UNIVERSIDADE FEDERAL DO PARANÁ

HENRIQUE GUARNERI

METHODOLOGY FOR THE ANALYSIS AND DESIGN OF INLAND WATERWAYS AND ITS APPLICATION FOR THE PARAGUAY RIVER WATERWAY

CURITIBA

2017

HENRIQUE GUARNERI

METHODOLOGY FOR THE ANALYSIS AND DESIGN OF INLAND WATERWAYS AND ITS APPLICATION FOR THE PARAGUAY RIVER WATERWAY

Dissertação do Curso de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental, Área de Concentração em Recursos Hídricos, Departamento de Hidráulica e Saneamento, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Ciências.
Orientador: Prof. Dr.-Ing. Tobias Bleninger

CURITIBA

2017

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS/UFPR
BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

G916m Guarneri, Henrique
Methodology for the analysis and design of inland waterways and its application for the Paraguay river waterway / Henrique Guarneri. – Curitiba, 2017.
183 p. : il. color.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental, 2017.

Orientador: Tobias Bleninger.

1. Rio Paraguai. 2. Modelagem de nível de água. 3. Modelos hidrodinâmicos. 4. Otimização de níveis de referência. 5. Precisão do volume de dragagem. I. Universidade Federal do Paraná. II. Bleninger, Tobias. III. Título.

CDD: 627.1

Bibliotecária: Romilda Santos - CRB-9/1214

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA DE RECURSOS HÍDRICOS E AMBIENTAL da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de HENRIQUE GUARNERI, intitulada: "METHODOLOGY FOR ANALYSIS AND DESIGN OF INLAND WATERWAYS AND ITS APPLICATION FOR THE PARAGUAY RIVER WATERWAY", após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua aprovação.

Curitiba, 05 de Maio de 2017.

TOBIAS BERNWARD BLENINGER
Presidente da Banca Examinadora (UFPR)

Gustavo O. S. Fernandes.

CRISTOVÃO VICENTE SCAPULATEMPO FERNANDES
Avaliador Interno (UFPR)

ELOY KAVISKI
Avaliador Interno (UFPR)

CALVIN T. CREECH

"This is one of man's oldest riddles.
How can the independence of human volition be harmonized with the fact that we are integral parts of a universe which is subject to the rigid order of nature's laws?"
(Max Planck)

RESUMO

No contexto das soluções técnicas para vias navegáveis interiores e dos esforços recentes do governo Brasileiro para equilibrar sua matriz modal de transporte, o objetivo geral desta dissertação é de fornecer conjuntos de ferramentas e interpretações para ajudar na melhoria e otimização de projetos e manutenções de hidrovias, focando principalmente no contexto Brasileiro. Inicialmente, a interpretação das diferenças entre um Nível de Redução para Batimetria (BRL) e um Nível de Redução para Dragagem (DRL) foram estabelecidos. Elas são fundamentais para um tratamento mais adequado das questões relativas à disponibilidade de profundidade e o desempenho de navegação. Na sequência, demonstrou-se como as influências plurianuais dos níveis de água dos rios podem ser de importância central para as definições de DRLs. Também não há evidência razoável para tratar o DRL como uma característica intrínseca de rios. E que os modelos de previsão podem ser ferramentas poderosas na busca de uma garantia de tempo operacional exata (por exemplo, 90%). Além disso, a abordagem de previsões visou avaliar até que ponto as previsões endógenas simples poderiam atingir em questão de precisão e acuracia. Apresentando uma melhoria significativa na definição de DRL, um ganho considerável de precisão, bem como a fixação de um ‘benchmark’ para abordagens futuras mais complexas (por exemplo, multi-estação, híbrido, exógeno). Além disso, também foi demonstrado como imprecisões de definição de DRL podem afetar os volumes de dragagem de um projeto. Da mesma forma, um modelo de nível de água melhor poderia "pagar por si mesmo" ao longo dos anos com os ganhos resultados de dragagens mais acuradas e precisas.

Palavras-chaves: Rio Paraguai, Brasil, vias navegáveis interiores, Navegação, otimização, disponibilidade de profundidade, previsão de níveis, modelagem de nível de água, modelos hidrodinâmicos, otimização de níveis de referência, precisão do volume de dragagem.

ABSTRACT

In the context of technical solutions for inland waterways, within the recent efforts of the Brazilian government to balance its modal matrix of transport, the overall aim of this dissertation is to provide a set of methodological tools and interpretations to assist the improvement and optimization of waterway's designs and maintenances, focusing primarily on the Brazilian context. Initially, the interpretation of the differences between a Bathymetric Reference Level (BRL) and a Dredging Reference Level (DRL) had to be set. They are key to a more appropriate treatment of the issues concerning waterways depth availability and performance. In the sequence, it was demonstrated how rivers' water level multi-annual influences can be of central importance to DRL definitions. Also that there is no reasonable evidence for treating the DRL as an intrinsic river's characteristic. And that forecast models can be powerful tools in the pursuit of an exact operational time assurance (*e.g.* 90%). Moreover, the forecast approach aimed at assessing how far could simple endogenous predictions reach precision wise. Presenting a significant improvement on DRL definition, a considerable gain of precision as well as setting a 'benchmark' for more complex approaches (*e.g.* multi-station, hybrid, exogenous). Furthermore, it was also demonstrated how DRL imprecisions can affect the dredging volumes of a project. Likewise, a better water level model is likely to 'pay for it self' along the years with the resulted gains due to more precise and accurate dredgings.

Key-words: Paraguay River, Brazil, Inland Waterway, Navigation, optimization, depth availability, stage forecast, water level modeling, hydrodynamic models, reference level improvement, dredging volume accuracy.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ANN	Artificial Neural Network
ANTAQ	Brazilian National Agency of Waterway Transportation
AR	Autoregressive
BRL	Bathymetric Reference Level
CNT	Brazilian National Confederation of Transport
DEM	Digital Elevation Model
DNIT	Brazilian National Department of Transport Infrastructure
DRL	Dredging Reference Level
EPL	Brazilian Federal Government's Logistic and Planning Company S.A.
ERM	Elevation Reference Marks
EVTEA	Technical-Economical and Environmental Feasibility Studies
IHO	International Hydrographic Association
ITTI	Technological Institute of Transport and Infrastructure
PIANC	The World Association for Waterborne Transport Infrastructure
PNLI	Brazilian National Plan of Integrated Logistics
PNLT	Brazilian National Transport Logistics Plan
UFPR	Brazilian Federal University of Paraná
UPRB	Upper Paraguay River Basin

CONTENTS

1 INTRODUCTION

The appeal of inland waterways relates to the transport matrix modal distribution and to the logistic dynamic of supply and demand. As a consequence, the existence of inland navigation requires navigable rivers and interest of use. Today, Brazil has a significant amount of navigable rivers. According to the National Confederation of Transport CNT (2013), it has one of the largest hydrographic networks in the world, with approximately 63 thousand kilometers of extension, connecting Brazil with 5 other South American countries. Henceforth, part of this network integrates a region of great agricultural production and economic expansion, the Mid-West. During the year 2012, 80.9 million tons were transported through waterways, although only 50.3% (20.956 km), of a total of 41.635 navigable kilometers, are economically navigated. This under-utilization also reflects the Brazilian modal distribution. Only 4% of the total inter-regional cargo were transported through inland waterways in 2015 (EPL, 2015). Figure 1 illustrates this distribution.

In this context, the Brazilian government in 2007, through the Transport and Defense Ministries, with the National Transport Logistics Plan (PNLT) (MT; MD, 2007) and subsequent updates in 2009 and 2011, agreed to 'make and effective change, with better balance, in the country's current freight transport matrix, as optimization and rationalization measures are associated with more appropriate and intensive use of railway and waterway modal, taking advantage of their energetic efficiencies and productivity in the displacement of higher density transport flows and distances.' With that, the PNLT foresaw in 2007, the increase in a 15 to 20 years horizon, of the waterway transportation share from 13% to 29% [1].

[1] Taking in consideration cabotage and inland navigation

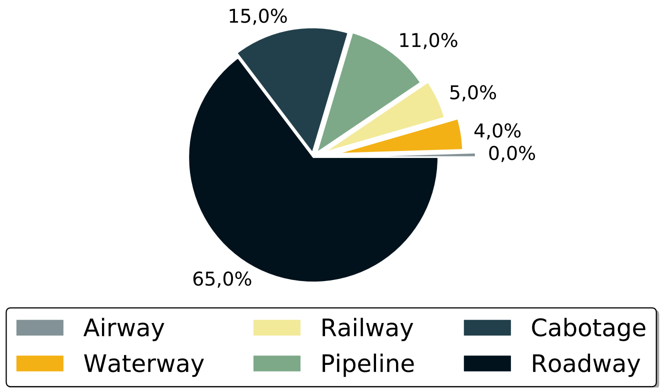


FIGURE 1 – Brazilian modal distribution of inter-regional transportation in 2015. Source: (EPL, 2016)

Following the PNLT, on February 19th, 2013, the National Agency of Waterway Transportation (ANTAQ) published the National Waterway Integration Plan (PNIH). This plan detailed the Brazilian waterways and the indication of opportune areas for port installations. In overview, it intended to identify alternatives for cargo transportation by waterway means when contrasted to the current and projected transport matrix. As a result, the plan delivered for the 6 Brazilian hydrographic basins the potential utilization of the waterway modal, terminals and paths, to the transportation of goods for the scenarios of 2015, 2020, 2025, 2030.

Currently, the Federal Government's Logistic and Planning Company S.A. (EPL) is developing the National Plan of Integrated Logistics (PNLI). The PNLI has as objective identify and analyze the alternatives for the optimization of transportation of cargo through railroads, cabotage and inland waterways, as high capacity systems, integrated to the regional roadways in a harmonic and synergistic way (EPL, 2016).

These efforts in the last decade were focused mostly to assess one of the requirements of navigation, the demand. The second requirement, the availability of navigable rivers, was approached by the National Department of Transport Infrastructure (DNIT) through the development of Technical, Economical and Environmental Feasibility Studies (EVTEA). These studies were performed for every operational waterway and those with potential of operation. Initially, 9 inland waterways were investigated in respect to its feasibilities (Figure 2).

These studies generally relied on established techniques and knowledge to reach its results. Nonetheless, due to the primary data gathered, questions raised, rationalizations and the overall understanding of the issues surrounding the waterway context, this studies may serve as base to further investigations on the natural processes and characteristics that constraint the feasibility of inland waterways. Therefore, advancing the reliability and generality of the findings.

In this context, specifically considering the EVTEA of the Paraguay River Waterway, developed in a partnership between the Brazilian National Department of Transport Infrastructure (DNIT) and the Brazilian Federal University of Paraná (UFPR), a wide set of question were raised and denote lack of formal literature aligned with the Brazilian background and the premises of optimization and rationalization of the PNLT. That is, given the complex natural scenario of inland navigation, very few has been done to the systematic arrangement of knowledge concerning the physical and material aspects of waterway design and management. Moreover, the Brazilian scenario, land proportions, river extensions, remoteness and low density of data gathering stations, set a background somewhat different from those of other countries where this systematic arrangement of knowledge has been specifically consolidated.

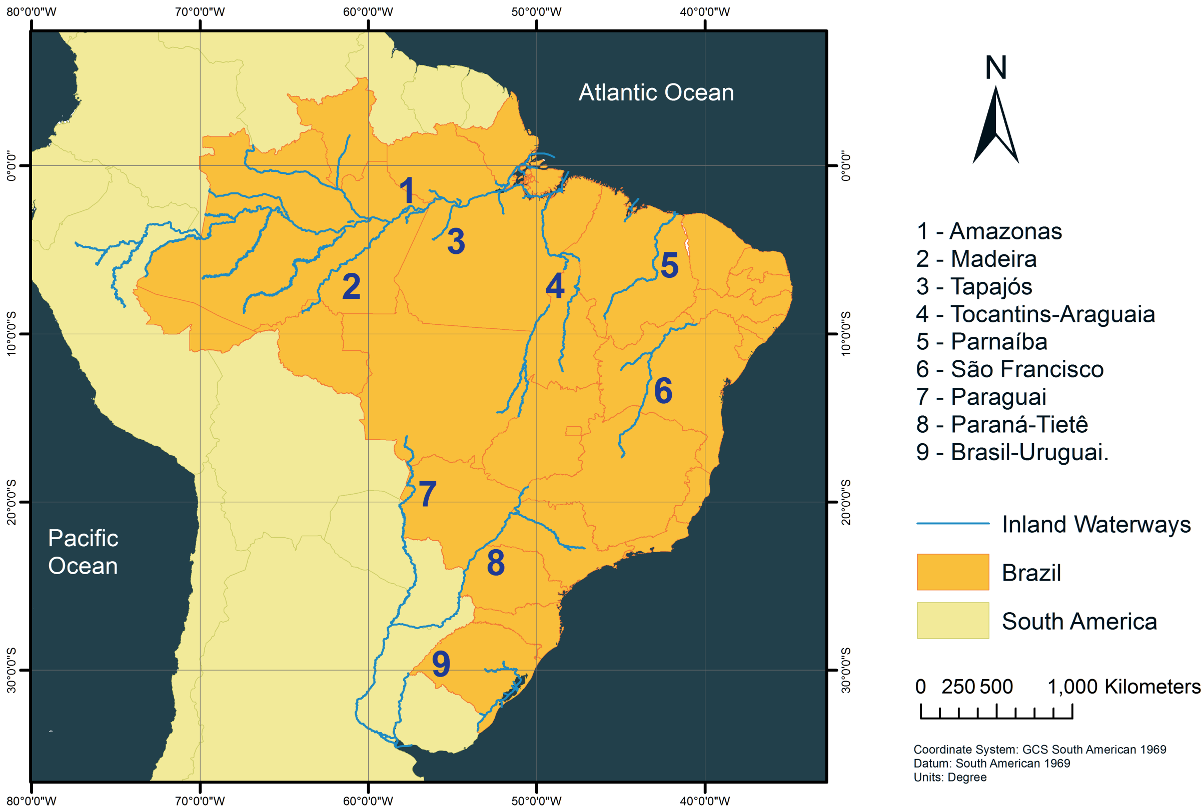


FIGURE 2 – The Brazilian inland waterways that initially received EVTEA's.

From this, one of the issues risen, as inland navigation and maintenance relies on depth availability, was the lack of a proper systematic way for defining water level scenarios as function of data availability and river characteristics. This affect the reliability of nautical chart's corrections and dredging projects reference levels for bathymetric surveys.

With this perspective, the following questions were raised: How many gaging stations are needed? What are the distances between each other necessary to achieve a desirable precisions? Which is the best way to propagate the water level readings between gaging stations, and how do they vary under different data availability and river characteristics scenarios?

Moreover, questions related to the dynamic of the river sediment transport system arise. How long does a bathymetric survey remain precise? How can the rivers morphological changing rates be defined to redo surveys when necessary? What is the role of dune motion? What are they characteristics (velocity, amplitude, period)? and how can this phenomena be taken into consideration when defining dredging solutions and reliability of bathymetric surveys?

Furthermore, the vessel and channel design criteria come into question. How can the geometrical characteristics of a river be defined to improve these designs? How to define radius of curvature of a river? As channel design has some degrees of freedom when defining it's path, how can a channel path radii of curvature be optimized within the bounds of a river margin and under the constraints of depth availability ? At last, how can a certain minimal depth be guaranteed for such vessel for a predetermined portion of the year (e.g 90%)?

In general the raised questions fall under the domain of the natural sciences — hydraulics, hydrology and geomorphology; naval engineering — vessel design; and transport engineering — channel design and maintenance; and with constraints related to the nature of inland navigation and data availability. For example, when planning a maintenance dredging over a long stretch of river, it is necessary to coordinate the propagation of the annual flood cycle with the availability and productivity of the dredging equipments — its not possible to dredge all at once. Ergo, the successful implementation of dredging endeavors relies vastly on the understanding of the periodical water level variations. These constraints may offer a new shape to problems of vastly researched topics, such as water level forecast, hydrodynamic modeling and stochastic hydrology, requiring further investigations to obtain a wider but focused understanding of the relationships in place. Figure 3 illustrates the amplitude of topics surrounding inland waterways planning and maintenance.

More specifically, the design of inland waterway channels and vessels require the knowledge of the depth's and radii of curvature that are available in the riverine system. Both this characteristics are subject to temporal and spatial variations. Although channel designs are more flexible to changes in time, it's possible to re-project the channel after each bathymetric survey — vessel's design tend to be permanent due to their long durability. As such, the correct understanding of the local flood cycle history, trends and propagations are essential for the correct definition of the vessel's draft. Also, once the vessel design has been committed, this understanding can aid the occasional maintenance of the channel's depths.

Furthermore, the depth availability problem strongly relates to radii of curvature. In a way that both are design criteria of channels that must conform simultaneously. In other words, the optimal depth related path does not necessarily fulfill the curvature requirements, making dredging sometimes necessary. This can lead to subjective trial and error processes, that rely a lot on the designer. A better understanding of both constraints can corroborate the pursue of objective methods. Figure 4, presents the overall characteristics of the problems. It's noticeable the importance of the distances between water level gage staffs, the relevance of a proper water level model, the flood cycle influence and the curvature constraint.

Given this scenario, this dissertation aims to assess and define quantitative and

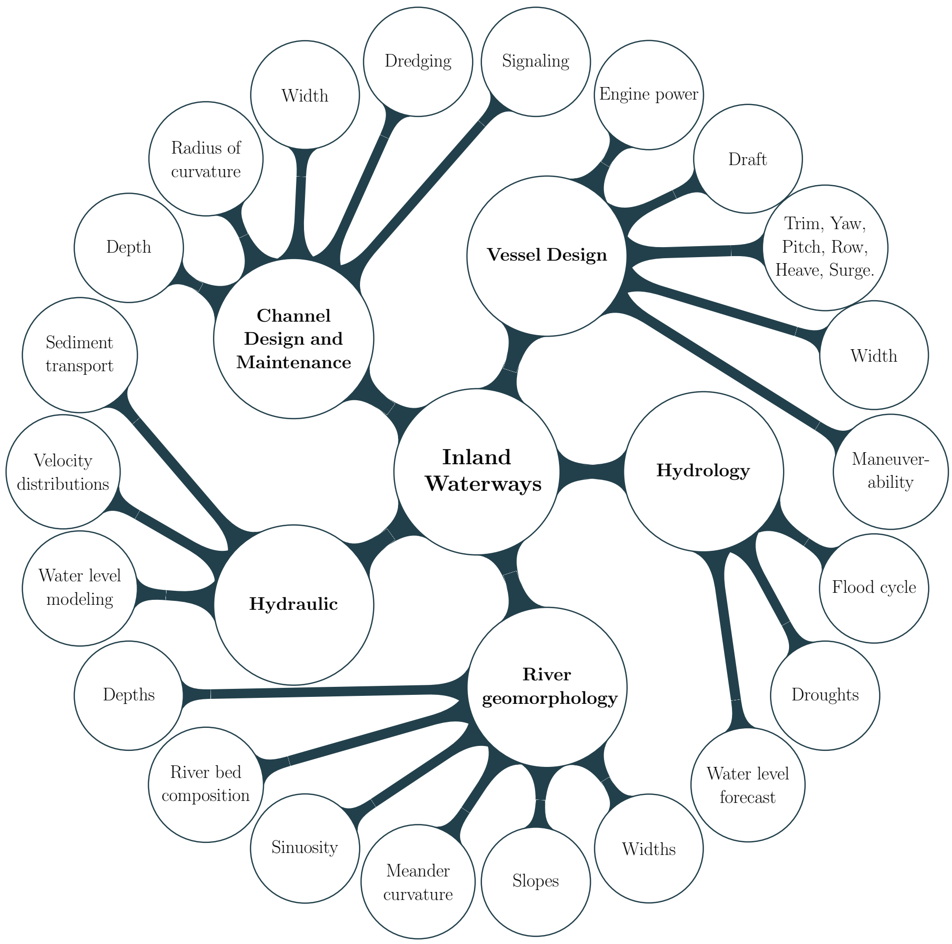


FIGURE 3 – Areas of knowledge related to the development of inland waterways.

qualitative criteria to optimize aspects of inland waterway management. Especially, those concerning the mechanics of depth maintenance, their representations, modeling, contour conditions and impact on dredging volumes with the data availability and technical limitations characteristics, specifically, of the Paraguay River Waterway.

In order to achieve the desirable goals, we sought to understand the important water level scenarios for inland waterways, assessing its temporal and spatial variability aspects and how it can be used to help ensure and optimize navigability. Through the application of water level forecast and modeling, a better method to characterize depth availability was pursued along with one to define a reference level for maintenance dredging. The figure 5 portrays the conception of the research. The focus is exclusive on natural flowing rivers (without dam's) and without tidal influences. As such, the proposed methods were tested for the Paraguay River Waterway.

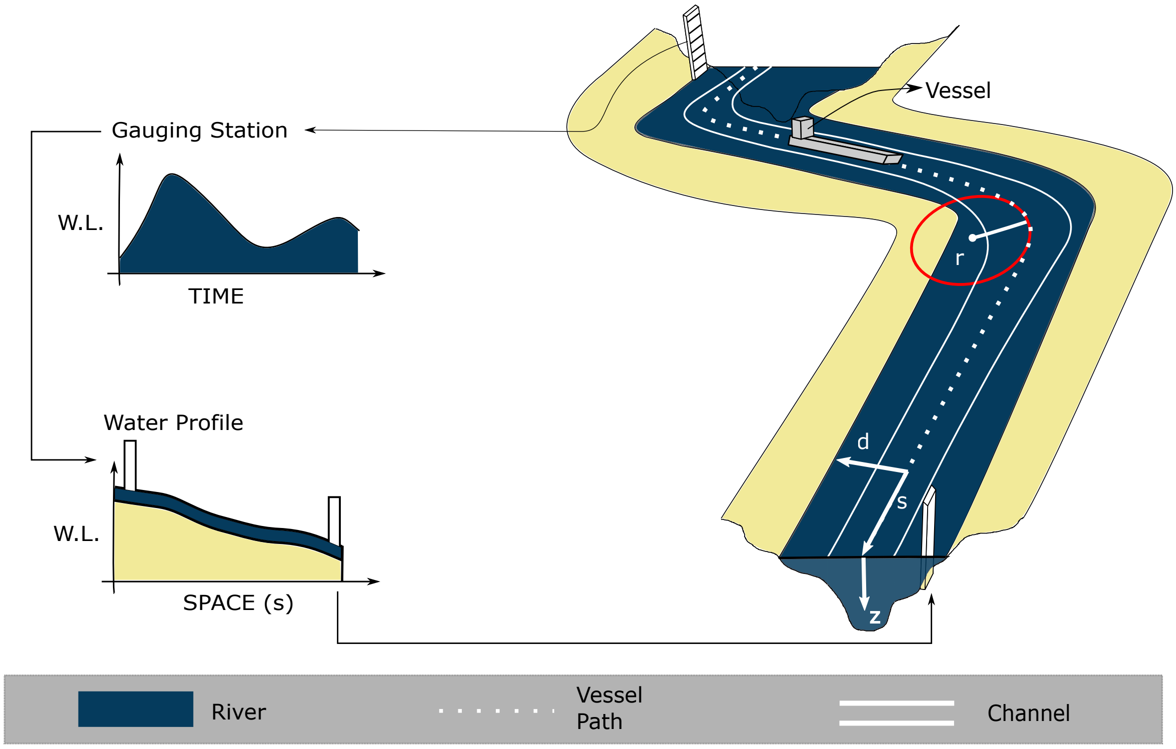


FIGURE 4 – Sketch of the water level (W.L.) variations and curvature related with inland navigation.

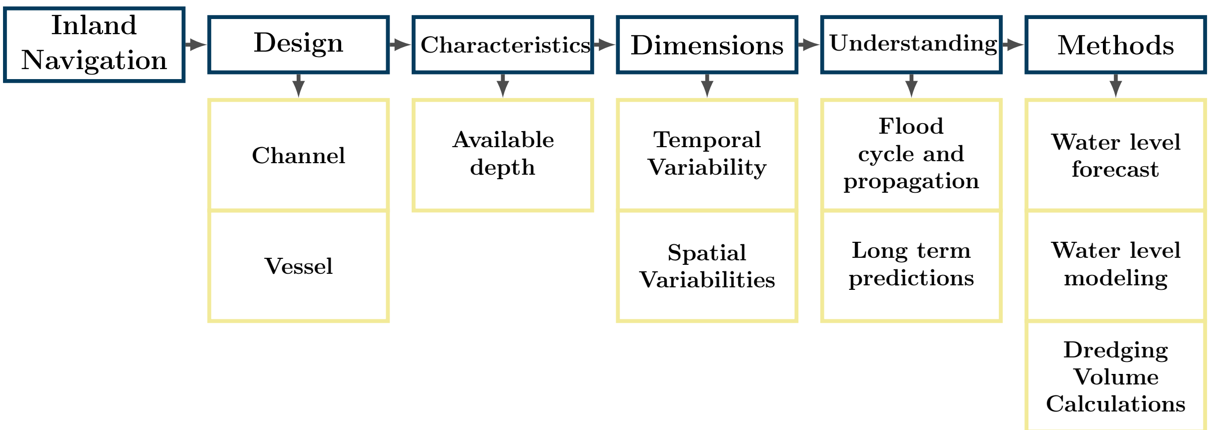


FIGURE 5 – Flow chart of the research concept.

2 LITERATURE REVIEW

The physical context of waterways is complex because it inherits from river's, its contexts and the characteristics of the hydrographic basins where they are inserted. They have distinct characteristics as to their geomorphology, morphology, course state and processes. Many authors have already demonstrated by means of methods of river classification, the extent of these characteristics variations. These include (STRAHLER, 1952; LEOPOLD; WOLMAN, 1957; WHITING; BRADLEY, 1993; ROSGEN, 1994; MONTGOMERY; BUFFINGTON, 1997).

For this reason, when reviewing the state of the art of waterway solutions, it is important to take into account these feature variations. For the state of the art in one context may not be applicable in another. In general, are rare the more generic technical solutions that take these variations into account. It is common that the state of practice won't follow the state of the art, precisely because it needs adaptations to the local characteristics. For example, a water level model where the level is linearly interpolated between two staff gages may work well for a context of high density of staffs with constant hydraulic radii and slopes. However, the same model can work very poorly in a context where the distance between the upstream and downstream staffs is large and the characteristics of the cross section, hydraulic radii, slopes, roughnesses, sinuosities vary greatly. Therefore, a possible benchmark for assessing the quality of a new method, guided by similar constraints, is the current state of practice, since this is what is objectively sought to be improved.

In this section, reviews of the state of the art and state of practice are presented for topics relevant to the achievement of the proposed objectives.

2.1 Elements of Navigation Channel Design

Inland waterway planners generally seek to optimize the economics of the overall transport chain, which includes a fair return on the infrastructure and equipment investments, and compliance with environmental criteria.

The pressure on inland waterways authorities by companies that operate inland waterways, to provide larger channels, that support larger vessel, with security and environmental resilience is a byproduct of the dynamics of economics and shipping. The costs per tonne-km of cargo, with respect to fuel, crew, and capital value for a ship, decrease as ship size increases (MCBRIDE, 2014).

The development of a successful inland waterway channel design is an on-going process, dependent on both the variations on the local characteristics (e.g. depth, meanders, sand banks, aquatic vegetation) and the economic demand of use, as the ship size may be fix but the numbers of ships also imposes pressures on the channel design as it increases the rate of ship crossings.

To elaborate, the design of navigation channels concerns to the appropriate definition of channel parameters in a way that (1) The waterway must be deep enough to ensure good maneuverability and control of the vessel in shallow water in order to prevent stranding; (2) The waterway must be large enough to allow normal boat traffic to pass safely at normal speed; (3) The vessel must reach enough speed to reduce transportation costs; (4) And the cross section should not be too large so that it does not reduce the flow velocity and cause greater deposition of sediment (AAPA, 1951). Figure 6, demonstrates a inland waterway channel design.

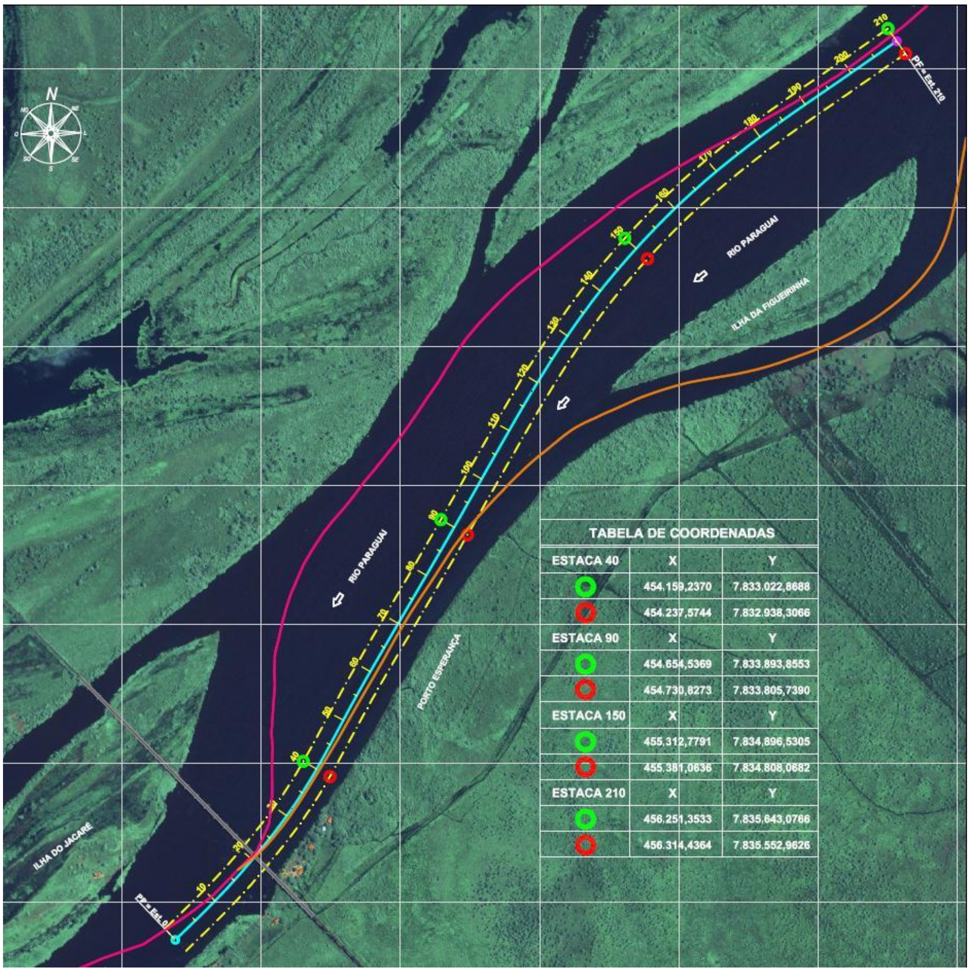


FIGURE 6 – Example of inland waterway channel design for the passo of Jacaré location, upstream of the Eurico Gaspar Dutra bridge, in the Paraguay river Waterway. Source: ITTI (2014)

In order to accomplish that, many author developed techniques that correlate the channel design parameters (e.g. width, depth and curvature) with the physical characteristics of vessel/traffic (e.g. Draft, width, length, maximal rudder angle, engine types, power , vessels/time, cargo type) and environmental characteristics (e.g. wind, current directions, water velocity distributions). In order to achieve that, some authors provide a set of rules and guidelines to design of such channels (AAPA, 1951; MCALEER; WICKER; JOHNSION, 1963; WICKER, 1965; WICKER, 1971; BOOGAARD, 1992; ABNT, 1995; PIANC, 1997, 1997). Among this authors, the recommendations of the International Association for Waterway Transportation Infrastructure (PIANC) (PIANC, 1997; MCBRIDE, 2014) found vast acceptance on harbor approaching channels design, but also for inland waterways. Currently, there is a PIANC commission handling exclusively inland waterways design guidelines, referenced as Inland Navigation Commission (InCom). To the date of the conclusion of this dissertation they hold none official final publications.

For the navigation channel projects, PIANC (1997) indicates that the minimum radii of curvature is proportional to the vessel/convoy length (L), to the ruder angle (α) and the vessel depth/channel depth ratio (R/L). Based on that, the Figure 7 presents the minimum radius of curvature requirements.

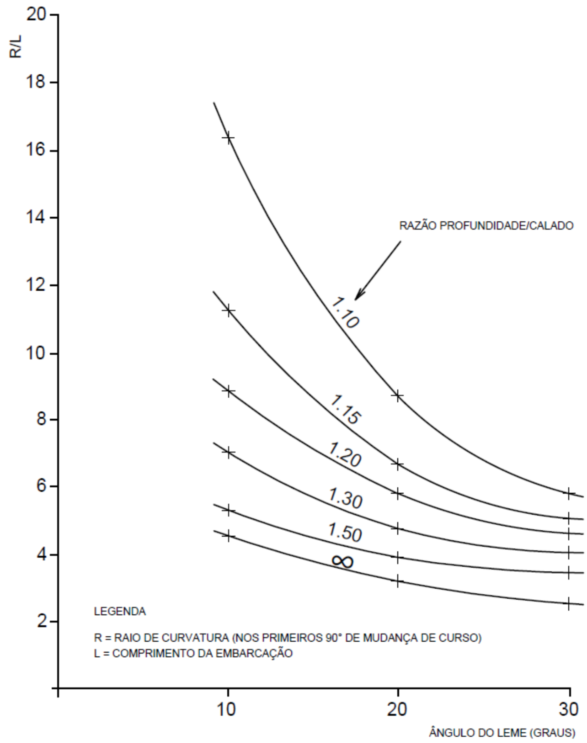


FIGURE 7 – Abacus for definition of minimum radius of curvature. In the y-axis the minimum radius of curvature over the lenght of the vessel. On the x-axis the maximum rudder angle of the vessel. The parallel lines define the depth/-draft ratio.(Source: Adapted from PIANC 1997).

The radius of curvature R at a given point in a curve is the radius of the osculating circle in the region, that is, the circle radius that best fits the curve at that point and is defined by

$$R = \frac{1}{|\kappa|}, \tag{2.1}$$

where κ is the curvature.

There are several ways of calculating the curvature, varying according to the dimensional characteristics of the curve under study. The simplest form of radius curvature calculation applies to functions of the form y = f (x). In this case the radius of curvature is calculated with

$$R = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{\frac{3}{2}}}{\left|\frac{d^2y}{dx^2}\right|}. \tag{2.2}$$

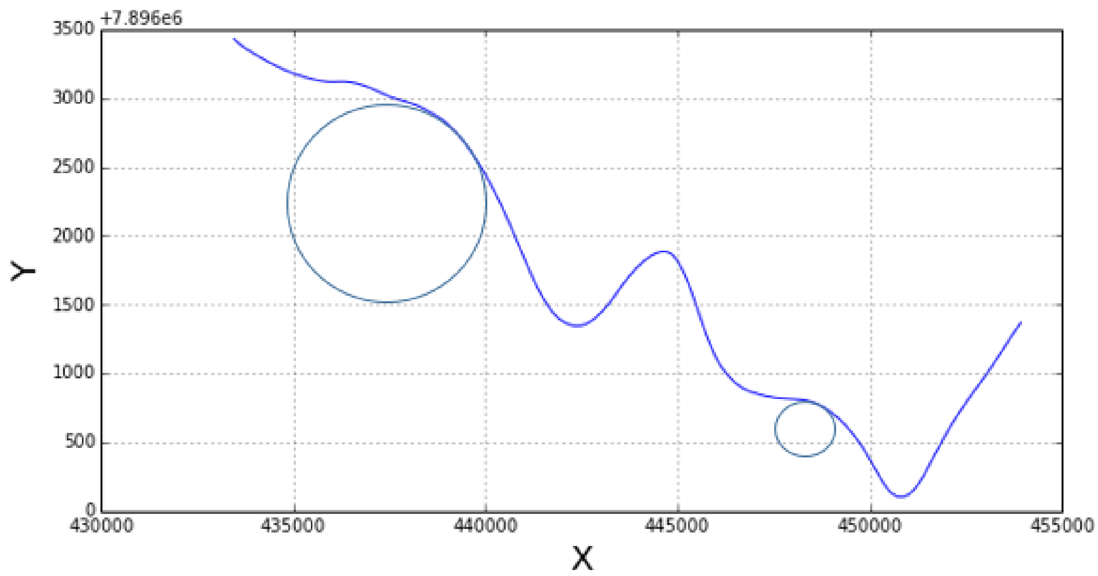


FIGURE 8 – Example of an univariate function.

The Brazilian Association of Technical Standards (ABNT) makes available through NBR-13246 the geometric parameters for the dimensioning of navigational channels of port approach, where it is stated that:

- For $\alpha < 25^o \rightarrow R = 10 \times L$ (where L is the length of the convoy and α is the deflection angle).
- For $25^o < \alpha < 35^o \rightarrow R = 5 \times L$
- For $\alpha > 35^o \rightarrow R = 3 \times L$

where L is the typical convoy length and α is the deflection angle. The deflection angle of a curve is given by the total variation of the angle accumulated in a single curve by a vessel considering that it starts that curve from a rectilinear displacement and ends it in the same way.

AAPA (1951) Channel Design method was developed based on four conditions that must be met. (1) The waterway must be deep enough to ensure good maneuverability and control of the vessel in shallow water in order to prevent stranding; (2) The waterway must be large enough to allow normal boat traffic to pass safely at normal speed; (3) The vessel must reach enough speed to reduce transportation costs; (4) And the cross section should not be too large so that it does not reduce the flow velocity and cause greater deposition of sediment. Thus, the following parameters can be calculated step by step for defining the width dimensions of the navigation channel in sinuous segments and in rectilinear segments:

- Projected vessel length;
- Width of the vessel;
- Width of the channel in still waters;
- Addition of width due to wind;
- Minimum bend radius;
- Increase width in curves;
- Average speed of current;
- Increase due to flow current;

McAleer, Wicker e Johnsion (1963) developed an empirical approach to solving the problem. The central consideration was that navigation should be performed on straight segments connected by circular arcs to make it safer. The radius of curvature of these arcs should be at least 5 times the length of the vessel and greater when the angle of drift of the vessel increases as a function of winds or currents.

Wicker (1965) Wicker (1971) developed a methodology based on the width of the navigation channel without specifying the minimum radius of curvature. Its empirical development was performed for single- and double-handed channels, considering some external effects such as yawing forces and ship controllability to obtain multiple of vessel's width that define the required channel width. The proposed function takes the following in consideration:

- Vessel size;
- Vessel maneuverability characteristics;
- Traffic density;

- Vessel velocity;
- Water depth;
- Channel type;
- Flow lines;
- Waves;
- Wind.

From these data the channel width is calculated by the composition of the maneuvering range, the sampling zone and the minimum distance between two intersecting ships.

Boogaard (1992) developed a method for dimensioning the width of navigation channels based on an equation for two-way channels and one for single channels. The calculation is based on multiples of the vessel typical width. The minimum radius of curvature of this method is expressed on the basis of multiples of the length of the vessel, and the depth of the channel projected to the depth of the vessel. The approach deals with the definition of depths and widths for rectilinear stretches and provides guidelines for compensating the width of the channel in sinuous stretches. The equation 2.3 refers to single-hand channels and 2.4 to double-hand channels.

$$W_O = W_{BM} + \sum_{i=l}^{N} W_i + W_{Br} + W_{Bg} \tag{2.3}$$

$$W_T = 2 \times W_{BM} + \sum_{i=l}^{N} W_i + W_{Br} + W_{Bg} + \sum W_p \tag{2.4}$$

where W_O and W_T are respectively the final inner width of the navigation channel in one- and two-handed design.

- W_{BM} is the channel width required for safe maneuverability under favorable environmental and operating conditions;
- W_i are additional widths due to ship speeds and waves;
- W_{Bg} are the banks to the slopes of the navigation channel;
- W_p is the crossing distance that takes into account the sum of the separation distance based on vessel speeds, and additional distances based on traffic density;

From the above equations and additional information on the characteristics of the vessels it is possible to calculate the total width of the waterway.

El-Sersawy e Ahmed (2005) compared the methodologies for the navigation channel design of Boogaard (1992), Wicker (1971), Wicker (1965), AAPA (1951), McAleer, Wicker e Johnsion (1963) and its applications to the Egyptian Nile river waterway. They obtained results similar to the analytical methods of Boogaard (1992) e AAPA (1951). with maximum variations of 5.1%. The variations compared with the other methods reach 20%. They also affirm that the physical characteristics assessment of existing natural channels must be the first step of any Waterway design. They enforce the such designs take into considerations the following topics:

- The width and depth of the channel and different moments of the annual hydrological regime;
- The sediment transport rate;
- The severity of the erosion of the banks;
- Magnitude and frequency of floods;
- Ecological considerations;
- Size of locks, ports and berths;
- Crossing visibility;
- Places and spans free of bridges;
- Weather conditions;
- Topographic and hydrographic characteristics of the canal;
- Type of waterway;
- Physical characteristics of vessels.

Among the many variables of the presented Channel Design methods, concerns related to the channel depth and depth availability have an imperative weight, thus demonstrating the importance of methods to obtain precise and accurate results for the related variables.

2.2 Navigation Related Reference Levels

pelo seguinte painel examinador In the context inland waterways water levels, several reference levels can serve as parameters to assist the maintenance of the necessary infrastructure and to provide information for navigation. They provide information on a wide range of situations, such as references to flood risk, minimum bridge height clearance, nautical chart depths and maintenance dredging. The most common reduction levels are related to nautical chart reference and dredging requirements.

Bathymetric surveys record a temporal 'photograph' of the river's bottom shape. To extend its validity, a reference level, also known as level of reduction, is commonly used.

It can be defined as the closest staff gage reading on the day of survey or a common preset level, used to reference more bathymetrys for a time interval. It is usually associated with the characteristics of the waterway navigation channel design. Its main purpose is to correlate depth values of different days and levels (ruler readings). In other words, it references the depths to a common water level, so that the differences between the survey periods do not have to be taken into account by the end user. This reference level will be referred to as Bathymetric Reference Level (BRL). Figure 9 illustrates the application of BRL.

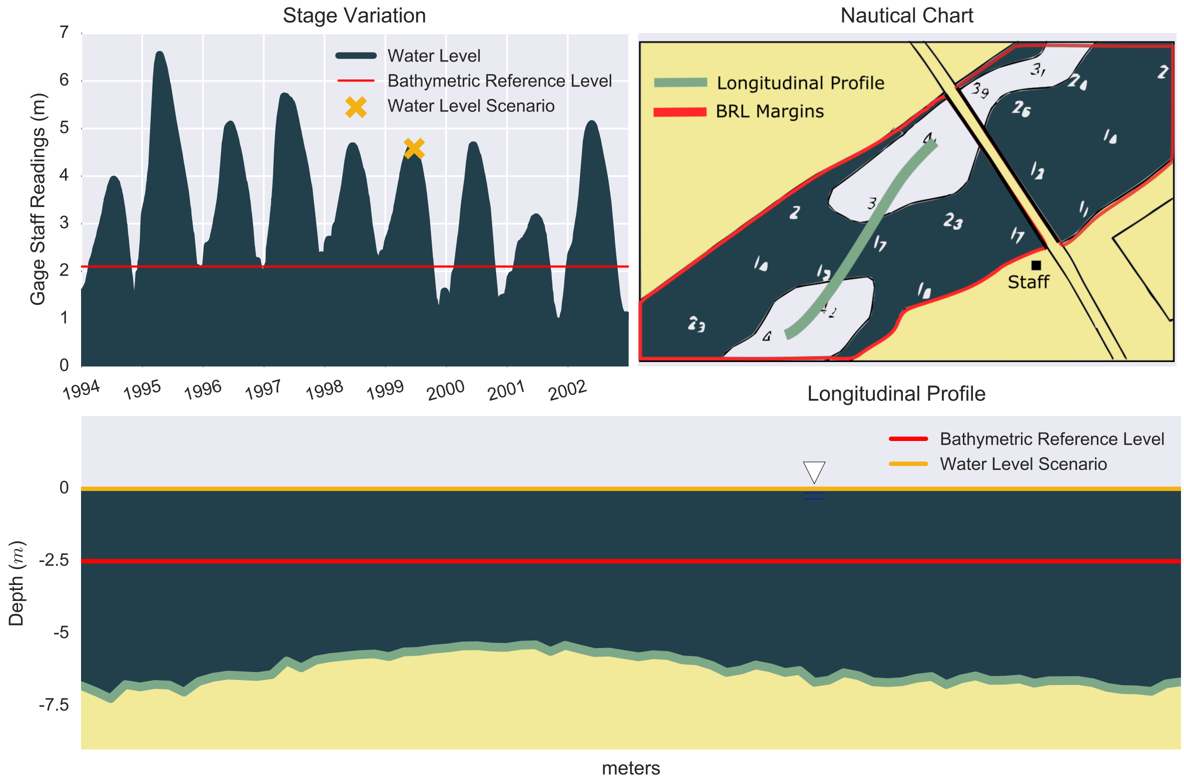


FIGURE 9 – Water level scenario and the need to relate the depths to a bathymetric reference level (BRL). Stage Variation: The evolution in time of the water level readings of a gage station. Nautical Chart: Illustration of a Nautical Chart, view in plant showing quoted points and margins at the a BRL. Logitudial Profile: A profile indicating depths in a water level scenario, different from the BRL. The values in the axis are just for the illustration of a concept and should not be taken as precise.

The safety and consistency of inland navigation operation services are closely related to operational maintenance requirements of the waterway, such as dredging depths and signaling information. The existence of sufficient depths for navigation over a pre-established period of time ensures the feasibility of implementing commercially navigable routes for cargo and passengers, allowing increased safety for navigation and investment returns. To achieve this, a detailed river characteristic's survey should be carried out to define the vessel's design of less impact and that fulfill commercial needs. However, even with the correct dimensioning of the appropriate vessel, the bottom morphology, characteristic of certain watercourses, can potentially generate shoals that interrupt navigation. In these cases, maintenance dredging is an option. To refer to the water level that corresponds to the minimum water level guaranteed throughout a predefined percentile of time, a reference level is required. This level will be refDissertação do Curso de Pós-Graduação em Engenharia de Recursos Hídricos e Ambientais, Área de Concentração em Recursos Hídricos, Departamento de Hidráulica e Saneamento, Setor de Tecnologia, Universidade Federal do ParanáeDissertação do Curso de Pós-Graduação em Engenharia de Recursos Hídricos e Ambientais, Área de Concentração em Recursos Hídricos, Departamento de Hidráulica e Saneamento, Setor de Tecnologia, Universidade Federal do Paranárred to as Maintenance Dredging Reference Level (DRL).

The maintenance cycle of inland waterways, in general, correspond to the annual cycle of flood waters and droughts. Dredging usually takes place after the peak period of the flood and before the waters become too shallow and endanger navigation. The parts of the river bed that require dredging are denoted by the plain defined by the DRL line minus the depth required for safe navigation or dredging depth (Figure 10). The depth required for safe navigation is usually composed by the vessels draft dimension plus a margin for the vessel's vertical movement (squat and trim). The dredging depth usually adds extra depths to the depth required for safe navigation related to the precision of the sounding equipment, the shoaling between two consecutive dredging and a dredging precision tolerance (ABNT, 1995).

This demonstrates the importance of the correct definition of DRL. If it is too high and the region undergoes severe drought, the vessels would not be able to navigate at their optimal configuration or not at all, resulting in losses for the industry segments and people who rely on the use of the waterway. If it is consistently too low, it would lead to unnecessary dredging of some areas, wasting significant resources and causing avoidable environmental impacts. This can occur when DRL is mistaken for BRL.

In Brazil, the concept of DRL is widely confused with BRL. The Brazilian Navy is responsible for the operation of navigation safety. For this, it provides nautical charts, nautical signals and warnings to navigators through radio and printed publications. The official reference levels for the gages staff are calculated according to the recommendations of International Hydrographic Association (IHO, 2007). The IHO recommends that the reference level definition should be adopted as the lower/upper 94-100 percentile of an appropriate series of long-term water levels. In other words, what the IHO recommends is the definition of a BRL as a considerably low level using data over a very long period, equivalent to the 0 to 6 % of the ascendantly ordered water level data series. The Navy considers a sufficient long series the period of 20 years.

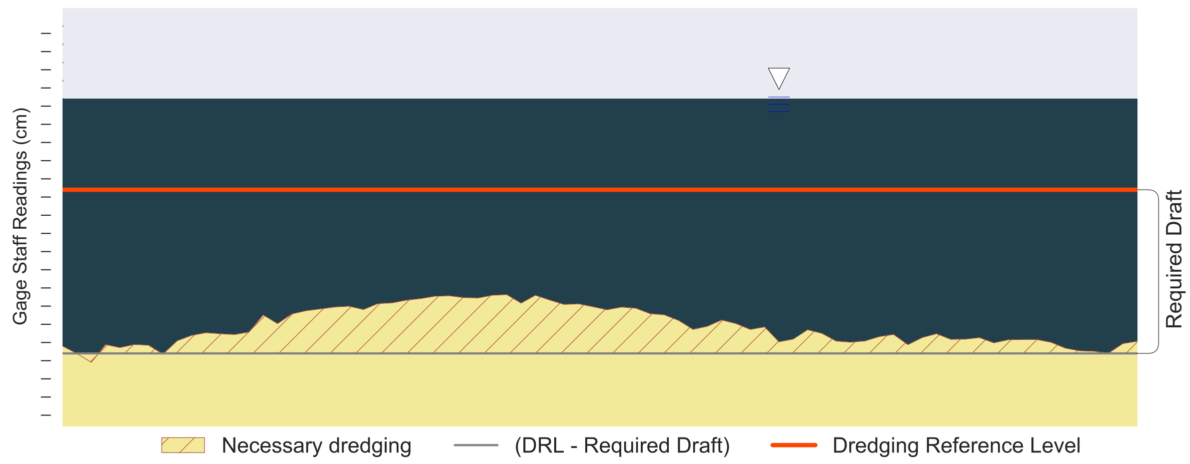


FIGURE 10 – Need to relate the depths to a reference level for maintenance dredging (DRL). The y-axis shows abstractly how water level relates to stage readings, depth availability and DRL.

Table 1 presents some Brazilian water level stations, the corresponding reference level, the percentile, the number of years and the period used for their BRL calculation. It shows that recent updates use the IHO recommendations, but the outdated ones use 10th percentiles.

Name	River	State	RL (cm)	Percentil	Years	Period
Amolar	Paraguay	MS	443	10%	20	1974 a 1999
Cáceres- PONTE BR 070	Paraguay	MT	208	10%	20	1974 a 2000
Forte Coimbra	Paraguay	MS	133	10%	20	1976 a 1999
Oriximiná	Trombetas	AM	118	6%	16	1991 a 2008
Óbitos	Amazonas	AM	109	10%	20	1985 a 2005
Humaitá	Madeira	AM	1018	10%	22	1967 a 1989

TABLE 1 – Some river stations operated by the Brazilian Navy.

The main objective of the Brazilian Navy is however, the definition of a reference level for the nautical charts it offers. A level small enough so water level corrections, most of the time, are addition operation, making it easier for end user (MIGUENS, 1996). This reference level is clearly a BRL. In this case, the method for it's calculation is appropriate. However, it may not be severe and flexible enough to serve as a DRL (FORRESTER, 1983). The Figure 11, exemplifies the calculation of the BRL performed by the Navy.

The maintenance of inland waterways is, however, an attribute of the National Department of Transport Infrastructure (DNIT). Services, such as waterways maintenance dredging and rock excavation, are contracted with a given reference level. This reference level usually considers local characteristics and demands.

Overall, to fulfill the demands of commercial uses of the waterways, a predetermined time frame validity needs to be met. What eventually happens is the misconception that the reference level of the Brazilian Navy represents a level that would allow navigation to operate within the given time frame requirement. Even if calculated with a period of 20 years and 10th percentile, it represents an old BRL for the correction of nautical charts depth values and has no purpose to attend dredging specifications. Generally, engineers take the 10th percentile of the last 20 years to refer maintenance dredgings, sometimes the lowest value of the last decade is taken (AHIPAR, 2015). With this prerogative, it is concluded that today there is no standard method in Brazil that can guarantee that the Maintenance Dredging Reference Level (DRL) is treated correctly. To that end, one of the objectives of this dissertation is to locally validate an approach to guarantee navigation by a predetermined minimum amount of time with an optimization/minimum impact mind frame, given a set of considerations and assumptions.

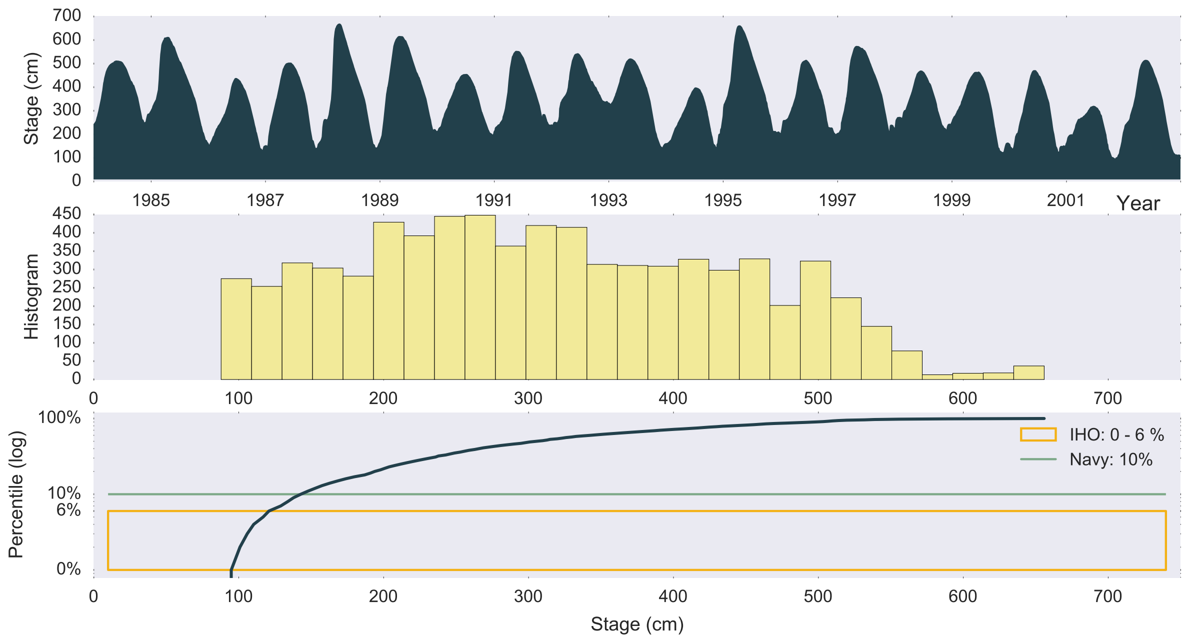


FIGURE 11 – Example of calculation of BRL using Percentile 0 to 6 and 10. First the water level data series (stage vs time). Second the stage histogram to demonstrate its distribution. Third the ascendantly ordered stage values, the line with respect to the 10th percentile and the interval corespondent to the 0 to 6th percentiles.

2.3 Stochastic Hydrological Forecast Methods

Hydrology forecast is an area of research and a set of techniques that aims to reproduce hydrological events before they take place. It has been applied to forecast droughts (MISHRA; DESAI, 2006; OCHOA-RIVERA, 2008; BELAYNEH et al., 2014; MEHR; KAHYA; ÖZGER, 2014; DJERBOUAI; SOUAG-GAMANE, 2016), floods and flow (YASEEN et al., 2016; KISI, 2005; MüLLER; FILL, 2003; ATIYA et al., 1999), water levels (WEIGANG et al., 1998, 1998; HERR; KRZYSZTOFOWICZ, 2010), sediment transport (ZOUNEMAT-KERMANI et al., 2016) and several other hydrological related parameters.

In general, it can be done using either physical/conceptual or data driven models. Physical/conceptual models, also known as deterministic, follow the premise that given a specified way things are at a time t, the way things go thereafter is fixed as a matter of natural law. While they may provide good introspections of the overall processes, due to the different types and the sheer volume of data required, they may be considered difficult to implement for forecasting applications, specially on contexts with data deficit (BELAYNEH et al., 2014). In disparity, data driven models concern to the identification of historical trends and its reproducibility. They have fast development times, minimum parameters requirements, and have been found to be accurate in various hydrological forecasting applications (ADAMOWSKI, 2008).

Several data driven forecast models are available in literature. Among them, Auto Regressive models are vastly applied in a wide set of variations. In general they aim at training (fitting) a linear function that 'learns' future event based on past behavior over, exclusively, a single variable data series (e.g water level). That is, it regresses on its own lagged values. Its plain form (AR), although it may present a useful first approach due to its easy application, has hardly any contemporary application in literature. In that sense, it falls along with more 'primitive' forecasting methods such as moving averages, weighted moving average, linear exponential smoothing and Kalman filters. These forecasting techniques are advantageous due to simplicity but disadvantageous due to *ad hoc* nature (KARTHIKEYAN; KUMAR, 2013).

