UNIVERSIDADE DO ALGARVE

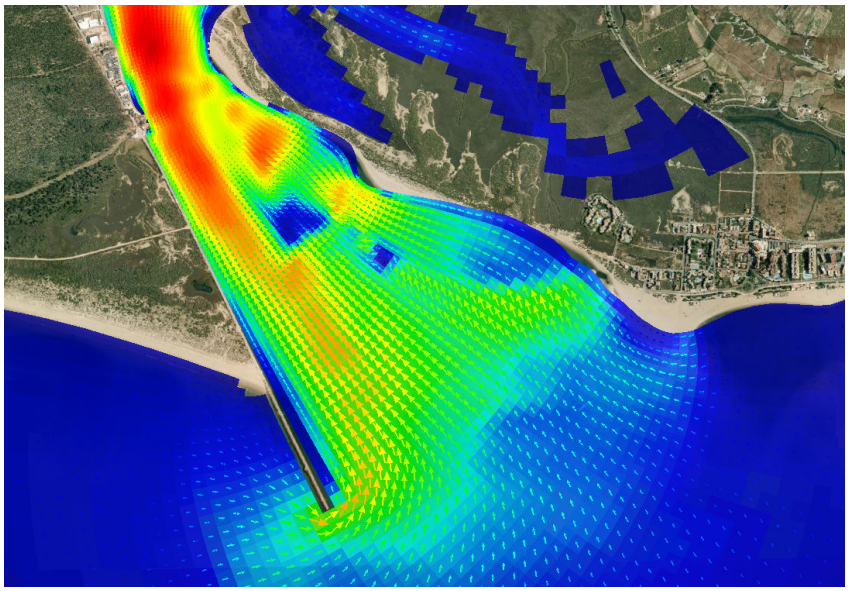
Faculdade de Ciências e Tecnologia e Instituto Superior de Engenharia

GIS as a tool to aid pre- and post-processing of hydrodynamic models. Application to the Guadiana Estuary

Nadiia Basos

Master thesis in Geomatics

(Specialization in Analysis of Environmental Systems)



2013

UNIVERSIDADE DO ALGARVE

Faculdade de Ciências e Tecnologia e Instituto Superior de Engenharia

GIS as a tool to aid pre- and post-processing of hydrodynamic models. Application to the Guadiana Estuary

(GIS como ferramenta de ajuda pré- e pós-processamento de modelos hidrodinâmicos. Aplicação ao Estuário do Guadiana)

Nadiia Basos

Mestrado em Geomática

(Especialização em Análise de Sistemas Ambientais)

Dissertação orientada por Flávio Martins e José I. Rodrigues

2013

Declaração de autoria de trabalho

"GIS as a tool to aid pre- and post-processing of hydrodynamic models. Application to the Guadiana Estuary."

Declaro ser a autora deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

_____

Acknowledgements

I would like to thank all the people who contributed to my work and made this thesis possible.

First of all, I would like to thank warmly my supervisors, professors Flávio Martins and José I. Rodrigues, for encouraging, personal guidance, valuable advices and sharing their wide knowledge. I could not wish better advisors for my research!

Many thanks to Erwan Garel, CIMA, University of the Algarve, for providing the data and for comments and suggestions which were very helpful for this study.

I am thankful to Juan Morales, University of Huelva, and Joaquim Luis, University of the Algarve, for providing data for this research.

I wish to thank Leonid Sokoletsky, East China Normal University, for valuable comments and reference material.

Also I would like to thank all my professors from the University of the Algarve, especially Fernando Martins, Oscar Ferreira, José Jacob, Cristina Veiga-Pires, Carla Rebelo and Helena Fernandez, for giving me deep knowledge in the fields of Geomatics and marine sciences.

I wish to acknowledge the academic and technical support of the University of the Algarve for providing the necessary computer facilities for this research. Thanks to my colleagues in the Hidrotec laboratory, especially to João Janeiro for help on resolving computer problems.

I also thank my first scientific advisor Yuriy Vergeles, who helped me to take this road.

And I sincerely thank my parents, Yuriy Basos and Galyna Basos, for their love, patience and support, as well for professional advices in computer science. To them I dedicate this thesis.

Finally, but most importantly, thanks to Jehovah God and Jesus Christ for giving me strength. “For all things I have the strength by virtue of him who imparts power to me” (Philippians 4:13).

This research was done in the frame of Erasmus Mundus ECW Lot 7 BMU scholarship.

Abstract

The main objective of this work is to develop GIS based techniques to aid the setup of hydrodynamic models and to improve model results. This dissertation analyzes well-known GIS methods and proposes new ones to prepare and process estuarine model data.

Estuarine hydrodynamic models require management of large quantities of georeferenced information. A Geographic Information System (GIS) can help to store, manage, analyze and display all these data during the input and the output phases.

The hydrodynamics of the Guadiana Estuary was simulated using a 2D configuration in MOHID Water Modelling System, based on a boundary fitted curvilinear grid.

GIS tools were used to pre-process the model grid and bathymetry. The water domain was extracted from the orthophoto using unsupervised classification of the image based on principal component analysis of the spectral bands. The large amount of bathymetric measurement points was decreased using a spatial regular pattern (hexagons). The missing bathymetry data in some very shallow parts of the estuary were estimated from the orthophoto using correlation between existing data and spectral band values. The bathymetry data were interpolated into curvilinear grids by several different methods, including an advanced method using river straightening (transformation to the along-channel coordinate system). The finite volume model MOHID was used to test these methods and evaluate the associated improvements.

The model results were in good agreement with the observations under well-mixed conditions. Including the bathymetry estimated from orthophoto improved the accuracy of the simulations; and using advanced interpolation methods improved the results even more. The bathymetry interpolation in the channel-oriented coordinates significantly improved the direction of the water current.

Good quality of the spatial input data was critical for obtaining good model results. The use of GIS tools to produce model inputs proved to be a valuable aid to coastal hydrodynamic modelling increasing substantially the model accuracy.

This dissertation is useful in both theoretical and practical fields of science: in theoretical, by analyzing and developing GIS methods to prepare and process estuarine hydrodynamic model data, in practical, by creating an improved model setup for the Guadiana Estuary, using more accurate bathymetric data and a new curvilinear grid, thus obtaining more realistic results.

Keywords: GIS, bathymetry, interpolation, hydrodynamic model, Guadiana Estuary

Resumo

O objetivo principal deste trabalho é desenvolver técnicas baseadas em sistemas SIG de modo a melhorar os resultados da modelação hidrodinâmica. Esta dissertação analisa métodos SIG conhecidos e propõe novos métodos para preparar e processar os dados do modelo estuarino.

Os Estuários são sistemas ambientais especialmente vulneráveis às atividades humanas, por esse motivo a compreensão da hidrodinâmica do estuário é muito importante para a sua gestão sustentável. A utilização de modelos hidrodinâmicos estuarinos requer o acesso a elevada quantidade de dados georreferenciado. Os sistemas de informação geográfica permitem organizar, armazenar, aceder, analisar e visualizar esse tipo de dados. Os SIG permitem também que os modeladores integrem dados geográficos e informações de muitas fontes diferentes. Um benefício adicional dos SIG é a facultarem um grande número de ferramentas avançadas para analisar e modelar as relações entre as camadas individuais de dados, por exemplo, álgebra de mapas, sobreposições, animações, análises estatísticas, entre outras.

Muitas tentativas para integrar SIG e modelos hidrodinâmicos são descritas na literatura. Os SIG são considerados como ferramentas muito úteis para ajudar a discretização espacial, o processamento de dados de entrada e resultados de visualização por muitos autores. No entanto, a maioria dos trabalhos existentes concentram-se no desenvolvimento de aplicações SIG simples para visualização dos dados produzidos pelo modelo e para facilitar a operação do modelo através de uma interface amigável. Não foram identificados trabalhos que comprovem o benefício do uso de ferramentas de SIG para a precisão do modelo através da sua validação com medições.

Os modelos numéricos simulam o transporte das propriedadesda água em movimento, através da discretização das equações diferenciais de fluxo. A discretização espacial requer uma malha computacional que deve descrever a linha costeira de forma precisa. A hidrodinâmica do estuário do Guadiana foi simulada pela primeira vez neste trabalho utilizando uma configuração 2D no MOHID Water Modelling System, com base numa malha curvilínea adaptada à fronteira do domínio. O Módulo MOHID Water simula o fluxo de água e propriedades em corpos de água superficiais resolvendo as equações de águas rasas, pelo método de volumes finitos. O pacote inclui o módulo MOHID GIS, que trata dados variáveis no espaço e no tempo usando formatos específicos ou produzidos por módulos numéricos. No entanto o MOHID GIS não possui métodos complexos de tratamento de dados, não permitindo por isso executar todo o processamento de dados. Por essa razão foi necessário utilizar um software SIG mais poderoso.

A tarefa central do trabalho consistiu em utilizar ferramentas SIG para melhorar os resultados

produzidos pelo modelo matemático. Com esse objetivo foi criado um banco de dados GIS usando os softwares MOHID GIS e ESRI ArcGIS. O ArcGIS permitiu efetuar transformações entre os sistemas de coordenadas e o georreferenciamento.

Ferramentas de SIG foram utilizados para pré-processar a malha do modelo e a batimetria. A linha de costa foi extraída de um ortofotomapa por classificação não-supervisionada dos componentes principais da imagem usando o software IDRISI. Em seguida, utilizou-se como polígono de domínio para gerar a malha curvilínea. A grande quantidade de pontos de medição batimétricos foi diminuída usando um padrão espacial regular (hexágonos) usando um script em Python desenvolvido para esta tarefa. As coordenadas médias e valores de pontos de batimetria foram calculados dentro de cada hexágono. Quando inexistentes, os dados de batimetria em algumas partes muito rasas do estuário foram estimados a partir do ortofotomapa através da correlação entre os dados existentes e valores da banda espectral RGB, usando o programa de estatística R e programação em Python.

Os dados de batimetria foram interpolados para a malha curvilínea usando métodos diferentes, incluindo um método avançado baseado no alinhamento do rio. Este método avançado foi realizada utilizando ferramentas ArcGIS Referência Linear para a transformação dos dados de batimetria para o sistema de coordenadas ao longo do rio. O modelo de volumes finites MOHID, foi usado para testar estes métodos e avaliar as melhorias associadas.

O uso de ferramentas de geoprocessamento para produzir dados de entrada para o modelo provou ser uma ajuda valiosa aumentando substancialmente a precisão deste. O modelo melhorado através de métodos avançados baseados em SIG foi calibrado e validado usando medições hidrodinâmicas. Depois de estender o domínio do modelo para montante até ao limite da maré, os resultados do modelo estavam em bom acordo com as medições das estações de calibração, especialmente quando as condições produziam uma coluna de água bem-misturada (pouco fluxo do rio). A validação mostrou que a inclusão dos pontos de litoral e batimetria estimados a partir do ortofotomapa melhorou o resultado e que a utilizaçãode métodos de interpolação avançados melhorou os resultados ainda mais. A interpolação da batimetria nas coordenadas orientadas com o canal melhorou significativamente o sentido da corrente. O RMSE calculado para cada cenário mostrou que o último modelo com batimetria interpolados após alisamento rio produziu, em geral, o melhor resultado.

Esta dissertação é útil tanto no campo teórico como no campo prático da ciência; no teórico pela análise e desenvolvimento de métodos de GIS para preparar e processar dados hidrodinâmicos estuarinos para serem inseridos em modelos, no prático, através da criação de uma configuração melhorada do modelo existente para o estuário do Guadiana, usando

dados batimétricos mais precisos e uma malha curvilínea nova, obtendo assim resultados mais realistas.

Palavras chave: SIG, batimetria, interpolação, modelo hidrodinâmico, Estuário do Guadiana

Table of Contents

Declaração de autoria de trabalho 2

Acknowledgements 3

Abstract 4

Resumo 5

Table of Contents 8

Index of Figures 10

Index of Tables 14

1. Introduction 15

2. Literature review 18

2.1. GIS in hydrodynamic modelling 18

2.2. Numerical flow simulation 25

2.2.1. Modelling 25

2.2.2. The governing equations 26

2.2.3. Discretization 29

2.2.4. Turbulence 32

2.2.5. Model execution 33

2.3. Curvilinear grid 33

2.4. MOHID water modelling system 36

2.4.1. MOHID numerical model 36

2.4.2. MOHID and GIS integration 39

3. The Guadiana Estuary 42

3.1. Physical Characterization of the System 42

3.2. Existing Guadiana models 46

4. Methods 48

4.1. Model inputs 48

4.2. GIS tools and database 49

4.2.1. Data sources 49
4.2.2. Data processing 55
4.2.3. Water polygon extraction from orthophoto 59
4.2.4. Curvilinear grid creation 66
4.2.5. Bathymetry estimation from orthophoto 71
4.2.6. Bathymetry interpolations 79
4.3. Model setup 94
5. Model results and calibration 97
5.1. Calibration 97
5.2. Comparing results from different inputs processed in GIS 106
6. Discussion 110
7. Conclusion 117
References 119
Appendixes 127
Appendix 1. Python script “Creating regular grid of hexagons” 128
Appendix 2. Python script “Decreasing the number of data points” 130
Appendix 3. Python script “Relation between water color and bathymetry” 133
Appendix 4. Python script “Predicting bathymetry” 137
Appendix 5. Python script “RMSE of bathymetry prediction” 141

Index of Figures

Figure 1.1. Model geospatial data 16

Figure 1.2. The workflow 17

Figure 2.1. Modelling methods 26

Figure 2.2. Control Volume and system in a flow (where t is the time) 27

Figure 2.3. Cartesian grids, regular (left) and variable spacing (right) 30

Figure 2.4. Triangular mesh (Pinho et al., 2004) 30

Figure 2.5. Cartesian grid 15×10 cells and curvilinear grid 15×10 cells 34

Figure 2.6. Transformed region 35

Figure 2.7. Grid cell in MOHID (Martins et al., 2001) 37

Figure 3.1. The Guadiana drainage basin 42

Figure 3.2. The entire Guadiana Estuary 43

Figure 3.3. The Guadiana mouth, morphology (Morales, 1997), the blue line shows the submerged jetty, the yellow line is the deep channel 43

Figure 3.4. Water level (m) in the lower estuary during two months 44

Figure 3.5. The Guadiana mouth and the lower estuary, aerial view from south-east 45

Figure 4.1. MOHID hydrodynamic model data 49

Figure 4.2. The orthophotos of the lower estuary: a) IGP 2005, b) IGP 2010, c) Bing, d) Google, e) Yahoo, d) Navteq 51

Figure 4.3. Available bathymetry data 52

Figure 4.4. Flow data stations 53

Figure 4.5. Calibration data locations 54

Figure 4.6. Simpatico 2008-2009 data of velocity of the currents (m/s) 54

Figure 4.7. Comparing pattern types 57

Figure 4.8. Clustering bathymetry (initial data on the left and the result at the right side) 58

Figure 4.9. Joined bathymetry data from all sources 59

Figure 4.10. Spectral characteristics of some Earth's surface types, and Landsat bands 60

Figure 4.11. Landsat 7 composites for the Odeleite and Guadiana mouth, bands 3,2,1

(left), 4,3,2 (center) and 4,5,7 (right) 61
Figure 4.12. The scatterplots of the bands (R vs. G, R vs. B, G vs. B) 62
Figure 4.13. The three principal components, from PC1 (left) to PC3 (right) 63
Figure 4.14. The final PCA-based classification result 64
Figure 4.15. Comparing the best classification result using spectral bands (central) and classification on principal components (right) 65
Figure 4.16. The vectorized water class 65
Figure 4.17. Curvilinear grid structure 66
Figure 4.18. Domain polygon topology 66
Figure 4.19. Domain polygons and grids 68
Figure 4.20. Errors in tributaries 69
Figure 4.21. Tidal flat and ports 69
Figure 4.22. The final grid (the lower estuary) 70
Figure 4.23. The final grid topology 71
Figure 4.24. Relationships between bathymetry and colour intensity 76
Figure 4.25. Distribution of band values 77
Figure 4.26. Distribution of bathymetry data before and after transformation 77
Figure 4.27. Linear regression validation plots 78
Figure 4.28. Estimated bathymetry 79
Figure 4.29. MOHID GIS Triangulation 81
Figure 4.30. Trend analysis. Bathymetry data (yellow) projected on 3D planes 82
Figure 4.31. Variogram for Kriging 4.5*Spherical(200) 83
Figure 4.32. The semivariograms 84
Figure 4.33. Kriging with anisotropy 4.5*Spherical(200,300,80) 84
Figure 4.34. Cross-validation 85
Figure 4.35. Minimum curvature (upper left), Topo to Raster (upper right), Kriging spherical (lower left) and kriging with N-S anisotropy (lower right) 86
Figure 4.36. Isobaths (every 1 m) and data points along ship tracks 87

Figure 4.37. Minimum curvature (top left),Topo to Raster (top right), Kriging isotropic (bottom left) and kriging with N-S anisotropy (bottom right) 88
Figure 4.38. Topo to Raster surface and data points 89
Figure 4.39. Transformation into channel-oriented coordinates 90
Figure 4.40. Bathymetry of the lower estuary: normal (left) and transformed (right) 91
Figure 4.41. Kriging in RM space 91
Figure 4.42. Transformed grid centers and reconstructed in RM space grid 92
Figure 4.43. Kriging in RM space and back-transformed grid centers with mean values 92
Figure 4.44. Surface based on kriging respecting varying anisotropy 93
Figure 4.45. Comparison of the cross-sections 93
Figure 4.46. Selected high flow and low flow periods 94
Figure 4.47. Model domains 96
Figure 5.1. Model result comparing to measured data at the Simpatico station 97
Figure 5.2. Model result comparing to measured data at the Ayamonte station (spring tide) 98
Figure 5.3. Model result with extended domain at Simpatico station 98
Figure 5.4. Model result with extended domain at the Ayamonte station 99
Figure 5.5. Model calibration at the Simpatico station (high river flow) 100
Figure 5.6. The model with the chosen parameters at Simpatico station 101
Figure 5.7. Validation of the model with final parameters in Simpatico (very low river flow) 101
Figure 5.8. Validation of the model with final parameters at Simpatico station in recent time (rather low river flow) 102
Figure 5.9. Validation of the model with final parameters at Ayamonte station 102
Figure 5.10. Velocity components at Simpatico station in recent time 103
Figure 5.11. Validation of the model with final parameters at Odeleite station 103
Figure 5.12. Spring tide, ebb velocities 104
Figure 5.13. Spring tide, flood velocities 105
Figure 5.14. Neap tide, ebb velocities 106

Figure 5.15. Neap tide, flood velocities 106

Figure 5.16. Comparing the models with different inputs at Simpatico station (velocity modulus) 107

Figure 5.17. Comparing the models with different inputs at Simpatico station (velocity components) 108

Figure 5.18. RMSE of the models with different inputs at Simpatico station 109

Figure 6.1. Low tide at the mouth using old sparse data (above) and estimated bathymetry (below) 114

Figure 6.2. Velocities from two model scenarios using isotropic interpolation (above) and anisotropic along the centerline (below) 116

Index of Tables

Table 3.1. State of the Guadiana Estuary under different conditions 45

Table 3.2. Tidal asymmetry in the deep channel (depth-averaged currents) 46

Table 3.3. Circulation patterns in the lower estuary 46

Table 4.1. Transformation of coordinates 56

Table 4.2. PCA result 62

Table 4.3. Tidal harmonics for Guadiana (Vila Real S. A.) 95

1. Introduction

Estuaries are environmental systems especially vulnerable to human activities. Understanding of estuarine hydrodynamics is very important for their sustainable management. But field measurements are expensive and time-consuming and can be done only in several particular locations. Interpolation and extrapolation of measured properties in moving water often cannot give realistic results in distant places, therefore, a model is needed. Environmental system analysis of an estuarine system can include hydrodynamic, sediment transport and water quality modelling of the water. A model is an abstracted representation of a complex, "real world" system (i.e. a simplification of reality). It is useful for simulation, prediction and understanding of a process.

Estuarine hydrodynamic models always require management of large quantities of georeferenced information. The geospatial technologies such as Geographic Information Systems (GIS), Global Positioning System (GPS), Remote Sensing, Geostatistics and Geovisualization (3D view and animation) can provide powerful support for numerical water modelling. In particular, GIS can help to prepare, manage, analyze and display the model geospatial data (Jolma et al., 2008; Ng et al., 2007; Tsanis et al., 1996; Wang et al., 2012; Peng et al., 2010; Green and King, 2003). GIS allows the modellers to integrate geospatial data and information from many disparate sources. An additional benefit of GIS is the availability of the advanced tools for analyzing and modelling the relationships between individual layers of data; for instance, map algebra, overlays, animations, statistical analyses (Green and King, 2003).

Many attempts to integrate GIS and hydrodynamic models are described in the literature (Lichy, 1998; Tsanis and Boyle, 2001, Naoum et al., 2005; Ng et al., 2009; Green and King, 2003). GIS is considered to be a very useful tool to help spatial discretization, input data processing and results visualization by many authors (Naoum, 2005; Merwade et al., 2008; Tsanis et al., 1996; Peng et al., 2010; Wang et al., 2012). However, most of the existent works concentrated on developing simple GIS applications for model data visualization and easy model operation through a user-friendly interface, and none of the previous works has proved the benefit of using GIS tools for model accuracy by validation on real measurements.

The main objective of this dissertation is to analyze and develop GIS-based techniques to improve the setup of hydrodynamic models, namely to increase the accuracy of model results by advanced pre-processing using the geospatial technologies. The Guadiana Estuary, Portugal, is used to demonstrate the concepts which can be generalized to other systems. The model improved by advanced GIS-based methods then is calibrated and validated by real measurements.

Hydrodynamics of the Guadiana Estuary was simulated using a 2D configuration in MOHID Water Modelling System, based on a boundary fitted curvilinear grid. MOHID Water module simulates the flow and water properties in surface water bodies solving the shallow water equations by the Finite Volumes method. The MOHID package includes GIS module which handles spatial and temporal variable data in specific formats required or produced by the numerical modules (figure 1.1).

GIS database was created using MOHID GIS and ESRI ArcGIS 9.3[1] geospatial software. MOHID GIS is not able to perform all necessary data processing, so more powerful GIS software was needed. MOHID requires the data to be in the same coordinate system, but available data existed in many different coordinate systems. ArcGIS helped in transformations between the coordinate systems and georeferencing.

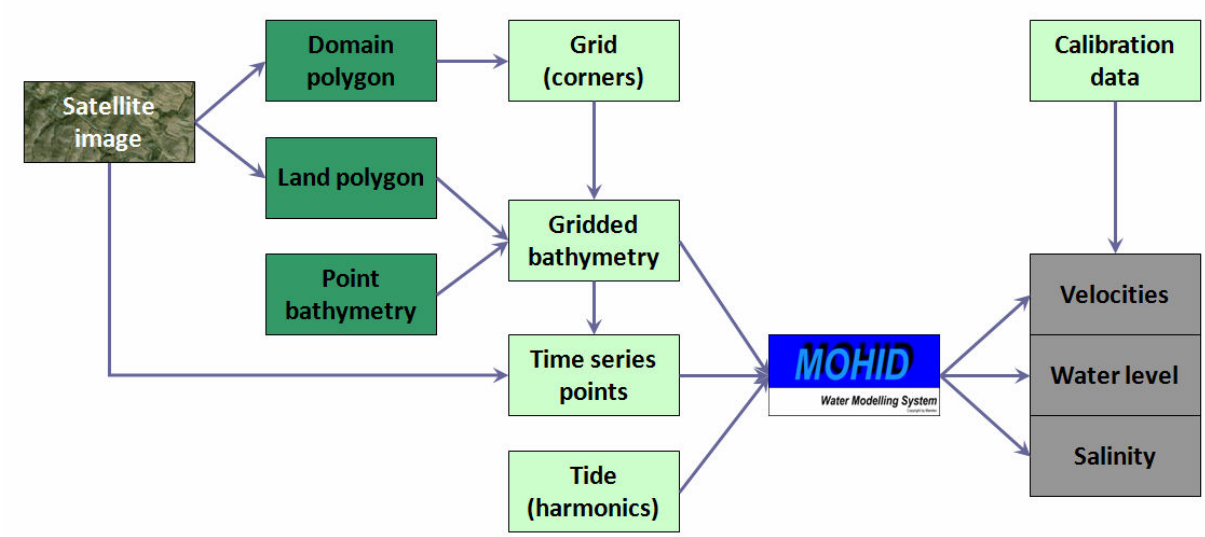


Figure 1.1. Model geospatial data.

The central task was to run simulations and to obtain good realistic results. The design of the computational grids was done using MOHID GIS and was improved using ArcGIS software by fitting cell corners to the shoreline. To produce the model inputs, several spatial operations using GIS are needed: vectorization of the estuary domain polygon, erasing it from surrounding polygon to have a land polygon, preparing bathymetry in the MOHID data format, creating time series locations (figure 1.1). In MOHID GIS only two bathymetry interpolation methods are included: average inside the cell and triangulation. ArcGIS can help in interpolating by advanced methods (IDW, kriging, etc.). MOHID GIS can provide animated visualization of the temporally variable modelling results.

Additional tasks included analyzing existent methods and proposing new ones to prepare and

[1] *ESRI 2009. ArcGIS Desktop: Release 9.3. Redlands, CA. Environmental Systems Research Institute.*

process estuarine model data, as well highlighting advantages and drawbacks of gridding and data pre-processing methods. These methods include data clustering, extraction of information from orthophotos, and other (figure 1.2).

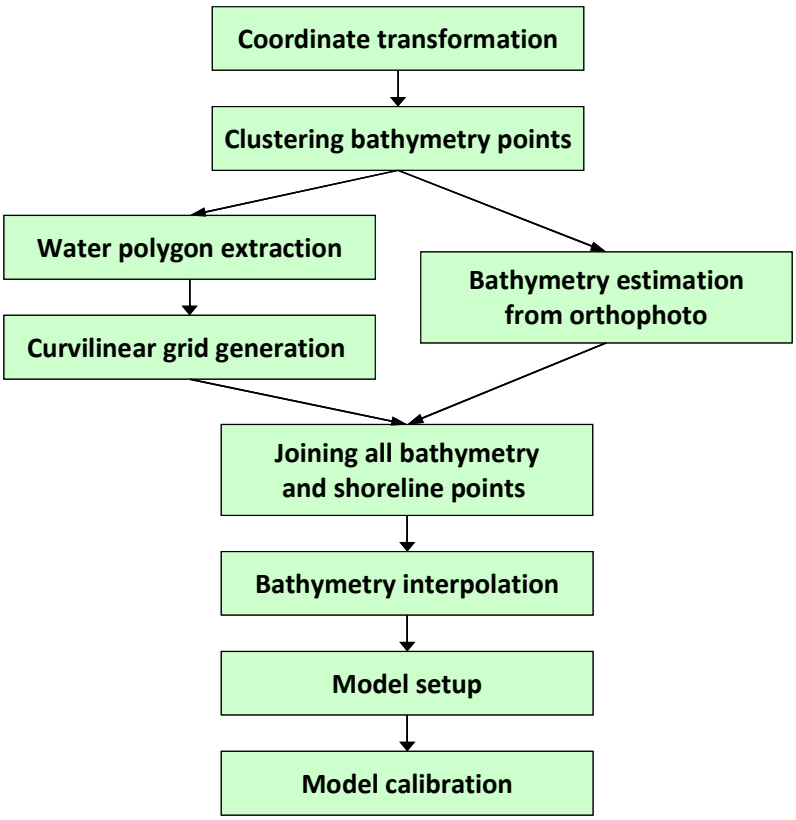


Figure 1.2. The workflow.

The thesis is organized as follows. The second chapter presents the state of the art of the work, containing review of the literature related to numerical modelling techniques and integration GIS and models. The third chapter describes the study area and the models which have already been developed for the estuary. The forth chapter shows the methods of the work, namely the model setup and GIS tools used. The fifth chapter describes the model results, calibration process, and validation of the models with different inputs. The sixth chapter contains discussion and the seventh is conclusion.

Some of the results of this work were presented at the two conferences:

1) Basos N., Martins F. and Rodrigues J. I., 2012. GIS methods to improve numerical model grids and bathymetries. Paper presented at the GeoMundus 2012 Conference on Geosciences, Geoinformation and Environment. November 9-10, Lisbon, Portugal. - 6 p.

2) Basos N., Martins F. and Rodrigues J. I., 2012. Using MOHID GIS to aid hydrodynamic modeling in the Guadiana Estuary. In Proceedings of the 5as Jornadas de Software Aberto para Sistemas de Informação Geográfica – SASIG 5. November 15-17, Faro, Portugal. pp. 15-28.

2. Literature review

This chapter describes efficiency of GIS for managing, analyzing and displaying the model geospatial data, in particular, integration from different sources, spatial discretization, pre-processing of input data and results visualization. The chapter provides an overview of attempts to integrate GIS and hydrodynamic models which are described in literature. It highlights that most of the works were aimed at developing simple GIS applications for data visualization and easy model operation through a user-friendly interface, and none of the previous works proved the benefit of using GIS tools for model accuracy by validation on real measurements.

The next section of the chapter describes the numerical modelling idea and techniques. Numerical models simulate transport of a property in moving water, using discretization of the flow differential equations. The aspects of spatial discretization using curvilinear grids are mentioned. Finally this chapter shows how MOHID Water Modelling System simulates flow and water properties solving the shallow water equations by the Finite Volumes method, and how it already has been integrated with GIS.

2.1. GIS in hydrodynamic modelling

Estuaries are environmental systems especially vulnerable to human activities. Environmental system analysis of an estuarine system can include hydrodynamic, sediment transport, and water quality modelling of surface water. Effective modelling of a coastal system has been found to require both a mathematical component and a geographical component (Green and King, 2003). The geographical component provides understanding of spatial effects by placing processes in a wider environmental context.

Estuarine hydrodynamic models always require management of large amounts of geographically referenced information. Model inputs are georeferenced vector or raster data, and results of the model are spatial (2D or 3D) and temporal distribution of scalar and vector properties, which can be visualized on maps at different time instants. The geospatial tools such as geographic information systems (GIS), global positioning system (GPS), remote sensing, geostatistics and geovisualization (3D view and animation) can provide powerful support for numerical water modelling. In particular, GIS can help to store, manage, analyze and display the model georeferenced data, both the input and the output data (Jolma et al., 2008; Ng et al., 2007; Tsanis et al., 1996; Wang et al., 2012; Peng et al., 2010; Green and King, 2003). GIS is also valuable for pre-processing of input data including editing, transformation, interpolation, and the derivation of parameters; as well as for spatial analyses, visualization, and a computational

environment (Green and King, 2003). GIS allows the modellers to integrate geospatial data and information from many disparate sources, and to visualize the spatial and temporal distribution of the simulated variables, thus to facilitate the modelling process. It is now possible to incorporate remote sensing data, video imagery, tabular data and seafloor mapping in a single data handling and processing environment (Green and King, 2003). GIS provides a tool to interpret the model results in a spatial context.

An additional benefit of GIS lies with the advanced tools for analyzing and modelling the relationships between individual layers of data; for example, map algebra, overlays, animations, statistical analyses (Green and King, 2003).

However, some specific methods and tools required for numerical water modelling are difficult to find in many of the general GIS software packages. For instance, many of them do not handle temporal data very well (Green and King, 2003). Non-Cartesian grids are also a challenge for the most of GIS software.

There are three main types (architectures) of the integration of GIS with models described in the literature. They are low (loose-coupled), medium (tightly-coupled) and high level integration (embedded-coupling). The type signifies the extent to which the model and GIS software are integrated (Green and King, 2003).

In low level integration the two components (model and GIS) are linked together through data transfer between the two independent systems. The GIS is used to get and preprocess data into the form required by the model (inputs to the model program). The model computes the results and returns the files for visualization to the GIS software. For the end-user the advantages are that there is a little knowledge of programming required, operation is quick, and the models are highly portable (Green and King, 2003).

Medium level integration involves a two component architecture allowing a master component to use the capabilities of an agent component. The GIS can be either the master or the agent (GIS calls the model or the model calls the GIS). Usually the model is developed outside the GIS, with its own data structures and exchange mechanisms, and linked to a GIS macro-language. Interaction between the model and the GIS is hidden from the end-user. Data is usually exported from the GIS to the model, and the results are returned for visualization. One benefit is that access to the spatial database is direct (Green and King, 2003).

In high level integration (embedded-coupling), the two elements are fully combined, sharing components such as a database management system and output system. Full integration means that the model is written using the analytical engine of the GIS or a simple GIS is developed and

added to the modelling system. The GIS is used to display results and provide interactive control for the user. The advantages are that transformation of the data to other formats is not needed, data structures do not have to be matched (Green and King, 2003).

All three of these approaches have been successfully used. However, in a case of developing a model within GIS, the cost for duplicating the existing model may be too high. Linking a model to GIS software via the development of a user-friendly interface is a rational approach. Loose or tight coupling is the best approach for an already existing good, long-standing and well-established numerical model (Green and King, 2003). However, nowadays commercial modelling software based on simpler models often rely on a different approach such as developing new GIS software components or modules for their models.

Two decades ago Wright and Bartlett (1999) noted: "many of the techniques involved in coupling marine and coastal models to GIS are still poorly investigated or understood, and thus the benefits and synergy that can arise from bringing these different tools together are rarely seen".

Recently, many attempts to couple GIS and hydrodynamic and pollution transport models are described in the literature (Lichy, 1998; Tsanis and Boyle, 2001, Naoum et al., 2005; Ng et al., 2009; Green and King, 2003). The most of the developers took advantage mainly from visualization capabilities of GIS, but not from advanced GIS tools to increase the model accuracy.

GIS is proved to be a very useful tool to help spatial discretization, input data processing and results visualization (Naoum, 2005; Merwade et al., 2008; Tsanis et al., 1996; Wang et al., 2012); especially when the numerical model is not equipped with any tools for spatial analysis and visualization (Ng et al., 2009), or has its own GUI with limited abilities of only reading and presenting simple geographic information (Peng et al., 2010).

In the year of 1998 a multidirectional and multifunctional gateway between GIS and hydrodynamic models was presented at the Third International Conference on Hydroinformatics by Lichy (1998). The author noticed that an efficient way to analyze and store hydrodynamic model data was to use GIS. Lichy (1998) noted that it was important to get realistic geometry of the modelling domain which was usually had to be corrected manually. System geometries were usually set up from different data sources and this might bring inconsistent datasets. At that time, none of the widely used numerical models offered GIS functionality or a direct interface for data exchange with GIS. Every model had its own data structure, formats and way of representation

and processing, which made comparison of data and results from different models impossible. Pre- and post-processing, interpolation and visualization of data depended on usually undocumented internal model algorithms because numerical modelling systems were concentrated on calculation algorithms. And some numerical models were not able to incorporate data from surveying services. Lichy (1998) developed a gateaway which made it possible to transfer data from many numerical models to and from GRASS GIS (open source GIS software with command line interface). The gateaway was able to convert and to compare model data, and simulation could be run based on GIS data and the results analyzed in the GIS. The gateaway was designed as a library on C++ with methods for reading and writing. It incorporated GIS functionality and also some additional special methods for problems of hydraulics and coastal engineering, in particular b-spline interpolation method. That gateaway firstly presented an effective approach to connect GIS technology and numerical models for efficient data management and analysis.

Then, Liang and Molkenthin (2001) developed a GIS-based hydrodynamic model system and applied in a long estuary dominated by the lunar semi-diurnal tide. The 3D hydrodynamic model WQMAP was applied to simulate flows using constant river depth and curvilinear grid. The main purpose was to develop a visualization tool for analyzing the data. 3D terrain model was created from DTM data and the satellite image, and the model results were visualized in GIS to investigate the salinity intrusion. The results were verbally compared to previous simulations by other scientists, but no any comparing with field measurements was mentioned. The authors predicted GIS to become widely applied in hydroinformatics.

Tsanis and Boyle (2001) developed 2D hydrodynamic and pollutant transport finite difference model IDOR 2D GIS operating within ArcView GIS. It provided data capture and editing, basic pre-processing (Spline and IDW interpolation from contour and point data) and result interpretation through a simple user-friendly GUI.

A GIS pre-processor also operating within ArcView software has been developed to produce bathymetric grids and shorelines as input to a hydrodynamic model IDOR3D to simulate the currents and pollutant transport. This application, in particular, allowed adding the shoreline points to the dataset, and creating bathymetric Cartesian grid by any interpolation method available in the ArcView Spatial Analyst extension (Naoum et al., 2005). The application provided user-friendly interface for preparing the model inputs.

In 2006 Tsanis et al. used GIS to provide water depth information for a two-dimensional hydrodynamic and sediment transport model of a part of a river. Water depth points and shorelines were digitized manually from georeferenced hydrographic map, then used to build

TIN and this TIN (namely linear interpolation) was sampled by the regularly spaced points to get a bathymetric mesh for the model. The study did not show any model validation.

Ng et al. (2009) integrated a GIS with a 3D hydrodynamic, sediment and heavy metal transport numerical model using ArcView software with help of VBA scripting. Simulations could be run using input data created in the GIS and the results could be analyzed through simple and user-friendly GIS interface. The model execution was a separate procedure which did not require GIS capabilities and performed outside of the GIS interface (Ng et al., 2009). The pre-processing through GIS interface included retrieval, manipulation, display, editing and export of model input data (bathymetry, water boundary, initial concentrations, boundary tidal elevation) containing in the GIS database, and also generation of unstructured triangular mesh and manual editing the mesh nodes by the mouse cursor. The post-processing included display the model output data in forms of spatial layers, time series or interpolated profiles. Several spatial interpolation methods (IDW, Spline, Kriging, Natural Neighbor, and Trend) could be applied to create raster layers from point layers just for display in the mapping window, because the input parameters and simulated properties were calculated in the mesh nodes only (Ng et al., 2009). However, it was not clear from the paper, how the mesh nodes retrieved the information (input bathymetry values) from the raw point data when the data point was not coinciding with the node. The functionality of the developed application was illustrated by the case study on the 2000 km^2 estuary in the southern China, but without any calibration or validation by real measurements. Ng et al. (2009) concluded that integration of a GIS and a numerical model enhanced model data management, making pre- and post-processing more convenient and efficient.

Integration of GIS with water quality model of a river was also conducted by Peng et al. (2010) by building a connection platform between the model and GIS through a geodatabase. GIS was used to store maps and DEM of the study area, and to enhance the collection and preparation of the model input data, in particular grid generation and water body boundary obtaining (as shapefiles), attribute data (inflow, outflow, pollutant loads) and water quality data of the monitoring stations. In the post-processing GIS was used for management and presentation of the model output data which also were stored in the geodatabase as spatial and attribute data. The model data could be analyzed with usual GIS functions, such as custom layer and data query, and easily shared with others. The main advantages of this integration were facilitation of data collection, management, visualization and share, via the user-friendly interface. The simulated water quality data (surface temperature and dissolved oxygen) were compared with the measured data at one monitoring station.

GIS was also integrated with a typhoon model and a numerical ocean model for 3D storm surge modelling in a coastal bay, in a form of programmed add-in module using the latest version of ArcGIS 10 (Wang et al., 2012). For the multidimensional data exchange the NetCDF file format was used. GIS user-friendly interface allowed pre-processing of model data (manipulation, editing and display of the bathymetry, coastline, wind data, tide at the boundary, computational mesh) and typhoon data (center, radius, speed and pressure). The intention of post-processing was to display the output data over time through a simple GUI. The meteorological model output was included into the finite volume ocean model via boundary conditions, where the triangular mesh and 10 sigma layers were used for simulation of water elevation and currents. The model was validated comparing water elevation with field measurements at two stations, with the normalized RMSE about 5%.

A GIS-based data management and publication framework was developed for visualization and analysis of hydrodynamic modelling results as desktop- and web-based applications, with Cartesian and unstructured triangular meshes, by Yu et al. (2012).

Actually, in the present time many water modelling software packages already contain pre-processing graphical modules including some GIS tools, such as ability to visualize model spatial data and results, create bathymetric grids and prepare other necessary model inputs. Among them are Delft3D Modelling Suite, SMS Surface-Water Modelling System, MIKE Marine GIS, ASA's HYDROMAP Hydrodynamic Modelling System (and WQMAP), Argus Open Numerical Environments, MOHID Water Modelling System, etc. (Green and King, 2003; Ng, 2006). These modelling systems have simple GIS functionality built into the software package, and thus use the high level integration strategy (embedded-coupling).

Furthermore, there was a study dedicated to improving the hydrodynamic model accuracy with GIS techniques by using advanced interpolation of bathymetry data for river models (Merwade et al, 2008, Merwade et al, 2005). The hydrodynamic model results are greatly affected by the geometric description of the river bathymetry, and the proposed method presented interpolation of river cross-sections in a channel-fitted coordinate system. It was mentioned that rigorous testing of the proposed technique should involve running actual hydrodynamic simulations using river terrain models obtained by different interpolation methods. But for that study only simple cross-validation of the interpolation was performed (namely excluding a few cross-sections from calculations), together with visual evaluation of the obtained 3D terrain models (Merwade et al, 2008). This interpolation method is described in details below in the Chapter 4 among other bathymetry interpolation methods.

However, most of these previous works were concentrated on developing simple GIS

applications for model data visualization for better planning and management (Ng et al., 2007; Yu et al., 2012). Actually, the developers of coupled hydrodynamic GIS applications connecting model and GIS with some pre-processing functions were aimed at providing simple and user-friendly interface to make the model execution more convenient and easy (Tsanis and Boyle, 2001; Ng et al., 2009; Wang et al., 2012). Moreover, usually developers of such applications did not perform calibration and validation of their models using field measurements; and none of these works proved the benefit of using GIS tools for model accuracy by validation of different simulation scenarios. Also, Green and King (2003) noted that GIS/model integration with very friendly GUI might have another problem: it allowed the end-user to operate the model too easily without deep knowledge and understanding of the both GIS and numerical model inner working, thus the user might get "beautiful" but wrong results and to interpret them incorrectly.

Some authors suggested that general GIS was not yet very useful for coastal modelling except for some pre-processing of data. Green and King (2003) guessed that the reason of limited use of GIS was that "support for floating point grids, essential for modelling, is considered unreliable and support for time series and meshes is still generally poor". They also noted that usual GIS were focused on working with maps and data represented by categories (traditional geography) rather than the multivariate fields required for modelling. A traditional 2D static GIS was not sufficient for dynamic and multidimensional processes (Mitasova, 1995). To fully support the use of GIS in coastal research, the GIS should have capabilities for storing, processing and analyzing large spatio-temporal and volumetric data sets; for visualization of dynamic multidimensional data along with the standard GIS data; and for multi-scale data representation and processing (Mitasova, 2000). So Green and King (2003) concluded that full integration of complex models in a GIS generally required extensions to the standard GIS functionality such as support for temporal, 3D and 4D data and meshes for finite element methods. And as mentioned above, developers of modelling systems often also develop simple but specific GIS functionality to support their numerical models by necessary geospatial tools (at least mesh generation and temporal animation).

So, at the present time there is already some experience in programmatic integration of GIS and numerical water models at different levels for convenient model operation and visualization. However, there are not enough studies about how advanced GIS methods can influence model results.

2.2. Numerical flow simulation

Hydrodynamics of the Guadiana Estuary was simulated using the numerical modelling technique. Numerical models simulate transport of a property in moving water, using discretization of the flow differential equations. MOHID Water module simulates flow and water properties in surface water bodies solving the shallow water equations by the Finite Volumes method.

2.2.1. Modelling

A model is an abstracted representation of a complex, "real world" system (i.e. a simplification of reality). It is useful for simulation, prediction and understanding of a process.

Models allow to refine hypotheses and to test sampling plans. Models can help to direct observations in real time, and they can interpolate observations into a more complete picture of processes. Field observations should provide information needed to calibrate and validate models, since models are tools to extrapolate information to a broader temporal or spatial setting (CoOp, 1998). Assumptions are the essential part of model development.

Models may be conceptual, laboratory, mathematical and statistical (CoOp, 1998). In the GIS context, a "model" usually means a combination of layers leading to the generation of the output using map algebra (Green and King, 2003).

Mathematical models are sets of equations determining how a system changes from one state to the next (differential equations) and how a variable depends on the values or state of other variables (state equations) (MacKay, 2006). The solution to these equations, used to describe changes in the system, can be numerical or analytical. Analytical models have a closed form solution, i.e. the solution to the equations can be expressed as an analytic function, i.e. as a finite number of certain "well-known" functions, such as the basic arithmetic operations, extraction of nth roots, exponentiation, logarithms, and trigonometric functions (MacKay, 2006). An analytical expression is ready to calculation.

Numerical methods replace the differential equations with systems of algebraic equations at discrete points, which only then can be solved (figure 2.1). These systems of algebraic equations are usually very large and require computers to solve them.

Computational fluid dynamics (CFD) is a branch of fluid mechanics that deals with numerical simulation of fluid flows. It simulates real flows by the numerical solution of partial differential equations (the governing equations), such as the Navier-Stokes, Reynolds or shallow water equations; or of conservation equations in integral form. The governing equations are extremely complicated and a closed-form analytical solution cannot be obtained without significant

simplifications. The goal of CFD is to find the values of the flow properties in a large number of discrete points in the system using computers which can be programmed to solve algebraic equations very fast (Kuzmin, 2010).

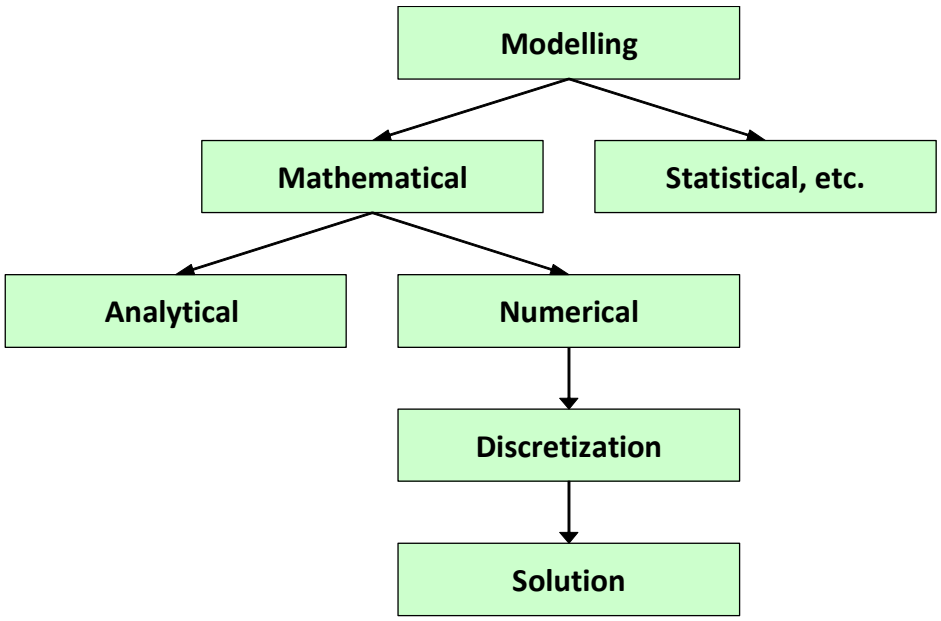


Figure 2.1. Modelling methods.

2.2.2. The governing equations

The governing equations of fluid flow represent the mathematical statements of the conservation laws of physics such as the conservation of mass, momentum, and energy (Versteeg and Malalasekera, 1995; Kuzmin, 2010).

Variation of a generic property is:

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + U\frac{\partial T}{\partial x}$$

where the first term on the right hand side is temporal variation and the second is advective variation (Martins, 2012).

Numerical models simulate transport of a property by advection and diffusion in moving water. This transport can be described by two different approaches, the Lagrangian approach and the Eulerian approach. The Lagrangian approach focuses on a small moving amount of water, a "particle". The Eulerian approach, which is usually used for hydrodynamics, focuses on a fixed portion of space – the control volume (figure 2.2), and the evolution of the properties inside this volume (Martins, 2012; Versteeg and Malalasekera, 1995).

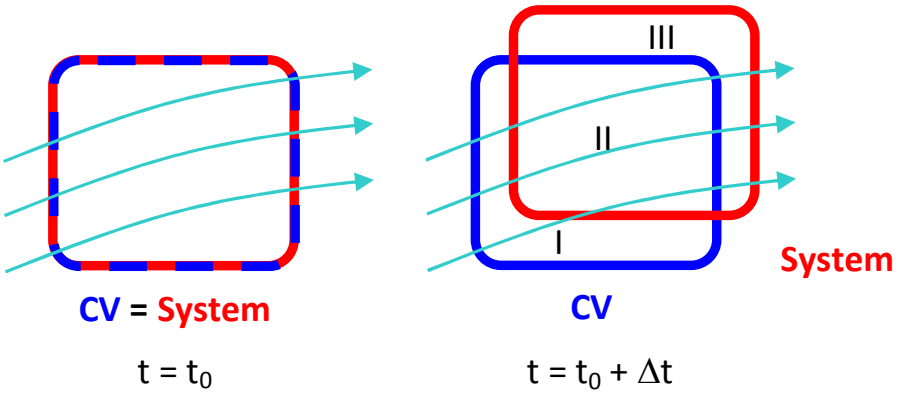


Figure 2.2. Control Volume and system in a flow (where t is the time).

Let's call a generic extensive property N and its intensive (per unit of mass) related $\eta=N/m$, where m is mass. Then the change in N inside the system between time instants t and $t+\Delta t$ is:

$$N_{syst_{t+\Delta t}} - N_{syst_t} = \left(\int_{II} \eta\rho \, dV + \int_{III} \eta\rho \, dV\right)_{t+\Delta t} - \left(\int_{I} \eta\rho \, dV + \int_{II} \eta\rho \, dV\right)_{t}$$

After some mathematical operations and taking the limit $\Delta t \rightarrow 0$:

$$\frac{DN_{syst}}{Dt} = \frac{\partial}{\partial t}\int_{CV} \eta\rho \, dV + \oint_{CV} \eta\rho\vec{v} \cdot d\vec{A}$$

where the operator D means a substantive derivative applied to the system (Martins, 2012). The first term on the right hand side is the temporal derivative and the second term is the advective term. This equation is the basis for the application of the finite volumes method (described later).

The complete equation for the generic property η in a control volume in the integral form is:

$$\frac{\partial}{\partial t}\int_{CV} \eta\rho \, dV + \oint_{CV} \eta\rho\vec{v} \cdot d\vec{A} = Sources - Sinks$$

where ρ is the water density. And this equation in the differential form is:

$$\left.\frac{\partial \rho\eta}{\partial t}\right|_{CV} + \left.\frac{\partial \rho\eta\vec{v}}{\partial \vec{x}}\right|_{CV} = Sources - Sinks$$

The sources and sinks of the property are the reasons for the variations (Martins, 2012).

The equations for the hydrodynamics are obtained applying these equations to mass and momentum.

The mass conservation equation ($N = m$, so $\eta = 1$), also known as continuity equation is:

$$\frac{\partial}{\partial t}\int_{CV} \rho \, dV + \oint_{CV} \rho\vec{v} \cdot d\vec{A} = 0$$

The sources and sinks of mass are zero.

The momentum equations are obtained from the generic equations by $N = \rho\vec{v}$, $\eta = \vec{v}$, leading to the Navier-Stokes equations (Martins, 2012):

$$\frac{\partial}{\partial t}\int_{CV} \rho\vec{v}\, dV + \oint_{CV} \rho\vec{v}\, \vec{v}\cdot d\vec{A} = Sources - Sinks = \sum \vec{F}$$

or:

$$\left.\frac{\partial \rho\vec{v}}{\partial t}\right|_{CV} + \vec{v}\left.\frac{\partial \rho\vec{v}}{\partial \vec{x}}\right|_{CV} = Sources - Sinks$$

The sources and the sinks of momentum are the forces.

For geophysical flows, assuming incompressible fluid and hydrostatic equilibrium, the hydrodynamic equation is (Martins, 2012, Martins, 1999):

$$\frac{\partial \rho\vec{v}}{\partial t} + \vec{v}\frac{\partial \rho\vec{v}}{\partial \vec{x}} = -\frac{1}{\rho_0}\frac{\partial p_{atm}}{\partial \vec{x}} - g\frac{\rho(\eta)}{\rho_0}\frac{\partial \eta}{\partial \vec{x}} - \frac{g}{\rho_0}\int_{x_3}^{\eta}\frac{\partial \rho'}{\partial \vec{x}}dx_3 + \frac{\partial}{\partial \vec{x}}\left(\nu\frac{\partial \vec{v}}{\partial \vec{x}}\right) - f\,\vec{z}\times\vec{v}$$

where $\vec{v}$ is only the horizontal components of the velocity, x_3 and $\vec{z}$ are the vertical coordinate and versor, $f = 2|\Omega|\sin\theta$ is the Coriolis parameter, with Ω as the earth rotation and θ as the Latitude. The terms on the right hand side are the forces applied to the water: the first term is the barotropic force produced by atmospheric pressure gradients, the second term is the barotropic force due to gradients in the water height, the third term is the baroclinic force produced by the vertical integral of horizontal density gradients, the forth term is the diffusive (friction) force, and the fifth term is the Coriolis force (an apparent force that appears due to movement within a rotating coordinate system (the earth)). The relative importance of these terms depends on the problem.

The evolution of the concentration α of a "real" water property (Martins, 2012):

$$\frac{\partial}{\partial t}\int_{CV} \rho\alpha\, dV + \oint_{CV} \rho\alpha\, \vec{v}\cdot d\vec{A} = Sources - Sinks$$

$$\frac{\partial \alpha}{\partial t} + \vec{v}\frac{\partial \alpha}{\partial \vec{x}} = \frac{\partial}{\partial \vec{x}}\left(K\frac{\partial \alpha}{\partial \vec{x}}\right) + Sources - Sinks$$

where α - salt, temperature, nitrate, etc., K – diffusion coefficient.

