UNIVERSIDADE FEDERAL DE SANTA CATARINA
NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Marco Aurélio Schmitz de Aguiar

Distributed Optimal Control of DAE Systems: Modeling, Algorithms, and Applications

Florianópolis
2022

Marco Aurélio Schmitz de Aguiar

Distributed Optimal Control of DAE Systems: Modeling, Algorithms, and Applications

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemasda Universidade Federal de Santa Catarina (UFSC) e pela Norwegian University of Science and Technology (NTNU) em regime de cotutela para a obtenção do título de Doutor em Engenheiro de Automação e Sistemas(UFSC) e Doutor em Engineering Cybernetics (NTNU).
Orientadores: Prof. Eduardo Camponogara, Ph.D. (UFSC) e Prof. Morten Hovd, Ph.D. (NTNU)

Florianópolis
2022

Marco Aurélio Schmitz de Aguiar

Distributed Optimal Control of DAE Systems: Modeling, Algorithms, and Applications

Thesis submitted Programa de Pós-Graduação em Engenharia de Automação e Sistemasof the Fedeal University of Santa Catarina (UFSC) and Norwegian University of Science and Technology (NTNU) in co-tutelle for obtaining the title of Doctor in Automation and Systems Engineering (UFSC) and Philosophiae Doctor in Engineering Cybernetics (NTNU).
Advisors: Prof. Eduardo Camponogara, Ph.D. (UFSC) and Prof. Morten Hovd, Ph.D. (NTNU)

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Aguiar, Marco Aurelio Schmitz de
Distributed Optimal Control of DAE Systems : Modeling, Algorithms, and Applications / Marco Aurelio Schmitz de Aguiar ; orientador, Eduardo Camponogara, coorientador, Morten Hovd, 2022.
168 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Sistemas algébrico-diferenciais. 3. Sistemas não-lineares. 4. Controle ótimo. 5. Algoritmos de Otimização. I. Camponogara, Eduardo. II. Hovd, Morten. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Marco Aurélio Schmitz de Aguiar

Distributed Optimal Control of DAE Systems: Modeling, Algorithms, and Applications

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Dominique Bonvin, Dr.
EPFL/Suiça

Mario Cesar Mello Massa de Campos, Dr.
SmartAutomation/Brasil

Profa. Cristina Stoica Maniu, Dra.
Centrale Supélec/França

Prof. Lars Imsland, Dr.
NTNU/Noruega

Prof. Lars Imsland, Dr.
NTNU/Noruega

Prof. Daniel Ferreira Coutinho, Dr.
UFSC/Brasil

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Doutor em Engenheiro de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Werner Kraus Junior
Data: 21/03/2022 08:46:22-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de Pós-Graduação

ICP-Edu
Documento assinado digitalmente
Eduardo Camponogara
Data: 20/03/2022 21:22:16-0300
CPF: 513.988.180-20
Verifique as assinaturas em https://v.ufsc.br

Eduardo Camponogara, Ph.D.
Orientador

Florianópolis, 2022.

RESUMO

Os sistemas não lineares em rede consistem em vários subsistemas que interagem entre si. Como os sistemas do mundo real raramente são isolados, os sistemas em rede representam uma parte considerável das aplicações de controle. Esta tese contribui para o campo do controle ótimo distribuído de sistemas em rede ao propor um framework para modelagem, formalização e solução dos problemas de controle ótimo (OCP, do inglês optimal control problems). O framework proposto baseia-se no método Lagrangiano aumentado para controle ótimo de equações algébrico diferenciais (DAEs, do inglês differential-algebraic equations), para o qual esta tese apresentou condições para convergência global, local e subótima. A estrutura inclui uma estratégia de modelagem que usa um grafo direcionado (directed graph) para representar os subsistemas da rede. Cada nó representa um subsistema, enquanto uma aresta modela a relação de entrada-saída entre dois subsistemas. Esta descrição do sistema baseada em componentes permite um desenvolvimento e manutenção mais fáceis dos modelos. A formulação de restrições de custo acoplado desacoplado (DCCC, do inglês decoupled cost coupled constraints) para o OCP de tais sistemas em rede é obtida seguindo um conjunto de diretrizes. Ao relaxar essas restrições, o método Lagrangiano aumentado converte as formulações DCCC em uma formulação de restrição desacoplada de custo acoplado (CCDC, do inglês coupled cost decoupled constraint) para o controle ideal de DAEs. Para resolver tais formulações CCDC, esta tese desenvolve algoritmos distribuídos baseados na descida coordenada (coordinate descent) e no método do multiplicador com direção alternada (ADMM, do inglês alternating direction multiplier method), que são adequados para esta formulação. Um artifício de modelagem foi proposto para permitir interações paralelas de nós conectados, que de outra forma teriam que iterar serialmente quando usando esses algoritmos. Experimentos computacionais realizados em um sistema benchmark mostraram resultados promissores, motivando novas investigações sobre o comportamento dos algoritmos no controle de modelos dinâmicos não lineares de aplicações reais. Especificamente, experimentos numéricos com uma rede de produção de petróleo representativa mostraram que os algoritmos podem controlar os sistemas de forma adequada, apesar das não linearidades e descontinuidades.

Palavras-chave: Sistemas algébrico-diferenciais. Sistemas não-lineares. Controle Ótimo. Algoritmos de Otimização.

RESUMO EXPANDIDO

INTRODUÇÃO

Os sistemas não lineares em rede consistem em vários subsistemas que interagem entre si. Como os sistemas do mundo real raramente são isolados, os sistemas em rede representam uma parte considerável das aplicações de controle. Portanto existe um grande beneficio em desenvolver metodologias que aproveitem a sua estrutura disitribuida para alcançarem soluções eficientes, robustas e escláveis.

OBJETIVOS

Esta tese busca contribuir para o campo do controle ótimo distribuído de sistemas em rede ao propor um framework para modelagem, formalização e solução dos problemas de controle ótimo (OCP, do inglês optimal control problems). Uma das propriedades esperadas desse framework é a facil modelagem e manutenção de modelos de controle. Em sistemas de larga escala, se os modelos não são estruturados apropriadamente os modelos podem acabar crescendo sem respeitar os limites de cada subsistema, o que aumenta a complexidade do modelo. Ao mesmo tempo, busca-se um formalismo para a espeficicação dos OCPs que permita obte-los de forma simples e pragmática. Um padronização nos OCPs permite uma comunicação facil de estrategias e abordagens de solução com a comunidade cientica. Por fim, esse framework precisa permitir que algoritmos de controle ótimo resolvam os OCPs de forma eficiente e acertiva.

METODOLOGIA

O primeiro passo para o desenvolvimento do framework é reforçar as propriedades do método Lagrangiano aumentado para controle ótimo de equações algébrico diferenciais (DAEs, do inglês differential-algebraic equations). Para este método algumas propriedades matemáticas são elaboradas sendo elas: condições para convergência para um mínimo global do algoritmo, condições para convergência para um mínimo local, condições para convergência atravéz atravéz de uma sequencia de soluções sub-ótimas para o problema auxiliar. A estratégia de modelagem proposta então é desenvolvida usando sistemas algébrico-diferenciais e grafo direcionados (directed graph) para representar os subsistemas da rede. No grafo, cada nó representa um subsistema, enquanto cada aresta modela a relação de entrada-saída entre dois subsistemas. Esta descrição do sistema baseada em componentes permite um desenvolvimento e manutenção mais fáceis dos modelos. A formulação de restrições de custo acoplado desacoplado (DCCC, do inglês decoupled cost coupled constraints) para o OCP de tais sistemas em rede é obtida seguindo um conjunto de diretrizes. Ao relaxar essas restrições, o método Lagrangiano aumentado converte as formulações DCCC em uma formulação de restrição desacoplada de custo acoplado (CCDC, do inglês coupled cost decoupled constraint) para o controle ideal de DAEs. Para resolver tais formulações CCDC, esta tese desenvolve algoritmos distribuídos baseados na descida coordenada (coordinate descent) e no método do multiplicador com direção alternada (ADMM, do inglês alternating direction multiplier method), que são adequados para esta formulação. Um artifício de modelagem foi proposto para permitir interações paralelas de nós conectados, que de outra forma teriam que iterar serialmente quando usando esses algoritmos.

RESULTADOS E DISCUSSÃO

Experimentos computacionais realizados em um sistema benchmark mostraram resultados promissores, motivando novas investigações sobre o comportamento dos algoritmos no controle de modelos dinâmicos não lineares de aplicações reais. Especificamente, experimentos numéricos com uma rede de produção de petróleo representativa mostraram que os algoritmos podem controlar os sistemas de forma adequada, apesar das não linearidades e descontinuidades.

CONSIDERAÇÕES FINAIS

Os experimentos apresentaram resultados inspiradores que incentivão novas pesquisas e mostram que tanto o framework proposto quanto as estratégias de solução empregadas atingiram os objetivos esperados. Na conclusão da tese possiveis continuações para a pesquisa são apresentadas, incluindo estudos com controlo ótimo estocástico, validação das estrategias empregadas em caso ainda mais realistas e elaborações de propriedades matemáticas das técnicas de controle desenvolvidas com através do framework.

Palavras-chave: Sistemas algébrico-diferenciais. Sistemas não-lineares. Controle Ótimo. Algorítmos de Otimzação

ABSTRACT

Networked nonlinear systems consist of several subsystems that interact with one another. Since real-world systems are seldom isolated, networked systems represent a considerable portion of the control applications. This thesis contributes to the field of distributed optimal control of networked systems by proposing a framework for modeling, formalizing, and solving the optimal control problems (OCP). The proposed framework relies on the augmented Lagrangian method for optimal control of differential-algebraic equations (DAEs), for which this thesis presented conditions for global, local, and suboptimal convergence. The framework includes a modeling strategy that uses a directed graph to represent the network subsystems. Each node represents a subsystem, while an edge models the input-output relation between two subsystems. This component-based description of the system allows for easier development and maintenance of the models. Following a set of guidelines, decoupled cost coupled constraints (DCCC) formulation for the OCP of such networked systems is obtained. Bhy relaxing these constraints, the augmented Lagrangian method converts the DCCC formulations into a coupled cost decoupled constraint (CCDC) formulation for optimal control of DAEs. To solve such CCDC formulations, this thesis develops distributed algorithms based on the coordinate descent and the alternating direction multiplier method (ADMM), which are well-suited for this formulation. A modeling artifice was proposed to enable parallel iterations of connected nodes, which otherwise would have to iterate serially according to these algorithms. Computational experiments performed in a benchmark system showed promising results, motivating further investigation of the behavior of the algorithms in the control of nonlinear dynamic models of real applications. Specifically, numerical experiments with a representative oil production network showed that the algorithms could control the systems properly, despite the nonlinearities and discontinuities.

Keywords: Differential-algebraic systems, nonlinear systems, optimal control, optimization algorithms

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

A great range of industrial processes shows nonlinear behavior. Most commonly, they are connected to other systems with their own dynamics. Such systems are classified as networked nonlinear systems, which are a representative class of practical systems. Networked systems are usually large and complex, which complicates control design, particularly if one desires optimal control actions. For this reason, we look for optimal control techniques that exploit the system structure enabling efficient and structured optimal control.

This thesis seeks to develop a framework for distributed optimal control of networked dynamic systems. One of the desired properties of such a framework is easy modeling and model maintenance. With large systems, if the models are not appropriately structured, the designer may end up building models with variables and equations disregarding the boundaries of the subsystems, which adds complexity. This practice causes a big issue when new equipment is added to the system, or a model of a particular subsystem needs to be changed. Likewise, a well-defined methodology for specifying the optimal control problem (OCP) from the ground up is desired for effectively operating such networked systems. The formalization of a structure allows for promptly finding in the literature strategies to solve the OCP. While having well-structured models and OCP is a great start, the final goal of the optimal control is to obtain optimal control actions – and for that, we need algorithms. Therefore a framework should provide means for algorithms to solve the problems efficiently and reliably to yield the optimal control actions.

1.2 CONTRIBUTIONS

This thesis contributes to the areas of optimal control and distributed optimal control, particularly for the control of systems described by continuous-time differential-algebraic equations (DAE); it does this with two significant contributions.

Strengthening the augmented Lagrangian method for Optimal Control (AGUIAR, 2016)

In (AGUIAR, 2016), an augmented Lagrangian method for optimal control of continuous-time DAEs was proposed. The algorithm is inspired by the augmented Lagrangian method for solving constrained optimization problems. It works by relaxing the algebraic equations of the OCP of DAEs, effectively transforming it into an OCP of ordinary differential equations (ODEs). The relaxed equation is put in the integral term of the objective, with a quadratic penalization term and an inner product with a multiplier estimate. This thesis formalizes the work-in-progress algorithm presented in

the master's dissertation and develops proofs of convergence, under varying conditions, that led to a paper published in a reputable journal of the field (AGUIAR et al., 2021). Two numerical experiments were performed to validate the proposed method.

Proposition of a framework for Distributed Optimal Control

The proposed framework models the system with a directed graph. The system is assumed to be composed of many subsystems, each with its own dynamics, states, inputs, and outputs variables. The inputs are the variables that are not defined by the given subsystem, while the outputs are the variables on which other subsystems depend. Each subsystem is represented as a node in the graph. The input-output relation between the nodes are directed edges in the graph, mathematically, they are trivial linear equations. This allows for a component-based description of the system so that, if any changes are needed, the system can be easily reconfigured locally without central intervention.

The specification of the optimal control problem also has some rules. In addition to the DAE equations for networked systems described in terms of directed graphs, the objective is assumed to be separable for each subsystem. A separable objective means that the objective of each subsystem does not depend on other subsystem variables directly. This objective leads to a decoupled cost coupled constraints (DCCC) formulation, the coupling constraint being the connection between the nodes. An OCP with a DCCC formulation has interesting properties which can be exploited. The restriction of only separable objectives seems extreme, but a few modeling tricks are shown to allow a vast group of objectives to be modeled with such conditions.

The proposed modeling structure allows for the augmented Lagrangian method for optimal control of DAE systems to be used to relax the algebraic equations that connect the subsystem. Since the coupling equations are relaxed and put into the objective, the optimal control problem becomes a coupled cost decoupled constraint (CCDC) problem. A CCDC optimization problem can be solved with several strategies, notably coordinate descent and alternating direction, to name a few. An optimal control interpretation of these strategies is developed. One downside of these strategies is that they typically do not allow for nodes that have a direct connection to be solved simultaneously; to circumvent this limitation, a modeling artifice is used to enable all nodes to be solved in parallel. Numerical experiments performed with the methods in a benchmark DAE system showed promising results. The promising results on the benchmark system led to a more sophisticated experiment with a small-scale oil production network, which is full of nonlinearities and other real-world challenges.

1.3 PUBLICATIONS

The author was involved in the following publications, which are directly related to this thesis

- AGUIAR, Marco Aurelio; CAMPONOGARA, Eduardo; FOSS, Bjarne. An Augmented Lagrangian for Optimal Control of DAE Systems: Algorithm and Properties. IEEE Transactions on Automatic Control, v. 66, p. 261–266, 1 Jan. 2021. DOI: 10.1109/TAC.2020.2976042

- KRISHNAMOORTHY, Dinesh; AGUIAR, Marco Aurelio; FOSS, Bjarne; SKOGESTAD, Sigurd. A Distributed Optimization Strategy for Large Scale Oil and Gas Production Systems. In: 2018 IEEE Conference on Control Technology and Applications (CCTA). IEEE, Aug. 2018. P. 521–526. DOI: 10.1109/CCTA.2018.8511385

- CAMPONOGARA, Eduardo; SILVA, Ricardo da; AGUIAR, Marco Aurelio. A distributed dual algorithm for distributed MPC with application to urban traffic control. In: 2017 IEEE Conference on Control Technology and Applications (CCTA). IEEE, Aug. 2017b. P. 1704–1709. DOI: 10.1109/CCTA.2017.8062702

- AGUIAR, Marco Aurelio; CAMPONOGARA, Eduardo; FOSS, Bjarne. An augmented Lagrangian method for optimal control of continuous time DAE systems. In: 2016 IEEE Conference on Control Applications (CCA). 2016. P. 1185–1190. DOI: 10.1109/CCA.2016.7587967

The following publications are not directly connected to distributed optimal control, however they are tangential the themes of this thesis

- JORDANOU, Jean P.; CAMPONOGARA, Eduardo; ANTONELO, Eric; AGUIAR, Marco Aurelio. Nonlinear Model Predictive Control of an Oil Well with Echo State Networks. IFAC-PapersOnLine, v. 51, n. 8, p. 13–18, 2018. DOI: 10.1016/j.ifacol.2018.06.348

- JORDANOU, Jean; ANTONELO, Eric; CAMPONOGARA, Eduardo; AGUIAR, Marco Aurelio. Recurrent Neural Network Based Control of an Oil Well. In: XIII Simpósio Brasileiro de Automação Inteligente. 2017

1.4 ORGANIZATION

This thesis is organized as follows:

- Chapter 2 begins with a review of concepts required for better following this dissertation: DAEs, OCPs, necessary conditions for OCPs, Pontryagin's minimum principle, necessary conditions for OCPs of DAEs, solution methods for OCPs,

and augmented Lagrangian method for optimization. The augmented Lagrangian method for optimal control of DAEs is presented. In the following, a series of essential proofs for this method are provided. The chapter ends with two numerical experiments illustrating the applicability of this method.

- Chapter 3 goes through a technical review of networked dynamic systems and distributed optimal control. An overview of some optimization techniques are provided, namely coordinate descent, augmented Lagrangian with coordinate descent, and alternating direction multiplier method (ADMM). Based on the relaxation provided by the augmented Lagrangian for optimal control, a version of the optimization methods is presented for distributed optimal control. A reformulation that leads to the full parallelization of the nodes is presented. To finalize the chapter, all the proposed methods are assessed using a benchmark plant.
- Chapter 4 provides a more realistic assessment of the distributed algorithm for optimal control of networked systems. For this, a network with two oil and gas production wells and a vertical riser is modeled based on models available in the literature. The numerical experiments setup is then provided, along with the indicators chosen for the analysis. The analysis of the results is provided.
- Chapter 5 gives the final thoughts of this thesis along with some possibilities for future work.

2 AUGMENTED LAGRANGIAN FOR OPTIMAL CONTROL PROBLEMS OF DAE

2.1 BACKGROUND

2.1.1 Differential Algebraic Equations

The usual mathematical tool for modeling dynamic systems is ordinary differential equations (ODEs), which have the form

$$\dot{x} = f(x, u, t) \tag{1}$$

where $x(t) \in \mathbb{R}^{N_x}$ is the state variable, $u \in \mathbb{R}^{N_u}$ is the control variable, $t \in [t_0, t_f]$ is the time variable, and f is the dynamic function. This class of equations allows obtaining a compact model, which benefits from the many mathematical and numerical tools for analysis and solution. As these models grow in size and complexity, they become hard to interpret and, therefore, to maintain.

An approach that has increased in popularity in recent years is combining algebraic equations with ODEs to keep the inherent structure of the physical process. This combination is known as differential-algebraic equations (DAEs), in which variables in the system of equations are categorized in two groups: the state variables, those that are defined by the differential equations, and the algebraic variables, those that are defined by the algebraic equations. The typical approach of obtaining an ODE often starts with a DAE, which is then simplified by eliminating the algebraic equations. Essentially, the same process is performed to obtain a DAE, but the simplification step is skipped, and the responsibility of dealing with a more complex set of equations is attributed to the numerical solvers.

A DAE can be transformed into an ODE via two main methods: substitution and differentiation. Those modeling an ODE system often use the substitution method without being aware that they first formulate a DAE system. The technique works by simply isolating the algebraic variables and substituting them in the differential equations with the resulting expression. Consider the following DAE system,

$$\dot{x} = -x + y + u \tag{2a}$$

$$0 = y - x^2 + 1 \tag{2b}$$

one could simply eliminate the y variable by obtaining $y = x^2 - 1$, and replacing y in the differential equation to obtain $\dot{x} = -x + x^2 - 1 + u$. However, through this process, information is lost. The information in y could be meaningful for the user of the model; it could provide a hint of the process behavior or be required to be bounded due to the very nature of the modeled system.

Differentiation, the second method to obtain an ODE of a DAE, uses the differentiation of the algebraic equation with respect to the time variable to transform the

algebraic equation into a differentiation. In the example (2) by differentiating the algebraic equation, the differential equation $\dot{y} = 2x\dot{x} = 2x(-x + y + u)$ is obtained, therefore making an ODE with two states. This method does not come without its hindrances: the obtained state requires an initial condition.

For more complex models, a single differentiation may not result in an ODE right away, requiring subsequent differentiations until no algebraic equation is left. This process of differentiating DAEs is known as index reduction. The index of a DAE system is the number of differentiations needed to convert the DAE into an ODE system. For instance, an ODE is an index-0 DAE because the system itself is an ODE. An equation like (2) is an index-1 DAE. The pendulum system can be modeled as index-3 (ASCHER; PETZOLD, 1998). Ultimately, the index of a DAE can be used as a measure of complexity.

In this work we are particularly interested in semi-explicit index-1 DAE, the class of DAE with the following form

$$\dot{x} = f(x, y, u, t) \tag{3a}$$

$$0 = g(x, y, u, t) \tag{3b}$$

where $y(t) \in \mathbb{R}^{N_y}$ is the algebraic variable, and $g(x, y, u, t) \in \mathbb{R}^{N_y}$ is the algebraic function, and (3b) is solvable w.r.t. y, equivalently, the Jacobian of g w.r.t. exists and is invertible. This particular kind of DAE was chosen because of its mathematical properties, which will be later employed to identify mathematical properties and design an OCP algorithm for DAEs. Although they exist, higher-order DAEs appear less often in the modeling of control applications. When they do, they are typically transformed into a lower index DAE by applying index reduction.

The differences between the two approaches are highlighted in the following example.

Example 1 (A simple tank: ODE vs. DAE)**.** *To demonstrate how one can approach obtaining a DAE model, a tank that holds liquid is studied in this example. The illustration of the tank is depicted in Figure 1.*

The mass balance is typically the first thing to address when doing phenomenological modeling. Let us assume that the liquid has constant density, therefore a volume balance can be equivalently applied. Hence, the variation of volume of liquid that the tank holds is the inflow decreased by the outflow

$$\dot{V} = q_{in} - q_{out} \tag{4}$$

where V is the volume of liquid, q_{in} is the tank feed, and q_{out} is the outflow.

Knowing the volume, the tank level can be obtained through the following equation

$$h = \frac{V}{A} \tag{5}$$

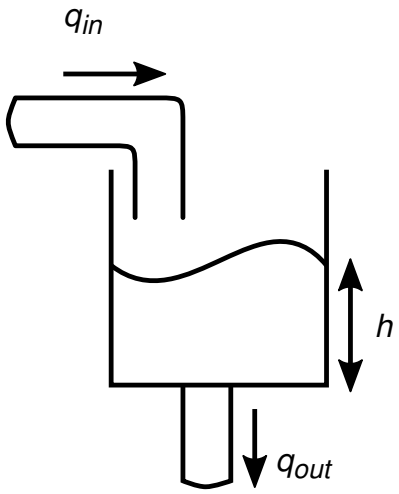


Figure 1 – Illustration of the tank

where the A is the cross section area of the tank.

Considering that the inflow q_{in} is given, the only unknown is the outflow, which can obtained using the orifice equation

$$q_{out} = a\sqrt{2gh} \tag{6}$$

where a is the cross section area of the hole, g is gravitational acceleration, and h is the height of the liquid inside of the tank.

Grouping the equations above, the DAE model is obtained

$$\dot{V} = q_{in} - q_{out} \tag{7a}$$

$$h = \frac{V}{A} \tag{7b}$$

$$q_{out} = a\sqrt{2gh} \tag{7c}$$

Assuming that one is interested in the model for the tank height (h), then the ODE model could be achieved by the substitution method that yields the following equation

$$\dot{h} = \frac{q_{in} - a\sqrt{2gh}}{A} \tag{8}$$

As expected due to the substitutions, ODE models are typically more compact, whereas DAE models are richer in information. This compactness of the model not always translates into a faster and more stable solution by numerical methods. The sparsity provided by the DAE can aid numerical algorithms by avoiding sharp derivatives.

The modularity of the DAE can be exploited to compose models with higher complexity while preserving meaning and maintainability. For instance, modeling of electrical submersible pumps (ESPs) and their interactions with gas and oil production

wells can be achieved while keeping meaning and understanding of the process (KRISHNAMOORTHY et al., 2016). For this reason, DAEs are an excellent tool, not only for centralized process control but for distributed control as well.

2.1.2 Optimal Control Problems for ODEs

In the previous subsection, we saw how to obtain a mathematical model that represents a dynamic system. Here we discuss how to make the system behave as we desire, particularly how to do so optimally. Clearly, if a controller (and system behavior) is *optimal* there can be no controller giving better system behavior, and there has to exist a criterion for making such comparisons.

Optimal control is a field of control theory that combines optimization and control to obtain the control actions that are the best for some given criterion. This criterion, which is commonly referred to as the objective function, is crucial to optimization. One can only say that A is better than B if there is a comparison metric. The objective function is typically expressed as the sum of an integrated cost and a final cost. The integrated cost is the combination of costs for the actions, penalties, and systems' behavior incurred during the prediction window. In contrast, the final cost is the cost resulting from the conditions at the final time.

$$\min_{u} \quad J(x, y, u) = \int_{t_0}^{t_f} L(x, y, u, t)\, dt + V(x(t_f), t_f) \tag{9}$$

where L is the dynamic cost function (also know as the Lagrange term), and V is the final cost function (Mayer term). The functions L and V are expected to be continuously differentiable w.r.t. to their arguments. Under some basic assumptions, an objective function described in terms of the dynamic cost L can be expressed purely in terms of the final cost V, with the opposite also holding, as shown in the following theorem.

Theorem 1. *(KIRK, 2004) Given an OCP with the following objective*

$$\min_{u} \quad V(x(t_f), t_f) + \int_{t_0}^{t_f} L(x, u, t)\, dt, \tag{10}$$

and $V(x(t_f), t_f)$ is continuously differentiable in x and t, then there is an equivalent objective that only has the Lagrangian term,

$$\min_{u} \quad \int_{t_0}^{t_f} \widehat{L}(x, u, t)\, dt. \tag{11}$$

Conversely, it is possible to find a representation of (10) *that has only the Mayer term,*

$$\min_{u} \quad \widehat{V}(x(t_f), t_f). \tag{12}$$

Proof. Starting with the identity

$$V(x(t_f), t_f) = V(x(t_0), t_0) + V(x(t_f), t_f) - V(x(t_0), t_0) \tag{13}$$

which can be developed into

$$V(x(t_f), t_f) = V(x(t_0), t_0) + \int_{t_0}^{t_f} \frac{dV}{dt}(x, t)\, dt \tag{14a}$$

$$= V(x(t_0)) + \int_{t_0}^{t_f} \left[\frac{\partial V}{\partial t}(x, t) + \frac{\partial V}{\partial x}(x, t) \frac{dx}{dt} \right] dt \tag{14b}$$

$$= V(x(t_0)) + \int_{t_0}^{t_f} \left[\frac{\partial V}{\partial t}(x, t) + \frac{\partial V}{\partial x}(x, t) f(x, u, t) \right] dt. \tag{14c}$$

Since $x(t_0)$ and t_0 are fixed, $V(x(t_0), t_0)$ is fixed by consequence. Therefore they do not affect the solution and can be disregarded. By choosing $\widehat{L}$ in the form

$$\widehat{L}(x, u, t) = L(x, u, t) + L_V(x, u, t). \tag{15}$$

where

$$L_V = \frac{\partial V}{\partial t}(x, t) + \frac{\partial V}{\partial x}(x, t) f(x, u, t) \tag{16}$$

the minimization of the integral of $\widehat{L}$ will be equivalent to (10).

For the second statement, let us introduce a new state x_L such that

$$\dot{x}_L = L(x, u, t), \tag{17}$$

with the initial condition being $x_L(t_0) = 0$. Therefore we can define a new final cost

$$\widehat{V}(x(t_f), t_f) = V(x(t_f), t_f) + \int_{t_0}^{t_f} L(x, u, t)\, dt \tag{18a}$$

$$= V(x(t_f), t_f) + \int_{t_0}^{t_f} \dot{x}_L(t)\, dt \tag{18b}$$

$$= V(x(t_f), t_f) + x_L(t_f). \tag{18c}$$

□

A common choice for the objective is to use a quadratic error between a desired target and the state variable,

$$\min_u \quad J(x, y, u) = \int_{t_0}^{t_f} (x_d - x)^T (x_d - x) + (u_d - u)^T (u_d - u)\, dt \tag{19}$$

where x_d and u_d are the desired state and control, respectively.

However, more generally, one could choose an economic cost that will lead to the optimum operation of processes of interest. For instance, one might choose to

maximize the oil produced by oil wells or minimize the time a rocket takes to reach a given altitude.

The optimal control problem of an ODE system is composed of an objective function and an ODE; it can also contain constraints on control and state variables, but a simple OCP is investigated as a start. Putting all together, the following problem is obtained

$$\min_{u,t_f} \quad J(x,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f} L(x,u,t)\,dt \tag{20a}$$

$$\text{s.t.:} \quad \dot{x} = f(x,u,t) \tag{20b}$$

$$x(t_0) = x_0 \tag{20c}$$

where x_0 is the initial condition for the state variable. (20)

A few assumptions on its underlying variables and functions are necessary for problem (20) to be well defined.

Assumption 1. *With respect to problem* (20)*, the following assumptions are made:*

1. *The function V is continuously differentiable with respect to x and t.*
2. *The function L is continuously differentiable with respect to x, u, and t.*
3. *The dynamics function f is continuously differentiable with respect to x, u, and t.*
4. *The function x is continuously differentiable with respect to t, and the function u is piecewise continuous with respect to t.*

If the third assumption holds and u is piecewise continuous, then the solution of the ODE system (20b) *exists and is unique (KHALIL, 2002, Theorem 3.2).*

A pair of state and control profiles x^* and u^* can only be said to be a solution to the optimization problem (20) if it satisfies the conditions given in the following theorem.

Theorem 2. *(KIRK, 2004; SASANE, 2004) Let* (20) *be an OCP for which Assumption 1 holds. If the control profile u^* defined in the interval $[t_0, t_f]$, which induces the state profile x^*, is a minimum for* (20) *then there exists a function $\lambda^* : [t_0, t_f] \to \mathbb{R}^{N_x} \in \mathcal{C}^1[t_0, t_f]$ that satisfies*

$$-\dot{\lambda}^* = \frac{\partial L}{\partial x}(x^*,u^*,t)^T + \frac{\partial f}{\partial x}(x^*,u^*,t)^T\lambda^*, \qquad t \in \left[t_0, t_f\right] \tag{21a}$$

$$0 = \frac{\partial L}{\partial u}(x^*,u^*,t) + \lambda^{*T}\frac{\partial f}{\partial u}(x^*,u^*,t) \qquad t \in \left[t_0, t_f\right], \tag{21b}$$

$$\lambda^*(t_f) = \frac{\partial V}{\partial x}(x(t_f),t_f) \tag{21c}$$

and x^* is given by

$$\dot{x}^* = f(x^*,u^*,t), \qquad t \in [t_0, t_f] \tag{21d}$$

$$x^*(t_0) = x_0. \tag{21e}$$

Proof. In optimization theory, the Lagrangian is a function that augments the objective function by incorporating the constraints multiplied by a vector, known as Lagrange multipliers. Likewise, we can define an equivalent augmented functional that combines the final cost, the dynamic cost function, and the system dynamic equation adjointed by a variable $\lambda(t) \in \mathcal{C}^1[t_0, t_f]$, as follows

$$J_a(x, u, \lambda) = V(x(t_f), t_f) + \int_{t_0}^{t_f} \left\{ L(x, u, t) + \lambda^T \left[f(x, u, t) - \dot{x} \right] \right\} dt \tag{22}$$

where λ can be referred to as the adjoint variable or the costate. Notice that if x^* satisfies the ODE equation $\dot{x}^* = f(x^*, u^*, t)$, then $J_a = J$ (20a).

From Theorem 10 (in Appendix A) if (x^*, u^*, λ^*) are an extremum of J_a, then the first variation of the augmented functional J_a has to be zero.

Let δx with $\delta x(t_0) = 0$ be the variation of the state x, δu be variation of u, the $\delta\lambda$ be the variation of the adjoint variable λ, and δt_f be a variation on the final time t_f. The first variation of J_a, using Theorem 12 (in Appendix A), at x^*, λ^*, and u^* is given by

$$\begin{aligned}
\delta J_a = \left[\frac{\partial V}{\partial x}(x^*(t_f), t_f) \right] \delta x(t_f) + \Bigg\{ & \frac{\partial V}{\partial t}(x^*(t_f), t_f) \\
& + \frac{\partial V}{\partial x}(x^*(t_f), t_f)\dot{x}^*(t_f) + L(x^*(t_f), u^*(t_f), t_f) \\
& + \lambda^{*T} \left[f(x^*(t_f), u^*(t_f), t_f) - \dot{x}^*(t_f) \right] \Bigg\} \delta t_f \\
+ \int_{t_0}^{t_f} \Bigg\{ & \left[\frac{\partial L}{\partial x}(x^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial x}(x^*, u^*, t) \right] \delta x \\
& + \left[\frac{\partial L}{\partial u}(x^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial u}(x^*, u^*, t) \right] \delta u \\
& + \left[f(x^*, u^*, t) - \dot{x}^* \right]^T \delta\lambda - \lambda^{*T} \delta\dot{x} \Bigg\} dt = 0
\end{aligned} \tag{23}$$

where $\delta\dot{x}$ is the derivative of the perturbation δx, and $\delta x(t_f)$ is the value of the perturbation δx at time t_f. The terms $L(x^*(t_f), u^*(t_f), t_f) + \lambda^{*T} \left[f(x^*(t_f), u^*(t_f), t_f) - \dot{x}^*(t_f) \right]$ (multiplied by δt_f) arise from the differentiation of the integral with respect to t_f, and the terms related to $V(x(t_f), t_f)$ originate from the first order Taylor's expansion

$$\delta V(x(t_f), t_f) = \left[\frac{\partial V}{\partial t}(x^*(t_f), t_f) + \frac{\partial V}{\partial x}(x^*(t_f), t_f)\dot{x} \right] \delta t_f + \frac{\partial V}{\partial x}(x^*(t_f), t_f)\delta x(t_f). \tag{24}$$

Using integration by parts, the term related to $\delta\dot{x}$ can be transformed

$$\int_{t_0}^{t_f} -\lambda^{*T} \delta\dot{x}\, dt = -\lambda^{*T}(t_f)\delta x(t_f) + \int_{t_0}^{t_f} \dot{\lambda}^{*T} \delta x\, dt \tag{25}$$

where the term $\lambda^{*T}(t_0)\delta x(t_0)$ vanishes since $\delta x(t_0) = 0$.

The term $\delta x(t_f)$ is a perturbation on the variable $x^*(t_f)$ at the time $t = t_f$. Further, let us recall that δt_f is a perturbation at the point $t = t_f$ in the time axis. Let us define

the perturbation δx_f as a perturbation on the final value of the trajectory of x^*, that is x_f. Notice that $\delta x(t_f) = \delta x_f$ only if t_f is the final time, but if the final time is perturbed then the final time is $t_f + \delta t_f$ and the final state is $x_f = x(t_f + \delta t_f)$.

Observing Figure 2, it can be noticed that by perturbing the value of $x(t)$ at $t = t_f$, that is $x(t_f)$, the value of the end of the trajectory x_f is directly affected. In addition, note that by perturbing the final time t_f, the end of trajectory (x_f) is also affected depending on the inclination of trajectory, given by $\dot{x}$.

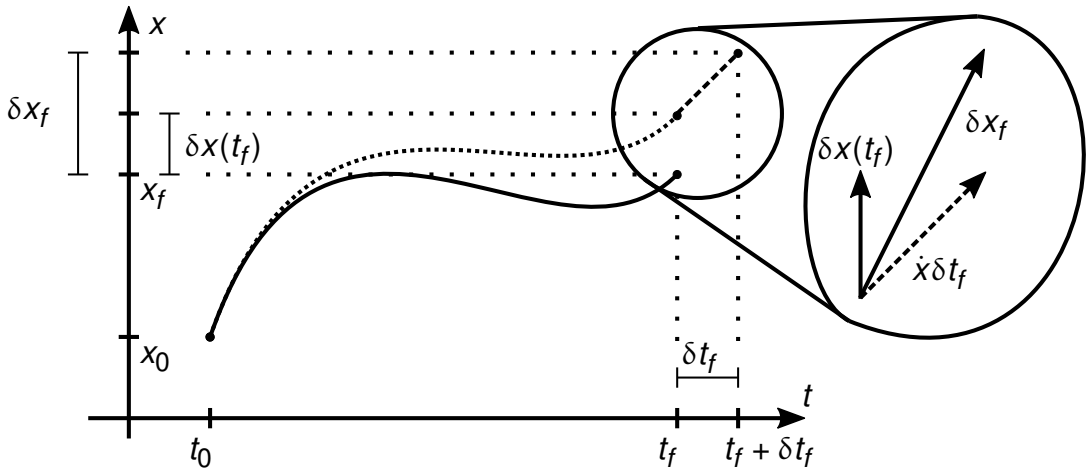


Figure 2 – Illustrations on the variations δx_f, $\delta x(t_f)$, and δt_f. The solid line is the original trajectory and dotted line is the perturbed trajectory

As observed in Figure 2, we have the following relation

$$\delta x_f = \delta x(t_f) + \dot{x}\delta t_f \tag{26}$$

which can be rearranged in the form

$$\delta x(t_f) = \delta x_f - \dot{x}\delta t_f \tag{27}$$

Using equations (25), (27), and the fact that $\dot{x}(t_f) = f(x(t_f), u(t_f), t_f)$, the first variation of the augmented functional can be rewritten as

$$\begin{aligned}
\delta J_a = \Bigg[& L(x^*(t_f), u^*(t_f), t_f) + \lambda^*(t_f)^T f(x^*(t_f), u^*(t_f), t_f) \\
& + \frac{\partial V}{\partial t}(x^*(t_f), t_f) \Bigg] \delta t_f + \left[\frac{\partial V}{\partial x}(x^*(t_f), t_f) - \lambda^*(t_f) \right] \delta x_f \\
& + \int_{t_0}^{t_f} \Bigg\{ \left[\frac{\partial L}{\partial x}(x^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial x}(x^*, u^*, t) + \dot{\lambda}^{*T} \right] \delta x \\
& + \left[\frac{\partial L}{\partial u}(x^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial u}(x^*, u^*, t) \right] \delta u \\
& + \left[f(x^*, u^*, t) - \dot{x}^* \right]^T \delta\lambda \Bigg\} \, dt = 0
\end{aligned} \tag{28}$$

Since in order to the trajectories (x^*, u^*, t) be an extremum, it is required that $\delta J_a = 0$, the Fundamental Lemma of variational calculus (Lemma 2, in Appendix A) can be applied to the integral term in (28) obtaining the necessary conditions

$$-\dot{\lambda}^* = \frac{\partial L}{\partial x}(x^*, u^*, t)^T + \frac{\partial f}{\partial x}(x^*, u^*, t)^T \lambda^* \tag{29a}$$

$$0 = \frac{\partial L}{\partial u}(x^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial u}(x^*, u^*, t) \tag{29b}$$

$$\dot{x}^* = f(x^*, u^*, t) \tag{29c}$$

for all $t \in [t_0, t_f]$.

By meeting the conditions (29), the first variation of δJ_a becomes

$$\delta J_a = \left[L(x^*(t_f), u^*(t_f), t_f) + \lambda^*(t_f)^T f(x^*(t_f), u^*(t_f), t_f) + \frac{\partial V}{\partial t_f}(x^*(t_f), t_f) \right] \delta t_f + \left[\frac{\partial V}{\partial x}(x^*(t_f), t_f) - \lambda^*(t_f) \right] \delta x_f \tag{30}$$

Under the assumption that the final time is fixed and the final state is free, the final time perturbation is zero ($\delta f_t = 0$). The first term of (30) vanishes, resulting in

$$\delta J_a = \left[\frac{\partial V}{\partial x}(x^*(t_f), t_f) - \lambda^*(t_f) \right] \delta x_f \tag{31}$$

In order to δJ_a to be zero for any pertubation x_f, the following equation must be satisfied

$$\lambda^*(t_f) = \frac{\partial V}{\partial x}(x^*(t_f), t_f) \tag{32}$$

If the final time were considered an optimization variable or the final condition was fixed or constrained, the necessary condition would take another form. The condition for those cases can be found in (KIRK, 2004). □

Sometimes the conditions presented in Theorem 2 are presented in terms of a Hamiltonian system, whose Hamiltonian function is given by

$$H(x, \lambda, u, t) = L(x, u, t) + \lambda^T f(x, u, t) \tag{33}$$

which leads to the necessary conditions to be presented as

$$-\dot{\lambda} = \frac{\partial H}{\partial x} = \frac{\partial L}{\partial x}(x, u, t)^T + \lambda^T \frac{\partial f}{\partial x}(x, u, t) \tag{34a}$$

$$\dot{x} = \frac{\partial H}{\partial \lambda} = f(x, u, t) \tag{34b}$$

$$\frac{\partial H}{\partial u} = \frac{\partial L}{\partial u}(x, u, t)^T + \lambda^T \frac{\partial f}{\partial u}(x, u, t) = 0 \tag{34c}$$

From the necessary conditions defined in terms of the Hamiltonian, we can notice that the optimal control u^* is the one that minimizes the Hamiltonian ($\frac{\partial H}{\partial u} = 0$).

2.1.3 Pontryagin's minimum principle

Theorem 2 assumed that the control was unbounded. Next consider the case where control is bounded, that is

$$u \in U_B = \{\mathrm{u} \in R^{N_u} | u_L \leq \mathrm{u} \leq u_U\} \tag{35}$$

where $u_l \in \mathbb{R}^{N_u}$ is the lower bound, and $\mathbb{R}^{N_u}$ is the upper bound.

The OCP (20) can be augmented to include the control bounds

$$\min_{u,t_f} \quad J(x,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f} L(x,u,t)\,dt \tag{36a}$$

$$\text{s.t.:} \quad \dot{x} = f(x,u,t) \tag{36b}$$

$$x(t_0) = x_0 \tag{36c}$$

$$u \in U_B \tag{36d}$$

If the solution of the OCP (36) has u^* such that $u_L < u(t) < u_U$ for all $t \in [t_0, t_f]$, then solving OCP (36) with or without (36d) has the same solution, and for such condition the optimality conditions ($\frac{\partial H}{\partial u} = 0$) of Theorem 2 are valid.

On the other hand, if for some interval $t \in [t_1, t_2]$ the optimal solution u^* touches the boundary of the set U_B, then it will not necessarily satisfy the optimality condition $\frac{\partial H}{\partial u} = 0$. In this case, a strategy similar to Theorem 2 using the variational calculus can not be used given that the pertubation on u may not lay inside the set of valid controls, that is $u(t) + \delta u(t) \notin U_B$, in particular if $u(t) = u_L$ or $u(t) = u_U$.

Pontryagin's minimum principle establishes a necessary optimal condition that accounts for constraints in the control variable. An intuitive demonstration will be given below, while a more detailed proof can be found in (PONTRYAGIN et al., 1962). If $u^* \in U_B$ induces an optimal value in the objective $J(u)$, assuming that the state equations are respected, then

$$\Delta J(u) = J(u) - J(u^*) \geq 0 \tag{37}$$

for all valid u in the neighborhood of u^*. If we define $u = u^* + \delta u$, the variation in the objective can be reprented by

$$\Delta J(u^*, \delta u) = \delta J(u^*, \delta u) + \text{higher-order variations} \tag{38}$$

If we express the variation δJ in terms of the Hamiltonian we get

$$\begin{aligned}\delta J = &\left[H(x^*(t_f),\lambda^*(t_f),u^*(t_f),t_f) + \frac{\partial V}{\partial t_f}(x^*(t_f),t_f)\right]\delta t_f + \left[\frac{\partial V}{\partial x}(x^*(t_f),t_f) - \lambda^*(t_f)^T\right]\delta x_f \\ &+ \int_{t_0}^{t_f}\left\{\left[\frac{\partial H}{\partial x}(x^*,\lambda^*,u^*,t)\right]\delta x + \left[\frac{\delta H}{\delta u}(x^*,\lambda^*,u^*,t)\right]\delta u \right. \\ &\left. + \left[\frac{\delta H}{\delta \lambda}(x^*,\lambda^*,u^*,t) - \dot{x}^{*T}\right]\delta\lambda\right\} dt \end{aligned} \tag{39}$$

If for all perturbations except δu the equations are set in order to make δJ be zero, then

$$\Delta J = \int_{t_0}^{t_f} \left[\frac{\partial H}{\partial u}(x^*, \lambda^*, u^*, t) \right] dt + \text{higher-order variations} \tag{40}$$

Using the first variation definition,

$$\left[\frac{\partial H}{\partial u}(x^*, \lambda^*, u^*, t) \right] \delta u = H(x^*(t), \lambda^*(t), u^*(t) + \delta u(t), t) - H(x^*(t), \lambda^*(t), u^*(t), t) \tag{41}$$

in the definition of ΔJ,

$$\Delta J = \int_{t_0}^{t_f} \left[H(x^*(t), \lambda^*(t), u^*(t) + \delta u(t), t) - H(x^*(t), \lambda^*(t), u^*(t), t) \right] dt + \text{higher-order variations} \tag{42}$$

By assuming that δu is sufficiently small and that $u^* + \delta u$ satisfies the neighborhood demand, then the integral term of (42) dominates over the higher-order terms. Hence acconding to (37), in order to u^* to be optimal

$$\int_{t_0}^{t_f} \left[H(x^*(t), \lambda^*(t), u^*(t) + \delta u(t), t) - H(x^*(t), \lambda^*(t), u^*(t), t) \right] dt \geq 0 \tag{43}$$

for all admissible δu.

Therefore, we must have

$$H(x^*(t), \lambda^*(t), u^*(t), t) \leq H(x^*(t), \lambda^*(t), u^*(t) + \delta u(t), t) \tag{44}$$

or in terms of u in a neighborhood of u^*,

$$H(x^*(t), \lambda^*(t), u^*(t), t) \leq H(x^*(t), \lambda^*(t), u(t), t) \tag{45}$$

for all $t \in [t_0, t_f]$. This means that the necessary condition to u^* to be optimal is that it minimize the Hamiltonian.

The intuitions introduced above are formalized in the following theorem.

Theorem 3 (Pontryagin's Minimum Principle (PONTRYAGIN et al., 1962))**.** *Let $u^*(t)$ for $t \in [t_0, t_f]$ be an admissible control such that the corresponding trajectory x^* is defined by*

$$\dot{x}^* = \frac{\partial H}{\partial \lambda} = f(x^*, u^*, t) \tag{46}$$

with the boundary condition $x^*(t_0) = x_0$. In order for $u^*(t)$ and $x^*(t)$ to be optimal it is necessary that there exists a nonzero continuous vector function $\lambda^* : [t_0, t_f] \to \mathbb{R}^{N_x}$ corresponding to $u^*(t)$ and $x^*(t)$ through the ODE such that

$$-\dot{\lambda}^* = \frac{\partial H}{\partial x} = \frac{\partial L}{\partial x}(x^*, u^*, t) + \frac{\partial f}{\partial x}(x^*, u^*, t), \tag{47}$$

with the boundary conditions given by the conditions

$$x^*(t_0) = f(x^*, u^*, t) \tag{48a}$$

$$\lambda^*(t_f) = \frac{\partial V}{\partial x}(x^*(t_f), t_f) \tag{48b}$$

and that

1. *The function* $H(x^*(t), \lambda^*(t), \mathrm{u}, t)$*, with* $\mathrm{u} \in U_B$*, attains its minimum at the point* $\mathrm{u} = u^*(t)$ *for all* $t \in [t_0, t_f]$*:*

$$H(x^*(t), \lambda^*(t), u^*(t), t) = \inf_{\mathrm{u} \in U_B} H(x^*(t), \lambda^*(t), \mathrm{u}, t) \tag{49}$$

   or, equivalently, for all $\mathrm{u} \in U_B$ *and* for all $t \in [t_0, t_f]$*:*

$$H(x^*(t), \lambda^*(t), u^*(t), t) \leq H(x^*(t), \lambda^*(t), \mathrm{u}, t). \tag{50}$$

2. *If the final time is fixed and the Hamiltonian does not depend explicitly on the time variable, then the Hamiltonian must be equal to a constant* c_1 *for all* $t \in [t_0, t_f]$*,*

$$H(x^*(t), \lambda^*(t), u^*(t), t) = c_1, \quad \forall t \in [t_0, t_f]. \tag{51}$$

3. *If the final time is free and the Hamiltonian does not depend explicitly on the time variable, then the Hamiltonian must be zero for all* $t \in [t_0, t_f]$*,*

$$H(x^*(t), \lambda^*(t), u^*(t), t) = 0, \quad \forall t \in [t_0, t_f]. \tag{52}$$

Proof. The proof of this Theorem can be found in (PONTRYAGIN et al., 1962, Chapter 2). □

The conditions presented so far are necessary conditions; sufficient conditions have been developed for OCPs of ODEs. These conditions are typically not explored in practice due to the difficulty of extracting control laws out of these conditions. The most well-known theorem among these sufficient conditions is the Hamilton-Jacobi-Bellman (HJB) equations, which can be described as a continuous version of the Bellman equations. The interested reader is referred to (BERTSEKAS, 2005) for further reading.

2.1.4 Optimal Control Problems of DAEs

So far, this chapter has presented the optimality conditions for OCPs of ODEs. However, we are interested in the OCP of DAEs.

Given the definition of a DAE (3) and the definition of an OCP of ODEs (20), let us define an OCP of DAEs

$$\min_{u,t_f} \quad J(x,y,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f} L(x,y,u,t)\,dt \tag{53a}$$

$$\text{s.t.:} \quad \dot{x} = f(x,y,u,t) \tag{53b}$$

$$0 = g(x,y,u,t) \tag{53c}$$

$$x(t_0) = x_0 \tag{53d}$$

where $x(t) \in \mathbb{R}^{N_x}$ is the state, $y(t) \in \mathbb{R}^{N_y}$ is the algebraic variable, $u(t) \in \mathbb{R}^{N_u}$ is the control variable. The function f characterizes the states, and g characterizes the algebraic variables. The final cost is given by the function V, while L defines the dynamic cost.

The OCP of DAEs is also known in the literature as an OCP with mixed constraints. In those works, the dynamic system is described solely using states and controls. The differential and algebraic equations are viewed as equality constraints. The former representation was chosen in this work because of the structure that distinguishes algebraic and states variables.

Similarly to the approach described above for studying the OCP for ODEs, we can associate a multiplier function with the equality (53c). Let $\nu : [t_0, t_f] \to \mathbb{R}^{N_y}$ be the multiplier associated with (53c). Notice since the algebraic equation should hold for all t in $[t_0, t_f]$, ν is a time dependent function.

Theorem 2 can be extended for including algebraic equations. Let the Hamiltonian function be defined by

$$H_{DAE}(x,\lambda,y,\nu,u,t) = L(x,y,u,t) + \lambda^T f(x,y,u,t) + \nu^T g(x,y,u,t) \tag{54}$$

Then the necessary conditions for the functions x^*, y^*, and u^* to be optimal are given in the following theorem.