Furthermore on the topic, Auto Regressive Moving Average Models (ARMA), introduced by Box (1970), have found vast applications for time series forecast in hydrology (BURLANDO et al., 1993; MOHAMMADI; ESLAMI; KAHAWITA, 2006; GÁMIZ-FORTIS et al., 2010), and has shown better generalization due to the combination of auto regressive characteristics with the insight that the regression error is in fact a linear combination of error terms that occurred at steps in the past. A generalization of this method is the Auto Regressive Integrated Moving Average (ARIMA) model. It replaces the values of the time series by the the difference between consecutive values — the reason for the extra 'I' in the name. They have been the most widely used stochastic models for hydrological drought forecasting (BELAYNEH et al., 2014). And have been successfully implemented to predict time series in a variety o scenarios (MISHRA; DESAI, 2005; HAN et al., 2010; ADAMOWSKI et al., 2012; VALIPOUR, 2015; TIAN; WANG; KHAN, 2016).

Stochastic models are linear models and have a limited ability to capture non-stationarities and nonlinearities in data. To effectively forecast non-linear data, researchers in recent decades have started using artificial neural networks (ANNs), Support Vector

Machines (SVMs) and other machine learning techniques to forecast hydrological data (DJERBOUAI; SOUAG-GAMANE, 2016).

ANNs have been used in a number of studies as forecasting tool (WEIGANG; NORDEMANN, 1996; GARDNER; CORLING, 1998; ATIYA et al., 1999; KISI, 2005; MüLLER; FILL, 2003; ADAMOWSKI; SUN, 2010; MEKANIK et al., 2013; BELAYNEH et al., 2014; DJERBOUAI; SOUAG-GAMANE, 2016). The advantage of using ANNs is their parsimonious data requirements, rapid execution time and ability to produce models where the relationship between inputs and outputs are not fully understood (BELAYNEH et al., 2014), with both endogenous and exogenous data. SVMs have been used in a number of studies as forecasting tool as well (BELAYNEH et al., 2014; SANG et al., 2016; KISI; PARMAR, 2016). SVMs can be applied to two separated problem types — classification and regression, with respective techniques named Support Vector Classification (SVC) and Support Vector Regression (SVR) (GAO et al., 2002). The eventual advantage of SVMs is that it embodies the structural risk minimization principle, while neural networks embody the empirical risk minimization principle. In contrast to ANNs that seek to minimize training error, SVMs attempt to minimize the generalization error (BELAYNEH; ADAMOWSKI, 2012).

Due to the convoluted nature of hydrological parameters, its many influences and nonlinearities, choosing the right set of input variables has beens a substantial challenge for researchers. In recent efforts to capture a wide spectrum of this variations, developments have been made towards the application of wavelet decomposed time series as input data. Wavelet analysis has been applied to the forecasting of streamflow(KISI, 2010), droughts (BELAYNEH; ADAMOWSKI, 2012; MEHR; KAHYA; ÖZGER, 2014; DJERBOUAI; SOUAG-GAMANE, 2016), rainfall (YANO; JAKUBIAK, 2016), groundwater level (ADAMOWSKI; CHAN, 2011), urban water demand (ADAMOWSKI et al., 2012), wind velocity (MENG et al., 2016) and others.

As this dissertation will focus in water level availability, more specifically on the low water scenarios, it falls under the domain of drought and water level forecasting. A wide set of input variables are available to this forecasting goal, in general, relying on drought indexes, climate indexes and/or hydro-meteorological variables. The Figure 12, adapted from the drought modeling review of Mishra e Singh (2011), presents the possible input variables, the forecasting methodology possibilities and the sought results.

A simpler machine learning technique that is often applied for forecasting, that can handle nonlinearities and take advantage of different kinds of input variables are the multiple regression models. Their main advantages are that they are significantly simpler to implement than ANN and SVMs; they have been successfully implemented in a large variations of problems; its a very flexible method, allowing the independent variables to be numeric of categorical; and can take multiple independent input variables. Its main assumptions are that the errors from the model are normally distributed; errors have constant variance; the mean of the errors is zero; and that the errors are independent (KEITH, 2015).

Within the area focus of this research, the Paraguay River Basin, some prediction modeling have already been attempt. Weigang e Nordemann (1996) applied a feedforward neural network with backpropagation learning law to predict the levels of the Paraguay River for the next 12 months. They obtained an NMSE <0.06 and a NALL <9. Although they where more interested in flood forecasting, they claim to believe that the method can be used for other predictions of future behavior depending on the range, size and quality of the available data.

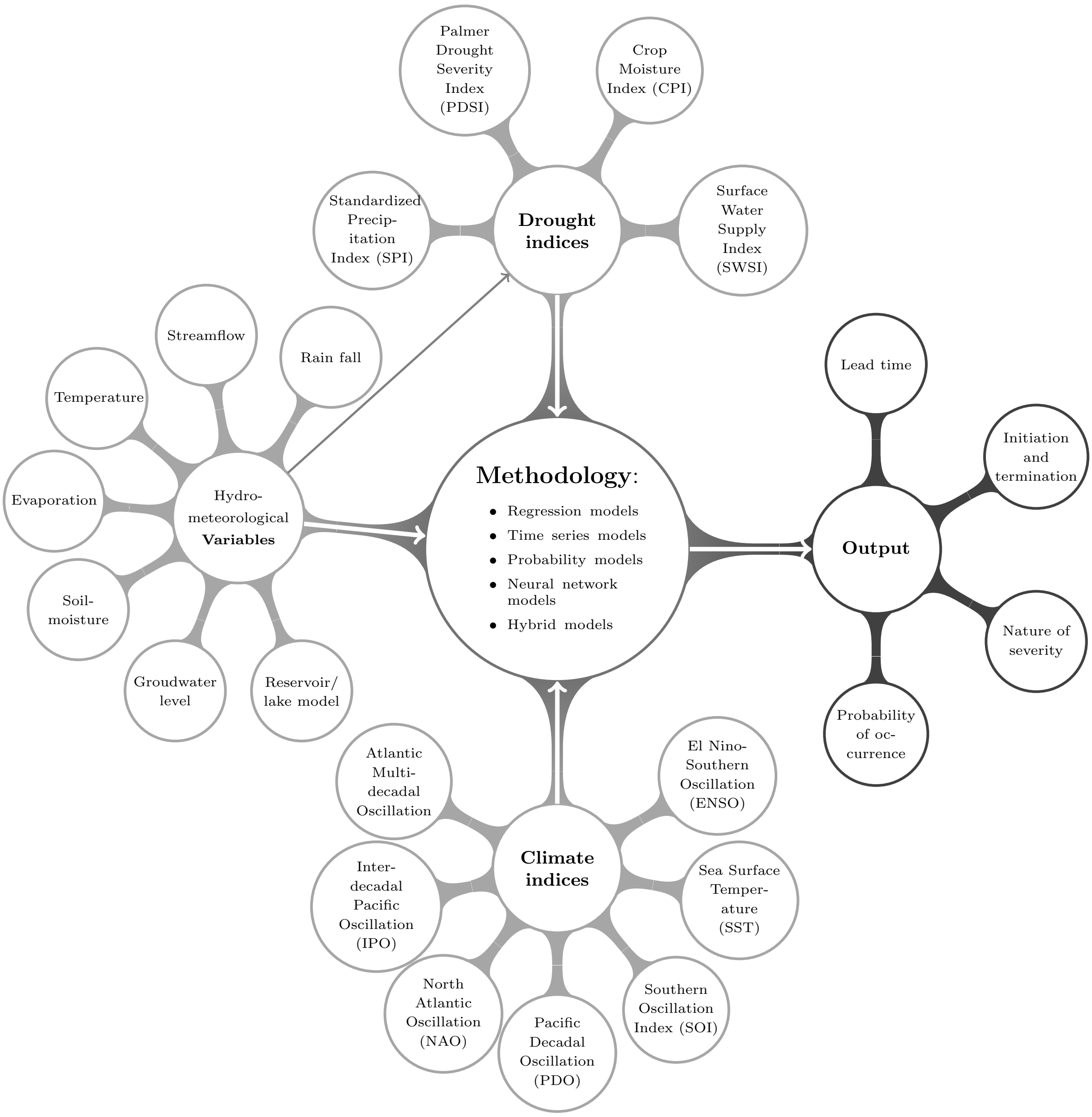


FIGURE 12 – Input data for drought forecasting. Adapted from Mishra e Singh (2011).

Weigang et al. (1998) applied a multi-layer perceptron with backpropagation learning algorithm for the Paraguay River series of levels for predictions of up to 4 months. They used 274 averages of monthly levels and reached errors of 17%.

Kono (2008) applied Learning Vector Quantilization Artificial Neural Network (LVQ) to the forecast of the water levels of the Paraguay river. The main horizon was of 4 months ahead to compare the results obtained by Weigang et al. (1998). Using only water level data the author claim to obtain 0% errors in some experiments.

Ratton (2015) applied a coupled hydrological deterministic/stochastic model to predict water levels and discharge in the Paraguay river basin. The applied deterministic hydrological model was the 3RV2 (A variation of the 3R model of Guetter, Georgakakos e Tsonis (1996)), and the stochastic prediction system was based on the Kalman Filters. The author claims the innovations were 'the development of a state updating system with assimilation of the observed flows to correct the state of the hydrological model; (2) adoption of an objective function for the calibration of the hydrological model that minimizes the differences between simulated and observed flows; (3) calibration of the Kalman filter parameters in relation to the uncertainties of the input data, observations and parameters of the hydrological model'.

The current definition of Maintenance Dredging Reference Levels (Section 2.2) opens room for improvement. As maintenance dredging has to occur prior to the drought period it suits, and yearly dredging campaigns are the norm due to variations derived from the annual (orbital revolution) hydrological cycle, a method to predict the arbitrary low percentile of the following year using only endogenous data (as currently done) has not been published yet. Initially, it is hypothesized that the application of forecast methods such as the ones reviewed in this section could yell better results than the current state of praxis.

2.4 Hydraulic Models.

Hydrodynamic modeling of riverine systems is a sub-area of hydraulics that aims in the application of mathematical models formulated over the physical understanding of water related behavior. In general, the intention is to define water related physical parameters were discrete surveys are not possible (*e.g.* Extreme events and lack of data).

Conceptual hydrodynamic models, in its many forms, derive from the Navier-Stokes equations, which describes widely the behavior of viscous fluid substances. The Navier-Stokes equations are separated into a set of equations, know as the equation of conservation of mass and the equation of conservation of momentum. They where obtained by applying the Newton's second law to fluid motion, along with the notion that the sum of a diffusive viscosity term and a pressure term can define the stress in a fluid.

As the resulting Navier-Stokes equations are too difficult to analyze for arbitrary flows (WHITE, 1998), several developments were made towards the dimensionality reduction of this equations and discrete numerical solution.

Today, helped by the increased computation power of the last decades, numerical models are vastly applied. They can be classified according to its applications and needs, are available in 1D, 2D and 3D variations and can be coupled with hydrological models. Generally, large-scale hydrological models apply simple flow routing models that focus only on the linearized Saint–Venant equations, representing wave advection and diffusion, kinematic wave approximations, or the Muskingum–Cunge model and its variations (PAIVA; COLLISCHONN; TUCCI, 2011).

Many commercial models are available and have been used for a wide set of research and engineering problems (Table 2). They have been applied to analyze the impact of river bed morphology on discharge and water levels (SALEH et al., 2013), to assess and map flood risk (SAKSENA; MERWADE, 2015; SEGURA-BELTRÁN et al., 2016), to model the swim of aquatic animals (WILLIS, 2011), to model water quality for urban rivers (XUE; YIN; XIE, 2015), to assess the variations on suspended sediment transport concentration (YANG et al., 2015), to derive flow calculations (HAN et al., 2011) and several others.

Table 2 presents a comparison of the numerical hydrodynamic models types, it's applications descriptions, typical computational times, outputs and examples of know commercial and institutional models. Historically, 1D models have been the preferred methods for river routing. According to Paiva, Collischonn e Tucci (2011) it's main advantages is that large scale 1D hydrodynamic models using limited data can have good performance. Thus, simplifying significantly the simulations and data gathering processes. Also, the necessary information for large scale hydrodynamic models can be extracted from SRTM, and that 1D hydrodynamic models can be used for flow routing in large scale hydrological models. They can also provide good water level and flood inundation results.

The low computation cost was a significant argument that boosted 1D models in the 80's and the 90's. The beginning of 21st century on, marked a significant increase of 2D modeling research for large stretches of rivers. Currently, higher dimension models are the norm for innovative research and engineering on the area, due to the increased data availability, decrease in computation times and the possibilities that the extra set of output parameters provide. Nonetheless, 1D models have a strong application were simple hydraulic output parameters are required (*e.g.* water levels), when river modeling is a secondary task to the results and where data availability still lacks.

According to Yörük e Sacher (2007), 1D models are preferred when the river has characteristics such as a 'V' shaped valleys, with high discharge in the main water course, and/or with compact rectangular cross-sections, with complex construction works and with full flowing channel calculations. The application of 2D models, in general, are applied in flatlands with large flood areas and high floodplain flows, being independent from the main channel flow. And 2D hybrid models are like 2D models, but with and expected model size of more than 100 km^2, and when the water bodies themselves are primarily ditch structures rather than larger water systems.

In general, engineers and scientists rely on the quality and quantity of available data and the objective of modeling (*e.g.* water levels, velocity profiles and discharges) to assess the cost benefit that encloses the decision towards most the suitable option.

TABLE 2 – Classification of inundation models, adapted from Néelz e Pender (2009)

Method	Description	Application	Typical computation times	Outputs	Example Models
1D	Solution of the one-dimensional StVenant equations.	Design scale modelling which can be of the order of 10s to 100s of km depending on catchment size.	Minutes	Water depth, cross-section averaged velocity, and discharge at each crosssection. Inundation extent if floodplains are part of 1D model, or through horizontal projection of water level.	Cascade, Mike 11, HEC-RAS, ISIS, InfoWorks and RS
1D+	1D plus a storage cell approach to the simulation of floodplain flow.	Design scale modelling which can be of the order of 10s to 100s of km depending on catchment size, also has the potential for broad scale application if used with sparse cross-section data.	Minutes	As for 1D models, plus water levels and inundation extent in floodplain storage cells	Cascade, Mike 11, HEC-RAS, ISIS, InfoWorks and RS
2D-	2D minus the law of conservation of momentum for the floodplain flow.	Broad scale modelling and applications where inertial effects are not important.	Hours	Inundation extent Water depths	LISFLOODFP JFLOW
2D	Solution of the two dimensional shallow water equations.	Design scale modeling of the order of 10s of km. May have the potential for use in broad scale modeling if applied with very coarse grids.	Hours or days	Inundation extent Water depths Depth-averaged velocities	TUFLOW, Mike 21, TELEMAC, SOBEK, InfoWorks- 2D
2D+	2D plus a solution for vertical velocities using continuity only.	Predominantly coastal modeling applications where 3D velocity profiles are important. Has also been applied to reach scale river modeling problems in research projects.	Days	Inundation extent Water depths 3D velocities	TELEMAC 3D
3D	Solution of the three dimensional Reynolds averaged Navier Stokes equations.	Local predictions of three-dimensional velocity fields in main channels and floodplains.	Days	Inundation extent Water depths 3D velocities	CFX

2.4.1 Water Level Modeling and Navigation

To complement the reference level definitions presented on Section 2.2, the events this reference levels define must be represented on regions in between gage stations. The upstream and downstream reference level of a region can be used as input parameters to obtain a water level model that serves this purpose. In that instance, several methods are available, separated by their intrinsic complexity and data requirements. The simplest conceivable model is the application of the reference levels in a 'step' manner with a vertical and sudden decrease in the midpoint between gages. A slightly more complex method consists on the linear interpolation of this reduction level along the river's channel. A more consistent approach resides on implementing physical hydrological/hydrodynamic models of water levels, that can vary from 1D to 3D with the possibility of a hydrological model coupling.

In Brazil, to solve the reference level transposition problem, the Brazilian Navy uses a linear interpolation between downstream and upstream Bathymetric Reference Levels (BRL) (as water level model). According to (MIGUENS, 1996), to evaluate the current depth in a certain point, the difference between the current gage reading and BRL must be interpolated as a function of their distance. Thus, the Navy presents its Nautical Charts with values referring to a low water scenario and offers its users correction *abaci* for the water depth conversion in different river levels as a function of the distance between gage rules. In some cases, when the disposition of the river is predominately longitudinal (north-south), instead of reduction as function of kilometer it is presented as a function of latitude, a simplification that may be valid or not depending on the characteristics of the river in question. The Figures 13 e 14 demonstrates both cases.

The German Federal Waterways Engineering and Research Institute (BAW) is ahead of the water level models of the German Inland Waterways network. For many years a 1D+ hydrodynamic model (CASCADE) was applied for water level calculations as described by (ZENTGRAF; FENTON; BLENINGER, 2006). Currently, a 2D model is applied for many circumstances, specially on the study of vessel/river interaction (ZENTGRAF; DETTMANN, 2010). 3D models are also being implemented mainly for morphological behavior studies of river bed and interactions with physical structures (HILLEBRAND; KLASSEN; OLSEN, 2016).

In Brazil applications of hydrodynamic models for inland navigations has been growing. As example, for engineering projects focused in local solutions, 2D models have been applied to provided morphological changing rates horizon for the best option of dredging and channel definition (ITTI, 2014); and to provide a water level for riverbed rock excavation in the Tocantins river (ITTI, 2013). As of research focused studies, it has been applied to the assessment of morphological impact of the Eurico Gaspar Dutra Bridge construction in the the Paraguay river and its resulting sedimentation and subsequent impact on navigation safety and transportation times (RATTON et al., 2012); and the effects of groins construction on the Paraguay river upstream of the Eurico Gaspar Dutra bridge (TOMAS, 2014)

Within the focus area of this research, the Paraguay River Basin, some large scale modeling have already been attempt. Bravo et al. (2012) implemented a detailed modeling of rainfall-runoff processes and flow routing along the Upper Paraguay River Basin (UPRB). The model was implemented in a two step processes '(1) simulation of the basin and part of the Paraguay River tributaries by means of the distributed large-scale hydrological model MGB-IPH using simpler flow routing methods; and (2) simulation of the main drainage network, approximately 4,800 km of river reaches, with a one-dimensional hydrodynamic model'. They obtained good results representing the hydrological regime of the basin, developing the model as a tool for understanding ecosystem functioning and assessing its resilience to anthropogenic pressure, climate change, and climate variability.

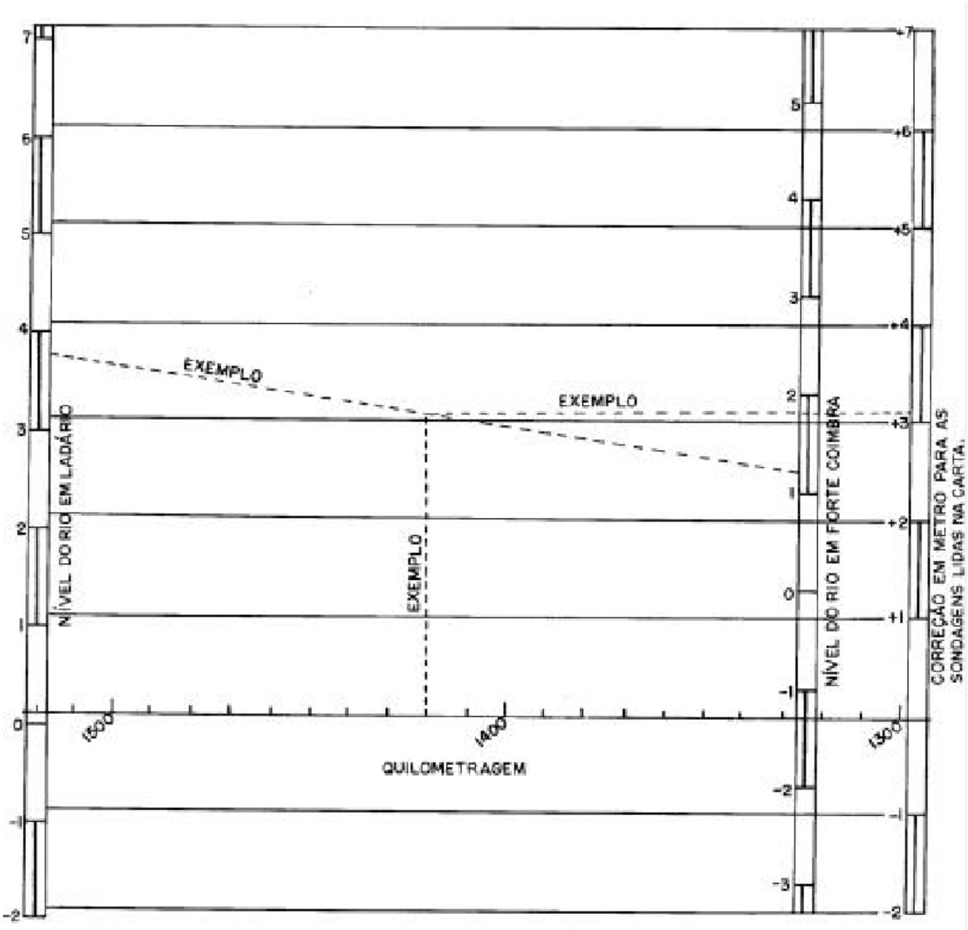


FIGURE 13 – Abacus of corrections according to the river kilometer. Source: (MIGUENS, 1996)

Aimed directly at the navigation constraints, Correia (2016) implemented a 1D hydrodynamic model of a Paraguay river stretch, for a scenario of low waters and equivalent to the reduction levels (BRL) of the local water level stations. The model was developed in a GIS environment through the extension HEC-GeoRAS. The water level calculations were performed with HEC-RAS, which is based on the 1D equations of mass conservation and momentum. The model was calibrated with 365 flow and velocity measurements along with 77 altitude checkpoints. The reference level scenario was simulated with flows analogous to the reduction levels (BRL) of the rules offered by the Navy. The same model conception was applied in UFPR/ITTI. (2015b), for the whole extension of the Paraguay River Waterway, in the context of the development of the Environmental Technical and Economical Feasibility Study of the Brazilian Paraguay River Waterway.

Although models were already implemented, there is no comparison in literature, in the context of the Paraguay river, of methods focused in low water level (DRL) models for navigation. Such comparisons are important due to the more complex and time/budget costly procedures of overall modeling. This results would also be strong motivators to the responsible entities such as the Brazilian Navy and DNIT, to reevaluate the requirements for dredging licenses and bathymetric correction *abaci*.

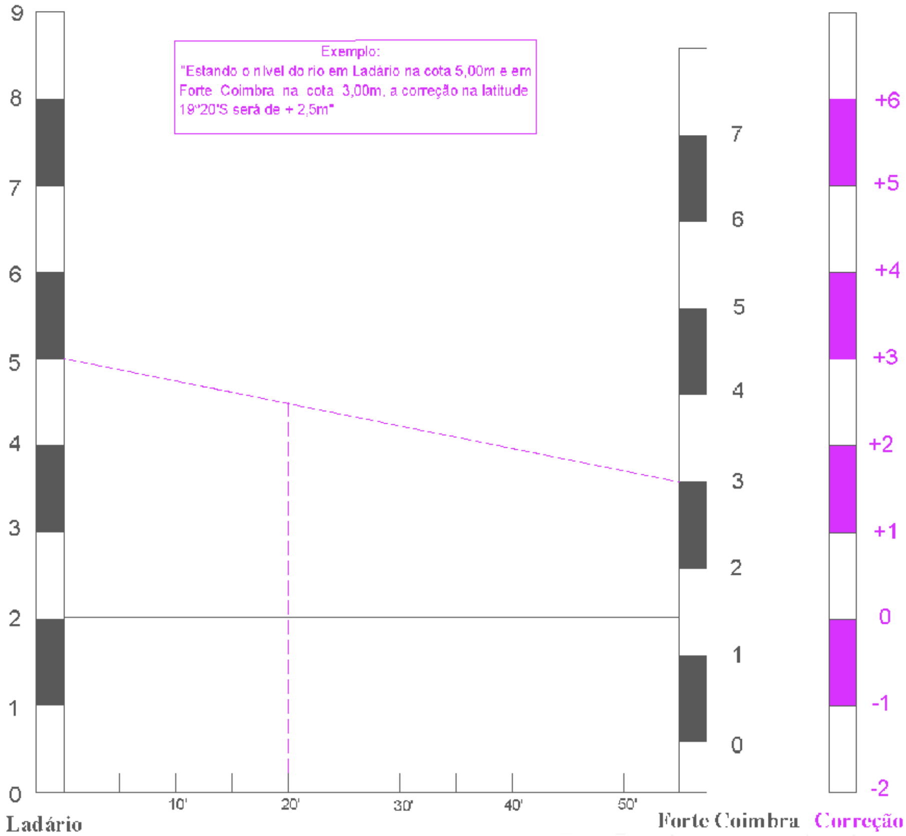


FIGURE 14 – Abacus of corrections according to the latitude. Source: Brazilian Navy Nautical Chart (BRASIL, 2016a)

2.5 Studied Area

The studied area, concern of this dissertation, is the Brazilian stretch of the Paraguay River Waterway. It is located in the states of Mato Grosso and Mato Grosso do Sul, integrating one of the largest rivers of South America an the World. The river represents an important national and international economical integration axis, and composes a rich ecological ecosystem — the Pantanal. This characteristics places it in the center of several political, economical, social and environmental interests. The Paraguay river stretch in analysis is located entirely in Brazilian territory, have approximately 1270 km and is located between Cáceres/MT and the Apa river confluence. Due to its operational and technical characteristics, the waterway is divided in two stretches 'North' between Cáceres/MT and Corumbá/MS ($\approx$ 670 km) and 'South' between Corumbá/MS and the Apa river junction at the Brazil/Paraguay border ($\approx$ 600 km) (Figure 15).

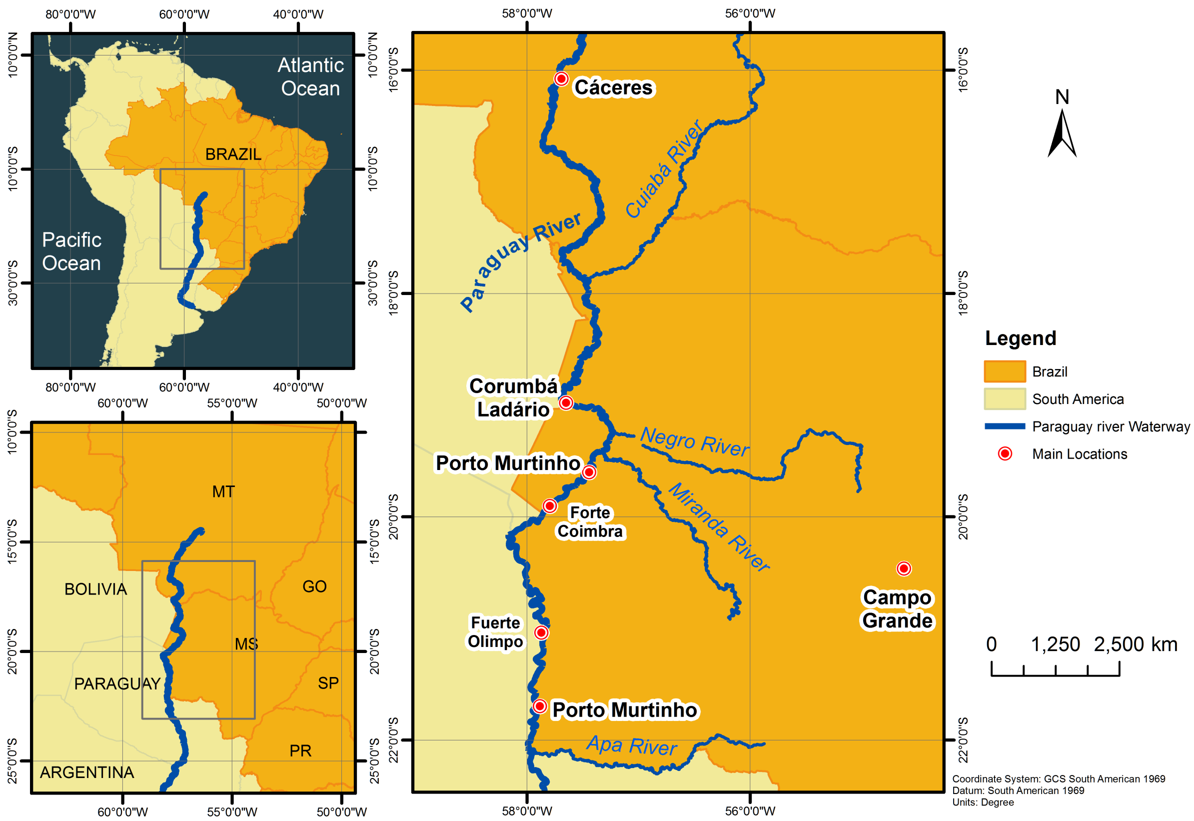


FIGURE 15 – Studied Area location and the separation between north and south stretches. Source: UFPR/ITTI. (2015b)

In the south stretch, the cargo transport is significantly more expressive than at the north stretch. The hydrogeomorphological characteristics of this stretch (wider and deeper channel) allow navigation of 4x4 convoys (Figure 18), with 16 vessels with 60 m x 12 m dimensions and one pusher of 50 m of length, reaching 2.6 m of draft (BRASIL, 2011). Adding to the draft 0.3 m of ship rotation margin and 0.1 m of error margin, the minimum depth required for full deck navigation is 3.0 m, for dredging purposes it can be added to this number a shoaling margin. The Figure 18 presents a schematic representation of the North Stretch convoy of the Paraguay River waterway. The Figure 16 presents a picture of a (4x5) convoy recently allowed by the Navy.

In the northern section, freight transport is currently inexpressive, with drastic reduction in the last 5 years. Maintenance dredges are designed to meet the traffic of tourism vessels (Figure 17b). The hydrogeomorphologic characteristics of this stretch make it an area of difficult navigability. It's know from exchanges of information with the Paraguayan Waterway Administration (AHIPAR) and with the Brazilian Navy that the region to the south of the city of Cáceres, in times of drought, presents difficulties of navigation, inclusive for vessels of Tourism with low draft. The sub-stretch called Rio Bracinho has sharp curves and problems of clogging by aquatic vegetation (balseiros and camalotes - Figure 17a). In the northern section, the maximum draft of the vessels is 1.50 m. Adding 0.30 m of boat movement clearance is defined as the minimum depth of 1.80 m required for navigation. Therefore, sites with depths less than 3.0 meters in the south section and 1.80 meters in the north section at the low water reference level are characterized as critical stretch requiring dredging.

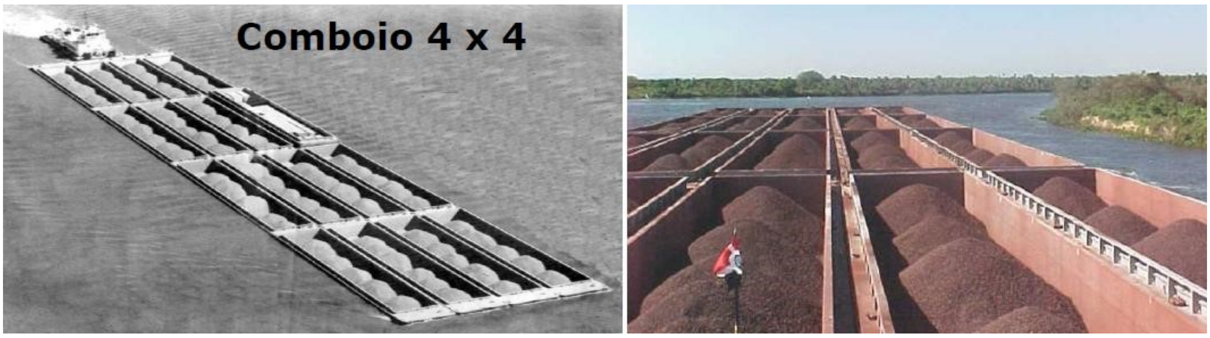


FIGURE 16 – 4x4 and 5x4 convoy used in the south stretch of the Paraguay River Waterway. Source: UFPR/ITTI. (2015b)

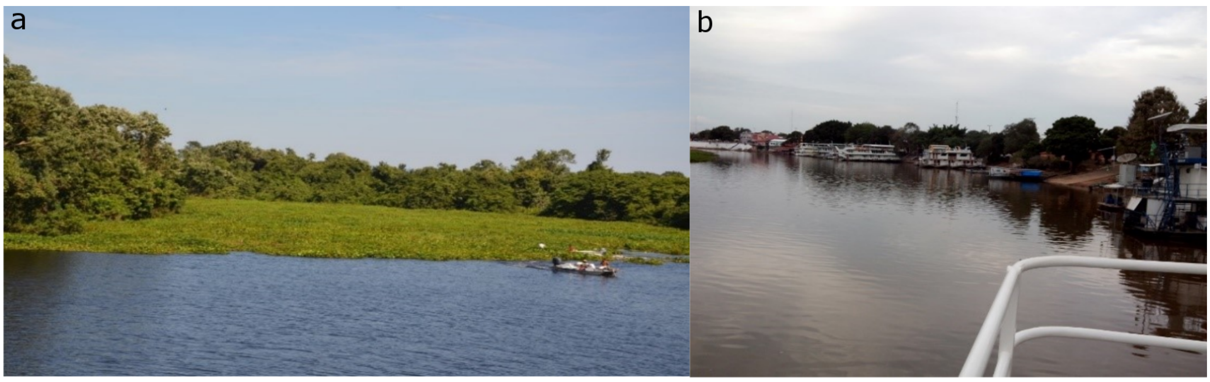


FIGURE 17 – a - Paraguay river problem of clogging by aquatic vegetation. b -Touristic vessels in the Port of Cáceres. Source:UFPR/ITTI. (2015b)

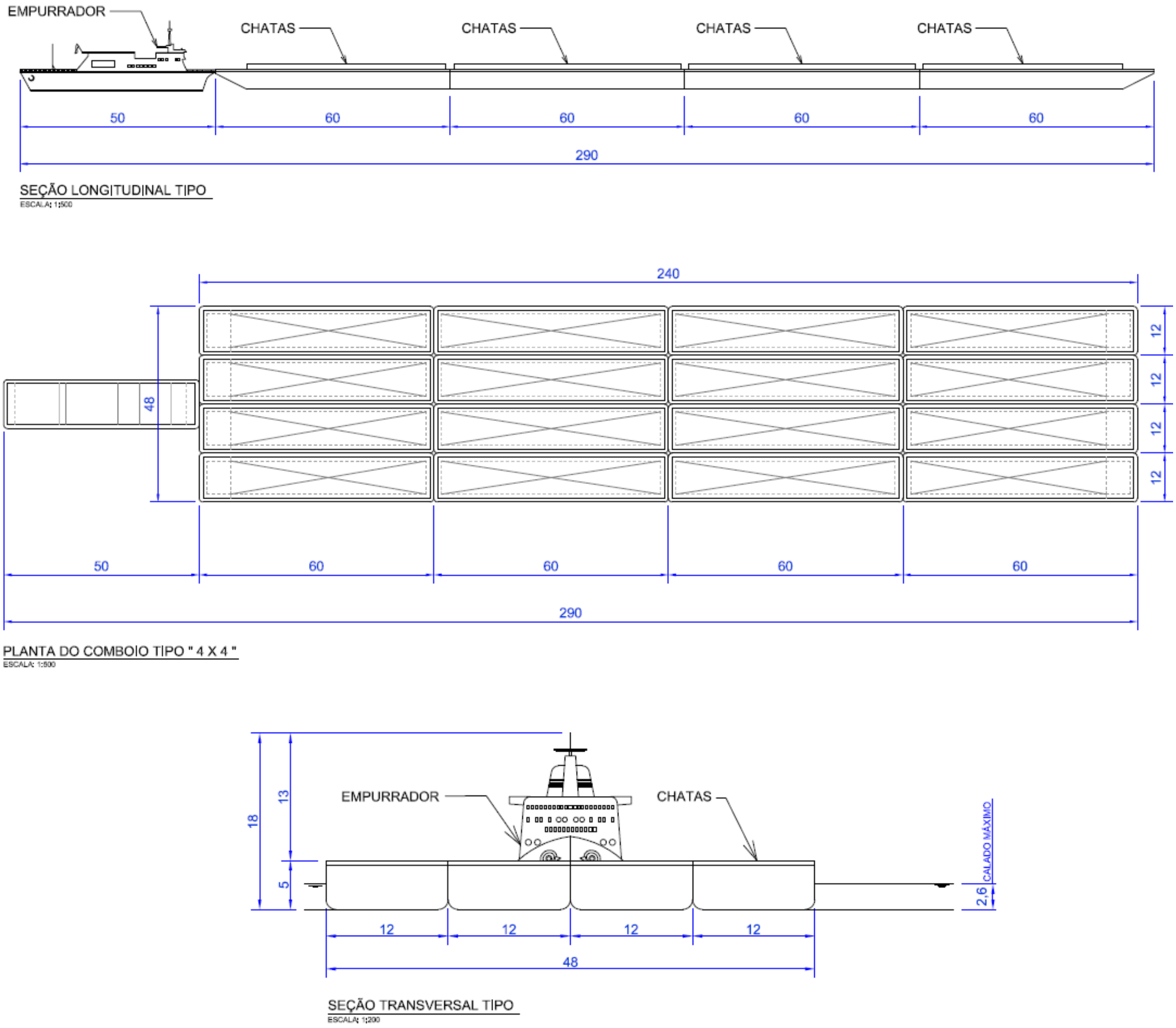


FIGURE 18 – 4x4 convoy designed for the south stretch. Source: UFPR/ITTI. (2015b)

2.5.1 Hydrological Characterization

The basin of the Paraguay River has, in its Brazilian portion, an drainage area of approximately 253,000 km^2. Along the Paraguay River and its tributaries, there are 9 fluviometric stations where 5 are used used for official bathymetric reference levels. (Table 3).

Station Name	Station N^o	Current BRL (m)	Period of Calculation	Missing Data
P. Murtinho*	67100000	3.54	1939-01 2015-09	2.29%
Ft. Coimbra*	66970000	1.34	1961-01 2015-09	8.61%
P. Esperança	66960008	1.77	1963-12 2015-09	4.47%
Porto Manga	66895000	4.41	1969-05 2015-09	17.71%
Ladário*	66825000	2.02	1900-01 2015-08	0.06%
Amolar	66800000	4.43	1967-11 2015-08	9.29%
Bela Vista do Norte*	66125000	3.38	1967-12 2015-09	1.37%
P. Conceição	66120000	3.09	1967-12 2015-08	26.15%
Cáceres - PONTE BR 070*	66070004	2.02	1965-12 2015-08	0.26%

TABLE 3 – Main fluvial stations along the Paraguay River. Source: (BRASIL, 2015).

The general behavior of the Paraguay river basin was already vastly researched in several aspects. Based on the behavior of the Ladários's stage staff, and it's secular water level data series (Figure 19), researchers classify the historical events of the basin in tree distinct periods (CLARKE; TUCCI; COLLISCHONN, 2003; RATTON, 2015; NORDEMANN, 1998). First (A) from 1900 and 1960, the second (B) from 1961 to 1973, and the third (C) from 1974 to current days.

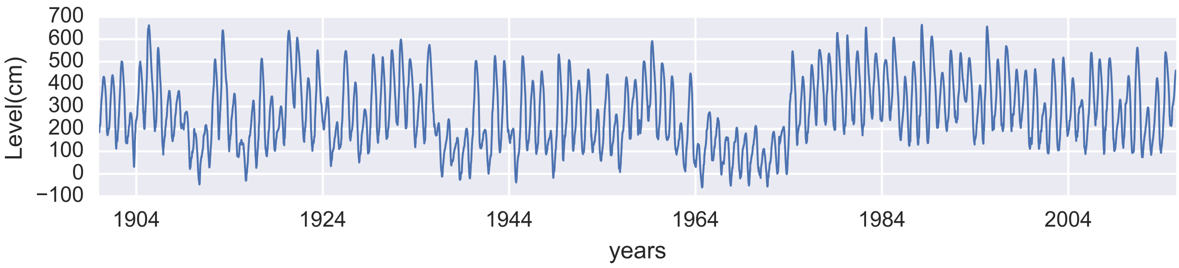


FIGURE 19 – Profile of the Ladário-MS stage staff readings (1900-2003).

The annual average and the annual minimum average of B are significantly lower the A and C. Thought, there is no significant variation on the average from A to C. In a relatively long data series as such, questions can be raised concerning the consistency of gages staff condition over the years. Possible changes of location, significant changes on cross-section, island and bypass stretches my have been formed and vanish trough the years, all of which can have an impact on the consistency of the data, rating curve variation and the ability using the data of primary forecast input.

Clarke, Tucci e Collischonn (2003) investigated the correlations between the prolonged drought B and the decrease of discharge in the basin. Normalized discharge data

$(\bar{Q}/Q_{std})$ of 20 stations were used. According to the authors, until 1973 normalized negative data were observed, and positive prevailed fro 1973 to 1980. This results reassure the general behavior of the water level stage staff of Ladário. To make sure of this conclusion, the authors also investigated the correlation between the low readings of B and the precipitation on the Paraguay river basin. In total 36 stations were analyzed and lead to the conclusion that the variations occurred from B to C strongly correlates to the increase of rainfall. This results indicates that the stage series has a good correlation with the discharge of the Paraguay river, specifically at Ladário-MS. This increases the security of using this data as an input for a forecast model.

The seasonality of the Ladário station can lay clues about the necessary forecast structure . Weigang e Nordemann (1996) described the periodical influences on the Ladário stage series from 1900 to 1995 by applying Fourier series. They concluded that the most significant influential period is of one year (Orbital revolution) corresponding to an amplitude of 130.0± 5.2 cm. Other significant influences included several amplitudes for periods of 2 to 4 years attributed to the Quasi Biannual Oscilation/El Niño Southern Oscillation (QBO/ENSO) (Table 4). The 3.8 year was atributed to the El Ninõ (ENSO), and the 28.9 to the luni-solar cycle.

Period (yr.)	Amplitude ± s.d. (cm)	Observations
1.0000 ± 0.0002	130.0 ± 5.2	Orbital revolution
28.4 ± 0.77	77.0 ± 13.7	
14.6 ± 0.34	44.5 ± 12.0	
8.9 ± 0.16	33.7 ± 11.4	
7.8 ± 0.14	33.6 ± 12.8	
6.6 ± 0.12	20.7 ± 10.2	
3.8 ± 0.03	32.6 ± 10.9	QBO/ENSO
4.8 ± 0.06	23.9 ± 10.7	
2.8 ± 0.02	22.7 ± 10.5	
2.3 ± 0.02	17.3 ± 10.1	
...	...	...

TABLE 4 – Most important periodicity parameters (Paraguay river, 1900-1995). Source: Weigang e Nordemann (1996)

This results indicates that the Paraguay River stage variation is influenced by the El Ninõ and several other periods, any reference level for navigation may has to take this consideration into account and use them to update values. The impact of this variation have not yet been studied for the complex maintenance processes of the Paraguay River Waterway. Also, there is an open space for testing if constant updating of the Dredging Reference Level is key to the proper maintenance of depths. It is hypothesized that a long period without updates may cause increasingly recurrent errors due to this unconsidered periodical variations, as it takes in consideration a past behavior that don't represents the current state of the river.

2.6 Literature Review Summary

In Section 2.1 we seen how important the accurate knowledge of depth availability is to inland navigation design. It was also concluded that today there is no standard method in Brazil that can guarantee that the Maintenance Dredging Reference Level is treated correctly (Section 2.2), and that there is room for improvements. From Section 3.5, it is hypothesized that the application of forecast methods, such as the ones reviewed, could yell better results than the current state of praxis. To solve the DRL in sections further from the water level stations, Subsection 2.4.1 showed that there are several options for choosing a proper model. And, although hydraulic models have been already implemented to low water scenarios (equivalent to a DRL), there is no literature concerned on demonstrating the variations of the available methods on the overall maintenance dredging volumes, in the context of the Paraguay river. This results would be strong motivators to the responsible entities such as the Brazilian Navy and DNIT, to reevaluate the requirements for dredging licenses and bathymetric correction *abaci*.

Objectively concerning the hydrological characteristics of the studied area, Section 2.5 demonstrated that the Paraguay River stage variation is non-stationary and influenced by the El Ninõ and several other periods. Also, that any reference level for navigation may have to take this considerations into account and use them to timely update its values. The impact of this variation have not yet been studied for the complex maintenance processes of the Paraguay River Waterway, this presents a opportunity for testing the rates of DRL updates and its quantitative impact on the proper maintenance of depths. It is hypothesized that a long period without updates may cause increasingly recurrent errors due to this unconsidered periodical variations, as it takes in consideration a past behavior that no longer represents the current state of the river. In the following chapter we present the material and methods applied to 'tackle' this reviewed issues.

3 MATERIALS AND METHODS

In this chapter are presented in details the material and methods applied to approach the reviewed issues.

3.1 Outline

First the results of Weigang e Nordemann (1996) were updated to the Ladário Stage Series by applying wavelet analysis, this helped theorize further on the necessary structures of a forecast models for DRL predictions.

In sequence, a test was developed to implemented as DRL an adaptation of the method the Brazilian Navy uses to calculate it's Nautical Chart's BRL. It consisted of checking the resulted error by varying the number of years to the past that are taken to calculate the lowest 10th percentile and by varying how long it takes to be updated. At the end, an autocorrelation analysis was performed to validate the previous findings and to contribute to the development hypothesis that set the DRL forecast approach as a viable possibility.

Based on that, five forecast models where developed, trained, tested and had it's results compared (A) the currently applied 20 years percentile; (B) the best result from the percentile parameters variation test; (C) a classical auto-regressive $AR(n)$ model, where n was best period indicated by the autocorrelation analysis; (D) a multiple regression ; (E) a multilayer perceptron trained with backpropagation learning law.

In addition, tree water level models were implemented, tested and had it's results compared for simulated dredging volume calculation. (1) The Navy's official linear interpolation method. (2) a linear interpolations between closer water level stations. (3) a 1D hydrodynamic model based on Correia et al. (2015).

Figure 20 demonstrates the differents approaches and their relationships.

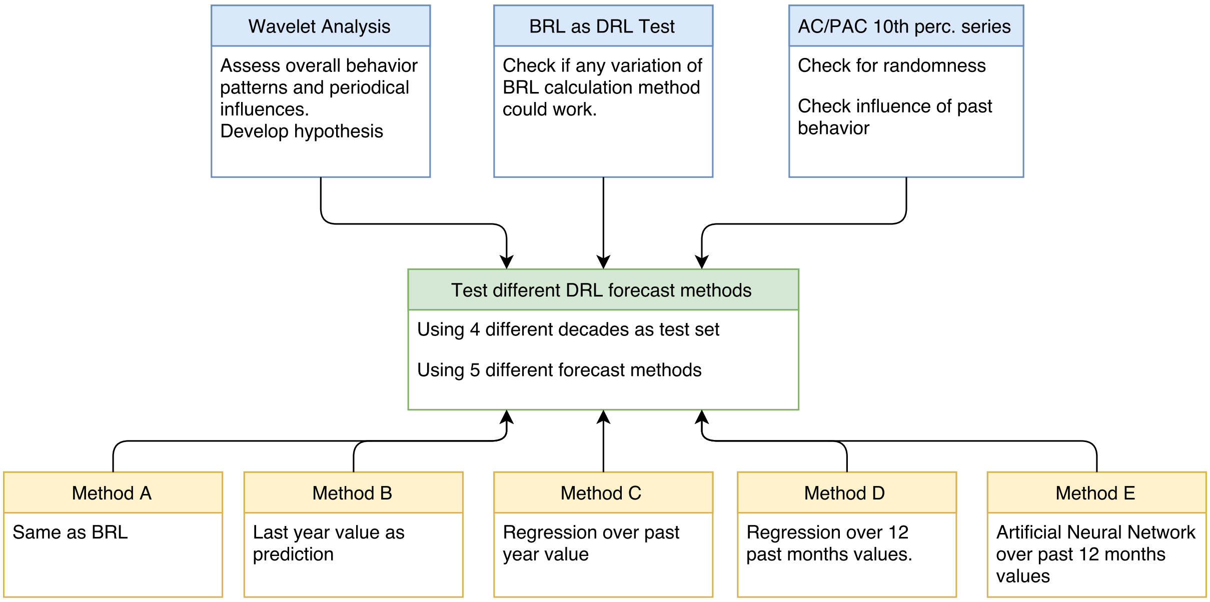


FIGURE 20 – Material and Methods outline showing the different approaches and their relationships.

3.2 Characterization of stage time series' harmonic influences

A good understanding of the periodical influences of the Ladários Stage series could assist the elaboration of hypothesis towards a better DRL forecasting method design. To accomplish so, the results of Weigang e Nordemann (1996) were updated by a more convenient approach then the plain Fourier Transform and to a longer time period. A contemporary approach was intended, so a wavelet analysis was implemented based on the developments of Torrence e Compo (1997) and Liu, Liang e Weisberg (2007). The normalized ($\frac{y-\mu}{\sigma}$) Ladário Stage Series (BRASIL, 2015) from 1900 to 2003 (Figure 19) was used as input data. The implementation was performed with the `wavelet` Python (ROSSUM, 1995) module of O'Leary (2013), that provides a lean Python implementation of the wavelet analysis outlined in Torrence e Compo (1997). The graphs and plot were implemented in Python as well. The overall script applied to this analysis is available in Appendix C.1.

3.3 Quantification of error related to the current DRL calculation method

To assess the impact of using the BRL as DRL, a water level data series was used. The main development was carried out on the Ladários Stage Series (Table 3) given its consistency and long period of records. The Ladário station is operated by the Brazilian Navy, and has one of the longest water level series of Brazil, with 115+ years of records starting at the year 1900. The records are available by the Brazilian National Water Agency (ANA) in BRASIL (2015).

Next, an experiment was developed. It consisted of calculating DRL, for all continuous subsets x with x_N elements of a given set X, and maintaining it as the DRL value for a set of x' continuous years of x'_N values, subsequent to x. With x_N ranging from 1 to 20 years and x'_N ranging from 1 to 5 years. Then calculating the errors in centimeters between the calculated and the real values. The overall intention was to verify the quantitative errors (in centimeters) associated with the current praxis ($x_N = 20$ years with non standard update restrictions), and to check which combination of numbers of year x_N and x'_N are associated with the smallest accumulated prediction errors. The last year of the period x carries the calculated error for further comparisons. The year considered for the historic Ladário series begins in January and ends in December, in subsequent analysis the years begins in July and ends 12 full months later at the end of June.

Elaborating, if the DRL is defined as the n_{th} percentile of a subset x of a set X.

It's possible to represent DRL as

$$DRL = P_n\{x\}. \tag{3.1}$$

So, for the DRL calculation of the j_{th} year

$$DRL_{\left(j,j+x'_N\right]} = P_n\{x_{(j-x_N,j]}\} \tag{3.2}$$

was implemented, where $(j, j + x'_N]$ is the period of years which the DRL will be maintained, and $(j - x_N, j]$ the period of years used for the percentile calculations.

The result is then compared with the real observed values of the set $x'_{(j,j+x'_N]}$ that follows $x_{(j-x_N,j]}$. To define the errors, the Equation 3.3 presents the absolute difference between foreseen and reality, and Equation 3.4 performs the average of this differences.

$$\epsilon_{x'_k} = \sqrt{\left(P_n\{x'_k\} - DRL_{(j,j+x'_N]}\right)^2} \tag{3.3}$$

$$\epsilon_{x'} = \frac{1}{x'_N}\sum_{k=1}^{x'_N} \epsilon_{x'_k} \tag{3.4}$$

The solutions and the final graphs were implemented with a Python script and is available at appendix C.2.

3.4 Yearly DRL Autocorrelation Analysis for the DRL forecast

To test and identify the reasons for the results of the previous section, and to assess the level of influence of past years over current ones, an autocorrelation and partial autocorrelation analysis were performed. The intention was to obtain more informations that could lead to better forecast methods design. The tests were implemented over the lowest 10th annual stage percentile of the Ladário station (3). The autocorrelation statistical test is commonly used to verify randomness in data sets, computing autocorrelation values of varying time shifts. The autocorrelations that result in values close to zero, to any and all time delay intervals, should be considered random. If one or more autocorrelation values is significantly different from zero (ranging from 0 to 1), they are considered not random. The partial autocorrelation test is commonly used to help decide the order or an autoregressive forecast model. The analysis was implemented in the Python scripting language, based on `statsmodels` [1] statistical library, and it's available in Appendix C.3.

[1] http://statsmodels.sourceforge.net

3.5 DRL forecasting methods for depth assessment and dredging volume definition

In this section the methods for the sought forecasting of DRL are described. In total 5 methods were tested, namely: (A) the current applied 20 years percentile; (B) the percentile of the numbers of years that obtained the least amount of average error from the test presented in Section 3.3; (C) a classical auto-regression $AR(n)$ model using as n the best period of the partial-autocorrelation analysis (Section 3.4); (D) a multiple regression using data from previous months; (E) and an multilayer perceptron trained with backpropagation learning law using the same input data as method (D).

An experiment was designed to compare the models' results, submitted to different training periods and hydrological scenarios. The methods were tested with four different continuous periods of 10 years each (Table 5). Among which two were quasi-stationary periods (1951: 1960) and (2001: 2010); And two non-stationary (1961: 1970) and (1971: 1980).

Period and Validation	Testing	Training
1951:1960	10 years	50 to 59 years
1961:1970	10 years	60 to 69 years
1971:1980	10 years	70 to 79 years
2001:2010	10 years	100 to 109 years

TABLE 5 – Forecasting benchmark periods.

Every scenario used the same dataset, varying only the preprocessing steps. The Ladário raw ASCII stage records from ANA (BRASIL, 2015) were used. First the values were stacked into rows by date and separated into consisted and not consisted data (according to ANA's classification). The consisted data was entirely used, and not-consisted data was used from 2003 to 2010. As it only represents 0.06% of the total amount of records, the sparse missing data were linearly interpolated between days. There weren't any significant long periods without records.

As the average peak of this stage series occurs in June, the hydrological year was defined starting in the 1st of July and ending at the 30th of June. For numerical reasons, the Julian year at the last day of the considered hydrological year, is the year that names it (*e.g.* July of 1990 to June of 1991 is the hydrological year of 1991). Every method used the series of hydrological years' 10th percentiles as target for the predictions. A Python module called `anastage` was developed to standardize the preprocessing procedures. Its general structure is presented in Figure 21 and the complete module source code is available at Appendix D.1.

To execute the methods a Python module called `forecast_methods` was developed. Its general structure is presented in Figure 22, its detailed structure at Appendix B.1

and its source code in Appendix D.2. In the following subsection the applied methods are generally described.

A) 10th percentile of 20 year.

This method follows the same methodology as currently applied for BRL calculations for inland nautical charts and many times used for DRL calculations. The calculations are the same as the ones seen in Section 3.3, where the 10th percentile of a 20 years period is taken to be used as the following years DRL. The solution was implemented through a Python script available at Appendix C.4, and the structure of the applied forecast class called `forecast_models.basic_forecast` is presented in Appendix B.1. The complete source code is available in Appendix D.2.

B) $10th$ yearly percentile.

This method is analogous to the method 'A' but with the percentile calculations using the past 1 year instead of 20. The 1 year percentile was implemented after the results presented in Figure 11 and the autocorrelation plot demonstrated in Figure 37. The solution was implemented trough a Python script (Appendix C.4), and the structure of the applied forecast class called `forecast_models.basic_forecast` is presented in Appendix B.1. The complete source code is available in Appendix D.2.

C) $10th$ yearly percentile AR(1) model.

This method consists of implementing an autoregressive model of order 1 to forecast the following year's 10th percentile. Assuming X as the set of the 10th percentiles of every year in the original dataset, where $X = \{x_0, x_1, ...x_t, ...x_n\}$. The autoregressive model considers the prediction values as a regression of the previous years values,

$$x_t = \theta x_{t-1} + \beta, \tag{3.5}$$

where θ is called the weight and β the bias values.

