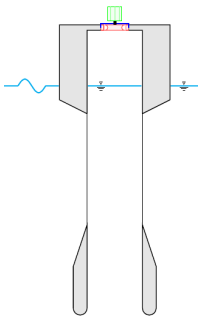
TÉCNICO LISBOA



A High-Order Discontinuous Galerkin Method for Optimal Control of an OWC Spar Buoy

Ricardo Filipe Gomes Duarte

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Duarte Pedro Mata de Oliveira Valério
Prof. João Carlos de Campos Henriques

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira
Supervisor: Prof. Duarte Pedro Mata de Oliveira Valério
Member of the Committee: Prof. João Manuel Lage de Miranda Lemos

January 2021

Acknowledgments

This section of the thesis is dedicated to the ones who have held an important role during the production of this work.

I will start by thanking my supervisors, who opened space and took the time to find me a thesis topic related with both the energy branch and the systems branch of the mechanical engineering course. They introduced me to the topic, guided me through the work and corrected my text many times, and some times on the same mistake. I would like to mention professor João Henriques enthusisam, who, even yesterday (one day before the thesis delivery) was suggesting how I could further improve my work and professor Duarte Valério, who responded to my messages with celerity and offered much needed advice and knowledge, even with time teaching me english.

I send a very special thank you for my girlfriend, Mariana Baptista and all the support she has shown me in these years. To my family who endured me and some of my lectures about ”things that do not matter to any one”.

Also, a direct thank you to José Bento, my colleague and friend, for the reminder on the thesis deadline.

I am also grateful to every one who made me company during this time, as the writing of a thesis, is known to be a lonesome task, namely: my family, my girlfriend, and my friends.

Also, a thank you to Jorge Silva, for sharing his research on Mutriku power plant for the first steps of the thesis.

Resumo

Os métodos pseudo-espectrais (PS) rapidamente tornaram-se a ferramenta padrão para resolver numericamente problemas de controlo óptimo (OCP). Uma das características principais dos métodos PS é a capacidade de atingir uma elevada precisão em OCPs contínuos. Esta precisão resulta da aproximação dos estados e do controlo ser feita usando um polinómio em todo o domínio. No entanto, a elevada precisão perde-se para problemas do tipo bang-bang devido à natureza descontínua do controlo. Esta tese pretende estudar uma abordagem alternativa aos métodos PS: um método de elementos finitos designado por Galerkin descontinuo (DG-FEM) usando o Príncipio do Máximo de Pontryagin. Contrastando com os métodos PS, o DG-FEM usa uma solução polinomial por troços para a trajéctória dos estados e do controlo, onde o refinamento de malha e polinomial são facilmente implementáveis. A aplicação de um refinamento de malha permite obter uma solução de elevada precisão mesmo para problemas tipo bang-bang. Foram considerados dois casos teste de OCPs: um contínuo e um bang-bang. Foi realizado um estudo detalhado da convergência do método para estes dois casos. Para isso, foi usada uma biblioteca que permite aritmética de ponto flutuante com precisão arbitrária. Os resultados demonstraram os problemas esperados pelo uso precisão dupla para polinónios de grau superior a seis. Finalmente, o método foi aplicado com sucesso a um problema de controlo óptimo de uma bóia tipo "OWC spar" de extração de energia das ondas. Os resultados mostraram um aumento de 20% na potência da turbina em comparação com o controlo padrão.

Palavras-chave: controlo óptimo não-linear, coluna de água oscilante, Galerkin descontínuo, princípio do máximo de Pontyagin, método de tiro

Abstract

Pseudospectral (PS) methods appeared in the 1990s and became an almost standard tool for the numerical solution of optimal control problems (OCPs). One of the fundamental characteristics of the PS methods is the ability to achieve high-order accuracy for smooth OCPs. This accuracy results from approximating the state and control trajectories by single polynomials across the entire domain. However, the high-order accuracy is usually lost for bang-bang OCPs because of the non-smooth nature of the control. The current thesis aims to study an alternative approach to PS methods: a high-order Discontinuous Galerkin Finite Element Method (DG-FEM) for the numerical solution of OCP based on Pontryagin's Maximum Principle. In contrast with the PS methods, the DG-FEM use a piecewise polynomial solution of the state and control trajectories where mesh and polynomial refinement are straightforward to implement. The application of mesh refinement allows obtaining high-order solutions even for bang-bang OCPs. To show the capabilities of the method, two test cases were considered: a continuous and a bang-bang time-solutions. A detailed study of the convergence properties of the method was performed for both cases. For that, a floating-point arithmetic library with arbitrary precision was used. The results demonstrated the expect problems of using double-precision arithmetic for polynomial approximations of degree above six. Finally, the method was successfully applied for optimal control of an OWC spar buoy wave energy converter. The results showed a 20% in the turbine output power in comparison with the standard non-optimal control.

Keywords: nonlinear optimal control, discontinuous Galerkin, shooting method, oscillating water column, Pontryagin's maximum principle

Contents

List of Tables

List of Figures

Nomenclature

Buoy nomeclature

$\dot{m}_{\mathrm{t}}$	Air mass flow (outlet of the turbine)
γ	Heat capacity ratio of air
Ω	Shaft rotation speed
Φ	Non-dimensional flow
Π	Non-Dimensional power
Ψ	Non-dimensional pressure head
ρ	Air density
ρ_{atm}	Air density at atmosferic pressure
ρ_{w}	Water density
$\mathbf{A}_{R_{ij}}$	Radiation state matrix affecting the radiation force of body i in body j
$\mathbf{B}_{R_{ij}}$	Radiation input matrix affecting the radiation force of body i in body j
$\mathbf{C}_{R_{ij}}$	Radiation output matrix affecting the radiation force of body i in body j
$\mathbf{M}$	Mass and added mass matrix
$\mathbf{s}_{ij}$	Radiation states affecting the radiation force of body i in body j
A_{ij}^{∞}	Added mass in body i due to body j
d	diameter of the turbine
$F_{\mathrm{d}i}$	Excitation force in body i
g	Gravity accelaration
h_0	Height of the air chamber at $x_1 = x_2 = 0$
I	Moment of intertia of the couple turbine-shaft-generator about axis of rotation
m_i	Mass of body i

p^* Dimesional pressure difference (to atmosferic)

p^* Nondimesional pressure difference (to atmosferic)

p_{atm} Atmosferic pressure

R_{ij} Radiation force in body i due to body j

S_i Surface are of body i

T_{gen} Torque of the generator

T_{t} Torque of the turbine

V Air chamber volume

v_1 Heave velocity of the buoy

v_2 Heave velocity of the water column

x_1 Heave position of the buoy

x_2 Heave position of the water column

Discontinuous Galerkin optimal control nomenclature

$\boldsymbol{\lambda}$ Vector of all co-states λ

Δt^e Time interval covered by element e

$\mathcal{H}$ Hamiltonian function $\text{IR}^n \rightarrow \text{IR}$

$\mathbf{f}$ State function $\text{IR}^n \rightarrow \text{IR}^n$

$\mathbf{u}$ Vector of all control variables u

$\mathbf{v}$ Arbitrary continuously differentiable function

$\mathbf{x}$ Vector of all state variables x

c_f Cost associated with a discrete state

I preformance index

J Functional

n_{p} number of polynomials on approximation

n_{x} number of state variables

t Time

t_{f}^e Final time of element e

t_{st}^e Starting time of element e

Buoy indexes

atm Referent to atmospheric values

gen Referent to the generator

t Referent to the turbine

i, j With respect to body i or j, where i or $j = 1$ corresponds to the buoy and i or $j = 2$ refers to the water column

Discontinuous Galerkin optimal control indexes

i, j Matrix/vector position (ex. $c_{1,2}$ at row with index 1 and collumn with index 2)

Mathematical operations and symbology

$\mathbf{\Phi}^{-1}$ Inverse matrix of $\mathbf{\Phi}$

$\dot{\phi}$ Time derivative of ϕ

$\frac{\mathrm{d}^n f}{\mathrm{d}\phi^n}$ n-th derivative of f (generic function) about ϕ (generic variable)

$\hat{\phi}$ Complete estimative of ϕ

$\int_a^b f \mathrm{d}\phi$ Integral of f, about ϕ between a and b

$\mathcal{J}_{\mathbf{f}}(\boldsymbol{\phi})$ Jacobian of vectorial function $\mathbf{f}$ about the vector $\boldsymbol{\phi}$

$\nabla_{\boldsymbol{\phi}} \delta$ Gradient of potential function δ about the vector $\boldsymbol{\phi}$

ϕ^T Transpose of $\mathbf{\Phi}$

Det($\mathbf{\Phi}$) Time derivative of ϕ

$\tilde{\phi}$ Local estimative of ϕ

Buoy supersctipts

∞ Calculated at ”inifite” frequency

Discontinuous Galerkin optimal control superscripts

e Referent to arbitrary element e

$e + 1$ Referent element imediatly after (in time) to arbitrary element e

$e - 1$ Referent element imediatly before (in time) to arbitrary element e

Chapter 1

Introduction

1.1 Motivation and Topic Review

Wave energy converters (WECs) still prove a challenge to engineering, either due to their inherent oscillatory behavior, which requires different forms of control to achieve higher efficiencies, or the complexity of the wave-device interactions, which motivates the existence of a manifold of technologies and devices for the extraction. A. F. de O. Falcão [1] did a thorough review of these. Penalba and Ringwood [2] also did a good review of WEC devices, along with a comparison of the used model's complexity.

The Oscillating Water Column (OWC) principle is one of many viable ways to extract energy from ocean waves. This principle uses the waves' oscillation to move a water column (inside a device, either floating or fixed), which changes the air pressure between a turbine and the environment. The pressure gradient forces air through the turbine and induces its rotor movement. The rotor of the turbine is attached to an electrical energy generator, which supplies power to the grid.

This principle was first introduced as an energy converter by Yoshio Masuda in 1947 [1]. The system was idealized with unidirectional turbine and control valves that would open according to the flow direction, redirection the air flow for the correct direction of the turbine. The most common design nowadays uses a self rectifying turbine, i.e. a turbine that keeps its sense of rotation independent of the sense of the air flow [3], thus eliminating the need for valves to control the flow direction. The most common turbine for this purpose is the Wells turbine (with guiding vanes) [4, 5] attached to a doubly fed induction generator (DFIG). The Wells turbine is used due to its simplicity and capability to deal with bi-directional flows [3]. Some designs replace the Wells turbine (reaction turbine) with an impulse turbine, which has been shown to increase the operating range with higher efficiency [3, 6]. A reaction turbine is kept rotating due to the deflection of air hitting the blades, while an impulse turbine relies on the impulse given by the fluid passing through the blades, figure 1.1 shows the most different designs of turbines [3].

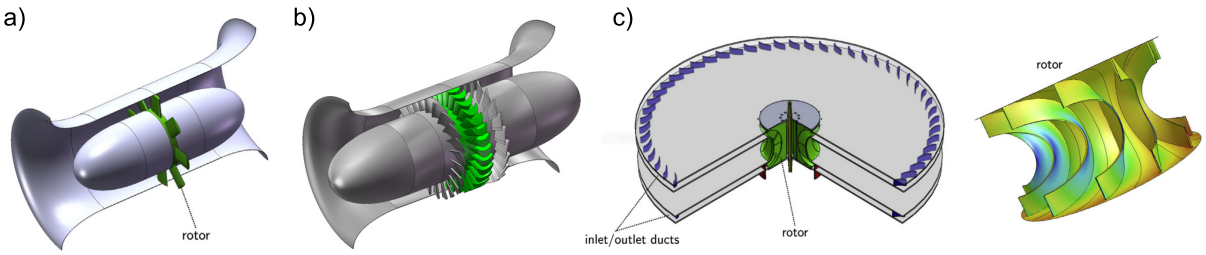


Figure 1.1: The most used turbine for OWC wave energy converters. a) Wells turbine. b) Axial impulse turbine. c) Bi-radial turbine.

Regarding control for OWCs, it is mostly implemented through the actuation of either a flow-limiting control valve or the electromagnetic power of the generator. There are also a variety of implemented strategies with different objectives. A common goal is to attain a steady power supply to the grid [7, 8]. Other control strategies focus on finding the best power output of the generator (power supplied to the grid) [9, 10]. Figure 1.2 shows a distribution of control strategies for a selection of 30 articles analyzed (Resultant from a search in EBSCO for the SU TERMS ”Oscillating water column” and ”Control”).

Recently, the Wave Energy Group of Instituto Superior Técnico (IST) developed a new self-rectifying bi-radial turbine with which proved to have a better performance than a Wells turbine, for this application [3, 6]. The turbine has been tested in laboratory and under real sea conditions at the Mutriku wave power plant, Basque Country, Spain. This turbine can be equipped with a simple high-speed stop valve that allows an open/closed operation as well as the partial closure of the valve. This thesis will only focus on the open/closed usage of the valve, which will be treated as a bang bang optimal control problem (OCP).

An upcoming method to solve OCP problems is the finite element method: Discontinuous Galerkin (DG-FEM). This FEM method was proposed by Reed and Hill [11] as a method to solve nonlinear differential equations. In this thesis, the DG-FEM with a mesh refinement technique will be used with the same goal as Henriques et al. [12] but with a different formulation, with a specific focus on the bang-bang control problems, as they are troublesome to pseudo-spectral methods.

The DG-FEM method to solve OCP problems, takes advantage of a discretization of the problem time domain element-wise and the weak form of the differential equations, which is very well known of most used finite element methods [13]. These characteristics ar combined a to achieve high order of accuracy and convergence characteristics [14]. Furthermore, the refinement of the time mesh is comes naturally and the discontinuities (like the solutions of bang-bang OCPs) are easily handled because the boundary between each element has a discontinuity.

Nowadays, the most established numerical solutions to optimal control problems, are the pseudo-spectral methods (PS), which were first introduced in the 1990s Elnagar et al. [15] and have gathered popularity, to the point of being the standard continuous-time OCP solving method, in research and industry [16]. These methods present high order of accuracy for smooth OCP [16].

The PS methods use polynomials to approximate the entire time domain of the OCP, where each polynomial used is pondered by a variable coefficient. These coefficients are computed according to the

problem; using the minimization of a norm calculated in a chosen quadrature (preferably, Gauss-Lobatto or Chebyshev–Gauss–Lobatto [16]). The refinement of solutions is made by increasing the number of polynomials and the number of points in the quadrature. Since the the variables are approximated by polynomials (which are continuous and smooth), these methods are bound to have accuracy problems for discontinuous our non smooth OCPs.

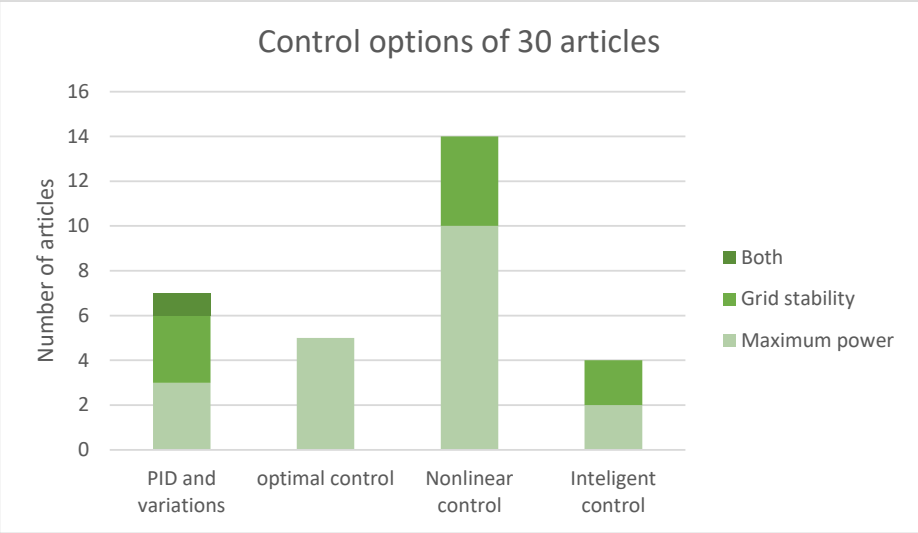


Figure 1.2: Control options distribution for 30 articles

1.2 Objectives

This thesis has the main goal of formulating and solving the optimal bang-bang control problem for the OWC spar buoy described in the motivation and topic review (section 1.1), where the control will rely, solely, on the high-speed stop valve. The valve, as said before, will only be used with open and close configurations. Remember that bang-bang control problems are a special type of on-off control where the objective function and state functions are linear with the control variable. This problem has been solved before by Henriques et al. [12] and Valério [17], but the bang-bang control showed a chattering problem for both cases (closing and opening the valve many times in a short time interval).

This work is an attempt to eliminate the previously observed chattering problems. This requires the development of an environment capable of solving optimal control problems of fixed initial conditions and terminal time, where the control may be of bang-bang type. The proposed method to solve this control problem is the finite element method (FEM), namely the discontinuous Galerkin (DG-FEM) method. This method will provide a better approximation to discontinuous functions (since a discontinuity is allowed at each element boundary) and will increase the capacity of refining the solutions when compared to other continuous time methods like the spectral methods. These methods approximate the whole time domain as a sum of several polynomials. Ross and Karpenko [16] wrote a comprehensive review on

this topic for a wide range of control problems. But this type of numerical formulation inherently fails to approximate discontinuous functions.

The developed method (DG-FEM) will follow a different approach to [12] and [17] and should allow the transition for any optimal control problem of this type. With this in mind, it is important not only to test the solver with the OWC spar buoy but with other systems as well, with a special focus on validating and proofing. It is expected that the optimal control solver environment will be able to numerically solve any other optimal control problems, with these type of restrictions.

1.3 Thesis Structure

This thesis is divided into 7 chapters including the introduction, where a short overview of the topic, the motivation and the objectives are presented.

Chapter 2 provides a review of the OWC spar buoy system and its dynamics, where the main physical and numerical principles will be described extensively, along with some remarks and simplifications. The assumptions for the optimal control of the OWC spar buoy will also be made here. The model described in this section will follow linear water wave theory and access the nonlinearities of the pressure, the turbine and the generator.

The third chapter offers the mathematical basis for the numerical formulation of the optimal control, the DG-FEM method and the algorithms that join these; namely: forward integration of the state variables and control followed by backward integration of the adjoint states and the shooting method.

Chapter 4 shows, graphically, the algorithms and numerical methods developed in chapter 3. This is done using flowcharts for each of the modules. These flowcharts are preceded by a short explanation to help following the schemes.

The fifth chapter is dedicated to the testing and validation of the algorithms shown in chapter 4. Initially, a continuous nonlinear control problem is solved, where the algorithms are tested for their precision and matched with the machine precision. Then follows the solution of a bang-bang problem, where the importance of refinement is accessed. Still, on this problem, the sensitivity of the shooting method is evaluated.

Chapter 6 tackles the problem of the optimal control of the OWC spar buoy, with one single wave and three waves with different frequencies and amplitudes.

Finally, in chapter 7 the conclusions are taken, regarding the methods and results of the control, followed by a description of possible future work that may be done to add or complete this thesis.

After the main body of the thesis, appendix A also presents some of the work done, but this work was not used for any of the parts of the previous formulation. Appendix B presents the tables for the excitation force of the waves and appendix C shows the documentation for the code developed in python, from the extensive description of each module to practical examples of their use.

Chapter 2

OWC Spar Buoy System

The OWC spar buoy system consists of a long, hollow and axisymmetric buoy with a Turbine and a generator attached to its head, approximately as it is shown in figure 2.1. Its long shape eliminates some of the radiation terms resultant from the interaction of the waves with the water column and makes the approximation of the water column to a rigid piston more viable. The axisymmetry assures the body will have the same behavior independently of the wave direction. Note, as well, the buoy has a mounted weight at the end of the tail. This weight is a method to reduce pitch and roll rotation modes. In general, these rotation modes are the more problematic movement modes of these types of devices [18].

As stated before the oscillating water column principle (OWC) is a method to extract energy from ocean waves (WEC). The Energy scheme 2.2 suggests, several key components ensure the energy from the waves is transmitted to the power grid. The ocean waves interact with the OWC spar buoy by diffraction F_{d_i}, on the other hand, the OWC spar buoy exchanges energy to the waves with the radiation forces. Then, the air chamber, located between the water column and turbine (see figure 2.1) receives the energy from the water column with the compression and decompression of air. The OWC spar buoy is affected by the air chamber due to impulsion effects (caused by the pressure p). Then the energy going to the air chamber is accumulated in the air pressure, which is traded with the turbine, in exchange for mass outtake or mass intake to the chamber. The turbine, then, transforms the mass flow and pressure into torque. Of course, part of the mass and pressure is lost to the atmosphere. The torque given by the turbine, along with a control torque provided by the generator will rotate both the generator and the turbine rotors, as they are rigidly coupled. In the presence of torque and rotational speed, the generator is then able to supply that power to the grid.

This chapter presents the Spar Buoy theoretical system with a description of the energy transmission effects. The formulation of the spar buoy dynamics present in this chapter mainly follows the time domain approach already developed by Valério [17] and Henriques et al. [19].

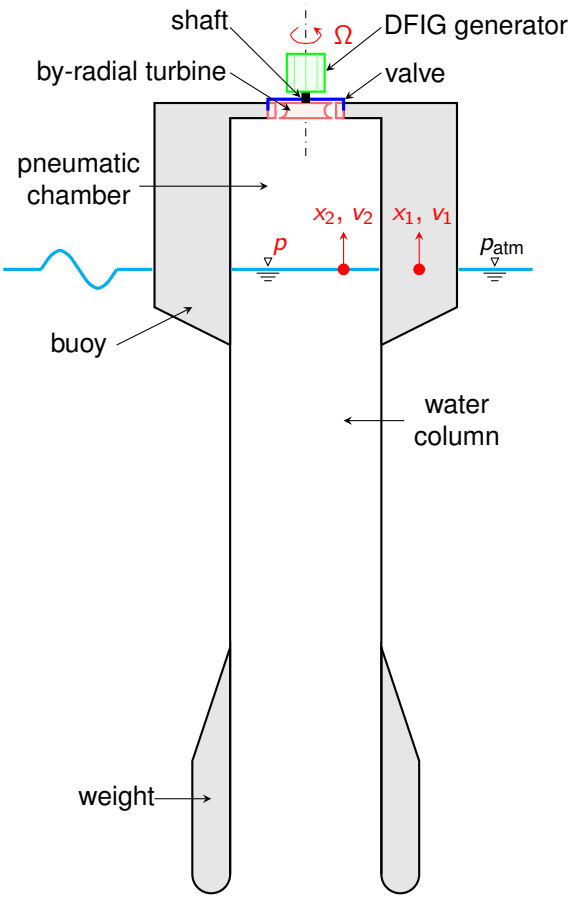


Figure 2.1: Cross section of the buoy scheme.

2.1 Physical Model of the Spar Buoy

2.1.1 Hydrodynamics

The OWC spar buoy is a two-body system composed of a floater and tail tube filled with water. The floater and the OWC free-surface are denoted as body 1 and 2, respectively, as depicted in Fig. 2.1. For computing the hydrodynamic coefficients in the frequency domain, body 2 is modeled as an imaginary neutrally buoyant rigid piston.

The discussed model will only consider the heave motion of the buoy (up and down motion). Furthermore, it is assumed that the surface of the inside water column is plane and horizontal. The vertical position of body i is named x_i, with the x-axes pointing upward. At the equilibrium position $x_i = 0$. With the previous assumptions, the first two equations of the buoy's dynamics, describing the heave motion, are:

$$(m_1 + A_{11}^{\infty})\ddot{x}_1 + A_{12}^{\infty}\ddot{x}_2 = -\rho_w g S_1 x_1 + S_2 p_{atm} p^* + F_{d_1} - R_{11}(\dot{x}_1) - R_{12}(\dot{x}_2), \tag{2.1}$$

$$A_{21}^{\infty}\ddot{x}_1 + (m_2 + A_{22}^{\infty})\ddot{x}_2 = -\rho_w g S_2 x_2 - S_2 p_{atm} p^* + F_{d_2} - R_{21}(\dot{x}_1) - R_{22}(\dot{x}_2), \tag{2.2}$$

where the mass of body i is denoted as m_i. The limiting value at infinite frequency of the added mass of body i as affected by the motion of body j is defined by A_{ij}^{∞}. The buoy and the inside OWC are under the

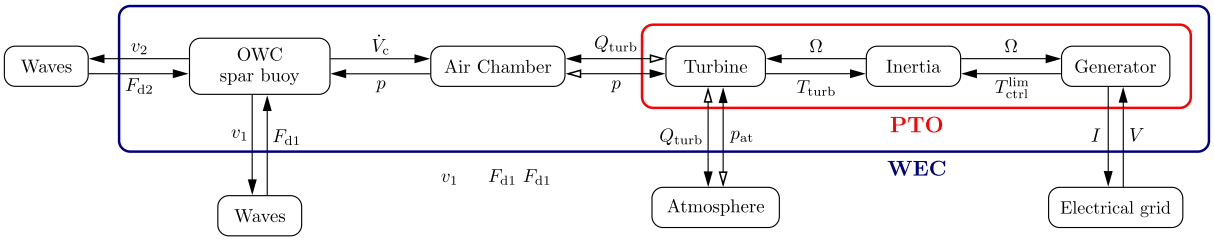


Figure 2.2: Wave-to-wire power-flow on an OWC wave energy converter. The bidirectional power-flow between the air chamber, the turbine and the atmosphere is represented by double arrows. In the figure, V and I stand for voltage and electrical current, respectively. Adapted from [4].

effect of excitation F_{d_i}, radiation R_{ij}, and the hydrostatic restoring forces $\rho_w g S_i x_i$, where ρ_w is the water density, g the gravity acceleration and S_i the area of the section parallel to the water surface for each body. The force resulting from the air chamber pressure is represented by the term $S_i p_{atm} p^*$, p_{atm} is the atmospheric pressure, and p^* is the dimensionless relative pressure inside the air chamber defined by

$$p^* = \frac{p - p_{atm}}{p_{atm}}. \tag{2.3}$$

Here p is the absolute pressure inside the air chamber.

The radiation forces are defined by the terms $A_{ij}^{\infty} \ddot{x}_j + R_{ij}$. R_{ij} are terms defined by a convolution integral:

$$R_i = \int_0^t K_{ij}(t-s)\dot{x}_j(s)ds. \tag{2.4}$$

For simplification, in this thesis, R_{ij} was be calculated as viscous dampers: $R_{12} = R_{21} = 0$ and $R_{11} = c_1\dot{x}_1$, $R_{22} = c_2\dot{x}_2$. This is possible due to the reduced order of magnitude of the cross terms (R_{12}, R_{21}) with respect to the diagonal terms (R_{11}, R_{22}).

The wave excitation forces are represented by F_{d_i}. With the assumption of linear water wave theory, it can be computed as:

$$F_{d_i} = \sum_{m=1}^{N} \Gamma_i(\omega_m) A(\omega_m) \cos(\omega_m t + \phi_{i_m} + \phi_r), \tag{2.5}$$

i.e. the sum of several regular waves with different angular frequency. Γ_i represents the excitation force coefficient for body i, A the frequency dependent wave amplitude, ϕ_{m_i} is the frequency response of body i and ϕ_r is just a random phase [19]. $A(\omega_m)$ may be calculated with:

$$A = \sqrt{\Delta\omega_m S_{\omega_m} \omega_m}. \tag{2.6}$$

where S_{ω_m} is the power spectral density of the wave climate and $\Delta\omega_m$ is a frequency interval resultant from the discretization of the frequency spectrum. For a given $\Delta\omega_m$ the correspondent ω_m should be placed in the middle of the interval:

$$\omega_m = \omega_{m-1} + \frac{1}{2}(\Delta\omega_{m-1} + \Delta\omega_m). \tag{2.7}$$

The power spectral density S_{ω_m} may be given by Pierson-Moscowitz formula:

$$S_{\omega_m} = 262.9 \frac{H_s^2}{\omega_m^5 T_e^4} e^{-\frac{1054}{(\omega_m T_e)^4}}, \tag{2.8}$$

where T_e represents the wave energy period and H_s the significant wave height (which are referent to the wave climate). The tables for $\Gamma(\omega)$ and ϕ_ω are present in annex B.

