature-Based solutions, also referred to as Building with Nature solutions (BwN), are implemented to counteract erosion, stabilise coasts and ensure protection from flooding (Wilmink et al., 2017). Examples of BwN solutions are beach and shoreface nourishments (Wilmink et al., 2017). This internship project is part of a larger EU Interreg project: Building with Nature. The aim of BwN is to maintain the coasts using natural forces.

Observations of applied BwN solutions indicate differences in nourishment behaviour along the North Sea Region (NSR) coasts. Analysing the observed behavioural differences of nourishments with respect to local coastal dynamics is a key part of the EU Interreg project. The project aims at generating the crucial knowledge needed for making the sandy coasts of the NSR more adaptable and resilient to the effects of climate change (Wilmink et al., 2017).

The objective of this internship is to develop a tool to quantify different coastal characteristics. By using the profile measurement data, gathered by each partner in the EU Interreg project. Also, the results from this quantification are analysed. Furthermore, the quantification could result in a classification on coasts along the NSR with comparable characteristics. This information is useful for Coastal zone managers to compare different behaviour of nourishments.

During this internship a dataset, containing the historical record of measured coastal cross-sections along the NSR coasts (Fig. 1), is analysed. Firstly, the datasets collected by the different partners, from Belgium, The Netherlands, Germany and Denmark, were made into 1 set (Chapter 2). There were several challenges before the data could work in one set, like changing the numbers used to identify each coastal area in the region and converting the data to the same structure.

Figure 1: Overview map of the study area, indicated in orange are the locations of all the measurement transects.

Morphological profile characteristics are defined and calculated for the transect profiles of the complete dataset (Section 3.1). After which the calculated characteristics are visualised in figures (Section 3.2). Additionally, the results are discussed, along with a comparison between different sections of the coast (Chapter 4).

In Chapter 5, potential future analyses which can be done with this dataset are discussed. Finally, the uncertainties of the results are reviewed and some remarkable findings and errors in the database are listed (Chapter 6). An overview of the report structure is given in figure 2.

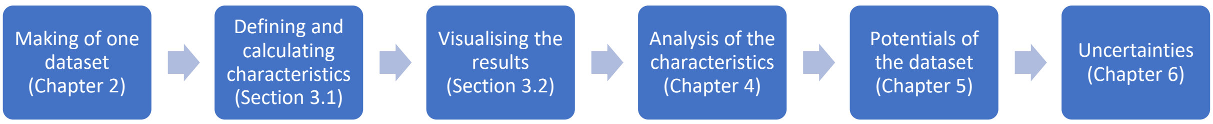


Figure 2: An overview of the structure of the report.

2. Making of one complete dataset

The EU Interreg project: Building with Nature, is a collaboration between different agencies from Flanders (Belgium), the Netherlands, Lower-Saxony (German state), Schleswig-Holstein (German state) and Denmark. Each organisation carries out yearly measurements of coastal profiles along their part of the North Sea coast. This data is stored in different data structures. One of the objectives of this internship is to analyse the whole North Sea coastal region with the same method. The first step to accomplish this is to make one complete dataset with all the data in the same data structure, which is described in this chapter.

2.1 MorphAn

The profile data from the North Sea region can be analysed with MorphAn. MorphAn is an application developed to do coastal analyses. Rijkswaterstaat uses MorphAn for the morphological analysis of the Dutch coast. As MorphAn is an open source application, this tool can be used by everyone.

MorphAn couples profile data, JARKUS files (*.jrk, text file with a certain structure), to certain cross-shore transects along the coast, of which the location (coordinates) and orientation is described in a grid file (*.grd). The coupling between transect grid locations and the measured profile data is based on the coastal area name and number, stated in a coastal area definition file (*.csv).

MorphAn is primarily developed for the analysis of the Dutch coast, consequently it is made to work with the Dutch data structure, JARKUS files (*.jrk), and transect grid files (*.grd). As the other project partners use different data structures to save their data, the other data cannot be analysed with MorphAn. However, the data can be converted to the JARKUS data structure and the grid structure used by Rijkswaterstaat. To make one usable dataset from all the data, the used coastal area numbers needed to be changed as well, as otherwise some numbers were used double and therefore MorphAn would not be able to read some data. Furthermore, the Danish data was combined into one coastal area for the whole Danish coast, which needed to be split into the original 6 coastal areas, based on the transect numbers.

2.2 Old data structures to JARKUS format

As each project partner saved the data in a different format, the data must be transformed to the same data structure to make a complete dataset which can be analysed with similar methods. It has been determined that all the data will be transformed to a JARKUS structure, used by Rijkswaterstaat, as MorphAn is compatible with this data structure. The data transformation tool was developed (Bregman, 2017), which was made to transform the data structure to the used JARKUS structure in MorphAn.

Using the data transformation tool, all the data was converted from the old data structures to the JARKUS data format. For the data from Sylt (Schleswig-Holstein), a little adjustment to the tool had to be made, as the measuring date was sometimes also stated on the same line as the transect number. In the function *F_SH_DataAlloc* the following was changed:

```
if length(tokens{li})>=2 %data lines exists of more than 2 numbers
        if strcmp(tokens{li}(2),'Bezeichnung')==1 %recognize start of
dataset by this word

        data.rough.raainr{ds}=char(tokens{li+1});

        %allocate metadata
```

To:

```
if length(tokens{li})>=2 %data lines exists of more than 2 numbers
        if strcmp(tokens{li}(2),'Bezeichnung')==1 %recognize start of
dataset by this word

            if size(tokens{li+1},2) == 2 % Recognizing if this token only
contains the profile number
                % or also the date of measurement, if both: only take the
                % first cell into account which is the profile number
                data.rough.raainr{ds}=char(tokens{li+1}(1));
            else
                data.rough.raainr{ds}=char(tokens{li+1});
            end

        %allocate metadata
```

2.3 Grid files

The grid files contain the position and orientation of the profiles of each coastal area. These files were also converted to a structure usable by MorphAn with the use of the data transformation tool. To be able to load all the measurement data into the same MorphAn project, the transect grid files from all the partners (*.grd) should contain coordinates in the same coordinate system. Therefore, the old coordinates in the grid files were transformed from the locally used coordinate system to the global coordinate system: WGS-84, EPSG: 3395. This transformation was done with the use of Python.

2.4 Changing coastal area numbers & names

All the data from the partners could be loaded individually into MorphAn after the JARKUS and grid data were transformed into the right structure. However, to load all the data into one MorphAn project, each coastal area should have a unique name and number. Unfortunately, this was not the case. The coastal area numbers were therefore changed to 8-digit numbers, based on the country code and locally used area number (Table 1). Some coastal area numbers in the JARKUS files were changed with the use of a developed MATLAB script, *Omzetten_kv_num_jrk_files** (Appendix Script 1), which could automatically identify the old area number and replace it with the new one. This was done for the data from the Netherlands, Flanders, Baltrum/Langeoog and Denmark. Other numbers were changed by transforming the old data again with the use of the data transformation tool and giving another in-/output number. The coastal area numbers of the grid files needed to be changed as well. This was easily done by hand within excel. The same applied for the coastal area definition files (*.csv), as these files only contain a combination of coastal area numbers and coastal area names.

The initial data from Denmark was grouped as if the whole Danish coast was one coastal area. While, the Danish partners use 6 coastal areas. Therefore, the dataset needed to be split into 6 different coastal areas based on the transect numbers, with each a unique name and number (Table 2). This splitting of the measurement data (*jrk) was done with the use of a MATLAB script, *Omzetten_kv_num_jrk_files_Vestkyst* (Appendix Script 2), which stored the data of the new coastal areas (with their new numbers) in separated JARKUS files. The names given to the coastal areas were based on local geographic areas. These names and numbers also needed to be changed in the grid file and the coastal area definition file (*.csv), which was done by hand. Note: Holmsland is a part of the coast which is measured in more detail (transects narrower spaced) and this area lies within the

Midtjylland (45000002) coastal area, indicated by the same coastal area number + one sub-number (the 9th digit).

The data from Flanders consist of four coastal areas, but they are treated as one coastal area during this internship project: Middelkerke. Because they are very small areas and close to Middelkerke. Although, in the data each of these four coastal areas have their own number and name. Table 1 contains the new coastal area name and number combinations and the coastal areas are ordered from south to north (west to east).

Table 1: An overview of the coastal area names in combination with the coastal area number, ordered from south to north (west to east)

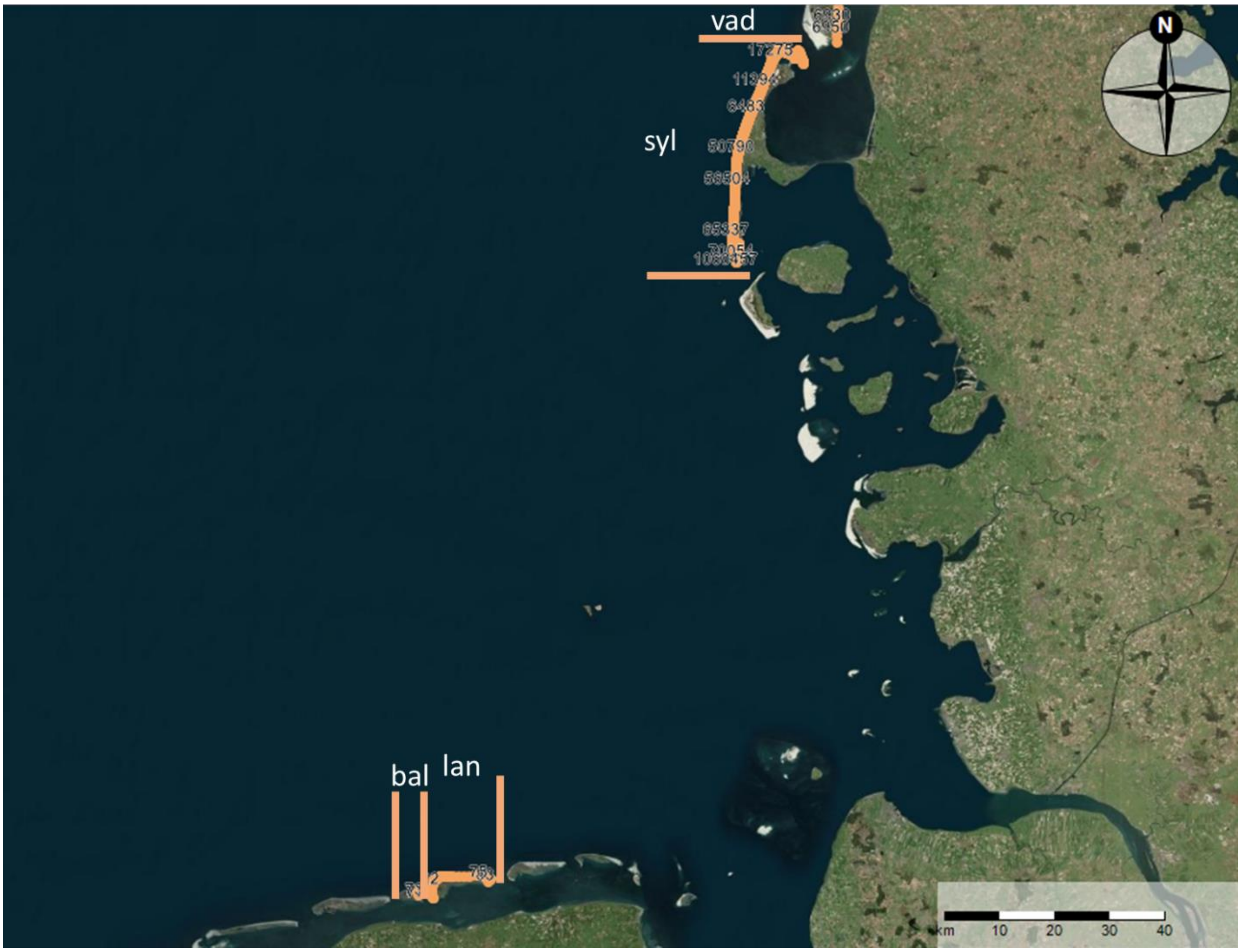
	Coastal area names	Coastal area numbers	Abbreviation
Flanders (Belgium)	Middelkerke: - Westende-Bad - De Krokodille - Middelkerke-Bad - Middelkerke-Oost	320000**: - 32000012 - 32000013 - 32000014 - 32000015	mdk
The Netherlands	Zeeuw-Vlaanderen	31000017	zws
	Walcheren	31000016	wal
	Noord-Beveland	31000015	nb
	Schouwen	31000013	sch
	Goeree	31000012	goe
	Voorne	31000011	voo
	Delfland	31000009	del
	Rijnland	31000008	rij
	Noord-Holland	31000007	nh
	Texel	31000006	tex
	Vlieland	31000005	vli
	Terschelling	31000004	ter
	Ameland	31000003	ame
	Schiermonnikoog	31000002	mon
Lower-Saxony (Germany)	Baltrum	49260040	bal
	Langeoog	49260050	lan
Schleswig-Holstein (Germany)	Sylt	49250107	syl
Denmark	Vadehavsoer	45000001	vad
	Holmsland	450000027	hol
	Midtjylland	45000002	mid
	Agger	45000003	agg
	Nationalpark-Thy	45000004	thy
	VigsoJammerbugten	45000005	vig
	Tannis-Bugt	45000006	tan

Table 2: The 6 coastal area names and numbers, plus the additional coastal area; Holmsland

Coastal area name	Coastal area number	Transect range
Vadehavsoer	45000001	6280 - 6970
Midtjylland	45000002	4210 - 6270
Holmsland	450000027	-
Agger	45000003	4010 - 4170
Nationalpark-Thy	45000004	3060 - 3670
Vigso-Jammerbugten	45000005	1510 - 3050
Tannis-Bugt	45000006	1010 - 1500

Now that all the original data files are converted to the JARKUS data structure files, the grid files contain the coordinates in the same global coordinate system and each coastal area has a unique coastal area number and name, the profile data could be loaded into the same MorphAn project. With the use of MorphAn an overview map was made to indicate the locations of each coastal area on this page and the following two pages (Fig. 3). The full dataset now consists of 5840 transect locations. The measurements were done between 1874 – 2017, which means that almost 150 years of measurements are available at some locations, although for most locations the consistent annual measurements were done since 1965 (see Section 3.1.3, Fig 9). In total the dataset contains approximately 135.000 measured transect profiles, leading to an average of a little more than 23 profiles per transect location.


mid
hol
vad
syl


vad
syl
bal
lan

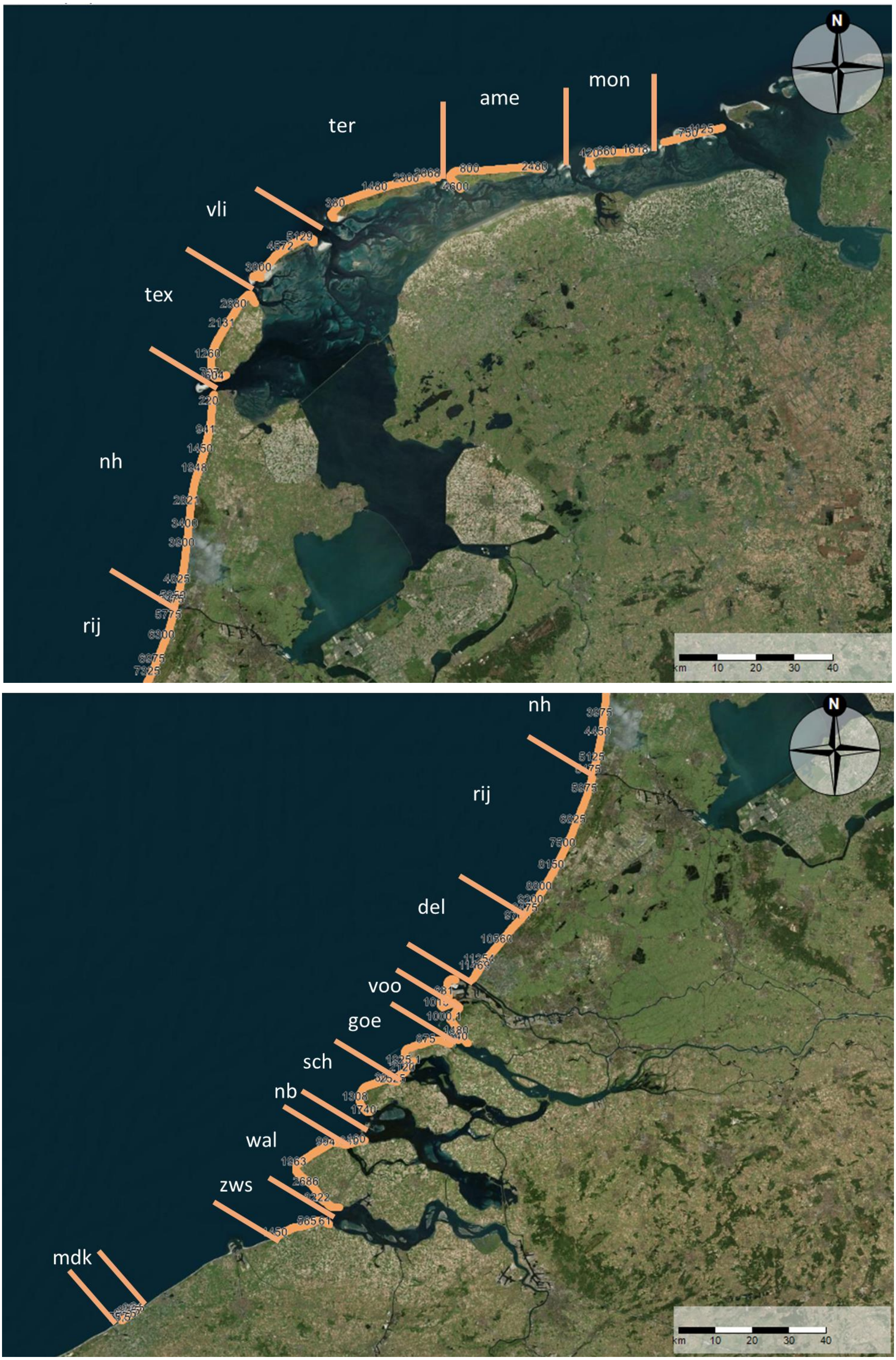


Figure 3: The locations of the coastal areas indicated in a map over 5 different windows. The abbreviations used are stated in Table 1. Only the coastal areas of which the transect measurement data was used are indicated.

2.5 Converting JARKUS data to a usable data structure

Now that the dataset was complete and structured in the same format, the calculation of certain morphological characteristics of the transects could be done. As MorphAn is provided with a scripting tool which uses the Python language, the calculation of the morphological characteristics was firstly attempted with the use of this scripting tool. Unfortunately, it was unsuccessful to load the measurement data from a MorphAn project with the use of the scripting tool. Therefore, there was no surplus value to quantify the characteristics with the use of Python, and consequently MATLAB was used to write scripts for the quantifications (Chapter 3).

Before the calculation of the characteristics with the use of MATLAB could be done, the JARKUS data structures needed to be converted and stored into MATLAB structure arrays (*.mat). This was necessarily as the JARKUS structure is very inconvenient to use for calculations. Since the data is structured in the following way:

```
31000002      2012       102         0       2203       1904       627
-5    3221   0    2901   5    3041   10    3051   15    3351
20    3601   25    3551   30    2991   35    2861   40    2531
45    2951   50    3381   55    3521   60    4481   65    5681
70    6491   75    7251   80    7951   85    7761   90    7661
95    7571   100    7731   105    7611   110    7501   115    6921
120    6081   125    5631   130    4201   135    3511   140    3601
145    3871   150    4241   155    4791   160    5851   165    6331
170    5781   175    5461   180    4821   185    3631   190    2931
195    2581   200    2161   205    1721   210    1761   215    1811
220    1791   225    1891   230    1761   235    1791   240    1711
245    1971   250    2361   255    2121   260    1871   265    1681
270    1641   275    1611   280    1631   285    1591   290    1601
295    1571   300    1561   305    1551   310    1571   315    1581
320    1501   325    1581   330    1461   335    1501   340    1471
345    1411   350    1411   355    1411   360    1391   365    1391
370    1271   375    1371   380    1381   385    1351   390    1311
395    1271   400    1251   405    1291   410    1261   415    1291
420    1241   425    1241   430    1211   435    1201   440    1171
445    1201   450    1201   455    1221   460    1181   465    1151
470    1171   475    1131   480    1221   485    1201   490    1131
495    1251   500    1141   505    1151   510    1191   515    1201
520    1211   525    1161   530    1171   535    1161   540    1201
545    1221   550    1211   555    1181   560    1181   565    1171
570    1161   575    1151   580    1161   585    1121   590    1101
595    1171   600    1071   605    1121   610    1111   615    1041
```

Where the first line, the header, consists of 7 numbers containing (Deltares, 2016):

- The coastal area number (in this case the one from Schiermonnikoog, see Table 1)
- The measurement year
- The transect number
- The indicator for the type of measurement (standard yearly or additional)
- The date of the terrestrial measurement
- The date of the bathymetric measurement
- Number of measurement points for this transect measurement

Followed by the measurement data in pairs of X and Ys. Where each number in the uneven columns (column 1, 3, 5 etc..) defines the distance from the reference point in meters (X) and each number in the even columns defines the vertical distance from the reference height in cm (Y). The last digit of the Y numbers is a tag, defining the type of measurement so that MorphAn can differ between bathymetric, terrestrial or interpolated values. All measured transects for each coastal area are stored below each other in a *.jrk text file, separated by the header lines (Deltares, 2016). Furthermore, each coastal area got its own JARKUS document.

For each coastal area (for each JARKUS file) a MATLAB structure array was made. Containing the measurement data for all the measured transects within that coastal area. Where each measurement is stored in a structure field; with the x and y values, the transect number, the measurement year, the number of data points and the coastal area number (Fig. 4). The coastal area number should be the same for all the measurements in the same structure, as each structure only contains the data for one coastal area. The data from the JARKUS files was converted to the new structures with a MATLAB script, *Load_jarkus.m* (Appendix Script 3). This script automatically runs through every line of the *.jrk text files and distributes the information into the right fields of the structure array. The different transect measurements are found by finding the header lines, which separates the measurement data for each transect from each other. From the Y data the last digit is removed, as this information is not used in the calculation of the characteristics. A more detailed description of the script, and how to use it, is found in Appendix Section 3.2.

Fields	x	y	transect	year	num_of_points	coastal_area_number
1	97x1 double	97x1 double	71872	1984	97	49250107
2	117x1 double	117x1 double	71823	1984	117	49250107
3	103x1 double	103x1 double	71773	1984	103	49250107
4	147x1 double	147x1 double	71723	1984	147	49250107
5	141x1 double	141x1 double	71673	1984	141	49250107
6	111x1 double	111x1 double	71623	1984	111	49250107
7	100x1 double	100x1 double	71573	1984	100	49250107
8	83x1 double	83x1 double	71523	1984	83	49250107
9	90x1 double	90x1 double	71473	1984	90	49250107
10	319x1 double	319x1 double	71404	1984	319	49250107
11	324x1 double	324x1 double	71354	1984	324	49250107
12	323x1 double	323x1 double	71304	1984	323	49250107
13	298x1 double	298x1 double	71254	1984	298	49250107
14	272x1 double	272x1 double	71204	1984	272	49250107

Figure 4: An example of a MATLAB structure array in which each transect measurement is saved.

For each coastal area a *.mat structure file was created and saved. An overview of the structure names is provided in table 3. It is advised not to change these names, as these names are also used in the structure in which each characteristic is saved after calculation (Chapter 3). Furthermore, the names are used to automatically call the characteristics from each coastal area in a loop. Therefore, if one would change the name of a structure file, one should change the name in several scripts (and within a function), and run some of these scripts again, as well.

The data from the coastal area Tannis-Bugt (45000006) measured in 2016, was reversed; the y values got the wrong sign. With the use of a simple MATLAB script (Appendix Script 4) the y data from 2016 was multiplied by (-1) to fix the problem.

Table 3: An overview of the MATLAB structure names given to each coastal area data file.

	Coastal area names	Structure names (*.mat)
Flanders (Belgium)	Middelkerke	Middelkerke_detail_320000newnum
The Netherlands	Zeeuw-Vlaanderen	zws_vlaanderen_31000017
	Walcheren	walcheren_31000016
	Noord-Beveland	nbeveland_31000015
	Schouwen	schouwen_31000013
	Goeree	goeree_31000012
	Voorne	voorne_31000011
	Delfland	delf_31000009
	Rijnland	rijnland_31000008
	Noord-Holland	nh_31000007
	Texel	texel_31000006
	Vlieland	vlieland_31000005
	Terschelling	terschelling_31000004
	Ameland	ameland_31000003
	Schiermonnikoog	schier_31000002
Lower-Saxony (Germany)	Baltrum	Baltrum_data_49260040
	Langeoog	Langeoog_data_49260050
Schleswig-Holstein (Germany)	Sylt	All_Sylt_49250107
Denmark	Vadehavsoer	Vestkyst_Vadehavsoer2_45000001
	Holmsland	Holmsland_data_450000027
	Midtjylland	Vestkyst_Midtjylland_45000002
	Agger	Vestkyst_Agger_45000003
	Nationalpark-Thy	Vestkyst_NationalparkThy_45000004
	VigsoJammerbugten	Vestkyst_VigsoJammerbugten_45000005
	Tannis-Bugt	Vestkyst_TannisBugt_45000006

3. Methods

One goal of this internship was to divide the North Sea coast (from Flanders to Skagen) into different areas based on the morphology. Adjacent cross-shore profiles with a similar morphology will together form an area. This will result in a distribution of the coast in different zones based on the morphology. To accomplish this goal several steps had to be taken. Firstly, the coastal morphological characteristics on which the division was based were determined. This also included that a method to quantify these morphological features needed to be defined. Hence the question was; which morphological characteristics will be used to characterise the coast and how will these characteristics be determined? After this was determined, a tool to determine/quantify each characteristic needed to be made. This tool could be made in MorphAn (Python) or in MATLAB. MorphAn is provided with a scripting tool which uses the Python language. Unfortunately, as previously stated in Section 2.5, it failed to use/load the measurement data from a MorphAn project with the use of the scripting tool (Python). Therefore, MATLAB was used to write scripts for the quantifications. Now that the measurement data was saved in usable structures (Section 2.5), the calculation of the characteristics could be done. The calculations are described in this chapter (Section 3.1). The MATLAB scripts used to calculate these characteristics can be found in Appendix Section 3. After the quantification, the results were post-analysed by visualising them in figures (Section 3.2). With the use of these figures the morphology was compared between the profiles, and the results are discussed in Chapter 4.

3.1 Defining and calculating coastal profile characteristics

This section will focus on how the morphology can be used to characterise a profile, using certain morphological elements. Furthermore, the calculations of the characteristics are described. Schematic cross-sections are used to visualise the quantification of these morphological characteristics. Due to the limited available time, not all characteristics which were made up were determined during this internship, some of these 'potential' characteristics are described in Chapter 5. Note: the described height points used in the schematic cross-sections (to determine certain characteristics) might be different from the height points which were actually used during this project.

3.1.1 Reference points

First, as each profile has its own horizontal reference point (the 0-meter cross-shore distance point is not located at relatively the same position, like in figure 5), a morphological reference point must be determined which is generally stable. A suitable point would be the peak of the first dune, as its position is commonly quite constant. This reference point can be used to refer other cross-shore positions too, like the position of a subtidal bar. In addition, the point where the mean sea level (0 m elevation) crosses the profile can also be used as a horizontal reference point. Furthermore, it is convenient to use intersections of horizontal height lines with the profile (e.g. the intersection of the profile with the 0 m elevation line) to determine the morphological features. The distance between two height points is relative to each other and not to the chosen reference point for each profile. Therefore, such a value is comparable to other profiles (which have different reference points). For example, figure 5 shows two identical cross-sections. They have a different horizontal reference point. At a specific height (example +4 and -2) the profiles have a different horizontal location. However, the distance between the points remains the same as it is not dependent on the reference point. This method, using the intersection/height points, is used in the calculation of the characteristics. The determination of the intersection/height points is further described in the next section.

Furthermore, a vertical reference point should be set. If the vertical reference points are not at relatively the same height for each profile, the profiles cannot be compared relative to each other. Therefore, the vertical reference point should be a point relative to the a hydrodynamical position:

the mean sea level (MSL). In this project it is presumed that the used vertical ordinance datum (like the NAP) for the measurements is (nearly) equal to MSL. Hence, the vertical profile values are used as they are (not relatively to other values).

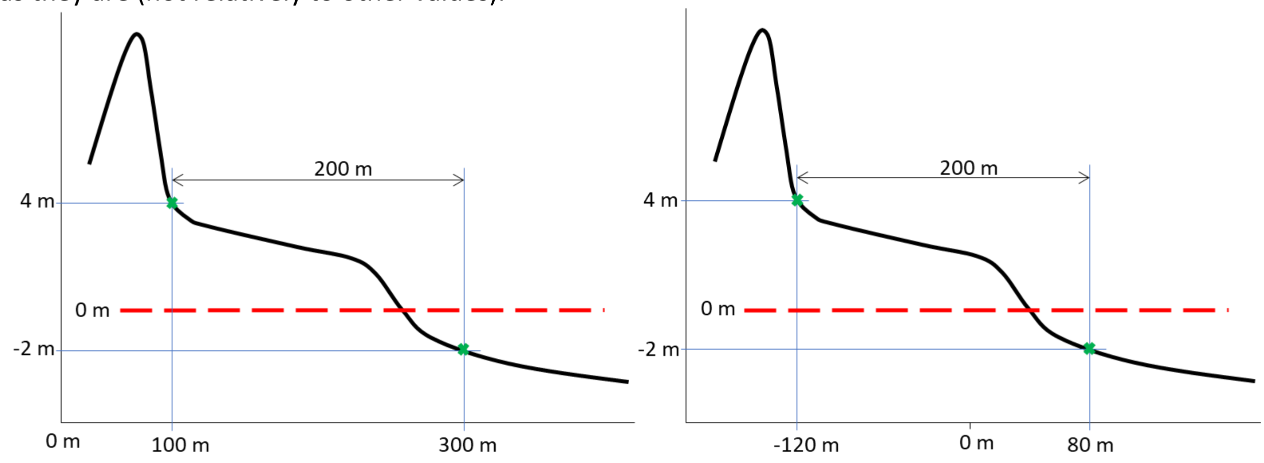
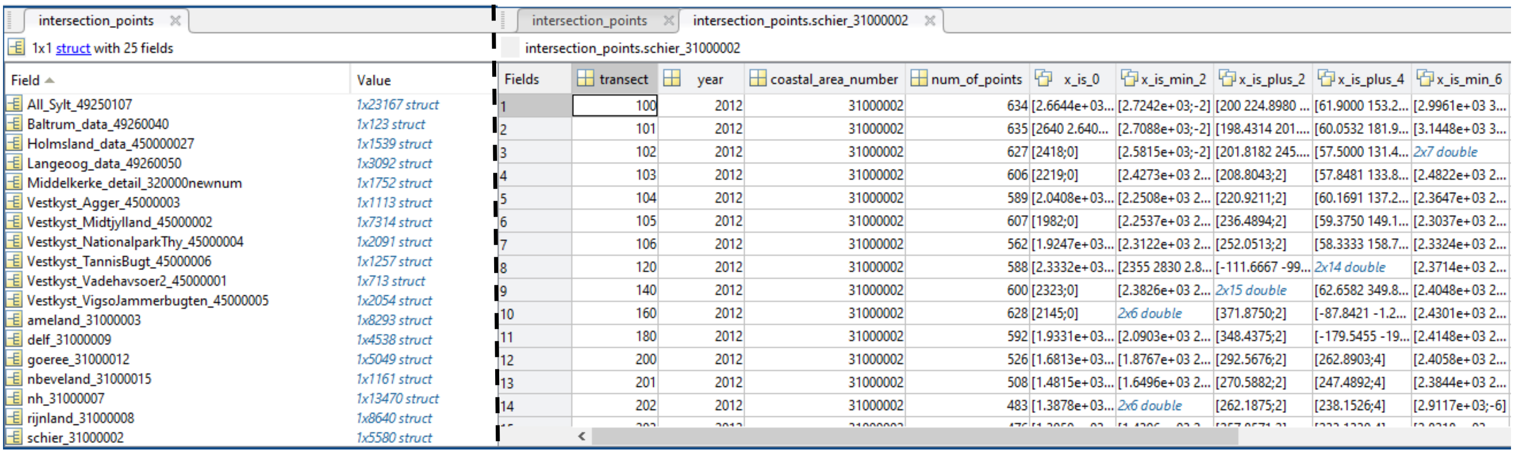


Figure 5: Two identical cross-shore profiles with different horizontal refence points.

