Adnotation

Thesis **Mathematical models and methods for improving the conversion efficiency of renewable energy based on aero-hydrodynamics effects**, presented by Viorel Bostan for the competition of doctor habilitate degree in techincal sciences, was developed at the Technical University of Moldova , Chisinau, is written in Romanian and contains 433 pages, 145 figures, 8 tables, and 223 references. The thesis consists of introduction, six chapters, conclusions and appendices. Appendicies contain additional 57 figures and 49 tables.

The thesis is dedicated to the study of aero-hydrodynamic effects in small power ($P < 20$ kW) wind turbine rotors and micro hydropower stations rotors using mathematical models to describe turbulent flows and modern methods of numerical simulation in the framework of computational fluid dynamics (CFD).

The purpose of this work is to increase the conversion efficiency and functional capacity of small power wind turbines and micro hydropower stations.

There were identified models and modern mathematical methods used to describe the turbulent flows specific to small power rotors with emphasis on aero-hydrodynamic unsteady effects and near blades effects. There was argued the geometry of efficient aero-hydrodynamics blades in terms of energy conversion efficiency based on which original concepts of aero-hydrodynamic rotors were developed.

Using the proposed CAD rotor models there were performed complex CFD simulations of the flow through the rotors and near the blades in order to determine the influence of the constructive and kinematic parameters on the power and performance factor characteristics of of the aero-hydrodynamic rotors used in wind turbines and micro hydropower stations; there were performed the boundary layer analyzis and identified the technical solutions that can assure the decrease of possible negative effects on the energy conversion efficiency.

Based on the obtained research results, there have been developed and manufactured new models of small power wind turbines and micro hydropower stations for various applications, including the concept of a wind turbine with tilting rotor and orientation to the wind direction through windrose-wheels. The developed technical solutions were protected by 17 patents and appreciated at International Innovations, Research and Technology Transfer Salons with 43 gold, 13 silver and 2 bronze medals.

Keywords: mathematical modeling; CFD numerical simulation; boundary layer; turbulent flow; aero-hydrodynamic rotor; wind turbine; small hydropower station.

Rezumat

Teza **Modele matematice şi metode de eficientizare a conversiei energiilor regenerabile în baza efectelor aero-hidrodinamice**, prezentată de către Viorel Bostan pentru conferirea gradului ştiinţific de doctor habilitat în tehnică, a fost elaborată la Universitatea Tehnică a Moldovei, Chişinău, este scrisă în limba română şi conţine 342 pagini, 90 de figuri, 38 tabele, şi 250 de titluri bibliografice. Structura tezei include: introducerea, 6 capitole, concluzii şi anexe. Anexele conţin 145 de pagini cu 52 de figuri şi 48 tabele.

Teza este consacrată studiului fenomenelor aero-hidrodinamice în rotoarele turbinelor eoliene (TE) şi microhidrocentralelor de flux (MCHF) de mică putere ($P < 20$ kW), cu aplicarea modelelor matematice de descriere a fizicii curgerii fluidelor şi a metodelor moderne de simulare numerică a turbulenţei din cadrul dinamicii fluidelor CFD.

Scopul lucrării constă în sporirea eficienţei conversiei şi a capacităţilor funcţionale ale turbinelor eoliene şi microhidrocentralelor de flux de mică putere.

Au fost identificate modelele şi metodele matematice moderne de descriere a curgerii turbulente a fluidului, specifică rotoarelor de mică putere, cu evidenţierea efectelor aero-hidrodinamice tranzitorii şi în vecinătatea palelor. A fost argumentate profilurile aero-hidrodinamice ale palelor eficiente din punct de vedere al randamentului conversiei energiei şi în baza lor au fost elaborate concepte originale de rotoare aero-hidrodinamice.

În baza modelelor CAD ale rotoarelor propuse: au fost efectuate simulări CFD complexe ale curgerii tranzitorii a fluidului prin rotoare şi în vecinătatea palelor, cu determinarea gradului de influenţă a parametrilor constructiv-cinematici asupra caracteristicilor de putere şi factorilor de performanţă aero-hidrodinamică a rotoarelor TE şi MHCF; a fost efectuată analiza fenomenului de curgere a fluidului în stratul limită şi identificate soluţii tehnice de control şi minimizare a impactului negativ al acestuia asupra eficienţei conversiei energiei.

În baza rezultatelor cercetărilor, au fost elaborate şi fabricate modele noi de TE şi MHCF pentru diverse aplicaţii, inclusiv conceptul TE cu rotor basculant şi orientare la direcţia curenţilor de aer cu windrose cu profil aerodinamic al palelor. Soluţiile tehnice elaborate au fost protejate cu 17 brevete de invenţie şi apreciate la saloanele internaţionale de inovaţii, cercetare şi transfer tehnologic cu 43 medalii de aur, 13 de argint şi 2 de bronz.

Cuvinte-cheie: modele matematice; simulare numerică CFD; strat limită; curgere turbulentă, rotor aero-hidrodinamic, turbină eoliană; microhidrocentrală.

Table of contents

List of Figures

Listings

Introduction

This is an introduction to the thesis. It should be 2–3 pages. Introduction should answer briefly to the following questions: what problem is being solved, why this problem is important, what functionalities should be implemented in the proposed information system, why, what other products do exist on the market with the same purpose, what kind of technologies are being used, what are their limitations, advantages, disadvantages etc. Also, at the end of your introduction you should describe shortly the contents of your thesis and chapters. See the end of this introduction for details.