Theorem 4. *(BIEGLER, L. T., 2010) Consider an OCP of DAE in the form* (53)*. If the control u^*, which induces the states x^* by* (53b)*, and the algebraic variables y^* by* (53c) *in the time interval $t \in [t_0, t_f]$, is a minimum of* (53a)*, then there exists a continuous differentiable function $\lambda^* : [t_0, t_f] \to \mathbb{R}^{N_x}$ and a function $\nu^* : [t_0, t_f] \to \mathbb{R}^{N_y}$, such that*

$$\frac{\partial H_{DAE}}{\partial x} = -\dot{\lambda}^{*T} = \frac{\partial L}{\partial x} + \lambda^{*T}\frac{\partial f}{\partial x} + \nu^{*T}\frac{\partial g}{\partial x} \tag{55a}$$

$$\frac{\partial H_{DAE}}{\partial y} = 0 = \frac{\partial L}{\partial y} + \lambda^{*T}\frac{\partial f}{\partial y} + \nu^{*T}\frac{\partial g}{\partial y} \tag{55b}$$

$$\frac{\partial H_{DAE}}{\partial u} = 0 = \frac{\partial L}{\partial u} + \lambda^{*T}\frac{\partial f}{\partial u} + \nu^{*T}\frac{\partial g}{\partial u} \tag{55c}$$

Proof. The proof follows the same steps of the proof of Theorem 2; therefore, a summary is provided here.

Let us define the an augmented functional J_a, given by

$$J_a(x,\lambda,y,\nu,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f}\left\{L(x,y,u,t)+\lambda^T\left[f(x,y,u,t)-\dot{x}\right]+\nu^T g(x,y,u,t)\right\}dt \quad (56)$$

Let δx be the perturbation on the state, δy be the perturbation on the algebraic variable, and δu be the perturbation on the control variable. Then we can perform the same process used to obtain the first variation of δJ_a for the ODE case (28). By doing so, the first variation of J_a at $(x^*,\lambda^*,y^*,\nu^*,u^*)$ is given by

$$\begin{aligned}\delta J_a = &\left[L(x^*(t_f),y^*(t_f),u^*(t_f),t_f)+\lambda^*(t_f)^T f(x^*(t_f),u^*(t_f),t_f)\right.\\ &\left.+\nu^*(t_f)^T g(x^*(t_f),y^*(t_f),u^*(t_f),t_f)+\frac{\partial V}{\partial t_f}(x^*(t_f),t_f)\right]\delta t_f\\ &+\left[\frac{\partial V}{\partial x}(x^*(t_f),t_f)-\lambda^*(t_f)^T\right]\delta x_f\\ &+\int_{t_0}^{t_f}\left\{\left[\frac{\partial L}{\partial x}+\lambda^{*T}\frac{\partial f}{\partial x}+\nu^{*T}\frac{\partial g}{\partial x}+\dot{\lambda}^{*T}\right]\delta x+\left[\frac{\partial L}{\partial y}+\lambda^{*T}\frac{\partial f}{\partial y}+\nu^{*T}\frac{\partial g}{\partial y}\right]\delta y\right.\\ &+\left[\frac{\partial L}{\partial u}+\lambda^{*T}\frac{\partial f}{\partial u}+\nu^{*T}\frac{\partial g}{\partial u}\right]\delta u+\left[f(x^*,y^*,u^*,t)-\dot{x}^*\right]^T\delta\lambda\\ &\left.+\left[g(x^*,y^*,u^*,t)\right]^T\delta\nu\right\}dt \quad (57)\end{aligned}$$

where the arguments of partial derivatives were suppressed for a better readability.

If $(x^*,\lambda^*,y^*,\nu^*,u^*)$ minimizes (53), then by Theorem 10 (in Appendix A) the first variation has to be zero. By the fundamental lemma of the calculus of variations (Lemma 2 in Appendix A) to J_a to be zero, the following system of equation has to be satisfied for all $t\in[t_0,t_f]$

$$-\dot{\lambda}^{*T} = \frac{\partial L}{\partial x}+\lambda^{*T}\frac{\partial f}{\partial x}+\nu^{*T}\frac{\partial g}{\partial x} \quad (58a)$$

$$0 = \frac{\partial L}{\partial y}+\lambda^{*T}\frac{\partial f}{\partial y}+\nu^{*T}\frac{\partial g}{\partial y} \quad (58b)$$

$$0 = \frac{\partial L}{\partial u}+\lambda^{*T}\frac{\partial f}{\partial u}+\nu^{*T}\frac{\partial g}{\partial u} \quad (58c)$$

$$\dot{x}^* = f(x^*,y^*,u^*,t) \quad (58d)$$

$$0 = g(x^*,y^*,u^*,t) \quad (58e)$$

If we assume that the final time t_f is fixed and the final state x_f is free, the boundary conditions are

$$x(t_0) = x_0 \quad (59a)$$

$$\lambda(t_f) = \frac{\partial V}{\partial x}(x^*(t_f),t_f)^T \quad (59b)$$

□

The OCP (53) can also be extended to include bounds on the controls

$$\min_{u,t_f} \quad J(x,y,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f} L(x,y,u,t)\,dt \tag{60a}$$

$$\text{s.t.:} \quad \dot{x} = f(x,y,u,t) \tag{60b}$$

$$0 = g(x,y,u,t) \tag{60c}$$

$$x(t_0) = x_0 \tag{60d}$$

$$u \in \mathcal{U}_B \tag{60e}$$

The Pontryagin's mininum principle can adjusted to account for the algebraic variables (BIEGLER, L. T., 2010).

Theorem 5. *Let the optimal control $u^*(t)$ satisfy $u^*(t) \in U_B$ for all $t \in [t_0, t_f]$, such that the corresponding trajectory x^* and y^* are defined by*

$$\frac{\partial H_{DAE}}{\partial \lambda}^T = \dot{x}^* = f(x^*,y^*,u^*,t) \tag{61a}$$

$$\frac{\partial H_{DAE}}{\partial \nu}^T = g(x^*,y^*,u^*,t) = 0 \tag{61b}$$

with the boundary condition $x^*(t_0) = x_0$.

If $x^*(t)$, $y^*(t)$, and $u^*(t)$ induce an optimal solution to (60)*, then there exists a nonzero continuous vector function $\lambda^* : [t_0, t_f] \to \mathbb{R}^{N_y}$ and a nonzero functions corresponding to $x^*(t)$, $y^*(t)$, and $u^*(t)$ through the DAE*

$$\frac{\partial H_{DAE}}{\partial x} = -\dot{\lambda}^{*T} = \frac{\partial L}{\partial x}(x^*,y^*,u^*,t) + \lambda^{*T}\frac{\partial f}{\partial x}(x^*,y^*,u^*,t) + \nu^{*T}\frac{\partial g}{\partial x}(x^*,y^*,u^*,t) \tag{62a}$$

$$\frac{\partial H_{DAE}}{\partial y} = \frac{\partial L}{\partial y}(x^*,y^*,u^*,t) + \lambda^{*T}\frac{\partial f}{\partial y}(x^*,y^*,u^*,t) + \nu^{*T}\frac{\partial g}{\partial y}(x^*,y^*,u^*,t) = 0 \tag{62b}$$

with the boundary conditions given by the conditions stated in Theorem 4, and that

1. *The function $H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),\mathrm{u},t)$ with $\mathrm{u} \in U_B$ attains its minimum at the point $\mathrm{u} = u^*(t)$ for all $t \in [t_0, t_f]$:*

$$H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),u^*(t),t) = \inf_{\mathrm{u}\in U_B} H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),\mathrm{u},t) \tag{63}$$

   or, equivalently, for all $\mathrm{u} \in U_B$ and for all $t \in [t_0, t_f]$:

$$H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),u^*(t),t) \leq H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),\mathrm{u},t) \tag{64}$$

2. *If the final time is fixed and the Hamiltonian does not depend explicitly on the time variable, then the Hamiltonian must be equal to a constant c_1 for all $t \in [t_0, t_f]$,*

$$H_{DAE}(x^*(t),\lambda^*(t),y^*(t),\nu^*(t),u^*(t),t) = c_1 \quad \forall t \in [t_0,t_f] \tag{65}$$

3. *If the final time is free and the Hamiltonian does not depend explicitly on the time variable, then the Hamiltonian must be zero for all* $t \in [t_0, t_f]$,

$$H_{DAE}(x^*(t), \lambda^*(t), y^*(t), \nu^*(t), u^*(t), t) = 0 \quad \forall t \in [t_0, t_f] \tag{66}$$

More general necessary conditions have been developed that account for a DAEs with an arbitrary index (CLARKE; PINHO, 2010; DE PINHO et al., 2001; GERDTS, 2006; KUNKEL; MEHRMANN, 2008; BOCCIA et al., 2016). Sufficient conditions for OCP of ODEs have also been developed based on the HJB equations (GALEWSKA; NOWAKOWSKI, 2005).

2.2 SOLUTION METHODS

So far, necessary conditions for OCPs of DAEs with and without control constraints have been provided, but no discussion on obtaining a solution in practice was presented. In this section, we are going to discuss two common approaches to solve the OCPs. These approaches restate the OCP in a form more compatible with optimization tools. The methods can be divided into two (SRINI et al., 2003):

- **Indirect Methods:** The indirect methods use optimality conditions to find a solution for the OCP. They are indirect because instead of using the OCP to find the optimal control, they try to find the solution for a set of equations obtained from the optimality conditions.
- **Direct Methods:** The direct methods try to find the controls that minimize the objective function, typically using optimization methods and solvers.

2.2.1 Indirect Method

As said previously, the methods that use the optimality conditions to obtain an optimal solution are known as indirect methods. In this section, a method that uses the necessary conditions given in Theorem 4 to obtain the optimal solution of an OCP with the form of (53) will be presented. This same approach can be used to solve (60) using the conditions defined in Theorem 5

The method will be presented following an example for better visualization. The Van der Pol oscillator is a common benchmark application (KHALIL, 2002), that is typically modelled as an ODE system of the form

$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u \tag{67a}$$

$$\dot{x}_2 = x_1 \tag{67b}$$

However, this system can be modelled as a DAE system,

$$\dot{x}_1 = y + u \tag{68a}$$
$$\dot{x}_2 = x_1 \tag{68b}$$
$$y = (1 - x_2^2)x_1 - x_2 \tag{68c}$$

Consider the objective function that brings the system to the unstable equilibrium at $x = (0, 0)$,

$$J = \int_{t_0}^{t_f} \left[x_1^2 + x_2^2 + u^2 \right] \, dt \tag{69}$$

Let the initial time t_0 be 0, and the final time t_f be 5 seconds, the initial conditions are $x(t_0) = [0, 1]^T$. Figures 3 and 4 depicts the system without any control law applied. It is possible to verify that the open-loop system goes to a limit cycle dynamic without a control law to stabilize it.

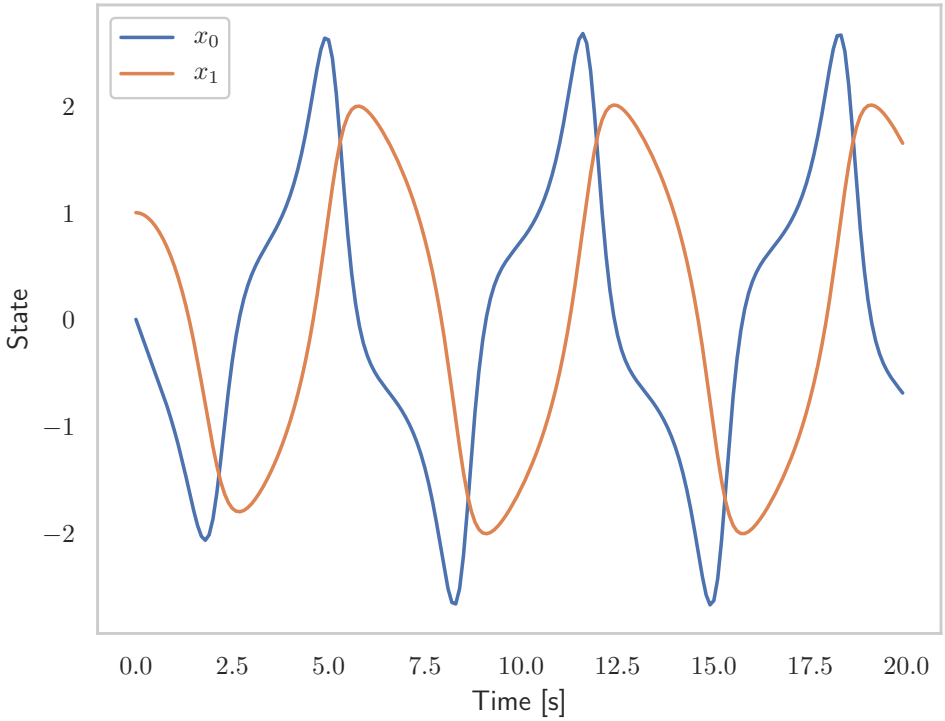


Figure 3 – States of the Van Der Pol oscillator in open loop

With the DAE system (68) and the objective function (69), the following OCP can be created

$$\min J = \int_{t_0}^{t_f} \left[x_1^2 + x_2^2 + u^2 \right] \, dt \tag{70a}$$
$$\text{s.t.: } \dot{x}_1 = y + u \tag{70b}$$
$$\dot{x}_2 = x_1 \tag{70c}$$
$$y = (1 - x_2^2)x_1 - x_2 \tag{70d}$$

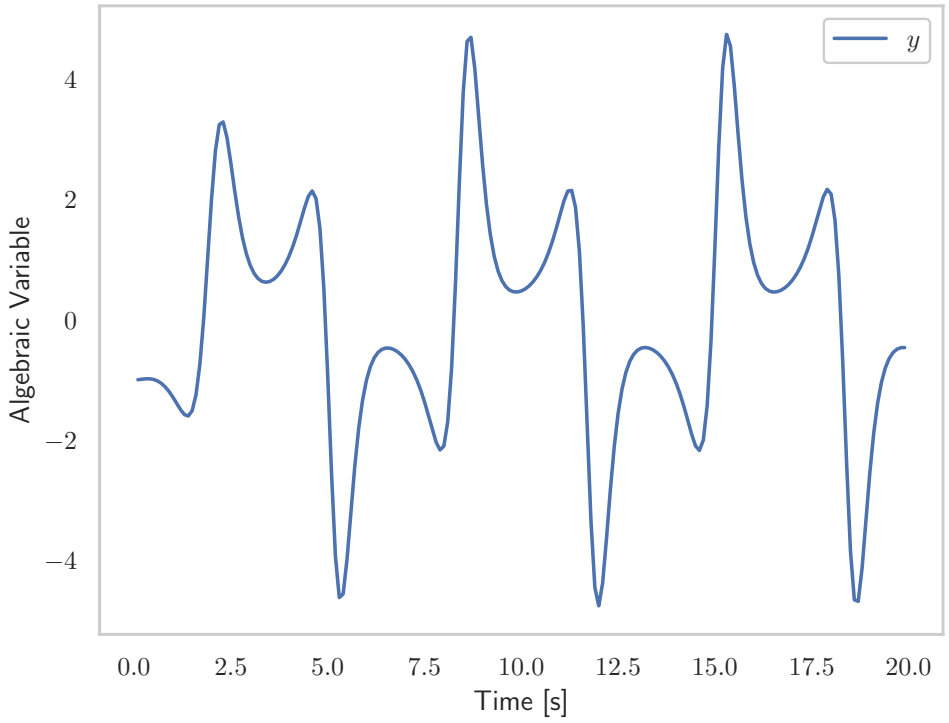


Figure 4 – Algebraic variable of the Van Der Pol oscillator in open loop

If we apply the necessary conditions discussed in Theorem 5 and 4 to OCP (70) then the following boundary value problem (BVP) arises

$$\frac{\partial H}{\partial x_1} = -\dot{\lambda}_1 = 2x_1 + \lambda_2 + \nu(1 - x_2^2) \tag{71a}$$

$$\frac{\partial H}{\partial x_2} = -\dot{\lambda}_2 = 2x_2 + \nu(-2x_2x_1 - 1) \tag{71b}$$

$$\frac{\partial H}{\partial y} = \lambda_1 - \nu = 0 \tag{71c}$$

$$\frac{\partial H}{\partial u} = 2u + \lambda_1 = 0 \tag{71d}$$

$$\frac{\partial H}{\partial \lambda_1} = \dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u \tag{71e}$$

$$\frac{\partial H}{\partial \lambda_2} = \dot{x}_2 = x_1 \tag{71f}$$

$$\frac{\partial H}{\partial \nu} = (1 - x_2^2)x_1 - x_2 - y = 0 \tag{71g}$$

$$x(0) = [1, 0]^T \tag{71h}$$

$$\lambda(t_f) = 0 \tag{71i}$$

From (71d), the optimal control strategy can be extracted

$$u^* = -\frac{\lambda_1}{2} \tag{72}$$

Since there is an equation for the control variable, there are two possibilities:

1. The control variable can be recast as an algebraic variable, with the associated algebraic equation being (72).
2. The substitution method can be applied to make u vanish from the system.

Either way, the resulting set of equations become a BVP. For instance, if the first approach is used the following BVP is obtained

$$-\dot{\lambda}_1 = 2x_1 + \lambda_2 + \nu(1 - x_2^2) \tag{73a}$$
$$-\dot{\lambda}_2 = 2x_2 + \nu(-2x_2x_1 - 1) \tag{73b}$$
$$\lambda_1 - \nu = 0 \tag{73c}$$
$$2u + \lambda_1 = 0 \tag{73d}$$
$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u \tag{73e}$$
$$\dot{x}_2 = x_1 \tag{73f}$$
$$(1 - x_2^2)x_1 - x_2 - y = 0 \tag{73g}$$
$$x(0) = [1, 0]^T \tag{73h}$$
$$\lambda(t_f) = 0 \tag{73i}$$

An off-the-shelf BVP with DAE capabilities solver can solve this BVP. Alternatively, it can be solved using one of the discretization techniques for OCPs that will be later described. Notice that BVPs are OCPs with no objective ($J = 0$) and a mixture of initial and terminal constraints.

Although indirect methods are pretty simple to implement, they have a few disadvantages. There is a stability duality between states and costates; if a state is stable, then its costate is unstable and vice-versa. This issue makes the BVP more challenging to solve due to numerical instability. Furthermore, dealing with general constraints, typical in OCPs, is a complex challenge for indirect methods. The necessary conditions for OCPs with general constraints were not discussed in this document. However, they are pretty intricate to use in practice and have to be hand-tailored to specific applications. For these reasons, a direct approach is often preferred in many industrial applications.

2.2.2 Direct Method

The direct methods try to solve the OCP using optimization techniques. For this reason, the OCP needs to be conditioned to be handled by an optimization solver. The preparation typically takes three steps.

The first step consists of transforming the objective. Optimization solvers are made to minimize objective functions and can not handle the minimization of integrals as is the case of the objective of the OCP (70). Theorem 1 can be used to transform the integral in a final cost. Let us introduce a new state x_c that is defined by

$$\dot{x}_c = L(x, y, u, t) = x_1^2 + x_2^2 + u^2 \tag{74}$$

where $x(t_0) = 0$.

Then the objective can be equivalently expressed as

$$\widehat{J} = x_c(t_f) \tag{75}$$

Since the objective can now be expressed in terms of a scalar value, it can be used as an objective function.

The second step to be able to solve an OCP directly is to parametrize the control variables. The unknown variables of optimization problems can only be scalars and vectors and can not handle arbitrary functions as it is stated in the OCP (70).

A common approach to parametrize the control is to approximate the control function by a piecewise constant function,

$$u(t) = \overline{u}_i, \qquad t \in [t_i, t_{i+1}), i \in \{1, \ldots, N_i\} \tag{76}$$

where $\overline{u}_i \in \mathbb{R}^{N_u}$ are the constant value that the control will be held during the period $[t_i, t_{i+1}]$, and N_i is an arbitrary number of segmentations of the control variable.

Another common approach is to use a piecewise polynomial approximation, for instance

$$u(t) = \overline{u}_{i,a}t^2 + \overline{u}_{i,b}t + \overline{u}_{i,c} \qquad t \in [t_i, t_{i+1}), i \in \{1, \ldots, N_i\} \tag{77}$$

where $\overline{u}_{i,a}$, $\overline{u}_{i,b}$, and $\overline{u}_{i,c}$ for all i are the parameters that charcterize the the polynomial. This polynomial representation has a problem. In order to constraint the controls, the polynomial has to be evaluated. For this reason, Lagrangian polynomials are often preferred. The piecewise polynomial approximation using the Lagrangian polynomial is given by

$$u(t) = \sum_{j=0}^{N_\ell} \ell_j \left(\frac{t - t_i}{t_{i+1} - t_i} \right) \overline{u}_{ij} \qquad t \in [t_i, t_{i+1}), i \in \{1, \ldots, N_i\} \tag{78}$$

where ℓ_j are the Lagrangian polynomial basis (which can be precomputed), N_ℓ is the degree of the polynomial, and $\overline{u}_{ij}$ is the parameters of the polynomial. Lagrangian polynomials have a property that for some particular values $\overline{\tau}$, the basis assume the value $\ell_j(\overline{\tau}) = 1$ for a particular j and $\ell_k(\overline{\tau}) = 0$ for all $k = \{1, \ldots, N_i\} - \{j\}$. Then, for some time $\overline{t}$ such that $\frac{\overline{t} - t_i}{t_{i+1} - t_i} = \overline{\tau}$

$$u(\overline{t}) = u_{ij} \tag{79}$$

which means that by constraining the parameters u_{ij}, the value of $u(t)$ is being constrained at some "breakpoints". We will revisit the Lagrangian polynomials in the next section because they are the foundation for the collocation method.

By parametrizing the control variable, the search space of u is being redefined to a subspace for the control function u. For instance, instead of specifying that the

control is a piecewise continuous function with a finite number of discontinuities, the method narrows the search space to the space of piecewise constants functions with a finite number of discontinuities.

The third step is to discretize the differential and algebraic equations to transform the states and algebraic variables from time-dependent functions in a sequence of vectors.

2.3 DISCRETIZATION SCHEMES

The discretization methods can be categorized into two groups: implicit and explicit methods. In the implicit methods, the optimization solver is oblivious to the discretization technique and its equations, effectively being "blind" to the state and algebraic variables other than at some specified breakpoints. On the other hand, the explicit methods put the discretization equations in the optimization problem, leaving them to be solved by the optimization solver. Take the Runge Kutta of 4-th order (RK4) discretization method as an example. It can be both an explicit and implicit method, depending on how the optimization problem is formulated.

The RK4 has the following equation to compute the next x_{k+1} at the time t_k (BIEGLER, L. T., 2010), assuming an autonomous ODE system,

$$k_1 = hf(x_k, t_k) \tag{80a}$$

$$k_2 = hf\left(x_k + \frac{k_1}{2}, t_k + \frac{h}{2}\right) \tag{80b}$$

$$k_3 = hf\left(x_k + \frac{k_2}{2}, t_k + \frac{h}{2}\right) \tag{80c}$$

$$k_4 = hf\left(x_k + k_3, t_k + h\right) \tag{80d}$$

$$x_{k+1} = x_k + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \tag{80e}$$

where $h = t_{k+1} - t_k$ is the integration step.

Notice that if we use the substitution method to substitute k_1, k_2, k_3, and k_4 into (80e) using (80a-80d), we are hiding from the solver the intermediate computation steps. This would characterize an implicit approach.

In contrast, if all equations of (80) are used to create the optimization problem, the optimization solver will solve the equations at the same time it optimizes, hence the explicit methods being known as simultaneous methods.

For this reason, implicit methods are more compact and require fewer variables to represent, while explicit methods, on the other hand, have more variables. The increased number of variables and equations typically results in a more sparse optimization problem, which some optimization solvers can exploit. Therefore, one can not say in general that either one of the approaches is better than the other.

2.3.1 Collocation Method

The collocation method is a technique for discretizing and solving differential equations (ODEs, DAEs, and partial differential equations (PDEs)). The method works by sectioning the time (and space in PDEs) for which the differential equation is being solved. For each time subinterval, an approximation function is defined such that for a specified set of points in that time subinterval, the system equations hold. These points are known as collocation points.

The approximation function is chosen from a predefined family of basis functions. The basis function can be freely chosen, but polynomials with the Lagrangian polynomial form are preferred for practical reasons. As mentioned in Section 2.2.2 the parameters of this class of polynomials are the values of the approximating function at the collocation points (time breakpoints), hence facilitating the inclusion of bound constraints in the approximated functions.

Let τ be the variable that represents time normalized within the subinterval, having the value 0 at the beginning and 1 at the end of the subinterval. For the i-th subinterval the time variables relate through

$$t = t_{i-1} + h_i \tau \tag{81}$$

where

$$t \in T_i = [t_{i-1}, t_i] \tag{82}$$

$$t_i = t_{i-1} + h_{i-1} \tag{83}$$

where T_i is the interval, h_i is the length of the i-th time interval, t_0 is given, $t_{N_i} = t_f$, and N_i is the number of subintervals.

The basis for a Lagrangian polynomial can be defined by

$$\ell_j(\tau) = \prod_{k=0,\neq j}^{N_k} \frac{\tau - \tau_k}{\tau_j - \tau_k} \tag{84}$$

where τ_j and τ_k are collocation points.

Notice that the polynomial ℓ_j has the property that for $\tau = \tau_j$,

$$\ell_j(\tau_j) = 1 \tag{85}$$

$$\ell_k(\tau_j) = 0, \qquad \forall k \in \{0, \ldots, N_k\} - \{j\} \tag{86}$$

this means that if we define an approximation function $x(t)$ such that

$$x(t) = \sum_{j=0}^{N_k} \ell_j(\tau) x_{ij} \tag{87}$$

Table 1 – Legendre-Gauss (LG) and Legendre-Gauss-Radau (LGR) roots as collocation points (Table 10.1 of (BIEGLER, L. T., 2010))

Degree (N_k)	LG roots	LGR roots
1	0.500000	1.000000
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000
5	0.046910	0.057104
	0.230765	0.276843
	0.500000	0.583590
	0.769235	0.860240
	0.953090	1.000000

for all t_{ij} such that

$$t_{ij} = t_{i-1} + h_i \tau_j \tag{88}$$

the value of the approximating function is given by

$$x(t_{ij}) = x_{ij} \tag{89}$$

This property is known as the orthogonality, therefore the collocation method is commonly referred as the orthogonal collocation method when using polynomials with this property.

The collocation points are chosen such that $\tau_j < \tau_{j+1}$ for $j \in \{0, \ldots, N_k - 1\}$, $\tau_0 = 0$, and τ_j, for $k \in \{1, \ldots, N_k\}$, are chosen accondingly to the Gaussian quadrature (BIEGLER, L. T., 2010). In particular, choosing the Legendre-Gauss-Radau points results in polynomials with high precision and, as can be seen in Table 1, which includes the final point of the interval which is of great use for setting up constraints (KAMESWARAN; BIEGLER, L., 2007). Typically, quadratures are presented in the interval [–1, 1]. However, we are interested in the interval $\tau \in [0, 1]$; therefore, the points presented in Table 1 are scaled to the appropriate interval. Legendre-Gauss and Gauss-Lobatto are other common alternatives for choosing the collocation points.

Thus far, tools have been developed for creating a polynomial that passes through some specific points (x_{ij}) at some specific times (τ_j). The question now is how to use these tools to represent the state of a system.

Let us assume that the state is defined by the differential equation

$$\dot{x} = \frac{dx}{dt} = f(x, t) \tag{90}$$

Let us approximate x by the piecewise polynomial (87). In order for x to approximate the true state well, the system equations must be satisfied

$$\frac{dx}{dt}(t_{ik}) = f(x(t_{ik}), t_{ik}) \tag{91}$$

for all $k \in \{0, \ldots, N_k - 1\}$, for all $i \in \{1, \ldots, N_i\}$.

The left hand side of (91) can be replaced with the derivative of the approximation polynonmial, which can be obtained by differentiating (87),

$$\frac{dx}{d\tau}(t) = \sum_{j=0}^{N_k} \frac{d\ell_j}{d\tau}(\tau) x_{ij} \tag{92}$$

where the relation between the dt and $d\tau$ can be obtained by differentiating (88)

$$dt = h_i d\tau \tag{93}$$

These relations result in the following equation

$$\sum_{j=0}^{N_k} \frac{d\ell_j}{d\tau}(\tau_k) x_{ij} = h_i f(x_{ik}, t_{ik}) \tag{94}$$

for $k \in \{1, \ldots, N_k\}$.

For a system of equations to be solvable, it needs as many equations as it has unknown variables. For each subinterval i, there are $N_k + 1$ unknown variables x_{ik}, but on (94) there are only N_k equations. For the first subinterval, the additional equation can be obtained by the initial condition,

$$x_{1,0} = x_0 \tag{95}$$

where $x_{1,0}$ is the value of the state at the collocation point τ_0, on the first subinterval.

For the following subintervals, the extra equation can be obtained by guaranteeing the continuity of the state, that is the value at end of the subinterval has to be equal to the value at the begining of the subsequent subinterval.

$$x_{i+1,0} = \left[\sum_{j=0}^{N_k} \ell_j(\tau) x_{ij} \right] \Bigg|_{\tau=1}, \qquad i \in \{1, \ldots, N_i - 1\} \tag{96}$$

where $x_{i+1,0}$ is the state value at begining of the $(i+1)$-th subinterval, and the right-hand side is the value the end of the previous subinterval. If the LGR collocation points are used, the equation reduces to

$$x_{i+1,0} = x_{i,N_k} \tag{97}$$

which is a much simpler and sparse equation. The final condition can be obtained in a similar fashion,

$$x_f = \left[\sum_{j=0}^{N_k} \ell_j(\tau) x_{N_i,j}\right]\Bigg|_{\tau=1} \tag{98}$$

In the case of the OCP having terminal conditions, an extra set of equations can be applied to x_f.

It can be shown that the orthogonal collocation method using Lagrangian polynomials is equivalent to implicit RK, and therefore share its good properties regarding error tolerance and numerical stability. The truncation error ranges from $\mathcal{O}(h_i^{2N_k-2})$ to $\mathcal{O}(h_i^{2N_k})$, depending on which set of collocation points is chosen (BIEGLER, L. T., 2010). For a sufficient number of collocation points $N_k + 1$ and subintervals N_i, the approximation error can be neglected and the approximation $x(t)$ is close enough to the solution of the system (90).

A similar approach can be used for the algebraic variables. Since the algebraic variables do not have initial conditions, there is no need for a collocation point at $\tau = 0$. Hence a new polynomial basis is used.

Let $\widehat{\ell}_j(\tau)$ be a Lagrangian polynomial basis such that

$$\widehat{\ell}_j(\tau) = \prod_{k=1,\neq j}^{N_k} \frac{\tau - \tau_k}{\tau_j - \tau_k} \tag{99}$$

notice that the product starts in $k = 1$ instead of $k = 0$.

For each subinterval i, the approximation of y is given by

$$y(\tau) = \sum_{j=1}^{N_k} \widehat{\ell}_j(\tau) y_{ij} \tag{100}$$

where y_{ij} is the value of the algebraic varaible at the collocation point τ_j and the subinterval i.

Recall that the Lagrangian polynomial has the property,

$$y(t_{ij}) = y_{ij} \tag{101}$$

therefore, to define the value of y_{ij}, the algebraic equation can be evaluated at $t = t_{ij}$,

$$g(x_{ij}, y_{ij}, u_{ij}, t_{ij}) = 0 \tag{102}$$

where u_{ij} is the value of the control at the collocation point.

Example 2 (Demonstration of Orthogonal Collocation (BIEGLER, L. T., 2010))**.** *Consider the following dynamic system,*

$$\frac{dx}{dt} = x^2 - 2x + 1 \tag{103}$$

$$x(0) = -3 \tag{104}$$

where $t \in [0, 1]$*. The analytic solution for this ODE is* $x(t) = \frac{4t-3}{4t+1}$*. Using the Lagrangian polynomial with* $N_k = 3$*, a single subinterval (*$N_i = 1$*), and applying the collocation equations with the continuity equations leads to*

$$\sum_{j=0}^{3} x_{ij} \frac{d\ell_j}{d\tau}(\tau_k) = h(x_{ik}^2 - 2x_{ik} + 1), \qquad \forall k \in \{1, \ldots, 3\}, i \in \{1, \ldots, N_i\} \tag{105a}$$

$$x_{1,0} = -3 \tag{105b}$$

$$x_{i+1,0} = \sum_{j=0}^{3} \ell_j(1) x_{ij}, \qquad \forall i \in \{1, \ldots, N_i\} \tag{105c}$$

$$x_f = \sum_{j=0}^{3} \ell_j(1) x_{N_i,j} \tag{105d}$$

where $h = 1$ *since* $\frac{t_f - t_0}{N_i} = 1$*.*

For the LGR tableau with $N_k = 3$*, the collocation points are* $\tau_0 = 0$*,* $\tau_1 = 0.155051$*,* $\tau_2 = 0.644949$*, and* $\tau_3 = 1$*. For* $N_i = 1$*, the collocation equations are*

$$\sum_{j=0}^{3} x_{1j} \frac{d\ell_j}{d\tau}(\tau_k) = x_k^2 - 2x_k + 1, \qquad k = \{1, \ldots, 3\} \tag{106}$$

which expands to

$$x_{1,0}(-30\tau_k^2 - 9) + x_{1,1}(46.7432\tau_k^2 - 51.2592\tau_k + 10.0488) + x_{1,2}(-26.7423\tau_k^2 + 20.595\tau_k - 1.38214) + x_{1,3}\left(10\tau_k^2 - \frac{16}{3}\tau_k + \frac{1}{3}\right) = (x_{1,k}^2 - 2x_{1,k} + 1) \tag{107}$$

for all $k \in 1, ..., 3$*.*

Solving the set of equations leads to $x_{1,0} = x_0 = -3$*,* $x_{1,1} = -1.65701$*,* $x_{1,2} = 0.032053$*,* $x_{1,3} = x_f = 0.207272$*. The plot of the polynomial is shown in Figure 5, for* $N_i = 1$ *and* $N_k = 3$*. Note that by increasing the number of segments* N_i*, by subdividing the original intreval, or the order of the polynomial* N_k *the error reduces and a better approximation can be achieved.*

2.3.2 Multiple-Shooting

The shooting methods are powerful methods for solving mathematical problems, in particular methods involving differential equations (STOER; BULIRSCH, 2002). There are two shooting methods: the single-shooting and the multiple-shooting method. The latter is an improvement over the former, being more numerically stable and, when applied in optimal control, allows for a better imposition of constraints. However, for the sake of explanation, let us first present the single-shooting method.

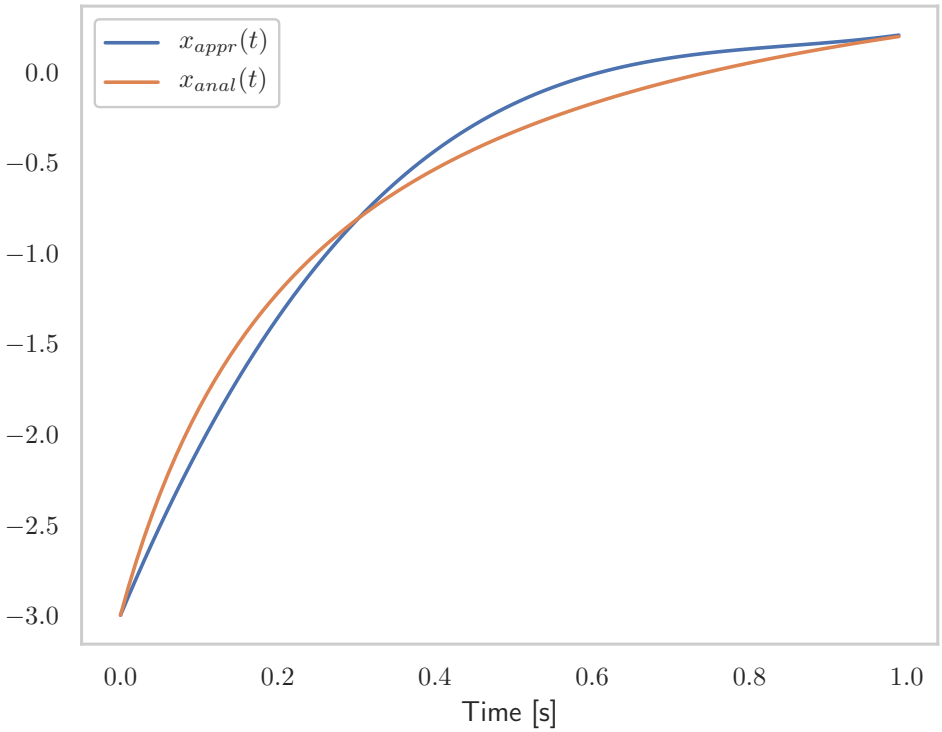


Figure 5 – Resulting polynomial

The idea of the single-shooting method can be better understood with an example. Without loss of generality, let us take the ODE system

$$\dot{x} = -x + u \tag{108}$$

with $x(t_0) = 0$ and we want to figure out what the constant value for the control u such that $x(t_f) = 5$, where $t_0 = 0$ and $t_f = 2$. Since the terminal value for x depends on the constant value chosen for the control u, let us use the notation $x(t_f|u)$ to imply the terminal value of x given a particular u. Let us define a function $F(u) = 5 - x(t_f|u)$ which has its root at $x(t_f) = 5$, for some particular u^*.

The problem then becomes finding a root for $F(u)$ given the dynamic equation (108) and its initial condition. Notice that $F(u)$ requires the solution of an initial value problem (IVP).

In order to find a zero for $F(u)$, one may start with the trivial solution $u = 5$, and by solving the IVP for this given control, one would obtain $x(t_f|5) = 4.326$, and $F(5) = 0.6740$. By observing that the final state is slightly lower than the desired, we could try with $u = 6$, which would result in $x(t_f|6) = 5.191$, and $F(6) = 0.191$. Which is closer to the root, and by following this process, just like a bisection algorithm, one would obtain the solution for our problem. A different algorithm could be used in place of the bisection approach, but, inevitably an IVP would need to be solved at each algorithm step. For instance, one could use a Newton step algorithm, where each iteration is

obtained through

$$u^{k+1} = u^k - \frac{F(u^{(k)})}{\frac{\partial F}{\partial u}(u^{(k)})} \tag{109}$$

Recall, however, that $F(u^k)$ is obtained from the solution of the IVP and therefore $\frac{\partial F}{\partial u}(u^{(k)})$ cannot be computed as a regular derivative. In order to obtain this derivative, a sensitivity analysis needs to be computed. The sensitivity analysis will be further discussed in this section.

In the example given above, the system was simple, and the integration period was short. Intuitively one can imagine that the relation between the control u and the final state $x(t_f)$ was simple, close to linear. On the other hand, if the system was complex or even unstable, it is easy to imagine that these relations would be much sharper. A slight variation on the control would result in a notably different final state. This issue is further pronounced with an extended time horizon. Another significant limitation of the single-shooting method is the inability to access variables that are not in extrema of the integration period. Since the IVP is solved from t_0 to t_f, a constraint in the state as $x(\frac{t_f - t_0}{2}) \leq 4$ becomes impossible to implement. Because of these aspects, the multiple shooting method is a clear improvement over single shooting for most applications.

The multiple shooting method breaks the time interval into smaller subintervals to avoid the long integration periods that can lead to the issues just mentioned. The shooting starts with an estimate of the state at the start of the subinterval and produces an estimate of the state at the end of the subinterval. This approach resembles the collocation method: a shooting (an IVP) is performed for each subinterval, much like the collocation method creates a polynomial for each subinterval.

Let us define $F_i(u_i)$, the solution of the IVP for a single subinterval, as

$$F_i(u_i, x_{0,i}) = x(t_{i+1}) \tag{110a}$$

where

$$\dot{x} = f(x, u_i, t) \tag{110b}$$

$$x(t_i) = x_{0,i} \tag{110c}$$

$$t \in [t_i, t_{i+1}] \tag{110d}$$

where $i \in \{1, ..., N_i\}$ is the subinterval, N_i is number of subintervals, u_i is the control in the i-th subinterval, and $x_{0,i}$ is the initial condition for the given subinterval.

Multiple shooting seems like running several single-shootings in parallel, one for each subinterval, but connecting them through continuity equations. For the first subintervals, the intial conditions are applyed to define the initial condition

$$x_{0,1} = x_0 \tag{111}$$

For the remaining subinterval, the initial equations should be equal to the state at the end of the previous subinterval

$$x_{0,i+1} = F_i(u_i, x_{0,i}) \tag{112}$$

for all $i \in \{1, ..., N_i\}$.

Notice that the equations that form the multiple shooting are (111) and (112), which sums to $N_x \times N_i$ equations. This number is far less than the number of equations required for implementing the collocation method. However, while using the multiple shooting method, constraints can only be set at the beginning and end of each subinterval. On the other hand, one can include constraints inside the subintervals, at each collocation point, when the collocation method is used.

As mentioned earlier, F_i is the solution of an IVP, which can be as simple as the RK4 presented earlier, or as complex as an IVP solver with varying integration steps and error control. For a simple integrator, as the RK4, the derivative can be obtained through the chain rule, but it is an unreasonable task to try to do so for more complex solvers.

One common approach to solve this issues is to use a finite difference strategy, for instance one could obtain an approximated derivative with respect to the control variable using

$$\frac{\partial F_i}{\partial u_i}(u_i, x_{0,i}) \approx \frac{F(u_i + \Delta u_i, x_{0,i}) - F(u_i, x_{0,i})}{\Delta u_i} \tag{113}$$

However, this approach becomes computationally intensive as the number of derivatives increases. On top of that, these derivatives are not exact, being disturbed by numerical noise resulting from the complex integration algorithms.

A better approach to this issue is to use sensitivity analysis, which computes the derivatives through a numerical integration along with the IVP. There are two sensitivity analysis methods, the forward and the backward method (also known as the adjoint method). Their denominations originate from how they are computed. The forward sensitivity is computed in the same direction as the IVP, while the backward is computed in the opposite direction.

Consider a 1-index DAE system with $x(t) \in \mathbb{R}^{N_x}$, $y(t) \in \mathbb{R}^{N_y}$, $t \in [t_0, t_f]$, and a vector of decision parameters $p \in \mathbb{R}^{N_p}$ for which we want to obtain the derivatives. Assume that the system functions f and g are continuously differentiable with respect to all of their arguments. The DAE can be described as

$$\dot{x} = f(x, y, t, p) \tag{114a}$$

$$0 = g(x, y, t, p) \tag{114b}$$

$$x(t_0) = x_0 \tag{114c}$$

where x_0 is the initial condition that can be parametrized by some parameter vector p.

Let $F(x(t_f), p)$ be a function for which the derivative must be taken with respect to p,

$$\frac{dF}{dp}(x(t_f), p) = \frac{\partial F}{\partial x}\frac{dx}{dp}(t_f) + \frac{\partial F}{\partial p} \tag{115}$$

where the partial derivatives can easily be obtained, leaving the term $\frac{dx}{dp}(t_f)$ to be defined.

If the DAE system (114) is differentiated with respect to p, the following set of equations is obtained

$$\frac{d\dot{x}}{dp} = \frac{df}{dp} = \frac{\partial f}{\partial x}\frac{dx}{dp} + \frac{\partial f}{\partial y}\frac{dy}{dp} + \frac{\partial f}{\partial p} \tag{116a}$$

$$\frac{dg}{dp} = \frac{dg}{dp} = \frac{\partial g}{\partial x}\frac{dx}{dp} + \frac{\partial g}{\partial y}\frac{dy}{dp} + \frac{\partial g}{\partial p} = 0 \tag{116b}$$

$$\frac{dx}{dp}(t_0) = \frac{dx_0}{dp} \tag{116c}$$

Notice that the initial condition x_0 might depend on p, in which case $\frac{dx_0}{dp}$ is not null.

Let us define two matrices variables

$$S = \frac{dx}{dp} = \begin{bmatrix} \frac{dx_1}{dp_1} & \cdots & \frac{dx_1}{dp_{N_p}} \\ \vdots & \ddots & \vdots \\ \frac{dx_{N_x}}{dp_1} & \cdots & \frac{dx_{N_x}}{dp_{N_p}} \end{bmatrix} \tag{117a}$$

$$R = \frac{dy}{dp} = \begin{bmatrix} \frac{dy_1}{dp_1} & \cdots & \frac{dy_1}{dp_{N_p}} \\ \vdots & \ddots & \vdots \\ \frac{dy_{N_y}}{dp_1} & \cdots & \frac{dy_{N_y}}{dp_{N_p}} \end{bmatrix} \tag{117b}$$

$$\tag{117c}$$

where $S(t) \in N_x \times N_p$ represents the Jacobian matrix of x with respect to p, and $R(t) \in N_y \times N_p$ represents the Jacobian matrix of y with respect to p.

Given that f is continuously differentiable, one can use the Schwartz's theorem (ALLEN, 1962) to change the order of the differentiations

$$\frac{d}{dp}\frac{dx}{dt} = \frac{d}{dt}\frac{dx}{dp} = \frac{dS}{dt} \tag{118}$$

Hence, the system (116) can be rewritten as

$$\frac{dS}{dt} = \frac{\partial f}{\partial x}S + \frac{\partial f}{\partial y}R + \frac{\partial f}{\partial p} \tag{119a}$$

$$0 = \frac{\partial g}{\partial x}S + \frac{\partial g}{\partial y}R + \frac{\partial g}{\partial p} \tag{119b}$$

$$S(t_0) = \frac{dx_0}{dp} \tag{119c}$$

and the derivative (115) is now given by

$$\frac{dF}{dp}(x(t_f), p) = \frac{\partial F}{\partial x} S(t_f) + \frac{\partial F}{\partial p} \tag{120}$$

By including (119) into the DAE system one can compute through an IVP not only the value of F but also the Jacobian. The sensitivity equations do not rely on the function of interest F, which is advantageous for cases where F has high dimensionality. The inclusion of the $(N_x + N_y) \times N_p$ equations is often more computationally efficient than using a finite difference method. Not only that but the sensitivity analysis method results in an exact Jacobian, while the finite differences method does not.

The following example shows how sensitivity analysis can be used in a practical scenario.

Example 3. *Let us have an DAE system given by*

$$\dot{x}_1 = x_1^2 + x_2^2 - 3y \tag{121a}$$

$$\dot{x}_2 = x_1 x_2 + x_1 (y + p_2) \tag{121b}$$

$$0 = x_1 y + p_3 x_2 \tag{121c}$$

$$x(t_0) = \begin{bmatrix} 5 \\ p_1 \end{bmatrix} \tag{121d}$$

with $t_0 = 0$. By defining the sensitivity states S and the sensitivity algebraic variables R, the sensitivity equations can be written as

$$\dot{S} = \begin{bmatrix} \dot{S}_{11} & \dot{S}_{12} & \dot{S}_{13} \\ \dot{S}_{21} & \dot{S}_{22} & \dot{S}_{23} \end{bmatrix} = \begin{bmatrix} 2x_1 & 2x_2 \\ x_2 + y + p_2 & x_1 \end{bmatrix} S + \begin{bmatrix} -3 \\ x_1 \end{bmatrix} R + \begin{bmatrix} 0 & 0 & 0 \\ 0 & x_1 & 0 \end{bmatrix} \tag{122a}$$

$$0 = \begin{bmatrix} y & p_3 \end{bmatrix} S + x_1 \begin{bmatrix} R_1 & R_2 & R_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & x_2 \end{bmatrix} \tag{122b}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} S_{11}(0) & S_{12}(0) & S_{13}(0) \\ S_{21}(0) & S_{22}(0) & S_{23}(0) \end{bmatrix} \tag{122c}$$

Let the function of interest F be given by

$$F(x(t_f), t_f) = \frac{1}{2} x(t_f)^T x(t_f) \tag{123}$$

Then the derivatives with respect to the parameters p are given by

$$\frac{dF}{dp}(x(t_f), t_f) = \frac{\partial F}{\partial x} S(t_f) = \begin{bmatrix} x_1(t_f) & x_2(t_f) \end{bmatrix} \begin{bmatrix} S_{11}(t_f) & S_{12}(t_f) & S_{13}(t_f) \\ S_{21}(t_f) & S_{22}(t_f) & S_{23}(t_f) \end{bmatrix} \tag{124}$$

which provides derivatives that can be used in an algorithm like (109) *to find a minimum or a root for F.*

2.4 AUGMENTED LAGRANGIAN

In mathematical programming, the augmented Lagrangian method (BERTSEKAS, 1996) is an algorithm developed to solve equality constrained optimization problems (COP), which can also be extended to inequality constraints. The algorithm achieves the solution of the constrained problem by solving a sequence of unconstrained optimization problems.

Let us describe a COP by

$$\min_{z} \quad V(z) \tag{125a}$$

$$\text{s.t.:} \quad c(z) = 0 \tag{125b}$$

where z is the optimization variable, V is the objective function, and c is the constraint function.

In order to obtain an unconstrained optimization problem, the augmented Lagrangian method relaxes the equality constraint (125b) and includes a penalization term in the objective function creating the augmented objective function

$$V_{\mu_k}(z, \lambda_k) = V(z) + \lambda_k^T c(z) + \frac{\mu_k}{2} \|c(z)\|^2 \tag{126}$$

where $\mu_k > 0$ is a scalar such that $\{\mu_k\} \to \infty$, and λ_k is an approximation of the Lagrange multiplier of the constraint $c(z)$, which belongs to a sequence $\{\lambda_k\} \to \lambda^*$ (BERTSEKAS, 1996).

The solution of (125) is obtained by a sequence of unconstrained minimizations of (126), determined by the scalar μ_k and the vector λ_k which are updated at each iteration. The method is outlined in Algorithm 1 (NOCEDAL; WRIGHT, 2006).

Algorithm 1 Augmented Lagrangian for Constrained Optimization

Require: $\mu_0 > 0$, $\varepsilon_{V,0} > 0$, starting points z_0^s and λ_0:
  repeat
    $z_k \leftarrow \arg\min_z V_{\mu_k}(z, \lambda_k)$, starting at z_k^s, satisfying $\left\| \frac{\partial V_{\mu_k}}{\partial z}(z_k, \lambda_k) \right\| \leq \varepsilon_{V,k}$,
    obtain λ_{k+1} with the equation $\lambda_{k+1} = \lambda_k + \mu_k c(z_k)$,
    choose a new parameter $\mu_{k+1} \geq \mu_k$,
    set the starting point for the next iteration $z_{k+1}^s = z_k$,
    select tolerance $\varepsilon_{V,k+1}$
    $k \leftarrow k + 1$
  until z_k satisfies a convergence condition

An update rule often used for the parameter μ_k is

$$\mu_{k+1} = \beta \mu_k \tag{127}$$

where β is a scalar greater than 1, usually in the range from 5 to 10. However, if μ_k is increased too much, the second order derivatives of the minimization of (126) might

become ill conditioned (BERTSEKAS, 1996). To this end, an alternative update rule is

$$\mu_{k+1} = \begin{cases} \beta\mu_k & \text{if } \beta\mu_k < \mu_{\max} \\ \mu_{\max} & \text{otherwise} \end{cases} \tag{128}$$

There exists a theoretical value μ^* such that for any $\mu > \mu^*$, $\{V_\mu(z_k, \lambda_k)\} \to V(z^*)$ where z^* is a solution for (125), if the tolerance $\varepsilon_{V,k+1} \to 0$ as $k \to \infty$ and the problem satisfies some conditions (NOCEDAL; WRIGHT, 2006).

The augmented Lagrangian method is the basis for many other optimization algorithms, including the alternating direction multiplier method (ADMM) (BOYD et al., 2010). In addition to solving convex constrained optimization problems, the ADMM can also be used to train machine learning and data science algorithms. ADMM allows the solution of these optimization problems in a distributed fashion, which has become increasingly important as the available data has grown to a size that can only be processed in computer clusters.