For a conservative property, the sources and sinks are zero.

2.2.3. Discretization

The above mentioned transport equations don't have the exact analytical solution. There are the two different approaches to approximate their solution (Martins, 2012):

- to simplify the equations by removing the less important terms, reaching a simple equation which can have analytical solution (old approach).

- to maintain all the terms but to simplify them, reaching an equation which can be solved by numerical methods.

Numerical methods transform the space and the time from continuous to discretized medium. Time and space are divided into a finite number of discrete intervals and the variables are evaluated approximately only in these discrete locations (Martins, 2012). The governing differential equations are replaced by a system of linear algebraic equations in these locations (Kuzmin, 2010). The complexity and size of the set of equations depends on the dimensionality of the problem, the number of the locations and the discretization method (Versteeg and Malalasekera, 1995).

Spatial discretization

To discretize the space and create a computational domain, the continuous space is divided into portions producing a computational grid (mesh), composed by cells (or points), inside (or around) which the flow properties are computed (Martins, 2012).

Computational grids can be of many types. Structured grids are the simplest grids used for discretization. In a structured grid any cell in the grid and the neighbouring cells can be identified by a set of indices (*I,J-1; I,J; I,J+1*). Structured grids can be Cartesian regular (where the cell size in each direction is constant), Cartesian with variable spacing (figure 2.3), and curvilinear. Cartesian grids are always orthogonal (the gridlines cross each other at 90° and the cells are always rectangular). Curvilinear grids can be orthogonal or non-orthogonal. Curvilinear grids are described in details below in the chapter 2.3 Curvilinear grid.

Unstructured grids are the most difficult to generate, but they are the most adaptable to discretize complex domains. They can be triangular or quadrilateral (figure 2.4). They are not restricted to one particular cell type, but it is possible to use a mixture of cell shapes (Versteeg and Malalasekera, 2007). In an unstructured grid the number of cell neighbours is variable and the position of a cell in the grid cannot be identified by a set of indices, so it is impossible to relate a cell with its neighbours directly. A connectivity table can be used to solve this problem, but it introduces complexity into the model code and increases the computational time (Martins, 2012). In triangular grid the cells should not be distorted but be close to the optimal equilateral

triangles to minimize computational errors (Pinho et al., 2004; Martins, 2012). The node positions (corners) of each cell have to be identified so the grid generation is complicated. The most widely used generation technique for triangular meshes is the Delaunay triangulation, where the mesh resolution near the boundary may depend on the resolution (number of vertices) of the boundary polygon line (Pinho et al., 2004). In triangular meshes the advective term is difficult to implement and this causes numerical diffusion and conservation problems (Martins, 2012).

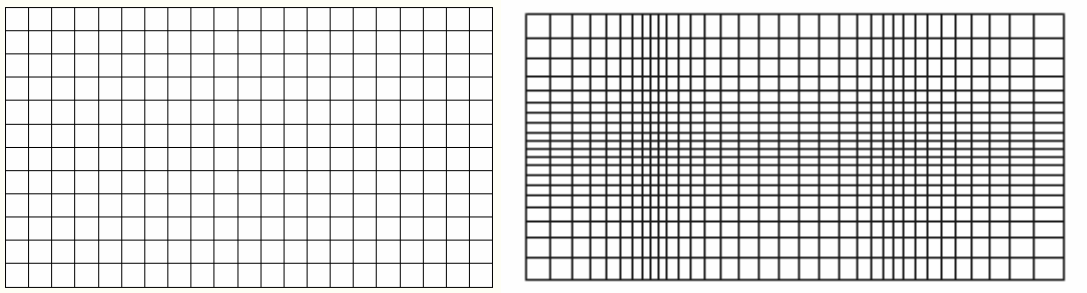


Figure 2.3. Cartesian grids, regular (left) and variable spacing (right)

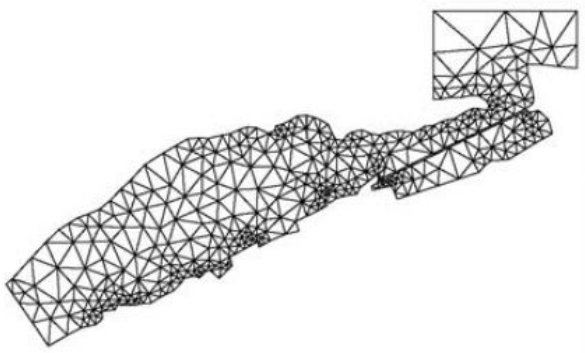


Figure 2.4. Triangular mesh (Pinho et al., 2004).

In 3D models the vertical shape of the grid layers can be different, but in 2D models there is only one vertical layer from the water surface to the bottom (Sigma layer), and all properties are the averages for the water column (depth-integrated) (Martins, 1999).

In 2D and 3D models the cells used to compute velocities and the cells used to compute the properties are not coincident for stability reasons (Arakawa and Lamb, 1977).

Temporal discretization

Time is discretized into time steps (intervals) and the values of the variables are determined only in the specific instants of time (Martins, 2012). Time discretization is related to the implicit or explicit character of the model (Versteeg and Malalasekera, 1995).

Implicit models calculate the variables at each time step using the values of the variables in the neighbour grid cells at the same time step:

$$\alpha^{n+1} = f\left(\alpha^{n+1}\right)$$

where α is a generic variable and n is the time instant. The large system of equations for all grid cells should be solved at each time step, which is slowly.

Explicit models calculate the variables at the current time step using values from the previous time instant:

$$\alpha^{n+1} = f\left(\alpha^{n}\right)$$

This method is much faster but creates some error because the variables are not evaluated at the correct time instant. The time interval between n and $n+1$ cannot be large (it can turn the model instable) and the model needs more iterations (Martins, 2012).

It is possible to use the variables from the two time instants, previous and current, considering the variables in one grid direction as implicit and the variables in the other direction as explicit:

$$\alpha^{n+1} = f\left(\alpha^{n};\alpha^{n+1}\right)$$

The directions are to be changed in the next time step. This method is called Alternate Direction Implicit (ADI) and is better than pure explicit or fully implicit methods (Abbott and Basco, 1989).

Numerical methods (discretization methods)

The Navier-Stokes and transport equations can be solved by different numerical modelling techniques (or methods of discretization of the flow equations). The most used are the Finite Differences, the Finite Volumes and the Finite Element methods.

The Finite Difference method is the simplest method where the derivatives in the differential equation are approximated by an operator which is applied to each grid cell. It can be used only with Cartesian grids. Curvilinear grids can be implemented only by using a transformation of the equations from real curvilinear coordinates to Cartesian coordinates, and it makes the equations very complicated (Versteeg and Malalasekera, 1995). One more drawback is that it is not possible to guarantee conservation of the properties in the cells (Martins, 2012; Martins, 1999).

The mostly used Finite Volumes method solves the transport equations in the integral form in the finite control volumes – the cells of the computational grid (Abbott and Basco, 1989; Martins, 1999). This numerical algorithm consists of the following steps: integration of the governing equations over all the control volumes, conversion of these integral equations into a system of algebraic equations (discretization), solution of these algebraic equations by an iterative method (Versteeg and Malalasekera, 2007). Finite Volume models guarantee

conservation of the properties in the grid cells. Since the equations are applied directly to the cell, the cell can have any shape (Abbott and Basco, 1989). Only geometric parameters of the cells (volume, faces) must be computed and included in the equations, which increases the computational time in a case of a moving grid in free surface models (Martins, 2012, Martins, 1999). The general equations in the integral form applied to a finite volume have been written above. Conservation of a property *a* considering a volume of finite dimensions is (Martins, 1999):

[Variation of a in V] = [Fluxes of a in the faces] + [Sources] – [Sinks]

In Finite Element models the solutions for the differential equations are approximated by a family of functions. It minimizes the error and produces a numerically stable solution. Some of its drawbacks in fluid dynamics are the conservative problems and the difficulty in application for the advective term (Martins, 2012). The advantage is that it is easily compatible with unstructured meshes.

Numerical properties of the modelling methods are

- consistence (difference equation tends to continuous equation),

- convergence (difference solution tends to exact solution, which cannot be proved since the exact solution is unknown),

- stability (a set of difference solutions is inside reasonable boundaries).

According to the Theorem of Lax, a consistent and stable model is convergent (Versteeg and Malalasekera, 1995). In other words, a stable model with dense discretization (small grid cells and small time step) tends to produce correct results.

Stability can be evaluated by the Courant number related to advection:

$$C_r = \frac{v \cdot \Delta t}{\Delta x}$$

where v is velocity, Δt and Δx are time and space steps. For fully explicit method it should not be higher than 1 (Abbott and Basco, 1989), for ADI methods it can be higher (about 4).

2.2.4. Turbulence

At high Reynolds number[2] the water flow becomes turbulent. The velocity and all other flow properties vary in a random and chaotic way. To describe this flow the velocity can be

[2] *Reynolds number expresses the ratio of the inertial forces to the viscous forces.*

decomposed into a steady mean value V with a fluctuating component v'. A turbulent flow can now be characterized in terms of the mean values of flow properties and some statistical properties of their fluctuations (Versteeg and Malalasekera, 1995). Usually it is not necessary to solve the details of the turbulent fluctuations. Most of turbulent flow computations are based on the Reynolds-averaged Navier-Stokes equations for turbulent flows. Turbulence models compute turbulent flows with these equations, predicting the Reynolds stresses and the scalar transport terms. The mixing length and k-ε models are most widely used and validated among turbulence models. They assume that there is an analogy between the action of viscous stresses and Reynolds stresses on the mean flow, and that the turbulent viscosity is isotropic (Versteeg and Malalasekera, 1995).

The k-ε model accounts the effects of transport of turbulence properties by convection and diffusion, and by production and destruction of turbulence. Two transport differential equations are solved: one for the turbulent kinetic energy k and another for the rate of dissipation of turbulent kinetic energy ε (Versteeg and Malalasekera, 1995).

Turbulence is filtered by the size of the grid cell. The eddies smaller than the cells are unknown since the Eulerian model does not solve them explicitly. So, circulations inside a grid cell cannot be calculated and are considered as turbulence for the Eulerian model, and are included in the turbulent diffusion term (Martins, 2012).

2.2.5. Model execution

CFD programs always contain 3 main elements: pre-processor, solver and post-processor.

Pre-processing stage includes at least (Versteeg and Malalasekera, 1995):

- definition of the geometry of the region: the computational domain,
- generation of the computational grid,
- selection of the physical and chemical phenomena to be modelled,
- definition of fluid properties,
- specification of the boundary and initial conditions.

Post-processing is usually related with visualization of model results.

2.3. Curvilinear grid

A numerical model requires the discretization of the continuous space into a collection of elementary volumes using a computational grid. This discretization must conform to the domain

boundaries for accurate representation of the boundary conditions (Thompson et al., 1985). In estuarine models the accurate description of the shoreline is a major issue because it influences the results strongly. Accurate resolution of complex geometries is a big challenge in estuarine modelling (Sheng, 2003).

In the traditional Cartesian grid the shoreline is discretized by steps producing a very rough description of the real geometry. Stepwise approximation of boundary description produces errors (Versteeg and Malalasekera, 1995). In addition a large number of inactive cells are always present occupying the computer memory and increasing the computational time. Also a large number of very small grid cells in distant regions of less interest need to be calculated (Versteeg and Malalasekera, 1995).

On the other hand, boundary-fitted curvilinear grids fit the coastline precisely (figure 2.5), have few unused grid cells and allow higher precision in narrow parts of the domain and lower precision in the parts of less interest. The grid cells are of different size and shape but the grid still can be nearly-orthogonal. Curvilinear grids are more difficult in implementing and processing the data. This type of grid is particularly well suited for long and narrow meandering rivers like Guadiana and for all coastal systems in general (Blumberg et al, 2000). However, Versteeg and Malalasekera (1995) note that despite of accurate representation of geometrical details and control of regions of interest, the governing equations with body-fitted grids are much more complicated when they are transformed into a curvilinear coordinate system (when the Finite Difference method is used). In the Finite Volume method it doesn't occur since a grid transformation is not needed.

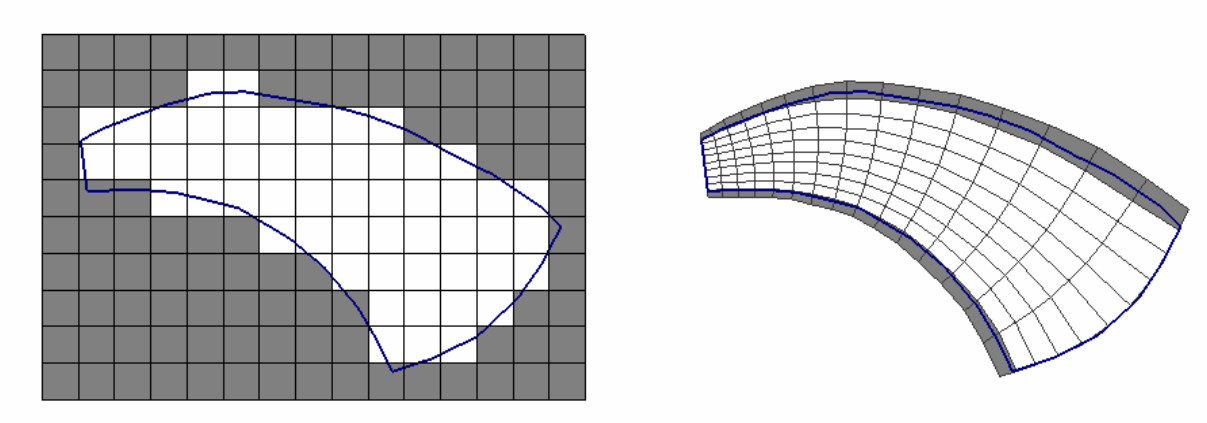


Figure 2.5. Cartesian grid 15×10 cells and curvilinear grid 15×10 cells.

Boundary-fitted curvilinear coordinate systems are generated numerically by determining the values of the physical Cartesian coordinates in the field from the values (and/or angles of intersection) on the boundary. This can be done in two ways: 1) by algebraic interpolation from the boundary values, or 2) by solving a set of partial differential equations with the boundary

values as boundary conditions (Thompson, 1982). The partial differential systems may be elliptic, parabolic or hyperbolic (Thompson et al., 1985).

There are many methods to generate boundary-fitted curvilinear grids developed in the past decades (Thompson, 1982; Sparis, 1985, Driscoll and Stephen, 1998; Akcelik et al., 2001). An extensive review of the early methods has been compiled by Thompson et al. (1982). There is a freedom of choice on the generation method because the generation of the boundary fitted coordinate system has no physical meaning in relation to the problem considered (Sparis, 1985).

The basic idea of a boundary-conforming curvilinear coordinate system is to have some coordinate line which coincides with each boundary segment. The other curvilinear coordinate will vary along the boundary segment and must do so monotonically (Thompson et al., 1985).

For grid generation, the curvilinear physical region is transformed to a rectangular region with a unique correspondence between the Cartesian and the curvilinear coordinates, so every point in the physical field corresponds to only one point in the transformed field, and vice versa (figure 2.6). The computational field (the field in the transformed space) thus has rectangular boundaries and is covered by rectangular grid (Thompson et al., 1985; Martins, 1999).

The first step in general is to position points on the physical boundary which will correspond to corners of the transformed region (Thompson et al., 1985).

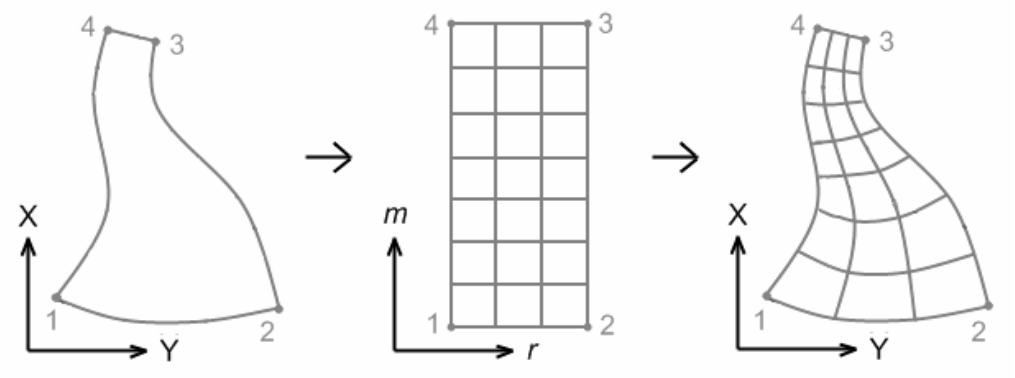


Figure 2.6. Transformed region.

The Cartesian coordinates of the grid points on a physical boundary may be specified or may move freely over the boundary in order to satisfy a condition, for example, orthogonality or the angle of intersecting the boundary by coordinate lines (Thompson et al., 1985).

Grid orthogonality is important for minimizing computational error. Coordinate systems that are orthogonal, or at least nearly orthogonal near the boundary, make the application of boundary conditions more straightforward (Thompson et al., 1985). Although strict orthogonality is not necessary, and the grid can be nearly-orthogonal.

In the Finite Volumes method the transformation of governing equations is not needed since the equations are solved in their integral form in a control volume coinciding with the curvilinear cell (and the cell can have any shape). One drawback of the curvilinear grid is that the grid resolution is a function of the domain boundaries and cannot be freely decided by the user in any desired place of the domain (Martins, 2012).

2.4. MOHID water modelling system

2.4.1. MOHID numerical model

MOHID is a water modelling system written in Fortran using object-oriented programming strategy, supporting graphical user interfaces for model pre- and post-processing (Miranda et al., 2000; MOHID, 2002; Braunschweig et al., 2004). It is an open-source modular system developed in the Technical University of Lisbon[3]. MOHID Water is a 3D numerical program to simulate surface water bodies, MOHID Land module is for simulating hydrographic basins, and there are other modules available. MOHID Water module simulates flow and water properties in surface water bodies solving the shallow water equations by the Finite Volume method.

The MOHID system includes a baroclinic hydrodynamic module for the water column and 3D for the sediments, and the correspondent Eulerian transport and Lagrangian transport modules (Pina et al., 2003). Parameters and processes involving non-conservative properties are computed by other specific modules (e.g. turbulence module, water quality, ecology and oil transformation).

The model solves the equations using the Finite Volume method in the real domain without any space transformation. The geometry information is presented as the areas and volumes which are needed to calculate the fluxes. So, there is a complete separation of the hydrodynamic variables and the geometry for all grid types; and the solution is independent of the mesh geometry. The geometry information is updated in each time step as a function of the grid type. The cells can have any initial shape and suffer any deformations. The same code can be used with any discretization, and different discretizations can be used in different regions of the domain at the same time (Martins et al., 2001, Martins et al., 1998).

The main numerical characteristics of the hydrodynamic module are listed on the MOHID web page (http://www.mohid.com/Hydrodynamics.htm).

MOHID uses Arakawa-C staggered grid type, where the water level and the properties are

[3] *http://www.mohid.com/*

computed at grid cell centers and velocity components at the mid-points of the grid cell faces (figure 2.7) (Arakawa and Lamb, 1977).

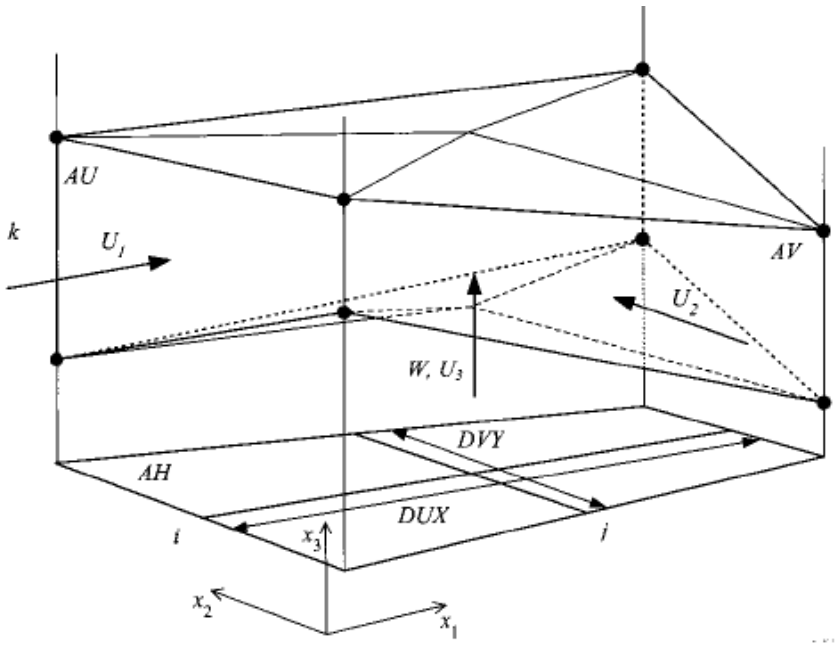


Figure 2.7. Grid cell in MOHID (Martins et al., 2001).

The hydrodynamic model solves the three-dimensional primitive Navier-Stokes equations in Cartesian coordinates for incompressible flows, assuming hydrostatic equilibrium and the Boussinesq approximation.

The mass and momentum evolution equations are (Martins et al., 2001):

$$\frac{\partial u}{\partial t}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

$$\frac{\partial u}{\partial t}+\frac{\partial (uu)}{\partial x}+\frac{\partial (vu)}{\partial y}+\frac{\partial (wu)}{\partial z}=fv-g\frac{\rho_\eta}{\rho_0}\frac{\partial \eta}{\partial x}-\frac{1}{\rho_0}\frac{\partial p_s}{\partial x}-\frac{g}{\rho_0}\int_z^\eta \frac{\partial \rho'}{\partial x}\partial z+\frac{\partial}{\partial x}\left(A_h\frac{\partial u}{\partial t}\right)+\frac{\partial}{\partial y}\left(A_h\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left(A_h\frac{\partial u}{\partial z}\right)$$

$$\frac{\partial v}{\partial t}+\frac{\partial (uv)}{\partial x}+\frac{\partial (vv)}{\partial y}+\frac{\partial (wv)}{\partial z}=-fu-g\frac{\rho_\eta}{\rho_0}\frac{\partial \eta}{\partial y}-\frac{1}{\rho_0}\frac{\partial p_s}{\partial y}-\frac{g}{\rho_0}\int_z^\eta \frac{\partial \rho'}{\partial y}\partial z+\frac{\partial}{\partial x}\left(A_h\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(A_h\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial z}\left(A_h\frac{\partial v}{\partial z}\right)$$

$$\frac{\partial p}{\partial z}=-\rho g$$

where *u, v* and *w* are the velocity vector components in the Cartezian *x, y* and *z* directions, η is the free surface elevation, *f* is the Coriolis parameter, A_h and A_v are the turbulent viscosity in horizontal and vertical directions and p_s is the atmospheric pressure. And ρ is the density and ρ' is its anomaly.

The computed flow field transports salinity, temperature and any other property using an

advection-diffusion equation. Salinity and temperature values are provided by a transport equation:

$$\frac{\partial P}{\partial t}+\frac{\partial (uP)}{\partial x}+\frac{\partial (vP)}{\partial y}+\frac{\partial (wP)}{\partial z}=\frac{\partial}{\partial x}\left(k_h \frac{\partial P}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_h \frac{\partial P}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_v \frac{\partial P}{\partial z}\right)+S_{ST}$$

where P stands for a property and S_{ST} is a source-sink term. k_h and k_v are the horizontal and vertical diffusivities respectively (Martins et al., 2001).

The Eulerian module used to transport these properties is based on the same finite volume method as the hydrodynamic model and is independent of the transported property. The same transport module is used in the sediment transport, water quality and ecological modules to transport different conservative and non-conservative properties (Pina et. al, 2003).

The density is calculated as a function of salinity and temperature by the constitutive law (the equation of state):

$$\rho = (5890 + 38T - 0.375T^2 + 3S)/((1779.5 + 11.25T - 0.074T^2) - (3.8 + 0.01T)S + 0.698 (5890 + 38T - 0.375T^2 + 3S))$$

The model uses the Finite Volume approach to discretize the equations. For discretization of the momentum equations, they are integrated for each cell volume. The final form of the u_i momentum equation is (Martins et al., 2001):

$$\frac{V_u (U_i^{t+1} - U_i^t)}{\Delta t} + \sum_j U_i \cdot U_{i\,flux} = \frac{1}{\rho_0} F_i$$

where F_i is the x_i component of the forces applied to the fluid mass due to barotropic, baroclinic, Coriolis, horizontal and vertical diffusion effects. U_{jflux} are the horizontal water fluxes across the U cell faces (Martins et al., 2001).

The discretized equations are described in details by Martins (1999) and Martins et al., (2001).

The model uses a semi-implicit ADI algorithm with two time levels per iteration for time discretization.

For stability reasons the vertical transport and barotropic pressure are computed implicitly, all the other terms in the momentum equation are calculated explicitly.

Five types of boundaries are used in MOHID: free surface, bottom, lateral closed boundary, lateral opened boundary, and moving boundary (Martins et al., 2001). The water flux across the free-surface boundary and the wind stress can be considered. The water flux at the bottom

boundary can be assumed and the bottom stress is implemented implicitly using a quadratic law. The closed boundaries correspond to land, and water fluxes and momentum diffusive fluxes are assumed null for the cell faces in contact with land. At the open boundaries, the tidal signal is imposed by specifying the free surface elevation of those cells, and also radiative conditions can be set. Moving boundaries are closed boundaries whose position varies with time; this often happens in tidal flats. A criteria for considering a cell as dry are described by Martins et al. (2001). This enables the drying and flooding of cells as a function of the water level.

The turbulent transport of momentum, mass and heat in the MOHID can be calculated in a simplified way using constant diffusion coefficients[4]. However in the vertical direction, the user can compute the evolution of turbulent flow properties in a more realistic way using the k-ε model GOTM (Global Ocean Turbulence Model) (Pina et al., 2003).

In this work the model is used with only one vertical layer, as a 2D depth-integrated model, because the Guadiana Estuary is rather shallow. In a 2D model the computed properties are the averages of the entire water column.

2.4.2. MOHID and GIS integration

MOHID numerical modules generate temporally variable three-dimensional data. The MOHID package includes GIS module which handles spatial and temporal variable data in specific formats required or produced by MOHID (Braunschweig et al., 2005). It allows to create and visualize data stored in different formats: MOHID input data files (ASCII files containing geographical information like points, lines, polygons, grids, structured in a similar way as XML files), visualize ESRI shapefiles, MOHID output HDF5 files (Braunschweig et al., 2005). MOHID GIS is written in Microsoft Visual Basic .NET and uses some executable extensions written in Fortran (MOHID, 2002). This shows the high-level GIS/model integration. The current version of MOHID GUI is 4.9.2.

MOHID GIS provides a grid generator that produces structured nearly-orthogonal curvilinear meshes based on the cross-ratios of the Delaunay triangulation (CRDT) algorithm (Driscoll and Stephen, 1998). Producing such grid requires a meshing domain polygon which must follow the shoreline. MOHID GIS can also interpolate bathymetry point data into a computational grid, using land polygon presenting non-computing areas.

MOHID GIS can open the data in any coordinate system, but it should be the same for all datasets since it cannot perform transformations. MOHID Studio, the new GUI for numerical

[4] *http://www.mohid.com/Turbulence.htm*

models (described below), is able to reproject the data of the same datum but cannot perform transformation between geographic coordinate systems.

There have been attempts to improve integration of MOHID with GIS. Pires (2006) developed methods to insert MOHID simulation results into a geographic information system using ArcGIS software. The work was applied to environmental risk assessment. The modelling results were exported from MOHID as raster images and manually georeferenced in the GIS by defining control points on a satellite image. Tironi et al. (2008) programmed a management tool which was a modified ArcView 3.3 interface that showed, in a simple and user-friendly way, the combined results of the hydrodynamic model in a GIS environment. Navas et al. (2011) also incorporated MOHID hydrodynamic model results into the ArcView GIS system for visualization, interpretation and future spatial analysis in aquaculture. However, the method of this incorporation was not described.

There is a new graphical user interface for the MOHID numerical engines (MOHID Water, MOHID Land and MOHID River), named MOHID Studio[5], which includes some properties of GIS. Braunschweig (2012) noted that results from numerical models were difficult to represent in classical GIS for several reasons: poorly addressed time dimension, difficult data exchange with numerical engines written in Fortran, and very large quantities of data. MOHID Studio tries to overcome these difficulties by using several open source libraries: an OpenGL extended version of SharpMap (map visualization), DotSpatial.Projection (coordinate transformation), HDF (data storage), NHiberante Spatial (data base access) and OpenMI (inter model data exchange) (Braunschweig, 2012). It is based on the .Net language and uses the XML format for vector data.

MOHID Studio version 1.2.4.0 (the Map tab), as well as MOHID GIS, is able to create and show Cartesian grids, vector data, grid data (slightly correspondent to raster), time-dependent gridded model results (and Lagrangian points), velocity vector fields (as arrows), and is also able to fill sinks and delineate watersheds, and some other specific features. The advantages of MOHID Studio over MOHID GIS are the improved file conversion between MOHID files and shapefiles, the additional interpolation methods (former triangulation and new IDW), and faster rendering of large datasets. But MOHID Studio cannot create curvilinear grids and cannot open a branched curvilinear grid (with tributaries) or grid data (gridded bathymetry) based on such grid.

A significant advantage of MOHID Studio is that it can reproject spatial data. However, it cannot do transformations between different datums. The used DotSpatial library is a port of the proj.4 C++ library to C# .Net. There is a known problem with datum transformations using this

[5] *http://www.actionmodulers.pt/default.aspx?canal=33*

library[6].

Since MOHID system is already integrated with GIS (having its own GIS module MOHID GIS) for necessary basic pre-processing, this work concentrates not on the integration and coupling but on investigating advanced GIS methods to improve model results.

[6] *http://dotspatial.codeplex.com/discussions/396921*

3. The Guadiana Estuary

This chapter shortly describes the study area, as well hydrodynamic models and the investigations which have been done for the Guadiana Estuary.

3.1. Physical Characterization of the System

The Guadiana Estuary is a narrow rock-bound estuary located at the southern Iberian Peninsula, between Portugal and Spain (figure 3.1). The regional climate is classified as semi-arid in general.

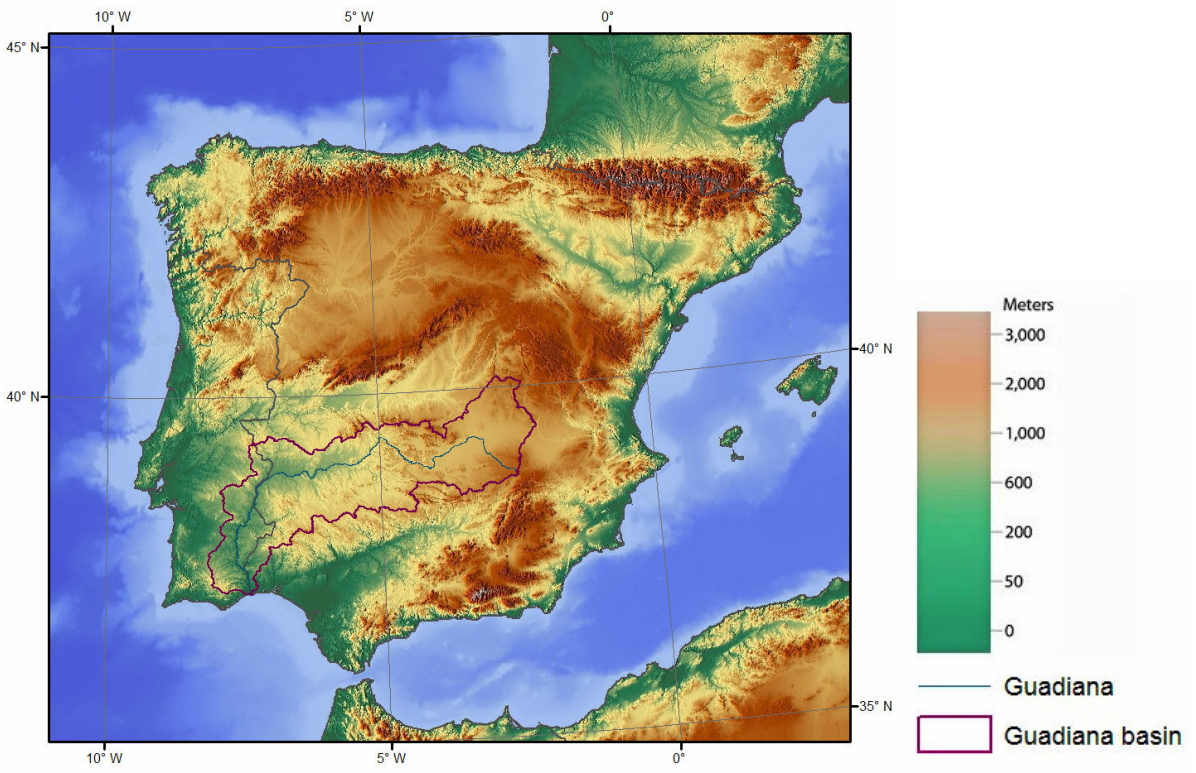


Figure 3.1. The Guadiana drainage basin[7].

The estuary extends for about 80 km from the mouth upstream (figure 3.2) and is prolonged offshore by a submerged delta (Garel et al., 2009a). It passes open coastal plain only in the last 7 km near the mouth (Gonzalez et al., 2005), which is a part of an old delta where salt marsh systems are dominant (figure 3.3) (Morales, 1997). The rest of the estuary upstream is a narrow valley.

The river discharge was highly seasonal in past decades, with higher values in winter months (averages about 400 m^3/s) and episodic floods (Lobo et al., 2004). But recently after closing the Alqueva dam (in 2002), the river discharge to the estuary is very low all year round (less than 10 m^3/s in summer and about 20 m^3/s in winter). Additionally, the estuary receives freshwater inputs from several tributaries (mainly right-side), such as Beliche, Odeleite, Vascao streams.

[7] *Elevation data from http://www.shadedrelief.com*

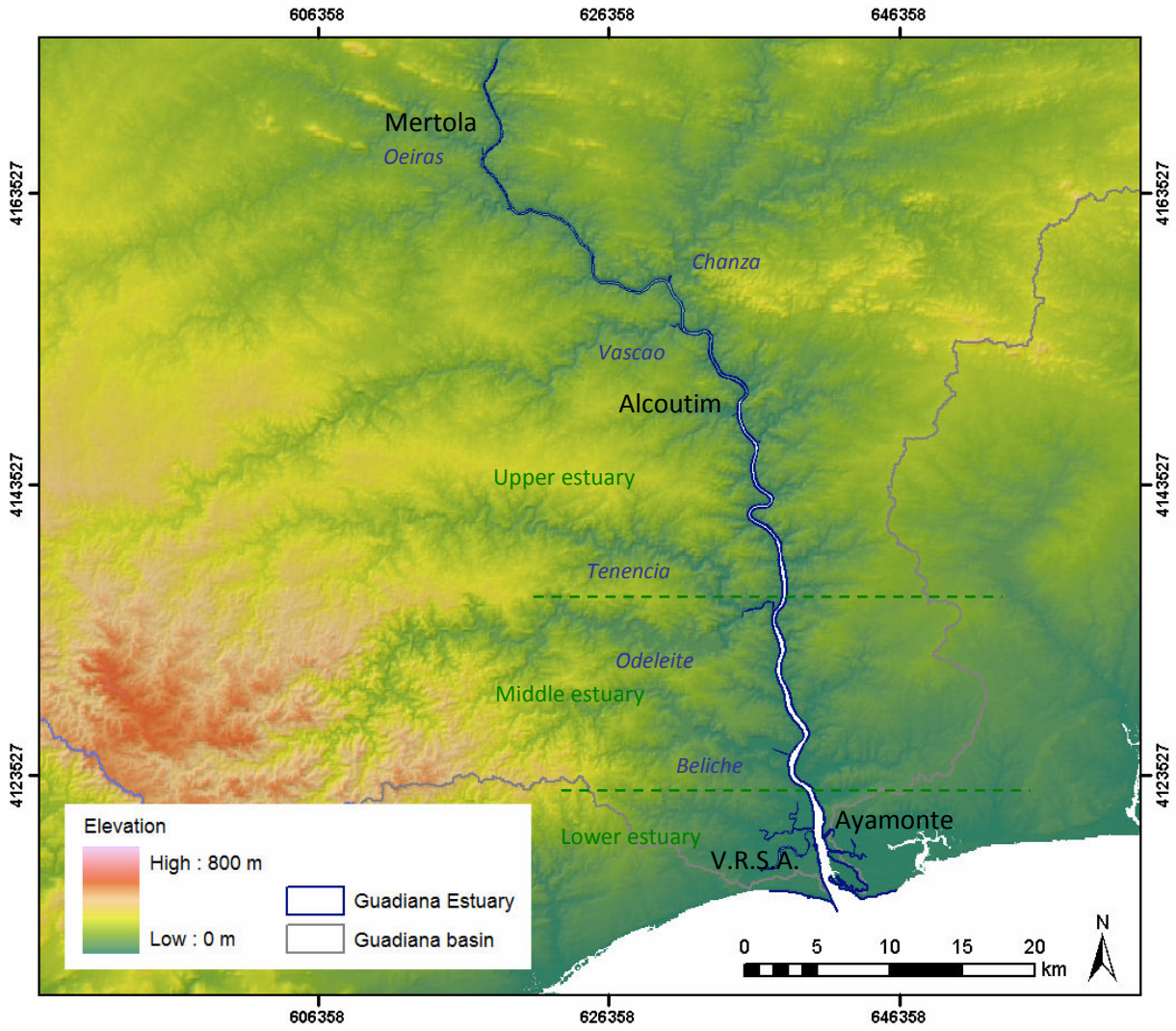


Figure 3.2. The entire Guadiana Estuary[8].

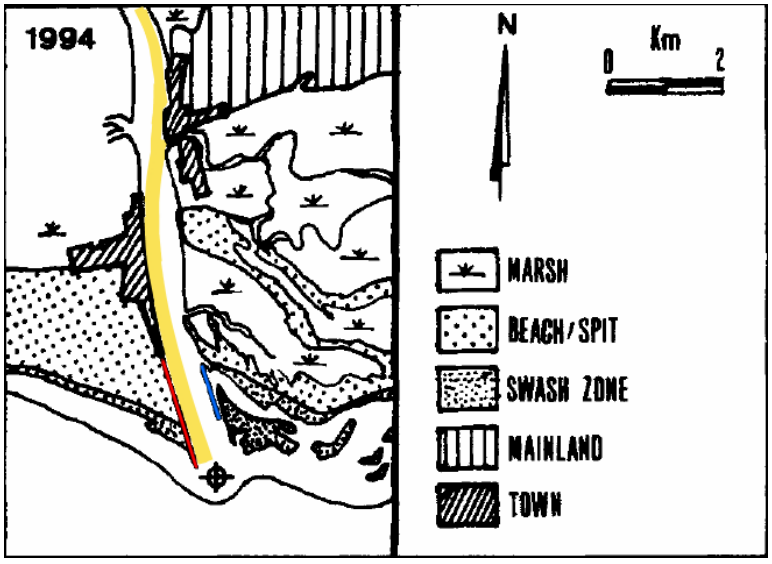


Figure 3.3. The Guadiana mouth, morphology (Morales, 1997), the red and blue lines show the jetties (the blue is submerged), the yellow line marks the deep channel.

[8] *SRTM elevation data. WGS 84 zone UTM 29 coordinates*

The estuary (until 50 km upstream) has an average depth of about 5 m with maximum depths up to 18 m (Lobo et al., 2004). The tidal signal is regular semidiurnal and mesotidal, with low diurnal inequality in tidal elevation (figure 3.4). Mean tidal amplitude is around 2 m and tidal ranges are 1.3 m for neap tides and 3.5 m for spring tides (Lobo et al., 2004). There is also inequality in fortnightly tide. Every month there is a spring tide with higher amplitude and one with smaller, and the same for neap tides (figure 3.4). The most important components are M2 and S2.

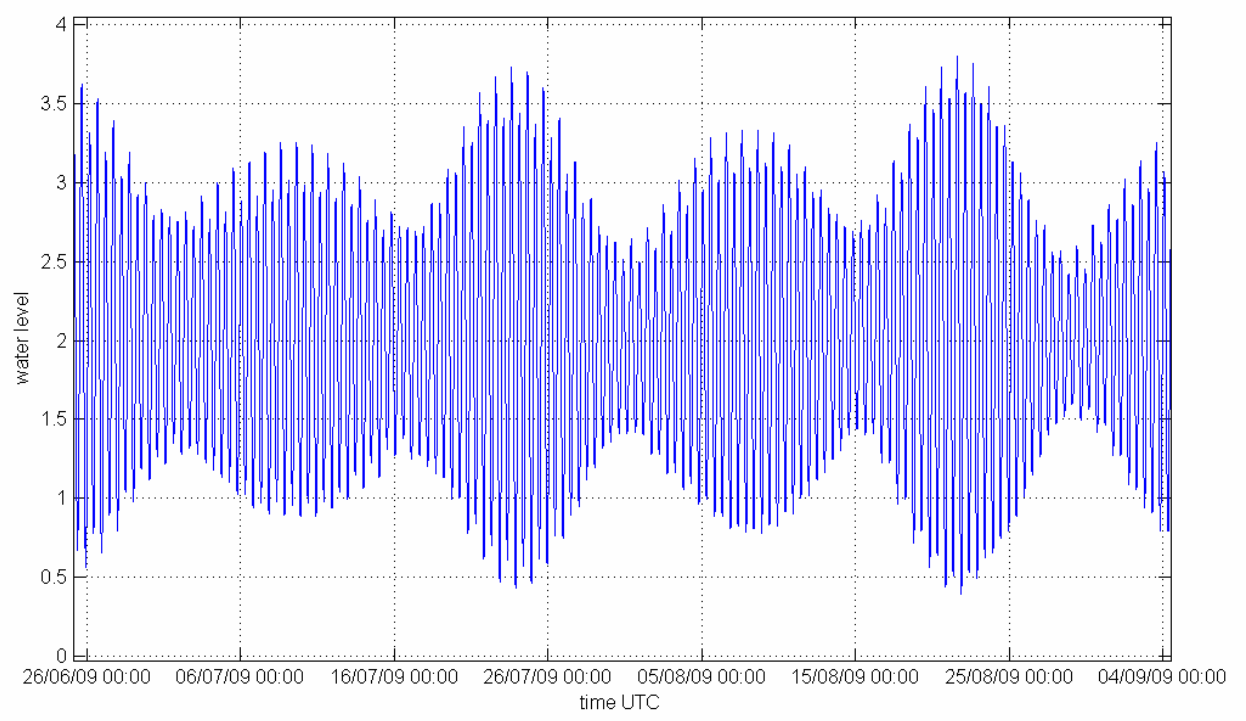


Figure 3.4. Water level (m) in the lower estuary during two months.

The estuary is under high human pressure, including effect of dams and different types of pollution, such as urban sewage, agricultural, mining (Wolanski et al., 2006; Delgado et al., 2010). The mouth is fixed by two jetties (breakwaters) built in 1974 (Morales, 1997), one of them is submerged (figure 3.3). They caused changes in land cover of the estuary, such as in sand marsh and marsh areas (Gonzalez et al., 2005).

The offshore coastal waves are coming mainly from south-west and west (Gonzalez et al., 2005) and mean significant wave height is 0.6 m (Morales, 1997). West-to-east currents in the coastal zone produced an asymmetry in the delta (figure 3.5) (Morales, 1997).

Water circulation in estuaries is usually driven by differences in density (baroclinic) and pressure (barotropic, tidally induced). The hydrodynamics of the Guadiana Estuary have been investigated by several authors. Garel et al. (2009a) examined the relative impacts of tidal and river discharge forcing on water circulations using field hydrographical data (vertical profiles of current, salinity and turbidity) collected at different regions of the estuary under different tidal

and discharge conditions. Flow velocities are found higher in the lower estuary and reducing upstream (Garel et al., 2009a). Maximum velocities are about 1.4 m/s.

Figure 3.5. The Guadiana mouth and the lower estuary, aerial view from south-east[9]

At spring tide and low river discharge the estuary is well-mixed. But at neap tide the estuary is partially stratified even at low river discharge, more in lower estuary (Garel et al., 2009a). It is better mixed during the flood than during the ebb. But at high river discharge the estuary is highly stratified, and a salt wedge is developed on the flood (Garel et al., 2009a). These differences are visualised in the table 3.1. So, the highest stratification occurs at high river discharge, neap tide, the ebb phase; and the best mixing is during flood of spring tide at low discharge.

Table 3.1. State of the Guadiana Estuary under different conditions.

	Spring tide	**Neap tide**
Low river discharge	well-mixed	partly stratified
High river discharge	highly stratified	highly stratified

[9] *Toksave 2007, under GNU Free License 1.2, http://commons.wikimedia.org/*

Garel and Ferreira (in press) also investigated the dynamics of tidal flow and residual currents at the lower estuary at a fortnightly time scale. At the deep channel at spring tide the depth-averaged currents are stronger and longer during the ebb phase, and during the flood at neap. This resulted in opposite residual transport directions in the two weeks cycle at the deep channel (Garel and Ferreira, in press). This dynamics is integrated in the table 3.2.

Table 3.2. Tidal asymmetry in the deep channel (depth-averaged currents).

	SPRING TIDE		**NEAP TIDE**	
	duration	**maximum velocities**	**duration**	**maximum velocities**
FLOOD (inflow, upstream current)	shorter (than ebb)	smaller	**longer**	**higher**
EBB (outflow, seaward current)	**longer**	**higher**	shorter	smaller

The net water transport across the entire channel is up-estuary at spring and down-estuary at neap, i.e., opposite to the one at the deep channel (Garel and Ferreira, in press). At spring tide there is the outflow in the deep channel and the inflow over the shoals; this fact results from the combination of the upstream Stokes transport (which is larger over the shoals) and compensating return flow. At neap tide the inflow at the deep channel is associated with gravitational circulation (Garel and Ferreira, in press). These changes in residual currents are presented in the table 3.3.

Table 3.3. Circulation patterns in the lower estuary.

	SPRING TIDE		**NEAP TIDE**	
	deep channel	**shoals**	**deep channel**	**shoals**
Residual currents (tidally averaged currents, net water mass transport)	**outflow** (downstream current)	**inflow** (upstream, due to Stokes transport)	**inflow** (due to gravitational circulation)	**outflow**
	upstream (entire channel)		**downstream** (entire channel)	

3.2. Existing Guadiana models

Numerical models have already been applied to the Guadiana Estuary for simulation of flow and tidal currents (Lopes, 2004; Lopes et al., 2003; Martins et al., 2004; Oliveira et al., 2006). Previous simulations were performed usually with rather coarse Cartesian grids and covered only the lower half of the estuary. In this work curvilinear grids are used in this estuary for the first time.

Hydrodynamic modelling of the estuary was done in 2001 during EMERGE[10] project, using variable Cartesian grid (spatial resolution up to 250 m) in MOHID in 2D (Dias and Ferreira, 2001). Bathymetry data for this simulation were collected in the same project. Also sedimentology, hydrology, ecological properties, bottom morphology, and geological structure of the estuary were investigated empirically during this project.

2D hydrodynamics, salinity and sediment transport of the Guadiana Estuary were also modelled in MOHID using Cartesian grid with resolution of 180×180 m (Lopes, 2004). Bathymetry was obtained from the Instituto Hidrográfico. The results were calibrated and validated using field measurements from the regions close to the mouth. The model calibration showed that results are comparable with the measurements. This model however needed to be adapted and improved, particularly with the construction of a higher resolution mesh. Grid dimensions did not allow to determine all desired parameters (Lopes et al., 2003). Based on this model, an approach to evaluate the impact of water discharge from dams on advection of fish larval stages off estuaries was developed using lagrangian modelling in MOHID (Morais et al., 2012).

MOHID Water Modelling System was used to perform simulations with different nitrogen load scenarios in several Portuguese estuaries. Phytoplankton, ammonia and nitrate average annual concentration and transport between boxes for Guadiana Estuary were simulated with a very coarse Cartesian grid (Saraiva et al., 2007; Martins et al., 2004).

The hydrodynamics of the Guadiana Estuary also was simulated in two and three dimensions using coarse unstructured triangular grid by Oliveira et al. (2006). Those numerical grids extended from the limit of tidal intrusion to the continental shelf. The model results were needed to investigate the flushing properties of the lower estuary for passive organisms and for organisms with vertical migration capabilities. Depth-averaged flow simulations were performed with the shallow water model ADCIRC and salinity and stratification were analyzed with the 3D baroclinic model ELCIRC. In that model good agreement between salinity results and observations was achieved for well-mixed conditions, but the accuracy of the 2D simulations decreased under partially mixed and stratified conditions (Oliveira et al., 2006). Stratified conditions were confirmed to occur at neap tide.

[10] *http://w3.ualg.pt/~jdias/JAD/eb_Emerge.html*

4. Methods

In estuarine modelling the high quality of spatial input data is essential to ensure good model results; therefore several pre-processing methods were developed using GIS tools. This chapter presents the methods used for preparing the model spatial inputs. The key issues of near-shore surface modelling include large heterogeneous data sets and anisotropy. The necessary data were obtained from many different sources in disparate coordinate systems. Sonar surveying method usually produces extremely large amounts of data, so it is necessary to cluster the data, prior to its interpolation. Bathymetric surveys also display gaps in the shallowest parts of the estuary and there is a need to recover bathymetry information.

The following sections describe the data used for model setup, coordinate transformation, bathymetry pre-processing (including aggregation and estimation from an orthophoto in the shallow areas), extraction of the water polygon, creation of a curvilinear grid, and interpolation of the bathymetry into the grid by several techniques, including anisotropic interpolation in the along-channel coordinate system.

4.1. Model inputs

For running a hydrodynamic model MOHID needs the following data. Obligatory inputs are gridded (spatially discretized) bathymetry and forcing (such as water discharges or tide). The tidal signal is included as changes in water level of the open boundary cells using a list of harmonics (presented in the section 4.3 Model setup). Water discharges are included as locations in certain grid cells with specified water flow value and concentrations of properties.

MOHID spatial inputs and outputs of the Guadiana Estuary model are schematically presented in the figure 4.1. The figure presents only spatial inputs for a simple hydrodynamic model forced by tide and river flow. MOHID GIS requires all the data to be in the same coordinate system.

Gridded bathymetry is a key spatial input requiring spatial data of high quality. MOHID GIS can interpolate bathymetry point data into a computational grid, using land polygon as non-computing areas (figure 4.1). In a case of a curvilinear grid, firstly a water domain polygon is needed to generate the grid (described later in the section 4.2.4. Curvilinear grid creation).

The time series point locations are defined at the locations of real measurements, based on the correspondent cells of the gridded bathymetry. The model creates output files with changes of properties at these points. The same can be done using polygons (“boxes”).

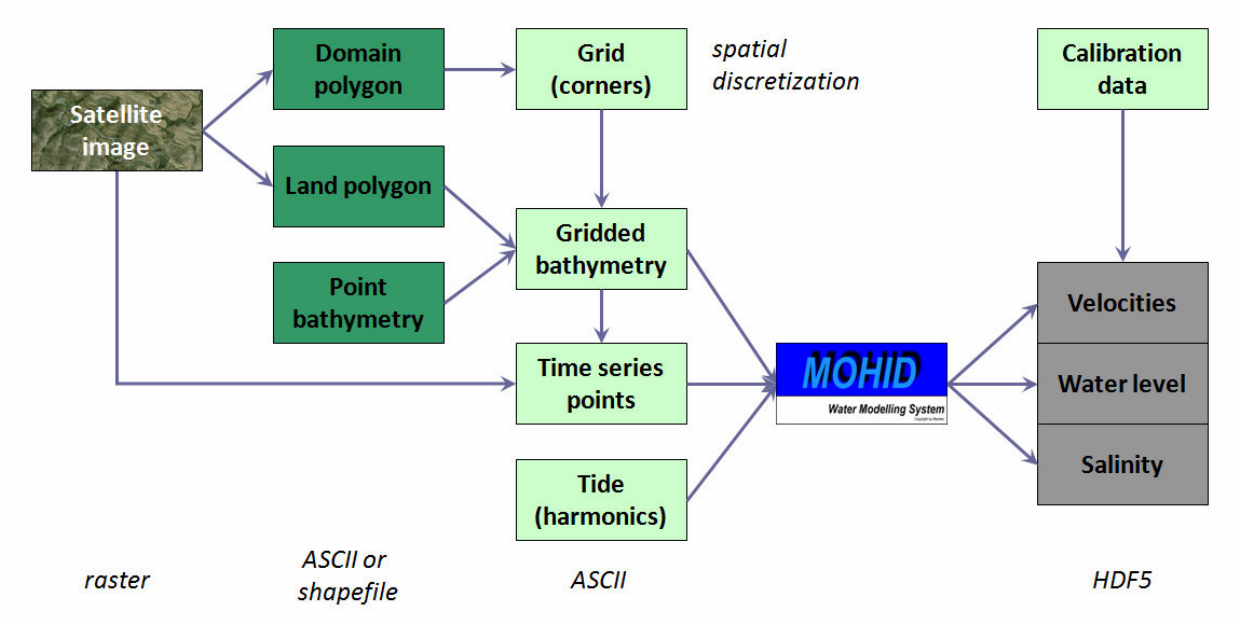


Figure 4.1. MOHID hydrodynamic model data.

River flow values are temporal data, but located in the certain grid cells. Tidal input is not really spatial in a case of one tidal gauge when it is applied to entire open boundary, however in a case of large study areas in the ocean several tidal gauges in different locations are needed requiring a spatiotemporal treatment of the data.

In a case of more complicated simulations with initial values of several water properties, wind, solar energy, pollutant concentrations etc. more spatial inputs are needed to input these conditions. They can be gridded fields (such as bathymetries) or defined in polygons (boxes).