3.1.2 Height points/intersection points

As stated above, to be able to determine the characteristics, certain points on the profile must be known; height points. The height points are the points on a profile where a horizontal line at a certain (chosen) height intersects with the profile. These points indicate where the profile has a certain elevation. Characteristics of a profile can be determined with the use of several height points, as described further in this chapter. Unfortunately, as not all measurements extended to a greater depth than -8 meters, the deepest point which was used in the calculations is -8 m. The intersection points of the profile with the horizontal lines at heights of +4, +2, 0, -2, -4, -6 and -8 m were determined for all transect measurements of each coastal area and saved in a MATLAB structure array. This was done with the *Calculate_intersection_points.m* script (Appendix Script 5). If there were multiple intersections of a profile with the same horizontal height line all intersection points were saved. Which intersection point (the most landward, most seaward or in between) was used during the calculation of the characteristics is described in the relevant sections.



intersection_points

1x1 struct with 25 fields

Field	Value
All_Sylt_49250107	1x23167 struct
Baltrum_data_49260040	1x123 struct
Holmsland_data_450000027	1x1539 struct
Langeoog_data_49260050	1x3092 struct
Middelkerke_detail_320000newnum	1x1752 struct
Vestkyst_Agger_45000003	1x1113 struct
Vestkyst_Midtjylland_45000002	1x7314 struct
Vestkyst_NationalparkThy_45000004	1x2091 struct
Vestkyst_TannisBugt_45000006	1x1257 struct
Vestkyst_Vadehavsoer2_45000001	1x713 struct
Vestkyst_VigsoJammerbugten_45000005	1x2054 struct
ameland_31000003	1x8293 struct
delf_31000009	1x4538 struct
goeree_31000012	1x5049 struct
nbeveland_31000015	1x1161 struct
nh_31000007	1x13470 struct
rijnland_31000008	1x8640 struct
schier_31000002	1x5580 struct

intersection_points.schier_31000002

Fields	transect	year	coastal_area_number	num_of_points	x_is_0	x_is_min_2	x_is_plus_2	x_is_plus_4	x_is_min_6
1	100	2012	31000002	634	[2.6644e+03...	[2.7242e+03;-2]	[200 224.8980 ...	[61.9000 153.2...	[2.9961e+03 3...
2	101	2012	31000002	635	[2640 2.640...	[2.7088e+03;-2]	[198.4314 201....	[60.0532 181.9...	[3.1448e+03 3...
3	102	2012	31000002	627	[2418;0]	[2.5815e+03;-2]	[201.8182 245....	[57.5000 131.4...	2x7 double
4	103	2012	31000002	606	[2219;0]	[2.4273e+03 2...	[208.8043;2]	[57.8481 133.8...	[2.4822e+03 2...
5	104	2012	31000002	589	[2.0408e+03...	[2.2508e+03 2...	[220.9211;2]	[60.1691 137.2...	[2.3647e+03 2...
6	105	2012	31000002	607	[1982;0]	[2.2537e+03 2...	[236.4894;2]	[59.3750 149.1...	[2.3037e+03 2...
7	106	2012	31000002	562	[1.9247e+03...	[2.3122e+03 2...	[252.0513;2]	[58.3333 158.7...	[2.3324e+03 2...
8	120	2012	31000002	588	[2.3332e+03...	[2355 2830 2.8...	[-111.6667 -99...	2x14 double	[2.3714e+03 2...
9	140	2012	31000002	600	[2323;0]	[2.3826e+03 2...	2x15 double	[62.6582 349.8...	[2.4048e+03 2...
10	160	2012	31000002	628	[2145;0]	2x6 double	[371.8750;2]	[-87.8421 -1.2...	[2.4301e+03 2...
11	180	2012	31000002	592	[1.9331e+03...	[2.0903e+03 2...	[348.4375;2]	[-179.5455 -19...	[2.4148e+03 2...
12	200	2012	31000002	526	[1.6813e+03...	[1.8767e+03 2...	[292.5676;2]	[262.8903;4]	[2.4058e+03 2...
13	201	2012	31000002	508	[1.4815e+03...	[1.6496e+03 2...	[270.5882;2]	[247.4892;4]	[2.3844e+03 2...
14	202	2012	31000002	483	[1.3878e+03...	2x6 double	[262.1875;2]	[238.1526;4]	[2.9117e+03;-6]

Figure 6: Left: An example of the first layer of a structure in which the intersection points are saved. Right: the second layer of the structure, containing the actual intersection points for a certain coastal area, in this case for Schiermonnikoog.

Figure 6 is an example of the structure in which the intersection points were saved (left side of the figure). Mind: the structure fields of the structure are also structures for each coastal area. And the fields of these structures (right side of the figure, example for Schiermonnikoog) contain all the intersection points for each transect within that coastal area under a certain name. Table 4 gives an overview of the names given to all the intersection point variables used in the structure.

Furthermore, the fields of the structures within the structure also contain information about the transect; transect number, measurement year, coastal area number and number of measuring points (Fig. 6).

Table 4: Overview of the variable names given to each intersection point variable used in the structure.

Height of the horizontal line	*Variable name in the struct*
+4	x_is_plus_4
+2	x_is_plus_2
0	x_is_0
-2	x_is_min_2
-4	x_is_min_4
-6	x_is_min_6
-8	x_is_min_8

3.1.3 Widths and Slopes

Now that the positions of certain height points in the profiles were determined, the distances between these height points could be calculated. This can give information about the width and slope of parts of the profile. For instance, figure 7 indicates the width (W_b) and averaged slope (β_b) of the beach (including the intertidal beach), between height points +4 and -2 meter. The W_b and β_b can be determined by finding the intersection points of the profile with those height lines and determine the distance between those points (Fig. 7), which can be used to determine the slope. Furthermore, the width (W_{sf}) and averaged slope (β_{sf}) of the shoreface can be determined between two height points. For example, in figure 8 the width and slope between height points -2 and -10 meter are indicated. The shoreface slope could also be determined in multiple smaller parts, for example in two parts: one in the active bar zone, and the other one seaward of the last bar (so outside of the active bar zone).

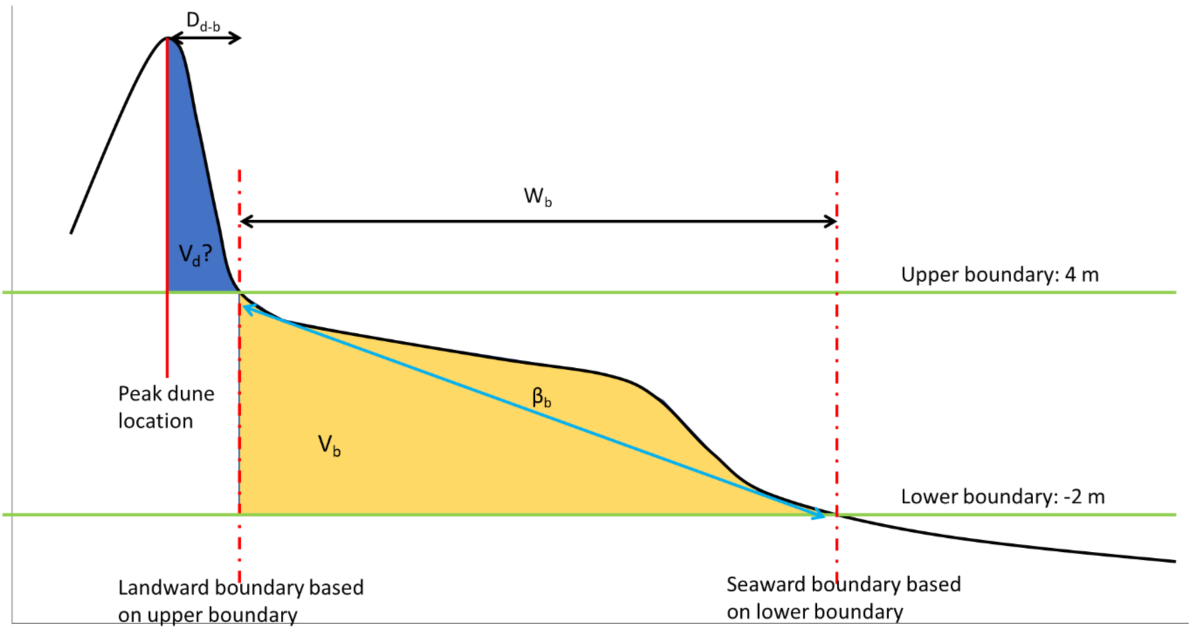


Figure 7: A schematic overview of morphological characteristics in the upper part (intertidal to dunes) of a cross-shore profile.

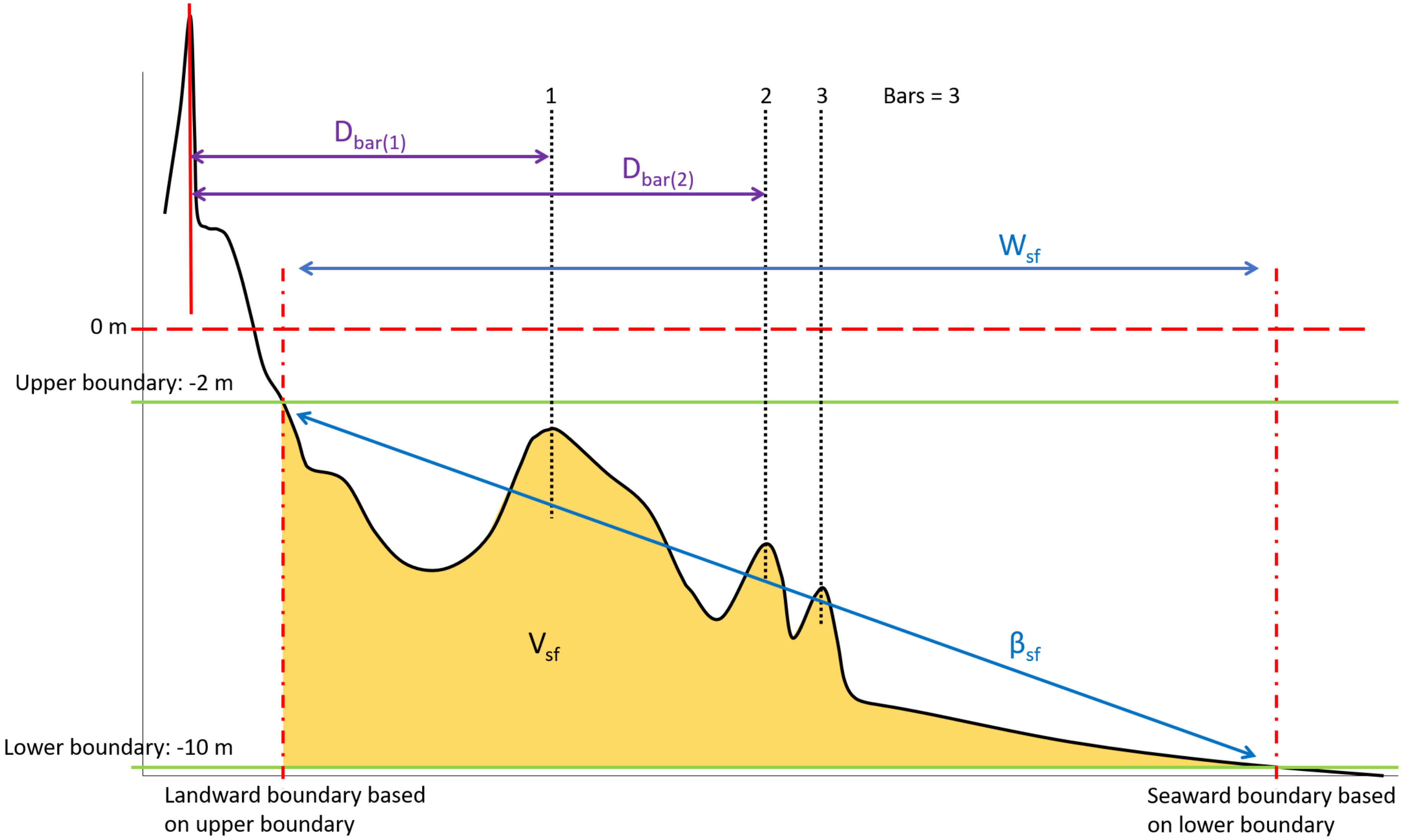


Figure 8: A schematic overview of some morphological characteristics of a cross-shore profile.

MATLAB scripts (Appendix Scripts 6 and 7) were written to determine the width and slope between certain height point positions (Table 5). The calculated widths and slopes were stored in a MATLAB structure array similar to the intersection points (Fig. 6). Having a structure for each coastal area within the structure, containing information about the transect; transect number, measurement year, coastal area number and number of measuring points as well as the values of the width/slope variables.

Some height points occur at multiple locations, therefore, for some height points, more positions can be used to calculate the width. Which height position was chosen during the calculation of each width variable is stated in Table 5. Basically, for the height points +4, +2 and 0 the most seaward located height points were taken, while for the height points -2, -4, -6 and -8 the most landward located height points were taken. This was done in an attempt to use the points which are closest to the MSL position. However, the most seaward location was used for the -4 height point and the most landward location was used for the -8 height point for the calculation of the width and slope between height points -4 and -8. This was done to determine the slope of the most seaward part of the shoreface which was measured, while avoiding the influence of sandbars on the results (finding points of -4 and -8 which are adjacent). If the most seaward point of -4 was located further seaward than the -8 point, the first -4 point located landward of the -8 point was found and used. If the result of any width calculation gave a negative width, the variable was left blank as this is not possible. Furthermore, Appendix Table 1 contains the names of each width and slope variable, within the structure, which are used in this project.

Additionally, the mean (time-averaged) widths and slopes for each part of the profiles was calculated. This was done by taking the width and slope values of every measurement done at each transect and average them. Besides the mean widths and slopes over the whole history of measurements, the means were also determined for the period 2006-2016. This was done to get (or at least attempt to get) equivalent results for all the transects, as some transects were monitored consistently for over 50 years while others were not (Fig. 9). Although, even for the period 2006-2016, the amount of annual measurements done per transect is not equal for all transects. This is also indicated in figure 9, where the dotted lines indicate periods during which some measurements were done, though not annually. So, for some coastal areas (vad, thy, vig and tan) only a few measurements were done during the period 2006-2016. Furthermore, the standard deviation was also determined along with the means, to indicate the amount of variation over time. The calculation of the means was only done for each transect which was measured in 2016, except for the coastal areas in Denmark, because most of the Danish transects were not measured in 2016 (Fig. 9). To get all the widths (throughout the measurement history) of one transect, the index numbers (location) on which these widths are saved in the structure, for that specific transect, must be known. The function *GetStructIndex* (Appendix Function 2) does this for all measurements measured for a given year. It has been decided to only use all the transects which were measured during the same year, because otherwise it was too difficult to select all the results for each transect. As mentioned, the index of each transect which was measured in 2016 was used. Except for the coastal areas in Denmark for which a different year was taken, because a lot of the Danish transects were not measured in 2016. The year taken for each Danish coastal area is based on the amount of transects measured during that year: Vadehavsoer: 2014, Midtjylland: 2014, Agger: 2016, Nationalpark-thy: 2009, VigsoJammerbugten: 1995, Tannis-Bugt: 2008 and Holmsland: 2014. The mean width and slope variable names used in the structure can be found in Table 5.

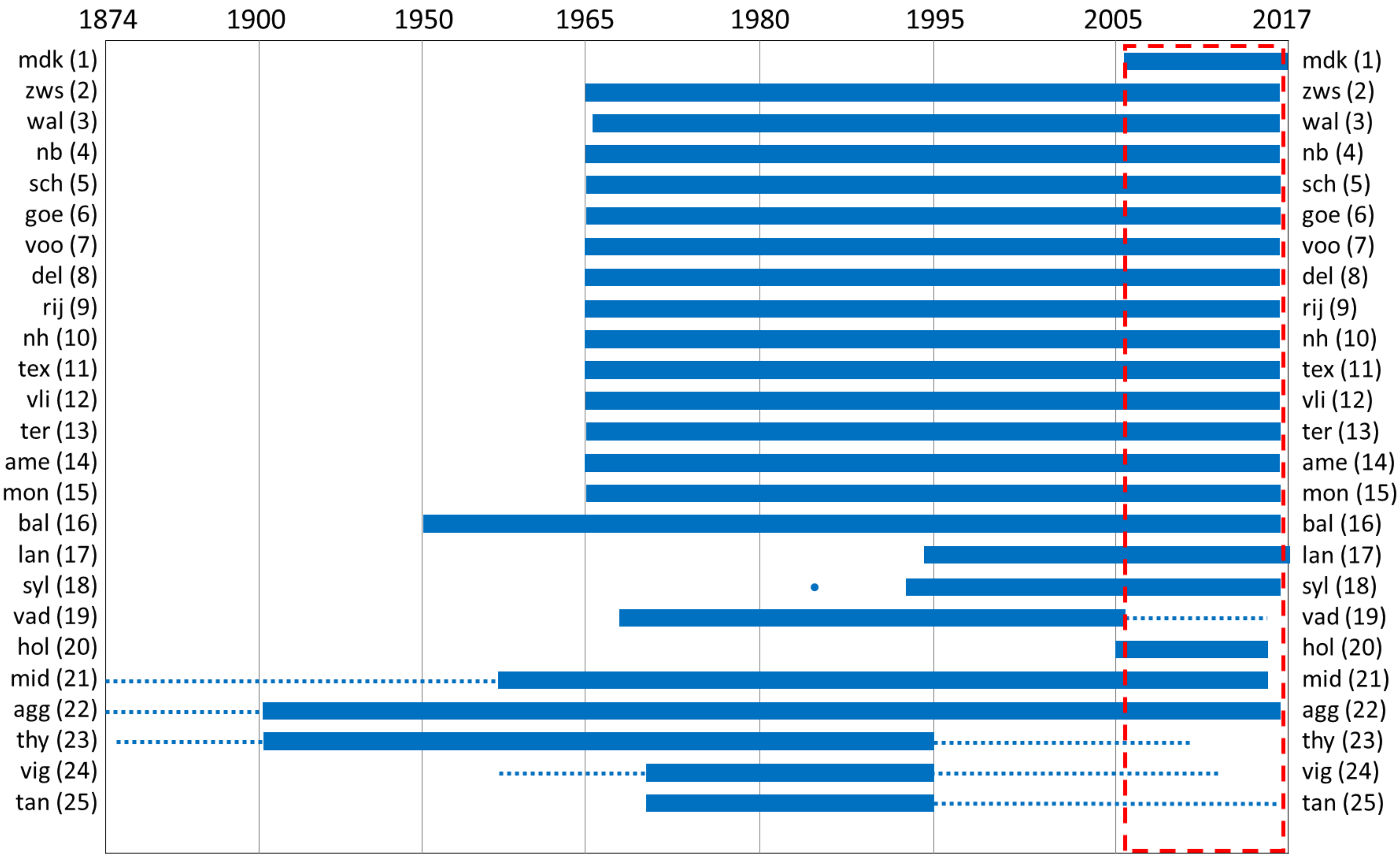


Figure 9: An overview indicating the periods over which the transects were measured per coastal area. The bars indicate periods with (almost) consistent annual measurements. The dotted lines indicated periods over which some measurements were done but not consistently annually. The dot (coastal areas Sylt) indicates a single measurement done in 1984. Note: the year spacing at the x-axis is not consistent.

Table 5: Mean width and slope variable names used in the structures within the main structure. Note: the variable names of the means from the period 2006-2016 are the same as the names in the table but with the extension '_2006_2016'.

1st height point (m)	2nd height point (m)	Mean width variable name	Mean slope variable name
+4 most seaward	0 most seaward	mean_width_0plus4	mean_slope_0plus4
+2 most seaward	0 most seaward	mean_width_0plus2	mean_slope_0plus2
+4 most seaward	-2 most landward	mean_width_min2plus4	mean_slope_min2plus4
+2 most seaward	-2 most landward	mean_width_plus2min2	mean_slope_plus2min2
0 most seaward	-2 most landward	mean_width_0min2	mean_slope_0min2
-2 most landward	-4 most landward	mean_width_min2min4	mean_slope_min2min4
-2 most landward	-6 most landward	mean_width_min2min6	mean_slope_min2min6
-2 most landward	-8 most landward	mean_width_min2min8	mean_slope_min2min8
-4 most seaward	-8 most landward	mean_width_min4min8	mean_slope_min4min8

3.1.4 Volumes

The volume of, for example, the beach (V_b) can be determined as the volume under a cross-shore profile between two height points (points where the profile intersects with a horizontal line at a chosen height). For example, in figure 7 y = +4 m and y = -2 m are taken, since the whole beach, including intertidal beach, is generally located between these points. This can be done by finding the intersection points of the profile with both height lines and determining the volume between those heights (upper and lower boundary in figure 7) and between those points (land- and seaward boundaries based on upper and lower boundaries, respectively, in figure 7). Note the potential problems with determining the positions of a depth point, as one depth can occur at multiple locations in one profile (see Chapter 6). Also note the influence of a seaward propagating (growing) dune on the volume of the beach when using this calculation method, further explained in Chapter 6. Furthermore, the volume of the shoreface (V_{sf}) can be determined between two height points. For example, in figure 8 y = -2 m and y = -10 m are taken. The calculation of the volume can be done with a similar approach as the calculation of the V_b. The shoreface might also be divided into an upper and lower part, as these often differ in morphology. When coupling the volume(s) of (the parts of) the shoreface to the averaged slope(s), the general shape of the profile might be determined. If a volume is less/more than the 'triangle' volume, determined as the volume below the straight line between two height points, the shape of the profile is generally convex/concave.

With the use of the determined height points (Section 3.1.2) and the data of the measured profiles, the volume between two height points could be determined for all the measured transects. A MATLAB script (Appendix Script 8) was developed to determine the volumes between certain height points. With the use of the Matlab OpenEarth Tools from Deltares a function was found which could calculate the volume under a profile between specified boundaries (Fig. 10). The function is used in the script which calculates the volumes and stores the results into a structure similar to the widths and slopes. Between which height points the volumes were calculated, along with the names given to the variables inside the structure, is stated in Appendix Table 2. The mean (time-averaged) volumes, over the whole measurement history and over the period 2006-2016, were also determined in the same way as the means of the widths and slopes. The mean volume variable names used in the structure can be found in Table 6. After the volumes were calculated, the trends in the volume changes over the period 2006-2016 were determined. Per transect profile a linear trend was fitted through the volume values, one for each year in this period (if there was data available), and the slope of this linear trend was stored in a structure. This slope indicates the amount of sediment volume change (in m^3), erosion or deposition, that occurred on average per year over that period. The names of these trend variables are also presented in Appendix Table 2.

Table 6: Mean volume variable names used in the structures within the main structure.

1st height point (m)	2nd height point (m)	Mean volume variable name	Mean volume variable name period 2006-2016
+4 most seaward	0 most seaward	mean_vol_0plus4	mean_vol_0plus4_2006_2016
+2 most seaward	0 most seaward	mean_vol_0plus2	mean_vol_0plus2_2006_2016
+4 most seaward	-2 most landward	mean_vol_min2plus4	mean_vol_min2plus4_2006_2016
+2 most seaward	-2 most landward	mean_vol_plus2min2	mean_vol_plus2min2_2006_2016
-2 most landward	-4 most landward	mean_vol_min2min4	mean_vol_min2min4_2006_2016
-2 most landward	-6 most landward	mean_vol_min2min6	mean_vol_min2min6_2006_2016
-2 most landward	-8 most landward	mean_vol_min2min8	mean_vol_min2min8_2006_2016

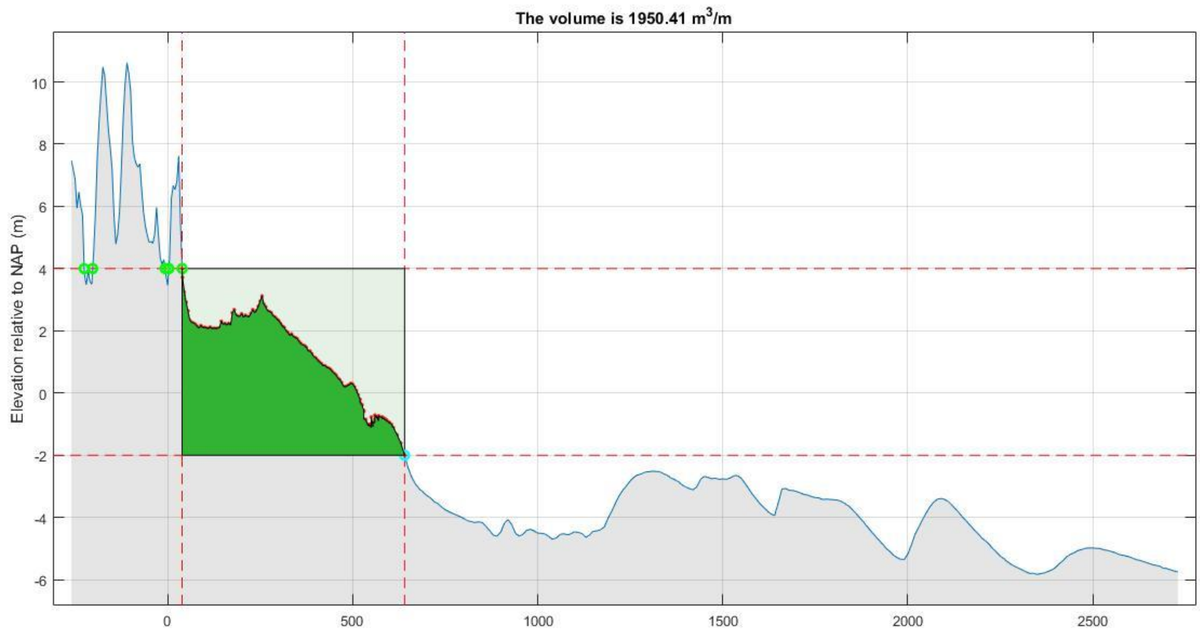


Figure 10: A figure representing how the volume is calculated for one transect. The horizontal dashed lines indicate the upper and lower boundaries, whereas the vertical dashed lines indicate the land and seaward boundaries. The volume is determined between the boundaries indicated by the green area inside the rectangle.

Figure 11 was made to indicate the relative differences, due to different axis ranges, between the figure displaying the full profile (Fig. 8) and the zoomed in figure (Fig. 7). This figure is also an overview of several coastal characteristics. Mind, some of the characteristics indicated in the figures (7, 8 and 11) were not calculated and are discussed as potential characteristics in the analysis of this dataset, in Chapter 5. Note: both axes are exaggerated in the figures, where the x-axis is more exaggerated than the y-axis. Furthermore, the y-axis is more exaggerated for the full profile figures than for the zoomed in figures (Fig. 11).

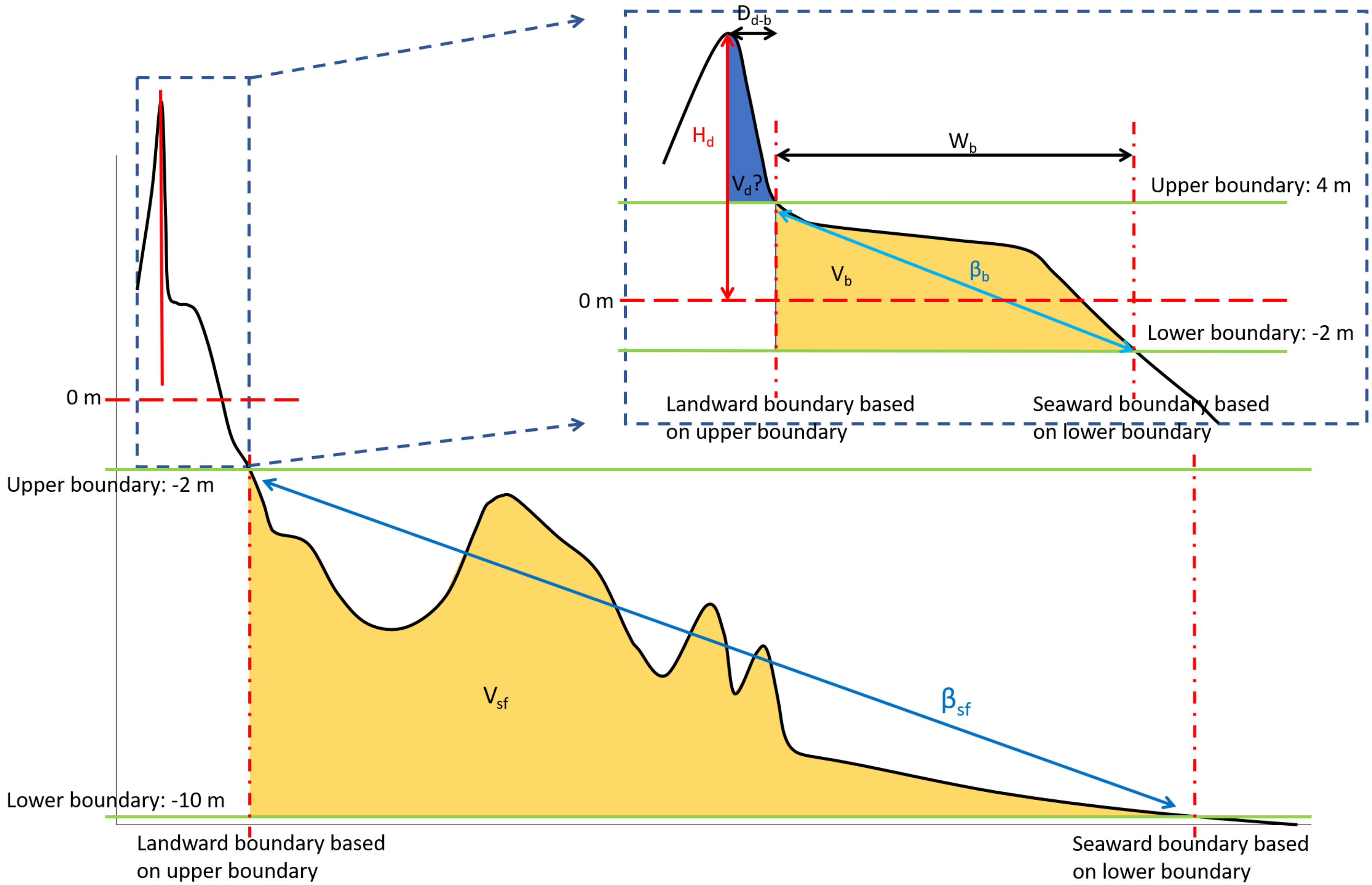


Figure 11: A schematic overview of a cross-shore profile along with a zoom in on the upper part of the profile to indicate the relative differences, due to different axis ranges, between figures 7 and 8.

3.2 Plotting characteristics

3.2.1 Overview figures

To analyse the characteristics of the profiles, the results of the calculations must be visualised. Using MATLAB, scripts were written to plot the characteristics (Appendix Section 3.6), in an order from south to north, of each transect of each coastal area. The plotted results are the mean (time-averaged) values of a specific characteristic for each transect. The means were determined over the full history of measurements and for the period 2006-2016 (Section 3.1). They are used to indicate the average state of each profile, rather than only a snapshot. The means were also calculated for the period 2006-2016 to get (or at least attempt to get) equivalent results for all the transects, as some transects were monitored for over 50 years while others were not (Fig. 9). Although, even for the period 2006-2016, the amount of measurements done per transect is not equal for all transects, as some were not measured annually. Figures 14 and 15 indicate the mean slopes for the full history of measurements and for the period 2006-2016, respectively. The transects are plotted in different colours based on the mean slope values to amplify the differences in the mean slopes. The mean slope (and width, Appendix figures 1 and 2) value ranges for each colour are specified in Table 7. In addition, the standard deviations of the slopes were also plotted (Appendix figures 3 and 4), indicating the amount of variation/dynamics in the profiles.

Table 7: Table indicating which value ranges are plotted in which colour per characteristic, for the figures 14 and 15.

+4 & 0	**Slope range (1/xx)**	**Width range (m)**	**Colour**		**+2 & -2**	**Slope range (1/xx)**	**Width range (m)**	**Colour**
	0 – 20	0 – 75	Red			0 – 25	0 – 100	Red
	20 – 40	75 – 150	Magenta			25 – 50	100 – 175	Magenta
	40 – 60	150 – 225	Blue			50 – 75	175 – 250	Blue
	60 – 80	225 – 300	Cyan			75 – 100	250 – 325	Cyan
	80 – 100	300 – 375	Green			100 – 125	325 – 400	Green
	100+	375+	Yellow			125+	400+	Yellow
-2 & -4	**Slope range (1/xx)**	**Width range (m)**	**Colour**		**-4 & -8**	**Slope range (1/xx)**	**Width range (m)**	**Colour**
	0 – 30	0 – 75	Red			0 – 50	0 – 300	Red
	30 – 60	75 – 150	Magenta			50 – 100	300 – 600	Magenta
	60 – 90	150 – 225	Blue			100 – 150	600 – 900	Blue
	90 – 120	225 – 300	Cyan			150 – 200	900 - 1200	Cyan
	120 – 150	300+	Green			200+	1200+	Green
	150+	-	Yellow			-	-	Yellow

As described in Section 3.1.4, the linear trends in the volume changes over the period 2006-2016 were determined, for selected parts of the profiles. The slopes of the trends, indicating the average amount of sediment volume change per year over that period, were saved. The volume trend figure (Fig. 16) indicates these linear trends in the volume change per year, in an order from south to north, for each transect of each coastal area. The transects are plotted in different colours, based on the values of the trend in volume change, to amplify the differences in the trends. These value ranges for each colour are specified below in Table 8.

Table 8: Table indicating which trend volume value ranges are plotted in which colour per characteristic, for figure 16.

+4 & 0	Trend Vol range (m³/yr)	Colour	+2 & -2	Trend Vol range (m³/yr)	Colour
	< -40	Red		< -40	Red
	-40 – -10	Yellow		-40 – -10	Yellow
	-10 – 10	Black		-10 – 10	Black
	10 – 40	Dark green		10 – 40	Dark green
	40+	Light green		40+	Light green
-2 & -4	**Trend Vol range (m³/yr)**	**Colour**	**-2 & -8**	**Trend Vol range (m³/yr)**	**Colour**
	< -40	Red		< -40	Red
	-40 – -10	Yellow		-40 – -10	Yellow
	-10 – 10	Black		-10 – 10	Black
	10 – 40	Dark green		10 – 40	Dark green
	40+	Light green		40+	Light green

3.2.2 Boxplots

For the slopes, calculated for the chosen section of each transect, boxplots are made to indicate the variation in the slopes per transect over the whole measurement history and the period 2006-2016. A small description of the script generating the boxplots is stated in Appendix section 2.2. The boxplots are plotted per coastal area, as they become unreadable when plotting more data points (Figures 17-24, Section 4.2). Although, when plotting the boxplots of only one coastal area the details of the plots are also hard to distinguish. Therefore, an example of one boxplot is shown in figure 12. The MATLAB function *boxplot* was used to plot the data in boxplots. What the boxplot shows: the central horizontal line indicates the median (not visible in the figures due to the number of boxplots in one figure), the top and bottom edges of the box show the 75th and 25th percentiles, respectively, the whiskers (also not visible in the figures, these mark the end of the dashed lines) extend to the most extreme data points which are not considered as outliers, and the outliers are marked with a '+' symbol. The *boxplot* function sets a data point as an outlier if it is greater than $q_3 + w \times (q_3 - q_1)$ or less than $q_1 - w \times (q_3 - q_1)$, where w is the maximum whisker length, q_1 is the 25th and q_3 is the 75th percentile of the data. The whisker value is approximately +/- 2.7σ by default which has a 99.3 percent coverage if the data are normally distributed. The plotted whisker extends to the 'adjacent value', which is the most extreme value that is not considered as outlier (MathWorks, 2017).