2.5 AUGMENTED LAGRANGIAN ALGORITHM FOR OPTIMAL CONTROL PROBLEMS

Based on the augmented Lagrangian for constrained optimization, a relax-and-discretize approach for optimal control of continuous-time differential-algebraic systems (DAE) has been proposed. The algorithm works by relaxing the algebraic equations and penalizing its violation into the objective function using the augmented Lagrangian, which converts the original problem into a sequence of optimal control problems (OCPs) of ordinary differential equations (ODEs). The relax-and-discretize approach of the algorithm provides flexibility by allowing the OCPs of ODEs to be solved by the method of choice, such as direct or indirect methods. Conditions are developed for global, local, and sub-optimal convergence in terms of the solution of the underlying OCPs. The method is applied to an illustrative example.

The proposed algorithm solves the OCP in the form $\mathcal{P}$

$$\mathcal{P}: \quad \min_{u,t_f} \quad J(x,y,u) = V(x(t_f),t_f) + \int_{t_0}^{t_f} L(x,y,u,t)\,dt \tag{129a}$$

$$\text{s.t.:} \quad \dot{x} = f(x,y,u,t) \tag{129b}$$

$$0 = g(x,y,u,t) \tag{129c}$$

$$x(t_0) = x_0 \tag{129d}$$

$$u \in U_B,\ t \in [t_0, t_f] \tag{129e}$$

The first step is to relax the algebraic constraint (129c), and then introduce the new objective functional,

$$J_\mu(x,y,u,\nu) = \int_{t_0}^{t_f} \mathcal{L}_\mu(x,y,u,\nu,t)\,dt \tag{130}$$

where the function $\mathcal{L}_\mu$ is given by

$$\mathcal{L}_\mu(x, y, u, \nu, t) = L(x, y, u, t) + \nu(t)^T g(x, y, u, t) + \frac{\mu}{2} \|g(x, y, u, t)\|^2 \tag{131}$$

with $\mu > 0$ being a scalar, and the function $\nu : [t_0, t_f] \to \mathbb{R}^{N_y}$ being an estimate of the multiplier function ν^*, which will be driven by the algorithm towards satisfying the optimality conditions (21b) of problem $\mathcal{P}$.

The just introduced functional (130) is used as the objective of the auxiliary OCP, which is solved by the algorithm at each iteration k,

$$\mathcal{P}_\mathcal{L}(\mu_k, \nu_k) : \min_{y,u} J_{\mu_k} = \int_{t_0}^{t_f} \mathcal{L}_{\mu_k}(x, y, u, \nu_k, t)\, dt \tag{132a}$$

$$\text{s.t.: } \dot{x} = f(x, y, u, t) \tag{132b}$$

$$x(t_0) = x_0 \tag{132c}$$

$$u \in U_B,\ t \in [t_0, t_f] \tag{132d}$$

Notice that with the algebraic equation relaxed, y is free and becomes an optimization variable. For this reason, the algebraic variable acts similarly to the control variable u. Problem (132) can be reformulated with the aggregation of the y and u into a new extended control variable $\widehat{u} = [u, y]$, where $\widehat{u}(t) \in \widehat{U} = U_B \times Y$. With this reformulation, problem $\mathcal{P}_\mathcal{L}$ meets the standard form of an OCP of ODE, whose optimality conditions are well established (KIRK, 2004).

2.5.1 Algorithm

The proposed algorithm follows the same structure of the augmented Lagrangian for standard constrained optimization (BERTSEKAS, 1996). Let μ_0 be an initial value for the sequence of penalty values $\{\mu_k\}$, ν_0 be an initial estimate for the sequence of multipliers $\{\nu_k\}$, and ε_g be a tolerance on the violation of the algebraic constraint. Starting with these parameters, at each iteration k, the problem (132) is solved, the multiplier estimate and penalty are updated, and the process is repeated until an acceptable tolerance is achieved, as detailed in Algorithm 2.

The pseudo-function *solve* yields a solution for the sub-problem $\mathcal{P}_\mathcal{L}$ and returns the functional values J_k and the trajectories for the states, algebraic and control variables. The pseudo-function *update_mu* represents the use of an update rule for the penalization μ_k. For the convergence analysis it is assumed that $\mu_{k+1} = \beta\mu_k$ with a $\beta > 1$ to ensure that $\mu_k \to \infty$. In practice, however, a $\mu_k \to \infty$ will cause ill-conditioning on the Hessian of the sub-problem $\mathcal{P}_\mathcal{L}$, therefore when performing a computational implementation, it is recommended to use an upper bound $\mu_{\max}$ for the penalization.

Algorithm 2 Augmented Lagrangian for Optimal Control

Require: μ_0, ν_0, and ε_g:
1: **for** $k = 1, 2, \ldots$ **do**
2: $(J_k, x_k, y_k, u_k) \leftarrow solve\{\mathcal{P}_{\mathcal{L}}(\mu_k, \nu_k)\}$
3: $\nu_{k+1} \leftarrow \nu_k + \mu_k g(x_k, y_k, u_k)$
4: $\mu_{k+1} \leftarrow update\_mu\{\mu_k\}$
5: **if** $\|g(x_k, y_k, u_k)\| < \varepsilon_g$ **then**
6: **return** u_k
7: **end if**
8: **end for**

2.5.2 Mathematical Properties

In this section, the necessary conditions are established for the solution sequence produced by the algorithm to arrive at a global solution of the OCP for DAE. Furthermore, conditions are then presented for convergence to local solutions and convergence under a suboptimal solution sequence, which reflects situations typically found in practice.

The development of the conditions follows a handful of assumptions that are presented in the next.

Assumption 2 (Regularity)**.** *For problem* $\mathcal{P}$ (129) *and* $\mathcal{P}_{\mathcal{L}}(\mu_k, \nu_k)$ (132) *to be well-conditioned, we assume that*

1. $x : [t_0, t_f] \to \mathbb{R}^{N_x}$ *is continuously differentiable;* $y : [t_0, t_f] \to \mathbb{R}^{N_y}$, $u : [t_0, t_f] \to U_B$, *and* $\nu_k : [t_0, t_f] \to \mathbb{R}^{N_y}$ *are continuous,*
2. *L, g, and f are continuously differentiable with respect to all the arguments,*
3. *the space of feasible functions for problems* $\mathcal{P}$ *and* $\mathcal{P}_{\mathcal{L}}$ *are compact,*
4. *the Jacobian* $\frac{\partial g}{\partial y}(x(t), y(t), u(t), t)$ *has full rank for all* $x(t) \in X$, $y(t) \in Y$, $u(t) \in U_B$, *and* $t \in [t_0, t_f]$,
5. *the sequence* $\{\mu_k\}$ *has the property that* $0 < \mu_k < \mu_{k+1}$ *for all k, and* $\mu_k \to \infty$ *as* $k \to \infty$,
6. *problem* $\mathcal{P}$ *and* $\mathcal{P}_{\mathcal{L}}(\mu_k, \nu_k)$ *are solvable.*

From condition 4 of the Assumption 2, the algorithm does not apply to OCP with DAE of index greater than one.

The following theorems will make use of uniform convergence and uniform norm for functions; their definitions follow.

Definition 1. *Let* $f : [t_0, t_f] \to \mathbb{R}^N$ *be a continuous function then* $\|f\|$ *is given by* $\|f\| = \max_{t \in [t_0, t_f]} \|f(t)\|_\infty$.

Definition 2. *Let $f_k : [t_0, t_f] \to \mathbb{R}^N$ be a function for every $k \in \mathbb{N}$. The sequence of functions $\{f_k\}$ converges uniformly to the limiting function $f^* : [t_0, t_f] \to \mathbb{R}^N$ if, for every $\varepsilon > 0$, there exists a number $K \in \mathbb{N}$ such that for all $t \in [t_0, t_f]$ and all $k \geq K$, we have $\|f_k(t) - f^*(t)\| < \varepsilon$.*

The following theorem states the necessary conditions for the algorithm to converge to a global optimum.

Theorem 6. *Let the functions $\langle x_k, y_k, u_k \rangle$ be global minima of the problem $\mathcal{P}_\mathcal{L}(\mu_k, \nu_k)$ (Eq. 132) at each iteration k. In addition, assume that $\{\langle x_k, y_k, u_k \rangle\}_K$ and $\{\nu_k\}_K$ are uniformly convergent subsequences. Then, under Assumption 2, the limiting functions of every subsequences $\{\langle x_k, y_k, u_k \rangle\}_K$ are a global minimizer of problem $\mathcal{P}$ and the subsequence $\{J_{\mu_k}(x_k, y_k, u_k, \nu_k)\}_K$ converges to the optimum objective of $\mathcal{P}$.*

Proof. Let $\langle x^*, y^*, u^* \rangle$ be limiting functions of the subsequence $\{\langle x_k, y_k, u_k \rangle\}_K$. By definition of x_k, y_k, and u_k, for a given k

$$J_{\mu_k}(x_k, y_k, u_k, \nu_k) \leq J_{\mu_k}(x, y, u, \nu_k) \tag{133}$$

for all feasible x, y, and u.

Let J^* denote the optimal value of $\mathcal{P}$. We have that

$$J^* = \min_{\substack{u \\ \text{s.t. (129b)-(129e)}}} \qquad J = \min_{\substack{y,u \\ \text{s.t. (132b)-(132d)} \\ g(x,y,u,t)=0}} J_{\mu_k}(\mu_k, \nu_k) \tag{134}$$

the last term implies the minimization of the problem $\mathcal{P}_\mathcal{L}$ over y and u with the additional equation $g(x, y, u, t) = 0$. The first equality holds by definition. The second equality holds because $\mathcal{P}$ and $\mathcal{P}_\mathcal{L}$ are equivalent when the equation $g(x, y, u, t) = 0$ is included in $\mathcal{P}_\mathcal{L}$.

The inequality (133) holds for any x, y, and u, including a minimizer of (134). Therefore, we can substitute the optimum value J^* on the right-hand side of (133), and on the left-hand side we substitute $J_{\mu_k}(x_k, y_k, u_k, \nu_k)$ with its definition to obtain

$$\int_{t_0}^{t_f} L(x_k, y_k, u_k, t) + \nu_k^T g(x_k, y_k, u_k, t) + \frac{\mu_k}{2} \|g(x_k, y_k, u_k, t)\|^2 \, dt \leq J^* \tag{135}$$

Given that the subsequence $\{\nu_k\}_K$ is uniformly convergent, it has a limiting function ν^*. By taking the limit with $k \to \infty$ in the inequality (135) we obtain

$$\int_{t_0}^{t_f} \left[L(x^*, y^*, u^*, t) + {\nu^*}^T g(x^*, y^*, u^*, t) \right] dt + \lim_{k \to \infty} \frac{\mu_k}{2} \int_{t_0}^{t_f} \|g(x_k, y_k, u_k, t)\|^2 \, dt \leq J^* \tag{136}$$

Since $\|g(x_k, y_k, u_k, t)\|^2 \geq 0$ and $\mu_k \to \infty$, it follows that we must have $g(x_k, y_k, u_k, t) \to 0$ and

$$g(x^*, y^*, u^*, t) = 0 \qquad \forall t \in [t_0, t_f] \tag{137}$$

otherwise the limit on the left-hand side of (136) would go to $+\infty$ which does not hold since J^* is finite. Therefore,

$$J(x^*, y^*, u^*) = \int_{t_0}^{t_f} L(x^*, y^*, u^*, t)\, dt \leq J^* \tag{138}$$

Any solution to problem $\mathcal{P}_\mathcal{L}$ satisfies all of the constraints of $\mathcal{P}$ except the relaxed algebraic equations. However (137) ensures that the limiting functions x^*, y^*, and u^* do satisfy the algebraic equation. By definition, J^* is less or equal to the objective of any feasible functions for problem $\mathcal{P}$, therefore we have

$$J^* \leq J(x^*, y^*, u^*) \tag{139}$$

Using (138) and (139), we conclude that

$$J^* \leq J(x^*, y^*, u^*) \leq J^* \implies J^* = J(x^*, y^*, u^*) \tag{140}$$

which proves that the limiting functions x^*, y^*, and u^* are global minimizers for problem $\mathcal{P}$ and that $\{J_{\mu_k}(x_k, y_k, u_k, \nu_k)\}_K \to J^*$. □

Theorem 6 assumes that the original $\mathcal{P}$ and the augmented problems $\mathcal{P}_\mathcal{L}$ are solved to global optimality. The subsequent theorem shows that the sequence of problems $\mathcal{P}_\mathcal{L}$ that reaches a local minimum converges to a local minimum of problem $\mathcal{P}$.

Definition 3. *Let $\mathcal{V}$ be a function space, then a nonempty set $\mathcal{V}^* \subset \mathcal{V}$ is said to be an isolated set of local minima of problem $\mathcal{P}$ if each function $v^* \in \mathcal{V}^*$ is a local minimum of problem $\mathcal{P}$ and, for some $\varepsilon > 0$, the set*

$$\mathcal{V}_\varepsilon^* = \{v \in \mathcal{V} : \|v - v^*\| \leq \varepsilon \text{ for some } v^* \in \mathcal{V}^*\} \tag{141}$$

contains no local minima of problem $\mathcal{P}$ other than the functions of $\mathcal{V}^*$.

An isolated set of local minima consisting of a single function is a strict local minimum.

Theorem 7. *Suppose that the regularity Assumption 2 holds, and that $\mathcal{V}^*$ is a compact and isolated set of local minima of problem $\mathcal{P}$. If $\langle x_k, y_k, u_k\rangle$ is a local minimizer for problem $\mathcal{P}_\mathcal{L}$ for each k, then there exists a subsequence $\{\langle x_k, y_k, u_k\rangle\}_K$ converging to a limiting function $\langle x^*, y^*, u^*\rangle \in \mathcal{V}^*$. Furthermore, if $\mathcal{V}^*$ consists of a single function $\langle x^*, y^*, u^*\rangle$, then there exists a sequence $\{\langle x_k, y_k, u_k\rangle\}$ such that $\{\langle x_k, y_k, u_k\rangle\} \to \langle x^*, y^*, u^*\rangle$.*

Proof. Consider the set

$$\mathcal{V}_{\tilde{\varepsilon}}^* = \{v \in \mathcal{V} : \|v - v^*\| \leq \tilde{\varepsilon} \text{ for some } v^* \in \mathcal{V}^*\} \tag{142}$$

where $\mathcal{V}$ is the set of feasible functions of $\mathcal{P}_{\mathcal{L}}$, with some $0 < \tilde{\varepsilon} < \varepsilon$, and ε is as in (141). From (142) and because $\mathcal{V}$ is compact by Assumption 2, it follows that $\mathcal{V}^*_{\tilde{\varepsilon}}$ is also compact, and hence the problem

$$\min_{x,y,u} J_{\mu_k} = \int_{t_0}^{t_f} \mathcal{L}_{\mu_k}(x, y, u, \nu_k, t)\, dt \tag{143a}$$

$$\text{s.t.: } \dot{x} = f(x, y, u, t) \qquad \forall t \in [t_0, t_f] \tag{143b}$$

$$u(t) \in U_B \qquad \forall t \in [t_0, t_f] \tag{143c}$$

$$\langle x, y, u\rangle \in \mathcal{V}^*_{\tilde{\varepsilon}}, \quad x(t_0) = x_0 \tag{143d}$$

has a global minimum $\langle x_k, y_k, u_k\rangle \in \mathcal{V}^*_{\tilde{\varepsilon}}$. By Theorem 6, every limiting function $\langle x^*, y^*, u^*\rangle$ of $\{\langle x_k, y_k, u_k\rangle\}_K$ is a global minimum of the problem

$$\min_{x,y,u} J = \int_{t_0}^{t_f} L(x, y, u, t)\, dt \tag{144a}$$

$$\text{s.t.: } \dot{x} = f(x, y, u, t) \qquad \forall t \in [t_0, t_f] \tag{144b}$$

$$g(x, y, u, t) = 0 \qquad \forall t \in [t_0, t_f] \tag{144c}$$

$$u(t) \in U_B \qquad \forall t \in [t_0, t_f] \tag{144d}$$

$$\langle x, y, u\rangle \in \mathcal{V}^*_{\tilde{\varepsilon}}, \quad x(t_0) = x_0 \tag{144e}$$

Furthermore, each global minimum of the problem above must belong to $\mathcal{V}^*$ by the definition of $\mathcal{V}^*_{\tilde{\varepsilon}}$. Thus there is a subsequence $\{\langle x_k, y_k, u_k\rangle\}_K$ converging to $\langle x^*, y^*, u^*\rangle \in \mathcal{V}^*$. If $\mathcal{V}^*$ contains only one local optimum, then all the subsequences will lead to this local optimum, therefore $\{\langle x_k, y_k, u_k\rangle\} \to \langle x^*, y^*, u^*\rangle \in \mathcal{V}^*$. □

Theorems 6 and 7 implicitly assume that a local or global minimum solutions are found for the augmented Lagrangian problem at each iteration. From a practical point of view, numerical methods are expected to terminate when the optimality conditions of $\mathcal{P}_{\mathcal{L}}$ are almost satisfied, meaning that for a small scalar $\varepsilon_k > 0$ the necessary optimality conditions (KIRK, 2004) are

$$\|f(x_k, y_k, u_k, t) - \dot{x}\| \leq \varepsilon_k, \tag{145a}$$

$$\left\| \frac{\partial \mathcal{L}_{\mu_k}}{\partial x}(x_k, y_k, u_k, \nu_k, t)^T + \frac{\partial f}{\partial x}(x_k, y_k, u_k, t)^T \lambda_k + \dot{\lambda}_k \right\| \leq \varepsilon_k, \tag{145b}$$

$$\left\| u_k(t) - \arg \inf_{u \in U_B} H(x_k(t), \lambda_k(t), y_k, \nu_k, u, t) \right\| \leq \varepsilon_k, \tag{145c}$$

$$\left\| \frac{\partial \mathcal{L}_{\mu_k}}{\partial y}(x_k, y_k, u_k, \nu_k, t)^T + \frac{\partial f}{\partial y}(x_k, y_k, u_k, t)^T \lambda_k \right\| \leq \varepsilon_k. \tag{145d}$$

The following theorem shows that if $\varepsilon_k \to 0$, the algorithm still converges. To prove this properties the therem requires the folllowing lemma.

Lemma 1. *Let $g : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ be a continuous function, and the sequence of functions $\{f_n\}$ to converge uniformly to f, where $f_n : [0,1] \to \mathbb{R}^{d_1}$. Let the function norm $\|\cdot\|$ be given by $\|g\| = \max_{x\in[0,1]} \|g(x)\|_\infty$. Then $\{g(f_n)\}$ converges uniformly to $g(f)$.*

Proof. If f_n converges uniformly to f, then for all ε_f exists N, such that $\|f_n - f\| < \varepsilon_f$ for all $n > N$, and exists an upper bound M s.t. $\|f_n\| \leq M$ for all $n \in \mathbb{N}$.

Then, consider $g : [-M, M]^{d_1} \to \mathbb{R}^{d_2}$. As g is continuous in a compact set, for all $\varepsilon_g > 0$, there exists a $\delta_g > 0$ such that $\|g(z_1) - g(z_2)\| < \varepsilon_g$ for all $\|z_1 - z_2\| < \delta_g$. Using $\varepsilon_f = \delta_g$, $\|f_n - f\| < \varepsilon_f = \delta_g$ for all $n > N$. Therefore, $\|g(f_n) - g(f)\| < \varepsilon_g$ for all $n > N$. □

Theorem 8. *Suppose that Assumption 2 holds and let $\langle x_k, y_k, u_k\rangle$ be a suboptimal solution obtained for $\mathcal{P}_{\mathcal{L}}(\mu_k, \nu_k)$ such that the violation of the optimality conditions are given by* (145)*, for which inequality* (145d) *is fundamental, where $0 \leq \varepsilon_k$, and $\varepsilon_k \to 0$ as $k \to \infty$, $\{\nu_k\}$ is a uniform convergent sequence, and λ_k is the costate at the k-th algorithm iteration. Assume that a subsequence $\{\langle x_k, y_k, u_k\rangle\}_K$ converges uniformly to $\langle x^*, y^*, u^*\rangle$ such that $\frac{\partial g}{\partial y}(x^*, y^*, u^*, t)$ has full rank and is bounded for all $t \in [t_0, t_f]$.*

Then the subsequence $\{\nu_k + \mu_k g(x_k, y_k, u_k, t)\}_K$ converges uniformly to $\widetilde{\nu}^*$, such that the following relations are obtained, with respect to y

$$\frac{\partial L}{\partial y}(x^*, y^*, u^*, t)^T + \frac{\partial f}{\partial y}(x^*, y^*, u^*, t)^T \lambda^* + \frac{\partial g}{\partial y}(x^*, y^*, u^*, t)^T \widetilde{\nu}^* = 0 \tag{146a}$$

and with respect to λ, u, and x are

$$-\dot{\lambda}^* = \frac{\partial L}{\partial x}(x^*, y^*, u^*, t)^T + \frac{\partial f}{\partial x}(x^*, y^*, u^*, t)^T \lambda^* + \frac{\partial g}{\partial x}(x^*, y^*, u^*, t)^T \widetilde{\nu}^* \tag{146b}$$

$$u^*(t) = \arg \inf_{u\in U_B} H(x^*(t), \lambda^*(t), y^*, \nu^*, u, t) \tag{146c}$$

$$\dot{x}^* = f(x^*, y^*, u^*, t). \tag{146d}$$

Proof. The derivative of $\mathcal{L}_{\mu_k}$ w.r.t. y results in

$$\frac{\partial \mathcal{L}_{\mu_k}}{\partial y}(x_k, y_k, u_k, \nu_k, t) = \frac{\partial L}{\partial y}(x_k, y_k, u_k, t) + [\nu_k + \mu_k g(x_k, y_k, u_k, t)]^T \frac{\partial g}{\partial y}(x_k, y_k, u_k, t) \tag{147}$$

Then, by defining for all k

$$\widetilde{\nu}_k = \nu_k + \mu_k g(x_k, y_k, u_k, t) \tag{148}$$

replacing $\widetilde{\nu}_k$ into (147) results in

$$\frac{\partial \mathcal{L}_{\mu_k}}{\partial y}(x_k, y_k, u_k, \nu_k, t) = \frac{\partial L}{\partial y}(x_k, y_k, u_k, t) + \widetilde{\nu}_k^T \frac{\partial g}{\partial y}(x_k, y_k, u_k, t). \tag{149}$$

Since $\frac{\partial g}{\partial y}$ is invertible, we can derive the following expression for $\widetilde{\nu}_k$,

$$\widetilde{\nu}_k = \left[\frac{\partial g}{\partial y}(x_k, y_k, u_k, t)^T\right]^{-1} \left[\frac{\partial \mathcal{L}_{\mu_k}}{\partial y}(x_k, y_k, u_k, \nu_k, t)^T - \frac{\partial L}{\partial y}(x_k, y_k, u_k, t)^T\right] \tag{150}$$

From (150) we can say that there exists an F such that

$$\widetilde{\nu}_k = F(x_k, y_k, u_k, \nu_k) \tag{151}$$

which is continuous since all the functions in (150) are continuous. Given that a subsequence $\{\langle x_k, y_k, u_k\rangle\}_K$ converges to $\langle x^*, y^*, u^*\rangle$ and $\{\nu_k\}$ converges to ν^*, Lemma 1 is invoked to conclude that

$$\{\widetilde{\nu}_k = F(x_k, y_k, u_k, \nu_k)\}_K \to \widetilde{\nu}^* = F(x^*, y^*, u^*, \nu^*) \tag{152}$$

which shows that $\{\nu_k + \mu_k g(x_k, y_k, u_k, t)\}_K \to \widetilde{\nu}^*$ uniformly, and $\widetilde{\nu}^*$ is given by

$$\widetilde{\nu}^* = \left[\frac{\partial g}{\partial y}(x^*, y^*, u^*, t)^T\right]^{-1} \left[\frac{\partial \mathcal{L}_{\mu^*}}{\partial y}(x^*, y^*, u^*, \nu^*, t)^T - \frac{\partial L}{\partial y}(x^*, y^*, u^*, t)^T\right]. \tag{153}$$

Considering the optimality conditions for y, given in (145d), and taking the limit $k \to \infty$, we obtain

$$\frac{\partial \mathcal{L}_{\mu^*}}{\partial y}(x^*, y^*, u^*, \nu^*, t) = -\lambda^{*T} \frac{\partial f}{\partial y}(x^*, y^*, u^*, t) \tag{154}$$

which can be substituted into (153) to obtain

$$\widetilde{\nu}^* = \left[\frac{\partial g}{\partial y}(x^*, y^*, u^*, t)^T\right]^{-1} \left[-\frac{\partial L}{\partial y}(x^*, y^*, u^*, t)^T - \frac{\partial f}{\partial y}(x^*, y^*, u^*, t)^T \lambda^*\right] \tag{155}$$

which can be rearranged into

$$\frac{\partial L}{\partial y}(x^*, y^*, u^*, t) + \lambda^{*T} \frac{\partial f}{\partial y}(x^*, y^*, u^*, t) + \widetilde{\nu}^{*T} \frac{\partial g}{\partial y}(x^*, y^*, u^*, t) = 0 \tag{156}$$

and related to the necessary conditions developed in Theorem 4, applying to the original OCP $\mathcal{P}$. A similar approach can be used to obtain the conditions for x, u, and λ.

Since the sequence $\{\nu_k\}$ is bounded and $\{\nu_k + \mu_k g(x_k, y_k, u_k, t)\}_K \to \widetilde{\nu}^*$ from (152), it follows that $\{\mu_k g(x_k, y_k, u_k, t)\}_K$ is bounded. Given that $\mu_k \to \infty$ we must have $g(x_k, y_k, u_k, t) \to 0$ with $g(x^*, y^*, u^*, t) = 0$ for all t. □

Notice that the sequence $\{\nu_k\}$ was never specified, other than it is a uniformly convergent sequence. From Theorem 8, an update rule can be derived such that $\{\nu_k\} \to \widetilde{\nu}^*$.

Corollary 1. *By defining* $\nu_{k+1} = \nu_k + \mu_k g(x_k, y_k, u_k, t)$ *we have that* $\{\nu_k\} \to \widetilde{\nu}^*$ *and* $\{\mu_k g(x_k, y_k, u_k, t)\} \to 0$.

Proof. For any uniformly convergent sequence $\{\nu_k\}$, Theorem 8 ensures that $\{\nu_k + \mu_k g(x_k, y_k, u_k, t)\} \to \widetilde{\nu}^*$. Therefore, we can define $\nu_{k+1} = \nu_k + \mu_k g(x_k, y_k, u_k, t)$, which makes the sequence become $\{\nu_{k+1}\} \to \widetilde{\nu}^*$. □

2.6 APPLICATION

The algorithm is applied to two optimal control problems to illustrate the algorithm behavior and remark some implementation details. The first problem is the stabilization of the Van der Pol Oscillator. The second scenario is the control of the four tanks system. Both are popular benchmark systems.

2.6.1 Van der Pol Oscillator

To illustrate the algorithm behavior and to remark some implementation details, the algorithm is applied to the optimal control problem of stabilizing the Van der Pol oscillator (KHALIL, 2002), which is nonlinear and has an attractive limit cycle. These features render the oscillator a widely used benchmark for the control of nonlinear systems.

The Van der Pol oscillator is typically modeled in the form of an ODE system as

$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u \tag{157a}$$

$$\dot{x}_2 = x_1 \tag{157b}$$

For the purpose of our analysis, the ODE system (157) is remodeled as a DAE system,

$$\dot{x}_1 = y + u \tag{158a}$$

$$\dot{x}_2 = x_1 \tag{158b}$$

$$y = (1 - x_2^2)x_1 - x_2 \tag{158c}$$

With the objective of keeping the system at the unstable equilibrium $(0, 0)$, the following objective is chosen

$$J(x, y, u) = \int_{t_0}^{t_f} \left[x_1^2 + x_2^2 + u^2 \right] \, dt \tag{159}$$

Let us define an optimal control problem that minimizes the functional J (159), while being subject to the DAE system (158).

$$\min_{x,y,u} \; J = \int_{t_0}^{t_f} \left[x_1^2 + x_2^2 + u^2 \right] \, dt \tag{160a}$$

$$\text{s.t.:} \quad \dot{x}_1 = y + u \tag{160b}$$

$$\dot{x}_2 = x_1 \tag{160c}$$

$$y = (1 - x_2^2)x_1 - x_2 \tag{160d}$$

$$x(0) = x_0, \qquad t \in [t_0, t_f] \tag{160e}$$

where $x_0 = [0, 1]$, $t_0 = 0$, and $t_f = 5$.

To investigate the properties of the proposed algorithm, two cases are considered:

- Case 1, the OCP (160) is solved as stated.
- Case 2, solves (160) subject to the bound constraints: $-0.3 \leq u(t) \leq 1$ on the control variables, and the constraint $-0.4 \leq x_1(t)$ on the state x_1.

Case 2 highlights an interesting practical property of the algorithm; the algorithm allows state constraints to be easily expressed and solved. Solving OCP with state constraints is not easily achieved when using indirect methods. The algorithm's underlying OCP is solved with the indirect method applying multiple shooting for both cases.

Case 1: Unconstrained OCP

To apply the algorithm for Case 1, the algebraic equation (160) needs to be relaxed, and a the new augmented cost function needs to be defined

$$\mathcal{L}_\mu = \left(x_1^2 + x_2^2 + u^2\right) + \nu\left[(1-x_2^2)x_1 - x_2 - y\right] + \frac{\mu}{2}\left\|(1-x_2^2)x_1 - x_2 - y\right\|^2 \quad (161)$$

which allows us to formulate the auxiliary problem

$$\begin{aligned}
\mathcal{P}_{\mathcal{L}}(\mu_k, \nu_k): \quad \min\ & J_{\mu_k} = \int_{t_0}^{t_f} \mathcal{L}_{\mu_k}(x, y, u, t)\, dt && (162a)\\
\text{s.t.:}\ & \dot{x}_1 = y + u && (162b)\\
& \dot{x}_2 = x_1 && (162c)\\
& x(0) = x_0 && (162d)\\
& x(t) \in X,\ y(t) \in Y,\ u(t) \in U && (162e)\\
& t \in [t_0, t_f] && (162f)
\end{aligned}$$

The update of ν_k and μ_k are achieved through

$$\begin{aligned}
\nu_{k+1} &= \nu_k + \mu_k\left[(1-x_2^2)x_1 - x_2 - y\right] && (163a)\\
\mu_{k+1} &= \beta\mu_k && (163b)
\end{aligned}$$

with the parameters $\beta = 8$, $\mu_0 = 2$, and $\nu_0 = 0$ for all $t \in [t_0, t_f]$.

A solution for (162) can be achieved by applying the indirect method. In order to find the optimal solution, let us establish the Hamiltonian of the auxiliary problem

$$\begin{aligned}
H_k = \left(x_1^2 + x_2^2 + u^2\right) + \nu_k\left[(1-x_2^2)x_1 - x_2 - y\right] \\
+ \frac{\mu_k}{2}\left\|(1-x_2^2)x_1 - x_2 - y\right\|^2 + \lambda_1(y+u) + \lambda_2 x_1 \quad (164)
\end{aligned}$$

By defining $\hat{u} = [u, y] \in X = U \times Y$ and using the conditions of Theorem 2, the optimality conditions for OCP of ODEs (162) can be expressed as

$$\dot{x}_1^* = y^* + u^*, \qquad \dot{x}_2^* = x_1^* \tag{165a}$$

$$-\dot{\lambda}_1^* = (2x_1^*) + \left[\nu_k + \mu_k[(1 - x_2^{*2})x_1^* - x_2^* - y^*]\right](1 - x_2^{*2}) + \lambda_2^* \tag{165b}$$

$$-\dot{\lambda}_2^* = (2x_2^*) + \left[\nu_k + \mu_k[(1 - x_2^{*2})x_1^* - x_2^* - y^*]\right](-2x_1^* x_2^* - 1) \tag{165c}$$

$$0 = 2u^* + \lambda_1^*, \tag{165d}$$

$$0 = \nu_k(-1) + \mu_k[(1 - x_2^{*2})x_1^* - x_2^* - y^*](-1) + \lambda_1^* \tag{165e}$$

and the boundary conditions $x^*(t_0) = x_0$ and $\lambda_1^*(t_f) = \lambda_2^*(t_f) = 0$.

From (165), the extended controls that mininize (162) can be deduced to be

$$u^* = -\frac{\lambda_1^*}{2}, \qquad y^* = -\frac{\nu}{\mu} - (1 - x_2^{*2})x_1^* - x_2^* + \frac{\lambda_1^*}{\mu} \tag{166a}$$

The combination of the optimal extended controls (166a) and the differential equations (165a-165d) compose the system of equations that need to be solved sequentially in order to obtain a solution to the original problem (160). Let the $F(x_{init}, \lambda_{init})$, be the function that solves the IVP for $t \in [0, \frac{t_f - t_0}{N_i}]$. Then an optimization problem for the indirect multiple shooting can be expressed as

$$\min_{x_i, \lambda_i} 0 \tag{167a}$$

$$\text{s.t.:} \begin{bmatrix} x_{i+1} \\ \lambda_{i+1} \end{bmatrix} = F(x_{0,i}, \lambda_{0,i}) \tag{167b}$$

$$x_{0,1} = x_0 \tag{167c}$$

$$\lambda_{f,N_i} = \lambda_f = 0 \tag{167d}$$

which can be solved with a proper nonlinear optimization solver, as for instance Ipopt (WÄCHTER; BIEGLER, L. T., 2006). To easily solve the IVP and obtain the sensitivity analysis, the solution can be implemented in CasADi framework (ANDERSSON et al., 2019).

By using the number of subintervals $N = 40$ and a third-order polynomial to describe ν_k, the solution displayed in Figures 6 and 7 is achieved. The controls and state trajectories obtained using the method achieved the desired goal of stabilizing the VDP oscillator. The results of the OCP solution using the augmented Lagrangian are very similar to those obtained using the indirect multiple-shooting.

Case 2: Constrained OCP

In Case 2, the OCP has two constraints, one in the control variables ($-0.3 \leq u(t) \leq 1$), and one in the state variables ($-0.4 \leq x_1(t)$). The first constraint can be held

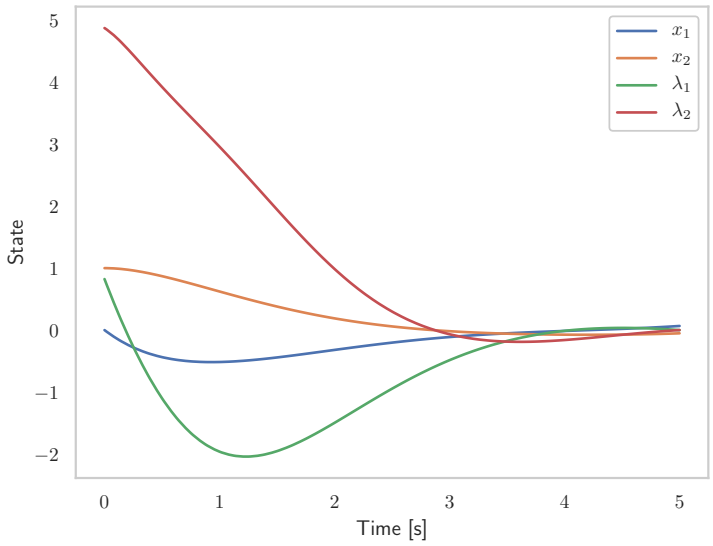


Figure 6 – Optimal state trajectories for the unconstrained stabilization of the VDP oscillator

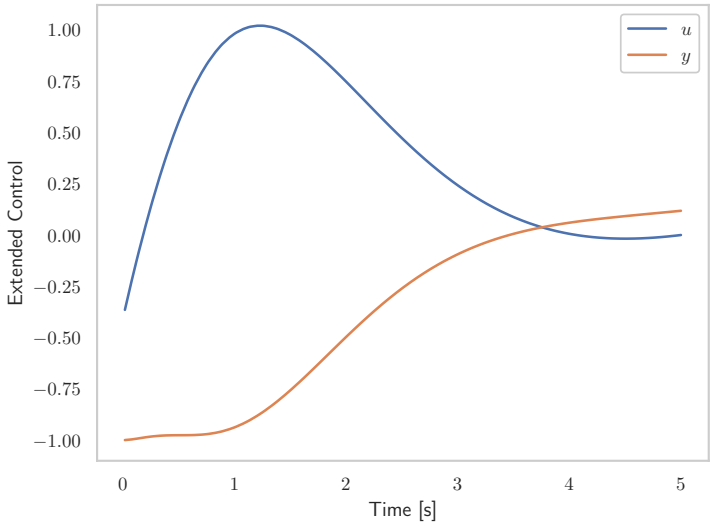


Figure 7 – Optimal control for the unconstrained stabilization of the VDP oscillator

by satisfying Pontryagin's minimum principle, which imposes new necessary conditions into the problem. To handle the state constraint, however, a more substantial change in the problem structure is required.

Handling state constraints for indirect methods is difficult, but handling control constraints can be handled by Pontryagin's condition. Furthermore, by relaxing the algebraic equations, the proposed augmented Lagrangian algorithm transforms the algebraic variables into control variables. This transformation by itself allows for enforcing constraints into the algebraic variables. But by introducing a new algebraic variable

$$y_{x_1} = x_1 \tag{168}$$

the value of state x_1 is bound to an algebraic variable.

The OCP with the new variable can then have its algebraic equations relaxed, with the objective function

$$\mathcal{L}_\mu = \left(x_1^2 + x_2^2 + u^2\right) + \nu_1 \left[(1-x_2^2)x_1 - x_2 - y\right] + \nu_2 \left[x_1 - y_{x_1}\right] + \frac{\mu}{2}\left\|(1-x_2^2)x_1 - x_2 - y\right\|^2 + \frac{\mu}{2}\left\|x_1 - y_{x_1}\right\|^2 \quad (169)$$

where

$$g(x_1, x_2, y, y_{x_1}) = \begin{bmatrix} (1-x_2^2)x_1 - x_2 - y \\ x_1 - y_{x_1} \end{bmatrix} \quad (170)$$

The Hamiltonian for the each iteration of the relaxed OCP then becomes

$$H_k = \left(x_1^2 + x_2^2 + u^2\right) + \nu_{1,k} \left[(1-x_2^2)x_1 - x_2 - y\right] + \nu_{2,k} \left[x_1 - y_{x_1}\right] + \frac{\mu_k}{2}\left\|(1-x_2^2)x_1 - x_2 - y\right\|^2 + \frac{\mu_k}{2}\left\|x_1 - y_{x_1}\right\|^2 + \lambda_1 (y+u) + \lambda_2 x_1 \quad (171)$$

By applying the conditions of Pontryagin's minimum principle (Theorem 3), with the Hamiltonian (171), the following conditions are obtained

$$\dot{x}_1^* = y^* + u^*, \qquad \dot{x}_2^* = x_1^* \quad (172a)$$

$$-\dot{\lambda}_1^* = (2x_1^*) + \left[\nu_{1,k} + \mu_k[(1-x_2^{*2})x_1^* - x_2^* - y^*]\right](1-x_2^{*2}) + \left[\nu_{2,k} + \mu_k(x_1 - y_{x_1})\right] + \lambda_2^* \quad (172b)$$

$$-\dot{\lambda}_2^* = (2x_2^*) + \left[\nu_{1,k} + \mu_k[(1-x_2^{*2})x_1^* - x_2^* - y^*]\right](-2x_1^* x_2^* - 1) \quad (172c)$$

$$u^* = \underset{-0.3\le u\le 1}{\arg\min}\, H(x_1^*, x_2^*, \lambda_1^*, \lambda_2^*, y^*, y_{x_1}^*, u), \quad (172d)$$

$$0 = \nu_{1,k}(-1) + \mu_k[(1-x_2^{*2})x_1^* - x_2^* - y^*](-1) + \lambda_1^* \quad (172e)$$

$$y_{x_1} = \underset{-0.4\le y_{x_1}}{\arg\min}\, H(x_1^*, x_2^*, \lambda_1^*, \lambda_2^*, y^*, y_{x_1}, u^*) \quad (172f)$$

The optimal equation for y^* can be obtained by isolating the variable in equation (172e),

$$y^* = -\frac{\nu_{1,k}(-1) - \mu_k[(1-x_2^{*2})x_1^* - x_2^*] + \lambda_1^*}{\mu_k} \quad (173)$$

To obtain the equations that define the optimal laws for the free variables u and y_{x_1}, the minimization of the Hamiltonian needs to solved. A necessary condition for

a minimum is that its derivative at the minimum is zero. Therefore, if there was no constraints on u and y_{x_1}, the minimum would be attained at

$$u^* = -\frac{\lambda_1^*}{2} \tag{174a}$$

$$y_{x_1}^* = -\frac{\nu_{2,k}(-1) - \mu_k x_1}{\mu_k} \tag{174b}$$

To enforce the bound constraints into this variables, a feasibility projection can be used. That is, project the variables into the feasible set when they violate the bounds.

$$u^* = \text{mid}\left(-0.3, -\frac{\lambda_1^*}{2}, 1\right) \tag{175a}$$

$$y_{x_1}^* = \max\left(-0.4, -\frac{\nu_{2,k}(-1) - \mu_k x_1}{\mu_k}\right) \tag{175b}$$

where mid is a function that the return the intermediary value between its three arguments, while max is the function that returns the maximum value between its two arguments.

Using the equations (173)-(174) that define the optimum values for u^*, y^* and y_{x_1}, and equations (172a-172c), a nonlinear optimization problem can be formulated based on the multiple shooting. Just like it was formulated for Case 1. By doing so, the optimal profiles for the states, algebraic, and control variables are obtained, as depicted in Figure 8 and 9. The optimal profiles obtained with the augmented Lagrangian method follow closely the profiles obtained by solving the original problem (160) with an indirect multiple-shooting method. As can be seen in the figures, the introduced algebraic variable y_{x_1}, mimics the dynamics of the state x_1, and ensures that the constraint $-0.4 \leq x_1$ is satisfied. Most likely due to the constraints on x_1, the bound on u are also satisfied.

2.6.2 Four Tanks

The computation experiment presented here was originally developed for (AGUIAR et al., 2021), where the algorithm was applied to find the optimal controls that stabilize the four-tank system, which is portrayed in Figure 10.

The four-tank system is typically modeled using an ODE system (JOHANSSON, 2000), here the model is represented as a DAE system. A DAE representation is obtained by using algebraic variables to describe the outflow of each tank.

For each tank i, the outflow is discribed by

$$q_{t,i} = a_i\sqrt{2gh_i}, \tag{176}$$

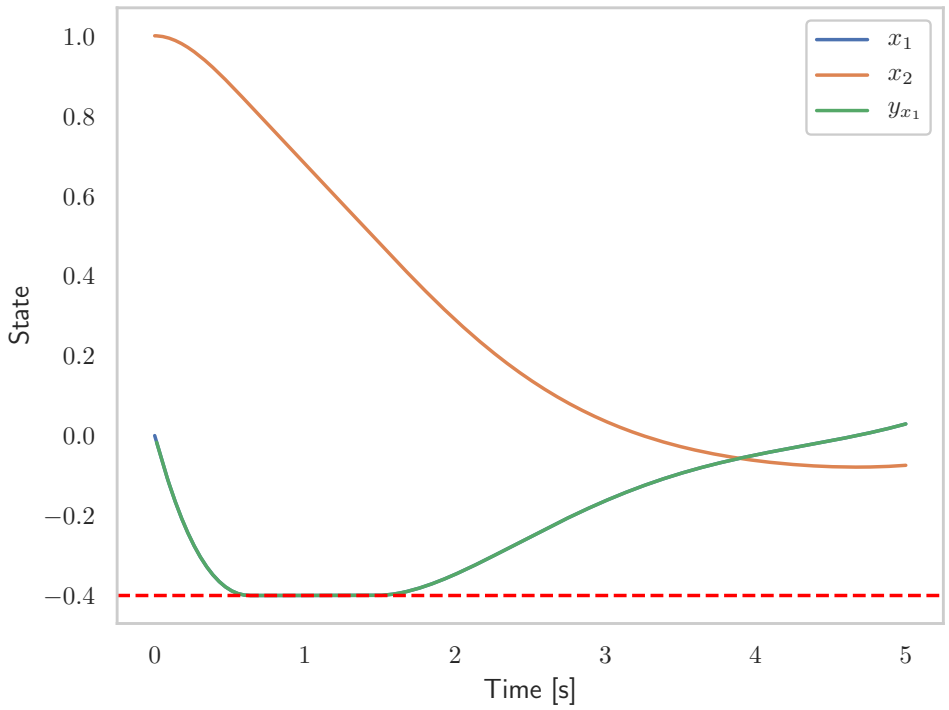


Figure 8 – Optimal state trajectories for the constrained stabilization of the VDP oscillator, with the introduced algebraic variable that was relaxed

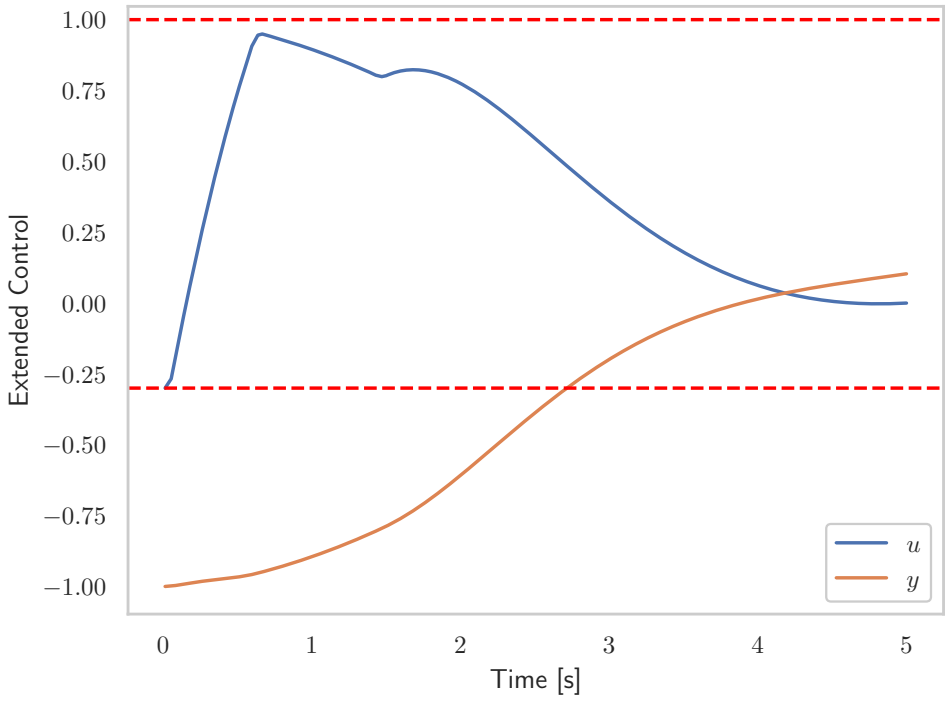


Figure 9 – Optimal control and the relaxed algebraic variable for the constrained stabilization of the VDP oscillator

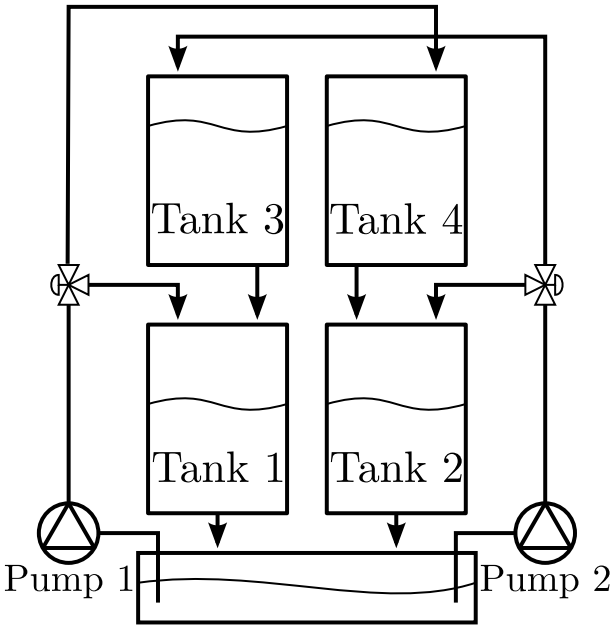


Figure 10 – Illustration of the four tank system

where a_i is the cross section area of the orifice, g is the gravity constant, and h_i is the fluid level of tank i. Equations that define the dynamics for the level of each tank are

$$\dot{h}_1 = \frac{q_{t,3} + \gamma_1 q_{p,1} - q_{t,1}}{A_1}, \tag{177a}$$

$$\dot{h}_2 = \frac{q_{t,4} + \gamma_2 q_{p,2} - q_{t,2}}{A_2}, \tag{177b}$$

$$\dot{h}_3 = \frac{(1-\gamma_2) q_{p,2} - q_{t,3}}{A_3}, \tag{177c}$$

$$\dot{h}_4 = \frac{(1-\gamma_1) q_{p,1} - q_{t,4}}{A_4}, \tag{177d}$$

where A_i is the cross section area of the i-th tank, γ_j is the split ratio on the three-way valves, and the flow pump j is given by the differential equation

$$\dot{q}_{p,j} = \delta_j. \tag{178}$$

where δ_j is the variation on the flow-rate.

The objective of the optimal control problem is to stabilize the tanks 1 and 2 at a given setpoint, while keeping the variation in the pump at a minimum, which is expressed by the following objective

$$\min_u J = \int_{t_0}^{t_f} \Delta x^T \Delta x + u^T u \, dt \tag{179}$$

with $\Delta x = x - x_{ref}$, $x = [h_1, h_2, h_3, h_4, q_{p,1}, q_{p,2}]$, and $u = [\delta_1, \delta_2]$.

Putting the system equations and the objective together, the following optimal

control problem can be formulated

$$\min_u J = \int_{t_0}^{t_f} \Delta x^T \Delta x + u^T u \, dt \tag{180a}$$

$$\text{s.t.: } \dot{h}_1 = \frac{q_{t,3} + \gamma_1 q_{p,1} - q_{t,1}}{A_1}, \tag{180b}$$

$$\dot{h}_2 = \frac{q_{t,4} + \gamma_2 q_{p,2} - q_{t,2}}{A_2} \tag{180c}$$

$$\dot{h}_3 = \frac{(1-\gamma_2) q_{p,2} - q_{t,3}}{A_3}, \tag{180d}$$

$$\dot{h}_4 = \frac{(1-\gamma_1) q_{p,1} - q_{t,4}}{A_4} \tag{180e}$$

$$\dot{q}_{p,1} = \delta_1 \tag{180f}$$

$$\dot{q}_{p,2} = \delta_2 \tag{180g}$$

$$\text{for } i \in \{1, \ldots, 4\} :$$

$$q_{t,i} = a_i \sqrt{2gh_i} \tag{180h}$$

To use the algorithm, the algebraic equation (180h) is relaxed, which ressults in the following auxiliary problem

$$\min_{u,y} \quad J_{\mu_k}, \tag{181a}$$

$$\text{s.t.: } \dot{h}_1 = \frac{q_{t,3} + \gamma_1 q_{p,1} - q_{t,1}}{A_1}, \tag{181b}$$

$$\dot{h}_2 = \frac{q_{t,4} + \gamma_2 q_{p,2} - q_{t,2}}{A_2} \tag{181c}$$

$$\dot{h}_3 = \frac{(1-\gamma_2) q_{p,2} - q_{t,3}}{A_3}, \tag{181d}$$

$$\dot{h}_4 = \frac{(1-\gamma_1) q_{p,1} - q_{t,4}}{A_4} \tag{181e}$$

$$\dot{q}_{p,1} = \delta_1 \tag{181f}$$

$$\dot{q}_{p,2} = \delta_2 \tag{181g}$$

where

$$J_{\mu_k} = \int_{t_0}^{t_f} \Delta x^T \Delta x + u^T u + \sum_{i=1}^{4} \left[\nu_{i,k} \left(q_{t,i} - a_i \sqrt{2gh_i} \right) + \frac{\mu_k}{2} \left\| q_{t,i} - a_i \sqrt{2gh_i} \right\|^2 \right] dt \tag{182}$$

The auxiliary problem (181) is solved at each algorithm iteration. The optimal solution is used to compute the new multiplier estimates $\nu_{i,k+1}$ using the update rule

$$\nu_{i,k+1} = \nu_{i,k} + \mu_k \left[q_{t,i} - a_i \sqrt{2gh_i} \right] \tag{183}$$

the multiplier penalty is them updated using

$$\mu_{k+1} = \beta \mu_k \tag{184}$$

As discussed in (AGUIAR et al., 2016), since $\nu_{i,k}$ is a function that can assume any shape, a piecewise polynomial approximation with a finite number of terms is used instead. For this application, a piecewise Lagrangian polynomial (similar to those used in the collocation method) was chosen to facilitate the computation of updates.