Basically, GIS can help in processing all these input datasets. It is especially useful for spatial discretization, i.e. creating and processing gridded bathymetry and its sources (grid, polygons, survey points). It is described in details in the next sections.

4.2. GIS tools and database

The coordinate system of the project was chosen as WGS 1984 projection UTM Zone 29.

4.2.1. Data sources

The data needed for the model inputs were obtained from many different sources.

Aerial and satellite imagery

➢ RGB orthophotos of the lower estuary from 2005 year with 0.5 m spatial resolution, in the projected coordinate system Datum 1973 Hayford-Gauss (IPCC). The images were taken at high tide (figure 4.2), by the Instituto Geográfico Português.

➢ RGB orthophotos[11] of the lower estuary from the Instituto Geográfico Português of 2010 year with 0.5 m spatial resolution, in the ETRS 1989 PT-TM06 coordinate system. The images show high tide (figure 4.2).

➢ RGB orthophotos for the entire estuary obtained from free public web mapping services of aerial and satellite imagery, such as Google Maps/Earth, Bing (Microsoft) Maps, and others. The images were downloaded with 4.7 m, 2.4 m and 1.2 m resolution and georeferenced in the chosen WGS 1984 coordinate system. Bing image is rather recent and shows low tide, Google image is from year 2007 and was taken between high and low tides. Nokia (and former Yahoo and Navteq) services have clouds and old low-resolution images for upper estuary.

➢ Satellite image of Landsat 7 (sensor ETM+) from 25/05/2003, which consists of eight spectral bands with a spatial resolution of 30 m for Bands 1 to 7 (visible and infrared). The resolution for panchromatic band (Band 8) is 15 m. The wavelengths are listed on the official website[12].

Bathymetry data

➢ Bathymetry data for the outer ebb delta of the estuary from December 2010, corrected from tidal variations. The data are in the Datum 1973 Hayford-Gauss (IPCC) coordinate system, very dense points along ship tracks (figure 4.3). Obtained in CIMA, UALG (by E. Garel).

➢ Water depth data for the lower Guadiana right tributaries (Carrasqueira and Leziria) from 2005 year, georeferenced in Lisboa Hayford-Gauss, Military (IGeoE) coordinate system. Collected by Blueedge.

➢ Gridded old bathymetry data (60 m resolution, probably 2001 year) with no georeference.

➢ Old bathymetry data of the lower estuary and nearby ocean in WGS 1984 geographic coordinate system (professor J. Luis).

➢ Raw water depth data for the lower estuary obtained in University of Huelva (professor J. Morales) in June-July 2010. The data were very dense points along ship tracks in the WGS 84 UTM Zone 31 coordinate system, with no correction from tidal height. This correction was done during pre-processing.

---

[11] Orthophoto is a geometrically corrected vertical aerial photograph ("orthorectified", in order to remove perspective and terrain distortions by transforming the central projection of the photograph into an orthogonal view of the ground with the uniform scale, so the photo becomes equivalent to a map).

[12] http://landsat.usgs.gov/band_designations_landsat_satellites.php

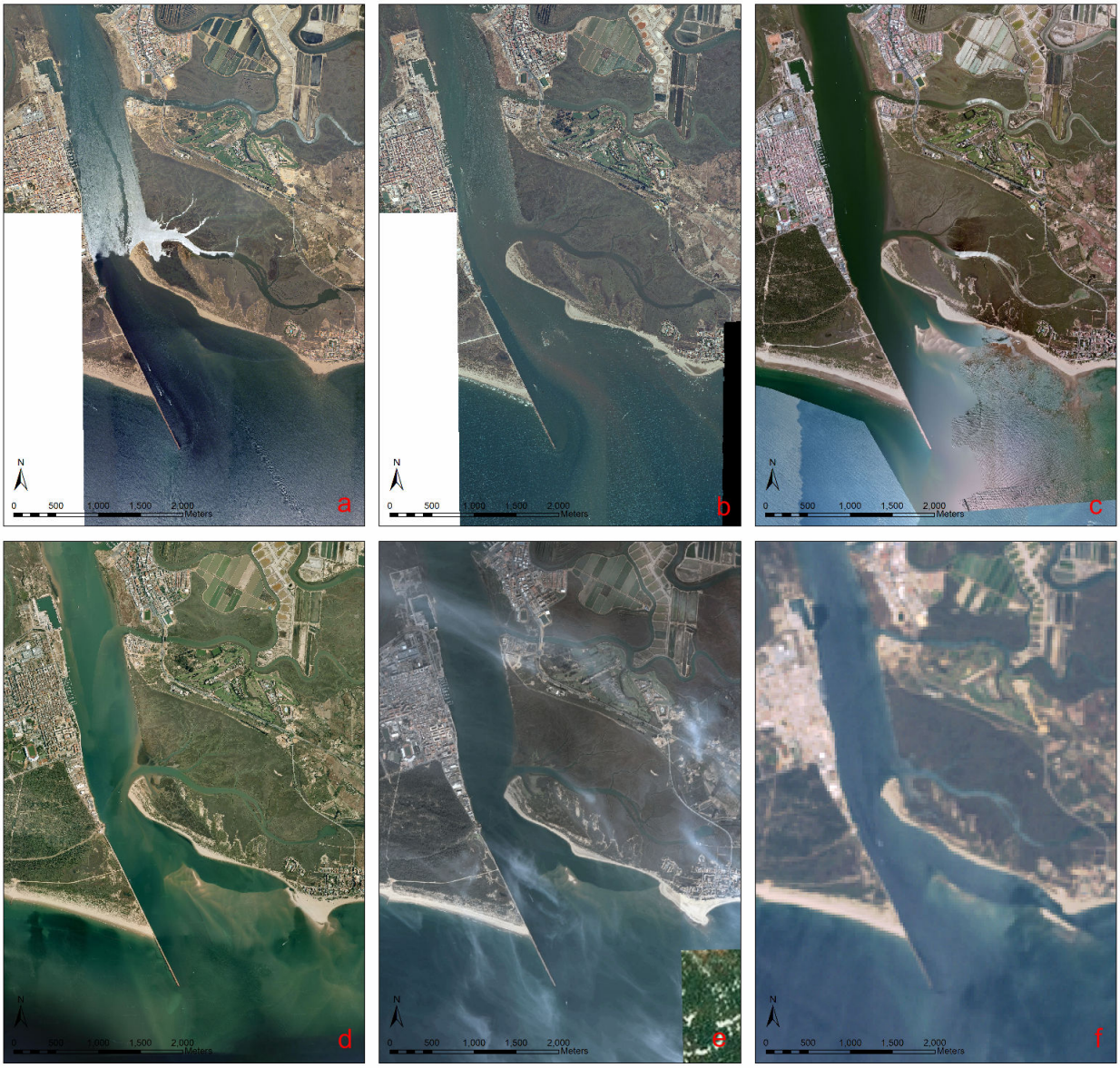


Figure 4.2. The remotely sensed imagery of the lower estuary: a) IGP 2005, b) IGP 2010, c) Bing, d) Google, e) Nokia, d) Landsat 7 (RGB composite).

➢ Gridded bathymetry data from the EMERGE 2000 project for the lower and middle estuary, in Datum Lisboa projection Hayford-Gauss (IPCC). The grid size was 5x25 m.

➢ Bathymetry data around the location of the international bridge between Portugal and Spain, in WGS 1984 geographic coordinate system (collected by P. Valdes, Spain).

➢ Bathymetry data from the Instituto Hidrográfico, available at their website[13]. The data were collected in different years: in 1990 for the upper and middle estuary (filtered points along ship tracks), in 2010 for the lower estuary (gridded data, 100 m grid).

All bathymetry data are presented in the figure 4.3.

[13] *http://www.hidrografico.pt/download-gratuito.php*

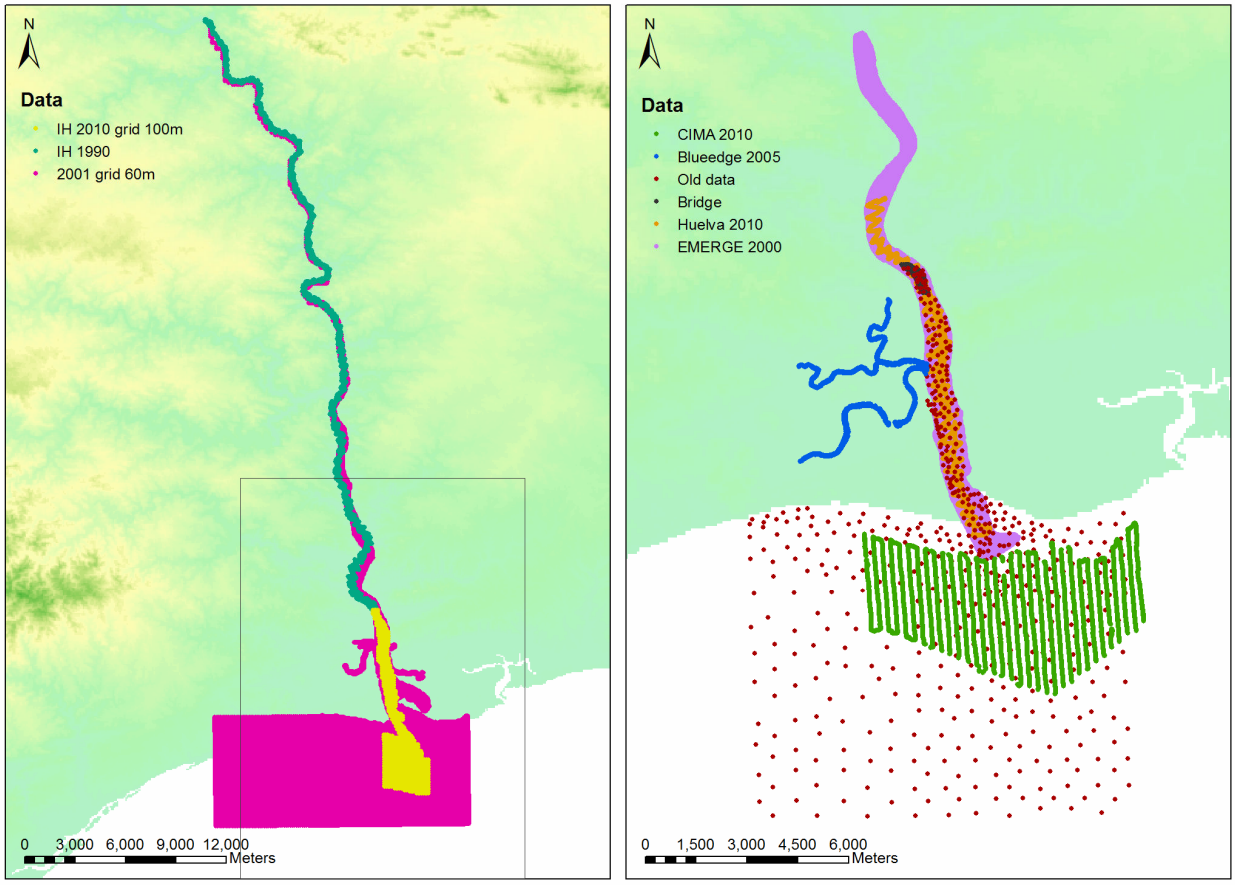


Figure 4.3. Available bathymetry data.

River flow data

Flow data for the Guadiana Estuary were obtained from the SNIRH[14] database. The stations used in this work are presented in the figure 4.4. Daily flow values for the model were taken from the Pulo do Lobo station and complemented by the stations on lower tributaries, or calculated as sum of the upper stations in the period when the Pulo do Lobo was not working. As for the tributaries included into the model (Odeleite and Beliche), mostly the monthly mean flow values were taken due to the lack of daily data.

[14] *Sistema Nacional de Informação de Recursos Hídricos http://snirh.pt/*

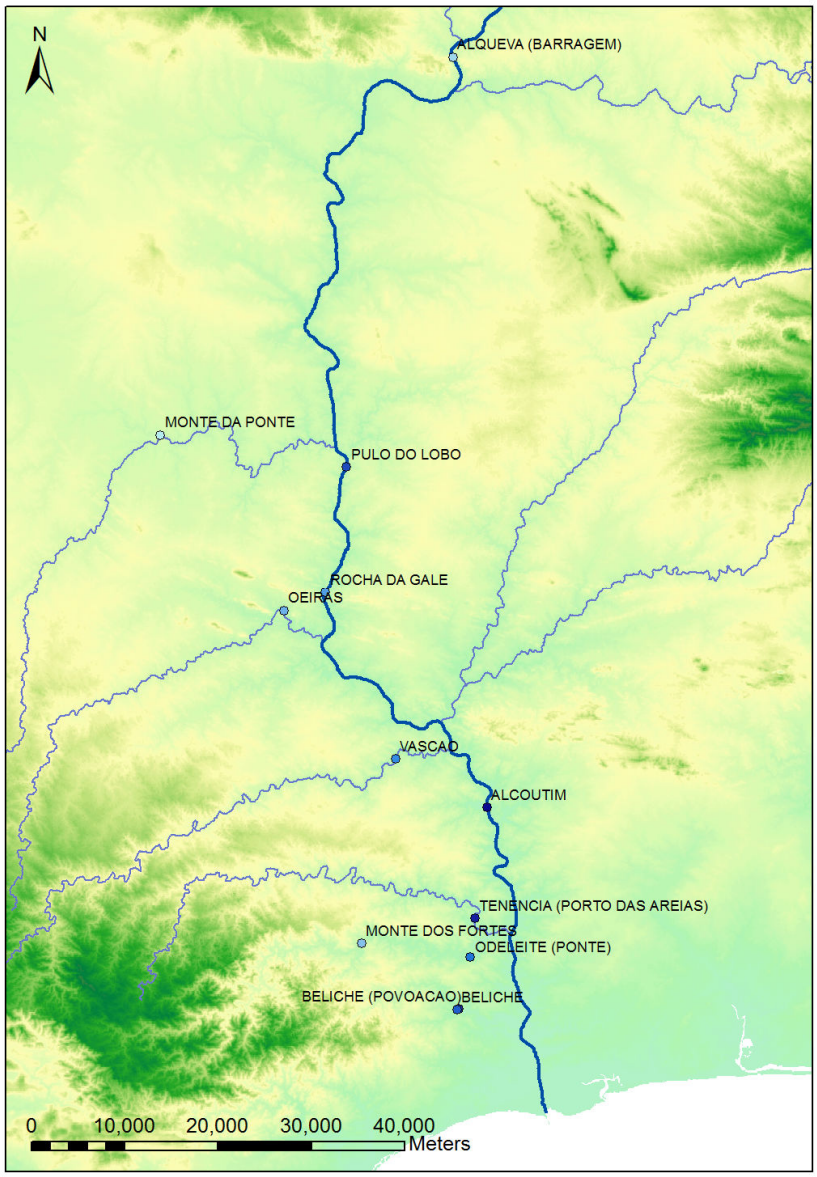


Figure 4.4. Flow data stations.

Calibration data

➢ Velocity components integrated for the whole water column, and water level data (in the form of pressure) were obtained from the Simpatico system (Garel et al., 2009b; Garel et al., 2011) located in the lower estuary (figure 4.5). The data are from 2008, 2009, and 2012 years (figure 4.6).

➢ Water height and surface, bottom and middle velocities at station near Ayamonte (figure 4.5). The data taken with ADCP system at water surface on 5th and 15th of July 2010. The coordinates of the station were given in Datum ED 50, Spain and Portugal.

➢ CTD vertical profiles (every meter) at fixed stations from the Campanha de Primavera on

the Guadiana River carried out on 23rd of May 2001 and 29th of May 2001.

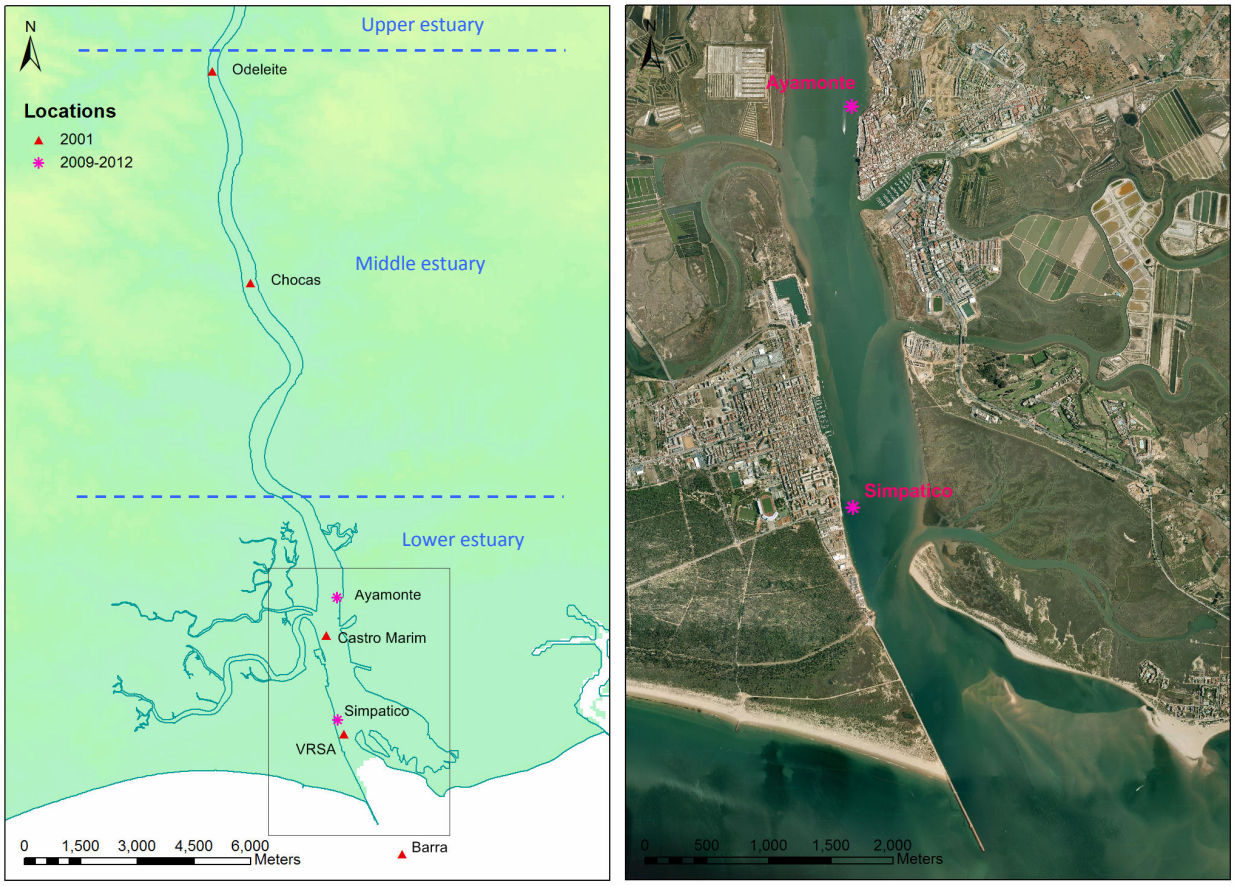


Figure 4.5. Calibration data locations.

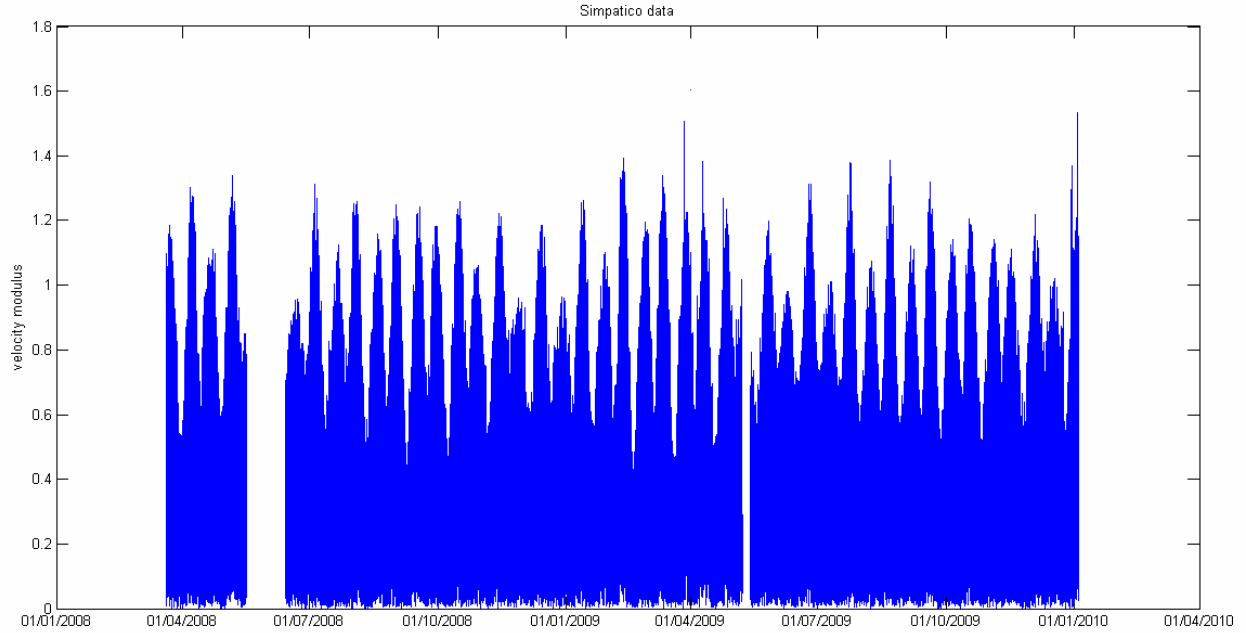


Figure 4.6. Simpatico 2008-2009 data of velocity of the currents (m/s).

Elevation and coastline data

For investigation of surrounding topography the elevation data SRTM (90 m resolution) and ASTER GDEM (30 m resolution) were used. SRTM data[15] are the processed Shuttle Radar Topography Mission data version 4.1 derived from the USGS/NASA SRTM data (Jarvis et al. 2008). ASTER GDEM Version 2 (The Version 2 of Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Global Digital Elevation Model (GDEM)), released jointly in October 17, 2011 by the Ministry of Economy, Trade, and Industry (METI) of Japan and the United States National Aeronautics and Space Administration (NASA)[16].

Algarve coastline was obtained from NOAA National Geophysical Data Center[17], GSHHG Version 2.2.0 (Wessel and Smith, 1996).

The Guadiana river basin was obtained from CCM River and Catchment Database[18], version 2.1 (CCM2) of 2007, created by Joint Research Centre (JRC) of European Commission (Vogt et al., 2007).

4.2.2. Data processing

In estuarine modelling, the quality of bathymetry is essential to ensure good model results, therefore GIS tools were used to pre-process the model spatial inputs.

After collecting available bathymetry data, the datasets with water depth values were re-referred to the Portuguese Hydrographic Zero which is 2 m below the mean sea level (measured in Cascais). In order to obtain bathymetry values in the vertical coordinate system required by MOHID, positive values were directed to the deep water and negative values upland. All the datasets in xyz text files of different structures were converted into shapefiles.

It was assumed that the along-river slope (slope of the water surface) for the estuary is zero, and everything is referred to the HZ. The real river slope along the Guadiana estuary is very small (between the mouth and Alcoutim it is about 20 cm). Because the estuary is deep and calm, the water surface and bed elevation slightly increase upstream. Only after the cascades and rapids above Mertola (where the estuary actually ends) the river slope starts increasing and water surface becomes significantly above the ocean level.

The bathymetry data points also had to be transformed into one coordinate system, clustered in a case of too dense point distribution, and joined into one dataset for future interpolation.

[15] *http://srtm.csi.cgiar.org*
[16] *https://lpdaac.usgs.gov/products/aster_products_table/astgtm*
[17] *http://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html*
[18] *http://ccm.jrc.ec.europa.eu*

Coordinate conversion

The projected coordinate system WGS 1984 UTM Zone 29 was selected for several reasons. First, the direct transformation parameters between some old, rarely used and national Portuguese datums are not developed[19], and transformation between them usually goes through WGS 84 datum (or ETRS) as double transformation: Datum 1 –> WGS 84 –> Datum 2 (for instance, as in the Proj.4 library). But each transformation produces an error, and an additional transformation accumulates and increases this error. Second, WGS 84 is the most widely used datum nowadays, so the data can be easily published in the Internet or shared with other people. Finally, the projected coordinates were preferred to geographic ones, because visualization of the data and model results is better perceived when the distances are in meters.

The data from different sources were integrated with transformation of different initial coordinate systems into WGS 84 system using ArcGIS 9.3. The chosen datum transformation methods are presented in the table 4.1. The seven-parameter methods were preferred because they are more complex and usually more accurate.

Table 4.1. Transformation of coordinates.

Coordinate system	**Type**	**Data**	**Datum transformation method**	**Method name in ArcGIS**	**Method type**
Datum 73 Hayford Gauss IPCC	projected	Bathymetry submerged delta, Orthophotos 2005	Coordinate Frame (Bursa-Wolf method)	Datum 73 To WGS 1984 2	seven-parameter
Lisboa Hayford Gauss IGeoE	projected	Bathymetry tributaries, Old calibration data, Flow stations	Coordinate Frame (Bursa-Wolf method)	Datum Lisboa Hayford To WGS 1984 2	seven-parameter
Lisboa Hayford Gauss IPCC	projected	Bathymetry lower and middle estuary (EMERGE)			
ETRS 1989 PT TM06	projected	Orthophotos 2010	Position Vector (Bursa-Wolf, opposite rotation) – zero valuess	ETRS 1989 To WGS 1984	seven-parameter
ED 50 (Spain and Portugal)	geog-raphic	Ayamonte calibration data	Geocentric Translation	ED 1950 To WGS 1984 13	three-parameter
WGS 1984	geog-raphic	Bathymetry ocean, Bathymetry IH, Simpatico calibration data, DEMs, Watershed	–	–	–
WGS 84 UTM Zone 31	projected	Bathymetry lower estuary	(Reprojection)	–	–

[19] *http://www.igeo.pt/produtos/Geodesia/Inf_tecnica/parametros_transformacao/parametros_Portugal.htm*

Aggregating bathymetry (clustering and joining)

The available bathymetric data were complemented with Sonar bathymetric surveys. This surveying method produces extremely large amounts of data. All bathymetry data had about two millions of points and the distances between data points in some datasets were about 20 centimetres, which is very redundant for the model. Processing and interpolation of such a large dataset takes much computer time and memory. MOHID is not able to handle it at all.

Thus, techniques were needed to cluster the information, prior to its interpolation. The measured points were firstly separated using ArcGIS 9.3 software into groups by a regular net of hexagons using the *Spatial join* function. For creation of the hexagons as a shapefile, a script in Python 2.5 was developed (Appendix 1). The diameter was chosen 2 m, being several times smaller than the smallest grid cell but about 10 times larger than average distance between Sonar points. The hexagon pattern was chosen in order to avoid complete coincidence of element borders with sequences of points along ship tracks (figure 4.7). Common regular patterns like squares, rectangles, or triangles have straight-line borders and can cause this problem.

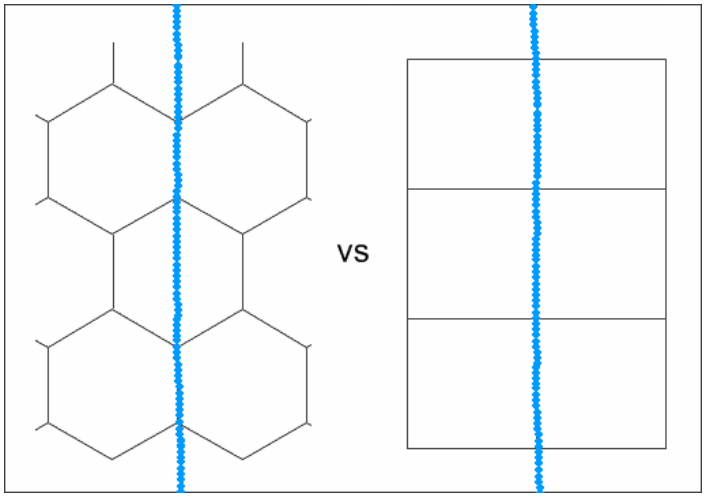


Figure 4.7. Comparing pattern types.

Then average x, y, and bathymetry values were calculated for each group (inside each hexagon) using another Python script (Appendix 2). The developed script allows to avoid spatial outliers before the final averaging. In each hexagon, outliers are points with bathymetry value different from the initial average inside this hexagon more than the tolerance (the tolerance value was set 2 m, the same as the hexagon size). Then average x, y, and bathymetry are computed only using "good" points inside the tolerance borders (figure 4.8) (Appendix 2).

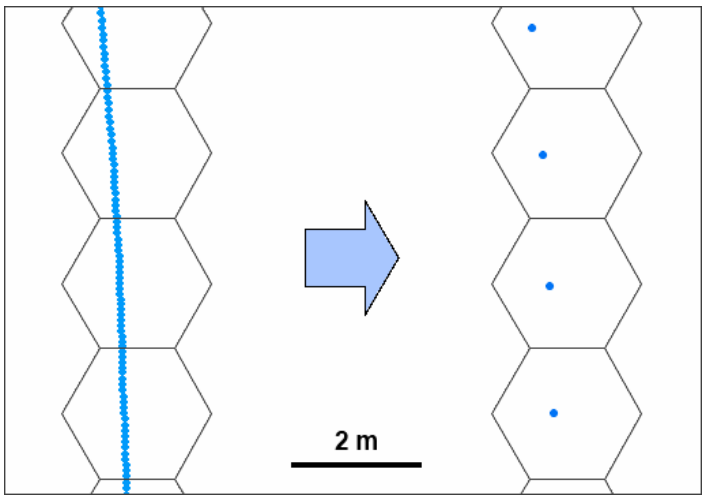


Figure 4.8. Clustering bathymetry (initial data on the left and the result at the right side).

This clustering reduced the number of points to more than ten times than initial datasets.

Bailly du Bois (2011) proposed a methodology to facilitate the construction of gridded bathymetry data for coastal hydrodynamic models. It consisted of the three tasks: 1) selection of records of better quality among overlapping data, 2) elimination of data points located on land, 3) taking into account the shoreline as bathymetric data. These tasks were accounted in the present work.

Finally, all the bathymetry data were combined together into one shapefile (figure 4.9). In the places where two or more datasets overlaid, priority was given to surveyed and recent data over old and gridded data. The final joined dataset had about 130 000 points.

Areas of shallow water adjacent to land are poorly represented in ship-borne bathymetric data. The shoreline can be used as a series of bathymetric records whose elevation corresponds to the mean sea level (or other level according to the origin of the shoreline) (Bailly du Bois, 2011). Then the slope close to the coast will be more realistic after interpolation. Bailly du Bois (2011) advised that the shoreline points should be spaced at a shorter distance than the minimum resolution of the model, but it was found redundant for this work because the range of cell sizes of a curvilinear grid is very large (described in the section 4.2.4). Points along the shoreline were added into the final bathymetry dataset with 0 depth (-2 bathymetry) value. It was done only for the lower estuary where the slope is gentle and bathymetry data is dense (in the upper estuary with very steep slope of river banks and very sparse data points this would induce errors into the interpolation). The points were taken from the shoreline at every 40 m using *Densify* tool[20], which is the common cell size of the computational grid. The procedure of obtaining the shoreline is described in the next section.

[20] *ET GeoWizards extension for ArcGIS, http://www.ian-ko.com/*

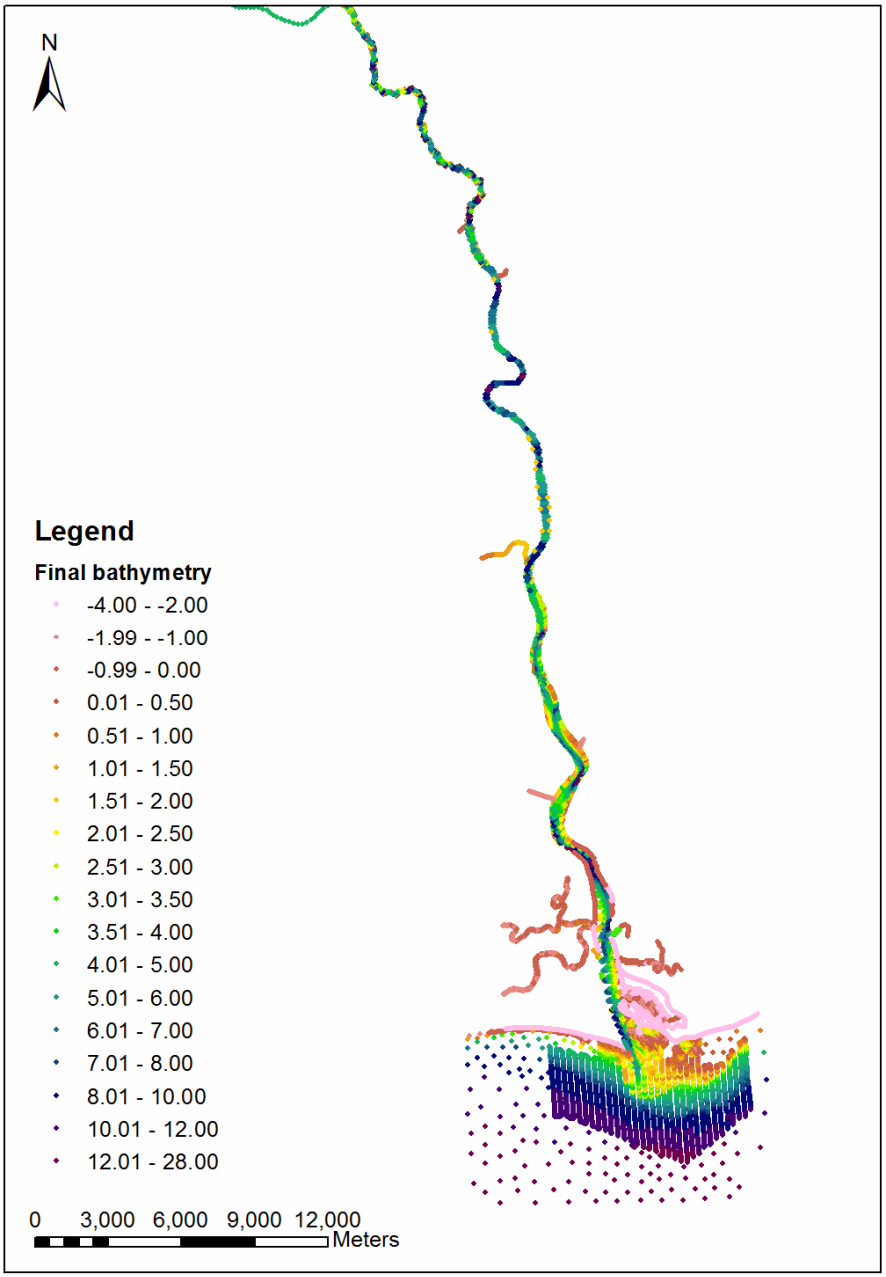


Figure 4.9. Joined bathymetry data from all sources.

4.2.3. Water polygon extraction from orthophoto

Numerical models require a meshing domain polygon which should follow the shoreline, and the opposite land polygon to input non-computing areas.

In the present work the available coastline data from NOAA was rather coarse and didn't enter inside the estuary. Manual digitizing of a long estuary would be very time-consuming. Therefore, an alternative method to obtain the shoreline was needed.

Shorelines can be obtained by manual digitizing (usually from orthophotos), intersecting water level with DTM, extracting from satellite multispectral/hyperspectral or SAR (Radar)

imagery (or LIDAR data), photogrammetry using aerial imagery pairs, video imaging, and with help of different image fusion techniques (i.e. combining data from several different sensors) (Gens, 2010, Li et al., 2001; Lipakis et al., 2008). There are many methods developed for water body and shoreline extraction from satellite imagery, including various classification methods. Shih (1986) firstly used Landsat MSS imagery to delineate the water body from the surrounding land. A throughout recent review and comparison have been done by Nath and Deb (2010) and Gens (2010), and review of early methods by Zhu and Neto (Nd) and Du et al. (2002). But none of the developed algorithms are accepted as universal and most of them are application specific, so precise separation of water from land is still a challenge (Nath and Deb, 2010).

The spectral reflectance of water in visible and especially in infrared bands is very different from the land features (figure 4.10). The best region in the electromagnetic spectrum covers the near and middle infrared portion of the spectrum (Zhu and Neto, Nd). Thus, existing methods usually require presence of infrared bands in the satellite imagery (Shih, 1986; Frazier and Page, 2000; Nguyen, 2012). Reflected infrared is widely used for water bodies determination, for example, using band ratios or Normalized Difference Water Index (McFeeters, 1996; Qiao et al., 2012). But for a small estuary the image must be of very high resolution and quality, which usually requires an orthophoto. In high-resolution orthophotos the infrared band is usually missing, and sometimes only RGB photos taken by aircraft are available for the modelling area.

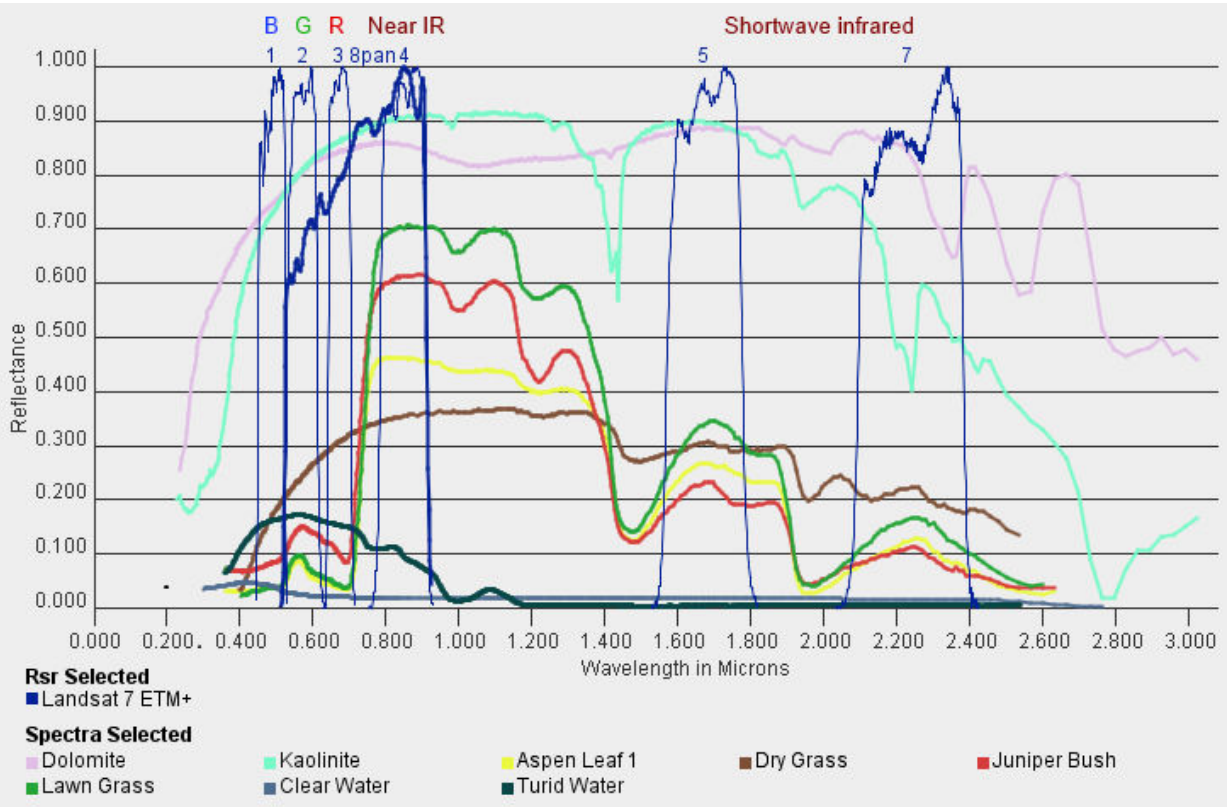


Figure 4.10. Spectral characteristics of some Earth's surface types, and Landsat bands.

Automatic extraction of shoreline from colour orthophotos is not so common and often is a complicated and multi-step process. It can include smoothing and segmentation (Liu and Ramirez, 1997), dividing each land cover class into subclasses “in shadow” and “not in shadow” before classification (Le Bris and Boldo, 2007), using thresholds on calculated greyness, blueness and smoothness values (Tasseled Cap transformation) (Pierce and Law, 2008), segmentation and Delaunay triangulation for edge detection (Sharma et al., 2008), integrating LiDAR point cloud and aerial orthophotos (Lee et al., 2009).

Principal component analysis (PCA) is a very common statistical method in remote sensing for reducing dimensionality of hyperspectral data, and also widely used for identifying features in multispectral imagery as an image processing technique. PCA transforms a set of initial correlated bands into a few uncorrelated bands (principal components) which are easy to interpret (Avena, 1999). Including the principal components generated by PCA into classification together with initial spectral bands have significantly improved classification accuracy for multiple land cover types (Shataee and Najjarlou, 2007; Fisher et al., 2002). There was also experience of including the first principal component into classification, together with Landsat TM bands, NDVI and DEM (Li et al., 2009), and together with three RGB bands of aerial photography (Cleve et al., 2008), for land cover classification.

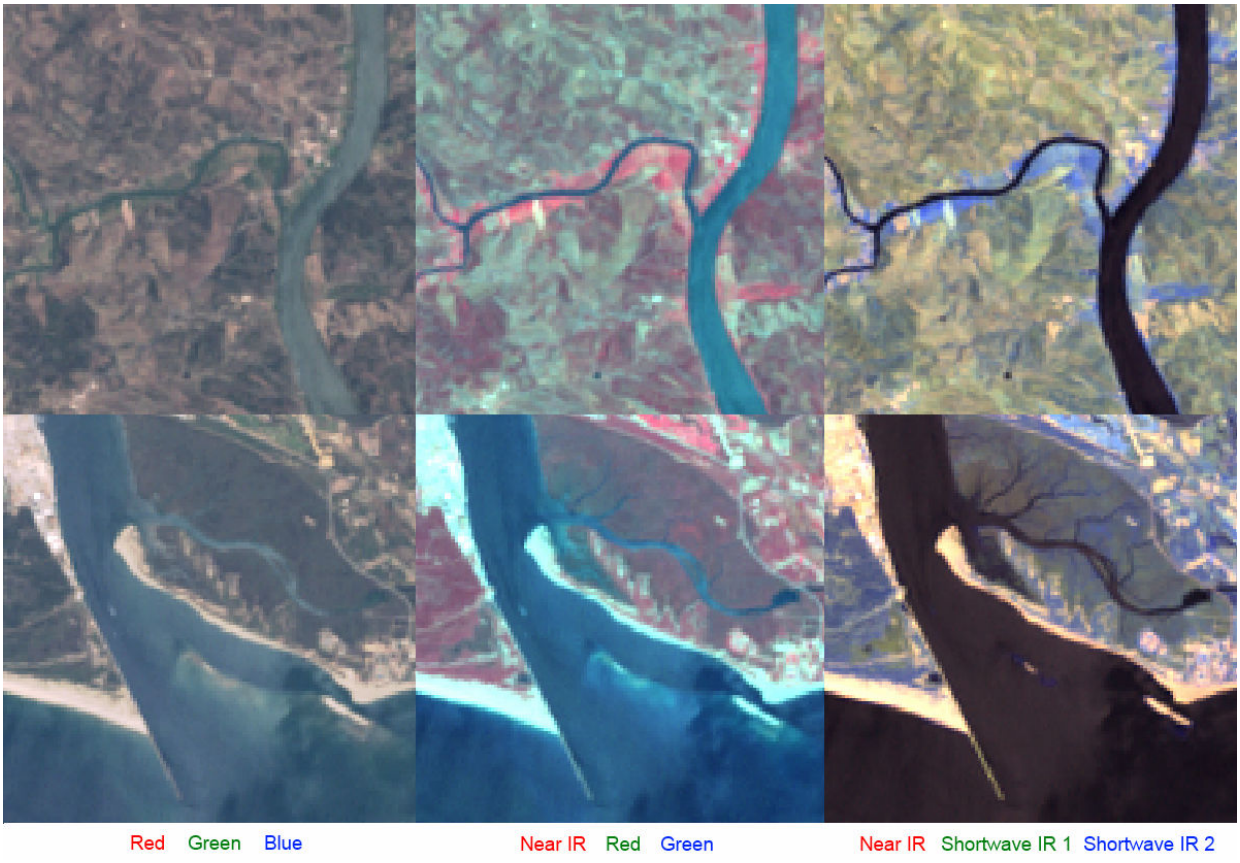


Figure 4.11. Landsat 7 composites for the Odeleite and Guadiana mouth, bands 3,2,1 (left), 4,3,2 (center) and 4,5,7 (right).

The available infrared data for the Guadiana Estuary was coming from Landsat 7 satellite with 30 m resolution (which can be fused with the panchromatic band to 15 m). Infrared bands could help to clearly distinguish the water surface for the entire estuary (figure 4.11). However, the resolution was to low for obtaining an accurate domain for the hydrodynamic model.

In this work the water domain was extracted from an orthophoto by unsupervised classification. Several orthophotos were tested, and finally the image obtained from Google Maps was selected due to several advantages: recent image with clear water, free of clouds, no sun glints, high enough but not too high resolution (4.75 m pixel), taken not at extremely high flood or low ebb.

The best result was achieved with unsupervised classification of the image based on PCA using IDRISI Andes software[21]. Several classification methods (including supervised classification minimum distance to means and unsupervised k-means) were tried for the red, green and blue bands, but they showed very bad, noisy and misclassified results. Then PCA for the three spectral bands was performed. PCA split all the spectral information into the three independent orthogonal variables. Almost all the information (99% of the variance) from the three bands was included into the first principal component which can be explained as colour intensity (table 4.2), because the image bands were highly correlated (figure 4.12).

Table 4.2. PCA result.

Loading	PC 1	PC 2	PC 3
Band 1	0.995042	-0.095441	-0.028001
Band 2	0.997767	-0.023890	0.062365
Band 3	0.991061	0.132534	-0.015253
% var.	98.886068	0.972918	0.141017

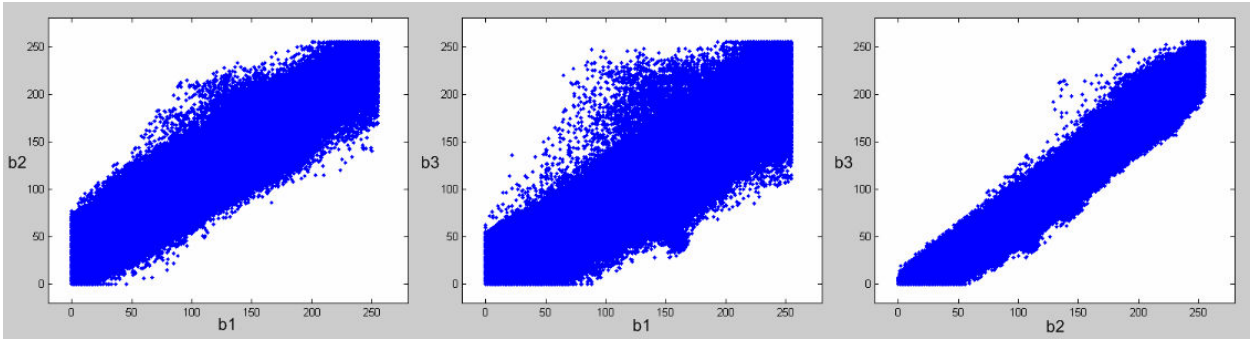


Figure 4.12. The scatterplots of the bands (R vs. G, R vs. B, G vs. B).

[21] *Clark Labs, Clark University, http://clarklabs.org/*

However, some information in the second and third components was related especially to the water and turned out critically important for classification (figure 4.13). After performing PCA, unsupervised clustering based on a histogram peak technique of cluster analysis (the method CLUSTER[22] in IDRISI) was run on the three principal components instead of the initial bands. Twenty classes were obtained and then reclassified as land, water and very shallow water (figure 4.14). The clustering result was much better but other classifications did not improve much.

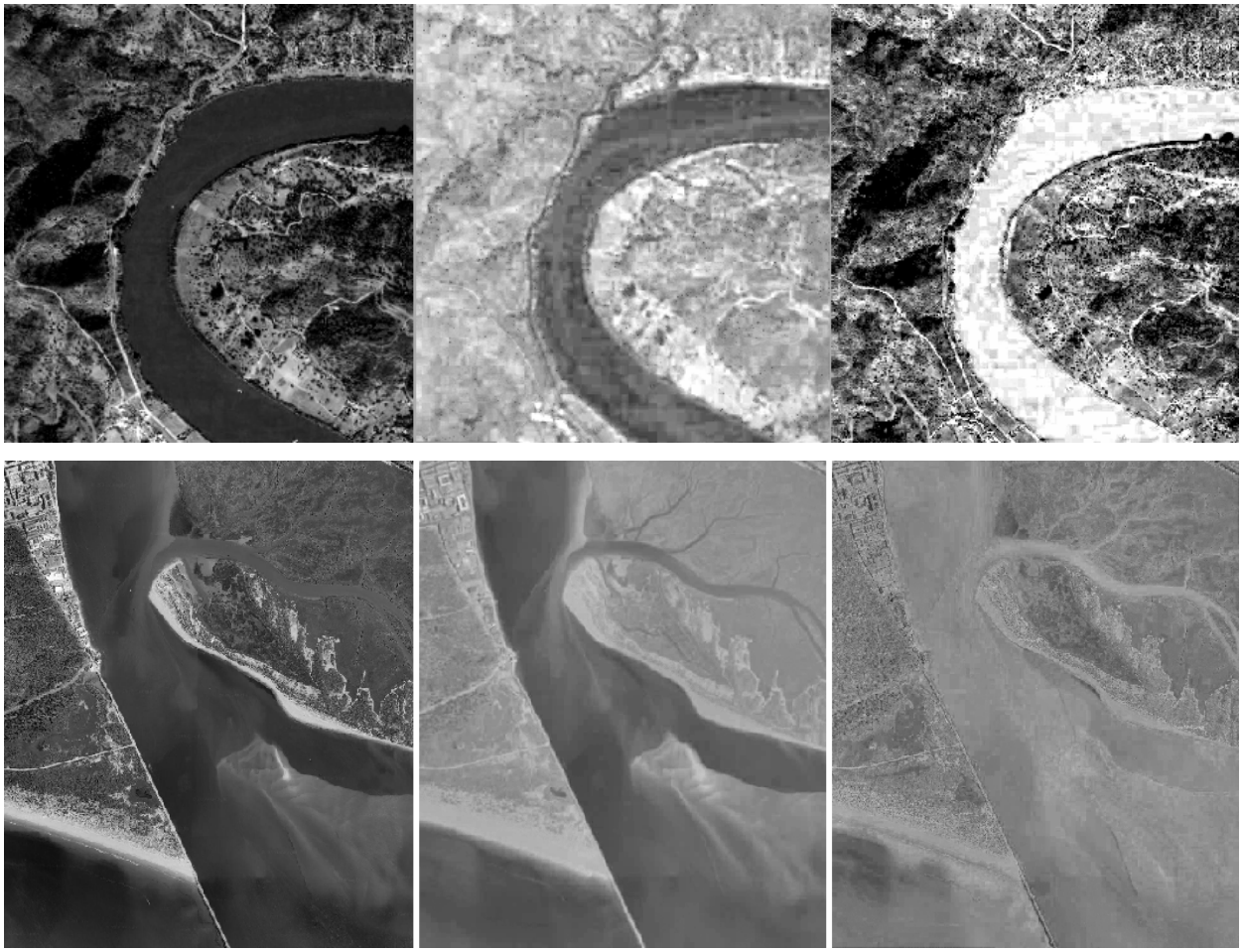


Figure 4.13. The three principal components, from PC1 (left) to PC3 (right).

So, the best land cover separation was found using PCA for cluster analysis, namely due to using the last principal components. Figure 4.15 shows the results in some problematic areas: hills, dirty waters and tidal flats. The possible statistical explanation of this effect is that PCA applies normalization to the data set and this gives all three dimensions roughly equal weight, thus weights of the last components increase. The components are used as equal-weight variables for classification, and the influence of information stored in the last principal components is much higher. But during classification on image bands this important information was negligible.

[22] *CLUSTER performs unsupervised pixel-based classification using a variant of the histogram peak technique.*

Figure 4.14. The final PCA-based classification result.

The classification result was estimated visually, however, the more precise method can be used. Classification accuracy can be tested by comparing a set of known points with the correspondent pixels, and the best method would show the highest percentage of coincidences.

The classified image was vectorized in ArcGIS 9.3 software and processed using ArcToolbox as follows. The amount of small polygons appeared due to pixel-based classification was decreased by the *Aggregate* tool, finally leaving only one large polygon for each class. These polygons were generalized and smoothed using the *Simplify* tool (figure 4.16). The result also needed some manual editing in several places with orthophoto defects (figure 4.16, red square), and in dynamic sandy parts in the mouth according to the most recent orthophoto of 2010 from IGP.