Also, introduction may contain figures. Figures in introduction should be numbered I.1, I.2 and so on. Don't forget to comment your figures and make refference to them in the text.

For example, Boeing Commercial Airplanes has leveraged academia- and NASA-developed CFD technology, some developed under contract by Boeing Commercial Airplanes, into engineering tools used in new airplane development. As a result of the use of these CFD tools, the number of wings designed and wind tunnel tested for high-speed cruise lines definition during an airplane development program has steadily decreased, being substituted with complex CFD simulations, as shown in figure I.1. Also, it should be noted that wind tunnel simulations are still present, but are mainly used for validation of the numerical results.

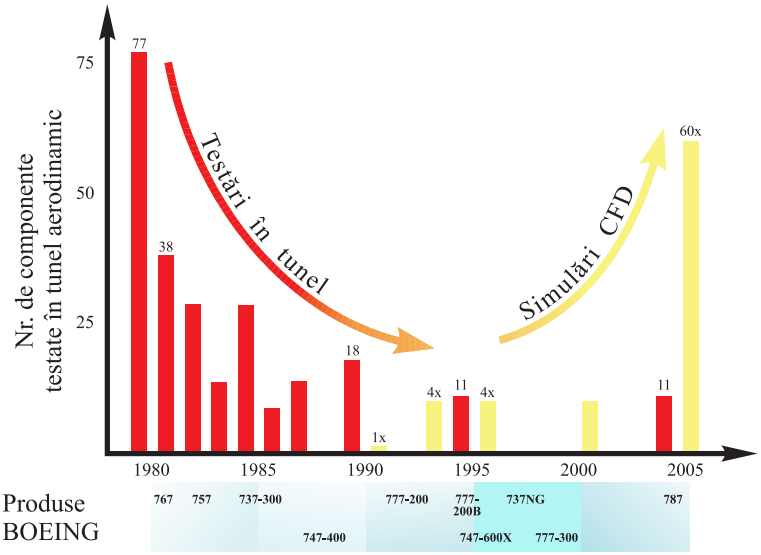

Figure I.1 – Evolution of the aerodynamic tunnel simulations versus the CFD simulations, adapted from [9]

A few words about figures should be mentioned. Figures, as well as tables, listings and formulas, must be numbered within chapters. Use only high quality figures, and don't use figures with low pixelisation. For example, the quality of the figures 2.1–2.3 is considered barely acceptable, and you must have a better definition/quality.

In other words avoid copy/paste behaviour. Copy/paste is reserved for ULIM or USM students. Don't forget to mention where did figure come from and give an exact citation from bibliography, citations can be placed either in the text or in the figure caption (citations are enclosed in brackets), see for example figures I.1 and I.2. As a rule of thumb, the font size of the text in a figure should not

exceed the font size of the main text. Figures should be relevant to your text, don't place a figure just to fill the main body. The size of the figure also should be appropriate, neither too big, nor too small. Figures should be readable and clear. Note that in figure I.2 the subfigures are being used. You can refer to individual subfigure as well, see for example figure I.2 (c). Don't forget to comment your figures and to refer them in text. These considerations also apply to the tables, listings and formulas used.

Figure should be either on the page where it is referenced to or on the next page. Don't refer to a figure for the first time, if the figure is on previous page. If the figure or group of figures is big (and its size is appropriate) you can place it on a single page alone. Figures can be placed in any position on the page, but avoid big empty spaces in the page.

(a)RePower 5 MW turbines

(b)GE 2.5 MW turbines

(c)Fântânele-Cogealac (Romania) wind farm

Figure I.2 – Wind park Thornton Bank with RePower 5 MW turbines located in North Sea (Belgium, 2012) (a) [11], wind park with GE 2.5 MW turbines located in Fântânele and Cogealac area (Romania, 2011-2012) (b,c) [12, 13].

The thesis consists of 4 chapters and contains 86 pages. In chapter 1 there are shown some examples for using formulas, tables and listings of the code. Chapter 2 gives an example of several pages from a past FAF thesis. In chapter 3 there are presented blah-blah-blah and so on. Finally, there are presented the conclusions, final recommendations and the future work. Also, the apendix section contains blah-blah.

1 Examples with formulas, tables and listings

1.1 Examples of formulas

Below is a text with some formulas. You should specify each variable being used, unless this variable was not mentioned in a formula mentioned befiore.

Consider a symmetric profile of a hydrodynamic blade in a fluid flow with uniform velocity $\overrightarrow{V}_\infty$. In the fixing point of the symmetric blade with the boom OO' we consider two coordinate systems, namely: the $O'xy$ system with axis $O'y$ oriented in the direction of velocity vector $\overrightarrow{V}_\infty$, and axis $O'x$ normal to this direction; and the $O'x'y'$ system with axis $O'y'$ oriented along the boom direction OO', and axis $O'x'$ normal to this direction. Points A and B correspond to the trailing edge and the leading edge, respectively. The angle of attack α is the angle between the profile chord AB and $\overrightarrow{V}_\infty$, and the positioning angle φ is the angle between the boom OO' and $\overrightarrow{V}_\infty$.