An indirect collocation method with order three polynomials was used to solve the relaxed subproblem at each iteration. The indirect collocation used 40 finite elements and was implemented using YAOCPTool, and CasADi (ANDERSSON et al., 2019). The same settings were used to solve the original problem (180) for comparison purposes.

The nonlinear programming problems obtained from the discretization process were solved using the IPOPT solver (WÄCHTER; BIEGLER, L. T., 2006). By using indirect methods to solve the original problem, the multipliers are computed automatically. The multipliers from the original problem can be compared to the estimated ν_i obtained by the algorithm. The multiplier estimates ν_i are also approximated with a piecewise polynomial of degree 3 with 40 finite elements.

Given that no prior information is available for the multipliers, the algorithm is initialized with the multiplier estimates as zero ($\nu_{i,0} = 0$ for all $t \in [t_0, t_f]$ and all $i \in N_i$). The penalization term starts with $\mu_0 = 0.1$ and increases at a rate $\beta = 4$.

The trajectories of the proposed algorithm coincide with the optimal trajectories obtained by the indirect method, which are shown in Figures 11 and 12. To evaluate if the algorithm is converging to the optimal solution of the original problem, in Fig. 13, the relaxed objective (J_{μ_k}) and the evaluation of the solution iteration on the original objective (J_k) are compared to the optimal cost obtained with the indirect method (J^*). It can be seen that the objectives converge to the same objective value J^* as the indirect method. As for the violation of algebraic equations, the line in blue of Fig. 14 shows the violation rapidly converging to zero; the line in red shows the norm of the difference between the multiplier obtained with the indirect method and the multiplier estimate computed by the proposed algorithm, which decreases as the algorithm iterates.

An experiment was performed using the proposed algorithm with direct multiple shooting to solve the subproblems (AL-DMS). To solve the ODE of the subproblems, we apply Sundials CVODES and a 4^{th} order Runge-Kutta method (RK4). The proposed algorithm was compared against DMS applied directly to the original OCP, whereby Sundials IDAS was used to solve the DAE. Table 2 presents the results that indicate faster convergence of the proposed algorithm with RK4, which might be more suitable for embedded applications with limited computational power.

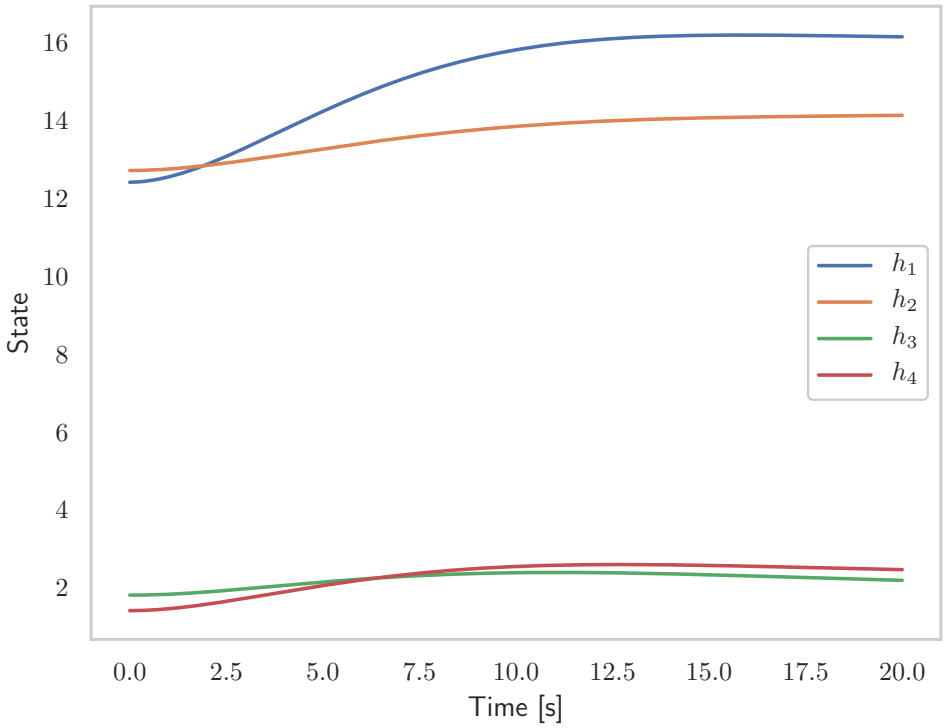


Figure 11 – Optimal state trajectories for the stabilization of the four-tank system obtained with the proposed augmented Lagrangian method and which coincide with the trajectories obtained with the trajectories obtained with indirect multiple shooting of the baseline OCP

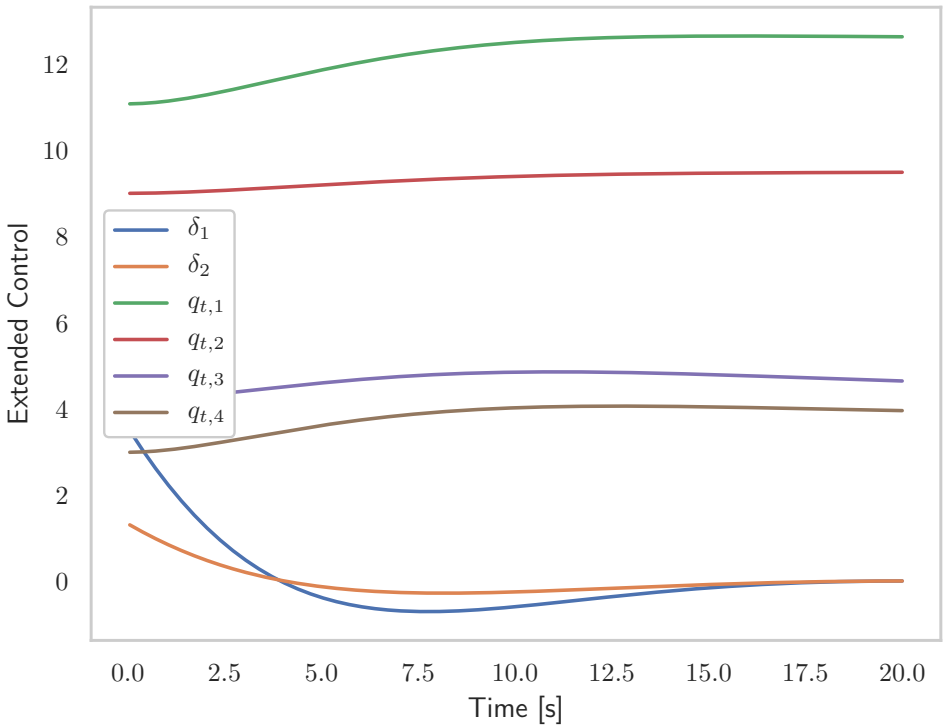


Figure 12 – Optimal control for the constrained stabilization of the four-tank system

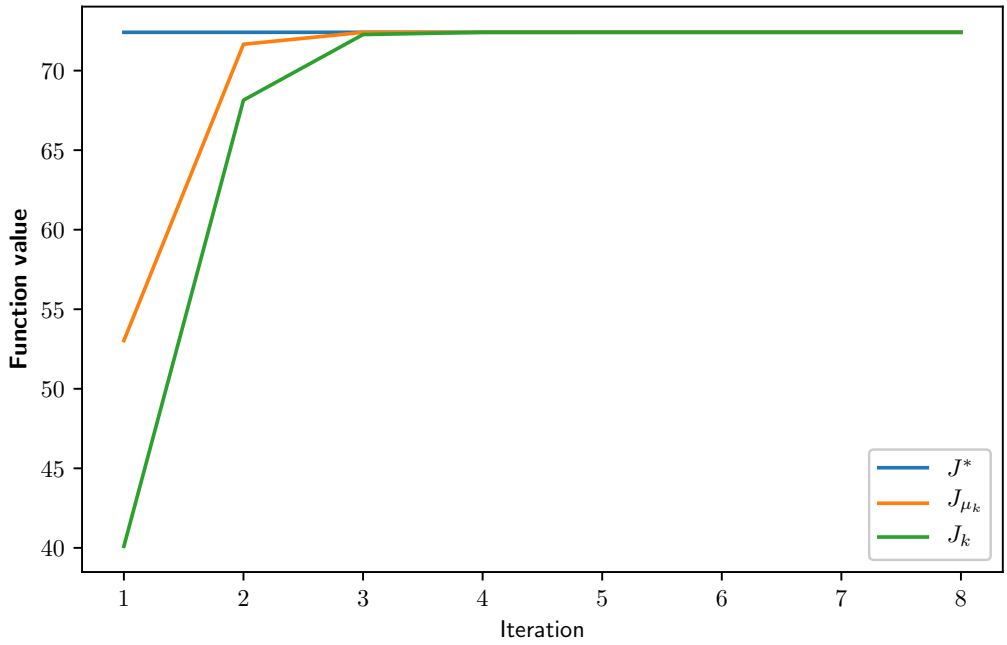


Figure 13 – Comparison of the objective functions.

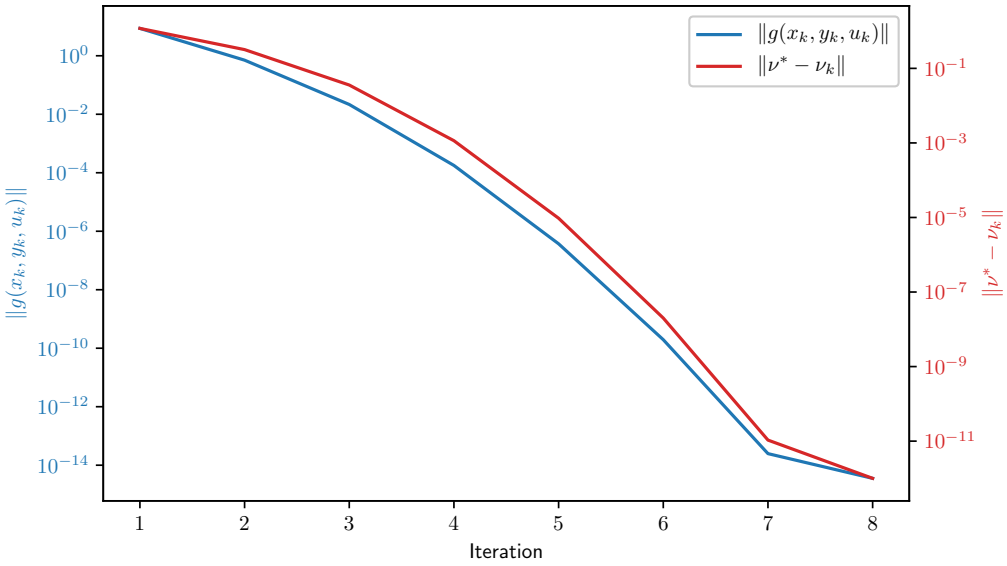


Figure 14 – Convergence of the algebraic function to zero (blue) and the multiplier estimate converging to the original problem multiplier (red).

Table 2 – Algorithm Solving Time (in seconds)

AL-DMS (CVODES)	AL-DMS (RK4)	DMS (IDAS)
24.15	0.11	2.42

3 DISTRIBUTED OPTIMAL CONTROL

3.1 LITERATURE REVIEW

3.1.1 Distributed Dynamic Systems

Distributed dynamic systems, also known as networked dynamic systems or multi-agent systems, are systems composed of multiple subsystems, each having dynamics or resources that couple them to the others.

Distributed systems can be classified concerning a wide range of characteristics, for instance, according to their subsystems' similarity. Distributed systems appear in several fields, with different compositions of subsystems, for example:

- In urban traffic control, each traffic junction can be represented as a subsystem (CAMPONOGARA et al., 2017a; DE OLIVEIRA; CAMPONOGARA, 2010). Considering that the queues of cars in front of the semaphoric lights are the states, then the streets between two junctions connect these subsystems. Streets and junctions are an excellent example of a dynamic network with systems of the exact nature, referred to as *homogeneous* networks.
- An offshore oil production platform comprises various components: production wells, pipelines, risers, separators, compressors, and gas-lift injection lines (AGUIAR et al., 2015). This system has different sorts of subsystems and can be named a *heterogeneous* network.

Besides classifying a distributed system regarding its subsystems, a system can be classified according to its structure. For instance:

- When the network is composed of many subsystems connected to a single node, we call it a "star" configuration, as depicted in Figure 15. This type of system might occur, for example, when there are one power generator and several consumers.
- In the offshore oil production platform, the system is configured as a "tree". One layer has several nodes that connect to a node in a higher layer. Those nodes in the higher layer also connect to a node in an even higher layer, and so on. This structure can be seen in Figure 15.
- Another type of formation that might occur is the cycle configuration, where every subsystem is connected to two neighbors, as shown in Figure 15.
- Very similar to the cycle configuration is the small-world configuration. This configuration resembles the popular thinking that every person in the world is, on average, six persons of distance from one another. However, with globalization,

such a number is probably down to 4. For this configuration, every node is connected to two neighbors, to the neighbors of these neighbors, and occasionally to some other node in the network. Figure 15 illustrates this kind of network.

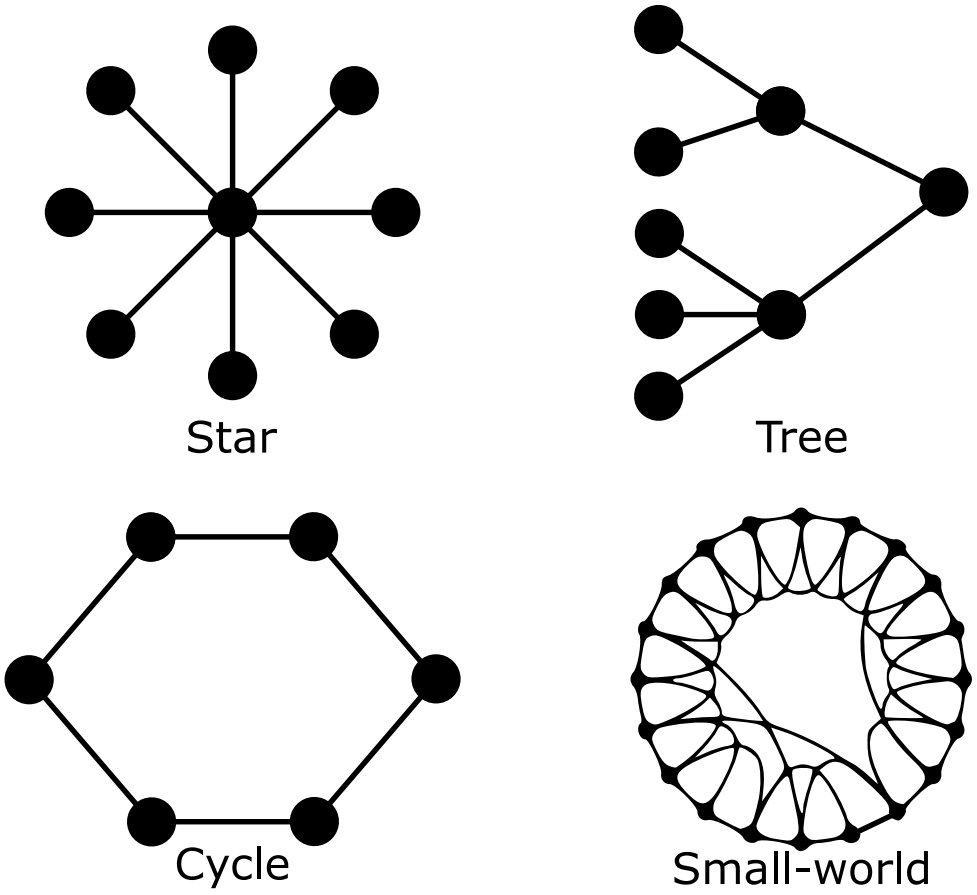


Figure 15 – Different topologies for dynamic networks

For many applications, the resulting networks do not fully suit one of the groups above or any other category for that matter. Nevertheless, frequently those systems materialize as an amalgam of different known structures.

In the following, let us see how the four tank system is modeled as a dynamic network.

Example 4 (The four-tank system)**.** *The four-tank system is a classical benchmark system, which has been used for several purposes (JOHANSSON, 2000). Even though it has relatively simple internal dynamics, its topology makes it an interesting control problem.*

The system is composed of four tanks, two pumps, and two three-way valves. The first pump fills Tanks 1 and 4, according to the settings of the first three-way valve. The second pump fills Tanks 2 and 3, according to the second three-way valve. As Tank 3 is on top of Tank 1, its outflow goes into Tank 1. Likewise, as Tank 4 is on top of Tank 2, its outflow goes to Tank 2. The system setup is illustrated in Figure 16.

In Example 1, a model for a single tank was developed using DAE equations. For that tank, the exogenous variables were the inflow q_{in}, and the output was the outflow q_{out}. In the four-tank problem, the connections between these subsystems are dictated by where the entering flows come from and where the exiting flows go.

If we consider each subsystem as a node and each connection as an arc, then a distributed system can be represented by a directed graph. Figure 17 shows the four tank system represented as a graph, where the nodes P_1 and P_2 represent the pumps and valves; T_1, T_2, T_3, and T_4 represent the tanks.

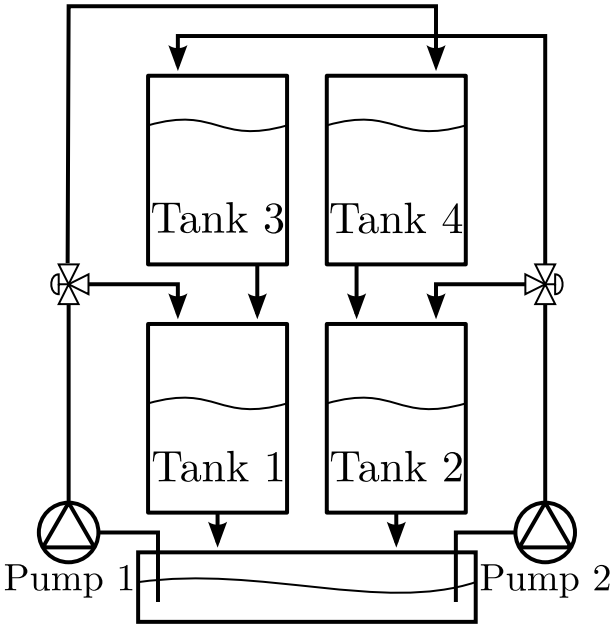


Figure 16 – Illustration of the four-tank system

The example above shows how the four tank system, one of the most common benchmark systems, can be modeled as a distributed system. In fact, many systems that are often tackled as a monolithic structure have a distributed nature, either for being geographically sparse or having loosely coupled dynamics.

In the sequence, we will be discussing how to develop controllers that can take advantage of the sparse structure of distributed systems.

3.1.2 Controlling Distributed Systems

Various control architectures have been proposed for distributed systems for which different properties have been developed, especially regarding convergence and stability. Here we do not discuss these methods but rather classify these methods using a classification proposed by Venkat et al. (2005), and Rawlings and Stewart (2008). By using this classification, we can understand what each method can accomplish and by what means.

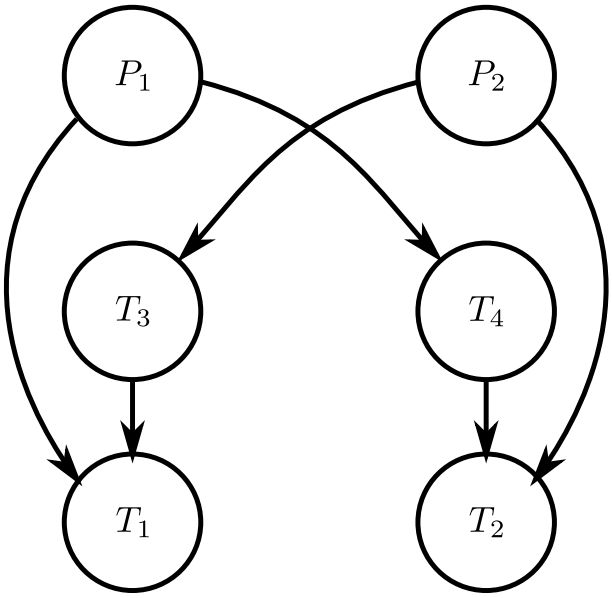


Figure 17 – Four tank system represented as a dynamic network

The standard approach to control a distributed system is to consider the system as a monolithic system and use a centralized controller. The upside is that there is no need to develop a communication framework, given that a single controller determines all the decisions. Another positive factor is that the solution of the optimal control problem is optimal, at least for the cases that the model reflects the dynamics of the plant flawlessly.

However, according to Venkat et al. (2005), for large networked systems, the centralized MPC is too impractical and inflexible, rendering it unsuitable for such applications. Often the problem of developing a centralized controller is not computational but rather organizational. When different subsystems are developed and managed by different teams, it becomes challenging to agree on a single monolithic and inflexible centralized MPC. Therefore, unless these barriers can be overcome, centralized controllers serve more as a benchmark than a *de-facto* practice.

At first, the centralized approach might seem the most common technique; however, decentralized approaches are even more prevalent. In the decentralized approach, the local controllers can be single-input single-output (like PIDs), or multivariable (locally centralized), but overall have a decentralized structure (SCATTOLINI, 2009). These decentralized configurations benefit from being easy to implement, being flexible, and having no communication or interoperability issues. On the other hand, a decentralized configuration can have some drawbacks; for instance, the control actions of one system might become perturbations to another. Those interactions can either be direct: the input of one system affects the other; or indirect: the states of one subsystem affect another. Anyhow, this interaction can cause poor performance or even make the process unstable. As Scattolini (2009) points out, some methods for decentralized control that

have stability guarantees have been proposed, however not many have been proposed thus far. This lack of popularity has distinct reasons; one of them is that the nature of MPC allows to simply couple the system into a centralized MPC. Also, the proof of stability for MPCs typically comes from finding an implicit law for the controllers and showing that the cost is a Lyapunov function, which is not easily achieved when there are several coupled systems.

Another approach for controlling a distributed system is a distributed controller. This class of controllers can be separated into two groups (RAWLINGS; STEWART, 2008):

- Communication-based distributed control,
- Cooperative distributed control.

The communication-based controller is the next step of a decentralized controller in the direction of a centralized controller. The typical procedure for communication-based controllers is that the local controller calculates the control actions for that sampling time; afterward, it informs the neighbors about its future actions. Each subsystem has a model of how the controls of the neighboring subsystems affect their controlled variables and local objectives. Such an approach can be understood as the competition among the controllers, which falls in the non-cooperative game theory. The centralized controller converges to a Pareto optimal solution when considering the same weight to all local objective functions (RAWLINGS; STEWART, 2008). In contrast, controllers under a non-cooperative game converge to the Nash equilibrium, which might be unstable.

Rawlings and Stewart (2008) show an example with three variants. Their results are summarized in Figure 18. In the first case, the communication-based controller converges to a Nash equilibrium close to the Pareto solution. In this case, the solution of both approaches are similar, and the communication-based controller might result in an excellent closed-loop response. In the second case, the objective function is modified. The result is that the communication-based method converges to a Nash equilibrium far from the Pareto set, the closed-loop implementation might be unstable. In the third case, despite the Nash equilibrium being close to the Pareto optimal, the Nash equilibrium is unstable. Therefore the communication-based controller diverges from the Nash equilibrium to a point on the boundary of the feasible region. The closed-loop system, most likely, is unstable.

To synthesize, the problem of the communication-based controller is that despite knowing how its neighbors are going to affect their local objective, there is no negotiation between the controllers of each subsystem. For instance, if one of the subsystems is close to losing controllability, it can not “ask for help” to its neighbors. At the same time,

neighbors are not pro-actively trying to help because they only have information about their objectives.

In order to circumvent the issues of a communication-based controller, one can use a cooperative distributed controller, which was first proposed by Venkat et al. (2005). The most significant advantage of this approach is that it can overcome the problems that the decentralized and the communication-based controllers have (CHRISTOFIDES et al., 2013). Typically, cooperative controllers have a model for how the other variables affect their system and how their decision variables affect the objective of their neighboring subsystems. Ultimately, the cooperative controller solves the global optimization problem, where each subsystem has only the terms that they affect. While most communication-based algorithms are non-iterative, cooperative algorithms have an iterative nature due to their negotiating aspect. For the same sampling time, they will solve the optimization problem several times, informing their neighbors of the decisions they are willing to take; this allows each controller to adjust based on their neighbor's actions before applying the control actions into the system.

According to Rawlings and Stewart (2008), a cooperative controller has the following properties:

- The iterations generated by the cooperative MPC algorithm are systemwide feasible,
- Control based on any intermediate termination of the algorithm provides nominal closed-loop stability and zero steady-state offsets,
- If iterated to convergence, the distributed MPC algorithm achieves optimal, centralized MPC control,
- To handle output instead of state feedback, a distributed estimator design strategy can be implemented. Each estimator is stable and uses only local measurements to estimate subsystem states. The combined distributed estimator-distributed regulator is feasible and closed-loop stable for all iterations (in the case of decaying estimate error).

So the cooperative distributed controller has the best of both worlds; it has a distributed implementation, and, at the same time, it keeps the performance of the global controller. The disadvantages of the cooperative distributed controller include the requirement of an infrastructure to be used, including communication and multiple agents that can solve OCPs. While this is less of a problem with technological advances, the concept of "mainframes" remains popular. Also, depending on the sparsity of the system, it can be more computationally costly to obtain a solution compared to centralized controllers.

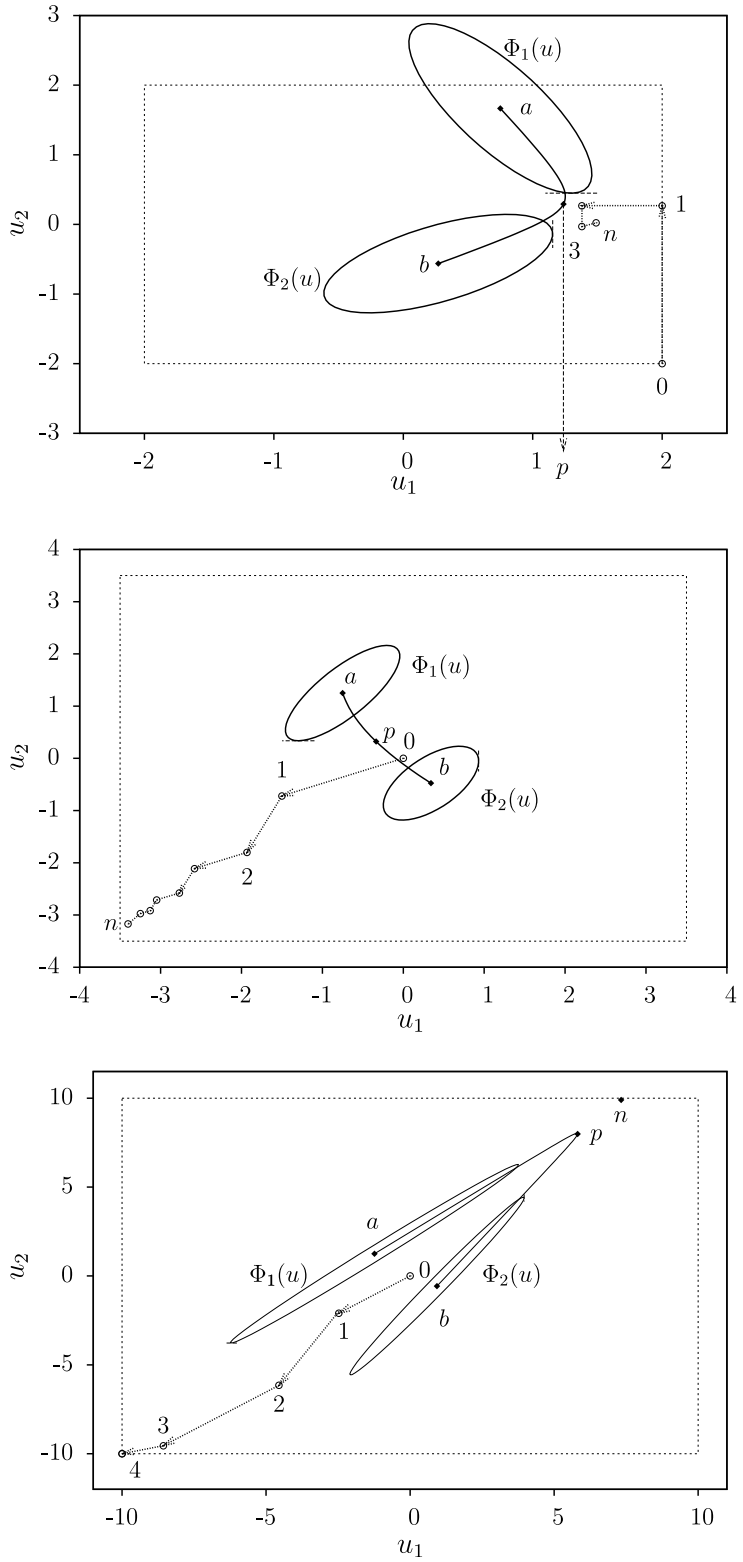


Figure 18 – Experiment demonstrating the convergence of communication-based controller. The functions $\Phi_1(u)$ and $\Phi_2(u)$ are the objective functions of each subsystem, *a* and *b* being their optimal solution (disregarding the other subsystem), the ellipsoid around each optimal point being the contour curve, the line connecting *a* and *b* are the Pareto set, where the point *p* is the Pareto optimal point balancing the objective of both systems. First: the algorithm converges to a Nash-point close to the Pareto optimal. Second: the algorithm converges to a Nash-point far from the Pareto optimal. Third: the algorithm diverges from the Nash-point despite starting at a point close to the Pareto optimal. Extracted from (RAWLINGS; STEWART, 2008)

3.1.3 Related Works

There are not many works that use a continuous-time model for modeling distributed systems. However, the most common application is for systems whose subsystems are not physically coupled but coupled by constraint. Dunbar and Murray (2006) proposes a distributed control for a multi-vehicle system whose dynamics and constraints are uncoupled but the cost function couples the states. Murray (2007) surveys of methods for continuous-time linear models of multi-vehicle systems; while this is a multi-agent problem, the systems were not coupled physically but by constraints.

Regarding works that propose solutions for problems with continuous-time systems with coupled dynamics, some of them are described briefly in the following. In (YAO, J. et al., 2009), a continuous-time nonlinear dynamic network was proposed, but then there is no complete decoupling of the local models. In (GUAN et al., 2012), a similar approach is used to solve a consensus problem in continuous time networks. In (FARINA et al., 2014), a continuous-time linear model has been used; however, the concept of networks is not established. In (BESTLER; GRAICHEN, 2017), an ADMM-based algorithm is proposed for a network modeled with state and control variables but no algebraic variables.

Other works have been using discrete-time dynamic networks (DAI et al., 2017; FERRAMOSCA et al., 2013; HERNANDEZ et al., 2016; VENKAT et al., 2005; CAMPONOGARA et al., 2002; MENDES et al., 2017), although they serve as a reference on what can be pursued, they are not directly comparable with this work.

3.1.4 Contribution

As seen in the previous sections, the cooperative distributed controller can be implemented honoring the distributed characteristics of the system while achieving the performance of a centralized controller. For these reasons, this class of controllers is studied in this work.

First, a framework for modeling distributed networks based on continuous-time DAE models is proposed. This framework is detailed in Section 3.3. This framework renders a mathematical structure that has a decoupled cost and coupled constraint structure. However, the only coupling constraints are the constraints that connect the output of the upstream subsystem to the input of the downstream one. This framework resembles “object-oriented programming”, in the sense that each subsystem has its equations and interacts only by an interface (input-output connecting equations).

Based on this modeling framework, three methods are proposed using the augmented Lagrangian algorithm developed by Aguiar (2016). All proposed algorithms are partially connected cooperating iterative algorithms according to the classification of (SCATTOLINI, 2009). Meaning that a local controller only shares information with

neighboring controllers (partially connected), the information is transmitted multiple times at the same sampling time (iterative), and they minimize a global cost function (cooperative algorithm).

The first algorithm is based on the coordinate descent method (BERTSEKAS, 1995). The main idea is to use the augmented Lagrangian algorithm for OCP to transform the model given by the framework, which is a decoupled cost coupled constraint, into a coupled cost decoupled constraint problem. Problems with coupled cost decoupled constraints can be solved using the coordinate descent algorithm, which splits the decision space into smaller subspaces and solves the optimal control problem iterating between the solution subspaces. Conveniently, we choose the subspaces to match the spaces of variables of each subsystem. This way, each subsystem has a local optimal control problem which is the projection of the global optimal control problem onto the subsystem space of decision variables. In this approach, the optimal control problems resulting from the augmented Lagrangian algorithm are solved to optimality, and then the multiplier estimates and penalty parameter are updated.

The second approach is based on the alternating direction multiplier method (ADMM) (BOYD et al., 2010). It has the same general idea as the previous method. The augmented Lagrangian algorithm is used to obtain a coupled cost decoupled constraint problem. The algorithms differ from the way that this subproblem is solved. In the previous algorithm, the resulting subproblem is solved to convergence, and then the parameters of the augmented Lagrangian algorithm are updated. In this method, only one step of the coordinate descent algorithm is performed for each subsystem, and then the penalty of the augmented Lagrangian is updated.

Both methods require some coordination among the subsystems. Due to some characteristics of the optimal control problem generated by the augmented Lagrangian method, not all subsystems can simultaneously solve their local optimal control problems. The subsystems can be grouped so that no two neighbors are solving their optimal control problem simultaneously. This allows a certain degree of parallelization, but it is limited to how coupled or sparse the system is. For highly sparse systems, this might not be a problem for some structures; for instance, in a star or tree formation (Figure 15), it is possible to split the whole network into two groups. However, for tightly coupled systems, the number of groups can grow to the number of subsystems in the network.

For this reason, a variant of both methods is proposed. This variant includes virtual subsystems in the network; by doing so, no two original subsystems are neighbors, which allows full parallelization of the network, and the cost of solving the optimal control problem of the virtual nodes is minor.

Finally, the third proposed method is a variation of the ADMM method, which through algebraic manipulation of the resulting equations exploits the network topogra-

phy. This method achieves simultaneous updates of all subsystems.

3.2 ALGORITHMS

3.2.1 Coordinate descent[1]

Coordinate descent (CD) algorithms are among the more straightforward classes of methods for solving optimization problems. CD origins are dated to the foundation of the optimization discipline and have many variations due to its age and straightforwardness (WRIGHT, 2015).

CD algorithms are iterative; at each iteration, most components of the decision vector (x) are fixed, and the objective is minimized with the remaining components. Each iteration is associated with a lower-dimensional subproblem that can be solved with less effort than the original problem.

The simplicity of these algorithms and their acceptable performance arguably is one of the main reasons for their popularity among practitioners. Nevertheless, the scarcity of intricacies makes these algorithms overlooked by the scientific community, who often favor investigating more elaborate methods. However, with the popularization of machine learning and the ever-increasing demand for computing power, CD algorithms have shown to be a competitive alternative to traditional approaches for solving problems that arise from machine learning. Simultaneously, the development of mathematical guarantees and parallelization variants makes these algorithms even more relevant, particularly for large-scale optimization problems.

Formulation

The simplest problem that CD solves is the unbounded optimization problem,

$$\min_x \ f(x) \tag{185}$$

where $f : \mathbb{R}^{N_c} \to \mathbb{R}$ is a continuous function. More specific assumptions might be required, depending on the particular CD algorithm. For instance, some algorithms assume that f is smooth and convex, other algorithms accept it to be smooth and nonconvex, or even f being smooth with a restricted domain.

There is another typical formulation that includes a regularization term,

$$\min_x \ f(x) + \lambda\Omega(x) \tag{186}$$

where the function f is smooth, Ω is a nonsmooth and extended-value[2] regularization function, and $\lambda > 0$ is the regularization parameter. Algorithms exploit the structure of f

[1] Wright (2015) was the main source for this section, however other works were used as Bertsekas and Tsitsiklis (1989)

[2] Extended-value functions are functions that have infinty in its counter-domain, e.g. $f : X \to Y \cup \{\infty\}$

and the regularization function Ω. Often Ω is convex and separable (or block separable). The regularization term can represent methods, as ℓ_1 norm, ℓ_2 norm, empirical risk minimization, and Lagrangian dual. Algorithm 3 shows a general CD algorithm that assumes that the objective function is differentiable. In each step, the component i_k of the gradient ∇f is evaluated at the current point x^k, followed by an update of the i_k-th component of the decision variable x. The selection of which component of x iterates can follow different sets of rules. These rules are further explained later in the section. The update of the component can be via a fixed step α^k, a varying α^k that satisfies a line-search condition, or an exact minimization in the direction of the component i_k which would give an optimal α^k. The vector e_{i_k} that appears in Algorithm 3 is a vector with 1 in the i_k-th element and 0 everywhere else.

Algorithm 3 Coordinate descent for optimization

Set $k \leftarrow 0$, requires x^0
repeat
  choose an index $i_k \in \{1, \ldots, N_c\}$
  $x^{k+1} \leftarrow x^k - \alpha^k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha^k > 0$
  $k \leftarrow k + 1$
until pass convergence test

CD algorithms have apparent similarity to the Gauss-Seidel method for solving linear equations (GOLUB; VAN LOAN, 2013). They iterate in each component and step towards minimizing the objective function, just like finding a solution for a set of linear equations.

Example 5 (Coordinate descent)**.** *Consider the following optimization problem*

$$\min_{x_1, x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{187}$$

For a given start point for x_1 and x_2, the coordinate descent algorithm solves the optimization problem above by only solving in the direction x_1, keeping x_2 fixed. After finding a new x_1, the problem is solved in the direction x_2, keeping x_1 fixed. The process repeats until a convergence test is satisfied.

If an optimal step is used, solving problem (187) *in the direction of x_1 is the equivalent of solving the problem*

$$\mathcal{P}_1(\overline{x}_2): \quad \min_{x_1} x_1^2 - 2x_1\overline{x}_2 \tag{188}$$

where the $\overline{x}_2$ is the previous value obtained for variable x_2. Likewise, solving (187) *in the direction x_2 is equivalent to solving the problem*

$$\mathcal{P}_2(\overline{x}_1): \quad \min_{x_2} -2\overline{x}_1 x_2 + 2\overline{x}_1^2 - 4x_2 \tag{189}$$

where the $\overline{x}_1$ is the previous value obtained for variable x_1.

Figure 19 illustrates the algorithm converging to the unbounded minimum for the initial condition $x_1 = 5$ and $x_2 = 2$ using optimal steps in each iteration. Notice that the centralized method travels from the initial guess to the optimum following a diagonal, going in a straight line. On the other hand, since the coordinate descent iterates over each direction, it takes a longer path.

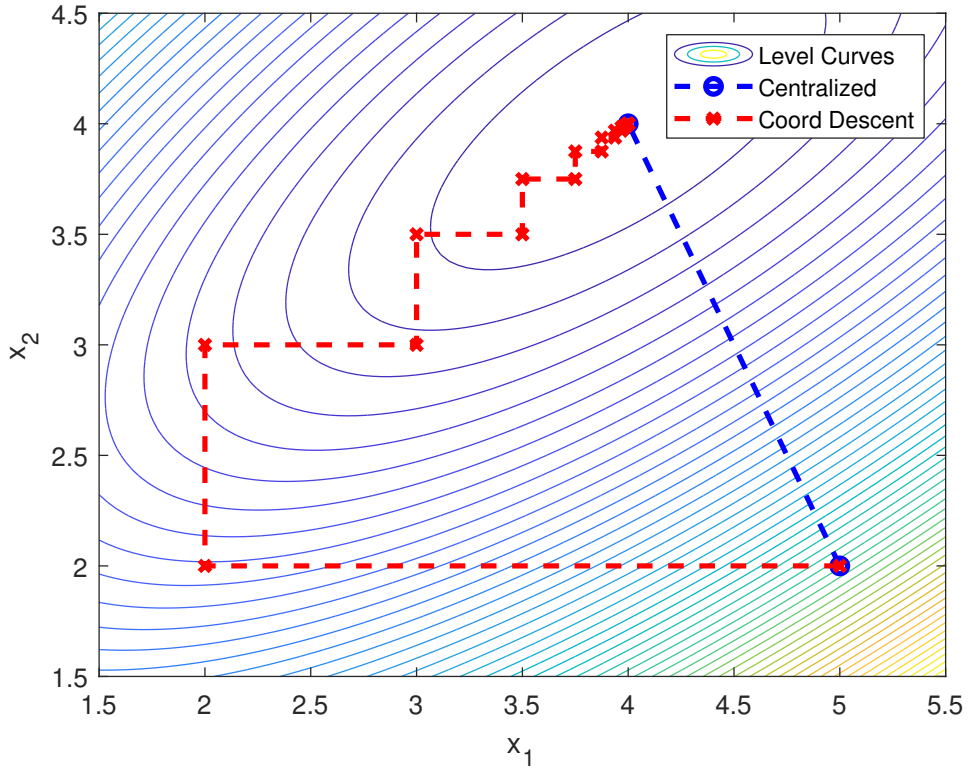


Figure 19 – Solution of problem (187) using a centralized method and a coordinate descent (decentralized)

Variants

Algorithm 3 is a platform for many algorithm variations. A possible point of variation for the CD algorithms is choosing the component at each iteration, as previously stated. Iterations might be sequential or parallel; in parallel iterations, the iterations may be synchronous or asynchronous. Also, each iteration step might have a fixed length, a varying length according to some rule, or even be of optimal length. Additionaly, there are algorithms designed to solve particular sub-classes of problem (185).

The choice of the iterating component varies according to the technique. The i_k-th component can be chosen following a deterministic cycle (e.g., $1, 2, \ldots, n$), where every n iterations, the algorithm iterates over the same component. The choice of the component can follow a sampling rule. The component is chosen randomly at each iteration, allowing to avoid any bias caused by the cyclic iterations. On the other hand,

a completely random choice may cause some directions to be chosen more often than others, leading to occasional slow convergence. A third approach, sampling with replacement, combines the previous two. A component is picked off out of a pool of possible components. When there are no more components in the pool, it refills with all components, allowing for a balanced iteration without hindering one component. Numerical experiments by Wright (2015) show that these three variants generally behave the same. However, in some cases, the sampling with replacement was faster than the sampling without replacement; in other cases, the cyclic case had worse convergence rates than the randomized variants.

So far, the CD has been described as iterating over a single component at each iteration. However, a simple extension is to iterate a more significant number of components at a time. This type of approach might increase the efficiency of block decomposable problems, that is, problems in which a subset of components is tightly connected. Also, as described further in the section, it might be possible to solve problems with constraints that couple multiple components.

In recent years, with increased demand for efficient algorithms for solving optimization problems with a high volume of data and numerous features, several parallel variants have been proposed. CD algorithms can be parallelized for problem-specific applications, exploiting the problem structure. Another form to achieve parallelization is with a more generic approach, having the same algorithm running in different instances, operating over the same shared decision vector x. The synchronous variants typically perform some iterations over a local copy of the decision vector. The instances then perform a synchronization step to update the local copy with the information from other instances every few steps. Bradley et al. (2011) propose *Shotgun*, a parallel algorithm for ℓ_1-Regularized Loss minimization based on the shooting method developed by Fu (1998). Jaggi et al. (2014) propose an algorithm for convex regularization functions with lower communication cost to reduce synchronization costs; the proposed method has some similarities to the Gauss-Jacobi method for solving linear equations, instead of the Gauss-Seidel. Richtárik and Takáč (2016) propose a randomized block CD algorithm for smooth convex functions whose benefits come from exploiting the objective function's separability. The method is enhanced by an accelerated version on Fercoq et al. (2014), taking advantage of the developments of Lee and Sidford (2013). A similar approach is used in Mareček et al. (2015), where the decision variable is split into chunks, and each instance is responsible for a chunk; the same approach of Richtárik and Takáč (2016) is applied to each chunk for another increased parallelization. In the asynchronous algorithms, the decision vector is centralized, available for all instances to update its local copy and write their updates. Each instance runs its CD algorithm that does not communicate or synchronize with other instances other than updating the same shared decision vector. An illustrative example of an asynchronous algorithm can be obtained

from modifying Algorithm 3, displayed in Algorithm 4. Notice that Algorithms 3 and 4 differ only by the argument of the gradient computation. The differentiation happens because, by the time an instance is updating the vector (x^{k+1}), the current value for the decision vector (x^k) is no longer the same used to compute the gradient ($\tilde{x}^k$): it was changed by another thread running the same method. It is challenging to show mathematical properties of asynchronous algorithms due to the mismatch between several instances running Algorithm 4. However, due to its importance to practical applications, advances have been achieved to support these methods (LIU; WRIGHT, 2015; LIU et al., 2015).

Algorithm 4 Asynchronous coordinate descent for optimization

Set $k \leftarrow 0$, requires x^0
repeat
  choose an index $i_k \in \{1, \ldots, n_c\}$
  $x^{k+1} \leftarrow x^k - \alpha^k [\nabla f(\tilde{x}^k)]_{i_k} e_{i_k}$ for some $\alpha^k > 0$
  $k \leftarrow k + 1$
until termination

Convergence

Generically speaking, the CD may fail to converge. In a seminal work, Powel shows that there exists an unbounded optimization problem such that a CD algorithm may fail to converge for a problem (185). In the counter-examples, Powel proposes a nonconvex continuously differentiable objective function in $\mathbb{R}^3$, in which a cyclic CD algorithm that performs exact minimization and starts at a specific region will not converge, cycling indefinitely. Powell provides further examples, which show that convergence cannot be expected without further assumptions.

Bertsekas (2005) shows that even with a nonconvex problem, a cyclic CD may converge if for every point x in the domain, there is only one minimizer along any coordinate direction. Beck and Tetruashvili (2013) show that under the assumption that the objective function f is convex and Lipschitz continuously differentiable, and the problem attains finite minima in its domain; a cyclic block CD algorithm with constant step sizes converges. This property can be generalized to the case where each block has a single component.

Multi-Block CD

Multi-block CD algorithms can solve a specialized type of problem (185). These algorithms can solve problems that have orthogonal constraints in each block, for in-

stance

$$\min_x f(x) \tag{190a}$$

$$\text{s.t.: } g_b(x_b) \leq 0 \qquad \forall b \in B \tag{190b}$$

$$h_b(x_b) = 0 \qquad \forall b \in B \tag{190c}$$

where $x_b \in \mathbb{R}^{N_c^b}$ is the components of block b, a subvector of x, N_c^b is the number of components in the block $b \in B = \{1, \ldots, N_b\}$, $g_b(x_b) \leq 0$ and $h_b(x_b) = 0$ are the inequality and equality constraints of block b, which define the feasible set $X_b \subseteq \mathbb{R}^{N_c^b}$.

To solve problem (190), an algorithm similar to Algorithm 3 can be used. However, a minor modification is required. A purely fixed step algorithm cannot work because it may violate the constraints. An approach to handle this issue is to perform a feasible set projection. Alternatively, an optimal step for the block will provide a block-feasible update. An algorithm that solves (190) is displayed in Algorithm 5.

Algorithm 5 Coordinate descent for multi-block optimization

Set $k \leftarrow 0$, requires x^0
repeat
  choose a block $b \in B$
  $x^{k+1} \leftarrow x^k - \alpha^k [\nabla f(x^k)]_b e_b$ for some $\alpha^k > 0$
  $k \leftarrow k + 1$
until pass convergence test

The following example expands on example 5 by including a constraint in each block (in each component, for the two dimensions case).

Example 6 (Coordinate descent with intra-block constaint)**.** *Let us include the following box constraints into* (187)*,*

$$2.5 \leq x_1 \leq 5 \tag{191a}$$

$$1.5 \leq x_2 \leq 3.75 \tag{191b}$$

which results in the following optimization problem,

$$\mathcal{P}: \quad \min_{x_1, x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{192a}$$

$$\text{s.t.: } 2.5 \leq x_1 \leq 5 \tag{192b}$$

$$1.5 \leq x_2 \leq 3.75 \tag{192c}$$

Notice that with the intent of visualizing the iterations, (192) *is a two-dimensional problem with one component in each block. Hence box constraints in* (192) *are the only possible type of intra-block constaint. In higher-order problems, more general constraints could be included.*

For the problem with box constraints with optimal steps, the directional subproblems are

$$\mathcal{P}_1(\overline{x}_2): \quad \min_{x_1} \; x_1^2 - 2x_1\overline{x}_2 \tag{193a}$$

$$\text{s.t.: } 2.5 \leq x_1 \leq 5 \tag{193b}$$

for x_1*, and*

$$\mathcal{P}_2(\overline{x}_1): \quad \min_{x_2} \; -2\overline{x}_1 x_2 + 2x_1^2 + 4x_2 \tag{194a}$$

$$\text{s.t.: } 1.5 \leq x_2 \leq 3.75 \tag{194b}$$

for x_2*.*

Figure 20 shows the algorithm iterations for the initial conditions $x_1 = 5$ *and* $x_2 = 2$*, using optimal steps at each iteration.*

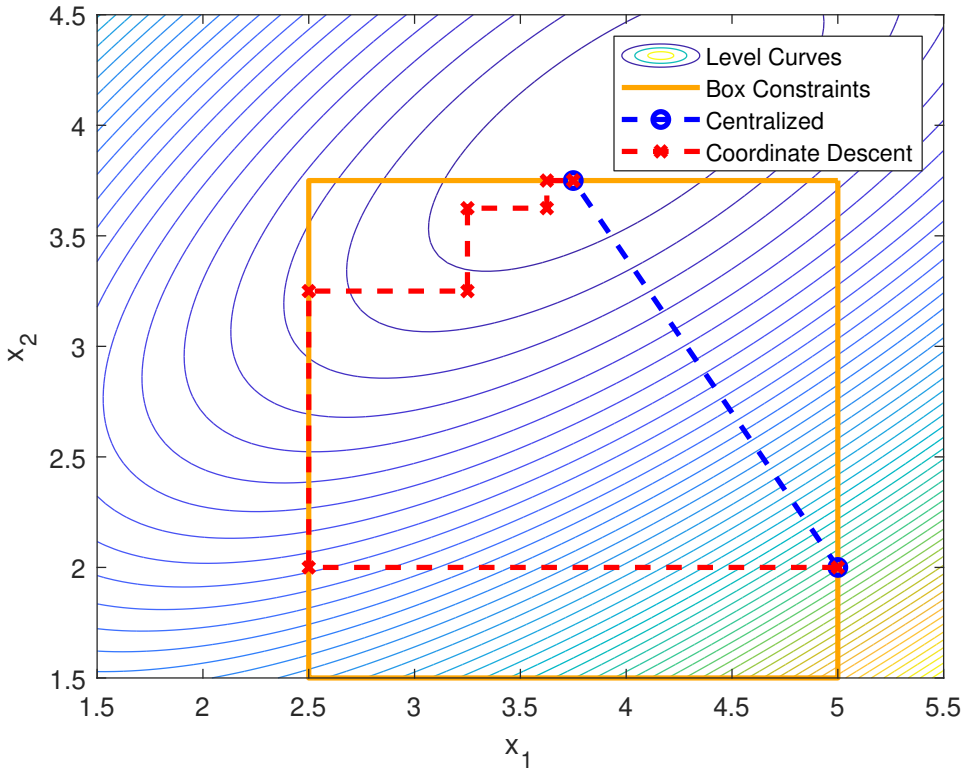


Figure 20 – Solution of problem (192) using a centralized method and a coordinate descent (decentralized)

Notice that problem (190) does not contain inter-block coupling constraints, for instance inequality

$$g(x_1, x_2) \leq 0 \tag{195}$$

or an equality constraint

$$h(x_1, x_2) = 0 \tag{196}$$

Without transforming the problem and altering the algorithm, the general coordinate descent algorithm cannot guarantee an optimal solution that satisfies the constraints. This issue happens because Algorithm 5 can only solve problems with coupled cost and (block) decoupled constraints (CCDC), meaning that the problem may have cross-terms in the objective function but no constraint coupling any two or more blocks.