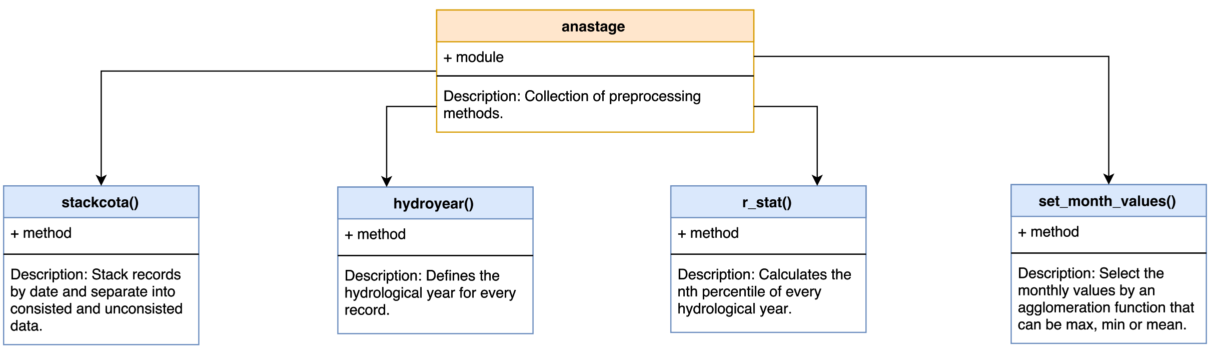


FIGURE 21 – Python module 'anastage' general structure.

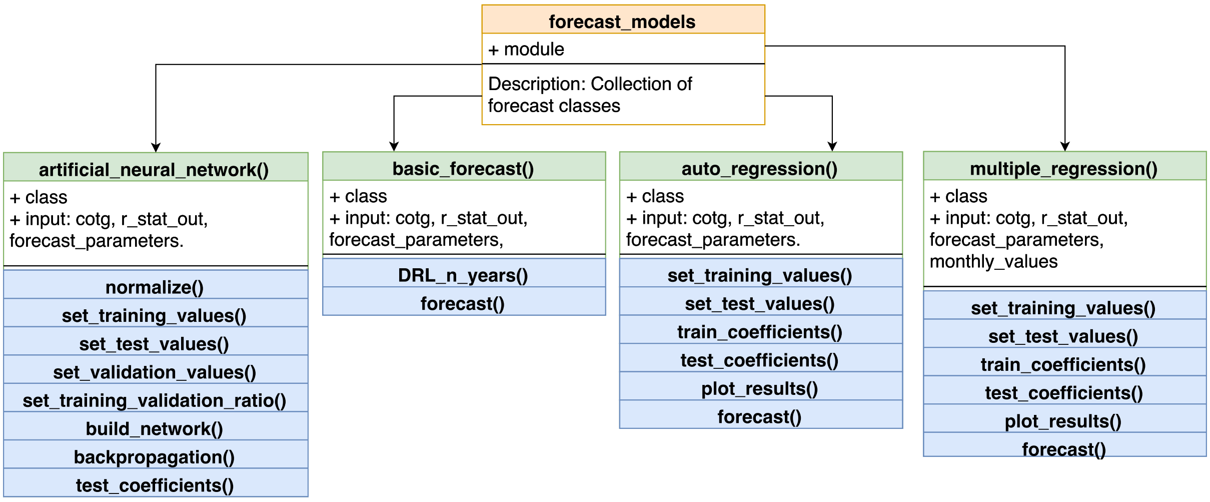


FIGURE 22 – Python module 'forecast_models' general structure.

Following this idea, given the matrices

$$\bar{X} = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_{n-1} \end{bmatrix} \text{ and } \bar{Y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{3.6}$$

we can train the model finding θ and β, by solving

$$\bar{\Theta} = (\bar{X}^T \times \bar{X})^{-1} \times \bar{X} \times \bar{Y}, \tag{3.7}$$

where $\Theta = \begin{bmatrix} \beta & \theta \end{bmatrix}$.

The solution was implemented trough a Python script (Appendix C.4). The structure of the applied forecast class called `forecast_models.auto_regression` is presented in Appendix B.1 and the source code presented in Appendix D.2.

D) Multiple Linear Regression Model

This model consists of implementing a Multiple Linear Regression Model, fitting previous 12 monthly maximum stage values to predict the following years 10th percentiles. Given that a year has 12 monthly maximum stage values of $M = \{m_1, m_2, ...m_t, ...m_n\}$ and that X is the set of the 10th percentiles of every year in the original dataset, where $X = \{x_0, x_1, ...x_t, ...x_n\}$. The method assumes that a forecast for a given year t can be represented as

$$x_t = \theta_1 m_1^{t-1} + \theta_2 m_2^{t-1} + \ldots + \theta_{12} m_{12}^{t-1} + \alpha + \epsilon, \tag{3.8}$$

where ϵ is the forecast error or residual. The values of θ to achieve the best results, can be found applying the matrices

$$\bar{\bar{M}} = \begin{bmatrix} 1 & m_1^1 & m_2^1 & \dots & m_{12}^1 \\ 1 & m_1^2 & m_2^2 & \dots & m_{12}^2 \\ \vdots & & & \ddots & \\ 1 & m_1^n & m_2^n & \dots & m_{12}^n \end{bmatrix} \text{ and } \bar{Y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \tag{3.9}$$

to solve

$$\bar{\theta} = (\bar{\bar{M}}^T \times \bar{\bar{M}})^{-1} \times \bar{\bar{M}} \times \bar{Y}, \tag{3.10}$$

where $\bar{\theta} = \Big[\alpha, \theta_1, \theta_2, \dots, \theta_n\Big]$.

This solution was then applied recurrently to every year. The training periods for each forecast consists of the input samples $\bar{M}$ of every year prior to the year the forecast is aimed. The solution was implemented trough a Python script (Appendix C.4) The structure of the applied forecast class called `forecast_models.multiple_regression` is presented in Appendix B.1 and the source code presented in Appendix D.2.

E) Artificial neural network.

The idea that supports this method is that an artificial neural network could be trained with monthly stage values (maximum, minimum or medium) to predict the following year's 10th percentile (DRL). It lays on the assumption that the disposition of the stage on this previous months carry information about the current hydro/meteorological state of the basin and region, thus allowing a forecast within certain bounds of uncertainty. The artificial neural network training process aims at defining the weights of the structure that the results of a feed-forward simulation are as similar to real ones as possible. As input data the stacked stage/date series was pivoted by the hydrological years' months (Table 6). In summary, for a test year t composed of input set (month values) and target value (10th percentile), all the year that precedes it are selected to be separated into training and validation sets.

A Principal Component Analysis (PCA) was applied over monthly stage values and transformed into a 2 component series for every year including the test year. In this transformed set the 10% to 5% closest points (Euclidean distance) to the transformed values of the test period input where selected as validation period, given its similarities with the test period input (thus, assuming, with the test period target as well). The rest of the points were selected as training input. It's important to emphasize that this analysis is only used to select the years for the training and validation sets, the network training is performed over the original non-transformed data.

Hydro. Year	Jul	Ago	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
1900	NaN	NaN	NaN	NaN	NaN	NaN	212.0	245.0	314.0	358.0	410.0	432.0
1901	432.0	422.0	384.0	330.0	222.0	187.0	202.0	247.0	328.0	389.0	432.0	439.0
1902	431.0	397.0	334.0	214.0	150.0	165.0	220.0	282.0	354.0	443.0	493.0	500.0
1903	496.0	451.0	397.0	320.0	204.0	142.0	164.0	186.0	215.0	261.0	271.0	269.0
1904	253.0	201.0	115.0	102.0	195.0	240.0	257.0	308.0	363.0	462.0	495.0	500.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2009	507.0	465.0	400.5	293.0	157.0	115.0	129.5	172.5	206.5	228.5	265.0	306.0
2010	330.0	330.0	297.0	218.0	145.0	162.0	236.0	313.0	319.0	350.0	416.0	436.0
2011	432.0	400.0	317.0	182.0	108.0	102.0	139.0	244.0	433.0	522.5	562.0	562.0

TABLE 6 – Maximum monthly stage series in centimeters.

The network shape chosen was built with a shape of 12 cells and 1 bias in the input layer; 1 hidden layer with 8 cells and 1 bias; and 1 output layer of 1 cell. For the inner layer a sigmoid activation function was used, and for the output layer a linear activation layer was used. A back-propagation algorithm was used to train the weights that connect each cell. The learning rate was set to 0.0001, the maximum number of epochs to 2000 and the weights started randomly.

After several trial and error attempts to find the best training strategy, the following design was implemented. First the the PCA analysis is implement to separate train and validation sets, then the network is build with the standard shape and started with random weights, after that it is trained through backpropagation until convergence (with the training set). The validation set is then feed-forward through the network and its results compared with real values. If the average error from that is greater than 30 cm the network is randomly started again and backpropagation restarted(this tends to avoid the convergence of local minimum error associated with the cost function). If the the averaged error is smaller than 30 cm the test year input parameters are fed to the network and its forecast and error stored for the analysis. Figure 23 illustrates the steps of the method.

The method was implemented with a Python script available at Appendix C.4. The structure of the applied forecast class called `forecast_models.artificial_neural_network` is presented in Appendix B.1 and the source code presented in Appendix D.2.

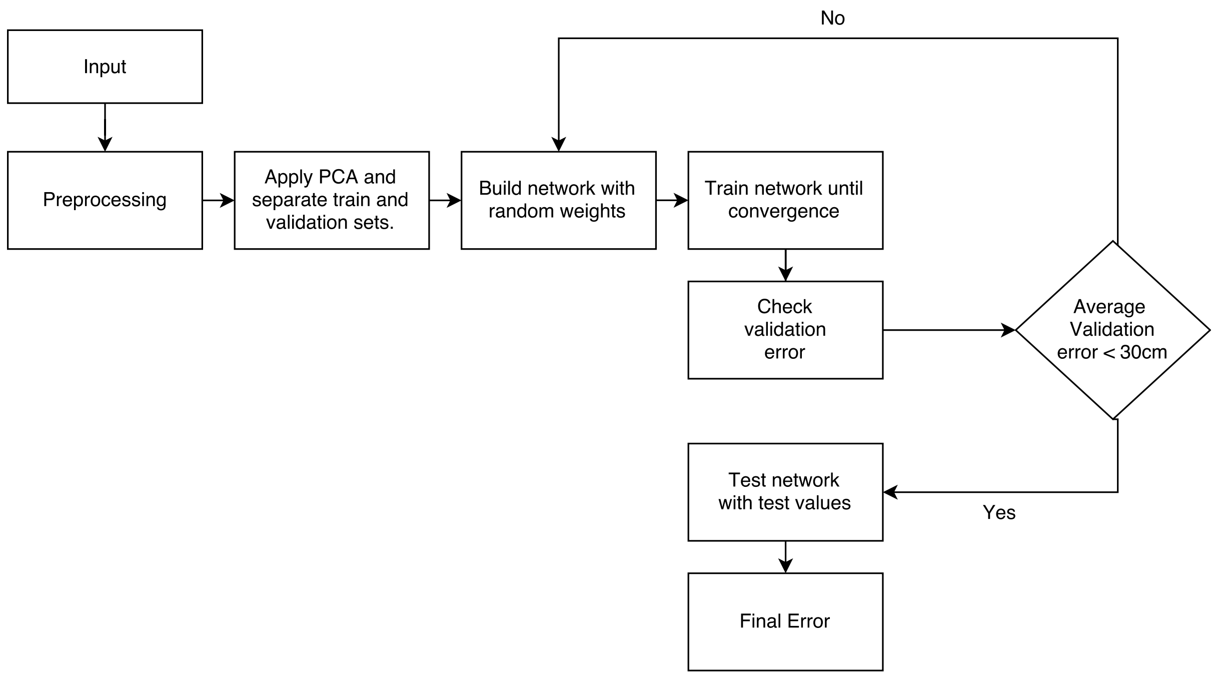


FIGURE 23 – Flow chart of the Artificial Neural Network based method.

3.6 Water level modeling for dredging volume definitions

This section presents the methodology and the materials used to assess the impact of water level models in a simulated dredging case study. The intention is to test the resulted variations between using a 1D model and a linear model, also to test the impact of using non-update traditional DRL and forecasted DRLs. The studied area is the stretch of 126 km that extends from Ladário (MS) to Porto Esperança (MS). More specifically, the critical stretch known as Caraguatá Island will be used as benchmark to display the volumetric variations of each method. Figure 25 presents the location map of the studied area.

The experiment was conducted by firstly linearizing the bathymetric survey data with a method that attributes a 'longitudinal' distance from the station of origin (Ladário) through the the river's center point to all surveyed points. Secondly, the surveyed points received an altitude attribute by linear interpolation of the elevation reference points (Table 7) along the river. In sequence, the depths were recalculated by subtracting the surveyed points altitude from the water level of the different methods. Once the depths were calculated, the ones within the channel were selected an with a buffer to avoid interpolation bias, interpolated into a surface raster for volume calculations. The dredging volumes were calculated for the surface sections that stayed above the depth of 3.3 m composed of the minimum depth requirement (3.0 m for the Paraguay River south stretch) added of a shoaling margin of 10%.

Figure 24 illustrates the overall strategy for the dredging volume definitions and errors assessment.

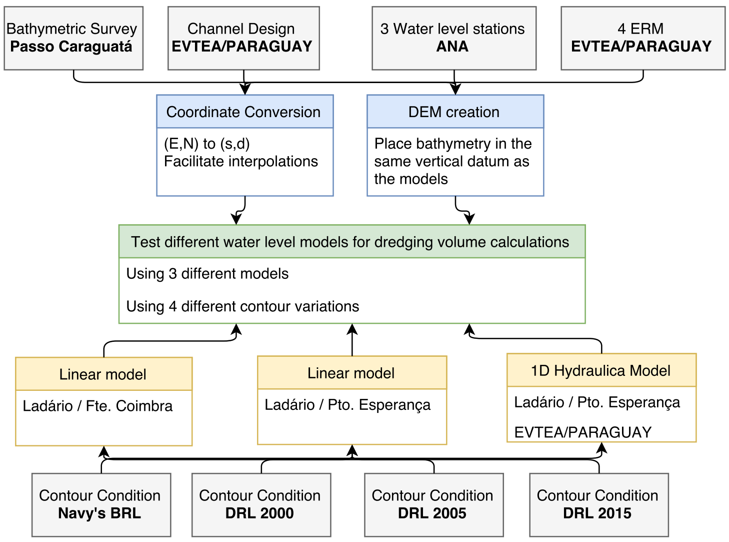


FIGURE 24 – Strategy for dredging volume definitions and error assessment

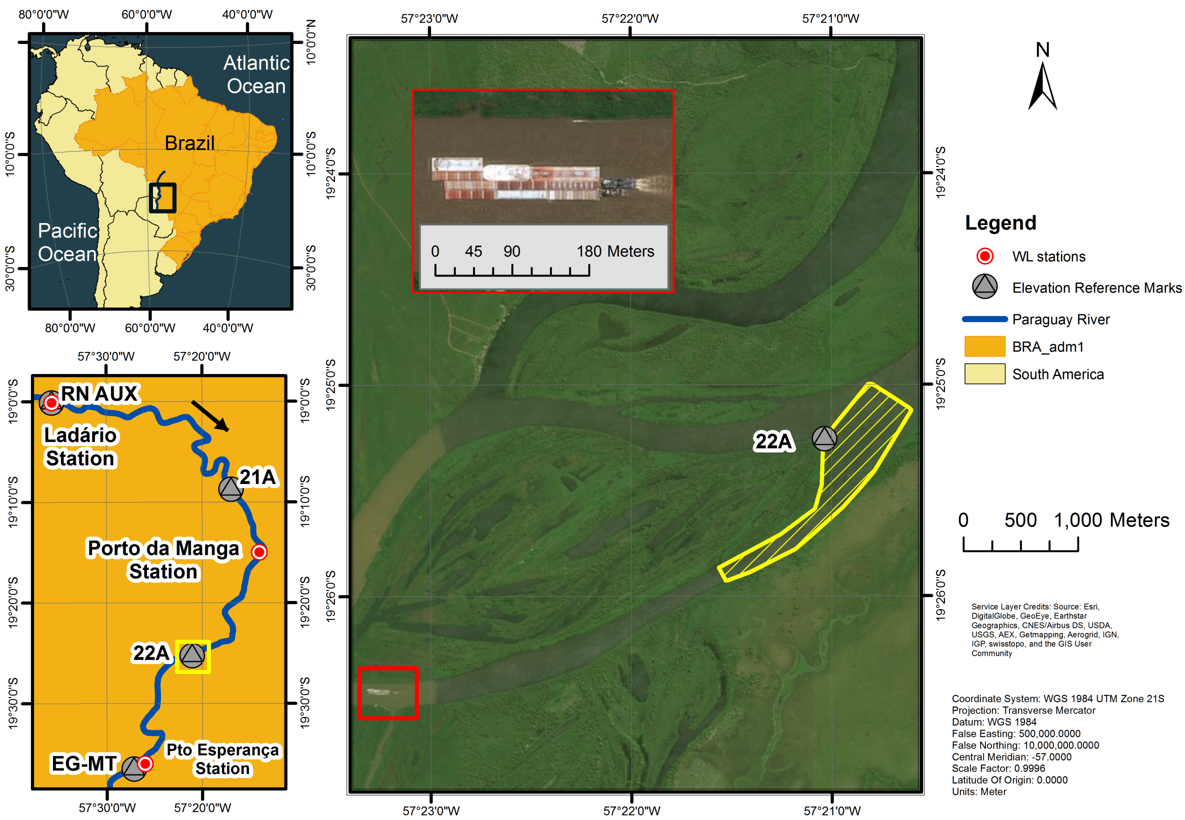


FIGURE 25 – Location of the focus area, detailed location of 'passo' Caraguatá. For a sense of scale a Vessel that navigates the stretch in a 3x3 +1 configuration is highlighted.

3.6.1 Focus Area

The focus stretch is located at the extreme west of Mato Grosso do Sul state, between the port of Ladário at the Corumbá district and the Porto Esperança district. It has approximately 126 km and 4 junctions with smaller rivers (Paraguay-Mirim, Miranda, Taquari and Negro). Two loci in this stretch that have historically presented difficulties for navigation due to shallow waters and severe turning radius. They are also referenced in the Navy's 'Notice to Seafarers' (BRASIL, 2016b) and are known as passo[2] Caraguatá and passo of Jacaré. Along this stretch there are 4 elevation reference marks that were implemented and served as calibration scenario for the waterway model of UFPR/ITTI. (2015b). There are also three water level stations along the stretch, Ladário at the upstream limit, Porto da Manga and Porto Esperança at the downstream end of the stretch (Table 8).

[2] Portuguese word, in inland waterway context, means a 'critical stretch for navigation'

ID	Location	E(m)	N (m)	WL (m)	Date	s (m)	WL - BRL (m)
RN AUX LAD.	Ladário (naval base)	437401.20	7898820.51	84.91	07/11/2014	0	84.75
21A	Tira Catinga Island	470315.00	7883059.00	83.53	10/01/2015	58,616	83.35
22A	Caraguatá Island	463196.00	7852557.00	82.40	11/01/2015	99,215	82.21
EG-MT	Porto Esperança	452567.00	7831768.00	81.58	11/01/2015	129,394	81.38

TABLE 7 – Elevation reference marks of the focus stretch (WGS 84 - UTM - 21S). Source: (UFPR/ITTI., 2015b)

Station Name	ID	N(m)	E(m)	Drainage Area (km^2)	Current BRL (m)	Period of Calculation	Missing Data
Ladário	66825000	7899070.80	438606.89	253,000	2.02	1900-01 2015-08	0.06%
Porto Manga	66895000	7871493.77	475482.05	327,000	4.41	1969-05 2015-09	17.71%
Pto Esperança	66960008	7832724.07	454560.83	371,000	1.77	1963-12 2015-09	4.47%

TABLE 8 – Water level stations of the focus stretch (WGS 84 - UTM - 21S). Source: (BRASIL, 2015)

3.6.2 Bathymetric Survey and Channel Design

The bathymetric survey of AHIPAR (2015) was used for the volumetric calculations comparisons. The depths were surveyed in November 28th, 2014 and covered an area of 59 ha. The readings on the stretches water levels stations were: Ladário 242.0 cm ; Porto Manga 475.0 cm; Porto Esperança 247.5 cm. The reduction level used (0.93 m) was a fixed one for all surveys from Ladário to Porto Esperança. This correction was undue to obtain the original surveyed values, they were further referenced to the Navy BRL in order to maintain the same vertical datum as the Elevation Reference Marks. The altitude of the water level in the survey site on the day of surface was not provided. Figure 26 presents the bathymetric surveyed points of the passo Caraguatá.

In respect to the Channel Design, the option for keeping the same design to all water level scenarios was taken. The intention was keep a low degree of freedom for the dredging volumetric estimation experiment. This allowed to focus exclusively on the results varying the water level model and the contour parameters of the models. A topic further argued in the discussion section (Chapter 5), contemplates the possibility of assuming the channel's design related degrees of freedom. Nonetheless, such approach is not the focus of this dissertation.

The Channel Design considered for the dredging volume estimations is the navigation channel designed for the Executive Project of Paraguay Waterway Feasibility

Study(UFPR/ITTI., 2015a). It was designed respecting the curvature limitations suggested in (PIANC, 1997) and aiming at minimal dredging volumes. Further explanations are available in (RATTON et al., 2016). The Figure 27 presents the navigation channel area within the focused area.

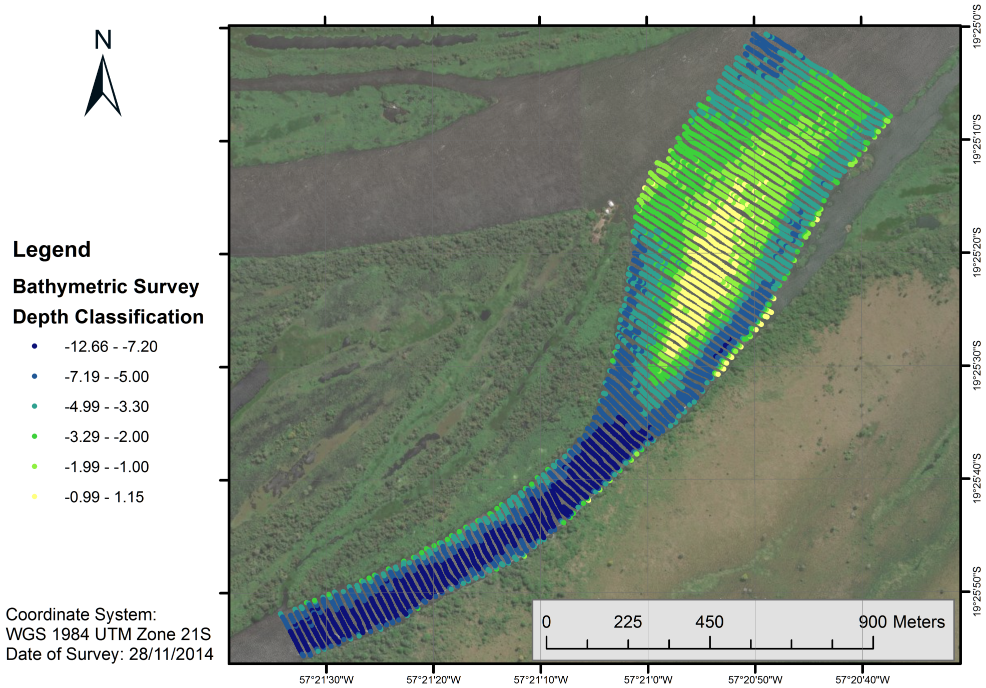


FIGURE 26 – Bathymetric survey of passo Caraguatá. Source: (AHIPAR, 2015)

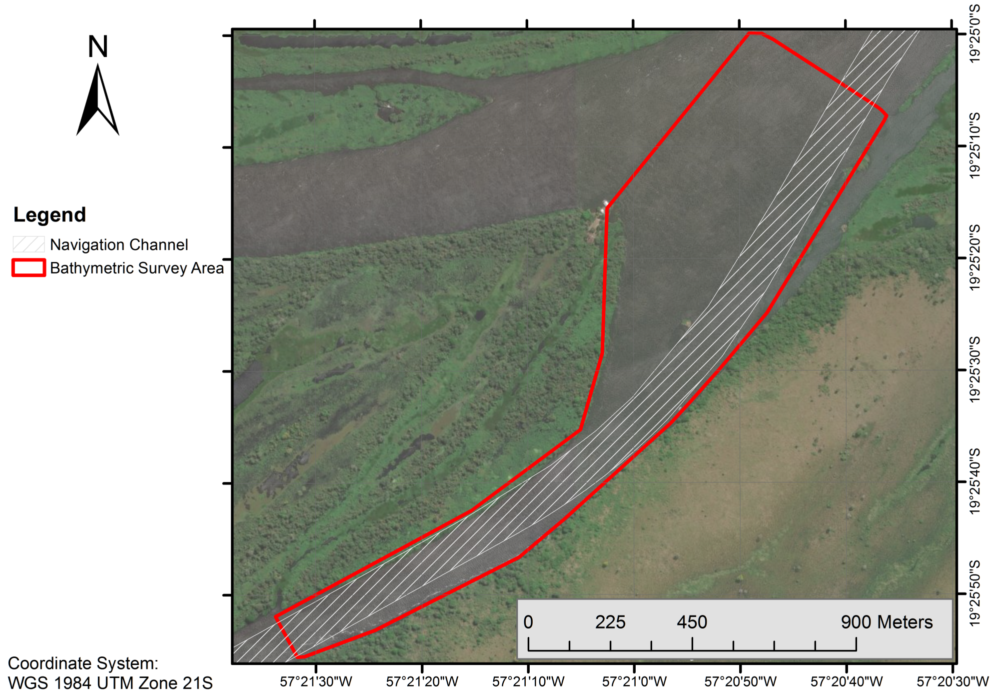


FIGURE 27 – Channel Design at passo Caraguatá. Source: (UFPR/ITTI., 2015b)

3.6.3 Bathymetric Coordinate System Transformation

To assist the definition of longitudinal characteristics of a river (*e.g.* elevation) as a function of its longitudinal coordinate (middle of the channel), the planimetric coordinates of a Cartesian system (E, N) can be transformed to a coordinate system oriented to the flow direction (s, d), as in Figure 28, where s is the longitudinal coordinate along the navigation channel and d is the transverse distance of the measured points of the channel line. To accomplish that, a method similar to the linearizion mentioned in Merwade, Maidment e Hodges (2005) was developed and implemented using a longitudinal trace of the river as reference. The transformation is described by the following equation:

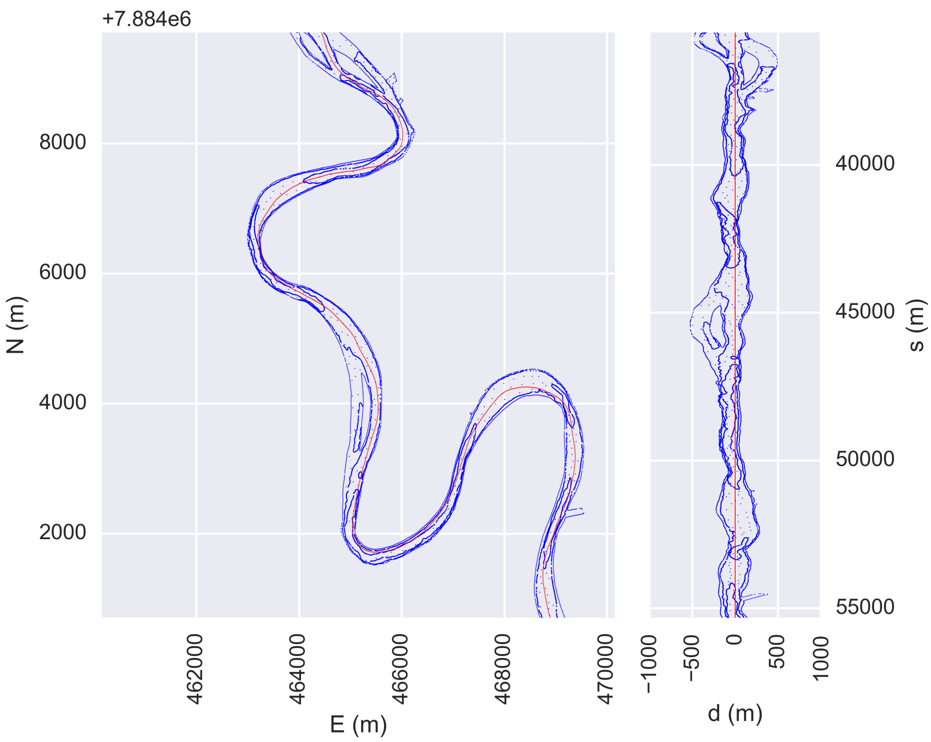


FIGURE 28 – Coordinate transformation (E,N) to (s,d).

$$[s, d] = [S_b + (|\overline{PB}|cos(\alpha)), \Psi(|\overline{PB}|sen(\alpha))], \tag{3.11}$$

where:

α is the angle between ($|\overline{Ap}|$) e ($|\overline{AB}|$).

S_B is the longitudinal distance of B from the arbitrary beginning;

B is the planimetric coordinate $[E_B, N_B]$ of the upstream point;

C is the planimetric coordinate $[E_C, N_C]$ of the downstream point;

P is the planimetric coordinate $[E_P, N_P]$ of the surveyed point;

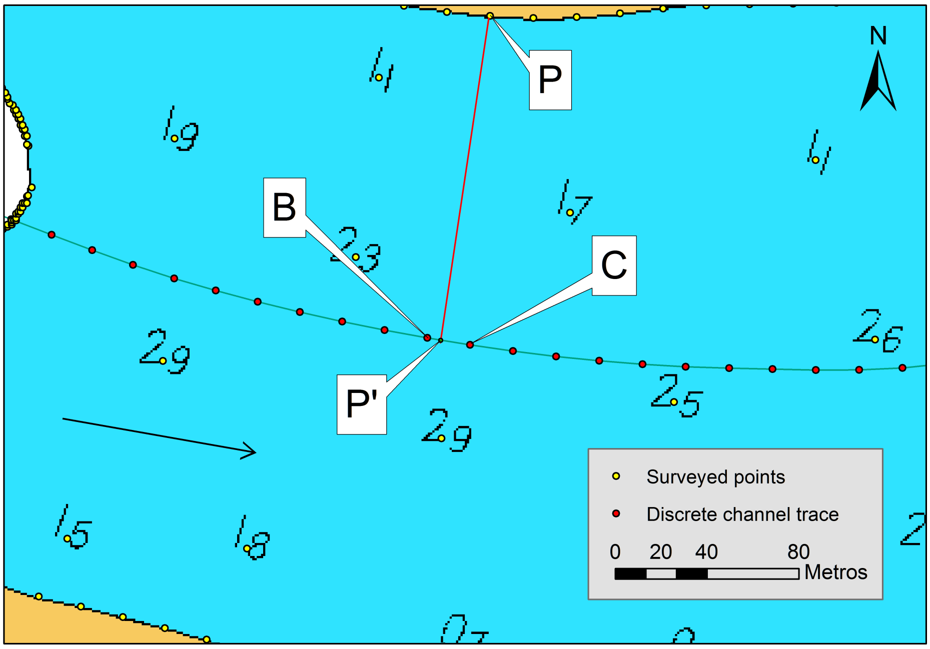


FIGURE 29 – Concept and main elements on the application of the projection transformation.

$$\psi = \frac{\begin{vmatrix} |\overline{CB_x}| & |\overline{CB_y}| \\ |\overline{PB_x}| & |\overline{PB_y}| \end{vmatrix}}{\left\| \begin{matrix} |\overline{CB_x}| & |\overline{CB_y}| \\ |\overline{PB_x}| & |\overline{PB_y}| \end{matrix} \right\|}$$ was developed so that it equals -1 for points at the right side and 1 to points at the left side of the longitudinal axis of the channel. In Figure 29 is possible to visualize the basic concept of the method.

The center idea behind the algorithm implementation is to find the pair $[B, C]$ that corresponds to each P. A possible solution is to verify for which points the Equation 3.12 is respected for a given point P.

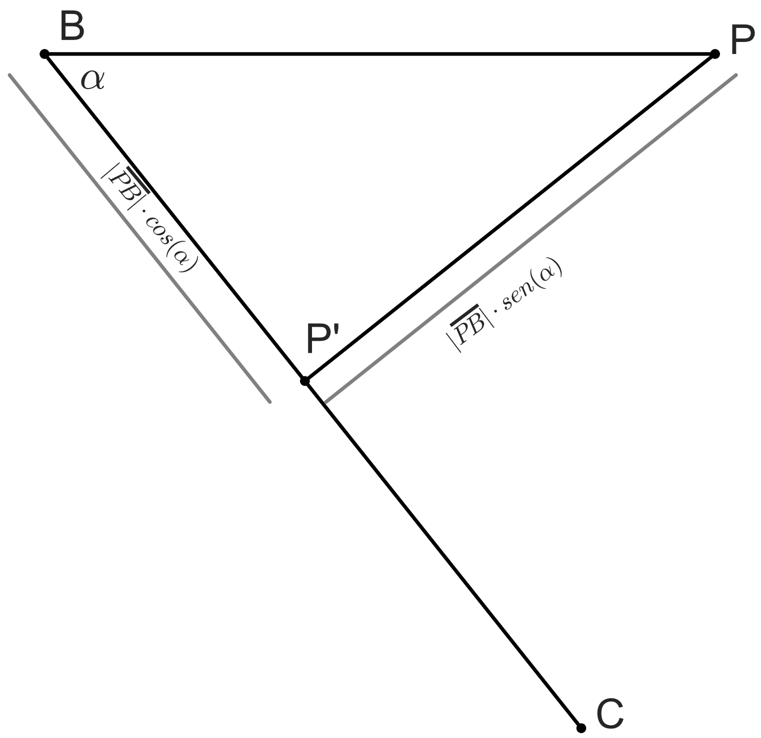


FIGURE 30 – Scheme for the longitudinal coordinate (s) attribution.

$$|\overline{CB}| = |\overline{P'B}| + |\overline{CP'}|, \tag{3.12}$$

where

$$P' = B + |\overline{PB}| \cdot cos(\alpha) \cdot \frac{\overline{CB}}{|\overline{CB}|} \tag{3.13}$$

In the case of a point P having multiple pairs (B, C), the nearest one is probably the correct pair. However, a verifying parameter must be set in accordance with the characteristics of the river in question in case the closest pair found is not in fact the correct pair. This can happen due to an effect called shading, where the point P becomes 'invisible' to the nearest pairs (B, C) (Figure 31). In order to minimize the effects of shading, one must smooth the channel's center trace maintaining the same input and output tangent for each node, thus smoothing the path and avoiding sharp curves. The smoothing algorithm implemented was the Polynomial Approximation with Exponential Kernel (PAEK) and the method was implemented in Python scripting language (Appendix D.3).

This method is important to correct the detailed bathymetric data as a function of the river mileage, and to assign the altitudes to the counted and margin points along the channel. For this, a zero was chosen arbitrarily and a reference axis $[s, d = 0]$ defined.

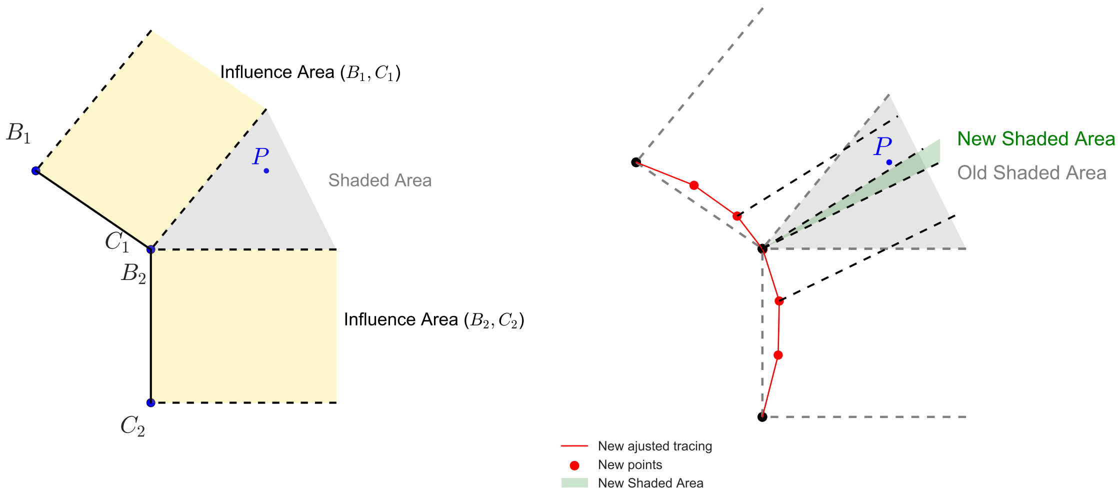


FIGURE 31 – Example of shadow effect (left) and correction by curve adjustment (right). Notice that the shadows don't vanish, they are just diluted.

The chosen zero was the section in front of the navy base at Caceres-MS, and the chosen route was a central channel tracing defined from the midpoint between two margins.

3.6.4 Digital Elevation Model

In this section the method to create the Digital Elevation Model (DEM) of ‘passo Caraguatá’ is demonstrated. The purpose of this DEM is to put the riverbed information in the same vertical datum as the water level models output. This allows the calculations of the new depths that are used in dredging volume estimations.

A problem faced here is the water level at the Elevation Reference Marks were not taken in the days of survey. The only information available is the water level at the stations in the stretch (Ladário, Porto Manga and Porto Esperança). As the elevation at the marks and stations are known, it is possible to interpolate the altitudes between this points to attribute them to the bathymetric survey points. This creates a entangled dependence, as one needs the water level model to define the altitudes of the depths points, but also needs the elevation model to run a hydrodynamic model. The solution is to attribute the altitude values to the depth records linearly. Given that the method is aimed towards relative volumetric values of dredging volumes given different water level models, the uncertainties added by this approximation will be ignored as they are included in every scenario. Nonetheless, had the water level orthometric altitude been propagated from a local Elevation Reference Mark this problem would not exist.

The same method applied by UFPR/ITTI. (2015b) in the creation o the DEM for the hydrodynamic model was applied. Once all the values $[s, d]$ are assigned to their respective quoted points, the altitudes can be interpolated linearly along the channel as a function of s. The calculated altitudes should be in the same reference as the depth values, for example if the quoted depths are referenced to the Bathymetric Reference Level (BRL) of the stations along the Waterway, the altitudes should also be compatible with those levels. Each point has altitude defined according to Equation 3.14.

$$H_p = H(s_p) - Y \tag{3.14}$$

where H_p is the altitude of a point; $H(s_p)$ is the water level altitude at s meters from the origin; and Y is the depth at the given point.

The altitude at each point is calculated by interpolating the altitude values (H) between the upstream Elevation Reference Mark (ERM) i and the downstream ERM $i+1$. The generalization of this reasoning is described by the following equation:

$$H(s) = \frac{(H_{i+1} - H_i)}{(S_{i+1} - S_i)} s + H_i - \frac{(H_{i+1} - H_i)}{(S_{i+1} - S_i)} S_i, \tag{3.15}$$

where S is the longitudinal distance between the RRNN and the start point. In the operations performed above it’s important to keep the ‘[E, N]’ data of each point, because at the end they are associated with the orthometric values.

Figure 32 shows the same section in planimetric coordinates (E, N) and coordinates in the direction of flow (s, d) after receiving altitude values (H, d). Note that only the values referring to the margins are represented since the values of bathymetry have their altitude subtracted from their depth.

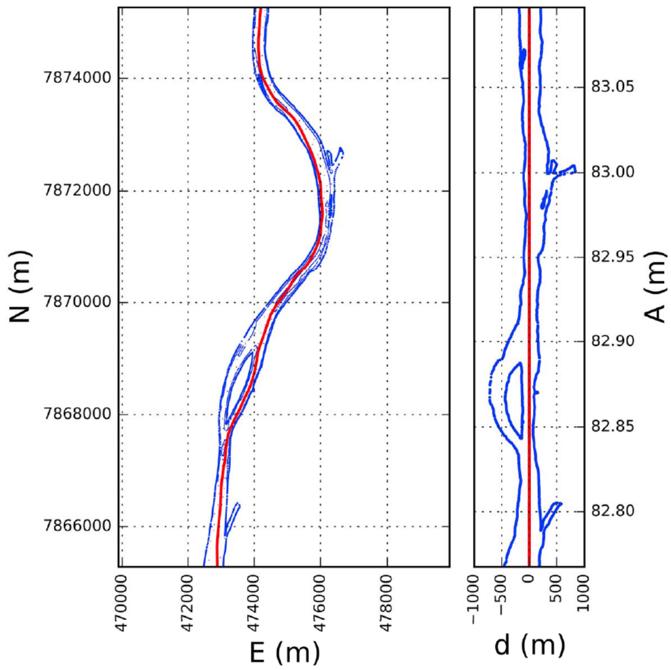


FIGURE 32 – Coordinate transformation from (E,N) to (a,d).

3.6.5 Hydrodynamic model

The hydrodynamic 1D model applied for the dreading volume comparisons is the same used in (UFPR/ITTI., 2015b) and (CORREIA, 2016). The original model was separated into subsections. Only the subsection from Ladário to Porto Esperança was taken for this analysis. As contour conditions the model in this stretch was applied for the water level of the official Navy's BRL at the downstream Cross-Section (Porto Esperança) and the discharge equivalent to the official Navy's BRL at the upstream cross-section (Ladário).

The model was developed with the HEC-RAS 4.1 software (USACE, 2010). It was calibrated with discharge and average flow velocity from ADCP field surveys, and with the water levels altitude determined along the Elevation Reference Marks (ERM) implemented.

The cross-sections and depth profiles were defined on top of the vectorized Navy's Nautical Charts with the assistance of the HEC-GeoRAS software (USACE, 2009).The digital elevation model was created with the Elevation Reference Marks (ERM) implemented and described in (UFPR/ITTI., 2015b). The complete procedural steps are described in (GUARNERI et al., 2015).

3.6.6 Linear Model

The linear interpolation method consists of the same interpolation applied in the Subsection 3.6.4. However, replacing in Equation 3.15 the reference marks at the BRL by a linear interpolation along the s coordinate of the upstream and downstream station's stage variations of the scenario wanted from the official BRLs.

That is, given a river stretch limited by two water level stations, with known BRL altitude; Given a set of BRL water level altitude reference marks along this stretch; The new altitude H of this references is given by:

$$H = H_{BRL} + \Delta V(s) \tag{3.16}$$

where V is the variation interpolation between stations, given by:

$$\Delta V(s) = \frac{(\Delta V_{ds} - \Delta V_{us})}{(S_{ds} - S_{us})} s + \Delta V_{us} - \frac{(\Delta V_{ds} - \Delta V_{us})}{(S_{ds} - S_{us})} S_{us} \tag{3.17}$$

where ds and us subscripts mean respectively upstream and downstream.

The resultant water level model is the linear interpolation along the rivers s coordinate of the updated heights of the water level altitude respective to each altitude reference mark along the stretch.

3.6.7 Depth Correction and Dredging Volume Estimation

The methods used to correct the depths according to each water level model and the subsequent dredging volume estimations are presented in this section. A different method is used to each of the two models.

The linear model uses the method described in Subsection 3.6.3 to attributed the water level height of Subsection 3.6.6 to each point value of the bathymetric survey. The new depths are given by the subtraction of each points DEM depth value (Subsection 3.6.4) from the scenario's water level height. The kriging interpolation is used to transform the resulting depth points into a continuous raster file.

For the hydraulic model, since the output of the HEC-RAS model can be a raster file of the water level altitude, the calculations were performed using raster operations. A kriging interpolation was implemented to create the surface that originated the bathymetry's DEM raster file. The new depths are the result of the DEM's raster subtraction from the water level raster.

All of the kriging interpolations were performed using a cell size of 3 meters, a circular semivariogram model and a search radius of 50 meters. The parameters were chosen by manual trial and error aiming at the lowest errors using a 10% portion of randomly chosen samples. To estimate the dredging volumes for both models, all the cells with depth superior to the dredging depth (3.3 m) had its excesses sum integrated. The calculations were developed and implemented through a Python module called `drl_volume_scenario.py` (Appendix D.4) which uses several functions `ArcPy` (ESRI, 2014). The kringing interpolation was implemented with the 3D Analyst Raster Interpolation toolset function `Kriging_3d` and the volume estimation with the 3D Analyst Functional Surface toolset function `SurfaceVolume_3d`, references to both functions are available in Esri (2017). Figure 33 illustrates the `SurfaceVolume_3d` function.

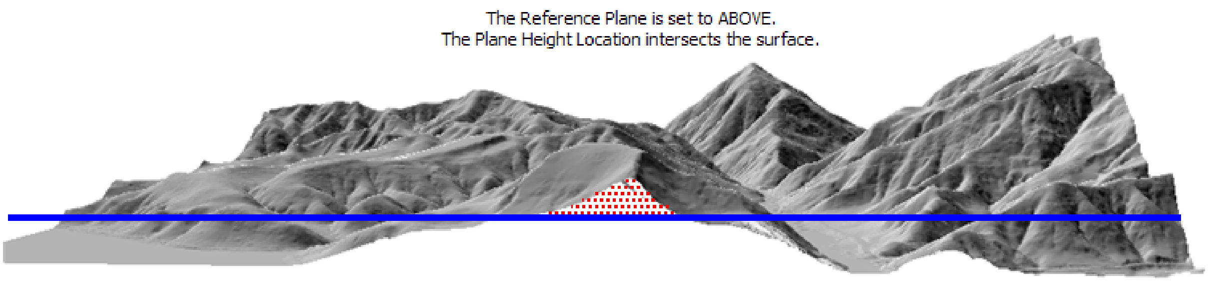


FIGURE 33 – `SurfaceVolume_3d` - Calculates the area and volume of the region between a surface and a reference plane. Source: Esri (2017)

The implementations of the `drl_volume_scenario.py` module were performed through Python scrips and two examples of it's application are available at Appendix C.5 and C.6 for the linear models and the hydrodynamic models respectively.

3.6.8 Volumetric Calculations Experiment

To assess the dredging volume variations associated with different water level models and DRL yearly variations an experiment was designed . In this subsection all the supporting elements of this experiment are described.

In total 12 scenarios were tested. The 12 scenarios were divided into tree groups: Group 1 aggregate the scenarios that use linear interpolation between Ladário and Forte Coimbra as the water level model; Group 2 aggregate the scenarios that use linear interpolation between Ladário and Porto Esperança as the water level model; Group 3 the scenarios that use the hydrodynamic model as water level model. All the Groups have the same subgroups: (1) the simulation with the official navy BRL values; (2) the simulation with the real DRL (10th percentile) of the year 2000; (3) the simulation with the real DRL (10th percentile) of the year 2005; (4) the simulation with the real DRL (10th percentile) of the year 2015. The years were chosen according to its values looking for a wide spectrum of DRLs.

Scenario	Ladário (cm)	Porto Esperança (cm)	Forte Coimbra (cm)
BRL	202	177	134
2000	124	67	15
2005	168	117	75
2015	220	184	151

TABLE 9 – Real DRL (10th percentile) per station and scenario.

Scenario	Ladário's Stage (cm)	Ladários Discharge (m^3/s)	Porto da Manga Discharge (m^3/s)
BRL	202	1106	1340
2000	124	873	1039
2005	168	992	1195
2015	220	1179	1452

TABLE 10 – Reference level discharge per water level station and scenario.

Groups 1 and 2 use the scenarios' stage variation from the BRLs at the downstream and upstream stations to correct the altitude of the elevation reference marks along the stretch. Table 9 presents the reference level for each year and the official BRL. Group 3 uses as input, at the upstream section (Ladário), the discharge related to the scenario's stage, and at the downstream station (Porto Esperança), the scenario's stage altitude. There is also a control point for the modeled scenarios at Porto da Manga where, due to the inflow of a tributary, the discharge left to complete the discharge at the downstream section equivalent to the scenarios reference level is added. The values used as contour conditions for the modeled scenarios are presented in Table 10.

4 RESULTS

In this section the results of the applied methods are presented. Descriptive analysis are presented when they are advantageous for understanding further steps. A deep more focused analysis is reserved for the discussion section.

First we present the results of the wavelet analysis and the characteristics that led to the development of hypothesis concerning the DRL forecast models. In the sequence the results from the DRL error quantification analysis are presented and also the key results that led to the formulation of further hypothesis concerning the forecast models. Following, the results from the autocorrelation analysis helped provide some analytic validation over the previous analysis and to define the structure of the forecast models. The results of the forecast models are then presented demonstrating the errors associated with each scenario.

At last, the results for the volumetric calculations with water level model variations and a DRL sensitivity analysis is presented along with the dredging volume values for each method, the profile plots and tables with relative percentages.

4.1 Water level wavelet analysis for the DRL forecast

The initial approach towards solving the introduction chapter's reviewed issues was to assess the overall behavior of the time series. To accomplish that, a wavelet analysis was performed in order to identify patters and periodical influences on the Ladário stage series from 1900 to 2003. In Figure 34, three graphs are presented: First the Ladários stage time series used for the analyses (1900-2003). Second the global wavelet power spectrum, that is, the averaged normalized harmonic influences of the wavelet power spectrum. Third the wavelet power spectrum or the normalized harmonic influences on each discrete time step of the analysis (days). Similar results to (NORDEMANN, 1998) were obtained. Other than the annual cycle, influence with periods of 3, 5, 8, 15 and 27 years were find.

As maintenance dredging focus on temporary dredging solutions to depth problems, the expected expiration date of a dredging endeavor due to sedimentation is short, requiring almost yearly interventions in some stretches. Within this scenario, this results are important to the correct definition of a local DRL. Harmonic influences present hints about the DRL calculation procedures to optimal maintenance dredging operations. From that, the following hypothesis can be formulated:

1. A DRL calculated with a base period to long, including several of these oscillations, may increase uncertainty and account for a period that no longer represents the

river current situation.

2. A constant DRL update could be used for proper depth maintenance performance.
3. An extended non-updated period can cause a number of recurring errors due to minor periodic variations.
4. Small base periods may provide more accurate results as it remains within the same harmonic influences.

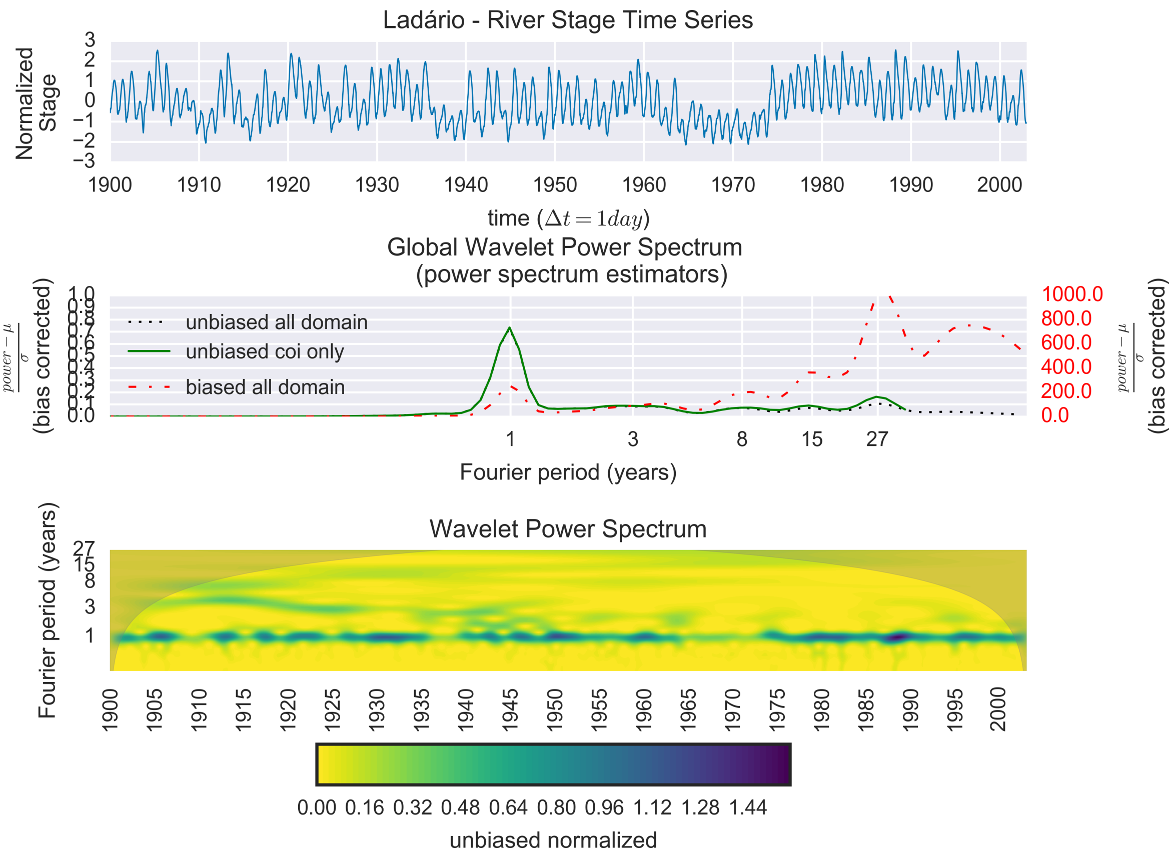


FIGURE 34 – Wavelet analysis results from the stage series Ladário station (1900-2003). From top to bottom: (a) The Ladário's stage time series used for the wavelet analysis. (b) The global wavelet power spectrum. (c) The local wavelet power spectrum using the Morlet wavelet, normalized by $\frac{x-\mu}{\sigma}$. The left axis is the Fourier period (in yr). The bottom axis is time (yr). The colors are the normalized variances. The shaded regions on either end indicate the "cone of influence," where edge effects become important.

4.2 Quantification of error related to the current DRL calculation method

The hypothesis raised after the results of the previous section require some testing. To accomplish that, the method described in Section 3.3 was implemented with x_N ranging from 1 to 20 years and x'_N ranging from 1 to 5 years. The Ladario's yearly 10th percentile series from 1920 to 2010 where used (Figure 35).

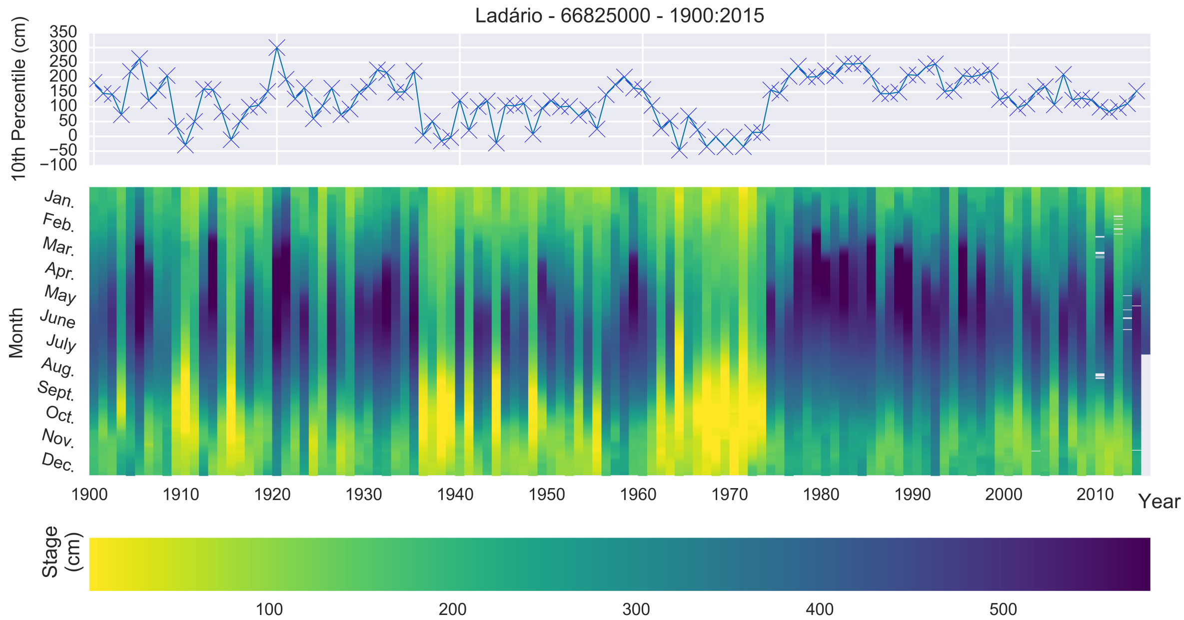


FIGURE 35 – At the top the calculated annual 10th percentiles that are used in the method. At the bottom the stage values presented in colors by year at the horizontal axis and months at the vertical axis.

The Figure 36 presents the error results of the testing with $x'_N = 1$. The colored matrix present the error between defined DRL and real 10th percentile varying by year (x-axis) and amount of years in the base period (y-axis). The graph on the top presents the base period that resulted in the least amount of error (y-axis) for each year (x-axis). The graph on the right presents the box plot of the errors for the whole period by base period size. The resulting graphs for the other values of x'_N are presented in Appendix A.1.