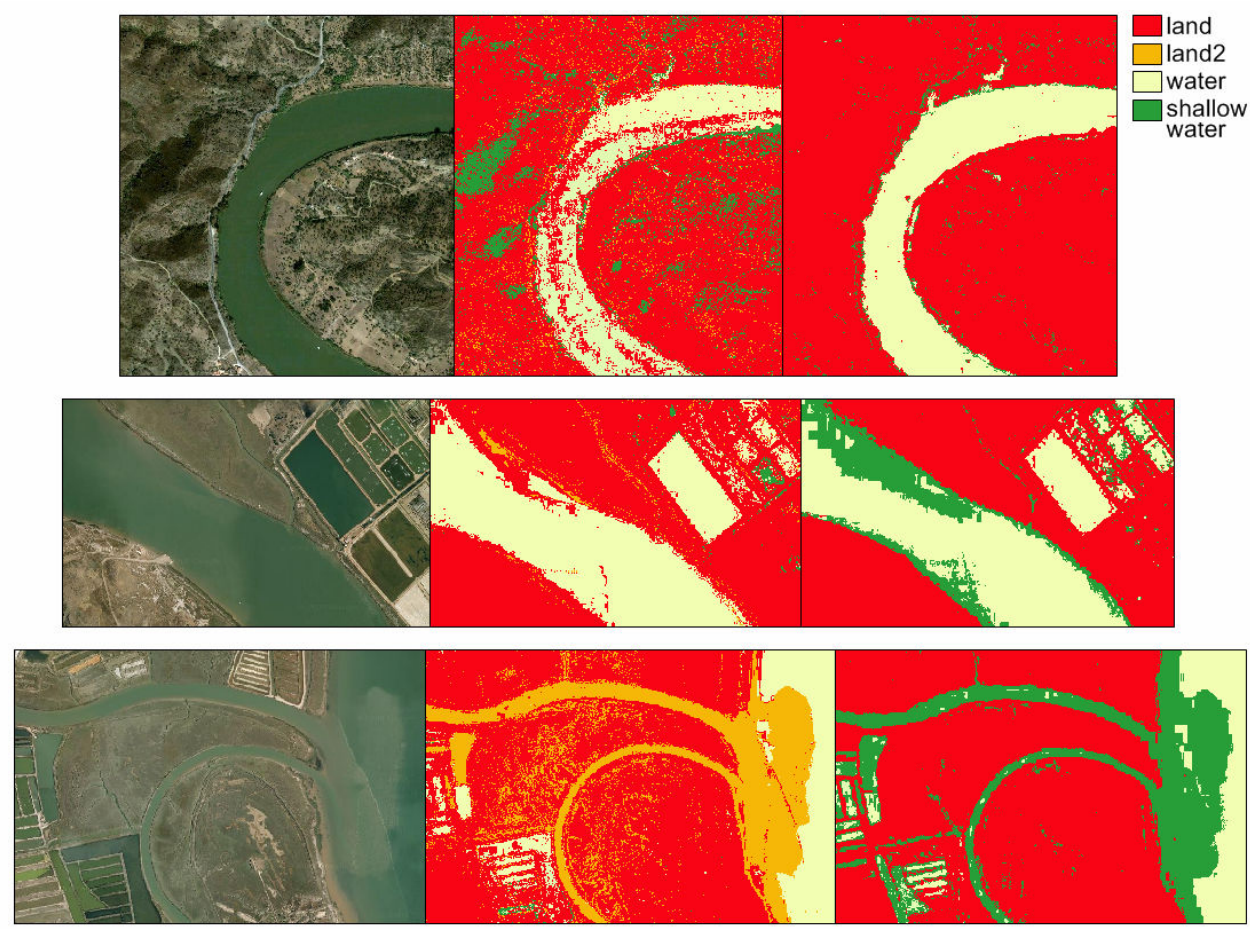


Figure 4.15. Comparing the best classification result using spectral bands (center) and classification on principal components (right).

Figure 4.16. The vectorized water class.

4.2.4. Curvilinear grid creation

MOHID GIS provides a grid generator that produces structured nearly-orthogonal curvilinear grids. The water polygon obtained by classification was imported into MOHID GIS and the curvilinear grid was generated in this domain.

MOHID GIS allows displaying satellite images as a background, and creating and editing vector data (lines, points, polygons) in the MOHID ASCII format. This format presents a list of X,Y coordinates with a specific keyword in the beginning and at the end. A curvilinear grid file contains the list of cell corner coordinates (in the sequential order with the code -9.9e+15 for unexisting vertices), information about georeference (coordinates of the origin, needed mostly for Cartesian grids) and the number of rows and columns (figure 4.17).

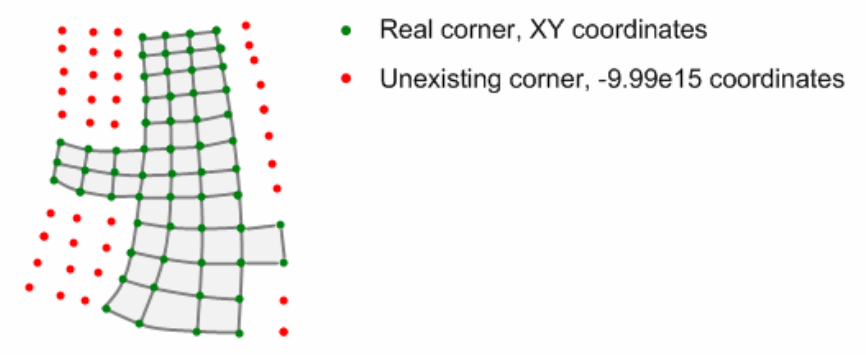


Figure 4.17. Curvilinear grid structure.

To generate a curvilinear grid, the boundary polygon must have a defined topology, namely marked vertices of the polygon representing "corners". Vertices can be Right turn or Left turn, and the number of marked vertices in each part of the polygon must be equal to the number of corners of the corresponding topological rectangle (figure 4.18). Therefore, simple river channel must have only four Right corners, but a branched estuary should have a very complicated corner definition. Islands are not allowed for grid generation.

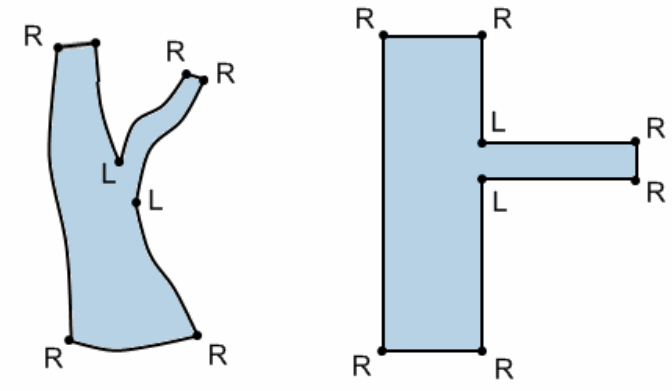


Figure 4.18. Domain polygon topology.

The Guadiana Estuary with its tributaries has a very complex branched shape. MOHID GIS provides the tools for editing and defining corners of the domain polygon, but the decision on how to define these corners to obtain a grid, which would perfectly represent the geometry of the estuary, is more of an art than a science.

Many attempts were performed and many issues resolved before an acceptable good curvilinear grid was obtained for the Guadiana Estuary. The biggest problem was the jetty (breakwater) at the mouth of the estuary. For correct modelling, the grid cells covering the breakwater must be marked as land, but the jetty is very narrow and prolonged far offshore, and the grid needs to describe it precisely. Figure 4.19 shows the main steps of improving the domain polygon and the resulting grid. The first grids had described the breakwater shape very roughly, and only the last attempt produced the cell size comparable to its width and the grid direction following it.

Among other difficulties, representation of the tributaries, ports and tidal flats was resolved (figures 4.20, 4.21).

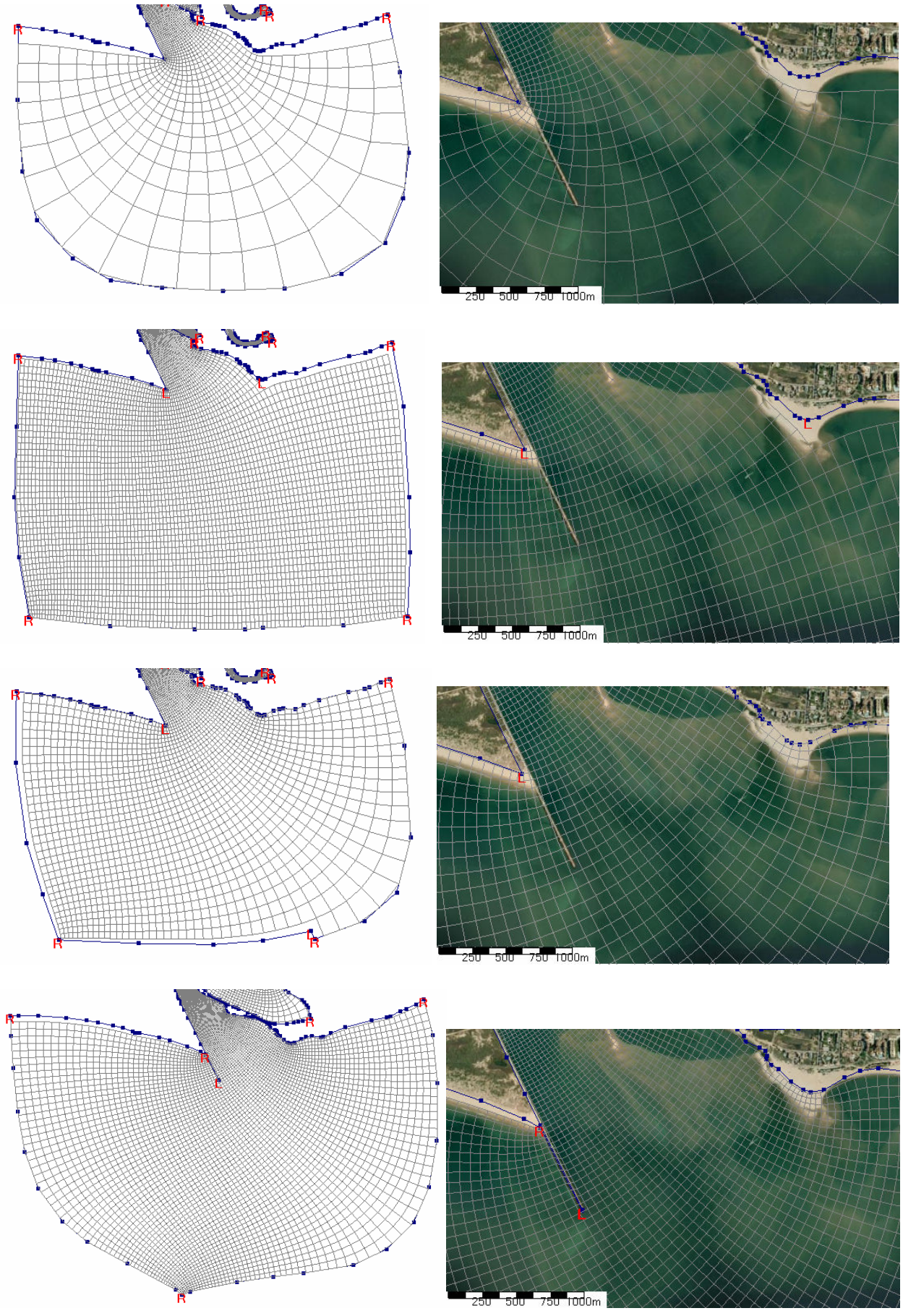


Figure 4.19. Domain polygons and grids.

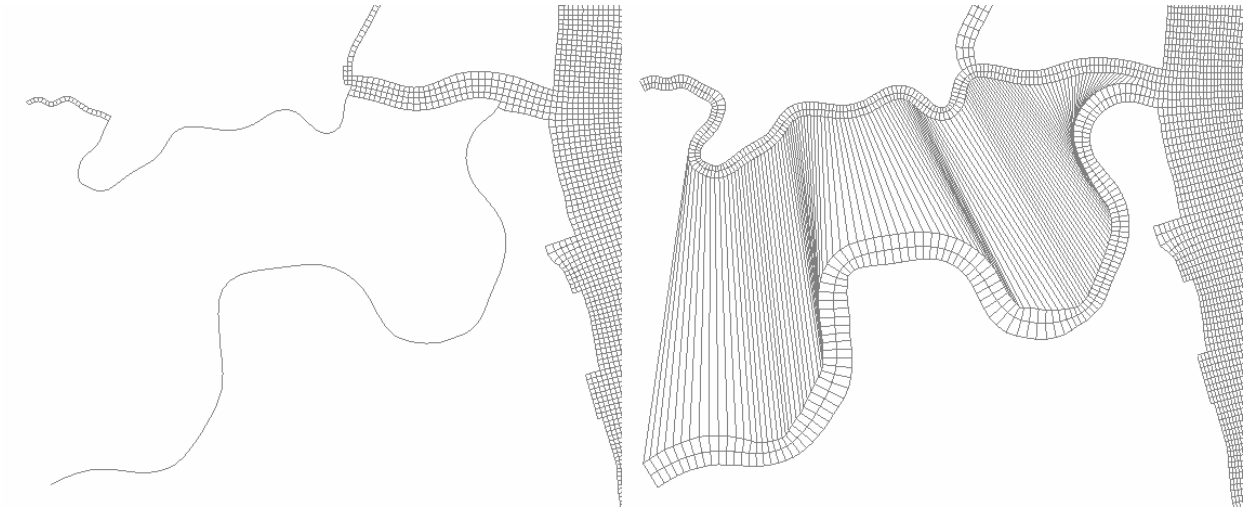


Figure 4.20. Errors in tributaries.

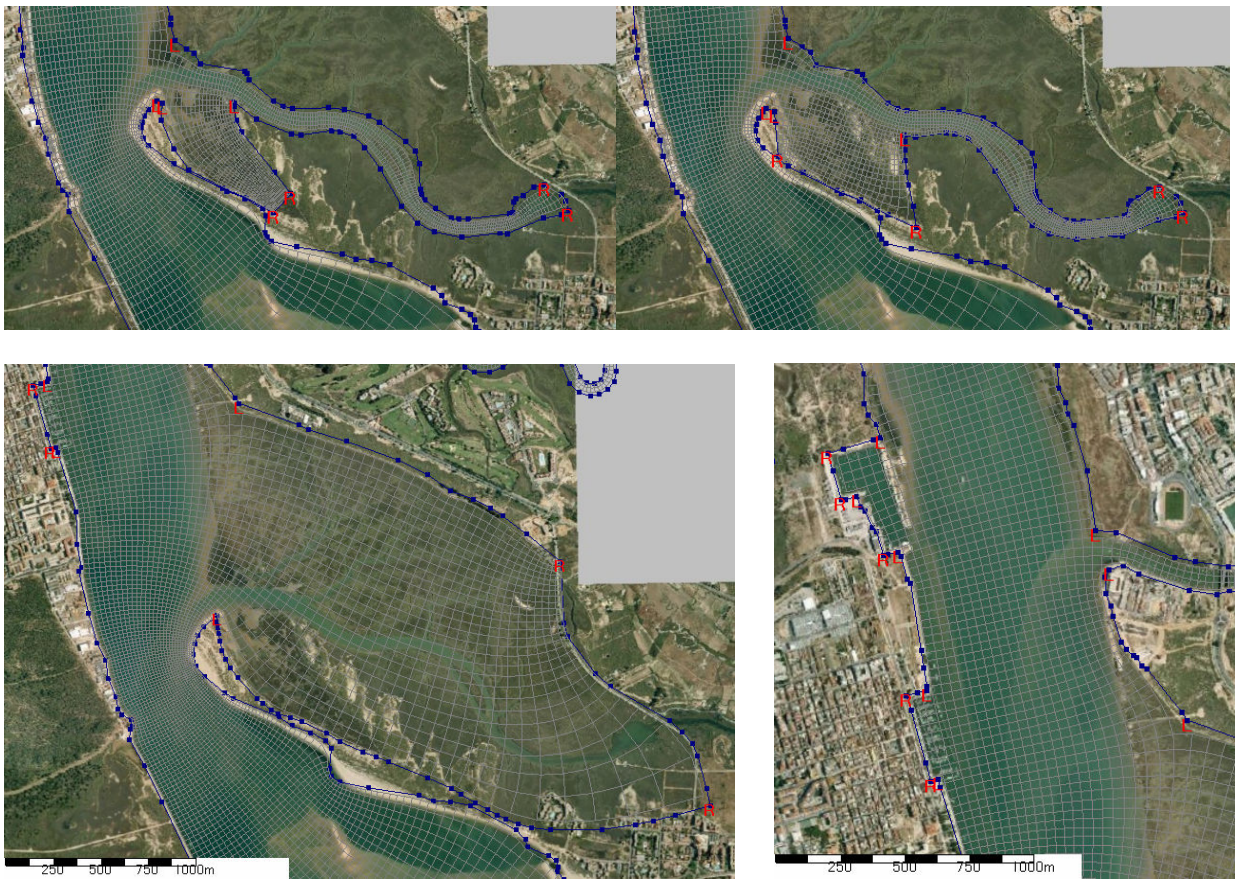


Figure 4.21. Tidal flat and ports.

The final curvilinear grid is presented in figure 4.22 and its topology in figure 4.23.

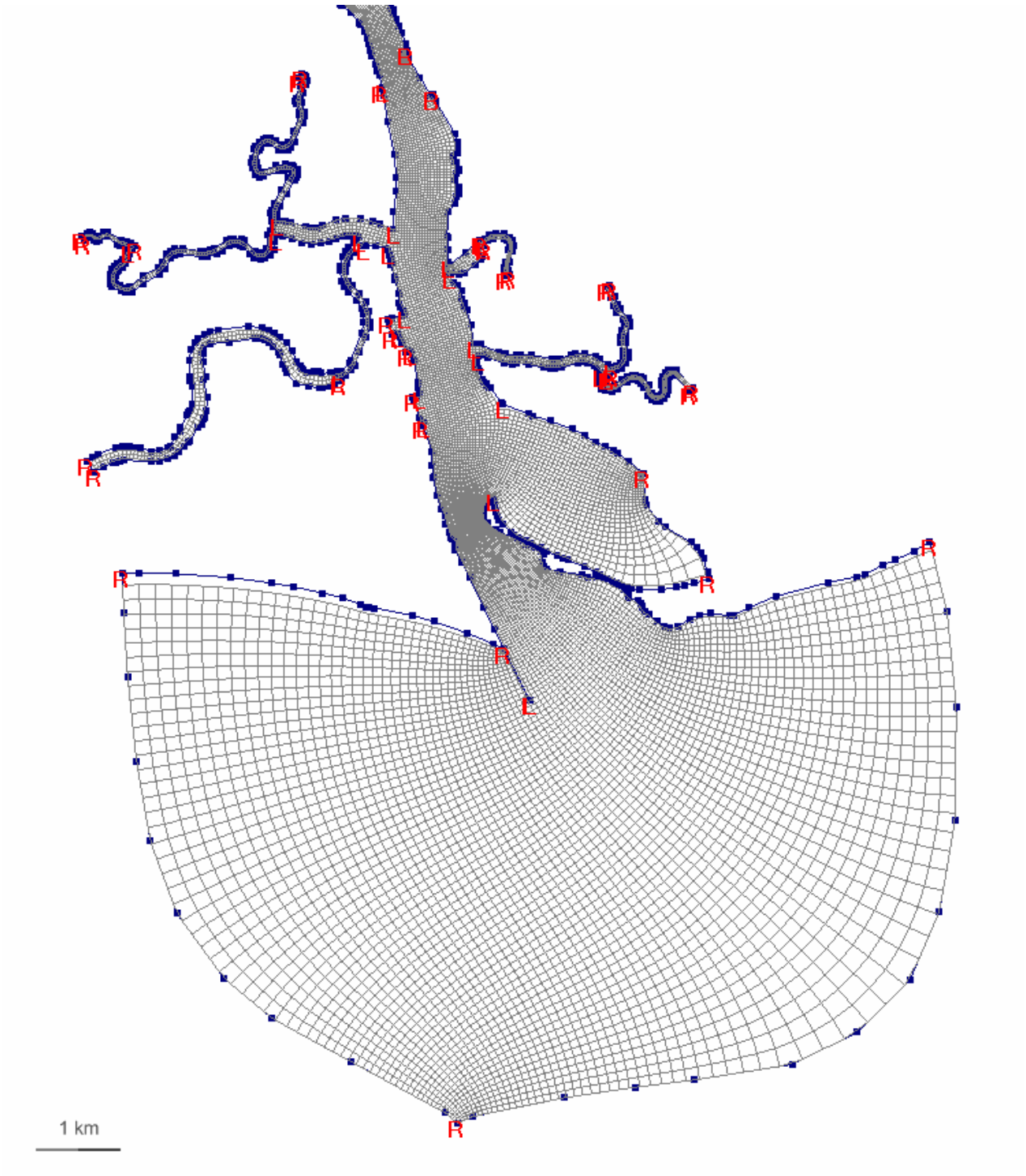


Figure 4.22. The final grid (the lower estuary).

The final grid dimensions were 2209×122 cells and the cell size varied from 10 m to 70 m inside the estuary and up to 300 m at the outer submerged delta.

The grid obtained was used to produce a gridded bathymetry using interpolation of data points into grid cells, which is described later (section 4.2.6).

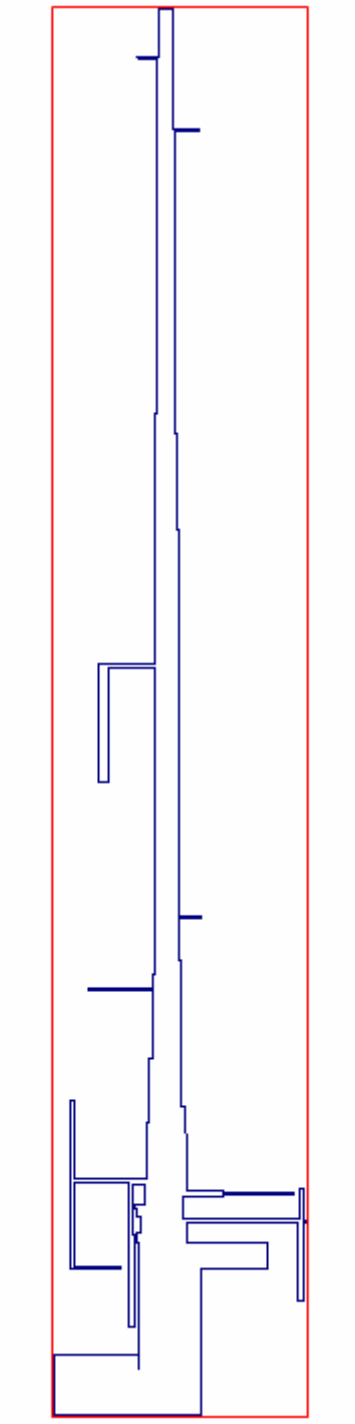


Figure 4.23. The final grid topology.

4.2.5. Bathymetry estimation from orthophoto

Sonar ship-borne surveys always display gaps in the shallow parts of the estuary. Such areas are very dynamic and often experience quick and significant changes in bottom shape, so acquired data become old rather quickly. Since the water is usually clear and the depth is small in those areas, sunlight reflected from the bottom can be detected by a satellite or aircraft. The recorded light intensity depends on the water depth, attenuation coefficient for its wavelength in

this water, and the reflection coefficient of the bottom (Benny and Dawson, 1983). Thus, it is possible to extract water depth information from satellite images (or orthophotos) of clear shallow areas.

This possibility has been examined by many authors for about 40 years (Lyzenga, 1978; Clark et al., 1987; Benny and Dawson, 1983; Stove, 1985; Stumpf et al., 2003). A review of studies of passive remote sensing of shallow-water bathymetry has been done by Yarbrough and Easson (2003), and evaluation of different algorithms by Baban (1993) and Bramante et al. (2010).

According to the Beer-Lambert Law, the intensity of light at the water depth d (m) is:

$$L = I\, exp(-dK)$$

where I is the intensity of the incident light and K is the attenuation coefficient which varies with wavelength (and water quality). So there is the exponential decrease in radiance with depth.

Linear Band algorithm for bathymetry modelling from remote sensing data was developed by Lyzenga and improved by Clark, and was based on the Beer-Lambert Law.

Lyzenga (1978) showed that the relationship of observed reflectance (or radiance) to depth and bottom albedo could be written as:

$$R = R_{deep} + (A_d - R_{deep})\, exp(-kZ)$$

where R is reflectance below the ocean surface at any wavelength, R_{deep} - is reflectance of an infinitely deep water, A_d is the bottom albedo, Z is the depth, and k is the sum of the upwelling and downwelling diffuse attenuation coefficients of light (Lyzenga, 1978). Deep water was subtracted in order to remove the water surface reflection, water-column contributions and atmospheric scattering.

Similarly, Clark et al. (1987) used the Beer-Lambert Law for creating a bathymetry model for Landsat data. The radiance in wavelength band i at depth Z was:

$$L_i = L_{ideep} + c_i\, R_{ai}\, exp(-2k_i\, Z_i)$$

where L_i is the radiance value in band i, L_{ideep} is the average signal over deep water, c_i is a constant that is a function of several optical parameters (solar irradiance, atmospheric and water surface transmittance, and water surface refraction), R_{ai} is the bottom reflectance in band i over bottom type a, and k_i is the attenuation coefficient (Clark et al., 1987).

Then this model can be inverted to find depth:

$$Z = 1/k \cdot [ln(A_d - R_{deep}) - ln(R - R_{deep})] \quad \text{(Lyzenga, 1978)}$$

$$Z = ln(c_i\, R_{ai})/2k_i - ln(L_i - L_{i\,deep})/2k_i \quad \text{(Clark et al., 1987)}$$

However, changes in the bottom reflectance or water attenuation caused errors when only one band was used. The bottom can have a large effect on the reflectivity, especially for small optical depths (Gordon and Brown, 1974). Lyzenga (1978) attempted to account for bottom type variability by using multiple spectral bands and a rotational matrix (similar to PCA). The radiances were log-transformed to create a linear relationship between the radiance and the depth. But this algorithm did not account differences in water quality.

So, to account the variation of the bottom reflectances, two or more bands were needed (Clark et al., 1987; Lyzenga, 1985). Using two bands 1 and 2 the depth was modelled by Clark et al. (1987) as:

$$Z = [1/2(k_1 - k_2)] \cdot [\ln(c_1 R_{a1}/c_2 R_{a2}) + \ln(L_1 - L_{1deep}) - \ln(L_2 - L_{2deep})$$

And for several bands:

$$Z = \sum w_i\ (1/2\ k_i) \cdot [\ln(c_i R_{ai}) - \ln(L_i - L_{ideep})] \quad \text{(Clark et al., 1987)}$$

where w_i are weights and $\sum w_i = 1$.

The assumption was that the ratio $c_1 \cdot R_{a1}/c_2 \cdot R_{a2}$ remains constant for all bottom types (Polcyn et al., 1970). And if this ratio is constant, then there are constants s_i and b independent from the bottom type a (Paredes and Spero, 1983), so:

$$(c_1 \cdot R_{a1})s_1 \cdot (c_2 \cdot R_{a2})s_2 \cdot (c_3 \cdot R_{a3})s_3 \ldots = b$$

Then equation for depth Z independent of bottom types was:

$$Z = (1/2 \sum s_i k_i) \cdot [1 - s_1 \ln(L_1 - L_{1deep}) - s_2 \ln(L_2 - L_{2deep}) \ldots] \quad \text{(Clark et al., 1987)}$$

And for n bands it might be written as:

$$Z = a_0 + a_1 \cdot X_1 + a_2 \cdot X_2 + \ldots + a_n \cdot X_n$$

where $X_n = \ln(L_n - L_{n\ deep})$, coefficients a_0, a_n are constants independent of the bottom type (Clark et al., 1987), accounting the attenuation coefficients (a_0 *is proportional to* $1/2k_i \cdot \ln(c_i R_{ai})$, *and* a_n *to* $-1/2k_i$). This simple equation corresponds to a statistical equation of multiple linear regression. The coefficients were determined empirically using a set of measured depths.

Lyzenga (1985) also showed that two bands could provide a correction for albedo and created linear equation for two channels i and j:

$$Z = a_0 + a_i \cdot X_i + a_j \cdot X_j$$

where $X_i = \ln[R_i - R_{ideep}]$ and a_0 is the offset for a depth of 0 m (Z = 0). The constants a_0, a_i, a_j, R_{ideep} and R_{jdeep} were determined empirically from multiple linear regression. This algorithm

was updated by Lyzenga et al. (2006) to account for water quality heterogeneity, finally modelling depth as:

$$Z = h_0 - \sum h_j \cdot X_j$$

where h_0 and each h_j are constants defining a linear relationship between X_j and depth. Variables h_0 and each h_j were again determined through multiple linear regression between a set of known depths and the log-transformed radiances at those depths (Lyzenga et al., 2006)

The Linear Band algorithm was slightly modified by Bramante et al. (2010). The log-transformed radiance was defined as in Stumpf et al. (2003):

$$X_j = ln(n \cdot R_j)$$

where $n = 1000$. The value of n is chosen to be sure that the logarithm will be always positive and that the ratio will produce a linear response with depth (Stumpf et al., 2003). Subtraction of deep-water signals was not necessary when atmospheric corrections have been made (Bramante et al., 2010).

Another algorithm called **Linear Ratio** transform was developed by Stumpf et al. (2003). The ratio method requires only bands with different water absorbtion:

$$Z = a_1 \frac{ln(nR_i)}{ln(nR_j)} - a_0$$

where a_1 is a tuneable constant to scale the ratio to depth, n is a fixed constant for all areas, and a_0 is the offset for a depth of 0 m (Z = 0). The band ratio method helped to get correct results over variable bottom types. The ratio algorithm does not require subtraction of deep water and has only two empirical coefficients to be tuned using known depths (a_1 and a_0), and has superior depth penetration for clear water. But there is an increased level of noise.

The determination of R_{deep} is problematic due to variations of scattering and absorbtion, and it can vary throughout a scene (Stumpf et al., 2003). In the Linear Ratio method band ratios can be chosen based on calculated correlation of band pairs with known depth (Bramante et al., 2010).

Benny and Dawson (1983) developed another model that incorporates the attenuation of light as a function of depth, where they subtracted digital numbers of pixels from digital number of deep and shallow water (also using logarithm).

The Stove (1985) model used a unitless parameter relating the optically deep and optically shallow waters (using digital numbers of pixels):

$$VR = 255 \, (V_{deep})^2 / (V_{shallow})^2$$

Then VR was plotted against depth, z, and a regression was calculated.

There is also a physics-based hyperspectral remote-sensing reflectance model for shallow water which does not require calibration measurements (Lee et al., 1999; Lee, 2010).

The Linear Band method was found the most accurate by Bramante et al. (2010) and almost the best (after the Benny and Dawson's method) by Baban (1993).

In general, all models assume the Beer-Lambert's Law. Models with high spatial resolution (<10m - 30m) imagery provided more accurate results than those using low resolution (>200m) imagery (Yarbrough and Easson, 2003). However, all those models have several parameters to be tuned empirically using some known bathymetry points. These tuneable coefficients partially represent attenuation and other local water properties.

Generally, the relationships between depth and optical signal are sensor and site specific and the coefficients and optical parameters derived for pixels in one scene cannot be applied to another scene (Stumpf 2003, Yarbrough and Easson, 2003; Lee, 2010).

However, the optical signal accepted at the water surface can provide information only about very surface layers. Bathymetry can be estimated correctly only until certain depth. At large depths the bottom is just not visible because the light scattering becomes higher than the reflectance. Gordon and Brown (1974) and Sokoletsky (2005) showed that the deepest depth that might be determined from optical signal (upwelling radiance) was about 3-4 optical depths (that depends on the mean downwelling attenuation coefficient). This is also local specific. Bramante et al. (2010) estimated depth up to 2 m in turbid waters (Secchi depth was also 2 m) from high-resolution image with RMSE about 0.5 m. Lyzenga (1978) found the error quickly increasing at depth >10 m (but less than 1 m until 8 m depth). Lyzenga (1985) calculated depths until 12 m with RMSE 0.7-0.9 m. Clark et al. (1987) reported RMSE 0.95 m for depth <5 m and 1.79 m for depth up to 16 m. Lyzenga et al. (2006) estimated depth until 20 m with RMSE about 2 m.

In the case of Guadiana the band ratios did not correlate with depth better than single bands, so the Linear Band regression method was used.

So, in this work bathymetry was estimated from the orthophoto by simple linear regression using correlation between existing data and spectral band values (digital numbers of pixels). The same image obtained from Google Maps was selected due to high water clarity on this image. The bathymetry points for the analysis were chosen in the clean shallow area, with values

shallower than 8 m depth (6 m bathymetry), where the relation between bathymetry and water colour was strong according to the plots (figure 4.24). The deep water was not subtracted assuming that its contribution can be accounted by the regression coefficients. The digital numbers of pixels were used, like Baban (1993) did.

A script in Python was developed to read georeferenced raster cell values correspondent to the bathymetry points using GDAL and OGR libraries (Appendix 3). The script includes the possibility to read a single pixel containing a bathymetry point, or to compute a median from 9 pixels (the pixel containing point and its 8 neighbours), which can be useful in a case of a noisy image.

Light attenuation exponentially growths with increasing depth. The log-transformation of the bathymetry values was used to achieve linear relationship between bathymetry and color intensity. The simple log-transformation did not give good linear relationship, and it was selected empirically according to the plots (and R-squared and residuals of the resulting regressions):

$$Z = ln(10 \cdot (bathym+2.5))$$

where 2.5 was added to remove negative values up to -2 near the mean sea level (due to the bathymetry reference to the Hydrographic Zero), 10 was empirically chosen as n in $ln(n \cdot R)$ (Stumpf et al., 2003; Bramante et al., 2010).

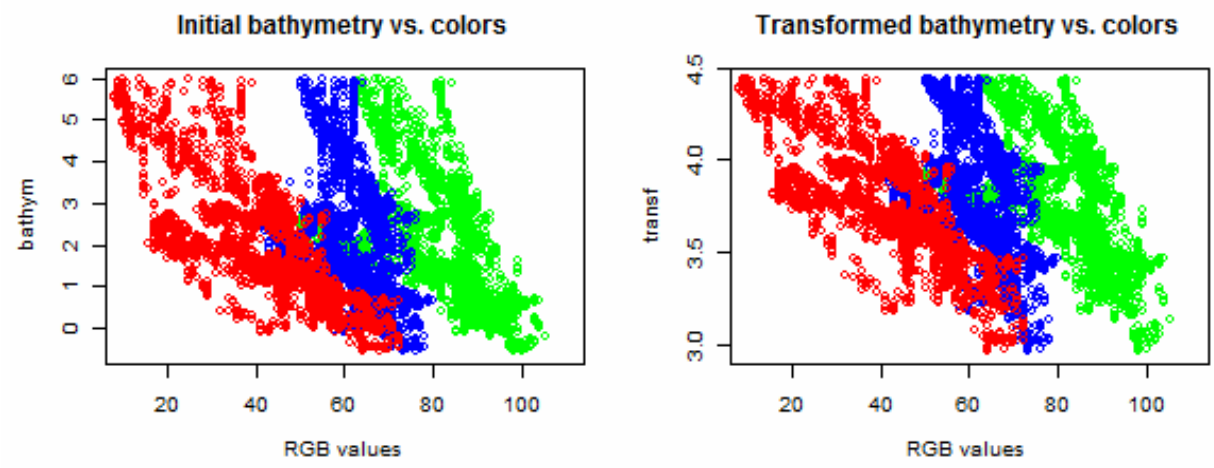


Figure 4.24. Relationships between bathymetry and colour intensity.

The bathymetry values, but not spectral band values, were log-transformed for several statistical reasons. First, band values already had nearly normal Gaussian distribution (figure 4.25) which is required for linear regression, and after transformation they would be skewed. In opposite, the bathymetry values after log transformation had the distribution closer to the normal (figure 4.26). Secondly, with transformation of a response (dependent variable, bathymetry) with fixed predictors (explanatory variables, colours) the R-squared is a proxy for the variance of the

residuals and is trustworthy (but when independent variables are re-expressed then R^2 can be misleading). Actually, for regression mathematical algorithm it doesn't matter what was transformed, and for prediction the best way is that gives the least error and normal residuals.

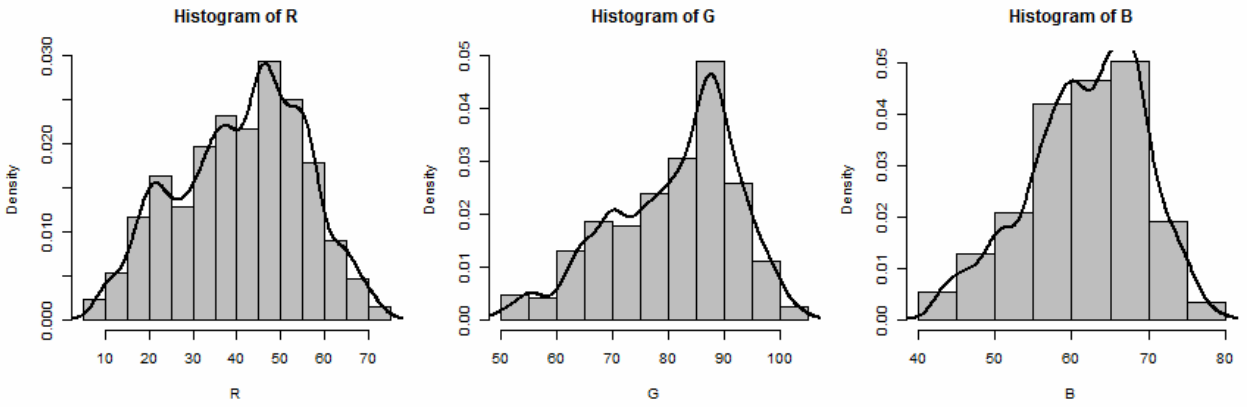


Figure 4.25. Distribution of band values.

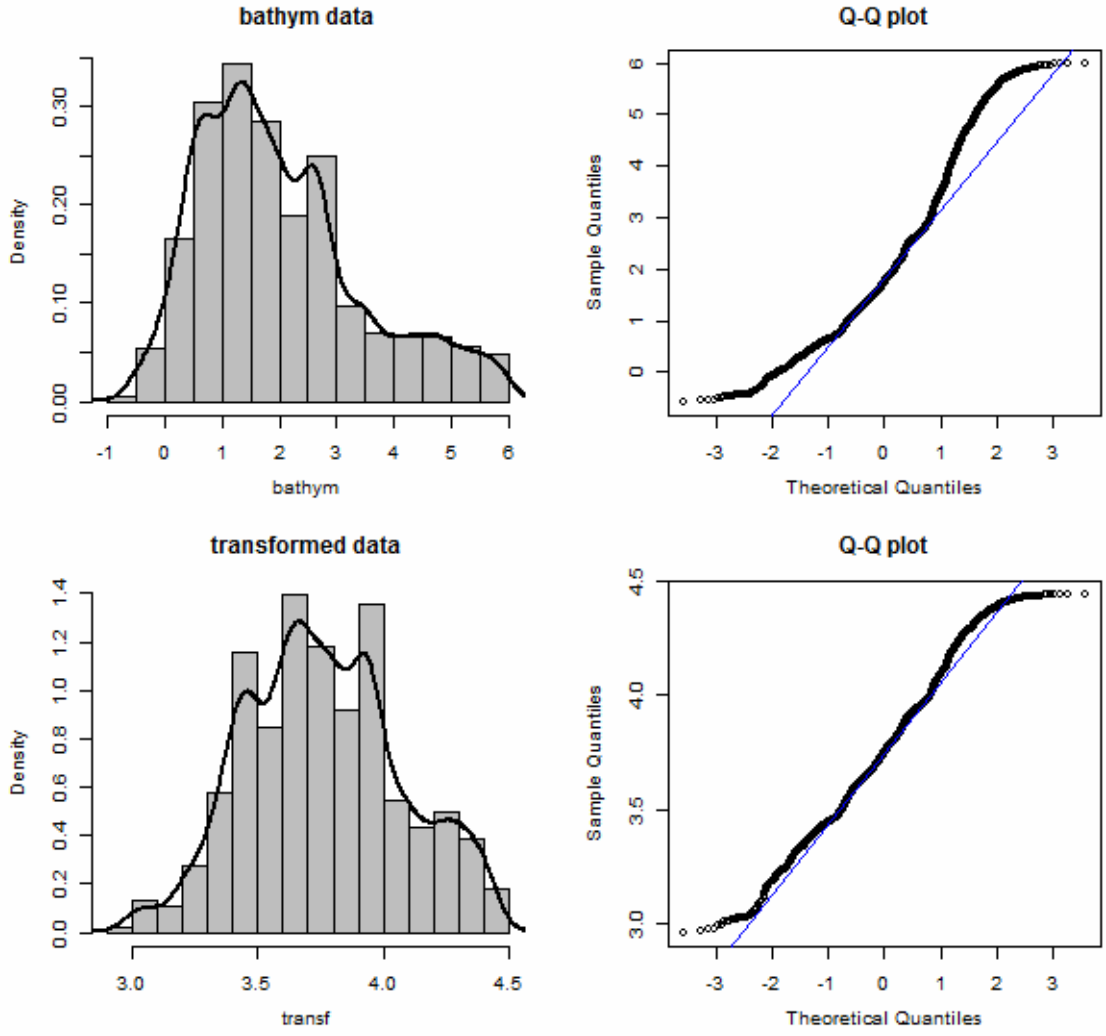


Figure 4.26. Distribution of bathymetry data before and after transformation.

The bathymetry data were separated into two subsets: 75% for regression and 25% for accuracy evaluation. A multiple linear regression was performed for the three bands in the form

of the following equation (Lyzenga, 1978; Clark et al., 1987) using R program 2.11.1 (R Development Core Team., 2010) (figure 4.27):

$$Z = a_0 + a_1 \cdot R + a_2 \cdot G + a_3 \cdot B$$

where R, G, B – red, green and blue digital numbers of pixels, a_0, a_1, a_2, a_3 – coefficients.

Then the result was returned to the normal bathymetry scale by inverting the transformation. The final equation to calculate bathymetry with the obtained coefficients was:

$$bathym = exp(3.247676 - 0.027756 \cdot R - 0.010246 \cdot G + 0.040201 \cdot B) / 10 - 2.5$$

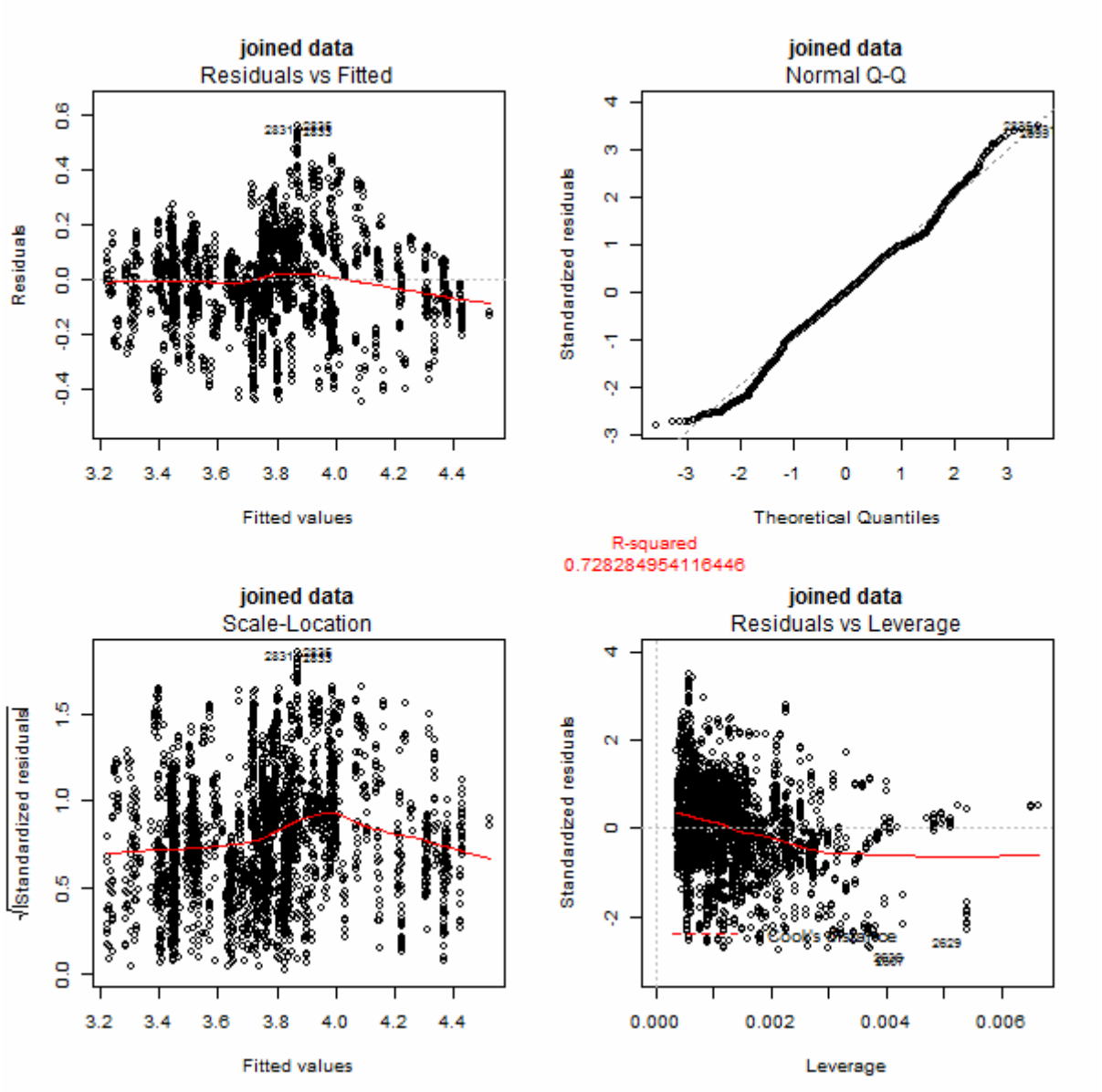


Figure 4.27. Linear regression validation plots.

R-squared of the regression was 0.73. The regression result was used for determining bathymetry in the data-missing areas using Python (Appendix 4). The point locations for estimation, shown in figure 4.28, were selected in very clean shallow areas (marked by red

outline) at the centers and corners of cells of the computational grid. The values of the estimated bathymetry are very close to the topographic zero at the points located almost on the shoreline, which increases the confidence in the method. RMSE for the evaluation subdataset (Appendix 5) was 0.6 m which is a good result for depths in the range 1.5 – 8 m. Depths were predicted in the range 0 – 6.5 m (-2 – 4.5 bathymetry). Since smaller the depth less the error (Lyzenga, 1978), the actual error in the predicted depths is even less than 0.6 m.

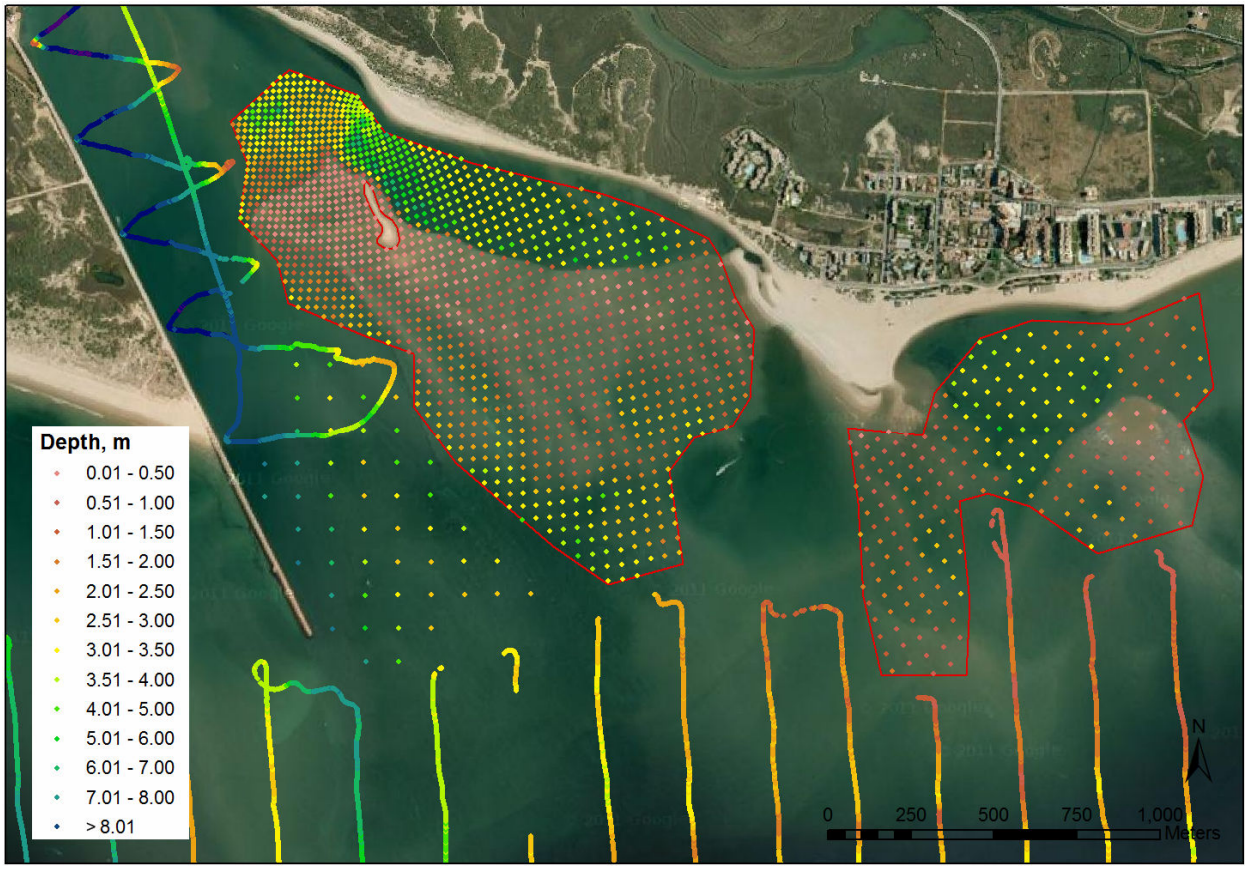


Figure 4.28. Estimated bathymetry.

The estimated bathymetry points were included into the total bathymetry dataset for the MOHID model replacing the old sparse data in this place.

4.2.6. Bathymetry interpolations

The key issues of interpolation and surface modelling for near-shore studies include large heterogeneous data sets, incorporation of anisotropy and break-lines (Mitasova, 2000). Several interpolation methods were tested to find the best suited for the Guadiana Estuary.

Common interpolation methods

There are many articles dedicated to interpolation of bathymetry data. Most of them conclude that the best method is Kriging (Carter and Shankar, 1997; Bernert and Sullivan, 1998; Bello-

Pineda and Hernández-Stefanoni, 2007; Medved et al., 2010). Dost and Mannaerts (2004) preferred kriging because this method gave more control over the interpolation due to the use of a semivariogram model. Kriging[23] is a geostatistical interpolation method that estimates a surface calculating a weighted moving average. It is based on a statistical model that includes autocorrelation and minimizes the prediction error. Kriging assumes that the spatial variation (quantified by the semivariogram) is statistically homogeneous throughout the surface. Kriging is usually an exact method (when semivariogram does not have a nugget effect), so the resulting surface passes exactly through the input points. But kriging is difficult to apply to very large datasets.

Some works suggest Spline with tension (Minimum Curvature) method (Smith and Wessel, 1990; Hell and Jakobsson, 2011; Amante et al., 2010). Spline is an exact bicubic interpolation method which computes values using a function that minimizes surface curvature. The resulting surface is smooth and passes through the input points. A tension factor reduces the problems with artificial extremes between the input points.

A few of works describe local interpolation methods as Inverse Distance Weighted (IDW) (Volakos and Barber, 2001), Natural Neighbor (Dost and Mannaerts, 2008; Ledoux and Gold, 2004) and obviously crude linear Triangulation method (Rogala, 1999), but these methods are proved to produce worse results comparing to global methods (Kriging and Spline). Medved et al. (2010, in Croatian) compared kriging, IDW and minimum curvature, and showed that kriging produced the least error. Bello-Pineda and Hernández-Stefanoni (2007) also compared kriging and IDW for a digital bathymetric model and found that kriging model produced more accurate estimates (reducing the error in 18 % compared with IDW).

There was also a successful attempt to use ANUDEM interpolation method of Hutchinson (1989) for producing bathymetry (Daniell, 2008). Daniell (2008) also derived bathymetry from Landsat satellite imagery to supplement traditionally acquired bathymetric data in shallow waters (using band ratio method), and used SRTM data and coastline for topographic control. Daniell (2008) tried kriging but refused it because it required an excessive amount of time to complete.

Topo to Raster method in ArcGIS is based on the ANUDEM program. ANUDEM (and Topo to Raster) is an interpolation method designed for the creation of hydrologically correct digital elevation models. Since water is the primary erosive force determining the general shape of most landscapes, this method takes advantage of drainage characteristics of elevation surfaces. ANUDEM uses an iterative finite difference technique. It combines the surface continuity of

[23] *developed by Matheron, G. 1963. Principles of geostatistics. Economic Geology, 58, pp. 1246–1266, after Krige, D.G. 1951. A statistical approach to some mine valuations and allied problems at the Witwatersrand, Master's thesis of the University of Witwatersrand.*

global interpolation methods with the computational efficiency of local interpolation methods. The adjustable roughness penalty in ANUDEM is similar to a tension parameter for splines. It includes an optional drainage enforcement algorithm for removing sinks in the output DEM. (Hutchinson, 1989; ArcGIS 9.3 Help; Hutchinson, 1988). Since the ocean bottom is also an elevation surface primary shaped by water, Topo to Raster seems to be a good method for bathymetry interpolation (but without drainage enforcement as Daniell (2008) used).

Actually, kriging is a very good method for interpolating some spatial variables, and Topo to Raster is the best for interpolating realistic ground surfaces.

MOHID GIS includes Triangulation interpolation method. But it does not perform Delaunay triangulation for input points, but constructs triangles just from the centers of the grid cells, assigning values to triangle vertexes as averages in these cells (figure 4.29).