This situation is illustrated in the following example.

Example 7 (Coordinate descent with inter-block constaint)**.** *If we include the following constraints into the unconstrained problem* (187)*,*

$$2.5 \leq x_1 \tag{197a}$$

$$1.5 \leq x_2 \leq 3.75 \tag{197b}$$

$$-x_1 + 2x_2 \leq 3.5 \tag{197c}$$

the coordinate descent algorithm might not converge to the centralized solution depending on the initial condition, as shown in Figure 21.

This issue happens because when the coupling constraint is active, the set of feasible directions that minimizes the objective function may not include either coordinate, x_1 and x_2. Therefore, the algorithm becomes "stuck" at the coupling constraint.

As seen in Example 7, CD is not able to solve coupled cost coupled constraint problems (CCCC). Therefore a way to solve this kind of issue is to apply a technique that transforms a CCCC problem into a CCDC.

There exist a variety of methods to perform this transformation: proximal methods, barrier methods, Lagrangian methods, augmented Lagrangian, among others. The following section explores how to use the augmented Lagrangian method to convert a CCCC problem into a CCDC problem, especially for dealing with coupling equality constraints.

3.2.2 Augmented Lagrangian with Coordinate Descent

The augmented Lagrangian (Section 2.4) is a method for solving equality constrained optimization problems also extensible to inequality constraints. It works by relaxing the equality constraint and including its violation as a penalty term in the objective function. By doing so, the method is removing a coupling constraint from the constraint and moving it to the objective.

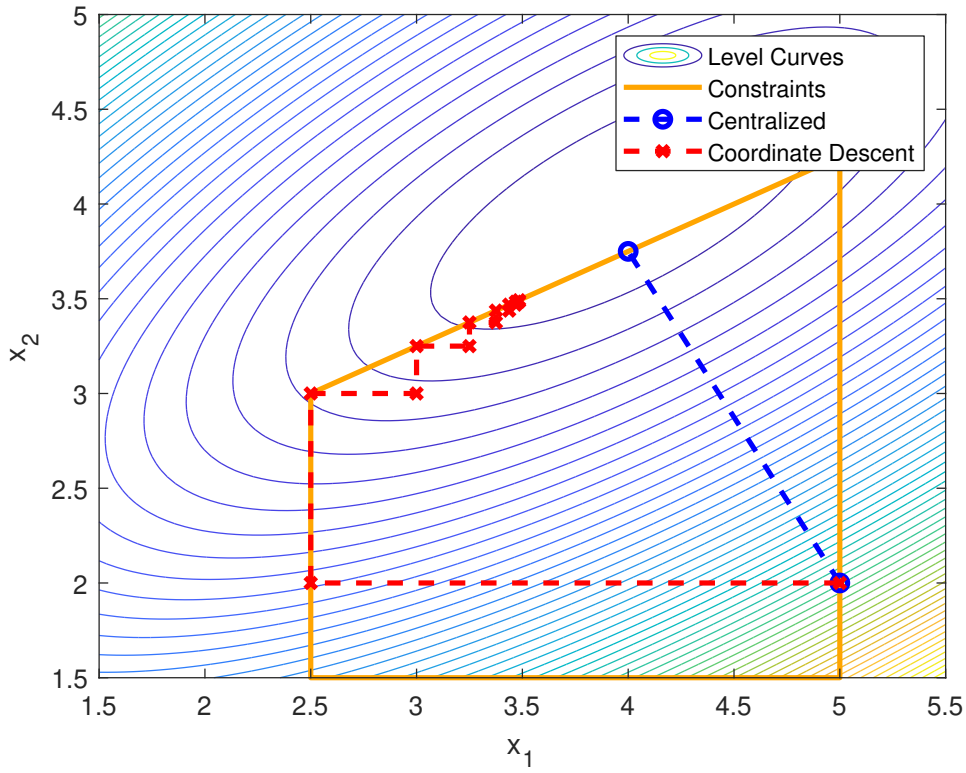


Figure 21 – Solution of problem (192) with coupling constraint

Let us extend (190) with an coupling equality constaint

$$\min_x f(x) \tag{198a}$$

s.t.: for all $b \in B$:

$$g_b(x_b) \leq 0 \tag{198b}$$

$$h_b(x_b) = 0 \tag{198c}$$

$$h(x) = 0 \tag{198d}$$

Notice that (198) is CCCC and cannot be solved with coordinate descent but can be relaxed by the augmented Lagrangian. The relaxation of (198d) results in the subproblem

$$\min_x f(x) + \lambda^T h(x) + \frac{\mu}{2} \|h(x)\|^2 \tag{199a}$$

s.t.: for all $b \in B$:

$$g_b(x_b) \leq 0 \tag{199b}$$

$$h_b(x_b) = 0 \tag{199c}$$

which is a CCDC problem, consequently solvable by the coordinate descent method.

The result of combining the augmented Lagrangian with coordinate descent is a double iterating algorithm. The augmented Lagrangian's outer loop requires the

subproblem solution to update the penalization μ and the multiplier λ. The inner loop, the coordinate descent loop, solves the subproblem by successively iterating over each block. Algorithm 6 synthesizes the amalgam of the augmented Lagrangian and the coordinate descent.

Algorithm 6 Augmented Lagrangian with Coordinate Descent

```
Set k ← 0, requires x^0, μ^0, and λ^0
repeat
    repeat
        choose a block b ∈ B
        update x^{k+1} in the directions of block b
    until pass inner loop convergence test
    update μ^{k+1}
    update λ^{k+1}
    k ← k + 1
until pass outer loop convergence test
```

The following example exhibits how the combination of these two algorithms can solve an equality-constrained optimization problem.

Example 8. *Let us include the following equality constraint into* (187)*,*

$$x_1 + x_2 = 5 \tag{200}$$

which yields the following equality constrained optimization problem,

$$\mathcal{P}: \quad \min_{x_1,x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{201a}$$

$$\text{s.t.: } x_1 + x_2 = 5 \tag{201b}$$

which cannot be solved by coordinate descent.

To obtain the augmented Lagrangian subproblem, let us relax the equality constraint, which produces

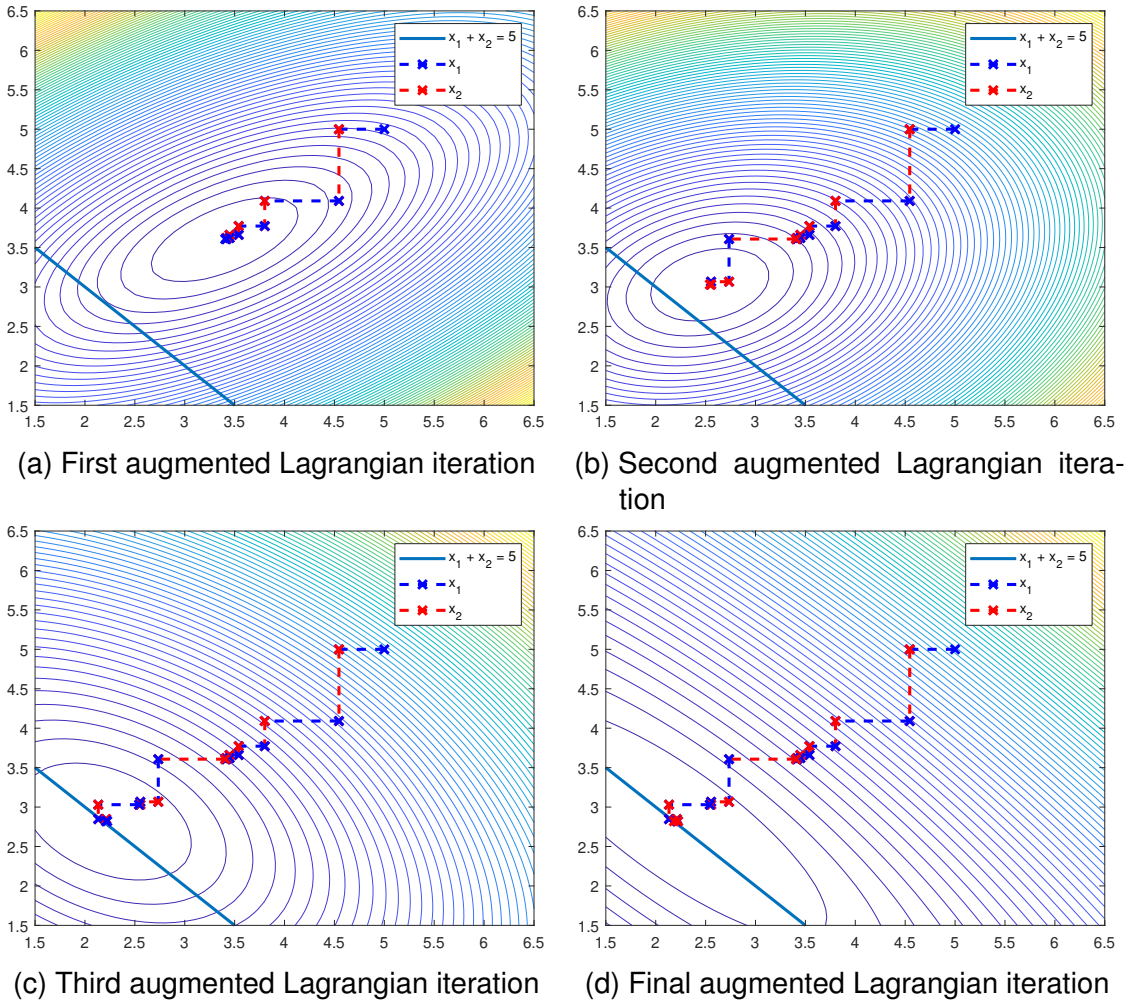
$$\widehat{\mathcal{P}}(\mu,\lambda): \quad \min_{x_1,x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \lambda(x_1 + x_2 - 5) + \frac{\mu}{2} \|x_1 + x_2 - 5\|^2 \tag{202}$$

The directional subproblems are

$$\widehat{\mathcal{P}}_1(\overline{x}_2,\mu,\lambda): \quad \min_{x_1} x_1^2 - 2x_1\overline{x}_2 + \lambda(x_1 - 5) + \frac{\mu}{2} \|x_1 + \overline{x}_2 - 5\|^2 \tag{203}$$

for x_1, and

$$\widehat{\mathcal{P}}_2(\overline{x}_1,\mu,\lambda): \quad \min_{x_2} -2\overline{x}_1 x_2 + 2x_2^2 + -4x_2 + \lambda(x_2 - 5) + \frac{\mu}{2} \|\overline{x}_1 + x_2 - 5\|^2 \tag{204}$$



(a) First augmented Lagrangian iteration

(b) Second augmented Lagrangian iteration

(c) Third augmented Lagrangian iteration

(d) Final augmented Lagrangian iteration

Figure 22 – Solution of problem (201) using the augmented Lagrangian and the coordinate descent to solve the subproblem. In the figures above, the level curves are readjusted in each iteration, and the path of previous iteration steps are depicted to full visualization of the algorithm behavior.

for x_2.

Figure 22 shows the algorithm iterations for the initial conditions $x_1 = 5$ and $x_2 = 5$, using optimal steps at each iteration. Notice that each augmented Lagrangian iteration has different level curves; this happens because of the increase of the penalization μ^k and the change of the multiplier approximation λ^k.

The combination of these two methods allows for a simple way to solve equality-constrained optimization problems. The requirement of solving each augmented Lagrangian subproblem to its optimum is wasteful. Notice how small the steps become at the end of an augmented Lagrangian iteration, followed by an increase in the stepsizes with adjustments of the objective function at the beginning of the following iteration. Supported by (BERTSEKAS, 1982, Propostion 2.3), it is possible to achieve a more efficient strategy by starting with a higher tolerance for the subproblem solution and decreasing it with each iteration.

The following method further builds upon this idea and dramatically reduces the overall number of iterations.

3.2.3 Alternating Direction Multiplier Method (ADMM)[3]

The alternating-direction method of multipliers (ADMM) (BOYD et al., 2010) has reached significant recognition in recent times due to its value in designing parallel algorithms and solving regularized problems for statistics and machine learning. Each ADMM iteration consists of obtaining an approximate minimizer of the augmented Lagrangian function over each block in turns, succeeded by an update of the Lagrange multiplier estimates.

The ADMM is typically presented as an algorithm to solve two-block equality constrained problems of the form

$$\min_{x,y} f(x) + g(y) \tag{205a}$$

$$\text{s.t.: } Ax + By = c \tag{205b}$$

where $x \in X \subset \mathbb{R}^{N_x}$, $y \in Y \subset \mathbb{R}^{N_y}$, $A \in \mathbb{R}^{N_c \times N_x}$, $B \in \mathbb{R}^{N_c \times N_y}$, and $c \in \mathbb{R}^{N_c}$. The functions f and g are assumed to be convex.

Applying the augmented Lagrangian to (205) yields the following objective function

$$L_\mu(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - c) + \frac{\mu}{2} \|Ax + By - c\|^2 \tag{206}$$

The augmented Lagrangian subproblem then becomes minimizing L_μ without any constraints.

The ADMM iterates over each direction, just like the augmented Lagrangian with the CD algorithm. However, it does only once for every direction, followed by an update of multipliers. On the other hand, if we were to solve the subproblem with the augmented Lagrangian with CD, we would iterate over the directions x and y until the CD loop achieved convergence and only then update the multipliers and the penalization. Algorithm 7 exhibits the steps for the ADMM. Notice that the ADMM does not assume an ever-increasing penalization term $\mu > 0$. On the contrary, such an assumption may cause convergence proofs to be hard to achieve.

[3] This section was written mainly based on Boyd et al. (2010), however other sources were used as (BERTSEKAS, 1982).

Algorithm 7 Alternating direction multiplier method (ADMM)

Set $k \leftarrow 0$, requires x_0, y_0, λ_0, μ
repeat
  $x^{k+1} \leftarrow \arg\min_x L_\mu(x, y^k, \lambda^k)$
  $y^{k+1} \leftarrow \arg\min_y L_\mu(x^{k+1}, y, \lambda^k)$
  $\lambda^{k+1} \leftarrow \lambda_k + \mu(Ax^{k+1} + By^{k+1} - c)$
  $k \leftarrow k + 1$
until pass convergence test

The ADMM uses a similar structure as its originating algorithm, the multiplier method (BERTSEKAS, 1982). There is no distinction between x and y variables in the multiplier method, and all variables are updated simultaneously.

$$(x^{k+1}, y^{k+1}) \leftarrow \arg\min_{x,y} L_\mu(x, y, \lambda^k) \tag{207}$$

$$\lambda^{k+1} \leftarrow \lambda_k + \mu(Ax^{k+1} + By^{k+1} - c) \tag{208}$$

The ADMM has better convergence properties, and the distinction of two separate steps for x and y is what enables the decomposition when f and g are separable.

The ADMM can be written in an alternative form, where the penalization μ scales the penalty and the Lagrangian term. Let $r = Ax + By - c$, then

$$\lambda^T r + \frac{\mu}{2}\|r\|^2 = \frac{\mu}{2}\left\|r + \frac{1}{\mu}\lambda\right\|^2 - \frac{1}{2\mu}\|\lambda\|^2 \tag{209a}$$

$$= \frac{\mu}{2}\|r + \sigma\|^2 - \frac{\mu}{2}\|\sigma\|^2 \tag{209b}$$

where $\sigma = \frac{1}{\mu}\lambda$ is the scaled multiplier variable. Using the scaled variables, the updates in Algorithm 7 can be reestated as

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{\mu}{2}\left\|Ax + By^k - c + \sigma_k\right\|^2\right) \tag{210a}$$

$$y^{k+1} = \arg\min_y \left(g(y) + \frac{\mu}{2}\left\|Ax^{k+1} + By - c + \sigma_k\right\|^2\right) \tag{210b}$$

$$\sigma^{k+1} = \sigma^k + \left(Ax^{k+1} + By^{k+1} - c\right) \tag{210c}$$

since the term $\frac{\mu}{2}\|\sigma\|^2$ does not vary with either x or y, it is disregarded for the update steps. By defining the k-th residual as $r^k = Ax^k + By^k - c$, the k-th scaled multiplier is given by

$$\sigma^k = \sigma_0 + \sum_{j=1}^{k} r^j \tag{211}$$

The scaled and the unscaled version are equivalent and can be used interchangeably.

Example 9. *Let us use the same equality constrained problem* (201)*,*

$$\mathcal{P}: \quad \min_{x_1,x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{212a}$$

$$\text{s.t.: } x_1 + x_2 = 5 \tag{212b}$$

which has the following augmented Lagrangian subproblem:

$$\widehat{\mathcal{P}}(\mu,\lambda): \quad \min_{x_1,x_2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \lambda(x_1 + x_2 - 5) + \frac{\mu}{2} \|x_1 + x_2 - 5\|^2 \tag{213}$$

with the directional subproblem being

$$\widehat{\mathcal{P}}_1(\overline{x}_2,\mu,\lambda): \quad \min_{x_1} x_1^2 - 2x_1\overline{x}_2 + \lambda(x_1 - 5) + \frac{\mu}{2} \|x_1 + \overline{x}_2 - 5\|^2 \tag{214}$$

for x_1, and

$$\widehat{\mathcal{P}}_2(\overline{x}_1,\mu,\lambda): \quad \min_{x_2} -2\overline{x}_1 x_2 + 2x_2^2 + -4x_2 + \lambda(x_2 - 5) + \frac{\mu}{2} \|\overline{x}_1 + x_2 - 5\|^2 \tag{215}$$

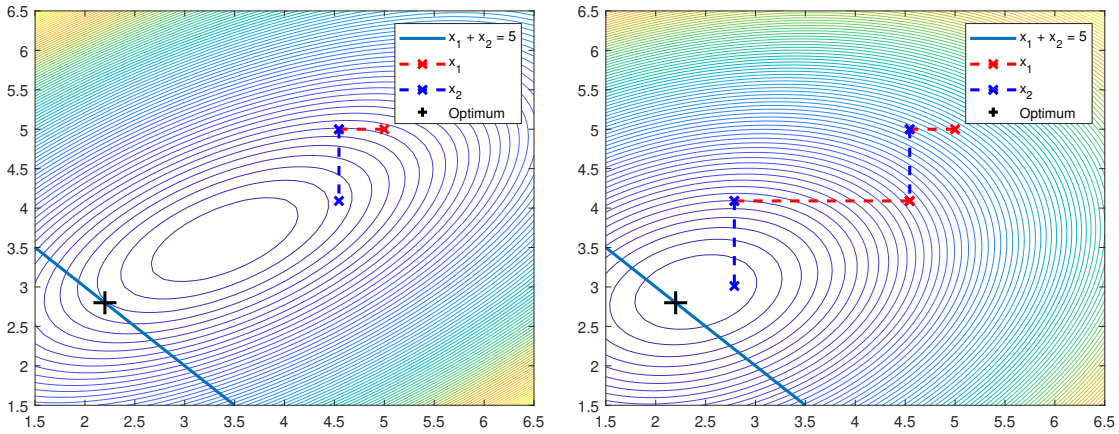
for x_2.

For an initial condition $x_1, x_2 = 5,5$, Figure 23 depicts the iterations of the ADMM until it reaches a point close to the optimum. Notice how the objective changes after only one iteration in each direction (represented in the level curves). This change makes the ADMM reach a point close to the optimum in fewer steps than the augmented Lagrangian combined with the coordinate descent.

The example above shows the ADMM converging in fewer steps than the augmented Lagrangian with CD. Indeed, empirical experiments show that ADMM is often more efficient than the augmented Lagrangian with coordinate descent (CHEN, L. et al., 2019; ECKSTEIN; YAO, W., 2015). However, if no warm start technique is used, ADMM iteration steps might be more complex and time-consuming, depending on the problem's size and the nonlinearities.

Convergence

Over the years, several works developed convergence proofs for the ADMM. Under the assumption that *f* and *g* are closed, proper, convex, and extended valued functions, it has been shown that *x*-update and *y*-update are solvable; that is, there exists an *x* and a *y* that minimizes the augmented Lagrangian function. Such assumption allows for *f* and *g* to be non-differentiable or even be an indicator function that assumes the value $+\infty$.



(a) First ADMM iteration

(b) Second ADMM iteration

(c) Third ADMM iteration

(d) Final ADMM iteration

Figure 23 – Solution of problem (212) using ADMM to solve the subproblem. In the figures above, the level curves are readjusted in each iteration, and the path of previous iterations iteration steps are depicted to are full visualization of the algorithm behavior.

Further, let us assume that there exists a saddle point for the unaugmented Lagrangian, that is, there exists a (x^*, y^*, λ^*), such that

$$L_0(x^*, y^*, \lambda) \leq L_0(x^*, y^*, \lambda^*) \leq L_0(x, y, \lambda^*) \tag{216}$$

for any x, y, λ. By the first assumption, there exists a finite objective $L_0(x^*, y^*, \lambda^*)$ for all (x^*, y^*, λ^*). This implies that (x^*, y^*) solves (205), $Ax^* + By^* = c$, $f(x^*) \leq \infty$, and $g(y^*) \leq \infty$. This further implies that λ^* is dual optimal.

Under such assumptions, it is possible to show

- Residual convergence: $r^k \to 0$ as $k \to \infty$, meaning that the algorithm approaches primal feasibility with the iterations.
- Objective convergence: $f(x^*) + g(y^*) \to p^*$ as $k \to \infty$, where p^* is the optimal objective of (205); implying that with the iterations the objective appoaches the optimal value.

- Dual variable convergence: $\lambda^k \to \lambda^*$ as $k \to \infty$, where λ^* is the optimal multiplier of (205).

The proofs are presented in (Appendix A, BOYD et al., 2010).

Practical experiments show that ADMM has a slow convergence for high-accuracy solutions. However, ADMM has been shown to achieve reasonable accuracy within a few iterations, which suffice for many applications. For applications that require high precision, the augmented Lagrangian can be combined with better methods that have better convergence when close to optimality, such as Newton's method (ECKSTEIN; FERRIS, 1998). However, the usage of Newton's method might be limited by the size of the problem. The augmented Lagrangian is commonly applied to large-scale problems in statistics and machine learning, where high accuracy is not required. On the contrary, high accuracy is typically associated with overfitting, therefore avoided, making the ADMM an exceptional fit.

Multi-block ADMM

Most of the scientific and practical work was developed for two-block ADMM, as in (205), but it often happens that we have more block separable functions and variables. For instance,

$$\min_{x,y,z} f(x) + g(y) + h(z) \tag{217a}$$

$$\text{s.t.: } A_1 x + A_2 y + A_3 z = c \tag{217b}$$

where the augmented Lagrangian funcion is

$$L_\mu(x, y, z, \lambda) = f(x) + g(y) + h(z) + \lambda^T (A_1 x + A_2 y + A_3 z - c) + \frac{\mu}{2} \|A_1 x + A_2 y + A_3 z - c\| \tag{218}$$

Indeed problem (217) can be solved using a two-block strategy by grouping x and y, or y and z. However, by doing so, we are not taking advantage of the problem structure.

A way to benefit from the problem structure is to produce a direct extension to Algorithm 7. Let us have a four steps update, one for each block and one for the multipliers,

$$x^{k+1} = \arg\min_x L_\mu(x, y^k, z^k, \lambda^k) \tag{219a}$$

$$y^{k+1} = \arg\min_y L_\mu(x^{k+1}, y, z^k, \lambda^k) \tag{219b}$$

$$z^{k+1} = \arg\min_z L_\mu(x^{k+1}, y^{k+1}, z, \lambda^k) \tag{219c}$$

$$\lambda^{k+1} = \lambda_k + \mu(A_1 x^{k+1} + A_2 y^{k+1} + A_3 z^{k+1} - c) \tag{219d}$$

which makes use of the problem's structure.

A long-standing question was if such extension would inherit the convergence properties from the two-blocks ADMM, given that it has shown to be empirically efficient (HE, B.; YUAN, X., 2018). Caihua Chen et al. (2016) showed that a multi-block ADMM is not necessarily convergent by developing a sufficient condition for convergence and producing a violating counter-example. The sufficient condition is that any two coefficient matrices of (205), are orthogonal. That is, one of the following holds:

- $A_1^T A_2 = 0$,
- $A_2^T A_3 = 0$,
- $A_1^T A_3 = 0$.

The proof follows by showing that if any two of the matrices are orthogonal, then the update (219) is equivalent to two block ADMM. For instance, if $A_1^T A_2 = 0$, then

$$(x^{k+1}, y^{k+1}) = \arg\min_{x,y} L_\mu(x, y, z^k, \lambda^k) \tag{220a}$$

$$z^{k+1} = \arg\min_{z} L_\mu(x^{k+1}, y^{k+1}, z, \lambda^k) \tag{220b}$$

$$\lambda^{k+1} = \lambda^k + \mu(A_1 x^{k+1} + A_2 y^{k+1} + A_3 z^{k+1} - c) \tag{220c}$$

which is a two-block update by grouping x and y. For more details, the reader is forwarded to (CHEN, C. et al., 2016, Section 2)

Caihua Chen et al. (2016) also claim that this proof can easily be extended for a m-block ADMM. Given a problem with the form

$$\min_x \sum_{i=1}^{m} \theta_i(x_i) \tag{221a}$$

$$\text{s.t.: } \sum_{i=1}^{m} A_i x_i = b \tag{221b}$$

$$x_i \in X_i, \qquad i = 1, \ldots, m \tag{221c}$$

where $m > 3$ is the number of blocks, $x_i \in X_i \subset \mathbb{R}^{N_{n_i}}$ are the variables of the block $i \in \{1, \ldots, m\}$, and $X_i \subset \mathbb{R}^{n_i}$ are closed convex sets. The sum of all $\theta_i : \mathbb{R}^{n_i} \to \mathbf{R}$ defines the decomposable objective function, which is closed and convex but not necessarily smooth functions. The coupling constraint is defined by the matrices $A_i \in \mathbb{R}^{p \times n_i}$, and the vector $b \in \mathbb{R}^p$.

In order to ensure the convergence of the extended ADMM applied to (221), the sufficient condition is recast as

> [...] There exist two integers i and j such that any two matrices in the sets $\{A_i, A_{i+1}, \ldots, A_{i+j}\}$ and $\{A_{i+j+1}, A_{i+j+2}, \ldots, A_m, A_1, A_2, ..., A_{i-1}\}$ are orthogonal (CHEN, C. et al., 2016) .

which can be interpreted as if we can group the sequential iteration steps into two sets with all matrices being orthogonal between each other, then the algorithm will converge.

Lin et al. (2015) propose a sufficient alternate condition that depends only on the objective function and on the penalty parameter that has sublinear convergence.

Varying Penalty

The standard ADMM uses a fixed parameter μ; experiments show that if μ is kept too small or too large, it can hinder computation performance (FORTIN; GLOWINSKI, 2000; FUKUSHIMA, 1992; KONTOGIORGIS; MEYER, 1998) or even make the problem ill-conditioned (TOSSERAMS, 2008). The augmented Lagrangian uses an ever-increasing $\mu^k \to \infty$, and the ADMM can use a similar strategy. By doing so, the ADMM can achieve better practical convergence, making it less susceptible to the initial conditions. In the case of the augmented Lagrangian, Rockafellar (1976) shows that a superlinear convergence is achievable with $\mu^k \to \infty$. While the convergence of the ADMM with μ^k is hard to prove, the increase of μ might result in faster convergence in practice. Combining both strategies by capping μ^k to a maximum value $\mu_{\max}$ increases the convergence and maintains the same properties of the fixed-parameter ADMM. By following this strategy, the update of the penalty parameter is defined by,

$$\mu^{k+1} = \min(\beta\mu^k, \mu_{\max}) \tag{222}$$

where $\beta > 1$.

Another usual practice to vary the penalty parameter is to couple it with the primal residual (XU et al., 2014, 2015). Given the problem (205), the primal residual is given by the relaxed equation

$$r^k = Ax^k + By^k - c \tag{223}$$

The strategy is to increase the penalty parameter if there is not enough decrease to the primal residual

$$\mu^{k+1} = \begin{cases} \mu^k & \text{if } \left\|r^k\right\| \leq \gamma \left\|r^{k-1}\right\| \\ \beta\mu^k, & \text{otherwise} \end{cases} \tag{224}$$

where $\beta > 1$, and $0 < \gamma < 1$.

These two previous approaches have an issue when the penalty parameter becomes too high: the augmented problem gets close to the unrelaxed problem. This issue hinders the method, making it sluggishly progress towards the optimal point since it gives much importance to satisfying the constraints.

B. S. He et al. (2000) propose a modified version of the ADMM, where the penalty parameter μ is allowed to increase and decrease. The authors present a self-adaptive

rule and exhibit the convergence proofs for such modifications. In total, three versions are presented: μ is monotonically increasing, μ is monotonically decreasing, and μ is adjusted by self-adaptative rule. B. S. He et al. (2000) compare the strategies where

- $\{\mu^k\}$ is constant, the traditional approach,
- $\{\mu^k\}$ is monotonically increasing,
- $\{\mu^k\}$ is monotonically decreasing,
- $\{\mu^k\}$ is a self-adaptative sequence that can decrease and increase,

They show that only the self-adaptative approach can achieve a low number of convergence iterations independently from the initial parameter μ_0.

The self-adaptative approach follows the rule

$$\mu^{k+1} = \begin{cases} \beta\mu^k & \text{if } \left\|r^k\right\| > \gamma \left\|s^k\right\| \\ \beta^{-1}\mu^k & \text{if } \left\|s^k\right\| > \gamma \left\|r^k\right\| \\ \mu^k, & \text{otherwise} \end{cases} \tag{225}$$

where $\beta > 1$, $\gamma > 1$, r^k is the primal residual (223), and s^k is the dual residual (BOYD et al., 2010),

$$s^k = \mu^k A^T B(y^k - y^{k-1}) \tag{226}$$

Other related works are found in the literature. Wang and Liao (2001) show the same approach applied to other situations of interest. Xu et al. (2014, 2015) propose using the same update steps for the augmented Lagrangian coordination algorithm. Wohlberg (2017) proposes a normalized version of (225) that makes the update unaffected by the difference of magnitudes of the objective function, equality constraints, and variables.

3.3 DISTRIBUTED SYSTEMS

This section presents the proposed framework for modeling distributed dynamic systems composed of nonlinear subsystems. A distributed dynamic network can be represented by a directed graph $G = (N, C)$, where each node n in the set of nodes N represents a subsystem, and each arc (n_1, n_2) in the set of connection arcs C represents a connection between the outputs of the subsystem n_1 and the inputs of the subsystem n_2.

Regarding the modeling of each subsystem dynamics, we have, for each node $n \in N$,

$$\dot{x}_n = f_n(x_n, y_n, u_n, t) \tag{227}$$

$$0 = g_n(x_n, y_n, u_n, t) \tag{228}$$

where x_n are the states, y_n are the algebraic variables, and u_n are the inputs of subsystem n. The proposed modeling approach assigns a broader meaning for some of these variables. The variable u_n represents the variables that are not defined by the equations of n: the exogenous variables (variables defined by a connection with another subsystem) and the control variables (variables that the controller of n can manipulate). Also, the variables y_n are the algebraic variables, but they are also used to represent the subsystem's output.

Regarding the connection between the subsystems, for every connection $(n_1, n_2) \in C$, the connecting equations are given by

$$M^{\text{in}}_{n_1,n_2} u_{n_2} = M^{\text{out}}_{n_1,n_2} y_{n_1} \tag{229}$$

where $M^{\text{out}}_{n_1,n_2}$ is a matrix of size $N^c_{n_1,n_2} \times N^y_{n_1}$, and $M^{\text{in}}_{n_1,n_2}$ is a matrix of size $N^c_{n_1,n_2} \times N^u_{n_2}$, $N^c_{n_1,n_2}$ being the number of variables that the system n_2 has connected to system n_1, $N^y_{n_1}$ the number of algebraic variables in the subsystem n_1, and $N^u_{n_2}$ the number of inputs of the subsystem n_2. Notice that if all of the outputs of system n_1 are inputs of the system n_2 then

$$M^{\text{in}}_{n_1,n_2} = M^{\text{out}}_{n_1,n_2} = I \tag{230}$$

and

$$u_{n_2} = y_{n_1} \tag{231}$$

Regarding the objective function, the modeling assumes a fully decoupled objective, that is

$$J = \sum_{n \in N} \int_{t_0}^{t_f} L_n(x_n, y_n, u_n, t)\, dt \tag{232}$$

where L_n is the objective function of the subsystem n.

Further, it is possible to include all kinds of constraints in the variables of each subsystem. For instance, for the subsystem n the following constraints can be included

$$h_{ineq}(x_n, y_n, u_n, t) \leq 0, \qquad \forall t \tag{233a}$$
$$h_{eq}(x_n, y_n, u_n, t) = 0, \qquad \forall t \tag{233b}$$
$$h_{final}(x_n(t_f), y_n(t_f), u_n(t_f), t_f) = 0 \tag{233c}$$
$$x_n(t_0) = x_{n,0} \tag{233d}$$
$$x_{n,L} \leq x_n \leq x_{n,U} \tag{233e}$$
$$y_{n,L} \leq y_n \leq y_{n,U} \tag{233f}$$
$$u_{n,L} \leq u_n \leq u_{n,U} \tag{233g}$$

given that they do not require variables from other subsystems directly.

Therefore, an optimal control for such dynamic network can be written as

$$\min J = \sum_{n \in N} \int_{t_0}^{t_f} L_n(x_n, y_n, u_n, t)\, dt \tag{234a}$$

s.t.: for all n in N:

$$\dot{x}_n = f_n(x_n, y_n, u_n, t) \tag{234b}$$

$$0 = g_n(x_n, y_n, u_n, t) \tag{234c}$$

$$h_{ineq}(x_n, y_n, u_n, t) \leq 0, \qquad \forall t \tag{234d}$$

$$h_{eq}(x_n, y_n, u_n, t) = 0, \qquad \forall t \tag{234e}$$

$$h_{final}(x_n(t_f), y_n(t_f), u_n(t_f), t_f) = 0 \tag{234f}$$

$$x_n(t_0) = x_{n,0} \tag{234g}$$

$$x_{n,L} \leq x_n \leq x_{n,U} \tag{234h}$$

$$y_{n,L} \leq y_n \leq y_{n,U} \tag{234i}$$

$$u_{n,L} \leq u_n \leq u_{n,U} \tag{234j}$$

for all (n_1, n_2) in C:

$$M^{\text{out}}_{n_1,n_2} y_{n_1} - M^{\text{in}}_{n_1,n_2} u_{n_2} = 0 \tag{234k}$$

or in a simplified form,

$$\min J = \sum_{n \in N} \int_{t_0}^{t_f} L_n(x_n, y_n, u_n, t)\, dt \tag{235a}$$

s.t.: for all n in N:

$$\dot{x}_n = f_n(x_n, y_n, u_n, t) \tag{235b}$$

$$0 = g_n(x_n, y_n, u_n, t) \tag{235c}$$

$$x_n(t_0) = x_{n,0} \tag{235d}$$

for all (n_1, n_2) in C:

$$M^{\text{in}}_{n_1,n_2} u_{n_2} - M^{\text{out}}_{n_1,n_2} y_{n_1} = 0 \tag{235e}$$

Notice that the global optimal control problem (234) has a decoupled cost and coupled constraints (DCCC). However, the only coupling constraints are given by the connection (234k), equivalently (235e), which means that one subsystem does not affect the others directly but by an input-output interface.

Despite the constraints present on formulation (234), with a few modeling tricks, other constraints can be modeled, for instance:

- **A subsystem being affected by a non-output variable of another subsystem:** for instance, a case where state or input variables of subsystem n_2 are affecting the dynamics of subsystem n_1:

$$\dot{x}_{n_1} = -x_{n_1} + u_{n_1} + u_{n_2} \tag{236}$$

notice that the state x_{n_1} is affected by u_{n_2}. To put the dynamics of this subsystem in way that fits the form (234), we can rewrite the ODE of the state x_{n_1} using a new input $\widehat{u}_{n_1}$ as

$$\dot{x}_{n_1} = -x_{n_1} + u_{n_1} + \widehat{u}_{n_1} \tag{237}$$

where $\widehat{u}_{n_1}$ is defined by the connection equation

$$\widehat{u}_{n_1} = \widehat{y}_{n_2} \tag{238}$$

and the variable $\widehat{y}_{n_2}$ is defined as an output in the subsystem n_2 as

$$\widehat{y}_{n_2} = u_{n_2} \tag{239}$$

This transformation allows us to fit equations like (236) into the form specified by (234).

- **Objective functions with cross terms**: for example, if the global objective function is given by

$$J = \int_{t_0}^{t_f} x_{n_1}^T x_{n_1} + x_{n_1}^T x_{n_2} + x_{n_2}^T x_{n_2}\, dt \tag{240}$$

A problem with such objective function can fit into the form (234), by including the input variable $\widehat{u}_{n_1}$ and the output variable $\widehat{y}_{n_1}$ into the subsystem n_1, and including $\widehat{u}_{n_2}$ and the output variable $\widehat{y}_{n_2}$ into the subsystem n_2. In addition, include the following equations to the respective subsystems,

$$\widehat{y}_{n_1} = x_{n_1} \tag{241a}$$
$$\widehat{y}_{n_2} = x_{n_2} \tag{241b}$$

and the coupling constraints

$$\widehat{u}_{n_1} = \widehat{y}_{n_2} \tag{242a}$$
$$\widehat{u}_{n_2} = \widehat{y}_{n_1} \tag{242b}$$

Therefore the optimal control problem can be stated as

$$\min J = \int_{t_0}^{t_f} \left[x_{n_1}^T x_{n_1} + \frac{1}{2} x_{n_1}^T \widehat{u}_{n_1} \right] + \left[\frac{1}{2} \widehat{u}_{n_2}^T x_{n_2} + x_{n_2}^T x_{n_2} \right] dt \tag{243a}$$

s.t.: for subsystem n_1:

$$\widehat{y}_{n_1} = x_{n_1} \tag{243b}$$

$$\text{other equations of the subsystem } n_1 \tag{243c}$$

for subsystem n_2:

$$\widehat{y}_{n_2} = x_{n_2} \tag{243d}$$

$$\text{other equations of the subsystem } n_2 \tag{243e}$$

for all (n_1, n_2) in C:

$$M^{\text{out}}_{n_1,n_2} y_{n_1} - M^{\text{in}}_{n_1,n_2} u_{n_2} = 0 \tag{243f}$$

and the new connections:

$$\widehat{u}_{n_1} = \widehat{y}_{n_2} \tag{243g}$$

$$\widehat{u}_{n_2} = \widehat{y}_{n_1} \tag{243h}$$

which has the same structure of (234).

- **Coupling constraints:** for example

$$x_{n_1} + x_{n_2} + x_{n_3} \leq 1 \tag{244}$$

This type of constraint can be included inside one of the subsystems. The exogenous variables for that subsystem can be treated as input variables, as was done in the previous case. The constraint then becomes:

$$x_{n_1} + \widehat{u}_{1,n_1} + \widehat{u}_{2,n_1} \leq 1 \tag{245}$$

with the connections

$$\widehat{u}_{1,n_1} = \widehat{y}_{n_2} \tag{246a}$$

$$\widehat{u}_{2,n_1} = \widehat{y}_{n_3} \tag{246b}$$

and in the subsystems n_2 and n_3, the following equalities are included

$$\widehat{y}_{n_2} = x_{n_2} \tag{247a}$$

$$\widehat{y}_{n_3} = x_{n_3} \tag{247b}$$

Another approach is to create a new subsystem n_4 with no objective function, only with an equivalent formulation of the constraint (244). The optimal control problem

of the subsystem n_4 is given by

$$\min J_{n_4} = 0 \tag{248a}$$

$$\text{s.t.: } \widehat{u}_{1,n_4} + \widehat{u}_{2,n_4} + \widehat{u}_{3,n_4} \leq 1 \tag{248b}$$

and the following connections are included into the network,

$$\widehat{u}_{1,n_4} = \widehat{y}_{n_1} \tag{249a}$$

$$\widehat{u}_{2,n_4} = \widehat{y}_{n_2} \tag{249b}$$

$$\widehat{u}_{3,n_4} = \widehat{y}_{n_3} \tag{249c}$$

and in the subsystems n_1, n_2, and n_3, the following equalities are included

$$\widehat{y}_{n_1} = x_{n_1} \tag{250a}$$

$$\widehat{y}_{n_2} = x_{n_2} \tag{250b}$$

$$\widehat{y}_{n_3} = x_{n_3} \tag{250c}$$

The following example presents the modeling of the four tank system using formulation (234).

Example 10 (Dynamic Network of the four tank system)**.** *From Example 4 we established that a network for the four tank system should have a structure like the one shown in Figure 24, where the subsystems are* $N = \{T_1, T_2, T_3, T_3, P_1, P_2\}$*, and the connections are* $C = \{(P_1, T_4), (P_1, T_1), (P_2, T_3), (P_2, T_2), (T_3, T_1), (T_4, T_2)\}$*.*

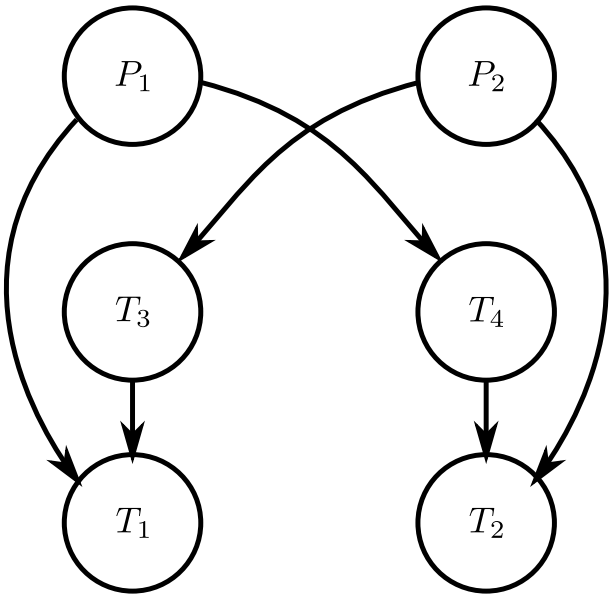


Figure 24 – Four tank system represented as a dynamic network

In the Example 1 a DAE model for a tank was developed, whose equations were

$$\dot{V} = q_{in} - q_{out} \tag{251a}$$
$$h = \frac{V}{A} \tag{251b}$$
$$q_{out} = a\sqrt{2gh} \tag{251c}$$

From these equations, we can separate the states, algebraic, and input variables for the tank subsystem T_n

$$x_{T_n} = \begin{bmatrix} V \end{bmatrix}, y_{T_n} = \begin{bmatrix} h \\ q_{out} \end{bmatrix}, u_{T_n} = \begin{bmatrix} q_{in} \end{bmatrix} \tag{252}$$

and the functions f_{T_n} and g_{T_n} that fit accordingly.

The two lower tanks receive fluids from two sources, so there is an additional equation

$$q_{in} = q_{in,1} + q_{in,2} \tag{253}$$

which makes the variables for those subsystems to be

$$x_{T_n} = \begin{bmatrix} V \end{bmatrix}, y_{T_n} = \begin{bmatrix} h \\ q_{out} \\ q_{in} \end{bmatrix}, u_{T_n} = \begin{bmatrix} q_{in,1} \\ q_{in,2} \end{bmatrix} \tag{254}$$

The pump-valve subsystems can have their dynamics modeled by the following equations

$$\dot{\omega} = K_V v \tag{255a}$$
$$q = K_{flow}\omega \tag{255b}$$
$$q_{out,1} = \gamma q \tag{255c}$$
$$q_{out,2} = (1-\gamma)q \tag{255d}$$

where the state ω is the pump's speed, being controlled by the voltage v; q is the flow emanating from the pump; and the flows through the three-way valve are $q_{out,1}$ and $q_{out,2}$, which are determined by γ. The model parameters are γ, K_v, and K_{flow}.

In order to put the model of a subsystems P_n in the form (234)*, the variables are grouped as follows*

$$x_{P_n} = \begin{bmatrix} \omega \end{bmatrix}, y_{P_n} = \begin{bmatrix} q \\ q_{out,1} \\ q_{out,2} \end{bmatrix}, u_{P_n} = \begin{bmatrix} v \end{bmatrix} \tag{256a}$$

and the functions f_{P_n} and g_{P_n} that fit accordingly

Assuming that the goal is to keep the two lower tanks' height at some desired reference while using the least power on the pumps, the local objectives are defined as

$$L_{T_1} = (h_{T_1} - h_{ref,T_1})^2 \tag{257a}$$
$$L_{T_2} = (h_{T_1} - h_{ref,T_1})^2 \tag{257b}$$
$$L_{T_3} = 0 \tag{257c}$$
$$L_{T_4} = 0 \tag{257d}$$
$$L_{P_1} = v_{P_1}^2 \tag{257e}$$
$$L_{P_2} = v_{P_2}^2 \tag{257f}$$

Regarding the connecting equations, they are defined as follows

$$\begin{bmatrix} 1 & 0 \end{bmatrix} u_{T_1} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} y_{P_1} \tag{258a}$$
$$u_{T_4} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} y_{P_1} \tag{258b}$$
$$\begin{bmatrix} 1 & 0 \end{bmatrix} u_{T_2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} y_{P_2} \tag{258c}$$
$$u_{T_3} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} y_{P_2} \tag{258d}$$
$$\begin{bmatrix} 0 & 1 \end{bmatrix} u_{T_1} = \begin{bmatrix} 0 & 1 \end{bmatrix} y_{T_3} \tag{258e}$$
$$\begin{bmatrix} 0 & 1 \end{bmatrix} u_{T_4} = \begin{bmatrix} 0 & 1 \end{bmatrix} y_{T_4} \tag{258f}$$

Therefore the global optimal control problem can be written as

$$\min J = \int_{t_0}^{t_f} \Big[(h_{T_1} - h_{ref,T_1})^2 + (h_{T_2} - h_{ref,T_2})^2 + v_{P_1}^2 + v_{P_2}^2 \Big] \, dt \tag{259a}$$

s.t.: for all $T \in \{T_1, T_2\}$:

$$\dot{V}_T = q_{T,in,1} + q_{T,in,2} - q_{T,out} \tag{259b}$$

$$h_T = \frac{V_T}{A_T} \tag{259c}$$

$$q_{T,out} = a_T \sqrt{2gh_T} \tag{259d}$$

for all $T \in \{T_3, T_4\}$:

$$\dot{V}_T = q_{T,in} - q_{T,out} \tag{259e}$$

$$h_T = \frac{V_T}{A_T} \tag{259f}$$

$$q_{T,out} = a_T \sqrt{2gh_T} \tag{259g}$$

for all $P \in \{P_1, P_2\}$:

$$\dot{\omega} = K_v v \tag{259h}$$

$$q = K_{flow} \omega \tag{259i}$$

$$q_{out,1} = \gamma q \tag{259j}$$

$$q_{out,2} = (1 - \gamma) q \tag{259k}$$

with the connections:

$$\begin{bmatrix} 1 & 0 \end{bmatrix} u_{T_1} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} y_{P_1} \tag{259l}$$

$$u_{T_4} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} y_{P_1} \tag{259m}$$

$$\begin{bmatrix} 1 & 0 \end{bmatrix} u_{T_2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} y_{P_2} \tag{259n}$$

$$u_{T_3} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} y_{P_2} \tag{259o}$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} u_{T_1} = \begin{bmatrix} 0 & 1 \end{bmatrix} y_{T_3} \tag{259p}$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} u_{T_4} = \begin{bmatrix} 0 & 1 \end{bmatrix} y_{T_4} \tag{259q}$$

3.4 PROPOSED ALGORITHMS FOR DISTRIBUTED OPTIMAL CONTROL

This work aims to propose a distributed framework for solving optimal control problems for networked systems. Using the proposed modeling (234) and the developments of the Section 2.5, a few proposed algorithms are presented inspired by the algorithms from Section 3.2.2 and 3.2.3.

In this section, the OCP form (235) is used for the sake of brevity. However, the same developments are valid for OCPs with the form (234).

The formulation (235) is a decoupled cost with coupled constraint (DCCC), which coordinate descent algorithms cannot solve due to the connection between the subsystems. To achieve a CCDC problem, which such algorithms can solve, one can use the augmented Lagrangian for OCP presented in Section 2.5. By applying the relaxation to the coupling constraints, the following subproblem is obtained

$$
\begin{aligned}
&\mathcal{P}_{\mathcal{L}}(\mu, \nu): \\
&\min J = \sum_{n \in N} \int_{t_0}^{t_f} \Big\{ L_n(x_n, y_n, u_n, t) \\
&\qquad + \sum_{(n_1,n_2) \in C} \Big[\nu_{(n_1,n_2)}^T \left(M_{n_1,n_2}^{\text{in}} u_{n_2} - M_{n_1,n_2}^{\text{out}} y_{n_1} \right) \\
&\qquad + \frac{\mu}{2} \left\| M_{n_1,n_2}^{\text{in}} u_{n_2} - M_{n_1,n_2}^{\text{out}} y_{n_1} \right\|^2 \Big] \Big\} \, dt && \text{(260a)} \\
&\text{s.t.: for all } n \text{ in } N: && \text{(260b)} \\
&\qquad \dot{x}_n = f_n(x_n, y_n, u_n, t) && \text{(260c)} \\
&\qquad 0 = g_n(x_n, y_n, u_n, t) && \text{(260d)} \\
&\qquad x_n(t_0) = x_{n,0} && \text{(260e)}
\end{aligned}
$$

Subproblem (260) can be projected to subsystem n, resulting in the subproblem

$$
\begin{aligned}
&\mathcal{P}_{\mathcal{L},n}(\mu, \nu_n, y_n^{\text{ext}}, u_n^{\text{ext}}): \\
&\min J = \int_{t_0}^{t_f} L_n(x_n, y_n, u_n, t) \\
&\qquad + \sum_{(n_1,n_2) \in C(n)} \Big\{ \nu_{(n_1,n_2)}^T \left[M_{n_1,n_2}^{\text{in}} u_{n_2} - M_{n_1,n_2}^{\text{out}} y_{n_1} \right] \\
&\qquad + \frac{\mu}{2} \left\| M_{n_1,n_2}^{\text{in}} u_{n_2} - M_{n_1,n_2}^{\text{out}} y_{n_1} \right\|^2 \Big\} \, dt && \text{(261a)} \\
&\text{s.t.: } \dot{x}_n = f_n(x_n, y_n, u_n, t) && \text{(261b)} \\
&\qquad 0 = g_n(x_n, y_n, u_n, t) && \text{(261c)} \\
&\qquad x_n(t_0) = x_{n,0} && \text{(261d)}
\end{aligned}
$$

where:

- $C(n)$ is the set of connections of the subsystem n, mathematically

$$C(n) = \{(n_1, n_2) \in C : n_1 = n \vee n_2 = n\} \tag{262}$$

- ν_n is the vector containing the multipliers estimates $\nu_{(n_1,n_2)}$ of the connections $(n_1, n_2) \in C(n)$,
- y_n^{ext} is the vector of external algebraic variables, that is $y_n^{\text{ext}} = [y_{n_1} : (n_1, n) \in C(n)]$,

- u_n^{ext} is the vector of inputs, with $u_n^{\text{ext}} = [u_{n_2} : (n, n_2) \in C(n)]$.