The resulting errors calculated with x'_N varying from 1 to 5 can be seen in Figure 11a. It's shown that for the Ladário's station, throughout a period of 90 years, fewer mistakes would have been made by assuming a 1 year period for the calculation of the DRL – with up to 30 cm less error. It also shows that, not updating the DRL every year could increase the error significantly in this case an average of 15 cm. Also that not updating the DRL for 5 years and using $x_N = 1$ results in smaller average error than using $x_N = 20$ and updating every year. Figure 11b presents the averaged errors encountered.

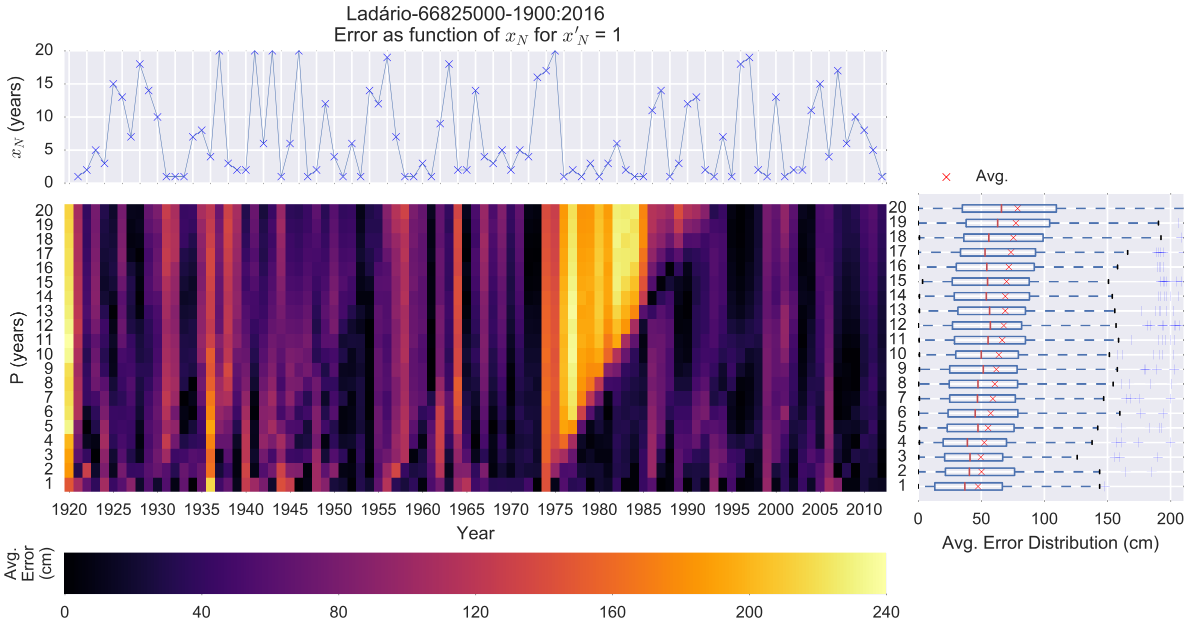
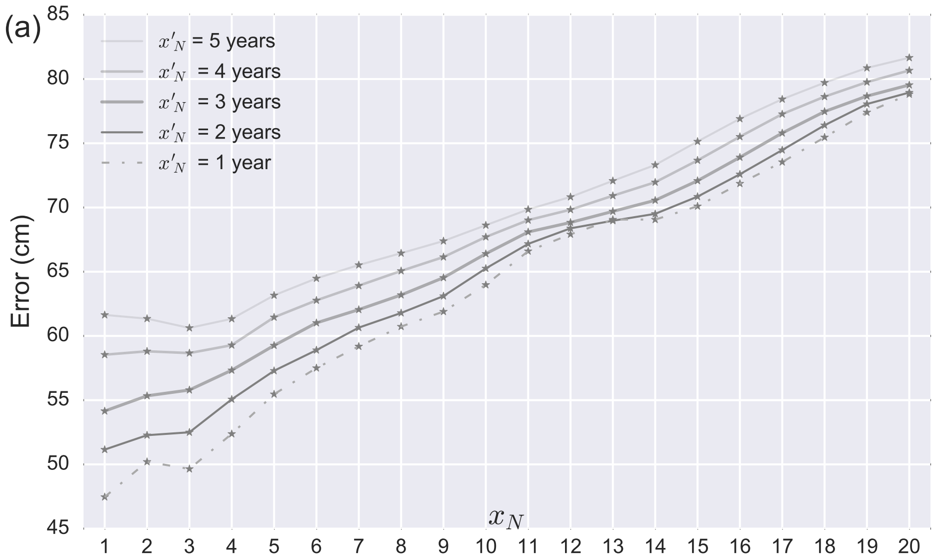


FIGURE 36 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 1$.



(b)

x'_N \ x_N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	47	50	49	52	55	57	59	60	61	63	66	67	69	69	70	71	73	75	77	78
2	51	52	52	55	57	58	60	61	63	65	67	68	68	69	70	72	74	76	78	78
3	54	55	55	57	59	61	62	63	64	66	68	68	69	70	72	73	75	77	78	79
4	58	58	58	59	61	62	63	65	66	67	69	69	70	71	73	75	77	78	79	80
5	61	61	60	61	63	64	65	66	67	68	69	70	72	73	75	76	78	79	80	81

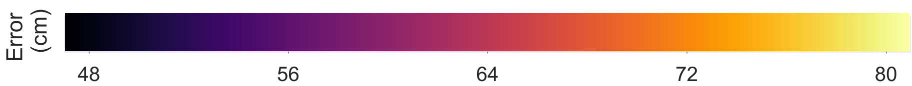


TABLE 11 – Resulting average errors from the current DRL method analysis. (a) Graph with the resulting errors calculated with x'_N varying from 1 to 5 (b) Table with the resulting errors calculated with x'_N varying from 1 to 5 showing in colors the variation of magnitudes.

4.3 Autocorrelation Analysis of annual stage 10th percentiles

To test the extent of past years stage influence in the current year stage, an Autocorrelation analysis was performed over the annual 10th percentile series. The method of this analysis is described in Section 3.4 and was applied to the data presented in Figure 35.

The Figure 37 presents the autocorrelation analysis results. It shows that, a given year in the data set analyzed has some correlation with the previous one, two and three years but it fades quickly after that (zero has always a perfect autocorrelation). This means that using the percentile of more than 3 year would only add random data to the DRL calculations leading to an increase of prediction error. This provides an important result that support and explain the findings of Appendix A.1. It also strongly suggest that stochastic forecast methods don't need to 'look too far to the past to predict what is going to happen in the following year. And, as far as autoregressive models go, and the characteristics of the time series under analysis, the past year may contain alone the most useful informations towards making a good prediction.

FIGURE 37 – Autocorrelation and partial autocorrelation plots for the annual 10th percentile – Ladário-MS.

4.4 DRL Forecasting Methods for Depth Assessment and Dredging Volume Definitions

After the results from the previous sections, a decision was made towards designing forecast methods. All of the methods were based on the 10th percentile or monthly values of the year prior to the one being forecasted. They were chosen to cover a wide range of endogenous approaches. The aims were to obtain a better comprehension of the variations of errors resulted from different training periods and which methods and datasets would provide the better results.

In this section the results from the methods of Section 3.5 are presented. Initially, an assessment of the predictions are presented for each method and test period. In the sequence the distribution of error for each method and tested period is presented followed by the description of the obtained results.

Method A - 10th percentile of 20 year.

Method 'A' results (Figure 38) shows that the method is very conservative as it takes the trend from several years and reproduces it for the year aim of forecast. In that sense it lacks flexibility as it cannot adapt to changes in trend. So, it works better when yearly variations are not significant and stay around the average of the past values. For the periods forecasted the following inferences can be withdrawn:

a Stayed around the average, have a slight variability and presented the best results for the method;

b Mostly overestimated predictions. Results would lead to higher dredging volumes, much larger than necessary, costing more and causing unnecessary impact on the environment. Predictions followed the trend of the drought period of the 1960's.

c Mostly underestimated predictions. Results would lead to lower dredging volumes, lower than necessary, causing extended interruptions in navigation and/or suboptimal loading.

d Mostly overestimated predictions. Results would lead to higher dredging volumes, much larger than necessary, costing more and causing unnecessary impact on the environment.

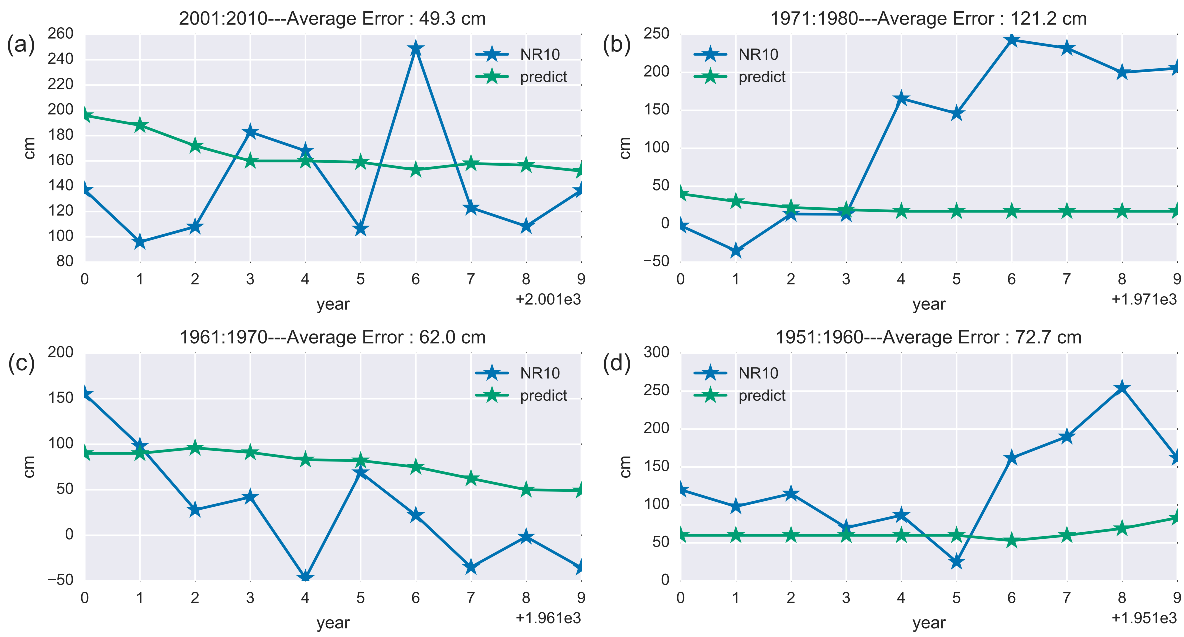


FIGURE 38 – Method A prediction results.

Method B - 10th percentile of 1 year.

Method 'B' results (Figure 39) shows that it's more flexible than Method 'A' as it demonstrates a tendency to adapt to changes in trend better. The fact that it takes the most recent available annual water level 10th percentile as the forecast for the next year is likely the reason for that. So, it works better when the values are stable around an average or have a slight change in trend. Sudden changes on trend result in poor predictions. For the periods forecast the following inferences can be withdrawn:

a Presented better results for the periods with small variability and sudden changes in trend. The overall performance was compromised due to the poor results produced by the sudden variations of 2007 and 2008.

b Followed the rising trend with 1 year delay. When yearly variations were low the method performed well. The high increase of 1975 resulted in a bigger error. Nonetheless, produced significantly better results than Method 'A' for the period.

c This period has a a non-stationary behavior with decreasing forecasting target. The method performed better when the series had a stable decrease in the begging of the decade and at the end when had stable values. The sudden changes of 1965 and 1966 generated bigger errors.

d Performed better in the beginning and at the end of the decade when target values trend were stable. Did not perform well at the sudden variations o 1957.

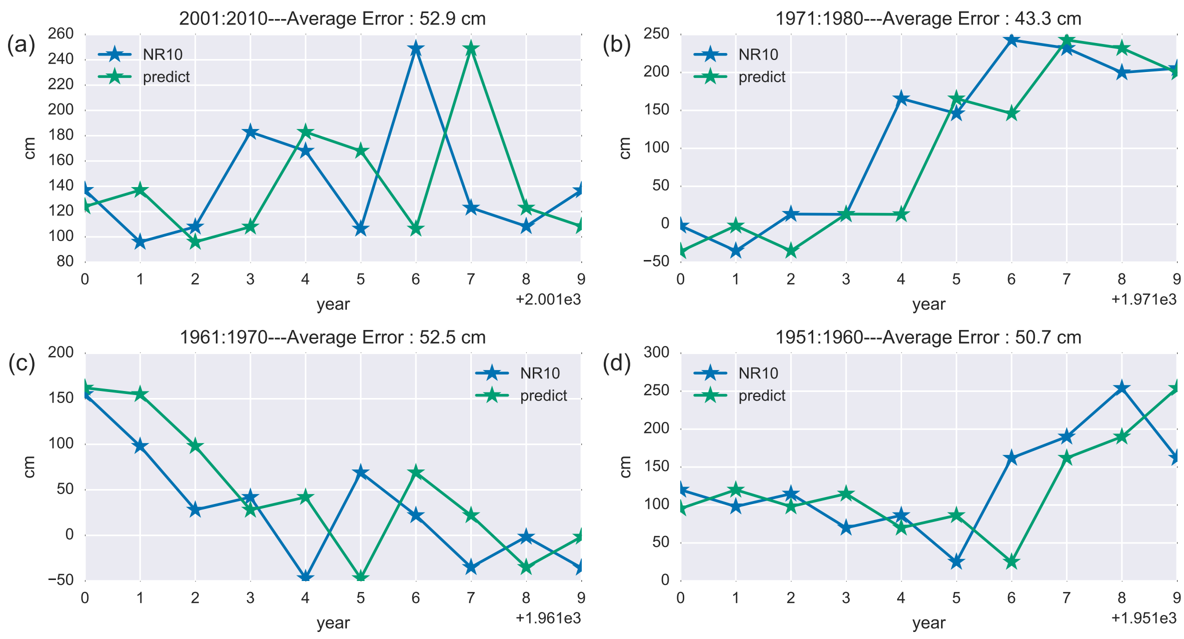


FIGURE 39 – Method 'B' prediction results.

Method C - AR(1) model of the annual 10th percentile.

Method 'C' results (Figure 39) shows that, as 'B', it's more flexible than Method 'A' as it shows a tendency to adapt to changes in trend as it takes a regression of the most recent available annual water level 10th percentile as the forecast for the next year. So, it works better when the values are stable around an average or have a slight change in trend. Sudden changes on trend result in poor predictions. For the periods forecast the following inferences can be withdrawn:

a Presented a delayed response to variations with a dumping effect. In that sense, performed similarly to the method B but with better performance over abrupt changes.

b The same shifted dumping effect is observed. However, it's observable an underestimation for low target values and an overestimation for high target values.

c Underestimated most of the predictions. Possibly due to the inexistent descending patter in the training set, the method couldn't handle it properly.

d Similar to method B, the beginning of this decade showed better prediction performance. It presented the same dumped delay effect as periods 'a' and 'b' and the same overestimation of higher target values.

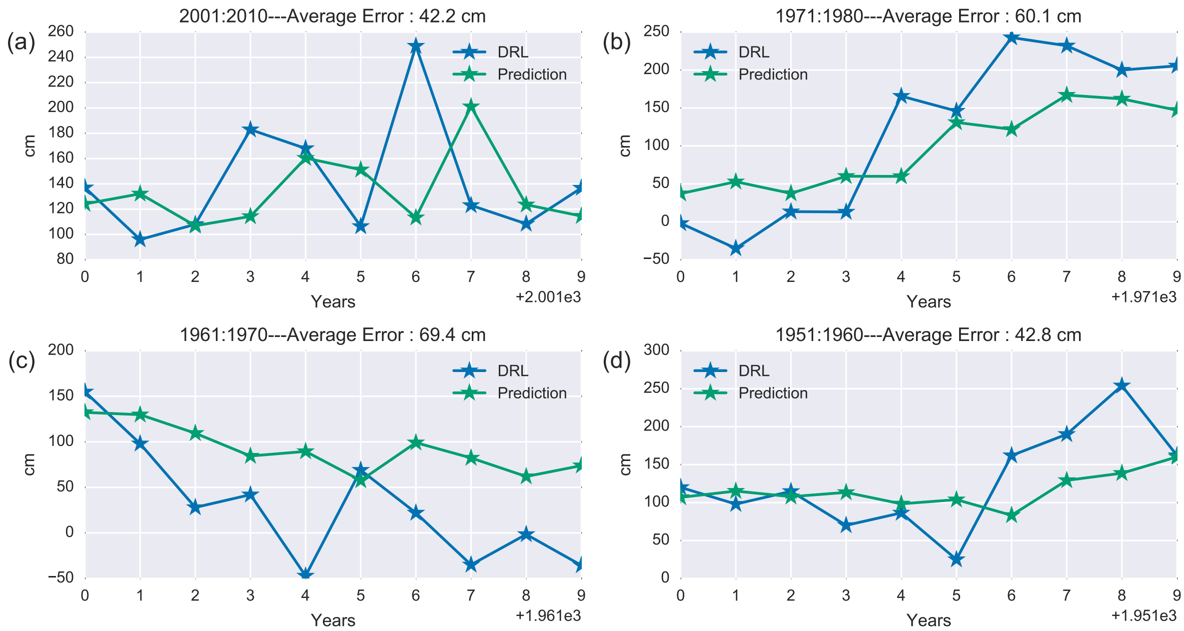


FIGURE 40 – Method 'C' prediction results.

Method D - Multi-variable Linear Regression Model with monthly maximum stage.

Method 'D' results (Figure 41) shows that it's much more flexible than Method 'A'. Its ability to adapt to changes becomes visible. The fact the method breaks the behavior of the year prior to the forecast in monthly values, add new information to the methods training that is now able to generalize much more accurately the predictions. The method showed better performance for the lower water period (c) with an average prediction error of 24 cm. In general, sudden changes in trend do not impact the predictions results. Overall the method seem to struggle more when the trend is stable as an intense variability between years is a strong characteristics of the overall series. For the periods forecasted the following inferences can be withdrawn:

a Followed the target reasonably well for the first half of the decade. The second half produced some inconsistent results.

b Had an overall good performance for this period. The stable target values of the end of the decade presented the higher errors.

c Performed well for this period also. During the first half of the decade it got 3 out 5 predictions very close to the targets. The second half had even better results with 4 out of 5 predictions very close to the target.

d Achieved reasonably good results for 6 out of 10 predictions (avg. 30 cm errors). For the other 4 years it achieved higher errors (avg. 50cm). Performed well for the abrupt change of 1956-1957.

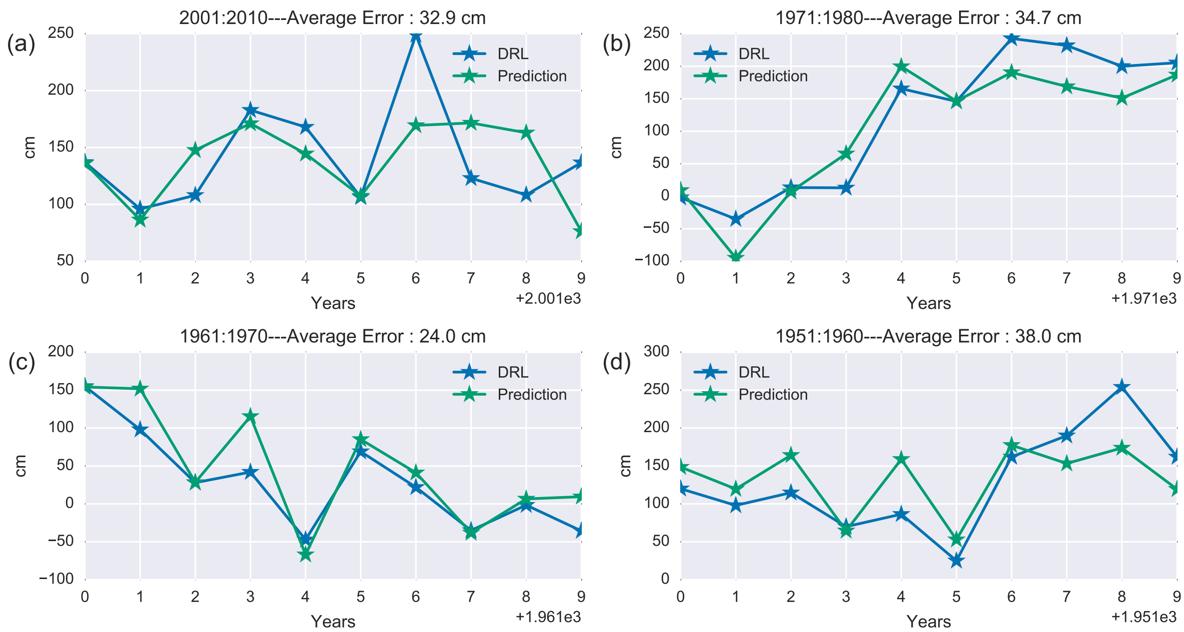


FIGURE 41 – Method 'D' prediction results.

Method E - Artificial Neural Network Model over monthly maximum stage.

Method ‘E’ results (Figure 42) showed similar results to Method ‘D’ and a much more flexible performance than Method ‘A’. Artificial Neural Network are know for their ability to reproduce well non-linear behaviors. They also tent to perform better in the context of large datasets. This may be the reason why period d showed results not as good as the other periods. For the periods forecasted the following inferences can be withdrawn:

a Followed the target reasonably well for the first half of the decade. The second half produced some inconsistent results but it stayed close to it average

b Followed rising trend well. Better for low targets. Overestimated the higher DRLs values of the end of the decade.

c Performed well for this period also. During the first half of the decade it got 4 out 5 predictions very close to the targets. The second half had even better results with 4 out of 5 predictions very close to the target as well.

d Descent performance in the begging of the decade. Poor behavior on the rising trend of targets at the end of the decade.

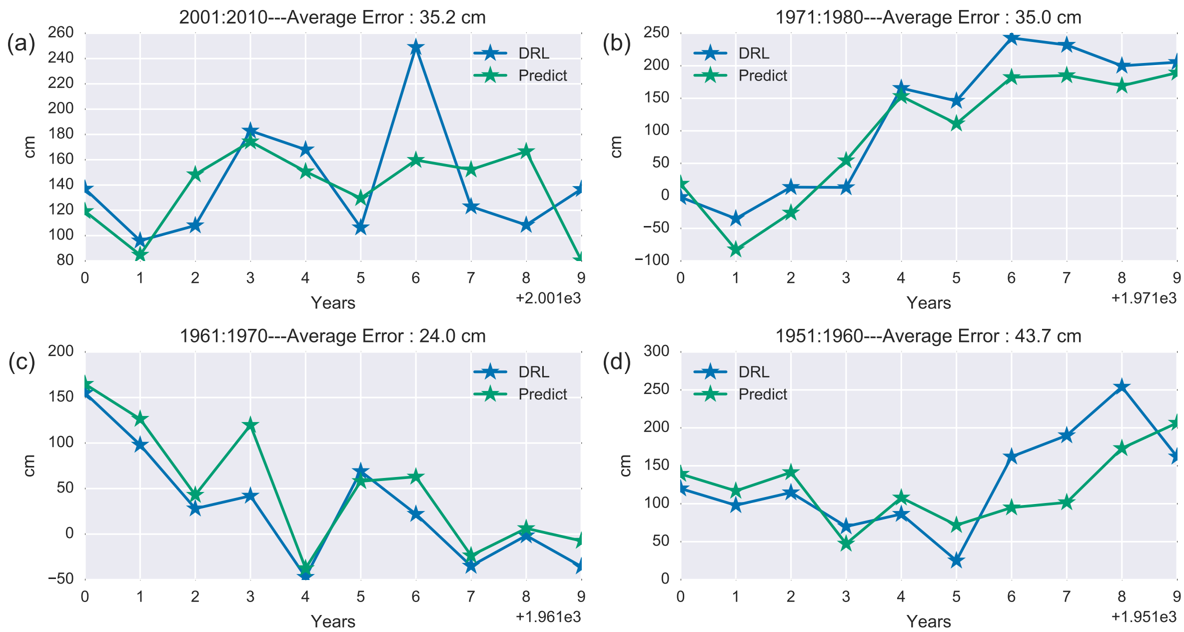


FIGURE 42 – Method ‘E’ prediction results.

Methods Comparison

Table 12 presents the magnitude of the errors for each method and years of tested periods. From that, it is observable the error variations and how well each method reproduced the real values of annual water level 10th percentile (DRL). In colors the magnitude of the errors.

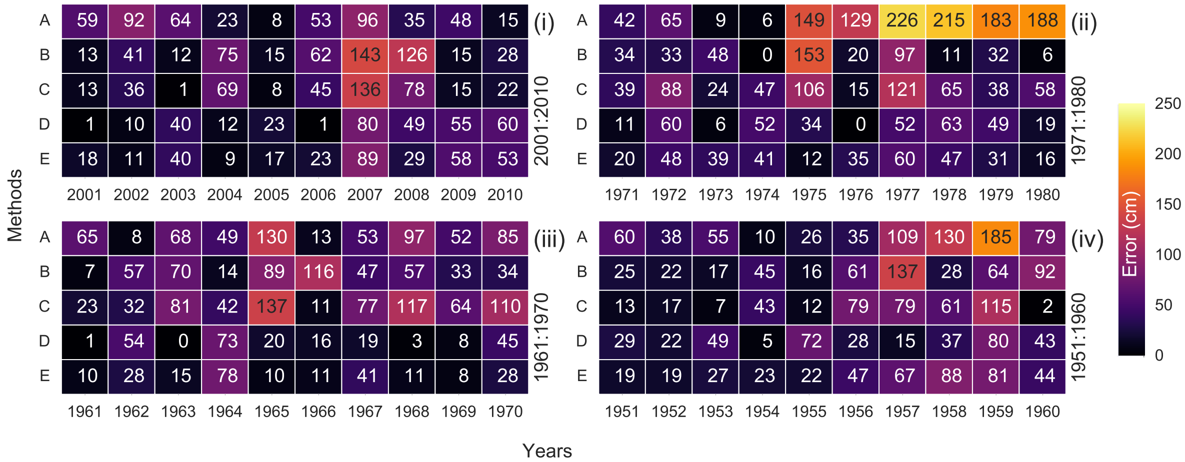


TABLE 12 – Error magnitude in centimeters for each method (vertical axis) by years of each period (horizontal axis). (i) Period of 2001:2010; (ii) Period of 1971:1980; (iii) Period of 1961:1970; (iv) Period of 1951:1960.

Figure 43 and Table 13 present the distribution of errors for each method and tested period. The following characteristics of the methods' results are highlighted: the mean; the standard deviation; the minimal and maximal error values; and the 25th percentile, median and 75th percentile.

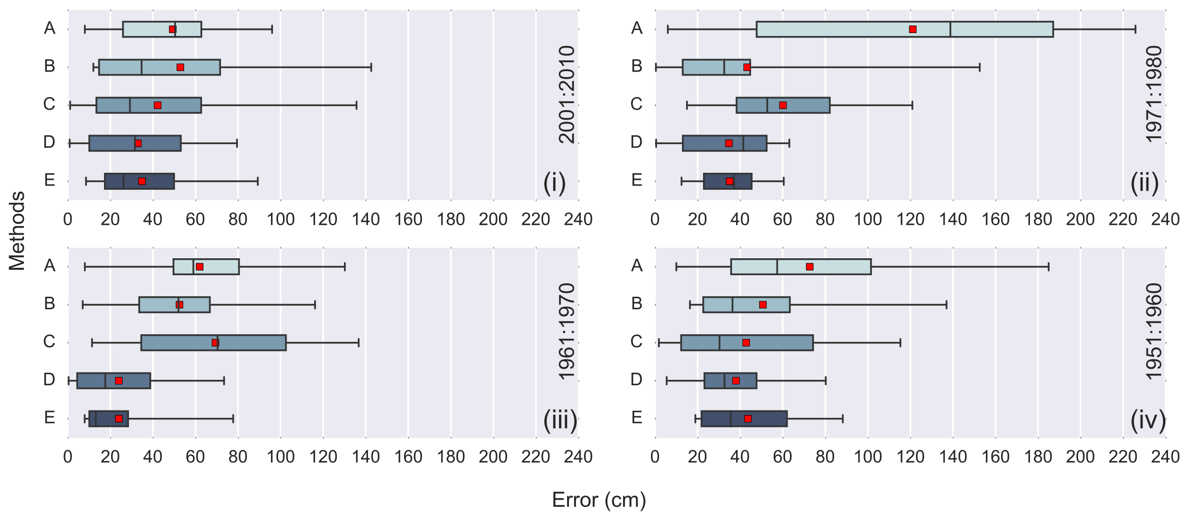


FIGURE 43 – Distribution of errors for each method (vertical axis) and tested period (horizontal axis). (i) Period of 2001:2010; (ii) Period of 1971:1980; (iii) Period of 1961:1970; (iv) Period of 1951:1960.

Of all the methods, Method D (multiple regression) showed the best overall results. It presented smaller values for all the parameters and an average improvement of 43.9 cm when compared with the current Method 'A'. This represents an advance of

	A	B	C	D	E
count	40.0	40.0	40.0	40.0	40.0
mean	76.3	49.8	53.6	32.4	34.5
std	60.0	40.9	39.9	24.7	23.0
min	6.0	0.4	1.1	0.4	7.9
25%	35.0	16.7	16.6	10.8	17.0
50%	59.5	33.8	44.1	28.2	28.3
75%	100.3	65.5	78.8	52.3	46.8
max	225.8	152.6	136.7	80.2	89.3

TABLE 13 – Overall description of errors' distribution in centimeters for each method.

57%, that is, the new error is less than half of the original. The values of the median, 25th and 75th percentile showed great improvement as well, specially for the larger errors indicator. The 75th percentile had an error of 100.3 cm for Method 'A' while for Method 'D' only 52.3 cm, which represents a little over half of the original value. When comparing the maximum error values, Method 'A' is significantly worse than Method 'D', and has a maximum error tree times higher with a variation of 145.6 cm. When compared with the other methods, 'D' also had significantly better results (except for Method 'E'). As to the results of the other methods, Method 'E' (ANN) showed very similar results to Method 'D', with slightly worse results for the period with the smaller number of training years. All of this methods showed better results than Method 'A'. The implications of this results are further discussed in the discussion section (5).

4.5 Water Modeling and Dredging Volumes

In this section the results from the water modeling and dredging volumes estimations are presented.

4.5.1 Digital Elevation Model

The results of the 'passo' Caraguatá bathymetric survey's coordinate transformation and digital elevation model (Sections 3.6.3 and 3.6.4) are presented in Figure 44.

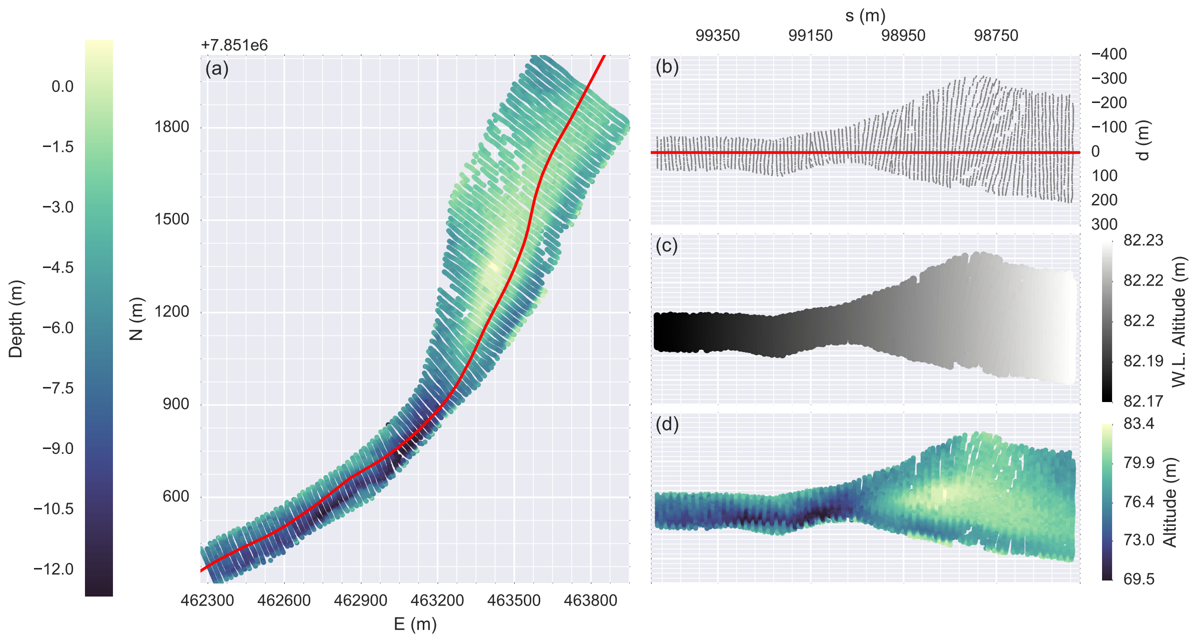


FIGURE 44 – DEM development results (a) Original bathymetric survey at the (E,N) coordinate system (WGS84 - UTM 21S); (b) Bathymetric surveyed points with the coordinated points transformed to the (s,d) coordinate system; (c) Water level values interpolated with altitude reference marks and bathymetric reference levels (BRL) interpolations ((s,d) coordinate system); (d) Resulting bathymetry altitude points originated by (c) - (a) at the (s,d) coordinate system.

4.5.2 Depth and Dredging Volumes Variations

In this section the depth and dredging volume variations due to different water level models in different DRL scenarios are presented. The depth variations are first presented qualitatively with profile graphs. The objective with this is to present visually the resulting differences from each method. In the sequence the total dredging volumes are presented. The result description is detailed but a deeper analysis is left for the discussion section.

In total 12 variations of water level models and contour values were tested. The details of each method are presented in Section 3.6. Figure 45 presents resulting average depth profiles of the bathymetric survey of Passo Caraguatá with its depths corrected by the different water level models. It's possible to observe the dimension of this variations and the potential volumetric variations that derive from it when contrasted with the depth requirement level. Table 14 presents the overall volumes for each of the 12 methods tested.

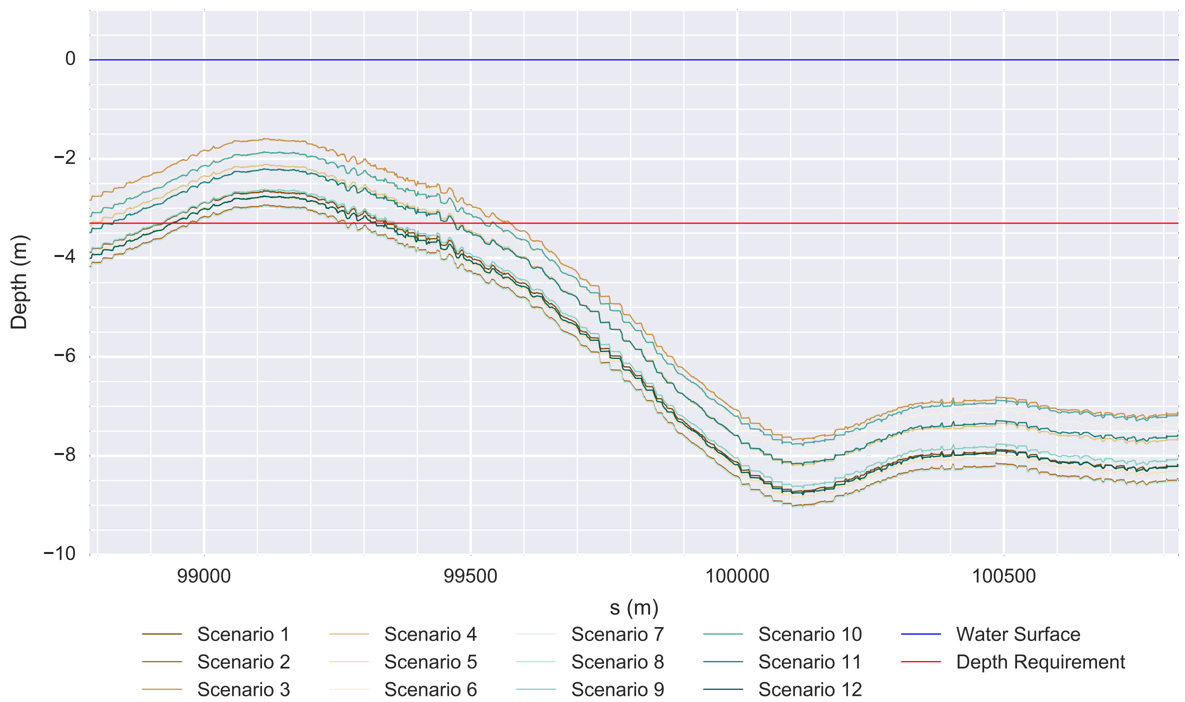


FIGURE 45 – Averaged cross-section depths for all of the scenarios tested. 1 - The current applied method, with the Navys BRL and linear interpolation between Ladário and Forte Coimbra. 2 - Linear interpolation between Ladário and Porto Esperança with the Navys BRL. 3 - 1D Hydrodynamic model with BRL contours. 4 - Linear interpolation between Ladário and Forte Coimbra with DRLs of the year 2000. 5 - Linear interpolation between Ladário and Porto Esperança with DRLs of the year 2000. 6 - 1D Hydrodynamic model with contours of the year 2000's DRL. 7 - Linear interpolation between Ladário and Forte Coimbra with DRLs of the year 2005. 8 - Linear interpolation between Ladário and Porto Esperança with DRLs of the year 2005. 9 - 1D Hydrodynamic model with contours of the year 2005's DRL. 10 - Linear interpolation between Ladário and Forte Coimbra with DRLs of the year 2015. 11 - Linear interpolation between Ladário and Porte Esperança with DRLs of the year 2015. 12 - 1D Hydrodynamic model with contours of the year 2015's DRL. Depth requirement - 3.3 meters.

Comparing the scenarios with current praxis.

The averaged depth profile of the critical depth section for each variation of the experiment is presented in Figure 46. The plots are arranged in a way to vary the

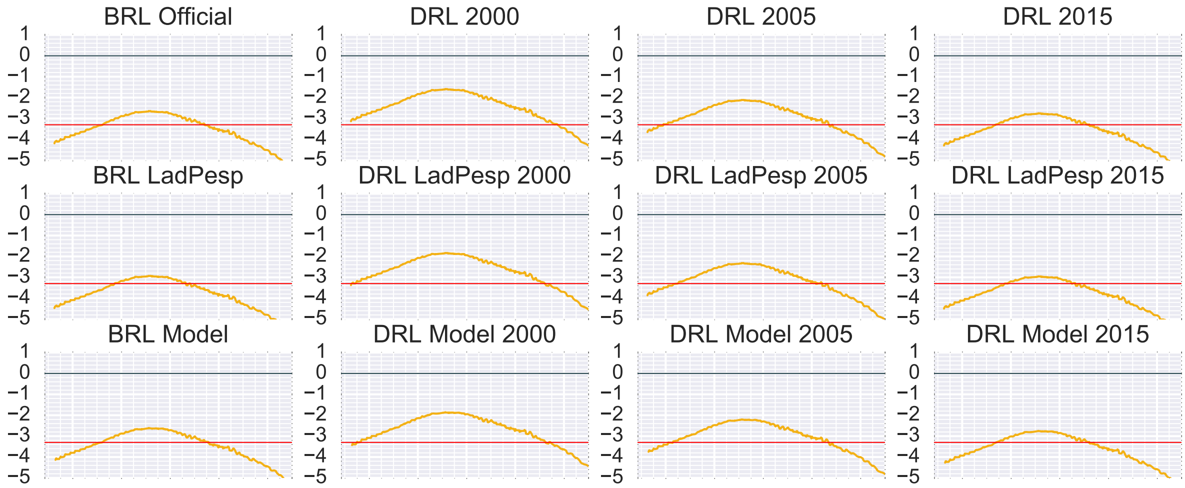


FIGURE 46 – Average depths for the various scenarios in the critical part of the stretch.

water level models in the columns and the contour conditions in the lines. With that, the qualitative variations of the methods are observable.

Table 15 shows the percentage variation of all the methods relative to the current method (linear interpolation of Navy's BRL between Ladário and Forte Coimbra.). From that, it is observable that there are significant variations in dredging volume related to the chosen method and contour conditions. For example, the dredging volumes to maintain this stretch navigable during 90% of the time of year 2000, using the interpolation between Ladário and Forte Coimbra, is 3.5 times higher than the volumes calculated by the traditional method (using the BRL). On the other hand, the dredging volumes to maintain this stretch navigable for 90% of the year 2015, using the interpolation between Ladário and Porto Esperança, is almost half the volumes calculated by the traditional method. This clearly indicates the importance of the precise definition of Dredging Reference Level and Water Level modeling as it can have profound economical impact on the maintenance budget and at businesses the rely on the navigability of the waterway.

TABLE 14 – Total of dredging volumes for the tested scenarios in cubic meters.

Model type	Contour	BRL	2000	2005	2015
Linear	Ladário-Forte Coimbra	27.059	97.038	57.495	22.607
Linear	Ladário-Pto. Esperança	16.162	77.679	44.147	15.387
Hydrodynamic 1D	Ladário-Pto. Esperança	27.939	66.931	47.084	22.637

TABLE 15 – Percentage variation of dredging volumes comparing the various scenarios with the current state of praxis (BRL - Ladário / Forte Coimbra).

Model type	Contour	BRL	2000	2005	2015
Linear	Ladário-Forte Coimbra	100%	359%	212%	84%
Linear	Ladário-Pto. Esperança	60%	287%	163%	57%
Hydrodynamic 1D	Ladário-Pto. Esperança	103%	247%	174%	84%

Comparing the scenarios with the same contour conditions.

The averaged depth profile of the critical depth section separated by contour conditions is presented in Figure 47. From that, the depth variation amplitude derived from the variations of the water level models (what propagates the contour conditions from the stations to the *loci*) can be checked. The linear models, one that uses the Navy's official stations and other that uses the closest valid stations, are compared with the 1D hydrodynamic model implemented.

The Table 16 shows the dredging volume percentage variations of the linear model and the hydrodynamic model in relation to the linear model between the Navy's BRL official stations. From that, it is observable that there are significant variations in dredging volume related to the chosen water level model. In general, the linear models interpolated from Ladário to Porto Esperança showed significant less dredging volumes than those interpolated from Ladário to Forte Coimbra (as currently done for the BRL in the region), resulting in an average of 71% of the original value. The dredging volume difference from the current linear interpolation and the 1D model increased with the severity of the scenario. That is, the lower the water level (thus also water flow), the greater the difference between the volumes. In this cases (years 2000 and 2005), the volume calculated with the 1D model is in average 75% of the value calculated with the current linear interpolation model. The BRL case and the year 2015 showed similar results because the 10th percentile of 2015 is coincides with the Navy's BRL value.

TABLE 16 – Percentage variation of dredging volumes comparing the scenarios with the same contour conditions (BRL, 2000, 2005 and 2015).

Model type	Contour	BRL	2000	2005	2015
Linear	Ladário-Forte Coimbra	100%	100%	100%	100%
Linear	Ladário-Pto. Esperança	60%	80%	77%	68%
Hydrodynamic 1D	Ladário-Pto. Esperança	103%	69%	82%	100%

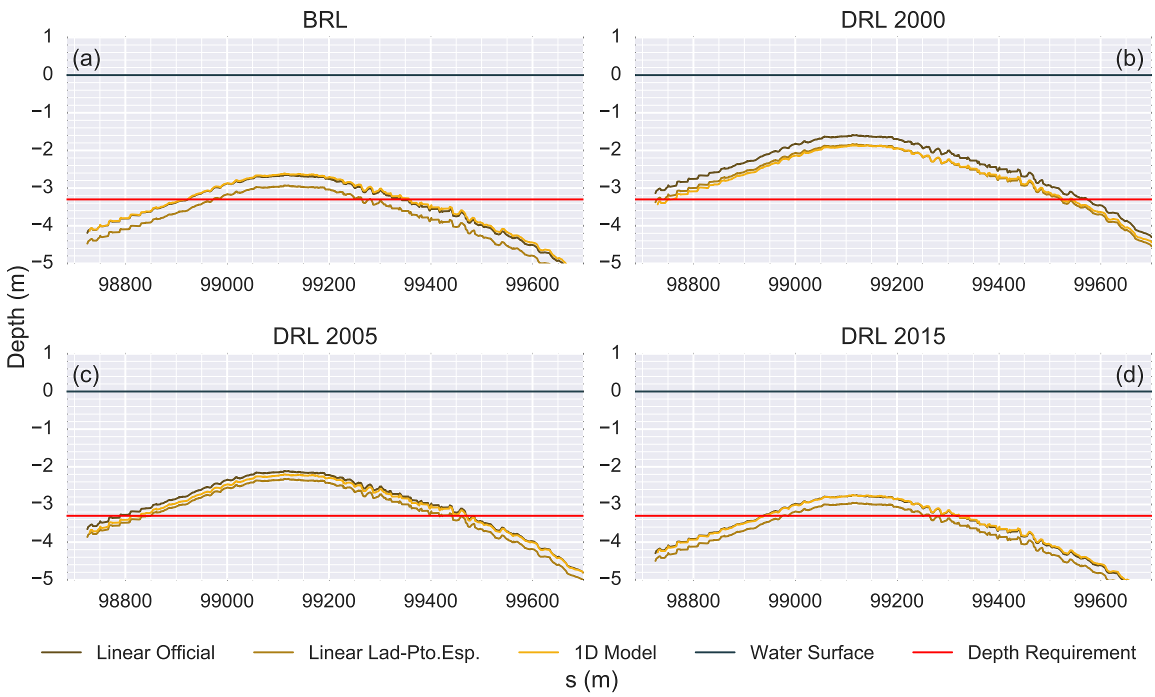


FIGURE 47 – Average depths variations. a) Comparison between methods that use the Navy's BRL as contour conditions. B) Comparison between methods that use the year 2000's DRL as contour conditions. c) Comparison between methods that use the year 2005's DRL as contour conditions. d) Comparison between methods that use the year 2015's DRL as contour conditions.

Comparing results with the same water level model.

The averaged depth profile of the critical depth section separated by water level method is presented in Figure 48. From that, the depth variation amplitude derived from the variations of the contour conditions (the DRL definition method) can be checked. The current method for defining the DRL (Navy's BRL) is compared with the real value of the 10th percentile of the years 2000, 2005 and 2015. As all of the DRL forecast methods obtained better results than the BRL method for the definition of DRL, by induction the profiles would also be more accurate.

The depth variation amplitude is clearly greater than in the case described in Figure 47, indicating that the DRL definition has more impact on the volume definitions than the water level method chosen (given the characteristics of the studied area).

Table 16 shows the dredging volume percentage variations of the real DRL values in relation to the Navy's BRL values as contour conditions. From that, it is observable that there are significant variations in dredging volume related to yearly 10th percentile (DRL) variation. In general, independent of the water level model chosen the volumes vary accordingly with the year water DRL values. For example, the dredging volume required to maintain the studied stretch navigable for 90% of the year 2000, when using the linear

TABLE 17 – Percentage variation of dredging volumes comparing the scenarios of the years 2000, 2005 and 2015 with it's respective scenario calculated with the Navy's BRL as contour condition.

Model type	Contour	BRL	2000	2005	2015
Linear	Ladário-Forte Coimbra	100%	359%	212%	84%
Linear	Ladário-Pto. Esperança	100%	481%	273%	95%
Hydrodynamic 1D	Ladário-Pto. Esperança	100%	240%	169%	81%

interpolation model from Ladário to Pto Esperança, was 4.8 times higher than when using the Navy's BRL reference and the same model. A variation like that would definitely have a significant impact on dredging budgets and businesses requirements. In general, the 1D hydrodynamic model showed less variation due to yearly changes.

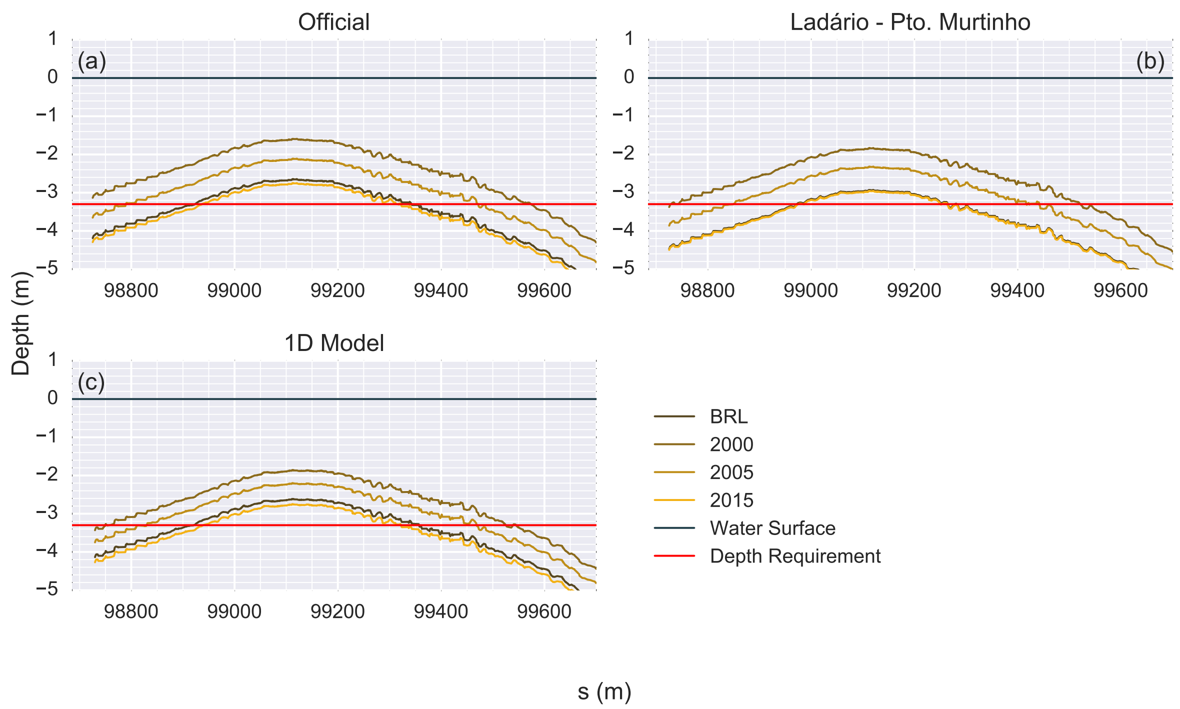


FIGURE 48 – Average depths variations. a) Comparison between methods that use the Navy's official BRL stations as contour conditions (Ladário - Forte Coimbra). B) Comparison between methods that use linear water level model and Ladário and Pto. Esperança as contour stations. c) Comparison between methods that use the 1D hydrodynamic water level model to set the depths.

5 DISCUSSION

In this chapter the discussion of the results of this dissertation is conducted. The results are described and the specific and general contributions and claims highlighted. The order was arranged as linear as possible, referencing the interconnections when appropriate.

In the first section it is presented the conclusions and findings concerning the current DRL method. In the sequence the methodological insights and its contributions towards the development of a better approach to the problem. After that the forecast hypotheses is discussed presenting the related finding, contributions and potential of uses.

In the second section the findings concerning the water level modeling and depths variations are discussed and the local aspects and characteristics that lead to the end result discussed. The DRL and water level modeling methods are weighted and prioritized. As the finding of this thesis have a local inclination, along the section, suggestions towards a model approach to investigate the generality of the findings are presented as well as recommendations for future research.

Forecasting Dredging Reference Levels

The study presented to propose a new method to define the Dredging Reference Level started with a Wavelet Analysis. This analysis allowed a general review of the water level behavior and periodic influences over the whole of the 20th century for the Ladário station. With that, it was possible to observe not only which were the predominant influences, but also what influences were relevant to each period. In that sense, it presented a gain of knowledge when compared with the results of (NORDEMANN, 1998). For example, in the beginning of the century a predominant periodic influence of 5 to 8 years were observed. From 1920 to 1940 the 15 and 3 years periodic influences predominated. From 1940 to 1955 several periodic influences were seen, mainly 2, 3, 8, 15, 27 years. The period from 1955 to 1975 was similar to the prior period except for the 3 years influence which was less relevant. After 1975, only the 2-year influence was eventually relevant which resulted in a much more stable behavior of the series.

What is important to filter from these results when assessing a method for DRL definition is that for the studied area there were a lot of water level behavior variation on the last century. There is no straight forward way to determine until when will the 2-year influence be the single most relevant influence (as currently). Due to edge effects it is hard to assess the periodical influences over recent years (e.g. it's only possible to assess, without biases, if the current behavior of the water level is influenced by a 27 years period, 27 years from now.). With these, it is reasonable to assume that a DRL definition method should be dynamic, flexible and take these past variability knowledge into account. It also presents evidences that the DRL wouldn't work well if treated as a river intrinsic characteristic.

After the considerations withdrawn from the first analysis, the quantification of error residue due to the current praxis was assessed. The visual inspection of the yearly 10th percentile series (Figure 35) demonstrated a similar behavior as encountered in the wavelet analysis. The results of the test developed clearly indicates that calculating DRL with 1 year to the past and updating it every year had better results than any other combination, specially with 20 years to the past and updating every 5 years — which almost doubles the error. These findings support the hypotheses developed in the wavelet analysis.

It is also clear that the drought period of the 1960s had a strong influence on the end result. The calculations with high numbers of years carried large errors until the beginning of the 1990s. Phenomena that didn't happen using small number of years for DRL calculation as the method's memory fades quicker and the recent events have exclusive importance. Nonetheless, as part of the water level history it should be taken into consideration as there are aways the possibility of happening again.

To confirm the findings so far, the autocorrelation analysis showed that, considering the studied water level series, an endogenous stochastic prediction don't need to look too far to the past in order to infer what the 10th percentile of the following year will be. As the last year presents the highest correlation coefficient, it's better if it doesn't. If from 10 to 1 past years the correlation increases in a seemingly exponential way, it's also reasonable to assume that the past months could also present good correlation and additional information and aid the DRL definition problem. Thus, the autocorrelation analysis helped the development of the necessary data structures for the DRL definition. As the wavelet analysis suggested that the DRL wouldn't work well if treated as an intrinsic river characteristic, this analysis showed there is a strong potential in treating it as a forecast problem to ensure an annual 90% of the time (or any other similar value) navigability.

Considering the forecast models developed and the implemented error assessment experiment, it's possible to conclude that the overall forecasting approach for yearly percentiles is better than the current longterm percentile approach. It allowed and average error reduction from 76.3 cm of a typical current method (Method 'A') to 32.4 cm of the best prediction method (Method 'D'). This increase in precision proves that it is possible to have a more precise and accurate method keeping the 'endogenous only' input data format. Hence, delivering an optimized solution to assist the maintenance of inland waterways.

Related specifically to the forecasting methods, it is worth mentioning that for the 20 year's percentile method (current), although reasonable for long periods of stationary behavior, in the long term it is a risky approach and can produce expressive errors, potentially leading to the interruption and harm of navigability for long periods of time.

As for the multiple regression, it was a straight forward and practical way for forecasting drought percentiles, with good precision and accuracy. It showed the best overall ability of mapping and reproducing behavior for the studied station. Keeping the errors low enough to be feasibly included as 'safety' margin on real operations.

The artificial neural network showed very good results as well. The fine-tuning of the many ANN parameters might result in even better results. One of its disadvantages however, is the significant higher cost in computational time than the fast multiple regression. If with refinements in precision this method precision improves, today's computation optimizations and the use of graphic cards for calculations can significantly improve computational times. Further investigations can be performed to test different cost and validation functions, different structures and training approaches.

Although this results achieved for the Ladario's station can be considered good, they cannot be generalized for the hole extension of the river. Aside from data availability issues, local characteristics of the river and the upstream basin certainly have key physical influences on the composition and shape of which input data would produce the best results (*e.g.* the number of months or years used to forecast future behavior). Further investigations could be performed to assess the correlation of physical characteristics with the input data shape (*e.g.* size of basin and number of months used to the forecast).

An important aspect while assessing the generalization is the data availability and quality for each station. Temporal bias may induce some methods to show better results than others, specially when a wide variability of scenarios have not been recorded. A clear example of that is to imagine if there were only available data for the Ladario's station after the intense drought of the 1960s. Another key thing to remember is that the Wavelet analysis, the percentile parameter variation test and the autocorrelation analysis, all need homogeneous continuous data to be implemented. Thus, highlighting the importance of the data continuity on the water level stations.

From that perspective, it is important to learn with the behavior of other stations on the same basin. This also indicates the possibility of having/testing forecast methods that include not only endogenous local data but also data from other stations (exogenous), specially on the upstream basin of the station which water level behavior is sought to be forecast. With that, it is also expected a rise on forecast precision.