To transform the system in a state space representation, let $v_1 = \dot{x}_1$ and $v_2 = \dot{x}_2$. Introducing the total mass matrix (2.9), the closed form of equations (2.1) and (2.2) is derived as:

$$M = \begin{bmatrix} \mathrm{m}_1 + \mathrm{A}_{11}^{\infty} & \mathrm{A}_{12}^{\infty} \\ \mathrm{A}_{21}^{\infty} & \mathrm{m}_2 + A_{22}^{\infty} \end{bmatrix} \tag{2.9}$$

where

$$\dot{v}_1 = \mathrm{Det}(M)^{-1}\Big((m_2 + A_{22}^{\infty})\mathcal{F}_1 - A_{12}^{\infty}\mathcal{F}_2\Big), \tag{2.10}$$

$$\dot{v}_2 = \mathrm{Det}(M)^{-1}\big((m_1 + A_{11}^{\infty})\mathcal{F}_2 - A_{21}^{\infty}\mathcal{F}_1\big), \tag{2.11}$$

$$\mathcal{F}_1 = -\rho_{\mathrm{w}} g \mathrm{S}_1 x_1 + S_2 p_{\mathrm{atm}} p^* + F_{\mathrm{d}_1} + R_{11}(v_1) + R_{12}(v_2),$$

$$\mathcal{F}_2 = -\rho_{\mathrm{w}} g \mathrm{S}_2 x_2 - S_2 p_{\mathrm{atm}} p^* + F_{\mathrm{d}_2} + R_{21}(v_1) + R_{22}(v_2).$$

2.1.2 Pneumatic Chamber

The pneumatic chamber is located between the water column and the turbine. For the developed model, the water column will act as a rigid ”piston”, whose dynamics are already described in (2.10) and (2.11) (x_1, x_2, v_1, v_2). The turbine will then acquire some of the pneumatic energy and allow the movement of mass from the system $\dot{m}_{\mathrm{t}}$ (which is defined in subsection 2.1.3. Below, there is a short demonstration for the dynamics equation of the pressure accounting with compressibility effects and starting on the classic continuity equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\rho V) = -\dot{m}_{\mathrm{t}} \tag{2.12}$$

$$\dot{\rho} V + \rho \dot{V} = -\dot{m}_{\mathrm{t}} \tag{2.13}$$

The volume of the air chamber V and its variation can be defined as:

$$V = (h_0 + x_1 - x_2) S_2, \tag{2.14}$$

for which, h_0 is the height of the chamber at the equilibrium position. The compression/decompression of air is assumed to be an isentropic process and the air is modeled as a perfect gas:

$$\frac{p}{\rho^{\gamma}} = \frac{p_{\mathrm{atm}}}{\rho_{\mathrm{atm}}^{\gamma}} \tag{2.15}$$

where γ is the specific heat ratio for air (1.4) and ρ_{atm} is the air density at atmospheric pressure. Rearranging:

$$\rho = \rho_{\text{atm}} \left(\frac{p}{p_{\text{atm}}} \right)^{1/\gamma} = \rho_{\text{atm}} (p^* + 1)^{1/\gamma} \tag{2.16}$$

$$\dot{\rho} = \rho_{\text{atm}} \frac{1}{\gamma} (p^* + 1)^{1/\gamma - 1} \dot{p}^* \tag{2.17}$$

From the algebraic manipulation of equation (2.13) and acknowledging the definitions in (2.14) and (2.16), results expression

$$\dot{p}^* = -\frac{\gamma}{\rho S_2} \frac{\dot{m}_{\text{t}} (p^* + 1)}{h_0 + x_1 - x_2} - \gamma \frac{(p^* + 1)(\dot{x}_1 - \dot{x}_2)}{h_0 + x_1 - x_2} \tag{2.18}$$

that shows, explicitly, all the variables and constants, or equation

$$\dot{p}^* = -\gamma \left(\frac{\dot{m}_{\text{t}}}{\rho V} + \frac{\dot{V}}{V} \right) (p^* + 1) \tag{2.19}$$

which is physically more intuitive. p^* is the fifth state variable. Note that for this non-dimensional pressure definition, the time variation is the same as the pressure (excluding the p_{atm} constant ($\dot{p} = p_{\text{atm}} \dot{p}^*$)).

With these expressions, the system's first non-linearity is, finally, introduced.

2.1.3 Power Take-off System Dynamics

The final power transmission to the generator is assured by a shaft coupling turbine and generator. The dynamics equation for the rotation of the shaft is:

$$I\dot{\Omega} = T_{\text{t}} - T_{\text{gen}} \tag{2.20}$$

in which, T_{t} is the Torque supplied by the turbine, T_{gen} is the torque absorbed by the generator, I is the moment of inertia along the axis of rotation and Ω is the rotational speed of the shaft. T_{t} and T_{gen} are usually a function of the flow characteristics, and type of devices and their size. It is typical, when dealing with turbo-machines such as the bi-radial turbine, to normalize the variables, by the characteristics of the flow and the machine (like the rotor diameter d). The expressions for the dimensionless pressure head Ψ, flow Φ and power Π are:

$$\Psi = \frac{p^* p_{\text{atm}}}{\rho_{\text{t}} \text{d}^2 \Omega^2} \qquad \Phi = \frac{\dot{m}_{\text{t}}}{\rho_{\text{t}} \text{d}^3 \Omega} \qquad \Pi = \frac{P_{\text{t}}}{\rho_{\text{t}} \text{d}^5 \Omega^3} \tag{2.21}$$

$$p^* = \frac{\Psi \rho_{\text{t}} \text{d}^2 \Omega^2}{p_{\text{atm}}} \qquad \dot{m}_{\text{t}} = \rho_{\text{t}} d^3 \Omega \qquad P_{\text{t}} = \Pi \rho_{\text{t}} \text{d}^5 \Omega^3 \tag{2.22}$$

For the dimensionalization just apply the inverse transformation (2.22), being p^* the non dimensional pressure, $\dot{m}_{\text{t}}$ the mass air flow and P_{t} the turbine power. To obtain T_{t} for expression (2.20) remember that $T_{\text{t}} = \frac{P_{\text{t}}}{\Omega}$. On these expressions, it is also present the term ρ_{t} this air density is different from ρ in the air chamber (2.18, 2.19), since it is correspondent to the density of the inlet/outlet of air through the

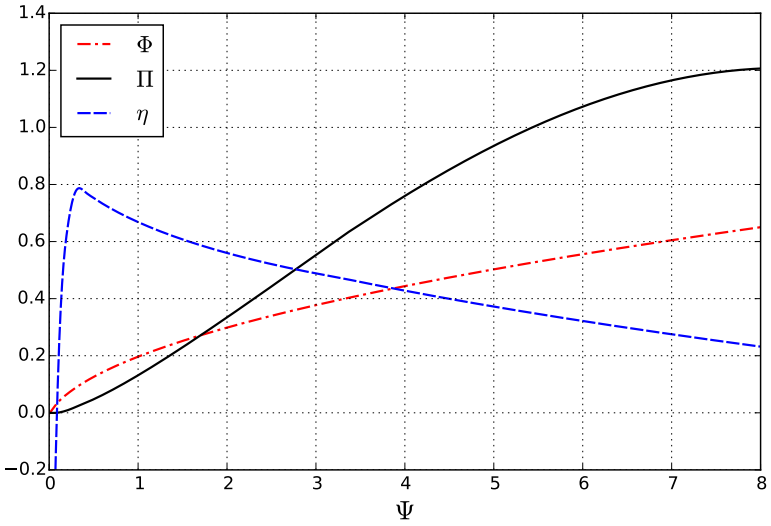


Figure 2.3: Turbine curves

turbine:

$$\rho_t = \begin{cases} (p^* + 1)^{\frac{1}{\gamma}} \rho_{atm}, p^* > 0 \\ \rho_{atm}, \text{otherwise} \end{cases} \tag{2.23}$$

To calculate T_t and $\dot{m}_t$ the data from [20] can be used. Figure 2.3 shows the curves of the data regarding the bi-radial turbine.

In fact, the data that was used can be seen in annex A.1 where there is a more detailed version of these curves, contempleting the option of partial closure of the valve. The curves in figure 2.3 were generated with the valve completely open. The expressions used to model the turbine for the bang-bang problem were:

$$\hat{\Phi}(\Psi, u) = \frac{0.12695\Psi^4 - 0.71\Psi^3 + 5.068\Psi^2 + 4.289\Psi}{\Psi^3 - 2.561\Psi^2 + 37.46\Psi + 6.278} \tag{2.24}$$

$$\hat{\Pi}(\Phi) = -272\Phi^{10} + 252\Phi^8 - 84.26\Phi^6 + 12.9\Phi^4 + 2.605\Phi^2 - 0.00657 \tag{2.25}$$

The fitting error for this curves can be seen as well in annex A.1.

Without the effect of partial valve closure in annex A, the control problem may be transformed into a specific type of on-off control: bang-bang control problem. For these types of problems, the state function and the objective function need to be linear with the control [21].

The control for the buoy is the valve u, with $u = 1$ corresponding the the open valve and $u = 0$ to the closed valve. The adapted problem for the application of the bang-bang principle, requires

$$T_t = u\Pi\Omega^2 d^5 \rho_t \tag{2.26}$$

$$\dot{m}_t = u\Phi\Omega d^3 \rho_t \tag{2.27}$$

and for the maximization of the turbine power

$$P_t = u\Pi\Omega^3 d^5 \rho_t \tag{2.28}$$

The bang-bang control problems will be better discussed in chapter 4.

Regarding the generator electromagnetic torque, equation (2.22) shows that the turbine output power should be proportional to Ω^3 if the time-averaged turbine aerodynamic efficiency is to be maximized. In practice, if the coupling between the turbine aerodynamics and the spar-buoy OWC hydrodynamics is taken into account, we can use a relation of the type [22]

$$P_{gen}^{opt} = a\Omega^b \tag{2.29}$$

where b is about 3 (in fact 3.33 was used) and the constant a used was 0.025. To obtain the torque for the optimal power, just aknowledge that

$$T_{gen} = P_{gen}/\Omega = a\Omega^{b-1} \tag{2.30}$$

Chapter 3

Optimal Control Solution Using a Discontinuous Galerkin Finite Element Method

In this section a viable solver is developed for linear and nonlinear optimal control problems, with fixed terminal time and in continuous time (finite elements). First, it starts with a review of optimal control theory, then follows a review of the finite element method, specifically the Discontinuous Galerkin method (DG-FEM), and finally the integration of optimal control and finite elements. It provides the base for the next chapter where this theory is validated.

3.1 Optimal Control Theory

The original Pontryagin's Maximum Principle (PMP) focuses on finding an optimal path for a set of variables for the maximization of a specified performance index J given by a final state cost $\mathbf{c}_f$ and a Lagrangian function $\mathcal{L}$ pondered over time. This originates the problem described by the following expressions [21][17]:
Maximize:

$$J = c_f\big(\mathbf{x}(t_f)\big) + \int_{t_0}^{t_f} \mathcal{L}(t, \mathbf{x}, \mathbf{u})\,\mathrm{d}t, \tag{3.1}$$

subject to:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \tag{3.2}$$

and the boundary condition:

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{3.3}$$

where $\mathbf{x}$ is the array of state variables, $\mathbf{u}$ are the control variables and t is the time. The use of **bold**

symbolizes vectors and **bold** capital letters matrices. By default vectors are column. c_f is the cost associated with final states, t_0 and t_f are the initial and final time of the problem respectively. The upper dot denotes time derivative.

Now, we may add the term $-\int_{t_0}^{t_f} \boldsymbol{\lambda}^T(\dot{\mathbf{x}} - \mathbf{f}(t, \mathbf{x}, \mathbf{u}))\,\mathrm{d}t$ to the performance index J:

$$I = c_f\big(\mathbf{x}(t_f)\big) + \int_{t_0}^{t_f} \mathcal{L}\,\mathrm{d}t - \int_{t_0}^{t_f} \boldsymbol{\lambda}^T(\dot{\mathbf{x}} - \mathbf{f}(t, \mathbf{x}, \mathbf{u}))\,\mathrm{d}t, \tag{3.4}$$

where $\boldsymbol{\lambda}$ are Lagrange multipliers, also called co-state variables. This term ensures the systems dynamics (3.2) are satisfied, since for optimal I: $\int_{t_0}^{t_f} \boldsymbol{\lambda}^T(\dot{\mathbf{x}} - \mathbf{f}(t, \mathbf{x}, \mathbf{u}))\,\mathrm{d}t = 0$. Re-arranging (3.4) results in:

$$I = c_f\big(\mathbf{x}(t_f)\big) + \int_{t_0}^{t_f} \big(\mathcal{L}(t, \mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^T\mathbf{f}(t, \mathbf{x}, \mathbf{u})\big)\,\mathrm{d}t - \int_{t_0}^{t_f} \big(\boldsymbol{\lambda}^T\dot{\mathbf{x}}\big).\,\mathrm{d}t \tag{3.5}$$

Now the Hamiltonian function can be defined as

$$\mathcal{H} = \mathcal{L}(t, \mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^T\mathbf{f}(t, \mathbf{x}, \mathbf{u}) \tag{3.6}$$

and I becomes:

$$I = c_f\big(\mathbf{x}(t_f)\big) + \int_{t_0}^{t_f} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})\,\mathrm{d}t - \int_{t_0}^{t_f} \big(\boldsymbol{\lambda}^T\dot{\mathbf{x}}\big)\,\mathrm{d}t \tag{3.7}$$

To remove the time derivative of $\mathbf{x}$ in this expression, integrate by parts the last term: $\int_{t_0}^{t_f} \big(\boldsymbol{\lambda}^T\dot{\mathbf{x}}\big)\,\mathrm{d}t$. Which results in:

$$I = c_f\big(\mathbf{x}(t_f)\big) - \boldsymbol{\lambda}^T(t_f)\mathbf{x}(t_f) + \boldsymbol{\lambda}^T(t_0)\mathbf{x}(t_0) + \int_{t_0}^{t_f} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})\,\mathrm{d}t + \int_{t_0}^{t_f} \big(\dot{\boldsymbol{\lambda}}^T\mathbf{x}\big)\,\mathrm{d}t. \tag{3.8}$$

If $\mathbf{u}$ is the optimal control, then $\mathbf{x}$ will also need to follow the optimal path, which implies that $\nabla_{\mathbf{x}} I$ and $\nabla_{\mathbf{u}} I$ need to be 0 (∇ represents the gradient operator and $\nabla_{\boldsymbol{\phi}} f$ the gradient of f along $\boldsymbol{\phi}$: $\frac{\partial f}{\partial \phi_i}$). Remembering that $\mathbf{x}(t_0)$ is fixed and its derivative about $\mathbf{x}$ is 0, the expressions for this condition stay:

$$\nabla_{\mathbf{u}} I = \int_{t_0}^{t_f} \nabla_{\mathbf{u}} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})\,\mathrm{d}t = 0, \tag{3.9}$$

$$\nabla_{\mathbf{x}} I = \nabla_{\mathbf{x}} c_f\big(\mathbf{x}(t_f)\big) - \nabla_{\mathbf{x}}(\boldsymbol{\lambda}^T\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \nabla_{\mathbf{x}} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) + \dot{\boldsymbol{\lambda}}\,\mathrm{d}t = 0. \tag{3.10}$$

Forcing the conditions (3.9) and (3.10) to be true for any time, then at the final time:

$$\boldsymbol{\lambda}(t_f) = \nabla_{\mathbf{x}(t_f)} c_f\big(\mathbf{x}(t_f)\big) \tag{3.11}$$

and at any intermediate time

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\mathbf{x}} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}), \tag{3.12}$$

$$0 = \nabla_{\mathbf{u}} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}), \tag{3.13}$$

Additionally, if $\mathbf{x}$ and $\mathbf{u}$ are optimal, Lagrange multipliers will also need to be optimal. A way to ensure

they are is to recall (3.7) and make $\nabla_{\boldsymbol{\lambda}} I = 0$ for any time instant:

$$\dot{\mathbf{x}} = \nabla_{\boldsymbol{\lambda}} \mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) \tag{3.14}$$

The PMP states the maximization of $\mathcal{H}$ can be done in these three stages: solving the state variables, through expression (3.14); solving the co-state variables, with the solution of (3.12) (Lagrange multipliers) and solving the optimal control equation (3.13). Expression (3.13) changes for the case where $\mathbf{u}$ as restrictions to a more generic form $\mathbf{u} : \mathcal{H} = \max(\mathcal{H})$

Expression (3.11), is the boundary condition for $\boldsymbol{\lambda}$. With a boundary condition placed at t_f the most natural way to solve equation (3.12) is to integrate it backwards in time (from t_f to t_0). The first method proposed in this thesis for the PMP problem is to solve the control and the state variables forward in time and then solving the co-state variables backward (iteratively).

An alternative to the forward and backward integration can be found in Lewis et al. [21] book. Where the problem is treated as a two-point boundary condition differential equation problem. This formulation is also developed in this thesis, even though, for the generic case, the numerical solution for the coupled systems (3.14)-(3.12) does not contemplate a changing variable through iterations (namely $\mathbf{u}$) [23].

3.2 Discontinuous Galerkin Finite Element Method

The problem depicted in the previous section 3.1 requires the solution for two vectorial differential equations which describe a system in state-space form: (3.14) and (3.12). These equations do not always have an analytical solution or the solution may be less accurate or convenient than the numerical. So for that purpose, on the course of this work, the Discontinuous Galerkin Finite Element Method (DG-FEM) will be used [13],[14], whose formulation can be followed in (3.15) through (3.25), for a generic state-space system (3.15).

The formulation with finite elements is particularly useful if the grid (points in time) is large to the point where the sampling has the same frequency as some of the systems' dynamics. Solving these problems with a discrete time method is also viable, but the choice of time intervals needs to be well planned. Even so, it is possible to miss some detail like the exact switching time of a bang-bang control problem (time where the control changes from 1 to 0 or vice-versa), which would require a sampling time refinement that is more easily made with a finite element formulation (see section 4.2 for a solution to this kind of problem). Spectral methods can be used as well to solve optimal control problems [16], but it is harder to deal with on/off nonlinearities when the problem is looked at from the whole domain, as is the case of the spectral formulation. The weak form, that is used for the formulation of FEM may be also very useful for some problems as it turns the constraints weaker and may offer a better solution for the problems.

3.2.1 DG-FEM formulation

Assume there is a system formulated by equation (3.15). Recall that: **x** is the array of state variables; **u** is the array of control variables; t is the time and **f** is the state function (vectorial function with the same length as **x**), which in expression (3.15) is nonlinear and time-dependent.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \text{where } \mathbf{u}(t) \text{ is known.} \tag{3.15}$$

Multiplying (3.15) by a continuous smooth function v and integrating along the domain (t_0 to t_f):

$$\int_{t_0}^{t_f} \dot{\mathbf{x}}\mathbf{v}\,\mathrm{d}t = \int_{t_0}^{t_f} \mathbf{f}\mathbf{v}.\,\mathrm{d}t \tag{3.16}$$

Defining now an approximation for **x** as

$$\hat{\mathbf{x}}(t) = \sum_{e=1}^{n_{\text{elem}}} \tilde{\mathbf{x}}^e(t), \tag{3.17}$$

$$\tilde{\mathbf{x}}^e(t) = \begin{cases} \mathbf{C}^e\mathbf{p}(\tau^e) & \text{, if } t_{\text{st}}^e < t < t_{\text{st}}^e + \Delta t^e \\ \mathbf{0} & \text{, otherwise} \end{cases}, \tag{3.18}$$

with:

$$\mathbf{C}^e = c_{i,j}^e, \qquad 1 \leq i \leq n_{\text{x}}, \tag{3.19}$$

$$\mathbf{p}(\tau^e) = p_j(\tau^e), \qquad 0 \leq j \leq n_{\text{p}}, \tag{3.20}$$

$$\tau^e = \frac{2t - t_{\text{st}}^e - t_{\text{f}}^e}{\Delta t^e}, \tag{3.21}$$

$$\Delta t^e = t_{\text{f}}^e - t_{\text{st}}^e, \tag{3.22}$$

where $\mathbf{p} = p_i$ (polynomials of degree i, $0 \leq i \leq n_{\text{p}}$) and n_{p} the number of polynomials (which affects the quality of the approximation), t_{st}^e is the starting time of element e, t_{f}^e the final time and Δt^e the time interval covered by element e, also, n_{x} represents the number of state variables. τ^e, the local time, is defined in expression (3.21). Matrix $\mathbf{C}^e$ contains constants where each line corresponds to the approximation of one **x** and each column multiplies with a polynomial of a certain degree, as shows equation (3.18).

Function **v** is approximated by a set containing the same polynomials as the ones used in the approximation ($\tilde{\mathbf{v}} = p_i$). Substituting, as well, **x** in (3.16) by (3.18), divides the integral by elements. The calculations for each element are:

$$\frac{\Delta t^e}{2}\int_{-1}^{1} \frac{2}{\Delta t^e}\tilde{\mathbf{x}}^e\tilde{\mathbf{v}}^{\mathsf{T}}\,\mathrm{d}\tau^e = \frac{\Delta t^e}{2}\int_{-1}^{1} \tilde{\mathbf{f}}^e\tilde{\mathbf{v}}^{\mathsf{T}}\,\mathrm{d}\tau^e. \tag{3.23}$$

The left hand side of the equation integrated by parts originates the boundary condition. Rearranging the terms we get the final expressions (per element) (3.24), for forward integration and (3.25), for backwards

integration.

$$-\tilde{\mathbf{x}}^e(1)\big(\tilde{\mathbf{v}}(1)\big)^{\mathsf{T}} + \int_{-1}^{1} \tilde{\mathbf{x}}^e \dot{\tilde{\mathbf{v}}}^{\mathsf{T}}\, \mathrm{d}\tau^e = -\int_{-1}^{1} \mathbf{f}(\tilde{\mathbf{x}}^e, \mathbf{u}, \tau^e)\tilde{\mathbf{v}}^{\mathsf{T}}\, \mathrm{d}\tau^e - \hat{\mathbf{x}}(t_f^{e-1})\big(\tilde{\mathbf{v}}(-1)\big)^{\mathsf{T}}, \tag{3.24}$$

$$\tilde{\mathbf{x}}^e(-1)\big(\tilde{\mathbf{v}}(-1)\big)^{\mathsf{T}} + \int_{-1}^{1} \tilde{\mathbf{x}}^e \dot{\tilde{\mathbf{v}}}^{\mathsf{T}}\, \mathrm{d}\tau^e = -\int_{-1}^{1} \mathbf{f}(\tilde{\mathbf{x}}^e, \mathbf{u}, \tau^e)\tilde{\mathbf{v}}^{\mathsf{T}}\, \mathrm{d}\tau^e + \hat{\mathbf{x}}(t_0^{e+1})\big(\tilde{\mathbf{v}}(1)\big)^{\mathsf{T}}. \tag{3.25}$$

In equation (3.24), $\hat{\mathbf{x}}(t_f^{e-1})$ is the approximation of $\mathbf{x}$ evaluated at the end of the previous element ($\tilde{\mathbf{x}}^{e-1}(1)$). On the other hand, in (3.25), $\hat{\mathbf{x}}(t_0^{e+1})$ represents the approximation of $\mathbf{x}$ calculated at the beginning of the next element ($\tilde{\mathbf{x}}^{e+1}(-1)$). Figure 3.1 shows a scheme of the position of the boundary conditions for each of the cases.

(a) Forward (b) Backward

Figure 3.1: Element time definitions for forward and backward integration

3.2.2 Linear System Simulation with DG-FEM

Expressions (3.26) and (3.27) represent the calculations that are required for the simulation of a linear system of the type $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)$, derived from (3.24) and (3.25) respectively.

$$\Big(\mathbf{P}(1) + \mathbf{D} + \frac{\Delta t^e}{2}\mathbf{I}_A\Big)\mathbf{a} = -\frac{\Delta t^e}{2}\mathbf{I}_B + \mathbf{BC}(-1)\hat{\mathbf{x}}(t_f^{e-1}), \tag{3.26}$$

$$\Big(\mathbf{P}(-1) - \mathbf{D} - \frac{\Delta t^e}{2}\mathbf{I}_A\Big)\mathbf{a} = \frac{\Delta t^e}{2}\mathbf{I}_B + \mathbf{BC}(1)\hat{\mathbf{x}}(t_0^{e+1}), \tag{3.27}$$

With the following definitions:

$$\begin{aligned}
&\mathbf{\Phi}(\tau) = \mathbf{p}(\tau)\big(\mathbf{p}(\tau)\big)^{\mathsf{T}};\ \mathbf{M} = \int_{-1}^{1} \dot{\mathbf{p}}\mathbf{p}^{\mathsf{T}} d\tau,\\
&\mathbf{P}(\tau) = \mathrm{diag}(\mathbf{\Phi}(\tau), \cdots, \mathbf{\Phi}(\tau)),\ n_{\mathsf{x}} \text{ blocks},\\
&\mathbf{D} = \mathrm{diag}(\mathbf{M}, \cdots, \mathbf{M}),\ n_{\mathsf{x}} \text{ blocks},\\
&\mathbf{I}_A = \int_{-1}^{1} \mathbf{\Phi}\mathbf{A}_{i,j}\, \mathrm{d}\tau\ ;\ \mathbf{I}_B = \int_{-1}^{1} \mathbf{p}B_i\, \mathrm{d}\tau\ ;\ 1 \le i \le n_{\mathsf{x}},\ 1 \le j \le n_{\mathsf{x}},\\
&\mathbf{BC}(\tau) = \mathrm{diag}(\mathbf{p}(\tau), \cdots, \mathbf{p}(\tau)),\ n_{\mathsf{x}} \text{ blocks};\ \mathbf{a} = \begin{bmatrix}\mathbf{c}_1^{\mathsf{T}} & \cdots & \mathbf{c}_{n_{\mathsf{x}}}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}},
\end{aligned} \tag{3.28}$$

in which, n_{x} represents the number of state variables, $\mathrm{diag}(\phi)$, n_{x} blocks is a block diagonal matrix where the block ϕ is repeated n_{x} times. $\mathbf{a}$ represents the matrix of constants $\mathbf{C}$ from expression (3.18), but the constants are positioned in vector form: the first entries of $\mathbf{a}$, 0 up to n_{p} are the constants of x_1, then from $n_{\mathsf{p}} + 1$ up to $2n_{\mathsf{p}} + 1$ are constants of x_2 and so on.

3.2.3 Nonlinear System Simulation with DG-FEM

In nonlinear systems, the simulation method is bound to be iterative. Expressions (3.29) and (3.30) show the equations to solve (for **a**) for forward and backward simulation.

$$\Big(\mathbf{P}(1)+\mathbf{D}\Big)\mathbf{a} = -\frac{\Delta t^e}{2}\mathbf{I}_f + \mathbf{BC}(-1)\hat{\mathbf{x}}(t_f^{e-1}) \tag{3.29}$$

$$\Big(\mathbf{P}(-1)-\mathbf{D}\Big)\mathbf{a} = \frac{\Delta t^e}{2}\mathbf{I}_f + \mathbf{BC}(1)\hat{\mathbf{x}}(t_0^{e+1}) \tag{3.30}$$

Using the definitions in (3.28) and (3.31).

$$\mathbf{I}_f = \int_{-1}^{1} \mathbf{p} f_i \,\mathrm{d}\tau \;;\; 1 \leq i \leq n_x. \tag{3.31}$$

The method chosen to solve these nonlinear vector expressions is the fixed point method. This method suggests that for iteration k the next guess for solution is calculated using $\mathbf{a}^{k+1} = \mathbf{g}(\mathbf{a}^k)$ Furthermore, with this method is possible to assure convergence for Δt^e low enough and a close enough initial guess.

Convergence of nonlinear simulation (fixed point method)

The fixed point theorem states that, for a function $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ if:

$$\begin{aligned} &\text{a) } \mathbf{g}(D) : D \to D, \text{ solution} \in D \subset \mathbb{R}^n, \\ &\text{b) } \mathbf{g} \text{ is continuously differentiable in } D, \\ &\text{c) } \left|\frac{\mathrm{d}\mathbf{g}_i}{\mathrm{d}x_j}\right| < 1, \end{aligned} \tag{3.32}$$

then there is a unique solution for $\mathbf{x} = \mathbf{g}(\mathbf{x})$ in D and the fixed point method will converge towards the solution [24].