Based on these relaxed subproblems, the following techinques are proposed:

- Augmented Lagrangian method with coordinate descent (AL-CD),
- Alternating direction multiplier method (ADMM),
- Fully decoupling the subsystems,
- Bipartite-Jacobi ADMM,

3.4.1 Augmented Lagrangian with Coordinate Descent

With the CCDC subproblem (260) we can develop a coordinate descent like algorithm. Because the iteration of one subsystem depends on a few others, given the network structure, it is not possible to iterate over all subsystems simultaneously. Therefore, we need to define the iteration blocks. Intuitively, each node can be placed in a block. For instance, we can separate the blocks for the four tank system, as portrayed in Figure 25.

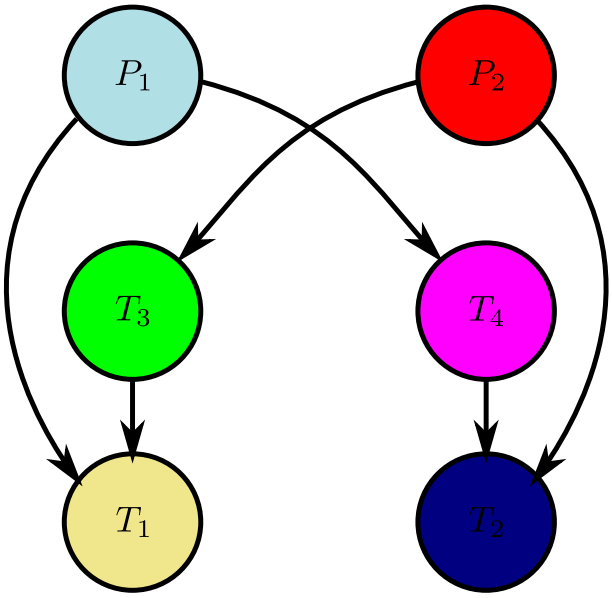


Figure 25 – Four tank system represented as a dynamic network, each subsystem in a block (represented by a color)

While this is a simple solution, there exist more efficient groupings. The limiting condition for all subsystems simultaneously iterating is that two subsystems that share a variable cannot iterate alongside. By definition of the modeling framework, two subsystems share a variable if they are neighbors[4]. Therefore, the condition translates into

[4] A subsystem is a neighbor to another subsystem if it is an in-neighbor or an out-neighbor, meaning that it has a connection from or to another subsystem.

two neighboring subsystems not being allowed to iterate alongside if they are adjacent. This condition matches the description of the map coloring problem[5], a well-studied graph problem (HARTSFIELD; RINGEL, 2003; BONDY; MURTY, 2008).

Given a graph $G = (N, C)$, a k-coloring is a map $c : N \to S$, where S is a set of k colors. Hence the coloring c assigns each vertex one of the k colors. A coloring is said proper if no two adjacent vertices have the same color. A coloring separates the vertices into k groups $\{N_1, N_2, \ldots, N_k\}$ of N, where each N_i is the set of vertices that where assigned the color i.

The only graphs that are 1-colorable are graphs with no connections between the nodes. 2-colorable graphs are bipartite graphs, that is, graphs that can be split into two sets. Graphs can be 3-, 4-, ..., N_N-colorable, where N_N is the number of nodes in the network. For instance, if a network is a complete graph, it is necessary N_N colors because all vertices are adjacent to each other. To have the most subsystems iterating concurrently in the coordinate descent, we want to have the least number of colors, which translates into the least number of blocks. The minimum k for which a graph is k-colorable is known as the chromatic number. Therefore the coloring problem consists of finding the minimum k for which a graph is colorable.

There exists a polynomial-time algorithm for finding if a graph is 2-colorable because it is bipartite. For $k > 2$, generally the problem is *NP*-complete. Therefore it is common to rely upon heuristics to obtain a k-coloring for a graph. In this work, we focus on the OCP solution and assume that the graph coloring solution is known a prirori. There are a few algorithms for solving this problem. In particular, for our case, distributed algorithms could be used for assigning a block for each subsystem in a startup phase (DUBHASHI, 2008; GHAFFARI; KUHN, 2020).

In this work context, we know that for any distributed subsystem, there is a minimum number of blocks N_B that minimize the number of iterations required by the coordinate descent, such that $2 \leq N_B \leq N_N$. For instance, as previously described, the four tank system can be separated into three blocks, as depicted in Figure 26, or as two blocks, as shown in Figure 27.

Notice that the grouping is only proposed for concurrent optimization, not that a single node or subsystem performs all the computations for that group. Each subsystem is associated with a controller that can compute its decisions locally by only exchanging information with its neighbors.

Once the blocks are defined, a coordinate descent algorithm that solves the augmented Lagrangian subproblem (260) can be presented. The algorithm has two iterating loops; the outer loop is the augmented Lagrangian loop, which requires the subproblem's solution for updating the multipliers and the penalty parameter. The inner loop is the coordinate descent loop, responsible for solving the subproblems until it

[5] also known as vertex coloring problem.

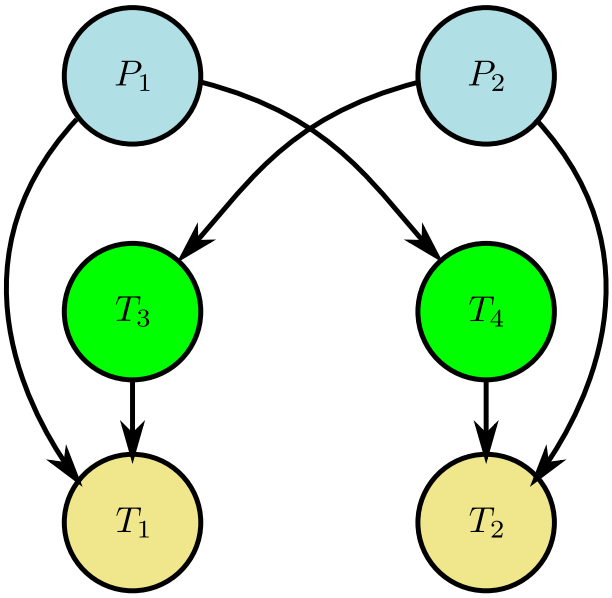


Figure 26 – Dynamic network graph colored using three colors

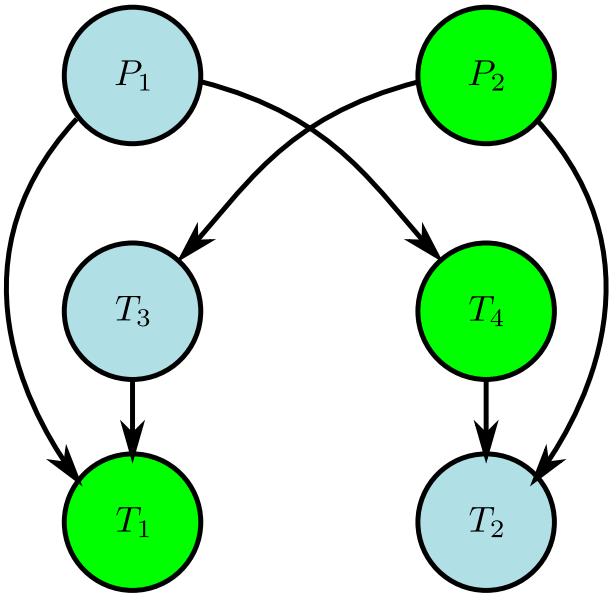


Figure 27 – Dynamic network graph colored with the optimal solution using two colors

passes a convergence test. The algorithm is presented in Algorithm 8, where y_b and u_b are the algebraic and control variables for all subsystems in the block b, the variables $y_n^{\text{ext},k}$ and $u_n^{\text{ext},k}$ are defined as

$$u_n^{\text{ext},k} = \left\{ u^k_{\widehat{b},j^{\widehat{b}}_{last}} : \forall \widehat{b} \in B \text{ such that } \widehat{b} \text{ has connections to } b \right\} \tag{263}$$

$$y_n^{\text{ext},k} = \left\{ y^k_{\widehat{b},j^{\widehat{b}}_{last}} : \forall \widehat{b} \in B \text{ such that } \widehat{b} \text{ has connections to } b \right\} \tag{264}$$

where n belongs to block b, and $j^{\widehat{b}}_{last}$ is the last coordinate descent iteration over the block $\widehat{b}$, $j^{\widehat{b}}_{last} = 0$ if it was not iterated in the current augmented Lagrangian iteration k.

As this method uses the augmented Lagrangian for OCP, the subproblems may use any available method to solve its subproblem. The method that each subsystem uses may differ, even in the same block. For instance, one subsystem may use a direct multiple shooting while another uses an indirect collocation. This flexibility makes it possible to integrate commercial control solutions from different vendors, as long as they can produce optimal control and algebraic variables.

The algorithm requires an initial value for the multiplier estimates ν_n^0 and the penalty parameter μ^0. Each block iteration requires other blocks' previous values. Hence the algorithm needs an initial guess for the algebraic and control variable ($y^0_{b,0}$ and $u^0_{b,0}$) for each block over time. Depending on the solution method used to solve each subproblem, more initialization parameters may be required. In the case of a direction collocation, an initial guess of the profiles for the state, the algebraic, and the control variables are needed. An initial guess for the final value of the adjoint variables is necessary for an indirect multiple shooting.

Algorithm 8 Coordinate descent with Augmented Lagrangian

repeat
  repeat
    choose a block $b \in B$
    $\left(y^k_{b,j+1}, u^k_{b,j+1}\right) \leftarrow \left(solve\left\{\mathcal{P}_{\mathcal{L},n}\left(\mu^k, \nu^k_n, y_n^{\text{ext},k}, u_n^{\text{ext},k}\right)\right\},\ \forall n \in b\right)$
    $j \leftarrow j + 1$
  until coordinate descent converged
  $(y^{k+1}_{b,0}, u^{k+1}_{b,0}) \leftarrow (y^k_{b,j}, u^k_{b,j})$ for all b
  Update multipliers ν^k
  Update penalty factor μ^k
  $k \leftarrow k + 1$
until augmented Lagrangian converged

The algorithm assumes perfect communication and synchronization. The local convergence can be defined by a minimum decrease in the local objective function or a more elaborate approach. The whole system convergence is verifiable by a consensus algorithm.

3.4.2 Alternating Directions Multiplier Method (ADMM)

The ADMM uses the relaxed subproblem (260), but instead of iterating with two loops like augmented Lagrangian with coordinate descent, it iterates with a single loop. To have a single loop, the ADMM performs a single iteration in each block, followed by an update of the multipliers, where the penalty parameter may or may not vary. Under the modeling framework (235), the resulting network may not be bipartite; therefore, the solution does not fit the common two-block ADMM, requiring a multi-block ADMM.

The sufficient condition shown in Section 3.2.3 is equivalent to the separation of blocks discussed in the previous section. Therefore, if a distributed system can be separated into two groups, meaning that its underlying graph is bipartite, it inherits its property from the two-block ADMM. Cascading subsystems like the star and the tree formations in Figure 15, can be easily separated into two groups. The cycle topology depends on the particular system; a six-node cycle is splittable in two, but a five-node cycle is not bipartite.

The ADMM algorithm proposed in this work has a single loop that solves a block of subsystems at a time, followed by the update of the multipliers, as summarized in Algorithm 9. The proposed ADMM algorithm has the variables $u_n^{\text{ext},k}$ and $y_n^{\text{ext},k}$ defined as

$$u_n^{\text{ext},k} = \left\{ u_{\widehat{b}}^{k+1} : \forall \widehat{b} \in B \text{ such that } \widehat{b} < b \right\} \bigcup \left\{ u_{\widehat{b}}^{k} : \forall \widehat{b} \in B \text{ such that } \widehat{b} > b \right\} \quad (265)$$

$$y_n^{\text{ext},k} = \left\{ y_{\widehat{b}}^{k+1} : \forall \widehat{b} \in B \text{ such that } \widehat{b} < b \right\} \bigcup \left\{ y_{\widehat{b}}^{k} : \forall \widehat{b} \in B \text{ such that } \widehat{b} > b \right\} \quad (266)$$

Like the previous algorithm, the ADMM requires an initial guess for the multipliers ν^0 and the penalty μ^0, as well as initial guesses for the control and algebraic variables (u_b^0 and y_b^0).

Like the previous algorithm, due to the usage of the augmented Lagrangian for OCP, each subsystem's solution may be achieved by any available solving method, not requiring to be the same method in all subsystems. The update of the penalty factor is optional and can be one of the following:

- μ is constant,
- μ changes with the primal residual,
- μ is updated using a self-adaptive strategy that adjusts based on the primal and dual residuals.

Algorithm 9 Alternating Direction Multiplier Method for OCP

repeat
  for $b \in B$ **do**
    $\left(y_b^{k+1}, u_b^{k+1}\right) \leftarrow \left(solve\left\{\mathcal{P}_{\mathcal{L},n}\left(\mu^k, \nu_n^k, y_n^{\text{ext},k}, u_n^{\text{ext},k}\right)\right\},\ \forall n \in b\right)$
  end for
  Update multipliers ν^{k+1}
  Update penalty factor μ^{k+1}
  $k \leftarrow k+1$
until convergence

For problem (260), the primal residual is obtained through the relaxed connections

$$r^k = \sum_{(n_1,n_2)\in C} \nu_{(n_1,n_2)}^T \left(M_{n_1,n_2}^{\text{in}} u_{n_2} - M_{n_1,n_2}^{\text{out}} y_{n_1}\right) \tag{267}$$

To derive the dual residual, first let us assume that (260) is two-block separable, and that $v_1^k = (y_1^k, u_1^k)$ is the vector with all algebraic and control variables of the systems in the block $b = 1$, and likewise $v_2^k = (y_2^k, u_2^k)$ for the block $b = 2$. Then with ADMM steps

$$v_1^{k+1} = \underset{v_1}{solve}\left\{\mathcal{P}_{\mathcal{L},1}\left(\mu^k, \nu_1^k, v_2^k\right)\right\} \tag{268a}$$

$$v_2^{k+1} = \underset{v_2}{solve}\left\{\mathcal{P}_{\mathcal{L},2}\left(\mu^k, \nu_2^k, v_1^{k+1}\right)\right\} \tag{268b}$$

followed by the update of the multipliers ν_1^k and ν_2^k. $\mathcal{P}_{\mathcal{L},1}$ is an ensemble of OCP, with form (261), of the subsystems in the first block, similarly for $\mathcal{P}_{\mathcal{L},2}$. Notice that v_1^{k+1} depends on v_2^k, since v_2^{k+1} is computed in a later step; this will relevant in the necessary conditions for OCP $\mathcal{P}_{\mathcal{L},1}$. For the sake of readability, let us put the relaxed constraints in the form,

$$g_c(v) = A_1 v_1 + A_2 v_2 = 0 \tag{269}$$

and the multipliers update are given by

$$\nu^{k+1} = \nu^k + \mu^k \left(A_1 v_1 + A_2 v_2\right) \tag{270}$$

Under such conditions, the Hamiltonian[6] for the relaxed OCP is given by

$$H = L_1 + L_2 + \lambda_1^T f_1 + \lambda_2^T f_2 + \nu_1^T g_1 + \nu_2^T g_2 + \left(\nu_c^k\right)^T \left(A_1 v_1 + A_2 v_2\right) + \frac{\mu^k}{2} \left\|A_1 v_1 + A_2 v_2\right\| \tag{271}$$

where the variables with the subscript 1 are associated with $b = 1$, and the ones with the subscript 2 are associated with $b = 2$. Therefore, the conditions for $\mathcal{P}_{\mathcal{L},1}$ and $\mathcal{P}_{\mathcal{L},2}$ can be obtained by only taking the terms for $b = 1$ and $b = 2$, respectively.

[6] presented in Section 2.1.2

The necessary condition for the OCP $\mathcal{P}_{\mathcal{L},2}$ obtained in (268b), under the perspective of v_2, is [7]

$$\frac{\partial H}{\partial v_2}^{k+1} = \frac{\partial L_2}{\partial v_2}^{k+1} + \left(\lambda_2^{k+1}\right)^T \frac{\partial f_2}{\partial v_2}^{k+1} + \left(\nu_2^{k+1}\right)^T \frac{\partial g_2}{\partial v_2}^{k+1} + \left(\nu_c^k\right)^T A_2 + \mu^k A_2 \left(A_1 v_1^{k+1} + A_2 v_2^{k+1}\right) \tag{272a}$$

$$= \frac{\partial L_2}{\partial v_2}^{k+1} + \left(\lambda_2^{k+1}\right)^T \frac{\partial f_2}{\partial v_2}^{k+1} + \left(\nu_2^{k+1}\right)^T \frac{\partial g_2}{\partial v_2}^{k+1} + A_2^T \left[\nu_c^k + \mu^k \left(A_1 v_1^{k+1} + A_2 v_2^{k+1}\right)\right] \tag{272b}$$

$$= \frac{\partial L_2}{\partial v_2}^{k+1} + \left(\lambda_2^{k+1}\right)^T \frac{\partial f_2}{\partial v_2}^{k+1} + \left(\nu_2^{k+1}\right)^T \frac{\partial g_2}{\partial v_2}^{k+1} + A_2^T \left[\nu_c^k + \mu^k r^{k+1}\right] \tag{272c}$$

$$= \frac{\partial L_2}{\partial v_2}^{k+1} + \left(\lambda_2^{k+1}\right)^T \frac{\partial f_2}{\partial v_2}^{k+1} + \left(\nu_2^{k+1}\right)^T \frac{\partial g_2}{\partial v_2}^{k+1} + A_2^T \nu_c^{k+1} \tag{272d}$$

which means that v_2 always satisfy the necessary conditions.

Similarly, the necessary conditions for v_1 in (268a) can be derived

$$\frac{\partial H}{\partial v_1}^{k+1} = \frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + \left(\nu_c^k\right)^T A_1 + \mu^k A_1 \left(A_1 v_1^{k+1} + A_2 v_2^k\right) \tag{273a}$$

$$= \frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + A_1^T \left[\nu_c^{k+1} + \mu^k \left(A_1 v_1^{k+1} + A_2 v_2^k\right)\right] \tag{273b}$$

$$= \frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + A_1^T \left[\nu_c^k + \mu^k r^{k+1} + \mu^k A_2 \left(v_2^k - v_2^{k+1}\right)\right] \tag{273c}$$

$$= \frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + A_1^T \nu_c^{k+1} + \mu^k A_1^T A_2 \left(v_2^k - v_2^{k+1}\right) \tag{273d}$$

Notice that $\frac{\partial H}{\partial v_1}$ satifies the necessary conditions only if

$$\frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + A_1^T \nu_c^{k+1} = \mu^k A_1^T A_2 \left(v_2^k - v_2^{k+1}\right) = 0 \tag{274}$$

therefore the vector-function $s : [t_0, t_f] \to \mathbb{R}^{N_{v_2}}$, defined as

$$s^{k+1} = \mu^k A_1^T A_2 \left(v_2^k - v_2^{k+1}\right) \tag{275}$$

[7] Here the notation f^k represent the function evaluated with its arguments at the iteration k, $f^k = f(x^k)$.

can be viewed as the dual residual.

For a three-block ADMM, where the coupling constraints are given by $Av_1 + Bv_2 + Cv_3 = 0$, similar development can be made to obtain the equations,

$$\frac{\partial H}{\partial v_1} = \frac{\partial L_1}{\partial v_1}^{k+1} + \left(\lambda_1^{k+1}\right)^T \frac{\partial f_1}{\partial v_1}^{k+1} + \left(\nu_1^{k+1}\right)^T \frac{\partial g_1}{\partial v_1}^{k+1} + A_1^T \nu_c^{k+1} + \mu^k A_1^T A_2 \left(v_1^k - v_2^{k+1}\right) + \mu^k A_1^T A_3 \left(v_3^k - v_3^{k+1}\right) \tag{276a}$$

$$\frac{\partial H}{\partial v_2} = \frac{\partial L_2}{\partial v_2}^{k+1} + \left(\lambda_2^{k+1}\right)^T \frac{\partial f_2}{\partial v_2}^{k+1} + \left(\nu_2^{k+1}\right)^T \frac{\partial g_2}{\partial v_2}^{k+1} + A_2^T \nu_c^{k+1} + \mu^k A_2^T A_3 \left(v_3^k - v_3^{k+1}\right) \tag{276b}$$

$$\frac{\partial H}{\partial v_3} = \frac{\partial L_3}{\partial v_3}^{k+1} + \left(\lambda_3^{k+1}\right)^T \frac{\partial f_3}{\partial v_3}^{k+1} + \left(\nu_3^{k+1}\right)^T \frac{\partial g_3}{\partial v_3}^{k+1} + A_3^T \nu_c^{k+1} \tag{276c}$$

The dual residual is then given by

$$s_1^{k+1} = \mu^k A_1^T A_2 \left(v_2^k - v_2^{k+1}\right) + \mu^k A_1^T A_3 \left(v_3^k - v_3^{k+1}\right) \tag{277a}$$

$$s_2^{k+1} = \mu^k A_2^T A_3 \left(v_3^k - v_3^{k+1}\right) \tag{277b}$$

where s_1^{k+1} and s_2^{k+1} are the residuals for the first and second blocks, with the third block having no residual.

This development can be generalized for a multi-block ADMM, for the n-th block the dual residual is given by

$$s_n^{k+1} = \mu^k \left(\frac{\partial g_c}{\partial v_n}^{k+1}\right)^T \sum_{m=n+1}^{N_b} \frac{\partial g_c}{\partial v_m} \left(v_m^{k+1} - v_m^k\right) \tag{278}$$

The dual variable can be written in terms of the subsystems by applying (278) to each node n in the block b,

$$s_{b,n}^k = \sum_{(n,\widehat{n}) \in C_{\widehat{b}>b}^{out}(n)} -\left(M_{(n,\widehat{n})}^{out}\right)^T M_{(n,\widehat{n})}^{in} \left(u_{\widehat{n}}^{k+1} - u_{\widehat{n}}^k\right) + \sum_{(\widehat{n},n) \in C_{\widehat{b}>b}^{in}(n)} -\left(M_{(n,\widehat{n})}^{in}\right)^T M_{(n,\widehat{n})}^{out} \left(y_{\widehat{n}}^{k+1} - y_{\widehat{n}}^k\right) \tag{279}$$

where

$$C_{\widehat{b}>b}^{in}(n) = \left\{(n_1, n) \in C \;\middle|\; n_1 \in \widehat{b} \wedge \widehat{b} > b\right\} \tag{280a}$$

$$C_{\widehat{b}>b}^{out}(n) = \left\{(n, n_2) \in C \;\middle|\; n_2 \in \widehat{b} \wedge \widehat{b} > b\right\} \tag{280b}$$

for a given node n in the block b. The notation $\widehat{b} > b$, means a block $\widehat{b}$ computed after block b.

We can interpret the subsystem's dual residual equation as the sum of mismatch of all variables that the subsystem depends on and appear in a later iteration block.

Once the equations for the residuals have been determined, we can derive the penalty parameter update strategies based on the rules presented in Section 3.2.3. The update rule based solely on the primal residual is given by

$$\mu^{k+1} = \begin{cases} \mu^k & \text{if } \left\| r^{k+1} \right\|_\infty \leq \gamma \left\| r^k \right\|_\infty \\ \beta\mu^k, & \text{otherwise} \end{cases} \tag{281}$$

The update rule based on the primal and dual residuals is given by

$$\mu^{k+1} = \begin{cases} \beta\mu^k & \text{if } \left\| r^{k+1} \right\|_\infty > \gamma \left\| s^{k+1} \right\|_\infty \\ \beta^{-1}\mu^k & \text{if } \left\| s^{k+1} \right\|_\infty > \gamma \left\| r^{k+1} \right\|_\infty \\ \mu^k, & \text{otherwise} \end{cases} \tag{282}$$

where $\beta > 1$ and $\gamma > 1$. Here the norms are in the functional sense since r^k and s^k are functions.

3.4.3 Fully Decoupling the Subsystems

The ADMM and the coordinate descent with augmented Lagrangian cannot iterate over all subsystems simultaneously; this becomes an issue as the network becomes more complex and denser, increasing the number of iteration blocks. The subsystems spend more time waiting for the computation of other subsystems than performing their computations. In the extreme, with a fully connected graph, each subsystem computes at a time, making the algorithm completely sequential. On the other hand, with a favorable network structure, there are only two computational steps.

For this reason, a modification to the structure of the dynamic network is proposed. This modification allows the parallelization of the solution of the subsystem OCPs, even for dense graphs.

The optimal control problems cannot be solved in parallel because two neighboring subsystems can not simultaneously solve their OCP. This issue can be avoided by including a new node between these two subsystems, such that they become non-neighbors, as shown in Figure 28.

For every connection $(n_1, n_2) \in C$, a new subsystem n_{n_1,n_2} with the following OCP is included in the network

$$\min J_{n_{1,2}} = 0 \tag{283a}$$

$$\text{s.t.: } y_{n_{1,2}} = u_{n_{1,2}} \tag{283b}$$

whose solution is trivial.

The connection (n_1, n_2) is then replaced by two new connections

$$u_{n_{1,2}} = M^{out}_{n_1,n_2} y_{n_1} \tag{284}$$

$$M^{in}_{n_1,n_2} u_{n_2} = y_{n_{1,2}} \tag{285}$$

where $u_{n_{1,2}} : [t_0, t_f] \to \mathbb{R}^{N^C_{n_{1,2}}}$ and $y_{n_{1,2}} : [t_0, t_f] \to \mathbb{R}^{N^C_{n_{1,2}}}$ are the inputs and outputs of the new subsystem, with $N^C_{n_{1,2}}$ being the number of variables shared in the connection between the subsystems n_1 and n_2.

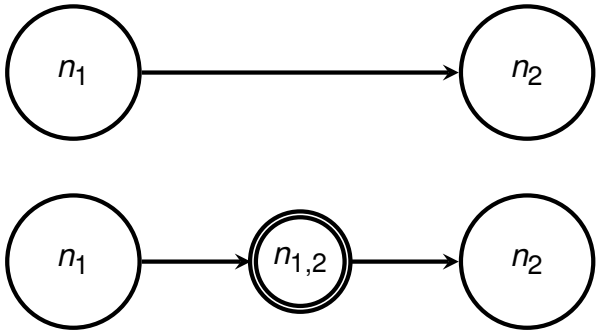


Figure 28 – Illustration of the new subsystem

When we apply the augmented Lagrangian to relax the connections of the network, the relaxed problem of the OCP (283) is given by

$$\begin{aligned} \min\ J_\mu(\nu_{n_1,n_{1,2}}, \nu_{n_{1,2},n_2}, y_{n_1}, u_{n_2}) = \int_{t_0}^{t_f} & \nu^T_{n_{1,2},n_2} \left[M^{in}_{n_1,n_2} u_{n_2} - y_{n_{1,2}} \right] \\ & + \nu^T_{n_1,n_{1,2}} \left[u_{n_{1,2}} - M^{out}_{n_1,n_2} y_{n_1} \right] \\ & + \frac{\mu}{2} \left\| M^{in}_{n_1,n_2} u_{n_2} - y_{n_{1,2}} \right\|^2 \\ & + \frac{\mu}{2} \left\| u_{n_{1,2}} - M^{out}_{n_1,n_2} y_{n_1} \right\|^2 dt \end{aligned} \tag{286a}$$

$$\text{s.t.: } y_{n_{1,2}} = u_{n_{1,2}} \tag{286b}$$

which can be easily solved as well.

If we apply these modifications for the network shown in Figure 26, the result is portrayed in Figure 29, with the graph already colored. With this modification, every original subsystem is a non-neighbor of any other original subsystem, allowing all the original subsystems to make their computations simultaneously. The second step is of easy solution, negligible compared to the OCP of each subsystem, and can similarly be performed in parallel.

This gain in parallelization probably does not come without cost. The introduction of intermediate nodes may act as a filter that slows down the communication between two neighbors, hence reducing the convergence speed. So there is a trade-off between the number of iterations versus the number of nodes computing simultaneously.

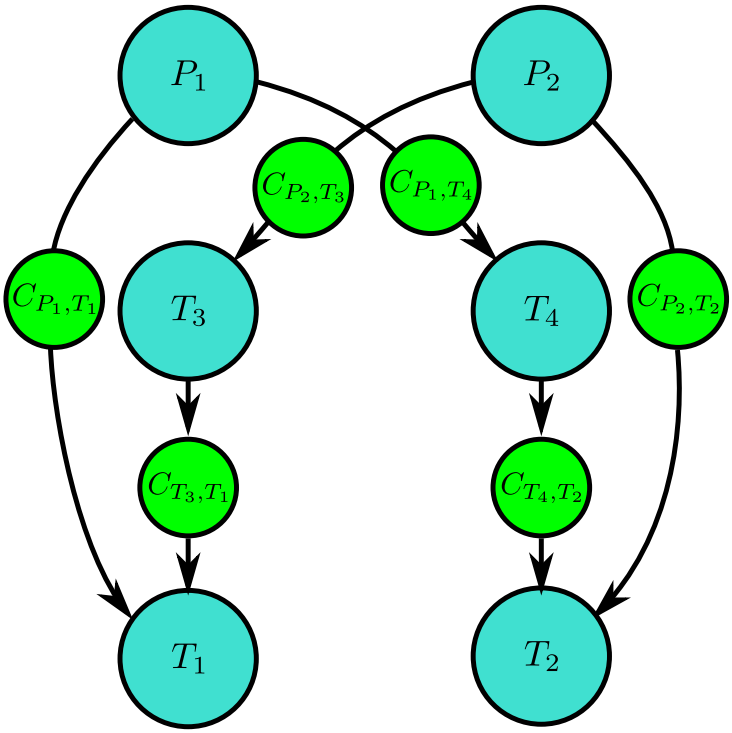


Figure 29 – Illustration of the network for the four tank problem with the introduction of the intermediate nodes

Alternatively, it is possible to strategically put these intermediate nodes to reduce the number of blocks in the graph. For instance, consider a five node cycle network (Figure 30a), the minimum number of colors is three. By strategically including one intermediate node between any of the nodes, in this case, the network becomes bipartite, as shown in Figure 30b. Because the application of this approach is highly situational dependent, this work does not explore this possibility further.

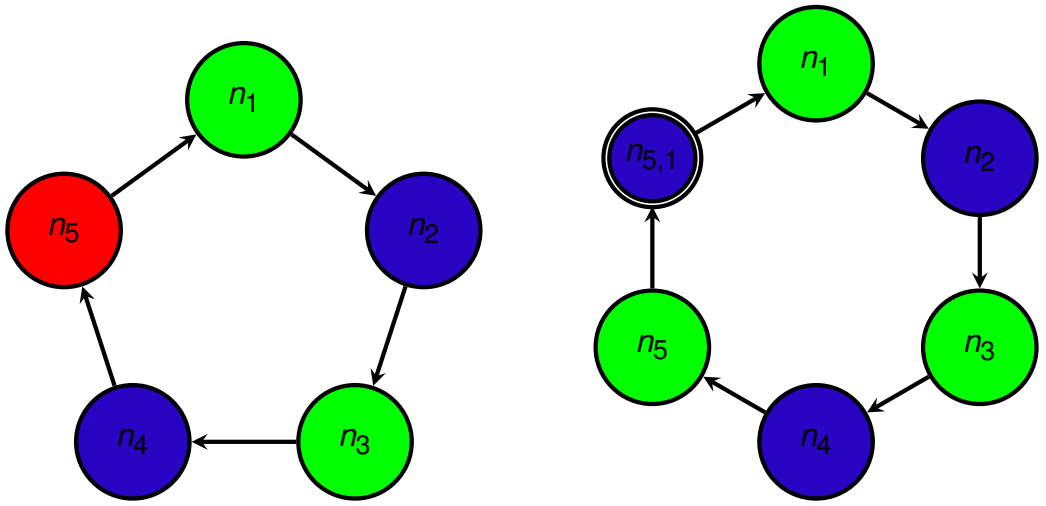
3.4.4 Jacobi ADMM

The relaxed subproblem (286) obtained for the introduced intermediate nodes are trivially solvable. As a matter of fact, it has an analytical solution.

Consider a system with two subsystems fully connected: all algebraic variables of the first subsystem are inputs of the second. These subsystems can be isolated by including an intermediate node between, as Figure 28 represents. For the intermediate node $n_{1,2}$, the Hamiltonian of subproblem (286) is given by

$$H = \nu_{n_{1,2}}^T \left(u_{n_{1,2}} - y_{n_{1,2}}\right) + \nu_{n_1,n_{1,2}}^T \left[u_{n_{1,2}} - y_{n_1}\right] + \nu_{n_{1,2},n_2}^T \left[u_{n_2} - y_{n_{1,2}}\right] + \frac{\mu}{2} \left\| u_{n_{1,2}} - y_{n_1} \right\|^2 + \frac{\mu}{2} \left\| u_{n_2} - y_{n_{1,2}} \right\|^2 \quad (287)$$

With the Hamiltonian, the necessary conditions for the subproblem can be devel-



(a) Network with 5 nodes require a minimum of 3 colors

(b) Network with 6 nodes can be colored with only 2 colors

Figure 30 – When a intermediate node is included in the cyclic network with 5 nodes (left), the minimum number of colors reduce from 3 to 2 (right)

oped for the variables of the subproblem

$$\frac{\partial H}{\partial u_{n_{1,2}}} = \nu^T_{n_{1,2}} + \nu^T_{n_1,n_{1,2}} + \mu\left(u_{n_{1,2}} - y_{n_1}\right)^T = 0 \tag{288a}$$

$$\frac{\partial H}{\partial y_{n_{1,2}}} = -\nu^T_{n_{1,2}} - \nu^T_{n_{1,2},n_2} - \mu\left(u_{n_2} - y_{n_{1,2}}\right)^T = 0 \tag{288b}$$

which implies

$$u_{n_{1,2}} = \frac{-\nu - \nu_{n_1,n_{1,2}} + \mu y_{n_1}}{\mu} \tag{289a}$$

$$\nu = -\nu_{n_{1,2},n_2} - \mu\left(u_{n_2} - y_{n_{1,2}}\right) \tag{289b}$$

By substituting ν in the equation above, we get

$$u_{n_{1,2}} = \frac{\nu_{n_{1,2},n_2} + \mu\left(u_{n_2} - y_{n_{1,2}}\right) - \nu_{n_1,n_{1,2}} + \mu y_{n_1}}{\mu} \tag{290a}$$

$$= \frac{\nu_{n_{1,2},n_2} - \nu_{n_1,n_{1,2}}}{\mu} + \left(u_{n_2} + y_{n_1}\right) - y_{n_{1,2}} \tag{290b}$$

$$= \frac{\nu_{n_{1,2},n_2} - \nu_{n_1,n_{1,2}}}{\mu} + \left(u_{n_2} + y_{n_1}\right) - u_{n_{1,2}} \tag{290c}$$

where the last equation is obtained by using the subsystem equations $y_{n_{1,2}} = u_{n_{1,2}}$. This allows us to find an optimal equation for the subsystem's control variable

$$u_{n_{1,2}} = \frac{\nu_{n_{1,2},n_2} - \nu_{n_1,n_{1,2}}}{2\mu} + \frac{\left(u_{n_2} + y_{n_1}\right)}{2} \tag{291}$$

which implies that the solution for subsystem OCP is the average of the two connected variables, but with the influence of the connections multipliers. The same analysis is valid for the algebraic variable.

The equations for the optimal control and algebraic variables given by (291) replace the need to solve the OCP for the intermediate nodes, making the network modification proposed in the previous section easier to be implemented. As the solution only requires the variables of the nodes involved in the connection, this implementation can be done in both n_1 and n_2, each one obtaining its identical local copy of $u_{n_{1,2}}$ and $y_{n_{1,2}}$. After the computation of the intermediate node is done, the multipliers are updated. For the connections between the nodes n_1 and n_2 with the intermediate node n_{n_1,n_2}, the multiplier updates are

$$\nu^{k+1}_{n_1,n_{1,2}} = \nu^{k}_{n_1,n_{1,2}} + \mu^k (u^k_{n_{1,2}} - y^k_{n_1}) \tag{292a}$$

$$\nu^{k+1}_{n_{1,2},n_2} = \nu^{k}_{n_{1,2},n_2} + \mu^k (u^k_{n_2} - y^k_{n_{1,2}}) \tag{292b}$$

By replacing the optimal solution (291) into the multipliers update, we get

$$\nu^{k+1}_{n_1,n_{1,2}} = \nu^{k}_{n_1,n_{1,2}} + \mu^k \left[\frac{\nu^k_{n_{1,2},n_2} - \nu^k_{n_1,n_{1,2}}}{2\mu^k} + \frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2} - y^k_{n_1} \right] \tag{293a}$$

$$= \frac{\nu^k_{n_1,n_{1,2}} + \nu^k_{n_{1,2},n_2}}{2} + \mu^k \frac{u^k_{n_2} - y^k_{n_1}}{2} \tag{293b}$$

likewise, for the other multiplier

$$\nu^{k+1}_{n_{1,2},n_2} = \nu^{k}_{n_{1,2},n_2} + \mu^k \left[u^k_{n_2} - \frac{\nu^k_{n_{1,2},n_2} - \nu^k_{n_1,n_{1,2}}}{2\mu^k} - \frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2} \right] \tag{294a}$$

$$= \frac{\nu^k_{n_1,n_{1,2}} + \nu^k_{n_{1,2},n_2}}{2} + \mu^k \frac{u^k_{n_2} - y^k_{n_1}}{2} \tag{294b}$$

By guaranteeing that $\nu^0_{n_1,n_{1,2}} = \nu^0_{n_{1,2},n_2}$, we have that $\nu^k_{n_1,n_{1,2}} = \nu^k_{n_{1,2},n_2}$ for all k. Let us call $\nu_{n_1,n_{1,2}}$ and $\nu_{n_{1,2},n_2}$ as ν_{n_1,n_2}. Therefore,

$$\nu^{k+1}_{n_1,n_{1,2}} = \nu^{k+1}_{n_{1,2},n_2} = \frac{\nu^k_{n_1,n_{1,2}} + \nu^k_{n_{1,2},n_2}}{2} + \mu^k \frac{u^k_{n_2} - y^k_{n_1}}{2} \tag{295a}$$

$$\nu^{k+1}_{n_1,n_2} = \nu^{k}_{n_1,n_2} + \mu^k \frac{u^k_{n_2} - y^k_{n_1}}{2} \tag{295b}$$

Notice that by stating that $\nu^k_{n_1,n_{1,2}} = \nu^k_{n_{1,2},n_2}$, the equation for the optimal control of the intermediate node (291) can be simplified

$$u_{n_{1,2}} = \frac{\left(u_{n_2} + y_{n_1}\right)}{2} \tag{296}$$

For the subsystem that comes before the intermediate node, in the case of Figure 29, the OCP objective is given

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \left(\nu^k_{n_1,n_2}\right)^T \left[u^k_{n_{1,2}} - y_{n_1}\right] + \frac{\mu^k}{2} \left\| u^k_{n_{1,2}} - y_{n_1} \right\|^2 dt \tag{297}$$

For a given k, the intermediate node optimal control is given by

$$u^k_{n_{1,2}} = \frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2} \tag{298}$$

By replacing (298), into (297) we get

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \left(\nu^k_{n_1,n_2}\right)^T \left[\frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2} - y_{n_1}\right] + \frac{\mu^k}{2}\left\|\frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2} - y_{n_1}\right\|^2 dt \tag{299}$$

which depends solely on the values of the subsystems in the previous iterations. The same can be achieved with the node after the intermediate node,

$$J_{n_2} = \int_{t_0}^{t_f} L_n + \left(\nu^k_{n_1,n_2}\right)^T \left[u_{n_2} - \frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2}\right] + \frac{\mu^k}{2}\left\|u_{n_2} - \frac{\left(u^k_{n_2} + y^k_{n_1}\right)}{2}\right\|^2 dt \tag{300}$$

which also does not depend on the intermediate node.

By doing these substitutions, we are effectively removing the dependence of the subsystems in the intermediate nodes. Thus, the intermediate nodes are disregarded and the system is back into its original configuration.

These developments can be generalized for any node in a network. Let us define the subproblem $\mathcal{P}_{BJ,n}$, which have the following form

$$\begin{aligned}
\mathcal{P}_{BJ,n} \min\ J_n = &\int_{t_0}^{t_f} L_n(x_n, y_n, u_n, t) \\
&+ \sum_{(n,n_2)\in C^{out}(n)} \left\{ \left(\nu^k_{n,n_2}\right)^T \left[\frac{M^{\text{in}}_{n,n_2} u^k_{n_2} + M^{\text{out}}_{n,n_2} y^k_n}{2} - M^{\text{out}}_{n,n_2} y_n\right] \right. \\
&\qquad \left. + \frac{\mu}{2}\left\|\frac{M^{\text{in}}_{n,n_2} u^k_{n_2} + M^{\text{out}}_{n,n_2} y^k_n}{2} - M^{\text{out}}_{n,n_2} y_n\right\|^2 \right\} \\
&+ \sum_{(n_1,n)\in C^{in}(n)} \left\{ \left(\nu^k_{n_1,n}\right)^T \left[M^{\text{in}}_{n_1,n} u_n - \frac{M^{\text{in}}_{n_1,n} u_n + M^{\text{out}}_{n_1,n} y^k_{n_1}}{2}\right] \right. \\
&\qquad \left. + \frac{\mu}{2}\left\|M^{\text{in}}_{n_1,n} u_n - \frac{M^{\text{in}}_{n_1,n} u_n + M^{\text{out}}_{n_1,n} y^k_{n_1}}{2}\right\|^2 \right\} dt
\end{aligned} \tag{301}$$

and the dynaminc equations of the subsystem n.

This OCP formulation allows for the development of a two-step algorithm that has all the subsystems iterating simultaneously. As depicted in Algorithm 10, in the first step, all the subsystems solve their OCPs, and the following step updates the multipliers. Here the algorithm is named Bipartite-Jacobi ADMM, as a reference to the Jacobi method for solving linear systems that iterate over all coordinates iterating in parallel, in opposition to the Gauss-Seidel method, which computes the coordinates

serially. The proposed algorithm is obtained by transforming the bipartite network in the ADMM framework. This strategy only requires modifying the objective function and the multiplier update, having no additional variable or constraint.

The algorithm requires an initial value for the multipliers estimates ν_n^0, for the penalty parameter μ^0, and initial guesses for u_n^0 and y_n^0. For this algorithm, the variables $u_n^{\text{ext},k}$ and $y_n^{\text{ext},k}$ are defned by

$$u_n^{\text{ext},k} = \left\{ u_{n_2}^k : (n, n_2) \in C^{out}(n) \right\} \tag{302a}$$

$$y_n^{\text{ext},k} = \left\{ y_{n_1}^k : (n_1, n) \in C^{in}(n) \right\} \tag{302b}$$

Algorithm 10 Bipartite-Jacobi ADMM

repeat
  for $n \in N$ **do in parallel:**
    $\left(y_n^{k+1}, u_n^{k+1}\right) \leftarrow solve \left\{ \mathcal{P}_{BJ,n} \left(\mu^k, \nu_n^k, y_n^{\text{ext},k}, u_n^{\text{ext},k}, u_n^k, y_n^k \right) \right\}$
  end for
  Update multipliers ν^k
  Update penalty factor μ^k
  $k \leftarrow k + 1$
until convergence

It is important to mention that Algorithm 10 is not equivalent to pure Jacobi ADMM for constrained optimization (HE, B., 2009; HE, B. et al., 2015). The Jacobi ADMM iterates over all variables in parallel, and at the computation of each coordinate, the other variables are kept at their last computed value. The Jacobi ADMM can be easily adapted for the solution of OCPs. An equivalent algorithm can be obtained by solving all the subsystems simultaneously, keeping the external values at their previously computed values. That is, if n_1 and n_2 have a connection (n_1, n_2), we substitute u_{n_2} with $u_{n_2}^k$, in the objective of the subsystem n_1. The objective of n_1 then becomes

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \left(\nu_{n_1,n_2}^k\right)^T \left[u_{n_2}^k - y_{n_1} \right] + \frac{\mu^k}{2} \left\| u_{n_2}^k - y_{n_1} \right\|^2 dt \tag{303}$$

While this is a much simpler strategy, it does not have good convergence properties, even for the two-block case (HE, B. et al., 2015). A few adjustments can be made to the algorithm to give it better convergence properties, as under relaxation steps. The reader can find more information about this strategy in (HE, B. et al., 2015; HE, B., 2009; HAN et al., 2014; JIANG; YUAN, X. M., 2010).

Deng et al. (2017) propose a modified version of the Jacobi ADMM with the inclusion of a proximal term along with the augmented Lagrangian term. The proximal term puts a penalization term between the computed value and the one obtained in the previous iteration. The algorithm can also be adapted to the OCP context by using

the augmented Lagrangian for OCP as a base and use equivalent algorithm steps. For instance, for a subsystem that has a relaxed connections $u_{n_2} = y_{n_1}$, and its variable is y_{n_1}, the proximal term is $\left\| y_{n_1}^k - y_{n_1} \right\|^2$. Therefore, the objective of the subsystem is given by

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \left(\nu_{n_1,n_2}^k\right)^T \left[u_{n_2}^k - y_{n_1}\right] + \frac{\mu^k}{2} \left\| u_{n_2}^k - y_{n_1} \right\|^2 + \frac{1}{2} \left\| y_{n_1}^k - y_{n_1} \right\|_P^2 dt \tag{304}$$

where P is a scalling matrix. The same objective can also be expressed in the scaled form

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \frac{\mu^k}{2} \left\| u_{n_2}^k - y_{n_1} + \frac{\nu_{n_1,n_2}^k}{\mu^k} \right\|^2 + \frac{1}{2} \left\| y_{n_1}^k - y_{n_1} \right\|_P^2 + \frac{1}{\mu^k} \left\| \nu_{n_1,n_2}^k \right\|^2 dt \tag{305}$$

as it is presented in the orignal paper. Notice that the last term is constant and can be disregarded.

The multiplier update is also adjusted,

$$\nu_{n_1,n_2}^{k+1} = \nu_{n_1,n_2}^k + \gamma\mu(u_{n_2}^{k+1} - y_{n_1}) \tag{306}$$

where $\gamma > 0$ is a dampig parameter.

The original Proximal-Jacobi ADMM makes no additional assumption on the problem, only on the additional parameters P and γ. By doing so, it can guarantee convergence of the algorithm.

In the generic form, the adatpted algorithm can solve a problem with the form

$$\begin{aligned}
\mathcal{P}_{PJ,n} \min\ J_n = \int_{t_0}^{t_f} & L_n(x_n, y_n, u_n, t) \\
& + \sum_{(n,n_2)\in C^{out}(n)} \left\{ \left(\nu_{n,n_2}^k\right)^T \left[M_{n,n_2}^{\text{in}} u_{n_2}^k - M_{n,n_2}^{\text{out}} y_n\right] \right. \\
& \left. + \frac{\mu}{2} \left\| M_{n,n_2}^{\text{in}} u_{n_2}^k - M_{n,n_2}^{\text{out}} y_n \right\|^2 + \frac{1}{2} \left\| M_{n,n_2}^{\text{out}} y_n - M_{n,n_2}^{\text{out}} y_n^k \right\|_{P_{n,n_2}^{\text{out}}}^2 \right\} \\
& + \sum_{(n_1,n)\in C^{in}(n)} \left\{ \nu_{(n_1,n)} \left[M_{n_1,n}^{\text{in}} u_n^k - M_{n_1,n}^{\text{out}} y_{n_1}^k\right] \right. \\
& \left. + \frac{\mu}{2} \left\| M_{n_1,n}^{\text{in}} u_n - M_{n_1,n}^{\text{out}} y_{n_1}^k \right\|^2 + \frac{1}{2} \left\| M_{n_1,n}^{\text{in}} u_n - M_{n_1,n}^{\text{in}} u_n^k \right\|_{P_{n_1,n}^{\text{in}}}^2 \right\} dt
\end{aligned} \tag{307}$$

and the same constraints (261b-261d). Algorithm 11 shows the steps of the algorithm.

Algorithm 11 Proximal-Jacobi ADMM

repeat
  for $n \in N$ **do in parallel:**
    $\left(y_n^k, u_n^k\right) \leftarrow solve\left\{\mathcal{P}_{PJ,n}\left(\mu^k, \nu_n^k, y_n^{\text{ext},k}, u_n^{\text{ext},k}, u_n^k, y_n^k\right)\right\}$
  end for
  Update multipliers ν^k
  Update penalty factor μ^k
  $k \leftarrow k+1$
until convergence

Deng's Jacobi-Proximal ADMM and the proposed Bipartite-Jacobi ADMM share some similarities. We can show that (299) and (304) have a similar objective. For brevity, the example of the connection between n_1 and n_2 is used.

Let us put (299) in the scaled form

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \frac{\mu^k}{2}\left\|\frac{\left(u_{n_2}^k + y_{n_1}^k\right)}{2} - y_{n_1} + \frac{\nu_{n_1,n_2}^k}{\mu^k}\right\|^2 + \frac{1}{\mu^k}\left\|\nu_{n_1,n_2}^k\right\|^2 dt \tag{308}$$

which can be rewritten as

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \frac{\mu^k}{2}\left\|\frac{\left(u_{n_2}^k - y_{n_1}\right)}{2} + \frac{y_{n_1}^k - y_{n_1}}{2} + \frac{\nu_{n_1,n_2}^k}{\mu^k}\right\|^2 + \frac{1}{\mu^k}\left\|\nu_{n_1,n_2}^k\right\|^2 dt \tag{309}$$

By applying the identity

$$\|a+b\|^2 = \|a\|^2 + 2a^T b + \|b\|^2 \tag{310}$$

into (309), the following is obtained

$$\begin{aligned} J_{n_1} = \int_{t_0}^{t_f} L_n + \frac{\mu^k}{2}\left\|\frac{\left(u_{n_2}^k - y_{n_1}\right)}{2} + \frac{\nu_{n_1,n_2}^k}{\mu^k}\right\|^2 \\ + \mu^k\left(\frac{\left(u_{n_2}^k - y_{n_1}\right)}{2} + \frac{\nu_{n_1,n_2}^k}{\mu^k}\right)^T\left(\frac{y_{n_1}^k - y_{n_1}}{2}\right) \\ + \frac{\mu^k}{2}\left\|\frac{y_{n_1}^k - y_{n_1}}{2}\right\|^2 + \frac{1}{\mu^k}\left\|\nu_{n_1,n_2}^k\right\|^2 dt \end{aligned} \tag{311}$$

If we define $P = \mu^k I$, we get

$$J_{n_1} = \int_{t_0}^{t_f} L_n + \frac{\mu^k}{2} \left\| \frac{\left(u_{n_2}^k - y_{n_1}\right)}{2} + \frac{\nu_{n_1,n_2}^k}{\mu^k} \right\|^2 + \mu^k \left(\frac{\left(u_{n_2}^k - y_{n_1}\right)}{2} + \frac{\nu_{n_1,n_2}^k}{\mu^k} \right)^T \left(\frac{y_{n_1}^k - y_{n_1}}{2} \right) + \frac{1}{2} \left\| \frac{y_{n_1}^k - y_{n_1}}{2} \right\|_P^2 + \frac{1}{\mu^k} \left\| \nu_{n_1,n_2}^k \right\|^2 dt \quad (312)$$

Notice that (312) is equivalent to (305) for the relaxed constraint $\frac{y_{n_1} - u_{n_2}}{2} = 0$ and the additional cross term. While it is not a clear indication that the Bipartite-Jacobi ADMM has the same properties as the Jacobi-Proximal ADMM, it instigates future developments.