In general, considering the unseen approach to define maintenance Dredging Reference Levels (DRL) in formal literature, the application of formal forecast models to the issue and the overall approach presented in this dissertation can be considered an advance in the inland waterways' optimization field, specially for the Paraguay River Inland Waterway characteristics and the Brazilian transport management and data availability contexts.

Digital Elevation Model

The objective of the creation of a Digital Elevation Model (DEM), in the context of this dissertation, was to attribute to surveyed depth points an altitude value. The resulting river bottom altitude would then be used to run a hydraulic model and to update depth values. Given that, the challenge faced here was that both bathymetric datasets used (for the 1D model and for the dredging calculations) had no water level altitude reference for the days of their surveys. So, in order to define a local water level altitude for the day of the survey (or reference of the survey), the only information available was the altitude of the water level stations, their readings and several altitude reference marks along the river. The options for water level model were then narrowed given that 1D hydraulic models require DEMs.

The convoluted nature of this issue forced the application of a linear water level model between stations knowing that it is likely to be less precise. Consequently, it induced one of the central assumptions that sustained the results of the section concerning dredging volume calculations. That is, considering that the dredging volume results from the different water level models and scenarios have all the same DEM basis, any variation related to the creation of the DEM are 'felt' by all the models — thus assumed proportional at the end result and reasonably neglected for the purpose of relative comparison.

This issue raises questions that can be covered in future investigations. Specially, concerning what would happen if after applying the linear model to define the DEM, a hydraulic model was run for the same contour conditions and then used to update the DEM altitude again. Questions like the following could be answered: If iterated, would this method converge to a final altitude value? Would these variations be significant? And what would their magnitude be?

Nonetheless, the method implemented to obtain a DEM, in this scenario, was not located in any formal literature reviewed. Given that there are several bathymetric surveys of the Paraguay river (and others) that don't have water level altitude references, the described method can be considered one of the contributions of this dissertation towards the optimizations of inland waterways and related areas.

Depth and Dredging Volumes Variations

The central objective of this section was to assess the impact of water level models and DRL definitions on a tangible engineering parameter, the dredging volume. This is significant because it is central to estimate overall feasibility and costs of inland waterway projects.

The experiment developed allowed the verification of the dredging volume for 3 different water level models and 4 different contour conditions (BRL, 2000, 2005 and 2010). These calculations were then compared to the results of the current praxis (Navy BRL interpolation), by model type, and by contour condition.

Furthermore, the number of contour conditions analyzed didn't allow a statistical inference, neither was it the aim of this analysis. Nonetheless, as the dredging volume is directly proportional to the DRL, and that received a statistical analysis in the previous section, by induction, it is possible to translate the results obtained there here.

Importantly, given that the minimum depth requirement for navigation is fixed as a waterway vessel's characteristic; that the water level equivalent to the 10th lowest annual percentile changes every year; and that the dredging volume is a function of the DRL. It is possible to infer that keeping the DRL calculation method as is, results in big dredging volume mistakes every year. That said, it jeopardizes the inland waterway operations and characterizes a miss use of public funds (in the Brazilian context).

Similarly, the water level model used to define the dredging volumes is also important. Its correct treatment can avoid additional project errors. To that end, the use of hydrodynamic models are standard in literature. Then again, data, technical and fund availability can be a hindrance in the Brazilian context as well as in other developing countries. Provided that and acknowledging the inherited imprecision, linear models can be considered. That being the case, the spacing between stations used for interpolation must be considered carefully as it can have a significant impact in the end result.

All things considered, this dissertation presented a DRL definition and water level models elements that can lead to a better use of inland waterway's maintenance budgets. For example, for the year 2000 if it's considered an approximated cost of 15 US dollars per cubic meter of dredged material [1], the traditional method of linear interpolation and Navy's BRL would result in a cost of $405,900.00. For the same year using the 1D hydrodynamic model and the exact contour values of 2000s 10th percentile, the cost would be $1,004,000.00. This means that a significant amount of money would be spent and the expected results wouldn't be met. Another example is that, by using the precise DRL values and the hydrodynamic model instead of the linear interpolation model, there would

[1] Averaged value found in AHIPAR (2015)

be savings of $450,000, which also means none unnecessary local environmental impact.

According to (LIMA, 2005) along the Paraguay river, in different areas with different characteristics, the year 2000 had a total of 298,000 cubic meters dredged. If the same ratio from the results of this study was maintained, the dredged volume using the hydrodynamic model and exact DRL values would have been close to 750.000 cubic meters. Using the unitary cost assumed, the difference would have been of $6.75 million. If the exact DRL values were used, the difference between using a linear model and a hydrodynamic model would have been 300,000 cubic meters. This means that using a hydrodynamic model would have saved the maintenance operation a total of a $4.5 million for 1 year. Off course, the characteristics of the river in other regions must be taken into consideration. Then again, the purpose of this example is to show the potential impact the findings can have in the overall dredging maintenance services.

In conclusion, for the given experiment and its characteristics, the key findings were:

- Different water level models can produce significantly different dredging volumes;
- The impact of using the current BRL as DRL is significant when comparing dredging volumes calculated with this different contour conditions.
- The kind of water level models implemented, yet still relevant, is less significant than the DRL definition methods concerning the dredging volume totals.
- If linear, the spacing between stations used for interpolation must be considered carefully.

One assumption that had to be made was the bed morphology evolution simplification. It may in fact change the location and the amount of dredged volumes. Indeed, further developments need to be made with that focus. Nonetheless, it's hardly possible that this changes over the period of one year would affect local water levels. Thus, not altering the results of the forecast. This issue could be 'patched' by a bathymetric survey prior to the effective dredging of the areas.

6 FINAL CONSIDERATIONS

The present exposition of depth-related inland waterways processes, considering the characteristics of the locations under study, rest upon some basic assumptions:

- There is a demand for inland waterway usage that requires a fix percentage of operational time assurance;
- An annual 90% of operational time assurance is considered;
- Riverbed morphology in the period of one year don't alter significantly local water levels;
- The river's DEM is calculated by linear interpolation of water levels for all the scenarios;
- To date there is no clear distinction between reference levels for Nautical Charts and for Dredging.
- The river under study has no tidal or dam influence;
- The Navigation Channel is considered fixed for all dredging simulations;
- The impact of yearly water level trends variations on dredging volumes has not been covered in literature;
- Although water level models were already implemented, the comparison of its impacts on dredging volume calculations have not been published.

That being said, the overall aim of this dissertation was to provide tool-sets and interpretations to aid the improvement and optimization of waterway's designs and maintenance, focusing primarily on the Brazilian context.

Initially the interpretation of the differences between a BRL and a DRL had to be set. They were key to a more appropriate treatment of the issues concerning waterways depth availability and performance.

In the sequence it was demonstrated how rivers' water level multi-annual influences can be of central importance to DRL definitions. Also, that there is no reasonable evidence for treating the DRL as an intrinsic river's characteristic. And that forecast models can be powerful tools in the pursuit of an exact operational time assurance (*e.g.* 90%).

Moreover, the forecast approach aimed at assessing how far could simple endogenous predictions reach precision wise. Presenting a significant improvement on DRL definition, a considerable gain of precision as well as setting a 'benchmark' for more complex approaches (*e.g.* multi-station, hybrid, exogenous).

Furthermore, it was also demonstrated how DRL imprecision can affect the dredging volumes of a project. Likewise, that a better water level model is likely to ‘pay for it self’ along the years with the resulted gains due to more precise and accurate dredging.

All things considered, this dissertation is a significant structured gain of knowledge for the studied area. Although derived generalizations of the conclusions are unreasonable, the generalization of the methodology can serve as a framework for studies contemplating other stretches and waterways. Additionally, it could also assist the formulation of hypothesis for studies that have as goal the convergence of generalized practices.

As a final recommendation, it is stated that the bureaucratic aspects of inland waterway management should be shaped and adapted around the best technical possibilities not the other way around. This ensures that inland waterways operates at maximum feasible capacity within real environmental constraints.

A COMPLEMENTARY RESULTS

A.1 Current DRL methods Error Quantifications.

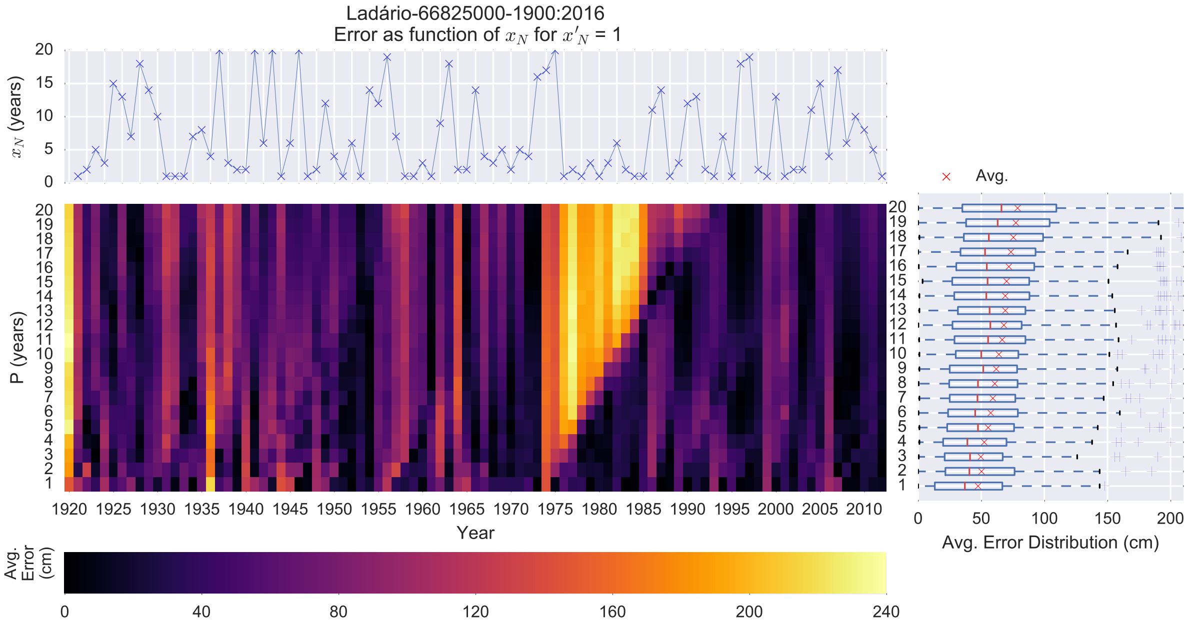


FIGURE 49 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 1$.

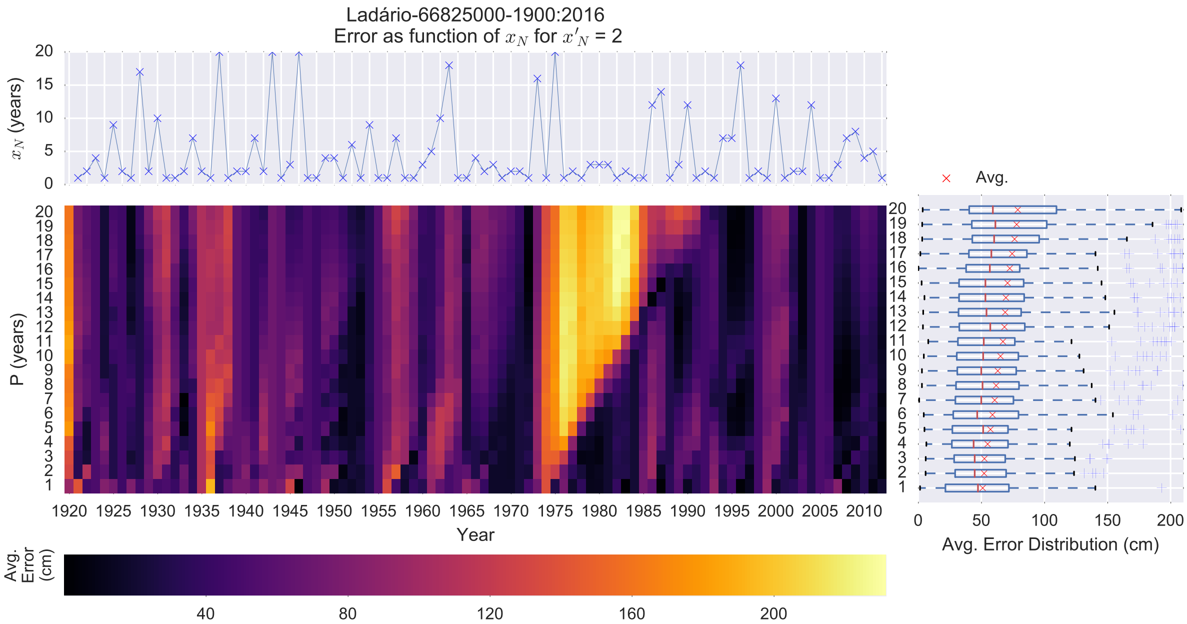


FIGURE 50 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 2$.

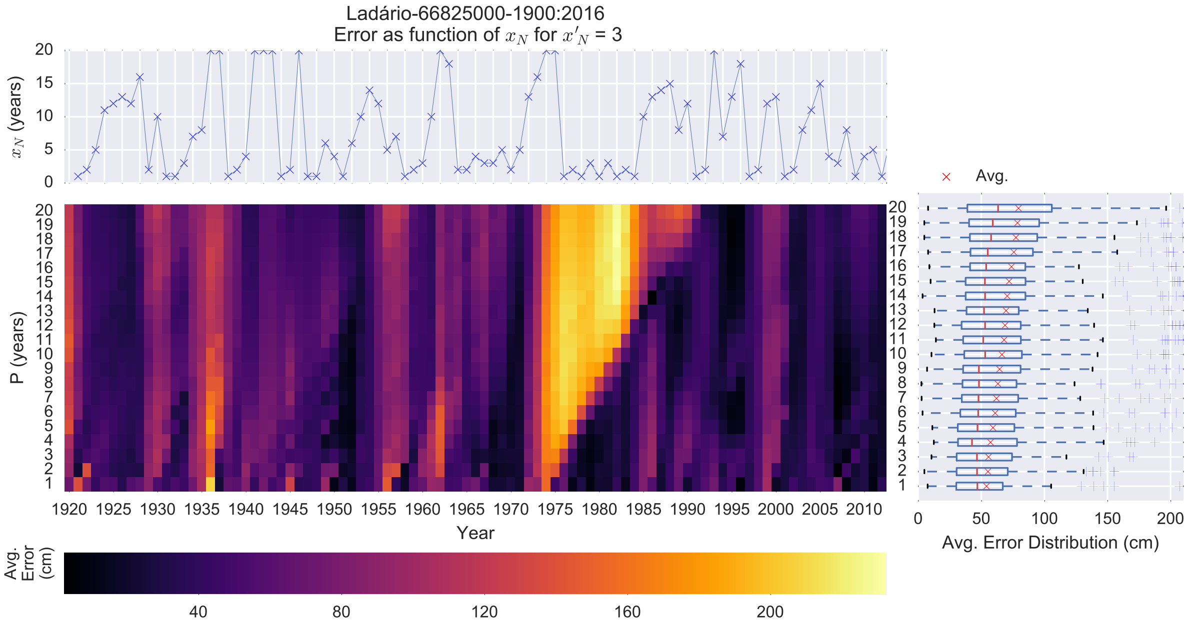


FIGURE 51 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 3$.

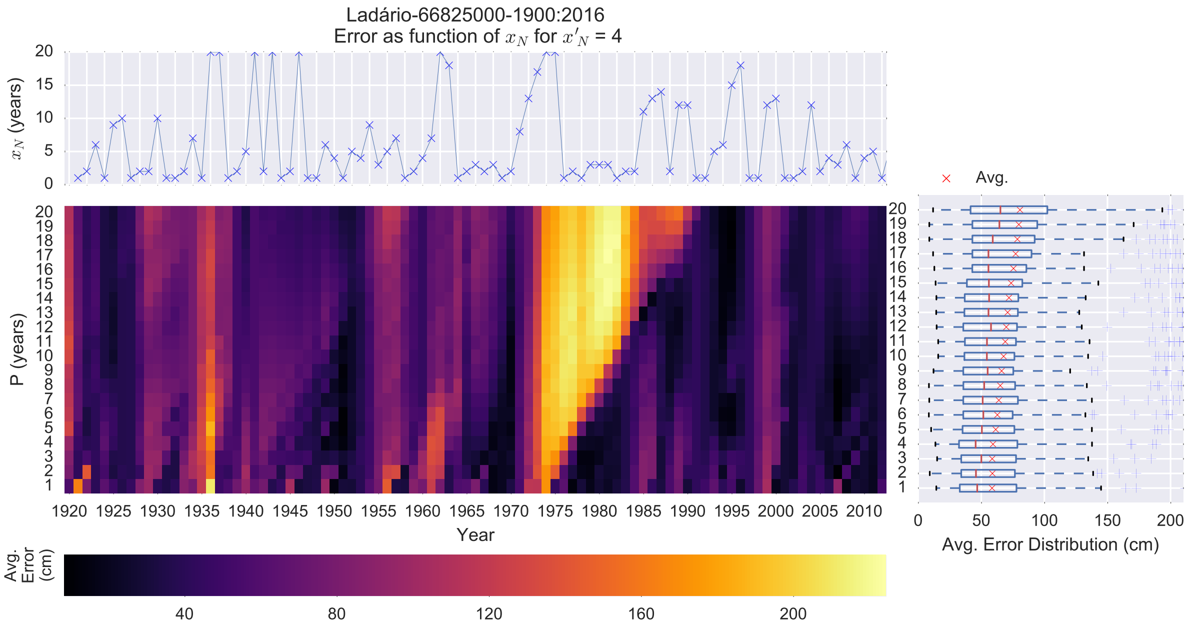


FIGURE 52 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 4$.

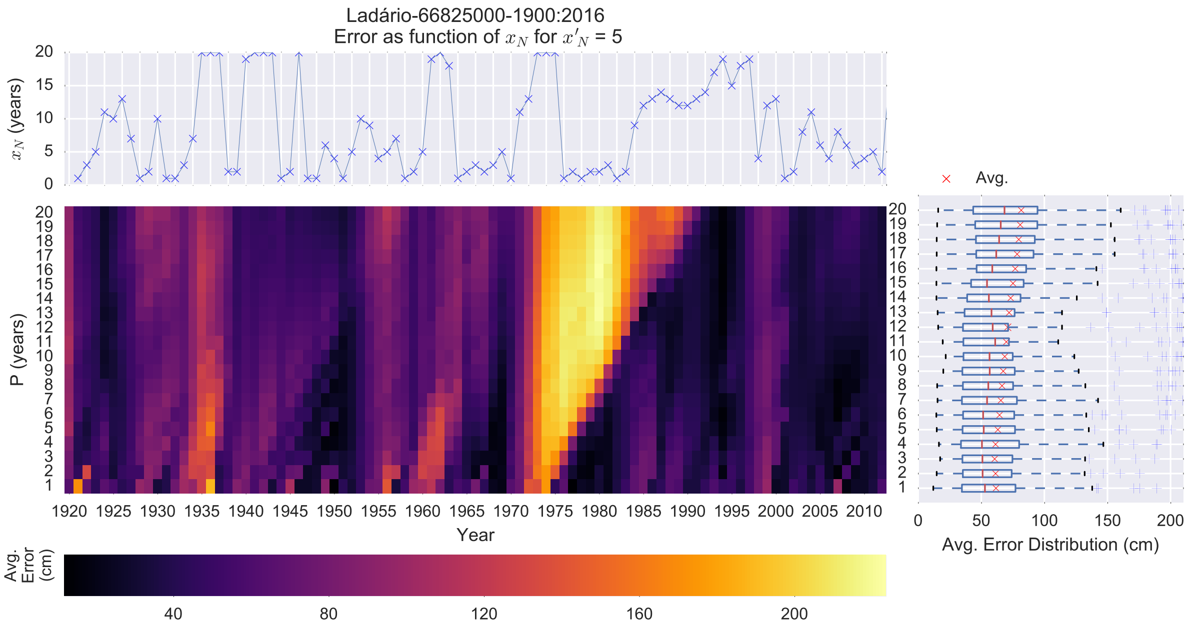


FIGURE 53 – Resulting average prediction errors and average errors distributions varying P from 1 to 20 with $x'_N = 5$.

B STRUCTURES

B.1 forecast_models structure

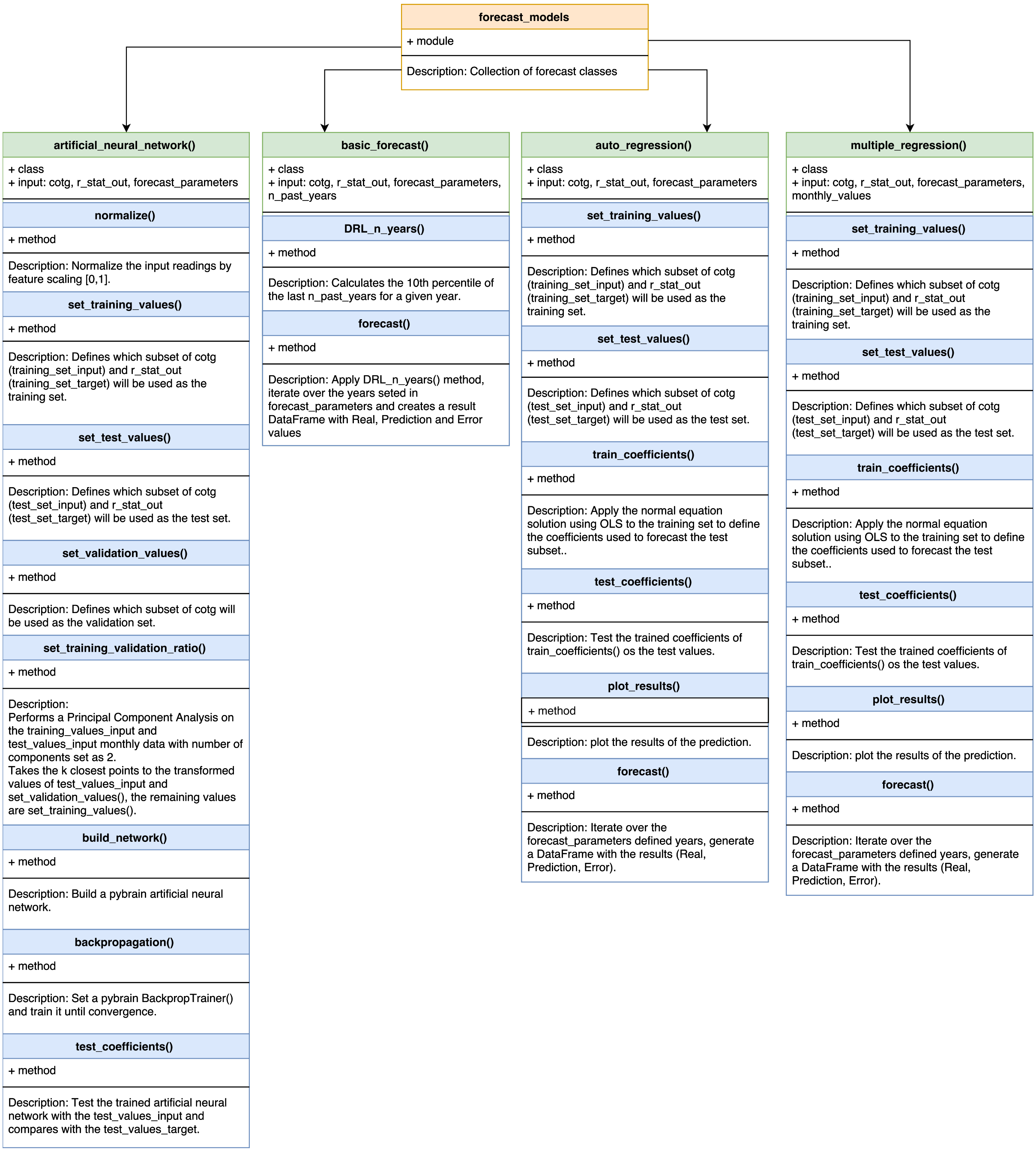


FIGURE 54 – forecast_models module structure

C SCRIPTS

C.1 Wavelet Analysis Script

```
In [1]: import sys
        print(sys.version)
```

```
2.7.11 |Anaconda 2.3.0 (64-bit)| (default, Feb 16 2016, 09:58:36) [MSC v.1500 64 bit (AMD64)]
```

```
In [2]: import pandas as pd
        import seaborn as sns
        import os
        import matplotlib.ticker as ticker
        from wavelets import WaveletAnalysis
        get_ipython().magic(u'pylab inline')
        pd.options.mode.chained_assignment = None
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [3]: figuras = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\figuras\\'
        tabelas = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\tabelas\\'
```

Reading of the Laradario stage series.

```
In [4]: cotg = pd.read_pickle(tabelas+'oGrafico.p')
```

```
In [5]: #print the first 5 lines
        print(cotg.head(5))
        #print the last 5 lines
        print(cotg.tail(5))
```

```
Data_l      Data_r  Dia   Cota  Mes   Ano
1900-01-01 1900-01-01 1900-01-01    1  186.0    1  1900
1900-01-02 1900-01-02 1900-01-02    2  185.0    1  1900
1900-01-03 1900-01-03 1900-01-03    3  185.0    1  1900
1900-01-04 1900-01-04 1900-01-04    4  185.0    1  1900
1900-01-05 1900-01-05 1900-01-05    5  185.0    1  1900
               Data_l      Data_r  Dia   Cota  Mes   Ano
2015-07-28 2015-07-28 2015-07-28  209  455.5    7  2015
2015-07-29 2015-07-29 2015-07-29  210  455.0    7  2015
2015-07-30 2015-07-30 2015-07-30  211  454.5    7  2015
2015-07-31 2015-07-31 2015-07-31  212  453.5    7  2015
2015-08-01 2015-08-01        NaT  213    NaN    8  2015
```

Separation of the x(days) and y(stage) of the period of interest. Normalizations of y by $\left(\frac{y-\mu}{\sigma}\right)$.

```
In [6]: t  = cotg.index[cotg.index.year < 2003]
        x = (np.array(cotg.Cota[cotg.index.year < 2003].tolist(),dtype='float') -
            cotg.Cota[cotg.index.year < 2003].mean())/cotg.Cota[cotg.index.year < 2003].std()
```

Given a signal x(t),

```
In [7]: t
```

```
Out[7]: DatetimeIndex(['1900-01-01', '1900-01-02', '1900-01-03', '1900-01-04',
                       '1900-01-05', '1900-01-06', '1900-01-07', '1900-01-08',
                       '1900-01-09', '1900-01-10',
                       ...
                       '2002-12-22', '2002-12-23', '2002-12-24', '2002-12-25',
                       '2002-12-26', '2002-12-27', '2002-12-28', '2002-12-29',
                       '2002-12-30', '2002-12-31'],
                      dtype='datetime64[ns]', length=37620, freq=None)
```

```
In [8]: x
```

```
Out[8]: array([-0.54777357, -0.55425061, -0.55425061, ..., -1.07241311,
               -1.07241311, -1.06593608])
```

and a sample spacing,

```
In [9]: dt = 1 #day
```

the following, implements the wavelet transform:

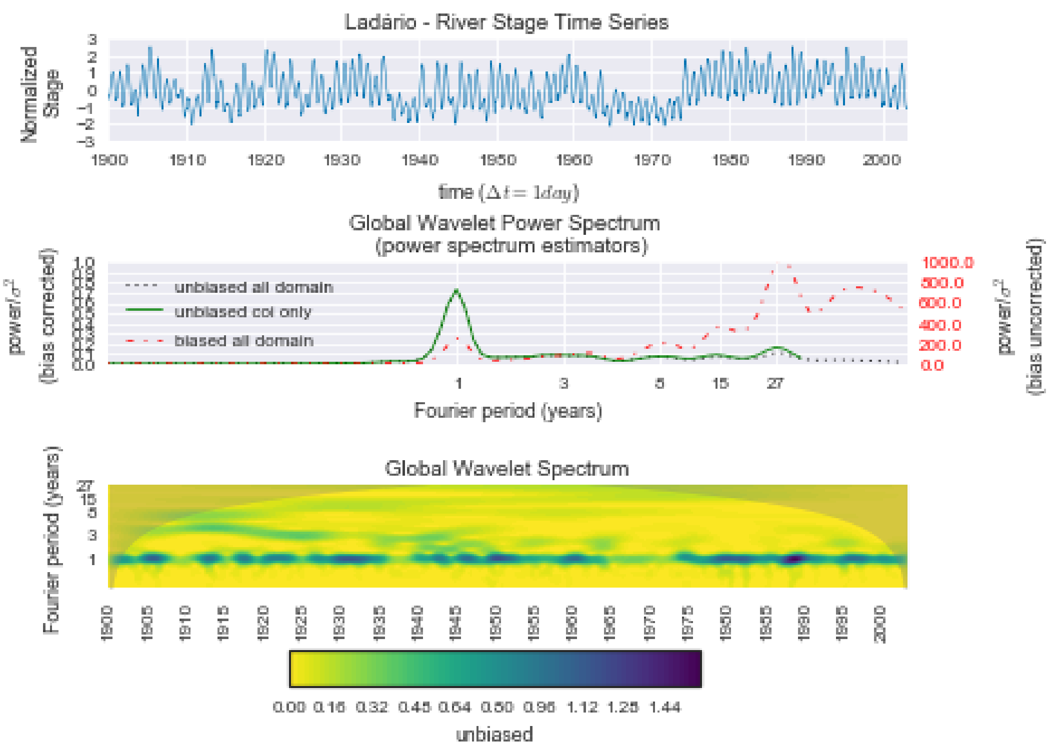
```
In [10]: wa = WaveletAnalysis(x, dt=dt, unbias=False)
         power_biased = wa.global_wavelet_spectrum # wavelet power spectrum with bias
         wa.unbias = True
         power = wa.global_wavelet_spectrum # wavelet power spectrum with bias correction
         wa.mask_coi = True
         power_coi = wa.global_wavelet_spectrum
         freqs = wa.fourier_periods
```

and this produces the graphical visualization:

```
In [11]: rc('font',family='Senasds Serif',size=10)
         sns.set_style("dark")
         sns.set(palette="colorblind")
         sns.set_context("paper")
         fig, ax = plt.subplots(nrows=3)
         figsize(18,20)

         ########################
         #PROFILE PLOT###########
         ########################

         ax_profile = ax[0]
         ax_profile.set_title(u'Ladário - River Stage Time Series')
         ax_profile.plot(t, x,lw=0.5)
         ax_profile.set_xlabel('time ($\Delta t = 1 day$)')
         ax_profile.set_ylabel('Normalized'+'\n'+u'Stage')
         ax_profile.set_xticks(pd.date_range(start='31/12/1899', end='01/01/2003', freq='10AS'))
```

```
########################
#UNBIASED POWER PLOTS###
########################

ax_power = ax[1]
ax_power.set_title(u'Global Wavelet Power Spectrum \n (power spectrum estimators)')
ax_power.plot(freqs, power, 'k', label=r'unbiased all domain',lw=0.8,ls=':')
ax_power.plot(freqs, power_coi, 'g', label=r'unbiased coi only',lw=0.8,ls='-')
grid(0)
ax_power.set_xscale('log')
ax_power.set_ylim(0,1)
ax_power.set_yticks(linspace(0,1,11))
ax_power.set_xlim(10 * dt, wa.time.max())
ax_power.set_xticks([365,1095,8*365,15*365,27*365])
ax_power.set_xticklabels(['1','3','8','15','27'])
ax_power.set_xlabel(u'Fourier period (years)')
ax_power.set_ylabel(r'power/$\sigma^2$'+'\n'+'(bias corrected)')


########################
#BIASED POWER PLOTS#####
########################

ax_power_bi = ax_power.twinx()
ax_power_bi.plot(freqs, power_biased, 'r', label='biased all domain',lw=0.8,ls='-.')
ax_power_bi.set_xlim(10 * dt, wa.time.max())
ax_power_bi.set_ylabel(u'power / $\sigma^2$ \\n (bias uncorrected)')
ylabel(r'power/$\sigma^2$''\n'+'(bias uncorrected)')
ax_power_bi.set_yticks(linspace(0,1000,6))
ax_power_bi.set_yticklabels(ax_power_bi.get_yticks(), color='r')
ax_power_bi.set_ylim(0,1000)
label = "T={0}"
'''for T in (T1, T2, T3):
    ax_power.axvline(T)
    ax_power.annotate(label.format(T), (T, 1))'''
grid(0)

ax_power_bi.legend( loc=3)
ax_power.legend(loc=2)

########################
#POWER SPECTRUM PLOT####
########################


ax_transform = ax[2]
sns.set_style("white")

X, Y = np.meshgrid(wa.time, wa.fourier_periods)
ax_transform.set_title(u'Global Wavelet Spectrum')
#ax_transform.set_xlabel(u'time \n($\Delta t = 1 day$)')
ax_transform.set_ylabel(u'Fourier period (years)')
ax_transform.set_ylim(100 * dt, 10000 * dt)
ax_transform.set_yscale('log')
ax_transform.set_yticks([365,1095,8*365,15*365,27*365])
ax_transform.set_yticklabels(['1','3','8','15','27'])
```

```
ax_transform.set_xticks((365*np.array(range(0,101,5))))
ax_transform.set_xticklabels(5*np.array(range(21))+1900,rotation=90)
figura2 = ax_transform.contourf(X, Y, wa.wavelet_power, 100,cmap='viridis_r')
#rect = l,b,w,h
rect = 0.3,-0.015,0.4,0.05
colorbar_ax = fig.add_axes(rect)
fig.colorbar(figura2, ax=ax_transform, label='unbiased',orientation='horizontal',use_gridspec=True,cax
grid(0)

# Shade the region between the edge and coi
C, S = wa.coi
F = wa.fourier_period(S)
f_max = F.max()
ax_transform.fill_between(x=C, y1=F, y2=f_max, color='gray', alpha=0.3)
fig.subplots_adjust(wspace=1.25
                    , hspace=1.2)
#fig.tight_layout()
fig.savefig(figuras+'test_power_bias3.pdf',bbox_inches='tight')
```



C.2 Percentile based DRL calculations and errors quantification.

System Version

```
In [1]: import sys
        print(sys.version)
```

```
2.7.11 |Anaconda 2.3.0 (64-bit)| (default, Feb 16 2016, 09:58:36) [MSC v.1500 64 bit (AMD64)]
```

Necessary Libraries

```
In [2]: import pandas as pd
        import seaborn as sns
        import os
        import matplotlib.ticker as ticker
        from wavelets import WaveletAnalysis
        import calendar
        from datetime import timedelta
        import numpy as np
        get_ipython().magic(u'pylab inline')
        pd.options.mode.chained_assignment = None
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [3]: def add_months(sourcedate,months):
            ''' Method to assist stack_cota with month shifts'''
            month = sourcedate.month - 1 + months
            year = int(sourcedate.year + month / 12 )
            month = month % 12 + 1
            day = min(sourcedate.day,calendar.monthrange(year,month)[1])
            return datetime.date(year,month,day)
```

```
In [4]: def stack_cota(df,consistencia):
            '''
            Recives a structured stage dataframe (pd.DataFrame)
            COTA.txt of ANA (with header set as 15),
            the consistency number wished:

            1 - Unconsisted
            2 - Consisted

            and statck it in a date/stage dataframe.

            in type: pd.DataFrame
            out type: pd.DataFrame

            '''
            dfc = df[df.NivelConsistencia == consistencia]
            if consistencia == 1:
                dfc = dfc[dfc.MediaDiaria == 1]
            dfc.index = range(len(dfc))
            dft = dfc.ix[:,16:47]
            dft.index = dfc.Data
```

```
            dft.columns = range(1,32)
            df = dft
            print(len(df))
            if len(df)!=0:
                vert = pd.DataFrame(index = pd.date_range(dft.index.min(),
                                                          add_months(dft.index.max(),1)))
                dfcol = pd.DataFrame(df.stack())
                dfcol.columns =  ['Data']
                dfcol.to_csv('temp.csv',sep='\t',header=['Cota'])
                vert.to_csv('temp2.csv',sep='\t')
                dateparse2 = lambda x: pd.datetime.strptime(x, '%Y-%m-%d')
                df = pd.read_csv('temp.csv',sep='\t', parse_dates=['Data'], date_parser=dateparse2)
                df.columns = ['Data', 'Dia', 'Cota']
                df.index = df.Data+ map(lambda x: timedelta(days=float(x) - 1.), df.Dia.tolist())
                vert['Data'] = vert.index
                df['Data'] = df.index
                finalmente = pd.merge(vert,df,on='Data')
                finalmente = vert.join(df,lsuffix='_l',rsuffix='_r')
                finalmente['Mes'] = finalmente.index.month
                finalmente['Dia'] = finalmente.index.dayofyear
                finalmente['Ano'] = finalmente.index.year
                return finalmente
            else:
                return pd.Series(np.nan)
```

```
In [5]: def merge_consistido(consistido,inconsistido):
            '''
            Merge consisted and unconsisted dataframes
            from stack_cota.

            in type: pd.DataFrame
            out type: pd.DataFrame

            '''
            if (any(consistido.notnull())) & (any(inconsistido.notnull())):
                frames = [consistido, inconsistido[inconsistido.Data_l>consistido.max().Data_l]]
                cotag = pd.concat(frames)
                cotag = cotag.sort_index()
                return cotag
            elif any(consistido.notnull()):
                cotag = consistido
                cotag = cotag.sort_index()
                return cotag
            elif any(inconsistido.notnull()):
                cotag = inconsistido
                cotag = cotag.sort_index()
                return cotag
            else:
                print('Error: Stage series is probably empty')
                kill()
```

```
In [6]: def pctil(cotg, percentil_,f,nr_p):
            '''
            Function auxiliary of NR_P;
            Receives the date\stage data (as of stack_cota()), the sought percentile (E.g 10),
```

```
    the number of years to the furure it whats the percentile, and nr_p dataframe.
    Calculates the percentile of the f to the future years for all years of cotg.

    input type: (pd.DataFrame, float, int, pd.DataFrame)
    output type: numpy array
    '''
    auxi = []
    for x in range(len(nr_p)):
        if cotg.Cota[(cotg.Ano>(nr_p[0][x]+f))&(cotg.Ano<=(nr_p[0][x]+f+1))].empty:
            auxi += [np.nan]
        else:
            auxi += [np.percentile(cotg.Cota[(cotg.Ano>(nr_p[0][x]+f))&
                                            (cotg.Ano<=(nr_p[0][x]+f+1))],percentil_)]
    return np.array(auxi)
```

```
In [7]: def pctilp(cotg, percentil_,nr_p):
            '''
            Function auxiliary of NR_P;
            Receives the date\stage data (as of stack_cota()), the sought percentile (E.g 10),
            and nr_p dataframe.
            Calculates the percentile to the past of all P values of nr_p.

            input type: (pd.DataFrame, float, pd.DataFrame)
            output type: numpy array
            '''
            auxi = []
            for x in range(len(nr_p)):
                if cotg.Cota[(cotg.Ano>nr_p[0][x]-nr_p['P'][x])&(cotg.Ano<=nr_p[0][x])].empty:
                    auxi += [np.nan]
                else:
                    auxi += [np.percentile(cotg.Cota[(cotg.Ano>nr_p[0][x]-nr_p['P'][x])&
                                                    (cotg.Ano<=nr_p[0][x])],percentil_)]
            return np.array(auxi)
```

```
In [8]: def NR_P(cotg_i,Pmax):
            '''
            Receives a data/stage dataframe (output of merge or stack_cota);

            Calculates the percentile of the range 1 to Pmax previous years
            (E.g 1 to 20 years) (p10);

            Calculates the percentile of the folowing 5 years (p10_1,p10_2,...,p10_5);

            Calculates the difference error from p10 to each p10_1,p10_2,...,p10_5
            (Erro1,...,Erro5);

            Calculates the error averages of Erro1, [Erro1, Erro2], ..., [Erro1,...,Erro5]
            (Erro_med_1,...,Erro_med_5);

            Returns dataframe columns (year,P,p10,p10_1,p10_2,p10_3,p10_4,p10_5,Erro1,Erro2
            Erro3,Erro4,Erro5,Erro_med_5,Erro_med_4,Erro_med_3,Erro_med_2,Erro_med_1)

            in type: (pd.DataFrame, integer)
            out type: pd.DataFrame
```

```
    '''
    nlinhas = Pmax *((cotg_i.index.year.max()-4) - (cotg_i.index.year.min()+Pmax-1))
    if nlinhas >0:
        nr_p = pd.DataFrame(np.zeros(nlinhas))
        lista_anos = []
        for i in range(cotg_i.index.year.min()+Pmax-1,cotg_i.index.year.max()-4):
            lista_anos += [i]*Pmax
        nr_p[0] = lista_anos
        P = np.array(range(Pmax) * len(range((cotg_i.index.year.min()+Pmax-1),
                                            (cotg_i.index.year.max()-4))))+1

        nr_p['P'] = P
        nr_p['p10'] = pctilp(cotg_i, 10,nr_p)
        nr_p['p10_1'] = pctil(cotg_i, 10, 0,nr_p)
        nr_p['p10_2'] = pctil(cotg_i, 10, 1,nr_p)
        nr_p['p10_3'] = pctil(cotg_i, 10, 2,nr_p)
        nr_p['p10_4'] = pctil(cotg_i, 10, 3,nr_p)
        nr_p['p10_5'] = pctil(cotg_i, 10, 4,nr_p)

        nr_p['Erro1'] = nr_p['p10'] - nr_p['p10_1']
        nr_p['Erro2'] = nr_p['p10'] - nr_p['p10_2']
        nr_p['Erro3'] = nr_p['p10'] - nr_p['p10_3']
        nr_p['Erro4'] = nr_p['p10'] - nr_p['p10_4']
        nr_p['Erro5'] = nr_p['p10'] - nr_p['p10_5']
        nr_p['Erro_med_5'] = nr_p.ix[:,8:13].T.abs().mean(skipna=0)
        nr_p['Erro_med_4'] = nr_p.ix[:,8:12].T.abs().mean(skipna=0)
        nr_p['Erro_med_3'] = nr_p.ix[:,8:11].T.abs().mean(skipna=0)
        nr_p['Erro_med_2'] = nr_p.ix[:,8:10].T.abs().mean(skipna=0)
        nr_p['Erro_med_1'] = nr_p.ix[:,8:9].abs()

    else: # To avoid cotg < Pmax years.
        nr_p = 0
        print("Try a smaller Pmax")
    return nr_p
```

```
In [9]: def var_erro_anual(cotg,nr_p,estacao,figuras,nome,Pmax):
    '''
    Receives the date/stage dataframe (cotg),
    the percentile and errors dataframe (nr_p),
    the station number and name for the title (estacao, nome),
    the address to save the output figure (figuras),
    and the maximum values of years to the past
    used for percentile calculations (Pmax).

    Built the plot that represents the errors distributions
    with different P values (1 to Pmax).

    input type: (pd.DataFrame, pd.DataFrame, int, str, str, int)
    output type: matplotlib figure and pdf file.

    '''

    rc('font',family='Sans-Serif',size=10)
    dados_anuais = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_1']
    dados_anuais2 = dados_anuais.dropna(subset = [range(1,Pmax)])
    left, width = 0.125, .7750
```

```
        bottom, height = 0.12, 0.55
        bottom_h = left_h = left+width+0.04
        lsup= (cotg.index.year.max()-4)
        linf= (cotg.index.year.min()+Pmax-1)
        nlinhas = Pmax *(lsup - linf)

        plt.figure(figsize(9,4.5))
        dados_anuais.index+=1
        sns.heatmap(dados_anuais[dados_anuais.notnull()].T[::-1],cmap="inferno",
                    xticklabels=5,cbar_kws={"orientation": "horizontal"})
        xlabel('Year')
        ylabel('P (years)')
        plt.yticks(rotation=0)
        plt.text(-4,-5,u'Avg.\nError\n(cm)',ha='center',va='center',rotation='vertical')

        nullfmt = NullFormatter()
        rect_histx = [left, bottom_h, width, 0.25] # dimensions of x-histogram
        axHistx = plt.axes(rect_histx) # x histogram
        axHistx.xaxis.set_major_formatter(nullfmt)
        axHistx.scatter(dados_anuais.index.tolist(),dados_anuais.T.idxmin().tolist(),
                        edgecolor=None,marker='x')
        axHistx.plot(dados_anuais.index.tolist(),dados_anuais.T.idxmin().tolist(),lw=0.3)

        axHistx.set_xlim(linf-0.5,lsup-.5)
        axHistx.set_ylim(0,20)#Pmax)
        axHistx.set_xticks(range(linf,lsup,2))
        axHistx.set_title(nome+'-'+str(estacao)+'-'+str(cotg.index.year.min())+
                          ':'+str(cotg.index.year.max())+
                          u'\n Error as function of P for 1 year prediction.')

        axHistx.set_ylabel( 'P (years)')

        left, width = 0.1, 0.81
        bottom, height = 0.34, 0.58
        bottom_h = left_h = left+width+0.02

        rect_histy = [left_h, bottom, 0.25, height]
        axHisty = plt.axes(rect_histy)

        axHisty.scatter(dados_anuais2.mean(skipna=0).tolist(),dados_anuais2.T.index.tolist()[::],
                        edgecolor=None,marker='x',label='Avg.',c='r')
        dados_anuais2.boxplot(vert=0,showmeans=0,ax=axHisty)

        startx, endx = axHisty.get_xlim()
        starty, endy = axHisty.get_ylim()

        axHisty.set_xlim(0,210)
        axHisty.set_yticks(range(int(starty)+1,int(endy)+1,1))
        axHisty.set_xlabel(u'Avg. Error Distribution (cm)')
        axHisty.legend(bbox_to_anchor=(.02, 1.1), loc=2, borderaxespad=0.)
        axHisty.set_ylim(0,21)#Pmax)
        savefig(figuras+str(estacao)+'_VariacoesAnuais_P_small_sides_1_eng.pdf',
                bbox_inches='tight')

In [10]: def Med_Erros(nr_p,Pmax):
             '''
```

```
    Receives the percentile and errors dataframe (nr_p),
    and the maximum values of years to the past used for
    percentile calculations (Pmax).

    Organizes the averagef errors (Erro_med_5,...,Erro_med_1)

    Built and return a new dataframe (med_erros).

    input type:( pd.DataFrame, int)
    output type: pd.DataFrame
    '''

    med_erros_grupos_5 = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_5']
    med_erros_grupos_5 = med_erros_grupos_5.dropna(subset =
                                    [range(1,Pmax)]).abs().mean(skipna=0)

    med_erros_grupos_4 = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_4']
    med_erros_grupos_4 = med_erros_grupos_4.dropna(subset =
                                    [range(1,Pmax)]).abs().mean(skipna=0)

    med_erros_grupos_3 = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_3']
    med_erros_grupos_3 = med_erros_grupos_3.dropna(subset =
                                    [range(1,Pmax)]).abs().mean(skipna=0)

    med_erros_grupos_2 = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_2']
    med_erros_grupos_2 = med_erros_grupos_2.dropna(subset =
                                    [range(1,Pmax)]).abs().mean(skipna=0)

    med_erros_grupos_1 = pd.pivot_table(nr_p,index=[0], columns='P')['Erro_med_1']
    med_erros_grupos_1 = med_erros_grupos_1.dropna(subset =
                                    [range(1,Pmax)]).abs().mean(skipna=0)

    frame = [med_erros_grupos_5,med_erros_grupos_4,med_erros_grupos_3,med_erros_grupos_2,
             med_erros_grupos_1]

    nomes = ['med5','med4','med3','med2','med1']
    med_erros = pd.DataFrame(frame,nomes).T

    return med_erros
```

```
In [11]: def Plot_Green(med_erros,estacao,figuras,nome):
    '''
    Receives the average errors dataframe (med_erros),
    the station number and name (estacao, nome) and
    the output address (figuras).

    Plot the average erros per year to the past and per year maintened.

    input type:( pd.DataFrame, int, str, str)
    output type: matplotlib figure and pdf file
    '''
    rc('font',family='Senasds Serif',size=10)
    sns.set_style("dark")
    sns.set(palette="colorblind")
    sns.set_context("paper")
    fig, ax = plt.subplots()
```

```
figsize(9,1.5)

med_erros.med5.plot(color='green',alpha=0.2,label='F = 5 Anos',lw=0.8)
plt.scatter(med_erros.med5.index,med_erros.med5,color='grey',label=None,s=10,marker='*')
legend(loc=2)

med_erros.med4.plot(color='green',alpha=0.4, label='F = 4 Anos',lw=1)
plt.scatter(med_erros.med4.index,med_erros.med4,color='grey',label=None,s=10,marker='*')
legend(loc=2)

med_erros.med3.plot(color='green',alpha=0.6, label='F = 3 Anos',lw=1.2)
plt.scatter(med_erros.med3.index,med_erros.med3,color='grey',label=None,s=10,marker='*')
legend(loc=2)

med_erros.med2.plot(color='green', label='F = 2 Anos',lw=0.8)
plt.scatter(med_erros.med2.index,med_erros.med2,color='grey',label=None,s=10,marker='*')
legend(loc=2)

med_erros.med1.plot(color='darkgreen', label='F = 1 Anos',lw=0.8,ls='-.')
plt.scatter(med_erros.med1.index,med_erros.med1,color='grey',label=None,s=10,marker='*')
legend(loc=2)

startx, endx = ax.get_xlim()
ax.xaxis.set_ticks(np.arange(startx, endx, 1))

starty, endy = ax.get_ylim()
ax.yaxis.set_ticks(np.arange(starty, endy, 5))


plt.plot([20,20],[starty,endy],c='r',lw=1,alpha=0.5)
plt.title(nome+' - '+str(estacao))

ax.set_ylim(starty, endy)
ylabel(u'Avg. Error (cm)')
xlabel('P (years)')
box = ax.get_position()
ax.set_position([box.x0, box.y0 + box.height * 0.1,
                 box.width, box.height * 0.9])

# Put a legend below current axis
ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.10),
      fancybox=True, shadow=True, ncol=5)

#fig.savefig(figuras+estacao+'_NR_Variacoes2_eng.png',bbox_inches='tight',dpi=1200)
```

Example

In [12]:
```
# Set a function to read datetime data.
dateparse = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')
```

In [13]:
```
# Set the number of the ANA code station wanted.
estacao = 66825000
```

```
In [14]: # Set the number of maximum year to the past used for percentile calc.
         Pmax = 20
```

```
In [15]: # Set root, figures and tables addresses.
         root = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Cotas\\'+str(estacao)
         figuras = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\figuras\\'
         tabelas = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\tabelas\\'
```

```
In [16]: # Read ANA's csv stage data.
         df = pd.read_csv(root+'\\COTAS.TXT',header=15,sep=';',
                          parse_dates=['Data'], date_parser=dateparse,decimal=',')
```

```
In [17]: # Print the brut dataframe for inspection.
         print(df.head(2))
         print('.')
         print('.')
         print('.')
         print(df.tail(2))
```

```
//EstacaoCodigo  NivelConsistencia       Data Hora  MediaDiaria  \
0         66825000                  1 1900-01-01  NaN            1
1         66825000                  1 1900-02-01  NaN            1

   TipoMedicaoCotas  Maxima  Minima Media  DiaMaxima     ...      \
0                 1   212.0   185.0   194       26.0     ...
1                 1   245.0   212.0   221       28.0     ...

   Cota23Status  Cota24Status  Cota25Status  Cota26Status  Cota27Status  \
0           1.0           1.0           1.0           1.0           1.0
1           1.0           1.0           1.0           1.0           1.0

   Cota28Status Cota29Status Cota30Status Cota31Status Unnamed: 78
0           1.0          1.0          1.0          1.0         NaN
1           1.0          0.0          0.0          0.0         NaN

[2 rows x 79 columns]
.
.
.
      //EstacaoCodigo  NivelConsistencia       Data Hora  MediaDiaria  \
3463         66825000                  2 2003-10-01  NaN            1
3464         66825000                  2 2003-11-01  NaN            1

      TipoMedicaoCotas  Maxima  Minima Media  DiaMaxima     ...      \
3463                 1   334.0   228.0   276        1.0     ...
3464                 1   230.0   175.0   199        1.0     ...

      Cota23Status  Cota24Status  Cota25Status  Cota26Status  Cota27Status  \
3463           1.0           1.0           1.0           1.0           1.0
3464           1.0           1.0           1.0           1.0           1.0

      Cota28Status Cota29Status Cota30Status Cota31Status Unnamed: 78
3463           1.0          1.0          1.0          1.0         NaN
3464           1.0          1.0          1.0          0.0         NaN
```

```
[2 rows x 79 columns]
```

```
In [18]: # Apply stack_cota to df.
         inconsistido = stack_cota(df,1)
         consistido = stack_cota(df,2)
```

```
1387
1247
```

```
In [19]: # Merge consisted and unconsisted dataframes.
         cotg = merge_consistido(consistido,inconsistido)
```

```
In [20]: # Print merged dateframe (cotg) for inspection.
         print(cotg.head(2))
         print('.')
         print('.')
         print('.')
         print(cotg.tail(2))
```

```
Data_l     Data_r  Dia   Cota  Mes   Ano
1900-01-01 1900-01-01 1900-01-01    1  186.0    1  1900
1900-01-02 1900-01-02 1900-01-02    2  185.0    1  1900
.
.
.
               Data_l     Data_r  Dia   Cota  Mes   Ano
2015-07-31 2015-07-31 2015-07-31  212  453.5    7  2015
2015-08-01 2015-08-01        NaT  213    NaN    8  2015
```

```
In [21]: # Apply NR_P function to all non null values of cotg.
         nr_p = NR_P(cotg[cotg.Cota.notnull()],Pmax)
```

```
In [22]: # Print resulting dataframe (nr_p) for inspection.
         print(nr_p.head(2))
         print('.')
         print('.')
         print('.')
         print(nr_p.tail(2))
```

```
0   P    p10  p10_1  p10_2  p10_3  p10_4  p10_5  Erro1  Erro2  Erro3  \
0  1919  1  152.0  300.0  192.8  126.4  164.0   59.0 -148.0  -40.8   25.6
1  1919  2  114.9  300.0  192.8  126.4  164.0   59.0 -185.1  -77.9  -11.5

   Erro4  Erro5  Erro_med_5  Erro_med_4  Erro_med_3  Erro_med_2  Erro_med_1
0  -12.0   93.0       63.88        56.6   71.466667        94.4       148.0
1  -49.1   55.9       75.90        80.9   91.500000       131.5       185.1
.
.
.
         0   P    p10  p10_1  p10_2  p10_3  p10_4   p10_5  Erro1  Erro2  \
1838  2010  19  140.0   83.4   97.0  110.0  153.2  218.05   56.6   43.0
1839  2010  20  143.0   83.4   97.0  110.0  153.2  218.05   59.6   46.0
```

```
      Erro3  Erro4  Erro5  Erro_med_5  Erro_med_4  Erro_med_3  Erro_med_2  \
1838   30.0  -13.2 -78.05       44.17        35.7        43.2        49.8
1839   33.0  -10.2 -75.05       44.77        37.2        46.2        52.8

      Erro_med_1
1838        56.6
1839        59.6
```

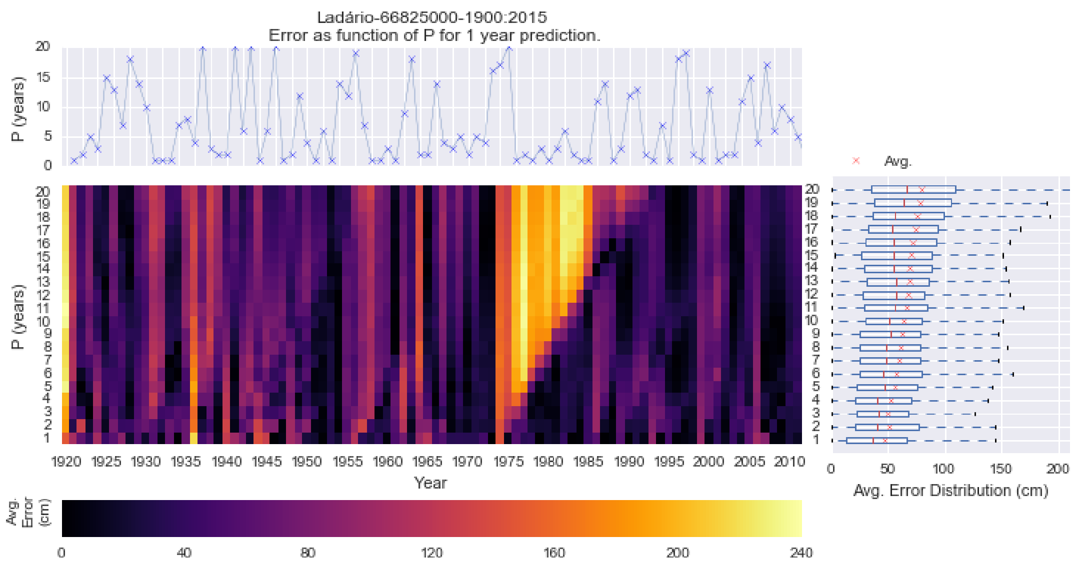
```
In [23]: # Apply Med_Erros function to nr_p values.
         med_erros = Med_Erros(nr_p,Pmax)
```

```
In [24]: # Print resulting dataframe (med_erros) for inspection.
         print(med_erros.head(2))
         print('.')
         print('.')
         print('.')
         print(med_erros.tail(2))
```