For this purpose, **g** is defined as:

$$\mathbf{g} = \Big(\mathbf{P}(1)+\mathbf{D}\Big)^{-1}\Big(-\frac{\Delta t^e}{2}\mathbf{I}_f + \mathbf{BC}(-1)\hat{\mathbf{x}}(t_f^{e-1})\Big). \tag{3.33}$$

Condition a) may be ”removed” if we assure that $\mathbf{x}^0$ (initial guess of **x**) is close enough to the solution. Expression (3.33) demonstrates that $\hat{\mathbf{x}}(t_f^{e-1})$ is a suitable guess to start the iterations, for a low Δt^e. This is done by removing the integral term $\mathbf{I}_f$, which will make $\tilde{\mathbf{x}}^e = \hat{\mathbf{x}}(t_f^{e-1})$.

It is assumed that **f** represents real dynamics, so b) is also assured.

Finally c) will hold true for low enough Δt^e.

$$\lim_{\Delta t^e \to 0}\left|\frac{\mathrm{d}\mathbf{g}}{\mathrm{d}a_i}\right| = \lim_{\Delta t^e \to 0}\left|\Big(-\mathbf{P}(1)+\mathbf{M}\Big)^{-1}\Big(-\frac{\Delta t^e}{2}\frac{\mathrm{d}\mathbf{I}_f}{\mathrm{d}\mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a_i}\Big)\right| = 0 < 1. \tag{3.34}$$

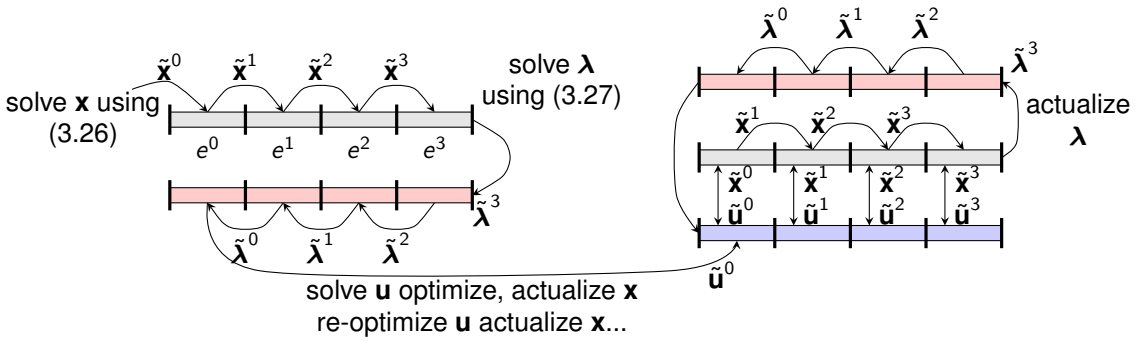


Figure 3.2: Algorithm outline.

3.3 Solving PMP Problems with DG-FEM

As seen above PMP problems require the solution of two differential equations (3.14)-(3.12), one for the state variables $\mathbf{x}$ and one for the co-state variables $\boldsymbol{\lambda}$. Unfortunately these cannot be solved together, as $\boldsymbol{\lambda}$ requires the knowledge of the control and state variables variables ($\mathbf{u}$ and $\mathbf{x}$) at the final time T, since its boundary condition is positioned there (at $t = T$) for specified initial states and fixed time problems.

In (3.14), $\mathbf{f}$ may be a nonlinear expression, leading to the iterations defined in the previous section 3.2.3. Looking at expressions (3.12) and (3.6), it is possible to infer that $\boldsymbol{\lambda}$ is defined by a linear, time-dependent system (3.35), which is easier to solve using the expressions defined for DG-FEM linear backwards simulation, present in expression (3.27).

$$\dot{\boldsymbol{\lambda}} = \left(\boldsymbol{\mathcal{J}}_{\mathbf{f}}(\mathbf{x})\right)^{\mathsf{T}} \boldsymbol{\lambda} + \nabla_{\mathbf{x}} J, \tag{3.35}$$

in which $\boldsymbol{\mathcal{J}}_{\mathbf{f}}(\mathbf{x})$ stands for the Jacobian of $\mathbf{f}$ about $\mathbf{x}$, i.e. $\frac{\mathrm{d}f_i}{\mathrm{d}x_j}$.

Regarding equation (3.13), it represents the maximization of the Hamiltonian $\mathcal{H}$ about the control variables $\mathbf{u}$, which can be solved with an optimization algorithm. In this work, the optimization algorithm used was the conjugate gradient (of *python*'s toolbox *scipy* ("cg")) for continuous control problems, but any other would do (attending to the characteristics of the problem). For *on-off* control problems, an exhaustive search is valid because all that is required is a solution using one of 2 values.

3.4 Solving PMP Problems with DG-FEM and Shooting method

In the previous section 3.3 is discussed a method to solve the PMP problem, using the DG-FEM method, by going forwards (to calculate $\hat{\mathbf{x}}$ and $\hat{\mathbf{u}}$) and backwards (to calculate $\hat{\boldsymbol{\lambda}}$) because the boundary conditions for $\boldsymbol{\lambda}$ are positioned in the final time.

In this section is discussed an alternative method, where everything is calculated forward, making the problem, possibly, faster to solve and with a better convergence: the shooting method [23].

The shooting method is mainly used to solve differential equations with boundaries in an initial time and at a final time. A typical problem of application is described in equation (3.36), where, to a N-th order

differential equation (described as a system of first-order differential equations), there are n_1 boundary conditions placed at $t = 0$ and $n_2 = N - n_1$ boundary conditions placed at $t = T$.

$$\begin{aligned} &\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{x}), \\ &x_i(t = 0) - x_i^{\mathrm{BC}(t=0)} = 0 \qquad\qquad i = 1, \cdots, n_1, \\ &x_j(t = T) - x_j^{\mathrm{BC}(t=T)} = 0 \qquad\qquad j = n_1 + 1, \cdots, N. \end{aligned} \tag{3.36}$$

This method consists on guessing the value of the variables with boundary conditions at $t = T$ in $t = 0$ (aiming), which will be imposed as a condition at $t = 0$. Then the problem can be solved with forward integration to reach T (shooting). If the initial guess for $x_j^{\mathrm{BC}(t=0)}$ is not the exact solution at $t = 0$, then there will be a discrepancy between $x_j(T)$ and the original boundary condition $x_j^{\mathrm{BC}(t=T)}$. For the purpose of the formulation let $\mathbf{V}$ be the guess of the imposed value of x_j in $t = 0$ and $\mathbf{F}$ the discrepancies, computed as $F_j = x_j - x_j^{\mathrm{BC}(t=T)}$.

3.4.1 Aiming

While the first value of $\mathbf{V}$ may be arbitrary, it is required a way to renew the imposed initial value to restart the cycle. The used method for this work is the Newton method as suggested by Press et al. [23]. Expression (3.37) demonstrates how a new step is calculated.

$$\mathbf{V}^{\mathrm{new}} = \mathbf{V}^{\mathrm{old}} - \left(\mathcal{J}_{\mathbf{F}}\mathbf{V}\right)^{-1}\mathbf{F}. \tag{3.37}$$

Unfortunately, the Jacobian $\mathcal{J}_{\mathbf{F}}\mathbf{V}$ required for the calculations is usually not available. So, these should be calculated as numerical Jacobians (3.38). Keep in mind the calculation of the Jacobian requires the integration of the problem n_2 extra times (one time for each δV_j). Then, each step for the shooting iterations, requires $n_2 + 1$ solutions for the problem accounting for the original calculation of $\mathbf{F}$.

$$(\mathcal{J}_{\mathbf{F}}\mathbf{V})_{ij} = \frac{\Delta F_i}{\Delta V_j} = \frac{F_i(\mathbf{V} + \delta V_j) - F_i(\mathbf{V})}{\delta V_j}. \tag{3.38}$$

Chapter 4

Implemented Algorithms

On this chapter is shown, with more detail, how the methods discussed in the previous chapter 3 were combined and implemented. This will be done with flowcharts to represent how the numerical environment works. In fact, every diagram shown in this chapter corresponds to a function of the implementation.

The control methods will also be described here for three types of problems: continuous control, on-off control and bang-bang control. The main algorithms outline is kept independent from the designed control methods, to allow the application of any other problem.

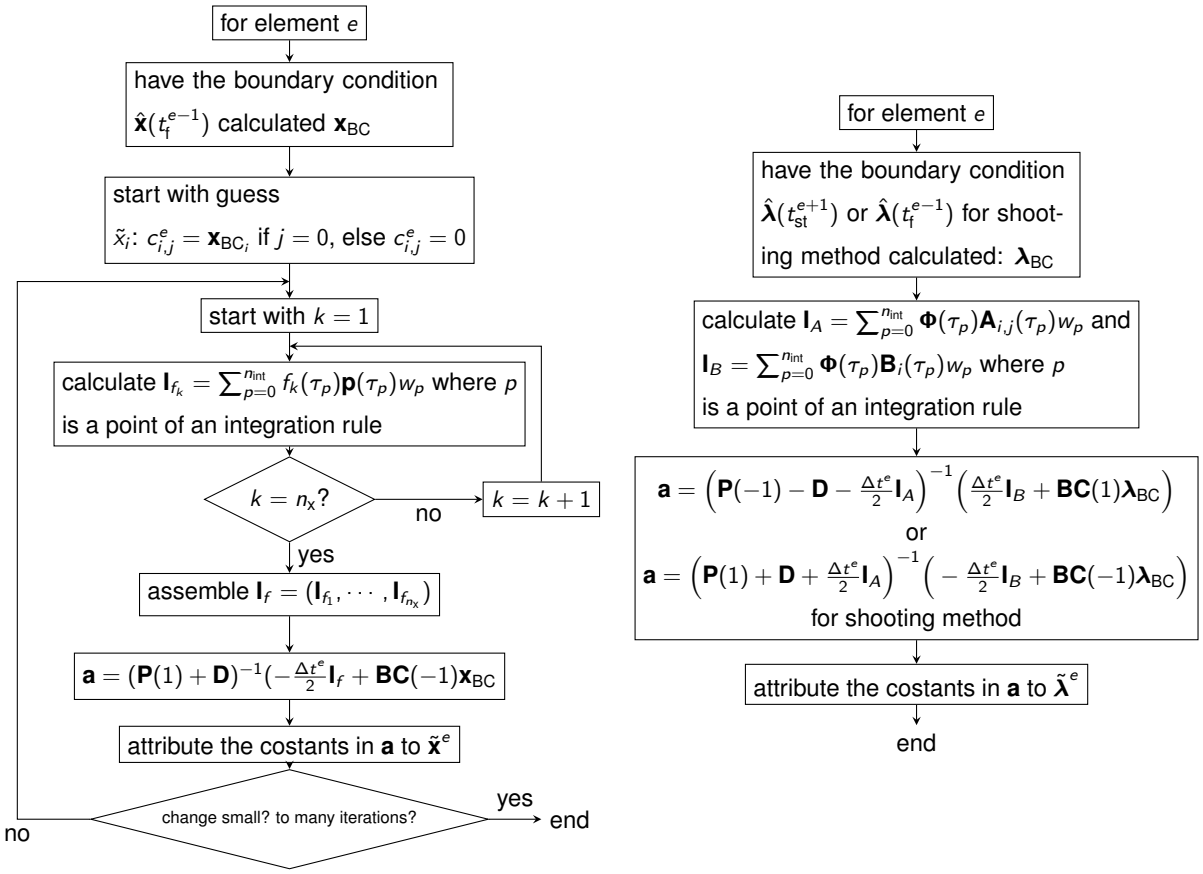
4.1 System Simulation

Diagrams 4.1(a) and (b) specify how the implementation of equations (3.29) for $\mathbf{x}$ and (3.27-3.26) for $\boldsymbol{\lambda}$ are calculated. As the diagram 4.1(b) suggests, it can be used to calculate $\tilde{\boldsymbol{\lambda}}$ backwards and forward (shooting method), as it is only a matter of changing the boundary conditions and some signs. Remember these expressions are used since $\mathbf{x}$ is assumed to be nonlinear and $\boldsymbol{\lambda}$ is linear. Recall, also, that for nonlinear systems, the fixed point method is being used, whose convergence is assured for Δt^e small enough and a close enough initial guess, which for small Δt^e may be $\hat{\mathbf{x}}(t_{\mathrm{f}}^{e-1})$. The fixed point method is present in figure 4.1(a) and it corresponds to everything inside the big cycle. Being the calculation of $\mathbf{I}_f$ done with the previous value of $\tilde{\mathbf{x}}$. The small cycle (incrementing k) is only done, to avoid large calculation times.

Note that, due to the iterative nature of the calculations of $\mathbf{x}$, a stopping criterion is needed. The criteria used here was a maximum number of iterations and a comparison between two different iterations (to check if the change was significant or not). These parameters are arbitrary and should be changed according to the desired precision. In these there is a reference to an integration rule; the one used for this work was Gauss-Lobatto rule because it uses the extremes of the element (discontinuity points). The number of integration points is arbitrary, but remembering that the Gauss-Lobato method integrates exactly polynomials of degree up to $2n_{\mathrm{int}} - 3$, a possible rule of thumb for the choice of n_{int} (number of

integration points) is:

$$2n_{\text{int}} - 3 > degree(\boldsymbol{\lambda}^T \mathbf{f}) \approx 2n_{\text{p}},$$
$$n_{\text{int}} > n_{\text{p}} + \frac{3}{2}. \tag{4.1}$$



(a) $\tilde{\mathbf{x}}$ forward

(b) $\tilde{\boldsymbol{\lambda}}$ backward or forward for shooting method

Figure 4.1: State variables simulation.

4.2 Control Calculations

After the solutions of $\mathbf{x}$ and $\boldsymbol{\lambda}$, the control is left to be defined. Here, the calculations may differ on the type of control, raising the need for different calculations. Remember that for any of these, the goal is to obtain the maximum $\int_0^T \mathcal{H}\,\mathrm{d}t$.

4.2.1 Continuous Control

As referenced in section 3.3, the continuous control problems are being calculated recurring to the conjugate gradient method, but any other would suffice. Figure 4.2 illustrates the algorithm used to solve the continuous control. It includes the simulation of the state variables as these will change the Hamiltonian, and thus, the control.

Note, again, that the algorithm in 4.2 is iterative and, therefore, it requires criteria to stop. The criteria used here was the number of iterations and the change, this time in $\mathcal{H}$. The thresholds are, once again, arbitrary. Each iteration starts by building a function $I_{\mathcal{H}}$ only dependent on $\tilde{\mathbf{u}}^e$ constants and, if required by the optimization, the construction of the gradient of $\int_{-1}^{1} \mathcal{H}\,\mathrm{d}\tau$ about the $\tilde{\mathbf{u}}^e$ constants: $\nabla_{\mathbf{C}_u^e} I_{\mathcal{H}}$. After the control is changed, the remaining variables should change to. With this, $\mathcal{H}$ will change, so the optimal control will need to be recalculated (restarting the iteration).

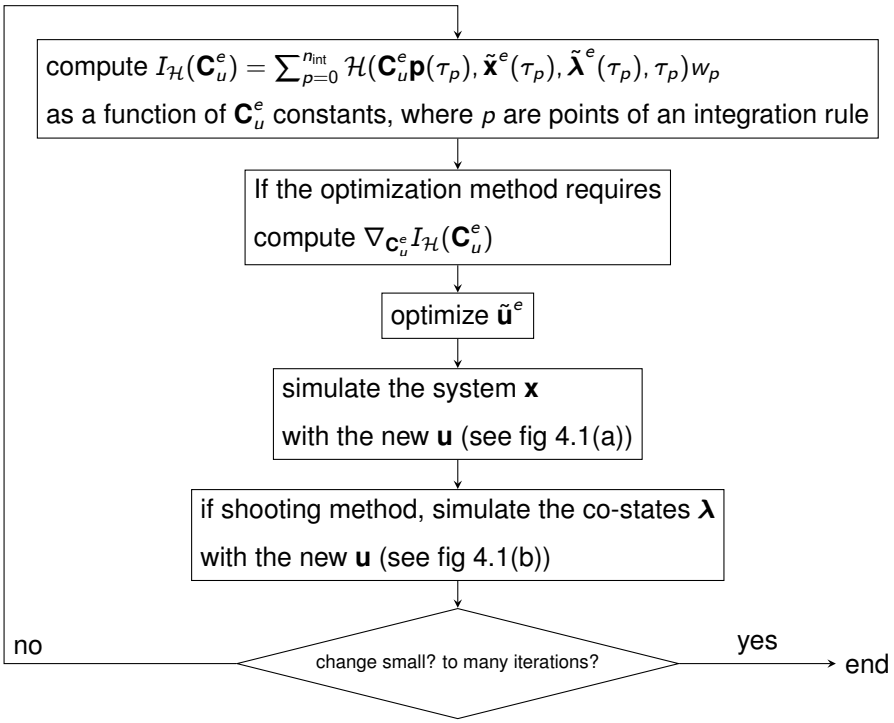


Figure 4.2: Continuous control calculations.

4.2.2 On-Off Control

This type of control has the particularity of offering just two options for u (this time, scalar) $u \in \{0, 1\}$. With this control, the maximization of the Hamiltonian is assured by one of two choices, making an exhaustive search viable. Figure 4.3 shows how it may be done. With this control the expression (3.13) $\left(\frac{\partial \mathcal{H}}{\partial u} = 0\right)$ may not be assured (in fact, most of the times it is not), but there is still and admissible maximum.

To calculate the on-off control, the approximations $\tilde{u}^e$ will be kept constant inside the element, meaning that one element only presents the value of $u = 1$ or $u = 0$. This may be undesirable for meshes with larger Δt^e, for which, figure 4.3 also proposes a refinement method, integrated inline.

This optimization algorithm starts with the calculation of $\mathcal{H}$ for $u = 1$ and for $u = 0$, performing the state variables and co state variables changes with each u tested. Instead of directly calculating the integral value of each $\mathcal{H}$ ($u = 0$ and $u = 1$), each point in the integral quadrature is compared, if at all points the $\mathcal{H}$ calculated with $u = 1$ is greater than $\mathcal{H}$ calculated with $u = 0$ then, the control should be $u = 1$. On the other hand, if $\mathcal{H}$ calculated with $u = 1$ is lesser than $\mathcal{H}$ calculated with $u = 0$, then the

control should be $u = 0$. If there is a change between points, then this is a element with a shifting time, and it should be refined.

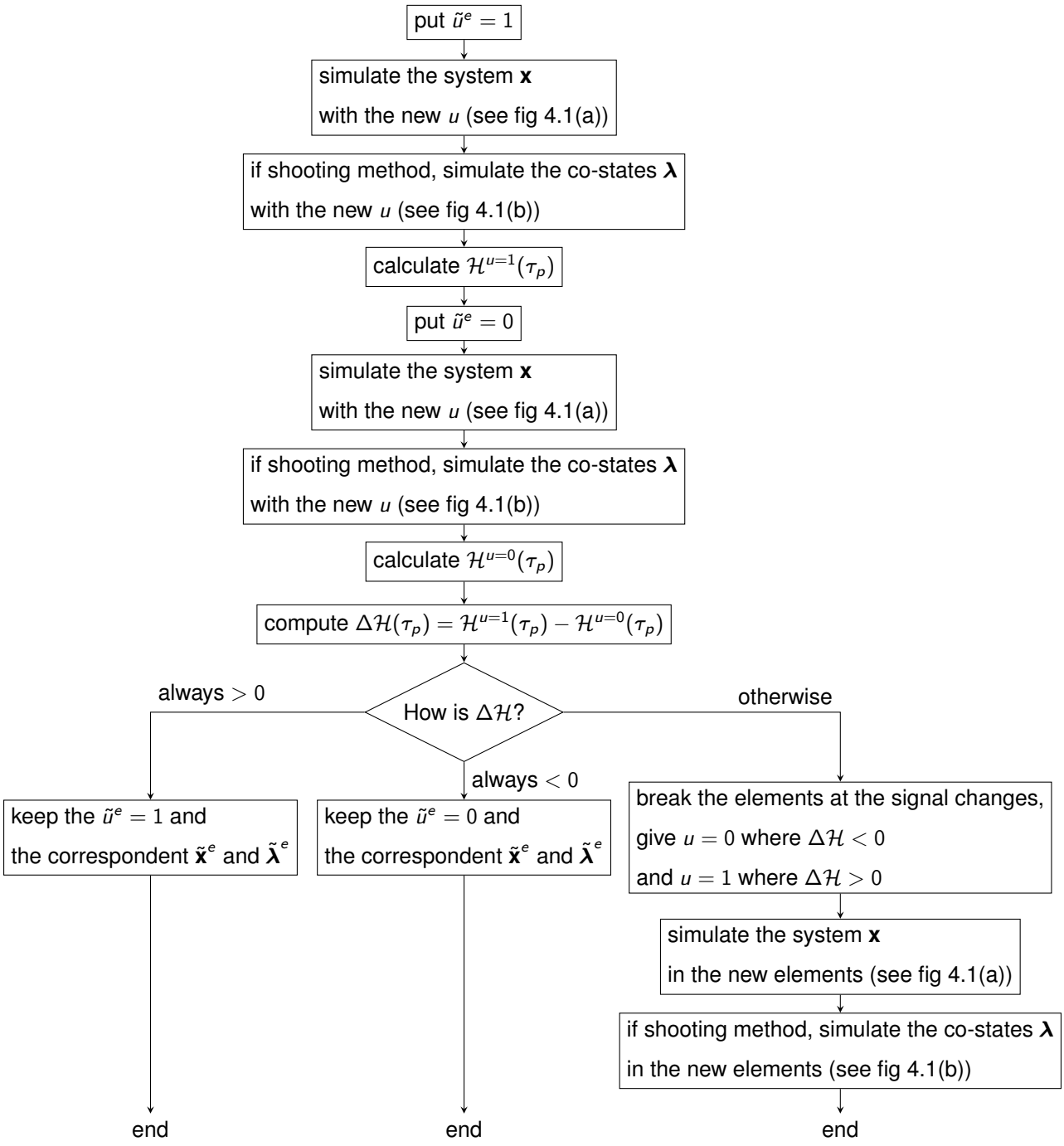


Figure 4.3: On-Off control calculations.

4.2.3 Bang-Bang Control Problems

This type of problems are a variation from the on-off control problem, described above. They only allow two possible values for $u - u \in \{0, 1\}$. With the particularity of having a functional J linear with u and a **f** function as well. With this specification, $\frac{\partial \mathcal{H}}{\partial u}$ is independent from u, or, at least, explicitly, because

x and $\boldsymbol{\lambda}$ are still dependent and they change $\frac{\partial \mathcal{H}}{\partial u}$. From which follows that if $\frac{\partial \mathcal{H}}{\partial u} > 0$ then u should be 1, if $\frac{\partial \mathcal{H}}{\partial u} < 0$ then u should be 0.

Again, the approximation with u will be constant within each element, presenting the values 1 or 0. But if it is assumed that $\frac{\partial \mathcal{H}}{\partial u}$ is independent from u, then the refinement of an element is trivial: find the zero of $\frac{\partial \mathcal{H}}{\partial u}$ at the element (assuming **x** and $\boldsymbol{\lambda}$ don't change). The following diagram 4.4, presents the control solution implemented for this type of problems.

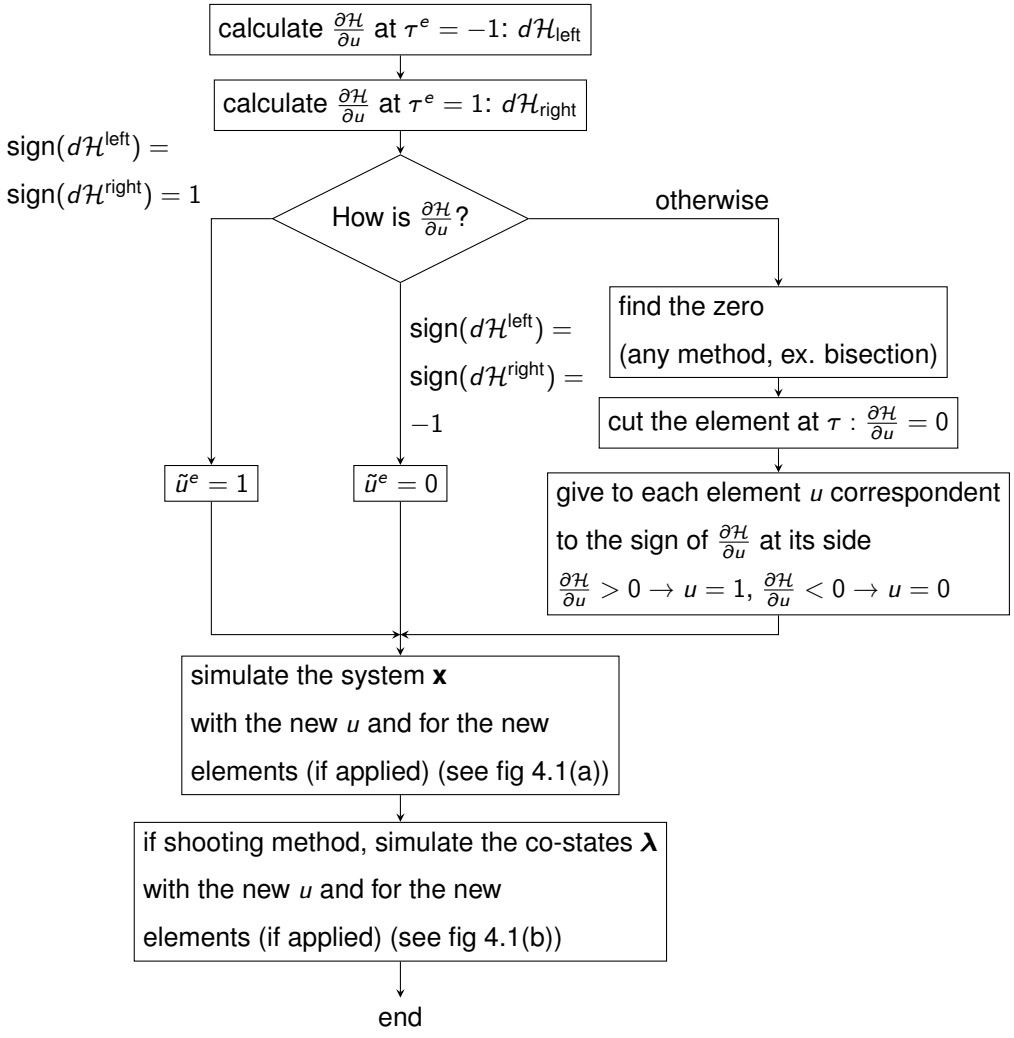


Figure 4.4: Bang-bang control calculations.

4.3 PMP Solutions

Now that the control and the simulation of state and co-state variables are defined (for one element), what is left to specify is the progression of the solutions proposed. Figure 4.5 shows how to solve a PMP problem with forward integration of the state variables and backwards for the co-state variables, while figure 4.6 illustrates the shooting method solution. These are independent for any of the control methods discussed in the previous diagrams 4.2, 4.3 and 4.4 or any other different method for a different control problem.

Figure 4.5 starts with the guess of an initial $\hat{\mathbf{u}}$, which will be used to calculate the state variables $\tilde{\mathbf{x}}^e$ from the initial element to the last. With these calculated, the main cycle can begin: cycling by the calculation of the co-state variables $\tilde{\boldsymbol{\lambda}}^e$ from the last element to the first, then proceeding to the recalculation of control state variables as was explained in the previous section 4.2, from the first to the last element.