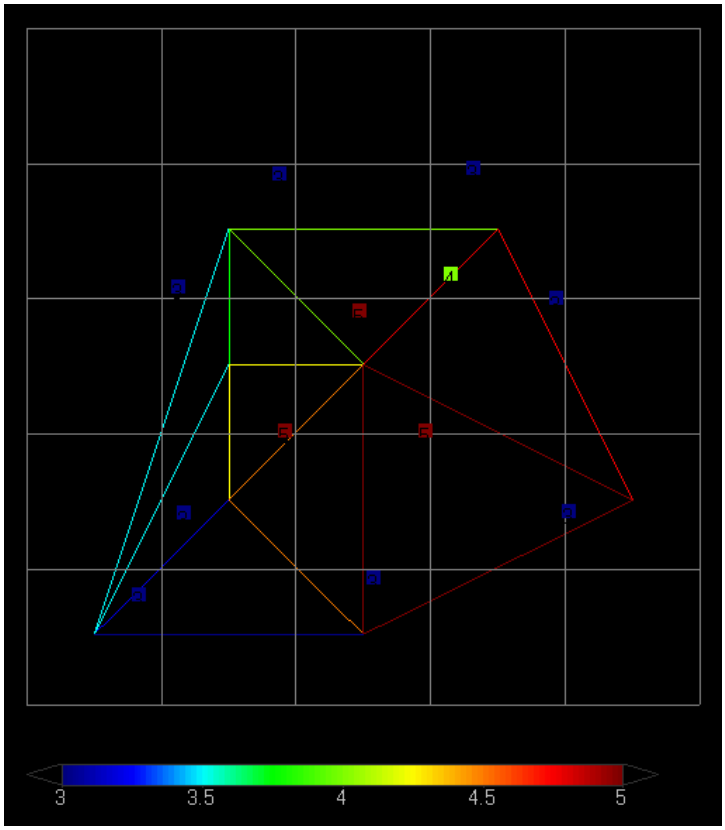


Figure 4.29. MOHID GIS Triangulation.

For interpolation of bathymetry points for the Guadiana Estuary several methods in MOHID GIS 4.9.2, ArcGIS 9.3 and Surfer 10 softwares were tested.

IDW with powers 1-3, TIN (and MOHID's triangulation) and Natural Neighbor interpolations showed not very good, crude and unrealistic results for the estuary. Minimum Curvature, Kriging, and Topo to Raster produced much better results.

Minimum Curvature was performed in Surfer with the internal and boundary tension equal to 1. This tension value showed better result.

Topo to Raster interpolation was performed using Spatial Analyst in ArcGIS with drainage enforcement turned off.

Kriging was performed using Geostatistical Analyst in ArcGIS in the two ways described below.

To prevent loss of information during interpolation, the size of the cell should be less than a quarter of the smallest calculation grid cell used by the hydrodynamic model (Bailly du Bois, 2011). The rasters were interpolated with 5 m cell size (which is only less than a half of the smallest curvilinear cell because the model domain is very large). Then the rasters were overlaid by the curvilinear grid polygons. Using the *Zonal statistics* tool in ArcGIS, the average values of raster cells located inside each curvilinear cell were calculated and then attached to curvilinear cell centers and imported into MOHID as model input bathymetries.

The **first kriging attempt** was a spherical model with Sill 4.5 and Range 200 m, and second order trend removal. Before that, ordinary kriging without trend removal was tried but showed worse result because the data had a trend (figure 4.30).

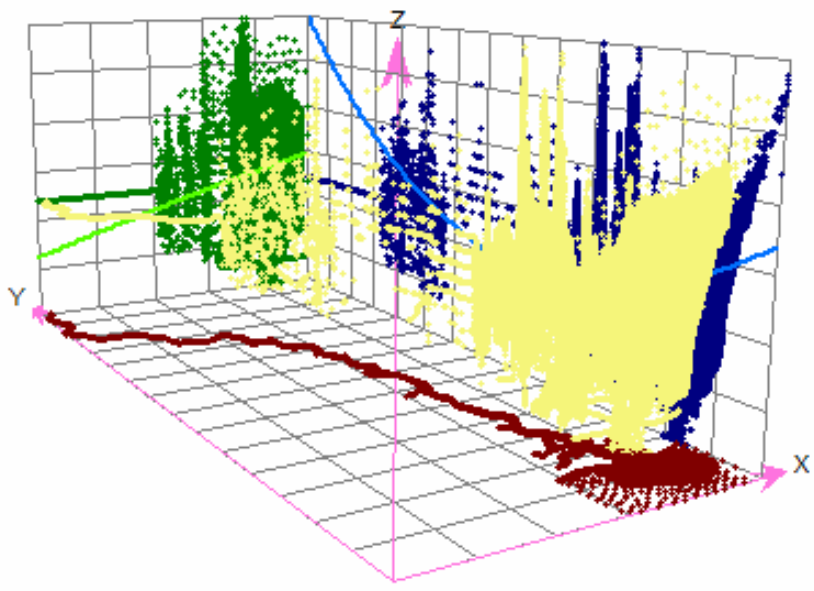


Figure 4.30. Trend analysis. Bathymetry data (yellow) projected on 3D planes.

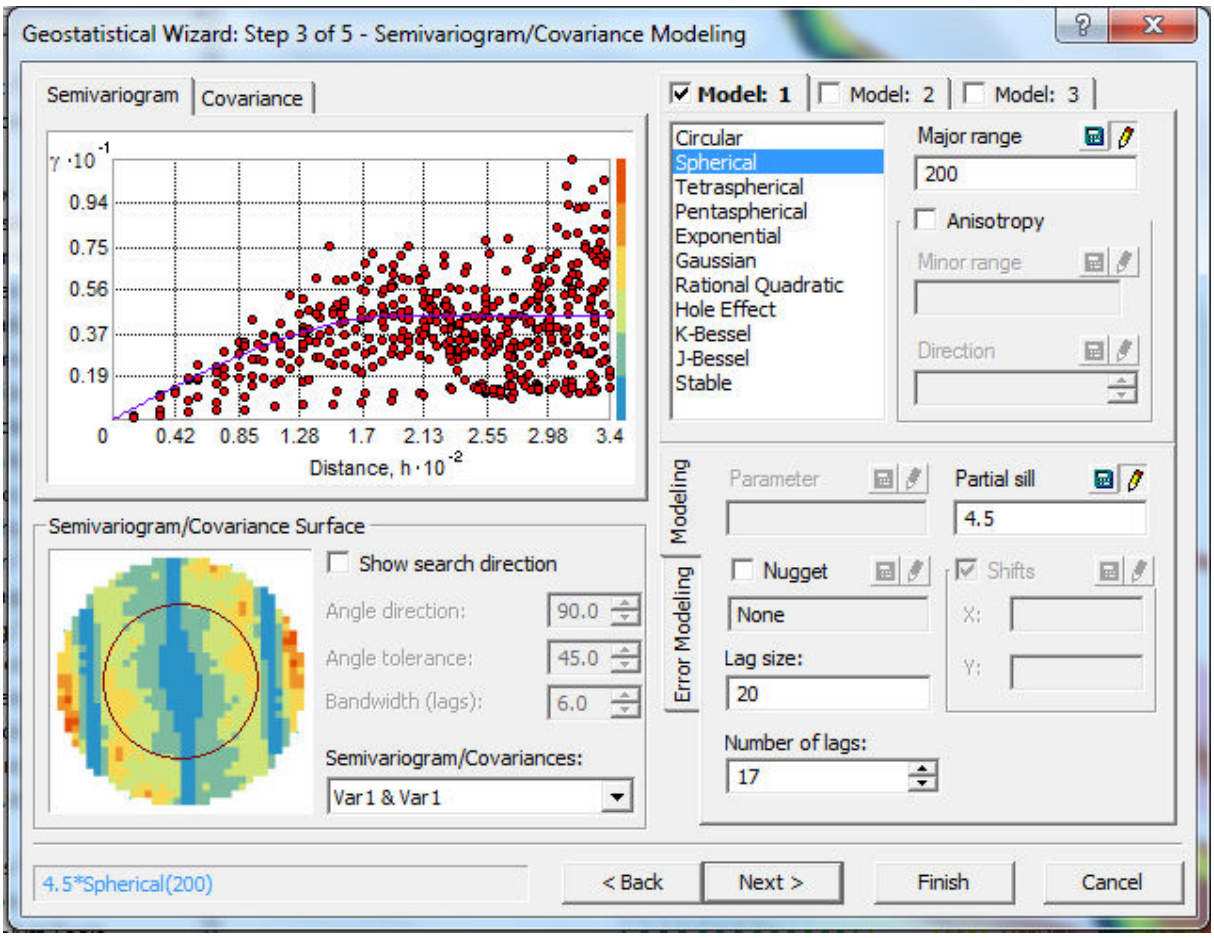


Figure 4.31. Semivariogram for Kriging 4.5*Spherical(200).

Nevertheless, it was obvious from the variogram (figure 4.31) that the data had anisotropy. Its spatial autocorrelation depends on direction. The main direction of anisotropy is nearly north-south vs. west-east. Thus, the bathymetry changes more rapidly in the west-east direction and more slowly in the north-south direction, in general. And setting the same parameters for all directions in kriging (and as well in all other isotropic interpolations above) actually was not correct.

So, the **second kriging model** was spherical with Sill 4.5 and second order trend removal, with anisotropy angle 80 degrees from north and range 200 m in this direction, and range 300 m in the perpendicular direction (figures 4.32, 4.33).

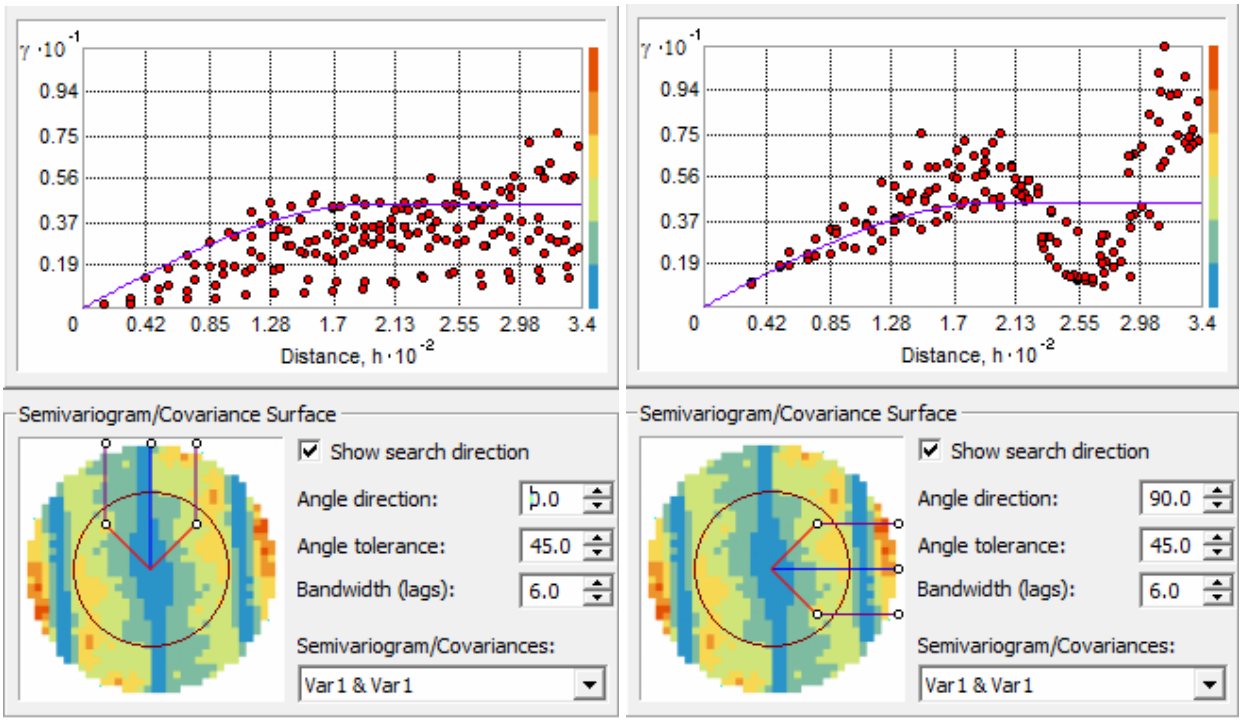


Figure 4.32. The semivariograms.

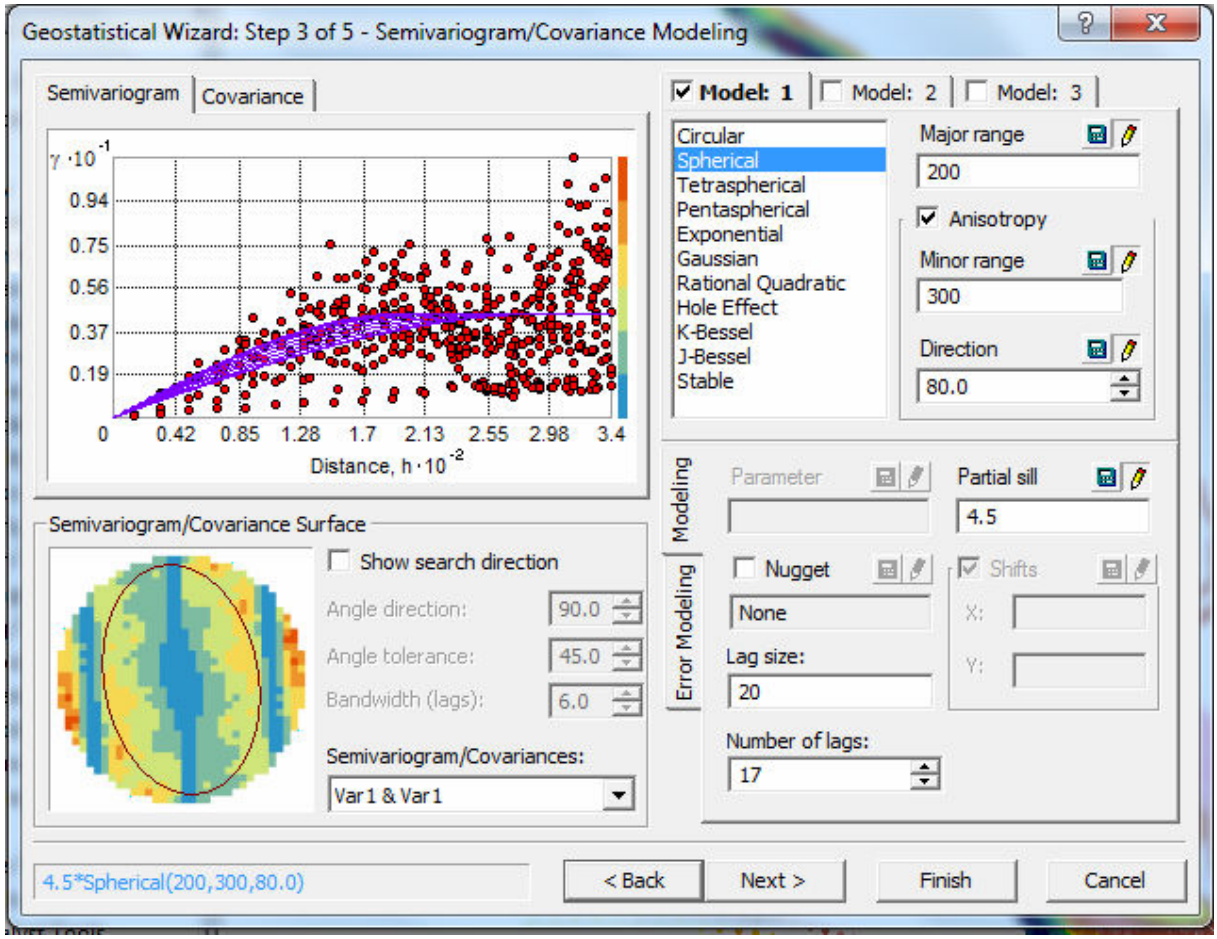


Figure 4.33. Kriging with anisotropy 4.5*Spherical(200,300,80).

For these two kriging models cross-validation was performed (figure 4.34). It showed that the model with anisotropy was slightly more accurate.

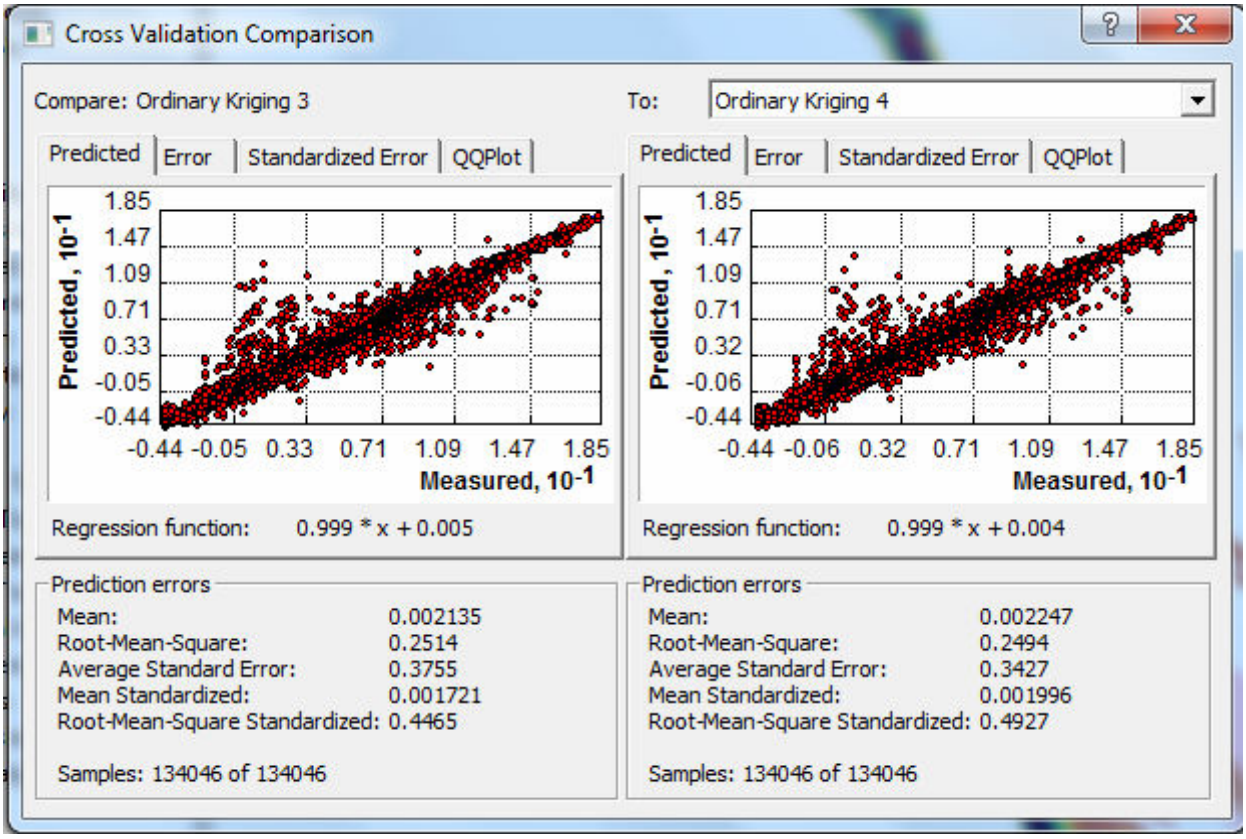


Figure 4.34. Cross-validation.

Visual observation of the resulting surfaces gives better understanding of the quality of the interpolation (figure 4.35). It is visible very good that all interpolations not respecting anisotropy (isotropic) have artefacts on the surfaces, such as artificial hills and deep holes along the channel. These errors occur due to highly irregular distribution of the survey points along ship tracks, namely at sharp turns of the tracks near the shore. Isotropic interpolations assume circular influence of each point and result in “wavy” interpolation. Figure 4.36 illustrates isobaths obtained from Topo to Raster interpolation with this effect.

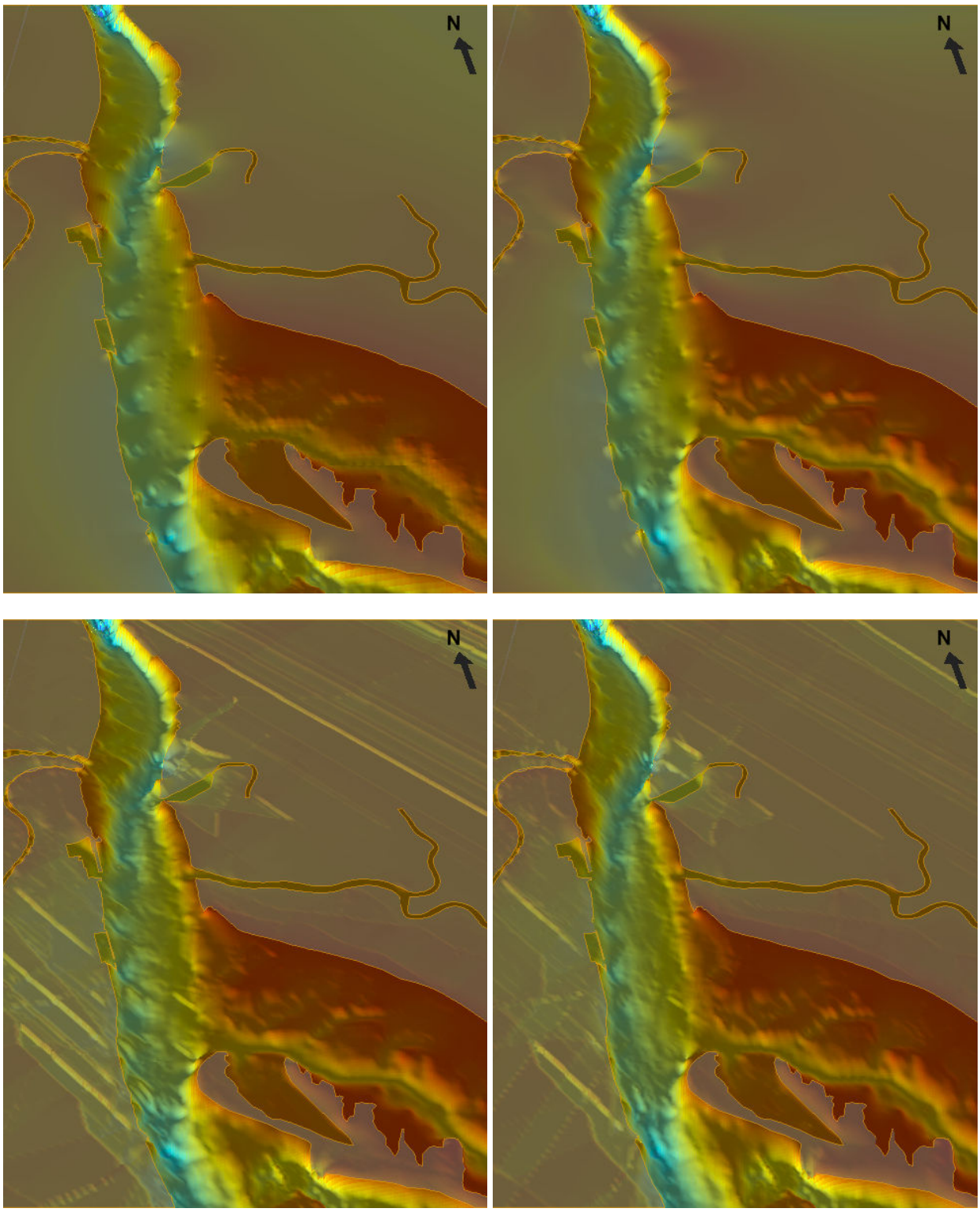


Figure 4.35. Minimum curvature (upper left), Topo to Raster (upper right), Kriging spherical (lower left) and kriging with N-S anisotropy (lower right).

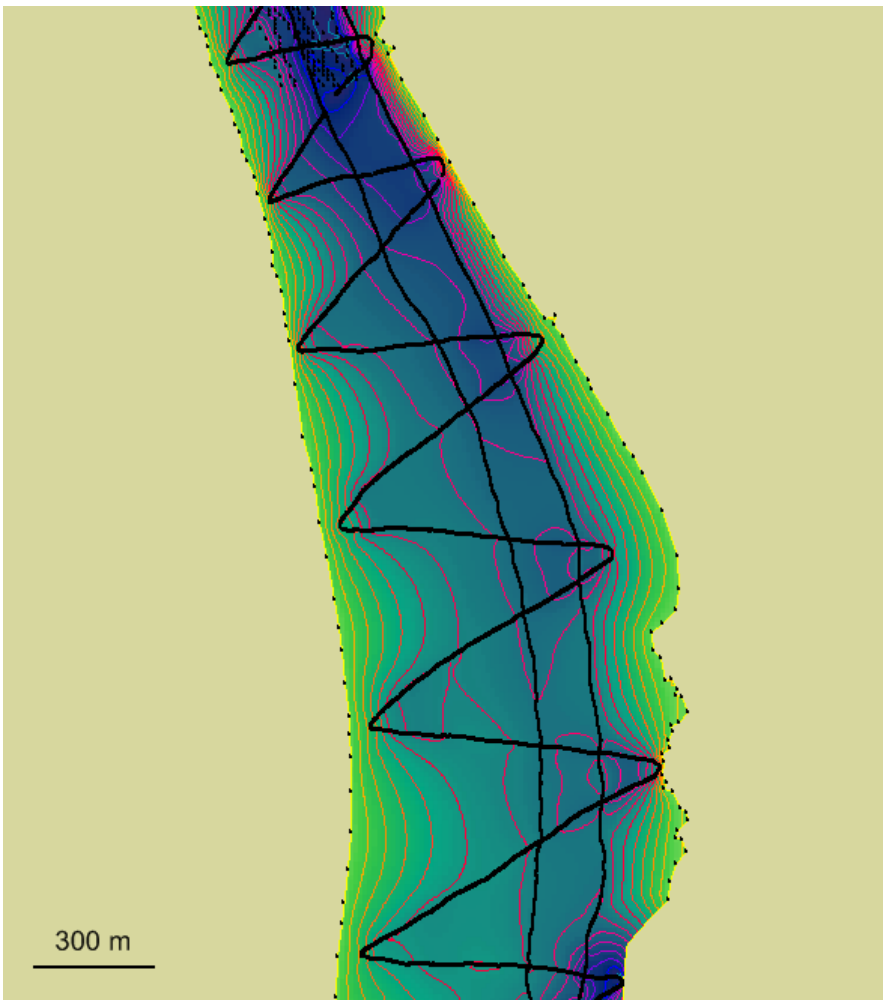


Figure 4.36. Isobaths (every 1 m) and data points along ship tracks.

But, kriging with anisotropy showed not good result as well. Despite the improved bottom in the north-south parts of the estuary, it produced much worse result in the curved parts of the channel (not aligned in the north-south direction). Figure 4.37 (red arrows) show one of the east-west parts of the channel where all interpolations interrupted the deep channel, producing unexisting shallow barriers, and artificial hills on the shoreline. The anisotropic kriging produced even the worst result in this place (figure 4.37).

So, among all the common interpolations performed, the best (realistic, smooth enough and having less visual errors) interpolation was Topo to Raster (figure 4.38). But, it still was not good enough because of problems due to varying anisotropy.

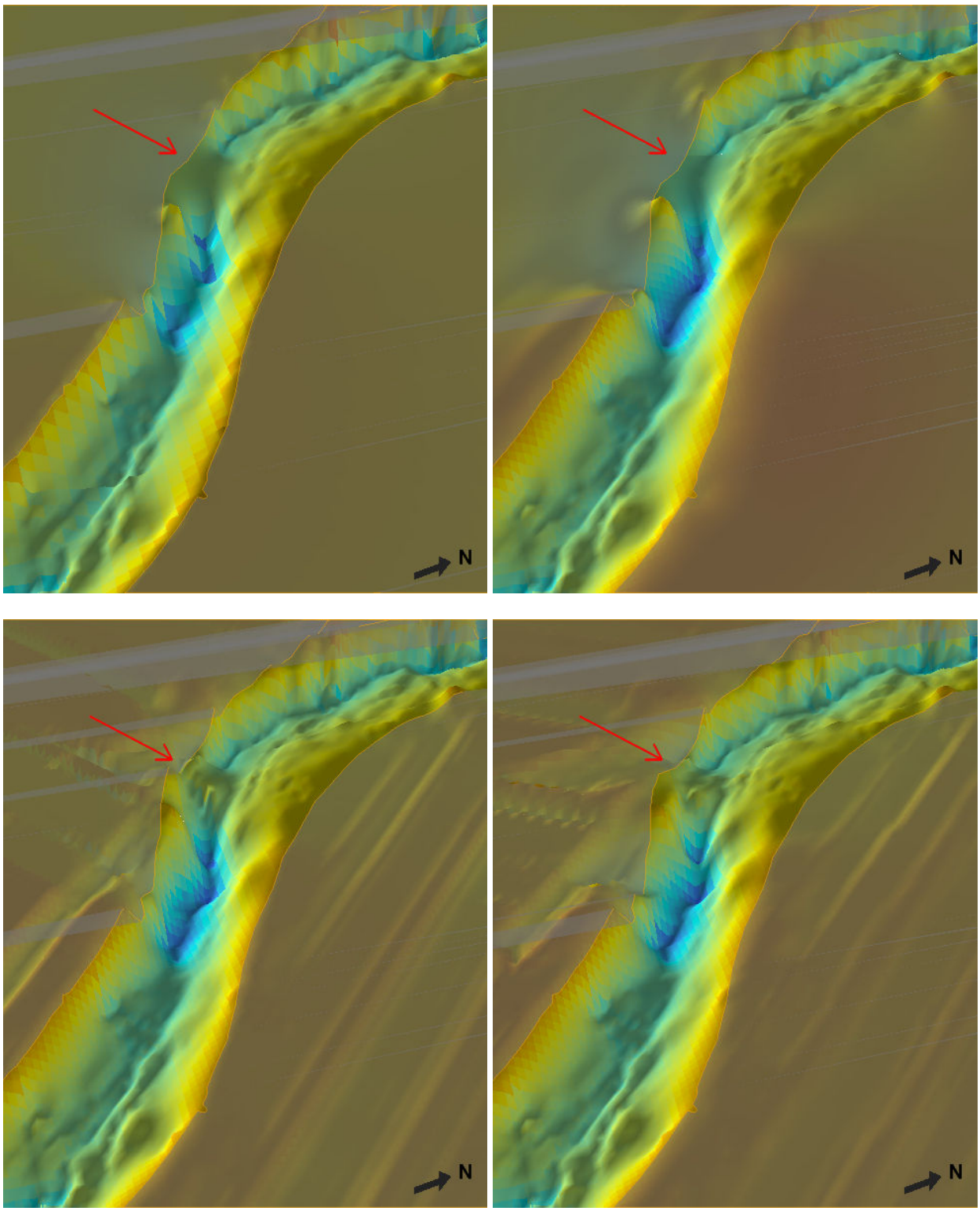


Figure 4.37. Minimum curvature (top left),Topo to Raster (top right),
Kriging isotropic (bottom left) and kriging with N-S anisotropy (bottom right).

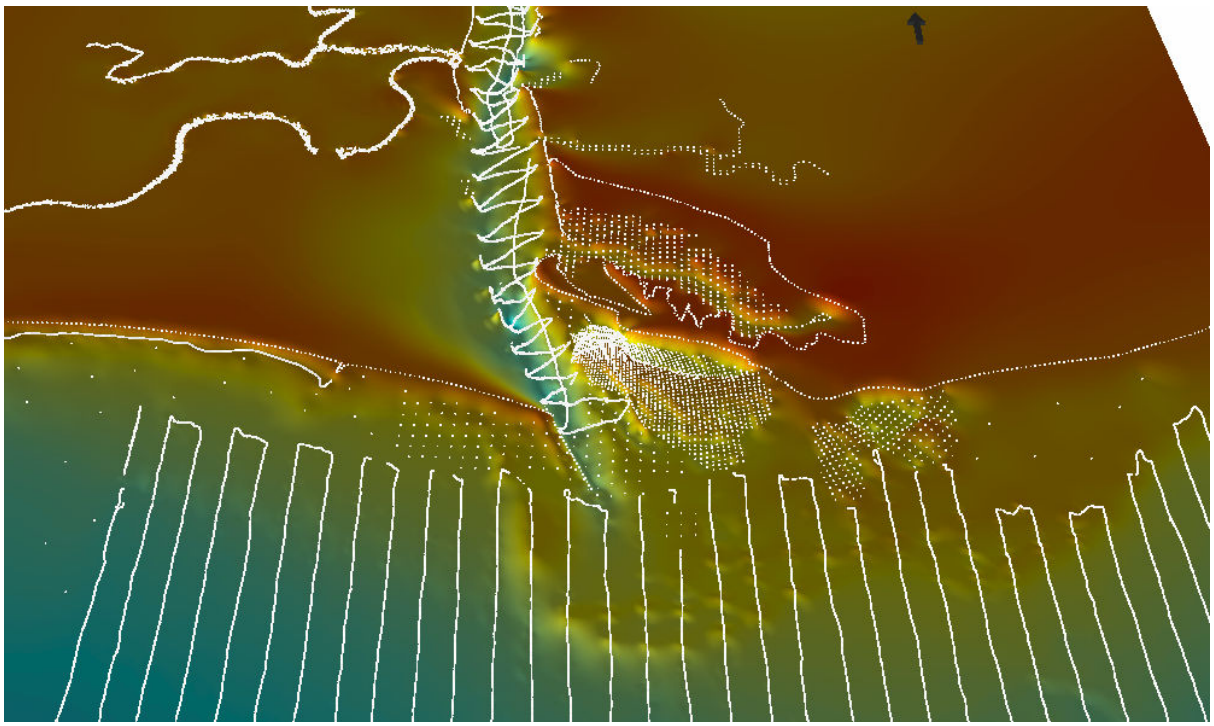


Figure 4.38. Topo to Raster surface and data points.

Advanced method using river straightening

It was clear from the interpolation attempts that common methods cannot interpolate a curved river correctly (and a long estuary as well), because anisotropy of river bottom is variable and follows the river centerline (and specifically the thalweg). Bathymetry usually changes very slowly along the centerline and rather quickly across-channel.

Wadzuk and Hodges (2001) proposed a methodology for straightening a sinuous river. This methodology assumed converting the Cartesian coordinates (x,y) into R(x,y) and M(x,y) coordinates, where M is the distance along the river centerline for any point and R is the perpendicular distance of this point from the centerline. Thus, a curved river can be transformed into a straight river by mapping the points in R and M coordinates (figure 4.39). Interpolation and other processing can be performed in this transformed (R,M) space. Then, all the points can be transformed back to the original coordinates when necessary, so the original geometry actually is not affected (Merwade et al., 2005). However, this back-transformation is mathematically quite complicated and cannot be done automatically and easily in GIS environment. This methodology was improved by Merwade et al. (2005), Merwade et al. (2006) and Merwade et al. (2008), who also compared several interpolation methods and found that anisotropic interpolation methods, after coordinate transformation into (R,M) coordinates, performed significantly better results (40% reduction in RMSE) than regular isotropic interpolation methods (Merwade et al., 2006).

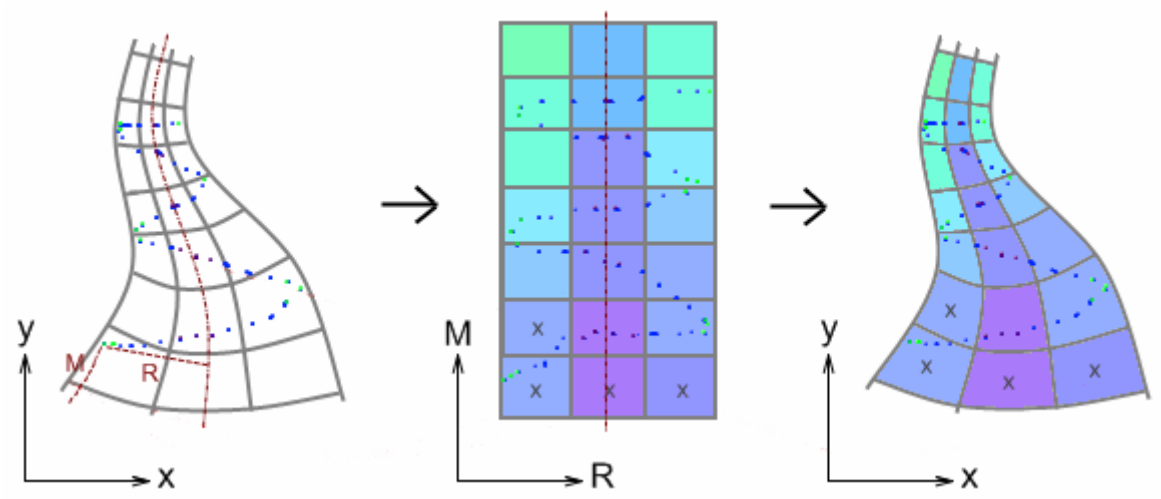


Figure 4.39. Transformation into channel-oriented coordinates.

So, for obtaining realistic interpolation of the estuary, it was necessary to implement this method, transforming bathymetry points into RM space and performing there interpolation with anisotropy along the centerline.

The centerline was drawn from the estuary polygon using ArcScan extension in ArcGIS and converted into a "route" (Polyline M) using the *Linear Referencing toolbox*. Then R and M coordinates were calculated to each point by the tool *Locate features along routes*, calculating distance to the route at the same time. Bathymetry points then were plotted using R instead of X, and M instead of Y coordinates (figure 4.40). Kriging with anisotropy in the centerline direction was performed using spherical model (figure 4.41) with 5 m raster cell.

As the creators and developers of the method showed, there still is no quick, simple and correct method to transform the points (or the interpolated raster) from RM back to Cartesian coordinates using GIS. But it was not necessary to do this in this case. Actually, the only result needed for the MOHID model was a bathymetry value for each curvilinear grid cell. So, curvilinear grid centers and corners (points) were also transformed into channel-oriented (R,M) coordinates (figure 4.39). The curvilinear grid cells in normal Cartesian space coincide almost exactly with the Voronoi polygons computed for the cell centers. So, Voronoi diagram was selected as a representation of the curvilinear grid in RM space (figure 4.42). The Voronoi polygons were created and clipped to the grid outline (obtained from the transformed grid corners). Then the raster produced by kriging was overlapped by the Voronoi polygons and average bathymetry value for each polygon was calculated from overlaid raster cells by the *Zonal Statistics tool*, in RM space. These average bathymetry values were attached to the curvilinear grid centers and then these centers were plotted using their original X and Y coordinates (stored in the attribute table). Thus, a very simple and spatially correct back-transformation from RM space to XY space was performed (figure 4.43).

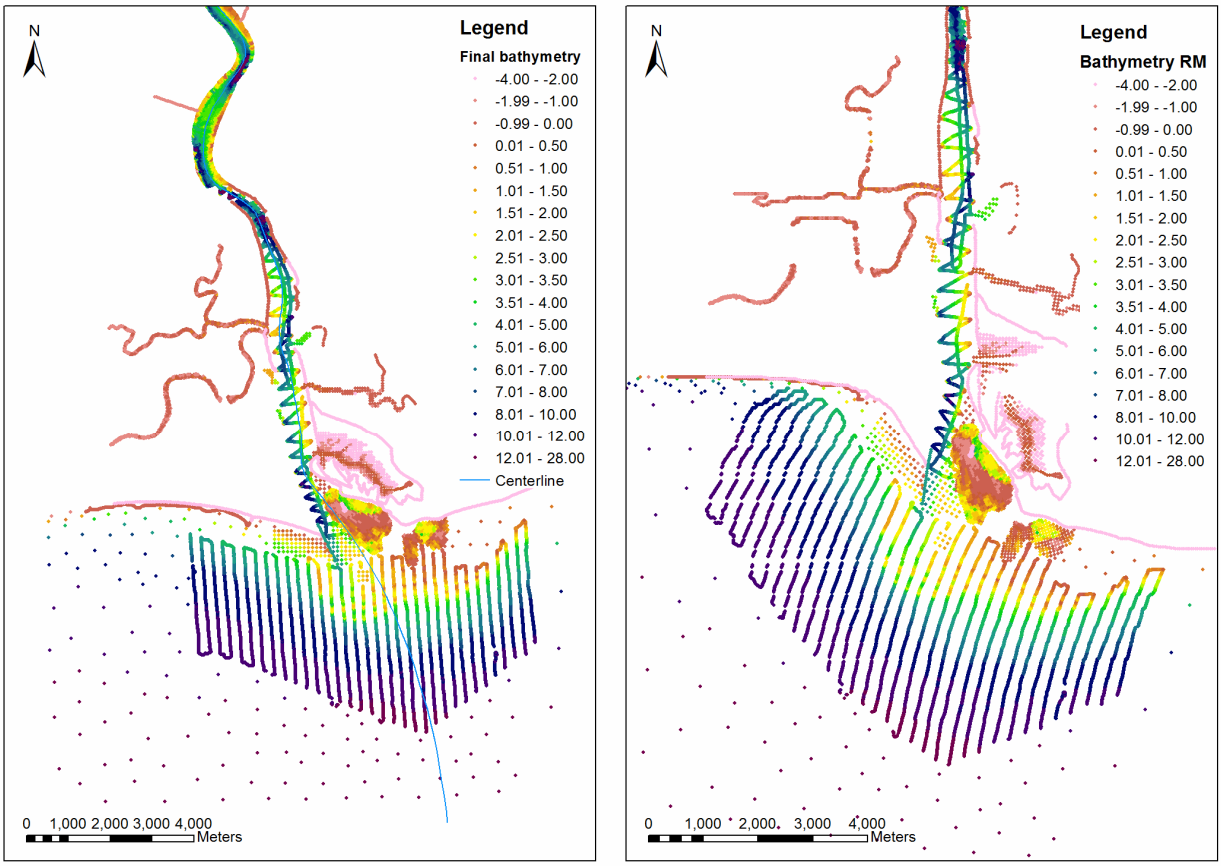


Figure 4.40. Bathymetry of the lower estuary: normal (left) and transformed (right).

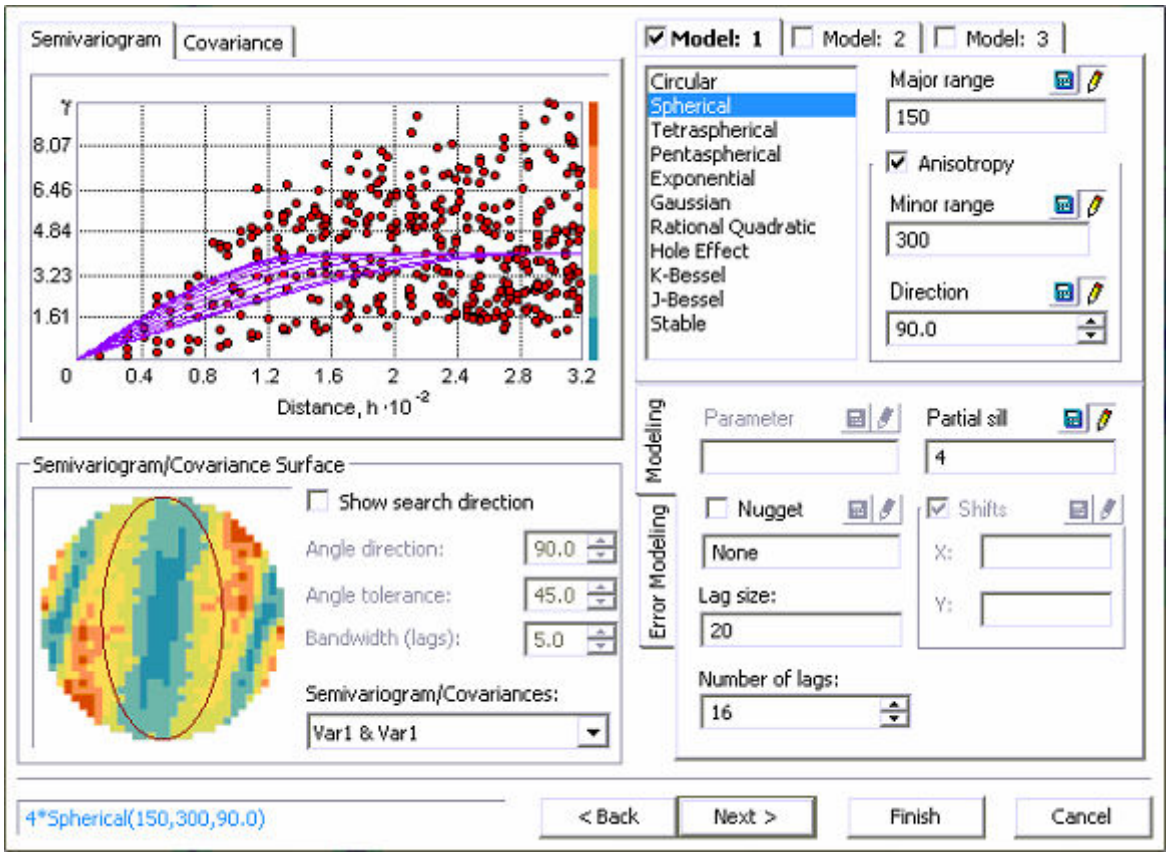


Figure 4.41. Kriging in RM space.

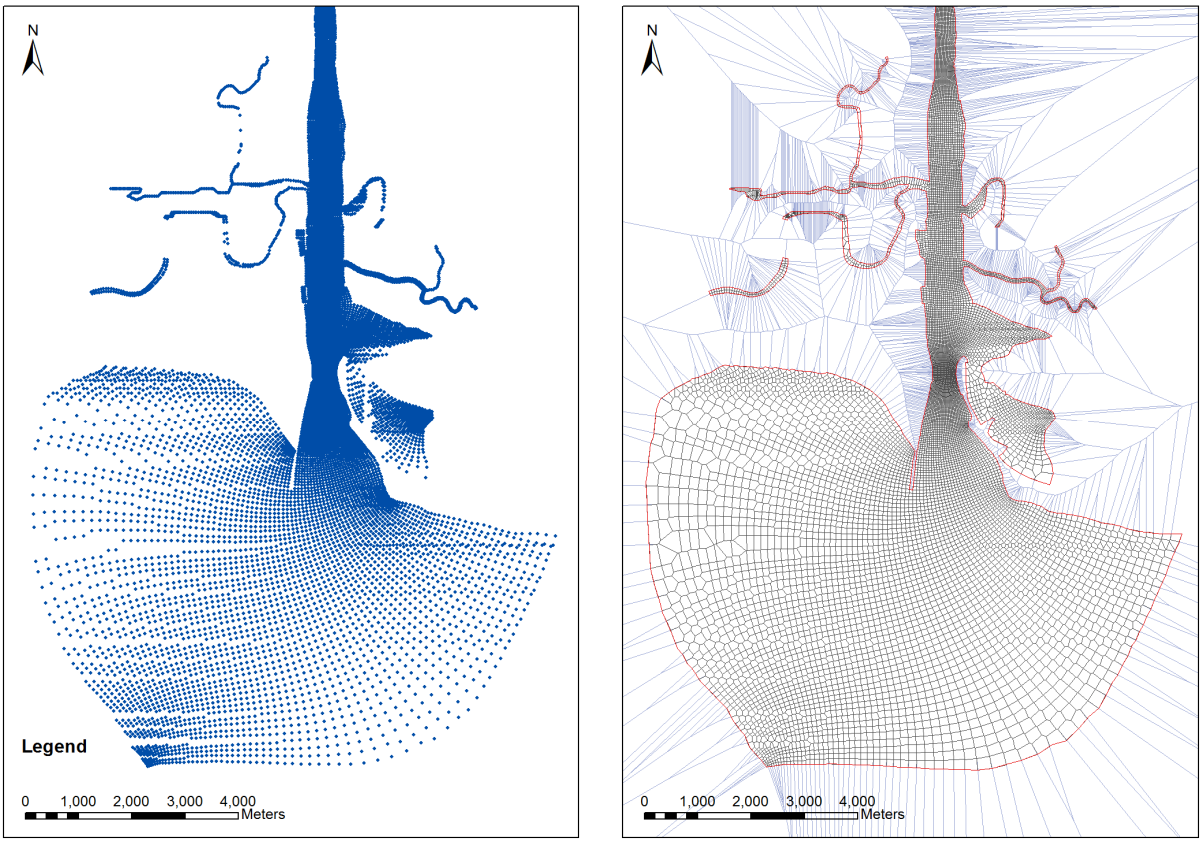


Figure 4.42. Transformed grid centers and reconstructed grid in RM space.

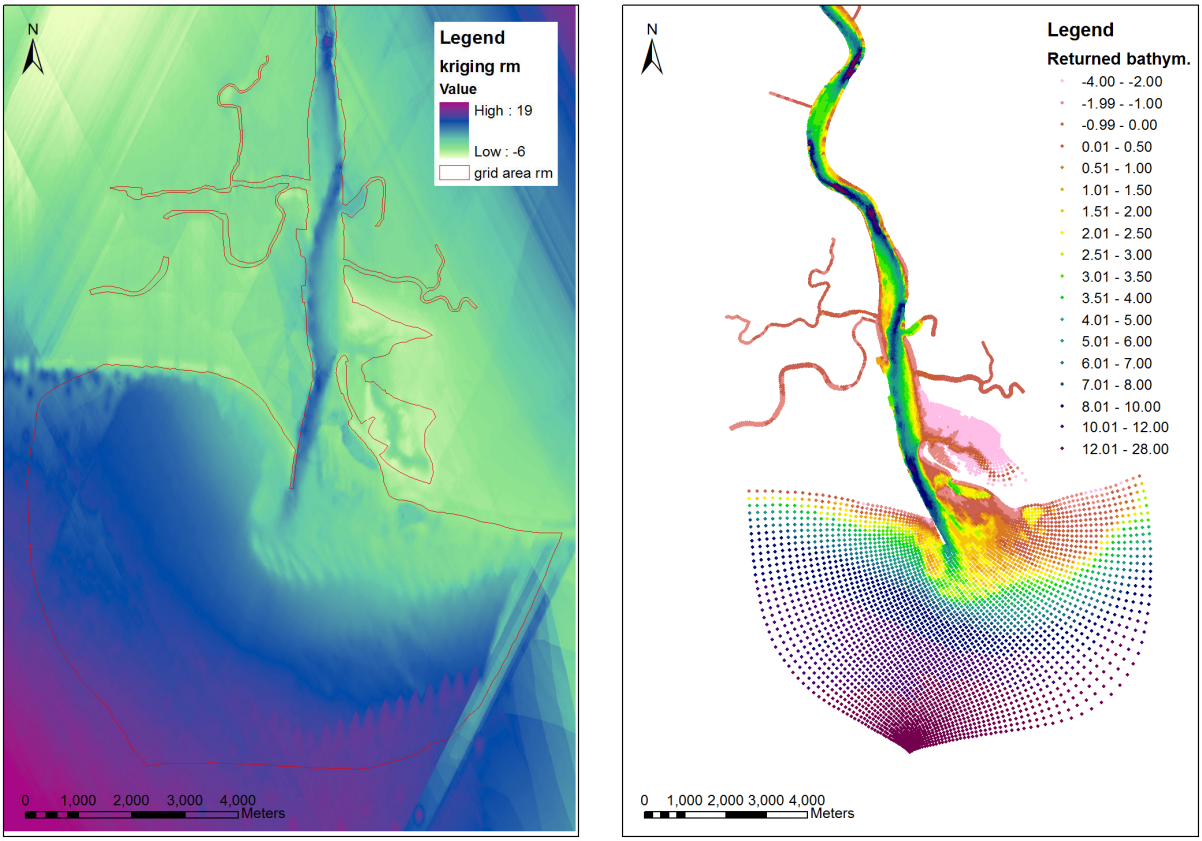


Figure 4.43. Kriging in RM space and back-transformed grid centers with mean values.

The resulting surface was significantly different from the other interpolations. The holes and hills at the ship track turns almost disappeared and the deep thalweg was preserved everywhere (figure 4.44). This method produced the most realistic interpolation at the gaps in bathymetry data restoring the thalweg and the near-shore shallowing (figure 4.45).

This method induced some errors in the tributaries and ocean in the places distant from the centerline, but these places near the model boundary were not important for simulations.

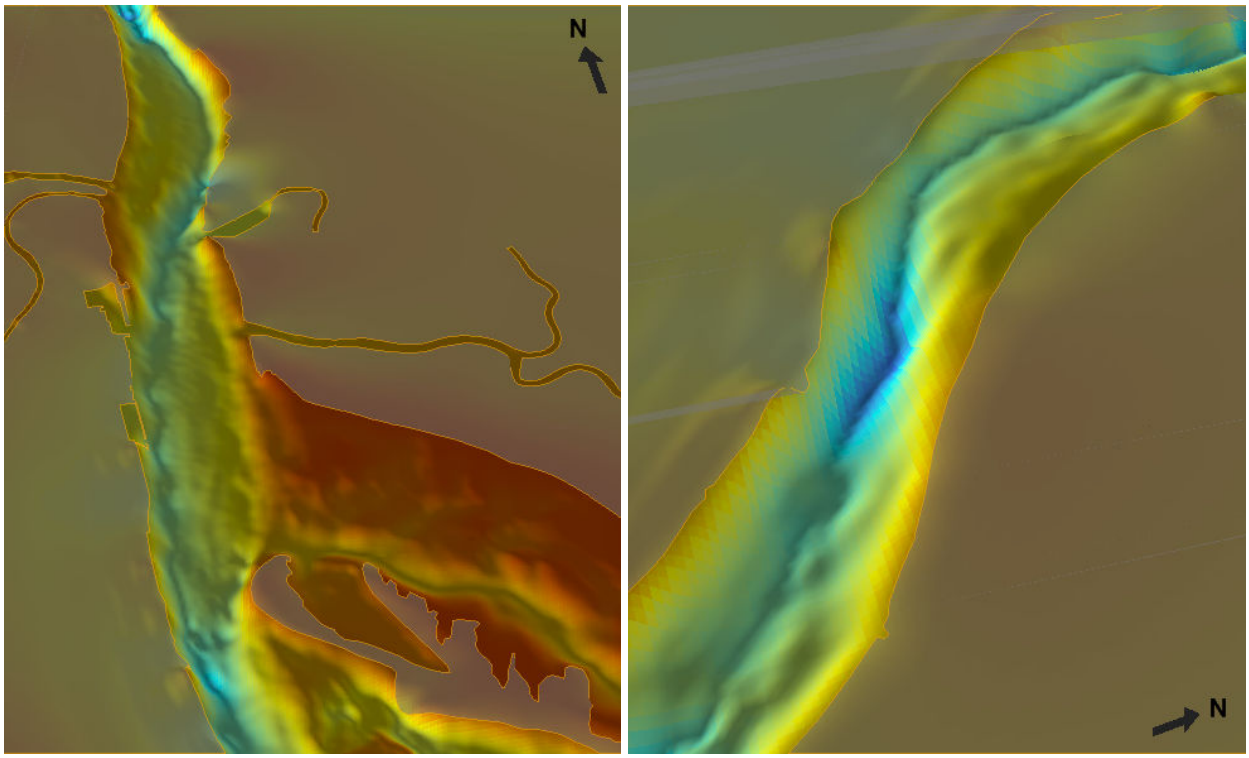


Figure 4.44. Surface based on kriging respecting varying anisotropy.