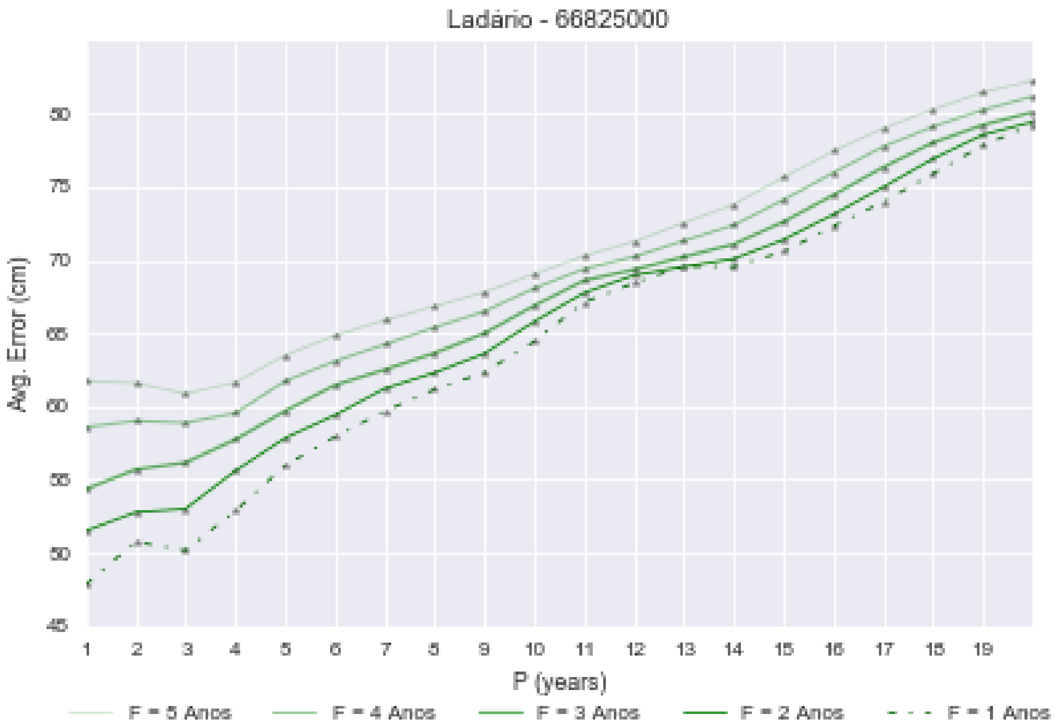
```
med5        med4        med3        med2        med1
P
1  61.736196  58.592391  54.343478  51.490217  47.834783
2  61.590870  59.008152  55.690036  52.771196  50.749457
.
.
.
         med5        med4        med3        med2        med1
P
19  81.457935  80.258696  79.232609  78.559783  77.834783
20  82.268043  81.180435  80.102717  79.442935  79.237500
```

```
In [25]: # Set station name.
         nome = u'Ladário'
```

```
In [26]: # Apply the var_erro_anual() funtion to build the error graph.
         var_erro_anual(cotg,nr_p,estacao,figuras,nome,Pmax)
```



```
In [27]: # Apply the Plot_Green() funtion to build the error summary graph.
         Plot_Green(med_erros,estacao,figuras,nome)
```



C.3 Autocorrelation Analysis Script

```
In [1]: import pandas as pd
        import seaborn as sns
        import os
        import matplotlib.ticker as ticker
        import calendar
        from datetime import timedelta
        import numpy as np
        import statsmodels.api as sm
```

```
In [2]: get_ipython().magic(u'pylab inline')
        pd.options.mode.chained_assignment = None
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [3]: # anastage: Library with the functions add_months,
        # stack_cota, merge_consistido seen in DRL_Error_Quantification.py
        from anastage import *
```

```
In [4]: # Set a function to read datetime data.
        dateparse = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')
        # Set the number of the ANA code station wanted.
        estacao = 66825000
        # Set root, figures and tables addresses.
        root = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Cotas\\'+str(estacao)
        figuras = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\figuras\\'
        tabelas = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\tabelas\\'
        # Read ANA's csv stage data.
        df = pd.read_csv(root+'\\COTAS.TXT',header=15,sep=';',
                         parse_dates=['Data'], date_parser=dateparse,decimal=',')
```

```
In [5]: # Apply stack_cota to df.
        inconsistido = stack_cota(df,1)
        consistido = stack_cota(df,2)
```

```
1387
1247
```

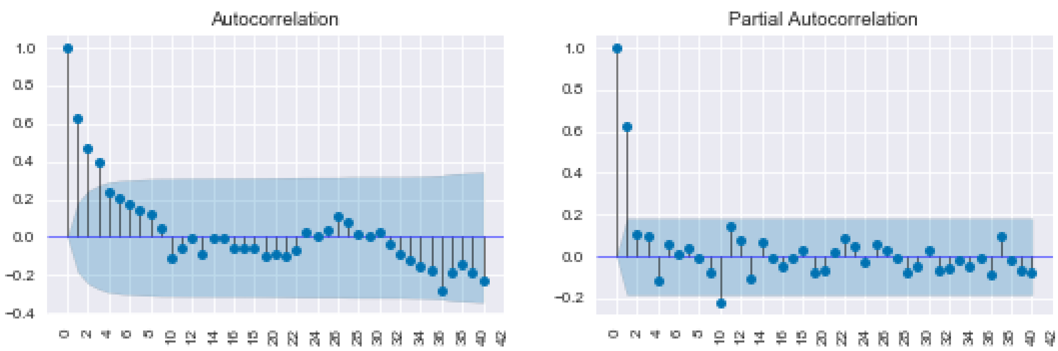
```
In [6]: # Merge consisted and unconsisted dataframes.
        cotg = merge_consistido(consistido,inconsistido)
```

```
In [7]: caa = pd.pivot_table(cotg,index=['Dia','Mes'],
                             columns='Ano')['Cota'].describe(percentiles
                                                              =[.1]).T['10%']
```

```
In [8]: fig1 = plt.figure(1,figsize(9,3.5))

        rc('font',family='Senasds Serif',size=10)
        sns.set_style("dark")
        sns.set(palette="colorblind")
        sns.set_context("paper")
```

```
fig = plt.figure(figsize=(9,2.5))
ax1 = fig.add_subplot(121)
plt.xticks(range(0,100,2),rotation='vertical')

fig = sm.graphics.tsa.plot_acf(caa.values.squeeze(),
                               lags=40, ax=ax1,lw=0.5,unbiased=True)
ax2 = fig.add_subplot(122)
plt.xticks(range(0,100,2),rotation='vertical')
fig = sm.graphics.tsa.plot_pacf(caa, lags=40, ax=ax2,lw=0.5)
savefig(figuras+str(estacao)+'_acf_pacf_eng.pdf',bbox_inches='tight')
```

```
<matplotlib.figure.Figure at 0x41c02b0>
```



C.4 Forecast Models Scripts

```
In [1]: import sys
        print(sys.version)
```

```
2.7.11 |Anaconda 2.3.0 (64-bit)| (default, Feb 16 2016, 09:58:36) [MSC v.1500 64 bit (AMD64)]
```

```
In [2]: from __future__ import print_function
        import numpy as np
        import matplotlib.pyplot as plt
        import statsmodels.tsa.arima_process as tsp
        from statsmodels.sandbox.tsa.fftarma import ArmaFft as FftArmaProcess
        import statsmodels.tsa.stattools as tss
        from statsmodels.graphics.tsaplots import plot_acf

        from math import *
        from scipy import *
        from cmath import *
        from numpy import *
        import numpy as np
        import fileinput
        import re
        import pandas as pd
        from scipy.interpolate import spline
        from sklearn.metrics import *
        #import seaborn as sns
        from datetime import timedelta
        import datetime
        import calendar
        import os
        import matplotlib.ticker as ticker
        #from wavelets import WaveletAnalysis
        #get_ipython().magic('matplotlib inline')
        #get_ipython().magic(u'pylab inline')
        pd.options.mode.chained_assignment = None
        %config InlineBackend.figure_format = 'svg'
```

```
In [3]: from optimizador_de_NR import *
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
WARNING: pylab import has clobbered these variables: ['gamma', 'test', 'polar']
`%matplotlib` prevents importing * from pylab and numpy
```

```
In [4]: def stack_cota(df,consistencia):
            """Recebe Dataframe de cotas da ANA (header=15) e retorna dataframe com cotas em colunas
            (Dia, Cota, Mes, Ano).Também recebe o Nível de Consistencia desejado
            (1 - Inconsistido, 2- Consistido)"""
            dfc = df[df.NivelConsistencia == consistencia]
            if consistencia == 1:
                dfc = dfc[dfc.MediaDiaria == 1]
            dfc.index = range(len(dfc))
            dft = dfc.ix[:,16:47]
            dft.index = dfc.Data
```

```
        dft.columns = range(1,32)
        df = dft
        print(len(df))
        if len(df)!=0:
            vert = pd.DataFrame(index = pd.date_range(dft.index.min(),add_months(dft.index.max(),1)))
            dfcol = pd.DataFrame(df.stack())
            dfcol.columns =  ['Data']
            dfcol.to_csv('temp.csv',sep='\t',header=['Cota'])
            vert.to_csv('temp2.csv',sep='\t')
            dateparse2 = lambda x: pd.datetime.strptime(x, '%Y-%m-%d')
            df = pd.read_csv('temp.csv',sep='\t', parse_dates=['Data'], date_parser=dateparse2)
            df.columns = ['Data', 'Dia', 'Cota']
            df.index = df.Data+ map(lambda x: timedelta(days=float(x)-1.), df.Dia.tolist())
            vert['Data'] = vert.index
            df['Data'] = df.index
            finalmente = pd.merge(vert,df,on='Data')
            finalmente = vert.join(df,lsuffix='_l',rsuffix='_r')
            finalmente['Mes'] = finalmente.index.month
            finalmente['Dia'] = finalmente.index.dayofyear
            finalmente['Ano'] = finalmente.index.year
            return finalmente
        else:
            return pd.Series(np.nan)
```

Here we define were pictures and tables should be saved.

```
In [5]: figuras = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\figuras\\'
        tabelas = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\tabelas\\'
```

Here we define the river station that we want.

```
In [6]: estacao = 66825000 #Ladario Station
```

This defines a function for Date reading, defines the origin of the data, read the data, copy then into 2 separate groups (Consisted and Inconsisted) data, apply the Stack_cota function for both groups and then merge them (optional).

```
In [7]: dateparse = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')
        root = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Cotas\\'+str(estacao)
        df = pd.read_csv(root+'\\COTAS.TXT',header=15,sep=';', parse_dates=['Data'], date_parser=dateparse,deci
        inconsistido = stack_cota(df,1)
        consistido = stack_cota(df,2)
        cotg = merge_consistido(consistido,inconsistido)
```

```
1395
1247
```

This rotine makes the years start in july. ex: july of 1990 will now be the first month of 1991. To accomplish that, it gives to all julys the value of the following year in the ano2 columns.

```
In [8]: cotg['new_year']=np.NAN
        cotg['Ano2']=np.NAN
```

```
cotg['Ano2'] = cotg['Ano']
cotg['Ano2'][cotg.index.month>6] = np.array(cotg.Ano[cotg.index.month>6].tolist()) +1
aux = cotg.Ano-1
cotg['new_year'] = aux.astype('str')+cotg.Ano.astype('str')
cotg['new_year'][cotg.index.month>6] = cotg.Ano[cotg.index.month>6].astype('str')+
                                        cotg.Ano2[cotg.index.month>6].astype('str')
```

Here, a dataframe with the annual percentiles description is created.

```
In [9]: resumo =  cotg[cotg.Cota.notnull()].pivot_table(values = 'Cota', index=['Dia','Mes'], columns='Ano2')
        r_stat = resumo.describe(percentiles=linspace(0,1,21)).T
```

```
In [10]: print(resumo.head(2))
```

```
Ano2      1900   1901   1902   1903   1904   1905   1906   1907   1908   1909  \
Dia Mes
1   1    186.0  191.0  168.0  141.0  240.0  298.0  190.0  150.0  168.0  210.0
2   1    185.0  193.0  172.0  140.0  240.0  300.0  190.0  151.0  168.0  210.0

Ano2     ...     2007   2008   2009   2010   2011  2012   2013   2014   2015  \
Dia Mes  ...
1   1    ...    243.0  196.0  108.0  164.0  100.0  82.0  124.0  117.5  222.5
2   1    ...    245.0  196.0  107.0  164.0  100.0  82.0  126.0  116.0  222.0

Ano2      2016
Dia Mes
1   1    145.5
2   1    146.5

[2 rows x 117 columns]
```

The following define the forecast parameters.

```
In [11]: T1Va=[40,1991,0,10]
         T1Ta=[40,1991,10,10]

         ###SECOND (BASIC - DIFERENT PERIODS)

         T2Va=[40,1996,0,5]
         T2Ta=[40,1996,5,10]


         ###THIRD (COMBINATION - 2 FORECAST EVENTS)

         T3Va=[40,1961,0,10]
         T3Ta=[40,1961,10,10]

         T3Vb=[40,1966,0,5]
         T3Tb=[40,1966,5,10]

         ###FOURTH (COMBINATION - 2 FORECAST EVENTS)
         T4Va=[40,1951,0,10]
         T4Ta=[40,1951,10,10]
```

```
T4Vb=[40,1956,0,5]
T4Tb=[40,1956,5,10]

###FIFTH (TESTE WITH COMPLEX VARIATION)

T5Va=[40,1941,0,10]
T5Ta=[40,1941,10,10]

T5Vb=[40,1946,0,5]
T5Tb=[40,1946,5,10]
```

In [12]:
```
parametros=[T1Va,
T1Ta,
T2Va,
T2Ta,
T3Va,
T3Ta,
T3Vb,
T3Tb,
T4Va,
T4Ta,
T4Vb,
T4Tb,
T5Va,
T5Ta,
T5Vb,
T5Tb]
```

For this analysis, only the parameters 0, 4, 8 and 12 were taken.

In [13]:
```
forecast_parameters = pd.DataFrame(pd.DataFrame(parametros).ix[[0,4,8,12],:].as_matrix())
```

Here the developed forecast_models module is imported

In [14]:
```
import foracast_models
```

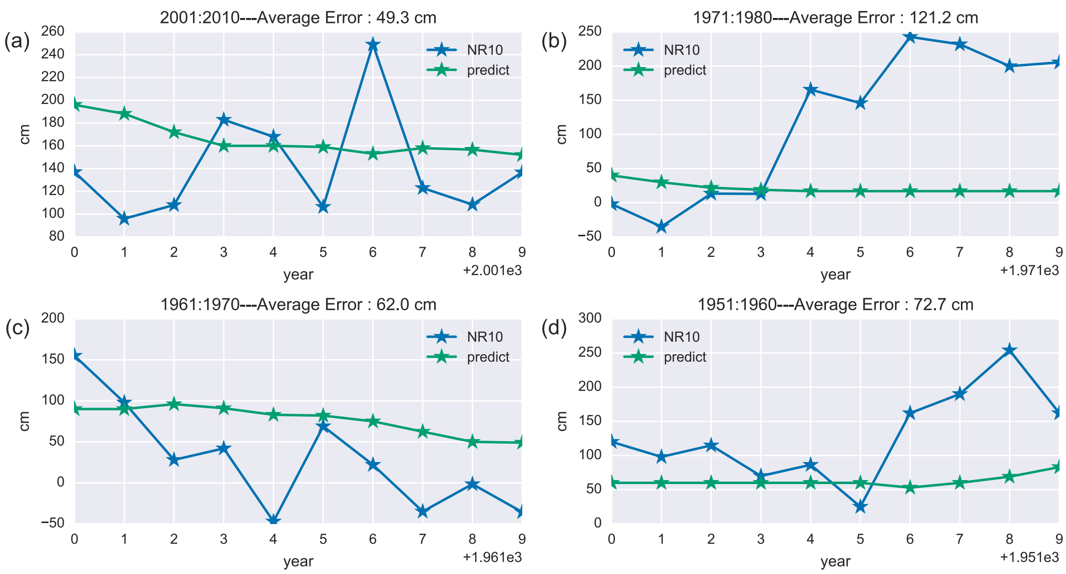
In [15]:
```
rc('font',family='Senasds Serif',size=10)
sns.set_style("darkgrid")
sns.set(palette="colorblind")
sns.set_context("paper")
```

C.4.1 Basic Model - Percentil 20

In [16]:
```
reload(foracast_models)

bm_20 = foracast_models.basic_forecast()

bm_20.set_parameters(forecast_parameters,20,r_stat,cotg)

bm_20.forecast(cotg)
```

In [17]:
```
grid_kws = {"height_ratios": (.5,.5), "hspace": (0.40)}
f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(10,5))
```

```
ax_t2 = [ax,ax1,ax2,ax3]
for i in range(len(forecast_parameters)):
    a = ax_t2[i]
    fig = plt.figure(figsize=(4,2))
    bm_20.test_results['BM_nr'][i].plot(marker='*',markersize=10,ax=ax_t2[i])
    bm_20.test_results['BM_Prediction'][i].plot(marker='*',markersize=10,ax=ax_t2[i])
    #bm_20.test_results['BM_Error'][i].plot(kind='bar')
    #plt.yticks(range(0,100,10))
    a.legend(loc='best')
    a.set_ylabel('cm')
    a.set_title(str(bm_20.forecast_parameters.ix[i:i,1:2].values[0][0]+
                    bm_20.forecast_parameters.ix[i:i,3:4].values[0][0])+
                    ':'+str(bm_20.forecast_parameters.ix[i:i,1:2].values[0][0]+
                    bm_20.forecast_parameters.ix[i:i,3:4].values[0][0]+
                    bm_20.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
                    '---Average Error : '+str(round(bm_20.test_results['BM_Error'][i].mean(),1))+' cm'
    #plt.show()
ax.text(-0.1, 1., '(a)', transform=ax.transAxes,
  fontsize=12, va='top', ha='right')
ax1.text(-0.1, 1., '(b)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='right')
ax2.text(-0.1, 1., '(c)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='right')
ax3.text(-0.1, 1., '(d)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='right')
f.savefig(figuras+'bm_20_teste_multiple.pdf',bbox_inches='tight')
```

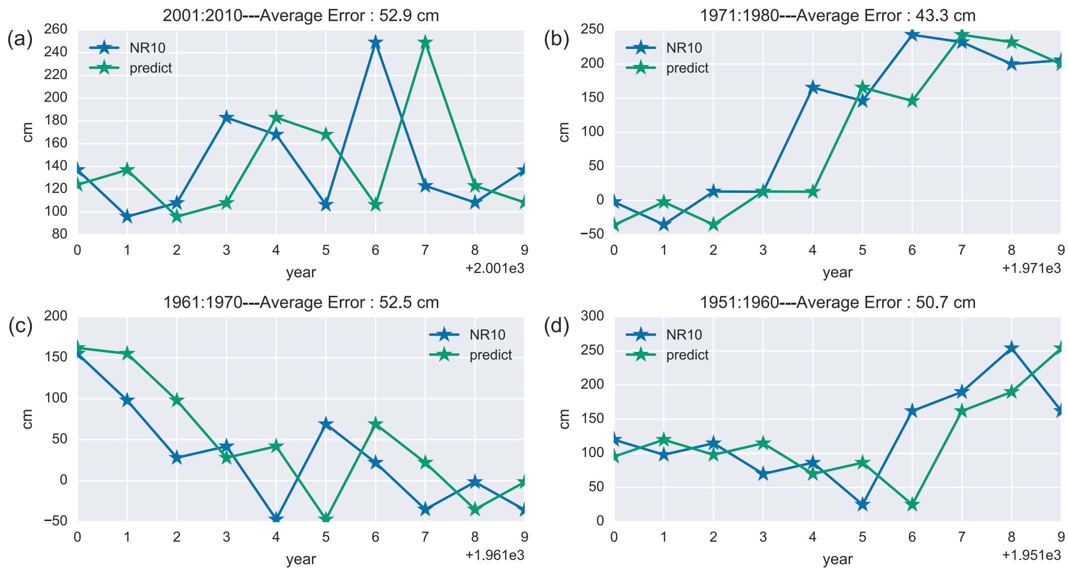


C.4.2 Basic Model - Percentil 1

In [22]:
```
reload(foracast_models)
# Create the forecast object
bm_01 = foracast_models.basic_forecast()
# Set the paramenters
```

```
bm_01.set_parameters(forecast_parameters,1,r_stat,cotg)
# Run the model
bm_01.forecast(cotg)
```

```
In [23]: #Plot the results
         grid_kws = {"height_ratios": (.5,.5), "hspace": (0.40)}
         f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(10,5))
         ax_t = [ax,ax1,ax2,ax3]
         for i in range(len(forecast_parameters)):
             a = ax_t[i]
             fig = plt.figure(figsize=(4,2))
             bm_01.test_results['BM_nr'][i].plot(marker='*',markersize=10,ax=ax_t[i])
             bm_01.test_results['BM_Prediction'][i].plot(marker='*',markersize=10,ax=ax_t[i])
             #bm_01.test_results['BM_Error'][i].plot(kind='bar')
             #plt.yticks(range(0,100,10))
             a.legend(loc='best')
             a.set_ylabel('cm')
             a.set_title(str(bm_01.forecast_parameters.ix[i:i,1:2].values[0][0]+
                             bm_01.forecast_parameters.ix[i:i,3:4].values[0][0])+
                       ':'+str(bm_01.forecast_parameters.ix[i:i,1:2].values[0][0]+
                               bm_01.forecast_parameters.ix[i:i,3:4].values[0][0]+
                       bm_01.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
                         '---Average Error : '+str(round(bm_01.test_results['BM_Error'][i].mean(),1))+' cm')
         ax.text(-0.1, 1., '(a)', transform=ax.transAxes,
           fontsize=12, va='top', ha='right')
         ax1.text(-0.1, 1., '(b)', transform=ax1.transAxes,
               fontsize=12, va='top', ha='right')
         ax2.text(-0.1, 1., '(c)', transform=ax2.transAxes,
           fontsize=12, va='top', ha='right')
         ax3.text(-0.1, 1., '(d)', transform=ax3.transAxes,
               fontsize=12, va='top', ha='right')
         f.savefig(figuras+'bm_1_teste_multiple.pdf',bbox_inches='tight')
             #plt.show()
```



C.4.3 AR(1)

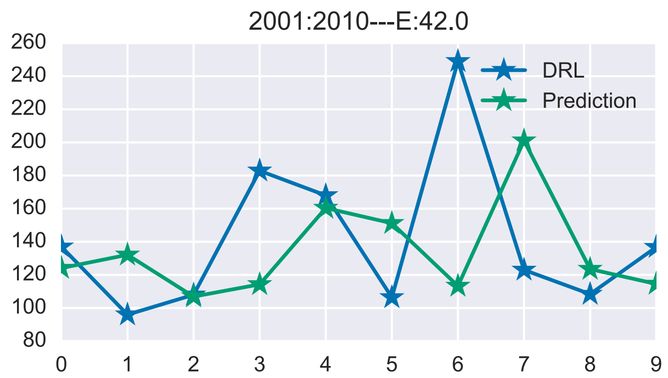
```
In [24]: reload(foracast_models)
```

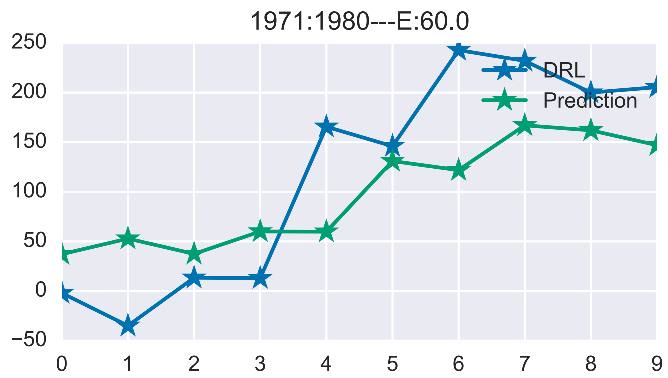
```
Out[24]: <module 'foracast_models' from 'foracast_models.pyc'>
```

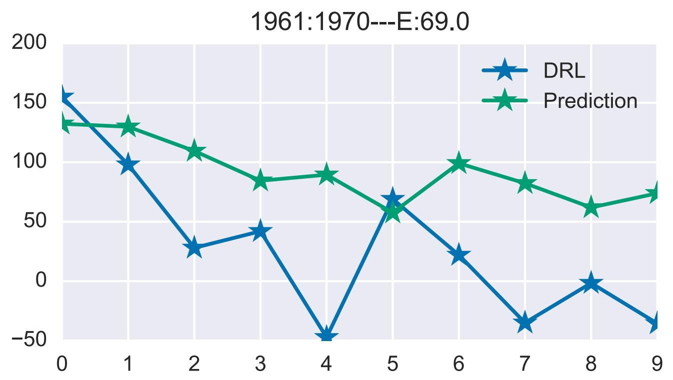
```
In [25]: # Create object
         ar_test_1 = foracast_models.auto_regression()
```

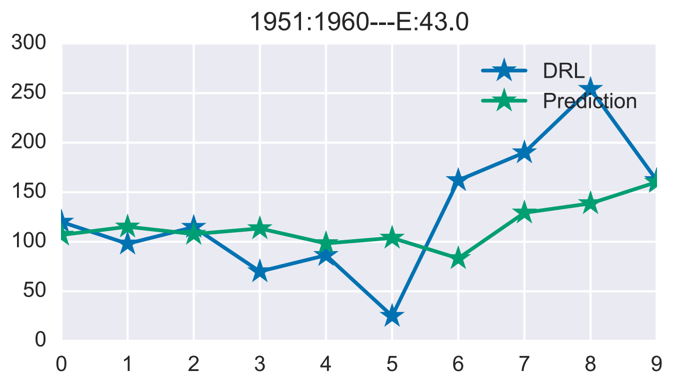
```
In [26]: # Set parameters
         ar_test_1.set_parameters(forecast_parameters,1,r_stat,cotg)
```

```
In [27]: # Run the model
         ar_test_1.forecast()
```



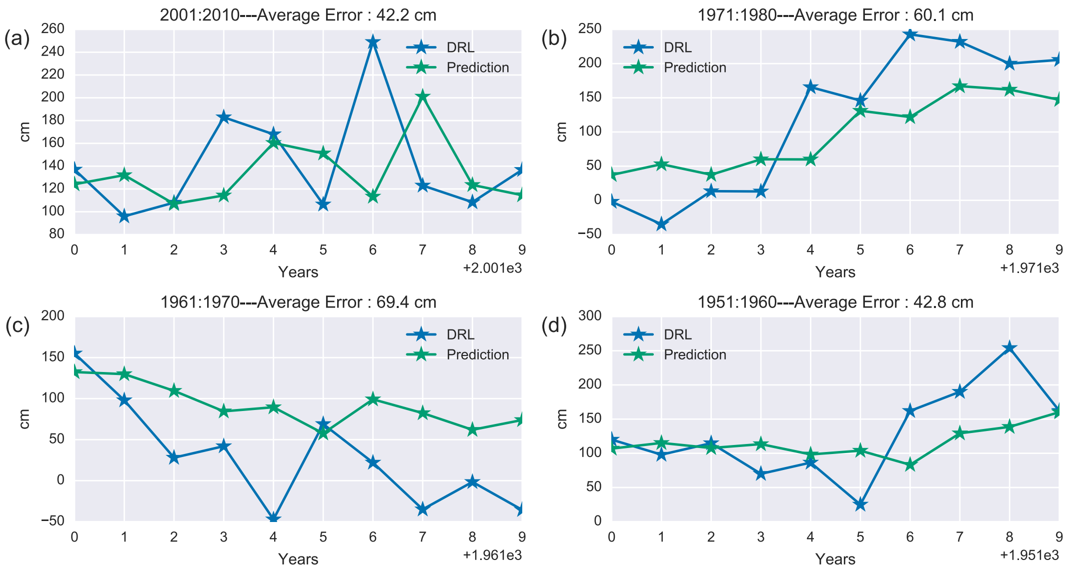






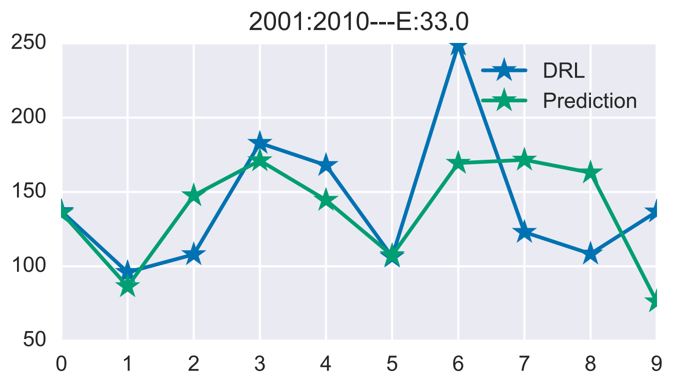

```
In [28]: # Plot the results
         grid_kws = {"height_ratios": (.5,.5), "hspace": (0.40)}
         f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(10,5))

         ax_t = [ax,ax1,ax2,ax3]
         for i in range(len(forecast_parameters)):
             a = ax_t[i]
             ar_test_1.test_results_summary[i].index = ar_test_1.test_results_summary[i].Years
             ar_test_1.test_results_summary[i]['DRL'].plot(marker='*',markersize=10,ax=ax_t[i])

             ar_test_1.test_results_summary[i]['Prediction'].plot(marker='*',markersize=10,ax=ax_t[i])
             #self.test_results['Error'].plot(kind='bar')
             #plt.yticks(range(0,100,10))
             a.legend(loc='best')
             a.set_ylabel('cm')
             a.set_title(str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                             ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0])+
                        ':'+str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                               ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]+
                        ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
                          '---Average Error : '+
                          str(round(ar_test_1.test_results_summary[i]['Error'].mean(),1))+' cm')
```

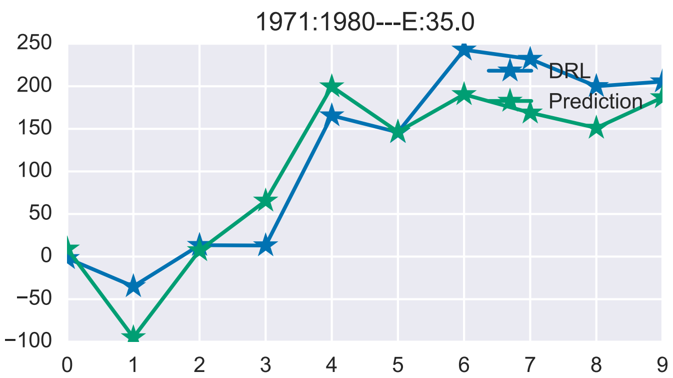
```
ax.text(-0.1, 1., '(a)', transform=ax.transAxes,
  fontsize=12, va='top', ha='right')
ax1.text(-0.1, 1., '(b)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='right')
ax2.text(-0.1, 1., '(c)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='right')
ax3.text(-0.1, 1., '(d)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='right')
f.savefig(figuras+'ar1_teste_multiple.pdf',bbox_inches='tight')
```

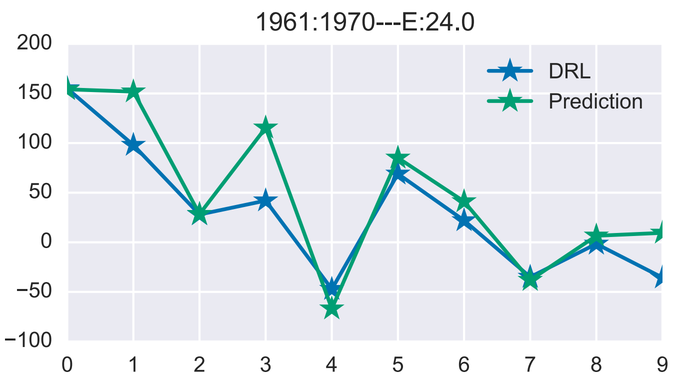


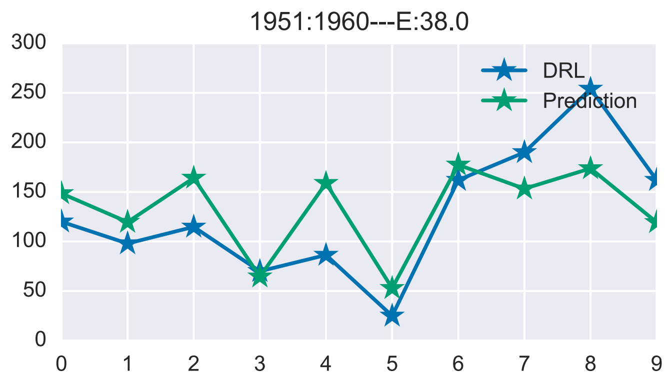
C.4.4 Multiple Regression

```
In [29]: reload(foracast_models)

Out[29]: <module 'foracast_models' from 'foracast_models.pyc'>

In [30]: # Create object
         mr1 =  foracast_models.multiple_regression()

In [31]: # Set the parameters
         mr1.set_parameters(forecast_parameters,1,r_stat,cotg,12)

In [32]: # Run the model
         mr1.forecast()
```


2001:2010---E:33.0
DRL
Prediction
250
200
150
100
50
0
1
2
3
4
5
6
7
8
9


1971:1980---E:35.0
DRL
Prediction
250
200
150
100
50
0
−50
−100
0
1
2
3
4
5
6
7
8
9

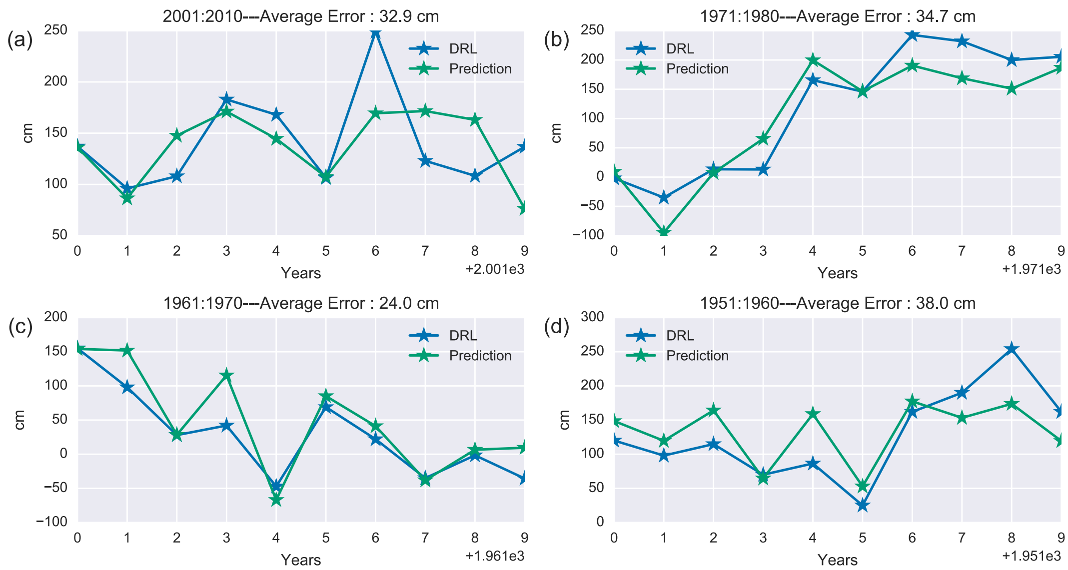

1961:1970---E:24.0
DRL
Prediction
200
150
100
50
0
−50
−100
0
1
2
3
4
5
6
7
8
9



In [33]: mr1.forecast_results.Error.mean()

Out[33]: 32.42261497444389

In [34]:
```
# Plot the results
grid_kws = {"height_ratios": (.5,.5), "hspace": (0.40)}
f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(10,5))
ax_t = [ax,ax1,ax2,ax3]
for i in range(len(forecast_parameters)):
    a = ax_t[i]
    mr1.test_results_summary[i].index = mr1.test_results_summary[i].Years
    mr1.test_results_summary[i]['DRL'].plot(marker='*',markersize=10,ax=ax_t[i])

    mr1.test_results_summary[i]['Prediction'].plot(marker='*',markersize=10,ax=ax_t[i])
    #self.test_results['Error'].plot(kind='bar')
    #plt.yticks(range(0,100,10))
    a.legend(loc='best')
    a.set_ylabel('cm')
    a.set_title(str(mr1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                    mr1.forecast_parameters.ix[i:i,3:4].values[0][0])+
               ':'+str(mr1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                    mr1.forecast_parameters.ix[i:i,3:4].values[0][0]+
               mr1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
                 '---Average Error : '+str(round(mr1.test_results_summary[i]['Error'].mean(),1))+' cm')
ax.text(-0.1, 1., '(a)', transform=ax.transAxes,
  fontsize=12, va='top', ha='right')
ax1.text(-0.1, 1., '(b)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='right')
ax2.text(-0.1, 1., '(c)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='right')
ax3.text(-0.1, 1., '(d)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='right')
f.savefig(figuras+'mr1_teste_multiple.pdf',bbox_inches='tight')
```



C.4.5 ANN

```
In [35]: # Read the models results the were stored localy
         models = []
         for i in range(1949,2010):
             models.append(pd.read_pickle('best_result'+str(i)+'.P')[0][0])
```

```
In [36]: # Extracts the error values for the tested periods
         def forecast_error(s_model):
             s_model.test_prediction2 = s_model.network.activate(s_model.test_values_x_norm.values[0])
             print(s_model.test_values_y_norm.index)
             ymax = s_model.training_values_y.max()
             ymin = s_model.training_values_y.min()
             ypredict = pd.DataFrame((np.array(s_model.test_prediction2)*(ymax-ymin))+ymin)
             yreal = pd.DataFrame((np.array(s_model.test_values_y_norm)*(ymax-ymin))+ymin)
             return([abs(yreal[0].values[0] -ypredict[0].values[0]),ypredict[0].values[0], yreal[0].values[0]])
```

```
In [37]: # Structure the results
         error_results= map(lambda x: forecast_error(models[x]),range(len(models)))
```

```
Int64Index([1951], dtype='int64', name=u'Ano2')
Int64Index([1952], dtype='int64', name=u'Ano2')
Int64Index([1953], dtype='int64', name=u'Ano2')
Int64Index([1954], dtype='int64', name=u'Ano2')
Int64Index([1955], dtype='int64', name=u'Ano2')
Int64Index([1956], dtype='int64', name=u'Ano2')
Int64Index([1957], dtype='int64', name=u'Ano2')
Int64Index([1958], dtype='int64', name=u'Ano2')
Int64Index([1959], dtype='int64', name=u'Ano2')
Int64Index([1960], dtype='int64', name=u'Ano2')
Int64Index([1961], dtype='int64', name=u'Ano2')
Int64Index([1962], dtype='int64', name=u'Ano2')
```

```
Int64Index([1963], dtype='int64', name=u'Ano2')
Int64Index([1964], dtype='int64', name=u'Ano2')
Int64Index([1965], dtype='int64', name=u'Ano2')
Int64Index([1966], dtype='int64', name=u'Ano2')
Int64Index([1967], dtype='int64', name=u'Ano2')
Int64Index([1968], dtype='int64', name=u'Ano2')
Int64Index([1969], dtype='int64', name=u'Ano2')
Int64Index([1970], dtype='int64', name=u'Ano2')
Int64Index([1971], dtype='int64', name=u'Ano2')
Int64Index([1972], dtype='int64', name=u'Ano2')
Int64Index([1973], dtype='int64', name=u'Ano2')
Int64Index([1974], dtype='int64', name=u'Ano2')
Int64Index([1975], dtype='int64', name=u'Ano2')
Int64Index([1976], dtype='int64', name=u'Ano2')
Int64Index([1977], dtype='int64', name=u'Ano2')
Int64Index([1978], dtype='int64', name=u'Ano2')
Int64Index([1979], dtype='int64', name=u'Ano2')
Int64Index([1980], dtype='int64', name=u'Ano2')
Int64Index([1981], dtype='int64', name=u'Ano2')
Int64Index([1982], dtype='int64', name=u'Ano2')
Int64Index([1983], dtype='int64', name=u'Ano2')
Int64Index([1984], dtype='int64', name=u'Ano2')
Int64Index([1985], dtype='int64', name=u'Ano2')
Int64Index([1986], dtype='int64', name=u'Ano2')
Int64Index([1987], dtype='int64', name=u'Ano2')
Int64Index([1988], dtype='int64', name=u'Ano2')
Int64Index([1989], dtype='int64', name=u'Ano2')
Int64Index([1990], dtype='int64', name=u'Ano2')
Int64Index([1991], dtype='int64', name=u'Ano2')
Int64Index([1992], dtype='int64', name=u'Ano2')
Int64Index([1993], dtype='int64', name=u'Ano2')
Int64Index([1994], dtype='int64', name=u'Ano2')
Int64Index([1995], dtype='int64', name=u'Ano2')
Int64Index([1996], dtype='int64', name=u'Ano2')
Int64Index([1997], dtype='int64', name=u'Ano2')
Int64Index([1998], dtype='int64', name=u'Ano2')
Int64Index([1999], dtype='int64', name=u'Ano2')
Int64Index([2000], dtype='int64', name=u'Ano2')
Int64Index([2001], dtype='int64', name=u'Ano2')
Int64Index([2002], dtype='int64', name=u'Ano2')
Int64Index([2003], dtype='int64', name=u'Ano2')
Int64Index([2004], dtype='int64', name=u'Ano2')
Int64Index([2005], dtype='int64', name=u'Ano2')
Int64Index([2006], dtype='int64', name=u'Ano2')
Int64Index([2007], dtype='int64', name=u'Ano2')
Int64Index([2008], dtype='int64', name=u'Ano2')
Int64Index([2009], dtype='int64', name=u'Ano2')
Int64Index([2010], dtype='int64', name=u'Ano2')
Int64Index([2011], dtype='int64', name=u'Ano2')
```

```
In [38]: error_results  = pd.DataFrame(error_results)
```

```
In [39]: error_results.columns = ['Error', 'Predict', 'Real']
```

```
In [40]: error_results.head(5)
```

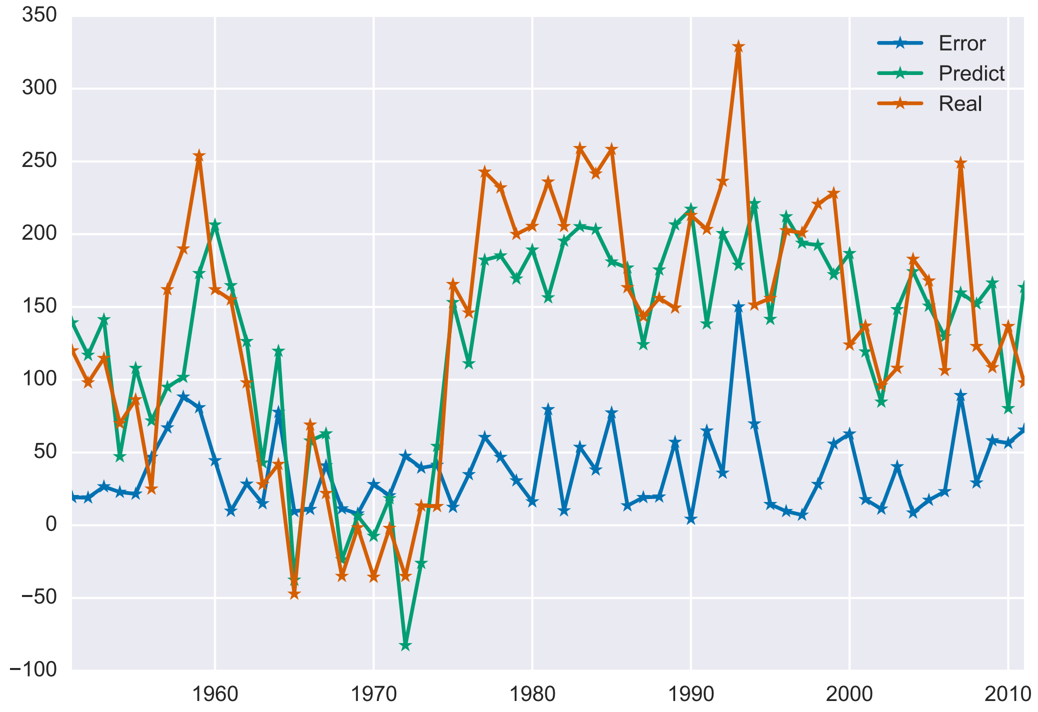
Out[40]:

	Error	Predict	Real
0	19.202509	139.202509	120.0
1	18.936566	116.936566	98.0
2	26.566223	141.366223	114.8
3	22.781741	47.218259	70.0
4	21.552307	107.952307	86.4

In [41]:
```
error_results.index = range(1951,2012)
```

In [42]:
```
#Plot the results
error_results.plot(marker='*')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0xeea00b8>



In [43]:
```
error_results.Error.mean()
```

Out[43]: 37.947928057954506

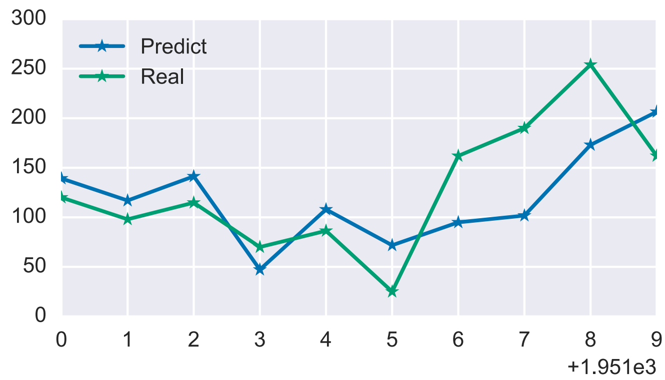
In [44]:
```
plt.figure(figsize(4,2))
error_results.ix[1951:1960,1:].plot(marker='*')
error_results.ix[1951:1960,:1].describe()
```

Out[44]:

	Error
count	10.000000
mean	43.653152
std	26.580620

```
min      18.936566
25%      21.859666
50%      35.526985
75%      61.988231
max      88.233623
```

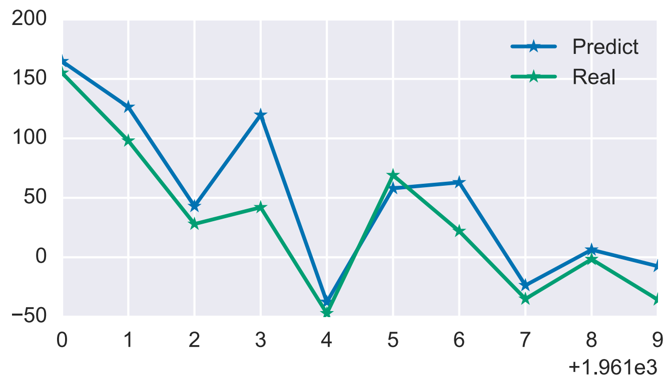
```
<matplotlib.figure.Figure at 0xe896518>
```




```
In [45]: plt.figure(figsize(4,2))
         error_results.ix[1961:1970,1:].plot(marker='*')
         error_results.ix[1961:1970,:1].describe()
```

```
Out[45]:            Error
         count  10.000000
         mean   23.979903
         std    21.825914
         min     7.921928
         25%    10.085332
         50%    13.169231
         75%    28.320725
         max    77.714865
```

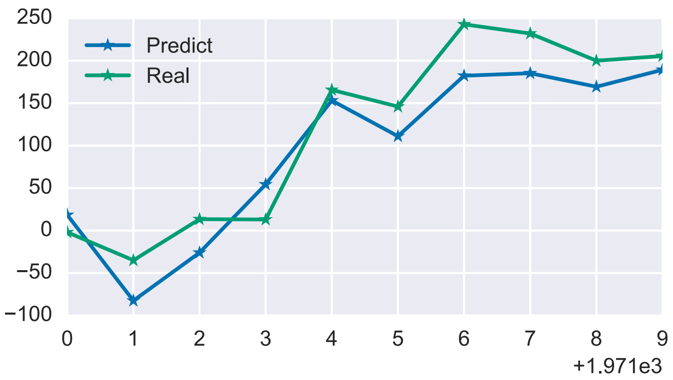
```
<matplotlib.figure.Figure at 0xf4e91d0>
```



```
In [46]: plt.figure(figsize(4,2))
         error_results.ix[1971:1980,1:].plot(marker='*')
         error_results.ix[1971:1980,:1].describe()
```

```
Out[46]:            Error
         count  10.000000
         mean   35.021891
         std    15.267889
         min    12.450043
         25%    22.957608
         50%    37.176476
         75%    45.405703
         max    60.485110
```

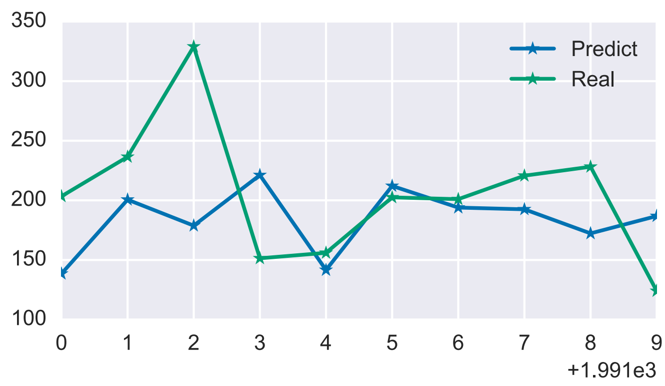
```
<matplotlib.figure.Figure at 0xf4fc6a0>
```




```
In [47]: plt.figure(figsize(4,2))
         error_results.ix[1991:2000,1:].plot(marker='*')
         error_results.ix[1991:2000,:1].describe()
```

```
Out[47]:             Error
         count   10.000000
         mean    49.882213
         std     42.519058
         min      7.066173
         25%     17.848499
         50%     45.911937
         75%     64.399160
         max    150.204034
```

```
<matplotlib.figure.Figure at 0x15a9fc88>
```

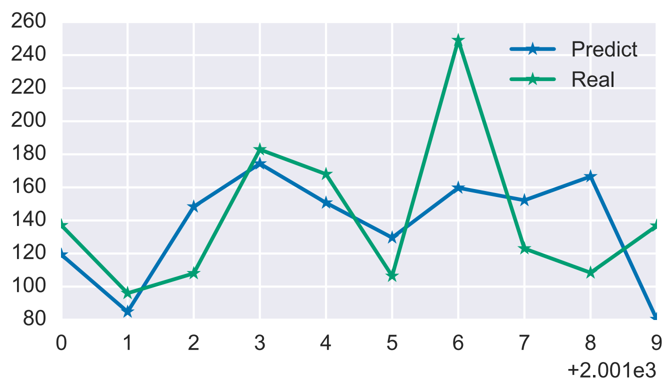



```
In [48]: plt.figure(figsize(4,2))
         error_results.ix[2001:2010,1:].plot(marker='*')
         error_results.ix[2001:2010,:1].describe()
```

Out[48]:

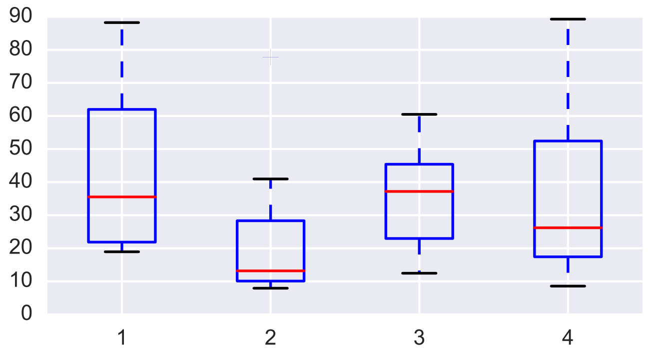
	Error
count	10.000000
mean	35.152071
std	25.864464
min	8.556163
25%	17.419322
50%	26.195472
75%	52.432578
max	89.280348

```
<matplotlib.figure.Figure at 0x29b7e6d8>
```




```
In [49]: a = error_results.ix[1951:1960,:1]
         b =error_results.ix[1961:1970,:1]
         c = error_results.ix[1971:1980,:1]
         d = error_results.ix[2001:2010,:1]
```

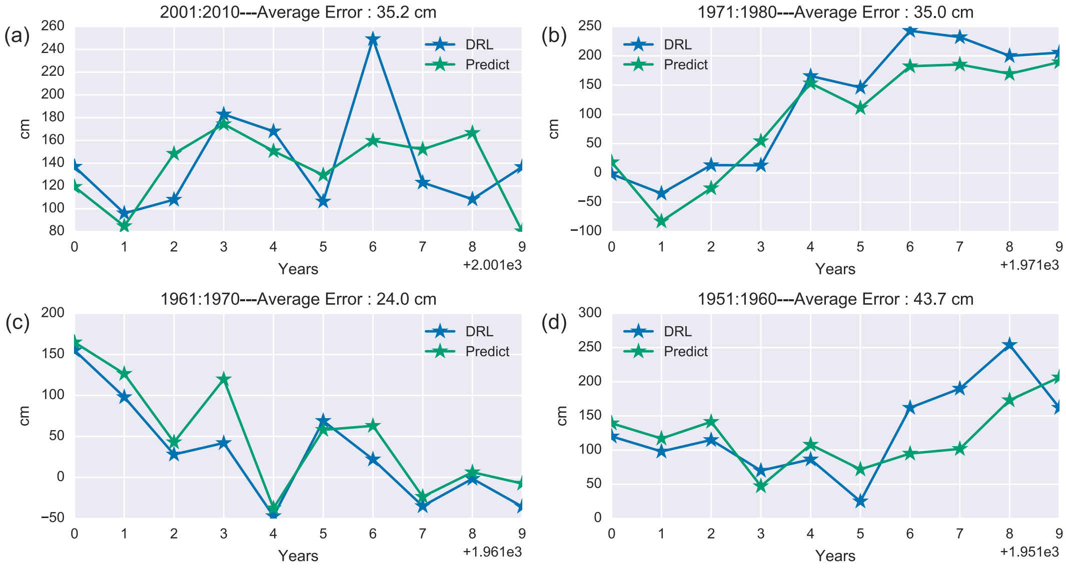
```
In [50]: a.values

Out[50]: array([[ 19.20250929],
                [ 18.93656586],
                [ 26.56622263],
                [ 22.78174124],
                [ 21.55230726],
                [ 46.80485262],
                [ 67.04935728],
                [ 88.23362252],
                [ 80.91659353],
                [ 44.48774681]])

In [51]: plt.boxplot([a.values,b.values,c.values,d.values])
         plt.show()
```




```
In [52]: error_results.columns = [u'Error', u'Predict', u'DRL']

In [53]: error_results.ix[2001:2010].Error.mean()

Out[53]: 35.152070925252204

In [54]: error_results = error_results[['Error','DRL','Predict']]

In [55]: grid_kws = {"height_ratios": (.5,.5), "hspace": (0.40)}
         f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(10,5))
         ax_t = [ax,ax1,ax2,ax3]


         error_results.ix[2001:2010,1:].plot(marker='*',markersize=10,ax=ax)
         error_results.ix[1971:1980,1:].plot(marker='*',markersize=10,ax=ax1)
         error_results.ix[1961:1970,1:].plot(marker='*',markersize=10,ax=ax2)
         error_results.ix[1951:1960,1:].plot(marker='*',markersize=10,ax=ax3)


             #self.test_results['Error'].plot(kind='bar')
             #plt.yticks(range(0,100,10))
```