The hydrodynamic force $\overrightarrow{F}$ has its lift and drag components in directions $O'x$ and $O'y$, respectively, given by:

$$F_L = \frac{1}{2} C_L \rho_\infty V_\infty^2 S_p, \tag{1.1}$$

$$F_D = \frac{1}{2} C_D \rho_\infty V_\infty^2 S_p, \tag{1.2}$$

where ρ is the fluid density, V_∞ is the flow velocity, $S_p = ch$ (c is the chord length, h is the blade height) represents the lateral surface area of the blade, and C_L and C_D are the dimensionless hydrodynamic coefficients, lift and drag coefficients, respectively. Coefficients C_L and C_D are dependent on the angle of attack α, the Reynolds number Re and the hydrodynamic shape of the blade profile. The components of the hydrodynamic force in the coordinate system $O'x'y'$ are given by

$$F_{x'} = -F_L \sin\varphi + F_D \cos\varphi, \tag{1.3}$$

$$F_{y'} = F_L \cos\varphi + F_D \sin\varphi, \tag{1.4}$$

where F_L and F_D are determined form relations (1.1) and (1.2).

The torque at the rotor axis O developed by the blade i is

$$T_{r,i} = F_{x'} \cdot |OO'| \tag{1.5}$$

and the total torque developed by all blades

$$T_{r\Sigma} = \sum_{i=1}^{N_b} T_{r,i}, \tag{1.6}$$

where N_b is the number of the rotor's blades.

Since the hydrodynamic force with components (1.3–1.4) does not have its application point in the origin of the blade axis system O', it will produce a pitching moment with respect to a reference point chosen to be located at $\frac{1}{4}$ of the chord distance from the leading edge. The pitching moment, is computed by

$$M = \frac{1}{2} C_M \rho_\infty V_\infty^2 c S_p \tag{1.7}$$

where C_M represents the pitching moment hydrodynamic coefficient.

1.2 Examples of tables

You might use a large variety of styles for tables. Don't forget to refer to the tables in the text and comment them.

Table 1.1– Estimation for computational effort associated to RANS and LES methods

Method	Cells	Timesteps	No. of internal ops per timestep	Relativ effort compared with RANS
RANS	$\approx 10^6$	$\approx 10^2 - 10^3$	1	1
LES	$\approx 10^9$	$\approx 10^5$	$1 - 10$	$\approx 10^5 - 10^6$

Table 1.2– Discretisation parameters for windrose wheel surface

	Element Size	Curvature Normal Angle	Growth Rate
Hub	9 mm	10^o	1,17
Root	$4,5$ mm	4^o	1,17
TLE	$0,7$ mm	1^o	1,135
Tips	$0,4$ mm	1^o	1,135
Blades	$8,5$ mm	5^o	1,165

1.3 Examples of listings

The listings of the code can be inserted using the package listings and setting up your own style, that will enhance greatly the readability of your code. The style for listing is defined in the thesis style fyle. For more details check `http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings` or listings package documentation. Below there are presented two examples. In listing 1.1 the code is taken directly from the file, while in the second example the code is inserted directly in the text, see listing 1.2.

```
1  class ViewFacade {
2  // ... code removed ...
3  public void mousePressed(MouseEvent e) {
4          selectedView.selectHandle(e.getX(), e.getY());
5      }
6
7      public void mouseDragged(MouseEvent e) {
8          if (selectedView.moveSelected(e.getX(), e.getY()))
9          {
10             panel.repaint();
11         }
12     }
13
```

```
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2)
        {
            if (selectedView == sectorView)
            {
                if (selectedView.switchRotationDirection(e.getX(), e.getY()))
                {
                    panel.repaint();
                }
            }
        }
    }
// ... code removed ...
}
```

Listing 1.1– Java example, code is included from a fyle

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Listing 1.2– Python example, code is listed

2 Software Design

This chapter contains the description of the software development process used to bulid the application. We start with the description of the iterative and incremental model of developing software and justify its usage for this project. In the spirit of the chosen methodology we describe the major iterations of developing the application. For each iteration we state what subset of the initial requirements was implemented and state them more formally as a use case scenario that are used to get more insight into what should be implemented and verify if the designed module fullfills the outcomes set at the start of the iteration. We continue by describing the implementation details and presenting portions of source code in this context. In the last section we present a general overview of modular representation of the application and what are the connections between various application components.

2.1 The Software Development Process

In order to develop the application for this thesis project there was chosen the Iterative and Incremental approach. Iterative development is an approach to building software in which the overall lifecycle is composed of several iterations in sequence [7]. The iterations are organized into a series of short, usually fixed length steps called iterations, each iteration providing a tested implementation of a part of the system. The system is successively enlarged and refined during the iterations. An important input into each iteration is the feedback received from the client and the readaptation of the system to the changing requirements as the user understands and refines his view upon what functionality the software should deliver. The process is illustrated in figure 2.1

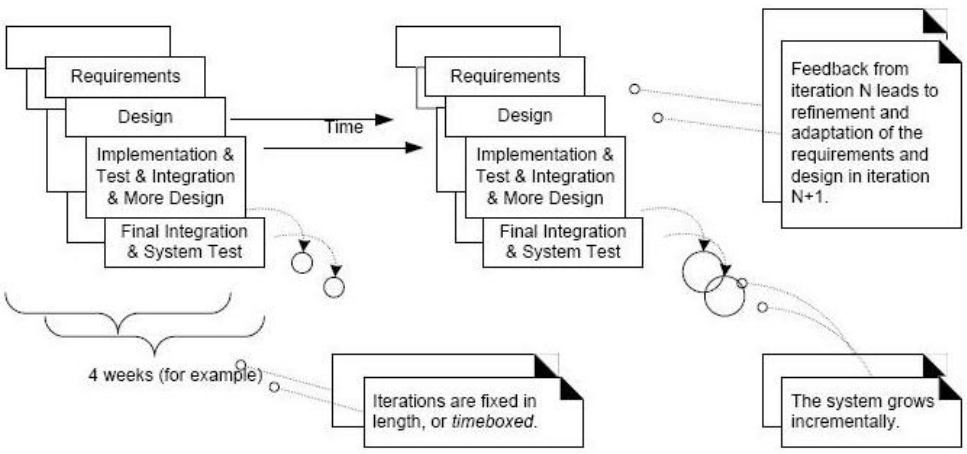

Figure 2.1– Iterative and Incremental Development, [8]