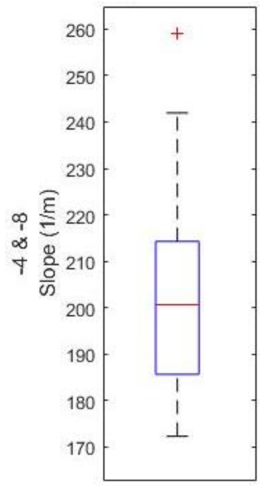


Figure 12: An example of one single boxplot

3.2.3 Removing island heads from figures

The objective is to analyse differences between the main parts of the coast. Therefore, a tool was fabricated as a function in MATLAB to remove the transects located at the island heads and some other locations which were clearly not located on the straight coast (like behind a dam), from the figures. Appendix table 3 specifies for each coastal area which transects were removed from the figures and the reason to remove them. The function uses the name of the coastal area used for the structure names within the structure to determine which transects are removed from the figure, therefore the names of the structures are advised to keep the same. Although, some names might be a little unusual as they were used as working names. The names used for each coastal area in the structure are found in Table 3. For the plotting of the boxplots a variation of the function is used to split the island heads, as the variables for the boxplot needed to be arrays (all the annual values for the chosen period) instead of only one (mean) value.

3.2.4 Categorization

One goal of this project is to investigate if parts of the North Sea coast can be distinguished from each other based on certain profile characteristics. A way of doing this is by assigning the transects to a category based on the characteristics. The method used during this project is making categories based on the slopes of certain parts of the profile, where the relative slope of one part to the adjacent parts was determined to check which part is steeper. Because, if the relative differences in slopes between small parts of the whole profile are known, a schematic representation of a that profile can be made, indicating the general shape of the profile. An extra step can be to split the profiles, which have the same general shape (placed in the same category), based on the actual slope values of certain parts. More ways of doing these categorizations are proposed in Chapter 5. Here 5 distinct categories are made based on the mean slopes between the following parts: +4 & 0, +2 & -2, -2 & -4 and -4 & -8. Where the 5 categories are (Table 9):

Table 9: Table indicating the 5 categories in which the transects were subdivided. Note: the ‘>’ sign (red) indicates that the left slope is steeper than the right slope, and the ‘<’ sign (green) indicates that the left slope is gentler than the right slope.

Category #	Slope		Slope		Slope		Slope
1	+4 & 0	>	+2 & -2	>	-2 & -4	>	-4 & -8
2	+4 & 0	>	+2 & -2	<	-2 & -4	>	-4 & -8
3	+4 & 0	>	+2 & -2	>	-2 & -4	<	-4 & -8
4	+4 & 0	>	+2 & -2	<	-2 & -4	<	-4 & -8
5 (other)	+4 & 0	<	+2 & -2		-2 & -4		-4 & -8

The schematic representations of the shape of the profiles that were distributed in the first 4 categories are illustrated in figure 13. Note: both axes of the figures are exaggerated, where the x-axis is more exaggerated than the y-axis.

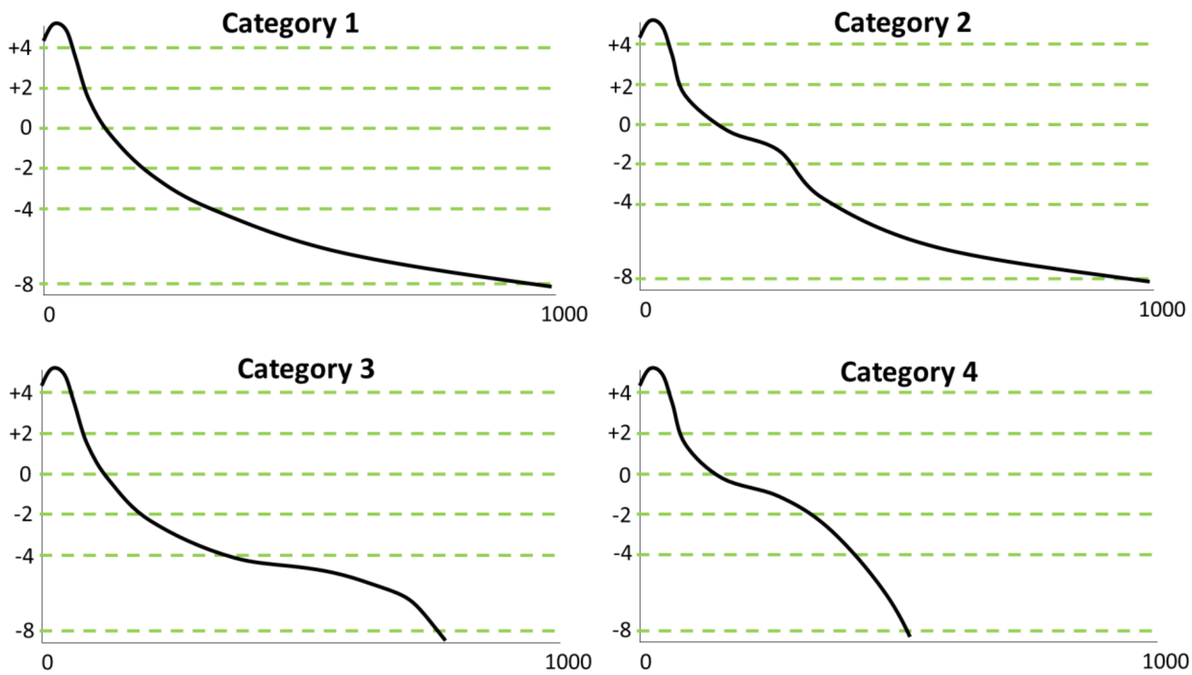


Figure 13: A schematic representations of the shape of the profiles in the 4 categories.

On top of this categorization, a second categorization was added. This one was done based on the mean slope value of the part between height points -4 and -8, to distinguish between profiles with a steeper/gentler slope in the deeper part of the shoreface. Four more categories were made (Table 10), where each category has a colour in which it is plotted in the resulting categories figure (Section 4.3, Fig. 26):

Table 10: Table indicating the 4 extra categories and the colour in which each category is plotted in the figure (Fig. 26).

Category #	**Slope (-4 & -8)**	**Colour in figure**
1: Very gentle slope	Gentler than 1/200	Green
2: Medium-gentle slope	Between 1/200 and 1/125	Blue
3: Medium-steep slope	Between 1/125 and 1/50	Magenta
4: Very steep slope	Steeper than 1/50	Red

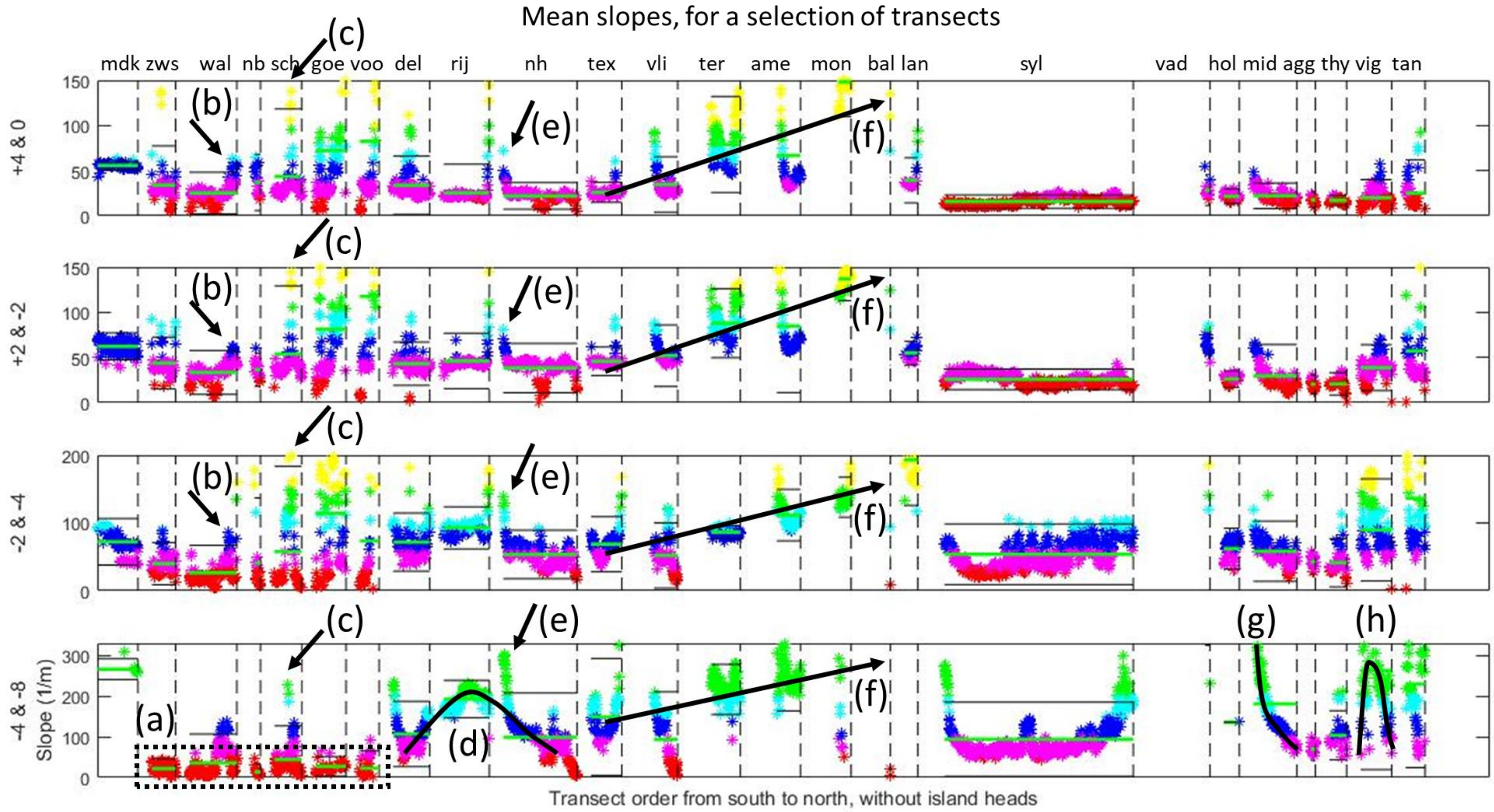


Figure 14: Figure indicating the mean slopes over the whole measurement history along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Table 7. The horizontal green line, per coastal area indicates the mean of the mean slopes of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean. Indicated with letters are areas discussed in Section 4.1.1.

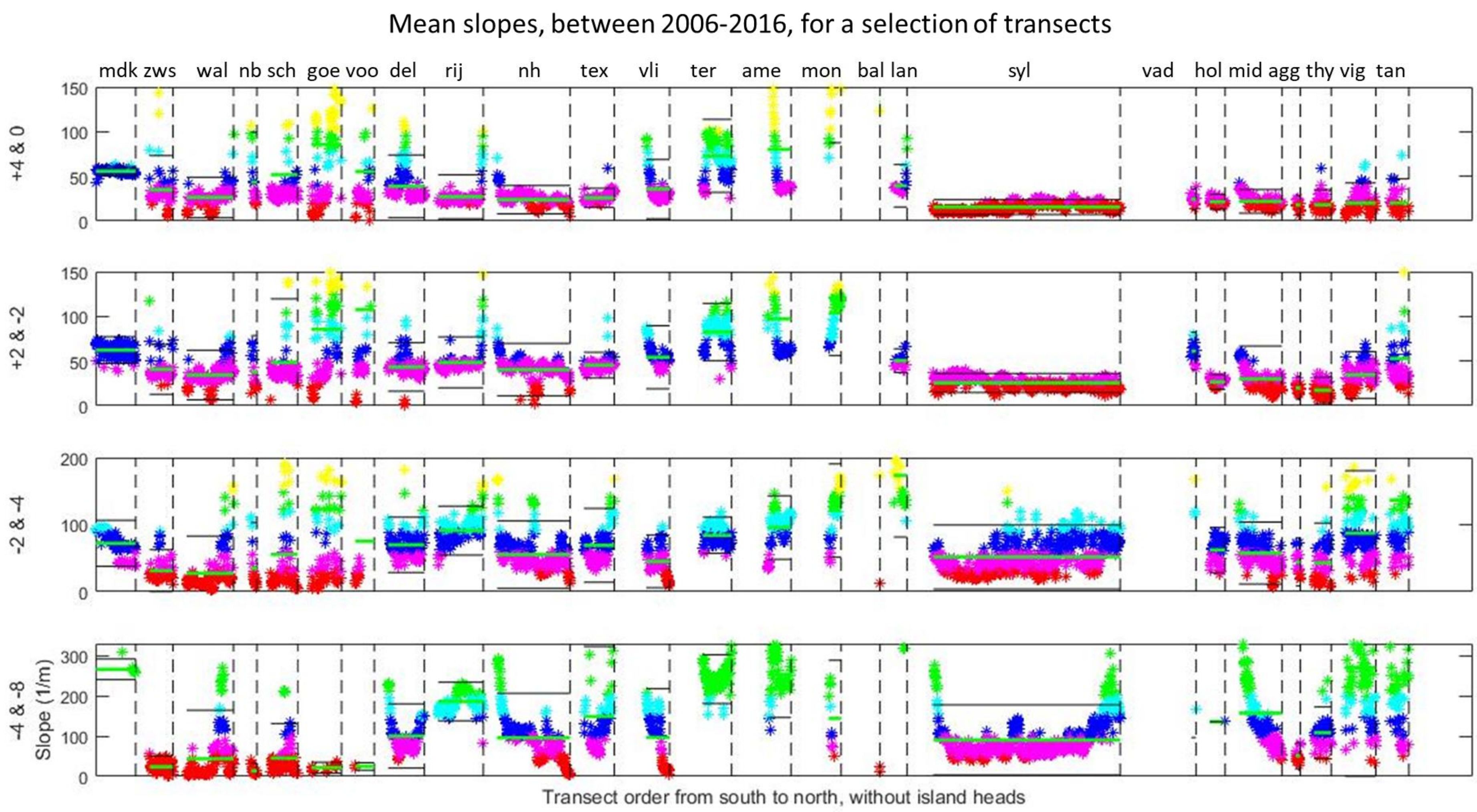


Figure 15: Figure indicating the mean slopes over the period 2006-2016 along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Table 7. The horizontal green line, per coastal area indicates the mean of the mean slopes of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean.

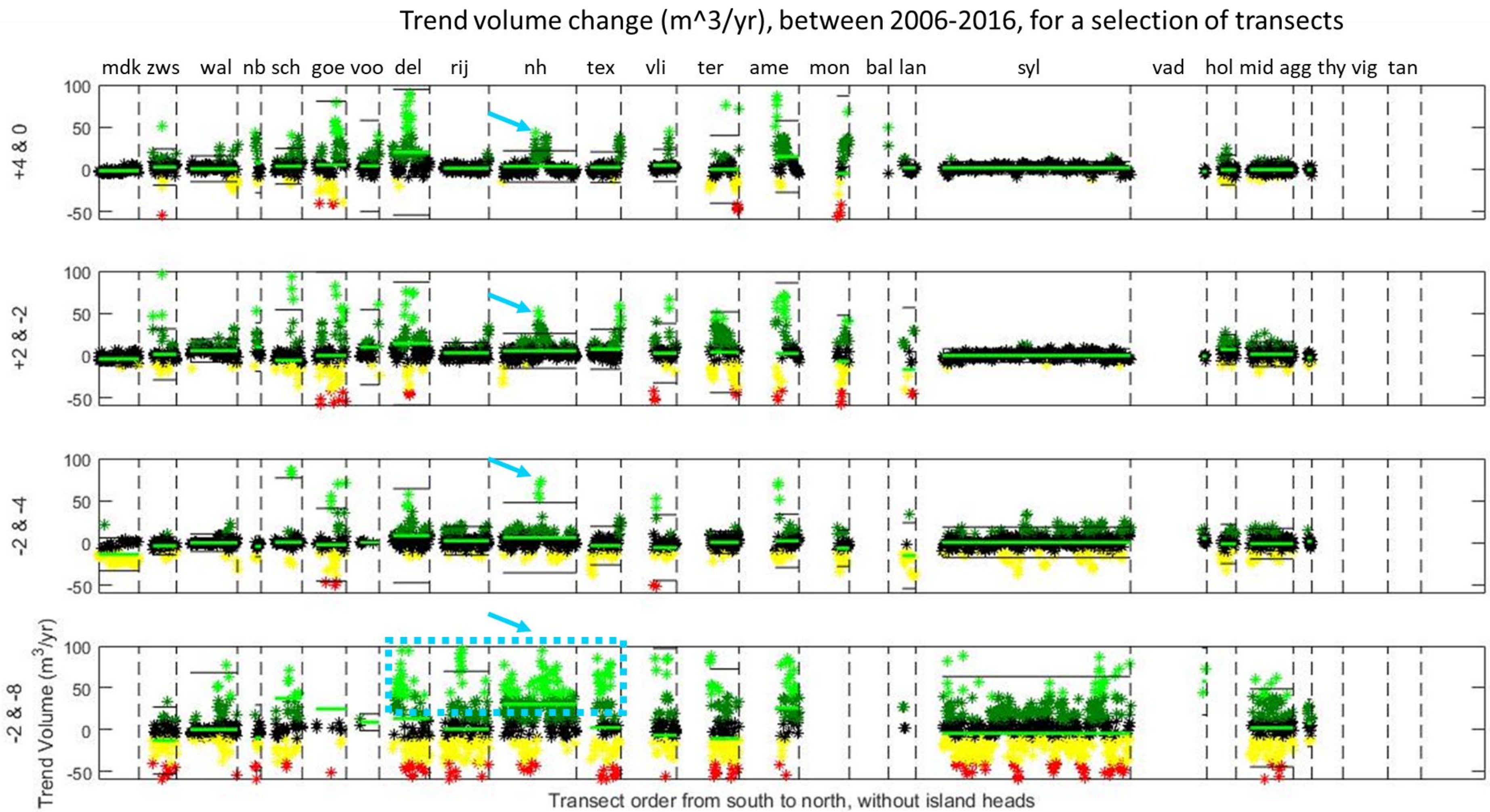


Figure 16: Figure indicating the trend in the volume change per year over the period 2006-20016 along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Table 8. The horizontal green line, per coastal area indicates the mean of the trend in the volume change per year of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean. The areas indicated with the blue box and arrows are discussed in Section 4.1.2.

4. Analysis of the characteristics

The results from the calculation of the characteristics are analysed in this section. The analysis of the characteristics can be done quite extensively. Though, due to the lack of available time, the analysis during this project is limited. Chapter 5 discusses about some of the potential analyses that could be done in the future with this dataset and the calculated characteristics, including possible characteristics that are yet to be calculated.

A selection of the figures is discussed here. First the 'overview' figures are treated, which give an overview of the characteristic values along the North Sea coast, from south to north. These include the mean (time-averaged) slope figures (Section 4.1.1) and the figure indicating the trend in the volume change over the period 2006-2016 (Section 4.1.2). Furthermore, some coastal areas with notable results from the 'overview' figures, are looked at in greater detail. By looking at the corresponding boxplots (Section 4.2). The boxplots indicate the variation of the slope value of a certain part of the profile. Finally, the basic categorization is analysed (Section 4.3).

4.1 Analysis of the overview figures

4.1.1 Mean slope

Figure 14 shows the mean slopes over the whole measurement history, for a selection of transects (specified in Section 3.1.3), for each coastal area along this part of the North Sea coast in an order from south to north. Whereas figure 15 shows the means slopes of only the period 2006-2016. The most striking observations in the figures are discussed here, in an order from south to north.

Middelkerke

Starting from the south, the Middelkerke coastal area has a gentler slope, for all the parts of the profile, compared to most other coastal areas. Especially the deeper part is very gentle, indicating a quite flat coastal shelf.

Zeeland

The deep parts of the coastal areas of the province of Zeeland have very steep slopes compared to almost all other coastal areas (indicated with (a) in figure 14), except some profiles (outside of Zeeland) near tidal inlets. This is probably due to the tidal channels located next to these coastal areas. At the far north part of Walcheren, the slopes become gentler (indicated with (b) in figure 14), this likely due to a curve in the coastline which allows the sediment to accumulate easier. For Schouwen, in the middle of the area, a sort of spit is attached to the coast, decreasing (even) the mean slope steepness at all ranges (indicated with (c) in figure 14), which suggests that this feature occurs often at this point (also very dynamic point, see variation Section 4.2). In the north part of Goeree the slopes of the upper part of the profile are very gentle, while this is not the case for the deepest part of the profile. Furthermore, the -2 & -4 slope indicates a lot of variation in slope along the whole coastal area.

Dutch mainland

When looking at the overall trend in slopes along the coast, the trend along the coastal areas Delfland, Rijnland and Noord-Holland is quite outstanding for the somewhat deeper parts. Here, clearest noticeable for the -4 & -8 slope, the mean slope trend a bit towards the south, in Delfland, is quite steep, then becomes gradually gentler towards the north, after which the slope steepens again (indicated with (d) in figure 14). This can be explained by the old sediment deposition fan of the former Rhine discharge mouth, which was located around the middle of the Rijnland coastal area (Stouthamer et al., 2010). The effect of the harbour jetties of Ijmuiden, marking the border of Rijnland and Noord-Holland, is clearly visible on the slopes, which are gentler near the jetties (indicated with (e) in figure 14). The far north part of Noord-Holland is steeper due to the nearby tidal inlet.

Wadden islands

The slopes of all the parts of the profile seem to become generally gentler going from Wadden island to island (indicated with (f) in figure 14), until Langeoog (note: some transects of Schiermonnikoog are plotted higher than the y-axis maximum range). This might be due to a different orientation to the general wind field or due to tidal influences. Furthermore, the '*Afsluitdijk*' might affect this by causing sediment import to the Wadden sea, especially to the basins of the Texel and Vlieland inlets (Elias et al., 2012). Along the island of Sylt the slopes seem to remain quite constant, being fairly steep in the upper two parts of the profile, but average for the deepest two parts of the profile, compared to the other coastal areas.

Midtjylland

The coastal area Midtjylland indicates a gradual increase in slope steepness towards the north for all parts of the profile, which is quite extreme for the slope between the -4 and -8 height points (indicated with (g) in figure 14), as this slope is very gentle in the south of this area. This might be due to Horns Rev, which is a shallow area (a submerged shoal) off the coast at Blåvands Huk, the core of Horns Rev consists of a moraine from the Saalian glaciation (Aagaard, 2011). This feature possibly shields the southern end of Midtjylland from the waves of the main winds and therefore sand is able to accumulate there. Additionally, the longshore sediment transport along the coastal area Midtjylland is directed to the south, transporting sediment from the northern parts to the south where some sediment accumulates at the coast (Aagaard, 2011). Furthermore, at the far northern an inlet is located towards a fjord. This might serve as a sediment sink which would explain the steepness of the slope by decreasing the available sand in the nearby profiles, making them steeper.

VigsoJammerbugten

The slopes in coastal area VigsoJammerbugten, indicate an arch form, from south to north, being gentler in the middle of the coastal area compared to both sides (indicated with (h) in figure 14). This might be explained by the overall coastline shape of this area, as the coastline describes a nice curve (further described in Section 4.2). Making it possible that more sediment is accumulating in the middle area of this curve; the centre of the coastal area.

Comparison between coastal areas

Overall, the slopes higher up in the profile (+4 & 0, +2 & -2) are quite comparable for the whole Dutch mainland. For Flanders the slopes are gentler. Also, the slopes on the Wadden islands are gentler, which increases towards the northern/eastern islands, except for Langeoog and Sylt. The latter having even steeper slopes than the Dutch mainland. The Danish mainland has generally slightly steeper slopes compared to the Dutch mainland.

For the deeper parts this is different. Where, in general, the slopes in Flanders are quite gentle, comparable to the slopes on Terschelling, Ameland and Schiermonnikoog. Furthermore, the slopes of the coastal areas in the province of Zeeland are the steepest slopes of all the slopes along the coast. The mean slopes values of the Dutch mainland (Delfland, Rijnland and Noord-Holland), like mentioned above, describe a sort of arch (indicated with (d) in figure 14). Having comparable slopes to Texel, Vlieland, Sylt and some parts of the Danish coast at the sides of this area, whereas the middle of the area (gentler slopes) has slopes comparable to those found at Flanders, Terschelling, Ameland, Schiermonnikoog and other parts of the Danish coast. Along the Danish coast the mean slope values vary quite considerably, making it difficult to compare.

The general arch forms in the slope values, previously identified, along the Dutch mainland (Delfland, Rijnland and Noord-Holland) and along VigsoJammerbugten, are most likely not caused by the same mechanism. As the first one is caused by the remains of the deposited sediments from the old discharge mouth of the rhine, while the second one might be caused by the general shape of the coastline.

4.1.2 Volume trend

The volume trend figure (Fig. 16) indicates the linear trends in the volume change per year, taken over the period 2006-2016. For most coastal areas all the transect points seem to alternate above and below the zero-volume change, indicating a relative stable volume condition over a large scale. Although, in the middle of the Noord-Holland coastal area an area with a positive sediment volume trend is revealed (indicated with blue arrows in figure 16). Furthermore, the volumes of the higher (> -2m) parts of the profiles do not appear to change (at all) for Middelkerke, Rijnland, Sylt and large parts of the Danish coast. Large variations in the volume trend of the shoreface (between -2 m and -8 m, indicated with a blue box in figure 16) from coastal area Delfland to Texel, which overall seems to be a little more positive than negative (note the y-axis scale), might be caused by shoreface nourishments implemented in these coastal areas. In addition, the subtidal bar behaviour may very well also influence the dynamic volume trends of the deeper parts of the profiles.

4.2 Variation of the different slopes (boxplots)

The boxplots of some selected coastal areas, which show some interesting results discussed above, are discussed here. The selected coastal areas of interest are treated from south to north.

Starting at the coastal area Middelkerke, the boxplot is plotted for the slope of the +4 & 0 part of the profile (Fig. 17). The boxplot indicates that the slope of this part of the profile, between the +4 and 0 m height points, is rather stable between 1/50 to 1/60 for the whole coastal area. Furthermore, there is almost no variation between the transect profiles within this coastal area.

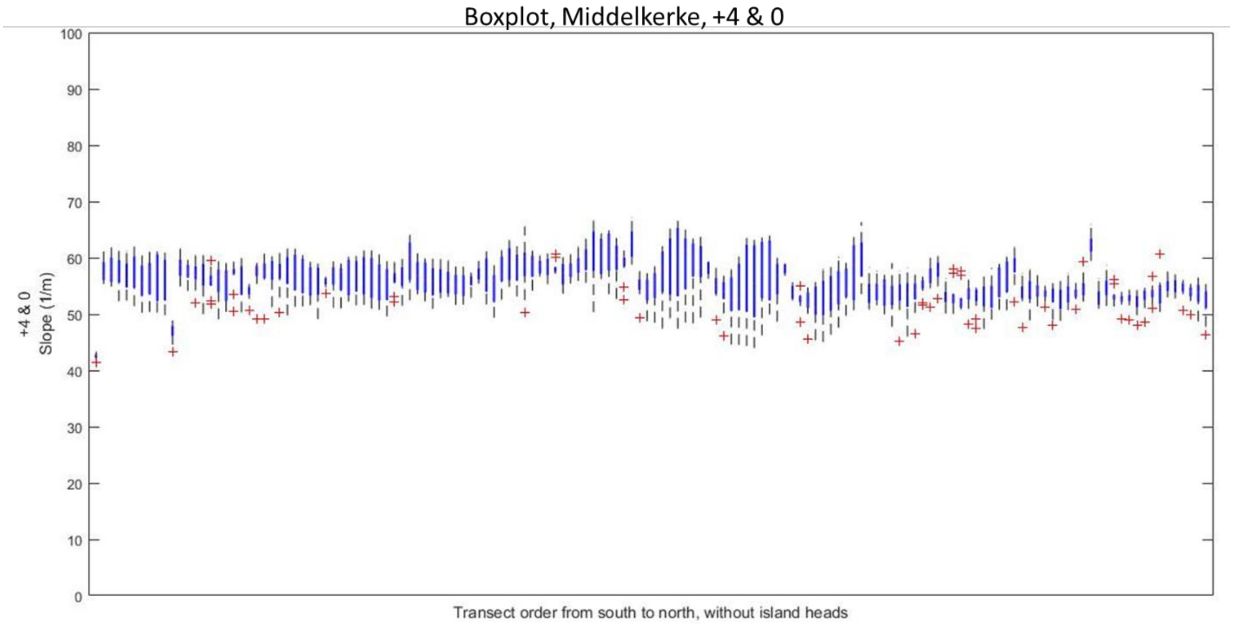


Figure 17: Boxplot of the slope between +4 and 0 meter, for the coastal area Middelkerke

Also, for Walcheren the boxplot for the +4 & 0 part is plotted (Fig. 18). This was mainly done for the interesting part in the far north of Walcheren where the slopes become gentler. Overall the amount of variation, indicated by the boxplots, is different along the coastal area. With rather stable slopes for the southern part of the area around 1/25 to 1/30, very steep slopes in the middle of the area which are extremely stable probably due to the hard structure near Westkapelle and in the northern part the slopes were more variable over the measurement history, indicated by the wider spread of the boxplots. This indicates a more dynamic beach in the north part of Walcheren.

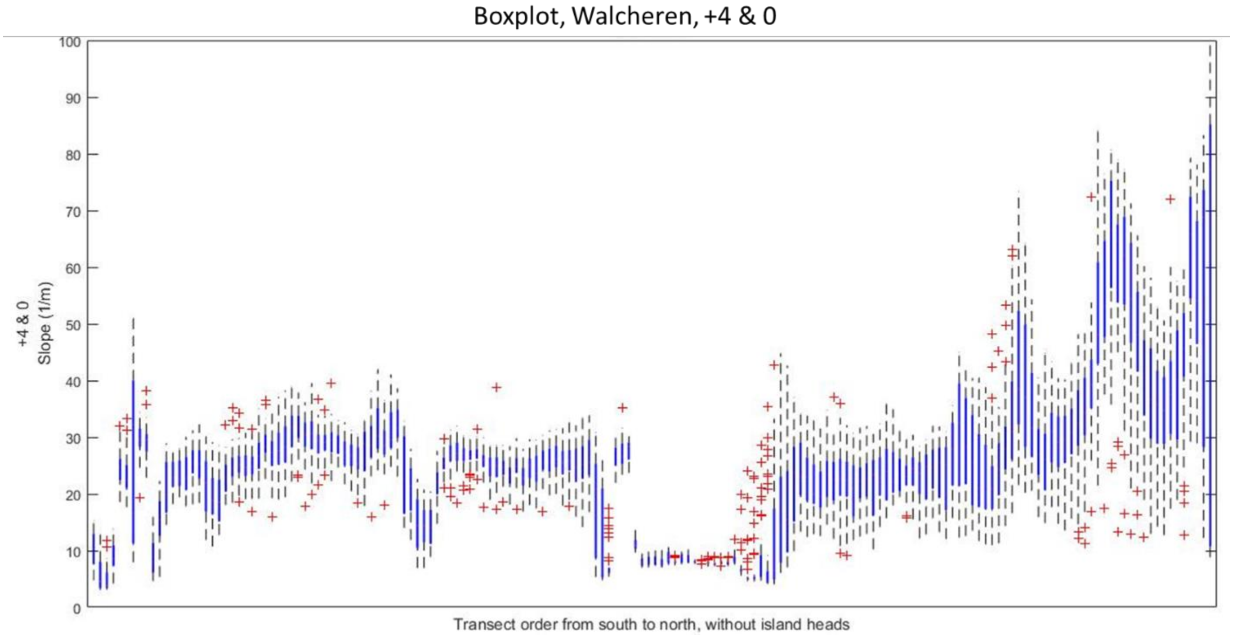


Figure 18: Boxplot of the slope between +4 and 0 meter, for the coastal area Walcheren.

For Schouwen slope between -2 & -4 is plotted as a boxplot (Fig. 19). The slope in the southern and northern part of this coastal area is rather stable with minor variation. Although the amount of variation in the middle part of the coastal area is very striking. This part is located on the west-north-western part of the area. This part is apparently very dynamic, which might be caused by occasional welding of sandspits to the beach.