```
ax.text(-0.1, 1., '(a)', transform=ax.transAxes,
  fontsize=12, va='top', ha='right')
ax1.text(-0.1, 1., '(b)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='right')
ax2.text(-0.1, 1., '(c)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='right')
ax3.text(-0.1, 1., '(d)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='right')

ax.legend(loc='best')
ax.set_ylabel('cm')
ax.set_xlabel('Years')
i = 0
ax.set_title(str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                 ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0])+
          ':'+str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]+
          ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
             '---Average Error : '+str(round(error_results.ix[2001:2010].Error.mean(),1))+' cm')

i = 1
ax1.set_title(str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0])+
          ':'+str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]+
          ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
              '---Average Error : '+str(round(error_results.ix[1971:1980].Error.mean(),1))+' cm')
ax1.legend(loc='best')
ax1.set_ylabel('cm')
ax1.set_xlabel('Years')


i = 2
ax2.set_title(str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0])+
          ':'+str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]+
          ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
              '---Average Error : '+str(round(error_results.ix[1961:1970].Error.mean(),1))+' cm')
ax2.legend(loc='best')
ax2.set_ylabel('cm')
ax2.set_xlabel('Years')


i = 3
ax3.set_title(str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0])+
          ':'+str(ar_test_1.forecast_parameters.ix[i:i,1:2].values[0][0]+
                  ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]+
          ar_test_1.forecast_parameters.ix[i:i,3:4].values[0][0]-1) +
              '---Average Error : '+str(round(error_results.ix[1951:1960].Error.mean(),1))+' cm')
ax3.legend(loc='best')
ax3.set_ylabel('cm')
ax3.set_xlabel('Years')
```

```
f.savefig(figuras+'ann_teste_multiple.pdf',bbox_inches='tight')
```

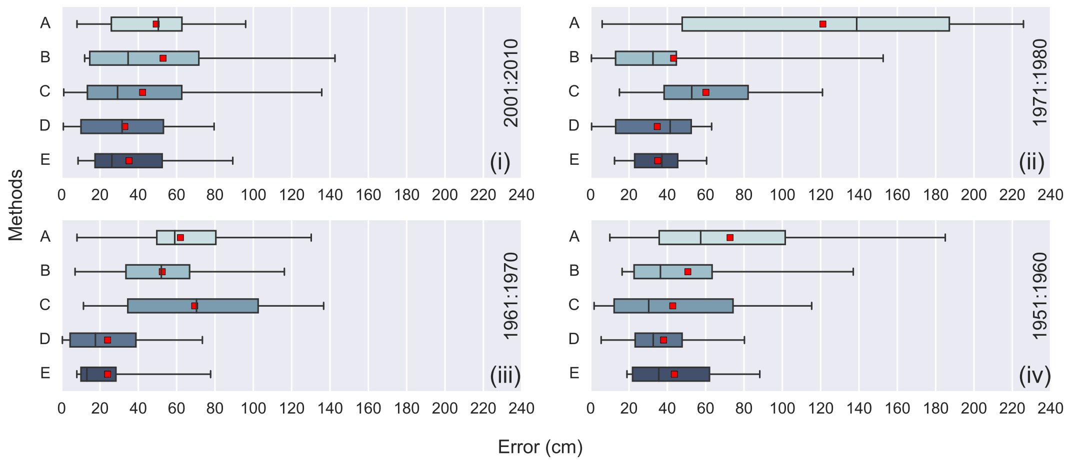


C.4.6 Error Summary

```
In [56]: from seaborn import boxplot
         sns.set_context("paper")
         a = [[2001,2010],[1971,1980],[1961,1970],[1951,1960]]
         grid_kws = {"height_ratios": (.5,.5), "hspace": (0.25), "wspace": (0.15)}
         f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(9,3.5))
         ax_t = [ax,ax1,ax2,ax3]
         for i in range(4):
             axis = ax_t[i]
             grafico=pd.DataFrame()

             grafico['A'] = bm20.BM_Error[i]
             grafico['B'] = bm_01.test_results.BM_Error[i]
             grafico['C'] = ar_test_1.test_results_summary[i].Error
             grafico['D'] = mr1.test_results_summary[i].Error
             grafico['E'] = error_results.ix[a[i][0]:a[i][1]].Error
             boxplot(grafico,fliersize=10,whis=200,width=0.4,linewidth=0.8,showmeans=False,
                     ax=ax_t[i],vert=0,color='Black',
                     palette=sns.cubehelix_palette(5,12,dark=0.3,rot=-.20))

             axis.set_title(str(a[i][0])+':'+str(a[i][1]))
             #ax2.legend(loc='best')
             #axis.set_xlabel('Error (cm)')
             #axis.set_ylabel('Methods')
             axis.set_xlim(0,200)
             axis.set_xticks(range(0,250,20))

             axis.scatter(grafico.mean(),[0,1,2,3,4],c='r',marker='s',s=10)

             title = str(a[i][0])+':'+ str(a[i][1])
```

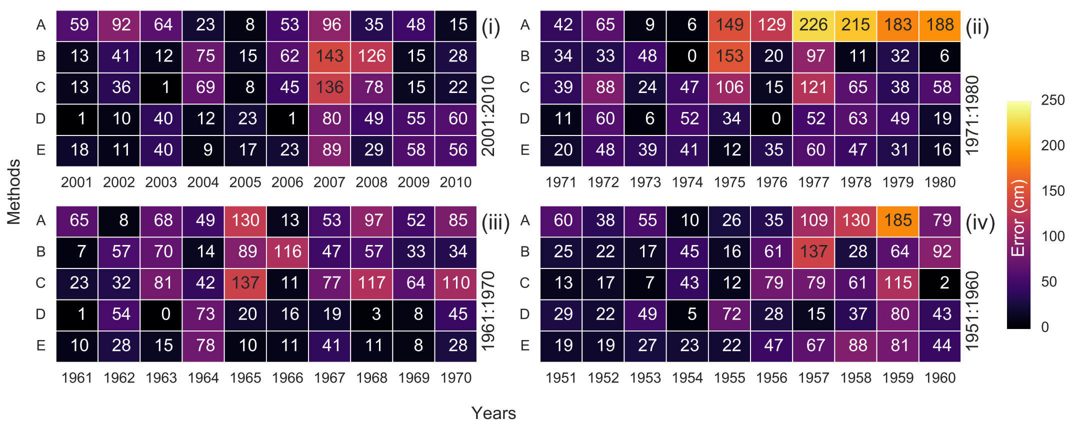
```
    axis.set_title(title,x=0.98,y=0.68,rotation='vertical')


f.text(0.09,0.5, 'Methods', ha='center', va='center',rotation='vertical')
f.text(0.5,0.01, 'Error (cm)', ha='center', va='center',rotation='horizontal')


ax.text(0.93, .15, '(i)', transform=ax.transAxes,
  fontsize=12, va='top', ha='left')
ax1.text(0.93, .15, '(ii)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='left')
ax2.text(0.93, .15, '(iii)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='left')
ax3.text(0.93, .15, '(iv)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='left')


f.savefig(figuras+'\\boxplota_2.pdf',bbox_inches='tight')
```

```
C:\Anaconda\lib\site-packages\seaborn\categorical.py:2125: UserWarning: The boxplot API has been changed. Attem
  warnings.warn(msg, UserWarning)
```




```
In [57]: sns.set_context("paper")
         #rc('font',family='Senasds Serif',size=10)
         a = [[2001,2010],[1971,1980],[1961,1970],[1951,1960]]
         grid_kws = {"height_ratios": (.5,.5), "hspace": (0.25), "wspace": (0.15)}
         f, ((ax,ax1),(ax2,ax3)) = plt.subplots(2,2, gridspec_kw=grid_kws,figsize=(9,3.5))
         ax_t = [ax,ax1,ax2,ax3]
         for i in range(4):
             axis = ax_t[i]
             grafico=pd.DataFrame()

             grafico['A'] = bm20.BM_Error[i]
             grafico['B'] = bm_01.test_results.BM_Error[i]
             grafico['C'] = ar_test_1.test_results_summary[i].Error
             grafico['D'] = mr1.test_results_summary[i].Error
```

```
    grafico['E'] = error_results.ix[a[i][0]:a[i][1]].Error
    grafico.index =grafico.index.map(int)


    cbar_ax = f.add_axes([.94, .2, .02,0.5])
    title = str(a[i][0])+':'+ str(a[i][1])
    axis.set_title(title,x=1.03,y=0.45,rotation='vertical')
    sns.heatmap(grafico.T,ax=axis,annot=True,cmap='inferno',cbar=i/3. == 0,
                fmt='3.0f',vmin=0, vmax=250,linewidths=.5,square=0,cbar_ax=cbar_ax)
    axis.set_xlabel('')
    axis.set_yticklabels(axis.yaxis.get_majorticklabels(), rotation=0)


ax.text(1.01, .95, '(i)', transform=ax.transAxes,
  fontsize=12, va='top', ha='left')
ax1.text(1.01, .95, '(ii)', transform=ax1.transAxes,
      fontsize=12, va='top', ha='left')
ax2.text(1.01, .95, '(iii)', transform=ax2.transAxes,
  fontsize=12, va='top', ha='left')
ax3.text(1.01, .95, '(iv)', transform=ax3.transAxes,
      fontsize=12, va='top', ha='left')

f.text(0.09,0.5, 'Methods', ha='center', va='center',rotation='vertical')
f.text(0.5,0.01, 'Years', ha='center', va='center',rotation='horizontal')

plt.text(0.15,0.6,'Error (cm)',rotation=90,color='white')
f.savefig(figuras+'\\error_2.pdf',bbox_inches='tight')
```




```
In [58]: bm_01.test_results.BM_Error[1].describe()
```

```
Out[58]: count     10.000000
         mean      43.270000
         std       47.199107
         min        0.400000
         25%       13.000000
         50%       32.500000
         75%       44.700000
```

```
max      152.600000
Name: erro, dtype: float64
```

```
In [61]: A = []
         B = []
         C = []
         D = []
         E = []
```

```
In [62]: # Erro médio BM20
         map(lambda x: A.append(bm20.BM_Error[x].tolist()),range(4))

         # Erro médio BM01
         map(lambda x: B.append(bm_01.test_results.BM_Error[x].tolist()),range(4))

         # Erro médio AR01
         map(lambda x: C.append(ar_test_1.test_results_summary[x].Error.tolist()),range(4))

         # Erro médio MR
         map(lambda x:  D.append(mr1.test_results_summary[x].Error.tolist()),range(4))

         # Erro médio ARR
         map(lambda x:  E.append(error_results.ix[a[x][0]:a[x][1]].Error.tolist()),range(4))
```

```
Out[62]: [None, None, None, None]
```

```
In [63]: resumo = pd.DataFrame()

         resumo['A'] = pd.DataFrame(np.array(A).reshape(40)).describe()[0]
         resumo['B'] = pd.DataFrame(np.array(B).reshape(40)).describe()[0]
         resumo['C'] = pd.DataFrame(np.array(C).reshape(40)).describe()[0]
         resumo['D'] = pd.DataFrame(np.array(D).reshape(40)).describe()[0]
         resumo['E'] = pd.DataFrame(np.array(E).reshape(40)).describe()[0]

         #resumo.index = pd.DataFrame(np.array(A).reshape(40)).describe().index
```

```
In [64]: resumo.round(1)
```

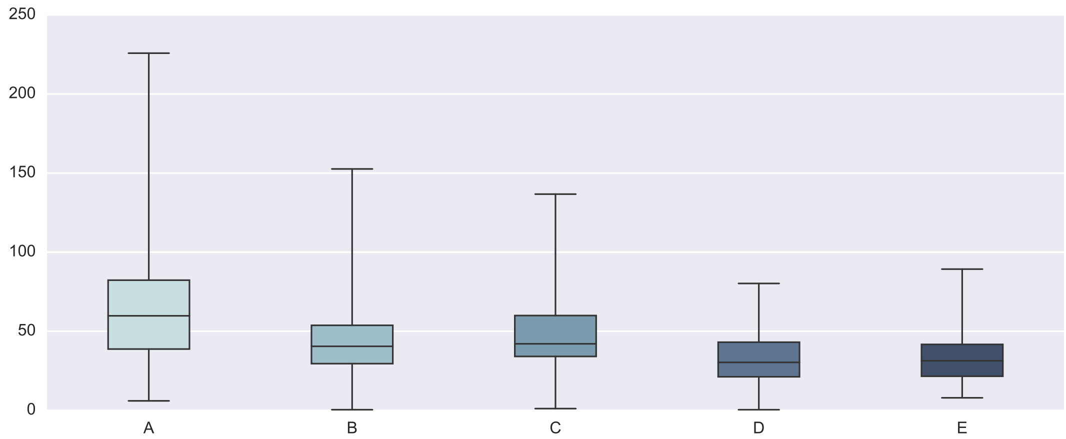
Out[64]:

	A	B	C	D	E
count	40.0	40.0	40.0	40.0	40.0
mean	76.3	49.8	53.6	32.4	34.5
std	60.0	40.9	39.9	24.7	23.0
min	6.0	0.4	1.1	0.4	7.9
25%	35.0	16.7	16.6	10.8	17.0
50%	59.5	33.8	44.1	28.2	28.3
75%	100.3	65.5	78.8	52.3	46.8
max	225.8	152.6	136.7	80.2	89.3

```
In [66]: resumo.round(1).to_latex(tabelas+'resumo_forecast_methods.tex')
```

```
In [67]: f =  plt.subplots(figsize=(9,3.5))
         boxplot(resumo,fliersize=10,whis=200,width=0.4,linewidth=0.8,showmeans=False,vert=1,
                 color='Black', palette=sns.cubehelix_palette(5,12,dark=0.3,rot=-.20))
```

Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x29d1dc50>



C.5 Water level linear model volume calculation example

```
In [1]: import pandas as pd
        import seaborn as sns
        import os
        import matplotlib.ticker as ticker
        import calendar
        from datetime import timedelta
        import numpy as np
        import statsmodels.api as sm
```

```
In [2]: get_ipython().magic(u'pylab inline')
        pd.options.mode.chained_assignment = None
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [3]: # anastage: Library with the functions add_months,
        # stack_cota, merge_consistido seen in DRL_Error_Quantification.py
        from anastage import *

        # Set a function to read datetime data.
        dateparse = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')


        def get_df(estacao):
            root = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Cotas\\'+str(estacao)
            df = pd.read_csv(root+'\\COTAS.TXT',header=15,sep=';',
                             parse_dates=['Data'], date_parser=dateparse,decimal=',')
            # Apply stack_cota to df.
            inconsistido = stack_cota(df,1)
            consistido = stack_cota(df,2)
            cotg = merge_consistido(consistido,inconsistido)
            return cotg

        figuras = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\figuras\\'
        tabelas = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Artigos\\PIANC\\RN\\Dados\\Saida\\tabelas\\'
        # Set the number of the ANA code station wanted.
        estacao1 = 66825000
        estacao2 = 66895000
        estacao3 = 66960008
        estacao4 = 66970000


        cotg1 = get_df(estacao1)
        cotg2 = get_df(estacao2)
        cotg3 = get_df(estacao3)
        cotg4 = get_df(estacao4)
```

```
1395
1247
491
391
```

```
C:\Anaconda\lib\site-packages\numpy\core\_methods.py:26: RuntimeWarning: tp_compare didn't return -1 or -2 for e
  return umr_maximum(a, axis, None, out, keepdims)
```

```
603
217
607
474
```

```
In [4]: reads = []
        reads.append(cotg1[(cotg1.Ano==2014)&(cotg1.Mes==11)&(cotg1.index.day==28)].Cota[0])
        reads.append(cotg2[(cotg2.Ano==2014)&(cotg2.Mes==11)&(cotg2.index.day==28)].Cota[0])
        reads.append(cotg3[(cotg3.Ano==2014)&(cotg3.Mes==11)&(cotg3.index.day==28)].Cota[0])
        reads.append(cotg4[(cotg4.Ano==2014)&(cotg4.Mes==11)&(cotg4.index.day==28)].Cota[0])
```

```
In [5]: reads
```

```
Out[5]: [242.0, 475.0, 247.5, 278.0]
```

C.5.1 Official BRL values of Ladário, Manga and Porto Esperança.

```
In [6]: root2 = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Volumetric Calculations\\'
```

```
In [7]: NR_ladario =2.02
        NR_manga = 4.41
        NR_esperanca = 1.17
        NR_coimbra = 1.34
```

```
In [8]: st = pd.read_csv(root2+'Entrada\\stations.csv',sep='\t')
        st
```

```
Out[8]:     Station Name        ID             N              E Drainage Area   BRL  \
        0        Ladário  66825000  7.899071e+06  438606.896500       253,000  2.02
        1    Porto Manga  66895000  7.871494e+06  475482.050000       327,000  4.41
        2  Pto Esperança  66960008  7.832724e+06  454560.830000       371,000  1.77
        3  Forte Coimbra  66970000  7.797546e+06  418017.735563           NaN  1.34

          Period of Calculation  Missing Data
        0       1900-01 2015-08          0.06
        1       1969-05 2015-09         17.71
        2       1963-12 2015-09          4.47
        3                   NaN           NaN
```

```
In [9]: st['reads'] = reads
```

```
In [10]: st['Var'] = st.BRL*100 - st.reads
```

```
In [11]: st
```

```
Out[11]:     Station Name        ID             N              E Drainage Area   BRL  \
         0        Ladário  66825000  7.899071e+06  438606.896500       253,000  2.02
         1    Porto Manga  66895000  7.871494e+06  475482.050000       327,000  4.41
         2  Pto Esperança  66960008  7.832724e+06  454560.830000       371,000  1.77
         3  Forte Coimbra  66970000  7.797546e+06  418017.735563           NaN  1.34
```

```
  Period of Calculation  Missing Data  reads    Var
0       1900-01 2015-08          0.06  242.0  -40.0
1       1969-05 2015-09         17.71  475.0  -34.0
2       1963-12 2015-09          4.47  247.5  -70.5
3                   NaN           NaN  278.0 -144.0
```

```
In [12]: rn = pd.read_csv(root2+'Entrada\\RNS_Perfil_corrigido.csv',sep='\t')
         rn
```

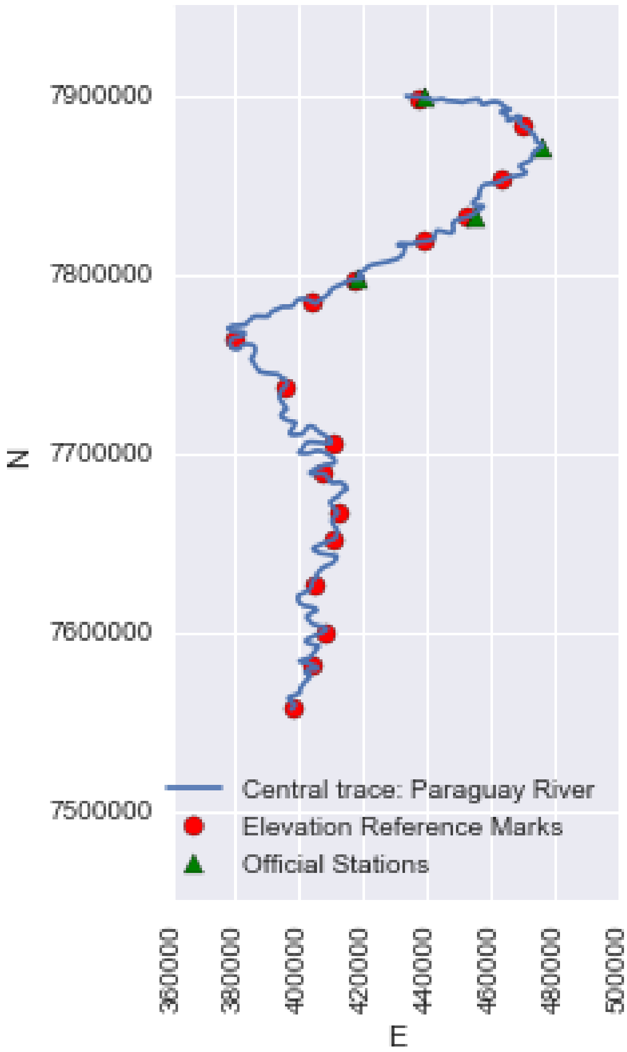
```
Out[12]:          ID_PTO              E             N   HONA                 DATA  \
         0   AUX LÁDARIO  437401.197562  7.898821e+06  84.91  2015-01-08 13:53:00
         1           21A  470315.346338  7.883059e+06  83.53  2015-01-10 14:00:00
         2           22A  463196.548287  7.852557e+06  82.40  2015-01-11 07:34:00
         3         EG-MT  452567.169338  7.831768e+06  81.58  2015-01-11 09:47:00
         4           24A  438741.946866  7.818247e+06  81.04  2015-01-11 09:47:00
         5           25A  417151.589068  7.797112e+06  80.55  2015-01-12 08:20:00
         6           26A  403802.197192  7.784611e+06  79.55  2015-01-12 09:44:00
         7           27A  379905.691593  7.764325e+06  78.18  2015-01-13 12:05:29
         8           28A  395085.869812  7.736215e+06  77.99  2015-01-14 11:13:59
         9           29A  410195.112291  7.705711e+06  77.40  2015-01-15 11:13:29
         10          30A  406911.543682  7.688514e+06  77.05  2015-01-16 11:26:59
         11          31A  412279.833071  7.666674e+06  76.66  2015-01-17 07:24:29
         12          32A  410179.194606  7.651107e+06  76.38  2015-01-18 08:17:14
         13          33A  404288.143665  7.626883e+06  75.63  2015-01-18 09:18:44
         14          34A  407765.287102  7.599972e+06  75.48  2015-01-19 10:11:29
         15          35A  403929.568010  7.581667e+06  74.44  2015-01-20 09:07:14
         16          36A  397761.701895  7.556918e+06  74.27  2015-01-20 10:06:29

              Indice           Dist        A-R
         0     203.0    4079.898305  84.750000
         1    2930.0   58616.982996  83.355830
         2    4960.0   99215.494555  82.215064
         3    6469.0  129394.593600  81.387061
         4    7714.0  154313.968525  80.848362
         5    9731.0  194652.297776  80.350398
         6   10753.0  215091.789713  79.369901
         7   13057.0  261189.914884  78.037524
         8   15317.0  306388.695126  77.886148
         9   17856.0  357187.349975  77.355975
         10  20049.0  401045.356079  77.051727
         11  21517.0  430404.046435  76.698966
         12  22358.0  447222.777124  76.428841
         13  24084.0  481741.816388  75.718010
         14  25900.0  518080.496579  75.610000
         15  27322.0  546506.587076  74.570000
         16  28958.0  579225.582136  74.400000
```

```
In [13]: tr = pd.read_pickle(root2+'tracado_total.p')
```

```
In [14]: plt.figure(figsize(3,6))

         plt.scatter(rn.E,rn.N,c='r',label='Elevation Reference Marks',s=50)
         #plt.scatter(stations.E,stations.N,marker='^',s=50,c='y',label='Official Stations')
         plt.scatter(st.E,st.N,marker='^',s=50,c='g',label='Official Stations')
         plt.plot(tr.E,tr.N,label='Central trace: Paraguay River')
```

```
xticks(rotation=90)
xlabel('E')
ylabel('N')
ylim(7450000,7950000)
legend(loc=4)
```

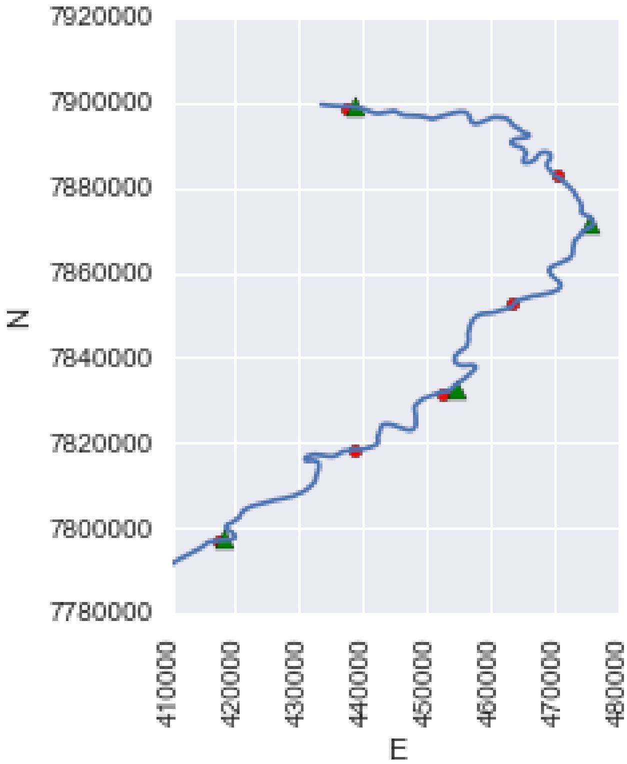
Out[14]: <matplotlib.legend.Legend at 0xd8c1c18>



In [15]:
```
plt.figure(figsize(3,4))

plt.scatter(rn.E,rn.N,c='r',label='Elevation Reference Marks')
#plt.scatter(brl.Er,brl.Nr,marker='^',s=50,c='y',label='Official Stations')
plt.scatter(st.E,st.N,marker='^',s=50,c='g',label='Official Stations')
plt.plot(tr.E,tr.N,label='Central trace: Paraguay River')
xticks(rotation=90)
xlabel('E')
ylabel('N')
ylim(7780000,7920000)
```

```
xlim(410000,480000)
#legend(loc=4)
```

Out[15]: (410000, 480000)



C.5.2 Stations dist(s)

In [16]:
```
from scipy.spatial.distance import *
```

In [17]:
```
# Calculate distances from every trace point to stations
dist_aux_ladario = cdist(tr.ix[:,1:3].as_matrix(), [[st.E[0],st.N[0]]], 'euclidean')
dist_aux_manga = cdist(tr.ix[:,1:3].as_matrix(), [[st.E[1],st.N[1]]], 'euclidean')
dist_aux_esperanca = cdist(tr.ix[:,1:3].as_matrix(), [[st.E[2],st.N[2]]], 'euclidean')
dist_aux_coimbra = cdist(tr.ix[:,1:3].as_matrix(), [[st.E[3],st.N[3]]], 'euclidean')
```

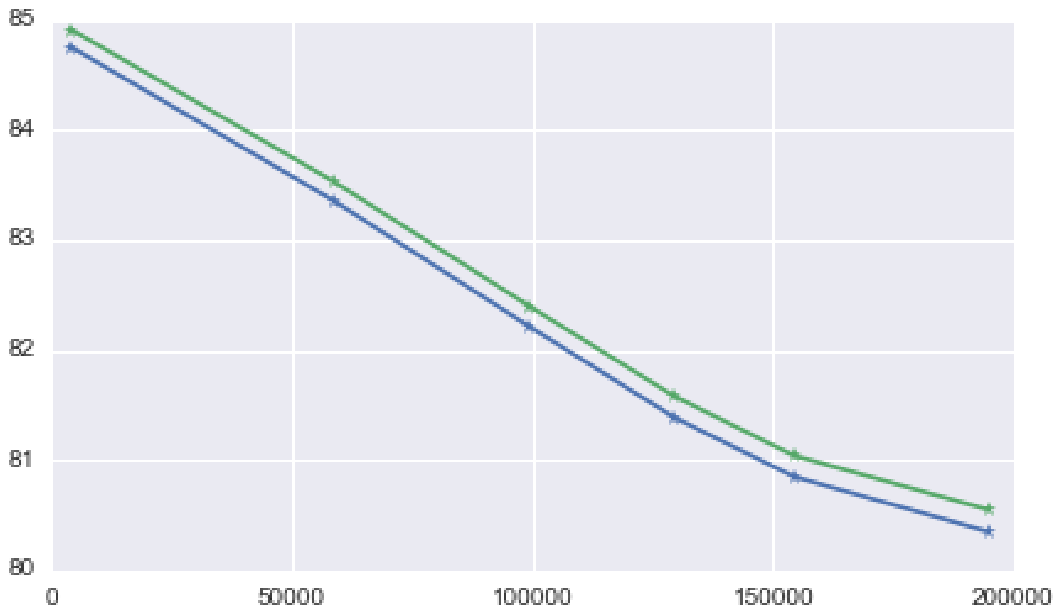
In [18]:
```
# Insert calculated distances into trace dataframe
tr['Dist_Ladario'] = dist_aux_ladario
tr['Dist_Manga'] = dist_aux_manga
tr['Dist_Esperanca'] = dist_aux_esperanca
tr['Dist_Coimbra'] = dist_aux_coimbra
```

In [19]:
```
# Get trace point with minimum distance and atributes its long distance to the respectice station
dl = tr.Dist[tr.Dist_Ladario == tr.Dist_Ladario.min()].values[0]
dm = tr.Dist[tr.Dist_Manga==tr.Dist_Manga.min()].values[0]
de = tr.Dist[tr.Dist_Esperanca==tr.Dist_Esperanca.min()].values[0]
```

```
dc = tr.Dist[tr.Dist_Coimbra==tr.Dist_Coimbra.min()].values[0]
st['Dist'] = [dl,dm,de,dc]
```

In [20]:
```
plt.figure(figsize(7,4))
plt.plot(rn.Dist[:6],rn['A-R'][:6],marker='*')
plt.plot(rn.Dist[:6],rn['HONA'][:6],marker='*')
```

Out[20]: [<matplotlib.lines.Line2D at 0x111d4e10>]



In [21]:
```
rn['IER'] = np.nan
rn['IER'][rn.Dist<st.Dist.min()] = 0
rn['IER'][(rn.Dist>=st.Dist.min())&(rn.Dist<=st.Dist.max())] = 1
rn['IER'][rn.Dist>st.Dist.max()] = 2
```

In [22]:
```
st.Var
```

Out[22]:
```
0     -40.0
1     -34.0
2     -70.5
3    -144.0
Name: Var, dtype: float64
```

In [23]:
```
def correction(ier,s):
    if ier == 0:
        r = st.Var[0]
    elif ier == 2:
        r = st.Var[3]
    else:
        y = [st.Var[0],st.Var[3]]
```

```
            x = [st.Dist[0], st.Dist[3]]
            regression = np.polyfit(x,y,1)
            poli = np.poly1d(regression)
            polynomial = poli
            r = poli(s)
        return r
```

```
In [24]: rn['Var'] = np.nan
```

```
In [25]: rn['Var'][:6] = map(lambda x: correction(rn['IER'][x],rn['Dist'][x]),range(6))
```

```
In [26]: rn['Var'][:6]
```

```
Out[26]: 0     -40.000000
         1     -69.488915
         2     -91.884552
         3    -108.532456
         4    -122.278902
         5    -144.000000
         Name: Var, dtype: float64
```

```
In [29]: rn.to_pickle(tabelas+'rn_1_1_var.p')
         st.to_pickle(tabelas+'st_1_1_var.p')
```

C.5.3 Volume Calculations

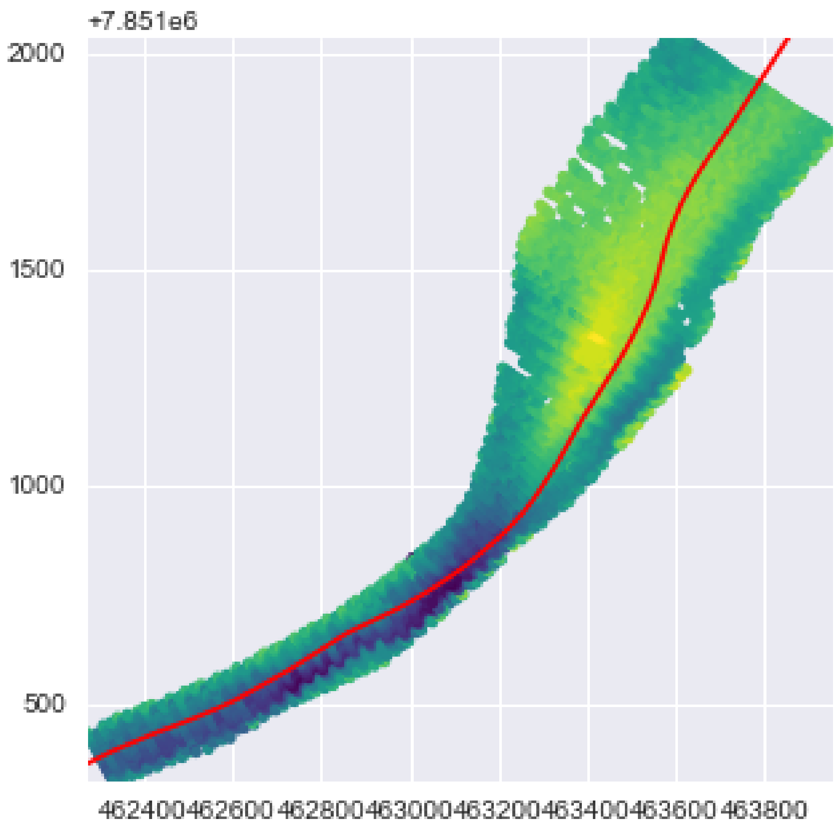
```
In [29]: b = pd.read_pickle(root2+'Batimetria_Caruso_Altitude.p')
```

```
In [30]: bc = b[b.Local=='CARAGUATA'][1:]
```

```
In [31]: cm = plt.cm.get_cmap('viridis')
```

```
In [32]: plt.figure(figsize=(5,5))
         plt.scatter(bc.X,bc.Y,c=bc.Z.tolist(),cmap=cm,edgecolor='None')
         plt.plot(tr.E,tr.N,c='r')
         xlim(bc.X.min(),bc.X.max())
         ylim(bc.Y.min(),bc.Y.max())
```

```
Out[32]: (7851321.8300000001, 7853035.6500000004)
```




```
In [33]: np.array(bc.Z.tolist())

Out[33]: array([-3.58, -3.81, -5.11, ..., -4.98, -5.12, -4.81])

In [34]: np.array(bc.Z.tolist())-rn.Var[2]/100

Out[34]: array([-2.66115448, -2.89115448, -4.19115448, ..., -4.06115448,
                -4.20115448, -3.89115448])

In [35]: bc['Z_new'] = np.array(bc.Z.tolist())-1.56-rn.Var[2]/100

In [36]: bc.to_csv(root2+'SC1_Caraguata_Bathymetry.csv',sep='\t')

In [30]: f_list  = ['scn_5_cgt_dreging.tif','scn_6_cgt_dreging.tif'',scn_7_cgt_dreging.tif','scn_8_cgt_dreging.

In [37]: import drl_scenario
         from drl_scenario import Drl_scenario

         reload(drl_scenario)
```

```
In [39]: workspace = "C:\\Users\\Usuario\\OneDrive\\Mestrado\\Volumetric Calculations"
         bathymetry = pd.read_csv(workspace+'\\SC1_Caraguata_Bathymetry.csv',sep='\t')

         scenario_1 = Drl_scenario(bathymetry,workspace)

         scenario_1.create_shape('scn_1_')

         scenario_1.kriging_interpolation('Z')

In [43]: scenario_1.define_dredging_volume("-3.3")
```

```
Executing: SurfaceVolume "C:\Users\Usuario\OneDrive\Mestrado\Volumetric Calculations\scn_1__krig_channel.tif" #
Start Time: Tue Feb 14 09:32:50 2017
Completed Surface Volume: 2D Area=44849,739973859  3D Area=44902,698379175  Volume=27059,056579308
Succeeded at Tue Feb 14 09:32:50 2017 (Elapsed Time: 0,08 seconds)
```

C.6 Water level hydraulic model volume calculation example

```
In [1]: import sys
        print(sys.version)
```

```
2.7.11 |Anaconda 2.3.0 (64-bit)| (default, Feb 16 2016, 09:58:36) [MSC v.1500 64 bit (AMD64)]
```

```
In [4]: import pandas as pd
```

```
In [5]: import arcpy, arcinfo
        from arcpy import env,TinRaster_3d
        import exceptions, sys, traceback
        # Obtain a license for the ArcGIS Spatial Analyst extension
        arcpy.CheckOutExtension("Spatial")
        # Obtain a license for the ArcGIS 3D Analyst extension
        arcpy.CheckOutExtension("3D")
        from arcpy.sa import *
        import drl_scenario
        from drl_scenario import Drl_scenario
```

```
In [6]: reload(drl_scenario)
```

```
Out[6]: <module 'drl_scenario' from 'drl_scenario.pyc'>
```

```
In [7]: # Name: ExtractValuesToPoints_Ex_02.py
        # Description: Extracts the cells of a raster based on a set of points.
        # Requirements: Spatial Analyst Extension

        # Set environment settings
        env.workspace = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Volumetric Calculations'

        # Set local variables
        inPointFeatures = "tracado_2.shp"
        inRaster = "scn_8_model_wl.tif"
        outPointFeatures = "tracado2_scn_8.shp"

        # Check out the ArcGIS Spatial Analyst extension license
        arcpy.CheckOutExtension("Spatial")

        # Execute ExtractValuesToPoints
        ExtractValuesToPoints(inPointFeatures, inRaster, outPointFeatures,
                              "INTERPOLATE", "VALUE_ONLY")
```

```
Out[7]: <geoprocessing server result object at 0x3e20810>
```

```
In [4]: # Set Environmental Variables
        workspace = 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Volumetric Calculations'
        env.workspace = workspace
        env.overwriteOutput = True
```

```
In [5]: # Set Local Variables
        inTin = 'Modelo_Ladario_Pesper\\Scenario_8\\sc8\\tp001'
```

```
        outRaster = 'scn_8_model_wl.tif'
        # Transform hec-ras TIN to a Raster file.
        TinRaster_3d(inTin, outRaster, data_type="FLOAT", method="LINEAR",
                           sample_distance="CELLSIZE 3", z_factor=1)
```

```
Out[5]: <Result 'C:\\Users\\Usuario\\OneDrive\\Mestrado\\Volumetric Calculations\\scn_8_model_wl.tif'>
```

```
In [6]: # Set Local Variables
        inRaster = 'scn_8_model_wl.tif'
        survey_mask = 'Caraguata_Bathymetry_Mask.shp'
        outName = 'scn_8_model_wl_cgt.tif'
```

```
In [7]: # Get raster inside mask
        outExtractByMask = ExtractByMask(inRaster, survey_mask)
        outExtractByMask.save(outName)
```

```
In [8]: # Create dredging plane raster
        outMinus = Raster('scn_8_model_wl_cgt.tif') - 3.3
        outMinus.save("scn_8_model_wl_cgt_DP.tif")
```

```
In [9]: # Create interpolation of the rivers bottom on the models z datum.
        bathymetry = pd.read_csv(workspace+'\\SC5_Caraguata_Bathymetry.csv',sep='\t')

        scenario_1 = Drl_scenario(bathymetry,workspace)

        scenario_1.create_shape('scn_8')

        scenario_1.kriging_interpolation('Alt_Z')
```

```
STARTED
Kriging OK !
Mask Survey OK !
Mask Channel OK !
```

```
In [10]: # Botton interpolation minus modeled water level
         outDredging = Raster('scn_8_krig_survey.tif') - Raster("scn_8_model_wl_cgt_DP.tif")
         outDredging.save("scn_8_cgt_dreging.tif")
```

```
In [11]: # Get raster inside channel mask
         channel_mask = 'Channel_Caraguata.shp'
         outExtractByMask = ExtractByMask("scn_8_cgt_dreging.tif", channel_mask)
         outExtractByMask.save('scn_8_cgt_dreging_channel.tif')
```

```
In [12]: result = arcpy.SurfaceVolume_3d('scn_8_cgt_dreging_channel.tif', "", "ABOVE", 0, "1", "0")
         print result.getMessages()
```

```
Executing: SurfaceVolume "C:\Users\Usuario\OneDrive\Mestrado\Volumetric Calculations\scn_8_cgt_dreging_channel.
Start Time: Thu Feb 16 17:59:48 2017
Completed Surface Volume: 2D Area=62040.735522797   3D Area=62155.869263021   Volume=66931.007029734
Succeeded at Thu Feb 16 17:59:48 2017 (Elapsed Time: 0.02 seconds)
```

D MODULES

D.1 anastage.py

```
import pandas as pd
import seaborn as sns
import os
import matplotlib.ticker as ticker
import calendar
import datetime
from datetime import timedelta
import numpy as np

def add_months(sourcedate,months):
        ''' Method to assist stack_cota with month shifts '''
        month = sourcedate.month - 1 + months
        year = int(sourcedate.year + month / 12 )
        month = month % 12 + 1
        day = min(sourcedate.day,calendar.monthrange(year,month)[1])
        return datetime.date(year,month,day)

def stack_cota(df, consistencia):


    '''
    Recives a structured stage dataframe (pd.DataFrame)
    COTA.txt of ANA (with set header set as 15),
    the consistency number wished:

    1 - Unconsisted
    2 - Consisted

    and statck it in a date/stage dataframe.

    in type: pd.DataFrame
    out type: pd.DataFrame

    '''
    dfc = df[df.NivelConsistencia == consistencia]
    if consistencia == 1:
        dfc = dfc[dfc.MediaDiaria == 1]
    dfc.index = range(len(dfc))
    dft = dfc.ix[:,16:47]
    dft.index = dfc.Data
    dft.columns = range(1,32)
    df = dft
    print(len(df))
    if len(df)!=0:
        vert = pd.DataFrame(index = pd.date_range(dft.index.min(),
                                                   add_months(dft.index.max(),1)))
        dfcol = pd.DataFrame(df.stack())
        dfcol.columns =  ['Data']
        dfcol.to_csv('temp.csv',sep='\t',header=['Cota'])
```

```
        vert.to_csv('temp2.csv',sep='\t')
        dateparse2 = lambda x: pd.datetime.strptime(x, '%Y-%m-%d')
        df = pd.read_csv('temp.csv',sep='\t', parse_dates=['Data'], date_parser=
            ↪ dateparse2)
        df.columns = ['Data', 'Dia', 'Cota']
        df.index = df.Data+ map(lambda x: timedelta(days=float(x) - 1.), df.Dia.
            ↪ tolist())
        vert['Data'] = vert.index
        df['Data'] = df.index
        finalmente = pd.merge(vert,df,on='Data')
        finalmente = vert.join(df,lsuffix='_l',rsuffix='_r')
        finalmente['Mes'] = finalmente.index.month
        finalmente['Dia'] = finalmente.index.dayofyear
        finalmente['Ano'] = finalmente.index.year
        return finalmente
    else:
        return pd.Series(np.nan)

def merge_consistido(consistido,inconsistido):
        '''
        Merge consisted and unconsisted dataframes
        from stack_cota.

        in type: pd.DataFrame
        out type: pd.DataFrame

        '''
        if (any(consistido.notnull())) & (any(inconsistido.notnull())):
            frames = [consistido, inconsistido[inconsistido.Data_l>consistido.max
                ↪ ().Data_l]]
            cotag = pd.concat(frames)
            cotag = cotag.sort_index()
            return cotag
        elif any(consistido.notnull()):
            cotag = consistido
            cotag = cotag.sort_index()
            return cotag
        elif any(inconsistido.notnull()):
            cotag = inconsistido
            cotag = cotag.sort_index()
            return cotag
        else:
            print('Error: Stage series is probably empty')
            kill()
```

D.2 forecast_models.py

```
import pandas as pd
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
#matplotlib.use('TkAgg')
#import matplotlib.pyplot as plt
#######NEURAL NETWORK########
from pybrain.datasets import SupervisedDataSet
from pybrain.structure import SigmoidLayer, LinearLayer
from pybrain.tools.shortcuts import buildNetwork
from pybrain.supervised.trainers import BackpropTrainer
#from pybrain.tools.xml import NetworkWriter
#from pybrain.tools.xml import NetworkReader
#from pybrain.tools.neuralnets import NNregression
############################
#matplotlib.use('TkAgg')
from sklearn.svm import SVC
from sklearn import svm
from sklearn import decomposition
from sklearn import datasets

class basic_forecast(object):

        def __init__(self):
                pass
        def set_parameters(self,forecast_parameters,n_past_years,r_stat,cotg):
                'S,starting_year, delta_validation, delta_test, n_past_years'
                self.r_stat = r_stat
                self.forecast_parameters = forecast_parameters
                self.n_past_years = n_past_years
                self.cotg = cotg
        def  forecast(self, cotg):
                ''' Method to calculate the '''
                def DRL_n_years(self):
                        j=0
                        year2=[]
                        predict2=[]
                        NR102 = []
                        for j in range(len(self.r_stat)-self.n_past_years):
                            NR102.append(self.cotg.Cota[(self.cotg.Ano2==self.
                                ↪ r_stat.index.min()+j+self.n_past_years)].
                                ↪ describe(percentiles=linspace(0,1,21))['10%'])
                            predict2.append(self.cotg.Cota[(self.cotg.Ano2>=self.
                                ↪ r_stat.index.min()+j)&(self.cotg.Ano2<self.
                                ↪ r_stat.index.min()+j+self.n_past_years)].
                                ↪ describe(percentiles=linspace(0,1,21))['10%'])
                            year2.append(self.r_stat.index.min()+j+self.
                                ↪ n_past_years)
                        erro = abs(np.array(NR102 )- np.array(predict2))
                        erro20 = pd.DataFrame(erro)
                        erro20.index = year2
                        self.predict2 = predict2
                        self.NR102 = NR102
                        BM20 = pd.DataFrame([year2, NR102,predict2,abs(np.array(
```

```
            ↪ NR102)-np.array(predict2)).tolist()],index=['year',
            ↪ 'NR10','predict','erro']).T
        BM20.index=BM20.year
        self.bm = BM20
    DRL_n_years(self)
    self.test_results = self.forecast_parameters
    test_results = []
    BM_Error = []
    BM_Prediction = []
    BM_nr = []
    for i in range(len(self.forecast_parameters)):
        aux = self.bm[self.forecast_parameters[1][i]+self.
            ↪ forecast_parameters[2][i]+10:10+self.
            ↪ forecast_parameters[1][i]+self.forecast_parameters
            ↪ [2][i]+self.forecast_parameters[3][i]-1]
        test_results.append(aux)
        BM_Error.append(aux.erro)
        BM_Prediction.append(aux.predict)
        BM_nr.append(aux.NR10)
    self.test_results['BM_Error'] = pd.Series(BM_Error)
    self.test_results['BM_Prediction'] = pd.Series(BM_Prediction)
    self.test_results['BM_nr'] = pd.Series(BM_nr)
    self.results_dataframe = pd.concat(test_results)

class auto_regression(object):

    def __init__(self):
        pass
    def set_parameters(self,forecast_parameters,n_past_years,r_stat,cotg):
        'S,starting_year,␣delta_validation,␣delta_test,␣n_past_years'
        self.r_stat = r_stat
        self.forecast_parameters = forecast_parameters
        self.n_past_years = n_past_years
        self.cotg = cotg
    def forecast(self):
        # Define the training variables
        def set_training_values(self,i):
            self.training_values_x = self.r_stat['10%'].ix[self.
                ↪ r_stat.index.min()+1: self.forecast_parameters.ix[i
                ↪ :i,3:4].values[0][0]+self.forecast_parameters.ix[i:
                ↪ i,1:2].values[0][0]-2]
            self.training_values_y = self.r_stat['10%'].ix[self.
                ↪ r_stat.index.min()+2:self.forecast_parameters.ix[i:
                ↪ i,3:4].values[0][0]+self.forecast_parameters.ix[i:i
                ↪ ,1:2].values[0][0]-1]

        # Defines function to find the coefficients that best fit the
            ↪ data

        def train_coefficients(self):

            self.input_matrix = np.vstack((np.ones(len(self.
                ↪ training_values_x)),self.training_values_x.
                ↪ as_matrix().T)).T
            kron = (np.kron(np.eye(len(self.training_values_y)),np.
                ↪ ones((1,1))))
```

```
            kron[0][0] = 0
            kron_lambda =0
            self.coefficients = (np.mat(self.input_matrix.T)*np.
                ↪ linalg.pinv(np.mat(self.input_matrix)*np.mat(self.
                ↪ input_matrix.T)+kron_lambda*kron))*np.mat(self.
                ↪ training_values_y.values).T
            self.training_results = pd.DataFrame(np.zeros(len(self.
                ↪ training_values_y)))*np.NAN
            self.training_results['Years'] = self.training_values_y.
                ↪ index
            self.training_results['DRL'] = self.training_values_y.
                ↪ tolist()
            self.training_results['Prediction'] = np.squeeze(np.mat(
                ↪ self.input_matrix) * self.coefficients).tolist()[0]
            self.training_results['Error'] = np.sqrt((self.
                ↪ training_results['DRL'] - self.training_results['
                ↪ Prediction'])**2)

    def test_coefficients(self,i):
            # Need to be updated to the new format without validation
                ↪  period.
            # Slice [Ano_Inicio+10:Ano_Inicio+10+Delta]  Delta = 10;
                ↪ To be corrected need to input the right
                ↪ forecast_parameter and correct training values
                ↪ slicing as well.
            self.test_values_x = self.r_stat['10%'].ix[self.
                ↪ forecast_parameters.ix[i:i,1:2].values[0][0]+self.
                ↪ forecast_parameters.ix[i:i,3:4].values[0][0]-1:
                                        self.forecast_parameters.ix[i:i
                                            ↪ ,1:2].values[0][0]+self.
                                            ↪ forecast_parameters.ix[i:i
                                            ↪ ,3:4].values[0][0]+self.
                                            ↪ forecast_parameters.ix[i:i
                                            ↪ ,3:4].values[0][0]-2]
            self.test_values_y = self.r_stat['10%'].ix[self.
                ↪ forecast_parameters.ix[i:i,1:2].values[0][0]+self.
                ↪ forecast_parameters.ix[i:i,3:4].values[0][0]:
                                        self.forecast_parameters.ix[i:i
                                            ↪ ,1:2].values[0][0]+self.
                                            ↪ forecast_parameters.ix[i:i
                                            ↪ ,3:4].values[0][0]+self.
                                            ↪ forecast_parameters.ix[i:i
                                            ↪ ,3:4].values[0][0]-1]
            self.input_matrix_test = np.vstack((np.ones(len(self.
                ↪ test_values_x)),self.test_values_x.as_matrix().T)).
                ↪ T
            self.test_results = pd.DataFrame(np.zeros(len(self.
                ↪ test_values_y)))*np.NAN
            prediction_aux = np.mat(self.input_matrix_test)*self.
                ↪ coefficients
            self.test_results['Years'] = self.test_values_y.index
            self.test_results['DRL'] = self.test_values_y.tolist()
            self.test_results['Prediction'] = np.array(prediction_aux
                ↪ .T.tolist()[0])
            self.test_results['Error'] = np.sqrt((self.test_results['
                ↪ DRL'] - self.test_results['Prediction'])**2)
```

```
        def plot_results(self):
            fig = plt.figure(figsize=(4,2))
            self.test_results['DRL'].plot(marker='*',markersize=10)
            self.test_results['Prediction'].plot(marker='*',
                ↪ markersize=10)
            #self.test_results['Error'].plot(kind='bar')
            #plt.yticks(range(0,100,10))
            plt.legend(loc=1)
            plt.title(str(self.forecast_parameters.ix[i:i,1:2].values
                ↪ [0][0]+self.forecast_parameters.ix[i:i,3:4].values
                ↪ [0][0])+
          ': '+str(self.forecast_parameters.ix[i:i,1:2].values[0][0]+
            ↪ self.forecast_parameters.ix[i:i,3:4].values[0][0]+
          self.forecast_parameters.ix[i:i,3:4].values[0][0]-1) + '---
            ↪ E: '+str(round(self.test_results['Error'].mean())))

            plt.show()


        self.coefficients_summary = []
        self.training_results_summary = []
        self.test_results_summary = []
        self.error_summary = []
        self.predictions_summary = []
        self.tests_result = self.forecast_parameters

        for i in range(len(self.forecast_parameters)):

            set_training_values(self,i)
            train_coefficients(self)
            test_coefficients(self,i)
            plot_results(self)

            self.coefficients_summary.append(self.coefficients)
            self.training_results_summary.append(self.
                ↪ training_results)
            self.test_results_summary.append(self.test_results)
            self.error_summary.append(self.test_results.Error)
            self.predictions_summary.append(self.test_results.
                ↪ Prediction)

        self.forecast_results = pd.concat(self.test_results_summary)
        self.tests_result = self.forecast_parameters
        self.tests_result['AR_Error'] = self.error_summary
        self.tests_result['AR_Prediction'] = self.predictions_summary


class multiple_regression():

    def __init__(self):
        pass

    def set_month_values(self,aggfunc):

        self.monthly_values = self.cotg[self.cotg.Cota.notnull()].
```

```
        ↪ pivot_table(values = 'Cota', index=['Ano2'], columns=['Mes'
        ↪ ],aggfunc=aggfunc)
      self.monthly_values = self.monthly_values[[7, 8, 9, 10, 11,12,1,
        ↪ 2, 3, 4, 5, 6]]
      self.monthly_values.columns = range(12)

  def set_parameters(self,forecast_parameters,n_past_years,r_stat,cotg,
    ↪ number_of_months):
      'S,starting_year,␣delta_validation,␣delta_test,␣n_past_years'
      self.r_stat = r_stat
      self.forecast_parameters = forecast_parameters
      self.n_past_years = n_past_years
      self.cotg = cotg
      self.number_of_months = number_of_months
      self.set_month_values(np.max)


  def forecast(self):

      def set_training_values(self,i):
          self.training_values_x = self.monthly_values.ix[self.
            ↪ monthly_values.index.min()+1:
                    self.forecast_parameters.ix[i:i,1:2].values
                      ↪ [0][0]+self.forecast_parameters.ix[i:i
                      ↪ ,3:4].values[0][0]-1,
                          11-self.number_of_months:]
          self.training_values_y = self.r_stat['10%'].ix[self.
            ↪ monthly_values.index.min()+2:
                    self.forecast_parameters.ix[i:i,1:2].values
                      ↪ [0][0]+self.forecast_parameters.ix[i:i
                      ↪ ,3:4].values[0][0]]

      def train_coefficients(self):

          self.input_matrix = np.vstack((np.ones(len(self.
            ↪ training_values_x)),self.training_values_x.
            ↪ as_matrix().T)).T
          kron = (np.kron(np.eye(len(self.training_values_y )),np.
            ↪ ones((1,1))))
          kron[0][0] = 0
          kron_lambda = 0
          self.coefficients = (np.mat(self.input_matrix.T)*np.
            ↪ linalg.pinv(np.mat(self.input_matrix)*np.mat(self.
            ↪ input_matrix.T)+kron_lambda*kron))*np.mat(self.
            ↪ training_values_y.values).T
          self.training_results = pd.DataFrame(np.zeros(len(self.
            ↪ training_values_y )))*np.NAN
          self.training_results['Years'] = self.training_values_y.
            ↪ index
          self.training_results['DRL'] = self.training_values_y.
            ↪ tolist()
          self.training_results['Prediction'] = np.squeeze(np.mat(
            ↪ self.input_matrix) * self.coefficients).tolist()[0]
          self.training_results['Error'] = np.sqrt((self.
            ↪ training_results['DRL'] - self.training_results['
            ↪ Prediction'])**2)
```

```
def set_test_values(self,i):

        self.test_values_x = self.monthly_values.ix[self.
            ↪ forecast_parameters.ix[i:i,1:2].values[0][0]+self.
            ↪ forecast_parameters.ix[i:i,3:4].values[0][0]-1:
                                                    self.
                                                        ↪ forecast_parameters
                                                        ↪ .ix[i:i,1:2].
                                                        ↪ values[0][0]+
                                                        ↪ self.
                                                        ↪ forecast_parameters
                                                        ↪ .ix[i:i,3:4].
                                                        ↪ values[0][0]+
                                                    self.
                                                        ↪ forecast_parameters
                                                        ↪ .ix[i:i,3:4].
                                                        ↪ values[0][0]-2,
                                                    11-self.
                                                        ↪ number_of_months
                                                        ↪ :]

        self.test_values_y = self.r_stat['10%'].ix[self.
            ↪ forecast_parameters.ix[i:i,1:2].values[0][0]+self.
            ↪ forecast_parameters.ix[i:i,3:4].values[0][0]:
                                                self.forecast_parameters.ix
                                                    ↪ [i:i,1:2].values
                                                    ↪ [0][0]+self.
                                                    ↪ forecast_parameters.
                                                    ↪ ix[i:i,3:4].values
                                                    ↪ [0][0]+
                                                self.forecast_parameters.ix
                                                    ↪ [i:i,3:4].values
                                                    ↪ [0][0]-1]

def test_coefficients(self,i):
        self.input_matrix_test = np.vstack((np.ones(len(self.
            ↪ test_values_x)),self.test_values_x.as_matrix().T)).
            ↪ T
        self.test_results = pd.DataFrame(np.zeros(len(self.
            ↪ test_values_y)))*np.NAN
        prediction_aux = np.mat(self.input_matrix_test)*self.
            ↪ coefficients
        self.test_results['Years'] = self.test_values_y.index.
            ↪ tolist()
        self.test_results['DRL'] = self.test_values_y.tolist()
        self.test_results['Prediction'] = np.array(prediction_aux
            ↪ .T.tolist()[0])
        self.test_results['Error'] = np.sqrt((self.test_results['
            ↪ DRL'] - self.test_results['Prediction'])**2)

def plot_results(self):
        fig = plt.figure(figsize=(4,2))
        self.test_results['DRL'].plot(marker='*',markersize=10)
        self.test_results['Prediction'].plot(marker='*',
            ↪ markersize=10)
```