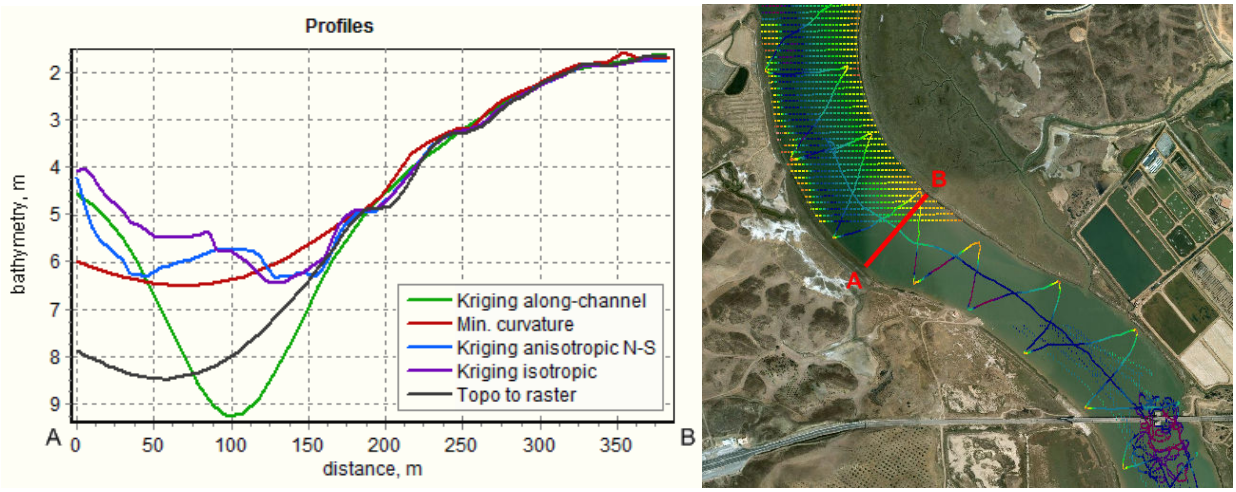


Figure 4.45. Comparison of the cross-sections.

Cross-validation of all the interpolation methods by a subset of data points was not performed because the points on ship tracks are very close to each other, so all interpolation methods show

very good results along the tracks. But the accurate interpolation was needed for gaps between the tracks with no points, where evaluation of interpolation could be done only by visual observation of the resulting surface or by evaluation of the hydrodynamic model results.

4.3. Model setup

Finally, the interpolated bathymetric grids were corrected in several locations (such as the bridge) in a try-and-error process to keep the model stable. This was done using MOHID GIS which has the advantage of allowing manual change of values of particular grid cells, which is not possible in many other GIS programs.

The time series point locations were created at the locations of real measurements in the estuary (figure 4.5), such as Simpatico (Garel et al., 2009b), based on the correspondent cells of the gridded bathymetry.

Several periods with different conditions were selected for simulation. These periods covered spring-neap tidal cycle and occurred in dry and wet seasons with relatively high and low river flow (figure 4.46).

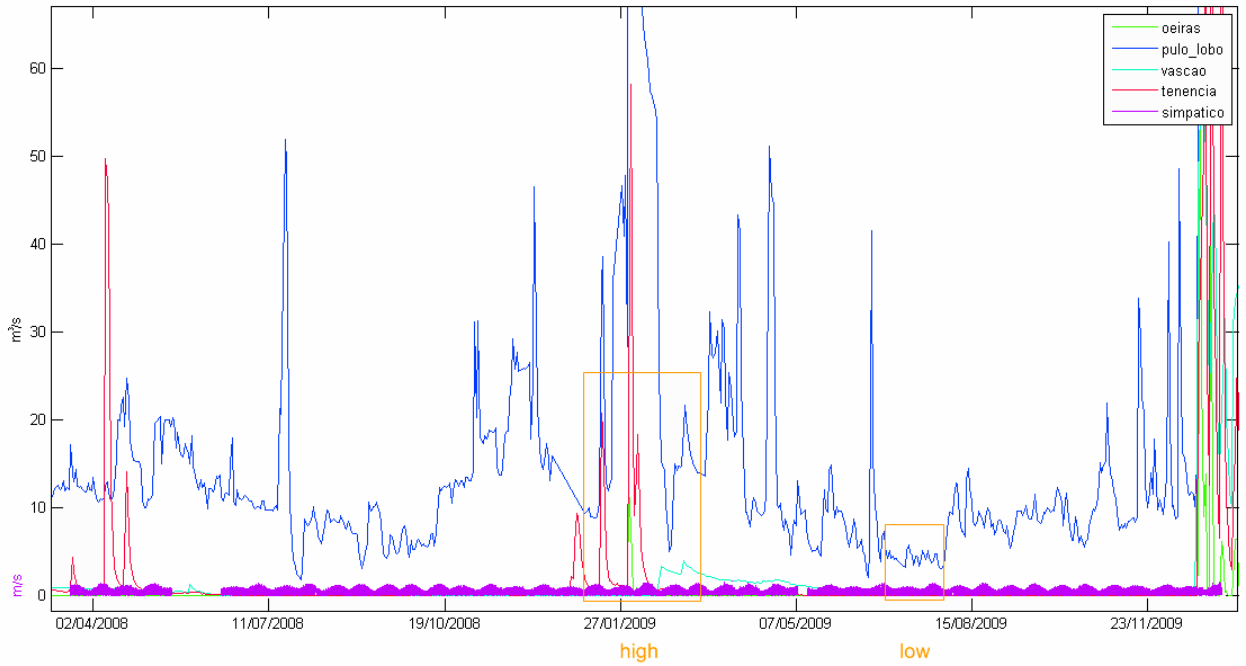


Figure 4.46. Selected high flow and low flow periods.

The hydrodynamic model was a 2D model with one vertical layer from bottom to surface with sigma coordinates and minimum depth of 0.1 m.

The hydrodynamic model was forced by tidal water elevations at the open boundary and using the river discharge at the upper end of the estuary and tributaries. The tide at the open boundary

was imposed using the data of tidal gauge of the Vila Real de Santo António port (table 4.3), with phase values corrected to the limits of the model domain in the ocean (Martins et al., 2006).

Table 4.3. Tidal harmonics for Guadiana (Vila Real S. A.).

Component	Amplitude	Period
M2	0.926	90.00973
S2	0.321	116.50973
N2	0.19	75.60973
K2	0.087	119.60973
M6	0.0	0.0
MS4	0.004	334.2937
M4	0.015	218.29369
L2	0.036	76.80973
MU2	0.019	56.00973
NU2	0.037	76.60973
O1	0.056	321.80243
K1	0.054	70.80243
P1	0.018	61.60243
T2	0.019	115.40973
2N2	0.025	58.39731
PI1	0.00103	70.80243

At the free-surface boundary, the fluxes across the surface (for example, rain) and the wind stress were assumed null (no waves). At the bottom boundary, the water flux was also assumed null. No differences in atmospheric pressure were assumed.

For the river and tributaries discharges the daily flow values from SNIRH database were used. Discharge for ETAR was included with default flow value 0.021757.

Initial model parameters were used as in the previous models (Lopes, 2004; Martins et al., 2006) but they were changed later during calibration:

- Vertical viscosity: 0.001
- Horizontal viscosity: 1.00
- Rugosity: 0.00005

The default value of density was 1027 mg/m^3. The hydrodynamic model used default barotropic conditions, implicit vertical advection and diffusion, upwind condition, Coriolis force,

horizontal and vertical advection and diffusion, bottom stress of 0.1, no momentum discharge, minimum horizontal advection of 0.5, hydrostatic conditions.

The time step was growing from 1 s at the cold start (from zero without initial velocities for stabilization) to 3 s for the simulations. Before each study period the warm-up periods of two days were simulated to stabilize the model.

Firstly the domain extending until Alcoutim was tested like the domains in the previous models of other authors (Lopes, 2004; Lopes et al., 2003 Morais et al., 2012; Saraiva et al., 2007; Martins et al., 2004; Dias and Ferreira, 2001). Later it was extended until the actual end of the estuary near Mertola (figure 4.47).

Initial bathymetry, containing sparse old points in the mouth over the sand bank and interpolated by MOHID's triangulation from original data points, was used for the calibration and validation process.

Figure 4.47. Model domains.

5. Model results and calibration

The hydrodynamics of the Guadiana Estuary was simulated for several periods with different conditions (spring-neap cycle and high and low river flow). In order to calibrate and validate the model, the results were compared to the measurements in several points along the estuary. The several input bathymetries based on the different processing methods in GIS were used for the simulations and then their results were compared.

5.1. Calibration

The results of the hydrodynamic model were compared to the measured water level and velocity modulus. Firstly the domain extending only until Alcoutim was tested for full tidal cycle at low and high river flow conditions.

The water height corresponded to the measurements quite well, but the velocity results were not good even after testing different calibration parameters (figure 5.1). The velocities were almost twice smaller comparing to the measured velocities in all calibration points. Additionally, the relation between ebb and flood velocities was wrong: stronger flood and smaller ebb velocities at spring tide in opposite to the measured stronger ebb and smaller flood (Erwan and Ferreira, in press), and at the neap tide the opposite situation occured.

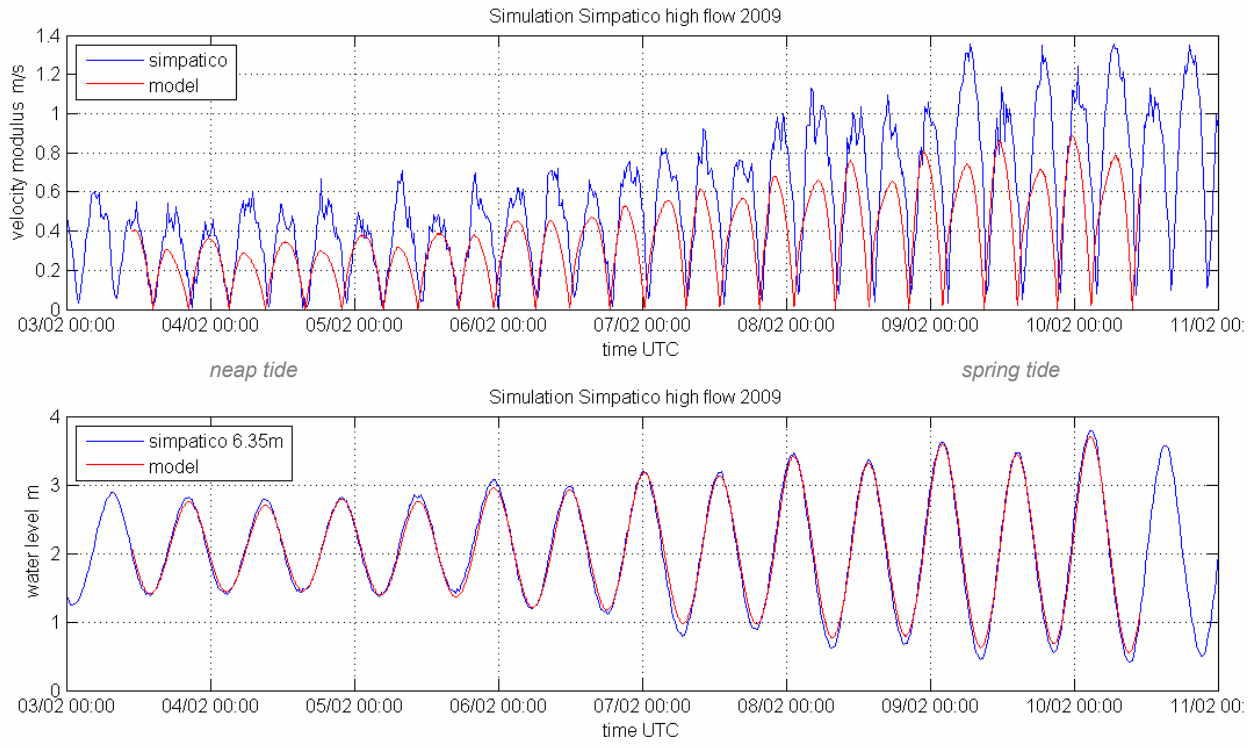


Figure 5.1. Model result comparing to measured data at the Simpatico station.

At the Ayamonte station the modelled velocities were two times smaller than measured as well (figure 5.2).

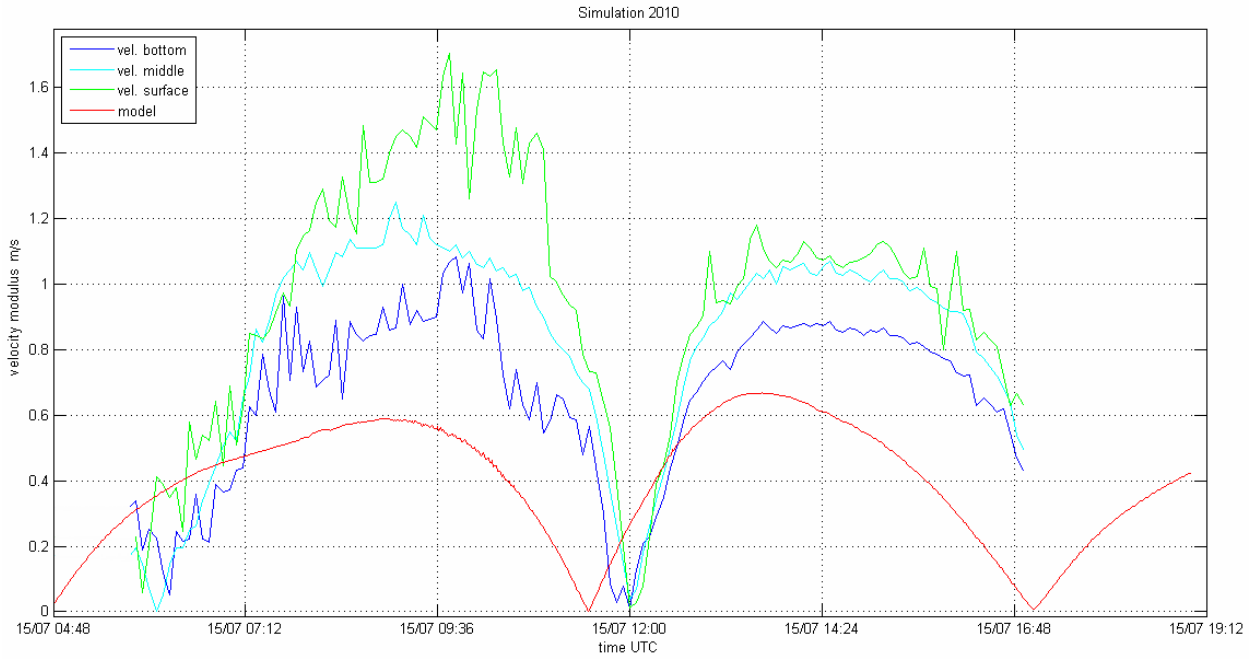


Figure 5.2. Model result comparing to measured data at the Ayamonte station (spring tide).

After extending the model domain up to its actual end near Mertola, the model results improved significantly (figure 5.3). Now the modelled velocities were comparable to the measured velocities in the both stations, Simpatico and Ayamonte (figure 5.4).

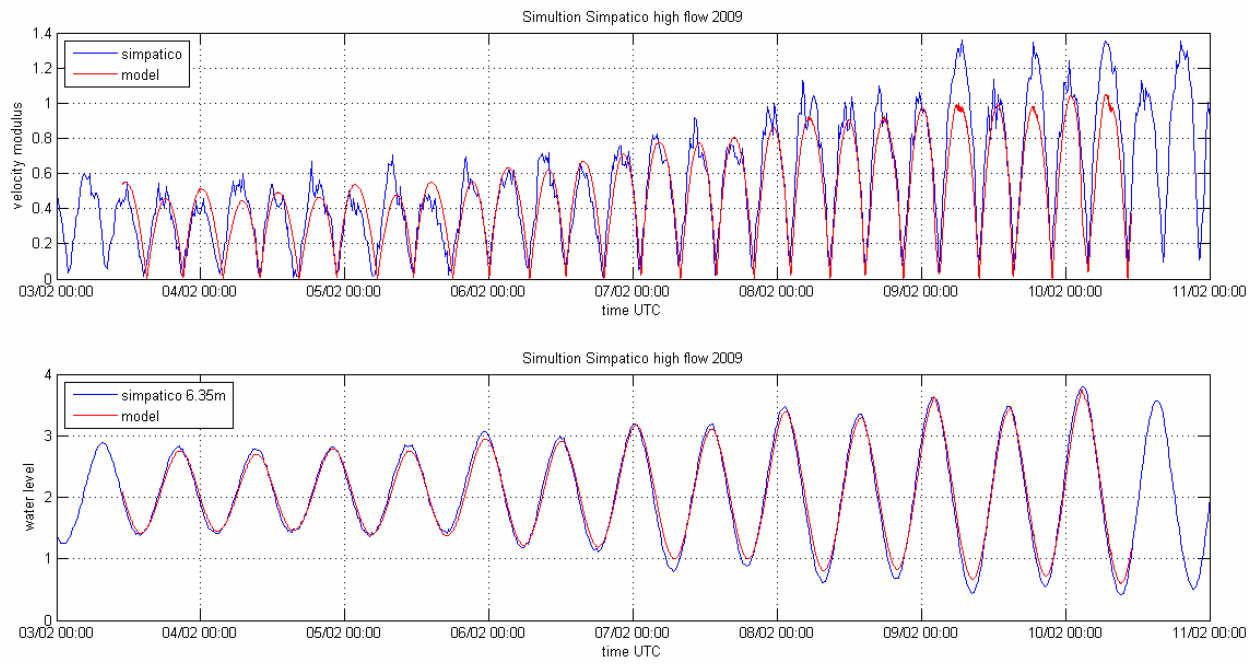


Figure 5.3. Model result with extended domain at Simpatico station[24].

[24] *On this figure and on all other plots the units are SI units: "m/s" for velocity and "m" for water lever (starting from the Hydrographic Zero)*

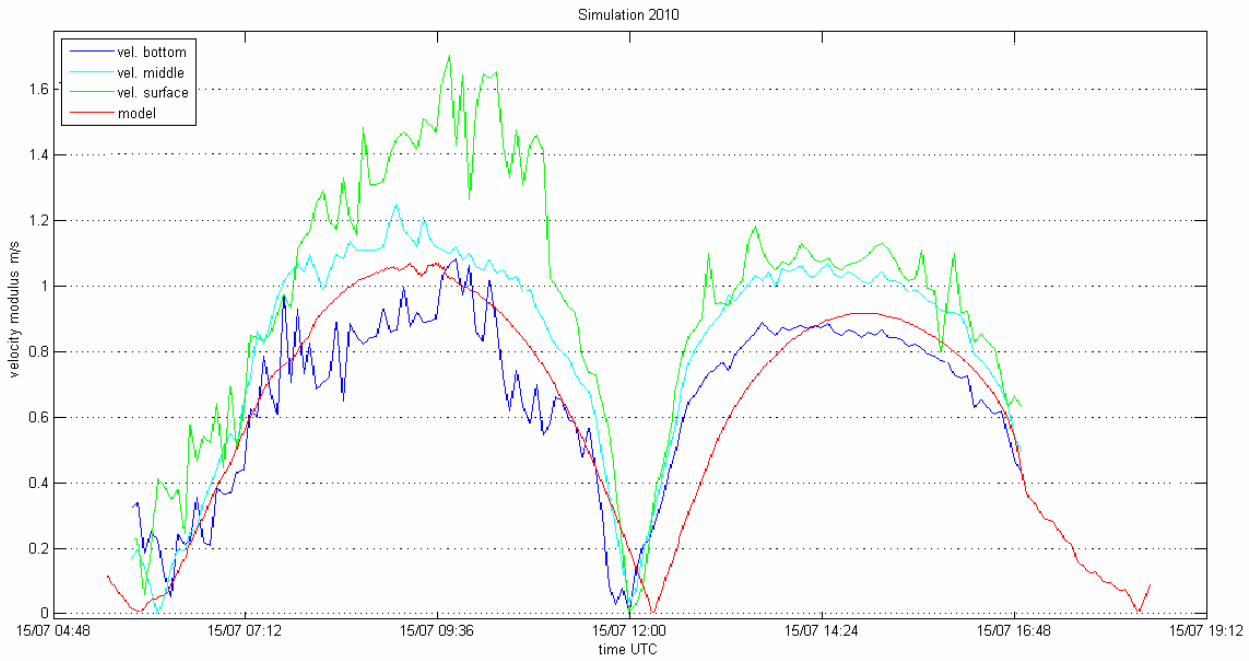


Figure 5.4. Model result with extended domain at the Ayamonte station.

Then, different values of the calibration parameters (numerical rugosity and viscosity) were tested and compared with the good dense set of measurements at the Simpatico station (figure 5.5). The calibration was performed at the worst conditions for the 2D model, namely stratification at high river flow (table 3.1). Figure 5.5 shows results of model runs with different parameters. It shows that the flow velocities increase with decreasing viscosity (v) and rugosity (r), but at very low values the model turns instable. However, with small rugosity the model started to produce wrong ebb-flood velocity variations. Since the modelled velocities were smaller than measured, the main calibration purpose was to increase velocities and also to respect the tidal asymmetry.

Finally, the best parameters for the model were chosen as:

- Vertical viscosity: 0.001
- Horizontal viscosity: 1.00
- Rugosity: 0.0001

Viscosity values were left as initial, and rugosity was increased from 0.00005 to 0.0001. These values allowed to have realistic velocities with stable model and correct tidal asymmetry at spring tide. The calibrated model results using these final parameters are presented in figure 5.6.

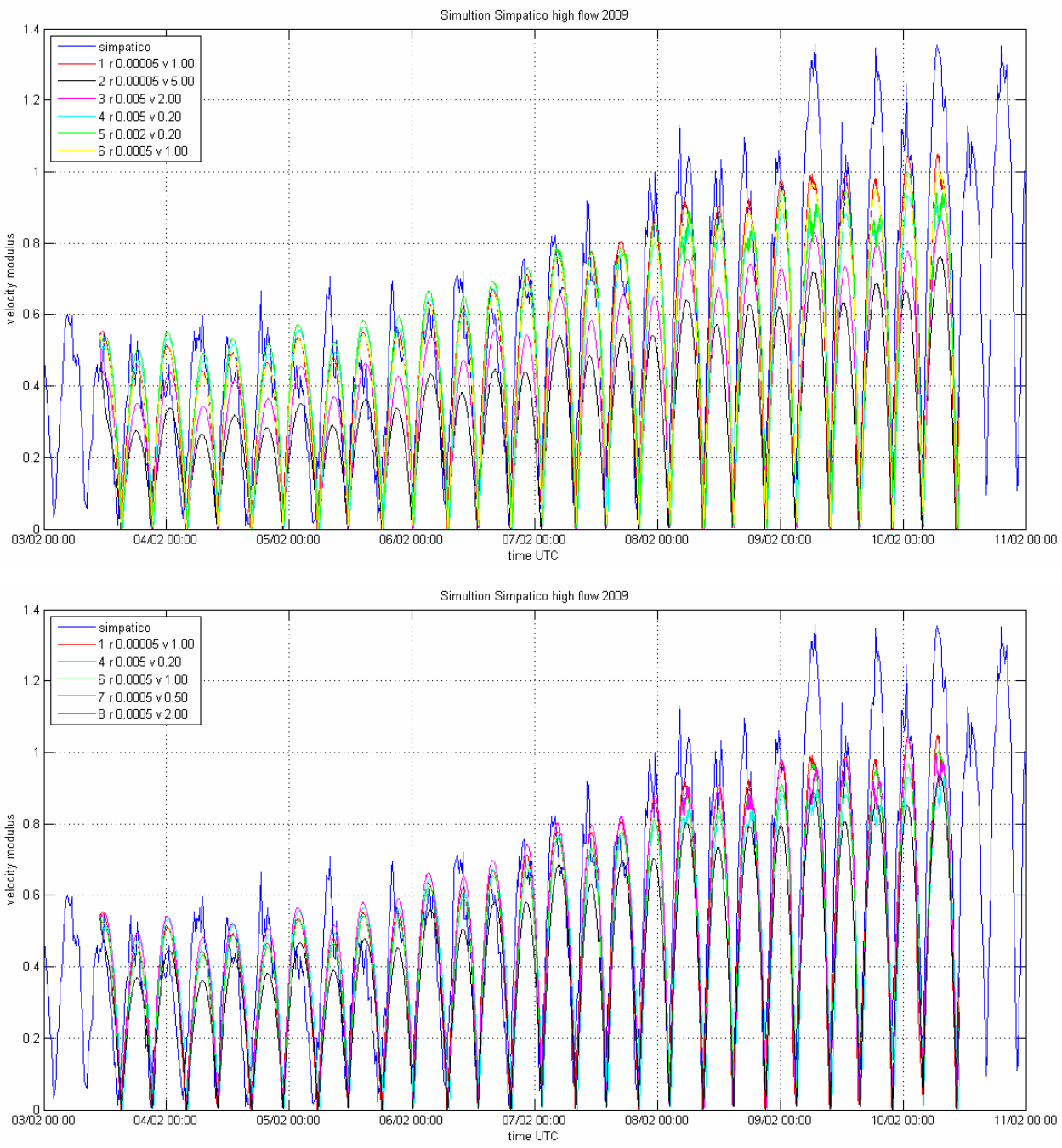


Figure 5.5. Model calibration at the Simpatico station (high river flow).

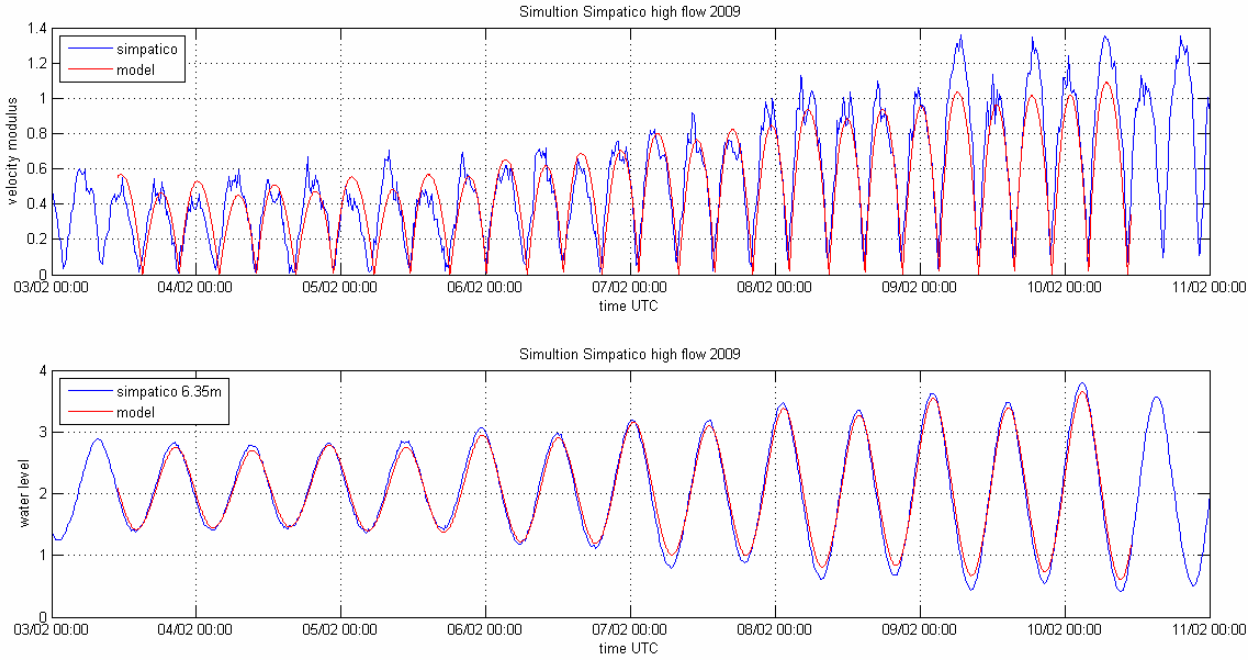


Figure 5.6. The model with the chosen parameters at Simpatico station.

Validation

The calibrated model then was validated in other time intervals with different river flows. The results of new model runs with final parameters were compared with other measured values (figures 5.7, 5.8, 5.9). The figures show that modelled velocities are now in agreement with the tidal asymmetry and their strength is similar to measured velocities.

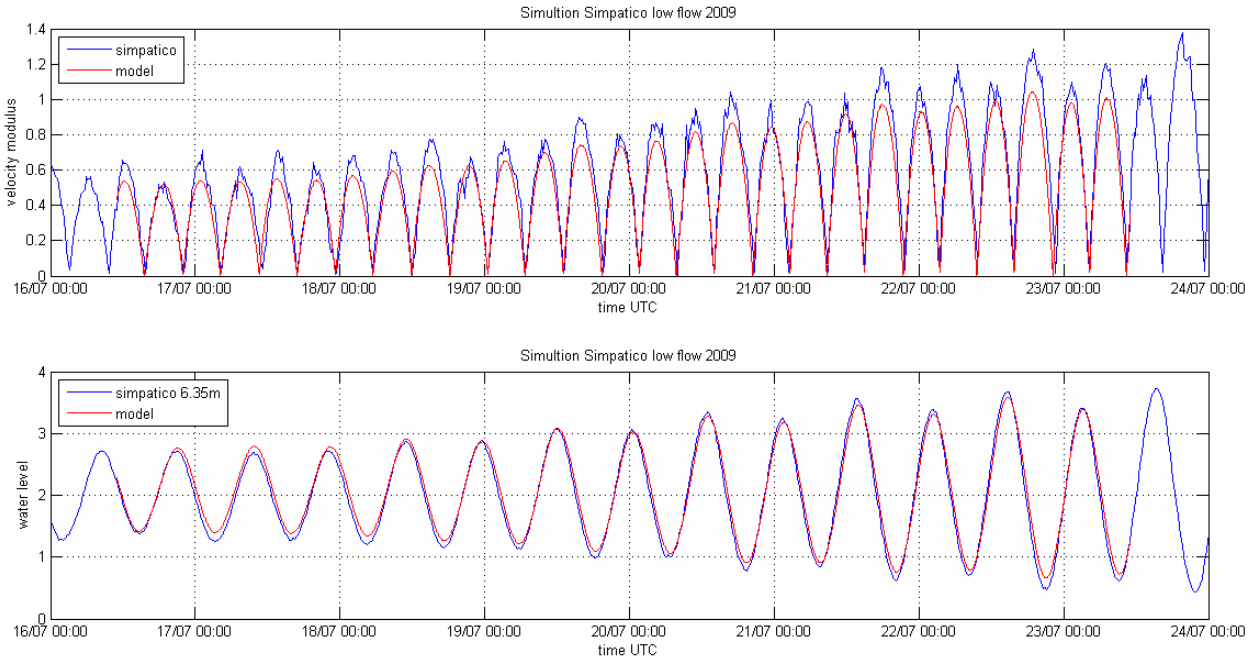


Figure 5.7. Validation of the model with final parameters in Simpatico (very low river flow).

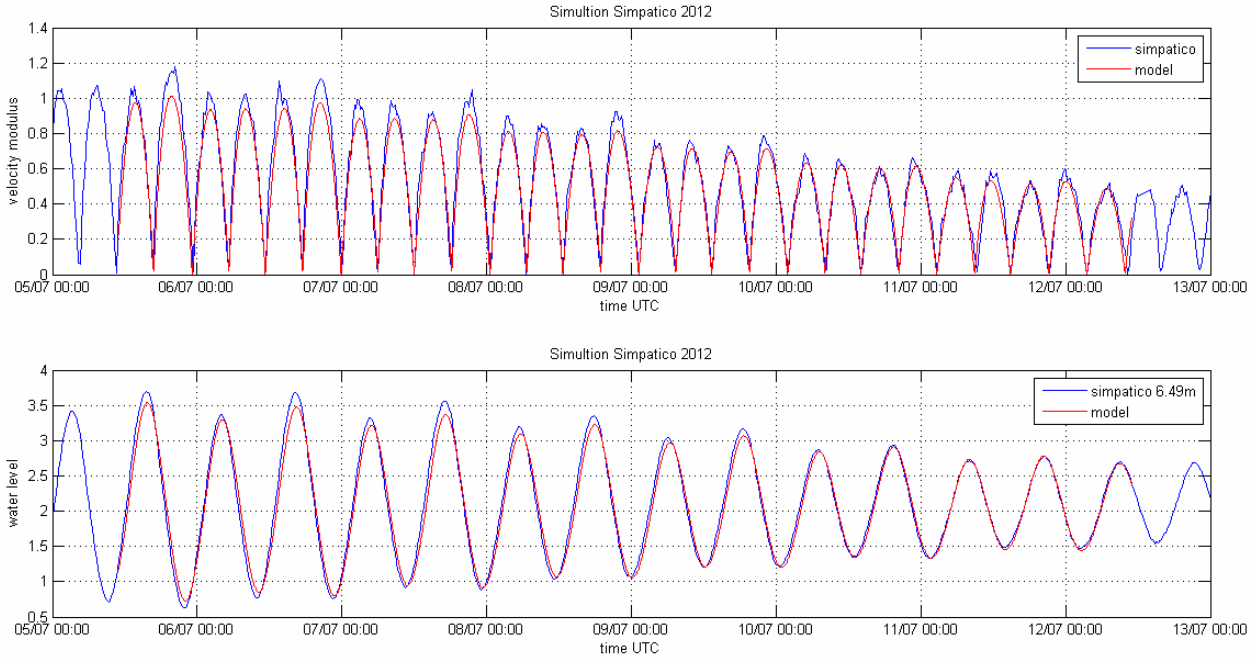


Figure 5.8. Validation of the model with final parameters at Simpatico station in recent time (rather low river flow).

However, the modelled velocities are still lower than the measured especially at spring tide when velocities are high.

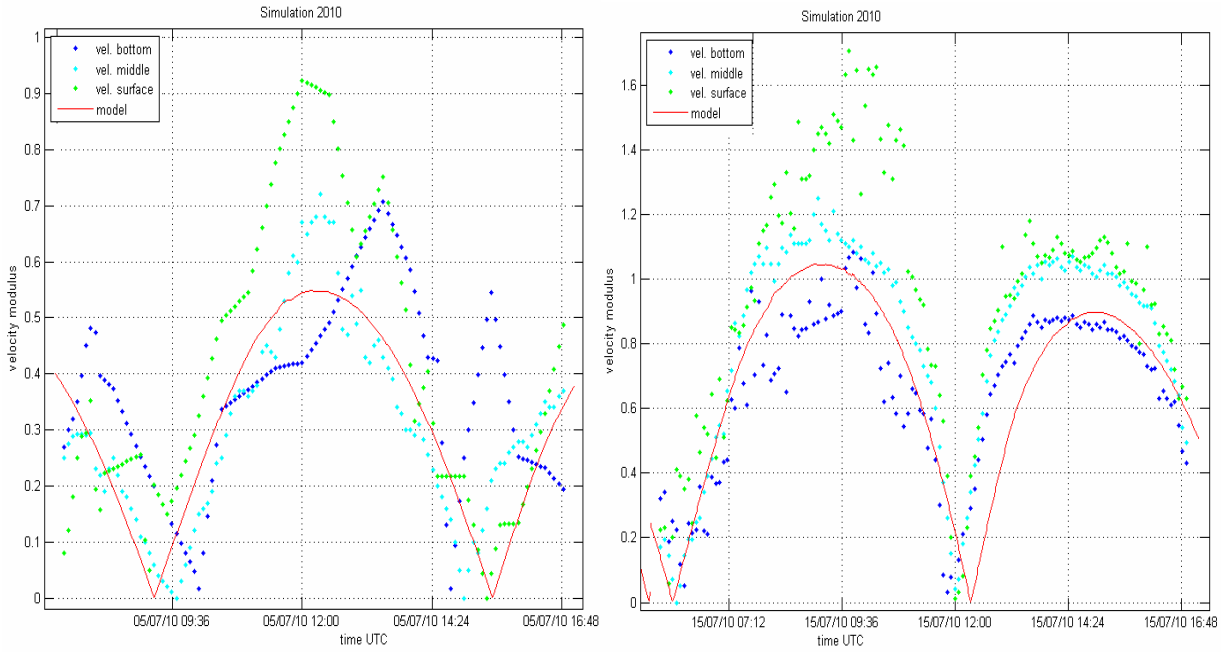


Figure 5.9. Validation of the model with final parameters at Ayamonte station.

However, despite rather good agreement between modelled and measured velocity modulus, velocity components have worse comparison, especially in the East direction (figure 5.10). So, the velocity of the flow was simulated correctly, but the direction was slightly wrong at Simpatico (Ayamonte station did not have velocity components data).

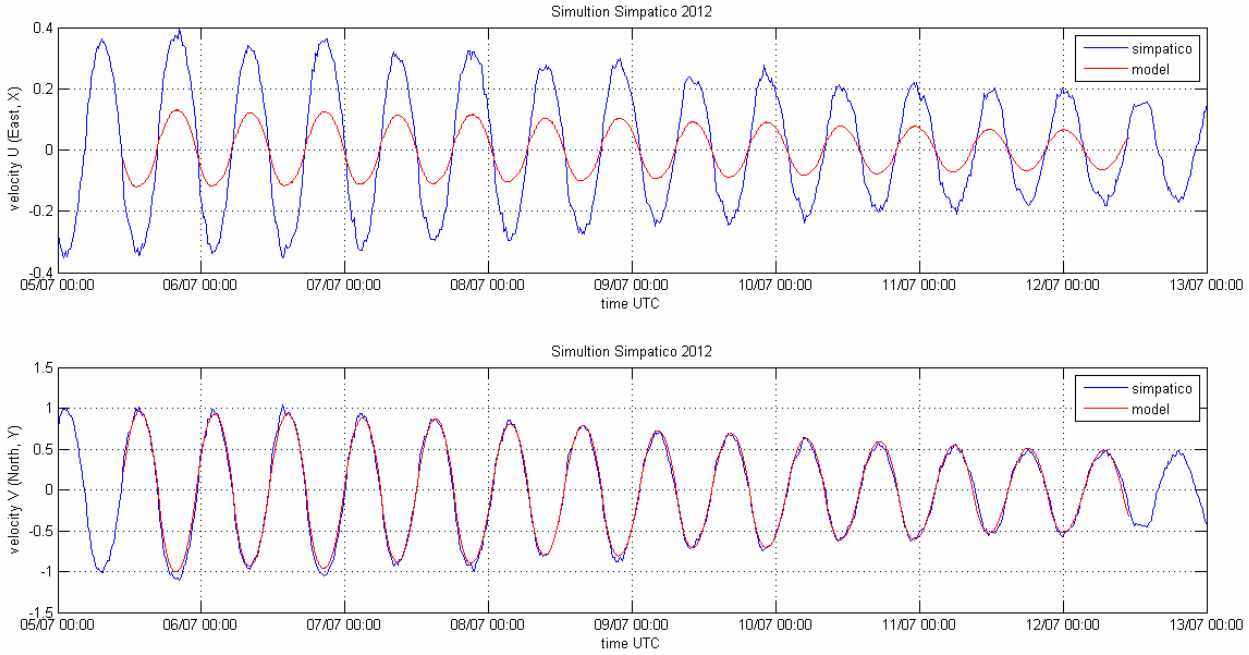


Figure 5.10. Velocity components at Simpatico station in recent time.

The stations with measurements of 2001 had very poor data, so the stations VRSA and Castro Marim, which are very close to Simpatico and Ayamonte, were not used for comparison. The station Odeleite is located far upstream near the Odeleite mouth between the middle and the upper estuary (figure 4.5). The modelled velocities were comparable to those measurements (figure 5.11).

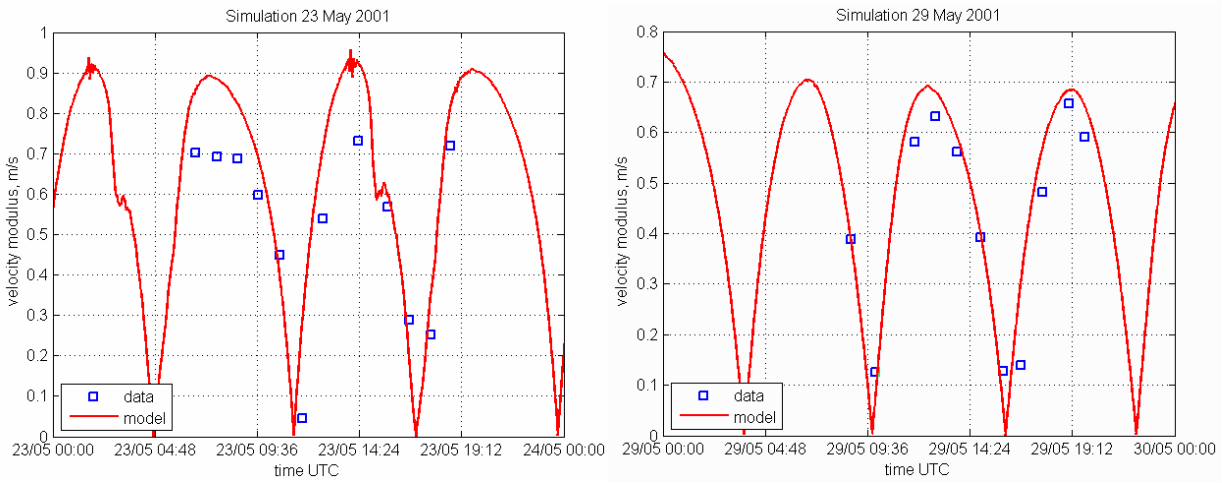
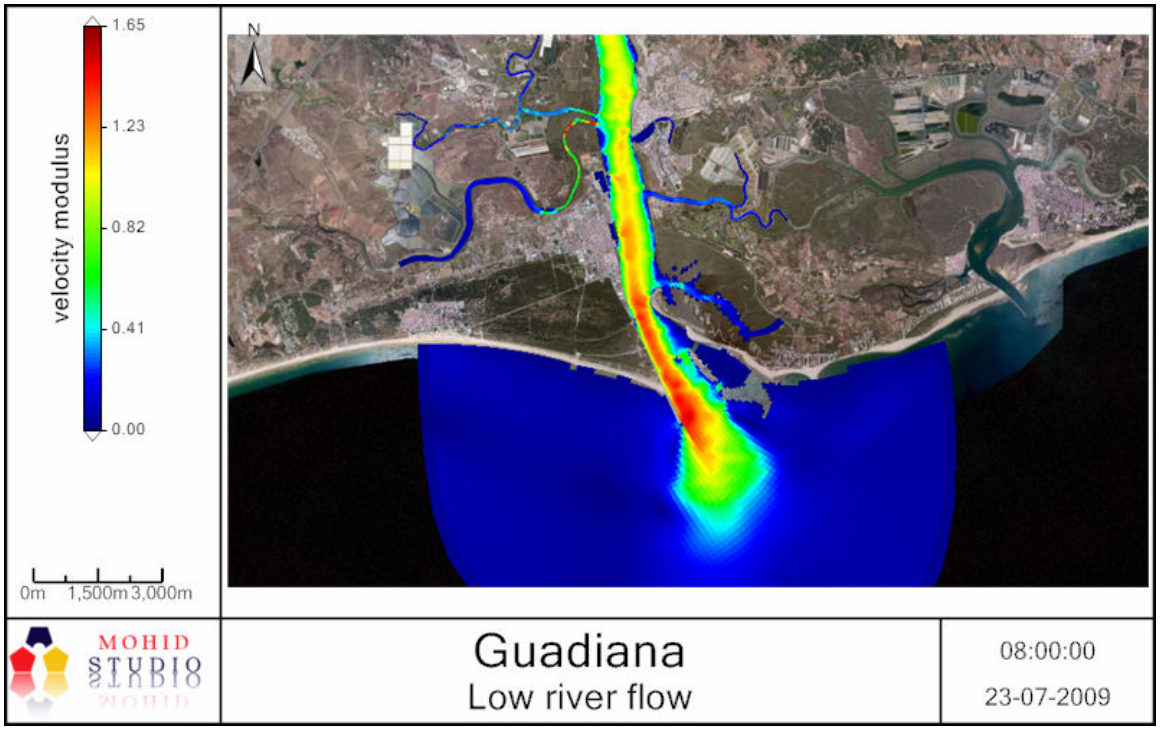


Figure 5.11. Validation of the model with final parameters at Odeleite station.

Velocity fields

The maps of the modelled velocities are presented in the figures 5.12-5.15. They show physically correct behaviour of the water flow: higher velocities at the deep channel, smaller velocities over the shoals (figures 5.12, 5.13), and the highest velocities in the narrowest parts of

the estuary. Flow velocities are higher in the lower estuary and reducing upstream as reported by Garel et al. (2009a).



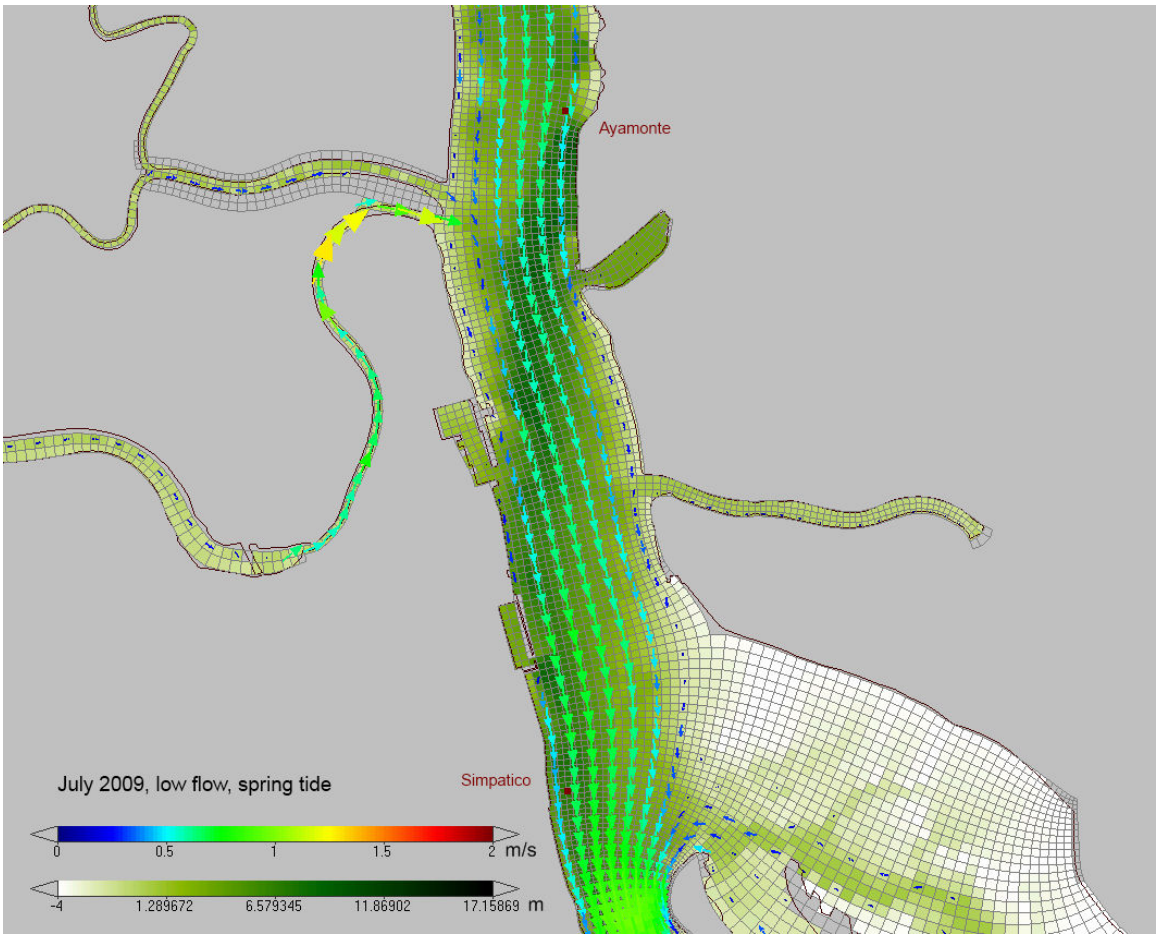
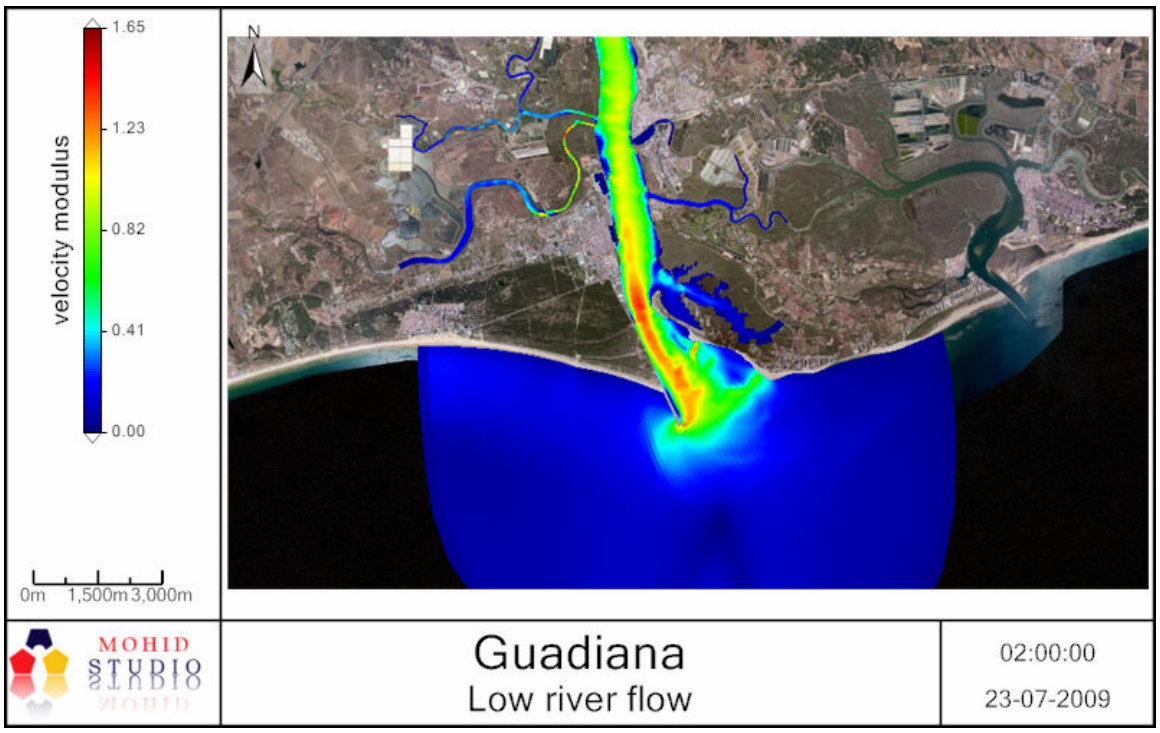


Figure 5.12. Spring tide, ebb velocities.