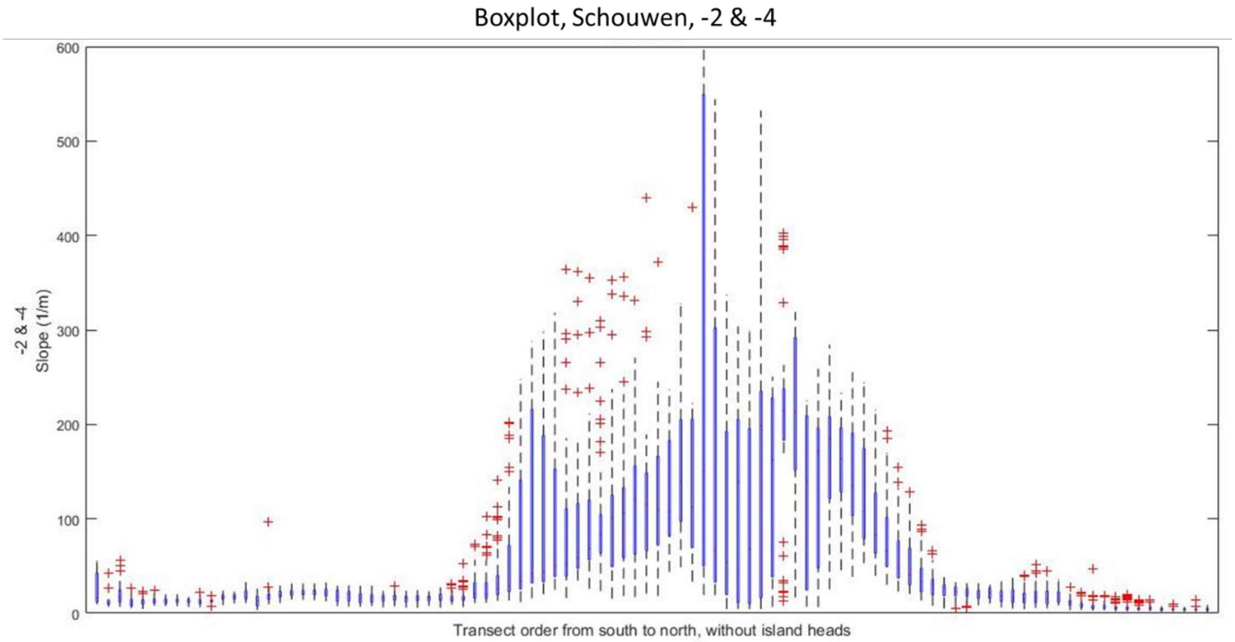


Figure 19: Boxplot of the slope between -2 and -4 meter, for the coastal area Schouwen.

The deeper part of the profile, -4 & -8, is plotted in a boxplot for Rijnland (Fig. 20). This part was interesting due to the gentle slope caused by the deposition fan of the old Rhine (see previous section). The bandwidth of the variation in the slope is very small for the southern part of the area, where some peculiar alongshore variation is indicated, which seems very stable according to the narrow bandwidth. Towards the north the slopes become somewhat gentler and the amount of variation also increases.

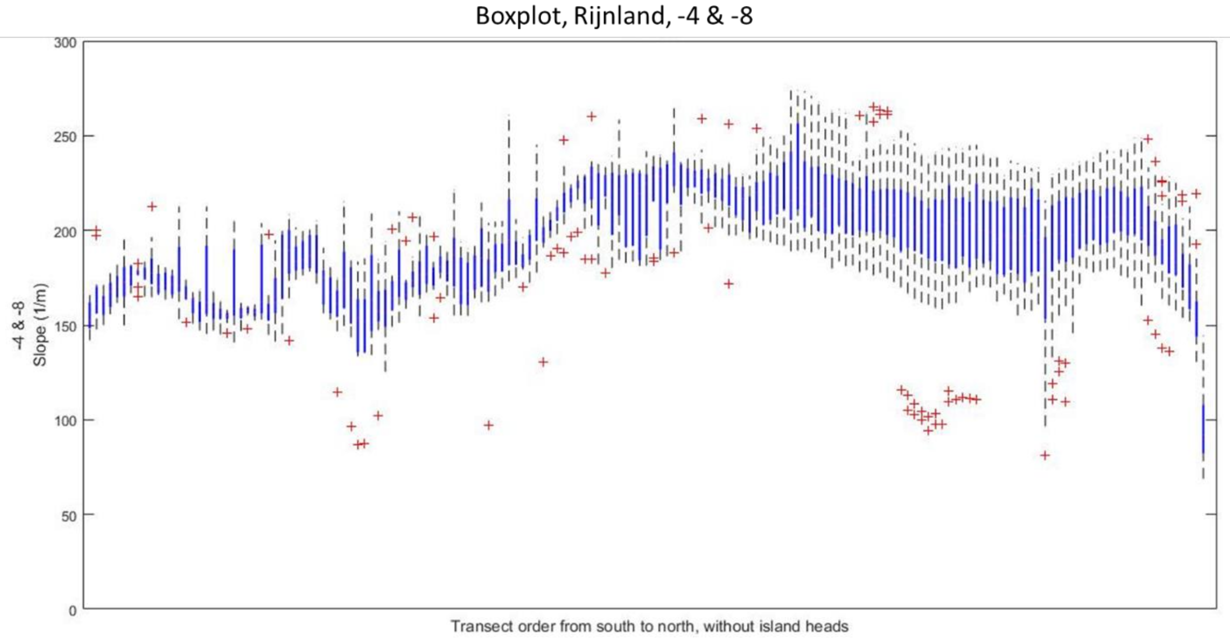


Figure 20: Boxplot of the slope between -4 and -8 meter, for the coastal area Rijnland.

For Noord-Holland the boxplot shows the variation in the slopes between the +2 and -2 height points (Fig. 21). Notable is the amount of variation, which is relatively large. However, the most remarkable area is the steep middle part of the coastal area. Which is probably caused by the “Hondsbossche Zeewering” which is located there. Furthermore, towards the north the profiles become steeper when approaching the tidal channel between Noord-Holland and Texel.

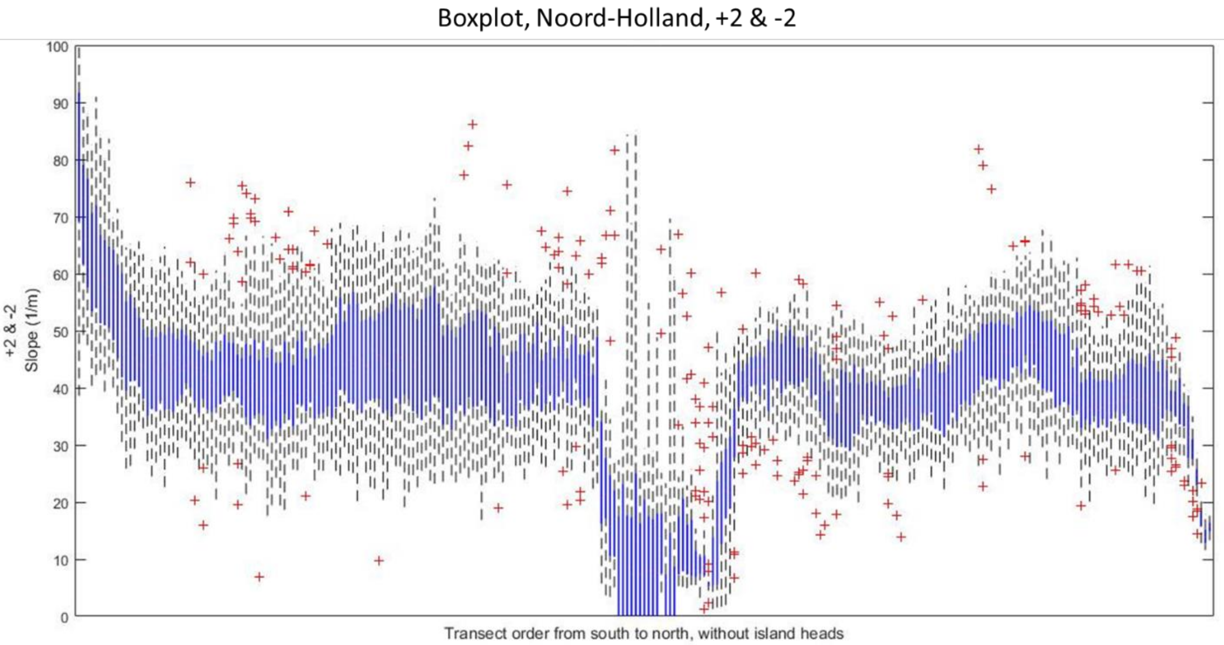


Figure 21: Boxplot of the slope between +2 and -2 meter, for the coastal area Noord-Holland.

The western part of Ameland was indicated to have a very wide beach (gentle beach slope) in the 'overview' figures (Fig. 14 and 15). Note: the island heads are not plotted. To get more information of the beach slope on the island the +4 & 0 slope at Ameland is plotted in a boxplot (Fig. 22). The slope is much gentler, but also much more variable, in the western part of the island compared to the rest. Where it is generally stable around a slope of 1/40. The variation in the slope in the western part of the island might be cause by occasionally huge sandspits/bars which attach to the beach at this point.

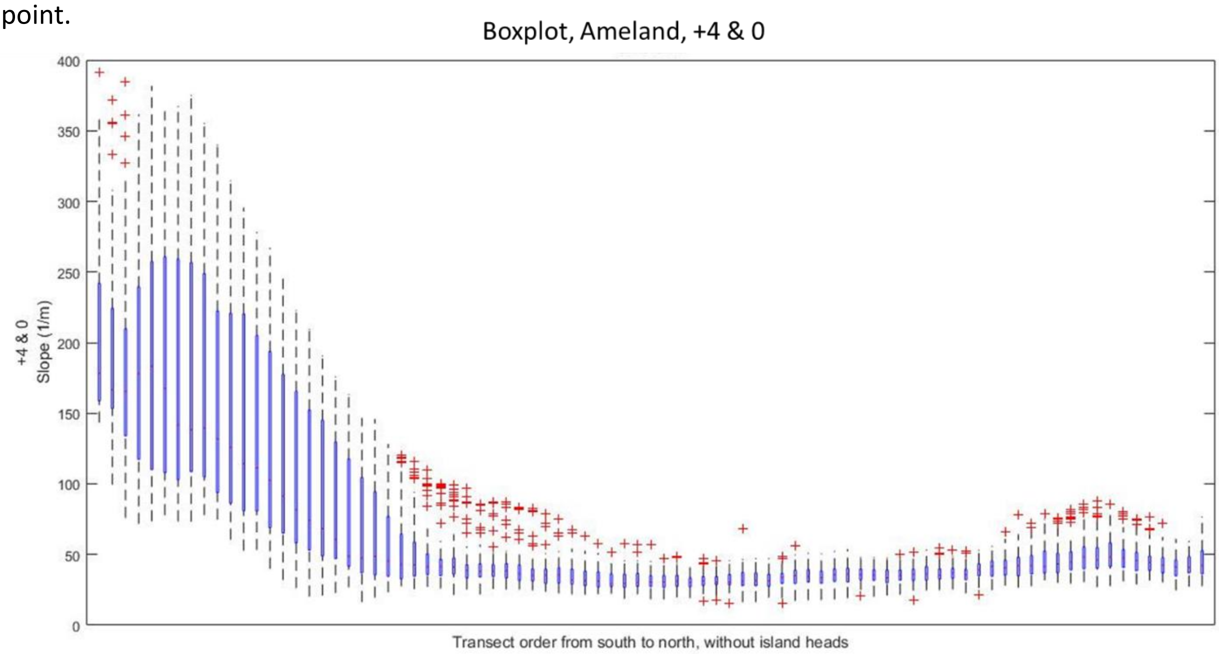


Figure 22: Boxplot of the slope between +4 and 0 meter, for the coastal area Ameland.

The slope of the deep parts of the profiles in Midtjylland describe the same alongshore trend as the beach slopes on Ameland. Therefore, Midtjylland slopes between the -4 and -8 height points are shown in a boxplot (Fig. 23). The same steepening, when going from south to north along the area, is visible in the boxplot. Furthermore, which is fairly remarkable, the amount of variation in the slope seems rather equal along the whole area. Indicating that the gentler slope in the south is most probably not caused by an occasionally sudden sediment input, but by a more permanent situation in the deeper shoreface, like the previously discussed Horns Rev (Section 4.1.1).

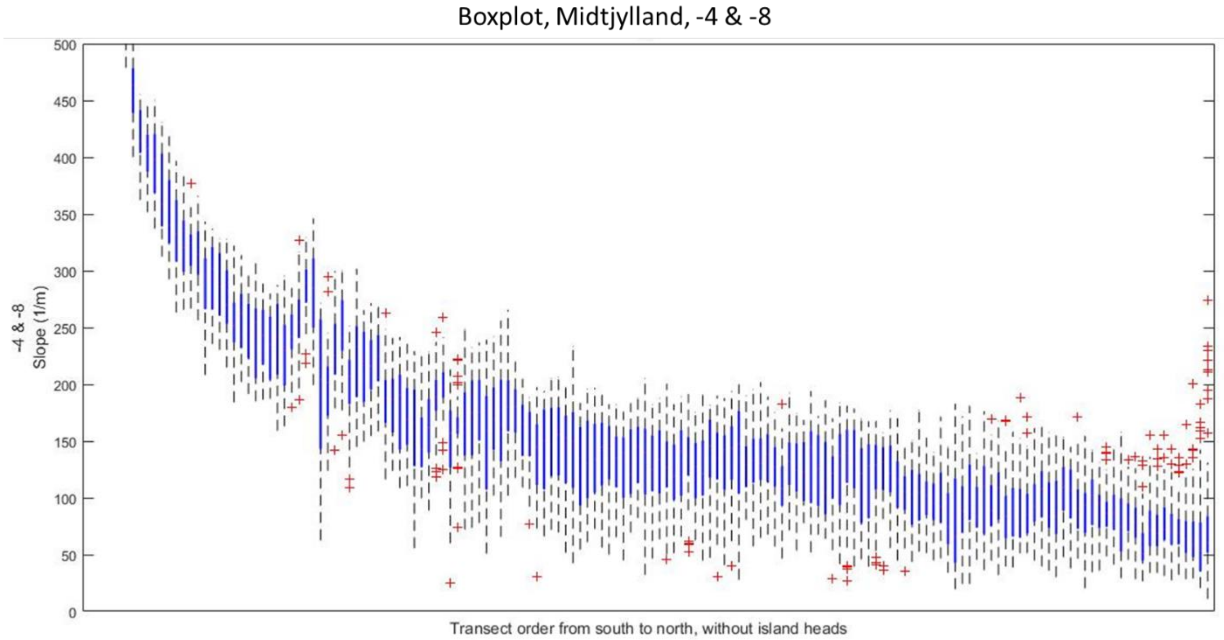


Figure 23: Boxplot of the slope between -4 and -8 meter, for the coastal area Midtjylland.

The VigsoJammerbugten slopes in the deeper part of the profiles have a sort of arch in the alongshore trend. A boxplot of the slopes between the -4 and -8 height points is shown (Fig. 24) to give more detailed information about this coastal area for this characteristic. When looking at the boxplot, which is also a zoom in into the coastal area, there are two (not one) arch shaped features in the alongshore trend, with even a small third one in the northern part (indicated in figure 24). This might be explained by the general shape of the coastline of this coastal area, as there are two arches and a weakly third one in the coastline (see figure 25). With the more southern one being smaller than the middle but greater than the northern one, which corresponds to the shapes indicated in the boxplot. Furthermore, the overall variation in the slope is nearly equal along this coastal area.

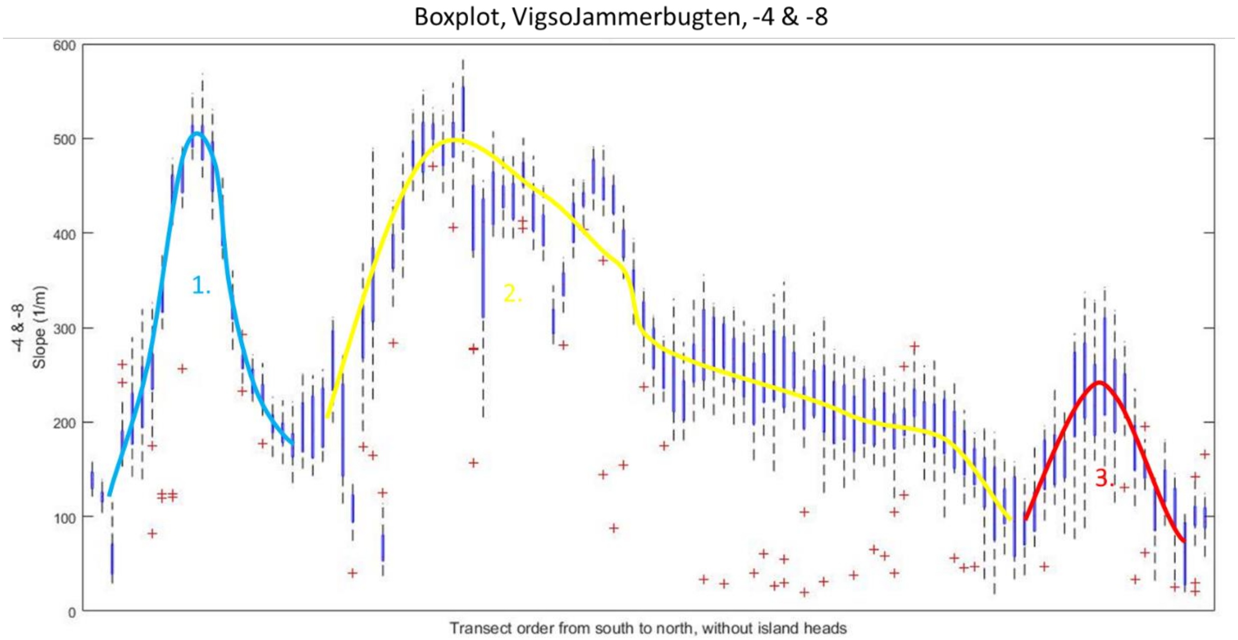


Figure 24: Boxplot of the slope between -4 and -8 meter, for the coastal area VigsoJammerbugten. Indicated are the three arch shaped features in the alongshore trend.

Figure 25: Map indicating the three arches in the coastline of the coastal area VigsoJammerbugten.

4.3 Analysis of the categories

Figure 26 displays the categories along the coastal areas. It is evident that category 1 occurs most often, which are the profiles with an increasingly gentler slope going seawards; a convex profile shape. Although, in the coastal areas of the province of Zeeland (zws, wal, nb, sch, goe and voo) category 1 is not represented as much (indicated with (a) in figure 26). Here the other three categories occur more often. Where categories 3 and 4 indicate a steeper sloping deep part compared to the rest of the profile, which occurs often when a (tidal) channel is located along the shore. Furthermore, the red colour of the plotted categories for these coastal areas indicate that the -4 & -8 slope is steeper than 1/50. The same is indicated at the northernmost point of Noord-Holland (indicated with (b)), where the transect profiles are located close to the tidal inlet between Texel and Noord-Holland. A part near the south of Sylt is dominated by category 2 (indicated with (c)), indicated that a bump is present in the general profile shape. This category also occurs on the southern part of Vlieland (also indicated with (c)).

Looking at the steepness of the deeper foreshore (slope -4 & -8), indicated by the colours, the Wadden islands have the gentlest sloping deeper foreshores (indicated with (d)). Whereas, the slopes in Zeeland are steepest, as mentioned before. In general, the slopes, and the general shape categories, are quite comparable between the coastal areas of the Dutch mainland (Delfland, Rijnland and Noord-Holland) and the coastal areas of the Danish mainland (indicated with (e)). As most of these transect profiles are within category 1, with only some alongshore different variations in -4 & -8 slopes.

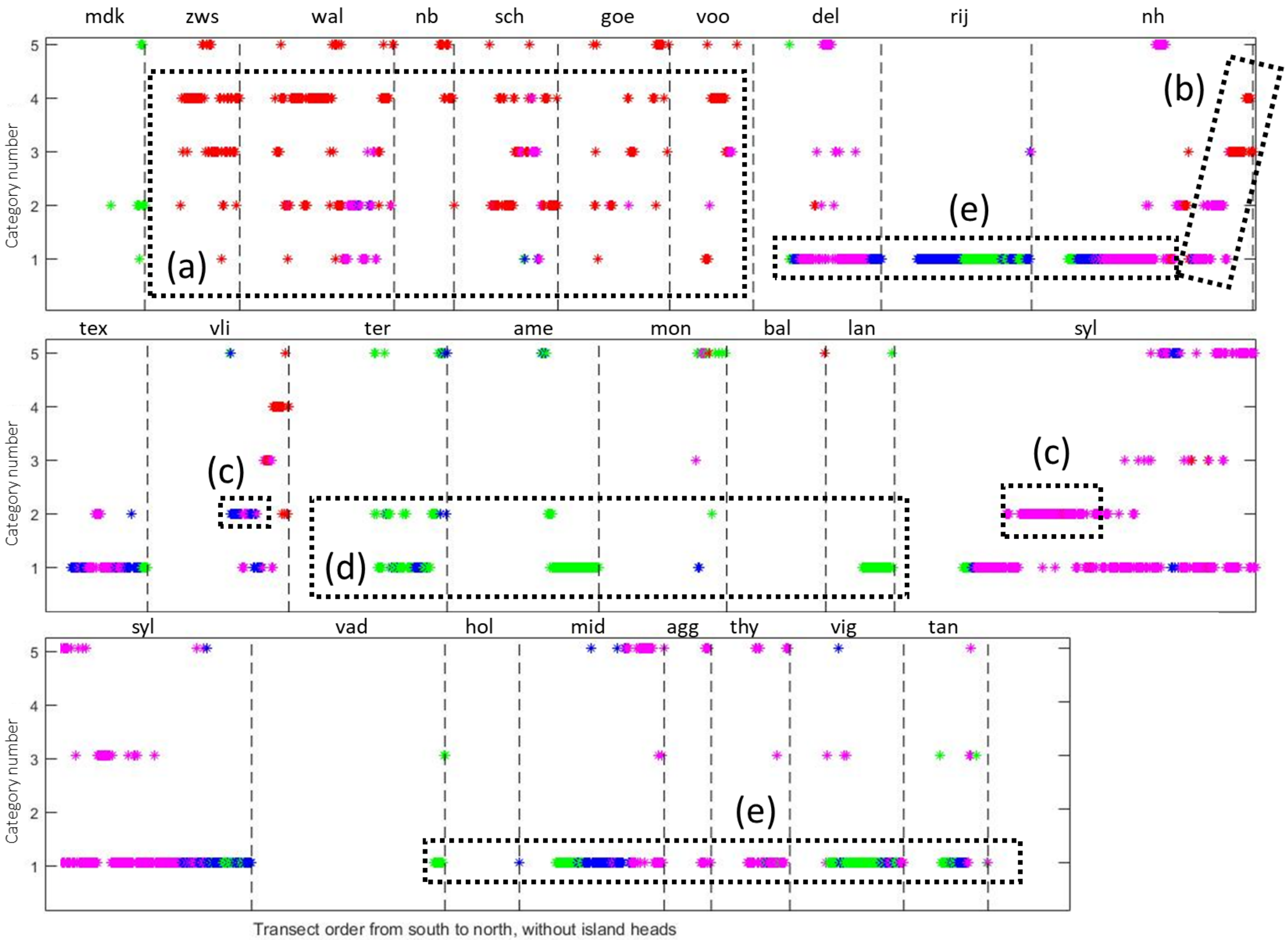


Figure 26: Figure indicating the distribution of the transect profiles into the categories. The first categorization is plotted on the y-axis. The second categorization is visualised by the colour of the points. Where: Green = category 1, Blue = category 2, Magenta = category 3 and Red = category 4. Indicated with letters are areas discussed in Section 4.3.

5. Potentials of this dataset

As mentioned before, the analysis of this dataset during this project is limited due to the available time. Therefore, some potential analyses which can be done in the future with this dataset are discussed here.

An averaged profile, per transect location, can be made, which indicates the general shape of the profile while eliminating some temporal features, like sandbars. This can be done by firstly interpolating the measurements done at the same transect profile to a defined transect grid, to make data points at the same positions. When the data points are available at the same position an average data point over time can be made for that position. Thereafter the average data points should be connected to make a new profile: the averaged profile.

The slope of the intertidal zone can also be determined per transect. The averaged slope of the intertidal zone (β_i), preferably coupled to the mean tidal range at each location, can be determined with the distance and height difference between the mean high and low waterline points. So, each calculation must be coupled to the averaged tidal range at the profile location. The β_i can be determined by finding the intersection points of the profile with the averaged low- and high-water height lines and calculating the distance and slope between those points (Fig. 27).

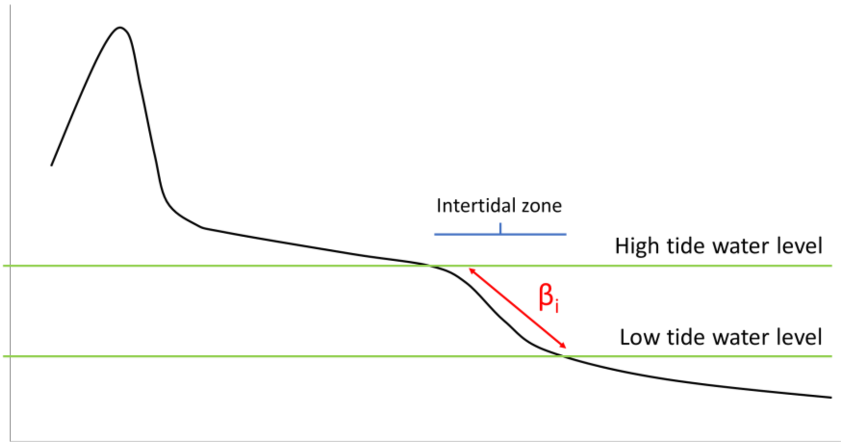


Figure 27: Schematic cross-shore profile indicating the intertidal zone and the averaged slope of the intertidal zone (β_i).

The volume of the dunes can be determined between, for example, height point z = +4 m and a further landward positioned point (Fig. 7, indicated by (V_d), Section 3.1.3). The landward point can be specified in multiple ways: another height point, an X distance landward of the seaward point or, like in figure 7, the peak of the first dune.

The number of bars present at each location can be determined between the position of the 0 meter elevation point and a more seaward located point, for example the -10 meter height point. This can be done by finding the peaks in the profile between these points and count them (Fig. 8, Section 3.1.3). The peaks must meet some conditions to filter some minor (measurement) fluctuations, to make sure that not every little positive height change in the profile data is considered a bar. For example, the peaks must be more than 0.5 meter high (can differ between locations) compared to its landward trough. Furthermore, the locations of the bars with respect to the 0-meter height point (mean-water line) can be determined (indicated by D_{bar}, in figure 8). This can be determined by finding the locations of the bars (like described above) and refer these to the 0 m height point location.

The distance between the chosen landward boundary of the beach (a height point) and the peak of the first dune indicates how steep the seaward slope of the dune is. This can be done by finding the location of the foredune peak and the intersection point of the profile with the chosen height line and determine the distance (indicated by D_{d-b} in figure 7, Section 3.1.3). Furthermore, the distance between the peak of the first dune and the mean sea level (0 m elevation) can be determined. To investigate differences in the distance from the coastline to the first dune.

The height of the foredune (H_d) can be determined by finding the vertical position of the foredune peak and calculating the vertical distance to the ordnance datum of the defined grid (Fig. 28).

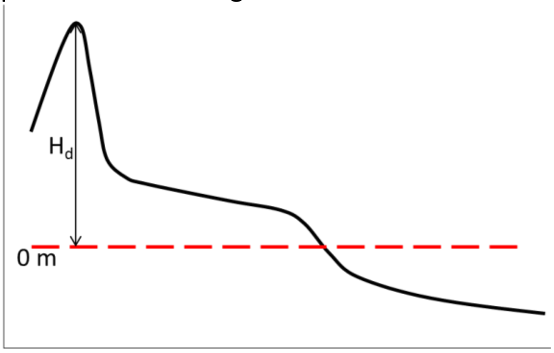


Figure 28: Schematic cross-shore profile indicating the height of the foredune (H_d).

An envelope study can also be done. This envelope describes the area of active morphological change within a profile over a certain period of time, the active bar zone. This can be determined by stacking up the yearly measurements. The thickness, length, position and volume of the morphologically active layer for each profile, taken over a fixed period of time, can indicate differences in the morphological activity (especially bars) between profiles. Figure 29 is an example of stacking yearly measurements, made in MorphAn. The grey area in the figure indicates the envelope.

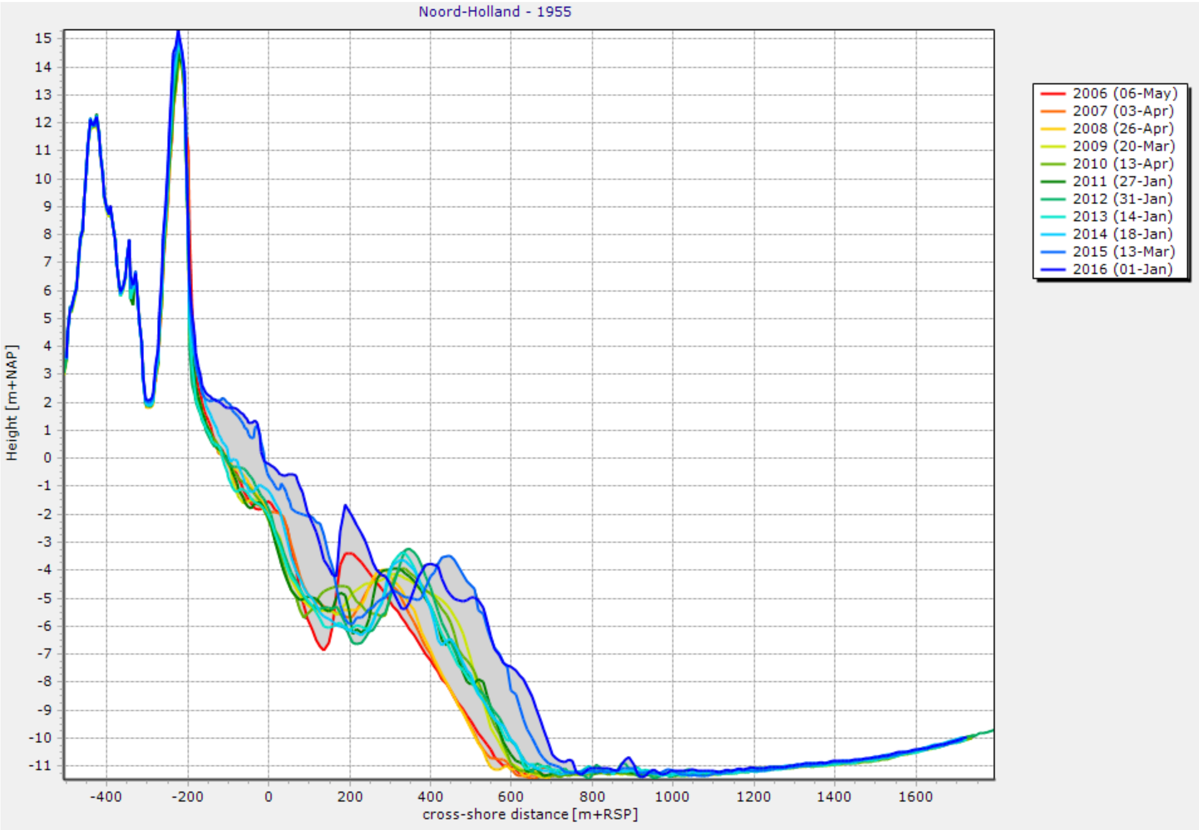


Figure 29: An example of a plot where different yearly measurements are plotted on top of each other, made with the use of MorphAn. The grey area indicates the maximum differences between the measurements. This example is from transect number 1955 in the Noord-Holland coastal area.

The effect of nourishments on the characteristics of the profiles can also be determined and compared to other profiles which are located near similar nourishments. This can be done by comparing the profile characteristics in the brief period before the nourishment with the characteristics over the couple of years after the nourishment and determine the difference (the effect of the nourishment). Thereafter the effect of the nourishments on the different transects can be compared.

The trends/development of each characteristic at each profile can be determined. This can be done similar to the volume trend. Furthermore, the trends might be incorporated in the categorization of the profiles. Where a distinction can be made between profiles with positive and negative trends, for example. Additionally, the effect of the nourishments on the trends can, and maybe should, be determined to make sure the 'natural' profile state can be compared. Moreover, to indicate differences in the profile responses at various locations to similar nourishments.

The amount of variation in the characteristics can also be compared. As this gives an indication of how dynamic a certain part of the coast is. This can be done by comparing the bandwidth around the mean values.

The categorization can be expanded or done in multiple ways. Firstly, a new categorization can be done based on other characteristics, like the average amount of bars in the profile or the trends in the development of each characteristic. But the categorization can also be expanded by adding more characteristics into the categories, making more detailed categories, although the number of categories might become too many. Multiple categorizations, with a small number of categories, can also be done so that each transect falls under a certain category per characteristic. Though, one should be cautious not to overdo the number of categories, resulting in noncomparable areas due to too much detail.

More detail can also be added to the existing categorization by including more of the actual mean slope values to each category. Or by splitting the profiles into smaller areas, which would result into more accurate schematic profiles linked to each category. Furthermore, the amount of variation in the slope per transect, visualised in the boxplots, can also be incorporated in the categories. Where a high variation indicates a highly dynamic profile and vice versa.

Finally, the positions of the isobaths in the profile, relative to the position of the dune peak or mean water line (z = 0 m) can also be determined. If these positions are known the positions relative to, distances between and the slope (β_{part}) between each isobath can be calculated (Fig. 30). From the β_{part} the maximum and minimum slope can be determined. Also, when combining the slopes of small parts of the profile, a schematic representation of the profile can be constructed. Furthermore, reverse slopes (β_{L_bar}), the landward side of a sandbar, might be calculated to indicate the shape of a sandbar. Note the potential problems with determining the positions of a depth point, as one depth can occur at multiple locations in one profile (see Chapter 6).