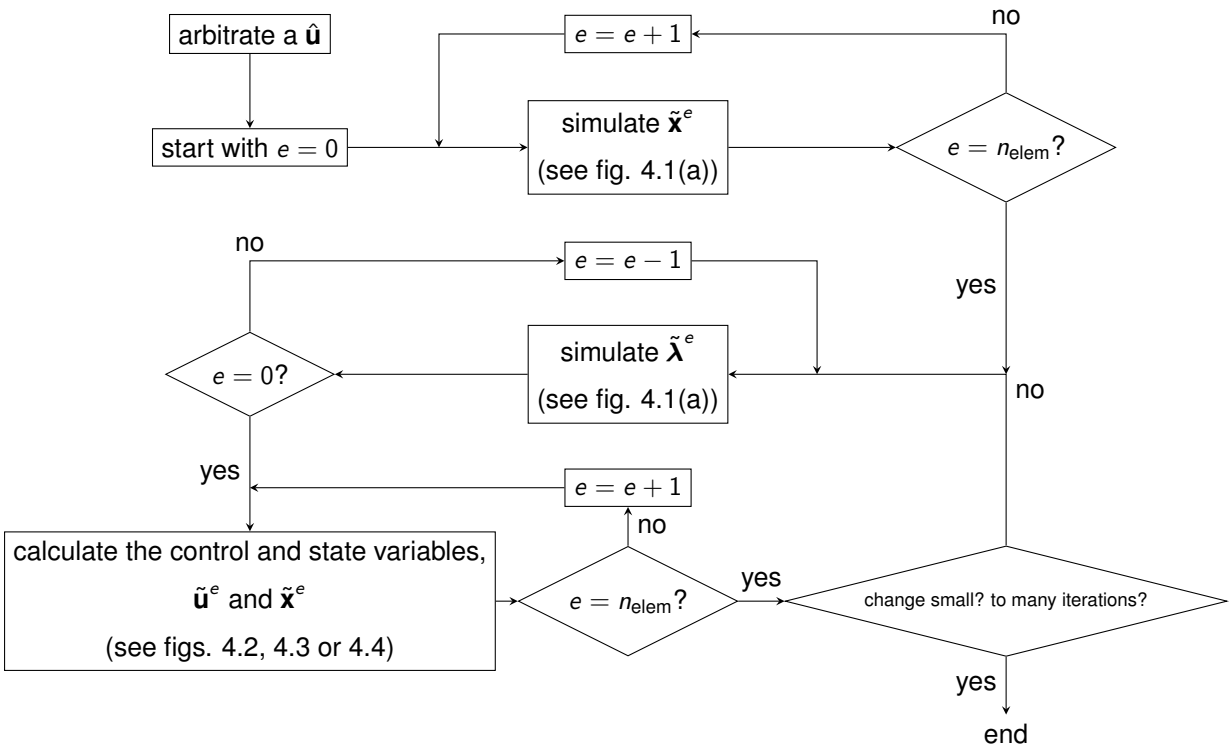
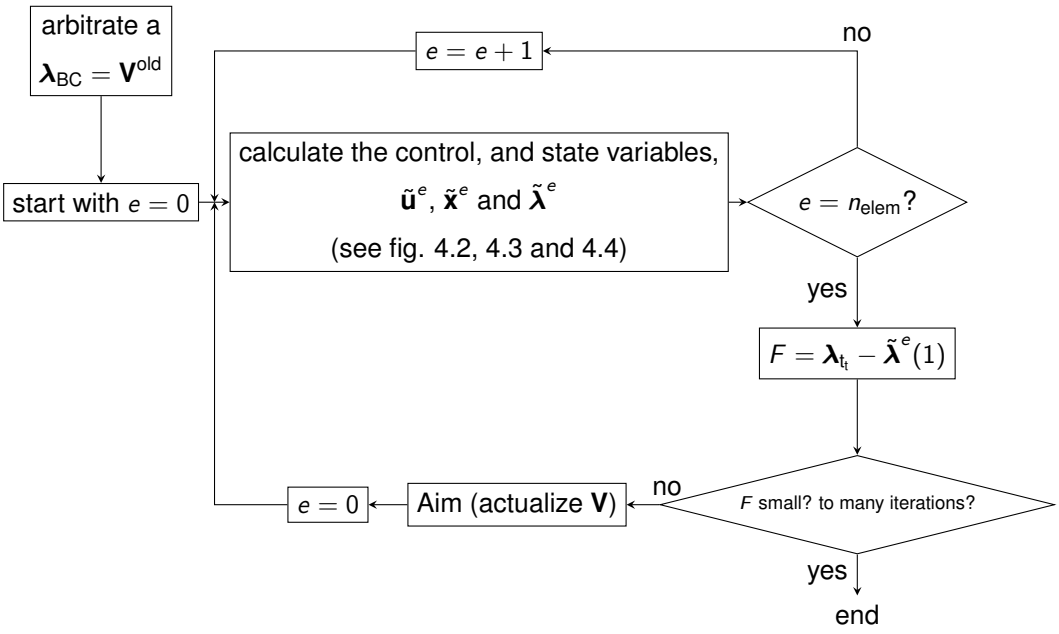


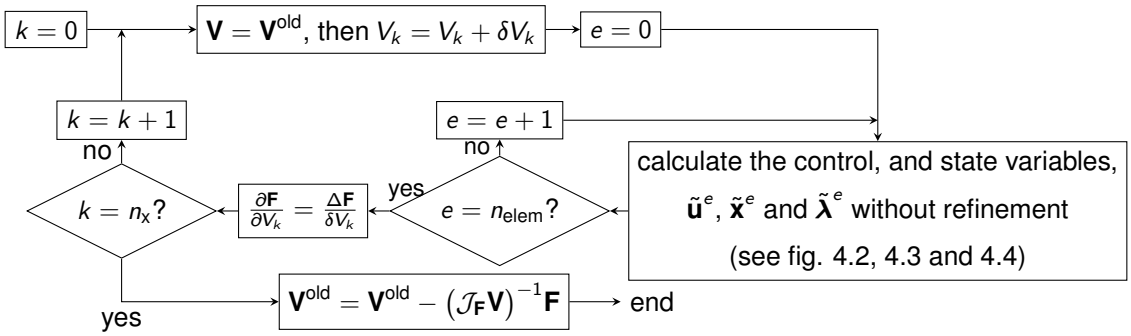
Figure 4.5: Forward and backward PMP solution.

Figure 4.6 introduces the shooting method. After choosing a plausible $\boldsymbol{\lambda}_{\text{BC}}$ the cycle begins with the calculation of the control, state and co-state variables as explained in section 4.2. These algorithms should now be adapted to include the forward calculation of $\boldsymbol{\lambda}$. The algorithms for the simulation (fig. 4.1(b)) and control (figs. 4.2, 4.3 and 4.4) already have this option described. After the simulations and control for each element, starting in the first, the discrepancy $\mathbf{F}$ will be calculated and $\boldsymbol{\lambda}_{\text{BC}}$ needs to be actualized for the cycle to re-start.

The actualization of $\boldsymbol{\lambda}_{\text{BC}}$ is described in figure 4.6(b), where, for each entry in $\boldsymbol{\lambda}_{\text{BC}}$ is inserted a displacement δV; with that displacement, re-simulate and optimize the system (starting, again, in the first element). Calculate the discrepancy with the displacement and return to the original system. Once all $\boldsymbol{\lambda}_{\text{BC}}$ has been displaced (one at a time), invert the Jacobean and calculate the new $\boldsymbol{\lambda}_{\text{BC}}$.



(a) Shooting method algorithm



(b) aiming function (newton method)

Figure 4.6: Shooting method.

Chapter 5

Validation

The validation of the developed formulation is now done by solving the problem proposed by Gong et al. [25] (simple continuous PMP problem), which has an analytical solution and tests the ability to solve continuous control problems. This is done in sub-section 5.1. The problem proposed by Luenberger, D. [26] (simple bang-bang problem) is then solved in sub-section 5.1, which tests the ability to solve bang-bang control problems, in this case, with one switching time instant. Different solutions for same problems were proposed by Henriques et al. [12] who implemented the discontinuous Galerkin method integrating expression (3.23) by parts twice, resulting in a different configuration of equations (3.24) and (3.25).

The following subsections show several error plots. The calculation of the error in these plots corresponds to the L^2 norm, as defined by [14]:

$$||\text{error}(\phi)||_2 = \sqrt{\frac{1}{T}\int_0^T (\phi(t) - \hat{\phi}(t))^2 \, \mathrm{d}t} \tag{5.1}$$

Remember that $\hat{\phi}$ is the approximation of ϕ by (3.17), on the other hand ϕ is the exact solution of the problem.

5.1 Validation: Simple Continuous PMP

This problem is a test to the algorithms in chapter 4, namely the continuous control algorithm with the conjugate gradient from *scypy* toolbox, in fact, the *scypy* toolbox had to be modified to allow arbitrary precision calculations. This was achieved by copying the existing module and changing the relevant functions to work with arbitrary precision. As referred before, Henriques et al. [12] already solved this problem numerically with a slightly different formulation. They achieved an order of convergence of about $n_\text{p} + 0.5$, even though, the optimal theoretical convergence should be of the order of $n_\text{p} + 1$ as proved by Baccouch [14].

This problem required a relaxation factor in order to converge with the forward ad backward integra-

tion.

$$\tilde{\mathbf{x}}^e = \omega \tilde{\mathbf{x}}^{e^{\text{new}}} + (1-\omega)\tilde{\mathbf{x}}^{e^{\text{old}}} \tag{5.2}$$

where, after each calculation of the state variables with the fixed point method, the state variables before the fixed point $\tilde{\mathbf{x}}^{e^{\text{old}}}$ will be added to the state variables calculated $\tilde{\mathbf{x}}^{e^{\text{new}}}$, through a relaxation factor $0 < \omega \leq 1$. In [24] the relaxation is used, as a form to speed the convergence of the fixed point method. This relaxation was not required when using the shooting method.

5.1.1 Problem Statement

maximize:

$$J(u) = \int_0^{\mathrm{T}} -u^2 dt - 4x_1(T) - x_2(T) \tag{5.3}$$

subject to:

$$\begin{cases} \dot{\mathbf{x}} = \begin{pmatrix} x_2^3 & , & u \end{pmatrix} \\ \mathbf{x}(0) = \begin{pmatrix} 0 & , & 1 \end{pmatrix} \\ T = 2 \end{cases} \tag{5.4}$$

The analytical solution is, as can be easily verified,

$$\mathbf{x} = \begin{bmatrix} \frac{2}{5} - \frac{64}{5(t+2)^5} \\ \frac{4}{(t+2)^2} \end{bmatrix} \tag{5.5}$$

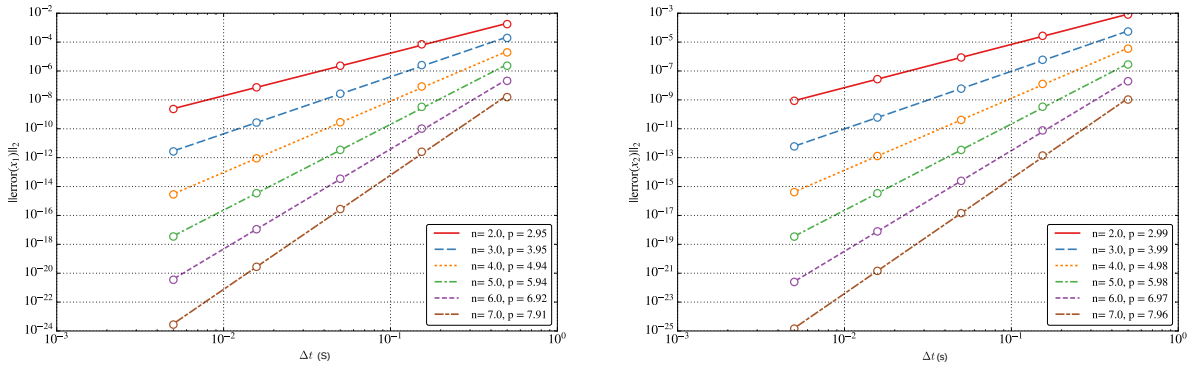
$$\boldsymbol{\lambda} = \begin{bmatrix} -4 \\ -\frac{64}{(t+2)^3} \end{bmatrix} \tag{5.6}$$

$$u = -\frac{8}{(t+2)^3} \tag{5.7}$$

5.1.2 Results: Numerical Errors and Error Trend-lines

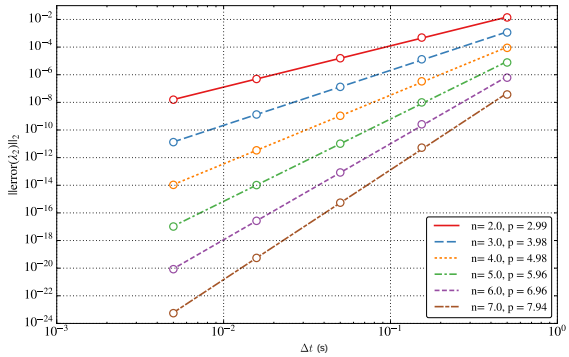
In this problem it was still possible to implement the integrals as symbolic; consequently, the numerical deviations from the analytical solution may only be caused by the DG-FEM method or the maximum precision of the operations. Figure 5.1 shows the variation of the numerical absolute error (square norm) with polynomial degree n and elements time interval Δt^e, with arbitrary precision (number of decimal places (dps) $= 100$). These figures were obtained using the forward and backward integration method described in section 4.3. Markers are the points calculated, lines are the trend-lines of the absolute error (with slope p). Note the log-log scale. Figure 5.4 plots the same quantities, but was generated with the shooting method.

Figures 5.2 and 5.3 plot the same as 5.1, but with limited precision, 8 decimal places for figure 5.2 and 16 for figure 5.3. In these figures the thinner continuous lines are the theoretical trend of the error $C\Delta t^{n_p+1}$[14] and the dashed thick lines are just a visual feature to follow the plotted points.

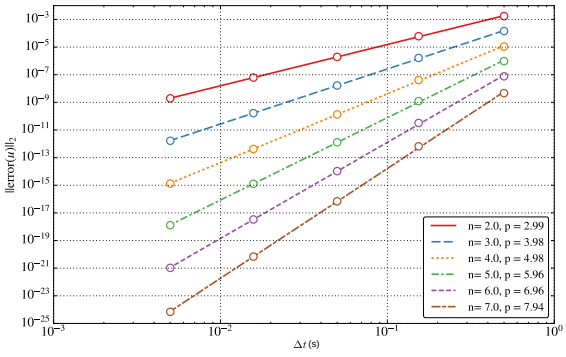


(a) x_1

(b) x_2



(c) λ_2



(d) u

Figure 5.1: Root mean square error and trend as function of Δt and the polynomial degree n (where p is the slope of the trend line).

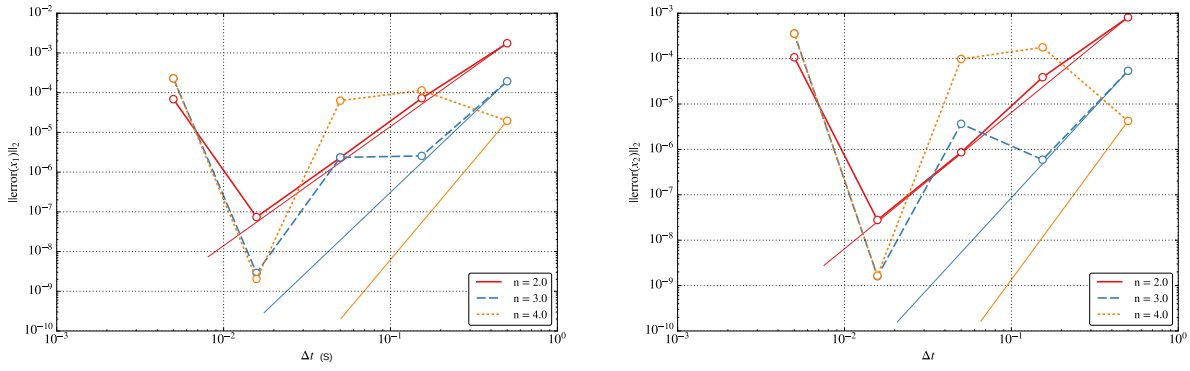
With these figures we will be able to access the impact of the machine precision in the DG method. The choice of $dps = 8$ and $dps = 16$ simulates approximately the regular single precision and double precision floats respectively.

5.1.3 Discussion

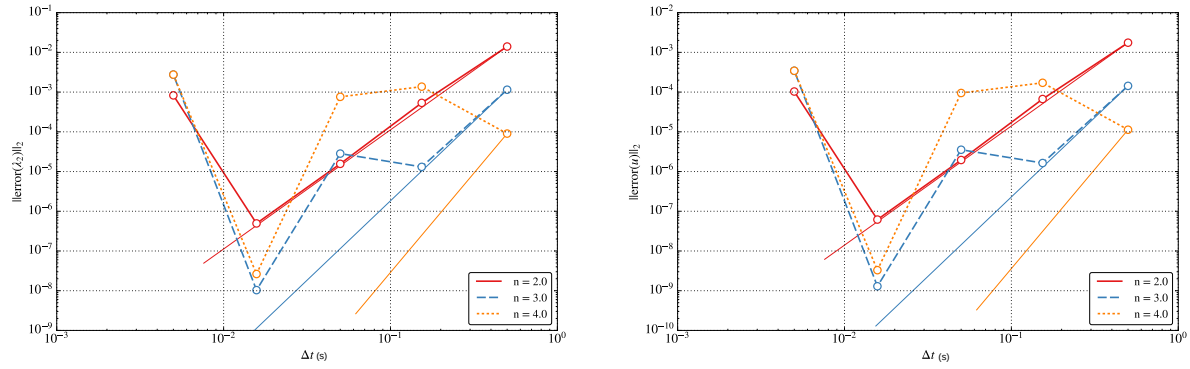
In figure 5.1 we can conclude the DG-FEM is working according to what is expected, following an error convergence of the order of $n_p + 1$ (for the "L^2-norm" of the error) [14]. Furthermore, the use of *mpmath* module with $dps = 100$ eliminated the precision portion of the error. On the other hand, in figures 5.2 and 5.3, the portion of the error due to precision is very visible, creating a threshold for the error. Once this threshold is surpassed, the error seems to develop a random behavior. These results also appear as an improvement to Henriques et al. [12].

These plots stress the relevance of high precision, as the solution calculated with almost single precision (fig. 5.2) is barely acceptable for most of the tested cases. This shows that increasing the order of the polynomial approximation n_p or decreasing the time interval Δt^e may be irrelevant or even harmful to the quality of the approximation if the precision is left unattended. If this test was made using only $dps = 8$, we would be led to believe the solution was badly implemented.

Similarly to figure 5.1, figure 5.4 also shows orders of convergence near the theoretical order $n_p + 1$.



(a) x_1 (b) x_2



(c) λ_2 (d) u

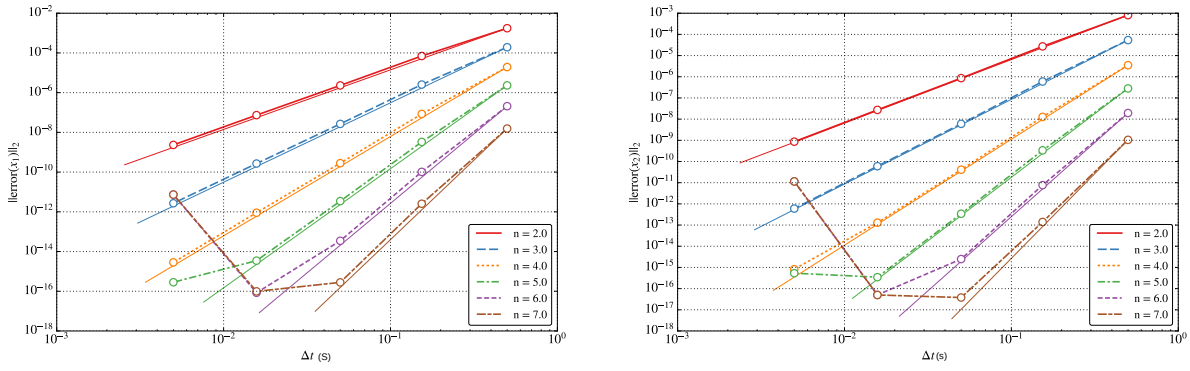
Figure 5.2: Root mean square error of as function of Δt and the polynomial degree n calculated with $dps = 8$

This implies that the shooting method is well implemented. This figure has fewer points because for higher degrees of precision, the acceptable discrepancy will also need to be lower, for it affects the errors as well, and for a lower discrepancy, more cycles need to run. This figure presents only x_1 and λ_2 because $u = \lambda_2/8$ and x_1 is derived from x_2.

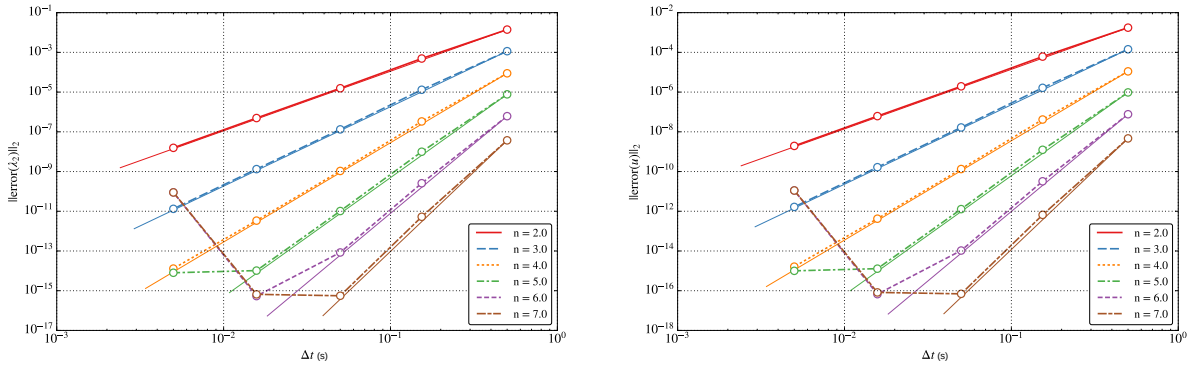
5.2 Validation: Simple Bang-bang Problem

This problem tests the algorithm of section 4.2, described by figure 4.4. As referenced before [21], bang bang problems are on-off problems with the particularity of having a functional and a state space equation linear with u. Looking at (5.8) and (5.9) it is evident that this problem is a bang bang control problem.

Henriques et al. [12] also solved this problem with the slightly different formulation and obtained a convergence of the error of the order of about $n_p + 0.5$ again, while the theoretical optimal convergence rate is of the order of $n_p + 1$ [14]. Unlike the continuous problem, this did not require the relaxation factor for the forward and backward integration algorithm, nor the shooting method.



(a) x_1 (b) x_2



(c) λ_2 (d) u

Figure 5.3: root mean square error of as function of Δt and the polynomial degree n calculated with $dps = 16$

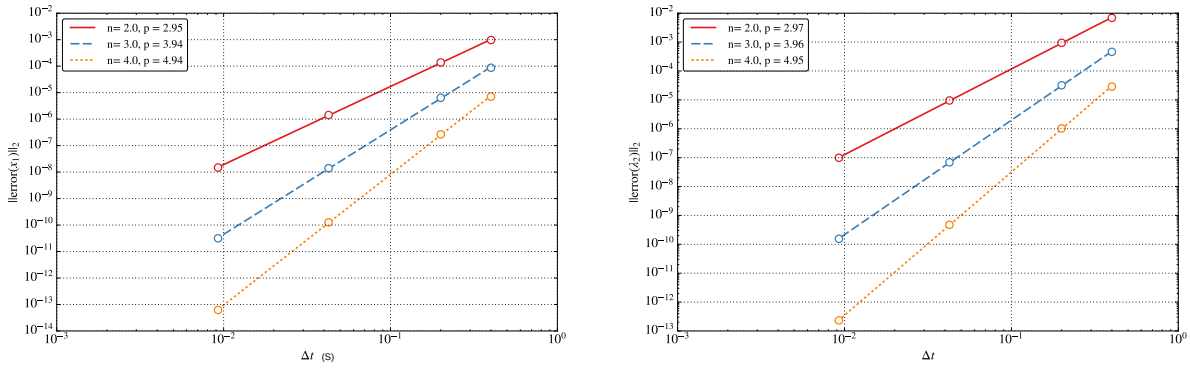
5.2.1 Problem Statement

Maximize:

$$J(u) = \int_0^T (1-u)x\,dt \tag{5.8}$$

subject to:

$$\begin{cases} \dot{x} = (u - 0.5)x \\ x(0) = 1 \\ T = 5 \end{cases} \tag{5.9}$$



(a) x_1 (b) λ_2

Figure 5.4: Root mean square error and trend as function of Δt and the polynomial degree n (where p is the solpe of the trend line). Calculated with shooting method.

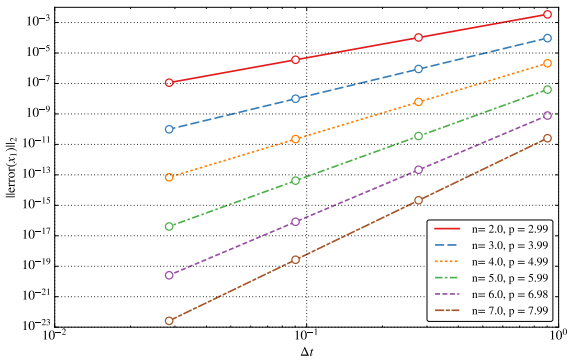


Figure 5.5: Error of simulation with t_s in element boundary

The analytical solution is, as can be easily verified,

$$t_s = T + 2\ln(0.5) \tag{5.10}$$

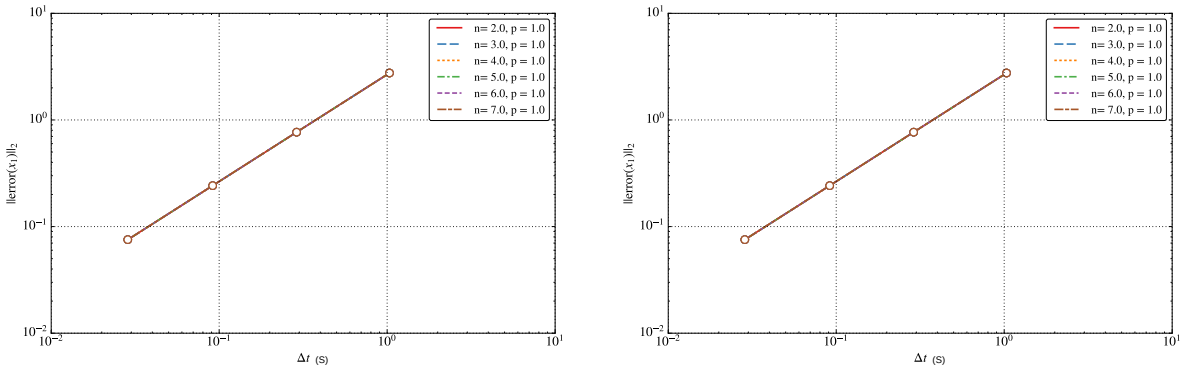
$$x(t) = \begin{cases} e^{0.5t}, 0 \leq t \leq t_s \\ e^{t_s - 0.5t}, t_s \leq t \leq T \end{cases} \tag{5.11}$$

$$\lambda(t) = \begin{cases} 2(1 - e^{0.5(t_s - T)})e^{-0.5(t - t_s)}, 0 \leq t \leq t_s \\ 2(1 - e^{0.5(t-T)}), t_s \leq t \leq T \end{cases} \tag{5.12}$$

$$u(t) = \begin{cases} 1, \ 0 \leq t \leq t_s \\ 0, \ t_s \leq t \leq T \end{cases} \tag{5.13}$$

5.2.2 Results: Numerical Errors and ”Switching time” Error

The errors calculated with this example represent mostly the error caused by the sifting time, t_s, estimation. To avoid the complications of the continuous problem in the previous section 5.1 These calculations were made with $dps = 100$. Figure 5.5 shows the error considering t_s to be a part of the boundary of two elements.



(a) (b)

Figure 5.6: Error of simulation with t_s: (a) at 50% of element length, (b) at 5% of element length from the left boundary.

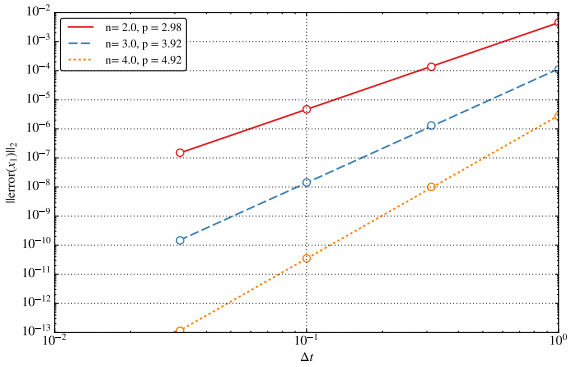


Figure 5.7: Mean square error of x with bang bang refinement and shooting method.

On the other hand, figures 5.6a and 5.6b represent the error considering t_s to be inside of one element. In figure 5.6a t_s is in the middle of an element. In figure 5.6b t_s is located at a distance of 5% of element length from the element right boundary.

Finally, figure 5.7 represents the error as a function of the initial element size and the polynomial degree of the solution found using the shooting method with refinement.