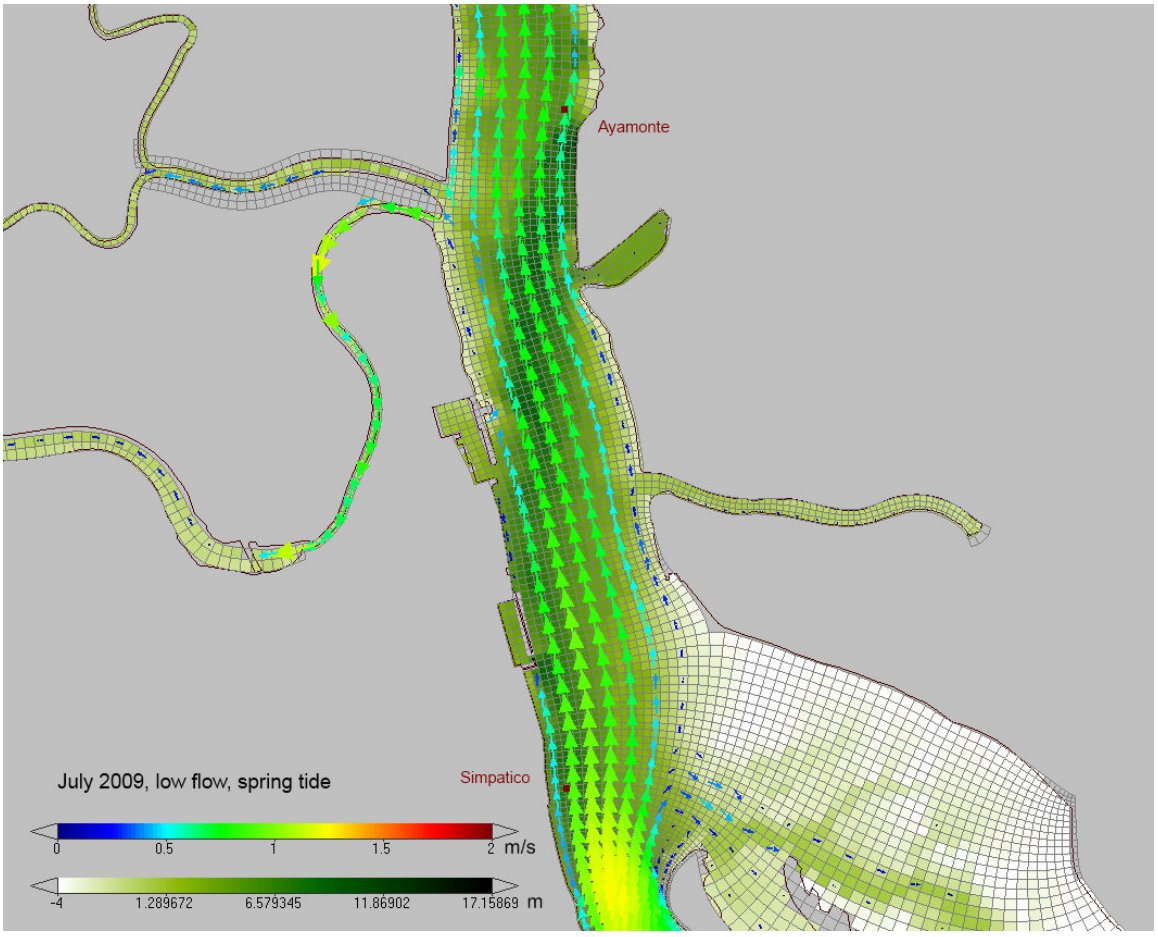


Figure 5.13. Spring tide, flood velocities.

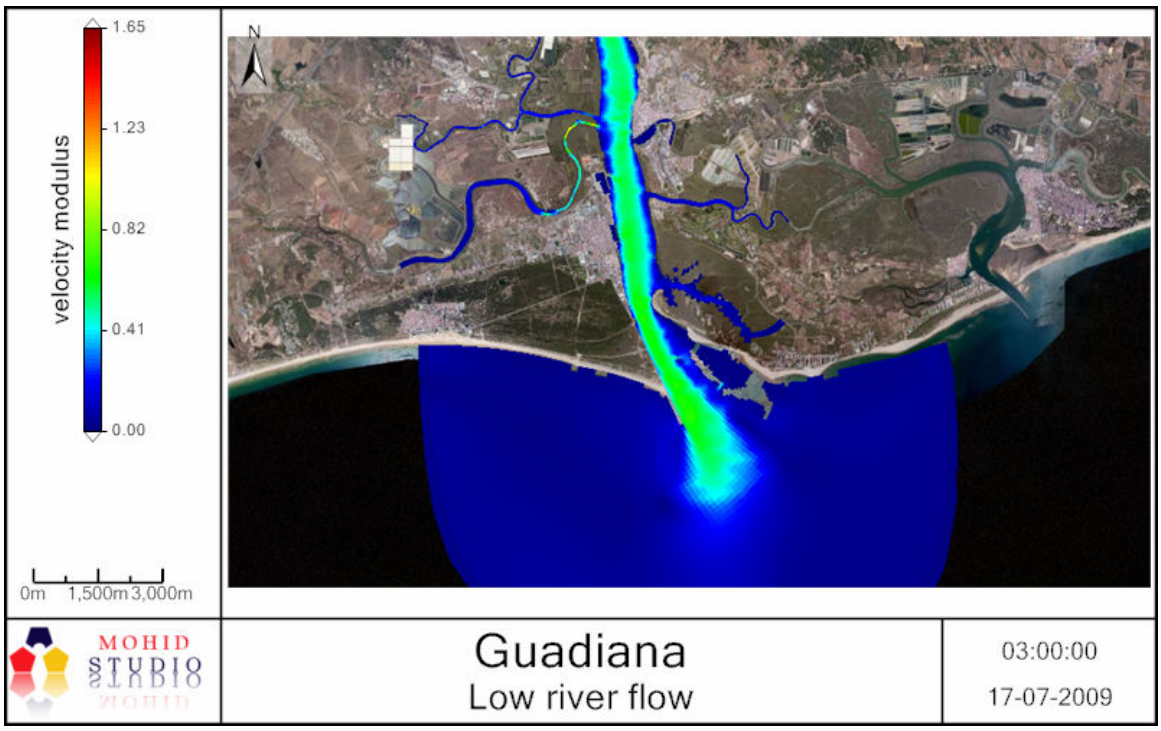


Figure 5.14. Neap tide, ebb velocities.

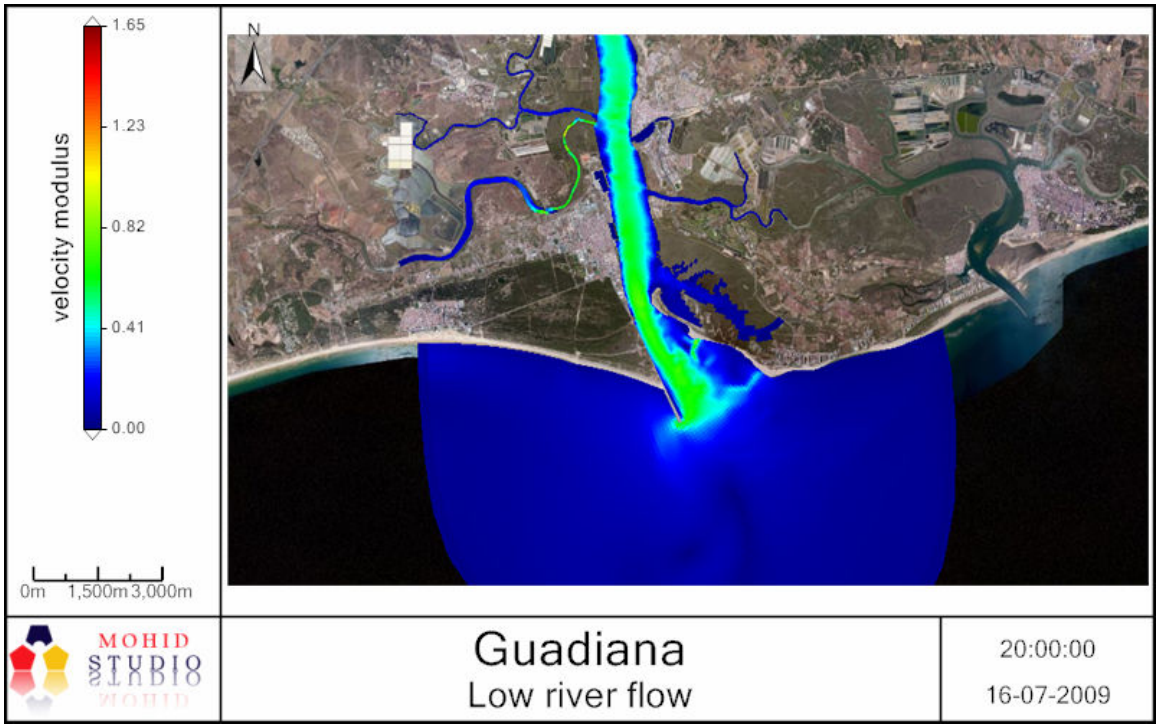


Figure 5.15. Neap tide, flood velocities.

5.2. Comparing results from different inputs processed in GIS

The initial model validated above was based on the grid data interpolated by MOHID's triangulation method from the original data points. The data included the shoreline (land) points but did not include the bathymetry estimated from the orthophoto. Then the several input

bathymetries for the model were created, based on the different processing methods in GIS. They were compared to that reference model (the initial model). The test bathymetries are:

0. Initial bathymetry interpolated without shoreline points.

1. Final bathymetry where old sparse points were replaced by bathymetry estimated from orthophoto, interpolated by triangulation.

2. Final bathymetry interpolated by Topo to Raster method in ArcGIS.

3. Final bathymetry interpolated by Kriging with varying anisotropy in (R,M) coordinates.

These bathymetry changes almost did not change resulting water lever but significantly changed velocities.

The results show that the absence of the shoreline points in the interpolated bathymetry lead to significantly worse results (figure 5.16). On the other hand, by including the bathymetry estimated from the orthophoto the result was improved, and by applying the advanced interpolation methods the results were improved even more (figure 5.16). And, the bathymetry interpolation in the channel-oriented coordinates significantly improved the direction of the current, which is visible in the change of the East velocity component (figure 5.17).

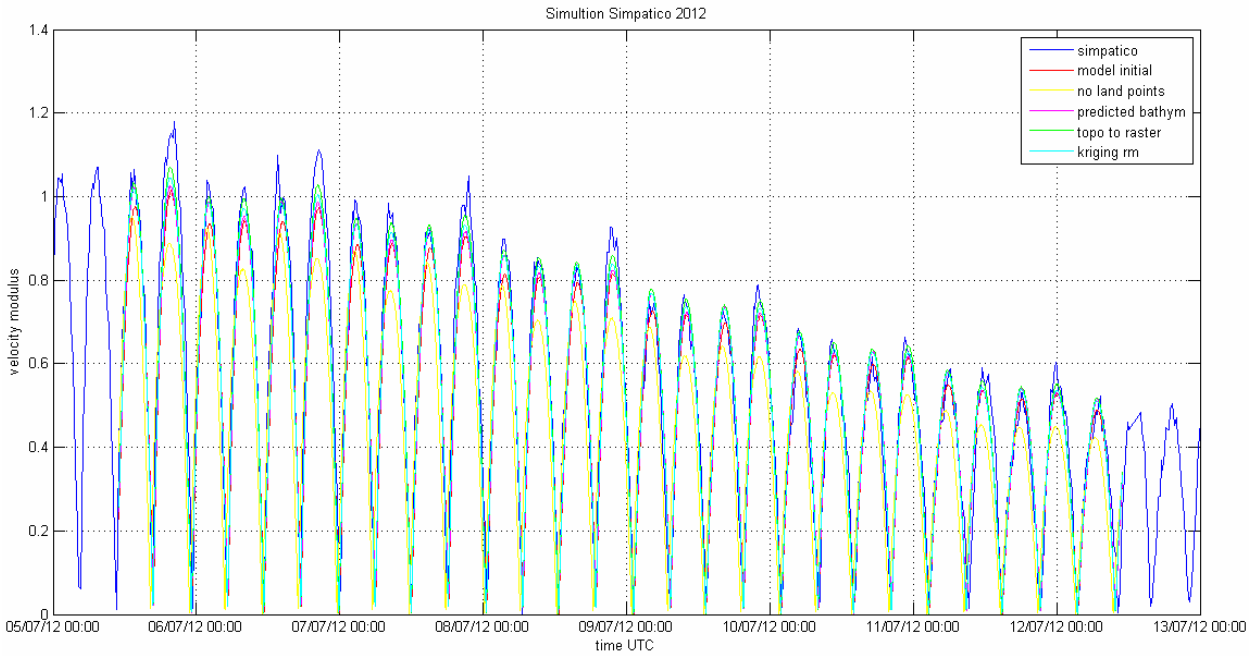


Figure 5.16. Comparing the models with different inputs at Simpatico station (velocity modulus).

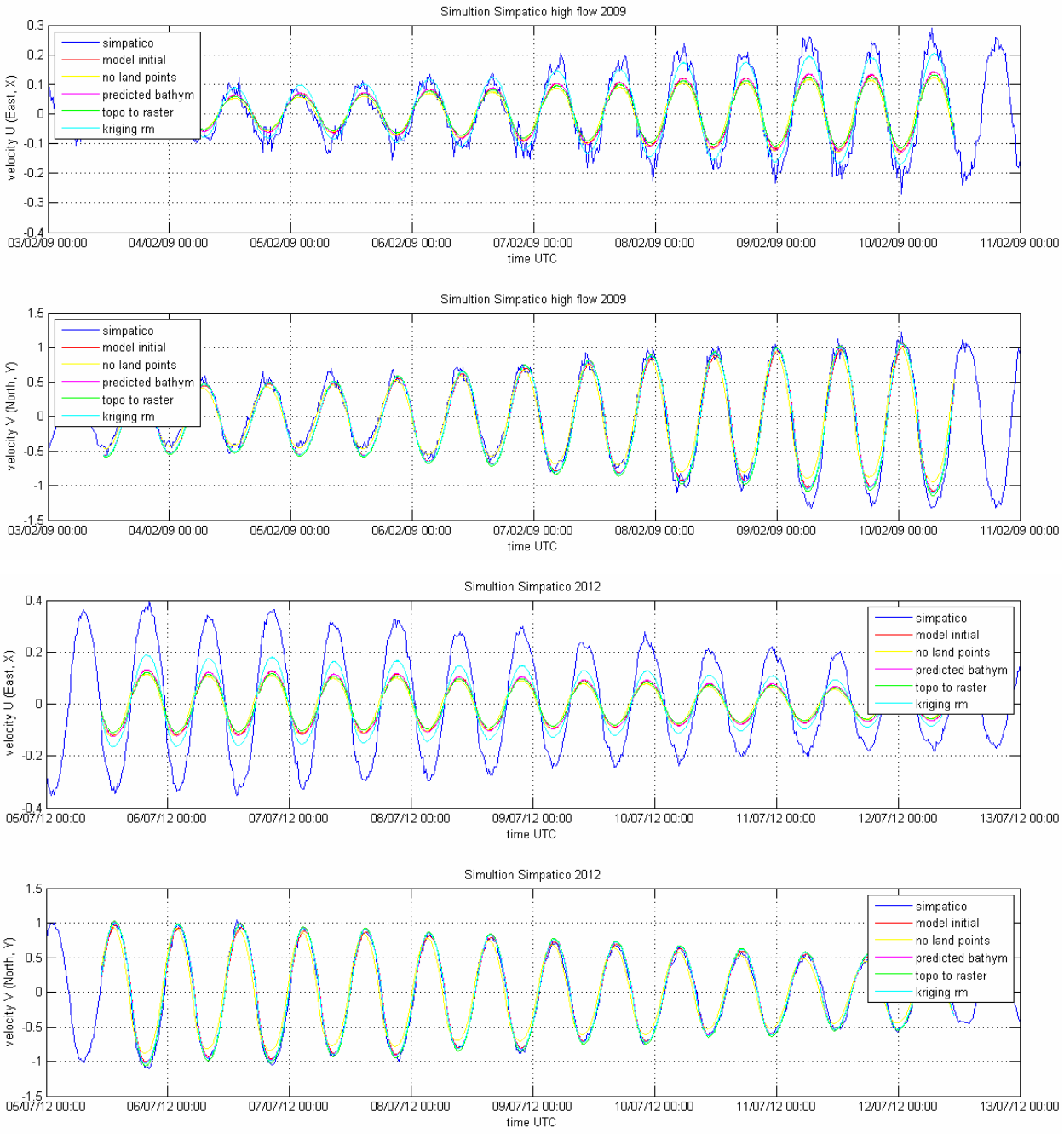


Figure 5.17. Comparing the models with different inputs at Simpatico station (velocity components).

The root mean square error was calculated for each scenario, and it showed that the last model with bathymetry interpolated after river straightening gave eventually the best result (figure 5.18).

The relative mean absolute error proposed by Walstra et al. (2001) revealed that velocity modulus and V component results could be qualified as excellent (with the lowest values in the last test run: about 0.18 for high flow 2009 and 0.08 for summer 2012), and velocity U improved from reasonable qualification in the models 0-3 to good level in the last run with the along-channel interpolation (from ~0.45 to 0.25 at high flow, from ~0.65 to 0.5 in 2012).

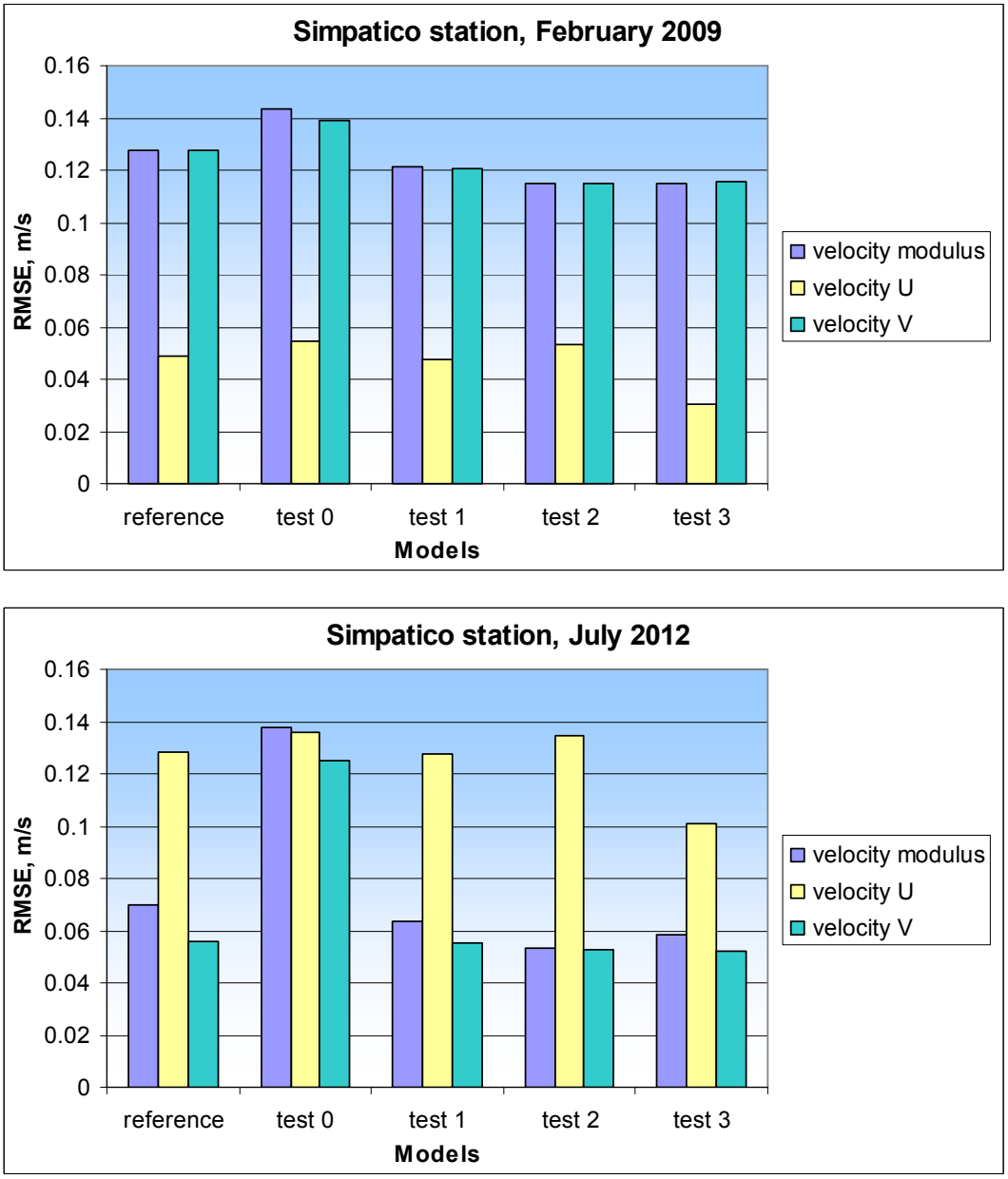


Figure 5.18. RMSE of the models with different inputs at Simpatico station.

6. Discussion

This chapter discusses the main achievements of this work. The main objective of this dissertation was to analyze and develop GIS-based techniques to improve the setup of hydrodynamic models, namely to increase the accuracy of model results by advanced pre-processing using the geospatial technologies. The Guadiana Estuary was used as a case study to demonstrate the methods and validate the results.

GIS/model integration

GIS is proved to be a very useful tool to help spatial discretization, input data processing and results visualization (Naoum, 2005; Merwade et al., 2008; Tsanis et al., 1996; Wang et al., 2012). However, a general GIS cannot fully support coastal modelling due to absence of capabilities for handling large temporal 3D datasets and curvilinear and unstructured grids. Indeed, the MOHID system is already integrated with GIS at high level (via the module MOHID GIS) for necessary basic pre- and post-processing. MOHID GIS allows to create curvilinear grids and to visualize animated results, which is necessary for the modelling process. On the other hand, advanced geospatial technologies can significantly improve, complement and extend spatial pre-processing of model inputs. In this work, remote sensing methods helped to obtain the model domain, geostatistics methods aided to the interpolation process, and they all together with GIS tools and programming helped to complement bathymetry dataset and to produce precisely described system geometry (the key spatial input - gridded bathymetry) which turned out critical for the model accuracy.

The most of the authors had concentrated on visualization capabilities of GIS, but not on advanced GIS tools for increasing the model accuracy (Ng et al., 2007; Yu et al., 2012; Tsanis and Boyle, 2001; Ng et al., 2009; Wang et al., 2012; Green and King, 2003). The literature mainly describes GIS/model integration via developing user-friendly GIS interface for visualization and easy operation, and none of the previous works proved the benefit of using GIS tools for model accuracy by validation on real measurements. There was only one study dedicated to improving the hydrodynamic model accuracy with GIS techniques by using along-channel interpolation of bathymetry data (Merwade et al., 2008), however, without actual hydrodynamic modelling. This work concentrated not on the coupling but on investigating GIS methods to improve model results via advanced pre-processing.

Workflow and methods

The workflow (figure 1.2) actually had several cycles and iterations.

The input data obtained from many different sources were significantly pre-processed. The

data needed for the inputs (figure 4.1) were obtained in 7 different coordinate systems. They were transformed into WGS 84 system using datum transformations (table 4.1), which induced small errors into the positions (unfortunately it is not documented in ArcGIS). Due to this fact and also due to the initial errors in the old data some bathymetry points appeared on the land and had to be removed.

The initial bathymetry datasets were redundant along the survey tracks but had gaps in shallow areas. Clustering in hexagons reduced the number of points to more than ten times than initial datasets and allowed to handle it easily. The water domain polygon was extracted from the orthophoto, and then used for removing wrong bathymetry points which appeared on the land. The best water body separation result was found using PCA for cluster analysis due to using the last principal components. PCA gave all three orthogonal components equal weights, thus the weights of the last components and the influence of their information significantly increased (figure 4.13). The classification result was estimated visually, however, a more precise method can be used. Classification accuracy can be tested by comparing with a set of known points. Also, object-based classification (with image segmentation) can produce better results than pixel-based.

The curvilinear grid for space discretization was generated in this water domain polygon. Then the first test interpolations were performed and the test simulations were run. This test result revealed significant problems in the interpolation and the grid. Then the domain polygon was edited manually in order to achieve such grid topology that allowed to implement open boundary conditions at the ocean (figure 4.23). The grid itself was improved by manually fitting some cell corners to the shoreline. The shoreline was converted into a sequence of points which were added into the bathymetry dataset.

The bathymetry data on the shoals were estimated from the orthophoto by the Linear Band method and the bathymetry dataset was also complemented by these new points. The empirical regression model was successful (the R^2 of 0.73 and RMSE of 0.6 m until 8 m depth) because the relationships between depth and optical signal are sensor and site specific and strongly depend on water clarity in the local. The most of estimated depths ranged from the topographic zero to about 6 m. Lyzenga (1978) showed that with smaller depths the error was less, so the actual error in the predicted depths is less than 0.6 m. Indeed, the values of estimated bathymetry are very close to the topographic zero at the points located near the shoreline. This result is better than results of the developers of the bathymetry estimation method (Lyzenga, 1985; Clark et al., 1987; Lyzenga et al., 2006).

Finally the different interpolation methods were tested and Kriging and Topo to Raster

(ANUDEM) were found the best. Kriging advantages were proved by many authors (Bello-Pineda and Hernández-Stefanoni, 2007; Medved et al., 2010) and ANUDEM was used by Daniell (2008). The semivariogram (figure 4.31) showed that the data had anisotropy so isotropic methods were not correct, however, kriging with uniform anisotropy was not good as well. Anisotropic kriging in channel-oriented coordinates produced much more realistic surface (figures 4.44, 4.45), as the method developers also showed (Wadzuk and Hodges, 2001; Merwade et al., 2005), because the anisotropy follows the centerline. RMSE of the interpolations could be calculated, however, all methods showed good results along the ship tracks (with very dense points), but the accurate interpolation was needed for the gaps between the tracks. Evaluation of interpolation results in the gaps was done by visual observation of the resulting surfaces and by validation of the hydrodynamic model results, which is discussed below.

Model results and limitations

The model results were quite good under most of the conditions and improved from the developed methods. However, the model had several limitations and assumptions which limited the quality of the results.

The available measurement data were not enough to calibrate precisely the long dynamic estuary under different conditions. There was only one calibration point with good dense recent data (Simpatico), one point with a few recent data (Ayamonte) and several old points with sparse and not very good measurements.

The river flow data from the SNIRH stations had some gaps especially for tributaries. The bathymetry data for the upper estuary were very sparse and old, and since the water is not clean there, it was impossible to estimate its depth from the orthophotos. In addition, recent orthophotos (or especially high-resolution satellite imagery with infrared bands) for the entire estuary taken in good weather at low river flow (with clean water) and different tidal conditions would have been very helpful for this study, for both depth estimation and shoreline extraction.

In the case of the model domain extending only until Alcoutim, like in the previous models of the other authors (Lopes, 2004; Lopes et al., 2003; Martins et al., 2004; Dias and Ferreira, 2001), the velocities were two times smaller comparing to the measured velocities in all calibration points (figures 5.1, 5.2). Additionally, the tidal asymmetry (relation between ebb and flood velocities) was completely wrong (figure 5.1). Calibration plots of Lopes (2004) and Lopes et al. (2003) also showed wrong tidal asymmetry and some underestimation of velocities, and they tried to fix it using very small rugosity value. They could decrease it because they had much coarser grid and thus still stable model.

The problem was that such domain presented only a half of the estuary which actually ends at small cascades above Mertola reached by the tidal signal (figure 4.47). After extending the model domain until its actual end the model results improved significantly at the all calibration stations (figures 5.3, 5.4). This could depend on the amount of water stored in the simulated system. The extended domain with the fine grid required much more computational power which was not available in the previous years.

The computed velocities are in good agreement with the observations under well-mixed conditions with low river flow. However, they are slightly lower than the measured velocities (figures 5.7, 5.8). This may result from the difference in bathymetry which highly influences flow velocities. The grid cell size in the area of measurement stations is about 30 m, and the station points may experience small local bathymetric variations.

But the presented model is 2D depth-integrated with only one vertical layer so it cannot produce good results when the river is stratified, which happens under high river flow conditions at neap tide (Garel et al., 2009a). Figure 5.6 shows that at these conditions the tidal asymmetry produced by the model (stronger ebb currents) is wrong comparing to the measurements (stronger flood currents) (Garel and Ferreira, in press). This may result in computing wrong net water transport. Also, 2D model is not able to show the opposite flow directions at the surface and the bottom which sometimes occur in the estuary under stratification (Garel and Ferreira, in press). Oliveira et al. (2006) also reported that the accuracy of the 2D simulations decreased under partially mixed and stratified conditions (at neap tide). Using a 3D model would allow to account influence of the baroclinic (density-driven) force and to predict the stratification and the secondary flow.

Advantages coming from using GIS

The several input bathymetries for the model were tested in a number of model runs and their results also compared to the measurements. These bathymetry changes significantly affected simulated velocities. Absence of the shoreline points in the interpolated bathymetry lead to significantly worse results (figure 6.1) and confirmed that including the shoreline into bathymetry data is essential for realistic interpolation (Bailly du Bois, 2011), and thus for correct model results.

Including the bathymetry estimated from orthophoto improved the accuracy of the simulations, and using advanced interpolation methods improved the results even more (figure 5.16). Because the current velocity in the shallow water depends a lot on the water depth, the precisely described system geometry due to correct interpolation (and fine curvilinear grid) is critical for the model results. As well, the model using very old bathymetry data in the mouth

produced wrong velocity fields and showed wrong places getting dry at low water. The model including the bathymetry estimated from the recent orthophoto has no such problems (figure 6.1)

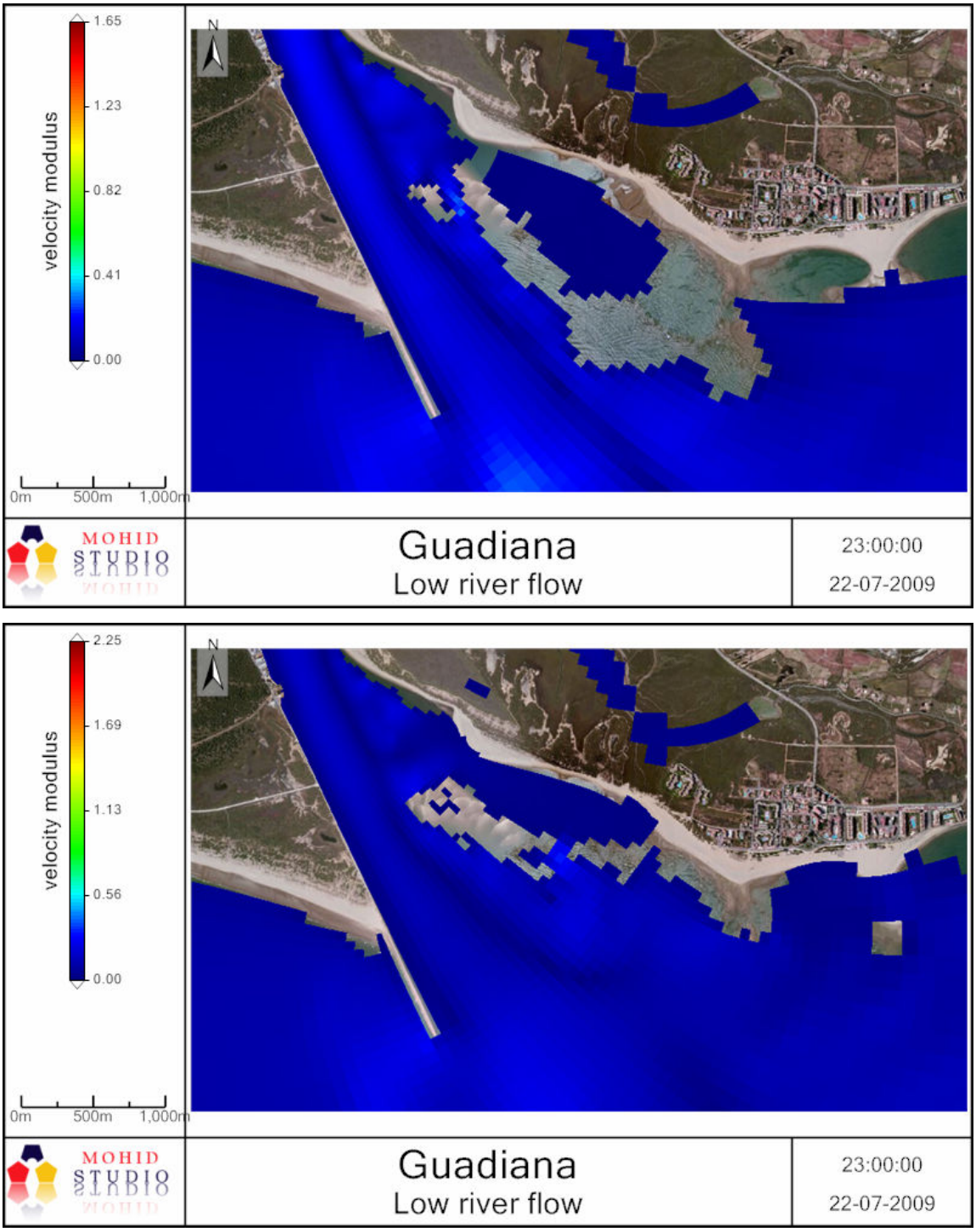


Figure 6.1. Low tide at the mouth using old sparse data (above) and estimated depths (below).

All isotropic interpolations of the long curved estuary produced artefacts on the resulting surfaces, and kriging with uniform anisotropy in north-south direction also showed wrong result at the curved parts of the channel (chapter 4.2.6. Bathymetry interpolations). The correct interpolation result was achieved only after transforming bathymetry points into channel-oriented coordinates using GIS tools and performing the interpolation with anisotropy along the

centerline in this transformed space (for “straightened” estuary). Because the bottom anisotropy is variable and follows the river centerline (thalweg), and the distribution of survey points is highly irregular along the ship tracks, only interpolation in channel-oriented coordinates which preserves the thalweg and the near-shore shoaling was acceptable for hydrodynamic modelling of the estuary (figure 4.45). This method induced small errors in the tributaries and ocean in some places far away from the centerline, but these distant places near the model boundary were not important for simulation. However, a wide estuary with large tributaries would require splitting the domain into parts and their separate interpolation.

The bathymetry interpolation in the channel-oriented coordinates significantly improved the direction of the water current (the East velocity component in the figure 5.17). Because the water flow in the narrow estuary is directed mainly along the thalweg and the shorelines, the anisotropic interpolation in the direction of the centerline allowed to represent the real geometry and to simulate this effect correctly. So the RMSE showed that the last model with bathymetry interpolated in the flow-oriented coordinates produced in general the best result (figure 5.18).

Indeed, the resulting velocities had significant errors in places where the bathymetry interpolation was not correct. Figure 6.2 shows the velocity field obtained using common isotropic interpolation and advanced method in the channel-oriented coordinates.

It allows to conclude that in a case of no bottom anisotropy the isotropic Topo to Raster (ANUDEM) method would be the best for bathymetry data, but in a case of long curved channel the only acceptable interpolation method is the channel-oriented anisotropic kriging.

So, validation of the model with different scenarios showed the truth of the common phrase in the field of computer science and modelling: “Garbage in, garbage out”. GIS tools are essentially needed for preparing accurate spatial inputs for coastal hydrodynamic modelling.

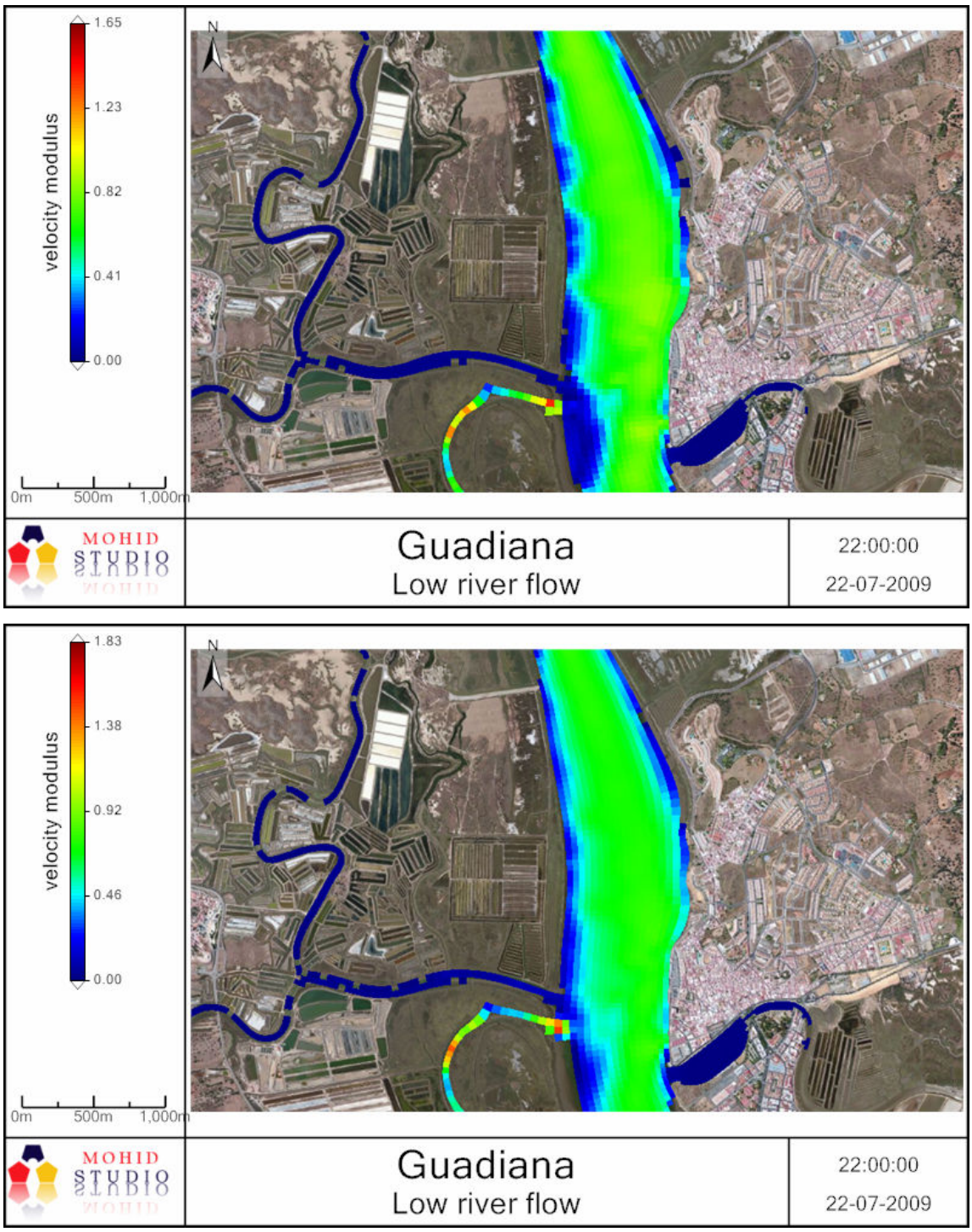


Figure 6.2. Velocities from two model scenarios using isotropic interpolation (above) and anisotropic along-channel (below).

7. Conclusion

The objective of this work was to analyze and develop GIS-based techniques to improve the setup of hydrodynamic models, namely to increase the accuracy of model results by advanced pre-processing methods. The finite volume model MOHID was used to test these methods and evaluate the associated improvements. The Guadiana Estuary was used to demonstrate the methods and validate the results.

The related literature mainly described development of user-friendly GIS interfaces for data visualization and easy model operation, and none of the previous works proved the benefit of using GIS tools for model accuracy by validation on real measurements.

GIS was proved to be a very useful tool to help spatial discretization, pre-processing of input data including editing, transformation, interpolation and computations. In addition, advanced GIS tools were used for analyzing and modelling the relationships between datasets (including remote sensing data) and statistical analyses.

GIS allowed to integrate geospatial data in several disparate coordinate systems using datum transformations. Then GIS tools were used to pre-process the model grid and bathymetry. MOHID GIS which is included into the modelling system helped with such necessary pre-processing tasks as curvilinear grid generation and grid data editing, and post-processing of temporal results. In addition, advanced GIS tools and geospatial technologies significantly improved, complemented and extended the pre-processing. Remote sensing methods helped to obtain the model domain, geostatistical methods helped the bathymetry interpolation, and they all together with GIS tools and programming helped to complement bathymetry dataset and to produce the correct gridded bathymetry, which turned out critical for the model accuracy.

The initial bathymetry datasets were redundant along survey tracks and clustering in hexagons reduced the number of points to reasonable amount. The water domain polygon was extracted from an orthophoto for generating the curvilinear grid for space discretization. The best land/water classification was based on PCA of spectral bands of the orthophoto and further pixel-to-pixel clustering. Regular points were generated along this shoreline and added into the bathymetry dataset. The bathymetry data in shallow areas of the river mouth were estimated from the orthophoto by linear band regression method and the bathymetry dataset was complemented by these points. This helped to fill the gaps and to replace sparse and outdated data in the bathymetry dataset. The bathymetry data was interpolated into curvilinear grids by several different methods and they were tested in the model and their results also compared to the measurements.

Including the shoreline points into bathymetry data was found essential for realistic interpolation and correct model results. Including bathymetry estimated from the orthophoto and using advanced interpolation methods improved the model accuracy comparing to the measurements. However, all isotropic interpolations produced artefacts on the resulting surfaces and then wrong velocity fields at these places. The correct interpolation result was achieved only after transforming bathymetry points into flow-oriented coordinates and performing the interpolation with anisotropy along the centerline. Using this channel-oriented interpolation in the model significantly improved the direction of the water current. The RMSE of simulated velocities showed that the last model with bathymetry interpolated in the flow-oriented coordinates produced in general the best result.

The hydrodynamic model results were quite good under most of the conditions and improved from the developed methods. However, the model had several limitations and assumptions which limited the quality of the results. The computed velocities were in good agreement with the observations under well-mixed conditions with low river flow. However, they were slightly lower than the measured velocities, and under stratified conditions the tidal asymmetry produced by the 2D model was wrong comparing to the measurements. The first model runs with the shorten domain produced wrong results but after extending the model domain until its actual end the model results improved significantly.

So, good quality of the spatial input data proved to be critical for model accuracy. Validation of the model with different scenarios showed that it is impossible to obtain good results with spatially incorrect input data, despite of all numerical calibration efforts. The model domain must cover entire estuary with all tributaries until disappearance of tidal signal. The bathymetry data should be interpolated into the grid by a method respecting the real bottom as precise as possible.

In general, the use of GIS tools to produce spatially accurate input data proved to be a valuable aid to modelling, significantly improving the model results.

References

Abbott, M., Basco, D., 1989. Computational fluid dynamics: an introduction for engineers, Longman Scientific and Technical, London.

Akcelik, V., Jaramaz, B., Ghattas O., 2001. Nearly Orthogonal Two-Dimensional Grid Generation with Aspect Ratio Control. Journal of Computational Physics 171, 805-821.

Amante, C., Eakins, B., Taylor, L., 2010. Spline interpolation of sparse bathymetric data for Digital Elevation Models (DEMs). Poster presented at The Fifth GEBCO Science Day, 15th September 2010, Callao, Peru.

Arakawa, A., Lamb, V.R., 1977. Computational design of the basic dynamical processes of the UCLA general circulation model. In Methods in Computational Physics, Chang, J. (Editor). Academic Press, New York, pp. 173-265.

Avena, G.C., Ricotta, C., Volpe, F., 1999. The influence of principal component analysis on the spatial structure of multispectral dataset. International Journal for Remote Sensing 20 (17), 3367-3376.

Baban, S.M.J., 1993. Evaluation of different algorithms for bathymetric charting of lakes using Landsat imagery. International Journal of Remote Sensing 14 (12), 2263-2273.

Bailly du Bois, P., 2011. Automatic calculation of bathymetry for coastal hydrodynamic models. Computers and Geosciences 37, 1303-1310.

Bello-Pineda, J., Hernández-Stefanoni, J.L., 2007. Comparing the performance of two spatial interpolation methods for creating a digital bathymetric model of the Yucatan submerged platform. Pan-American Journal of Aquatic Sciences 2 (3), 247-254.

Benny, A.H., Dawson, G., 1983. Satellite imagery as aid to bathymetric charting in the Red Sea. The Cartographic Journal 20 (1), 5-16.

Bernert, J.A., Sullivan, T.J., 1998. Bathymetric analysis of Tillamook Bay: comparison among bathymetric databases collected in 1867, 1957 and 1995. E and S Environmental Chemistry, Corvallis. - 46 p.

Blumberg, A.F., Kim, B.N., O'Neil, S., et al., 2000. Use of Orthogonal Curvilinear Grids for the Representation of the Littoral Ocean Environment. In Proceedings of Spring 2000 Simulation Interoperability Workshop, Paper No. 00S-SIW-118, March 2000, Simulation Interoperability Standards Organization, http://www.sisostds.org.

Bramante, J.F.; Raju, D.K., Min T.S., 2010. Derivation of bathymetry from multispectral imagery in the highly turbid waters of Singapore's south islands: A comparative study. DigitalGlobe 8 – Band Research Challenge 2010.

Braunschweig, F., 2012. MOHID Studio: An open source based graphical user interface for water models. Presented in the 3rd International MapWindow Open Source GIS Conference, 25-28 June, Velp (Netherlands). URL http://www.mapwindow.org/conference/2012/abstracts-overview.php#braunschweig

Braunschweig, F., Fernandes, L., Galvão, P., Trancoso, R., Pina, P., Neves, R., 2005. Mohid GIS – A geographical information system for water modeling software. In Geophysical Research Abstracts, EGU Meeting 05-A-08213, 25-29 April 2003, Wien (Austria).

Braunschweig, F., Leitao, P.C., Fernandes, L., Pina, P., Neves, R., 2004. The object-oriented design of the integrated water modelling system MOHID. Developments in Water Science 55, (2), 1079-1090.

Carter, G.S., Shankar, U., 1997. Creating rectangular bathymetry grids for environmental numerical modelling of gravel-bed rivers. Applied Mathematical Modelling 21 (11), 699-708.

Clark, R.K., Fay, T., Walker, C., 1987. Bathymetry calculations with Landsat 4 TM imagery under a generalized ratio assumption. Applied Optics 26 (19), 4036-4038.

Cleve, C., Kelly, M., Kearns, F.R., Moritz, M., 2008. Classification of the wildland-urban interface: A comparison of pixel- and object-based classifications using high-resolution aerial photography. Computers, Environment and Urban Systems 32 (4), 317-326.

Daniell, J.J., 2008. Development of a bathymetric grid for the Gulf of Papua and adjacent areas: A note describing its development. Journal of Geophysical Research 113 (F01S15).

Delgado, J., Nieto, J.M., T., Boski., 2010. Analysis of the spatial variation of heavy metals in the Guadiana Estuary sediments (SW Iberian Peninsula) based on GIS-mapping techniques. Estuarine, Coastal and Shelf Science 88 (2010) 71-83.

Dias, J., Ferreira, O., 2001. Projecto EMERGE "Estudo Multidisciplinar do Estuário do Guadiana". Relatorio final, CIACOMAR.

Dost, R.J.J., Mannaerts, C.M., 2004. Bathymetry generation using sonar and satellite imagery. Poster presented at ISPRS 2004: proceedings of the XXth ISPRS congress: Geo-imagery bridging continents, 12-23 July 2004, Istanbul, Turkey. Comm. IV.

Dost, R.J.J., Mannaerts, C.M., 2008. Generation of lake bathymetry using sonar, satellite imagery and GIS. In Proceedings of the 2008 ESRI international user conference: GIS, Geography in action, 4-8 August, San Diego, Florida. Ed. by J. Dangermond. Redmond: ESRI, 2008. - 5 p.

Driscoll, T.D., Stephen, A.V., 1998. Numerical conformal mapping using cross-ratios and Delaunay triangulation. SIAM Journal on Scientific Computing 19 (6), 1783-1803.

Du, J.-K., Feng, X.-Z., Wang, Z.-L., Huang, Y.-S., Ramadan, E., 2002. The methods of extracting water information from spot image. Chinese Geographical Science, 12 (1), 68 -72.

Fisher, C., Gustafson, W., Redmond. R., 2002. Mapping sagebrush/grasslands from Landsat TM-7 imagery: A comparison of methods. Report prepared for the USDI, Bu-reau of Land Management Montana/Dakota State Office. The University of Montana, 11 December 2002.

Frazier, P.S., Page, K.J., 2000. Water body detection and delineation with Landsat TM Data. Photogrammetric Engineering and Remote Sensing 66 (12), 1461-1467.

Garel, E., Ferreira Ó., 2011. Monitoring estuaries using non-permanent stations: practical aspects and data examples. Ocean Dynamics 61, 891-902.

Garel, E., Ferreira, Ó., Fortnightly changes in water transport direction across the mouth of a narrow estuary. Estuaries and Coasts, in press.

Garel, E., Pinto, L., Santos, A. Ferreira, Ó., 2009a. Tidal and river discharge forcing upon water and sediment circulation at a rock-bound estuary (Guadiana Estuary, Portugal). Estuarine, Coastal and Shelf Science 84 (2), 269-281.

Garel, E., Nunes, S., Neto, J.M., Fernandes, R., Neves, R., Marques, J.C., Ferreira, Ó. 2009b. The autonomous Simpatico system for realtime continuous water-quality and current velocity monitoring: examples of application in three Portuguese estuaries. Geo-Marine Letters 29, 331-341.

Gens, R., 2010. Remote sensing of coastlines: detection, extraction and monitoring. Review article. International Journal of Remote Sensing 31 (7), 1819-1836.

Gonzalez, R., Dias, J.A., Ferreira, Ó., 2005. Analysis of Land-Cover Shifts in Time and Their Significance. In High Resolution Morphodynamics and Sedimentary Evolution of Estuaries, FitzGerald, D., Knight, J. (Eds). Springer Netherlands, pp. 57-82.

Gordon, H.R., Brown, O.B., 1974. Influence of bottom depth and albedo on the diffuse

reflectance of a flat homogeneous ocean. Applied Optics 13 (9), 2153-2159.

Green, D., King. S., 2003. Chapter 21. Progress in Geographical information systems and coastal modeling: an overview. Lakhan, V.C. (editor), Advances in Coastal Modeling, 67, 553-580.

Hell, B., Jakobsson, M., 2011. Gridding heterogeneous bathymetric data sets with stacked continuous curvature splines in tension. Marine Geophysical Research 32 (4), 493-501.

Hutchinson, M.F., 1988. Calculation of hydrologically sound digital elevation models. Paper presented at the Third International Symposium on Spatial Data Handling at Sydney, Australia.

Hutchinson, M.F. 1989. A new procedure for gridding elevation and stream line data with automatic removal of spurious pits. Journal of Hydrology 106, 211-232.

Jarvis, A., Reuter, H.I., Nelson, A., Guevara, E., 2008. Hole-filled seamless SRTM data V4, International Centre for Tropical Agriculture (CIAT), available from http://srtm.csi.cgiar.org.

Jolma, A., Ames, D.P., Horning, N., Mitasova, H., Neteler, M., Racicot, A., Sutton, T., 2008. Free and open source geospatial tools for environmental modelling and management. Environmental Modelling, Software and Decision Support 3, 163-180.

Kuzmin, D., 2010. A Guide to Numerical Methods for Transport Equations. Friedrich-Alexander-Universität Erlangen-Nürnberg. - 220 p.

Le Bris, A., Boldo, D., 2007. Extraction of landcover themes out of aerial orthoimages in mountainous areas using external information. In: Stilla, U., et al (Eds) PIA07. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36 (3/W49A).

Ledoux, H., Gold, C.M., 2004. An efficient natural neighbour interpolation algorithm for geoscientific modelling. In Fisher P., (ed.) Developments in Spatial Data Handling – 11th International Symposium on Spatial Data Handling, Springer, pp. 97-108.

Lee, I.-C., Wu, B., Li, R., 2009. Shoreline extraction from the integration of LIDAR point cloud data and aerial orthophotos using mean shift segmentation. ASPRS 2009 Annual Conference, March 9-13, Baltimore, Maryland, USA.

Lee, Z., Hu, C., Casey, B., Shang, S., Dierssen, H., Arnone R., 2010. Global shallow-water bathymetry from satellite ocean color data. Eos, Transactions American Geophysical Union 91 (46), 429-430.

Lee, Z., Carder, K.L., Mobley, C.D., Steward, R.G., Patch J.S., 1999. Hyperspectral remote sensing for shallow waters: 2. Deriving bottom depths and water properties by optimization. Applied Optics 38 (18), 3831-3843.

Li, R., Di, K., Ma, R., 2001. A comparative study of shoreline mapping techniques. In: Proceeding of the 4th International Symposium on Computer Mapping and GIS for Coastal Zone Management, June 18. Halifax, Nova Scotia, Canada. Retrieved on March 2013 from http://shoreline.eng.ohio-state.edu/publications/coastalgis2001_li.pdf

Li, W., Liang, T., Wang X., 2009. Remote sensing image extraction and precision analysis for alpine wetland based on coupling analysis of multispectral factor PCA and decision tree. In Proceedings of the 2nd International Congress on Image and Signal Processing, CISP'09.

Liang, S.-J., Molkenthin, F., 2001. A virtual GIS-based hydrodynamic model system for Tamshui River. Journal of Hydroinformatics, 3 (4), 195-202.

Lichy, C., 1998. A multidirectional and multifunctional gateway between GIS and hydrodynamic models. In: Babovic, V., Larsen, L. (Eds.), Proceedings of Hydroinformatics 98, Copenhagen, Denmark, pp. 395-399.

Lipakis, M., Chrysoulakis, N., Kamarianakis, Y., 2008. Shoreline extraction using satellite

imagery. In: Pranzini, E. and Wetzel, E. (eds), Beach Erosion Monitoring. Results from BEACHMED/e-OpTIMAL Project (Optimization des Techniques Integrées de Monitorage Appliquées aux Lottoraux) INTERREG IIIC South. Nuova Grafica Fiorentina, Florence, Italy, pp. 81 – 95.

Liu, X., Ramirez, R., 1997. Automatic extraction of hydrographic objects in digital orthophoto images. In Digital Orthophoto Images Proceeding of GIS/LIS, 1997, pp. 365-373.

Lobo, J., Plaza, F., Gonzales, R., Dias, J., Kapsimalis, V., Mendes, I, Rio, V., 2004. Estimations of bedload sediment transport in the Guadiana Estuary (SW Iberian Peninsula) during low river discharge periods. Journal of Coastal Research 41 (Special Issue), 12-26.

Lopes, J., Neves, R., Dias, J., Martins, F., 2003. Calibração de um sistema de modelação para o Estuário do Guadiana. Thalassas (especial issue 4º simposio sobre el margen continental ibérico atlántico) 19 (2), 155-156.

Lopes, João M.M., 2004. Modelação Matemática do Transporte de Sedimentos no Estuário do Guadiana. Master thesis in Mechanical Engineering, Universidade do Minho, Portugal.

Lyzenga, D.R., 1977. Reflectance of a flat ocean in the limit of zero water depth. Applied Optics 16 (2), 282-283.

Lyzenga, D.R., 1978. Passive remote sensing techniques for mapping water depth and bottom features. Applied Optics 17 (3), 379-383.