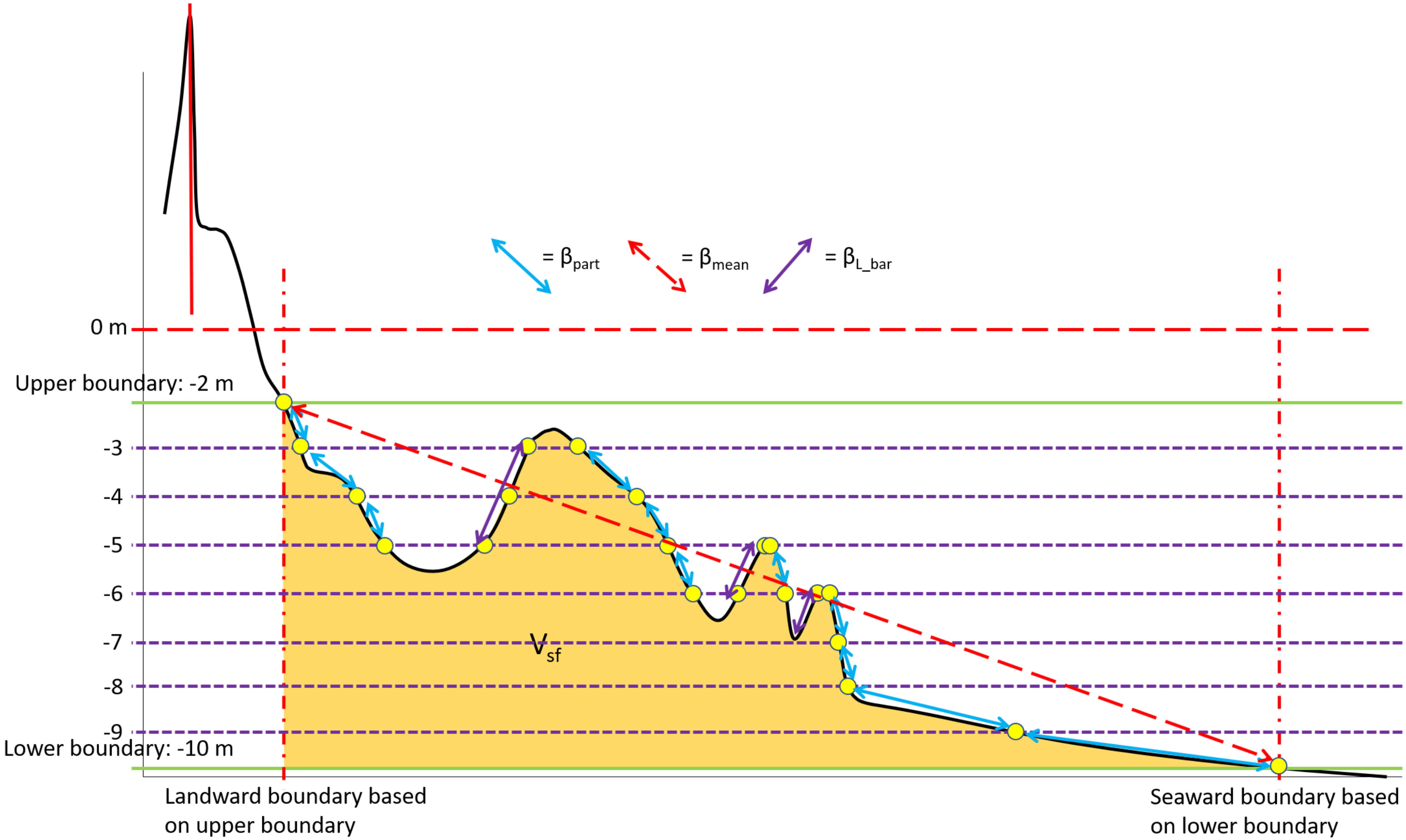


Figure 30: A schematic overview of the positions of isobaths and the different slopes present in a cross-section.

6. Uncertainties

In this chapter the possible errors and uncertainties during the calculation of the characteristics are discussed (Section 6.1). Followed by a list of some remarkable findings and errors in the database, of which some might lead to miscalculations (Section 6.2).

6.1 Uncertainties during the calculations

Firstly, if a height line intersects with the profile multiple times only one intersection point can be taken for the calculation of a certain characteristic. For the horizontal intersection lines lower than 0 meter the most landward intersection point is used in the calculation (Fig. 31, left window). While for the lines of 0 meter and higher the most seaward intersection point was used. This method assumes that these (most landward and most seaward) intersection points do not occur much more landward than the coastline, for the lower intersection points, or are positioned far seawards of the coastline, for the higher intersection points. If this does occur the characteristics are calculated over an incorrect area, which leads to a false result.

It might occur that the intersection point is "shifted" seaward by the presence of a sandbar (Fig. 31, right window). Due to this effect, the slope of a part of the profile might seem gentler/steeper according to the calculations than it was.

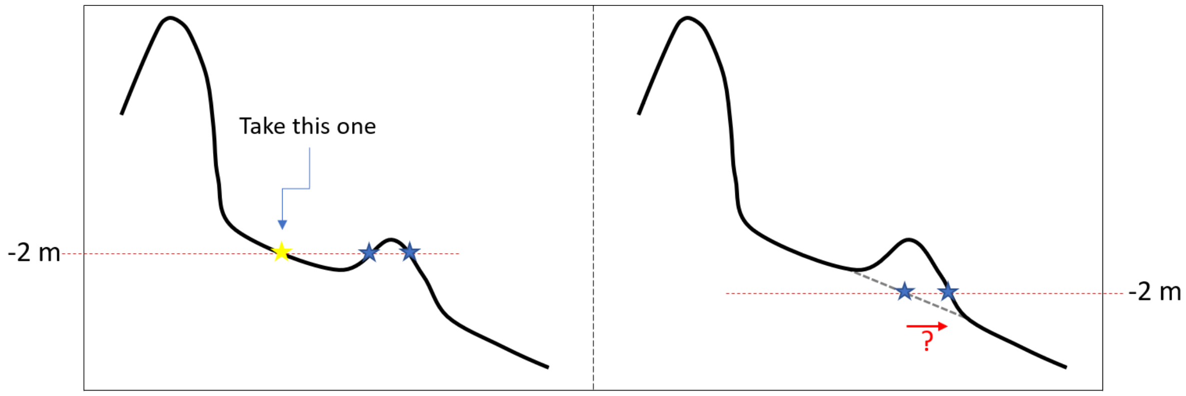


Figure 31: A schematic example of a height line which intersects with the profile at multiple positions, indicated is which intersection point should for a horizontal intersection line lower than 0 meter (left). An example of a seaward "shift" in cross-shore position of a height point due to the presence of a sandbar (right).

The date on which the measurements are done should be considered. For example: the beach width and volume are likely dependent on the timing of the measurements. Quartel et al. (2008) investigated seasonal variability in cross-shore positions of the high-, mean- and low- tide contours along with the beach width and volume. Furthermore, they studied the dependency on the offshore wave conditions. At the end of winter after more intense and frequent storm events, the beach was wide with a small volume, while during summer (with mild wave conditions) the beach became narrower with a large volume (Quartel et al., 2008). At the beginning of winter, the three contour lines straightened, simultaneously the dunefoot shifted landwards and the mean low water line shifted seawards. Consequently, the beach profile became flatter during winter. The gradual landward migration of the mean low water line, due to a lack of wave breaking, led to the steepening in summer. Therefore, the timing of the measurements might be crucial, as different measurement results can occur between measurements done during spring and during summer (generally there are no measurements done during autumn and winter).

During the calculation of the means and standard deviations of the characteristics the number of values used to determine the means (and standard deviations) was nonequal for all transects. This was due to unequal amounts of measurements done at each transect, where some were measured annually for more than 50 years, while others were measured less frequent and not always annually (see figure 9, Section 3.1.3). Even for the calculation of the means for the period of 2006-2016 the amount of measurements done is not equal. This might have caused some comparisons between dissimilar results. Because if there are less values to calculate the mean from, than certain measurement errors or unique situations have more influence on the result than if the mean is calculated over more values.

Due to the method of determining the volumes of the +4 & 0 part, a growing, seaward propagating, dune can cause a decline in the volume. Because, if the 0 meter height point remains at roughly the same position, while the +4 meter height point migrates seaward due to the accumulating dune. Then the width of the area over which the volume is calculated declines, resulting in a reduction of the calculated volume even though no erosion of sediment occurred. This is clarified in figure 32.

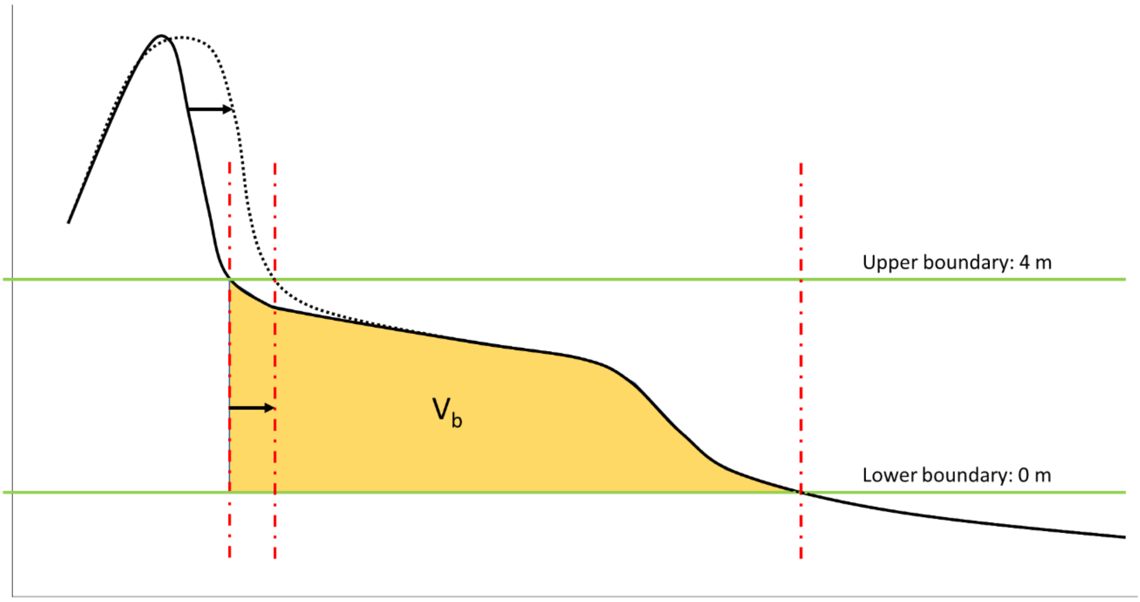


Figure 32: Figure indicating the effect of a seaward propagating dune on the calculated beach volume. The dotted black line indicates the profile after the dune propagation.

The transect profiles angles relative to the coastline were not considered during this project. While the angle of the transect profile is not always perpendicular to the coastline. This might lead to some small overestimates of the sediment volume or underestimates of the slope steepness.

Some measuring errors or data structuring errors which were unnoticed (and noticed, which are stated in the next Section, 6.2) might cause some incorrect results.

6.2 Remarkable findings and errors in the database

- ‘Gaps’ in the profile, when looking at a profile using the ‘side view’ window in MorphAn, occur when the horizontal distance between two measuring points is larger than 10 meters.
- For the quasi-synoptic data from Sylt the last measurements (one dry and one wet measurement) of the year were used to make one whole profile. If there were no new measurements available from that year than the measurements of the previous year were used. The dry and wet measurements were set together by a linear interpolation. If both measurements extended to the same position than the dry measurement was used over the wet measurement.

- During this project a few profiles were examined individually with the use of MorphAn, to get an insight in the data. A couple of these profiles show some errors in the measurement data, like in the following figure (Fig. 33): zones with a lot of 'spikes' (some of which were even more than 2 meters high) were found. This influences some results of the analyses.
    - Examples of profiles which show these profiles on Sylt: year 2000 & 1999. Transect number: 70154. Also, 2002: 21548 & 2002: 19726 for instance.

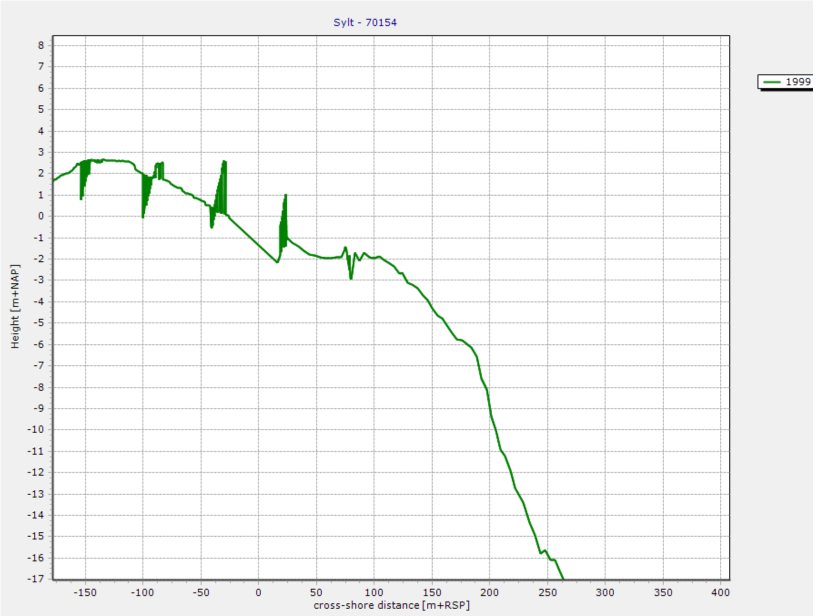


Figure 33: Example wrong measurement data.

- There is a backslash in the data from Vlieland (05_vlieland_65-16.jrk). Located in the following lines (with the use of "ctrl+f" the lines can be found):
  710 -275 720 -245 730 -235 999999999999 999999999999 \
  5 2001 3300 0 1506 2504 293
  This backslash should be removed as it gives errors when loading into MATLAB. This backslash has been removed by hand for the use of the data in this project.
- The data from coastal area Tannis-Bugt, measured during 2016, has the wrong sign: negative numbers are positive and vice versa. This must be fixed. This is fixed for the use of the data during this project and also fixed in the data folder of this project.

7. Concluding remarks

With the use of the database, containing data from the different partners within the EU Interreg project: Building with Nature, a tool to quantify and analyse certain coastal profile characteristics was made. This tool can be expanded by some additional analyses and calculations of characteristics, like the development of the characteristics over time or the response to certain nourishments.

Some main findings during the analysis of the results are the slope differences along the North Sea coast. Where the deeper parts of the shoreface are quite steep along Zeeland, due to the occurrence of tidal channels. While being very gently sloped for the Wadden islands and large parts along the Danish coast. Furthermore, the old Rhine deposits can be distinguished, as the alongshore deeper shoreface slope trend has an arch shape along the coast for coastal areas Delfland, Rijnland and Noord-Holland. The slopes at Middelkerke are generally gentler than at the other mainland coastal areas. Also, some other coastal features are reflected in the slope values, like the arches in the coastline at the VigsoJammerbugten coastal area, which are clearly reflected in the slope of the deeper shoreface.

Furthermore, when categorizing the transects, it is possible to distribute different areas of the coast into categories. This analysing technique can be enhanced during future work on this dataset by including several other characteristics, like the developments over time.

Finally, it should be possible to determine the effect of nourishments on the characteristics of the profiles, as the BwN project desires to identify differences in nourishment behaviour. This can be achieved by analysing responses to similar nourishments between different areas. Comparing the profile characteristics before and after the nourishment will determine the impact of the nourishment. Subsequently, the effect of the nourishments on the different transects can be compared.

References

Aagaard, T. (2011). Sediment transfer from beach to shoreface: The sediment budget of an accreting beach on the Danish North Sea Coast. Geomorphology, 135(1-2), 143-157.

Bregman, M. (2017). Transformation of coastal morphological data of Interreg partners to a format suitable for MorphAn.

Deltares. (2016). MorphAn 1.5.0, Analytical tool for sandy coasts, User manual. The translation of this manual from Dutch to English was commissioned by Rijkswaterstaat Water, Traffic and the Environment (WVL), as part of the Interreg VB NSR project "Building with Nature".

Elias, E. P. L., Van der Spek, A. J. F., Wang, Z. B., & De Ronde, J. (2012). Morphodynamic development and sediment budget of the Dutch Wadden Sea over the last century. Netherlands Journal of Geosciences, 91(3), 293-310.

The MathWorks, Inc., Natick, Massachusetts, United States. Statistics and Machine Learning Toolbox, Release R2017b. Retrieved from https://nl.mathworks.com/help/stats/.

Quartel, S., Kroon, A., & Ruessink, B. G. (2008). Seasonal accretion and erosion patterns of a microtidal sandy beach. Marine Geology, 250(1), 19–33.

Stouthamer, E., Cohen, K. & Gouw, M. (2010). Avulsion and its Implications for Fluvial-Deltaic Architecture: Insights from the Holocene Rhine–Meuse Delta. SEPM Special Publication.

Wilmink R.J.A., Lodder Q.J., Sørensen P. (2017). Assessment of the design and behaviour of nourishments in the North Sea Region. Towards an NSR guideline for nourishments. Coastal Dynamics 2017, paper No. 043.

Appendix

1. Tables

1.1 Width and slope

Width and slope variable names used in the structures within the main structure.

Table 11: Width and slope variable names used in the structures within the main structure.

1st height point (m)	2nd height point (m)	Width variable name	Slope variable name
+4 most seaward	0 most seaward	Width_0_plus_4	Slope_0_plus_4
+2 most seaward	0 most seaward	Width_0_plus_2	Slope_0_plus_2
+4 most seaward	-2 most landward	Width_min_2_plus_4	Slope_min_2_plus_4
+2 most seaward	-2 most landward	Width_plus_2_min_2	Slope_plus_2_min_2
0 most seaward	-2 most landward	Width_0_min_2	Slope_0_min_2
-2 most landward	-4 most landward	Width_min_2_min_4	Slope_min_2_min_4
-2 most landward	-6 most landward	Width_min_2_min_6	Slope_min_2_min_6
-2 most landward	-8 most landward	Width_min_2_min_8	Slope_min_2_min_8
-4 most seaward	-8 most landward	Width_min_4_min_8	Slope_min_4_min_8

1.2 Volume

Table 12: Volume variable names used in the structures within the main structure.

1st height point (m)	2nd height point (m)	Volume variable name	Trend volume variable name
+4 most seaward	0 most seaward	Vol_0_plus_4	trend_vol_0plus4_2006_2016
+2 most seaward	0 most seaward	Vol_0_plus_2	trend_vol_0plus2_2006_2016
+4 most seaward	-2 most landward	Vol_min_2_plus_4	trend_vol_min2plus4_2006_2016
+2 most seaward	-2 most landward	Vol_plus_2_min_2	trend_vol_plus2min2_2006_2016
-2 most landward	-4 most landward	Vol_min_2_min_4	trend_vol_min2min4_2006_2016
-2 most landward	-6 most landward	Vol_min_2_min_6	trend_vol_min2min6_2006_2016
-2 most landward	-8 most landward	Vol_min_2_min_8	trend_vol_min2min8_2006_2016

1.3 Transects

An overview of the correct transect number order from south to north per coastal area. The transects which were removed as they were located at the island heads, and some other special cases, are also indicated. The current order column indicates the transect order as how it is saved in the matlab structures, this column is only filled in if it differs from the south-north order. So, if it is empty than the order was already from south to north.

Table 13: Table which states the south-north transect order per coastal area. Furthermore, the transects which were removed from the figures and the reason to remove them are specified.

Coastal area order number from south to north	Coastal area name	Current order	South-North (and W-E) order	Removed transects: Island heads (transect numbers), and other special cases	Special notes
1	Middelkerke (actually four coastal areas but treated as one)	-	74 – 78.5 79 – 82.5 83 – 87.5 88 – 92.5		Coastal areas 12-15: Westende-Bad De krokodille Middelkerke-Bad Middelkerke-Oost
2	Zeeuws-Vlaanderen	11 – 1487	1487 – 11		
3	Walcheren	540 – 3750	3750 – 540	3750-3526 is the southern end of Walcheren along the hard structure at the harbour of Vlissingen	
4	Noord-Beveland	0 – 520	520 – 0	100 – 0 lie behind the “Oosterscheldekering”	
5	Schouwen	68 – 1800	1800 – 68		
6	Goeree	280 – 2525	2525 – 280		
7	Voorne	400 – 1830	1830 – 400		
8	Delfland	9740 – 11850	11850 – 9740		
9	Rijnland	5625 – 9725	9725 – 5625		
10	Noord-Holland	0 – 5500	5500 – 0		
11	Texel	-	416 – 3452	416 – 860 southern island head. 2937 – 3211 northern island head. 3212 – 3452 behind the island (Wadden Sea side).	
12	Vlieland	-	3300 – 5460	3300 – 4060 southern (western) island head. 5367 – 5460 northern (eastern) island head.	
13	Terschelling	-	0 - 3004	5916 – 5902 single point with multiple transects.	5902 – 5916 is one extra transect on the

				0 – 540 southern (western) island head. 2660 – 3004 northern (eastern) island head.	south of the island: not analysed
14	Ameland	-	100 - 2516	4600 – 4966 behind the island (south-west side). 100 – 440 western island head. 2160 – 2516 eastern island head.	4600 – 4966 are transects on the Wadden Sea side of the island
15	Schiermonnikoo g	-	100 – 1618	100 – 520 western island head. 1440 – 1618 eastern island head.	
16	Baltrum	-	73-75-77		Only 3 transects and all are on the eastern island head.
17	Langeoog	-	1 – 87	1 – 35 western island head. 80 – 87 eastern island head.	
18	Sylt	5 – 22722 50045 – 71872 Not analysed: 1006233 – 1006333 1016124 – 1022848 1049855 - 1070404	71872 – 50045 5 – 22722 1020255 – 1022848 (not analysed)	71872 – 69789 behind the island (Wadden Sea side). 69739 – 67387 southern island head. 16462 – 22722 northern island head. 1020255 - 1022848 are located on the north-east point of the island. 1049855 - 1070404 Some randomly located transects and others from the south point of the island.	
19	Vadehavsoer	-	6970 – 6280	6970 – 6950 southern island head of Rømø. 6850 – 6820 northern island head of Rømø. 6810 – 6680 in between islands.	

				6680 – 6510 island of Fanø. 6500 – 6460 in between islands. 6440 – 6450 south point of Skallingen	
20	Holmsland	4010000 - 4021000	4010000 – 4021000		Holmsland is a ‘close-up’ to a part within Midtjylland, where the transects are closer spaced
21	Midtjylland	-	6270 – 4210		
22	Agger	-	4170 – 4010		
23	Nationalpark - Thy	-	3670 – 3060		
24	VigsoJammerbu gten	-	3050 – 1510		
25	Tannis-Bugt	-	1500 – 1060	At the far north point of Denmark, near Skagen, there is an arc of transects which fall out of order, transects: 1010 - 1050	

2. Figures

2.1 Overview figures

Some extra figures which were made (Appendix figures 1-4). Along with a table which indicate which value ranges were plotted in what colour in the standard deviation figures.

Table 14: Table indicating which value ranges are plotted in which colour per characteristic, for the Appendix figures 4 and 5.

+4 & 0	**Stdv slope range (1/xx)**	**Colour**		**+2 & -2**	**Stdv slope range (1/xx)**	**Colour**
	0 – 6	Red			0 – 6	Red
	6 – 12	Magenta			6 – 12	Magenta
	12 – 18	Blue			12 – 18	Blue
	18 – 24	Cyan			18 – 24	Cyan
	24 – 30	Green			24 – 30	Green
	30+	Yellow			30+	Yellow
-2 & -4	**Stdv slope range (1/xx)**	**Colour**		**-4 & -8**	**Stdv slope range (1/xx)**	**Colour**
	0 – 10	Red			0 – 10	Red
	10 – 25	Magenta			10 – 20	Magenta
	25 – 40	Blue			20 – 30	Blue
	40 – 55	Cyan			30 – 40	Cyan
	55+	Green			40+	Green
	-	Yellow			-	Yellow

2.2 Boxplots

The boxplots were made with the use of the MATLAB *boxplot* fuction within the *Plotting_Slope2Mean_noheads_boxplot.m* script (not shown in the Appendix but it is located within the project folder). In the beginning of the script the coastal area and slope variable which will be plotted can be selected basen on the given numbers to the following variables: *Areanumber* and *plotted_characteristic*. The boxplots were made for the mean slope values for the small parts of the profiles.

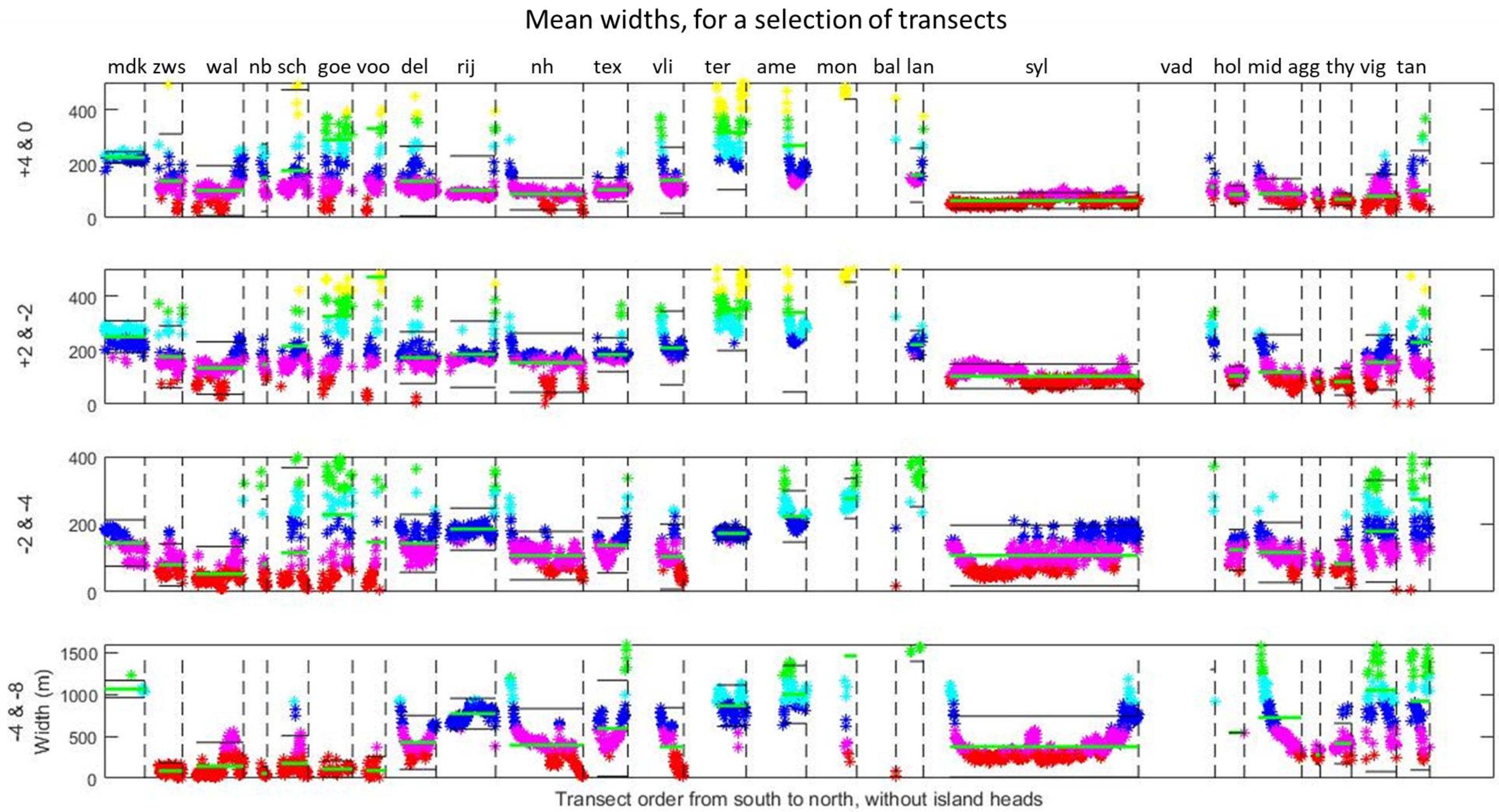


Figure 34: Figure indicating the mean widths over the whole measurement history along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Table 7. The horizontal green line, per coastal area indicates the mean of the mean widths of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean.

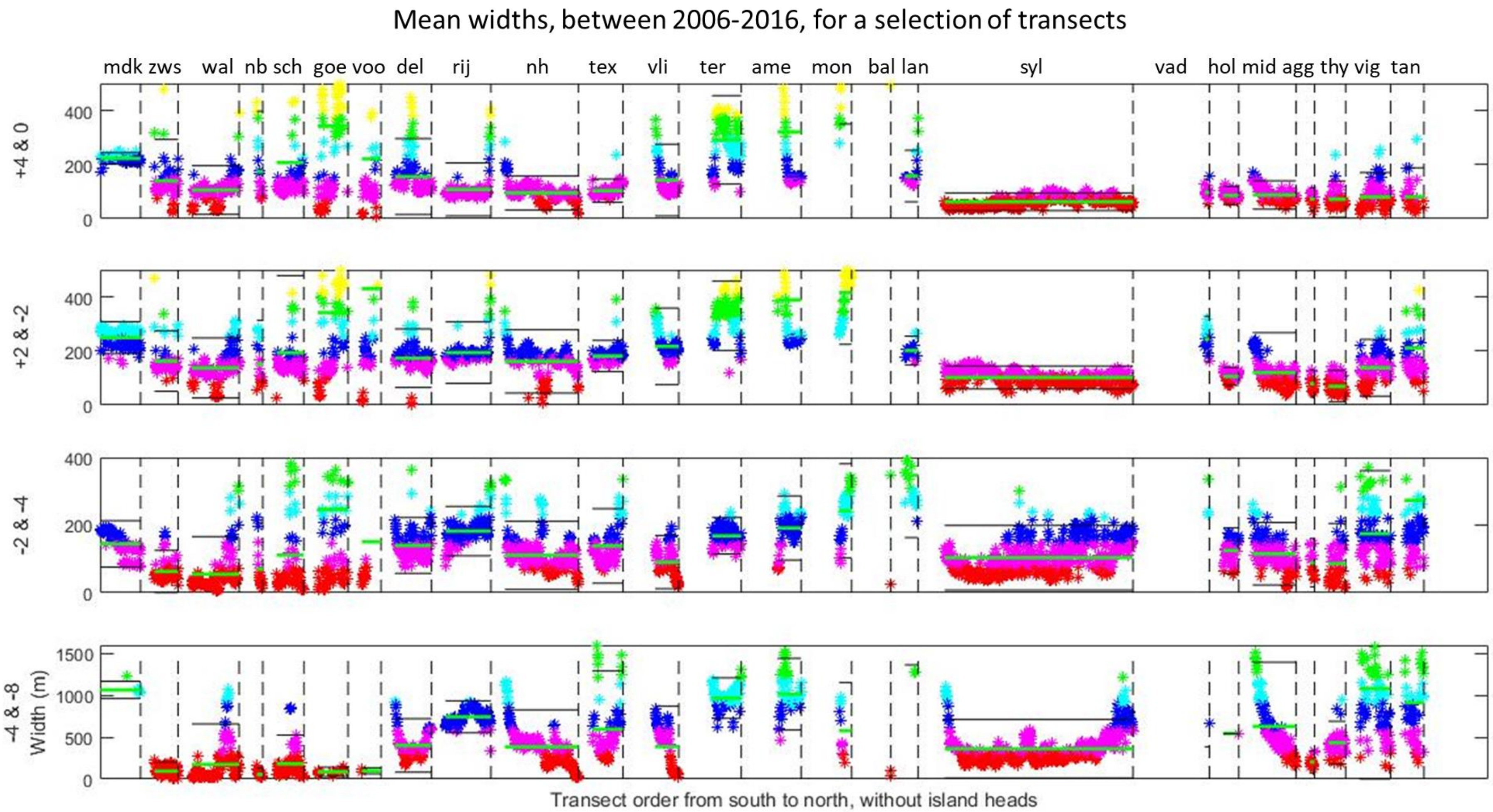


Figure 35: Figure indicating the mean widths over the period 2006-2016 along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Table 7. The horizontal green line, per coastal area indicates the mean of the mean widths of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean.