The output of each iteration is not an experimental prototype but a ready to use subset of the final system. During each iteration a subset of the software requirements is selected and implemented. However an interation shouldn't necessarily include the development of a new part of the system. Rather it could include some implementation steps for improving the performance of the existing modules.

The reason to use this approach for developing the application is that the application itself as well as the domain in which it will be used are relatively new. There are no current applications

on the market suited for the needs of the project simply because it is a part of a research project implementing an innovative idea in the field of renewable energy sources. Therefore there are no clear specifications of the software that can be taken as initial requirements. Also the approach itself allows the researchers to view intermediate versions of the software and refine their views upon how the workflow should be implemented, how the UI should look, etc.

2.2 Software Requirements

Although the application is built in small increments during successive iterations, a vision about the software under development and the list of requirements must be established at the beginning of the project. The description should contain the information about what kind of application is being developed (desktop, client-server, etc), who will be the final users and what are their goals, what domain processes or activities should the software implement. However at this stage it is impossible to define use case scenarios in detail. The main outcome of this step should be a list of system requirements prioritized by the value they have for the end user.

MHSim is a desktop application that will be used as a modeling and computational tool for the design and analysis of the working regime of the power station's rotor with rotating blades. The system will allow the user to specify the properties of the flow, the rotor and its blades. From a computational perspective the system must compute and render the plots for functional characteristics such as the forces which act on a blade, the torque produced by one blade, the resultant force and torque produced by all blades and the information about the power station floating stability. As a modeling tool the application should provide an ergonomic and intuitive UI, that is the structural parameters should be properly grouped and the UI should structure the steps of the workflow in the same way as if the modeling activity would be done in a real environment. Because the computational routines might last for a considerable amount of time, and the solution isn't guaranteed to converge for each case, the user must be able to control its execution. The application will be used by researchers to analyze a given configuration of the rotor with rotating blades and select the model which provides the best performance characteristics based on the data generated by the program. Since building the model of the rotor is only a stage in the overall power station design the application must provide the functionality of saving the data for further processing.

Next there are listed the software requirements derived from the high-level description made above and several discussion sessions with the end-user.

a) The system must provide the values of the hydrodynamic coefficients for a given airfoil profile.

b) The execution of the computational routine should be controlled by the user. This includes:

   1) completion status visualization;

   2) visualizing the computational strategies used;

   3) cancelling the solutions which don't converge.

c) The user should be able to select an airfoil by specifying a NACA profile or reading the coordinates of the airfoil from a file. The number of discretization points should be customizable.

d) The user must be able to enter the following properties of the fluid: density, viscosity and the freestream velocity.

e) The user must be able to specify the following parameters of the rotor and blades:

   1) the length of the rotor lever;

   2) the number of rotating blades;

   3) the height and chord length of the blades;

   4) the point where the blade is fixed to the rotor's lever;

   5) the number of regions in which a blade changes its direction relative to the flow. For each region the following information should be specified:

      -) the angle formed by the sector arms;

      -) the angle between the blade and the rotor lever;

      -) the angle between the blade and the flow at the start and end regions of the sector;

      -) an indicator showing if the blade is carried freely by the water flow (the blade in that sector won't contribute to the total moment generated by the rotor);

f) The system must show the direction of the flow and the direction of rotation of the power station.

g) The system must provide the graphs and numerical data for the following functional characteristics:

   1) Hydrodynamic coefficients versus angle of attack;

   2) Lift, drag, tangential and total force versus position angle;

   3) Torque generated by a blade and total torque generated by all blades;

   4) Buoyant stability data: the path traced by the center of application of buoyant forces and the vertical displacement of the center versus position angle.

The user requirements are organized into the use case diagram shown in figure 2.2

The sections that follow describe the software development process as a sequence of iterations illustrating the way that the system was built – using the incremental and iterative approach. Each iteration (except first which is mostly an implementation iteration) start with a use case scenario that describes the functionality of the system that is implemented in that iteration. We analyze the use case scenario to emphasize the details that affect the design of the implemented module. Then the design solution is presented along with the discussion of the implementation issues. During the iteration the usage and implementation of design patterns is emphasized and a short description of what design problems it mitigates is given.