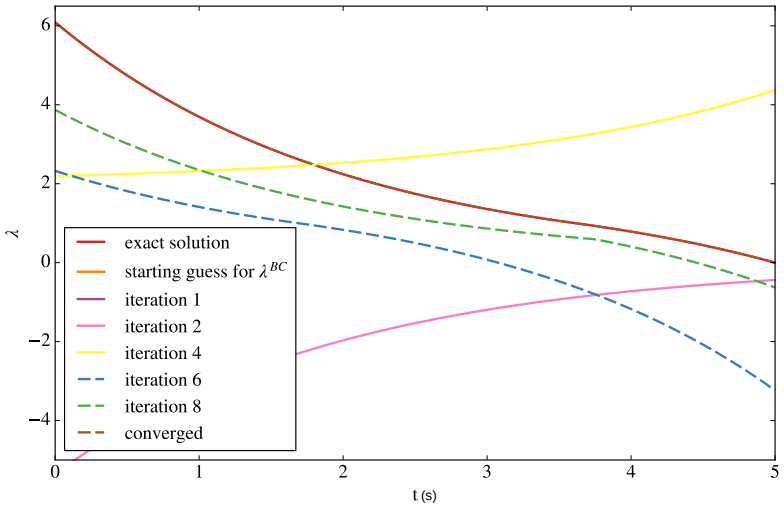
5.2.3 Discussion

Figure 5.5 encounters once more the limits of DG-FEM method, like the ones on sub-section 5.1.2.

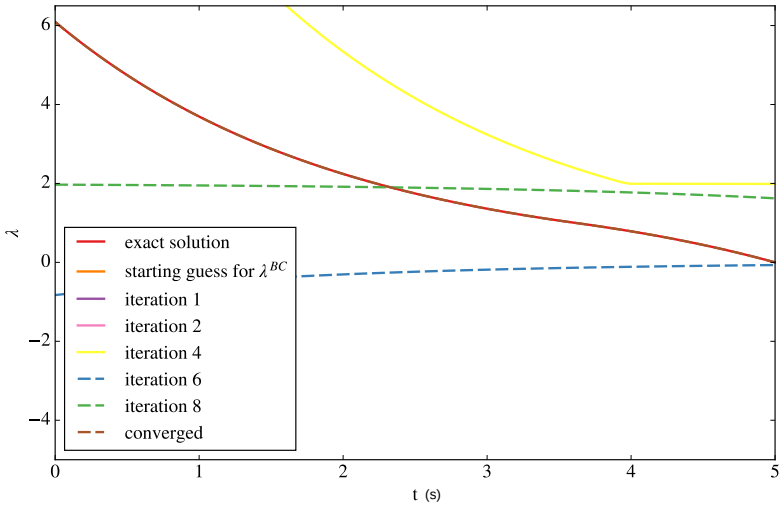
Figure 5.6 shows the dominance of the switching instant error about the DG-FEM error (note the different scales of 5.5 and 5.6 for the same points). The error respective to the estimation of t_s, with the distances to the boundary tested, is always larger than the DG-FEM error. So, the polynomial degree that is used is not reflected in the errors.

Between 5.6a and 5.6b there is almost no difference, figure 5.6a has more error associated with respect to figure 5.6b. These results reflects the importance of refinement, mainly near t_s.

Finally, 5.7 shows again the implementation of the algorithm appears correct. An Addition test was made for this problem regarding the shooting method. The algorithm was tested with different initial λ.



(a) Initial λ guess $= 100$



(b) Initial λ guess $= -100$

Figure 5.8: Sensitivity analysis of the shooting method.

The most extremes tests made are depicted in figure 5.8 $\lambda^{\text{BC}} = 100$ and $\lambda^{\text{BC}} = -100$. It was found that the shooting algorithm converged for both cases, demonstrating the method is not too sensitive to the initial guess for the boundary condition of $\boldsymbol{\lambda}$.

Chapter 6

Implementation on the OWC Spar Buoy

In this chapter the algorithm described in 4.5 (forwards integration of $\mathbf{x}$ along with u and backward integration of $\boldsymbol{\lambda}$) will be applied to the buoy, under regular and irregular wave circumstances, along with some tests to validate the system and the algorithm.

Unfortunately, the shooting method algorithm did not perform well for this system. The simulation of the $\boldsymbol{\lambda}$ states, at a certain point, becomes unstable (increasing without bounds). The instability starts with a big increase of the λ_5 state, from which follow all other $\boldsymbol{\lambda}$ states. Some fruitless efforts to repair the error (if it even is an error) were implemented: refining the simulation, simplifying the system, and normalizing Ω and P_t. Due to time constraints, this problem of unknown origin (for now) was left unsolved.

This chapter follows the reasoning of some decisions made for model testing. And will show some of the work made before performing the optimal control of the OWC spar buoy system under irregular wave climate.

6.1 OWC Spar Buoy System: Summary applied to the Optimization

This section compacts what was previously described about the OWC spar buoy dynamic system. Recall that the state variables are the heave position of the buoy and the water column x_1 and x_2, the heave speed of the buoy and the water column v_1 and v_2, the non dimensional pressure p^* and the shaft rotational speed Ω.The time variation of these is defined in chapter 2. This is just a summary, so the constants and their meaning will not be explained here, this information is already presented in chapter

2. For the hydrodynamic variables x_1, x_2, v_1 and v_2, the dynamics are:

$$\dot{x}_1 = v_1 \tag{6.1}$$

$$\dot{x}_2 = v_2 \tag{6.2}$$

$$\dot{v}_1 = \mathrm{Det}(M)^{-1}\Big((M_2 + A_{22}^{\infty})\mathcal{F}_1 - A_{12}^{\infty}\mathcal{F}_2\Big) \tag{6.3}$$

$$\dot{v}_2 = \mathrm{Det}(M)^{-1}\Big((M_2 + A_{11}^{\infty})\mathcal{F}_2 - A_{21}^{\infty}\mathcal{F}_1\Big), \tag{6.4}$$

where:

$$\mathcal{F}_1 = -\rho_{\mathrm{w}} g \mathrm{S}_1 x_1 + S_2 p_{\mathrm{atm}} p^* + F_{\mathrm{d}_1}(t) + c_1 v_1 \tag{6.5}$$

$$\mathcal{F}_2 = -\rho_{\mathrm{w}} g \mathrm{S}_2 x_2 - S_2 p_{\mathrm{atm}} p^* + F_{\mathrm{d}_2}(t) + c_2 v_2 \tag{6.6}$$

$$M = \begin{bmatrix} m_1 + A_{11}^{\infty} & A_{12}^{\infty} \\ A_{21}^{\infty} & m_2 + A_{22}^{\infty} \end{bmatrix}. \tag{6.7}$$

The dynamics of p^* with compressibility effects are:

$$\dot{p}^* = -\frac{\gamma}{\rho S_2}\frac{\dot{m}_{\mathrm{t}}(p^*+1)}{h_0 + x_1 - x_2} - \gamma\frac{(p^*+1)(\dot{x}_1 - \dot{x}_2)}{h_0 + x_1 - x_2}, \tag{6.8}$$

where $\rho = \rho_{\mathrm{atm}}(1+p^*)^{1/\gamma}$ and $\dot{m}_{\mathrm{t}}$ is defined by the turbine equations; $\dot{m}_{\mathrm{t}} = u\Phi\rho_{\mathrm{t}}\Omega \mathrm{d}^3$. The air density of the inlet/outlet of the turbine ρ_{t} has the expression $\rho_{\mathrm{t}} = \max(\rho, \rho_{\mathrm{atm}})$

The dynamics for the rotational speed of the shaft Ω are:

$$\dot{\Omega} = (T_{\mathrm{t}} - T_{\mathrm{gen}})/I. \tag{6.9}$$

T_{t} is derived from the turbine equations $T_{\mathrm{t}} = u\Pi\rho_{\mathrm{t}}\Omega^2\mathrm{d}^5$ and T_{gen} is being modeled by a control law $T_{\mathrm{gen}} = 0.025\Omega^{2.33}$. The turbine curves used follow the equations:

$$\hat{\Phi} = \frac{0.12695\Psi^4 - 0.71\Psi^3 + 5.068\Psi^2 + 4.289\Psi}{\Psi^3 - 2.561\Psi^2 + 37.46\Psi + 6.278}, \tag{6.10}$$

$$\hat{\Pi} = -272\Phi^{10} + 252\Phi^8 - 84.26\Phi^6 + 12.9\Phi^4 + 2.605\Phi^2 - 0.00657. \tag{6.11}$$

The control valve is u and only takes the values 0 (when closed) or 1 (when opened). For the problem to have a bang bang control, the Lagrangian for the optimization should be:

$$\mathcal{L} = P_{\mathrm{t}} = u\Pi\rho_{\mathrm{t}}\Omega^3\mathrm{d}^5 \tag{6.12}$$

Some previous works on this topic used a regularization term to avoid the chattering problem [17], [12]:

$$\mathcal{L} = P_{\mathrm{t}} + \epsilon(1-u)^2. \tag{6.13}$$

This extra term penalizes the time when the valve is open $u = 1$. This term was not used in this thesis,

because one of the objectives is to test the hypothesis that the chattering problem is caused by the DG-FEM formulation, the algorithm or the precision. Furthermore, for higher values of ϵ some of the power will be taken from the turbine.

6.2 Notes on the Solutions

For the next sections shown, the major concern will be the stability of the process and the physical rigor of system behavior. Assume, for the next sections (to avoid an extensive list of images) that if the system is stable and generating a reasonable u response, then the power is being maximized properly. In fact, at section 6.5, for the most complex scenario, the power is shown to ever increase for each iteration in the control calculation (figure 6.8).

The simulation and control of the buoy will be affected by the boundary conditions, which, can not be 0 for all the variables ($x_1, x_2, v_1, v_2, p^*, \Omega$). Specifically, Ω can never be 0, since the calculations of the non dimensional variables (2.21) will be impossible (division by Ω). So, the starting point for the simulation will have to be $(0, 0, 0, 0, 0, \Omega_0)$ with Ω_0 as an arbitrary value (which will vary in the following sections).

The element time for the simulation was set to be smaller than $\Delta t^e \leq 0.1$ s for larger values of Δt^e there were some instabilities, mainly presented in the p^* state and the λ co-states. As seen before, Δt^e is closely related to the error and the convergence of the fixed point method, but a smaller Δt^e has a cost in computation time. So, to demonstrate the working algorithm, only a Δt^e of 0.1 will be used. The length of the control interval will also be relevant since it provides the boundary condition for $\boldsymbol{\lambda}$. The larger the time interval the lesser influence will this boundary condition have on the first instants calculated, but, once again, the increase in the time interval will also mean an increase in computation time.

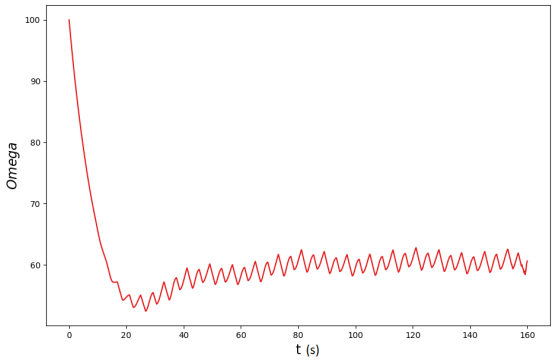
The constant c_{11} that appears as a simplification of (2.1)-(2.2) will be the maximum value of the damping 4.23×10^4 found in the tables and Henriques et al. [4] and c_{22} was set to $\frac{1}{10}$ of c_{11}. Also, for the purpose of testing the control, the significant wave height is fixed at 1.96m.

For stability reasons (regarding the boundary condition $v_1 = v_2 = 0$), the excitation force will need to have a transient time, where the wave height will rise (from 0 to the actual value), and then be ”stable” for the rest of the simulation.

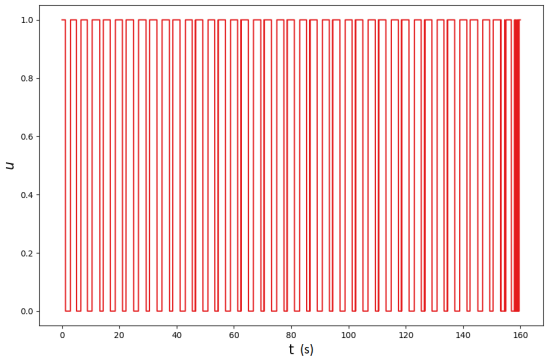
It is important that with regular waves u has a regular, symmetric, behavior, in stationary state [personal communication, J. C. C. Henriques, 2020]. The next subsections will focus on creating the conditions for this behavior to be visible in the calculated time interval. A good indicator of the system stability is the accumulation of mass, which, in theory, should not happen when the system reaches the stationary state.

6.3 Single Wave Simple Simulation

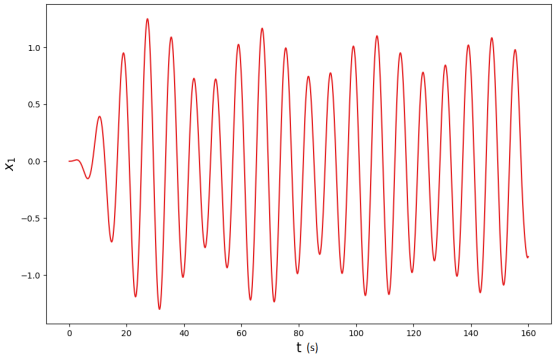
The initial, stable, test made with the buoy's system was with one wave at the period of $T = 9s$, significant wave height $H_e = 1.96$ and $\Omega_0 = 100$. Some of the results are shown bellow (in figure 6.1).



(a) Ω



(b) u



(c) x_1

Figure 6.1: Data from the simulation with one wave $T = 9$, $H_e = 1.96$ and $\Omega_0 = 100$

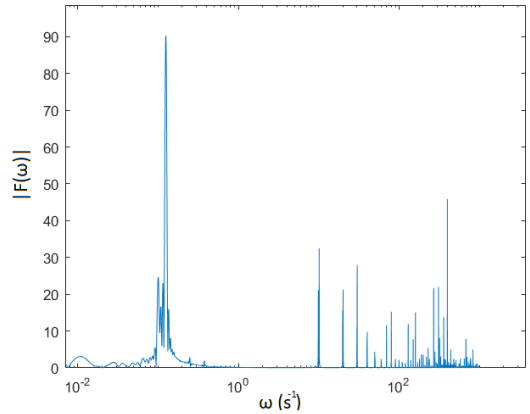


Figure 6.2: Fourier transform of the signal x_1 in figure 6.1c

6.3.1 Discussion

These variables were chosen to demonstrate that with a Ω_0 of 100 ms^{-1} there is an oscillatory behavior corresponding to the boundary conditions (described by 6.1b), from which it is hard to validate the simulation. After these results, the oscillatory behavior of the x_1 was characterized (fig 6.2), the simulation time was increased to 400 seconds and the rise time of the wave was changed from 20 seconds to 32, to accommodate multiples for the peak frequency of the x_1, x_2, v_1 and v_2 (which are not shown, but present a similar signal to x_1) responses: about 8 seconds. 6.1 also shows that Ω_0 is very superior to the average rotational speed of the turbine and, thus, should be changed to reduce the transient time of the buoy.

6.4 Single Wave Altered Excitation

Having in mind the comments about the previous simulation, and still pursuing the symmetry in u, an alteration to the excitation form is presented in figure 6.3, this excitation force will not only respect the boundary conditions of $\mathbf{x}$ in $t = 0$ but it will as well force the boundary condition at the terminal time to be respected. This is done by applying the symmetric of the rise effect to the end of the force (decreasing).

When subjected to this excitation, the algorithm reacted as in figure 6.4

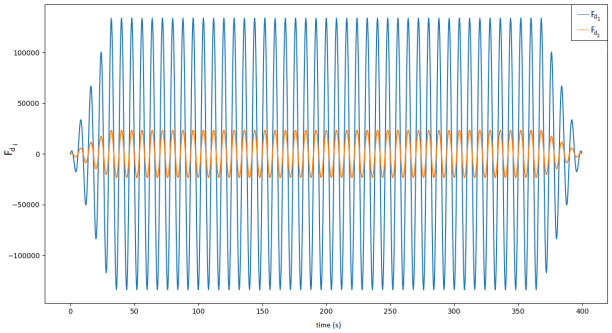
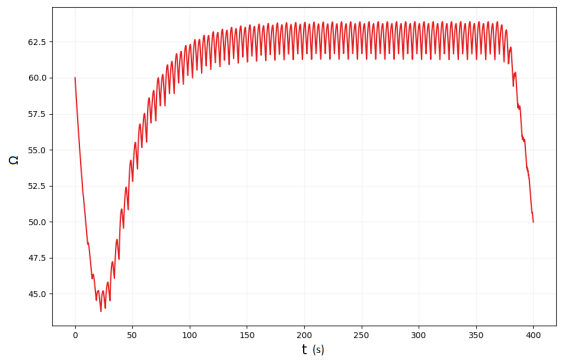
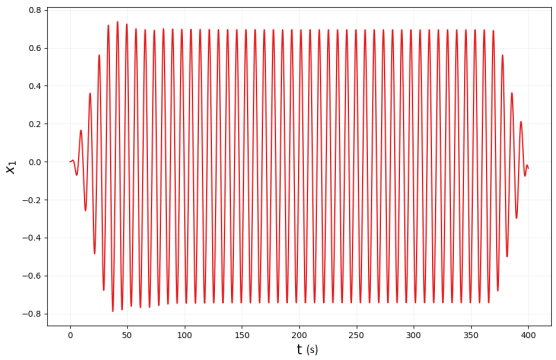


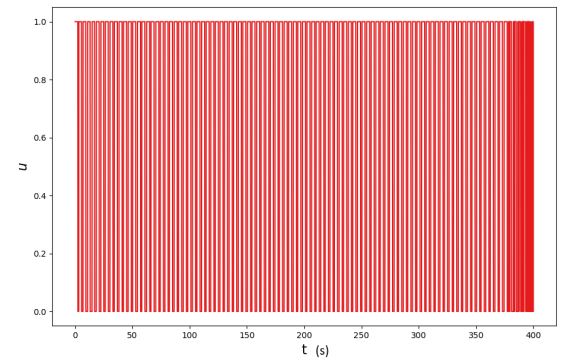
Figure 6.3: Altered excitation force



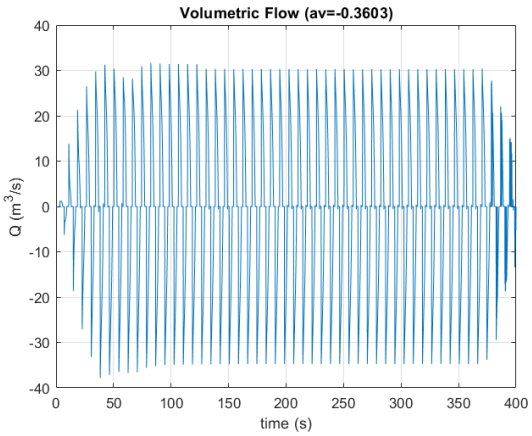
(a) Ω



(b) x_1



(c) u



(d) $\dot{m}_t$

Figure 6.4: Results with excitation force in 6.3 $T = 9$, $H_e = 1.96$ and $\Omega_0 = 60$.

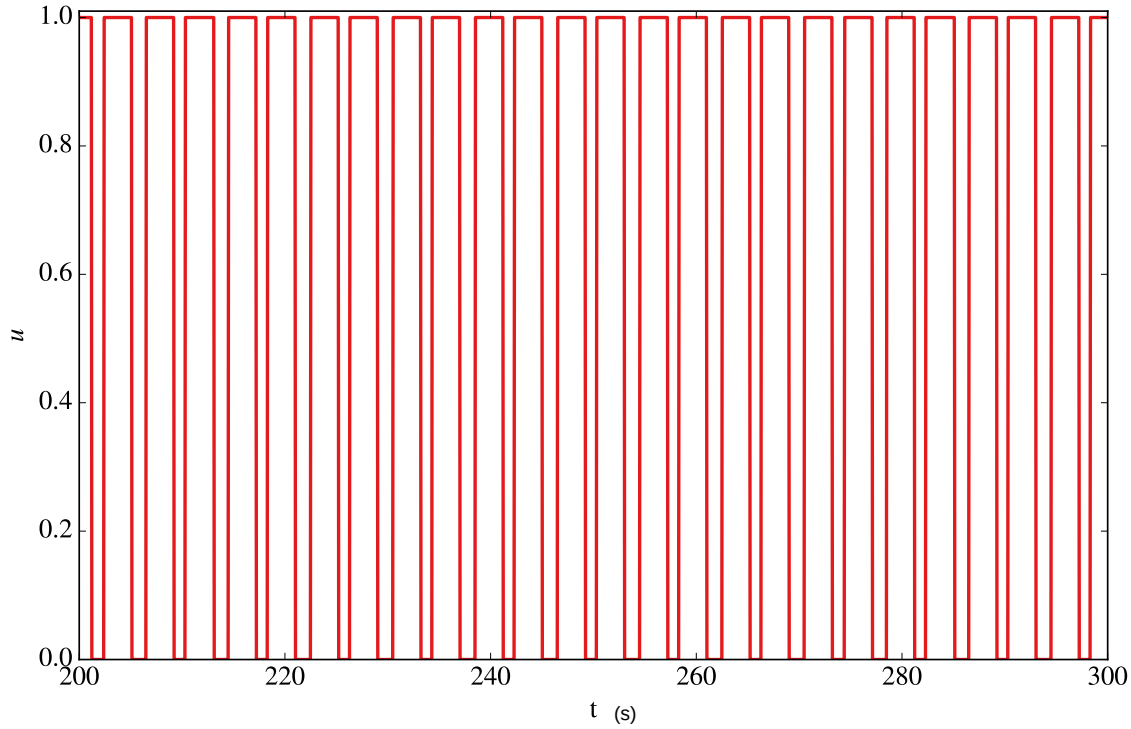


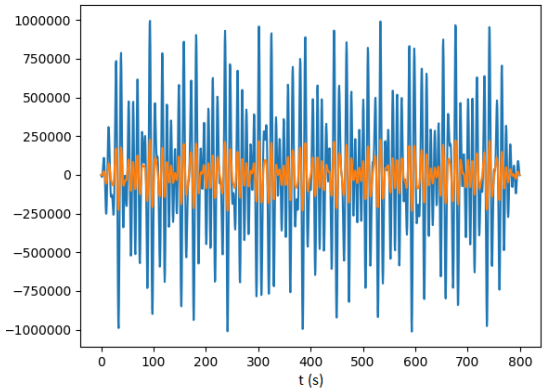
Figure 6.5: u zoomed from figure 6.4

6.4.1 Discussion

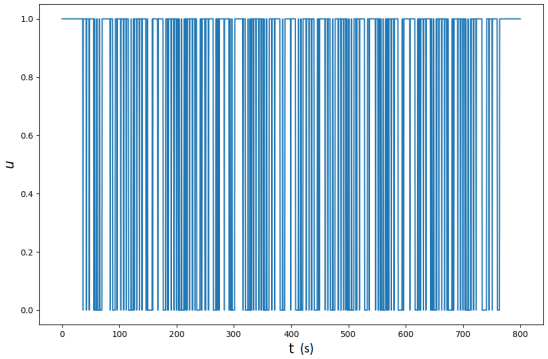
Figure 6.4b suggests that the oscillatory behavior in x_1 was almost mitigated. u in 6.4c also seems to have a more regular evolution, figure 6.5 shows a zoom in this plot for the time interval between 200 s and 300 s. Ω in 6.4a also starts closer to the average Ω. Finally, in 6.4d it is shown a plot of the flow during the calculated time, from whose inspection is concluded that the system is not accumulating mass (the average is almost 0, compared to the greatest peak, approximately 0.9% of the peak).

6.5 Irregular Waves

With the purpose of further testing the program, the system is now subjected to an irregular wave regimen with 3 waves. More waves (wave periods) may be added for a better approximation of reality, 3 unphased waves are already a fair proof of the algorithm. Figure 6.6 shows the excitation force F_{d_i} for this situation and the resultant control, which is zoomed for the times [200,300] in figure 6.7. Since the routine simulation of the system-control has been shown above that it is physically feasible, figure 6.8 presents the performance index convergence for various cycles of calculation for u and $\mathbf{x}$.



(a) Irregular wave excitation force



(b) Resultant control

Figure 6.6: a: Excitation force with 3 wave periods. b: Result of the control in irregular wave climate $H_e = 1.96, T = (8, 11, 13)$s

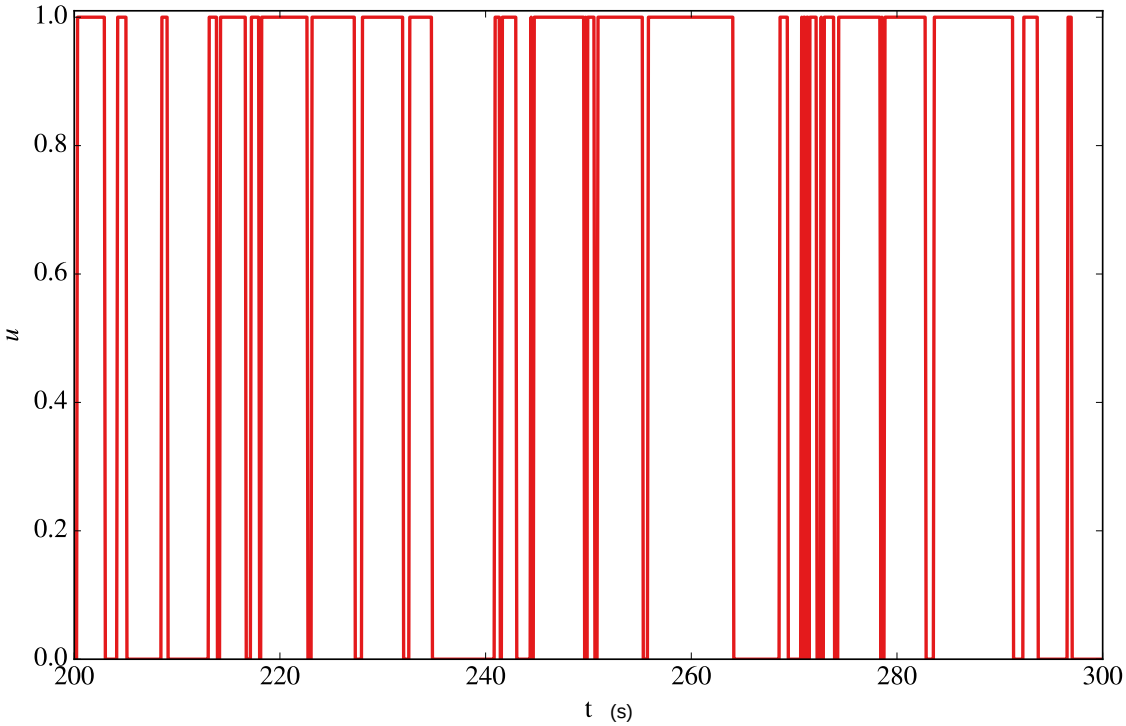


Figure 6.7: u zoomed from figure 6.6

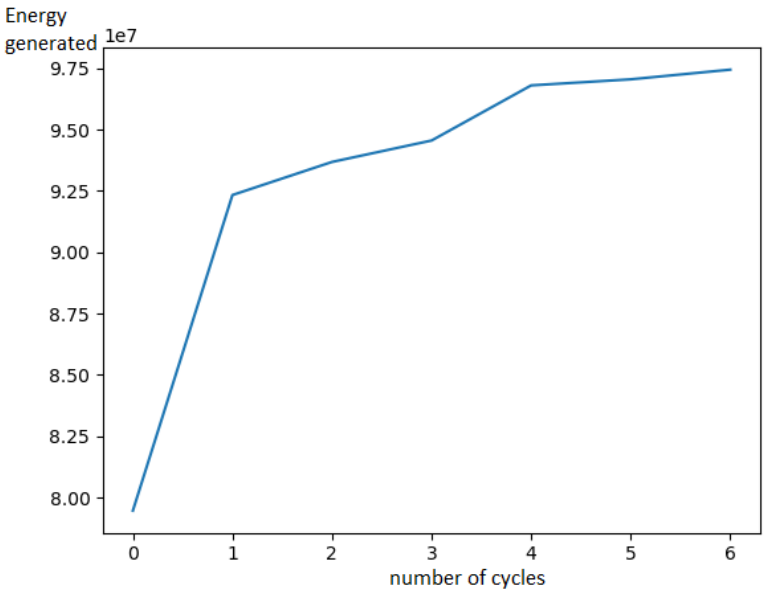


Figure 6.8: Performance index convergence over calculations of u and **x**

6.5.1 Discussion

Under these conditions, the algorithm seems to be working properly, or, at least, increasing the generated power of the non controlled system (figure 6.8 cycle 0). It seems safe to assume the maximum is being calculated (for this case).

u seems to have some chattering problems that were not shown in the regular wave signals (except at terminal time for figure 6.4). It seems that even with the introduction of arbitrary precision and another finite element formulation the chattering problem is still to be solved. But, on one hand, recalling that the regularization parameter $\epsilon(1-u)^2$ was not used, the chattering problem appears to be less evident than the one found in [12], where, even using a regularization parameter of $\epsilon = 0.01$, the control appears to have more chattering; on the other hand, since the wave behavior is irregular and there is a random phase in the model, it is hard to compare these behaviors.