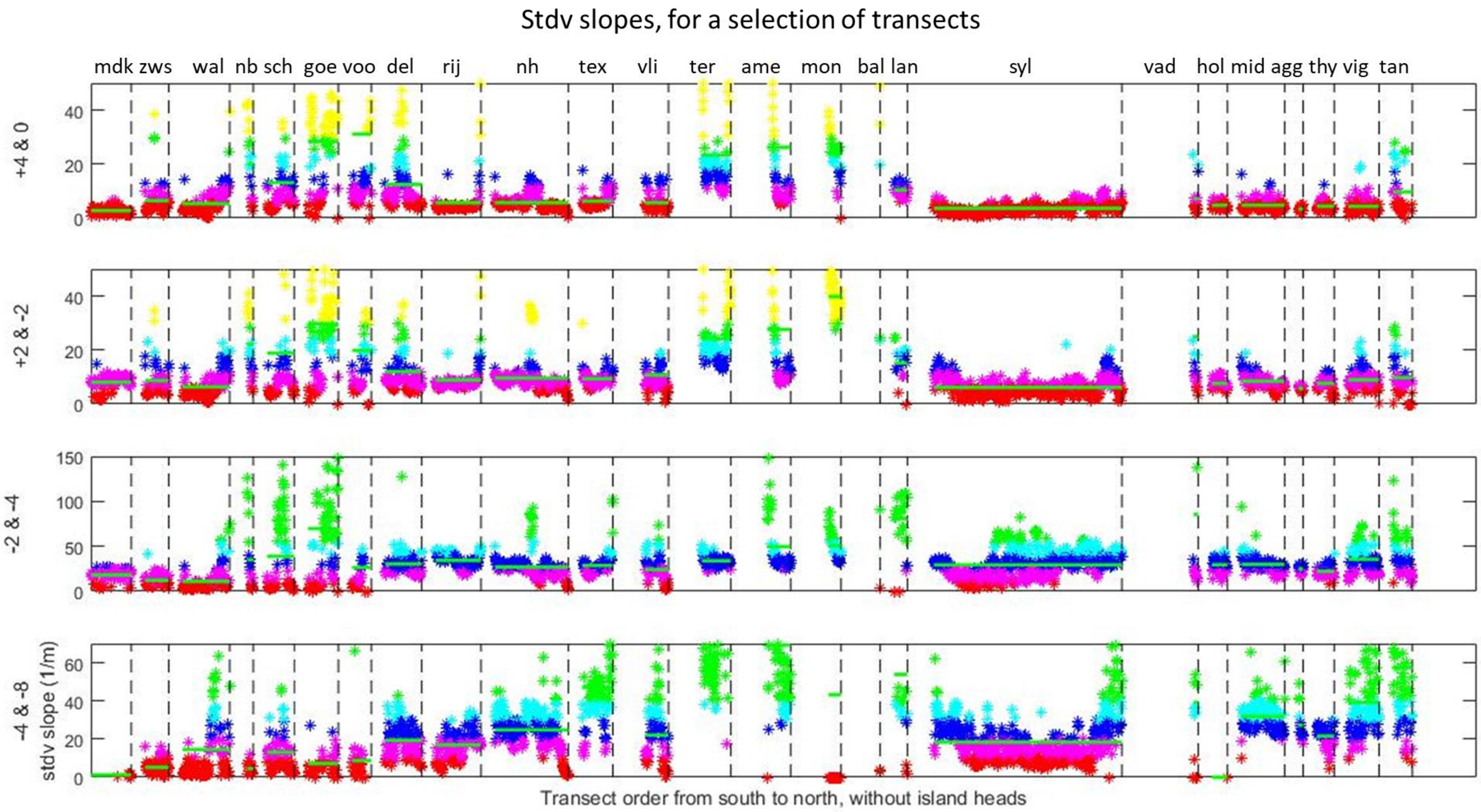


Figure 3: Figure indicating the standard deviation of the slopes over the whole measurement history along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Appendix Table 4. The horizontal green line, per coastal area indicates the mean of the standard deviation of the slopes of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean.

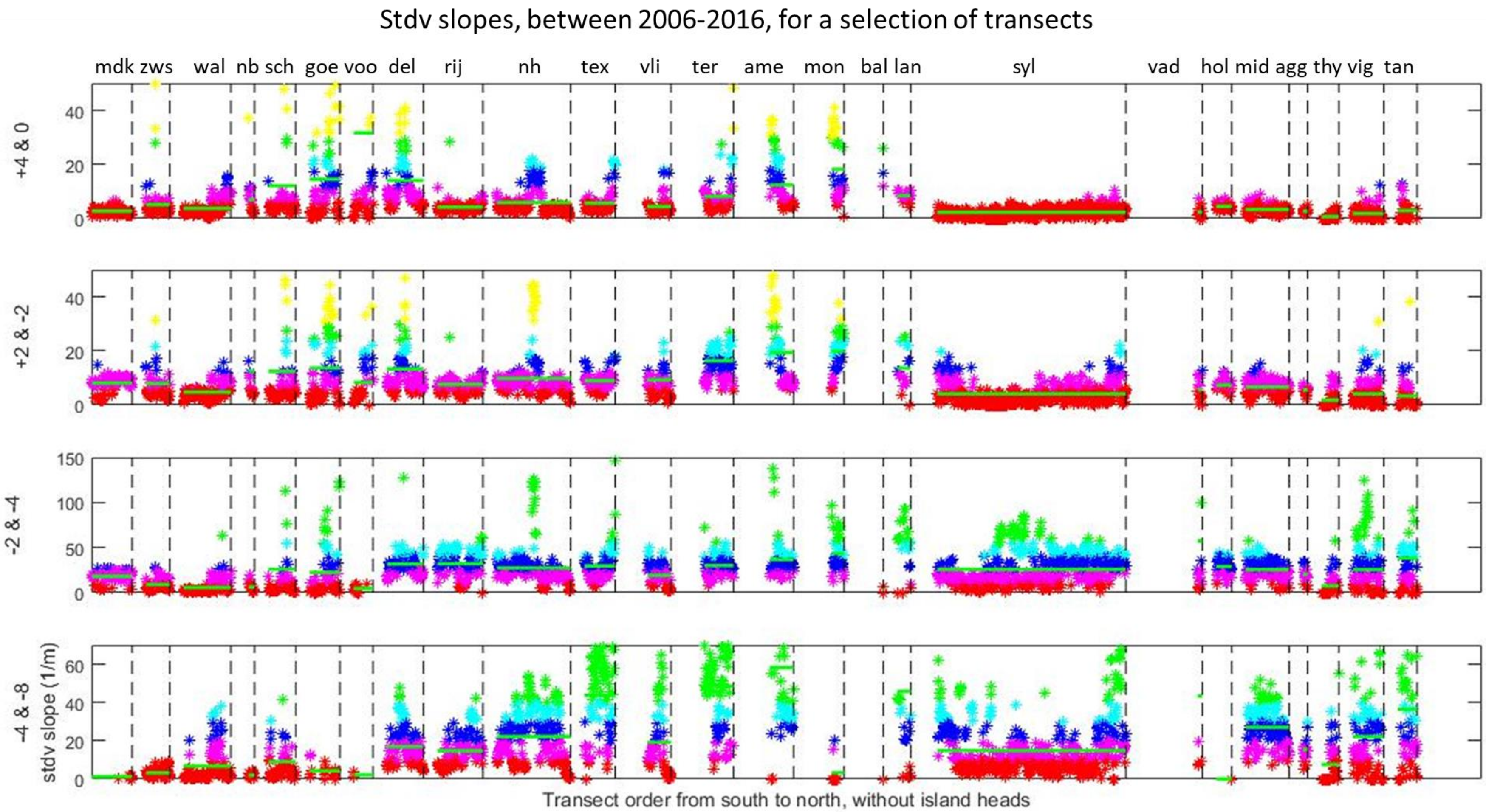


Figure 4: Figure indicating the standard deviation of the slopes over the period 2006-20016 along the North Sea coast, from south to north. The coastal areas are divided by the vertical dashed lines and the names are shown in the top of the figure. The colours are based on the values stated in Appendix Table 4. The horizontal green line, per coastal area indicates the mean of the standard deviation of the slopes of that coastal area. And the two horizontal black lines indicate the distance of 2 times standard deviation from the mean.

3. Scripts

3.1 Changing coastal area numbers

For some coastal areas a separate script has been made due to some differences in the data. The Danish measurements are also split-up into 6 coastal areas in the *Omzetten_kv_num_jrk_files_Vestkyst.m* script based on the chosen transect numbers (should be changed in the beginning of the script). Note: also for both the data from Middelkerke and Baltrum/Langeoog slightly different scripts are made, to handle the little variations in these datasets compared to the other, but those are not presented here.

Script 1: *Omzetten_kv_num_jrk_files.m*

```
%% Changing the Coastal area numbers in de jarkus files to the newly used numbers
% Ivo Naus

%% initialization
close all
clear all

tic
%% *************** Setting new number ************************************

% Important to set the new coastal area number value, the correct input
% file and the correct new file name.

new_kv_num = 49260040; % change this number

% Change the input file
input_file = 'C:\Users\ivo\Documents\1 . Rijkswaterstaat stage\Werkmap\Matlab\Ivo
scripts\Omzetten kustvaknummers\Oude nummers\test.jrk';

newfile_name = 'test.txt'; % change this name


%% load jrk file % Set new file name

open_file = fopen(input_file);

% new file
newfile = fopen(newfile_name,'wt');

%% read the jrk file line by line and change the coastal area number

oneline = fgetl(open_file); % read first line
while ischar(oneline)
    x = str2num(oneline);
    x2 = num2cell(x);
    if length(x) == 7 & x(7) ~= 9999 & x(7) ~= 99999 & x(7) ~= 999999 & x(7) ~= 9999999 & x(7)
~= 99999999 & x(7) ~= 999999999 & x(7) ~= 9999999999 & x(7) ~= 99999999999 & x(7) ~=
999999999999 & x(7) ~= 9999999999999 & x(7) ~= 99999999999999 & x(7) ~= 999999999999999 & x(7)
~= 9999999999999999 & x(7) ~= 99999999999999999 & x(7) ~= 999999999999999999 & x(7) ~=
9999999999999999999 & x(7) ~= 99999999999999999999
        % the length of the headline of each measurement
        x(1) = new_kv_num;  % change the coastal area number, which is in the first column of
a headline
        fprintf(newfile,'%s\n', num2str(x)); % print a new line in the new file with the
updated number
    elseif length(x) == 10 || length(x) == 8 || length(x) == 6 || length(x) == 4 || length(x)
== 2
        if length(x) == 10      % Check the length of a data line (lengths can be 2, 4, 6, 8
and 10)
            for i = 1:(length(x)/2)      % This only happens for the Y values
                if length(num2str(x(i*2))) == 1 % as the last number of the y values indicate
the type of measurement
                    x2{i*2} = ['0', num2str(x(i*2))];    % If there is only one number than the
actual y value was '0'
                end
```

```
            end
            x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ', ...
                num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6}),...
                '    ', num2str(x2{7}),'    ', num2str(x2{8}),'    ', num2str(x2{9}),...
                '    ', num2str(x2{10})];     % Make the new line correct
        elseif length(x) == 8
            for i = 1:(length(x)/2)
                if length(num2str(x(i*2))) == 1
                    x2{i*2} = ['0', num2str(x(i*2))];
                end
            end
            x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ', ...
                num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6}),...
                '    ', num2str(x2{7}),'    ', num2str(x2{8})];
        elseif length(x) == 6
            for i = 1:(length(x)/2)
                if length(num2str(x(i*2))) == 1
                    x2{i*2} = ['0', num2str(x(i*2))];
                end
            end
            x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ', ...
                num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6})];
        elseif length(x) == 4
            for i = 1:(length(x)/2)
                if length(num2str(x(i*2))) == 1
                    x2{i*2} = ['0', num2str(x(i*2))];
                end
            end
            x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ', ...
                num2str(x2{4})];
        elseif length(x) == 2
            if length(num2str(x(2))) == 1
                x2{2} = ['0', num2str(x(2))];
            end
            x = [num2str(x2{1}),'    ', num2str(x2{2})];
        end
        fprintf(newfile,'%s\n', x); % print a new line as the old line. NOTE: if a y value
        % was 0#, it is converted to # as the str2num makes it num (eating
        % the leading zero's). This part reflects if this happened and
        % returns '0#' to the correct locations if it happened
    else
        fprintf(newfile,'%s\n', num2str(x(1:(end-1))));
    end
    %disp(oneline)
    oneline = fgetl(open_file); % Set the next line in the original file to be read next
end
fclose(open_file);
fclose(newfile);

%% elapse time
toc
```

Script 2: *Omzetten_kv_num_jrk_files_Vestkyst.m*

```
%% Changing the Coastal area numbers in de jarkus files to the newly used numbers
% Ivo Naus

%% initialization
close all
clear all

tic
%% *************** Setting new number ***********************************

% Important to set the new coastal area number value, the correct input
% file and the correct new file name.

new_kv_num = 45000006; % change this number
```

```
% change the input file
input_file = 'C:\Users\ivo\Documents\1 . Rijkswaterstaat stage\Werkmap\Matlab\Ivo
scripts\Omzetten kustvaknummers\Oude nummers\Vestkyst_complete2.jrk';

% change this name
newfile_name = '45000006_Vestkyst_TannisBugt.txt';

% chose the transect range which should be taken
lower_transect_ID_number = 1010;

upper_transect_ID_number = 1500;

%% load jrk file % Set new file name

open_file = fopen(input_file);

% new file
newfile = fopen(newfile_name,'wt');

%% read the jrk file line by line and change the coastal area number

oneline = fgetl(open_file); % read first line
while ischar(oneline)
    x = str2num(oneline);
    x2 = num2cell(x);
    if length(x) == 7 && x(3) >= lower_transect_ID_number && x(3) <= upper_transect_ID_number
&& x(7) ~= 9999
        % the length of the headline of each measurement
        x(1) = new_kv_num;  % change the coastal area number, which is in the first column of
a headline
        fprintf(newfile,'%s\n', num2str(x)); % print a new line in the new file with the
updated number

        oneline = fgetl(open_file); % Set the next line in the original file to be read next

        while ischar(oneline) && length(str2num(oneline)) ~= 7
            x = str2num(oneline);
            x2 = num2cell(x);
            if length(x) == 10 || length(x) == 8 || length(x) == 6 || length(x) == 4 ||
length(x) == 2
                if length(x) == 10      % Check the length of a data line (lengths can be 2,
4, 6, 8 and 10)
                    for i = 1:(length(x)/2)      % This only happens for the Y values
                        if length(num2str(x(i*2))) == 1 % as the last number of the y values
indicate the type of measurement
                            x2{i*2} = ['0', num2str(x(i*2))];   % If there is only one number
than the actual y value was '0'
                        end
                    end
                    x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ',
...
                        num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6}),...
                        '    ', num2str(x2{7}),'    ', num2str(x2{8}),'    ', num2str(x2{9}),...
                        '    ', num2str(x2{10})];     % Make the new line correct
                elseif length(x) == 8
                    for i = 1:(length(x)/2)
                        if length(num2str(x(i*2))) == 1
                            x2{i*2} = ['0', num2str(x(i*2))];
                        end
                    end
                    x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ',
...
                        num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6}),...
                        '    ', num2str(x2{7}),'    ', num2str(x2{8})];
                elseif length(x) == 6
                    for i = 1:(length(x)/2)
                        if length(num2str(x(i*2))) == 1
                            x2{i*2} = ['0', num2str(x(i*2))];
                        end
                    end
                    x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ',
...
                        num2str(x2{4}),'    ', num2str(x2{5}),'    ', num2str(x2{6})];
                elseif length(x) == 4
```

```
                        for i = 1:(length(x)/2)
                            if length(num2str(x(i*2))) == 1
                                x2{i*2} = ['0', num2str(x(i*2))];
                            end
                        end
                        x = [num2str(x2{1}),'    ', num2str(x2{2}),'    ', num2str(x2{3}), '    ',
...
                            num2str(x2{4})];
                    elseif length(x) == 2
                        if length(num2str(x(2))) == 1
                            x2{2} = ['0', num2str(x(2))];
                        end
                        x = [num2str(x2{1}),'    ', num2str(x2{2})];
                    end

                    fprintf(newfile,'%s\n', x); % print a new line as the old line. NOTE: if a y
value
                    % was 0#, it is converted to # as the str2num makes it num (eating
                    % the leading zero's). This part reflects if this happened and
                    % returns '0#' to the correct locations if it happened
                end
                oneline = fgetl(open_file); % Set the next line in the original file to be read
next
            end
        else
            oneline = fgetl(open_file); % Set the next line in the original file to be read next
        end
end
fclose(open_file);
fclose(newfile);

%% elapse time
toc
```

3.2 Converting JARKUS to MATLAB structure

The *load_jarkus* script converts the JARKUS structure data to a MATLAB structure data. When using this script, the dir_in variable should be changed to the directory in which the user has saved the *.jrk files. Furthermore, the script saves the data structures in the 'Data_structs' output folder. It scans trough the lines of the *.jrk files which are located in the directory. The script makes use of the *GET_X_*Y function, which extracts the x and y values from the current data line and removes the last digit from the y value.

Script 3: *Load_jarkus.m*

```
%% Converting Jarkus file structure to a useable structure in matlab
% Ivo Naus

%% initialization
close all
clear all

tic

%% Adding paths

addpath('Functions','Data_jrk');

%% Automatically load files in the dir_in directory and convert to new structure

% the directory where the .jrk files are located
dir_in = 'C:\Users\ivo\Documents\1 . Rijkswaterstaat stage\Werkmap\Matlab\Ivo
scripts\Better\Load jarkus\Data_jrk\';
```

```
files = dir([dir_in,'*.jrk']);

% loop over all the files in the directory
for k = 1:length(files)
    % automatically generate output file name based on input file name
    inputfile = files(k).name;
    savestruct_name = files(k).name(1:end-4);

    open_file = fopen(inputfile);

    % generating a base struct (structure) layout for the data
    Data =
struct('x',[],'y',[],'transect',[],'year',[],'num_of_points',[],'coastal_area_number',[]);


%% Initialisation
    oneline = fgetl(open_file); % read first line

    n = 0; % used to scroll down in the struct (to the next measurement)

    %% filling in the data and metadata in the structure
    while ischar(oneline)

        split_line_first = strsplit(oneline);              % split the line into cells

        if isempty(oneline)
            split_line = split_line_first;
        elseif oneline(1) == ' '                          % otherwise the first cell is empty,
which gives an error
            split_line_second = split_line_first(2:end);
            if isempty(split_line_second{end})
                split_line = split_line_second(1:end-1);
            else
                split_line = split_line_second;
            end
        else
            split_line_second = split_line_first;
            if isempty(split_line_second{end})
                split_line = split_line_second(1:end-1);
            else
                split_line = split_line_second;
            end
        end

        if isempty(oneline)              % skip an empty line
            oneline = fgetl(open_file); % Set the next line in the original file to be read
next
            % Check if the line is a headline: metadata lines contain 7
            % collums, some data lines also contain 7 collumns as sometimes
            % they are filled with '9999'
        elseif length(split_line) == 7 && str2double(split_line{end}) < 9999

            % Getting the metadata from the line, later to be saved to the
            % structure
            coast_area_num = str2double(split_line{1});
            year = str2double(split_line{2});
            transect = str2double(split_line{3});
            meas_type = str2double(split_line{4});
            dry_date = str2double(split_line{5});
            wet_date = str2double(split_line{6});
            num_of_data_points = str2double(split_line{7});

            n = n+1;     % to scroll down in the struct

            % storing the metadata of the new measurement & storing the x and y data of the
old measurement
            if n == 1    % first measurement of the data: storing the metadata
                Data(1).transect = transect;
                Data(1).year = year;
                Data(1).num_of_points = num_of_data_points;
                Data(1).coastal_area_number = coast_area_num;
            else
                % storing the x and y data of the old measurement
                % and storing the metadata of the new measurement
```

```
                Data(n-1).x = x;
                Data(n-1).y = y;

                % In case there are more datapoints in the data than there were
                % measured. (i.e. if the data is filled with 9999999 at the
                % end)
                % Than: do not copy these 'false' points into the new structure
                if length(Data(n-1).x) - Data(n-1).num_of_points > 0
                    Data(n-1).x = Data(n-1).x(1:end-(length(Data(n-1).x) - Data(n-
1).num_of_points));
                    Data(n-1).y = Data(n-1).y(1:end-(length(Data(n-1).y) - Data(n-
1).num_of_points));
                end

                Data(n).transect = transect;
                Data(n).year = year;
                Data(n).num_of_points = num_of_data_points;
                Data(n).coastal_area_number = coast_area_num;
            end

            % empty x and y for the new measurement
            x = [];
            y = [];

            oneline = fgetl(open_file); % Set the next line in the original file to be read
next
        elseif length(split_line) == 10
            % adding to the x and y arrays
            [x, y] = GET_X_Y(split_line,x,y);

            oneline = fgetl(open_file); % Set the next line in the original file to be read
next
        else % when the number of collumns is not 10 or 7, which is only the case when it is
the last row of the data
            % adding to the x and y arrays
            [x, y] = GET_X_Y(split_line,x,y);

            oneline = fgetl(open_file); % Set the next line in the original file to be read
next
            while ischar(oneline) && length(strsplit(oneline)) ~= 7 % in case a metadata line
is incorrect (which leads to wrong results of this script)
                oneline = fgetl(open_file); % Set the next line in the original file to be
read next
            end
        end
    end

    % storing the x and y values of the last measurement
    Data(n).x = x;
    Data(n).y = y;

    %% Data saving

    save(['Data_structs\',savestruct_name,'.mat'],'Data');
end

toc
```

Script 4: *change_tannis_bugt.m*

```
%% Change data 2016 tannis bugt by multimpling by (-1)
% Ivo Naus

%% initialization
close all
clear all
```

```
tic

%% Adding paths

addpath('Functions','Data_structs');

%% automatisch bestanden in de betreffende directory inladen en wegschrijven

load('45000006_Vestkyst_TannisBugt.mat')

for k = 1:length(Data)
    if Data(k).year == 2016
        Data(k).y = Data(k).y * (-1);
    end
end


%% Data saving

save(['Data_structs\','45000006_Vestkyst_TannisBugt_2','.mat'],'Data');

toc
```

3.3 Calculating intersection points

Script 5: *Calculate_intersection_points.m*

```
%% get the intersection points of all the profiles with a certain height line
% Ivo Naus 2017, RWS WVL

clear all
close all

tic

addpath('Functions','Data_structs');

%% Automatically load files in the dir_in directory, and set a name for the resulting
structure
dir_in = 'C:\Users\ivo\Documents\1 . Rijkswaterstaat stage\Werkmap\Matlab\Ivo
scripts\Beter\Calculating Characteristics\Data_structs\';
files = dir([dir_in,'*.mat']);

% name of the saved file
save_struct_name_file = 'intersection_points';

save_struct_name = 'intersection_points';

%% Variables used

% heigt of horizontal line (intersection):
inter_height_zero = 0;
inter_height_min_2 = -2;
inter_height_plus_2 = 2;
inter_height_plus_4 = 4;
inter_height_min_6 = -6;
inter_height_min_8 = -8;
inter_height_min_4 = -4;
```

```
%% A loop over all the files within the directiory chosen as the dir_in variable

for k = 1:length(files)

    inputfile = files(k).name;

    load = load(inputfile);

    MakeCell = struct2cell(load);

    data = cell2struct(MakeCell, inputfile(1:end-4));

    % making a structure within the structure for every coastal area
    intersection_points.(inputfile(1:end-4)) =
struct('transect',[],'year',[],'coastal_area_number',[],'num_of_points',[],'x_is_0',[]);


    length_loop = length(data.(inputfile(1:end-4)));



    for i = 1:length_loop

        intersection_points.(inputfile(1:end-4))(i).transect = data.(inputfile(1:end-
4))(i).transect;
        intersection_points.(inputfile(1:end-4))(i).year = data.(inputfile(1:end-4))(i).year;
        intersection_points.(inputfile(1:end-4))(i).coastal_area_number =
data.(inputfile(1:end-4))(i).coastal_area_number;
        intersection_points.(inputfile(1:end-4))(i).num_of_points = data.(inputfile(1:end-
4))(i).num_of_points;

        % determine the intersection points with the horizontal line at a
        % height of 0 m
        Intersection_zero = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_zero inter_height_zero]]);

        % save the intersection point in the structure
        intersection_points.(inputfile(1:end-4))(i).x_is_0 = Intersection_zero;


    end

    %% Determine intersection point with y=-2

    intersection_points.(inputfile(1:end-4))(1).x_is_min_2 = [];

    for i = 1:length_loop

        Intersection_min_2 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_min_2 inter_height_min_2]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_min_2 = Intersection_min_2;

    end

    %% Determine intersection point with y=2

    intersection_points.(inputfile(1:end-4))(1).x_is_plus_2 = [];

    for i = 1:length(data.(inputfile(1:end-4)))

        Intersection_plus_2 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_plus_2 inter_height_plus_2]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_plus_2 = Intersection_plus_2;

    end
```

```
    %% Determine intersection point with y=4

    intersection_points.(inputfile(1:end-4))(1).x_is_plus_4 = [];

    for i = 1:length(data.(inputfile(1:end-4)))

        Intersection_plus_4 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_plus_4 inter_height_plus_4]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_plus_4 = Intersection_plus_4;

    end

    %% Determine intersection point with y=-6

    intersection_points.(inputfile(1:end-4))(1).x_is_min_6 = [];

    for i = 1:length(data.(inputfile(1:end-4)))

        Intersection_min_6 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_min_6 inter_height_min_6]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_min_6 = Intersection_min_6;

    end

    %% Determine intersection point with y=-8

    intersection_points.(inputfile(1:end-4))(1).x_is_min_8 = [];

    for i = 1:length(data.(inputfile(1:end-4)))

        Intersection_min_8 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_min_8 inter_height_min_8]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_min_8 = Intersection_min_8;

    end

    %% Determine intersection point with y=-4

    intersection_points.(inputfile(1:end-4))(1).x_is_min_4 = [];

    for i = 1:length(data.(inputfile(1:end-4)))

        Intersection_min_4 = InterX([data.(inputfile(1:end-4))(i).x';data.(inputfile(1:end-
4))(i).y'],[[data.(inputfile(1:end-4))(i).x(1) data.(inputfile(1:end-
4))(i).x(end)];[inter_height_min_4 inter_height_min_4]]);

        intersection_points.(inputfile(1:end-4))(i).x_is_min_4 = Intersection_min_4;

    end

    clear load
end

%% Save data

save(['Output\',save_struct_name_file,'.mat'],save_struct_name);

toc
```

3.4 Calculating the widths and slopes

The widths were calculated with the *Width_diff_years_final* script. This stands for the calculation of the width, using a different year for the Danish coastal areas to get the transects of which the mean was determined (described in Chapter 3) and this script was the last version which was used in this project. All the indices of the transects which were measured during a certain year were found using the *GetStructIndex* function, which searched through the whole structure to find the matching transect number and/or year. With the use of the calculated widths the slopes were determined in the *Slope_final* script.

<u>Script 6:</u> *Width_diff_years_final.m*

```
%% Calculation of the distance between certain height points of each profile
% Ivo Naus

clear all
close all

tic

%% load data

addpath('Functions','Data_structs','Characteristics');

intersection_points = load('intersection_points2.mat');

data_name = fieldnames(intersection_points);   % Get the name of the loaded struct (field 1 in
struct 'Data')
intersection_points = getfield(intersection_points, data_name{1});    % Change 'Data' to the
loaded struct (not a struct within a struct anymore

% save file names

save_struct_name_file = 'Widths2';

save_struct_name = 'Widths';

save_struct_name_file_2 = 'MeanWidths2';

save_struct_name_2 = 'MeanWidths';

%% Determine the distance (width) between two elevation points


FieldNames = fieldnames(intersection_points);

for k = 1:length(FieldNames)

    % Structure in which the widths will be saved

    Widths.(FieldNames{k}) =
struct('transect',[],'year',[],'coastal_area_number',[],'num_of_points',[],...

'Width_min_2_plus_4',[],'Width_0_plus_2',[],'Width_0_plus_4',[],'Width_min_2_min_6',[],...

'Width_min_2_min_8',[],'Width_plus_2_min_2',[],'Width_min_2_min_4',[],'Width_min_4_min_8',[],'
Width_0_min_2',[]);

    length_loop = length(intersection_points.(FieldNames{k}));
```

```
    for i = 1:length_loop

        Widths.(FieldNames{k})(i).transect = intersection_points.(FieldNames{k})(i).transect;
        Widths.(FieldNames{k})(i).year = intersection_points.(FieldNames{k})(i).year;
        Widths.(FieldNames{k})(i).coastal_area_number =
intersection_points.(FieldNames{k})(i).coastal_area_number;
        Widths.(FieldNames{k})(i).num_of_points =
intersection_points.(FieldNames{k})(i).num_of_points;

    end


    for n = 1:length_loop
        % calculating the distances between 2 height points
        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_2) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_plus_4)
            Width1 = [];
        else
            Width1 = intersection_points.(FieldNames{k})(n).x_is_min_2(1,1)-
intersection_points.(FieldNames{k})(n).x_is_plus_4(1,end);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_0) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_plus_2)
            Width2 = [];
        else
            Width2 = intersection_points.(FieldNames{k})(n).x_is_0(1,end)-
intersection_points.(FieldNames{k})(n).x_is_plus_2(1,end);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_0) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_plus_4)
            Width3 = [];
        else
            Width3 = intersection_points.(FieldNames{k})(n).x_is_0(1,end)-
intersection_points.(FieldNames{k})(n).x_is_plus_4(1,end);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_2) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_6)
            Width4 = [];
        else
            Width4 = intersection_points.(FieldNames{k})(n).x_is_min_6(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_2(1,1);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_2) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_8)
            Width5 = [];
        else
            Width5 = intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_2(1,1);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_2) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_plus_2)
            Width6 = [];
        else
            Width6 = intersection_points.(FieldNames{k})(n).x_is_min_2(1,1)-
intersection_points.(FieldNames{k})(n).x_is_plus_2(1,end);
        end


        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_4) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_2)
            Width7 = [];
        else
```

```
            Width7 = intersection_points.(FieldNames{k})(n).x_is_min_4(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_2(1,1);
        end

        % calculating the distance between -4 and -8, if the distance was
        % negative than take the next -4 point untill it is not negative,
        % or until 5 points have been skipped which almost certainly can
        % only be due to measurement errors
        if isempty(intersection_points.(FieldNames{k})(n).x_is_min_4) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_8)
            Width8 = [];
        elseif (intersection_points.(FieldNames{k})(n).x_is_min_8(1,1) -
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end)) < 0
            if length(intersection_points.(FieldNames{k})(n).x_is_min_4(1,:)) > 1
                if (intersection_points.(FieldNames{k})(n).x_is_min_8(1,1) -
intersection_points.(FieldNames{k})(n).x_is_min_4(1,(end-1))) < 0
                    if length(intersection_points.(FieldNames{k})(n).x_is_min_4(1,:)) > 2
                        if (intersection_points.(FieldNames{k})(n).x_is_min_8(1,1) -
intersection_points.(FieldNames{k})(n).x_is_min_4(1,(end-2))) < 0
                            if length(intersection_points.(FieldNames{k})(n).x_is_min_4(1,:))
> 3
                                if (intersection_points.(FieldNames{k})(n).x_is_min_8(1,1) -
intersection_points.(FieldNames{k})(n).x_is_min_4(1,(end-3))) < 0
                                    if
length(intersection_points.(FieldNames{k})(n).x_is_min_4(1,:)) > 4
                                        if
(intersection_points.(FieldNames{k})(n).x_is_min_8(1,1) -
intersection_points.(FieldNames{k})(n).x_is_min_4(1,(end-4))) < 0
                                            if
length(intersection_points.(FieldNames{k})(n).x_is_min_4(1,:)) > 5
                                                Width8 = [];
                                            else
                                                Width8 = [];
                                            end
                                        else
                                            Width8 =
intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end-4);
                                        end
                                    else
                                        Width8 = [];
                                    end
                                else
                                    Width8 =
intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end-3);
                                end
                            else
                                Width8 = [];
                            end
                        else
                            Width8 = intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end-2);
                        end
                    else
                        Width8 = [];
                    end
                else
                    Width8 = intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end-1);
                end
            else
                Width8 = [];
            end
        else
            Width8 = intersection_points.(FieldNames{k})(n).x_is_min_8(1,1)-
intersection_points.(FieldNames{k})(n).x_is_min_4(1,end);
        end

        if isempty(intersection_points.(FieldNames{k})(n).x_is_0) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_2)
            Width9 = [];
        elseif isempty(intersection_points.(FieldNames{k})(n).x_is_0) ||
isempty(intersection_points.(FieldNames{k})(n).x_is_min_2) < 0
            Width9 = [];
        else
```

```
            Width9 = intersection_points.(FieldNames{k})(n).x_is_min_2(1,1)-
intersection_points.(FieldNames{k})(n).x_is_0(1,end);
        end

        % Saving the widths in the structure


        if isempty(Width1)
            Widths.(FieldNames{k})(n).Width_min_2_plus_4 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_min_2_plus_4 = Width1;
        end

        if isempty(Width2)
            Widths.(FieldNames{k})(n).Width_0_plus_2 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_0_plus_2 = Width2;
        end

        if isempty(Width3)
            Widths.(FieldNames{k})(n).Width_0_plus_4 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_0_plus_4 = Width3;
        end

        if isempty(Width4)
            Widths.(FieldNames{k})(n).Width_min_2_min_6 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_min_2_min_6 = Width4;
        end

        if isempty(Width5)
            Widths.(FieldNames{k})(n).Width_min_2_min_8 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_min_2_min_8 = Width5;
        end

        if isempty(Width6)
            Widths.(FieldNames{k})(n).Width_plus_2_min_2 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_plus_2_min_2 = Width6;
        end

        if isempty(Width7)
            Widths.(FieldNames{k})(n).Width_min_2_min_4 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_min_2_min_4 = Width7;
        end

        if isempty(Width8)
            Widths.(FieldNames{k})(n).Width_min_4_min_8 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_min_4_min_8 = Width8;
        end

        if isempty(Width9)
            Widths.(FieldNames{k})(n).Width_0_min_2 = NaN;
        else
            Widths.(FieldNames{k})(n).Width_0_min_2 = Width9;
        end


    end


    %% Make a struct with the indices of each year for each transect which is measured in the
past
    % and one for those only measured in 2016
    % Determine the mean and standard deviation of the widths calculated
    % for theses specific measurements
    MeanWidths.(FieldNames{k}) =
struct('transect',[],'years',[],'indexdata',[],'mean_width_min2plus4', [],...
        'mean_width_0plus4',[],'mean_width_0plus2',[],'mean_width_min2min6',[],...
```

```
        'mean_width_min2min8',[],'mean_width_plus2min2',[],'mean_width_min2min4',[],...

'mean_width_min4min8',[],'mean_width_0min2',[],'mean_width_min2plus4_2006_2016',[],'mean_width
_0plus4_2006_2016',[],...
        'mean_width_0plus2_2006_2016',[],'mean_width_min2min6_2006_2016',[]...

,'mean_width_min2min8_2006_2016',[],'mean_width_plus2min2_2006_2016',[],'mean_width_min2min4_2
006_2016',[]...
        ,'mean_width_min4min8_2006_2016',[],'mean_width_0min2_2006_2016',[]...

,'stdv_width_min2plus4',[],'stdv_width_0plus4',[],'stdv_width_0plus2',[],'stdv_width_min2min6'
,[],...
        'stdv_width_min2min8',[],'stdv_width_plus2min2',[],...
        'stdv_width_min2min4',[],'stdv_width_min4min8',[],'stdv_width_0min2',[]...
        ,'stdv_width_min2plus4_2006_2016',[],'stdv_width_0plus4_2006_2016',[],...

'stdv_width_0plus2_2006_2016',[],'stdv_width_min2min6_2006_2016',[],'stdv_width_min2min8_2006_
2016',...
        [],'stdv_width_plus2min2_2006_2016',[],'stdv_width_min2min4_2006_2016',[]...
        ,'stdv_width_min4min8_2006_2016',[],'stdv_width_0min2_2006_2016',[]);

    % To get all the widths (throughout the years) of one transect, the
    % index numbers on which these widths are saved in the structure, for that specific
    % transect, have to be known. The function GetStructIndex does this for
    % all measurements measured for a given year. It has been decided to
    % use only all the transects which were measured during the same year.

    % Get the index of each transect which was measured in 2016, except for
    % the coastal areas in denmark, as a lot of transects weren't measured
    % in 2016 a different year has been taken for each coastal area, based
    % on the amount of transects measured during that year:
    % Vadehavsoer: 2014
    % Midtjylland: 2014
    % Agger: 2016 is good
    % Nationalpark-thy: 2009
    % VigsoJammerbugten: 1995
    % Tannis-Bugt: 2008
    % Holmsland: 2014
    if length((FieldNames{k})) == length('Vestkyst_Vadehavsoer2_45000001') & (FieldNames{k})
== 'Vestkyst_Vadehavsoer2_45000001'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2014, []);
    elseif length((FieldNames{k})) == length('Vestkyst_Midtjylland_45000002') &
(FieldNames{k}) == 'Vestkyst_Midtjylland_45000002'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2014, []);
    elseif length((FieldNames{k})) == length('Vestkyst_Agger_45000003') & (FieldNames{k}) ==
'Vestkyst_Agger_45000003'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2016, []);
    elseif length((FieldNames{k})) == length('Vestkyst_NationalparkThy_45000004') &
(FieldNames{k}) == 'Vestkyst_NationalparkThy_45000004'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2009, []);
    elseif length((FieldNames{k})) == length('Vestkyst_VigsoJammerbugten_45000005') &
(FieldNames{k}) == 'Vestkyst_VigsoJammerbugten_45000005'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 1995, []);
    elseif length((FieldNames{k})) == length('Vestkyst_TannisBugt_45000006') & (FieldNames{k})
== 'Vestkyst_TannisBugt_45000006'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2008, []);
    elseif length((FieldNames{k})) == length('Holmsland_data_450000027') & (FieldNames{k}) ==
'Holmsland_data_450000027'
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2014, []);
    else
        Index_2016 = GetStructIndex(intersection_points.(FieldNames{k}), 2016, []);
    end

    transect_num_2016 = NaN(length(Index_2016),1);

    for n = 1:length(Index_2016)
        transect_num_2016(n) = intersection_points.(FieldNames{k})(Index_2016(n)).transect;
    end

    % fill in the structure with metadata
    for n = 1:length(transect_num_2016)
        MeanWidths.(FieldNames{k})(n).transect = transect_num_2016(n);
        % indexdata is the index at which the data, used for the
        % calculation of the mean, is found in the structure
        MeanWidths.(FieldNames{k})(n).indexdata =
GetStructIndex(intersection_points.(FieldNames{k}), [], transect_num_2016(n));
```

```
    end

    for n = 1:length(MeanWidths.(FieldNames{k}))
        for m = 1:length(MeanWidths.(FieldNames{k})(n).indexdata)
            % the available years (years in which the transects were
            % measured) for each transect
            MeanWidths.(FieldNames{k})(n).years(m) =
intersection_points.(FieldNames{k})(MeanWidths.(FieldNames{k})(n).indexdata(m)).year;
        end
    end

    %% Mean width per transect that has been measured during the same year per coastal area
    % for most coastal areas this year is 2016, except for the Danish
    % coastal areas, the years taken for these areas is mentioned above

    for n = 1:length(MeanWidths.(FieldNames{k}))
        % calculate and save the mean widths in the structure
        MeanWidths.(FieldNames{k})(n).mean_width_min2plus4 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_plus_4]
);
        MeanWidths.(FieldNames{k})(n).mean_width_0plus4 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_plus_4]);
        MeanWidths.(FieldNames{k})(n).mean_width_0plus2 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_plus_2]);
        MeanWidths.(FieldNames{k})(n).mean_width_min2min6 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_6])
;
        MeanWidths.(FieldNames{k})(n).mean_width_min2min8 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_8])
;
        MeanWidths.(FieldNames{k})(n).mean_width_plus2min2 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_plus_2_min_2]
);
        MeanWidths.(FieldNames{k})(n).mean_width_min2min4 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_4])
;
        MeanWidths.(FieldNames{k})(n).mean_width_min4min8 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_4_min_8])
;
        MeanWidths.(FieldNames{k})(n).mean_width_0min2 =
nanmean([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_min_2]);
        % save the standard deviation in the structure
        MeanWidths.(FieldNames{k})(n).stdv_width_min2plus4 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_plus_4])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_0plus4 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_plus_4]);
        MeanWidths.(FieldNames{k})(n).stdv_width_0plus2 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_plus_2]);
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min6 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_6]);
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min8 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_8]);
        MeanWidths.(FieldNames{k})(n).stdv_width_plus2min2 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_plus_2_min_2])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min4 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_2_min_4]);
        MeanWidths.(FieldNames{k})(n).stdv_width_min4min8 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_min_4_min_8]);
        MeanWidths.(FieldNames{k})(n).stdv_width_0min2 =
nanstd([Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata)).Width_0_min_2]);
    end

    %% Mean width per transect that has been measured during the same year per coastal area

    for n = 1:length(MeanWidths.(FieldNames{k}))
        usable = find(MeanWidths.(FieldNames{k})(n).years >= 2006);
       % save the mean widths in the structure
        MeanWidths.(FieldNames{k})(n).mean_width_min2plus4_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_plus_4]
);
        MeanWidths.(FieldNames{k})(n).mean_width_0plus4_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_plus_4]);
```

```
        MeanWidths.(FieldNames{k})(n).mean_width_0plus2_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_plus_2]);
        MeanWidths.(FieldNames{k})(n).mean_width_min2min6_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_6])
;
        MeanWidths.(FieldNames{k})(n).mean_width_min2min8_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_8])
;
        MeanWidths.(FieldNames{k})(n).mean_width_plus2min2_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_plus_2_min_2]
);
        MeanWidths.(FieldNames{k})(n).mean_width_min2min4_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_4])
;
        MeanWidths.(FieldNames{k})(n).mean_width_min4min8_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_4_min_8])
;
        MeanWidths.(FieldNames{k})(n).mean_width_0min2_2006_2016 = nanmean(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_min_2]);
        % save the standard deviation in the structure
        MeanWidths.(FieldNames{k})(n).stdv_width_min2plus4_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_plus_4]
);
        MeanWidths.(FieldNames{k})(n).stdv_width_0plus4_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_plus_4]);
        MeanWidths.(FieldNames{k})(n).stdv_width_0plus2_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_plus_2]);
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min6_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_6])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min8_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_8])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_plus2min2_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_plus_2_min_2]
);
        MeanWidths.(FieldNames{k})(n).stdv_width_min2min4_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_2_min_4])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_min4min8_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_min_4_min_8])
;
        MeanWidths.(FieldNames{k})(n).stdv_width_0min2_2006_2016 = nanstd(...

[Widths.(FieldNames{k})((MeanWidths.(FieldNames{k})(n).indexdata(usable))).Width_0_min_2]);

        usable = [];
    end

end

%% Save data

save(['Output\',save_struct_name_file,'.mat'],save_struct_name);
save(['Output\',save_struct_name_file_2,'.mat'],save_struct_name_2);

toc
```