Lyzenga, D.R., 1985. Shallow-water bathymetry using combined LIDAR and passive multispectral scanner data. International Journal of Remote Sensing 6 (1), 115-125.

Lyzenga, D.R., Malinas, N.P., Tanis, F.J., 2006. Multispectral bathymetry using a simple physically based algorithm. IEEE Transactions on Geoscience and Remote Sensing 44 (8), 2251-2259.

MacKay, B., 2006. Mathematical and Statistical Models. Science Education Resource Center (SERC) at Carleton College. Retrieved in March 2013 from http://serc.carleton.edu/introgeo/ mathstatmodels/index.html

Martins, Flávio, 1999. Modelação Matemática Tridimensional de Escoamentos Costeiros e Estuarinos usando uma Abordagem de Coordenada Vertical Genérica. PhD Thesis, Technical University of Lisbon, Instituto Superior Técnico.

Martins, F., 2012. Numerical modelling of currents and pollutant dispersion in coastal and ocean systems. 2nd Summer School on Dynamics and Management of the Mediterranean Marine Environment (DME2), 18-29 August, Zakynthos, Greece, pp. 57-84.

Martins, F., Janeiro, J., Venâncio, A., Brito, A., Lopes, J., Neves, R., 2006. Programa de Monitorização: Na área de atendimento das Águas do Algarve. Relatório Final, Universidade do Algarve, Março de 2006. - 539 p.

Martins, F., Leitão, P.C., Silva, A., Neves, R., 2001. 3D modelling of the Sado Estuary using a new generic vertical discretization approach. Oceanologica Acta 24, 51-62.

Martins, F., Neves, R., Leitão, P., 1998. A three-dimensional hydrodynamic model with generic vertical coordinate. In: Babovic, V., Larsen, L. (Eds.), Proceedings of Hydroinformatics 98, Vol. 2. Copenhagen, Denmark, pp. 1403–1410.

Martins, F., Pina, P., Braunschweig, F., Saraiva, S., Santos, M., Neves, R., 2004. EU water framework directive: will the nitrate load reduction from diffuse sources produce the same results in all estuaries? In Proceedings of Ecohydraulics'04 International Conference, Madrid, 2004. - 4 p.

McFeeters, S.K., 1996. The use of the Normalized Difference Water Index (NDWI) in the

delineation of open water features. International Journal of Remote Sensing 17 (7), 1425-1432.

Medved, I., Pribicevic, B., Medak, D., Kuzmanic, I., 2010 (in Croatian). Comparison of interpolation methods of bathymetry data used for monitoring of lake volume change. (Usporedba metoda interpolacije batimetrijskih mjerenja za pracenje promjena volumena jezera). Geodetski list 64(87) (2), 71-86.

Merwade V.M., Maidment, D.R., Hodges, B.R., 2005. Geospatial representation of river channels. Journal of Hydrologic Engineering 10 (3), 243-251.

Merwade, V.M., Maidment, D.R., Goff, J.A., 2006. Anisotropic considerations while interpolating river channel bathymetry. Journal of Hydrology 331 (3-4), 731-741.

Merwade, V., Cook, A., Coonrod, J., 2008. GIS techniques for creating river terrain models for hydrodynamic modeling and flood inundation mapping. Environmental Modelling and Software 23, 1300-1311.

Miranda, R., Braunschweig, F., Leitão, P., Neves, R., Martins, F., Santos, A., 2000. MOHID 2000 – A coastal integrated object oriented model. Hydraulic Engineering Software VIII, Ed. W.R. Blain and C.A. Brebbia, WIT press, pp. 393-401.

Mitasova, H., 2000. Multi-Scale Characterization and Simulation of Near-Shore Environment Using Advanced GIS Technology. Detailed Proposal, 3 p. URL http://skagit.meas.ncsu.edu /~helena/publwork/coast02/finnrc.html.

Mitasova, H.L., Mitas, L., Brown, B.M., Gerdes, D.P., and Kosinovsky, I., 1995. Modeling spatially and temporally distributed phenomena: New methods and tools for GRASS GIS. International Journal of Geographical Information Systems: A special issue on integration of environmental modeling and GIS, 9 (4), 443-446.

MOHID Water Modelling System. 2002. URL http://www.mohid.com

Morais, P., Martins, F., Chícharo, M.A., Lopes, J., Chícharo, L., 2013. MOHID as a tool to evaluate the impact of water discharge from dams on the advection of estuarine fish larval stages. In Ocean modelling for coastal management - Case studies with MOHID. Eds. M. Mateus and R. Neves, Lisboa IST Press; pp. 143-160.

Morales, J.A., 1997. Evolution and facies architecture of the mesotidal Guadiana River delta (S.W. Spain--Portugal), Marine Geology 138 (1-2), 127-148.

Naoum, S., Tsanis, I.K., Fullarton, M., 2005. A GIS pre-processor for pollutant transport modelling. Environmental Modelling and Software 20 (1), 55-68.

Nath, R.K., Deb, S.K., 2010. Water-body area extraction from high resolution satellite images – an introduction, review, and comparison. International Journal of Image Processing (IJIP) 3 (6), 353-372.

Navas, J.M., Telfer, T.C., Ross, L.G. 2011., Application of 3D hydrodynamic and particle tracking models for better environmental management of finfish culture. Continental Shelf Research 31, 675-684.

Ng, S.M.Y., Wai, O., Li, Y.-Sh., Li, Z.-L., Jiang, Y., 2009. Integration of a GIS and a complex three-dimensional hydrodynamic, sediment and heavy metal transport numerical model. Advances in Engineering Software 40, 391-401.

Ng, S.M.Y., Wai, O.W.H., Li, Y.S., Xu, Z.H., Chen, H.L. Li, Z.L., 2007. Development of a GIS for managing dynamic, 3D coastal information of Pearl River Estuary. Journal of Hydroinformatics 9 (3), 215-232.

Ng, Sandy Man-Yi., 2006. Development of a Geographic Information System (GIS) for managing and predicting coastal water quality in the Pearl River Estuary (South China). PhD

thesis, The Hong Kong Polytechnic University.

Nguyen, D., 2012. Water body extraction from multi spectral image by spectral pattern analysis. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXIX-B8, 2012. XII ISPRS Congress, 25 August – 01 September, Melbourne, Australia.

Oliveira, A., Fortunato, A. and Pinto, L., 2006, Modelling the hydrodynamics and the fate of passive and active organisms in the Guadiana Estuary. Estuarine, Coastal and Shelf Science 70 (1-2), 76-84.

Paredes, J.M., Spero, R.E., 1983. Water depth mapping from passive remote sensing data under a generalized ratio assumption. Applied Optics 22 (8), 1134-1135.

Peng, S., Fu, G., Zhao, X., 2010. Integration of USEPA WASP model in a GIS platform. Journal of Zhejiang University-SCIENCE A (Applied Physics and Engineering) 11 (12), 1015-1024.

Pierce, B.G., Law, Y., 2008. High resolution lake edge extraction from colour orthophotography. Proceedings of the 10th Circumpolar Remote Sensing Symposium / 29th Canadian Symposium on Remote Sensing, Whitehorse, Yukon, Canada.

Pina, P., Martins, F., Chambel Leitão, P., Braunchweig, F., Neves, R., 2003. Development of an integrated system for coastal waters. In Proceedings of MERIS User Workshop (ESA SP-549). 10-13 November 2003, ESA-ESRIN, Frascati, Italy. Published on CDROM., p. 33.1.

Pinho, J.L.S., Pereira Vieira, J.M., Antunes do Carmo, J.S., 2004. Hydroinformatic environment for coastal waters hydrodynamics and water quality modelling. Advances in Engineering Software 35, 205–222.

Pires, Patrícia C.M., 2006. Desenvolvimento de uma metodologia de avaliação de riscos ambientais para apoiar a elaboração de planos de emergência. Dissertação de Mestrado em Ciência e Sistemas de Informação Geográfica. Instituto Superior de Estatística e Gestão de Informação (ISEGI), Universidade Nova de Lisboa.

Polcyn, F.C., Brown, W.L., Sattinger, I.J., 1970. The measurement of water depth by remote sensing techniques, Report 8973-26-F, Willow Run Laboratories, U. Michigan, Ann Arbor.

Qiao, C., Luo, J., Sheng, Y., Shen, Z., Zhu, Z., Ming, D., 2012. An adaptive water extraction method from remote sensing image based on NDWI. Journal of the Indian Society of Remote Sensing 40 (3), 421-433.

R Development Core Team., 2010. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Rogala, J.T., 1999. Methodologies employed for bathymetric mapping and sediment characterization as part of the Upper Mississippi River System Navigation Feasibility Study. Bathymetry – Chapter 2. ENV Report 13. Interim Report for the Upper Mississippi River - Illinois Waterway System Navigation Study.

Saraiva, S., Pina, P., Martins, F., Santos, M., Braunschweig, F. and Neves, R., 2007. Modelling the influence of nutrient loads on Portuguese estuaries. Hydrobiologia 587 (1), 5-18.

Sharma, O., Mioc, D., Anton F., 2008. Polygon feature extraction from satellite imagery based on colour image segmentation and medial axis. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B3a. Beijing 2008.

Shataee, S., Najjarlou S., 2007. Up to date mapping of reforested area using multi-dates ETM+ data. Journal of Applied Sciences 7 (7), 972-977.

Sheng, Y.P., 2003. Advances in modeling of estuarine and coastal environments. In Proceedings of Second MIT Conference on Computational Fluid and Solid Mechanics, 17-20 June, pp. 1805-1809.

Shih, S.F., 1986. Comparison of ELAS classifications and density slicing Landsat data for water-surface area assessment. Hydrologie Applications of Space Technology. In Proceedings of the Cocoa Beach Workshop, Florida, August 1985. IAHS Publ. no. 160.

Smith, W.H.F., Wessel, P., 1990. Gridding with continuous curvature splines in tension. Geophysics 55 (3), 293-305.

Sokoletsky, L., 2005. Comparative analysis of selected radiative transfer approaches for aquatic environments. Applied Optics 44 (1), 136-148.

Sparis, P.D., 1985. A method for generating boundary orthogonal curvilinear coordinate systems using biharmonic equation. Journal of Computational Physics 61, 445-462

Stove, G., 1985. Use of high resolution satellite imagery in optical and infrared wavebands as an aid to hydrographic and coastal engineering. Proceedings Conference on Electronics in Soil and Gas, London, January 1985 (Twickenham: Cahners Exhibitors), pp. 509-530.

Stumpf, R.P., Holderied K., Sinclair M., 2003. Determination of water depth with high-resolution satellite imagery over variable bottom types. Limnology and Oceanography 48 (1, part 2), 547-556.

Thompson, J.F., 1982. Foreword. Applied Mathematics and Computation, Volumes 10-11.

Thompson, J.F., Warsi, Z.U A., Mastin, C.W., 1982. Boundary-fitted coordinate systems for numerical solution of partial differential equations - A review. Journal of Computational Physics 47, 1-108.

Thompson, J.F., Warsi, Z.U.A., Mastin, C.W., 1985. Numerical Grid Generation: Foundations and Applications. Elsevier Science Publishing Co., Inc. - 483 p.

Tironi, A., Marin, V.H., Campuzano, F., 2008. A management tool for salmon aquaculture: Integrating MOHID and GIS applications for local waste management, In: Perspectives on Integrated Coastal Zone Management in South America, R. Neves, J. Baretta and M. Mateus (eds.), IST Press. pp. 585-595.

Tsanis I.K., C. Valeo, J. Wu, S. Boyle., 1996. Managing Contaminated Sediments Using a Hydrodynamic Model and a Gis, Environmental Technology 17 (8), 877-883.

Tsanis, I.K., Boyle, S., 2001. A 2D hydrodynamic-pollutant transport GIS model. Advances in engineering software 32, 353-361.

Versteeg, H.K., Malalasekera, W., 1995. An introduction to computational fluid dynamics the finite volume method. Longman Scientific and Technical. - 257 p.

Versteeg, H.K., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics. The finite volume method. Second Edition. Pearson Education Limited.

Vogt, J.V., et al., 2007. A pan-European River and Catchment Database. European Commission – JRC (Report EUR 22920 EN), Luxembourg. - 120 p.

Volakos, N.P., Barber, R.W., 2001. Comparing interpolation methods for processing randomly scattered bathymetric data. In Proceedings of the 5th International Conference on Computer Modelling of Sea and Coastal Regions (CE 2001), Rhodes, Greece, September 2001, WIT Press, pp. 161-172.

Wadzuk, B., Hodges, B.R., 2001. Model bathymetry for sinuous, dendritic reservoirs. Presented at the 6th International Workshop on Physical Processes in Natural Waters, University of Girona, Catalonia, Spain, June 2001.

Walstra, D.J.R., Van Rijn, L.C., Blogg, H., Van Ormondt, M., 2001. Evaluation of a hydrodynamic area model based on the COAST3D data at Teignmouth 1999. Report TR121-EC MAST, Project No MAS3-CT97-0086, HR Wallinford, UK, pp. D4.1-D4.4.

Wang, L., Zhao, X., Shen, Y., 2012. Coupling hydrodynamic models with GIS for storm surge simulation: application to the Yangtze Estuary and the Hangzhou Bay, China Front. Earth Science 6 (3), 261-275.

Wessel, P., Smith, W.H.F., 1996. A global self-consistent, hierarchical, high-resolution shoreline database. Journal of Geophysical Research 101 (B4), 8741-8743.

Wolanski, E., Chícharo, L., Chícharo, M.A., Morais, P., 2006. An ecohydrology model of the Guadiana Estuary (South Portugal). Estuarine, Coastal and Shelf Science 70, 132-143.

Wright, D., Bartlett, D., (Eds), 1999. Marine and Coastal Geographical Information Systems. Taylor and Francis.

Yarbrough, L.D., Easson, G., 2003. Comparison of techniques for deriving bathymetry from remotely sensed data. AMRS – Conference 2003: Hyperspectral Issues for Coastal Zone Environments.

Yu, J.J., Qin, X.S., Larsen, L.C., Larsen, O., Jayasooriya, A., Shen, X.L., 2012. A GIS-based management and publication framework for data handling of numerical model results. Advances in Engineering Software 45 (2012), 360-369.

Zhu, Y., Neto, J.C., Nd. Recognition of lakes from remotely sensed imagery. University of Nebraska – Lincoln - CSCE 896 Computational Aspects of GIS. - 6 p.

Appendixes

Appendix 1. Python script "Creating regular grid of hexagons"

```
#-------------------------------------------------------------------------------
# Name:        CREATING REGULAR GRID OF HEXAGONS
# Author:      Nadiia Basos a43682
#-------------------------------------------------------------------------------
import math as m

# Inputs                                                                    ##

#rad=100                # radius of circumcircle=side of hexagon
#rad=int(raw_input('Radius of circumcircle (Side of hexagon): '))
inrad=5                 # radius of incircle
rad=2.0*m.sqrt(inrad*inrad/3.0)
row=1000                # number of rows
col=225                 # double number of columns
Xorig=636630            # x coordinate of origin
Yorig=4116840           # y coordinate of origin
#-------------------------------------------------------------------------------

import time
start=time.time()

# Defining function
from math import *

def CriaHexagono(r):                    # function by professor J. Rodrigues
    p1=[-r+Xorig,0+Yorig]
    p2=[-r*cos(pi/3.0)+Xorig,r*sin(pi/3.0)+Yorig]
    p3=[r*cos(pi/3.0)+Xorig,r*sin(pi/3.0)+Yorig]
    p4=[r+Xorig,0+Yorig]
    p5=[r*cos(pi/3.0)+Xorig,-r*sin(pi/3.0)+Yorig]
    p6=[-r*cos(pi/3.0)+Xorig,-r*sin(pi/3.0)+Yorig]
    return [p1,p2,p3,p4,p5,p6,p1]

def Translacao(Tx, Ty, Hexagono):
    H=[]
    for p in Hexagono:
        H.append([p[0]+Tx, p[1]+Ty])
    return H
#-------------------------------------------------------------------------------


# Function to save hexagons as a shapefile (for ArcGIS)
import osgeo.ogr as ogr

def CreateShapefilePolyg(hexlist):                # defining module, hexagon -
parameter (list of lists (hexagons))
    drv=ogr.GetDriverByName("ESRI Shapefile")     # starting driver for shapefiles
    drv.DeleteDataSource("./output/Hexagons_tribut.shp")          # cleaning
existing file                          ##
    tema=drv.CreateDataSource("./output/Hexagons_tribut.shp")     # creating new
    layer=tema.CreateLayer("0",None,ogr.wkbPolygon,"") # creating layer, name=0,
SpatialReference=None,
                                                       # " " could be some values,
type - polygon
    # creating attributes (fields)
    field1=ogr.FieldDefn("Unique_ID",ogr.OFTInteger)     # defining a field,
name=idhex, type=integer
    layer.CreateField(field1)                         # creating a field in the layer
"layer"
```

```
    # creating attribute table and geometry (filling up the fields)
    for t in range(0,row*col):                      # for each hexagon
        entdfn=layer.GetLayerDefn()                 # getting definition of the layer
        entity=ogr.Feature(entdfn)                  # creating object of class feature
(line in table)
        entity.SetField("Unique_ID",int(t))         # writing value of the current key
in "idhex" field
        ring = ogr.Geometry(ogr.wkbLinearRing)      # creating empty ring
        for k in range(0,len(h)-1):
            X,Y=hexlist[t][k][0],hexlist[t][k][1]  # gettig coords x,y of each
hexagon vertex (0-6)
            ring.AddPoint(X,Y)                      # addind vertex to the ring
        polygon = ogr.Geometry(ogr.wkbPolygon)      # creating a polygon
        polygon.AddGeometry(ring)
        entity.SetGeometry(polygon)                 # setting geometry to the feature
        layer.CreateFeature(entity)                 # writing this feature in the
layer (line to shapefile table)
    tema.Release()                                  # shapefile is written on disk C
#-----------------------------------------------------------------------------


# Creating grid
import math as m

Kx, Ky=rad,rad+5
Ty=rad*m.sin(m.pi/3.0)+10
hexagons=[]
for i in range(row):
    Ty=rad*m.sin(m.pi/3.0)*i+Ky
    for j in range(col):
        h=CriaHexagono(rad)
        Tx=2.0*(rad+rad*m.cos(m.pi/3))*j+Kx+(rad+rad*m.cos(m.pi/3))*(i%2)  #n%m
ostatok deleniya
        h=Translacao(Tx, Ty, h)
#        print h
        hexagons.append(h)
#print hexagons
print ' '
CreateShapefilePolyg(hexagons)
#-----------------------------------------------------------------------------


end=time.time()
print 'took', end-start, 'seconds'
```

Appendix 2. Python script "Decreasing the number of data points"

```
#-------------------------------------------------------------------------------
# Name:        DECREASING THE NUMBER OF DATA POINTS
# Author:      Nadiia Basos a43682
# Inputs:      Bathymetry points with hexagon IDs shapefile
#-------------------------------------------------------------------------------


inshp='./input_decreasing_all_data/morales_2010_2h_SpatialJoin.shp'
outshp="./output_decreasing_all_data/morales_2010_2h_inh2m.shp"

import time
start=time.time()

import osgeo.ogr as ogr

# Reading the shapefile (bathymetry points with IDs of overlaying hexagons)
drv=ogr.GetDriverByName('ESRI Shapefile')   # starting driver for shapefiles
allbathym=drv.Open(inshp)                   # open shp-file
lay=allbathym.GetLayer()                    # get layer
laydef=lay.GetLayerDefn()                   # get its definition
#print "Geometry:",laydef.GetGeomType() # => Geometry: 1 (points)
ncol=laydef.GetFieldCount()                 # number of fields in attribute table
print ncol, 'fields in the "allbathym" attribute table'
listAttr=[]                                 # new empty list
for i in range(0,ncol):                     # do next two lines "ncol" times
    fld=laydef.GetFieldDefn(i)              # object of the class FieldDefn
    listAttr.append(fld.GetNameRef())       # get field name and put in into the list
print 'field names', listAttr
nFeat=lay.GetFeatureCount()                 # how many features (points) in shapefile
print nFeat, 'features (points) in the shapefile "allbathym"'
prj=lay.GetSpatialRef()
#-------------------------------------------------------------------------------


# Creating a dictionary with points related to the hexagons
feature=lay.GetNextFeature()                    # taking first feature (line in attr.
table)
hexIDfield=feature.GetFieldIndex('Unique_ID')  # what's the number of column
Unique_ID (ID hexagons)            ##
bathfield=feature.GetFieldIndex('bathym')      # get order number of bathym field   #
hexbathym={}                                   # new empty dictionary
while feature:                                 # doing this for every feature (until
None after last line)
    values=[]                              # new list
    for i in range(0,ncol):                # for each column (field) in table
        a=feature.GetFieldAsString(i)      # getting value of field (for current line-
feature)
        values.append(a)                   # writing to the list
    geom=feature.GetGeometryRef()          # geometry object ## print POINT
(646500.14572724677 4115052.1469114944)
#    print geom.GetGeometryName()          ## POINT
#    print values       # for every feature list "values" will be new, containing 6
values from table fields
    X=geom.GetX()                          # getting X coord. of the point
    Y=geom.GetY()
    hexID=int(values[hexIDfield])          # ID number of the hexagon correspondent to
this point     ## pointID deleted
    B=float(values[bathfield])
```

```
    point=[X,Y,B]                          # list of all needed info about the current
point to be written in dictionary
    if hexID not in hexbathym.keys():      # then write this key:
        hexbathym[hexID]=([point])         # hexID - the key, lists [X, Y, bathym] -
values (items)
    else:                                  # if this key exists, then attach to it one
more item
        hexbathym[hexID].append(point)     # attaching to the hex. key the list with
current point info
    # => dictionary {key:[list],[list],...}
    feature=lay.GetNextFeature()           # taking next feature (line in attr. table)
print '"hexbathym" dictionary length =', len(hexbathym)
#-----------------------------------------------------------------------------

import numpy

# Creating a dictionary with new bathymetry
newbathym={}
for j in hexbathym.keys():      # for every key (j gets value of the current key)
    xx=[]   # creating temporary lists of all X-s,Y-s and B-s from the current j key
    yy=[]
    bb=[]
    for k in range(0,len(hexbathym[j])):  # len() - number of items (lists with point
info) in the current key
        x=hexbathym[j][k][0]  # reading X of the current k point (item) in the
current hexagon key j
        y=hexbathym[j][k][1]
        b=hexbathym[j][k][2]
        xx.append(x)  # attaching next point from the hexagon key
        yy.append(y)  # for every key these lists will be new
        bb.append(b)
#        print xx     # at the end of 'k' cycle list xx contains all X-s from points
of current key (hexagon)
    tolerance=2     # in the hexagon, outliers are points with bathymetry value    ##
                    # different from meanvalue more than tolerance (will be avoided)
    bbok=[]
    xxok=[]
    yyok=[]
    meanvalue=numpy.mean(bb)  # can be mean or median                              ##
    for t in range(0,len(bb)):
        if numpy.abs(bb[t]-meanvalue)<tolerance:
            bbok.append(bb[t])  # list with values inside tolerance
            xxok.append(xx[t])  # -ok lists will be without outliers
            yyok.append(yy[t])
        else:
            print j, bb[t], 'outlier!'
    xmean=numpy.mean(xxok) # mean X (coord.) in the current key (hexagon)
    ymean=numpy.mean(yyok)
    bmean=numpy.mean(bbok) # mean bathymetry in this hexagon
    newbathym[j]=(xmean,ymean,bmean) # writing to the dictionary
                                     # the current hexagon key, and items: mean X,Y
coord-s and mean bathymetry
print len(newbathym), "points after processing"
#-----------------------------------------------------------------------------


# Saving new bathymetry as a shapefile (for ArcGIS)
def CreateShapefile(bathymdict):               # defining module, bathymdict -
parameter (dictionary {id:x,y,bathym})
#    drv=ogr.GetDriverByName('ESRI Shapefile')  # starting driver for shapefiles
    drv.DeleteDataSource(outshp)           # cleaning existing file
##
```

```
    tema=drv.CreateDataSource(outshp)      # creating new
    layer=tema.CreateLayer("0",None,ogr.wkbPoint,"") # creating layer, name=0,
SpatialReference=None, " " could be some values, type - point
    # creating attributes (fields)
    field1=ogr.FieldDefn("idhex",ogr.OFTInteger)   # defining a field, name=bathym,
type=float
    layer.CreateField(field1)                      # creating a field in the layer
"layer"
    field2=ogr.FieldDefn("ZBott",ogr.OFTReal)      # defining a field, type=float
    layer.CreateField(field2)                      # creating a field in the layer
    field3=ogr.FieldDefn("xwgs84",ogr.OFTReal)
    layer.CreateField(field3)
    field4=ogr.FieldDefn("ywgs84",ogr.OFTReal)
    layer.CreateField(field4)
    # creating attribute table and geometry (filling up the fields)
    for i in bathymdict.keys():                    # cheking every key in the
dictionary
        entdfn=layer.GetLayerDefn()                # getting definition of the layer
        entity=ogr.Feature(entdfn)                 # creating object of class feature
(line in table)
        entity.SetField("idhex",int(i))            # writing value of the current key
in "idhex" field
        entity.SetField('ZBott',bathymdict[i][2])  # writing the bathymetry value in
this field
        entity.SetField('xwgs84',bathymdict[i][0])
        entity.SetField('ywgs84',bathymdict[i][1])
        pnt=ogr.Geometry(ogr.wkbPoint)             # creating a point
        X,Y=bathymdict[i][0],bathymdict[i][1]      # gettig coords x,y from
dictionary
        pnt.AddPoint(X,Y)                          # writing the coords into the point
        #pnt.setX(bathymdict[i][0])                # for changing existing point (its
cooordinate)
        #pnt.setY(bathymdict[i][1])
        entity.SetGeometry(pnt)                    # setting geomtry pnt to the feature
        layer.CreateFeature(entity)                # writing this feature in the layer
(line to shapefile table)
    tema.Release()                                 # shpfile zapisyvaetsya na disk
CreateShapefile(newbathym)
#-----------------------------------------------------------------------------


end=time.time()
print 'took', end-start, 'seconds'
print 'or', (end-start)/60, 'minutes'


# Final message
print ''
print ''
print 'RESULTS'
print nFeat, 'intial points'
print len(newbathym), "points after processing"
```

Appendix 3. Python script "Relation between water color and bathymetry"

```
#-------------------------------------------------------------------------------
# Name:        RELATION BETWEEN WATER COLOR AND BATHYMETRY
# Author:      Nadiia Basos a43682
# Inputs:      New bathymetry points in hexagons (training data), orthophoto
#-------------------------------------------------------------------------------

#1# CALCULATING THE RELATION BETWEEN COLOR AND BATHYMETRY

from __future__ import division   # to avoid problems with dividing integers

inshp='./input_prediction/bathmeasured_training.shp'
satimage='./input_prediction/guadiana_shallow16_utm.jpg'
#satimage='./input_prediction/pca_result.bmp'
format='JPEG'
outtxt='./output_prediction/rgbbathym.txt'

import time
start=time.time()

import osgeo.ogr as ogr

# Reading the shapefile (new bathymetry points in hexagons)
drv=ogr.GetDriverByName('ESRI Shapefile')   # starting driver for shapefiles
allbathym=drv.Open(inshp)                   # open shp-file                    ##
lay=allbathym.GetLayer()                    # get layer
laydef=lay.GetLayerDefn()                   # get its definition
#print "Geometry:",laydef.GetGeomType()      # => Geometry: 1 (points)
ncol=laydef.GetFieldCount()                 # number of fields in attribute table
print ncol, 'fields in the attribute table'
listAttr=[]                                 # new empty list
for i in range(0,ncol):                     # do next two lines "ncol" times
    fld=laydef.GetFieldDefn(i)              # object of the class FieldDefn
    listAttr.append(fld.GetNameRef())       # get field name and put in into the list
print 'field names', listAttr
nFeat=lay.GetFeatureCount()                 # how many features (points) in shapefile
print nFeat, 'features (points) in the shapefile'
prj=lay.GetSpatialRef()
#-------------------------------------------------------------------------------


# Creating a dictionary
feature=lay.GetNextFeature()                    # taking first feature (line in attr.
table)
IDfield=feature.GetFieldIndex('OBJECTID')       # what's the number of columnn
OBJECTID (default field)
bathfield=feature.GetFieldIndex('bathym')       # poryadkovyi nomer stolbtsa bathym
##
newbathym={}                                    # new empty dictionary
while feature:                                  # doing this for every feature (until
None after last line)
    values=[]                           # new list
    for i in range(0,ncol):             # for each column (field) in the table
        a=feature.GetFieldAsString(i)   # getting value of field (for current line-
feature)
        values.append(a)                # writing to the list
    geom=feature.GetGeometryRef()       # geometry object ## print POINT
(646500.14572724677 4115052.1469114944)
#    print geom.GetGeometryName()        ## POINT
```

```
#    print values                        # for every feature list "values" will be new,
containing 6 values from table fields
    X=geom.GetX()                        # getting X coord. of the point
    Y=geom.GetY()
    ID=int(values[IDfield])              # ID number of the hexagon correspondent to
this point
    B=float(values[bathfield])
#    B=(B+10)**(float(1)/float(3))       # transformation                              ##
    newbathym[ID]=([X,Y,B])              # ID - the key, lists [X, Y, bathym] - values
(items)
    feature=lay.GetNextFeature()         # taking next feature (line in attr. table)
print len(newbathym), "points in the dictionary"
#------------------------------------------------------------------------------

import numpy

import osgeo.gdal as gdal
import sys

# Reading the satellite image
drvimage=gdal.GetDriverByName(format)          # image driver
drvimage.Register()                            # http://gdal.org/formats_list.html
dataset=gdal.Open(satimage,gdal.GA_ReadOnly)   # opening the file
nBands=dataset.RasterCount                     # number of raster bands, 3
nCols=dataset.RasterXSize                      # number of columns
nRows=dataset.RasterYSize                      # number of rows
print 'columns',nCols,'rows',nRows,'bands',nBands    ## 996 1571 3
georef=dataset.GetGeoTransform()       # the affine transformation coeffs (xtl, pixel,
angle, ytl, angle, -pixel)
refsyst=dataset.GetProjection()        # reading prj info
print georef                           # coords of top left corner of top left pixel
(w-file - center of the pixel!)
## (640566.9717115981, 5.0, 0.0, 4118862.5995785883, 0.0, -5.0) - list
originX,originY=georef[0],georef[3]            # x,y coords of top left corner of top
left pixel
pixelWidth,pixelHeight=georef[1],georef[5]     # pixel size
print 'pixel',pixelWidth,'x',pixelHeight,'m'
bands=[]
for i in range(nBands):                # for every of all 3 bands-objects
    bnd=dataset.GetRasterBand(i+1)     # bands 1,2,3, i cannot start from 0
    bands.append(bnd)                  # attaching current band (as ogr object) to the
list
#------------------------------------------------------------------------------


# Reading RGB values corresponding to (new) bathymetry points
rgb=[]
bath=[]
for i in newbathym.keys():     # for every key (j gets value of the current key)
    X=newbathym[i][0]          # reading coords of (new mean) points from dictionary
    Y=newbathym[i][1]
    B=newbathym[i][2]
    xOffset=int((X-originX)/pixelWidth)    # column in raster with current point from
dictionary
    yOffset=int((Y-originY)/pixelHeight)   # row
    cell=[]
    for j in bands:                                  # for every band-gdal object
#        near=j.ReadAsArray(xOffset-1,yOffset-1,3,3)  # reading value of block of
cells with current point in center
                                          # this block is 3x3 cells,
xOffset,yOffset - top left corner of the block
```

```
#
cells=[near[0][0],near[0][1],near[0][2],near[1][0],near[1][1],near[1][2],near[2][0],n
ear[2][1],near[2][2]]
#        median=numpy.median(cells)         # median of the block
#        cell.append(int(median))           # adding to list cell R, then G, then
Blue; for current point (key)
        vCell=j.ReadAsArray(xOffset,yOffset,1,1)   # reading value of block of cells
with current point
                                           # this block is 1x1, one cell,
xOffset,yOffset - top left corner of the block
        cell.append(int(vCell))            # adding to list cell R, then G, then
Blue; for current point (key)
    cell.append(1)             # cell becomes 4 elements list [r, g, b, 1]
    rgb.append(cell)           # list of (cell)lists, at the end - for all points
    bath.append(B)
rgbm=numpy.matrix(rgb)         # converting list of lists to a matrix
bathmh=numpy.matrix(bath)      # converting list to 1D horizontal matrix (vector)
bathmv=bathmh.T                # to vertical vector
#bathmv=numpy.transpose(bathmh)
#-------------------------------------------------------------------------------

import csv

# Saving ASCII file with R, G, B and correspondent bathymetry values (for R-program)
export=open(outtxt, "wb")      # where to save
writer=csv.writer(export, delimiter='\t', quoting=csv.QUOTE_NONE) #,
lineterminator='\n')
header=['R','G','B','bathym']  # list of strings [R G B bath]
writer.writerow(header)        # writes a sequence as a line to a file (here the
header)
for i in range(0, len(rgb)):   # rgb is a list of lists r,g,b,1
    line=[]                    # making the sequence to write (for each rgb cell-
bath.point)
    line.append(rgb[i][0])     # R
    line.append(rgb[i][1])     # G
    line.append(rgb[i][2])     # B
    line.append(bath[i])       # bathymetry point
    writer.writerow(line)      # writing it
export.close()
#-------------------------------------------------------------------------------


# Multiple regression (bathymetry ~ R + G + B)
coeffs=(rgbm.T*rgbm).I*rgbm.T*bathmv   # matrix equation to solve MR and get
coefficients
print coeffs                           # matrix (vector) of the coefficients
##
coefR=float(coeffs[0])
coefG=float(coeffs[1])
coefB=float(coeffs[2])
intercept=float(coeffs[3])
print 'bathymetry =',intercept,'+',coefR,'* R +',coefG,'* G +',coefB,'* B'
# R-squared (coefficient of determination)
residss=0
totalss=0
for j in range(0, len(rgb)):
    f=intercept+rgb[j][0]*coefR+rgb[j][1]*coefG+rgb[j][2]*coefB   # fitted
(predicted) value
    residss=residss+(bath[j]-f)**2                                # making sum of
squares of residuals
    totalss=totalss+(bath[j]-numpy.mean(bath))**2                 # making total sum
of squares
```

```
Rsquared=1-(residss/totalss)
RMSEtrain=residss/len(rgb)
print 'R-squared =',Rsquared
print 'RMSE training =', RMSEtrain
#-------------------------------------------------------------------------------


end=time.time()
print 'took', end-start, 'seconds'

print intercept
print coefR
print coefG
print coefB

# Final message
print ''
print inshp
print 'RESULTS'
print nFeat, 'bathymetry points'
print 'bathymetry =',intercept,'+',coefR,'* R +',coefG,'* G +',coefB,'* B'
print 'R-squared =',Rsquared

print ''
print 'intercept=', intercept
print 'coefR=', coefR
print 'coefG=', coefG
print 'coefB=', coefB
```

Appendix 4. Python script "Predicting bathymetry"

```
#-------------------------------------------------------------------------------
# Name:        PREDICTING BATHYMETRY
# Author:      Nadiia Basos a43682
# Inputs:      Grid corners and centers, orthophoto, predicting areas
#-------------------------------------------------------------------------------

#2# PREDICTING BATHYMETRY

incorners='./input_prediction/grids_2209-122_corners.shp'
incenters='./input_prediction/All_GuadT2_init_land_mertih_2209-122triang.shp'
satimage="./input_prediction/guadiana_shallow16_utm.jpg"
predictarea='./input_prediction/PredictArea.shp'
format='JPEG'
outbathym="./output_prediction/bathpredictedrgb_t.shp"

# MR results                                                                  ##
# transf
intercept= 3.24767622
coefR= -0.02775601
coefG= -0.01024637
coefB= 0.04020086

import time
start=time.time()

import osgeo.ogr as ogr

#1 Reading grid corners shapefile
drv=ogr.GetDriverByName('ESRI Shapefile')     # driver for shapefiles
temacorn=drv.Open(incorners)                  # hexagons opening
lay1=temacorn.GetLayer()                      # layer of hexagons
laydef1=lay1.GetLayerDefn()
ncol1=laydef1.GetFieldCount()
print ncol1,'fields in the corners attribute table'
listAttr1=[]                                  # new empty list for field names
listType1=[]
for i in range(0,ncol1):
    fld=laydef1.GetFieldDefn(i)               # object of the class FieldDefn (all
about field i)
    listAttr1.append(fld.GetNameRef())
    listType1.append(fld.GetFieldTypeName(fld.GetType()))  # checking field types
print listAttr1
print listType1
nfeat1=lay1.GetFeatureCount()
print nfeat1, 'grid corners'

#2 Reading grid centers shapefile
drv=ogr.GetDriverByName('ESRI Shapefile')     # driver for shapefiles
temacent=drv.Open(incenters)                  # hexagons opening
lay2=temacent.GetLayer()                      # layer of hexagons
laydef2=lay2.GetLayerDefn()
ncol2=laydef2.GetFieldCount()
print ncol2,'fields in the centers attribute table'
listAttr2=[]                                  # new empty list for field names
listType2=[]
for i in range(0,ncol2):
    fld=laydef2.GetFieldDefn(i)               # object of the class FieldDefn (all
about field i)
    listAttr2.append(fld.GetNameRef())
```

```
    listType2.append(fld.GetFieldTypeName(fld.GetType()))  # checking field types
print listAttr2
print listType2
nfeat2=lay2.GetFeatureCount()
print nfeat2, 'grid centers'
#-------------------------------------------------------------------------------

# Reading where to predict bathymetry
#drv=ogr.GetDriverByName('ESRI Shapefile')           #
temaAreas=drv.Open(predictarea)          # polygons for areas with no measurements
layAreas=temaAreas.GetLayer()                        #
laydefAreas=layAreas.GetLayerDefn()                  #
ncolAreas=laydefAreas.GetFieldCount()                #
nAreas=layAreas.GetFeatureCount()
print nAreas, 'areas for prediction'
#-------------------------------------------------------------------------------


# Getting points
feature1=lay1.GetNextFeature()               # getting feature from layer lay
gridpoints={}                                # dictionary for hexagon centers
while feature1:                              # for each feature1 (until None after
last line)
    values=[]                                # new list (for each line in attr table)
    for i in range(0,ncol1):                 # for each column in attr table
        a=feature1.GetFieldAsString(i)       # value from table
        values.append(a)                     # attributes, the line from table
    geom1=feature1.GetGeometryRef()          # geometry of current hexagon
    X1=geom1.GetX()                          # getting X coord. of the point
    Y1=geom1.GetY()
#    print geom1                              ## POINT (645053.60775042314
4118307.0084162918)
    for j in range(0,nAreas):
        featureArea=layAreas.GetFeature(j)
        geomArea=featureArea.GetGeometryRef()  # geometry of each predicting area
polygon
        if geom1.Intersect(geomArea):        # if current point intersects the
prediction area polygon
            gridpoints[len(gridpoints)+1]=(X1,Y1)  # hexagon ID - key; X, Y of center
- values
    feature1=lay1.GetNextFeature()
print len(gridpoints), 'corners in prediction area'
# adding centers
feature2=lay2.GetNextFeature()               # getting feature from layer lay
while feature2:                              # for each feature2 (until None after
last line)
    values=[]                                # new list (for each line in attr table)
    for i in range(0,ncol2):                 # for each column in attr table
        a=feature2.GetFieldAsString(i)       # value from table
        values.append(a)                     # attributes, the line from table
    geom2=feature2.GetGeometryRef()          # geometry of current hexagon
    X2=geom2.GetX()                          # getting X coord. of the point
    Y2=geom2.GetY()
#    print geom2                              ## POINT (645053.60775042314
4118307.0084162918)
    for j in range(0,nAreas):
        featureArea=layAreas.GetFeature(j)
        geomArea=featureArea.GetGeometryRef()  # geometry of each predicting area
polygon
        if geom2.Intersect(geomArea):        # if current point intersects the
prediction area polygon
```

```
            gridpoints[len(gridpoints)+1]=(X2,Y2) # hexagon ID - key; X, Y of center
- values
    feature2=lay2.GetNextFeature()
print len(gridpoints), 'corners and centers in prediction area'
#-------------------------------------------------------------------------------

import numpy
import osgeo.gdal as gdal
import sys


# Reading the satellite image
drvimage = gdal.GetDriverByName(format)       # image driver
drvimage.Register()                           # http://gdal.org/formats_list.html
dataset=gdal.Open(satimage,gdal.GA_ReadOnly)  # opening the file
##
nBands=dataset.RasterCount                    # number of raster bands, 3
nCols=dataset.RasterXSize                     # number of columns
nRows=dataset.RasterYSize                     # number of rows
print 'image columns',nCols,'rows',nRows,'bands',nBands    ## 1769 1410 3
georef=dataset.GetGeoTransform()     # the affine transformation coeffs (xtl, pixel,
angle, ytl, angle, -pixel)
refsyst=dataset.GetProjection()      # reading prj info
print georef                         # coords of top left corner of top left pixel
(w-file - center of the pixel!)
## (640566.9717115981, 5.0, 0.0, 4118862.5995785883, 0.0, -5.0) - list
originX,originY=georef[0],georef[3]           # x,y coords of top left corner of top
left pixel
pixelWidth,pixelHeight=georef[1],georef[5]    # pixel size
print 'pixel',pixelWidth,'x',pixelHeight,'m'
bands=[]
for i in range(nBands):               # for every of all 3 bands-objects
    bnd=dataset.GetRasterBand(i+1)   # bands 1,2,3, i cannot start from 0
    bands.append(bnd)                # attaching current band (as ogr object) to the
list
#-------------------------------------------------------------------------------


# Reading RGB values corresponding to points and predicting bathymetry
bathpredict={}
for i in gridpoints.keys():    # for every key (j gets value of the current key)
    X=gridpoints[i][0]         # reading coords of (new mean) points from dictionary
    Y=gridpoints[i][1]
    xOffset=int((X-originX)/pixelWidth)    # column in raster with current point from
dictionary
    yOffset=int((Y-originY)/pixelHeight)   # row
    cell=[]
    for j in bands:                                # for every band-gdal object
        vCell=j.ReadAsArray(xOffset,yOffset,1,1)   # reading value of block of cells
with current point
                                               # this block is 1x1, one cell,
xOffset,yOffset - top left corner of the block
        cell.append(int(vCell))                # adding to list cell R, then G, then
Blue; for current point (key)
    Bpredict=intercept+coefR*cell[0]+coefG*cell[1]+coefB*cell[2]
    Bpredict=(2.718282**Bpredict)/10.0-2.5         # back-transform              ##
    cell.append(Bpredict)       # cell becomes 4 elements list [r, g, b, bathymetry]
    cell.append(X)
    cell.append(Y)
    bathpredict[i]=(cell)       # [r, g, b, bathymetry, x, y]
print i,bathpredict[i]
#-------------------------------------------------------------------------------
```

```
# Saving bathymetry predicted in hexagon centroids as a shapefile (for ArcGIS)
def CreateShapefile2(centrdict):                # defining module, centrdict -
parameter (dictionary {id:r,g,b,bath,x,y})
#    drv=ogr.GetDriverByName('ESRI Shapefile')  # starting driver for shapefiles
    drv.DeleteDataSource(outbathym)           # cleaning existing file             ##
    tema=drv.CreateDataSource(outbathym)      # creating new
    layer=tema.CreateLayer("0",None,ogr.wkbPoint,"") # creating layer, name=0,
SpatialReference=None,
                                              # " " could be some values, type
- point
    # creating attributes (fields)
    field1=ogr.FieldDefn("id",ogr.OFTInteger)       # defining a field, name=id,
type=integer
    layer.CreateField(field1)                       # creating a field in the layer
"layer"
    field2=ogr.FieldDefn("xwgs84",ogr.OFTReal)      # defining a field, type=float
    layer.CreateField(field2)                       # creating a field in the layer
    field3=ogr.FieldDefn("ywgs84",ogr.OFTReal)
    layer.CreateField(field3)
    field4=ogr.FieldDefn("bathym",ogr.OFTReal)
    layer.CreateField(field4)
    # creating attribute table and geometry (filling up the fields)
    for i in centrdict.keys():                    # cheking every key in the dictionary
        entdfn=layer.GetLayerDefn()               # getting definition of the layer
        entity=ogr.Feature(entdfn)                # creating object of class feature
(line in table)
        entity.SetField("id",int(i))              # writing value of the current key in
"id" field
        X,Y=centrdict[i][4],centrdict[i][5]    # gettig coords x,y from the
dictionary
        entity.SetField('xwgs84',X)    # writing X value into this field
        entity.SetField('ywgs84',Y)
        entity.SetField('bathym',centrdict[i][3])
        pnt1=ogr.Geometry(ogr.wkbPoint)           # creating a point
        pnt1.AddPoint_2D(X,Y)                     # writing the coords into the point
#        print pnt1                                ## POINT (645053.60775042314
4118307.0084162918)
        entity.SetGeometry(pnt1)                  # set geomtry pnt1 to the feature
        layer.CreateFeature(entity)               # writing this featurein the layer
(line to shapefile table)
    tema.Release()                                # shapefile is written on the disk C
CreateShapefile2(bathpredict)
#-------------------------------------------------------------------------------


end=time.time()
print 'took', end-start, 'seconds'

# Final message
print ''
print ''
print 'RESULTS'
print 'bathymetry =',intercept,'+',coefR,'* R +',coefG,'* G +',coefB,'* B'
print len(gridpoints), 'predicted points'
```

Appendix 5. Python script "RMSE of bathymetry prediction"

```
#-------------------------------------------------------------------------------
# Name:        RMSE OF BATHYMETRY PREDICTION
# Author:      Nadiia Basos a43682
# Inputs:      New bathymetry points in hexagons (test subset), orthophoto
#-------------------------------------------------------------------------------

#3# RMSE

from __future__ import division   # to avoid problems with dividing integers

inshp='./input_prediction/bathmeasured_test.shp'
satimage='./input_prediction/guadiana_shallow16_utm.jpg'
#satimage='./input_prediction/pca_result.bmp'
format='JPEG'
outtxt='./output_prediction/rmsetest.txt'

# MR results                                                              ##
# transf
intercept= 3.24767622
coefR= -0.02775601
coefG= -0.01024637
coefB= 0.04020086


import time
start=time.time()

import osgeo.ogr as ogr

# Reading the shapefile (new bathymetry points in hexagons)
drv=ogr.GetDriverByName('ESRI Shapefile')   # starting driver for shapefiles
allbathym=drv.Open(inshp)                   # open shp-file
lay=allbathym.GetLayer()                    # get layer
laydef=lay.GetLayerDefn()                   # get its definition
#print "Geometry:",laydef.GetGeomType() # => Geometry: 1 (points)
ncol=laydef.GetFieldCount()                 # number of fields in attribute table
print ncol, 'fields in the attribute table'
listAttr=[]                                 # new empty list
for i in range(0,ncol):                     # do next two lines "ncol" times
    fld=laydef.GetFieldDefn(i)              # object of the class FieldDefn
    listAttr.append(fld.GetNameRef())       # get field name and put in into the list
print 'field names', listAttr
nFeat=lay.GetFeatureCount()                 # how many features (points) in shapefile
print nFeat, 'features (points) in the shapefile'
prj=lay.GetSpatialRef()
#-------------------------------------------------------------------------------


# Creating a dictionary
feature=lay.GetNextFeature()                # taking first feature (line in attr.
table)
IDfield=feature.GetFieldIndex('OBJECTID')   # what's the number of columnn OBJECTID
(default field)
bathfield=feature.GetFieldIndex('bathym')   # bathym field number               ##
newbathym={}                                # new empty dictionary
while feature:                              # doing this for every feature (until
None after last line)
    values=[]                           # new list
    for i in range(0,ncol):             # for each column (field) in table
```

```
        a=feature.GetFieldAsString(i)   # getting value of field (for current line-
feature)
        values.append(a)                # writing to the list
    geom=feature.GetGeometryRef()       # geometry object ## print POINT
(646500.14572724677 4115052.1469114944)
#    print geom.GetGeometryName()        ## POINT
#    print values                    # for every feature list "values" will be new,
containing 6 values from table fields
    X=geom.GetX()                       # getting X coord. of the point
    Y=geom.GetY()
    ID=int(values[IDfield])       # ID number of the hexagon correspondent to this
point
    B=float(values[bathfield])
#    B=(B+10)**(float(1)/float(3))      # transformation
##
    newbathym[ID]=([X,Y,B])             # ID - the key, lists [X, Y, bathym] - values
(items)
    feature=lay.GetNextFeature()        # taking next feature (line in attr. table)
print len(newbathym), "points in the dictionary"
#-----------------------------------------------------------------------------

import numpy

import osgeo.gdal as gdal
import sys
#from osgeo.gdalconst import *  # constant values defined in gdal

# Reading the satellite image
drvimage=gdal.GetDriverByName(format)       # image driver
drvimage.Register()                           # http://gdal.org/formats_list.html
dataset=gdal.Open(satimage,gdal.GA_ReadOnly)  # opening the file
##
nBands=dataset.RasterCount                    # number of raster bands, 3
nCols=dataset.RasterXSize                     # number of columns
nRows=dataset.RasterYSize                     # number of rows
print 'columns',nCols,'rows',nRows,'bands',nBands    ## 996 1571 3
georef=dataset.GetGeoTransform()     # the affine transformation coeffs (xtl, pixel,
angle, ytl, angle, -pixel)
refsyst=dataset.GetProjection()      # reading prj info
print georef                         # coords of top left corner of top left pixel
(w-file - center of the pixel!)
## (640566.9717115981, 5.0, 0.0, 4118862.5995785883, 0.0, -5.0) - list
originX,originY=georef[0],georef[3]           # x,y coords of top left corner of top
left pixel
pixelWidth,pixelHeight=georef[1],georef[5]    # pixel size
print 'pixel',pixelWidth,'x',pixelHeight,'m'
bands=[]
for i in range(nBands):              # for every of all 3 bands-objects
    bnd=dataset.GetRasterBand(i+1)   # bands 1,2,3, i cannot start from 0
    bands.append(bnd)                # attaching current band (as ogr object) to the
list
#-----------------------------------------------------------------------------


# Reading RGB values corresponding to test bathymetry points
rgb=[]
bath=[]
for i in newbathym.keys():    # for every key (j gets value of the current key)
    X=newbathym[i][0]         # reading coords of (new mean) points from dictionary
    Y=newbathym[i][1]
    B=newbathym[i][2]
```

```
    xOffset=int((X-originX)/pixelWidth)    # column in raster with current point from
dictionary
    yOffset=int((Y-originY)/pixelHeight)   # row
    cell=[]
    for j in bands:                                  # for every band-gdal object
#        near=j.ReadAsArray(xOffset-1,yOffset-1,3,3)  # reading value of block of
cells with current point in center
                                                     # this block is 3x3 cells,
xOffset,yOffset - top left corner of the block
#
cells=[near[0][0],near[0][1],near[0][2],near[1][0],near[1][1],near[1][2],near[2][0],n
ear[2][1],near[2][2]]
#        median=numpy.median(cells)          # median of the block
#        cell.append(int(median))            # adding to list cell R, then G, then
Blue; for current point (key)
        vCell=j.ReadAsArray(xOffset,yOffset,1,1)   # reading value of block of cells
with current point
                                            # this block is 1x1, one cell,
xOffset,yOffset - top left corner of the block
        cell.append(int(vCell))             # adding to list cell R, then G, then
Blue; for current point (key)
    cell.append(1)              # cell becomes 4 elements list [r, g, b, 1]
    rgb.append(cell)            # list of (cell)lists, at the end - for all points
    bath.append(B)
#rgbm=numpy.matrix(rgb)          # converting list of lists to a matrix
#bathmh=numpy.matrix(bath)       # converting list to 1D horizontal matrix (vector)
#bathmv=bathmh.T                 # to vertical vector
#bathmv=numpy.transpose(bathmh)
#-------------------------------------------------------------------------------

import csv

# Saving ASCII file with R, G, B and correspondent bathymetry values (for R-program)
export=open(outtxt, "wb")   # where to save                                      ##
writer=csv.writer(export, delimiter='\t', quoting=csv.QUOTE_NONE)#,
lineterminator='\n')
header=['R','G','B','bathym']   # list of strings [R G B bath]
writer.writerow(header)         # writes a sequence as a line to a file (here the
header)
for i in range(0, len(rgb)):  # rgb is a list of lists r,g,b,1
    line=[]                   # making the sequence to write (for each rgb cell-
bath.point)
    line.append(rgb[i][0])    # R
    line.append(rgb[i][1])    # G
    line.append(rgb[i][2])    # B
    line.append(bath[i])      # bathymetry point
    writer.writerow(line)     # writing it
export.close()
#-------------------------------------------------------------------------------


# Testing regression
print 'bathymetry =',intercept,'+',coefR,'* R +',coefG,'* G +',coefB,'* B'
# R-squared (coefficient of determination) and RMSE (root-mean-square error)
residss=0
totalss=0
for j in range(0, len(rgb)):
    f=intercept+rgb[j][0]*coefR+rgb[j][1]*coefG+rgb[j][2]*coefB   # fitted
(predicted) value
    f=(2.718282**f)/10.0-2.5                                      # back-
transformation (if was applied)           ##
```

```
    residss=residss+(bath[j]-f)**2                                  # making sum of
squares of residuals
    totalss=totalss+(bath[j]-numpy.mean(bath))**2                   # making total sum
of squares
Rsquared=1-(residss/totalss)
RMSE=residss/len(rgb)
print 'R-squared =',Rsquared
print 'RMSE test =', RMSE, 'm'


end=time.time()
print 'took', end-start, 'seconds'

# Final message
print ''
print inshp
print 'RESULTS'
print nFeat, 'evaluation points'
#print 'R-squared test=',Rsquared
print 'RMSE test =', RMSE, 'm'
```