Chapter 7

Conclusions

This chapter is dedicated to remarks, highlights and commentary on this thesis. First, a recapitulation of the main conclusions will be made, along with the achievements and advancements of the work.

Finally, is presented a list of possible future tasks to further complete the work developed in this thesis

7.1 Main Conclusions

The present thesis applied a Discontinuous Galerkin Finite Element Method (DG-FEM) to solve nonlinear optimal control problems based on Pontryagin's Maximum Principle. The methodology was extensively tested with arbitrary precision and refinement problems. The DG-FEM can be viewed as an alternative to the established spectral methods. Pontryagin's Maximum Principle DG-FEM has several advantages, namely the possibility of easily handling discontinuous control, as the control and the ability to use mesh refinement techniques to improve the accuracy. The spectral methods handle mesh refinement by adjusting the number of points and their global position of the mesh. However, the approximation of the solution spans across the computational domain. On the other hand, DG-FEM based significantly more versatile than the spectral methods. This method allows a simple implementation of local mesh and polynomial refinement (*hp*-refinement). The mesh refinement is based on element subdivision, while polynomial refinement uses the adjustment of the degree of the polynomial approximation within each element according to a specified criterion. Another important difference between the spectral methods and the proposed methodology concerns the maximization of the Hamiltonian. With spectral methods, the maximization is ensured at specific control points. The DG-FEM guarantees the maximization of the Hamiltonian using an integral approach that uses an element-wise continuous solution.

In this thesis, the mesh refinement in bang-bang optimal control problems was explored. An iterative approach was implemented to compute the switching instant. The results showed that it was possible to compute the switching points within the prescribed accuracy and, moreover, it was also demonstrated that the convergence was $p + 1$, where p is the order of the polynomial. This is an improvement in

comparison with previous results published in Henriques et al. [12], where the order of convergence was $p + 1/2$. This was due to a change in the DG-FEM formulation. The shooting method was also explored as a viable way to solve these problems. It performed well under all the test problems as well as the sensitivity analysis made to the admissible values of the arbitrary initial boundary condition. But, inconclusively, it was not possible to apply it to the OWC Spar buoy system.

Another important result was the demonstration that double precision (15 significant digits) is not enough to compute accurate solutions for high-order polynomial approximations. The results obtained in this thesis used a prescribed number of significant digits, typically 100, such that the order of convergence was always $p + 1$ independently of the mesh size.

To show the capabilities of the method, an optimal control problem with a continuous solution was also studied. The conclusions about the order of convergence were the same as in the case of the bang-bang problem.

The optimal control of an OWC spar buoy wave energy converter (WEC) was also implemented following the previous works made by the IST Wave Energy Group. The goal was to implement the so-called "Latching or phase control" of the WEC to maximize the power extraction. This is done by performing an optimal bang-bang control of a high-speed stop valve installed in series with the turbine rotor. The results presented for the studied OWC spar buoy showed several improvements when compared to previous published results by Henriques et al. [12]. The chattering phenomenon was largely reduced even without a regularization term used in Henriques et al. [12] to penalize the closing of the valve. Furthermore, the optimal control achieved a 20% of increase in the generated turbine power, in comparison with the uncontrolled scenario.

7.2 Future Work

Remembering that there is an annex, containing the unused work, where the buoy is modeled with the option of partial closure, it makes sense to solve the optimal control problem for this situation. This could be easily implemented in the code, but the computational time was already too high to be feasible within the framework of the current MSc thesis (a constraints module would need to be added). Still on this topic, a suggestion is made to optimize the code developed in what concerns computational time. It works well for small problems, but for cases with the complexity of the buoy, it takes a too much time to make the calculations, even with a low degree in the discontinuous Galerkin method.

The OWC spar buoy control of this thesis does not use a regularization term. Another test that can be implemented is to recalculate the control with regularization term and a very low ϵ as an attempt to completely eliminate the chattering. Furthermore, more tests with the buoy can be done (increasing the time, increasing the number of elements, making the system more complex and increase the number of waves). With this formulation, it is also possible to add the radiation terms with a more rigorous definition.

Another interesting implementation would be to control both the valve and the electromagnetic torque at the same time, having in mind the partial closure of the valve and the actual characteristics of the

doubly-fed induction generator (DFIG).

Finally, the routine written to solve the optimal control problems can always be improved with more features, for instance: the introduction of restrictions to the problem, which, once again could complete the buoy control, for example: with the addition of the u hard nonlinearity for the partial closure of the valve $u \in 0 \cup [0.4, 1]$ (due to suction forces the valve cannot close further than 40%) and the implementation of general boundary value problems (boundary conditions at the start and the end of the problem).

Bibliography

[1] A. F. de O. Falcão. Wave energy utilization : A review of the technologies. *Renewable and Sustainable Energy Reviews*, 14:899–918, 2009. doi: 10.1016/j.rser.2009.11.003.

[2] M. Penalba and J. V. Ringwood. A Review of Wave-to-Wire Models for Wave Energy Converters. *Energies*, 9, 2016. doi: 10.3390/en9070506.

[3] T. Karthikeyan, A. Samad, and R. Badhurshah. Review of air turbines for wave energy conversion. In *2013 International Conference on Renewable Energy and Sustainable Energy (ICRESE)*, pages 183–191, 2013. doi: 10.1109/ICRESE.2013.6927812.

[4] J. C. C. Henriques, J. C. C. Portillo, W. Sheng, L. M. C. Gato, and A. F. O. Falcão. Dynamics and control of air turbines for oscillating water columns : Case study. *Renewable and Sustainable Energy Reviews*, 112:571–589, 2019. doi: 10.1016/j.rser.2019.05.010.

[5] A. F. O. Falcão, J. C. C. Henriques, and L. M. C. Gato. Self-rectifying air turbines for wave energy conversion : A comparative analysis. *Renewable and Sustainable Energy Reviews*, 91(January): 1231–1241, 2018. doi: 10.1016/j.rser.2018.04.019.

[6] A. F. O. Falcão, J. C. C. Henriques, and L. M. C. Gato". Oscillating-water-column wave energy converters and air turbines: A review. *Renewable Energy*, 85:1391–1424, 01 2016. doi: 10.1016/j.renene.2015.07.086.

[7] G. Rajapakse, S. Jayasinghe, A. Fleming, and M. Negnevitsky. Grid integration and power Smoothing of an oscillating water column wave energy converter. *Energies*, 11, 2018. doi: 10.3390/en11071871.

[8] S. Ceballos, J. Rea, E. Robles, I. Lopez, J. Pou, and D. O'Sullivan. Control strategies for combining local energy storage with wells turbine oscillating water column devices. *Renewable Energy*, 83: 1097–1109, 2015. doi: 10.1016/j.renene.2015.05.030.

[9] T. Sun and S. R. K. Nielsen. Semi-active feedforward control of a floating owc point absorber for optimal power take-off. *IEEE Transactions on Sustainable Energy*, 11(3):1300–1308, 2020. doi: 10.1109/TSTE.2019.2923279.

[10] J. C. C. Henriques, L. M. C. Gato, J. M. Lemos, R. P. F. Gomes, and A. F. O. Falcão. Peak-power

control of a grid-integrated oscillating water column wave energy converter. *Energy*, 109:378 – 390, 2016. doi: 10.1016/j.energy.2016.04.098.

[11] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, University of California, Los Alamos Scientific Laboratory, 1973.

[12] J. C. C. Henriques, J. M. Lemos, L. Eça, L. M. C. Gato, and A. F. O. Falcão. A high-order Discontinuous Galerkin Method with mesh refinement for optimal control. *Automatica*, 85:70–82, 2017. doi: 10.1016/j.automatica.2017.07.029.

[13] J. N. Reddy. *Introduction to the finite element Method*. McGraw-Hill, 3rd edition, 2006. ISBN 007-124473-5.

[14] M. Baccouch. Analysis of a posteriori error estimates of the discontinuous Galerkin method for nonlinear ordinary differential equations. *Applied Numerical Mathematics*, 106:129–153, 2016. doi: 10.1016/j.apnum.2016.03.008.

[15] G. Elnagar, M. A. Kazemi, and M. Razzaghi. The pseudospectral legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995. doi: 10.1109/9.467672.

[16] I. M. Ross and M. Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182 – 197, 2012. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2012.09.002.

[17] J. N. H. Valério. Dimensionamento e controlo do sistema de conversão de energia de um dispositivo de flutuante de aproveitamento de energia das ondas baseado no princípio da coluna de água oscilante. Master's thesis, Instituto superior técnico, 2017.

[18] A. F. O. Falcão, J. C. C. Henriques, and J. J. Cândido. Dynamics and optimization of the owc spar buoy wave energy converter. *Renewable Energy*, 48:369 – 381, 2012. ISSN 0960-1481. doi: 10.1016/j.renene.2012.05.009.

[19] J. C. C. Henriques, A. F. de O. Falcão, R. P. F. Gomes, and L. M. C. Gato. Latching control of an OWC spar-buoy wave energy converter in regular waves. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 44915, pages 641–650. American Society of Mechanical Engineers, 2012.

[20] A. F. O. Falcão, L. M. C. Gato, and E. P. A. S. Nunes. A novel radial self-rectifying air turbine for use in wave energy converters. part 2. results from model testing. *Renewable Energy*, 53:159 – 164, 2013. doi: 10.1016/j.renene.2012.11.018.

[21] F. L. Lewis, D. Vrabie, and V. L. Syrmos. *Optimal Control*. John Wiley & Sons, inc., 3rd edition, 2012. ISBN 9780470633496.

[22] A. F. de O. Falcão. Control of an oscillating-water-column wave power plant for maximum energy production. *Applied Ocean Research*, 24(2):73 – 82, 2002. doi: 10.1016/S0141-1187(02)00021-4.

[23] W. H. Press, J. C. A. Wevers, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, B. Flannery, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Number v. 1 in Numerical Recipes in C book set. Cambridge University Press, 1992. ISBN 9780521431088.

[24] R. L. Burden and J. D. Faires. *Numerical Analysis*. Brooks/Cole Cengage Learning, 9th edition, 2011. ISBN 978-0-538-73351-9.

[25] Qi Gong, Wei Kang, and I. M. Ross. A pseudospectral method for the optimal control of constrained feedback linearizable systems. *IEEE Transactions on Automatic Control*, 51(7):1115–1129, July 2006. doi: 10.1109/TAC.2006.878570.

[26] D. G. Luenberger. *Introduction to dynamic systems: theory, models, and applications*. J. Wiley & Sons, New York, Chichester, Brisbane, 1979.

[27] *Curve fitting documentation*. The MathWorks, Inc. URL `https://www.mathworks.com/help/curvefit/fit.html`.

[28] D. C. Phan and S. Yamamoto. Maximum energy output of a dfig wind turbine using an improved mppt-curve method. *Energies*, 8(10):11718–11736, 2015. doi: 10.3390/en81011718.

[29] D. Yang, M. Kang, E. Muljadi, W. Gao, J. Hong, J. Choi, and Y. C. Kang. Short-term frequency response of a dfig-based wind turbine generator for rapid frequency stabilization. *Energies*, 10(11): 1863, 2017. doi: 10.3390/en10111863.

[30] J. C. C. Henriques, M. F. P. Lopes, R. P. F. Gomes, L. M. C. Gato, and A. F. O. Falcão. On the annual wave energy absorption by two-body heaving WECs with latching control. *Renewable Energy*, 45: 31–40, 2012. doi: 10.1016/j.renene.2012.01.102.

Appendix A

Numerical models for the turbine and generator

This chapter introduces numerical models for the turbine and the generator, with partial closure of the valve.

Unfortunately, due to time constraints, this work was not used for the optimal control of the turbine. The optimization of the turbine using partial closure of the valve would require the addition of constraints in the optimization formulation, which have not yet been introduced.

A.1 Numerical Model of the Turbine

The turbine will be modeled according to the data available in figures A.1 and A.2. Figure A.1 represents the non-dimensional flow (Φ) as a function of the valve opening ($u \in [0, 1]$, being 0 the closed position and 1 the open) and the non dimensional pressure head (Ψ). Figure A.2 represents the non-dimensional power (Π) as a function of the non-dimensional flow and the valve opening. These relate with the expressions in (2.21) and (2.22).

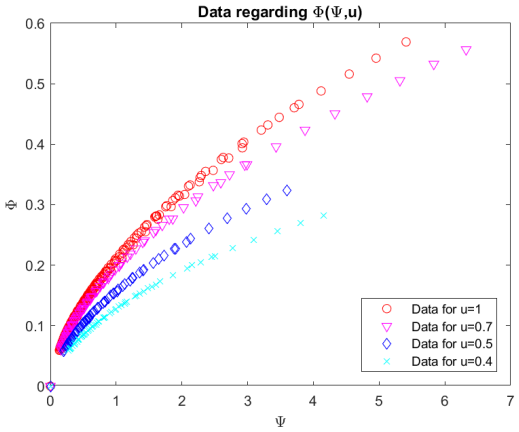


Figure A.1: Turbine non-dimensional flow

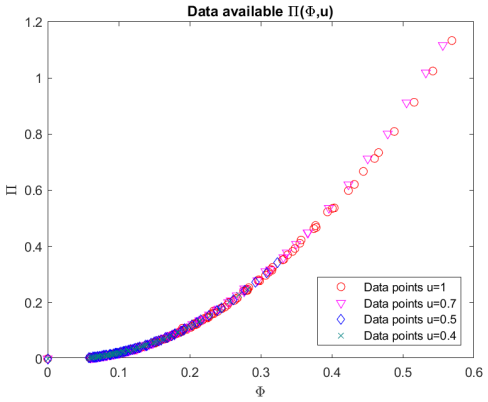


Figure A.2: Turbine non-dimensional power

A.1.1 Non Dimensional Flow Numerical Model

Inspecting the trend of the points in figure A.1, the evolution with Ψ appears to have either a rational or logarithmic form. Given the lack of points in u, it seems acceptable that a polynomial will fit. Two models were tested and refined for the fitting of Φ:

$$\hat{\Phi} = \text{sign}(p^*)(a_m u^m + \cdots + a_1 u)\log(b_n \Psi^n + b_{n-1}\Psi^{n-1} + \cdots + b_1 \Psi + 1) \tag{A.1}$$

$$\hat{\Phi} = \text{sign}(p^*)\frac{a_1\Psi + \cdots + a_{n-1}\Psi^{n-1} + (a_n + a_{n+1}u)\Psi^n}{\Psi^{n-1} + b_{n-2}\Psi^{n-2} + \cdots + b_1\Psi + b_0}(c_m u^m + \cdots + c_1 u) \tag{A.2}$$

with n and m dependent on the refinement. All a_i, b_i and c_i are independent constants to determine. Model (A.2) structure was chosen to have a linear assymptote for $\Psi \to \infty$. Furthermore, both models have been restricted to have $\Phi(\Psi = 0, u) = \Phi(\Psi, u = 0) = \Phi(0, 0) = 0$. It is assumed, of course, that the polynomial inside the logarithm in (A.1) is never less than 0 and the polynomial in the denominator in (A.2) is never 0 inside the testing domain ($\Psi \in [0, 7]$, $u \in [0, 1]$). The factor correspondent to sign(p^*) which represents a sign function (-1 if $p^* < 0$, 1 if $p^* > 0$, or 0 otherwise) describes the compression and decompression of the pressure chamber.

The estimation of the coefficients was calculated with a nonlinear least squares method [27]. Since

these are nonlinear functions of their coefficients, the initialization constants are relevant for the algorithm. So, for each optimization, 10 trials with different, random [27], starting points were made and the best of each was chosen. The result is shown in tables A.1 and A.2.

rmse Log	n=1	n=2	n=3	n=4	n=5
m=1	0.0344	0.0272	0.0259	0.0258	0.0257
m=2	0.0277	0.0272	0.0258	0.0258	0.0262
m=3	0.0104	0.0088	0.0038	0.0038	0.0040

Table A.1: Root mean square errors of proposed logarithmic function structure

rmse Rational	n=1	n=2	n=3	n=4
m=1	0.0557	0.0133	0.0117	0.0068
m=2	0.0557	0.0222	0.0190	0.0131
m=3	0.0545	0.0042	0.0231	0.0042

Table A.2: Root mean square errors of proposed rational function structure

Finally, the function that was selected (A.3) for the model was a rational function with some modifications to the above structure. (A.3) displays a root mean square error (RMSE) of 0.0025 relative to the data. The distribution of the error can be examined in figure A.3.

$$\hat{\Phi}(\Psi, u) = \frac{(0.1657 - 0.03875u)\Psi^4 - 0.71\Psi^3 + 5.068\Psi^2 + 4.289\Psi}{\Psi^3 - 2.561\Psi^2 + (33.17 - 29.74u + 34.03u^2)\Psi + 6.278u}u \tag{A.3}$$

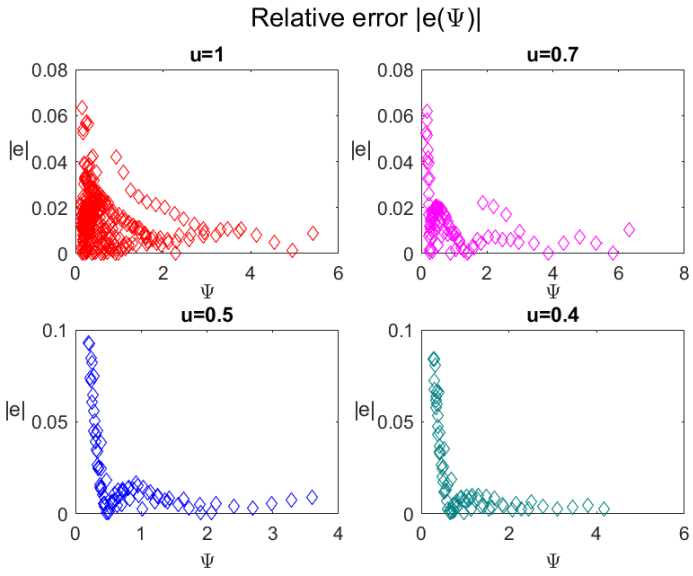


Figure A.3: Relative errors: $\frac{|\hat{\Phi}(\Psi,u)-\Phi(\Psi,u)|}{\Phi(\Psi,u)}$

Although there are some errors around 10%, it is to relevant to say that more points are available for lower Ψ and the data also has more noise on these points. This will influence the relative error

calculations. In figure A.4 it is clearly visible that these errors are almost imperceptible.

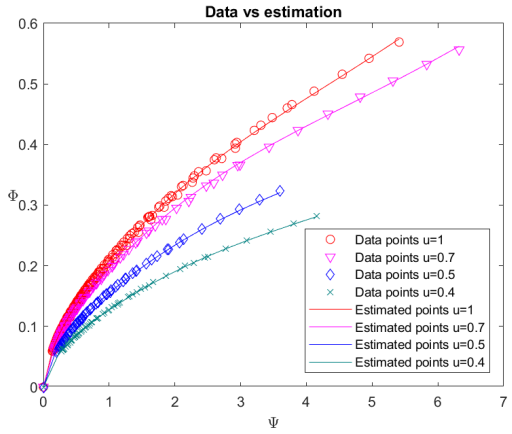


Figure A.4: Estimation $\tilde{\Phi}$

A.1.2 Non Dimensional Power Numerical Model

The data available concerning$\ll \Pi$ is represented in fig A.2. In here it is visible that Π is almost independent from u, as long as Φ is used for argument. The data corresponding to $u = 0.7$ deviated from $u = 1$ data for greater Φ, so these points were excluded for the least squares estimation, to avoid a erroneous overfiting of the data. With this, Π can be written as:

$$\hat{\Pi}(\Psi, u) = \Pi(\Phi, u) = \Pi(\Phi(\Psi, u)) \tag{A.4}$$

Still analyzing figure A.2, the curve seems to be approximately quadratic, so it is expected that a curve of that type will approximate the data well. This reasoning still leads to a rather big error due to the initial part of the curve. A way to reduce this error is to increase the polynomial degree. It is also reasonable to keep the degree of each term even, to represent a similar behavior in compression and decompression ($\Phi > 0$ and $\Phi < 0$ respectively). For low values of Φ, the shaft maintains a rotation even if there is almost no air flow (or pressure gradient), so the turbine is, in fact, consuming power, presenting a negative torque (friction effect). With that said, in this curve there should be a constant negative part. Summarizing what was said previously,

$$\hat{\Pi} = -a_0 + a_1\Phi^2 + \cdots + a_n\Phi^{2n} \tag{A.5}$$

Equation A.6 shows the fitting made for the tenth order polynomial (RMSE 0.0017) and the chosen function to use in the simulation, represented in figure A.5. Figure A.6 pictures the error distribution of the fitting. The other functions calculated and their root mean square errors can be found in table A.3

$$\hat{\Pi}(\Phi) = -272\Phi^{10} + 252\Phi^8 - 84.26\Phi^6 + 12.9\Phi^4 + 2.605\Phi^2 - 0.00657 \tag{A.6}$$

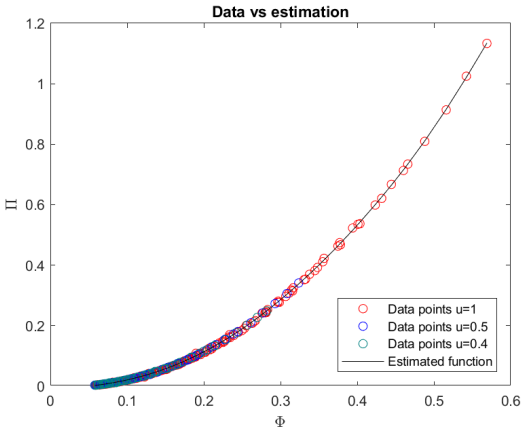


Figure A.5: Estimation $\hat{\Pi}$

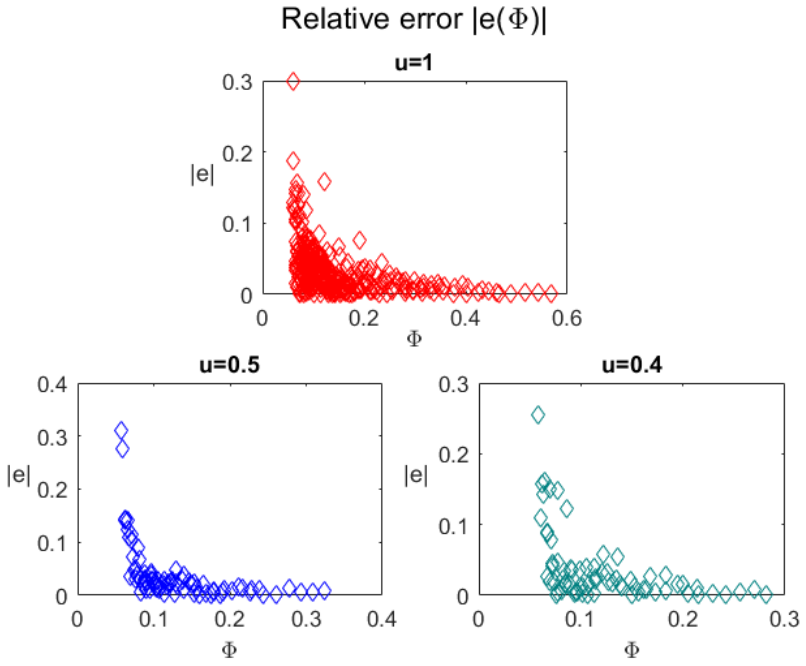


Figure A.6: Relative errors $\frac{|\hat{\Pi}(\Phi)-\Pi(\Phi)|}{\Pi(\Phi)}$

	n=1	n=2	n=3	n=4	n=5	n=6
rmse	0.0055	0.0029	0.0023	0.0018	0.0017	0.0017

Table A.3: Root mean square error for the $(2n)^{th}$ order polynomial

A.1.3 Turbine Efficiency

An optional tool to evaluate the turbine's characteristics is the efficiency. The efficiency of the turbine is defined as $\eta = \Pi/(\Phi\Psi)$ which is now possible to have as a sole function of Ψ and u. Figure A.7 pictures a surface of the efficiency as a function of these variables.

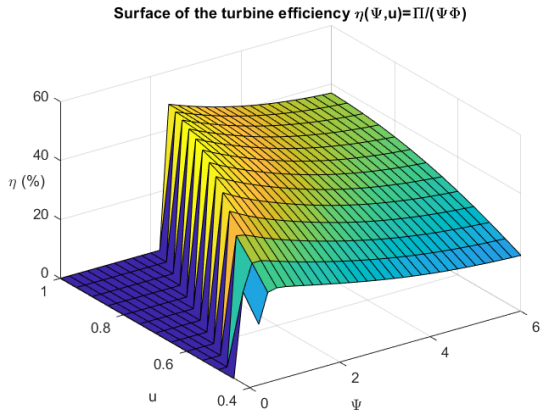


Figure A.7: Turbine efficiency surface

A.2 Numerical Model of the Generator

Since doubly fed induction generators are typically used as wind energy converters, the most common representation of the generated power comes associated with wind speed [28, 29], which is useful to skip some steps of the modeling. In this case it will be more useful to define the curve as a direct function of the torque, supplied by the turbine T_t, and the rotational speed of the system Ω.

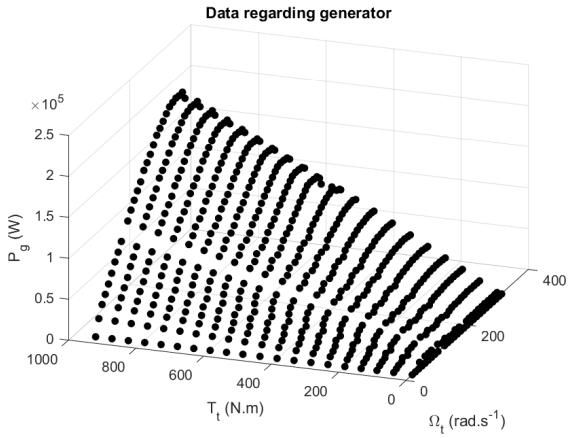


Figure A.8: Power of the generator data

Figure A.8 represents the available data about the generator. Upon inspection of this figure, a possible assumption is that the lines of constant rotational speed are nearly linear, whilst the lines of constant torque present a rather quadratic behavior. To this extent, equation (A.7) presents a generic model to apply to the generator. This model assumes that there can be higher order terms for the torque as well as the speed and assures the conditions $P_{\text{gen}}(\Omega = 0, T_\text{t}) = P_{\text{gen}}(\Omega, T_\text{t} = 0) = 0$ are respected. Table A.4 lists some of the tests made with this formula.