Script 7: *Slope_final.m*

```
%% Calculate the slopes from the widths, Ivo Naus


clear all
close all

tic
%% Load data/add paths

addpath('Functions','Data_structs','Characteristics');

MeanWidths = load('MeanWidths2.mat');
data_name = fieldnames(MeanWidths);   % Get the name of the loaded struct (field 1 in struct
'Data')
MeanWidths = getfield(MeanWidths, data_name{1});    % Change 'Data' to the loaded struct (not
a struct within a struct anymore

Widths = load('Widths2.mat');
data_name = fieldnames(Widths);   % Get the name of the loaded struct (field 1 in struct
'Data')
Widths = getfield(Widths, data_name{1});    % Change 'Data' to the loaded struct (not a struct
within a struct anymore

% save file names

save_struct_name_file = 'Slopes2';

save_struct_name = 'Slopes';

save_struct_name_file_2 = 'MeanSlopes2';

save_struct_name_2 = 'MeanSlopes';
%% Calculating the slopes

fieldnames_widths = fieldnames(Widths);


for n = 1:length(fieldnames_widths)
    for m = 1:length(Widths.(fieldnames_widths{n}))
        % Saving meta-data in the new structure for the slopes
        Slopes.(fieldnames_widths{n})(m).transect = Widths.(fieldnames_widths{n})(m).transect;
        Slopes.(fieldnames_widths{n})(m).year = Widths.(fieldnames_widths{n})(m).year;
        Slopes.(fieldnames_widths{n})(m).coastal_area_number =
Widths.(fieldnames_widths{n})(m).coastal_area_number;
        Slopes.(fieldnames_widths{n})(m).num_of_points =
Widths.(fieldnames_widths{n})(m).num_of_points;

        % calculating and saving the slopes in the struct
        Slopes.(fieldnames_widths{n})(m).Slope_min_2_plus_4 =
Widths.(fieldnames_widths{n})(m).Width_min_2_plus_4/6;
        Slopes.(fieldnames_widths{n})(m).Slope_0_plus_2 =
Widths.(fieldnames_widths{n})(m).Width_0_plus_2/2;
        Slopes.(fieldnames_widths{n})(m).Slope_0_plus_4 =
Widths.(fieldnames_widths{n})(m).Width_0_plus_4/4;
        Slopes.(fieldnames_widths{n})(m).Slope_min_2_min_6 =
Widths.(fieldnames_widths{n})(m).Width_min_2_min_6/4;
        Slopes.(fieldnames_widths{n})(m).Slope_min_2_min_8 =
Widths.(fieldnames_widths{n})(m).Width_min_2_min_8/6;
        Slopes.(fieldnames_widths{n})(m).Slope_plus_2_min_2 =
Widths.(fieldnames_widths{n})(m).Width_plus_2_min_2/4;
        Slopes.(fieldnames_widths{n})(m).Slope_min_2_min_4 =
Widths.(fieldnames_widths{n})(m).Width_min_2_min_4/2;
        Slopes.(fieldnames_widths{n})(m).Slope_min_4_min_8 =
Widths.(fieldnames_widths{n})(m).Width_min_4_min_8/4;
        Slopes.(fieldnames_widths{n})(m).Slope_0_min_2 =
Widths.(fieldnames_widths{n})(m).Width_0_min_2/2;
```

```
    end
end



for n = 1:length(fieldnames_widths)
    for m = 1:length(MeanWidths.(fieldnames_widths{n}))
        % Saving meta-data in the new structure for the mean slopes
        MeanSlopes.(fieldnames_widths{n})(m).transect =
MeanWidths.(fieldnames_widths{n})(m).transect;
        MeanSlopes.(fieldnames_widths{n})(m).years =
MeanWidths.(fieldnames_widths{n})(m).years;
        MeanSlopes.(fieldnames_widths{n})(m).indexdata =
MeanWidths.(fieldnames_widths{n})(m).indexdata;

        % calculating and saving the mean slopes in the struct
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2plus4 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2plus4/6;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0plus4 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0plus4/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0plus2 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0plus2/2;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min6 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min6/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min8 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min8/6;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_plus2min2 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_plus2min2/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min4 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min4/2;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min4min8 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min4min8/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0min2 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0min2/2;

        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2plus4_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2plus4_2006_2016/6;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0plus4_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0plus4_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0plus2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0plus2_2006_2016/2;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min6_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min6_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min8_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min8_2006_2016/6;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_plus2min2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_plus2min2_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min2min4_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min2min4_2006_2016/2;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_min4min8_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_min4min8_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).mean_slope_0min2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).mean_width_0min2_2006_2016/2;

        % calculating and saving the stdv slopes in the struct
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2plus4 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2plus4/6;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0plus4 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0plus4/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0plus2 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0plus2/2;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min6 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min6/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min8 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min8/6;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_plus2min2 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_plus2min2/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min4 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min4/2;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min4min8 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min4min8/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0min2 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0min2/2;

        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2plus4_2006_2016 = ...
```

```
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2plus4_2006_2016/6;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0plus4_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0plus4_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0plus2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0plus2_2006_2016/2;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min6_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min6_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min8_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min8_2006_2016/6;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_plus2min2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_plus2min2_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min2min4_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min2min4_2006_2016/2;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_min4min8_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_min4min8_2006_2016/4;
        MeanSlopes.(fieldnames_widths{n})(m).stdv_slope_0min2_2006_2016 = ...
            MeanWidths.(fieldnames_widths{n})(m).stdv_width_0min2_2006_2016/2;
    end
end

%% save data

save(['Output\',save_struct_name_file,'.mat'],save_struct_name);
save(['Output\',save_struct_name_file_2,'.mat'],save_struct_name_2);

toc
```

3.5 Calculating the Volume and trends in volume changes

By executing the the *oetsettings.m* script from the Deltares Matlab OpenEarth Tools, instructions found at: https://publicwiki.deltares.nl/display/OET/oetsettings, the paths needed to use the function *jarkus_getVolumeFast* are established. This function is used in the *GetVolumes_diff_years_final.m* script to determine the volumes under the profiles between the input boundaries, located at the chosen height points. The mean volumes, over the whole measurement history and over the period 2006-2016, were also determined in the same way this was done for the means of the widths and slopes. Furthermore, trends in the volume changes over the period 2006-2016 were determined in the *VolumeTrends.m* script. Per transect profile a linear trend was fitted through the volume values, one for each year in this period (if there was data available), and the slope of this linear trend was stored in a structure.

Script 8: *GetVolumes_diff_years_final.m*

```
%% Calculation of the volume of the sediment/sand between two height points in the profile.
% along with the mean and standard deviation of the volume, and save it in
% a struct.
% Ivo Naus


% Make sure the Shortcut "OET" has run before running this script! (to add
% the needed paths to the functions used)

clear all
close all

tic

%% load data

% save file names
```

```
save_struct_name_file = 'Volumes2';

save_struct_name = 'Volumes';

save_struct_name_file_2 = 'MeanVolumes2';

save_struct_name_2 = 'MeanVolumes';


addpath('Functions','Data_structs','Characteristics');


% load intersection points
intersection_points = load('intersection_points2.mat');

data_name = fieldnames(intersection_points);   % Get the name of the loaded struct (field 1 in
struct 'Data')
intersection_points = getfield(intersection_points, data_name{1});    % Change 'Data' to the
loaded struct (not a struct within a struct anymore

names_intersection_points = fieldnames(intersection_points);

% load the measurement data_structs, change the dir_in to your local
% directory where the measurement data structures are located
dir_in = 'C:\Users\ivo\Documents\1 . Rijkswaterstaat stage\Werkmap\Matlab\Ivo
scripts\Beter\Calculating Characteristics\Data_structs\';
files = dir([dir_in,'*.mat']);

%% Calculating the volumes

% loop over all the measurement data files in the directory
for k = 1:length(files)
    if length(files(k).name(1:end-4)) == length(names_intersection_points{k}) &
files(k).name(1:end-4) == names_intersection_points{k}

        inputfile = files(k).name;
        load = load(inputfile);
        MakeCell = struct2cell(load);
        data = cell2struct(MakeCell, inputfile(1:end-4));

        length_loop = length(data.(inputfile(1:end-4)));

        % Structure where all the determined volumes will be stored
        Volumes.(inputfile(1:end-4)) =
struct('transect',[],'year',[],'coastal_area_number',[],...

'num_of_points',[],'Vol_min_2_plus_4',[],'Vol_0_plus_2',[],'Vol_0_plus_4',[],'Vol_min_2_min_6'
,[],...
            'Vol_min_2_min_8',[],'Vol_plus_2_min_2',[],'Vol_min_2_min_4',[]);

        % Structure where all the mean volumes will be saved
        MeanVolumes.(inputfile(1:end-4)) =
struct('transect',[],'years',[],'indexdata',[],'mean_vol_min2plus4', [],...
        'mean_vol_0plus4',[],'mean_vol_0plus2',[],'mean_vol_min2min6',[],...
        'mean_vol_min2min8',[],'mean_vol_plus2min2',[],'mean_vol_min2min4',[],...
        'mean_vol_min2plus4_2006_2016',[],'mean_vol_0plus4_2006_2016',[],...

'mean_vol_0plus2_2006_2016',[],'mean_vol_min2min6_2006_2016',[],'mean_vol_min2min8_2006_2016',
...
        [],'mean_vol_plus2min2_2006_2016',[],'mean_vol_min2min4_2006_2016',[],...

'stdv_vol_min2plus4',[],'stdv_vol_0plus4',[],'stdv_vol_0plus2',[],'stdv_vol_min2min6',[],...
        'stdv_vol_min2min8',[],'stdv_vol_plus2min2',[],'stdv_vol_min2min4',[],...
        'stdv_vol_min2plus4_2006_2016',[],'stdv_vol_0plus4_2006_2016',[],...

'stdv_vol_0plus2_2006_2016',[],'stdv_vol_min2min6_2006_2016',[],'stdv_vol_min2min8_2006_2016',
[],...
        'stdv_vol_plus2min2_2006_2016',[],'stdv_vol_min2min4_2006_2016',[]);

        % loop over all the transects in the measurement structure file
        for i = 1:length_loop
```

```
            if intersection_points.(inputfile(1:end-4))(i).transect == data.(inputfile(1:end-
4))(i).transect...
                    && intersection_points.(inputfile(1:end-4))(i).year ==
data.(inputfile(1:end-4))(i).year

                % save the metadata in the new structure
                Volumes.(inputfile(1:end-4))(i).transect = data.(inputfile(1:end-
4))(i).transect;
                Volumes.(inputfile(1:end-4))(i).year = data.(inputfile(1:end-4))(i).year;
                Volumes.(inputfile(1:end-4))(i).coastal_area_number = data.(inputfile(1:end-
4))(i).coastal_area_number;
                Volumes.(inputfile(1:end-4))(i).num_of_points = data.(inputfile(1:end-
4))(i).num_of_points;

                % Calculate volume +4 -2
                if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_plus_4) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_2)...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) -
intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end) <= 0 ...
                        || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) < 0 ...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end) -
min(data.(inputfile(1:end-4))(i).x) < 0
                    Vol1 = [];
                else
                    % Input for the fuction: [Volume] = jarkus_getVolumeFast(x, z,
UpperBoundary, LowerBoundary, LandwardBoundary, SeawardBoundary,varargin)
                    Vol1 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(2,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(2,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1));
                end

                % Save volume in structure
                if isempty(Vol1)
                    Volumes.(inputfile(1:end-4))(i).Vol_min_2_plus_4 = NaN;
                else
                    Volumes.(inputfile(1:end-4))(i).Vol_min_2_plus_4 = Vol1;
                end

                % Calculate volume +2 0
                if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_plus_2) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_0)...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end) -
intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end) <= 0 ...
                        || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end) < 0 ...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end) -
min(data.(inputfile(1:end-4))(i).x) < 0
                    Vol2 = [];
                else
                    % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                    Vol2 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(2,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_0(2,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end));
                end

                % Save volume in structure
                if isempty(Vol2)
                    Volumes.(inputfile(1:end-4))(i).Vol_0_plus_2 = NaN;
                else
                    Volumes.(inputfile(1:end-4))(i).Vol_0_plus_2 = Vol2;
                end

                % Calculate volume +4 0
                if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_plus_4) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_0)...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end) -
intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end) <= 0 ...
```

```
                        || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end) < 0 ...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end) -
min(data.(inputfile(1:end-4))(i).x) < 0
                    Vol3 = [];
                else
                    % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                    Vol3 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(2,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_0(2,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_plus_4(1,end),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_0(1,end));
                end

                % Save volume in structure
                if isempty(Vol3)
                    Volumes.(inputfile(1:end-4))(i).Vol_0_plus_4 = NaN;
                else
                    Volumes.(inputfile(1:end-4))(i).Vol_0_plus_4 = Vol3;
                end

                % Calculate volume -2 -6
                if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_6) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_2)...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_min_6(1,1) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) <= 0 ...
                        || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_6(1,1) < 0 ...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) -
min(data.(inputfile(1:end-4))(i).x) < 0
                    Vol4 = [];
                else
                    % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                    Vol4 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(2,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_6(2,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_6(1,1));
                end

                % Save volume in structure
                if isempty(Vol4)
                    Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_6 = NaN;
                else
                    Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_6 = Vol4;
                end

                % Calculate volume -2 -8
                if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_8) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_2)...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_min_8(1,1) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) <= 0 ...
                        || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_8(1,1) < 0 ...
                        || intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) -
min(data.(inputfile(1:end-4))(i).x) < 0
                    Vol5 = [];
                else
                    % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                    Vol5 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(2,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_8(2,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1),...
                        intersection_points.(inputfile(1:end-4))(i).x_is_min_8(1,1));
                end

                % Save volume in structure
                if isempty(Vol5)
                    Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_8 = NaN;
                else
```

```
                Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_8 = Vol5;
            end

            % Calculate volume +2 -2
            if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_2) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_plus_2)...
                    || intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) -
intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end) <= 0 ...
                    || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) < 0 ...
                    || intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end) -
min(data.(inputfile(1:end-4))(i).x) < 0
                Vol6 = [];
            else
                % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                Vol6 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                    intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(2,end),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_2(2,1),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_plus_2(1,end),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1));
            end

            % Save volume in structure
            if isempty(Vol6)
                Volumes.(inputfile(1:end-4))(i).Vol_plus_2_min_2 = NaN;
            else
                Volumes.(inputfile(1:end-4))(i).Vol_plus_2_min_2 = Vol6;
            end


            % Calculate volume -2 -4
            if isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_2) ||
isempty(intersection_points.(inputfile(1:end-4))(i).x_is_min_4)...
                    || intersection_points.(inputfile(1:end-4))(i).x_is_min_4(1,1) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) <= 0 ...
                    || max(data.(inputfile(1:end-4))(i).x) -
intersection_points.(inputfile(1:end-4))(i).x_is_min_4(1,1) < 0 ...
                    || intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1) -
min(data.(inputfile(1:end-4))(i).x) < 0
                Vol7 = [];
            else
                % [Volume] = jarkus_getVolumeFast(x, z, UpperBoundary, LowerBoundary,
LandwardBoundary, SeawardBoundary,varargin)
                Vol7 = jarkus_getVolumeFast(data.(inputfile(1:end-4))(i).x,
data.(inputfile(1:end-4))(i).y,...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_2(2,1),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_4(2,1),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_2(1,1),...
                    intersection_points.(inputfile(1:end-4))(i).x_is_min_4(1,1));
            end

            % Save volume in structure
            if isempty(Vol7)
                Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_4 = NaN;
            else
                Volumes.(inputfile(1:end-4))(i).Vol_min_2_min_4 = Vol7;
            end

        else
            error('The measurement data and intersection data being used are not from the
same transect measurement')
        end
    end

else
    error(['Names of the data in intersection_points and data_struct are not the same,
possible data mismatch on struct num: ',num2str(k)])
end


%% Make a struct with the indices of each year for each transect which is measured in 2016
% Determine the mean and standard deviation of the widths calculated
% for theses specific measurements
```

```
    % Get the index of each transect which was measured in 2016, except for
    % the coastal areas in denmark, as a lot of transects weren't measured
    % in 2016 a different year has been taken for each coastal area:
    % Vadehavsoer: 2014
    % Midtjylland: 2014
    % Agger: 2016 is good
    % Nationalpark-thy: 2009
    % VigsoJammerbugten: 1995
    % Tannis-Bugt: 2008
    % Holmsland: 2014

    if length((inputfile(1:end-4))) == length('Vestkyst_Vadehavsoer2_45000001') &
(inputfile(1:end-4)) == 'Vestkyst_Vadehavsoer2_45000001'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2014, []);
    elseif length((inputfile(1:end-4))) == length('Vestkyst_Midtjylland_45000002') &
(inputfile(1:end-4)) == 'Vestkyst_Midtjylland_45000002'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2014, []);
    elseif length((inputfile(1:end-4))) == length('Vestkyst_Agger_45000003') &
(inputfile(1:end-4)) == 'Vestkyst_Agger_45000003'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2016, []);
    elseif length((inputfile(1:end-4))) == length('Vestkyst_NationalparkThy_45000004') &
(inputfile(1:end-4)) == 'Vestkyst_NationalparkThy_45000004'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2009, []);
    elseif length((inputfile(1:end-4))) == length('Vestkyst_VigsoJammerbugten_45000005') &
(inputfile(1:end-4)) == 'Vestkyst_VigsoJammerbugten_45000005'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 1995, []);
    elseif length((inputfile(1:end-4))) == length('Vestkyst_TannisBugt_45000006') &
(inputfile(1:end-4)) == 'Vestkyst_TannisBugt_45000006'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2008, []);
    elseif length((inputfile(1:end-4))) == length('Holmsland_data_450000027') &
(inputfile(1:end-4)) == 'Holmsland_data_450000027'
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2014, []);
    else
        Index_2016 = GetStructIndex(intersection_points.(inputfile(1:end-4)), 2016, []);
    end


    transect_num_2016 = NaN(length(Index_2016),1);

    for n = 1:length(Index_2016)
        transect_num_2016(n) = intersection_points.(inputfile(1:end-
4))(Index_2016(n)).transect;
    end

    % fill in the structure, in which the mean volumes will be saved, with metadata
    for n = 1:length(transect_num_2016)
        MeanVolumes.(inputfile(1:end-4))(n).transect = transect_num_2016(n);
        MeanVolumes.(inputfile(1:end-4))(n).indexdata =
GetStructIndex(intersection_points.(inputfile(1:end-4)), [], transect_num_2016(n));
    end

    for n = 1:length(MeanVolumes.(inputfile(1:end-4)))
        for m = 1:length(MeanVolumes.(inputfile(1:end-4))(n).indexdata)
            MeanVolumes.(inputfile(1:end-4))(n).years(m) =
intersection_points.(inputfile(1:end-4))(MeanVolumes.(inputfile(1:end-
4))(n).indexdata(m)).year;
        end
    end

    %% Mean width per transect that has been measured in 2016

    for n = 1:length(MeanVolumes.(inputfile(1:end-4)))
        % save the mean widths in the structure
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2plus4 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_0plus4 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_0_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_0plus2 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_0_plus_2]);
```

```
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min6 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_6]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min8 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_8]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_plus2min2 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_plus_2_min_2]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min4 =
nanmean([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_4]);
        % save the standard deviation in the structure
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2plus4 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_0plus4 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_0_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_0plus2 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_0_plus_2]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min6 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_6]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min8 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_8]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_plus2min2 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_plus_2_min_2]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min4 =
nanstd([Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata)).Vol_min_2_min_4]);
    end

        %% Mean width per transect that has been measured in 2016, between 2006 and 2016

    for n = 1:length(MeanVolumes.(inputfile(1:end-4)))
        usable = find(MeanVolumes.(inputfile(1:end-4))(n).years >= 2006);
       % save the mean widths in the structure
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2plus4_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_0plus4_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_0_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_0plus2_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_0_plus_2]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min6_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_6]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min8_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_8]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_plus2min2_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_plus_2_min_2]);
        MeanVolumes.(inputfile(1:end-4))(n).mean_vol_min2min4_2006_2016 = nanmean(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_4]);
        % save the standard deviation in the structure
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2plus4_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_0plus4_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_0_plus_4]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_0plus2_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_0_plus_2]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min6_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_6]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min8_2006_2016 = nanstd(...
```

```
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_8]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_plus2min2_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_plus_2_min_2]);
        MeanVolumes.(inputfile(1:end-4))(n).stdv_vol_min2min4_2006_2016 = nanstd(...
            [Volumes.(inputfile(1:end-4))((MeanVolumes.(inputfile(1:end-
4))(n).indexdata(usable))).Vol_min_2_min_4]);

        usable = [];
    end

    clear load
end

%% Save data to the Output folder map

save(['Output\',save_struct_name_file,'.mat'],save_struct_name);
save(['Output\',save_struct_name_file_2,'.mat'],save_struct_name_2);

toc
```

Script 9: *VolumeTrends.m*

```
%% Calculating the linear trend in volume change for each profile
% Ivo Naus

clear all
close all

tic

%% load data & add paths & create some variables

% save file names

save_struct_name_file = 'TrendVolumes';

save_struct_name = 'TrendVolumes';


addpath('Functions','Data_structs','Characteristics');

% load mean volume data, used as the indices which indicate the location of
% of the same transect inside the volume data structure
MeanVolumes = load('MeanVolumes2.mat');
data_name = fieldnames(MeanVolumes);   % Get the name of the loaded struct (field 1 in struct
'Data')
MeanVolumes = getfield(MeanVolumes, data_name{1});    % Change 'Data' to the loaded struct
(not a struct within a struct anymore

% load volume data
Volumes = load('Volumes2.mat');
data_name = fieldnames(Volumes);   % Get the name of the loaded struct (field 1 in struct
'Data')
Volumes = getfield(Volumes, data_name{1});    % Change 'Data' to the loaded struct (not a
struct within a struct anymore

areanames = fieldnames(MeanVolumes);

%% Calculating trends per transect in the selection of transects, between 2006 and 2016
% for every transect measured in 2016,
% except for the Danish coastal areas, for which different years are used

nanvalues = 1;
```

```
% First a x and y variable will be determined, than the NaN values will be removed
% after which the linear fit will be determined useing polyfit and the
% slope 'p(1)' will be saved in a structure

% loop over all the coastal areas
for k = 1:length(areanames)
    for n = 1:length(MeanVolumes.(areanames{k}))
        usable = find(MeanVolumes.(areanames{k})(n).years >= 2006);
        % make a x and a y value, where the x value represents the year
        % number within the period and the y value the volume present
        % during that year. This is done for every Volume variable
        x1 = NaN(1,12);
        y1 = NaN(1,12);

        x2 = NaN(1,12);
        y2 = NaN(1,12);

        x3 = NaN(1,12);
        y3 = NaN(1,12);

        x4 = NaN(1,12);
        y4 = NaN(1,12);

        x5 = NaN(1,12);
        y5 = NaN(1,12);

        x6 = NaN(1,12);
        y6 = NaN(1,12);

        x7 = NaN(1,12);
        y7 = NaN(1,12);

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(1) = 1;
                y1(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(2) = 2;
                y1(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(3) = 3;
                y1(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(4) = 4;
                y1(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(5) = 5;
                y1(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(6) = 6;
                y1(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(7) = 7;
                y1(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(8) = 8;
                y1(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(9) = 9;
                y1(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(10) = 10;
                y1(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(11) = 11;
                y1(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])~=1
                x1(12) = 12;
                y1(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
plus_4])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_plus_4
];
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(1) = 1;
                y2(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(2) = 2;
                y2(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(3) = 3;
                y2(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(4) = 4;
                y2(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(5) = 5;
                y2(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(6) = 6;
                y2(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(7) = 7;
                y2(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(8) = 8;
                y2(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(9) = 9;
                y2(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(10) = 10;
                y2(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(11) = 11;
                y2(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])~=1
                x2(12) = 12;
                y2(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
```

```
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_4])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_4];
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(1) = 1;
                y3(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(2) = 2;
                y3(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(3) = 3;
                y3(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(4) = 4;
                y3(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(5) = 5;
                y3(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(6) = 6;
                y3(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(7) = 7;
                y3(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(8) = 8;
                y3(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(9) = 9;
                y3(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(10) = 10;
                y3(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(11) = 11;
                y3(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])~=1
                x3(12) = 12;
                y3(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus
_2])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_0_plus_2];
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(1) = 1;
                y4(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(2) = 2;
                y4(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(3) = 3;
                y4(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(4) = 4;
                y4(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(5) = 5;
                y4(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(6) = 6;
```

```
                y4(6) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(7) = 7;
                y4(7) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(8) = 8;
                y4(8) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(9) = 9;
                y4(9) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(10) = 10;
                y4(10) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(11) = 11;
                y4(11) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])~=1
                x4(12) = 12;
                y4(12) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            elseif 
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_6])
                nanvalues(length(nanvalues)+1) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_6]
;
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ', 
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(1) = 1;
                y5(1) = 
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(2) = 2;
```

```
                y5(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(3) = 3;
                y5(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(4) = 4;
                y5(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(5) = 5;
                y5(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(6) = 6;
                y5(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(7) = 7;
                y5(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(8) = 8;
                y5(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(9) = 9;
                y5(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(10) = 10;
                y5(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(11) = 11;
                y5(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])~=1
                x5(12) = 12;
                y5(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_8])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_8]
;
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(1) = 1;
                y6(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(2) = 2;
                y6(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(3) = 3;
                y6(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(4) = 4;
                y6(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(5) = 5;
                y6(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(6) = 6;
                y6(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(7) = 7;
                y6(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(8) = 8;
                y6(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(9) = 9;
                y6(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(10) = 10;
                y6(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(11) = 11;
                y6(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])~=1
                x6(12) = 12;
                y6(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2
_min_2])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_plus_2_min_2
];
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end

        for m = 1:length(usable)
            if MeanVolumes.(areanames{k})(n).years(usable(m)) == 2006 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(1) = 1;
                y7(1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2007 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(2) = 2;
                y7(2) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2008 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(3) = 3;
                y7(3) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2009 && ...
```

```
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(4) = 4;
                y7(4) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2010 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(5) = 5;
                y7(5) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2011 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(6) = 6;
                y7(6) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2012 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(7) = 7;
                y7(7) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2013 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(8) = 8;
                y7(8) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2014 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(9) = 9;
                y7(9) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2015 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(10) = 10;
                y7(10) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2016 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(11) = 11;
                y7(11) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif MeanVolumes.(areanames{k})(n).years(usable(m)) == 2017 && ...

isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])~=1
                x7(12) = 12;
                y7(12) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
            elseif
isnan([Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_
min_4])
                nanvalues(length(nanvalues)+1) =
[Volumes.(areanames{k})((MeanVolumes.(areanames{k})(n).indexdata(usable(m)))).Vol_min_2_min_4]
;
```

```
            else
                disp(['Something went wrong: wrong year in selected years ', num2str(k),' ',
num2str(n),' ',num2str(m)])
            end
        end


        x1(isnan(x1)) = [];
        y1(isnan(y1)) = [];

        x2(isnan(x2)) = [];
        y2(isnan(y2)) = [];

        x3(isnan(x3)) = [];
        y3(isnan(y3)) = [];

        x4(isnan(x4)) = [];
        y4(isnan(y4)) = [];

        x5(isnan(x5)) = [];
        y5(isnan(y5)) = [];

        x6(isnan(x6)) = [];
        y6(isnan(y6)) = [];

        x7(isnan(x7)) = [];
        y7(isnan(y7)) = [];

        if length(y1) > 2
            trend_min2plus4 = polyfit(x1,y1,1);
        else
            trend_min2plus4 = NaN;
        end

        if length(y2) > 2
            trend_0_plus_4 = polyfit(x2,y2,1);
        else
            trend_0_plus_4 = NaN;
        end

        if length(y3) > 2
            trend_0_plus_2 = polyfit(x3,y3,1);
        else
            trend_0_plus_2 = NaN;
        end

        if length(y4) > 2
            trend_min_2_min_6 = polyfit(x4,y4,1);
        else
            trend_min_2_min_6 = NaN;
        end

        if length(y5) > 2
            trend_min_2_min_8 = polyfit(x5,y5,1);
        else
            trend_min_2_min_8 = NaN;
        end

        if length(y6) > 2
            trend_plus_2_min_2 = polyfit(x6,y6,1);
        else
            trend_plus_2_min_2 = NaN;
        end

        if length(y7) > 2
            trend_min_2_min_4 = polyfit(x7,y7,1);
        else
            trend_min_2_min_4 = NaN;
        end


        % save the trend slopes in the structure
        TrendVolumes.(areanames{k})(n).transect = MeanVolumes.(areanames{k})(n).transect;
```

```
        TrendVolumes.(areanames{k})(n).years = MeanVolumes.(areanames{k})(n).years;
        TrendVolumes.(areanames{k})(n).indexdata = MeanVolumes.(areanames{k})(n).indexdata;

        TrendVolumes.(areanames{k})(n).trend_vol_min2plus4_2006_2016 = trend_min2plus4(1);
        TrendVolumes.(areanames{k})(n).trend_vol_0plus4_2006_2016 = trend_0_plus_4(1);
        TrendVolumes.(areanames{k})(n).trend_vol_0plus2_2006_2016 = trend_0_plus_2(1);
        TrendVolumes.(areanames{k})(n).trend_vol_min2min6_2006_2016 = trend_min_2_min_6(1);
        TrendVolumes.(areanames{k})(n).trend_vol_min2min8_2006_2016 = trend_min_2_min_8(1);
        TrendVolumes.(areanames{k})(n).trend_vol_plus2min2_2006_2016 = trend_plus_2_min_2(1);
        TrendVolumes.(areanames{k})(n).trend_vol_min2min4_2006_2016 = trend_min_2_min_4(1);

        % clear the variables as they are re-used in the next loop
        usable = [];

        clear trend_min2plus4
        clear x1
        clear y1

        clear trend_0_plus_4
        clear x2
        clear y2

        clear trend_0_plus_2
        clear x3
        clear y3

        clear trend_min_2_min_6
        clear x4
        clear y4

        clear trend_min_2_min_8
        clear x5
        clear y5

        clear trend_plus_2_min_2
        clear x6
        clear y6

        clear trend_min_2_min_4
        clear x7
        clear y7
    end
end

%% Saving the data (trends) to the Output folder map

save(['Output\',save_struct_name_file,'.mat'],save_struct_name);

toc
```

3.6 Plotting the results in one overview figure

There are a couple of scripts written for the plotting of the mean characteristics. Because the scripts are mainly doing the same, only one of the scripts is shown here as an example.