2.3 Iteration 1 – linking to XFOIL code

During this iteration we develop a module that provides a simplified interface to the computational routines defined in XFOIL that later are used by the rest of the application to fullfil the

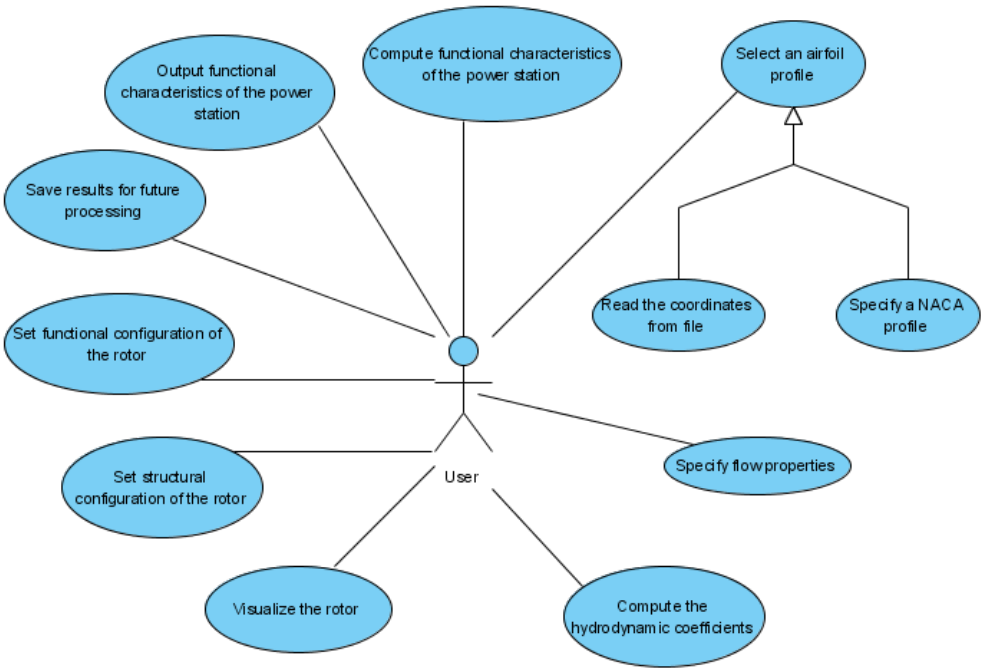

Figure 2.2– System use case diagram

first system requirement - to provide the values of the hydrodynamic coefficients for a given airfoil profile. The module provides the functions to generate and retrieve airfoil coordinates and extract the hydrodynamic coefficients at low angles of attack. Since this is an implementation step no use case scenarios are defined for this iteration.

2.3.1 Modifications to XFOIL code

The algorithms for determining hydrodynamic coefficients of lift, moment and drag forces require deep knowledge of CFD and dexterity in implementing complex computational methods. An example of the description of computation steps involved are described in [4]. An alternative approach would be to study the existing open source applications and use the already tested routines. The problem is that none of the analyzed programs provide such a module as a software library that can be added to our application. One well known program that allows to design and analyze isolated airfoils at subsonic speeds is XFOIL [5] . It is released under GNU General Public License which means that anyone is allowed to study and modify the source code. The first obvious problem is that XFOIL is designed as an interactive program: the user is presented with a command line menu where he enters commands and recieves response for each of the multiple options available. The other implementation problem is that XFOIL is written in Fortran 77, an old dialect of Fortran language approved as a standard in 1978 but the main part of the application will be written in Java. On the other side Fortran is an imperative language specifically designed for scientific and other computational intensive applications, meaning that the generated source code will perform faster than in most other programming langauges.

After studying the source code for 4 weeks solving the first problem proved to be a quite manageable task. The only modifications done at the first iteration involved transforming the conditional loops that accept user input and match against a set of predefined commands into linear routines. The input is passed from C++ language binding functions simulating the entered commands as if

typed by the user. Gathering the results of the computations also require a small amount of programming, but in order to better understand it, the programming model used in XFOIL is shortly described next.

XFOIL programming model. Following the classification of programming languages done in [3], the programming model used in XFOIL resembles the one used in Late Third-Generation Programming Languages shown in figure 2.3.

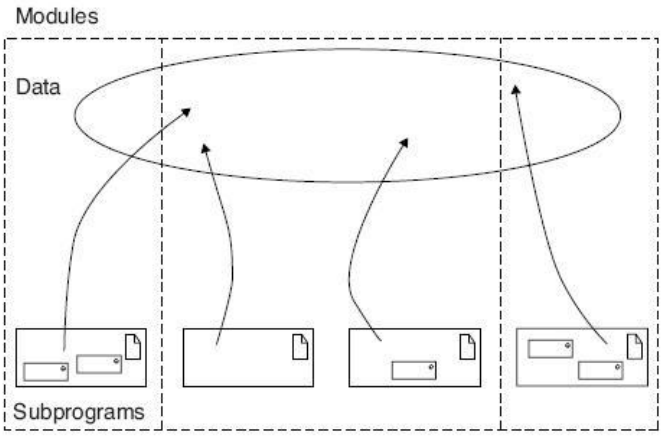

Figure 2.3 – The Topology of Late Third-Generation Programming Languages, [3]

The main entry point in XFOIL contains a loop that matches user input against the option from the top level menu. The selection of some option is followed either by the direct execution of the computational routines or the invocation of a lower lever menu that presents another set of the selected menu specific options. Among others there are 2 types of commands: commands that directly invoke the routines for implementing the selected option such as the command to generate a NACA airfoil profile or commands which sole purpose is to set a parameter used in further computations, for example setting the Reynolds number for the computation of the boundary layer equations for viscous fluids. The state of the computations performed in different routines is stored in *common blocks*, a Fortran specific language construct. Basically a common block represents a named block of memory that stores a set of variables. Data such as the coordinates of the airfoil profile or the current Reynolds number are stored in such blocks and may be accessed as long as the program is loaded into memory. New data may be freely added in this global storage by defining new common blocks. For the needs of the current application was added a block to hold the computed hydrodynamic coefficients for angles of attack between ± 20 degrees. In order to set the data the assignment statements that set the values of these coefficients must have been inserted in the proper place in the code (the code is located in EXEC_OPER_COMMAND a modified version of OPER subroutine which implemented a lower-lever menu which among others provides the options to compute the hydrodynamic coefficients).The same approach was taken in the next iteration when implementing the customizations for the coefficient extraction interface.