3.5 NUMERICAL ANALYSIS

A few numerical experiments were conducted to compare the multiple methods presented in the previous section. The model used to perform the experiments is the four-tank system. The experiments use the modeling framework (10), which models each component as a subsystem.

The experiments are divided into two levels. The different variations for each algorithm are studied at the first level, e.g., multiplier update rule and iteration block rule. The different algorithms are compared at the second level while using the best-performing combination of variants for each algorithm.

The experiments evaluate the algorithms and their variants for robustness, quality, and performance. The reference for these experiments is the solution of the same OCP using a centralized approach. The experiments use 50 scenarios with sampled initial conditions in a predefined region to avoid cherrypicking and biased results. For fairness, the random number generator uses the same seed in all experiments.

For robustness, in each scenario, the algorithms are evaluated as having succeeded or failed. The solution method fails to converge if it takes more than 180 seconds or if the final objective function differs by more than 10 % of the baseline (centralized solution).

The quality comparisons evaluate the violation of the relaxed equation, and the difference between the algorithms results and the baseline, for the objective function and control profiles. The control difference is used as a measure because obtaining the controls is the ultimate goal of solving an OCP; therefore, if two methods obtain the same controls, they can be used interchangeably.

The performance comparison considers the time to converge from an initial guess far from an optimal solution. When relevant, the performance comparison will also include the number of iterations. In a receding horizon implementation, a good initial point is typically available. However, during the algorithm's initialization, one often cannot expect to have a nearly-optimal initial point. Note that in these experiments, we are dealing with the algorithm's initialization.

Summarizing, the algorithms and variations will be analyzed with respect to the following criteria:

- Convergence (converged scenarios / total scenarios);
- Quality:
  - Relaxed equation violation,
  - Objective function (algorithms vs centralized approach),
  - Maximum difference between control variables (algorithms vs centralized approach);
- Performance:
  - Convergence time,
  - Number of iterations.

The OCP's objective is to take Tank 1 and Tank 2 to the reference at 16 cm and 14 cm, respectively, while minimizing the variation of the control variables. The initial conditions of the system are centered at $x_0^T = [12.4, 12.7, 1.8, 1.4]$ for the tanks and $[3.0, 3.0]$ for the pumps. The sampled initial conditions are in the range $(0\%, 200\%]$ of these values. In these experiments, no constraints were applied, except for the physical constraint that the level of each tank cannot go to zero [8].

The algorithms, OCPs, and subsystems were implemented in Python using YAOCPTool, and CasADi (ANDERSSON et al., 2019). The resulting optimization problems were solved using IPOPT (WÄCHTER; BIEGLER, L. T., 2006). Since this work is academic and performance is not the primary goal, a couple of design choices were made focusing on the ease of implementation. The main process manager performs the synchronization and data preparation for each optimization. The optimization problems are solved in different processes[9]. The experiments were performed in a computer equipped with an Intel 7700k (4 cores/8 threads) processing unit and 16 GB DDR4 of RAM, running Ubuntu 20.04 in WSL.

[8] The tank level cannot go to numbers lower than zero; otherwise, the flow is undefined due to the square root in the formula. The same square root causes the Jacobian to be undefined when the flow is zero since its derivative is $\frac{1}{h}$.

[9] The term process implies independent computer processes, which do not share memory addresses, and which need to communicate through a socket.

The following techniques are evaluated:

- Augmented Lagrangian method with coordinate descent (AL-CD),
- Alternating direction multiplier method (ADMM),
- Fully decoupling the subsystems,
- Bipartite-Jacobi ADMM.

Unless stated otherwise, the following configurations were used in the experiments. The OCP was discretized using the collocation method with 60 finite elements. The collocation method uses degree three polynomials, where Radau's tableau defines the collocation points. The initial value for the parameters are $\mu = 1$ and, for the multiplier estimate, $\nu(t) = 0$. We use a null multiplier estimate given the lack of previous information. A better initial estimate would undoubtedly accelerate the convergence. The bounded increase rule was used to update the penalty parameter (128), with $\beta = 2$ and $\mu_{max} = 10^4$. The exit condition for the algorithms is an absolute violation of the relaxed constraints smaller than 10^{-3}, which has to be reached within 1000 iterations.

3.5.1 Coordinate Descent with Augmented Lagrangian

The coordinate descent have many variations, as discussed in Section 3.2.1. In the experiments developed for the coordinate descent with the augmented Lagrangian method, we will investigate how the block choice rule affects the algorithm efficiency.

The rules are:

- Cyclic - the blocks are always chosen in the same order,
- Sample - choose a block without repeating until all blocks have been iterated over.
- Choices - choose a block of the ones available, regardless of the previous iterated blocks.

The CD loop continues until it reaches a relative objective decrease smaller than 10^{-4} or 30 iterations. At the first augmented Lagrangian iteration, a more significant number of CD iterations is allowed, up to 100 iterations.

The first experiment was using the network split into two blocks:

1. Pump 1, Tank 2, and Tank 3;
2. Pump 2, Tank 1, and Tank 4.

Two Blocks

The results for the first experiment is shown in Table 3. Except for the number of convergent scenarios, all numbers are the average value of all scenarios.

Regarding the number of converged scenarios, the choices method falls far behind, only solving 13 scenarios. Despite it being faster than other methods, it converges to solutions far from the baseline objective. This issue may be caused by it eventually solving two times the same block, which hinders dual convergence. Comparing cyclic and sample approaches, we can see an apparent compromise between the time and solution quality. While the latter converges about 25 % faster (on average), its controls are further away from the baseline. The same observation can be made for the objectives, with the cyclic method achieving virtually the same objective as the baseline.

Table 3 – Coord. Descent with Aug. Lagrangian for the network split into 2 blocks

	Cyclic	Sample	Choices
Convergence (total 50)	49	49	13
Objective	0.09%	1.19%	6.80%
Relaxed eq. violation	4.3×10^{-4}	3.7×10^{-4}	5.5×10^{-4}
Control difference	0.13	0.40	1.41
Convergence Time (s)	29.05	22.47	15.15
Iterations	305.05	151.10	88.46

Three Blocks

In the experiment with the three blocks, the blocks were grouped as follows

1. Pump 1 and 2,
2. Tank 1 and 2,
3. Tank 3 and 4.

In this experiment, we can discern a pattern for the three methods. As the results of Table 3 show, the choices method fails both in consistency and quality. The comparison of the cyclic and sample methods favors towards the cyclic, given that the sample method delivers an acceptable result in 43 of the 50 scenarios. In contrast, the cyclic approach converges in all scenarios. Although the sample is still a valid alternative.

Based on these results, we observe that the cyclic approach is more robust to variations in the quality of the initial condition and results in a better quality solution. It all comes with a cost; it is slower than the other compared methods.

Table 4 – Coord. Descent with Aug. Lagrangian for the network split into 3 blocks

	Cyclic	Sample	Choices
Convergence (total 50)	50	43	13
Objective	0.16%	2.62%	3.76%
Relaxed eq. violation	4.0×10^{-4}	5.3×10^{-4}	4.8×10^{-4}
Control difference	0.15	0.62	1.01
Convergence Time (s)	32.63	21.85	20.35
Iterations	308.34	187.18	159.23

3.5.2 ADMM

The ADMM experiments explore how different update rules affect the performance indicators. The three compared techniques are:

- Constant μ (constant),
- updates based on the primal violation (primal),
- updates based on the primal and dual violations (primal-dual).

The method's implementation uses an initialization phase to avoid the algorithm diverging due to a poor initial condition. The blocks are iterated until a satisfactory decrease in the objective, indicating a slight divergence between the subsystems. During this initialization phase, the update of the multiplier estimates and the penalty parameter is skipped.

The algorithms were parametrized with an initial penalty parameter $\mu_0 = 1$, a multiplier increase parameter $\beta = 3$, and a multiplier decrease parameter $\beta_{decrease} = 1.2$ (used in the primal-dual method).

Two Blocks

The first set of experiments compare the methods when the network is split into two blocks:

1. Pump 1, Tank 2, and Tank 3;
2. Pump 2, Tank 1, and Tank 4.

The experiments are presented in Table 5. The results show that the primal method is more reliable, solving all but one of the scenarios within the convergence constraints. The primal-dual follows behind, only missing three of the 50 scenarios. At last, the constant method was not able to solve eight cases. Concerning the quality of solutions, both constant and primal-dual approaches produced solutions virtually equal to the baseline. The primal method produces solutions close to the baseline;

however, not as good as the other two methods. Regarding convergence speed, the primal method is faster by a good margin but with the trade-off of the solution quality.

Table 5 – ADMM for the network split into 2 blocks

	Constant	Primal	Primal-Dual
Convergence (total 50)	42	49	47
Objective	0.01%	1.52%	0.01%
Relaxed eq. violation	9.1×10^{-4}	5.7×10^{-4}	9.2×10^{-4}
Control difference	0.03	0.35	0.02
Convergence Time (s)	29.25	19.95	32.09
Iterations	167.19	113.67	198.93

Three Blocks

A second experiment was performed with the ADMM, this time splitting the system into three blocks:

1. Pump 1 and 2,
2. Tank 1 and 2,
3. Tank 3 and 4.

The results for this experiments are displayed in Table 6. The conclusions for the three blocks case are very similar to the two blocks case. The primal method has a faster convergence at the cost of providing the worst quality solutions. As the constant and primal-dual methods are slower to converge, they end up hitting the time constraint and are considered to have failed. This issue is reflected in them having a worse perceived convergence rate.

One exciting factor when comparing the 2 and 3 blocks cases is that they both take about the same average time to solve, regardless of the number of blocks. In particular, the constant method takes 29 seconds on average to obtain a solution for both cases, even though the three blocks case has almost 50 % more iterations. This outcome might result from the increase in the subproblems' difficulty when there are fewer blocks.

In the ADMM method, the penalty parameter has a vital role in balancing primal and dual violations. Increasing the penalty makes the primal convergence reduce, at the cost of increasing the dual violation. The contrary is also valid; decreasing the penalty induces a lower dual violation and increases the primal violation.

Since the primal method can only increase the penalty parameter, the algorithm reaches faster the termination condition (primal violation $\leq 10^{-3}$), given that it will favor a reduction in the primal violation over the dual violation. Likewise, if we were to

Table 6 – ADMM for the network split into 3 blocks

	Constant	Primal	Primal-Dual
Convergence (total 50)	43	49	46
Objective	0.01%	1.44%	0.04%
Relaxed eq. violation	9.0×10^{-4}	6.3×10^{-4}	9.3×10^{-4}
Control difference	0.03	0.57	0.09
Convergence Time (s)	29.39	20.53	28.65
Iterations	249.14	169.95	164.44

increase the initial penalty in the constant method ($\mu_0 = 1$) to a higher value, it would make it converge faster, at the cost of the solution quality. On the other hand, the primal-dual method can adjust the penalty to keep a proportional decrease in primal and dual violations. This adjustment may increase the solution, which the constant method does not suffer. A more robust convergence is expected from the primal-dual, regardless of the initial penalty parameter, due to the adaptability.

This effect is further exacerbated by the fact that the ADMM has good general convergence. However, the ADMM does no accelerate as the optimality gap closes, as do other methods like Newton's step.

3.5.3 Fully Decoupling the System

The experiments on fully decoupling the system investigate how the strategy proposed in Section 3.4.3 changes the results of the previous investigated methods. The following combination of methods and variants are chosen for this experiment:

- Coordinate descent with augmented Lagrangian using cyclic block selection,
- ADMM with constant penalty parameter,
- ADMM with primal-dual penalty parameter adjustment rule.

These methods are chosen because they render a solution virtually equal to the baseline.

The results are in Table 7. With the decoupling of the systems, we can see that all the methods still produce solutions with less than a 1 % objective difference to the baseline. The same is valid for the difference in the control variables, although the coordinate descent method shows a more prominent difference. The coordinate descent was able to solve all the scenarios, proving to be quite reliable. On the other hand, both ADMM variants solved fewer scenarios than in the previous experiments. Finally, regarding performance, they are all slower than without the network modifications; this is likely a consequence of the increased number of iterations used in all methods. The increase in iterations is directly connected to an increase in the system sparsity, making

the interactions between subsystems take more iterations to propagate. Notice that the number of iterations presented in the table accounts for iterations on all nodes, original and intermediate nodes.

Table 7 – Fully decoupling the System

	CD-AL Cyclic	ADMM (Constant)	ADMM (Primal-Dual)
Convergence (total 50)	50	38	43
Objective	0.72%	0.15%	< 0.01%
Relaxed eq. violation	6.0×10^{-4}	9.2×10^{-4}	8.9×10^{-4}
Control difference	0.28	0.10	< 0.01
Convergence Time (s)	62.34	62.65	70.97
Iterations	392.88	298.18	370.36

3.5.4 Bipartite-Jacobi ADMM

The final set of experiments were performed using the Bipartite-Jacobi ADMM method. This experiment hopes to elucidate any possible benefits over the fully decoupled network strategy using the ADMM method, which is very similar in structure. For this matter, both variants for the ADMM were used

- ADMM with constant penalty parameter,
- ADMM with primal-dual penalty parameter adjustment rule.

The only difference is that the objective function and the update rule were adjusted accordingly.

From the results shown in Table 8, both methods achieve a meager difference in the objective value to the baseline. In the same manner, the control difference is negligible. The number of converged scenarios registers a good increase compared to the fully decoupled strategy; this increase is likely related to the reduction in the convergence time. One thing to notice is that the method uses a reduced number of iterations, fewer iterations than any of the other methods tested. A smaller number of iterations imply less communication time, which is particularly advantageous to geographically distributed systems.

3.5.5 Overall Comparison

To wrap up the analysis of the experiments, let us compare the outstanding performances of the experiments. The criteria for choosing the methods for this final analysis are algorithms that yielded results that are close to the baseline, had a good convergence in the test scenarios, and have a good performance. Under these criteria, the following methods and results are chosen:

Table 8 – Bipartite-Jacobi ADMM

	BJ-ADMM (Constant)	BJ-ADMM (Primal-Dual)
Convergence (total 50)	43	45
Objective	0.05%	< 0.01%
Relaxed eq. violation	9.5×10^{-4}	8.9×10^{-4}
Control difference	0.07	0.01
Convergence Time (s)	36.95	43.04
Iterations	97.60	128.44

- Augmented Lagrangian with coordinate descent (2 blocks) - denoted as a *AL-CD (Cyclic)*,
- ADMM with constant penalty parameter (3 blocks) - aliased as a *ADMM (Const)*,
- ADMM with primal-dual update rule (3 blocks) - aliased as a *ADMM (PD)*,
- Bipartite-Jacobi ADMM with primal-dual update rule (3 blocks) - aliased as a *BJ-ADMM (PD)*,

The results of each of these methods are synthesized in Table 9. In all of these results, the objective and control are very close to the baseline, which shows that all presented methods can achieve the same results as the centralized approach, at least for the problem presented in these experiments. Regarding the converge of these algorithms, they all solved 85+ % of the scenarios. Even greater converge rates could be achieved by tuning the parameters to make the algorithms more robust. The algorithms have very similar performance except for the BJ-ADMM, which was about 50 % slower than the others. On the other hand, the BJ-ADMM is the method with the least number of algorithm iterations, which means that an increase in the communication cost would worsen it the least. Also, the BJ-ADMM does not depend on finding the optimal block coloring strategy. The ADMM with a primal-dual rule is particularly interesting because its performance does not depend on the initial penalty, making it closer to a “plug-and-play” experience. The Augmented Lagrangian with coordinate descent performed surprisingly well. Other works have reported that this method was less reliable than ADMM, at least for optimization variants. However, from the results found in this work, the Augmented Lagrangian with coordinate descent was the method that solved almost all of the proposed scenarios while keeping comparable performance.

3.5.6 Discussion

The experiments presented in this section portray how the many techniques available for the traditional augmented Lagrangian method and ADMM can be reinterpreted in the optimal control context. The particular advantage of strategies like these

Table 9 – Overall comparison over the best results

	AL-CD (Cyclic)	ADMM (Const)	ADMM (PD)	BJ-ADMM (PD)
Convergence	49	43	46	45
Objective	0.09%	0.01%	0.04%	$< 0.01\%$
Rel. eq. violation	4.3×10^{-4}	9.0×10^{-4}	9.3×10^{-4}	8.9×10^{-4}
Control difference	0.13	0.03	0.09	0.01
Conv. Time (s)	29.05	29.39	28.65	43.04
Iterations	305.05	249.14	164.44	128.44

is that by decoupling the subsystem before the discretization step, we are free from the details of how each subsystem will solve its subproblems. Therefore, we can have subsystems using different methods and discretization strategies, and they still can "talk" with each other as long as they use this higher-level interface.

The matter of which and how these standard optimization strategies can be adapted to the optimal control context is a whole new area for investigation. Further, defining which combinations of methods perform better and what their trade-offs are is an ongoing process. Hence, these explorations are too broad to be covered in one thesis and can undoubtedly be in the scope of future publications.

An important note concerning the values presented in the experiments is that the methods were implemented with the practicality of implementation in mind. By doing so, the performance is most likely hindered in the process. There are indeed many points for improvement, given the vastity of implementation details, such as optimization solver, multiprocessing communication, and multilanguage libraries. These issues affect different algorithms in different ways. A method that relies a lot on communication would benefit the most by reducing the payload of the communication. In contrast, an algorithm that solves a few but challenging iterations would benefit the most from optimization solver improvement. Also, several tuning parameters could be adjusted to enhance the performance, but this kind of discovery is typically reserved for practitioners.

Nonetheless, by reporting the results of these experiments, the author feels that these methods have been validated and would invite further contributions on the topic.

4 APPLICATION: AN OIL AND GAS PRODUCTION NETWORK

4.1 MOTIVATION

The experiments shown at the end of the previous chapter, while extensive, do not adequately represent the kind of systems typically found on real-life applications. The four-tanks system has a quasi-linear model, given that the only nonlinearity comes from the square root on the flow through-an-orifice equation. These equations can cause numerical issues if the solver attempts to evaluate the equations with a height of zero or less, but it is well behaved on the regular operating region.

For this reason, in this chapter, we investigate how the distributed algorithms presented in the previous sections handle a more complex system. The system chosen is composed of two oil and gas production wells and a vertical riser. These systems have a more significant number of internal states and algebraic variables, whose equations are full of all kinds of nonlinearities.

Production wells and risers face a particular challenging phenomenon involving the formation of gas slugs in the pipelines. This issue causes a limit cycle which disturbs production and increases wear in the equipment. This is a well-known challenge that has been the focus of many studies.

In the early 2000's works focusing on using control strategies to stabilize the wells started to bloom. Hu and Golan (2003) used Schlumberger's OLGA simulator to develop a feedback loop controller to stabilize the well. Eikrem et al. (2004) developed a simplified well model which behaves very similar to OLGA; they proceeded to compare two approaches: a PI controller using an estimated downhole pressure and a nonlinear controller. Eikrem et al. (2008) showed a simple well model and proposed three control strategies that were able to stabilize the well in simulations and laboratory experiments. Plucenio et al. (2009) proposed the use of a nonlinear MPC, which relies upon a Hammerstein model to describe the behavior of the gas-lift well. Plucenio et al. (2012) used a nonlinear MPC strategy to control a simplified model, similar to phenomenological models proposed by Eikrem et al. (2004). To control a gas-lift well, Jahanshahi et al. (2012) propose a robust control and provided some robustness analysis. Krishnamoorthy et al. (2016) models and performs robustness analysis in wells that have electrical submersible pumps (ESPs). Diehl et al. (2017) developed a single model for the well and riser, where one is directly connected to the other; this situation is quite common in satellite wells. Jean P. Jordanou et al. (2018) used a echo-state neural network to model the riser and developed a MPC control strategy for the model.

On the riser side, a few works proposed strategies to eliminate oscillations caused by gas slugs. Early works focused on modeling this behavior solely for the intent of simulation and finding the settings that reduced the formation of gas slugs in the riser (TAITEL et al., 1989; TAITEL; BARNEA, 1997; MASELLA et al., 1998). Later

works focused on developing simplified and compact models for control and the development of MPCs. Di Meglio et al. (2009) proposed a first-principles model for vertical risers and performed a stability analysis. Jahanshahi and Skogestad (2011) proposed a riser model, including horizontal part (named pipeline), and compared it to other works available in the literature. Stasiak et al. (2012) proposed a discrete-time control strategy that can stabilize the riser simulated using OLGA.

The strategies proposed thus far have all relied on a centralized approach, which is not a huge issue if only a small number of wells and risers are part of the system or if it does not integrate with other network equipment. For instance, Sayda and Taylor (2007) proposed a three-state separator model that is well suited for control and MPC. Grong (2009) developed an elaborate compressor model that could be simplified for the intent of control. Budinis and Thornhill (2015) proposed a compact compressor model and MPC that can prevent compressor surges.

Some works have attempted integrating multiple components within the same problem. Willersrud et al. (2012) compared the use of two nonlinear MPC approaches for system modeled using the commercial modeling tool Dymola. Aguiar et al. (2015) developed a centralized solution for a simplified network containing wells, risers, separators, compressors, and a gas-lift injection line.

However, it is challenging to develop a centralized approach that achieves a systemwide control of the production, where each subsystem is cooperating with others to stabilize and reach the production goals and environmental and operational constraints. The problems faced range from technical to organizational.

Given the large number of components present in an oil production plant, the optimal control problem and the optimization problem resulting from the discretization will have several variables and constraints. A large number of variables will imply long-running optimization problems and require a computer with plenty of memory and processing power. Many constraints will define a narrow and complex solution space that might hinder initialization and algorithm convergence.

Regarding the organizational issues, it is often the case that different teams are responsible for different parts of the production plant. These teams are the specialists on the given process and are well-versed in its intricacies. Its also the responsibility of each team to develop and maintain models, operational boundaries, and constraints; and define the best control strategy for that particular part of the system. Therefore, it is quite a difficult challenge to coordinate several teams into a single highly-coupled control approach.

For these reasons, a distributed and cooperative strategy offers the distinct advantage of enabling each team to be responsible for its own model, optimal control problem and solution approach. Such a strategy still allows for an alignment that leads to an optimal plant operation, unlike a purely decentralized strategy.

In this chapter, a proof of concept is developed of such an approach. The test case used consists of a small-scale system with two wells and a riser. The system is modeled and controlled using an MPC strategy based on the algorithms presented in the previous chapter.

In Section 4.2 the network model is described, Section 4.3 describes the control objective and the implemented constraints, Section 4.4 goes through the details of the numerical experiments, Section 4.5 reports the experiment results and analysis, finally Section 4.6 concludes the chapter with final notes.

4.2 NETWORK MODEL

The experimental plant is composed of two oil and gas production wells connected to a common manifold, where the outflow goes to a vertical riser. Figure 31 shows the components of the network.

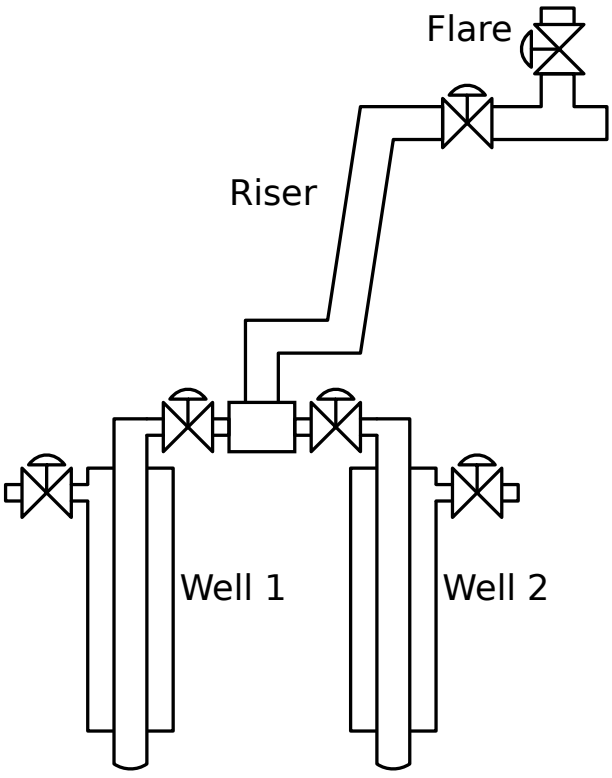


Figure 31 – Illustration of the oil and gas production network.

Notice that because the wells' outflow goes into the riser, the riser has a connection for each well subsystem. Also, by sharing the same riser, the wells are connected to the same input pressure of the riser, which is a consequence of the inflow and the riser's upstream separator pressure. From this analysis, the network's graph can be drawn, depicted in Figure 32.

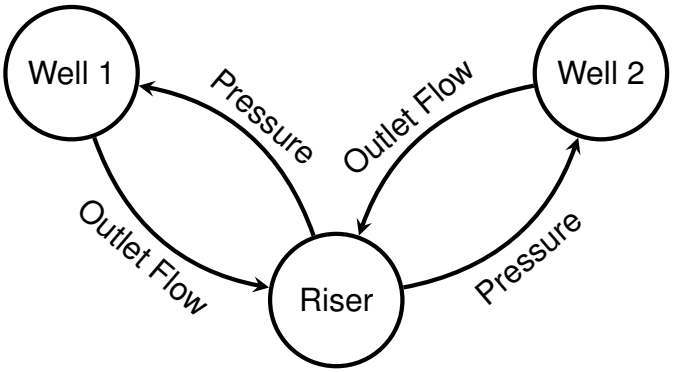


Figure 32 – Graph repsesentation of the oil production network

While not having as many components as the four-tanks system, this system contains a more complex topology, given that there is a reference loop between the well and riser nodes, and both well nodes depend on the same variable. The latter point makes the overall problem more complex since the inlet pressure needs to be "negotiated" between the three systems. This kind of flow-pressure pattern would repeat if we had more components in the network given the nature of the system[1].

In the following, the well and riser models are presented, along with a description of the physical process that occurs in each component.

4.2.1 Well model

In oil wells, the fluid emanates naturally if the reservoir pressure is high enough to push the liquid column to the surface. As the well is drained, the pressure of the reservoir drops, and eventually, it becomes insufficient to produce oil from the well without assistance. In such cases, artificial lift techniques can be applied. These techniques not only apply for extending the life of wells, but they can also boost the production of young wells, if the reservoir pressure is low.

Gas-lift is an artificial lift technique that consists of injecting pressurized gas in the annulus, where it flows into the bottom of the tubing. The inflow of gas reduces the density of the fluid column, making it lighter and increasing flow, since the pressure required to compensate for gravity is reduced.. The injected gas is recycled and later reinjected (JAHANSHAHI et al., 2012). In the continuous gas-lift process, there is a constant high-pressure gas injection into the production column, reducing the fluid density, increasing the net pressure available to cause flow. The intermittent gas-lift process uses an injection of gas into the fluid column to displace it. It is an artificial lifting method identical to others, where an ideal pressure difference is created to produce the desired flow. This can be used to restart a well after shutdown. There is

[1] Likewise, a similar current-voltage pattern would appear in electrical circuits

also artificial lift via pump off, which mechanically lifts the fluid from the reservoir with sequential up and down strokes (CAMPONOGARA et al., 2015).

Sometimes, a gas lift induces an unstable behavior by the formation of gas slugs. This is characterized by variations in both pressure and production rate and an oscillatory flow (JAHANSHAHI et al., 2012). Therefore the need for a controller to stabilize and induce a stable and safe operation.

As the gas-lift injection affects the flow, there is room for optimization. If no gas is injected, the outflow is too small; if too much gas is injected, actually, there is a reduction in the produced oil since most of the emanating fluid is the injected gas.

The gas lifted well model used in this case study was based on the model proposed in Krishnamoorthy et al. (2016). According to the authors, the model describes the production from each gas lifted well through the mass balance of the different phases, the density models, the pressure models, and the flow models.

The model states are the mass of gas in the annulus (m_{ga}), and the mass of gas and oil in the tubing (m_{gt} and m_{ot}, respectively). The mass balances in each well are given by:

$$\dot{m}_{ga} = w_{gl} - w_{iv} \tag{313}$$

$$\dot{m}_{gt} = w_{iv} - w_{pg} + w_{rg} \tag{314}$$

$$\dot{m}_{ot} = w_{ro} - w_{po} \tag{315}$$

where:

- w_{gl} is the gas-lift injection rate,
- w_{iv} is the gas flowing from the annulus into the tubing,
- w_{pg} is the produced gas flow rate,
- w_{po} is the produced oil flow rate,
- w_{rg} is the gas flowing from the reservoir,
- w_{ro} is the oil flowing from the reservoir.

The flows thrugh the injection valve (w_{iv}) and the total fluid flow through the production choken (w_{pc}) are defined by valve equations; w_{pg} and w_{po} are consequences of w_{pc} and the ratio of mass gas and oil in the tubing; and w_{ro} and w_{rg} are obtained

from the reservoir productivity equations. The equations are

$$w_{iv} = C_{iv}\sqrt{\rho_{ai}(p_{ai} - p_{wi})} \tag{316}$$

$$w_{pc} = C_{pc}\sqrt{\rho_m(p_{wh} - p_m)} \tag{317}$$

$$w_{pg} = \frac{m_{gt}}{m_{gt} + m_{ot}} w_{pc} \tag{318}$$

$$w_{po} = \frac{m_{ot}}{m_{gt} + m_{ot}} w_{pc} \tag{319}$$

$$w_{ro} = PI(p_{res} - p_{bh}) \tag{320}$$

$$w_{rg} = GORw_{ro} \tag{321}$$

where:

- p_{ai} - annulus pressure,
- p_{wh} - wellhead pressure,
- p_{wi} - well injection point pressure,
- w_{pc} is the total flow through the production choke,
- C_i is the valve flow coefficient for the downhole injection valve,
- C_p is the valve flow coefficients for the production choke,
- PI is the reservoir productivity index,
- p_{res} is the reservoir pressure,
- p_m is the manifold pressure and,
- GOR is the gas–oil ratio.

The annulus pressure at the injection point (p_{ai}) is obtained with the ideal gas law and the increase in pressure due to the gas column. The wellhead pressure (p_{wh}) is obtained with ideal gas law. The tubing pressure at the injection point (p_{wi}) is obtained by adding the fluid column to the wellhead pressure (p_{wh}). The bottom hole pressure (p_{bh}) is the pressure at the injection point plus the pressure from the liquid column down to the point of contact with the reservoir. These pressures are given by the following equations

$$p_{ai} = \left(\frac{RT_a}{V_a M_g} + \frac{gL_a}{V_a}\right) m_{ga} \tag{322}$$

$$p_{wh} = \frac{T_w R}{M_g}\left(\frac{m_{gt}}{L_w A_w + L_{bh}A_{bh} - \frac{m_{ot}}{\rho_o}}\right) \tag{323}$$

$$p_{wi} = p_{wh} + \frac{gL_w}{A_w L_w}(m_{ot} + m_{gt} - \rho_o L_{bh} A_{bh}) \tag{324}$$

$$p_{bh} = p_{wi} + g\rho_o L_{bh} \tag{325}$$

where:

- M_g is the molecular weight of the gas,
- R is the gas constant,
- T_a is the temperature in the annulus,
- L_a is the length of the annulus,
- V_a is the volume of the annulus,
- T_w is the temperature in the well tubing,
- L_{bh} and L_w is the vertical height of the well tubing below and above the injection point,
- g is the gravity acceleration constant.

The gas density in the annulus ρ_{ai} is obtained by assuming that the gas behave according to the ideal gas law; and the density of the fluid in the tubing (ρ_m) is obtained by the mass of oil and gas in the tubing, the gas density in the tubing, and the oil density:

$$\rho_{ai} = \frac{M_g}{RT_a} p_{ai} \tag{326}$$

$$\rho_m = \frac{(m_{gt} + m_{ot}) p_{wh} M_g \rho_o}{(m_{ot} p_{wh} M_g + \rho_o R T_w m_{gt})} \tag{327}$$

where:

- ρ_o is the density of oil in the reservoir,
- L_r and L_w is the length of the well above and below the injection point,
- A_r and A_w is the cross-sectional area of the well above and below the injection point.

The data used for the parametrization of both wells is available in (KRISHNAMOORTHY et al., 2016).

4.2.2 Riser Model

The riser is the pipeline that transports the produced fluids from the sea bed to the platform, where the fluids are processed. It has a simple structure, consisting of a simple vertical pipe. In the seabed, the pipe is connected to a manifold, which routes the outflow of the well into a specific riser. Here we are taking the manifold as part of the riser model. The riser outlet is connected to the separator and the flare. The flare

is a safety component; if the system cannot handle the produced gas for some reason, the flare can burn the excess gas. As one would expect, flaring gas is undesirable, it is a direct loss of revenue, and it is limited by environmental regulation. The flare is also included in this riser model.

The riser model is based on (JAHANSHAHI; SKOGESTAD, 2014). The authors propose a four-state model that represents a pipeline with a riser at the end. In this work, the pipeline is not included, so only the two states representing the riser dynamics are used.

The riser differential equations are simply mass balances for the gas (m_{gr}) and liquid (m_{or}) phases

$$\dot{m}_{gr} = \sum_{i=1}^{n_i} (w_{pg}^i) - w_{tg} \tag{328}$$

$$\dot{m}_{or} = \sum_{i=1}^{n_i} (w_{po}^i) - w_{to} \tag{329}$$

where:

- w_{pg} is the produced gas inflow from well i,
- w_{po} is the produced oil inflow from well i,
- w_{tg} is the gas outflow leaving at the top of the riser,
- w_{tg} is the oil outflow leaving at the top of the riser,
- n_i is the number of wells.

The gas and oil labeloutflow are given by the total flow through valve and by the ratio of gas and oil in the riser

$$w_{pr} = C_{pr}\sqrt{\rho_r(p_{rh} - p_s)} \tag{330}$$

$$w_{to} = (1 - \alpha_g)w_{pr} \tag{331}$$

$$w_{tg} = (\alpha_g)w_{pr} \tag{332}$$

$$\alpha_g = \frac{m_{gr}}{m_{gr} + m_{or}} \tag{333}$$

where:

- w_{pr} is the total fluid outflow at the top of ther riser,
- α_g is ratio of gas in the riser,
- p_{rh} is pressure at the riser outlet,
- p_s is the separator pressure.

The riser head pressure (p_{rh}) is the riser mean pressure minus the column above the upper half of the riser. The riser mean pressure can be computed with the ideal gas equation. Finally, the manifold pressure (p_m) can be obtained by adding the pressure from the fluid column and the pressure due to friction, which is obtained by the Darcy–Weisbach equation. The coefficient factor assumes a laminar flow, but a turbulent factor is a more realistic assumption.

$$p_{rh} = p_{rm} - g\frac{L_r}{2}\rho_{rm} \tag{334}$$

$$p_{rm} = RT_r\frac{\rho_{gm}}{M_g} \tag{335}$$

$$p_m = p_{rh} + gL_r\rho_{rm} + \frac{128}{\pi}\frac{\mu_{oil}L_r}{D_r^4}\frac{w_{pr}}{\rho_{rm}} \tag{336}$$

where:

- ρ_{rm} is the riser mean fluid density,
- ρ_{gm} is the riser mean gas density,
- D_r is the riser diameter,
- T_r is the riser temperature,
- L_r is the riser length,
- M_g is the gas molecular weight,
- μ_{oil} is the oil dynamic viscosity.

The gas mean density is obtained by dividing the gas mass by the riser volume discounted the oil volume,

$$\rho_{gm} = \frac{m_{gr}}{V_r - \frac{m_{or}}{\rho_o}} \tag{337}$$

$$\rho_{rm} = \frac{m_{gr} + m_{or}}{V_r} \tag{338}$$

where V_r is the riser volume.

4.3 SCENARIO DESCRIPTION

Control objectives and operation constraints are case dependent; they depend on operating conditions, an alignment of business decisions, and technical limitations. Often the operational situation varies day by day, for instance, when equipment breaks or goes into maintenance. It can also vary monthly, when the production field business team defines a new production target. Alternatively, even yearly, for instance, as new

regulations are defined. Changes on longer timescales may also occur, whether from changing regulations or from the gradual depletion of the reservoir. Depending on the solution architecture, some of this decisions can be delegated to static optimization (AGUIAR et al., 2012; KOSMIDIS et al., 2004, 2005; GUNNERUD; FOSS, 2010; SILVA et al., 2012; CODAS; CAMPONOGARA, 2012; SILVA; CAMPONOGARA, 2014).

Given the uncertainty of the conditions, in the experiments, a simple control objective is chosen. The ultimate goal is to assess how the algorithms developed in Chapter 3 behave on such a network, not to mimic real-life operations.

The objective of the wells is to minimize the variation of the injected gas to reduce the wear of valves and induce a smooth operating condition.

$$J_w = \int_{t_0}^{t_f} K_{\dot{w}_{gl}} \dot{w}_{gl}^2 \, dt \tag{339}$$

where:

- $K_{\dot{w}_{gl}}$ is an adjustabale penalty parameter;
- $\dot{w}_{gl}$ is the variation in the gas injection.

The objective of the riser is to minimize the gas burned in the flare while trying to keep the total oil outflow at a given target.

$$J_w = \int_{t_0}^{t_f} (\bar{w}_{to} - w_{to})^2 + K_{\dot{w}_{fl}} \dot{w}_{fl}^2 \, dt \tag{340}$$

where:

- w_{fl} is the amount of gas burned on the flare;
- $K_{\dot{w}_{fl}}$ is an adjustabale penalty parameter;
- $\bar{w}_{to}$ is the reference for the oil outflow.

Most of the implemented constraints merely keep the models sound with reality; for instance, flows should not go backward, and masses should not be negative. Other than that, each well optimal control problem implements a gas injection variation range and a minimum gas injection. Also, the riser implements an upper bound on the gas outflow, which could be interpreted as a limit on the separator gas handling capability.

To allow any residual dynamic to be dissipated and not influence the results, the MPC will only be activated after 250 seconds of simulation.

4.4 EXPERIMENTAL SETUP

The experiment studies the control of the previously described two wells and riser plant for a time horizon of 1000 seconds, bringing the system to a production

setpoint. The algorithms are implemented with an MPC control strategy, where at each time step, the control is computed based on the available measurements. For simplicity, the states are assumed to be measurable, or, equivalently, we have a perfect state estimator. The MPC uses a prediction window of 300 seconds, broken down into 60 intervals. The experiments used the direct collocation method to obtain the solution of the OCPs. The collocation method uses a polynomial of order 3.

If the OCP solver fails to obtain a solution (e.g., maximum iteration reached, IPOPT failed to recover), the iteration is disregarded, and the previous solution is used in the update steps. This represents the node communicating to its neighbors that it cannot move the shared variables in the direction that they "proposed".

The experiments are performed with 10 different scenarios varying the system oil production target, ranging from 39 to 44 kg/s. For each of the 10 scenarios the proposed algorithms are compared to a centralized approach, which also uses direct collection.

The algorithms, OCPs, and subsystems were implemented in Python using YAOCPTool, which relies on CasADi (ANDERSSON et al., 2019). IPOPT (WÄCHTER; BIEGLER, L. T., 2006) was used to solve the optimization problems. The experiments were performed in a computer equipped with an Intel 7700k (4 cores/8 threads) processing unit and 16 GB DDR4 of RAM, running Ubuntu 20.04 in WSL. Different from the experiments in the previous chapter, each node was solved sequentially using the same thread, rather than each in a separated thead. This was due to an issue with the implementation of the multi-threaded solution methods. It is not related to any limitation of the proposed algorithms or the modeled system. This most likely affects negatively the performance of the proposed algorithms.

In order to compare the algorithms, we need to define some metrics. To verify the numerical stability of the algorithms, the first metric is the convergence of scenarios. A scenario is said to be successful if it takes less than 3600 seconds to optimize for the whole simulation.

Regarding the quality of the solution, the algorithms will be evaluated concerning the realized cost. The realized cost uses the same function as the objective function in the OCP. The difference is that the realized cost comes from accumulating the incurred cost in the actual plant, accumulated over the entire duration of the experiment.

The performance is measured in two ways: the total time to perform the simulation, and the average time per MPC iteration, discarding the first MPC iteration. The first iteration is discarded because it is part of the algorithm initialization and does not reflect precisely the algorithm behavior during operation.

To summarize, the algorithms and variations will be analyzed with respect to the following criteria:

- Convergence (successful scenarios / total scenarios);

- Quality: Realized cost;
- Performance:
  - Total time;
  - Average MPC iterartion after the first iteration.

The strategies that will be compared are the best performing strategies from the previous chapter. Namely,

- Centralized,
- Augmented Lagrangian with Coordinate Descent, using a cyclic block choice rule and constant penalty parameter update rule;
- Alternating Direction Multiplier Method, using a primal-dual penalty parameter update rule;
- Bipartite-Jacobi ADMM, with a primal-dual μ parameter update rule.

4.5 ANALYSIS OF EXPERIMENTAL RESULTS

The numerical experiments were performed for all four strategies. The following figures show the behavior of the system in Scenario 1 for each of the methods tested. Similar results are obtained also in the other scenarios. The results are summarized in Table 10.

The profile of the produced oil is very similar between the different algorithms, with only slight difference. Figures 33, 34, 35, and 36 show the profile of the produced oil in the first scenario for the centralized, augmented Lagrangian with coordinate descent, ADMM, and the Bipartite-Jacobi ADMM, respectively.

Figures 37, 38, 39, and 40 display the subsystem's states for the first scenario with the different strategies: the centralized, augmented Lagrangian with coordinate descent, ADMM, and the Bipartite-Jacobi ADMM, respectively. The strategies display similar behavior for the states, just like the other compared variables.

Regarding the control actions, they all have the same behavior differing only due to numerical noise. Figures 41, 42, 43, and 44 show the control actions along the first scenario for the centralized, augmented Lagrangian with coordinate descent, ADMM, and the Bipartite-Jacobi ADMM, respectively.

For better visualization of the metrics, the results of each experiment were consolidated by averaging all the scenarios. The results of the consolidation are reported in Table 10.

All the tested algorithms were able to conclude the scenarios before the timeout, so they were considered all to be successful. Regarding the realized cost, both the

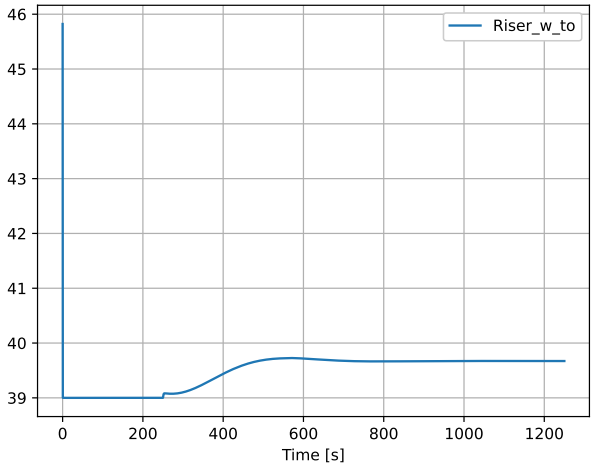


Figure 33 – Oil production profile for the centralized approach on Scenario 1.

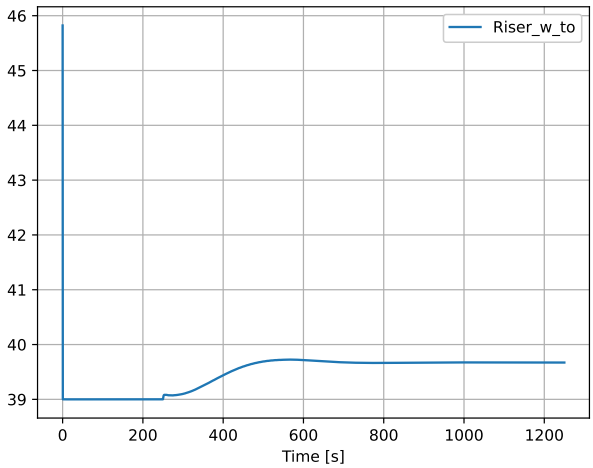


Figure 34 – Oil production profile for the augmented Lagrangian with coordinate descent approach on Scenario 1.

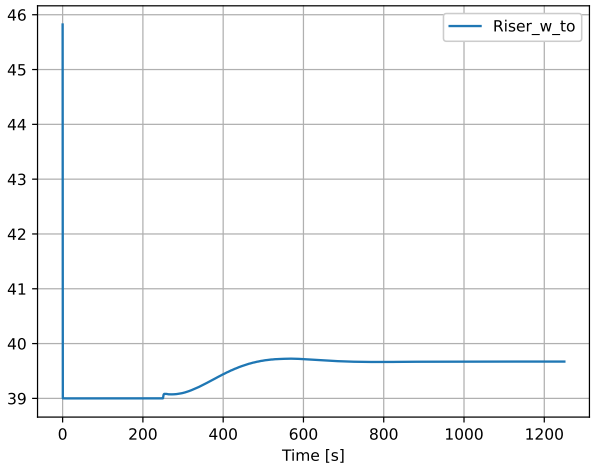


Figure 35 – Oil production profile for the ADMM approach on Scenario 1.

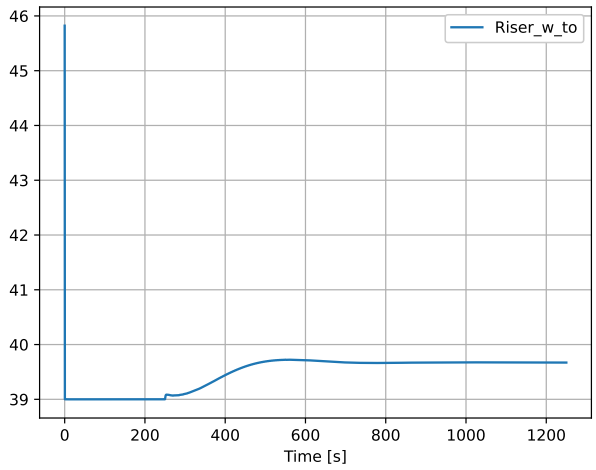


Figure 36 – Oil production profile for the Bipartite-Jacobi ADMM approach on Scenario 1.

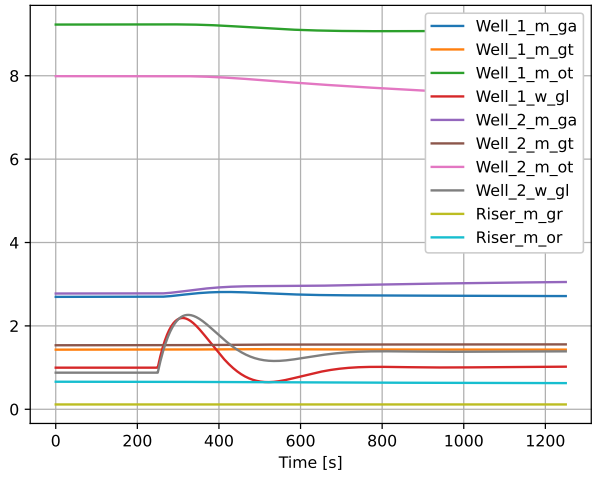


Figure 37 – State profile for the centralized approach on Scenario 1.

centralized approach and the ADMM were able to induce a better operation of the plant overall. The ADMM had the better result by a tiny margin. Concerning performance, the total computation time was longer with the distributed algorithms compared to the centralized approach. However, the ADMM had a very comparable time, despite the additional cost of initialization and communication. The average time per iteration, which does not consider the first iteration tells us another story, the ADMM ends up outperforming the centralized strategy ever so slightly. The augmented Lagrangian with the coordinate descent and the Bipartite-Jacobi ADMM had worse performance, both in terms of computation time and realized cost with the parameters tested in these experiments.

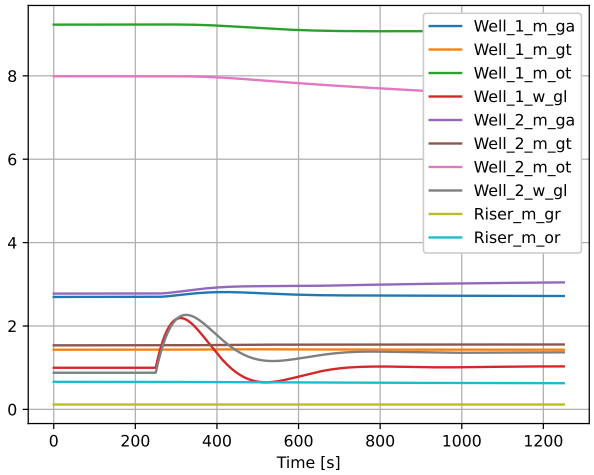


Figure 38 – State profile for the augmented Lagrangian with coordinate descent approach on Scenario 1.

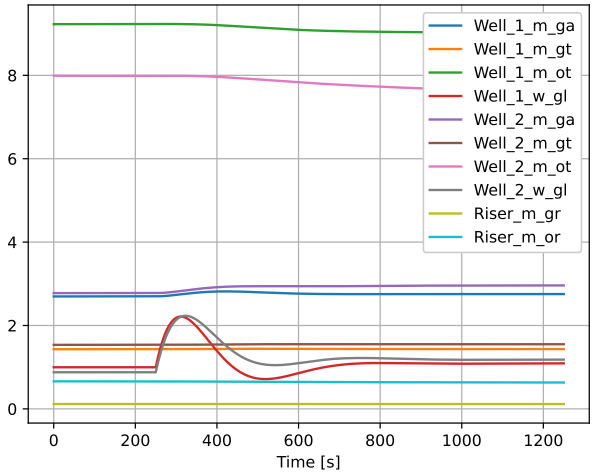


Figure 39 – State profile for the ADMM approach on Scenario 1.

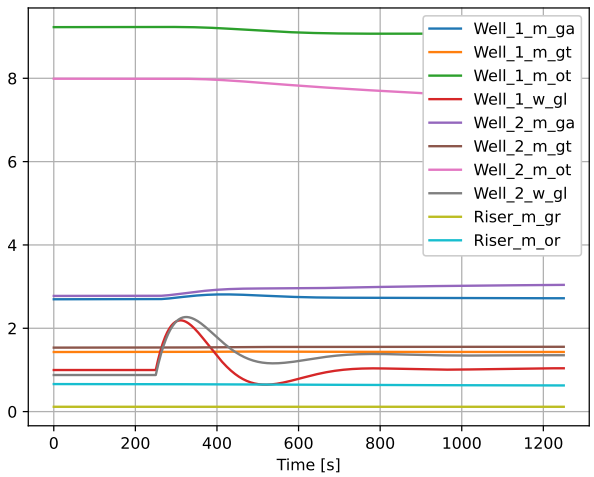


Figure 40 – State profile for the Bipartite-Jacobi ADMM approach on Scenario 1.

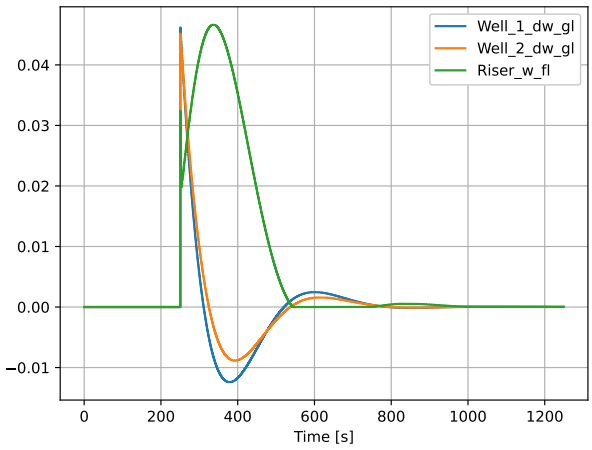


Figure 41 – Control profile for the centralized approach on Scenario 1.