```
            #self.test_results['Error'].plot(kind='bar')
            #plt.yticks(range(0,100,10))
            plt.legend(loc=1)
            plt.title(str(self.forecast_parameters.ix[i:i,1:2].values
                ↪ [0][0]+self.forecast_parameters.ix[i:i,3:4].values
                ↪ [0][0])+
          ':'+str(self.forecast_parameters.ix[i:i,1:2].values[0][0]+
              ↪ self.forecast_parameters.ix[i:i,3:4].values[0][0]+
          self.forecast_parameters.ix[i:i,3:4].values[0][0]-1) + '——
              ↪ E:'+str(round(self.test_results['Error'].mean())))
            plt.show()


        self.coefficients_summary = []
        self.training_results_summary = []
        self.test_results_summary = []
        self.error_summary = []
        self.predictions_summary = []
        self.tests_result = self.forecast_parameters


        for i in range(len(self.forecast_parameters)):

            set_training_values(self,i)
            train_coefficients(self)
            set_test_values(self,i)
            test_coefficients(self,i)
            plot_results(self)

            self.coefficients_summary.append(self.coefficients)
            self.training_results_summary.append(self.
                ↪ training_results)
            self.test_results_summary.append(self.test_results)
            self.error_summary.append(self.test_results.Error)
            self.predictions_summary.append(self.test_results.
                ↪ Prediction)

        self.forecast_results = pd.concat(self.test_results_summary)
        self.tests_result = self.forecast_parameters
        self.tests_result['MR_Error'] = self.error_summary
        self.tests_result['MR_Prediction'] = self.predictions_summary


class artificial_neural_network(object):


    def __init__(self):
        pass

    def set_month_values(self,aggfunc):

        self.monthly_values = self.cotg[self.cotg.Cota.notnull()].
            ↪ pivot_table(values = 'Cota', index=['Ano2'], columns=['Mes'
            ↪ ],aggfunc=aggfunc)
        self.monthly_values = self.monthly_values[[7, 8, 9, 10, 11,12,1,
            ↪ 2, 3, 4, 5, 6]]
```

```
        self.monthly_values.columns = range(12)


def set_parameters(self,forecast_parameters,r_stat,cotg,number_of_months)
    ↪ :
        'S,starting_year,␣delta_validation,␣delta_test,␣n_past_years'
        self.r_stat = r_stat
        self.forecast_parameters = forecast_parameters
        self.cotg = cotg
        self.number_of_months = number_of_months
        self.neurons_structure = [3,8,8,8]
        self.set_month_values(np.max)

def normalize(self,series):

        if isinstance(series,pd.DataFrame):
                series_max = series.T.stack().max()
                series_min = series.T.stack().min()
                series_mean = series.T.stack().mean()
                series_std = series.T.stack().std()
        else:
                series_max = series.T.max()
                series_min = series.T.min()
                series_mean = series.T.mean()
                series_std = series.T.std()

        return (series-series_min)/(series_max-series_min)

def set_training_values(self,training_values_x,training_values_y):

        self.training_values_x = training_values_x
        self.training_values_y = training_values_y
        self.input_matrix_x = self.normalize(self.training_values_x).
            ↪ as_matrix()
        self.input_matrix_y = self.normalize(self.training_values_y).
            ↪ as_matrix()
        self.input_matrix_y = self.input_matrix_y.reshape(-1,1)

        self.ds = SupervisedDataSet(self.input_matrix_x.shape[1],1)
        for x, y in zip(self.input_matrix_x, self.input_matrix_y):
                self.ds.addSample(tuple(x), (y))

def set_training_validation_ratio(self):
        # PCA - DIMENTION REDUCTION

        pca = decomposition.PCA(n_components=2)
        pca.fit(self.input_matrix_x)

        X = pca.transform(self.input_matrix_x)
        x_test = pca.transform(self.test_values_x_norm.values)

        def dist(X,x_test):
                euc_dist = map(lambda i: np.linalg.norm(np.array([X[i]])-
                    ↪ np.array([x_test])),range(len(X)))

                return(euc_dist)
```

```
distances = dist(X,x_test)
ordered_distances = pd.DataFrame(distances).sort(0)
self.distances = dist(X,x_test)
self.ordered_distances = pd.DataFrame(distances).sort(0)

# SLICE VALIDATION AND TRAINING PERIODS

self.training_set_input = self.input_matrix_x[ordered_distances.
    ↪ index[5:]]
self.training_set_target = self.input_matrix_y[ordered_distances.
    ↪ index[5:]]
self.validation_set_input = self.input_matrix_x[ordered_distances
    ↪ .index[:5]]
self.validation_set_target = self.input_matrix_y[
    ↪ ordered_distances.index[:5]]

#CREATE DATASETS FOR PYBRAIN MODEL
self.training_set = SupervisedDataSet(self.training_set_input.
    ↪ shape[1], self.training_set_target.shape[1])
for x, y in zip(self.training_set_input, self.training_set_target)
    ↪ :
        self.training_set.addSample(tuple(x), (y))

self.validation_set = SupervisedDataSet(self.validation_set_input
    ↪ .shape[1], self.validation_set_target.shape[1])
for x, y in zip(self.validation_set_input, self.
    ↪ validation_set_target):
        self.validation_set.addSample(tuple(x), (y))

#print(self.validation_set)

plt.figure(figsize=(2,2))
plt.scatter(X[ordered_distances.index[5:]][:, 0], X[
    ↪ ordered_distances.index[5:]][:, 1],c='black')
#plt.scatter(X[ordered_distances.index[len(X)/10:]], self.
    ↪ training_set_target,c='black',s=20)
plt.scatter(x_test[:, 0], x_test[:, 1],marker='v',c='red',
    ↪ edgecolor=None,s=100)
#plt.scatter(x_test,self.test_values_y_norm ,marker='v',c='red',
    ↪ edgecolor=None,s=100)
#plt.scatter(X[:, 0][ordered_distances.index[:len(X)/10]], X[:,
    ↪ 1][ordered_distances.index[:len(X)/10]],c='yellow')
plt.scatter(X[ordered_distances.index[:5]][:, 0], X[
    ↪ ordered_distances.index[:5]][:, 1],c='y' )#c=self.
    ↪ validation_set_target.reshape(-1), cmap=plt.cm.viridis)
#plt.scatter(X[ordered_distances.index[:len(X)/10]][,:0], self.
    ↪ validation_set_target,c='yellow')
#print((X[ordered_distances.index[:len(X)/10]]).reshape(-1,1))
#print((self.test_values_y[ordered_distances.index[:len(X)/10]]).
    ↪ reshape(-1,1))
plt.show()

# REMOVE THE WEIRD EXTRA ZEROS
#self.validation_set.data['input'] = np.array([self.
    ↪ validation_set.data['input'][0]])
```

```
        #self.validation_set.data['target'] = np.array([self.
            ↪ validation_set.data['target'][0]])

    def set_validation_values(self,validation_values_x,validation_values_y):

        self.validation_values_x = validation_values_x
        self.validation_values_y = validation_values_y

        self.validation_values_x_norm = (self.validation_values_x - self.
            ↪ training_values_x.min())/(self.training_values_x.max() -
            ↪ self.training_values_x.min())
        self.validation_values_y_norm = (self.validation_values_y - self.
            ↪ training_values_y.min())/(self.training_values_y.max() -
            ↪ self.training_values_y.min())

        self.val_data = SupervisedDataSet(self.validation_values_x_norm.
            ↪ as_matrix().shape[1],1)
        for x, y in zip(self.validation_values_x_norm.as_matrix(), self.
            ↪ validation_values_y_norm.as_matrix().reshape(-1,1)):
                self.val_data.addSample(tuple(x), (y))
        #self.val_data.data['input'] = self.val_data.data['input'][0]
        #self.val_data.data['target'] = self.val_data.data['target'][0]

    def set_test_values(self,training_values_x,training_values_y):

        self.test_values_x = training_values_x
        self.test_values_y = training_values_y

        self.test_values_x_norm = (self.test_values_x - self.
            ↪ training_values_x.values.min())/(self.training_values_x.
            ↪ values.max() - self.training_values_x.values.min())
        self.test_values_y_norm = (self.test_values_y - self.
            ↪ training_values_y.values.min())/(self.training_values_y.
            ↪ values.max() - self.training_values_y.values.min())

        self.test_data = SupervisedDataSet(self.test_values_x_norm.
            ↪ as_matrix().shape[1],1)
        for x, y in zip(self.test_values_x_norm.as_matrix(), self.
            ↪ test_values_y_norm.as_matrix().reshape(-1,1)):
                self.test_data.addSample(tuple(x), (y))
        #self.test_data.data['input'] = self.test_data.data['input'][0]
        #self.test_data.data['target'] = self.test_data.data['target'][0]

    def build_network(self,i):

        self.network = buildNetwork(self.input_matrix_x.shape[1],8,1,bias
            ↪ =True)#self.neurons_structure[i/4]

    def backpropagation(self):

        self.trainer = BackpropTrainer(self.network,self.ds, verbose =
            ↪ False, learningrate=0.0001)
        #self.training_error2 = self.trainer.trainEpochs(1000)
        self.training_error, self.validation_error = self.trainer.
            ↪ trainUntilConvergence( dataset=self.training_set ,maxEpochs
            ↪ =2000, verbose = False, continueEpochs=200,
```

```
            ↪ validationProportion=0.15)#, trainingData=self.training_set,
            ↪  validationData=self.validation_set)#,#trainingData=self.
            ↪ training_set, validationData=self.validation_set)
        #plt.figure()
        #plt.plot(self.training_error)
        #plt.plot(self.validation_error)
        #plt.show()
    def nn_regression(self):
        fig = plt.figure(figsize=(5,5))
        self.nnRegression = NNregression(self.ds, hidden=self.
            ↪ number_of_months - 2, VDS=self.val_data, epoinc=750,Graph=
            ↪ fig)# VDS=self.val_data
        self.nnRegression.setupNN()
        self.nnRegression.initGraphics(ymax=1, xmax=-1)
        self.run  = self.nnRegression.runTraining(convergence=0.00000001)

    def backpropagation_plot(self):

        plt.figure(figsize=(2,1))
        plt.plot(self.training_error[:], 'b', self.validation_error[:], 'r'
            ↪ )
        plt.legend(loc=2)
        plt.show()

    def test_coefficients(self):

        self.test_prediction=[]
        for k in self.test_values_x_norm.index:
                self.test_prediction.append(self.network.activate(self.
                    ↪ test_values_x_norm.ix[k:k,:].values[0]))

        ymax = self.training_values_y.max()
        ymin = self.training_values_y.min()
        self.test_error = abs(pd.DataFrame((np.array(self.test_prediction
            ↪ )*(ymax-ymin))+ymin)-pd.DataFrame(self.test_values_y.
            ↪ as_matrix()))

    def forecast_plot(self):
        plt.figure(figsize=(5,5))
        ymax = self.training_values_y.max()
        ymin = self.training_values_y.min()
        plt.plot(self.test_values_y.tolist(),marker='*',markersize=10)
        plt.plot((np.array(self.test_prediction)*(ymax-ymin))+ymin,marker
            ↪ ='*',markersize=10)
        print(self.test_error,self.test_error.mean())
        #plt.title(str(Ano_Inicio +10)+'-'+str(Ano_Inicio+20-1)+',
            ↪ Erro_av=:'+str(erro.mean().round(2)))
        plt.show()

class support_vector_regression(artificial_neural_network):

    def __init__(self):
        pass

    def svm_fit(self):
        X = self.input_matrix_x
```

```
        y = self.input_matrix_y.reshape(−1)
        self.clf = svm.SVR(kernel='linear',gamma=0.00000001)#, degree=2)
        self.clf.fit(X, y)

def svm_predict(self):
        self.test_prediction = self.clf.predict(self.
            ↪ validation_values_x_norm.values)
        ymax = self.training_values_y.values.max()
        ymin = self.training_values_y.values.min()
        self.predicted =( np.array(self.test_prediction)*(ymax−ymin))+
            ↪ ymin
        self.test_error = abs(pd.DataFrame((np.array(self.test_prediction
            ↪ )*(ymax−ymin))+ymin)−pd.DataFrame(self.test_values_y.
            ↪ as_matrix()))
```

D.3 ortoguay.py

```
#from datetime import datetime, date, time
from __future__ import print_function
from pandas import *
from math import *
from scipy import *
from cmath import *
from numpy import *
from matplotlib import *
import numpy as np
import pandas as pd
from socket import socket
import os
from dxfwrite import DXFEngine as dxf
#%pylab inline
########################################

########################################

#root = 'C:\\Users\\Henrique\\Documents\\EVTEA - Paraguai\\DTM\\'
root  = os.getcwd()
root2 = os.path.normpath(os.getcwd() + os.sep + os.pardir)

entrada = root2+'\\Entrada\\'
saida= root2+'\\Saida\\'

########################################
#Funcoes :
########################################


def distM(h,d,v):
        ''' Distancia de 'B' a 'Eta': '''
        dm = h* np.cos(np.arcsin(float(d)/float(h)))
        return dm

def cross(a, b):
    ''' Produto vetorial entre o vetor a[0,1,2] e b[0,1,2] '''
    c = [a[1]*b[2] - a[2]*b[1],
         a[2]*b[0] - a[0]*b[2],
         a[0]*b[1] - a[1]*b[0]]

    return c

def distPR(xp,yp,m,x1,y1):
    ''' Distancia de ponto (xp,yp) a reta y-y1 = m (x-x1)'''
    a = m
    b= -1
    c = -m*x1+y1
    d = abs(a*xp+b*yp+c)/math.pow((a**2+b**2),0.5)
    return d

def disti(x0,y0,x1,y1):
        ''' Distancia euclidiana entre dois pontos'''
        r = math.pow(math.pow(x1 - x0  , 2) + math.pow(y1 - y0,2),0.5)
```

```
        return r

def graph(formula, x_range):
        '''Plota um Array of points'''
        Xr = np.array(x_range)
        y = eval(formula)
        #print(Xr,y)
        plt.plot(x, y)
        #plt.show()

def angulo(x1,y1,x2,y2):
        ''' Retorna o angulo que um reta que contem (x1,y1) e (x2,y2) faz com o
           ↪ eixo x'''
        try:
                return np.arctan(float(y2-y1)/float(x1-x2))
        except:
                return np.arctan(np.inf)

def altitude(AC,AB,SC,SB,Sp):
        ''' Recebe:
                Altitude do ponto B e C,
                Coordenada S de B e C,
                e coordenada S de P.
                Retorna altitude de P. '''
        return ((AC-AB)/(SC-SB))*Sp + AB -((AC-AB)/(SC-SB))*SB

#Variavel auxiliar

k=range(1)
nome = raw_input(u'Nome do arquivo a ser interpolado (E,N,Z):')
nome2 =raw_input(u'Nome do arquivo base(tracado)(E,N,Z):')

root  = os.getcwd()
root2 = os.path.normpath(os.getcwd() + os.sep + os.pardir)


###########################################
#Leitura dos arquivos
###########################################
marg = pd.read_table(entrada+nome+'.csv')
df = pd.read_csv(entrada+nome2+'.xyz',sep='\t')

##########################################
#Dataframes de saida.
##########################################
index=range(len(df))
df3 = pd.DataFrame({'PtoE':[np.zeros(len(k))],'PtoN':[np.zeros(len(k))],'E':[np.
    ↪ zeros(len(k))], 'N':[np.zeros(len(k))], 'Z':[np.zeros(len(k))], 'Dist':[np.
    ↪ zeros(len(k))], 'RC':[np.zeros(len(k))], 'TanM':[np.zeros(len(k))], 'TanJ'
    ↪ :[np.zeros(len(k))], 'NormM':[np.zeros(len(k))], 'NormJ':[np.zeros(len(k))
    ↪ ], 'VecTM':[np.zeros(len(k))],'VecTJ':[np.zeros(len(k))],'VecNM':[np.zeros(
    ↪ len(k))],'VecNJ':[np.zeros(len(k))],'csi':[np.zeros(len(k))],'csf':[np.
    ↪ zeros(len(k))], 'mj':[np.zeros(len(k))],'mm':[np.zeros(len(k))]}, index=
    ↪ index)

index2=range(len(marg))
```

```
marg2 = pd.DataFrame({'Z':[np.zeros(len(k))],'PtoE':[np.zeros(len(k))],'PtoN':[np
    ↪ .zeros(len(k))],'E':[np.zeros(len(k))], 'N':[np.zeros(len(k))],'Alt':[np.
    ↪ zeros(len(k))],'s':[np.zeros(len(k))],'Alerta':[np.zeros(len(k))],'Eta':[np
    ↪ .zeros(len(k))],'IndexB':[np.zeros(len(k))],'pbc':[np.zeros(len(k))],'bp':[
    ↪ np.zeros(len(k))],'bc':[np.zeros(len(k))],'vbc':[np.zeros(len(k))],'vbp':[
    ↪ np.zeros(len(k))],'d':[np.zeros(len(k))]},index=index2)

############################################
#Gera as coordenadas S para o eixo
############################################
sum1=0
for i in range(1,len(df)):
    sum1 += disti(df.E[i-1],df.N[i-1],df.E[i],df.N[i])
    print(sum1)
    df3.Dist[i] = float(sum1)

print(u'Intervalo␣medio␣entre␣pontos␣de␣tracado:')
print(sum1/len(df))


############################################
#Largura das secoes transversais
############################################
fact = 400

for i in range(1,len(df)-1):
    #print(i)
    x0=df.E[i-1]
    y0=df.N[i-1]
    x1=df.E[i]
    y1=df.N[i]
    x2=df.E[i+1]
    y2=df.N[i+1]



    dist1 = disti(x1,y1,x2,y2)
    dyx = (y2 - y1)/(x2-x1) # coeficiente angular


    tangente = str(float(dyx)) + '*(x-' + str(x1)+')+'+ str(y1)
    #print(tangente)

    x = np.array([x1,x2])
    yl = np.array([y1,y2])

    #graph(tangente, range(int(x1),int(x2)))

    x = np.array([x0,x2])
    yl = np.array([y0,y2])

    #re = numpy.polyfit([x1,y2],[x2,y2],1)
    #reta = np.poly1d(re)


    xm = (x1+x2)/2.0
```

```
        ym= (y1+y2)/2.0


        #plt.scatter(xm,ym)
        #scatter(x,yl)

        if float(dyx)==0:
            print(i)
            normal = str(-9999999999999999999999) + '*(x-' + str(xm)+')+'+ str(ym)
        else: normal = str(-1.0/float(dyx)) + '*(x-' + str(xm)+')+'+ str(ym)

        #print(normal)
        #graph(normal,range(int(x0),int(x2)))
        #df3.Dist[i]=df.Dist[i]

        df3.VecTM[i+1]=np.array([x2-x1,y2-y1])/dist1
        df3.VecTJ[i]=np.array([x2-x1,y2-y1])/dist1
        df3.VecNM[i+1]=np.array([-(y2-y1),x2-x1,])/dist1
        df3.VecNJ[i]=np.array([-(y2-y1),x2-x1,])/dist1
        df3.TanM[i+1] = tangente
        df3.TanJ[i] = tangente
        df3.NormM[i+1] = normal
        df3.NormJ[i] = normal
        df3.E[i]=df.E[i]
        df3.N[i]=df.N[i]
        df3.csi[i]=[xm-fact*df3.VecNJ[i].tolist()[0],ym+(-fact)*df3.VecNJ[i].tolist()
            ↪ [1]]
        df3.csf[i]=[xm+fact*df3.VecNJ[i].tolist()[0],ym+fact*df3.VecNJ[i].tolist()
            ↪ [1]]
        df3.mm[i+1]=dyx
        df3.mj[i]=dyx

        #df3.Z[i]=df.z[i]
        #plt.plot([xm,xm-fact*df3.VecNJ[i].tolist()[0]],[ym,ym+(-fact)*df3.VecNJ[i].
            ↪ tolist()[1]],color='red')
        #plt.plot([xm,xm+fact*df3.VecNJ[i].tolist()[0]],[ym,ym+fact*df3.VecNJ[i].
            ↪ tolist()[1]],color='red')

        #plt.xlim((df.E[2000],df.E[2320]))
        #plt.ylim((df.N[2000],df.N[2320]))

#savefig('secoestrasversais6.png', transparent=0, bbox_inches='tight', pad_inches
    ↪ =0)#,autoscale_on=True)
df3.to_pickle(saida+'df3')


##############################################
#Saida das seoees trasversais
##############################################

drawing = dxf.drawing(saida+'cross_sections.dxf')
for i in range(1,len(df3)-1):
    #print(df3.csi[i][0])
    drawing.add(dxf.line((df3.csi[i][0], df3.csi[i][1]), (df3.csf[i][0], df3.csf[
        ↪ i][1]), color=7))
    #drawing.add_layer('TEXTLAYER', color=2)
```

```
    #drawing.add(dxf.text('Test', insert=(0, 0.2), layer='TEXTLAYER'))
drawing.save()


##############################################################
# Interpolacao das Altitudes - Step 1 - Gera coordenadas SD
##############################################################
if nome[0] != 'm':
        for i in range(1,len(marg)):
            P = [marg.X[i],marg.Y[i]]
            dmin = 2800
            a=(float(i)/len(marg))*100.0
            print("%.2f %%" % round(a,2))
            for j in range(2,len(df3)-1):
                        b = [df.E[j-1],df.N[j-1]]
                        c = [df.E[j],df.N[j]]
                        bc = disti(b[0],b[1],c[0],c[1])
                        bp = disti(b[0],b[1],P[0],P[1])
                        vbc = np.array([c[0]-b[0],c[1]-b[1]])/bc
                        vbp = np.array([P[0]-b[0],P[1]-b[1]])/bp
                        pbc =  np.arccos(np.dot(vbc,vbp))
                        versor=vbc
                        d = bp*np.sin(pbc)
                        pto = b + bp*np.cos(pbc)*versor
                        Ppto =disti(pto[0],pto[1],P[0],P[1])
                        versor2 = np.array([float(pto[0]-P[0]),float(pto[1]-P[1])
                            ↪ ])/Ppto
                        va= [float(versor[0]),float(versor[1]),0.0]
                        vb= [float(versor2[0]),float(versor2[1]),0.0]
                        croos = cross(va,vb)
                        #print(bc == disti(c[0],c[1],pto[0],pto[1]) + disti(b[0],
                            ↪ b[1],pto[0],pto[1]))
                        #plt.scatter(P[0],P[1])
                        if d<dmin:
                                if np.allclose(bc , disti(c[0],c[1],pto[0],pto
                                    ↪ [1]) + disti(b[0],b[1],pto[0],pto[1])) and
                                    ↪ (np.allclose(croos[2],1.0) or np.allclose(
                                    ↪ croos[2],-1.0)):
                                        #print(d)
                                        dmin=d
                                        #
                                            ↪ ##################################################
                                            ↪
                                        #Define se o ponto esta a direita ou a
                                            ↪ esquerda da reta bc#
                                        #
                                            ↪ ##################################################
                                            ↪
                                        position = sign( (c[0]-b[0])*(P[1]-b[1])
                                            ↪ - (c[1]-b[1])*(P[0]-b[0]) )
                                        #
                                            ↪ ##################################################
                                            ↪
                                        marg2['E'][i] = P[0]
                                        marg2['N'][i] = P[1]
                                        marg2['PtoE'][i] = pto[0]
```

```
                        marg2['PtoN'][i] = pto[1]
                        marg2['s'][i]=df3.Dist[j-1]+bp*np.cos(pbc
                            ↪ )
                        marg2['IndexB'][i] = df3.index[j-1]
                        marg2['pbc'][i] = pbc
                        marg2['bp'][i] = bp
                        marg2['bc'][i] = bc
                        marg2['vbc'][i] = [versor]
                        marg2['vbp'][i] = [vbp]
                        marg2['d'][i] = d * position
                        marg2['Z'][i] = marg.coord_Z[i]

else:
    for i in range(1,len(marg)):
        P = [marg.X[i],marg.Y[i]]
        dmin = 2800
        a=(float(i)/len(marg))*100.0
        print("%.2f %%" % round(a,2))
        for j in range(2,len(df3)-1):
            b = [df.E[j-1],df.N[j-1]]
            c = [df.E[j],df.N[j]]
            #if disti(P[0],P[1],b[0],b[1]) < 1000:
            #print(j)
            bc =disti(b[0],b[1],c[0],c[1])
            bp = disti(b[0],b[1],P[0],P[1])
            vbc = np.array([c[0]-b[0],c[1]-b[1]])/bc
            vbp = np.array([P[0]-b[0],P[1]-b[1]])/bp
            pbc =  np.arccos(np.dot(vbc,vbp))
            versor=vbc
            d = bp*np.sin(pbc)
            pto = b + bp*np.cos(pbc)*versor
            Ppto =disti(pto[0],pto[1],P[0],P[1])
            versor2 = np.array([float(pto[0]-P[0]),float(pto[1]-P[1])
                ↪ ])/Ppto
            va= [float(versor[0]),float(versor[1]),0.0]
            vb= [float(versor2[0]),float(versor2[1]),0.0]
            croos = cross(va,vb)
            #print(bc == disti(c[0],c[1],pto[0],pto[1]) + disti(b[0],
                ↪ b[1],pto[0],pto[1]))
            #plt.scatter(P[0],P[1])
            if d<dmin:
                if np.allclose(bc , disti(c[0],c[1],pto[0],pto
                    ↪ [1]) + disti(b[0],b[1],pto[0],pto[1])) and
                    ↪ (np.allclose(croos[2],1.0) or np.allclose(
                    ↪ croos[2],-1.0)):
                    #print(d)
                    dmin=d
                    #
                        ↪ ##################################################
                        ↪
                    #Define se o ponto esta a direita ou a
                        ↪ esquerda da reta bc#
                    #
                        ↪ ##################################################
                        ↪
                    position = sign( (c[0]-b[0])*(P[1]-b[1])
```

```
    ↪ -  (c[1]-b[1])*(P[0]-b[0])  )
#
    ↪ ################################################
    ↪
marg2['E'][i] = P[0]
marg2['N'][i] = P[1]
marg2['PtoE'][i] = pto[0]
marg2['PtoN'][i] = pto[1]
marg2['s'][i]=df3.Dist[j-1]+bp*np.cos(pbc
    ↪ )
marg2['IndexB'][i] = df3.index[j-1]
marg2['pbc'][i] = pbc
marg2['bp'][i] = bp
marg2['bc'][i] = bc
marg2['vbc'][i] = [versor]
marg2['vbp'][i] = [vbp]
marg2['d'][i] = d * position
#marg2['Z'] = marg.coord_Z[i]

marg2.to_pickle(saida+nome)
marg2.to_csv(saida+'Batimetria_Pontos_Altimetria'+'_'+nome+'.csv')
df3.to_pickle(saida+'Tracado_Referecia_ST')

####################################################
#END
####################################################
```

D.4 drl_volume_scenarios.py

```
import csv
import arcpy, arcinfo
from arcpy import env
import pandas as pd
import exceptions, sys, traceback
# Obtain a license for the ArcGIS Spatial Analyst extension
arcpy.CheckOutExtension("Spatial")
# Obtain a license for the ArcGIS 3D Analyst extension
arcpy.CheckOutExtension("3D")
from arcpy.sa import *

class Drl_scenario(object):
        def __init__(self, bathymetry, workspace):
                self.bathymetry = bathymetry
                self.workspace = workspace
                self.overwrite = True
        def create_shape(self,out_name):
                '''Creates the shapefile from the bathymetry dataframe. pd.
                    ↪ DataFrame with E,N,Alt,Alt_z'''

                # CREATE SHAPEFILE
                env.workspace = self.workspace
                env.overwriteOutput = self.overwrite
                out_path = self.workspace
                self.out_name = out_name
                geometry_type = "POINT"
                # Define the shp template (structure of columns)
                template = "Caraguata_Batimetria.shp"
                has_m = "DISABLED"
                has_z = "ENABLED" # Shape has Z value !

                # Use Describe to get a SpatialReference object
                spatial_reference = arcpy.Describe("Caraguata_Batimetria.shp").
                    ↪ spatialReference
                # Execute CreateFeatureclass
                arcpy.CreateFeatureclass_management(out_path, out_name,
                    ↪ geometry_type, template, has_m, has_z, spatial_reference)

                # FEED SHAPEFILE
                cursor = arcpy.InsertCursor(out_name+'.shp')
                for i in range(len(self.bathymetry)):
                        feature = cursor.newRow()
                        vertex = arcpy.CreateObject("Point")
                        vertex.X = self.bathymetry['E'].values[i]
                        vertex.Y = self.bathymetry['N'].values[i]
                        vertex.Z = self.bathymetry['Alt_Z'].values[i]
                        feature.shape = vertex
                        feature.Alt = self.bathymetry['Alt'].values[i]
                        feature.Alt_Z = self.bathymetry['Alt_Z'].values[i]
                        feature.Z = self.bathymetry['Z_new'].values[i]
                        cursor.insertRow(feature)
                del cursor

        def kriging_interpolation(self,field,cellSize = 3, kModel = "CIRCULAR",
```

```
↪ kRadius = 50, survey_mask="Caraguata_Bathymetry_Mask.shp",
↪ channel_mask = "Channel_Caraguata.shp"):

        # Set environment settings
        env.workspace = self.workspace

        # Set local variables
        inFeatures = self.out_name+'.shp'
        outRaster = self.out_name+'_'+"krgout"
        outVarRaster = self.out_name+'_'+"varout"
        cellSize = 3
        field = field
        kModel = "CIRCULAR"
        kRadius = 50
        print('STARTED')
        # Execute Kriging
        arcpy.Kriging_3d(inFeatures, field, outRaster, kModel, cellSize,
            ↪ kRadius, outVarRaster)
        print('Kriging␣OK␣!')

        # Extract values from Kriging output within the survey area and
            ↪ channel masks

        outExtractByMask = ExtractByMask(outRaster, survey_mask)
        outExtractByMask.save(self.out_name+'_'+'krig_survey.tif')
        print('Mask␣Survey␣OK␣!')

        outExtractByMask = ExtractByMask(outRaster, channel_mask)
        outExtractByMask.save(self.out_name+'_'+'krig_channel.tif')
        print('Mask␣Channel␣OK␣!')

    def define_dredging_volume(self,dredging_level):

        try:

            # Set environment settings
            env.workspace = self.workspace
            # Set Local Variables
            inSurface = self.out_name+'_'+'krig_channel.tif'
            #Execute SurfaceVolume
            result = arcpy.SurfaceVolume_3d(inSurface, "", "ABOVE",
                ↪ dredging_level, "1", "0")
            self.result = result
            print result.getMessages()

        except arcpy.ExecuteError:
            print arcpy.GetMessages()
        except:
            # Get the traceback object
            tb = sys.exc_info()[2]
            tbinfo = traceback.format_tb(tb)[0]
            # Concatenate error information into message string
            pymsg = 'PYTHON␣ERRORS:\nTraceback␣info:\n{0}\nError␣Info
                ↪ :\n{1}'\
              .format(tbinfo, str(sys.exc_info()[1]))
            msgs = 'ArcPy␣ERRORS:\n␣{0}\n'.format(arcpy.GetMessages
```

```
    ↪ (2))
# Return python error messages for script tool or Python
    ↪ Window
arcpy.AddError(pymsg)
arcpy.AddError(msgs)
```

REFERENCES

AAPA. *Regulatory Review of Sec. 404 of the Clean Water Task Act and the Fish and Wildlife Coordination Act - Recommendations to Presidential Task Force on Regulatory Relief.* [S.l.], 1951. Cited 5 times on pages 22, 23, 25, 26 e 27.

ABNT. *NBR-13246: Planejamento portuário - Aspectos náuticos - Procedimento.* [S.l.], 1995. Cited 2 times on pages 23 e 29.

ADAMOWSKI, J.; CHAN, H. F. A wavelet neural network conjunction model for groundwater level forecasting. *Journal of Hydrology*, Elsevier BV, v. 407, n. 1-4, p. 28–40, sep 2011. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2011.06.013>. Cited on page 33.

ADAMOWSKI, J. et al. Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in montreal, Canada. *Water Resources Research*, Wiley-Blackwell, v. 48, n. 1, jan 2012. Disponível em: <http://dx.doi.org/10.1029/2010wr009945>. Cited 2 times on pages 32 e 33.

ADAMOWSKI, J.; SUN, K. Development of a coupled wavelet transform and neural network method for flow forecasting of non-perennial rivers in semi-arid watersheds. *Journal of Hydrology*, Elsevier BV, v. 390, n. 1-2, p. 85–91, aug 2010. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2010.06.033>. Cited on page 33.

ADAMOWSKI, J. F. Development of a short-term river flood forecasting method for snowmelt driven floods based on wavelet and cross-wavelet analysis. *Journal of Hydrology*, Elsevier BV, v. 353, n. 3-4, p. 247–266, may 2008. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2008.02.013>. Cited on page 32.

AHIPAR. *Serviços de Engenharia para Levantamentos Topo-Batimetricos e Medições de Corrente do Rio Paraguai, no Trecho entre Corumbá-MS e a Foz do Rio Apa.* [S.l.], 2015. Cited 5 times on pages 8, 31, 59, 60 e 92.

ATIYA, A. F. et al. A comparison between neural-network forecasting techniques – case study: River flow forecasting. *IEEE Transactions on Neural Networks*, v. 10, n. 2, 1999. Cited 2 times on pages 31 e 33.

BELAYNEH, A.; ADAMOWSKI, J. Standard precipitation index drought forecasting using neural networks, wavelet neural networks, and support vector regression. *Applied Computational Intelligence and Soft Computing*, Hindawi Publishing Corporation, v. 2012, p. 1–13, 2012. Disponível em: <http://dx.doi.org/10.1155/2012/794061>. Cited on page 33.

BELAYNEH, A. et al. Long-term SPI drought forecasting in the awash river basin in Ethiopia using wavelet neural network and wavelet support vector regression models. *Journal of Hydrology*, Elsevier BV, v. 508, p. 418–429, jan 2014. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2013.10.052>. Cited 3 times on pages 31, 32 e 33.

BOOGAARD, A. Hydraulic Studies on the Nile River and its Structure, Phase II. *Delft Hydraulics of the Netherlands and the Hydraulic and Sediment Researsh Institute, Fairway Dimensions, Part 3, IWT Course.*, 1992. Cited 3 times on pages 23, 26 e 27.

BOX, G. E. P. *Time Series Analysis: Forecasting and Control*. [S.l.]: Holden-Day, 1970. ISBN 0816210942. Cited on page 32.

BRASIL. *Portaria Nº 24/CFPN, De 14 de Maio de 2011*. [S.l.], 2011. Cited on page 41.

BRASIL. *Agência Nacional de Águas (ANA) - HIDROWEB - Sistema de Informações Hidrológicas*. 2015. Online. Cited 5 times on pages 11, 44, 49, 51 e 59.

BRASIL. *Cartas Náuticas*. 2016. Internet. Disponível em: <http://www.mar.mil.br/dhn/chm/box-cartas-raster/raster_disponiveis.html>. Cited 2 times on pages 8 e 40.

BRASIL, M. do. *AVISOS AOS NAVEGANTES - HIDROVIA PARAGUAI-PARANÁ - DE ASSUNÇÃO A CÁCERES E CANAL TAMENGO - DH22 ISSN*. [S.l.], 2016. Cited on page 58.

BRAVO, J. M. et al. Coupled hydrologic-hydraulic modeling of the upper paraguay river basin. *Journal of Hydrologic Engineering*, American Society of Civil Engineers (ASCE), v. 17, n. 5, p. 635–646, may 2012. Disponível em: <https://doi.org/10.1061%2F%28asce%29he.1943-5584.0000494>. Cited on page 39.

BURLANDO, P. et al. Forecasting of short-term rainfall using ARMA models. *Journal of Hydrology*, Elsevier BV, v. 144, n. 1-4, p. 193–211, apr 1993. Disponível em: <http://dx.doi.org/10.1016/0022-1694(93)90172-6>. Cited on page 32.

CLARKE, R. T.; TUCCI, C. E. M.; COLLISCHONN, W. Variabilidade temporal no regime hidrológico da bacia do rio paraguai. *RBRH - Revista Brasileira de Recursos Hídricos.*, v. 8, n. 1, p. 201–21, 2003. Cited on page 44.

CNT. *Pesquisa CNT da navegação interior 2013*. [S.l.], 2013. Cited on page 15.

CORREIA, R. *Utilização de dados topo-batimétricos para a modelagem hidrodinâmica 1d com apoio de um sistema de informações geográficas – estudo de caso do rio Paraguai.* Dissertação (Mestrado) — PPGERHA/UFPR, 2016. Cited 2 times on pages 39 e 66.

CORREIA, R. et al. Modelagem hidrodinâmica para a avaliação da navegabilidade em um trecho da Hidrovia do rio Paraguai. *SOBENA*, TMD, 2015. Cited on page 47.

DJERBOUAI, S.; SOUAG-GAMANE, D. Drought forecasting using neural networks, wavelet neural networks, and stochastic models: Case of the algerois basin in north Algeria. *Water Resour Manage*, Springer Science Business Media, v. 30, n. 7, p. 2445–2464, mar 2016. Disponível em: <http://dx.doi.org/10.1007/s11269-016-1298-6>. Cited 2 times on pages 31 e 33.

EL-SERSAWY, H.; AHMED, A. F. Inland Waterways Design Criteria and its Applications in Egypt. *Ninth International Water Technology Conference*, 2005. Cited on page 26.

EPL. *Transporte inter-regional de carga no Brasil Panorama 2015. PNLI.* 2015. Apresentação. Cited on page 15.

EPL. *Plano Nacional de Logística Integrada*. 2016. Website. Disponível em: <http://www.epl.gov.br/plano-nacional-de-logistica-integrada-pnli>. Cited 3 times on pages 7, 15 e 16.

ESRI. *ArcPy*. 2014. Disponível em: <http://pro.arcgis.com/en/pro-app/arcpy/get-started/what-is-arcpy-.htm>. Cited on page 68.

ESRI. *ArcGIS Pro tool reference*. 2017. Online. Disponível em: <http://pro.arcgis.com/en/pro-app/tool-reference/main/arcgis-pro-tool-reference.htm>. Cited 2 times on pages 8 e 68.

FORRESTER, W. *Canadian Tidal Manual*. Otawa, Otario, 1983. Cited on page 30.

GÁMIZ-FORTIS, S. et al. Potential predictability of an iberian river flow based on its relationship with previous winter global SST. *Journal of Hydrology*, Elsevier BV, v. 385, n. 1-4, p. 143–149, may 2010. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2010.02.010>. Cited on page 32.

GAO, J. et al. A probabilistic framework for svm regression and error bar estimation. *Machine Learning*, Springer Nature, v. 46, n. 1/3, p. 71–89, 2002. Disponível em: <http://dx.doi.org/10.1023/A:1012494009640>. Cited on page 33.

GARDNER, M.; CORLING, S. Artificial neural networks (the multi-layer perceptron)—a review of applications in the atmospheric sciences. *Atmos Environ*, v. 32, p. 2627–2636, 1998. Cited on page 33.

GUARNERI, H. et al. Determinação do modelo digital de elevação de rios de grande extensão para uso em projetos de engenharia - estudo de caso do trecho brasileiro da hidrovia do rio paraguai. *9o Seminário de Transporte e Desenvolvimento Hidroviário Interior*, 2015. Cited on page 66.

GUETTER, A. K.; GEORGAKAKOS, K. P.; TSONIS, A. A. Hydrologic applications of satellite data: 2. flow simulation and soil water estimates. *Journal of Geophysical Research: Atmospheres*, Wiley-Blackwell, v. 101, n. D21, p. 26527–26538, nov 1996. Disponível em: <https://doi.org/10.1029%2F96jd01655>. Cited on page 35.

HAN, D. et al. A coupled 1-D and 2-D channel network mathematical model used for flow calculations in the middle reaches of the yangtze river. *Journal of Hydrodynamics, Ser. B*, Elsevier BV, v. 23, n. 4, p. 521–526, aug 2011. Disponível em: <https://doi.org/10.1016%2Fs1001-6058%2810%2960145-x>. Cited on page 36.

HAN, P. et al. Drought forecasting based on the remote sensing data using ARIMA models. *Mathematical and Computer Modelling*, Elsevier BV, v. 51, n. 11-12, p. 1398–1403, jun 2010. Disponível em: <http://dx.doi.org/10.1016/j.mcm.2009.10.031>. Cited on page 32.

HERR, H. D.; KRZYSZTOFOWICZ, R. Bayesian ensemble forecast of river stages and ensemble size requirements. *Journal of Hydrology*, Elsevier BV, v. 387, n. 3-4, p. 151–164, jun 2010. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2010.02.024>. Cited on page 31.

HILLEBRAND, G.; KLASSEN, I.; OLSEN, N. R. B. 3D CFD modelling of velocities and sediment transport in the iffezheim hydropower reservoir. *Hydrology Research*, IWA Publishing, apr 2016. Disponível em: <https://doi.org/10.2166%2Fnh.2016.197>. Cited on page 38.

IHO. *Resolutions of the International Hydrographic Organization*. Monaco, 2007. 64 p. Disponível em: <http://www.iho.int/iho{_}pubs/misc/M1Eversion07.> Cited on page 29.

ITTI. *Projeto de Derrocamento Hidrovia do Tocantins*. [S.l.], 2013. Cited on page 38.

ITTI. *Dragagem do Passo Do Jacaré Rio Paraguai, Km 1391. Hidrovia Paraguai-paraná*. [S.l.], 2014. Cited 3 times on pages 7, 22 e 38.

KARTHIKEYAN, L.; KUMAR, D. N. Predictability of nonstationary time series using wavelet and EMD based ARMA models. *Journal of Hydrology*, Elsevier BV, v. 502, p. 103–119, oct 2013. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2013.08.030>. Cited on page 32.

KEITH, T. Z. *Multiple Regression and Beyond - An Introduction to Multiple Regression and Structural Equation Solving*. [S.l.]: Routledge, 2015. Cited on page 34.

KISI, O. Wavelet regression model for short-term streamflow forecasting. *Journal of Hydrology*, Elsevier BV, v. 389, n. 3-4, p. 344–353, aug 2010. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2010.06.013>. Cited on page 33.

KISI, O.; PARMAR, K. S. Application of least square support vector machine and multivariate adaptive regression spline models in long term prediction of river water pollution. *Journal of Hydrology*, Elsevier BV, v. 534, p. 104–112, mar 2016. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2015.12.014>. Cited on page 33.

KISI Özgür. Daily river flow forecasting using artificial neural networks and auto-regressive models. *Turkish J. Eng. Env. Sci.*, v. 29, p. 9–20, 2005. Cited 2 times on pages 31 e 33.

KONO, Y. *Utilização de Rede Neural LVQ para a previsão de nível do rio Paraguai*. Dissertação (Mestrado) — INPE, 2008. Cited on page 35.

LEOPOLD, L. B.; WOLMAN, M. G. River channel patterns: braided, meandering and straight. *United States Geological Survey Professional Paper*, v. 282 - B, p. 39–84, 1957. Cited on page 21.

LIMA, C. A. N. Informações sobre as dragagens de manutenção do rio paraguai (nt 46/2005/cgmab/dpp). 2005. Cited on page 93.

LIU, Y.; LIANG, X. S.; WEISBERG, R. Rectification of the bias in the wavelet power spectrum. *Atmos. Oceanic Technol.*, v. 24, n. 12, p. 2093–2102, 2007. Cited on page 49.

MCALEER, J. B.; WICKER, C. F.; JOHNSION, J. R. *Design of Channels for Navigation*. [S.l.]: U.S. Army corps od Engineers, 1963. Cited 4 times on pages 23, 25, 26 e 27.

MCBRIDE, M. *PIANC Report No. 121 - 2014, Harbour Approach Channels - Design Guidelines*. [S.l.], 2014. Cited 2 times on pages 21 e 23.

MEHR, A. D.; KAHYA, E.; ÖZGER, M. A gene–wavelet model for long lead time drought forecasting. *Journal of Hydrology*, Elsevier BV, v. 517, p. 691–699, sep 2014. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2014.06.012>. Cited 2 times on pages 31 e 33.

MEKANIK, F. et al. Multiple regression and artificial neural network for long-term rainfall forecasting using large scale climate modes. *Journal of Hydrology*, Elsevier BV, v. 503, p. 11–21, oct 2013. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2013.08.035>. Cited on page 33.

MENG, A. et al. Wind speed forecasting based on wavelet packet decomposition and artificial neural networks trained by crisscross optimization algorithm. *Energy Conversion and Management*, Elsevier BV, v. 114, p. 75–88, apr 2016. Disponível em: <http://dx.doi.org/10.1016/j.enconman.2016.02.013>. Cited on page 33.

MERWADE, V. M.; MAIDMENT, D. R.; HODGES, B. R. Geospatial Representation of River Channels. *Journal of Hydrologic Engineering*, v. 10, n. 3, p. 243–251, 2005. ISSN 1084-0699. Disponível em: <http://ascelibrary.org/doi/abs/10.1061/(ASCE)1084-0699(2005)10:3(243)>. Cited on page 61.

MIGUENS, A. P. Navegação Fluvial. In: DHN (Ed.). *Navegação: A Ciência e a Arte.* 1ª. ed. Rio de Janeiro: [s.n.], 1996. III, cap. 40, p. 1489–1540. Cited 4 times on pages 7, 30, 38 e 39.

MISHRA, A.; DESAI, V. Drought forecasting using feed-forward recursive neural network. *Ecological Modelling*, Elsevier BV, v. 198, n. 1-2, p. 127–138, sep 2006. Disponível em: <http://dx.doi.org/10.1016/j.ecolmodel.2006.04.017>. Cited on page 31.

MISHRA, A. K.; DESAI, V. R. Drought forecasting using stochastic models. *Stochastic Environmental Research and Risk Assessment*, Springer Nature, v. 19, n. 5, p. 326–339, jun 2005. Disponível em: <http://dx.doi.org/10.1007/s00477-005-0238-4>. Cited on page 32.

MISHRA, A. K.; SINGH, V. P. Drought modeling – a review. *Journal of Hydrology*, Elsevier BV, v. 403, n. 1-2, p. 157–175, jun 2011. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2011.03.049>. Cited 3 times on pages 7, 33 e 34.

MüLLER, M.; FILL, H. D. Redes neurais aplicadas na propagação de vazões. *XV Simpósio Brasileiro de Recursos Hídricos*, 2003. Cited 2 times on pages 31 e 33.

MOHAMMADI, K.; ESLAMI, H.; KAHAWITA, R. Parameter estimation of an ARMA model for river flow forecasting using goal programming. *Journal of Hydrology*, Elsevier BV, v. 331, n. 1-2, p. 293–299, nov 2006. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2006.05.017>. Cited on page 32.

MONTGOMERY, D. R.; BUFFINGTON, J. M. Channel-reach morphology in mountain drainage basins. *GSA Bulletin*, v. 109, n. 5, p. 596–611, 1997. Cited on page 21.

MT; MD. *Plano Nacional de Logística e Transportes. Relatério Executivo.* [S.l.], 2007. Cited on page 15.

NéELZ, S.; PENDER, G. *Desktop Review of 2D Hydraulic Modelling Packages.* Environment Agency, 2009. ISBN 1849110794. Disponível em: <http://evidence.

environment-agency.gov.uk/FCERM/Libraries/FCERM_Project_Documents/SC080035_Desktop_review_of_2D_hydraulic_packages_Phase_1_Report.sflb.ashx>. Cited 2 times on pages 11 e 37.

NORDEMANN, D. J. R. Periodicidades, tendências e previsão a partir da análise espectral dinâmica da série dos níveis do rio paraguai, em ladário (1900/1995). *Pesq. agropec. bras.*, v. 33, p. 1787–1790, Oct. 1998. Cited 3 times on pages 44, 70 e 88.

OCHOA-RIVERA, J. C. Prospecting droughts with stochastic artificial neural networks. *Journal of Hydrology*, Elsevier BV, v. 352, n. 1-2, p. 174–180, apr 2008. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2008.01.006>. Cited on page 31.

O'LEARY, A. *Continuous wavelet transforms in Python*. 2013. Github. Disponível em: <https://github.com/aaren/wavelets>. Cited on page 49.

PAIVA, R. C.; COLLISCHONN, W.; TUCCI, C. E. Large scale hydrologic and hydrodynamic modeling using limited data and a GIS based approach. *Journal of Hydrology*, Elsevier BV, v. 406, n. 3-4, p. 170–181, sep 2011. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2011.06.007>. Cited on page 36.

PIANC. *Approach Channels – A guide for design*. [S.l.], 1997. Cited 2 times on pages 23 e 60.

RATTON, P. *Previsão de vazões na Hidrovia do rio Paraguai com aplicação do Filtro de Kalman*. Dissertação (Mestrado) — PPGERHA-UFPR., 2015. Cited 2 times on pages 35 e 44.

RATTON, P. et al. Aplicação de estudos prévios de modelagem para a definição da geometria estrutural de pontes. *XXVI ANPET*, p. 2042–2054, 2012. Cited on page 39.

RATTON, P. et al. Avaliação dos passos críticos de navegação no trecho brasileiro da hidrovia do rio paraguai. In: . [S.l.: s.n.], 2016. Cited on page 60.

ROSGEN, D. L. A classification of natural rivers. *Catena*, v. 22, n. 3, p. 169–199, June 1994. Cited on page 21.

ROSSUM, G. *Python reference manual*. [S.l.], 1995. Cited on page 49.

SAKSENA, S.; MERWADE, V. Incorporating the effect of DEM resolution and accuracy for improved flood inundation mapping. *Journal of Hydrology*, Elsevier BV, v. 530, p. 180–194, nov 2015. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2015.09.069>. Cited on page 36.

SALEH, F. et al. Impact of river bed morphology on discharge and water levels simulated by a 1D saint–venant hydraulic model at regional scale. *Journal of Hydrology*, Elsevier BV, v. 476, p. 169–177, jan 2013. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2012.10.027>. Cited on page 36.

SANG, Y.-F. et al. Wavelet-based hydrological time series forecasting. *Journal of Hydrologic Engineering*, American Society of Civil Engineers (ASCE), v. 21, n. 5, p. 06016001, may 2016. Disponível em: <http://dx.doi.org/10.1061/(asce)he.1943-5584.0001347>. Cited on page 33.

SEGURA-BELTRÁN, F. et al. Using post-flood surveys and geomorphologic mapping to evaluate hydrological and hydraulic models: The flash flood of the girona river (Spain) in 2007. *Journal of Hydrology*, Elsevier BV, v. 541, p. 310–329, oct 2016. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2016.04.039>. Cited on page 36.

STRAHLER, A. N. Hypsometric (area-altitude) analysis of erosional topography . *Geological Society American Bulletin*, v. 63, p. 1117–1142, 1952. Cited on page 21.

TIAN, M.; WANG, P.; KHAN, J. Drought forecasting with vegetation temperature condition index using ARIMA models in the guanzhong plain. *Remote Sensing*, MDPI AG, v. 8, n. 9, p. 690, aug 2016. Disponível em: <http://dx.doi.org/10.3390/rs8090690>. Cited on page 32.

TOMAS, G. P. *Avaliação Hidromorfológica do Uso de Espigões em Hidrovias.* Dissertação (Mestrado) — Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-graduação em Engenharia de Recursos Hídricos e Ambiental, 2014. Cited on page 39.

TORRENCE, C.; COMPO, G. P. A practical guide to wavelet analysis. *Bulletins of the American Meteorological Society.*, v. 79, n. 1, Jan. 1997. Cited on page 49.

UFPR/ITTI. *Evtea da Hidrovia do rio Paraguai. Volume 2 – Relatório dos Estudos e Projetos dos Melhoramentos Cotejados.* [S.l.], 2015. Cited on page 60.

UFPR/ITTI. *EVTEA da Hidrovia do rio Paraguai. Volume 3ª – Estudos Hidráulicos, hidrodinâmicos, de balizamento e sinalização.* [S.l.], 2015. Cited 11 times on pages 8, 11, 40, 41, 42, 43, 58, 59, 60, 65 e 66.

USACE. *HEC-GeoRAS. GIS Tools for Support of HEC-RAS using ArcGIS.* [S.l.], 2009. Cited on page 66.

USACE. *HEC-RAS. River Analysis System User's Manual . Version 4.1.* [S.l.], 2010. Cited on page 66.

VALIPOUR, M. Long-term runoff study using SARIMA and ARIMA models in the United States. *Meteorological Applications*, Wiley-Blackwell, v. 22, n. 3, p. 592–598, feb 2015. Disponível em: <http://dx.doi.org/10.1002/met.1491>. Cited on page 32.

WEIGANG, L.; NORDEMANN, D. J. R. Study and prediction of the paraguay river level by harmonic analysis and neural network. *Revista Brasileira de Geofísica*, v. 14, n. 2, 1996. Cited 6 times on pages 11, 33, 34, 45, 47 e 49.

WEIGANG, L. et al. Prediction of the level of paraguay river using neural networks. *Revista Agropec. Brasileira*, v. 33, p. 1791–1797, Out 1998. Cited 2 times on pages 31 e 35.

WHITE, F. *Fluid Mechanics.* McGraw-Hill Higher Education, 1998. ISBN 978-0073398273. Disponível em: <https://www.amazon.com/Fluid-Mechanics-Frank-M-White/dp/B0086PTBGK/ref=sr_1_7?s=books&ie=UTF8&qid=1484048280&sr=1-7&keywords=Fluid+Mechanics+4th+Edition>. Cited on page 36.

WHITING, P. J.; BRADLEY, J. B. A process-based classification system for headwater streams. *Earth Surface Processes and Landforms*, v. 18, n. 7, p. 603–612, 1993. Cited on page 21.

WICKER, C. F. *Evaluation of present State of Knowledge of factors affecting Tidal Hydraulic and Related Phenomena, Report No. 3. Committee on Tidal Hydraulics*. [S.l.], 1965. Cited 4 times on pages 23, 25, 26 e 27.

WICKER, C. F. Economic Channels and Maneuvering Areas for Ships. *journal of Water Ways, Harbors and Costal Engineering*, v. 97, p. 443–454, 1971. Cited 4 times on pages 23, 25, 26 e 27.

WILLIS, J. Modelling swimming aquatic animals in hydrodynamic models. *Ecological Modelling*, Elsevier BV, v. 222, n. 23-24, p. 3869–3887, dec 2011. Disponível em: <https://doi.org/10.1016%2Fj.ecolmodel.2011.10.004>. Cited on page 36.

XUE, C. hua; YIN, H. long; XIE, M. Development of integrated catchment and water quality model for urban rivers. *Journal of Hydrodynamics, Ser. B*, Elsevier BV, v. 27, n. 4, p. 593–603, aug 2015. Disponível em: <https://doi.org/10.1016%2Fs1001-6058%2815%2960521-2>. Cited on page 36.

YANG, Y. ping et al. The variations of suspended sediment concentration in yangtze river estuary. *Journal of Hydrodynamics, Ser. B*, Elsevier BV, v. 27, n. 6, p. 845–856, dec 2015. Disponível em: <https://doi.org/10.1016%2Fs1001-6058%2815%2960547-9>. Cited on page 36.

YANO, J.-I.; JAKUBIAK, B. Wavelet-based verification of the quantitative precipitation forecast. *Dynamics of Atmospheres and Oceans*, Elsevier BV, v. 74, p. 14–29, jun 2016. Disponível em: <http://dx.doi.org/10.1016/j.dynatmoce.2016.02.001>. Cited on page 33.

YASEEN, Z. M. et al. Stream-flow forecasting using extreme learning machines: A case study in a semi-arid region in Iraq. *Journal of Hydrology*, Elsevier BV, v. 542, p. 603–614, nov 2016. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2016.09.035>. Cited on page 31.

YöRüK, A.; SACHER, H. Methoden und qualität von modellrechnungen für hw-gefahrenflächen. In: 37. DRESDNER WASSERBAUKOLLOQUIUM 2014. *Simulationsverfahren und Modelle für Wasserbau und Wasserwirtschaft"*. [S.l.], 2007. Cited on page 36.

ZENTGRAF, R.; DETTMANN, T. River rhine - hydraulic and ship dynamic modelling. *River Flow 2010*, 2010. Cited on page 38.

ZENTGRAF, R.; FENTON, J.; BLENINGER, T. One-dimensional flow modelling and a case study of the river rhine. In: FERREIRA, R. M. et al. (Ed.). *River Flow 2006*. Informa UK Limited, 2006. Chapter 215. Disponível em: <https://doi.org/10.1201%2F9781439833865.ch215>. Cited on page 38.

ZOUNEMAT-KERMANI, M. et al. Evaluation of data driven models for river suspended sediment concentration modeling. *Journal of Hydrology*, Elsevier BV, v. 535, p. 457–472, apr 2016. Disponível em: <http://dx.doi.org/10.1016/j.jhydrol.2016.02.012>. Cited on page 32.