Linking Fortran and C++ code. In order to link the code written in XFOIL with the rest of the application we need to create C++ language bindings. By bindings we mean a set of functions that call the functions defined in the other language, extract data from global storage, etc. In the case of our application the bindings also must provide a simplified and restricted interface to XFOIL

code that would make the whole module more usable by the rest of the application code.

Linking C++ with Fortran code required the knowledge of the following typical mixed-language programming topics:

a) Name mangling;

b) Calling conventions used in both languages;

c) Specifics of parameter passing (is the parameter passed by value or reference, is the length of the array passed implicitly or not);

d) Primitive data type mappings (how similar data types are represented in memory, how are represented language specific data types, e.g. LOGICAL data type in Fortran);

e) How to represent language specific constructs (e.g. how to represent common blocks in C++ language).

If these aspects are not studied and handled properly the end-result usually manifests itself as a severe run-time error. A standard Fortran subroutine passes the parameters by address and has *__stdcall* calling convention. Name mangling can be handled by specifying additional compiler options or by declaring (in the binding code) the called functions with additional decoration symbols such as underscores (unless the mangling scheme is complex or results in invalid identifiers). A common block in Fortran maps to the declaration of a data structure in C++ with the same order of variables as declared in Fortran code, the common block itself is represented by an instance of the data structure with the same name as the common block in Fortran code. The listings 2.1 and 2.2 illustrate this mapping.

```
      COMMON/CR05/ X(IZX),Y(IZX),XP(IZX),YP(IZX),S(IZX),
     &             SLE,XLE,YLE,XTE,YTE,CHORD,YIMAGE,
     &             WGAP(IWX),WAKLEN
```

Listing 2.1– Common block declaration in Fortran

```
typedef struct {
  float x   [IZX]  ;
  float y   [IZX]  ;
  float xp    [IZX]  ;
  float yp    [IZX]  ;
  float s   [IZX]  ;
  float sle      ;
  float xle      ;
  float yle      ;
  float xte      ;
  float yte      ;
  float chord      ;
  float yimage       ;
  float wgap  [IWX]    ;
  float waklen       ;
} xfoil_cr05;

```

```
18 extern "C" {
19 extern xfoil_cr05 CR05;
20 }
```

Listing 2.2– Common block declaration in C++

2.3.2 C++ language bindings for XFOIL code

The class diagram shown in figure 2.4 shows the structural relationships between the concepts defined in the code that links to XFOIL program. In order to model C functions and structures using UML the approach described in [10] is used. The idea is to model a file containing C language code as a class stereotyped with <<File<<. As additional information we added a tagged value indicating in which language the file is written (on the diagram we have files written in C++ and Fortran languages). The functions and variables defined in a file represent the attributes and methods of the class. The visibility of attributes and methods can be either public or private. Public members represent code and data that is visible outside the file, that is functions and global variables declared in header files. On the other hand private members are functions and data inaccessible outside the file (functions and variables declared with *static* modifier). The simplified interface to XFOIL code is represented by the set of public methods in the *exports* class. The relationships between classes are mostly dependencies stereotyped as <<call>> and are used to indicate the files that contain functions that depend on functions defined in other files. We can also see associations going from *xfoil_data_structures* to C++ data structures declared in that file. The structures represent the C++ language equivalent of common blocks in Fortran from which functions defined in *exports* later will extract data such as the coordinates of the airfoil, the values of the hydrodynamic coefficients, etc. or set the data in those common blocks. On the diagram are shown only the fields from the common blocks which are used by the binding module. Next will be described the usage of each common block in the context of our application:

a) CI01 - stores the length of the airfoil identifier string;

b) CC01 - stores the airfoil identifier;

c) CR05 - stores the airfoil coordinates;

d) CL01 - stores flags set by XFOIL code (*lvisc* - used to determine if the viscous solution should be computed, *lnorm* - if the airfoil should be normalized [its chord must have unit length], *lvconv* - if the viscous solution converged for a given angle of attack);

e) CI04 - stores various integer variables among which the number of points on the current airfoil (n), maximal number of Netwon iterations to be performed before convergence is not attained ($itmax$) and the maximal number of points for which the solution may not converge ($nseqex$);

f) MYDATA - the common block added to XFOIL code that is used to store the hydrodynamic coefficients for angles of attack between ± 20 degrees. The block also contains an indicator set by XFOIL code to signal a file error in the routine for the extraction of airfoil coordinates from a file.