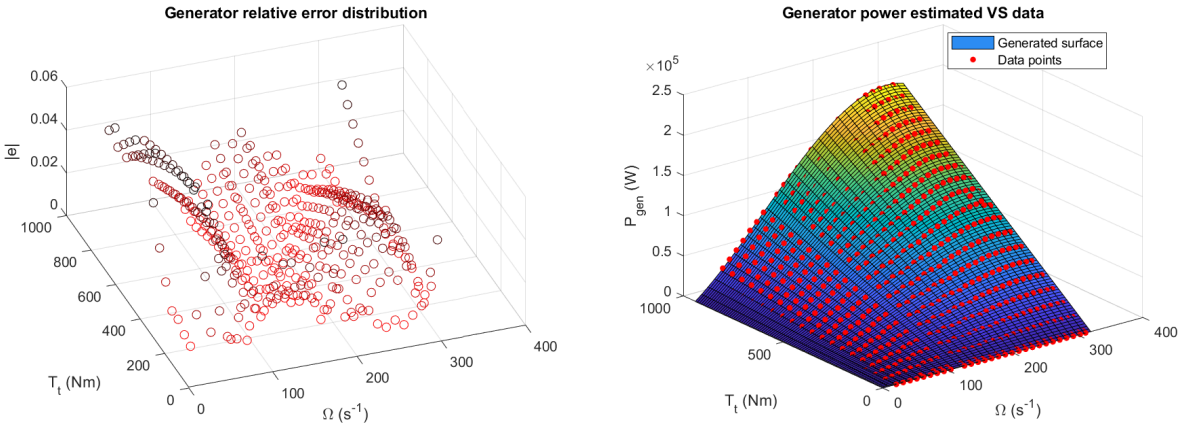
$$P_{\text{gen}} = \sum_{k=1}^{m} T_\text{t}^k (a_{nk}\Omega^n + a_{k(n-1)}\Omega^{n-1} + \cdots + a_{k1}\Omega) \tag{A.7}$$

rmse	n=2	n=3	n=4
m=1	194.9W	89.2W	88.2W
m=2	162.6W	83.5W	82.2W

Table A.4: Root mean square error for some values of m and n of equation A.7

Facing table A.4 the chosen model for this work was the model with $m = 1$ and $n = 3$, equation A.8, for its simplicity without much gain in the error compared to higher order ones. Below are figures A.9a and A.9b which depict the error distribution off the approximation and the surface created respectively.

$$P_{\text{gen}} = T_{\text{t}}(-9.18 \times 10^{-6}\Omega^3 + 2.89 \times 10^{-3}\Omega^2 + 6.84 \times 10^{-1}\Omega) \tag{A.8}$$



(a) relative error

(b) data comparison

Figure A.9: Relative error and comparison of (A.8) with the data in figure A.8

Similarly to the the turbine, an optional function that can be defined to describe the generator is the efficiency. In this case, it is easily described by (A.9). Figure A.10 shows its shape.

$$\eta = (-9.18 \times 10^{-6}\Omega^2 + 2.89 \times 10^{-3}\Omega + 6.84 \times 10^{-1}) \tag{A.9}$$

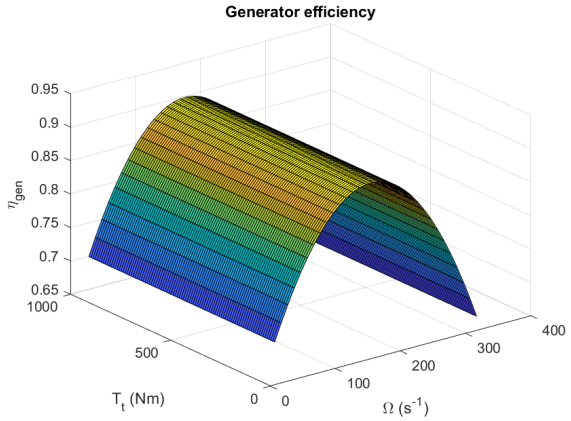


Figure A.10: Efficiency of the generator using the estimated function

Appendix B

Hydrodynamic characteristics

To calculate Γ and ϕ, the following tables were used [30], recurring, when needed, to linear interpolations between values.

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
0.0005	0.1957	0	0.2403	0.1704	0.0672	0.4802	0.1125	0.8379
0.0068	0.1957	0	0.2466	0.1692	0.0742	0.4865	0.1108	0.8769
0.0131	0.1956	0	0.253	0.1679	0.0816	0.4928	0.1092	0.9171
0.0194	0.1955	0	0.2593	0.1666	0.0896	0.4991	0.1076	0.9585
0.0258	0.1954	0	0.2656	0.1653	0.0981	0.5054	0.106	1.0013
0.0321	0.1952	0	0.2719	0.164	0.1071	0.5117	0.1043	1.0453
0.0384	0.195	0	0.2782	0.1626	0.1167	0.518	0.1027	1.0906
0.0447	0.1947	0.0001	0.2845	0.1612	0.1269	0.5243	0.1011	1.1373
0.051	0.1945	0.0002	0.2908	0.1598	0.1377	0.5307	0.0995	1.1852
0.0573	0.1942	0.0002	0.2971	0.1584	0.1491	0.537	0.0979	1.2347
0.0636	0.1938	0.0004	0.3034	0.157	0.1612	0.5433	0.0963	1.2855
0.0699	0.1934	0.0005	0.3098	0.1555	0.174	0.5496	0.0947	1.3378
0.0762	0.193	0.0008	0.3161	0.154	0.1874	0.5559	0.0931	1.3913
0.0826	0.1925	0.001	0.3224	0.1526	0.2015	0.5622	0.0916	1.4465
0.0889	0.192	0.0014	0.3287	0.1511	0.2164	0.5685	0.09	1.5031
0.0952	0.1915	0.0018	0.335	0.1495	0.232	0.5748	0.0885	1.5612
0.1015	0.1909	0.0023	0.3413	0.148	0.2483	0.5811	0.0869	1.6209
0.1078	0.1903	0.003	0.3476	0.1465	0.2655	0.5875	0.0854	1.6822
0.1141	0.1897	0.0037	0.3539	0.1449	0.2834	0.5938	0.0839	1.7449
0.1204	0.189	0.0046	0.3602	0.1434	0.3021	0.6001	0.0824	1.8096
0.1267	0.1883	0.0056	0.3666	0.1418	0.3217	0.6064	0.0809	1.8757
0.133	0.1876	0.0068	0.3729	0.1402	0.3421	0.6127	0.0794	1.9434
0.1394	0.1868	0.0082	0.3792	0.1386	0.3635	0.619	0.0779	2.0128
0.1457	0.186	0.0097	0.3855	0.137	0.3856	0.6253	0.0765	2.0841
0.152	0.1852	0.0115	0.3918	0.1354	0.4087	0.6316	0.075	2.1569
0.1583	0.1843	0.0134	0.3981	0.1338	0.4327	0.6379	0.0736	2.2317
0.1646	0.1834	0.0157	0.4044	0.1322	0.4577	0.6443	0.0722	2.3082
0.1709	0.1825	0.0181	0.4107	0.1305	0.4836	0.6506	0.0708	2.3866
0.1772	0.1816	0.0209	0.4171	0.1289	0.5105	0.6569	0.0694	2.4666
0.1835	0.1806	0.0239	0.4234	0.1273	0.5384	0.6632	0.068	2.5488
0.1898	0.1796	0.0272	0.4297	0.1256	0.5673	0.6695	0.0667	2.6328
0.1962	0.1785	0.0309	0.436	0.124	0.5973	0.6758	0.0653	2.7187
0.2025	0.1774	0.0349	0.4423	0.1224	0.6283	0.6821	0.064	2.8063
0.2088	0.1763	0.0393	0.4486	0.1207	0.6604	0.6884	0.0627	2.8963
0.2151	0.1752	0.044	0.4549	0.1191	0.6936	0.6947	0.0614	2.988
0.2214	0.1741	0.0492	0.4612	0.1174	0.7279	0.7011	0.0601	3.0821
0.2277	0.1729	0.0548	0.4675	0.1158	0.7634	0.7074	0.0588	3.1779
0.234	0.1717	0.0608	0.4739	0.1141	0.8001	0.7137	0.0576	3.2757

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
0.72	0.0563	3.3759	0.9598	0.0217	8.9419	1.1997	0.0067	18.3846
0.7263	0.0551	3.478	0.9661	0.0211	9.139	1.206	0.0065	18.6876
0.7326	0.0539	3.5826	0.9724	0.0205	9.3385	1.2123	0.0063	18.9936
0.7389	0.0527	3.6894	0.9788	0.02	9.5411	1.2186	0.0061	19.3026
0.7452	0.0516	3.7979	0.9851	0.0194	9.746	1.2249	0.0059	19.6143
0.7516	0.0504	3.909	0.9914	0.0188	9.954	1.2312	0.0057	19.9308
0.7579	0.0493	4.0219	0.9977	0.0183	10.1647	1.2375	0.0055	20.2471
0.7642	0.0482	4.1375	1.004	0.0178	10.3782	1.2438	0.0053	20.5678
0.7705	0.0471	4.2552	1.0103	0.0173	10.5948	1.2501	0.0052	20.8875
0.7768	0.046	4.3753	1.0166	0.0168	10.8134	1.2565	0.005	21.2145
0.7831	0.0449	4.4979	1.0229	0.0163	11.0353	1.2628	0.0048	21.5454
0.7894	0.0439	4.6225	1.0292	0.0158	11.2607	1.2691	0.0047	21.8763
0.7957	0.0428	4.7495	1.0356	0.0153	11.4882	1.2754	0.0045	22.211
0.802	0.0418	4.8792	1.0419	0.0149	11.7181	1.2817	0.0044	22.5459
0.8084	0.0408	5.0109	1.0482	0.0144	11.9512	1.288	0.0042	22.8876
0.8147	0.0398	5.1456	1.0545	0.014	12.1862	1.2943	0.0041	23.2317
0.821	0.0389	5.2821	1.0608	0.0136	12.4246	1.3006	0.0039	23.5787
0.8273	0.0379	5.4214	1.0671	0.0132	12.6669	1.3069	0.0038	23.9244
0.8336	0.037	5.5632	1.0734	0.0128	12.9109	1.3133	0.0037	24.2776
0.8399	0.0361	5.7075	1.0797	0.0124	13.1576	1.3196	0.0035	24.6311
0.8462	0.0352	5.8542	1.0861	0.012	13.4075	1.3259	0.0034	24.9882
0.8525	0.0343	6.0036	1.0924	0.0116	13.6604	1.3322	0.0033	25.3462
0.8588	0.0334	6.1553	1.0987	0.0113	13.9151	1.3385	0.0032	25.7113
0.8652	0.0326	6.3097	1.105	0.0109	14.1734	1.3448	0.0031	26.079
0.8715	0.0318	6.4669	1.1113	0.0106	14.4352	1.3511	0.003	26.4469
0.8778	0.0309	6.6262	1.1176	0.0103	14.6983	1.3574	0.0029	26.8162
0.8841	0.0301	6.7886	1.1239	0.0099	14.9653	1.3638	0.0028	27.1888
0.8904	0.0294	6.9539	1.1302	0.0096	15.2351	1.3701	0.0027	27.5644
0.8967	0.0286	7.121	1.1365	0.0093	15.5063	1.3764	0.0026	27.9443
0.903	0.0278	7.2912	1.1429	0.009	15.7825	1.3827	0.0025	28.331
0.9093	0.0271	7.4638	1.1492	0.0087	16.0596	1.389	0.0024	28.7153
0.9156	0.0264	7.6393	1.1555	0.0085	16.3406	1.3953	0.0023	29.0977
0.922	0.0257	7.817	1.1618	0.0082	16.6244	1.4016	0.0023	29.4888
0.9283	0.025	7.9979	1.1681	0.0079	16.9106	1.4079	0.0022	29.8938
0.9346	0.0243	8.1811	1.1744	0.0077	17.1989	1.4142	0.0021	30.286
0.9409	0.0236	8.3673	1.1807	0.0074	17.4924	1.4205	0.002	30.6799
0.9472	0.023	8.5565	1.187	0.0072	17.786	1.4269	0.002	31.0817
0.9535	0.0224	8.7477	1.1933	0.007	18.0841	1.4332	0.0019	31.4952

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
1.4395	0.0018	31.9029	1.6793	0.0005	49.5096	1.9191	0.0002	71.394
1.4458	0.0018	32.3097	1.6856	0.0005	49.9967	1.9255	0.0002	71.9872
1.4521	0.0017	32.7185	1.6919	0.0005	50.5633	1.9318	0.0002	72.6758
1.4584	0.0017	33.1417	1.6982	0.0005	51.0784	1.9381	0.0002	73.1555
1.4647	0.0016	33.5525	1.7046	0.0005	51.6402	1.9444	0.0002	73.8884
1.471	0.0015	33.9819	1.7109	0.0005	52.1375	1.9507	0.0002	74.5997
1.4774	0.0015	34.416	1.7172	0.0004	52.6335	1.957	0.0002	75.311
1.4837	0.0014	34.8366	1.7235	0.0004	53.1744	1.9633	0.0002	76.0661
1.49	0.0014	35.2485	1.7298	0.0004	53.7092	1.9696	0.0002	76.585
1.4963	0.0013	35.6986	1.7361	0.0004	54.3184	1.976	0.0002	77.223
1.5026	0.0013	36.1283	1.7424	0.0004	54.8046	1.9823	0.0002	78.0633
1.5089	0.0013	36.5667	1.7487	0.0004	55.3781	1.9886	0.0002	78.5046
1.5152	0.0012	37.0134	1.7551	0.0004	55.9164	1.9949	0.0001	79.275
1.5215	0.0012	37.464	1.7614	0.0004	56.4987	2.0012	0.0001	80.099
1.5278	0.0011	37.9107	1.7677	0.0004	57.0543	2.0075	0.0001	80.6935
1.5342	0.0011	38.3616	1.774	0.0003	57.5951	2.0138	0.0001	80.9793
1.5405	0.0011	38.8312	1.7803	0.0003	58.1507	2.0201	0.0001	81.8224
1.5468	0.001	39.2733	1.7866	0.0003	58.6616	2.0264	0.0001	82.5976
1.5531	0.001	39.7217	1.7929	0.0003	59.3178	2.0328	0.0001	83.383
1.5594	0.001	40.2214	1.7992	0.0003	59.8737	2.0391	0.0001	83.9755
1.5657	0.0009	40.6511	1.8055	0.0003	60.4444	2.0454	0.0001	84.7052
1.572	0.0009	41.1271	1.8119	0.0003	60.9868	2.0517	0.0001	85.2545
1.5783	0.0009	41.5905	1.8182	0.0003	61.7111	2.058	0.0001	85.9317
1.5846	0.0008	42.0703	1.8245	0.0003	62.2187	2.0643	0.0001	86.5997
1.591	0.0008	42.5444	1.8308	0.0003	62.8077	2.0706	0.0001	87.3484
1.5973	0.0008	43.0237	1.8371	0.0003	63.451	2.0769	0.0001	87.8355
1.6036	0.0008	43.4795	1.8434	0.0003	63.9639	2.0832	0.0001	88.7253
1.6099	0.0007	43.9737	1.8497	0.0003	64.6355	2.0895	0.0001	89.5154
1.6162	0.0007	44.4685	1.856	0.0002	65.1924	2.0959	0.0001	90.0191
1.6225	0.0007	44.9664	1.8623	0.0002	65.769	2.1022	0.0001	90.8624
1.6288	0.0007	45.4451	1.8687	0.0002	66.416	2.1085	0.0001	91.6364
1.6351	0.0007	45.9526	1.875	0.0002	66.99	2.1148	0.0001	92.0267
1.6414	0.0006	46.4462	1.8813	0.0002	67.6793	2.1211	0.0001	93.0351
1.6478	0.0006	46.9627	1.8876	0.0002	68.1907	2.1274	0.0001	93.4891
1.6541	0.0006	47.4606	1.8939	0.0002	68.8472	2.1337	0.0001	94.3442
1.6604	0.0006	47.9512	1.9002	0.0002	69.5684	2.14	0.0001	94.7804
1.6667	0.0006	48.4592	1.9065	0.0002	70.1646	2.1463	0.0001	95.9366
1.673	0.0005	49.0061	1.9128	0.0002	70.836	2.1527	0.0001	96.8302

ω	Γ	ϕ	ω	Γ	ϕ
2.159	0.0001	97.4482	2.3988	0	126.8652
2.1653	0.0001	97.8753	2.4051	0	128.429
2.1716	0.0001	98.9213	2.4114	0	128.6548
2.1779	0.0001	99.5715	2.4177	0	129.8596
2.1842	0.0001	100.4229	2.4241	0	130.4772
2.1905	0.0001	101.165	2.4304	0	131.4195
2.1968	0.0001	101.7972	2.4367	0	132.0587
2.2032	0.0001	102.6183	2.443	0	132.6356
2.2095	0.0001	103.3237	2.4493	0	133.9373
2.2158	0.0001	104.0371	2.4556	0	134.8715
2.2221	0.0001	104.9279	2.4619	0	135.5872
2.2284	0.0001	105.4146	2.4682	0	136.3218
2.2347	0.0001	106.4893	2.4745	0	137.178
2.241	0.0001	107.0218	2.4808	0	138.3713
2.2473	0.0001	108.1602	2.4872	0	138.6782
2.2536	0.0001	108.884	2.4935	0	139.88
2.26	0.0001	109.6022	2.4998	0	141.3747
2.2663	0.0001	110.2289	2.5061	0	141.7056
2.2726	0.0001	111.1854	2.5124	0	142.3253
2.2789	0.0001	112.0914	2.5187	0	143.6906
2.2852	0.0001	112.809	2.525	0	144.0796
2.2915	0.0001	113.4282			
2.2978	0.0001	114.5414			
2.3041	0.0001	115.5488			
2.3105	0.0001	116.2418			
2.3168	0.0001	116.601			
2.3231	0.0001	117.3532			
2.3294	0.0001	118.0722			
2.3357	0.0001	119.1078			
2.342	0	119.9322			
2.3483	0	120.7954			
2.3546	0	121.2256			
2.3609	0	122.3803			
2.3672	0	122.6994			
2.3736	0	123.7263			
2.3799	0	125.0194			
2.3862	0	125.8216			
2.3925	0	126.2476			

Table B.1: Data for the excitation force F_{d_1}

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
0.0005	0.1957	0	0.2403	0.1704	0.0672	0.4802	0.1125	0.8379
0.0068	0.1957	0	0.2466	0.1692	0.0742	0.4865	0.1108	0.8769
0.0131	0.1956	0	0.253	0.1679	0.0816	0.4928	0.1092	0.9171
0.0194	0.1955	0	0.2593	0.1666	0.0896	0.4991	0.1076	0.9585
0.0258	0.1954	0	0.2656	0.1653	0.0981	0.5054	0.106	1.0013
0.0321	0.1952	0	0.2719	0.164	0.1071	0.5117	0.1043	1.0453
0.0384	0.195	0	0.2782	0.1626	0.1167	0.518	0.1027	1.0906
0.0447	0.1947	0.0001	0.2845	0.1612	0.1269	0.5243	0.1011	1.1373
0.051	0.1945	0.0002	0.2908	0.1598	0.1377	0.5307	0.0995	1.1852
0.0573	0.1942	0.0002	0.2971	0.1584	0.1491	0.537	0.0979	1.2347
0.0636	0.1938	0.0004	0.3034	0.157	0.1612	0.5433	0.0963	1.2855
0.0699	0.1934	0.0005	0.3098	0.1555	0.174	0.5496	0.0947	1.3378
0.0762	0.193	0.0008	0.3161	0.154	0.1874	0.5559	0.0931	1.3913
0.0826	0.1925	0.001	0.3224	0.1526	0.2015	0.5622	0.0916	1.4465
0.0889	0.192	0.0014	0.3287	0.1511	0.2164	0.5685	0.09	1.5031
0.0952	0.1915	0.0018	0.335	0.1495	0.232	0.5748	0.0885	1.5612
0.1015	0.1909	0.0023	0.3413	0.148	0.2483	0.5811	0.0869	1.6209
0.1078	0.1903	0.003	0.3476	0.1465	0.2655	0.5875	0.0854	1.6822
0.1141	0.1897	0.0037	0.3539	0.1449	0.2834	0.5938	0.0839	1.7449
0.1204	0.189	0.0046	0.3602	0.1434	0.3021	0.6001	0.0824	1.8096
0.1267	0.1883	0.0056	0.3666	0.1418	0.3217	0.6064	0.0809	1.8757
0.133	0.1876	0.0068	0.3729	0.1402	0.3421	0.6127	0.0794	1.9434
0.1394	0.1868	0.0082	0.3792	0.1386	0.3635	0.619	0.0779	2.0128
0.1457	0.186	0.0097	0.3855	0.137	0.3856	0.6253	0.0765	2.0841
0.152	0.1852	0.0115	0.3918	0.1354	0.4087	0.6316	0.075	2.1569
0.1583	0.1843	0.0134	0.3981	0.1338	0.4327	0.6379	0.0736	2.2317
0.1646	0.1834	0.0157	0.4044	0.1322	0.4577	0.6443	0.0722	2.3082
0.1709	0.1825	0.0181	0.4107	0.1305	0.4836	0.6506	0.0708	2.3866
0.1772	0.1816	0.0209	0.4171	0.1289	0.5105	0.6569	0.0694	2.4666
0.1835	0.1806	0.0239	0.4234	0.1273	0.5384	0.6632	0.068	2.5488
0.1898	0.1796	0.0272	0.4297	0.1256	0.5673	0.6695	0.0667	2.6328
0.1962	0.1785	0.0309	0.436	0.124	0.5973	0.6758	0.0653	2.7187
0.2025	0.1774	0.0349	0.4423	0.1224	0.6283	0.6821	0.064	2.8063
0.2088	0.1763	0.0393	0.4486	0.1207	0.6604	0.6884	0.0627	2.8963
0.2151	0.1752	0.044	0.4549	0.1191	0.6936	0.6947	0.0614	2.988
0.2214	0.1741	0.0492	0.4612	0.1174	0.7279	0.7011	0.0601	3.0821
0.2277	0.1729	0.0548	0.4675	0.1158	0.7634	0.7074	0.0588	3.1779
0.234	0.1717	0.0608	0.4739	0.1141	0.8001	0.7137	0.0576	3.2757

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
0.72	0.0563	3.3759	0.9598	0.0217	8.9419	1.1997	0.0067	18.3846
0.7263	0.0551	3.478	0.9661	0.0211	9.139	1.206	0.0065	18.6876
0.7326	0.0539	3.5826	0.9724	0.0205	9.3385	1.2123	0.0063	18.9936
0.7389	0.0527	3.6894	0.9788	0.02	9.5411	1.2186	0.0061	19.3026
0.7452	0.0516	3.7979	0.9851	0.0194	9.746	1.2249	0.0059	19.6143
0.7516	0.0504	3.909	0.9914	0.0188	9.954	1.2312	0.0057	19.9308
0.7579	0.0493	4.0219	0.9977	0.0183	10.1647	1.2375	0.0055	20.2471
0.7642	0.0482	4.1375	1.004	0.0178	10.3782	1.2438	0.0053	20.5678
0.7705	0.0471	4.2552	1.0103	0.0173	10.5948	1.2501	0.0052	20.8875
0.7768	0.046	4.3753	1.0166	0.0168	10.8134	1.2565	0.005	21.2145
0.7831	0.0449	4.4979	1.0229	0.0163	11.0353	1.2628	0.0048	21.5454
0.7894	0.0439	4.6225	1.0292	0.0158	11.2607	1.2691	0.0047	21.8763
0.7957	0.0428	4.7495	1.0356	0.0153	11.4882	1.2754	0.0045	22.211
0.802	0.0418	4.8792	1.0419	0.0149	11.7181	1.2817	0.0044	22.5459
0.8084	0.0408	5.0109	1.0482	0.0144	11.9512	1.288	0.0042	22.8876
0.8147	0.0398	5.1456	1.0545	0.014	12.1862	1.2943	0.0041	23.2317
0.821	0.0389	5.2821	1.0608	0.0136	12.4246	1.3006	0.0039	23.5787
0.8273	0.0379	5.4214	1.0671	0.0132	12.6669	1.3069	0.0038	23.9244
0.8336	0.037	5.5632	1.0734	0.0128	12.9109	1.3133	0.0037	24.2776
0.8399	0.0361	5.7075	1.0797	0.0124	13.1576	1.3196	0.0035	24.6311
0.8462	0.0352	5.8542	1.0861	0.012	13.4075	1.3259	0.0034	24.9882
0.8525	0.0343	6.0036	1.0924	0.0116	13.6604	1.3322	0.0033	25.3462
0.8588	0.0334	6.1553	1.0987	0.0113	13.9151	1.3385	0.0032	25.7113
0.8652	0.0326	6.3097	1.105	0.0109	14.1734	1.3448	0.0031	26.079
0.8715	0.0318	6.4669	1.1113	0.0106	14.4352	1.3511	0.003	26.4469
0.8778	0.0309	6.6262	1.1176	0.0103	14.6983	1.3574	0.0029	26.8162
0.8841	0.0301	6.7886	1.1239	0.0099	14.9653	1.3638	0.0028	27.1888
0.8904	0.0294	6.9539	1.1302	0.0096	15.2351	1.3701	0.0027	27.5644
0.8967	0.0286	7.121	1.1365	0.0093	15.5063	1.3764	0.0026	27.9443
0.903	0.0278	7.2912	1.1429	0.009	15.7825	1.3827	0.0025	28.331
0.9093	0.0271	7.4638	1.1492	0.0087	16.0596	1.389	0.0024	28.7153
0.9156	0.0264	7.6393	1.1555	0.0085	16.3406	1.3953	0.0023	29.0977
0.922	0.0257	7.817	1.1618	0.0082	16.6244	1.4016	0.0023	29.4888
0.9283	0.025	7.9979	1.1681	0.0079	16.9106	1.4079	0.0022	29.8938
0.9346	0.0243	8.1811	1.1744	0.0077	17.1989	1.4142	0.0021	30.286
0.9409	0.0236	8.3673	1.1807	0.0074	17.4924	1.4205	0.002	30.6799
0.9472	0.023	8.5565	1.187	0.0072	17.786	1.4269	0.002	31.0817
0.9535	0.0224	8.7477	1.1933	0.007	18.0841	1.4332	0.0019	31.4952

ω	Γ	ϕ	ω	Γ	ϕ	ω	Γ	ϕ
1.4395	0.0018	31.9029	1.6793	0.0005	49.5096	1.9191	0.0002	71.394
1.4458	0.0018	32.3097	1.6856	0.0005	49.9967	1.9255	0.0002	71.9872
1.4521	0.0017	32.7185	1.6919	0.0005	50.5633	1.9318	0.0002	72.6758
1.4584	0.0017	33.1417	1.6982	0.0005	51.0784	1.9381	0.0002	73.1555
1.4647	0.0016	33.5525	1.7046	0.0005	51.6402	1.9444	0.0002	73.8884
1.471	0.0015	33.9819	1.7109	0.0005	52.1375	1.9507	0.0002	74.5997
1.4774	0.0015	34.416	1.7172	0.0004	52.6335	1.957	0.0002	75.311
1.4837	0.0014	34.8366	1.7235	0.0004	53.1744	1.9633	0.0002	76.0661
1.49	0.0014	35.2485	1.7298	0.0004	53.7092	1.9696	0.0002	76.585
1.4963	0.0013	35.6986	1.7361	0.0004	54.3184	1.976	0.0002	77.223
1.5026	0.0013	36.1283	1.7424	0.0004	54.8046	1.9823	0.0002	78.0633
1.5089	0.0013	36.5667	1.7487	0.0004	55.3781	1.9886	0.0002	78.5046
1.5152	0.0012	37.0134	1.7551	0.0004	55.9164	1.9949	0.0001	79.275
1.5215	0.0012	37.464	1.7614	0.0004	56.4987	2.0012	0.0001	80.099
1.5278	0.0011	37.9107	1.7677	0.0004	57.0543	2.0075	0.0001	80.6935
1.5342	0.0011	38.3616	1.774	0.0003	57.5951	2.0138	0.0001	80.9793
1.5405	0.0011	38.8312	1.7803	0.0003	58.1507	2.0201	0.0001	81.8224
1.5468	0.001	39.2733	1.7866	0.0003	58.6616	2.0264	0.0001	82.5976
1.5531	0.001	39.7217	1.7929	0.0003	59.3178	2.0328	0.0001	83.383
1.5594	0.001	40.2214	1.7992	0.0003	59.8737	2.0391	0.0001	83.9755
1.5657	0.0009	40.6511	1.8055	0.0003	60.4444	2.0454	0.0001	84.7052
1.572	0.0009	41.1271	1.8119	0.0003	60.9868	2.0517	0.0001	85.2545
1.5783	0.0009	41.5905	1.8182	0.0003	61.7111	2.058	0.0001	85.9317
1.5846	0.0008	42.0703	1.8245	0.0003	62.2187	2.0643	0.0001	86.5997
1.591	0.0008	42.5444	1.8308	0.0003	62.8077	2.0706	0.0001	87.3484
1.5973	0.0008	43.0237	1.8371	0.0003	63.451	2.0769	0.0001	87.8355
1.6036	0.0008	43.4795	1.8434	0.0003	63.9639	2.0832	0.0001	88.7253
1.6099	0.0007	43.9737	1.8497	0.0003	64.6355	2.0895	0.0001	89.5154
1.6162	0.0007	44.4685	1.856	0.0002	65.1924	2.0959	0.0001	90.0191
1.6225	0.0007	44.9664	1.8623	0.0002	65.769	2.1022	0.0001	90.8624
1.6288	0.0007	45.4451	1.8687	0.0002	66.416	2.1085	0.0001	91.6364
1.6351	0.0007	45.9526	1.875	0.0002	66.99	2.1148	0.0001	92.0267
1.6414	0.0006	46.4462	1.8813	0.0002	67.6793	2.1211	0.0001	93.0351
1.6478	0.0006	46.9627	1.8876	0.0002	68.1907	2.1274	0.0001	93.4891
1.6541	0.0006	47.4606	1.8939	0.0002	68.8472	2.1337	0.0001	94.3442
1.6604	0.0006	47.9512	1.9002	0.0002	69.5684	2.14	0.0001	94.7804
1.6667	0.0006	48.4592	1.9065	0.0002	70.1646	2.1463	0.0001	95.9366
1.673	0.0005	49.0061	1.9128	0.0002	70.836	2.1527	0.0001	96.8302

ω	Γ	ϕ	ω	Γ	ϕ
2.159	0.0001	97.4482	2.3988	0	126.8652
2.1653	0.0001	97.8753	2.4051	0	128.429
2.1716	0.0001	98.9213	2.4114	0	128.6548
2.1779	0.0001	99.5715	2.4177	0	129.8596
2.1842	0.0001	100.4229	2.4241	0	130.4772
2.1905	0.0001	101.165	2.4304	0	131.4195
2.1968	0.0001	101.7972	2.4367	0	132.0587
2.2032	0.0001	102.6183	2.443	0	132.6356
2.2095	0.0001	103.3237	2.4493	0	133.9373
2.2158	0.0001	104.0371	2.4556	0	134.8715
2.2221	0.0001	104.9279	2.4619	0	135.5872
2.2284	0.0001	105.4146	2.4682	0	136.3218
2.2347	0.0001	106.4893	2.4745	0	137.178
2.241	0.0001	107.0218	2.4808	0	138.3713
2.2473	0.0001	108.1602	2.4872	0	138.6782
2.2536	0.0001	108.884	2.4935	0	139.88
2.26	0.0001	109.6022	2.4998	0	141.3747
2.2663	0.0001	110.2289	2.5061	0	141.7056
2.2726	0.0001	111.1854	2.5124	0	142.3253
2.2789	0.0001	112.0914	2.5187	0	143.6906
2.2852	0.0001	112.809	2.525	0	144.0796
2.2915	0.0001	113.4282			
2.2978	0.0001	114.5414			
2.3041	0.0001	115.5488			
2.3105	0.0001	116.2418			
2.3168	0.0001	116.601			
2.3231	0.0001	117.3532			
2.3294	0.0001	118.0722			
2.3357	0.0001	119.1078			
2.342	0	119.9322			
2.3483	0	120.7954			
2.3546	0	121.2256			
2.3609	0	122.3803			
2.3672	0	122.6994			
2.3736	0	123.7263			
2.3799	0	125.0194			
2.3862	0	125.8216			
2.3925	0	126.2476			

Table B.2: Data for the excitation force F_{d_2}

Appendix C

Code hand book

C.1 Introduction

The next sections introduce a guide to the use of the developed code. The reading of this manual should always be accompanied by the theory in chapter 3. The manual will cover the required stages to setup and run the different modules of the code. This excerpt was only tested with *Microsoft Windows* and *Linux/Unix* (using WSL – Windows Subsystem Linux feature of *Windows*); nevertheless, *Python* should interpret equally on any other operating system (OS).