Script 10: *Plotting_Slope2Mean_noheads.m*

```
%% plotting the slopes from south to north

clear all
close all
```

```
tic
%% Load data/add paths

addpath('Functions','Data_structs','Characteristics');

% Load the structure under the same name
MeanSlopes = load('MeanSlopes2.mat');
data_name = fieldnames(MeanSlopes);   % Get the name of the loaded struct (field 1 in struct
'Data')
MeanSlopes = getfield(MeanSlopes, data_name{1});    % Change 'Data' to the loaded struct (not
a struct within a struct anymore

%% Setting the values at which the data will split ( to be plotted in different colours)
% slope min4-min8
slope_min4_min8_split_1 = 50;
slope_min4_min8_split_2 = 100;
slope_min4_min8_split_3 = 150;
slope_min4_min8_split_4 = 200;

% slope min2 - min4
slope_min2_min4_split_1 = 30;
slope_min2_min4_split_2 = 60;
slope_min2_min4_split_3 = 90;
slope_min2_min4_split_4 = 120;
slope_min2_min4_split_5 = 150;

% slope plus2-min2
slope_plus2_min2_split_1 = 25;
slope_plus2_min2_split_2 = 50;
slope_plus2_min2_split_3 = 75;
slope_plus2_min2_split_4 = 100;
slope_plus2_min2_split_5 = 125;

% slope plus4-0
slope_plus4_0_split_1 = 20;
slope_plus4_0_split_2 = 40;
slope_plus4_0_split_3 = 60;
slope_plus4_0_split_4 = 80;
slope_plus4_0_split_5 = 100;

%% Making a south-north with the coastal area names, So when using a loop over these names the
order is n-s

% array with the coastal area names in order from south to north. To be
% able plot the coastal areas automatically in the rigth order within one loop
AreaNames_S_N = [string('Middelkerke_detail_320000newnum');
string('zws_vlaanderen_31000017');...
    string('walcheren_31000016'); string('nbeveland_31000015');...
    string('schouwen_31000013'); string('goeree_31000012');...
    string('voorne_31000011'); string('delf_31000009');...
    string('rijnland_31000008'); string('nh_31000007');...
    string('texel_31000006'); string('vlieland_31000005');...
    string('terschelling_31000004'); string('ameland_31000003');...
    string('schier_31000002'); string('Baltrum_data_49260040');...
    string('Langeoog_data_49260050'); string('All_Sylt_49250107');...
    string('Vestkyst_Vadehavsoer2_45000001'); string('Holmsland_data_450000027');...
    string('Vestkyst_Midtjylland_45000002'); string('Vestkyst_Agger_45000003');...
    string('Vestkyst_NationalparkThy_45000004');
string('Vestkyst_VigsoJammerbugten_45000005');...
    string('Vestkyst_TannisBugt_45000006')];

% These characteristics are plotted within the figure loop, the names of
% the characteristics are called using these strings and with the use of a
% loop looping over the different strings in this variable (for chname=..)
characteristic_name_slope = [string('mean_slope_0plus4'); string('mean_slope_plus2min2');...
    string('mean_slope_min2min4'); string('mean_slope_min4min8');...
    string('mean_slope_min4min8'); string('mean_slope_0plus4_2006_2016');...
    string('mean_slope_plus2min2_2006_2016'); string('mean_slope_min2min4_2006_2016');...
    string('mean_slope_min4min8_2006_2016'); string('mean_slope_min4min8_2006_2016')];

%% subfigures in the right order from south to north and without island heads, showing the
slopes
```

```
for chname = 1:length(characteristic_name_slope)
    if chname == 1 || chname == 2 || chname == 3 || chname == 4
        figure(1)
        subplot(4,1,chname)
    elseif chname == 6 || chname == 7 || chname == 8 || chname == 9
        figure(2)
        subplot(4,1,(chname-5))
    else
    end

    for n = 1:length(MeanSlopes.(AreaNames_S_N{1}))
        if chname == 5 || chname == 10
        elseif chname == 4 || chname == 9
            if MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_min4_min8_split_1

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*r'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_1 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min4_min8_split_2

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*m'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_2 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min4_min8_split_3

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*b'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_3 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min4_min8_split_4

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*c'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_4

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*g'); hold on
            elseif isnan(MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}))

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*k'); hold on
            else
                disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
            end
        elseif chname == 3 || chname == 8
            if MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_min2_min4_split_1

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*r'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_1 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min2_min4_split_2

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*m'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_2 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min2_min4_split_3

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*b'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_3 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min2_min4_split_4

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*c'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_4 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) < slope_min2_min4_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*g'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*y'); hold on
            elseif isnan(MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}))

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*k'); hold on
            else
```

```
                disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
            end
        elseif chname == 2 || chname == 7
            if MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_1

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*r'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_1 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_2

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*m'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_2 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_3

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*b'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_3 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_4

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*c'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_4 &&
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*g'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*y'); hold on
            elseif isnan(MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}))

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*k'); hold on
            else
                disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
            end
        elseif chname == 1 || chname == 6
            if MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) <
slope_plus4_0_split_1

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*r'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus4_0_split_1 && MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname})
< slope_plus4_0_split_2

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*m'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus4_0_split_2 && MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname})
< slope_plus4_0_split_3

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*b'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus4_0_split_3 && MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname})
< slope_plus4_0_split_4

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*c'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus4_0_split_4 && MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname})
< slope_plus4_0_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*g'); hold on
            elseif MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}) >=
slope_plus4_0_split_5

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*y'); hold on
            elseif isnan(MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}))

plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*k'); hold on
            else
```

```
                disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
            end
        else
            plot(n,MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname}),'*b');
hold on
        end
        determine_mean(n) =
MeanSlopes.(AreaNames_S_N{1})(n).(characteristic_name_slope{chname});
    end

    plot([n n],[0 8000],'--k')
    plot([1 n],[nanmean(determine_mean) nanmean(determine_mean)],'g','LineWidth',2)
    plot([1 n],[(nanmean(determine_mean)+(2*nanstd(determine_mean)))
(nanmean(determine_mean)...
        +(2*nanstd(determine_mean)))],'k','LineWidth',0.3)
    plot([1 n],[(nanmean(determine_mean)-(2*nanstd(determine_mean)))
(nanmean(determine_mean)...
        -(2*nanstd(determine_mean)))],'k','LineWidth',0.3)

    clear determine_mean

    for k = 2:length(AreaNames_S_N)
        if isempty(MeanSlopes.(AreaNames_S_N{k})(1).transect)
            disp(['coastal area: ',num2str(k),' is empty (no data)'])

            for m = 1:length(MeanSlopes.(AreaNames_S_N{k}))
                if chname == 5 || chname == 10
                elseif chname == 4 || chname == 9
                    if MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_min4_min8_split_1
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*r'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_1 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min4_min8_split_2
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*m'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_2 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min4_min8_split_3
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*b'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_3 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min4_min8_split_4
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*c'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min4_min8_split_4
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*g'); hold on
                    elseif
isnan(MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}))
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*k'); hold on
                    else
                        disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
                    end
                elseif chname == 3 || chname == 8
                    if MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_min2_min4_split_1
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*r'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_1 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min2_min4_split_2
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*m'); hold on
```

```
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_2 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min2_min4_split_3
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*b'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_3 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min2_min4_split_4
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*c'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_4 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) < slope_min2_min4_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*g'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_min2_min4_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*y'); hold on
                    elseif
isnan(MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}))
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*k'); hold on
                    else
                        disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
                    end
                elseif chname == 2 || chname == 7
                    if MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_1
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*r'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_1 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_2
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*m'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_2 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_3
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*b'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_3 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_4
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*c'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_4 &&
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus2_min2_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*g'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >=
slope_plus2_min2_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*y'); hold on
                    elseif
isnan(MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}))
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*k'); hold on
                    else
                        disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
                    end
```

```
                elseif chname == 1 || chname == 6
                    if MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus4_0_split_1
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*r'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >= slope_plus4_0_split_1
&& MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus4_0_split_2
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*m'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >= slope_plus4_0_split_2
&& MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus4_0_split_3
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*b'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >= slope_plus4_0_split_3
&& MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus4_0_split_4
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*c'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >= slope_plus4_0_split_4
&& MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) <
slope_plus4_0_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*g'); hold on
                    elseif
MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}) >= slope_plus4_0_split_5
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*y'); hold on
                    elseif
isnan(MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}))
                        plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*k'); hold on
                    else
                        disp(['Did not plot some points! ',num2str(chname),' ',num2str(k),'
',num2str(n)])
                    end
                else
                    plot(m+n+((k-
1)*50),MeanSlopes.(AreaNames_S_N{k})(m).(characteristic_name_slope{chname}),'*','color',[0+(k/
25) 0 1-(k/25)]); hold on
                end
            end

            plot([n+m+((k-1)*50) n+m+((k-1)*50)],[0 8000],'--k')

        else
            if k == 2 || k == 3 || k == 4 || k == 5 || k == 6 || k == 7 || k == 8 || k == 9
...
                    || k == 10 || k == 19 || k == 21 || k == 22 || k == 23 || k == 24 || k ==
25
                transects_character = NaN(length(MeanSlopes.(AreaNames_S_N{k})),2);
                for l = 1:length(MeanSlopes.(AreaNames_S_N{k}))
                    transects_character(l,1) = MeanSlopes.(AreaNames_S_N{k})(l).transect;
                    transects_character(l,2) =
MeanSlopes.(AreaNames_S_N{k})(l).(characteristic_name_slope{chname});
                end

                order_transects_character = sortrows(transects_character,-1); % descend

                [order_transect_character_nohead, order_transect_character_head] =
Split_islandheads(order_transects_character,(AreaNames_S_N{k}));

                for m = 1:length(order_transect_character_nohead)
                    if chname == 5 || chname == 10
                    elseif chname == 4 || chname == 9
                        if order_transect_character_nohead(m,2) < slope_min4_min8_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_1
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_2
```

```
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_2
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_3
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 3 || chname == 8
                        if order_transect_character_nohead(m,2) < slope_min2_min4_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_1
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_2
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_3
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_4
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 2 || chname == 7
                        if order_transect_character_nohead(m,2) < slope_plus2_min2_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_1 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_2 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_3 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_4 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
```

```
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 1 || chname == 6
                        if order_transect_character_nohead(m,2) < slope_plus4_0_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_1
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_2
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_3
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_4
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    else
                        plot(m+n+((k-
1)*50),order_transect_character_nohead(m,2),'*','color',[0+(k/25) 0 1-(k/25)]); hold on
                    end

                end

                plot([n+m+((k-1)*50) n+m+((k-1)*50)],[0 8000],'--k')
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[nanmean(order_transect_character_nohead(:,2))
nanmean(order_transect_character_nohead(:,2))],'g','LineWidth',2)
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead(:,2))...
                    +(2*nanstd(order_transect_character_nohead(:,2))))
(nanmean(order_transect_character_nohead(:,2))...
                    +(2*nanstd(order_transect_character_nohead(:,2))))],'k','LineWidth',0.3)
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead(:,2))...
                    -(2*nanstd(order_transect_character_nohead(:,2))))
(nanmean(order_transect_character_nohead(:,2))...
                    -(2*nanstd(order_transect_character_nohead(:,2))))],'k','LineWidth',0.3)

                clear transects_character
                clear order_transects_character
                clear order_transect_character_nohead
                clear order_transect_character_head

            elseif k == 11 || k == 12 || k == 13 || k == 14 || k == 15 || k == 16 || k == 17
...
                    || k == 20
                transects_character = NaN(length(MeanSlopes.(AreaNames_S_N{k})),2);
                for l = 1:length(MeanSlopes.(AreaNames_S_N{k}))
                    transects_character(l,1) = MeanSlopes.(AreaNames_S_N{k})(l).transect;
                    transects_character(l,2) =
MeanSlopes.(AreaNames_S_N{k})(l).(characteristic_name_slope{chname});
                end

                order_transects_character = sortrows(transects_character,1); % ascend

                [order_transect_character_nohead, order_transect_character_head] =
Split_islandheads(order_transects_character,(AreaNames_S_N{k}));
```

```
                for m = 1:length(order_transect_character_nohead)
                    if chname == 5 || chname == 10
                    elseif chname == 4 || chname == 9
                        if order_transect_character_nohead(m,2) < slope_min4_min8_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_1
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_2
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_3
&& order_transect_character_nohead(m,2) < slope_min4_min8_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min4_min8_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 3 || chname == 8
                        if order_transect_character_nohead(m,2) < slope_min2_min4_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_1
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_2
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_3
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_4
&& order_transect_character_nohead(m,2) < slope_min2_min4_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_min2_min4_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 2 || chname == 7
                        if order_transect_character_nohead(m,2) < slope_plus2_min2_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_1 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_2 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_3 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_4 && order_transect_character_nohead(m,2) < slope_plus2_min2_split_5
```

```
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >=
slope_plus2_min2_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    elseif chname == 1 || chname == 6
                        if order_transect_character_nohead(m,2) < slope_plus4_0_split_1
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*r');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_1
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_2
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*m');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_2
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_3
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*b');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_3
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_4
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*c');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_4
&& order_transect_character_nohead(m,2) < slope_plus4_0_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*g');
hold on
                        elseif order_transect_character_nohead(m,2) >= slope_plus4_0_split_5
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*y');
hold on
                        elseif isnan(order_transect_character_nohead(m,2))
                            plot(m+n+((k-1)*50),order_transect_character_nohead(m,2),'*k');
hold on
                        else
                            disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                        end
                    else
                        plot(m+n+((k-
1)*50),order_transect_character_nohead(m,2),'*','color',[0+(k/25) 0 1-(k/25)]); hold on
                    end
                end

                plot([n+m+((k-1)*50) n+m+((k-1)*50)],[0 8000],'--k')
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[nanmean(order_transect_character_nohead(:,2))
nanmean(order_transect_character_nohead(:,2))],'g','LineWidth',2)
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead(:,2))...
                    +(2*nanstd(order_transect_character_nohead(:,2))))
(nanmean(order_transect_character_nohead(:,2))...
                    +(2*nanstd(order_transect_character_nohead(:,2))))],'k','LineWidth',0.3)
                plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead(:,2))...
                    -(2*nanstd(order_transect_character_nohead(:,2))))
(nanmean(order_transect_character_nohead(:,2))...
                    -(2*nanstd(order_transect_character_nohead(:,2))))],'k','LineWidth',0.3)

                clear transects_character
                clear order_transects_character
                clear order_transect_character_nohead
                clear order_transect_character_head

            else
                if k == 18


                    ind_1 = 1;
                    ind_2 = 1;
```

```
                    for kl = 1:length(MeanSlopes.(AreaNames_S_N{k}))
                        if MeanSlopes.(AreaNames_S_N{k})(kl).transect > 50000 && ...
                                MeanSlopes.(AreaNames_S_N{k})(kl).transect < 80000
                            transects_character_sylt_1(ind_1,1) = 
MeanSlopes.(AreaNames_S_N{k})(kl).transect;
                            transects_character_sylt_1(ind_1,2) = 
MeanSlopes.(AreaNames_S_N{k})(kl).(characteristic_name_slope{chname});

                            ind_1 = ind_1 + 1;
                        elseif MeanSlopes.(AreaNames_S_N{k})(kl).transect < 25000
                            transects_character_sylt_2(ind_2,1) = 
MeanSlopes.(AreaNames_S_N{k})(kl).transect;
                            transects_character_sylt_2(ind_2,2) = 
MeanSlopes.(AreaNames_S_N{k})(kl).(characteristic_name_slope{chname});

                            ind_2 = ind_2 + 1;
                        else
                        end
                    end

                    order_transects_character_sylt_1 = sortrows(transects_character_sylt_1,-
1); % descend
                    order_transects_character_sylt_2 = sortrows(transects_character_sylt_2,1); 
% ascend

                    order_transects_character_sylt = [transects_character_sylt_1; 
transects_character_sylt_2];

                    [order_transect_character_nohead_sylt, order_transect_character_head_sylt] 
= Split_islandheads(order_transects_character_sylt,(AreaNames_S_N{k}));

                    for m = 1:length(order_transect_character_nohead_sylt)
                        if chname == 5 || chname == 10
                        elseif chname == 4 || chname == 9
                            if order_transect_character_nohead_sylt(m,2) < 
slope_min4_min8_split_1
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*r'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min4_min8_split_1 && order_transect_character_nohead_sylt(m,2) < slope_min4_min8_split_2
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*m'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min4_min8_split_2 && order_transect_character_nohead_sylt(m,2) < slope_min4_min8_split_3
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*b'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min4_min8_split_3 && order_transect_character_nohead_sylt(m,2) < slope_min4_min8_split_4
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*c'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min4_min8_split_4
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*g'); hold on
                            elseif isnan(order_transect_character_nohead_sylt(m,2))
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*k'); hold on
                            else
                                disp(['Did not plot some points! ',num2str(chname),' 
',num2str(k),' ',num2str(n)])
                            end
                        elseif chname == 3 || chname == 8
                            if order_transect_character_nohead_sylt(m,2) < 
slope_min2_min4_split_1
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*r'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min2_min4_split_1 && order_transect_character_nohead_sylt(m,2) < slope_min2_min4_split_2
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*m'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >= 
slope_min2_min4_split_2 && order_transect_character_nohead_sylt(m,2) < slope_min2_min4_split_3
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*b'); hold on
```

```
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_min2_min4_split_3 && order_transect_character_nohead_sylt(m,2) < slope_min2_min4_split_4
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*c'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_min2_min4_split_4 && order_transect_character_nohead_sylt(m,2) < slope_min2_min4_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*g'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_min2_min4_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*y'); hold on
                            elseif isnan(order_transect_character_nohead_sylt(m,2))
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*k'); hold on
                            else
                                disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                            end
                        elseif chname == 2 || chname == 7
                            if order_transect_character_nohead_sylt(m,2) <
slope_plus2_min2_split_1
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*r'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus2_min2_split_1 && order_transect_character_nohead_sylt(m,2) <
slope_plus2_min2_split_2
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*m'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus2_min2_split_2 && order_transect_character_nohead_sylt(m,2) <
slope_plus2_min2_split_3
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*b'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus2_min2_split_3 && order_transect_character_nohead_sylt(m,2) <
slope_plus2_min2_split_4
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*c'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus2_min2_split_4 && order_transect_character_nohead_sylt(m,2) <
slope_plus2_min2_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*g'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus2_min2_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*y'); hold on
                            elseif isnan(order_transect_character_nohead_sylt(m,2))
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*k'); hold on
                            else
                                disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                            end
                        elseif chname == 1 || chname == 6
                            if order_transect_character_nohead_sylt(m,2) <
slope_plus4_0_split_1
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*r'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus4_0_split_1 && order_transect_character_nohead_sylt(m,2) < slope_plus4_0_split_2
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*m'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus4_0_split_2 && order_transect_character_nohead_sylt(m,2) < slope_plus4_0_split_3
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*b'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus4_0_split_3 && order_transect_character_nohead_sylt(m,2) < slope_plus4_0_split_4
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*c'); hold on
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus4_0_split_4 && order_transect_character_nohead_sylt(m,2) < slope_plus4_0_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*g'); hold on
```

```
                            elseif order_transect_character_nohead_sylt(m,2) >=
slope_plus4_0_split_5
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*y'); hold on
                            elseif isnan(order_transect_character_nohead_sylt(m,2))
                                plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*k'); hold on
                            else
                                disp(['Did not plot some points! ',num2str(chname),'
',num2str(k),' ',num2str(n)])
                            end
                        else
                            plot(m+n+((k-
1)*50),order_transect_character_nohead_sylt(m,2),'*','color',[0+(k/25) 0 1-(k/25)]); hold on
                        end
                    end


                    plot([n+m+((k-1)*50) n+m+((k-1)*50)],[0 8000],'--k')
                    plot([n+((k-1)*50) n+m+((k-
1)*50)],[nanmean(order_transect_character_nohead_sylt(:,2))
nanmean(order_transect_character_nohead_sylt(:,2))],'g','LineWidth',2)
                    plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead_sylt(:,2))...
                        +(2*nanstd(order_transect_character_nohead_sylt(:,2))))
(nanmean(order_transect_character_nohead_sylt(:,2))...

+(2*nanstd(order_transect_character_nohead_sylt(:,2))))],'k','LineWidth',0.3)
                    plot([n+((k-1)*50) n+m+((k-
1)*50)],[(nanmean(order_transect_character_nohead_sylt(:,2))...
                        -(2*nanstd(order_transect_character_nohead_sylt(:,2))))
(nanmean(order_transect_character_nohead_sylt(:,2))...
                        -
(2*nanstd(order_transect_character_nohead_sylt(:,2))))],'k','LineWidth',0.3)

                else
                    error('The input number of coastal areas larger than can be handled (>
25)')
                end
            end

        end
        n = n + length(MeanSlopes.(AreaNames_S_N{k}));
    end


    if chname == 1 || chname == 6
        ylabel({'+4 & 0';' '})
        ylim([0 150])
        set(gca, 'xtick',[])
    elseif chname == 2 || chname == 7
        ylabel({'+2 & -2';' '})
        ylim([0 150])
        set(gca, 'xtick',[])
    elseif chname == 3 || chname == 8
        ylabel({'-2 & -4';' '})
        ylim([0 200])
        set(gca, 'xtick',[])
    elseif chname == 4 || chname == 9
        disp('last subplot, y-label is inserted later')
        %ylabel({'-4 & -8';' '})
        ylim([0 330])
        set(gca, 'xtick',[])
    elseif chname == 5 || chname == 10
    else
        disp('No y-label')
    end


end
%

%
figure(1); hold on
set(gcf,'Name', 'Mean slope: for a selection of transects')
```

```
xlabel('Transect order from south to north, without island heads')
ylabel({'-4 & -8';'Slope (1/m)'})


figure(2); hold on
set(gcf,'Name', 'Mean slope between 2006-2016: for a selection of transects')
xlabel('Transect order from south to north, without island heads')
ylabel({'-4 & -8';'Slope (1/m)'})


toc
```

3.7 Functions

Function 1: *GET_X_Y.m*

```
function [ x, y ] = GET_X_Y(split_line, x, y)
%GET_X_Y Extracting x and y values from the jarkus files
%   Getting the x and y values from the jarkus structure and locating them
%   in new x and y arrays

len = length(split_line);

for i = 1:(len/2)
    split_line(i*2) = {split_line{i*2}(1:end-1)};

    X(i,1) = str2double(split_line{i*2-1});

    Y(i,1) = str2double(split_line{i*2})/100;
end

x(length(x)+1:length(x)+length(X),1) = X;

y(length(y)+1:length(y)+length(Y),1) = Y;

end
```

Function 2: *GetStructIndex.m*

```
function [index] = GetStructIndex(Struct, Year, Transect_Num)
%GetStructIndex find the field index of a data structure based on year and
%transect number
%   Detailed explanation:
%   The jarkus data is saved in a structure. Each transect (profile) has
%   a measurement per year. So for each combination of year + transect
%   number there is one measurement.
%   Struct = the data structure
%   Year = the year of the measurement which you want to extract
%   Transect_Num = the number of the trasect/profile of the desired
%   measurement
%   if either the Year or the Transect_Num is left empty than all the index
%   numbers belonging to the other one are returned

if isempty(Struct)
    error('ERROR: No data structure given');
end

index = [];
```

```
if isempty(Year)<1 && isempty(Transect_Num)<1
    for i = 1:length(Struct)
        if Struct(i).year == Year && Struct(i).transect == Transect_Num
            index = i;
        end
    end
    %disp('The index of the measurement at the input transect during input year is returned');
elseif isempty(Year) && isempty(Transect_Num)<1
    for i = 1:length(Struct)
        if Struct(i).transect == Transect_Num
            index = [index i];
        end
    end
    %disp('The indices of all measurements at the input transect are returned');
elseif isempty(Year)<1 && isempty(Transect_Num)
    for i = 1:length(Struct)
        if Struct(i).year == Year
            index = [index i];
        end
    end
    %disp('The indices of all transect measurements during input year are returned');
else
    error('ERROR: Both input Year and Transect_Num are empty. At least one should be given');

end
```

Function 3: *Split_islandheads.m*

```
function [characteristic_without_island_head, characteristic_island_head] =
Split_islandhead(characteristic, name_coastal_area)
%Split_islandhead Splitting the data from island heads from data of straight
%coastal parts
%   Detailed explanation goes here
% input:
% characteristic = a matrix with two collumns, 1: transect number, 2: the
% characteristic
% name_coastal_area = name of the coastal area which is being analysed,
% this name has to be the same as the name used in the if statment of this
% function (it is the same as the initial structure file names (*.mat),
% where the data was stored from the jarkus files.
%
% Output:
% characteristic_without_island_head = the same matrix as input but only containing
% the transects which are on the straight parts of the coast, so without
% the island heads and some other special cases
% characteristic_island_head = the same matrix as the input but only
% containing the transects of the island head and some other special cases.

ind_head = 1;
ind_nohead = 1;

characteristic_without_island_head = [];
characteristic_island_head = [];

if length(name_coastal_area) == length('Middelkerke_detail_320000newnum') & name_coastal_area
== 'Middelkerke_detail_320000newnum'
elseif length(name_coastal_area) == length('zws_vlaanderen_31000017') & name_coastal_area ==
'zws_vlaanderen_31000017'
elseif length(name_coastal_area) == length('walcheren_31000016') & name_coastal_area ==
'walcheren_31000016'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) < 3526
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
```

```
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('nbeveland_31000015') & name_coastal_area ==
'nbeveland_31000015'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 100
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('schouwen_31000013') & name_coastal_area ==
'schouwen_31000013'
elseif length(name_coastal_area) == length('goeree_31000012') & name_coastal_area ==
'goeree_31000012'
elseif length(name_coastal_area) == length('voorne_31000011') & name_coastal_area ==
'voorne_31000011'
elseif length(name_coastal_area) == length('delf_31000009') & name_coastal_area ==
'delf_31000009'
elseif length(name_coastal_area) == length('rijnland_31000008') & name_coastal_area ==
'rijnland_31000008'
elseif length(name_coastal_area) == length('nh_31000007') & name_coastal_area == 'nh_31000007'
elseif length(name_coastal_area) == length('texel_31000006') & name_coastal_area ==
'texel_31000006'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 860 && characteristic(oi,1) < 2937
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('vlieland_31000005') & name_coastal_area ==
'vlieland_31000005'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 4060 && characteristic(oi,1) < 5367
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('terschelling_31000004') & name_coastal_area ==
'terschelling_31000004'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 540 && characteristic(oi,1) < 2660
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
```

```
        end
    end
elseif length(name_coastal_area) == length('ameland_31000003') & name_coastal_area ==
'ameland_31000003'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 440 && characteristic(oi,1) < 2160
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('schier_31000002') & name_coastal_area ==
'schier_31000002'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 520 && characteristic(oi,1) < 1440
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('Baltrum_data_49260040') & name_coastal_area ==
'Baltrum_data_49260040'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 80 || characteristic(oi,1) < 70
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('Langeoog_data_49260050') & name_coastal_area ==
'Langeoog_data_49260050'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 35 && characteristic(oi,1) < 80
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('All_Sylt_49250107') & name_coastal_area ==
'All_Sylt_49250107'
    for oi = 1:length(characteristic)
        if (characteristic(oi,1) > 50000  && characteristic(oi,1) < 67387) ||
(characteristic(oi,1) > 0  && characteristic(oi,1) < 16462)
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);
```

```
            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('Vestkyst_Vadehavsoer2_45000001') &
name_coastal_area == 'Vestkyst_Vadehavsoer2_45000001'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 6270 && characteristic(oi,1) < 6450
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
elseif length(name_coastal_area) == length('Holmsland_data_450000027') & name_coastal_area ==
'Holmsland_data_450000027'
elseif length(name_coastal_area) == length('Vestkyst_Midtjylland_45000002') &
name_coastal_area == 'Vestkyst_Midtjylland_45000002'
elseif length(name_coastal_area) == length('Vestkyst_Agger_45000003') & name_coastal_area ==
'Vestkyst_Agger_45000003'
elseif length(name_coastal_area) == length('Vestkyst_NationalparkThy_45000004') &
name_coastal_area == 'Vestkyst_NationalparkThy_45000004'
elseif length(name_coastal_area) == length('Vestkyst_VigsoJammerbugten_45000005') &
name_coastal_area == 'Vestkyst_VigsoJammerbugten_45000005'
elseif length(name_coastal_area) == length('Vestkyst_TannisBugt_45000006') & name_coastal_area
== 'Vestkyst_TannisBugt_45000006'
    for oi = 1:length(characteristic)
        if characteristic(oi,1) > 1510 && characteristic(oi,1) < 1050
            characteristic_without_island_head(ind_nohead, 1) = characteristic(oi,1);
            characteristic_without_island_head(ind_nohead, 2) = characteristic(oi,2);

            ind_nohead = ind_nohead + 1;
        else
            characteristic_island_head(ind_head, 1) = characteristic(oi,1);
            characteristic_island_head(ind_head, 2) = characteristic(oi,2);

            ind_head = ind_head + 1;
        end
    end
else
    error('Input not correct')
end

if isempty(characteristic_without_island_head)
    characteristic_without_island_head = characteristic;
end


end
```