To illustrate how the interface simplifies the access to XFOIL code lets show an example of an actual sequence of function calls. The sequential diagram is shown in figure 2.5. The diagram illustrates how the client module sets and gets the coordinates of an airfoil. The client calls the method *setAirfoil()* to generate/read the coordinates of the airfoil and then store them in *CR05* block. The airfoil specifier could be either a file name or a NACA specifier. It is the responsiblity of the function to determine the type of specifier and what XFOIL command to generate. The function that simulates user input is called *xfoil_exec.* The function takes as parameters the command name and the argument(s). Because the client could request an option from 3 different XFOIL menus (top level menu, panel manipulation menu or airfoil analysis menu) we delegate the responsibility for determing the type of menu to code written in *xfoil_commands* file, namely to function *type_of_command*. The module contains a set of arrays that stores the subset of XFOIL commands used by the application. The returned identifier is used to select the function which dispatches the command to the appropriate XFOIL menu. Note that passing strings to Fortran requires passing also the length of the string (which is done implicitly in Fortran code). If the airfoil wasn't generated (for example if the file with airfoil coordinates could not be read) the function returns a failure indicator value, otherwise it continues by sending the commands to discretize the profile with the number of panels specified by the client code as a second parameter to *setAirfoil()* function. The details of dispatching those commands where not included in the diagram since they generate the same sequence of calls as in the case when first command was sent. After receiving a success indicator value, the client could extract the arrays containing the airfoil coordinates by calling *getAirfoil()* function. The coordinates of the airfoil are extracted from *CR05* common block.