C.2 Setup

As referred before, the routine was developed in *Python*. *Python* is an interpreted programming language, so the first requirements is to have *python3* installed. After that, it is also necessary to have the modules: *numpy*, *mpmath*, *scypy*, sympy, *pylab* and, optionally, *pymp* (only available on *Linux/Unix* OS). An alternative to the individual installation of these is the installation of a *Python* distribution platform, such as *Anaconda*.

It is also advisable to have a *python* editor as some of the features that can be changed are embedded on the code.

If the studied system is complex and the calculations are slow, consider using the *PyMP* module, and for that (if in *Windows*) use WSL or install a *Linux/Unix* SO, equipped with *openMP*, for parallel computing.

All the *python* packages are available with *pip* or *conda*; WSL is an open source feature of windows, available in the store (for free) and openMP should be in the list of installations of linux as ”sudo apt-get install libomp-dev”.

C.3 Overview

The algorithm is divided into three main modules:

- Elements.py – Module responsible for the elements and element list
- System.py – Module responsible for the system definitions and their simulation
- PMPsolver.py – Module for the optimization of the problem

The remaining files are auxiliary, either for calculations or for visualization/inspection tools.

- Graphics.py – Functions to make graphs of several types
- LobattoQuad.py – Gauss-Lobatto quadrature points and weights
- optimization_mp.py – Modified *scypy* module to work with *mpmath*
- linesearch_mp.py – Modified *scypy* module to work with *mpmath*

C.4 *Elements* module: useful definitions and methods

In this sections some of the useful definitions and methods present on the *Elements* module are discussed. There are more methods in this module, but some of those are mostly useful for the use of other modules, and not to the user directly. For more information on these, see the code and the commentary.

This module is used to create elements and lists of elements. For this, the **Element** and the **Elements** classes are defined here. The remaining functions present in this file are used to create **Elements** objects (lin_elements_dt, lin_elements_nelem, elements_nodes, elements_nodes, load) in different ways. There is also a method to calculate the complete integral (along all the elements) of any value dependent on the elements and time.

C.4.1 Element

Each element object contains the following information:

- Element.n – Degree of approximation n_p
- Element.np1 – $n_\mathrm{p} + 1$
- Element.dt – Time interval of the element Δt^e
- Element.ti – Initial time of the element t^e_st
- Element.tf – Final time of the element t^e_f (derived from the previous)
- Element.n_variables – Number of variables the element approximates (example for a PMP problem: $2n_\mathrm{x} + n_\mathrm{u} - n_\mathrm{x}$ for state variables, n_x for co-state variables and n_u for the control variables)
- Element.constants – Matrix with all the $c^e_{i,j}$ of the approximated variables. Each row is a different variable and each column a different polynomial degree

Element.assign

Still within the element definition, the Element.assign method may prove helpful to assign a values to a specific variable or all variables.

Inputs:

- constants_array – Array of the constants to assign to the specific variables
- variable_numbers – Correspondence of the variables to which assign the previous constants (the first variable is number 0)

Example: if the element approximates 3 variables, assign a constant value 2 to the first and the third variable (assuming $n_p = 2$).

```
>> example_element.assign([2,0,0,2,0,0],[0,2])
```

Element.calculate

Calculates the specific value of a variable in a given point.

Inputs:

- tau – Local time [-1,1]
- variable_numbers – Variable correspondence in the element to calculate (the first variable is number 0)

Example: if the element approximates 3 variables, get the value of the second variable at the middle of the element (τ = 0).

```
>> example_element.calculate(0, (1,) )
```

C.4.2 Elements

The Elements object contains the following attributes:

- Elements.n – n_p
- Elements.np1 – $n_p + 1$
- Elements.Element_list – List of element objects
- Elements.Node_list – List of time nodes (boundary of elements in Element_list)
- Elements.nelem – Number of elements in Element_list
- Elements.n_variables – Number of variables the element approximates (example for a PMP problem: $2n_x + n_u - n_x$ for state variables n_x, for co-state variables and n_u for the control variables)

Elements.save

This method, as the name suggests, saves the all the elements. It is done by writing the data to a separate text file. Keep in mind, if the file is already filled, the information will be erased.

Inputs:

- file – Name of the file (example: 'savefile.txt')

Example: for elements of degree $n_p = 2$, Elements.n_variables $= 3$ and 3 elements has the formatting in table C.1

Element list with nvars=3	and	n=2							
element ti= t_{st}^0, dt= Δt^0:	$c_{1,0}^0$;	$c_{1,1}^0$;	$c_{1,2}^0$;	$c_{2,0}^0$;	$c_{2,1}^0$;	$c_{2,2}^0$;	$c_{3,0}^0$;	$c_{3,1}^0$;	$c_{3,2}^0$;
element ti= t_{st}^1, dt= Δt^1:	$c_{1,0}^1$;	$c_{1,1}^1$;	$c_{1,2}^1$;	$c_{2,0}^1$;	$c_{2,1}^1$;	$c_{2,2}^1$;	$c_{3,0}^1$;	$c_{3,1}^1$;	$c_{3,2}^1$;
element ti= t_{st}^2, dt= Δt^2:	$c_{1,0}^2$;	$c_{1,1}^2$;	$c_{1,2}^2$;	$c_{2,0}^2$;	$c_{2,1}^2$;	$c_{2,2}^2$;	$c_{3,0}^2$;	$c_{3,1}^2$;	$c_{3,2}^2$;

Table C.1: data file formatting

Elements.calculate

This method can be used to calculate specific variables at a given time instant. It works similarly to Element.calculate method, but incorporates a search to find the element for the given time.

Inputs:

- time – Time point to calculate [0, T]
- variable_numbers – Variable correspondence in the element to calculate (the first variable is number 0)

Example: if the element approximates 3 variables, get the value of the second variable at 22 s.

```
>> example_elements.calculate(22, (1,) )
```

C.4.3 calculate_function

Calculates the integral or average of a given functions along all the elements.

Inputs:

- elements – **Elements** object
- var_num_inputs – Correspondence of the variables that input the function
- lambda_functions – List of lambda (*python* anonymous funtions) functions which take the exact correspondence of var_num_inputs to give the function value

- time (optional) – Set to ”True” if any of the functions is time dependents
- average (optional) – Set to ”False” if the desired outhput is the integral instead of the average
- n_int (optional) – Number of Gauss-Lobatto integration points (the default is 3)

Example: Calculate the average of $(x_1 - x_2)^2$. In the elements, x_1 is the first variable and x_2 the second.

```
>> f = lambda x1, x2: (x1 - x2)**2
>> average = calculate_function(example_elements, (0, 1), [f])
```

C.4.4 Elements creation methods

The rest of the ”Elements.py” file contains functions to create **Elements** objects.

lin_elements_dt

Creates equal elements using a constant Δt^e. If T is not a multiple of Δt^e the last element will be truncated.

Inputs:

- delta_t – The constant Δt^e
- T – Final time of the last element
- n – n_p
- nvars – Number of variables the element approximates (example for a PMP problem: $2n_\mathrm{x} + n_\mathrm{u} - n_\mathrm{x}$ for state variables n_x, for co-state variables and n_u for the control variables)

lin_elements_nelem

Creates equal elements using a constant Δt^e by specifying the number of elements.

Inputs:

- n_elem – Number of elements
- T – Final time of the last element
- n – n_p
- nvars – Number of variables the element approximates (example for a PMP problem: $2n_\mathrm{x} + n_\mathrm{u} - n_\mathrm{x}$ for state variables n_x, for co-state variables and n_u for the control variables)

elements_nodes

Creates elements with nodes at specified points. It assumes the first node in the list is 0 and the last is T.

Inputs:

- node_list – List of custom nodes (example: [0, 0.2, 1, 2])
- n – n_p
- nvars – Number of variables the element approximates (example for a PMP problem: $2n_x + n_u - n_x$ for state variables n_x, for co-state variables and n_u for the control variables)

elements_dt

Creates elements with specified Δt^e.

Inputs:

- dt_list – List of custom nodes (example: [0.2, 0.4, 0.2])
- n – n_p
- nvars – Number of variables the element approximates (example for a PMP problem: $2n_x + n_u - n_x$ for state variables n_x, for co-state variables and n_u for the control variables)

load

Recreates the elements using previous saved files (see table C.1).

Inputs:

- file – File with the formatting of C.1 (example: 'savefile.txt')

C.5 System module: useful definitions and methods

The System module is responsible for system definitions and simulation (forward or backward). This module defines only one class: "System". The creation of a system object requires several parameters. These are placed as arguments of System.

This module is completely independent from the PMPsolver module, as it can be used to simply simulate a system. It requires the Element module, with a valid "Elements" object to run a simulation. The results of the simulation are stored in the "Elements" object.

The creation of the **System** may take some time, as it transforms the f function into anonymous, and readies the integrals to be made.

C.5.1 System creation

The system object becomes fully created after its command ”system = System(...)” but it requires a series of inputs:

- xvars – Input variables. List of symbolic variables (sympy.Symbol())
- uvars – Output variables. List of symbolic variables (sympy.Symbol())
- n – n_p
- Ass (only if linear) – State space matrix **A**
- Bss (only if linear) – State space matrix **B**
- state_function (only if nonlinear) – State function **f**. Array of symbolic functions
- xcorr (optional, but recommended) – Correspondence of the index of xvars to the variables index in the elements. By default is set to be xcorr = (0, 1, ..., n_x)
- ucorr (optional, but recommended) – Correspondence of the index of uvars to the variables index in the elements. By default is set to be ucorr = (n_x, n_x+1, ..., n_x+n_u)
- linear (recommended) – Set it to ”True” if the system is linear (don’t forget the state functions change to matrices). By default is ”False”
- forward (optional) – Set it to ”False” if the simulation is supposed to be done backwards in time. By default is ”True”
- n_int (optional but recommended) – Number of Gauss-Lobatto integration points
- multiprocessing (optional) – Set to ”True” to simulate with multiprocessing capabilities. It is required *openMP* and *PyMP*
- tolerance (optional) – Safety parameter for nonlinear simulations. If the relative change between iterations is smaller than the tolerance value, then finish simulating the element

---

Example: create the system $\dot{\mathbf{x}} = (x_2^3, u)$:

```
>> xVariables = [sympy.Symbol('x1'), sympy.Symbol('x2')]
>> uVariables = [sympy.Symbol('u')]
>> f = [xVariables[1]**3, uVariables[0] ]
>> sys = System(xVariables, uVariables, state_function = f, n = 2, xcorr = (0,1), ucorr = (2,), n_int = 4)
   # assuming there is an Elements object where x_1, x_2 and u correspond to the first, second and
   #third variables respectively
```

System.simulate_el

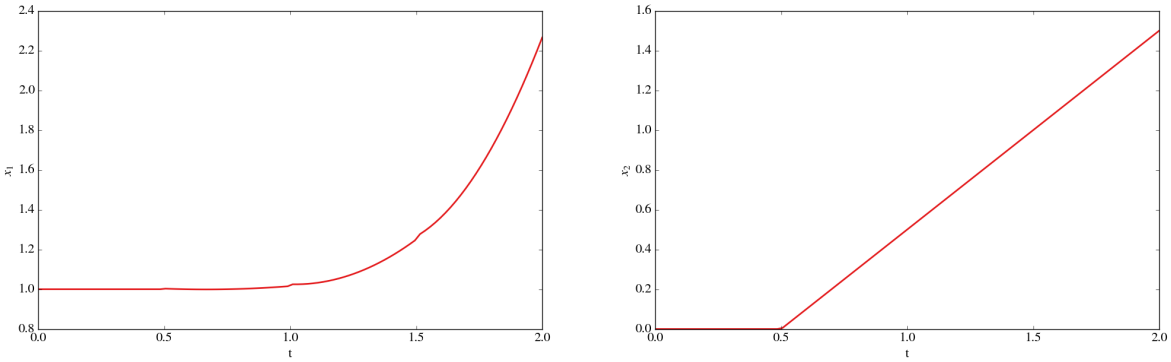
After the system creation, it is now possible to simulate it. This method performs the simulation for each element.

Inputs:

- element – **Element** object to simulate
- Boundary_condition – List containing the boundary conditions (initial or final value) of the **Element** object
- max_it (optional and only if nonlinear) – Limit of iterations to which apply the fixed point method. Default is set to 10

Example: simulate the previous system $\left(\text{sys: } \dot{\mathbf{x}} = (x_2^3, u)\right)$ with the boundary condition $\mathbf{x}(t = 0) = (1, 0)$ and $T = 2$. u is a unit step on $t = 0.5$.

```
>> # Sys is already created from previous example. Create Elements with 4 elements and n_p = 2
   #nodes: 0 - 0.5 - 1 - 1.5 - 2

>> elements_example = lin_elments_nelem(nelem = 4, T=2, n=2, nvars = 3)

>> for i in range(1, len(elements_exapmle.Element_list)): # build the step

>>     # assign to the third variable a constant 1 value, only for elements 1, 2 and 3 (ti = 0.5, 1 and 1.5)

>>     elements_example[i].assign(mpmath.matrix([1,0,0]), (2,))

>> bc = [1, 0]

>> for el in elements_example.Element_list:

>>     Sys.Simulate_el(el,bc)

>>     bc = el.calculate(1) #change boundary condition for the next element

>> Graphics.Plot_variables(example_elements,(0, 1)) # graphical function to be discussed later

>> Graphics.show()
```



(a) x_1 (b) x_2

Figure C.1: system simulation example output

C.6 PMPsolver module: useful methods

This module is responsible for the optimization of control problems with fixed terminal time and with boundary conditions at initial time. It requires all the previous modules to work (*Elements* and *System*).

It is defined a single class here: **PMPsolver**. The creation of a object from this class implies the calculation of an Hamiltionain function, the creation of the λ system (**System** object) and therefore the preparation of its simulation and finally it will prepare for the optimization of **u** be it continuous or "on/off".

C.6.1 PMPsolver creation

The **PMPsolver** object becomes fully created after its command "solver = PMPsolver(...)" but it requires a series of inputs:

- functional – Symbolic function (sympy) with the function to maximize (if minimization use -functionl)
- system – **System** object, imposed as a functional restriction
- T – Terminal time t_f
- n – n_p
- n_int – number of Gauss-Lobatto integration points
- cx (optional) – Cost associated with final states, used for λ_f boundary condition
- nelem (optional if there is already a **Elements** object) – To create a new **Elements** object
- delta_t (optional if there is already a **Elements** object) – To create a new **Elements** object
- xBC – Boundary condition of **x**. Input in the form os list. If not given the progam will assume $\mathbf{x}(t = 0) = \mathbf{0}$
- u_init – Initial value of **u** (constant). If not given, the itearions will start with $u = \mathbf{0}$

- ctrl_type – Control type: ”continuous” or ”on-off”
- save_file (optional) – Name of the file to save which each full iteration (through all elements), the **Elements** object will be saved. See Elements.save above. if not given the solution will not be saved between iterations
- start_from_file (optional) – To load an appropriate **Elements** object for the simulation
- display (optional) – ”Full” for all the information about the calculations to appear on screen, ”None” to show nothing. By deafault is set to ”Full”

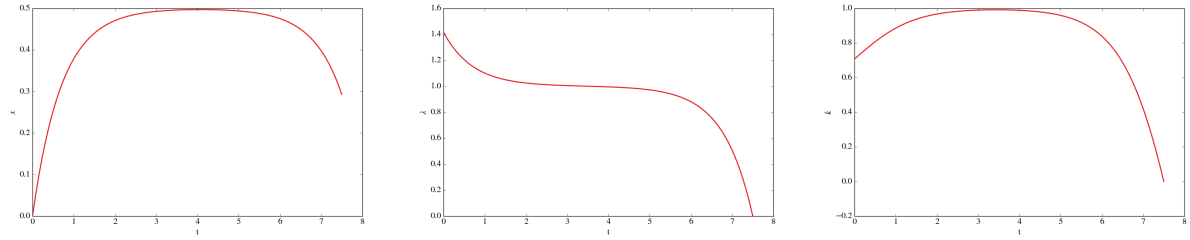
C.6.2 PMPsolver.solve()

This method will proceed to the simulation and optimization of the problem.

Example: solve the maximization problem $J = -x^2 - (k(1-x))^2$ subjected to $\dot{x} = -x + k(1-x)$ with $T = 7.5$. Calculate the average J

```
>> xVariables = [sympy.Symbol('x')]
>> uVariables = [sympy.Symbol('k')]
>> f = [-xVariables[0] + uVariables[0]*(1 - xVariables[0])]
>> J = -(xVariables[0]-1)**2 - (uVariables[0]*(1 - xVariables[0]))**2
>> Jl = sympy.lambdify( [[xVariables[0], uVariables[0]]], J, modules = ”mpmath”) #anonymous function to calculate the average
>> Sys = System(xVariables, uVariables, n = 3, state_function=f, n_int = 5)
>> solver = PMPsolver(J, Sys, 7.5, 3, 5, nelem = 40, xBC = [0], u_init = [0], save_file= ”example.txt”)
>> solver.solve()
>> print(calculate_function(solver.elements, (0,2), Jl)) #calculates the average of J
>> Graphics.Plot_Variables(solver.elements, (0, 1, 2), names = ('x', '\lambda', 'k'))
>> Graphics.show()
```

output: [-0.5666652958070061...]



(a) x_1 (b) λ (c) k

Figure C.2: System optimization example output

C.7 Graphics module

This is a secondary module to plot quantities. Its main methods are Plot_functions and Plot_variables. This module definition is separate from the System and PMPsolver modules, to keep Independence from them. it is possible to plot and analyze simulations and optimizations, without going through the definition of the system or the PMP problem.

C.7.1 Plot_variables

Plots the specified variables of an **Elements** object.

Inputs:

- elements – **Elements** object with the relevant data
- var_numbers – Tuple or list of the variables to plot
- nb_points (optional) – Number of time points to calculate
- names (optional) – List of names to each plotted variable (needs to be the same size as var_numbers)
- label (optional) – It is possible to plot in the same axis with a second call to the Plot_variables function (variables with the same name). This adds a legend to each plot of the same variable. If the legend is required, call it after all the plots are done for each figure

C.7.2 Plot_function

Plots specific custom functions of the **Elements** object and time.

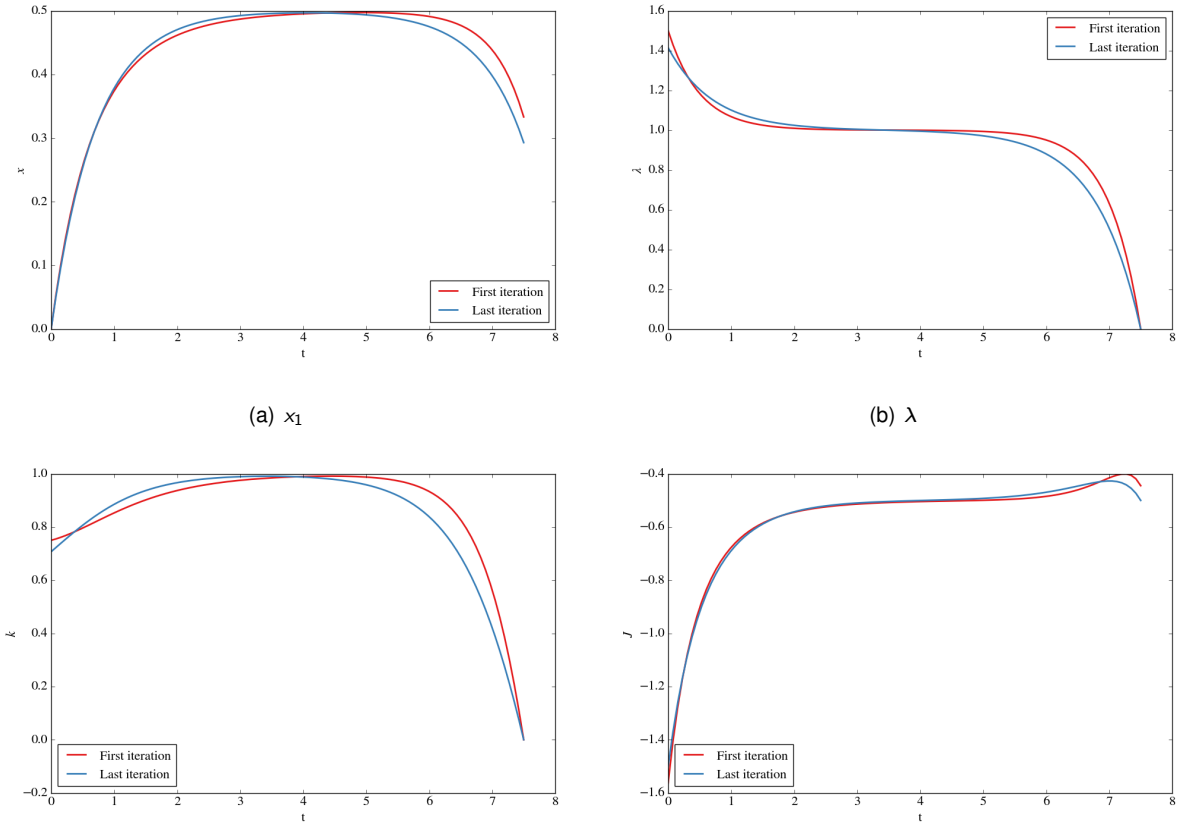
Inputs:

- elements – **Elements** object with the relevant data
- var_num_inputs – Each function to calculate need to have the same number of inputs and in the same order. This input specifies the order of the inputs
- lambda_functions – List of Functions to plot, in the form of lambda functions, whose inputs correspond to the var_num_inputs in the **Elements** object

- nb_points – number of points to evaluate
- names – name of the custom functions (needs to be the same size as lambda_functions)
- label (optional) – It is possible to plot in the same axis with a second call to the Plot_variables function (variables with the same name). This adds a legend to each plot of the same variable. If the legend is required, call it after all the plots are done for each figure
- time – Set to True if any of the functions is time dependent. If that is the case, the time argument should be the last argument of the lambda function. (example_fun = lambda vars,time: ...)

Example for Plot_variables and Plot_function: from the previous optimization, plot x, λ, k and J for the first iteration and the last iteration.

```
>> first = Load("control_it0_example.txt")
>> last = Load("control_it9_example.txt")
>> Graphics.Plot_Variables(first, (0, 1, 2), names = ('x', '\lambda', 'k'), label="First iteration")
>> Graphics.Plot_Variables(last, (0, 1, 2), names = ('x', '\lambda', 'k'), label="Last iteration")
>> #Jl is created in last example
>> Graphics.Plot_function(first, (0,2), Jl, names = ['J'], label = "First iteration")
>> Graphics.Plot_function(last, (0,2), Jl, names = ['J'], label = "Last iteration")
>> for num in Graphics.get_fignums():
>>     Graphics.figure(num)
>>     Graphics.legend(loc="best")
>> Graphics.show()
```



(a) x_1 (b) λ

(c) k (d) J

Figure C.3: Graphics example output

C.8 Convergence and stopping criteria

There are various locations on which it is possible to change stopping criteria and the speed of the convergence. Now it will be explained how to do this. Acknowledge first that there are several cycles that my need stopping criteria:

- Nonlinear simulation;
- Control and simulation;
- Main cycle

The different possibilities of criteria were not explored with this work, but they can be easily added if required.

C.8.1 Nonlinear simulation stopping criteria and convergence

The simulation of nonlinear systems is done recurring to the fixed point method. This method has some known ways of improving the speed of convergence, these also went beyond the scope of this

work, but may be added latter. It was implemented one off these, as an experiment, using a relaxation factor such that:

$$\mathbf{c}^{e^{it=i}} = (\omega - 1)\mathbf{c}^{e^{it=i-1}} + \omega \mathbf{g}\left(\mathbf{c}^{e^{it=i-1}}\right) \tag{C.1}$$

This factor ω is by default set to 1 (the regular fixed point method). It is possible to change it in a **PMPsolver** object as the .state_relaxation attribute (choose a value between 0 and 1)

Also, in the System module it is possible to change the tolerance factor for nonlinear systems, which, as explained before, is referent to the minimum change to continue the iterations.

Another way to stop the nonlinear simulations is to change the max_it parameter in the simulate method.

C.8.2 Continuous control and simulation

Unlike the on-off control, which is computed with the best of two scenarios, the continuous control offers a lot more flexibility. For this case, several optimizations and simulations may be needed to reach a convergence. Up to this date, the only one stopping criteria was added for this: maximum number of iterations. To change this, do it directly in the optimize_e_simulate method of **PMPsolver** and change the number in the ”while” cycle.

It can also be reasonable to add a tolerance for these (similarly to the nonlinear simulation), but this wasn't yet done since it would possibly slow down the calculations.

C.8.3 Main cycle

The main cycle is in the **PMPsolver** .solve() method, it provides the basis to go from the optimization and state simulation to the co-state calculations. The stop to this cycle is also being done in the function by the number of iterations. To change this number, do it in the said function and change the number in the ”while” cycle.

For this it is difficult to have a good measure of convergence, but something else may be added latter on.