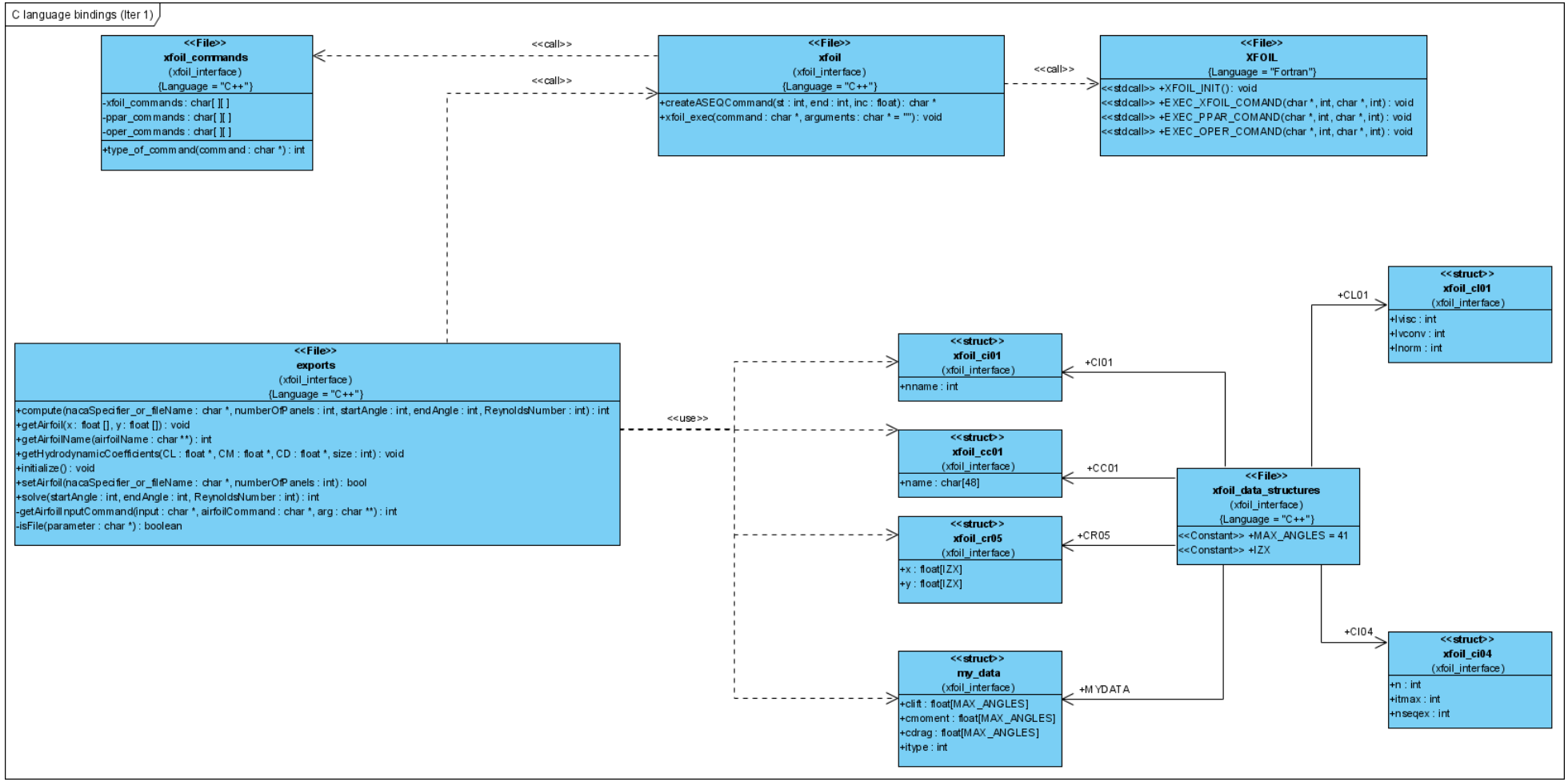

Figure 2.4– C++ language bindings – static view

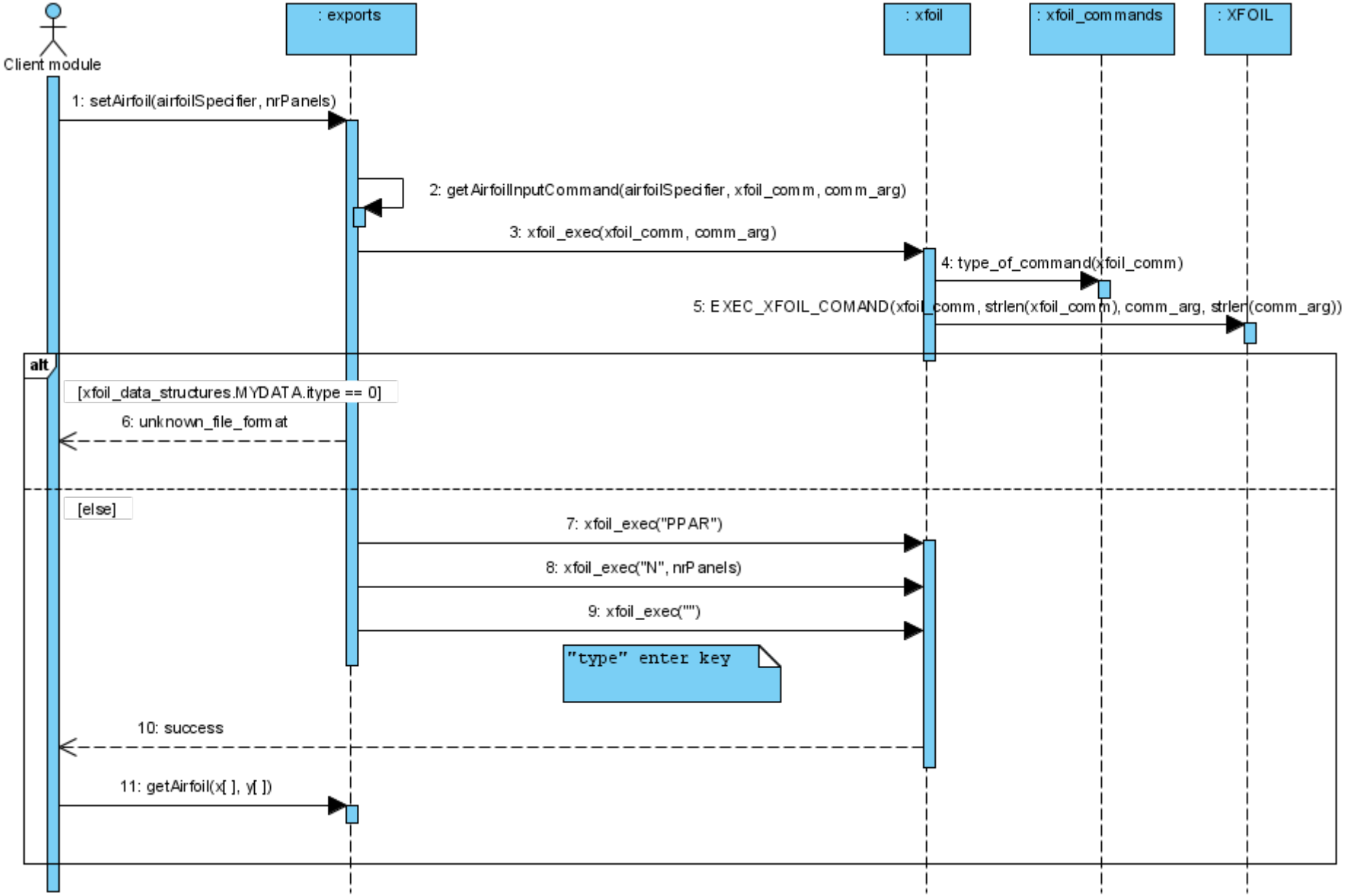

Figure 2.5– Setting and extracting airfoil coordinates

3 Title of third chapter

This is the third chapter
Here is some text.

4 Title of fourth chapter

This is the fourth chapter
Here is some text.

Conclusions

This is the conclusions section to the thesis. It should contain at least 2-3 pages. You can end conclusions using a description of the future development of your product/work.

References

1 Akins R.E. Performance evaluation of wind-energy conversion systems using the method of bins - current status. Internal.Report SAND-77-1375. Sandia Laboratory, USA, 1978.

2 Anderson J.D., Fundamentals of aerodynamics, 2nd Edition. McGraw-Hill, Singapore.

3 Booch G. et. al., Object-Oriented Analysis and Design with Applications, Addison-Wesley 2007.

4 Bostan I., Dulgheru V., Sobor I., Bostan V., Sochirean A., Sisteme de conversie a energiilor regenerabile, Technica-Info 2007.

5 Drela M. Xfoil : An Analysis and Design System for Low Reynolds Number Airfoils. Low Reynolds Number Aerodynamics, Springer-Verlag, Lec. Notes in Eng. 54, 1989.

6 Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley 1994.

7 Larman C., Agile and Iterative Development: A Manager's Guide, Adisson Wessley 2003.

8 Larman C., Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, Addison Wessley 2002.

9 President's Information Technology Advisory Committee, Computational Science: Ensuring America's Competitiveness, Report to President, `http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf`, June 2005.

10 `http://www.drdobbs.com/184401948`

11 Super-sized 6.15MW Wind Turbine Up and Running, 2050 Magazine, `http://www.2050publications.com/super-sized-6-15mw-wind-turbine-up-and-running/`

12 Largest Wind Farm in Europe Alive, `http://www.renewindians.com/2012/12/Largest-Wind-Farm-in-Europe.html`

13 GE Power & Water, Renewable energy, `http://www.ge-renewable-energy.com/uploads/tx_spdownloads/Fantanele_Romania_2.xWEA_03.JPG`

Appendix

This is an appendix to the thesis.

Here is some text. The figures, tables and listings from the appendix should be numbered Figure A.1, Figure A.2, Table A.1 etc.