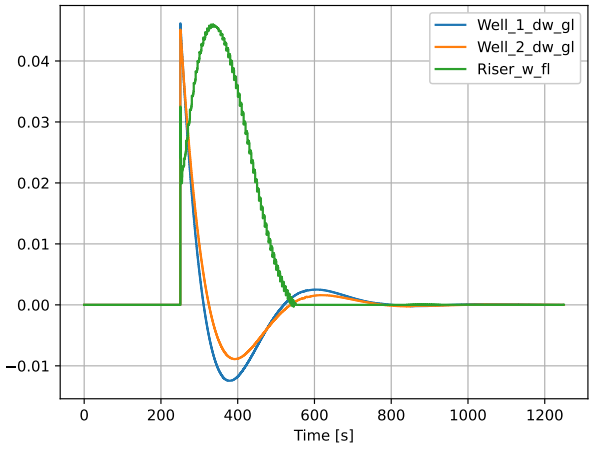


Figure 42 – Control profile for the augmented Lagrangian with coordinate descent approach on Scenario 1.

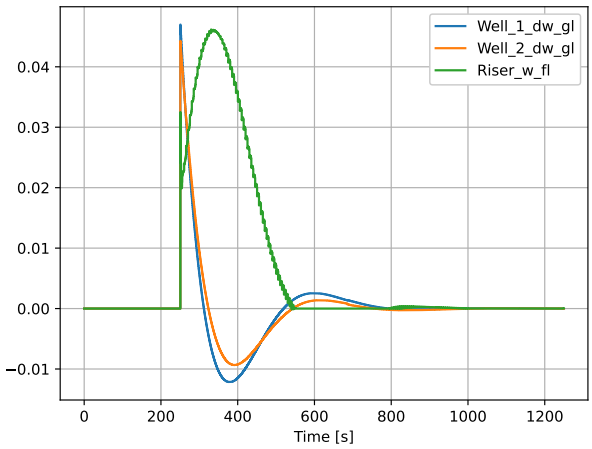


Figure 43 – Control profile for the ADMM approach on Scenario 1.

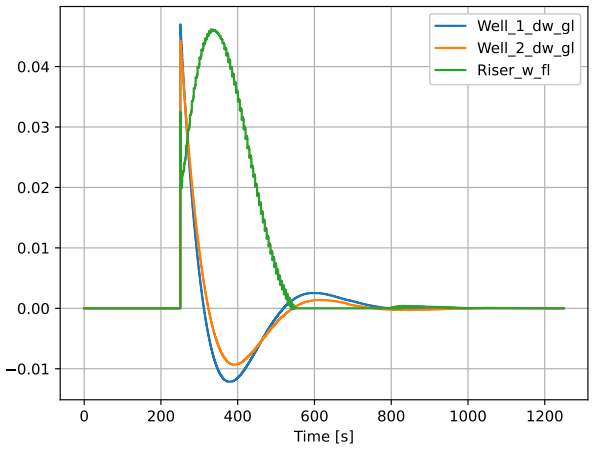


Figure 44 – Control profile for the Bipartite-Jacobi ADMM approach on Scenario 1.

Table 10 – Experiment Result

	Centralized	AL-CD	ADMM	Bipartite-Jacobi
Convergence (total 10)	10	10	10	10
Realized Cost	913.35	919.79	913.07	918.80
Total time (s)	100.16	255.86	109.27	268.70
Avg. Iteration (w.o. first)	0.485	1.20	0.450	1.18

4.6 CONCLUSION

In the previous chapters, we focused on evaluating the proposed algorithms with benchmark systems that are compact and well-behaved in all operating regions. In this chapter, the algorithms were challenged with a system that is much closer to those found in real control applications.

A simplified oil and gas production network was modeled by connecting well and riser models from the literature. The network comprises two oil and gas production wells and a vertical riser. The plant of choice here is a small part of a broader system. A more complex network with a greater diversity of equipment should be investigated for future work.

The results show exciting points. Even though the distributed methods solve the problems iteratively, they provide an equivalent result in a comparable time, particularly in the ADMM case. This can be justified by the nonlinearities embedded within each node problem and the relation between the nodes being linear. Each node is solving a smaller problem which has an iteration cost smaller than the original problem. The iteration cost comes from the number of variables and constraints, which directly affects the size of the Jacobian and the Hessian of the problem.

The experimental results clearly indicate that the proposed strategies are viable alternatives for controlling complex nonlinear systems, which encourages further investigation with the proposed methods, being them of practical or theoretical nature.

5 CONCLUSION

This thesis presents some advances in the area of optimal control and distributed optimal control.

The second chapter provides some proofs for an augmented Lagrangian method to solve the optimal control problem of DAE systems. The method that converts the solution of OCP of DAEs into a sequence of solutions to optimal control problems of ODEs. The theorems show that under certain circumstances, the algorithm can converge to a global minimum if its underlying subproblems are solved to global optimality; if the subproblems are solved to a local optimum, then the algorithm converges to a local optimum; and if at each iteration the subproblems are iterated to a point that is "close enough" to an optimum, and this distance decreases with the iterations, then the algorithm converges to the optimal solution. The demonstrations also show that the relaxed multiplier converges to a multiplier of the original problem.

Having good mathematical properties is not enough to make a valuable algorithm in practice[1]. To address this concern, the algorithm was put to the test with two classical benchmark control systems: the Van der Pol oscillator and the Quadruple Tank. The first has a limit cycle that makes it challenging to stabilize, and the second has heavy coupling between the many components of the system, which makes the dynamics intertwined. Both experiments showed promising results, which inspired further investigation. One highlight here is that the algorithm allows for an indirect solution method to deal with the algebraic and state variable constraints with great ease.

Part of the challenge of applying control of large plants is creating and maintaining control models and controllers. Given the system and organization's topologies, this task might include different teams that need to collaborate upon achieving a single control model, which is a difficult task and often leads to companies opting for decentralized control strategies. Within this context, a modeling framework for networked systems is proposed using differential-algebraic equations (DAEs). The framework models each subsystem as a node in a directional graph, the edges of the graph represent an input-output relation between the two subsystems. Mathematically, the edges are equivalent to a trivial linear algebraic equation.

Using the augmented Lagrangian method for optimal control to relax the algebraic equation of the edges, a set of distributed optimal control algorithms were proposed, inspired by optimization algorithms. The algorithms are the augmented Lagrangian method with coordinate descent and the alternating direction multiplier methods (ADMM). It was shown that by exploring the network structure and introducing intermediate nodes, it is possible to fully decouple the many subsystems and achieve parallelism in solving the underlying optimal control problems. Using some manipula-

[1] For instance, the ellipsoid method to solve linear programming problems has polynomial time but has limited usage due to numerical instability (DANTZIG; THAPA, 2006).

tions on the necessary conditions of the intermediate nodes, we were able to obtain an analytical solution for the subproblem. Further manipulations allowed for merging the solution of intermediate nodes back into the solution of the original nodes. These manipulations led to a new algorithm called Bipartite-Jacobi ADMM since it is a simultaneous iterative algorithm (like the Jacobi method). Some experiments with the Four-tanks benchmark system were made to validate the algorithms and their variations. The results were quite positive towards the proposed algorithms, where they have shown to solve a significant part of the scenarios with a solid performance.

Up to that point, the algorithms have faced benchmark systems, which are challenging but lack some of the complications that emerge in real-life systems. In the fourth chapter, the proposed distributed optimal control algorithms solve a problem with a plant composed of two oil and gas production wells and a vertical riser. The algorithms are used in a receding horizon MPC, where their solutions obtain a new control action at each iteration. The algorithms have been shown to solve the problem with results comparable to the centralized baseline.

These advances inspire further investigations in the area and could be part of further research:

- Use the augmented Lagrangian to solve stochastic optimal control problems – the algorithms in this thesis were used for decoupling subsystems in a network; these same algorithms could be used in the context of stochastic optimal control, where instead of dealing with the coupling between the subsystems, the relaxed equation would be the coupling between the multiple scenarios.

- Validation with a more complex network – as mentioned in the previous chapter, there are numerous ways to make the problem more complex, for instance, by including separators, compressors, gas injection lines, and gas exportation lines. An investigation in this direction would be pretty opportune for these algorithms.

- Theoretical developments on the proposed algorithms – now that they have been shown to provide sound practical results, it would be interesting to further develop mathematical properties to support their usage.

REFERENCES

AGUIAR, Marco Aurelio. **An Augmented Lagrangian Method for Optimal Control of Continuous Time Dae Systems**. 2016. MA thesis – Federal University of Santa Catarina.

AGUIAR, Marco Aurelio; CAMPONOGARA, Eduardo; FOSS, Bjarne. An Augmented Lagrangian for Optimal Control of DAE Systems: Algorithm and Properties. IEEE Transactions on Automatic Control, v. 66, p. 261–266, 1 Jan. 2021. DOI: 10.1109/TAC.2020.2976042.

AGUIAR, Marco Aurelio; CAMPONOGARA, Eduardo; FOSS, Bjarne. An augmented Lagrangian method for optimal control of continuous time DAE systems. In: 2016 IEEE Conference on Control Applications (CCA). 2016. P. 1185–1190. DOI: 10.1109/CCA.2016.7587967.

AGUIAR, Marco Aurelio; SILVA, Thiago; CAMPONOGARA, Eduardo. A Mixed-Integer Convex Formulation for Optimal Operation of Gas-Lifted Oil Fields with Facility, Routing, and Pressure Constraints. In: July. ENGOPT 2012 - International Conference on Engineering Optimization. Rio de Janeiro - Brazil, 2012. P. 1–10.

AGUIAR, Marco Aurélio; CODAS, Andres; CAMPONOGARA, Eduardo. Systemwide Optimal Control of Offshore Oil Production Networks with Time Dependent Constraints. **IFAC-PapersOnLine**, v. 48, n. 6, p. 200–207, 2015. 2nd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production OOGP 2015. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.08.032.

ALLEN, R. **Mathematical analysis for economists**. Macmillan, 1962.

ANDERSSON, Joel A E; GILLIS, Joris; HORN, Greg; RAWLINGS, James B; DIEHL, Moritz. CasADi – A software framework for nonlinear optimization and optimal control. **Mathematical Programming Computation**, Springer, v. 11, n. 1, p. 1–36, 2019. DOI: 10.1007/s12532-018-0139-4.

ASCHER, Uri M.; PETZOLD, Linda R. **Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations**. 1st. USA: Society for Industrial and Applied Mathematics, 1998. ISBN 0898714125.

BECK, Amir; TETRUASHVILI, Luba. On the Convergence of Block Coordinate Descent Type Methods. **SIAM Journal on Optimization**, v. 23, n. 4, p. 2037–2060, 2013. ISSN 1052-6234. DOI: 10.1137/120887679.

BERGER, M.S. **Nonlinearity and Functional Analysis: Lectures on Nonlinear Problems in Mathematical Analysis**. Elsevier Science, 1977. (Pure and Applied Mathematics). ISBN 9780080570440.

BERTSEKAS, Dimitri P. **Constrained Optimization and Lagrange Multiplier Methods**. Athena Scientific, 1982. ISBN 1-886529-04-3.

BERTSEKAS, Dimitri P. **Constrained Optimization and Lagrange Multiplier Methods**. Athena Scientific, 1996. (Athena scientific series in optimization and neural computation).

BERTSEKAS, Dimitri P. **Dynamic Programming and Optimal Control Vol I - Third Edition**. Athena Scientific, 2005. P. 543. ISBN 1-886529-26-4.

BERTSEKAS, Dimitri P. **Nonlinear Programming**. Athena Scientific, 1995.

BERTSEKAS, Dimitri P.; TSITSIKLIS, John. **Parallel and Distributed Computation: Numerical Methods**. 1989. P. 735. ISBN 978-1-886529-01-4.

BESTLER, Anja; GRAICHEN, Knut. Distributed model predictive control for continuous-time nonlinear systems based on suboptimal ADMM, 2017.

BIEGLER, Lorenz T. **Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Process**. Society for Industrial and Applied Mathematics, 2010.

BOCCIA, A.; PINHO, M. D. R. de; VINTER, R. B. Optimal Control Problems with Mixed and Pure State Constraints. **SIAM Journal on Control and Optimization**, v. 54, n. 6, p. 3061–3083, 2016. ISSN 0363-0129. DOI: 10.1137/15M1041845.

BONDY, Adrian; MURTY, U. S. R. **Graph Theory**. 2008. P. 655. ISBN 978-1-84628-969-9.

BOYD, Stephen; PARIKH, Neal; CHU, Eric; PELEATO, Borja; ECKSTEIN, Jonathan. Distributed optimization and statistical learning via the alternating direction method of multipliers. **Foundations and Trends in Machine Learning**, v. 3, n. 1, p. 1–122, 2010. ISSN 19358237. DOI: 10.1561/2200000016.

BRADLEY, Joseph K.; KYROLA, Aapo; BICKSON, Danny; GUESTRIN, Carlos. Parallel Coordinate Descent for L1-Regularized Loss Minimization. **Proceedings of the 28th International Conference on Machine Learning, ICML 2011**, n. 1998, p. 321–328, 2011. arXiv: 1105.5379.

BUDINIS, S.; THORNHILL, N.F. Control of centrifugal compressors via model predictive control for enhanced oil recovery applications. **IFAC-PapersOnLine**, Elsevier Ltd., v. 48, n. 6, p. 9–14, 2015. ISSN 24058963. DOI: 10.1016/j.ifacol.2015.08.002.

CAMPONOGARA, Eduardo; JIA, Dong; KROGH, Bruce H; TALUKDAR, Sarosh. Distributed model predictive control. **IEEE Control Systems**, IEEE, v. 22, n. 1, p. 44–52, 2002.

CAMPONOGARA, Eduardo; OLIVEIRA, Mateus Dubiela; DE AGUIAR, Marco Aurelio Schmitz. Scheduling pumpoff operations in onshore oilfields under electric-power constraints. **European Journal of Operational Research**, 2015. ISSN 03772217. DOI: 10.1016/j.ejor.2015.06.001.

CAMPONOGARA, Eduardo; SANTOS DA SILVA, Ricardo; AGUIAR, Marco Aurélio Schmitz de. A distributed dual algorithm for distributed MPC with application to urban traffic control. In: 2017 IEEE Conference on Control Technology and Applications (CCTA). 2017a. P. 1704–1709. DOI: 10.1109/CCTA.2017.8062702.

CAMPONOGARA, Eduardo; SILVA, Ricardo da; AGUIAR, Marco Aurelio. A distributed dual algorithm for distributed MPC with application to urban traffic control. In: 2017 IEEE Conference on Control Technology and Applications (CCTA). IEEE, Aug. 2017b. P. 1704–1709. DOI: 10.1109/CCTA.2017.8062702.

CHEN, Caihua; HE, Bingsheng; YE, Yinyu; YUAN, Xiaoming. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. **Mathematical Programming**, v. 155, n. 1-2, p. 57–79, 2016. ISSN 0025-5610. DOI: 10.1007/s10107-014-0826-5.

CHEN, Liang; LI, Xudong; SUN, Defeng; TOH, Kim Chuan. **On the equivalence of inexact proximal ALM and ADMM for a class of convex composite programming**. Springer Berlin Heidelberg, 2019. ISBN 1460002571. DOI: 10.1007/s10107-019-01423-x.

CHRISTOFIDES, Panagiotis D.; SCATTOLINI, Riccardo; MUÑOZ DE LA PEÑA, David; LIU, Jinfeng. Distributed model predictive control: A tutorial review and future research directions. **Computers and Chemical Engineering**, v. 51, p. 21–41, 2013.

CLARKE, Francis; PINHO, M. R. de. Optimal Control Problems with Mixed Constraints. **SIAM Journal on Control and Optimization**, v. 48, n. 7, p. 4500–4524, 2010. ISSN 0363-0129. DOI: 10.1137/090757642.

CODAS, Andres; CAMPONOGARA, Eduardo. Mixed-integer linear optimization for optimal lift-gas allocation with well-separator routing. **European Journal of Operational Research**, v. 217, n. 1, p. 222–231, 2012. ISSN 03772217. DOI: 10.1016/j.ejor.2011.08.027.

DAI, Li; CAO, Qun; XIA, Yuanqing; GAO, Yulong. Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance. **Journal of the Franklin Institute**, Elsevier Ltd, v. 354, n. 4, p. 2068–2085, 2017.

DANTZIG, G.B.; THAPA, M.N. **Linear Programming 1 Introduction**. Springer New York, 2006. (Springer Series in Operations Research and Financial Engineering). ISBN 9780387226330.

DE OLIVEIRA, Lucas Barcelos; CAMPONOGARA, Eduardo. Multi-agent model predictive control of signaling split in urban traffic networks. **Transportation Research Part C: Emerging Technologies**, v. 18, n. 1, p. 120–139, 2010. Information/Communication Technologies and Travel Behaviour Agents in Traffic and Transportation. ISSN 0968-090X. DOI: https://doi.org/10.1016/j.trc.2009.04.022.

DE PINHO, M. D.R.; VINTER, R. B.; ZHENG, H. A maximum principle for optimal control problems with mixed constraints. **IMA Journal of Mathematical Control and Information**, v. 18, n. 2, p. 189–205, 2001. ISSN 02650754. DOI: 10.1093/imamci/18.2.189.

DENG, Wei; LAI, Ming-Jun; PENG, Zhimin; YIN, Wotao. Parallel Multi-Block ADMM with o(1 / k) Convergence. **Journal of Scientific Computing**, Springer US, v. 71, n. 2, p. 712–736, 2017. ISSN 0885-7474. DOI: 10.1007/s10915-016-0318-2. arXiv: 1312.3040.

DI MEGLIO, Florent; KAASA, Glenn-Ole; PETIT, Nicolas. A first principle model for multiphase slugging flow in vertical risers. In: PROCEEDINGS of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference. IEEE, 2009. P. 8244–8251. DOI: 10.1109/CDC.2009.5400680.

DIEHL, Fabio C. et al. Fast Offshore Wells Model (FOWM): A practical dynamic model for multiphase oil production systems in deepwater and ultra-deepwater scenarios. **Computers and Chemical Engineering**, Elsevier Ltd, v. 99, p. 304–313, 2017. ISSN 00981354. DOI: 10.1016/j.compchemeng.2017.01.036.

DUBHASHI, Devdatt. Distributed Vertex Coloring. In: **Encyclopedia of Algorithms**. Ed. by Ming-Yang Kao. Boston, MA: Springer US, 2008. P. 258–260. ISBN 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_118.

DUNBAR, William B; MURRAY, Richard M. Distributed receding horizon control for multi-vehicle formation stabilization. **Automatica**, v. 42, n. 4, p. 549–558, 2006.

ECKSTEIN, Jonathan; FERRIS, Michael C. Operator-Splitting Methods for Monotone Affine Variational Inequalities, with a Parallel Application to Optimal Control. **INFORMS Journal on Computing**, v. 10, n. 2, p. 218–235, 1998. ISSN 1091-9856. DOI: 10.1287/ijoc.10.2.218.

ECKSTEIN, Jonathan; YAO, Wang. Understanding the Convergence of the Alternating Direction Method of Multipliers: Theoretical and Computational Perspectives *. **Pacific Journal of Optimization**, 2015.

EIKREM, Gisle; AAMO, Ole; FOSS, Bjarne. On Instability in Gas Lift Wells and Schemes for Stabilization by Automatic Control. **SPE Production & Operations**, May, p. 268–279, 2008.

EIKREM, Gisle; IMSLAND, Lars; FOSS, Bjarne. Stabilization of gas-lifted wells based on state estimation. In: IFAC Symposium Adchem. 2004. P. 1–6.

ERNZERHOF, Matthias. Taylor-series expansion of density functionals. **Physical Review A**, v. 50, n. 6, p. 4593–4607, 1994. DOI: 10.1103/PhysRevA.50.4593.

FARINA, Marcello; BETTI, Giulio; SCATTOLINI, Riccardo. Distributed predictive control of continuous-time systems. **Systems & Control Letters**, Elsevier B.V., v. 74, p. 32–40, 2014.

FERCOQ, Olivier; QU, Zheng; RICHTARIK, Peter; TAKAC, Martin. Fast distributed coordinate descent for non-strongly convex losses. In: 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2014. P. 1–6. DOI: 10.1109/MLSP.2014.6958862.

FERRAMOSCA, A.; LIMON, D.; ALVARADO, I.; CAMACHO, E.F. Cooperative distributed MPC for tracking. **Automatica**, v. 49, n. 4, p. 906–914, 2013.

FORTIN, M.; GLOWINSKI, R. **Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems**. 2000. ISBN 008087536X.

FU, Wenjiang J. Penalized Regressions: The Bridge versus the Lasso. **Journal of Computational and Graphical Statistics**, v. 7, n. 3, p. 397–416, 1998. ISSN 1061-8600. DOI: 10.1080/10618600.1998.10474784.

FUKUSHIMA, Masao. Application of the alternating direction method of multipliers to separable convex programming problems. **Computational Optimization and Applications**, v. 1, n. 1, p. 93–111, 1992. ISSN 0926-6003. DOI: 10.1007/BF00247655.

GALEWSKA, E.; NOWAKOWSKI, A. Sufficient conditions for optimal control problems with mixed constraints. **Optimal Control Applications and Methods**, v. 26, n. 5, p. 255–264, 2005. ISSN 01432087. DOI: 10.1002/oca.762.

GERDTS, M. Local minimum principle for optimal control problems subject to differential-algebraic equations of index two. **Journal of Optimization Theory and Applications**, v. 130, n. 3, p. 441–460, 2006. ISSN 00223239. DOI: 10.1007/s10957-006-9121-9.

GHAFFARI, Mohsen; KUHN, Fabian. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. **arXiv**, 2020. ISSN 23318422. arXiv: 2011.04511.

GOLUB, Gene; VAN LOAN, Charles. **Matrix Computation**. John Hopkins University Press, 2013. ISBN 9781421407944.

GRONG, Torbjørn Sønstebø. **Modeling of Compressor Characterisics and Active Surge Control**. 2009. S. 1–97. PhD thesis – Norwegian University of Science and Technology.

GUAN, Zhi-Hong; WU, Yonghong; FENG, Gang. Consensus analysis based on impulsive systems in multiagent networks. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEE, v. 59, n. 1, p. 170–178, 2012.

GUNNERUD, Vidar; FOSS, Bjarne. Oil production optimization—A piecewise linear model, solved with two decomposition strategies. **Computers & Chemical Engineering**, Elsevier Ltd, v. 34, n. 11, p. 1803–1812, 2010. ISSN 00981354. DOI: 10.1016/j.compchemeng.2009.10.019.

HAN, Deren; YUAN, Xiaoming; ZHANG, Wenxing. An augmented Lagrangian based parallel splitting method for separable convex minimization with applications to image processing. **Mathematics of Computation**, v. 83, n. 289, p. 2263–2291, 2014. ISSN 0025-5718. DOI: 10.1090/S0025-5718-2014-02829-9.

HARTSFIELD, Nora; RINGEL, Gerhard. **Pearls in Graph Theory: A Comprehensive Introduction**. 2003. P. 293. ISBN 9783540773405.

HE, B. S.; YANG, H.; WANG, S. L. Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities. **Journal of Optimization Theory and Applications**, v. 106, n. 2, p. 337–356, 2000. ISSN 0022-3239. DOI: 10.1023/A:1004603514434.

HE, Bingsheng. Parallel splitting augmented Lagrangian methods for monotone structured variational inequalities. **Computational Optimization and Applications**, v. 42, n. 2, p. 195–212, 2009. ISSN 09266003. DOI: 10.1007/s10589-007-9109-x.

HE, Bingsheng; HOU, Liusheng; YUAN, Xiaoming. On Full Jacobian Decomposition of the Augmented Lagrangian Method for Separable Convex Programming. **SIAM Journal on Optimization**, v. 25, n. 4, p. 2274–2312, 2015. ISSN 1052-6234. DOI: 10.1137/130922793.

HE, Bingsheng; YUAN, Xiaoming. A class of ADMM-based algorithms for three-block separable convex programming. **Computational Optimization and Applications**, Springer US, v. 70, n. 3, p. 791–826, 2018. ISSN 15732894. DOI: 10.1007/s10589-018-9994-1.

HERNANDEZ, Bernardo; BALDIVIESO MONASTERIOS, P.R.; TRODDEN, Paul. Distributed MPC: Guaranteeing Global Stabilizability from Locally Designed Tubes. **IFAC-PapersOnLine**, v. 50, n. 1, p. 12335–12340, 2016.

HU, Bin; GOLAN, Michael. Gas-lift Instability Resulted Production Loss and Its Remedy by Feedback Control: Dynamical Simulation Results. In: SPE International Improved Oil Recovery Conference in Asia Pacific. Society of Petroleum Engineers, 2003. DOI: 10.2523/84917-MS.

JAGGI, Martin; SMITH, Virginia; TAKÁČ, Martin; TERHORST, Jonathan; KRISHNAN, Sanjay; HOFMANN, Thomas; JORDAN, Michael I. Communication-Efficient Distributed Dual Coordinate Ascent. **Advances in Neural Information Processing Systems**, v. 4, January, p. 3068–3076, 2014. ISSN 10495258. arXiv: 1409.1458.

JAHANSHAHI, Esmaeil; SKOGESTAD, Sigurd. Simplified Dynamic Models for Control of Riser Slugging in Offshore Oil Production. **Oil and Gas Facilities**, v. 3, n. 06, p. 080–088, 2014. ISSN 2224-4514. DOI: 10.2118/172998-pa.

JAHANSHAHI, Esmaeil; SKOGESTAD, Sigurd. Simplified Dynamical Models for Control of Severe Slugging in Multiphase Risers. In: 8TH IFAC World Congress. 2011. P. 1634–1639. DOI: 10.3182/20110828-6-IT-1002.00981.

JAHANSHAHI, Esmaeil; SKOGESTAD, Sigurd; HANSEN, Henrik. Control structure design for stabilizing unstable gas-lift oil wells. In: 8TH IFAC Advanced Control of Chemical Processes. 2012. P. 93–100.

JIANG, Z. K.; YUAN, X. M. New Parallel Descent-like Method for Solving a Class of Variational Inequalities. **Journal of Optimization Theory and Applications**, v. 145, n. 2, p. 311–323, 2010. ISSN 0022-3239. DOI: 10.1007/s10957-009-9619-z.

JOHANSSON, K.H. The quadruple-tank process: a multivariable laboratory process with an adjustable zero. **IEEE Transactions on Control Systems Technology**, v. 8, n. 3, p. 456–465, 2000. ISSN 10636536. DOI: 10.1109/87.845876.

JORDANOU, Jean; ANTONELO, Eric; CAMPONOGARA, Eduardo; AGUIAR, Marco Aurelio. Recurrent Neural Network Based Control of an Oil Well. In: XIII Simpósio Brasileiro de Automação Inteligente. 2017.

JORDANOU, Jean P.; CAMPONOGARA, Eduardo; ANTONELO, Eric; AGUIAR, Marco Aurelio. Nonlinear Model Predictive Control of an Oil Well with Echo State Networks. IFAC-PapersOnLine, v. 51, n. 8, p. 13–18, 2018. DOI: 10.1016/j.ifacol.2018.06.348.

KAMESWARAN, Shivakumar; BIEGLER, Lorenz. Convergence rates for direct transcription of optimal control problems using collocation at Radau points. **Computational Optimization and Applications**, v. 41, n. 1, p. 81–126, 2007. ISSN 0926-6003. DOI: 10.1007/s10589-007-9098-9.

KHALIL, H.K. **Nonlinear Systems**. Prentice Hall, 2002. (Pearson Education). ISBN 9780130673893.

KIRK, D.E. **Optimal Control Theory: An Introduction**. Dover Publications, 2004. (Dover Books on Electrical Engineering Series).

KONTOGIORGIS, Spyridon; MEYER, Robert R. A variable-penalty alternating directions method for convex optimization. **Mathematical Programming, Series B**, 1998. ISSN 00255610. DOI: 10.1007/BF02680549.

KOSMIDIS, Vassileios D.; PERKINS, John D.; PISTIKOPOULOS, Efstratios N. A mixed integer optimization formulation for the well scheduling problem on petroleum fields. **Computers & Chemical Engineering**, Elsevier, v. 29, n. 7, p. 1523–1541, 2005. ISSN 00981354. DOI: 10.1016/j.compchemeng.2004.12.003.

KOSMIDIS, Vassileios D.; PERKINS, John D.; PISTIKOPOULOS, Efstratios N. Optimization of Well Oil Rate Allocations in Petroleum Fields. **Industrial & Engineering Chemistry Research**, v. 43, n. 14, p. 3513–3527, 2004. ISSN 0888-5885. DOI: 10.1021/ie034171z.

KRISHNAMOORTHY, Dinesh; AGUIAR, Marco Aurelio; FOSS, Bjarne; SKOGESTAD, Sigurd. A Distributed Optimization Strategy for Large Scale Oil and Gas Production Systems. In: 2018 IEEE Conference on Control Technology and Applications (CCTA). IEEE, Aug. 2018. P. 521–526. DOI: 10.1109/CCTA.2018.8511385.

KRISHNAMOORTHY, Dinesh; BERGHEIM, Elvira M.; PAVLOV, Alexey; FREDRIKSEN, Morten; FJALESTAD, Kjetil. Modelling and Robustness Analysis of Model Predictive Control for Electrical Submersible Pump Lifted Heavy Oil Wells. **IFAC-PapersOnLine**, Elsevier B.V., v. 49, n. 7, p. 544–549, 2016. ISSN 24058963. DOI: 10.1016/j.ifacol.2016.07.399.

KUNKEL, Peter; MEHRMANN, Volker. Optimal control for unstructured nonlinear differential-algebraic equations of arbitrary index. **Mathematics of Control, Signals, and Systems**, v. 20, n. 3, p. 227–269, 2008. ISSN 09324194. DOI: 10.1007/s00498-008-0032-1.

LEE, Yin Tat; SIDFORD, Aaron. Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. IEEE, 2013. P. 147–156. DOI: 10.1109/FOCS.2013.24. eprint: 1305.1922.

LIN, Tian Yi; MA, Shi Qian; ZHANG, Shu Zhong. On the Sublinear Convergence Rate of Multi-block ADMM. **Journal of the Operations Research Society of China**, Operations Research Society of China, v. 3, n. 3, p. 251–274, 2015. ISSN 21946698. DOI: 10.1007/s40305-015-0092-0. arXiv: 1408.4265.

LIU, Ji; WRIGHT, Stephen J. Asynchronous Stochastic Coordinate Descent: Parallelism and Convergence Properties. **SIAM Journal on Optimization**, v. 25, n. 1, p. 351–376, 2015. ISSN 1052-6234. DOI: 10.1137/140961134. arXiv: 1403.3862.

LIU, Ji; WRIGHT, Stephen J.; RÉ, Christopher; BITTORF, Victor; SRIDHAR, Srikrishna. An asynchronous parallel stochastic coordinate descent algorithm. **Journal of Machine Learning Research**, v. 16, p. 285–322, 2015. ISSN 15337928. arXiv: 1311.1873.

MAREČEK, Jakub; RICHTÁRIK, Peter; TAKÁČ, Martin. Distributed Block Coordinate Descent for Minimizing Partially Separable Functions. In: SPRINGER Proceedings in Mathematics and Statistics. 2015. v. 134. P. 261–288. ISBN 9783319176888. DOI: 10.1007/978-3-319-17689-5_11. arXiv: 1406.0238.

MASELLA, J.M.; TRAN, Q.H.; FERRE, D.; PAUCHON, C. Transient simulation of two-phase flows in pipes. **International Journal of Multiphase Flow**, v. 24, p. 739–755, 1998.

MENDES, Paulo RC; MAESTRE, Jose M; BORDONS, Carlos; NORMEY-RICO, Julio E. A practical approach for hybrid distributed MPC. **Journal of Process Control**, Elsevier, v. 55, p. 30–41, 2017.

MURRAY, Richard M. Recent research in cooperative control of multivehicle systems. **Journal of Dynamic Systems, Measurement, and Control**, American Society of Mechanical Engineers, v. 129, n. 5, p. 571–583, 2007.

NOCEDAL, J.; WRIGHT, Stephen J. **Numerical Optimization**. Springer, 2006.

PLUCENIO, Agustinho; GANZAROLI, Cleber; PAGANO, Daniel J. Stabilizing gas-lift well dynamics with free operating point. In: 2010. 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production. 2012. P. 95–100.

PLUCENIO, Agustinho; PAGANO, Daniel; CAMPONOGARA, Eduardo; TRAPLE, A.; TEXEIRA, Alex. Gas-lift optimization and control with nonlinear MPC. In: 1. ADVANCED Control of Chemical Processes. 2009. P. 904–909. DOI: 10.3182/20090712-4-TR-2008.00148.

PONTRYAGIN, Lev Semenovich; BOLTYANSKII, V. G.; GAMKRELIDZE, R. V.; MISHCHENKO, E. F. **Mathematical Theory of Optimal Processes**. English Ed: INTERSCIENCE PUBLISHERS, 1962. P. 362.

RAWLINGS, James B.; STEWART, Brett T. Coordinating multiple optimization-based controllers: New opportunities and challenges. **Journal of Process Control**, v. 18, n. 9, p. 839–845, 2008.

RICHTÁRIK, Peter; TAKÁČ, Martin. Parallel coordinate descent methods for big data optimization. **Mathematical Programming**, v. 156, n. 1-2, p. 433–484, 2016. ISSN 0025-5610. DOI: 10.1007/s10107-015-0901-6. arXiv: 1212.0873.

ROCKAFELLAR, R. Tyrrell. MONOTONE OPERATORS AND THE PROXIMAL POINT ALGORITHM. **SIAM Journal on Control and Optimization**, 1976. ISSN 03630129. DOI: 10.1137/0314056.

SASANE, Amol. **Class Notes: Calculus of Variations and Optimal Control**. 2004. P. 63.

SAYDA, Atalla; TAYLOR, James. Modeling and Control of Three-Phase Gravilty Separators in Oil Production Facilities. In: AMERICAN Control Conference ACC '07. 2007.

SCATTOLINI, Riccardo. Architectures for distributed and hierarchical Model Predictive Control – A review. **Journal of Process Control**, Elsevier Ltd, v. 19, n. 5, p. 723–731, 2009.

SILVA, Thiago Lima; CAMPONOGARA, Eduardo. A computational analysis of multidimensional piecewise-linear models with applications to oil production optimization. **European Journal of Operational Research**, Elsevier B.V., v. 232, p. 630–642, 2014. ISSN 03772217. DOI: 10.1016/j.ejor.2013.07.040.

SILVA, Thiago Lima; CODAS, Andres; CAMPONOGARA, Eduardo. A Computational Analysis of Convex Combination Models for Multidimensional Piecewise-Linear Approximation in Oil Production Optimization. **2012 IFAC Workshop on Automatic**, p. 292–298, 2012.

SRINI, B; PALANKI, S; BON, D. Dynamic optimization of batch processes I . Characterization of the nominal solution. **Computers and Chemical Engineering**, v. 27, 2003. DOI: 10.1016/S0098-1354(02)00116-3.

STASIAK, ME; PAGANO, DJ; PLUCENIO, A. A New Discrete Slug-Flow Controller for Production Pipeline Risers. **IFAC Proceedings Volumes**, IFAC, v. 45, n. 8, p. 122–127, 2012. ISSN 14746670. DOI: 10.3182/20120531-2-NO-4020.00032.

STOER, J.; BULIRSCH, R. **Introduction to Numerical Analysis**. New York, NY: Springer New York, 2002. v. 142, p. xiii–xiv. ISBN 978-1-4419-3006-4. DOI: 10.1007/978-0-387-21738-3.

TAITEL, Y; BARNEA, D. Simplified transient simulation of two phase flow using quasi-equilibrium momentum balances. **International Journal of Multiphase Flow**, v. 23, n. 3, p. 493–501, 1997. ISSN 03019322. DOI: 10.1016/S0301-9322(96)00084-5.

TAITEL, Yehuda; SHOHAM, Ovadia; BRILL, JP. Simplified transient solution and simulation of two-phase flow in pipelines. **Chemical Engineering Science**, v. 44, n. 6, p. 1353–1359, 1989. ISSN 00092509. DOI: 10.1016/0009-2509(89)85008-0.

TOSSERAMS, Simon. **Distributed Optimization for Systems Design: An Augmented Lagrangian Coordination Method**. 2008. PhD thesis – Technische Universiteit Eindhoven. ISBN 9789038613505. DOI: 10.6100/IR636822.

VENKAT, Aswin N.; RAWLINGS, James B.; WRIGHT, Stephen J. Stability and optimality of distributed model predictive control. In: PROCEEDINGS of the 44th IEEE Conference on Decision and Control. IEEE, 2005. P. 6680–6685.

WÄCHTER, Andreas; BIEGLER, Lorenz T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. **Mathematical Programming**, v. 106, n. 1, p. 25–57, 2006. ISSN 0025-5610. DOI: 10.1007/s10107-004-0559-y.

WANG, S. L.; LIAO, L. Z. Decomposition Method with a Variable Parameter for a Class of Monotone Variational Inequality Problems. **Journal of Optimization Theory and Applications**, v. 109, n. 2, p. 415–429, 2001. ISSN 0022-3239. DOI: 10.1023/A:1017522623963.

WILLERSRUD, Anders; IMSLAND, Lars; HAUGER, Svein Olav; KITTILSEN, Pål. Short-term production optimization of offshore oil and gas production using nonlinear model predictive control. **Journal of Process Control**, Elsevier Ltd, 2012. ISSN 09591524. DOI: 10.1016/j.jprocont.2012.08.005.

WOHLBERG, Brendt. ADMM Penalty Parameter Selection by Residual Balancing. n. 18, p. 1–13, 2017. arXiv: 1704.06209.

WRIGHT, Stephen J. Coordinate descent algorithms. **Mathematical Programming**, v. 151, n. 1, p. 3–34, 2015. ISSN 14364646. DOI: 10.1007/s10107-015-0892-3. arXiv: 1502.04759.

XU, Meng; FADEL, Georges; WIECEK, Margaret M. Dual Residual in Augmented Lagrangian Coordination for Decomposition-Based Optimization. In: VOLUME 2B: 40th Design Automation Conference. American Society of Mechanical Engineers, 2014. P. 1–9. DOI: 10.1115/DETC2014-35103.

XU, Meng; FADEL, Georges; WIECEK, Margaret M. **Dual Residual for Centralized Augmented Lagrangian Coordination Based on Optimality Conditions**. 2015. v. 137. ISBN 0001002376. DOI: 10.1115/1.4029788.

YAO, Jing; GUAN, Zhi-Hong; HILL, David J. Passivity-based control and synchronization of general complex dynamical networks. **Automatica**, Elsevier, v. 45, n. 9, p. 2107–2113, 2009.

APPENDIX A – CALCULUS OF VARIATIONS

Just like functions map a number from a domain of numbers to a counter-domain of numbers, functionals map a function from a domain of functions into a number in a counter domain of numbers.

Given a function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x_1, x_2) = x_1 + x_2$, f is a function and $f(1, 2)$ is the real number 3. Likewise, a function $x : [t_0, t_f] \to X$ applied into a functional $I : X \to \mathbb{R}$ results in a real number, where $[t_0, t_f]$ is the interval that x is defined and X is a function space.

The concept of functionals may be better understood with some illustrative examples:

1. **Area below a curve x:** The area below a curve characterizes the classical notion of integral. For a function x, the functional for its area $I_a(x)$ is given by
$$I_a(x) = \int_{t_0}^{t_f} x(t)\, dt \tag{341}$$
2. **Length of a curve x:** Having as input the function x in the set of rectifiable curves (curves that can be approximated by an infinite number of tiny straight lines), the length of the function x in the interval $t \in [t_0, t_f]$ can be represented by the functional
$$I_\ell(x) = \int_{t_0}^{t_f} \sqrt{1 + \dot{x}^2}\, dt \tag{342}$$
3. **Quadratic error:** The quadratic error is a classical measure in several areas of mathematics, including control theory. For a function x, the functional that calculates the quadratic error with respect to a reference x_{ref} during a period $t \in [t_0, t_f]$ is given by
$$I_e(x) = \int_{t_0}^{t_f} (x_{ref} - x(t))^2\, dt \tag{343}$$
4. **Maximum value:** The maximum value attained to a function in the interval $[t_0, t_f]$.
$$I_{\max}(x) = \max_{t \in [t_0, t_f]} x(t) \tag{344}$$

Although functionals are not required to be integrals, in this work we are particularly interested in functionals formed by integrals, in particular with the form

Definition 4 (Functional)**.** *A functional $I : X \to \mathbb{R}$ is given by the equation*
$$I(x) = \int_{t_0}^{t_f} F(x, \dot{x}, t)\, dt \tag{345}$$
where X is a function space, with $x : [t_0, t_f] \to \mathbb{R}^{N_x}$, and the function $F : \mathbb{R}^{N_x} \times [t_0, t_f] \to \mathbb{R}$ which is referred as the Lagrangian function.

Remark. *Although there are other types of functionals, e.g. the functional of the derivative at the point* 0 *($I(x) = \dot{x}(0)$), the functionals of interest here are those defined by an integral.*

In particular, we are usually interested in the critical curves $x^* \in X$ that are solutions for the problem of minimizing (or maximizing) a functional I. Before the problem can be solved, the space in which the minimization of functionals take place and its properties must be specified.

A.1 FUNCTION SPACE

The concept of continuity is essential for functionals, because the usage of a functional is bound to the function space of its domain, for instance

$$I = \int_{t_0}^{t_f} F(x, \dot{x}, t)\, dt \tag{346}$$

only makes sense if x belong to the space of the continuously differentiable functions in the interval $t \in [t_0, t_f]$.

To define these spaces, as the space of continuously differentiable functions first the definitions for linear space, norm, distance, and normed linear space needed to be established.

A linear space over $\mathbb{R}$ is defined by a set X together with the operations of addition, $+ : X \times X \to X$, and scalar multiplication, $\cdot : X \times X \to X$, that satisfy the following properties:

1. Associativity: $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
2. Commutative of addition: $x_1 + x_2 = x_2 + x_1$.
3. Identity element of addition: There exists an element $0 \in X$, such that $x_1 + 0 = x_1$.
4. Inverse element: For every $x_1 \in X$ there is a $-x_1 \in X$, such that $(x_1) + (-x_1) = 0$.
5. Compatibility of multiplication: $a(bx_1) = (ab)x_1$.
6. Identity element of scalar multiplication: There exists and element 1, such that $1x_1 = x_1$.
7. Distributivity of scalar multiplication with respect to vector addition: $a(x_1 + x_2) = ax_1 + ax_2$.
8. Distributivity of scalar multiplication with respect to field addition: $(a + b)x_1 = ax_1 + bx_1$.

In addition, a *linear function* $L : X \to \mathbb{R}$ is a map that satisfies:

1. $L(x_1 + x_2) = L(x_1) + L(x_2)$ for all $x_1, x_2 \in X$.
2. $L(ax_1) = aL(x_1)$ for all $a \in \mathbb{R}$ and for all $x_1 \in X$.

A linear space is considered a normed space, if there exists a function $\|\cdot\| : X \to [0, \infty)$, named the norm, that has the following properties:

1. Zero norm: $\|x\| = 0$ if and only if $x = 0$.
2. Absolute scalability: $\|ax\| = |a| \, \|x\|$ for all $a \in \mathbb{R}$ and for all $x \in X$.
3. Triangle inequality: $\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$.

In a normed space it makes sense to establish distances between elements. We define a distance function $d : X \times X \to \mathbb{R}$, where the distance between an element x_1 and x_2 is given by $d(x_1, x_2) = \|x_1 - x_2\|$. A distance function has the following properties:

1. $d(x, y) \geq 0$ for all x and y in X.
2. $d(x, y) = 0$ if and only if $x = y$.
3. $d(x, y) = d(y, x)$ for all x and y in X.
4. $d(x, z) \leq d(x, y) + d(y, z)$ for all x, y, and z in X.

A linear space X equipped with a distance function d is called a metric space.

The elements of a normed linear space can be of any type, *e.g.* numbers, matrices, functions, *etc*. However, working with functionals the following normed spaces are more important:

1. $C[t_0, t_f]$ is the normed space formed by all the continuous functions x in the interval $[t_0, t_f]$. The addition operation is defined by a pointwise addition, meaning $x_1(t) + x_2(t) = (x_1 + x_2)(t)$ for all $t \in [t_0, t_f]$, and the scalar multiplication is defined by $ax(t) = (ax)(t)$ for all $t \in [t_0, t_f]$. The norm function is defined by

$$\|x\| = \max_{t \in [t_0, t_f]} \|x(t)\|_\infty \tag{347}$$

   Therefore, saying that the distance between $x_1(t)$ and $x_2(t)$ does not exceed ε means that if we plot these curves we would see that $x_2(t)$ lays inside a band with the borders $(x_1 + \varepsilon)(t)$ and $(x_1 - \varepsilon)(t)$, as shown in Figure 45.

2. $C^1[t_0, t_f]$ is the normed space that contains all the functions $x(\cdot)$ with continuous first derivative in the interval $t \in [t_0, t_f]$. The addition and multiplication operations are the same of the space $C[t_0, t_f]$, however the norm is defined by

$$\|x\| = \max_{t \in [t_0, t_f]} \|x(t)\|_\infty + \max_{t \in [t_0, t_f]} \|\dot{x}(t)\|_\infty \tag{348}$$

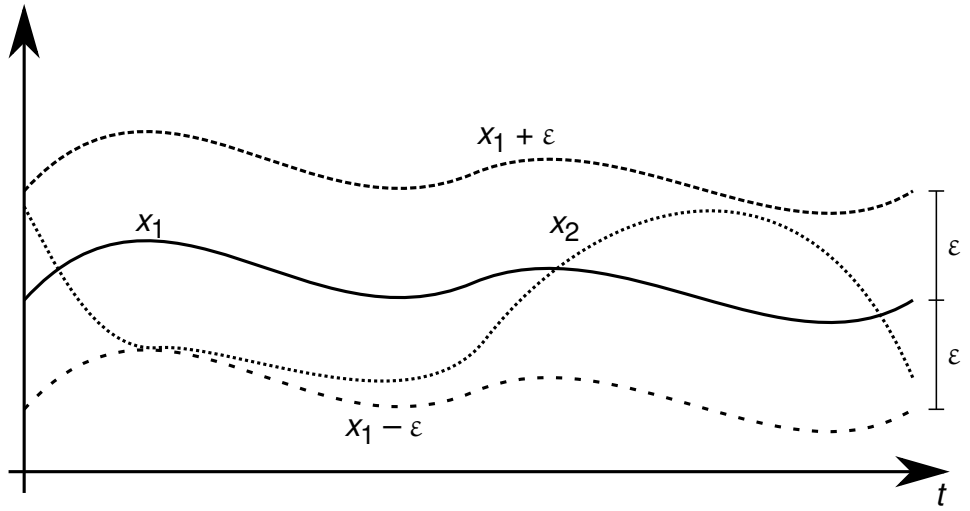


Figure 45 – The distance between functions x_1 and x_2 is ε

Thus, if $x_1(t)$ and $x_2(t)$ have a distance that does not exceed ε this means that

$$|x_1(t) - x_2(t)| < \varepsilon, \quad \forall t \in [t_0, t_f], \text{ and} \tag{349a}$$

$$|\dot{x}_1(t) - \dot{x}_2(t)| < \varepsilon, \quad \forall t \in [t_0, t_f] \tag{349b}$$

The function space presented so far contains the continuous and continuous differentiable functions with domain $[t_0, t_f]$ and codomain $\mathbb{R}$. However, when describing practical minimization problems we want functions from the interval $[t_0, t_f]$ to a particular space $\mathbb{R}^d$. Therefore we can define the space of functions $(\mathcal{C}[t_0, t_f])^d$, the space of continuous functions from $[t_0, t_f]$ to $\mathbb{R}^d$. The same definitions can be applied to define the space of $(\mathcal{C}^1[t_0, t_f])^d$.

Beside the function continuity, which is given by the chosen function space, the continuity of the functional itself must be considered. A functional $I : X \to \mathbb{R}$ is said to be continuous at $\bar{x}$ if for every $\varepsilon > 0$ if there exists a $\delta > 0$ such that

$$|I(x) - I(\bar{x})| < \varepsilon \text{ for all } x \in X \text{ such that } \|x - \bar{x}\| < \delta \tag{350}$$

where the norm used in the second inequality is the norm of the function space X. A functional $I : X \to \mathbb{R}$ is said to be continuous if it is continuous at all $x \in X$.

We define a local minimum of the functional I as the function x^* such that

$$I(x^*) \leq I(x), \text{ for all } x \text{ that satisfies } \|x^* - x\| < \varepsilon \tag{351}$$

for some $\varepsilon > 0$, where the norm is a norm of a function. Conversely, a local maximum of a functional I is a functional x^* such that

$$I(x^*) \geq I(x), \text{ for all } x \text{ that satisfies } \|x^* - x\| < \varepsilon \tag{352}$$

for some $\varepsilon > 0$. A function is said to be a local extremum, or merely an extremum, of the functional I if it is a local minimum or a local maximum. Moreover, the function x^* is

a global minimum for I if

$$I(x^*) \leq I(x), \text{ for all } x \in X \tag{353}$$

The concepts of global maximum and global extremum are analogous to their local counterparts.

A.2 DERIVATIVE OF FUNCTIONALS

In the previous subsection, we defined the grounds for the calculus of variations by introducing functionals, function spaces, and continuity in those contexts. However, oftentimes problems have candidate functions that are not in a linear space. For instance, consider a problem whose candidate functions, namely X_b, are the continuously differentiable functions that have the initial value at $x(0) = 0$ and the final value at $x(1) = 1$. It can be verified that X_b is not a linear space, given that for

$$x_3 = x_1 + x_2, \text{ where } x_1, x_2 \in X_b, \tag{354}$$

the function x_3 does not belong to the space X_b, since for $x_3(1) \neq 1$.

This often happens with problems of vectors and functions, and to cope with it we linearize the system around a region of interest. The same approach can be used for functionals. To this end, recall the definition of the derivative of functions, to later define the derivative of functionals.

The derivative of $f : \mathbb{R} \to \mathbb{R}$ at the point $\bar{z}$ is the approximation of f around $\bar{z}$ by an affine linear map $f'(\bar{z})$,

$$f(\bar{z} + h_z) = f(\bar{z}) + f'(\bar{z})h_z + \epsilon(h)|h_z| \tag{355}$$

where $h_z \in \mathbb{R}$ is a small scalar and the approximation error $\epsilon(h_z) \to 0$ as $h_z \to 0$.

In the same manner, for a functional I, the derivative at the *function* $\bar{x}$ is given by the linear map $I'(\bar{x})$, such that

$$I(\bar{x} + h) = I(\bar{x}) + \left[I'(\bar{x})\right](h) + \epsilon(h)\|h\| \tag{356}$$

where $h : [t_0, t_f] \to \mathbb{R}$ and the error functional $\epsilon(h) \to 0$ as $\|h\| \to 0$.

A linear map $L : A \to \mathbb{R}$ is always continuous only if A is a finite dimension space. If A is an infinite-dimensional space, then the continuity of the linear map must also be specified, because a linear map $L : X \to \mathbb{R}$ is a continuous linear functional only if it is linear and it is continuous.

The notion of functional derivatives can be formalized with the following definition.

Definition 5 (Fréchet Derivative)**.** *Let X be a normed linear space and $I : X \to \mathbb{R}$ be a functional. Then I is Fréchet differentiable at $\bar{x} \in X$ if there is a continuous linear map $A : X \to \mathbb{R}$ and a map $\epsilon : X \to \mathbb{R}$ such that, for all $h \in X$,*

$$I(\bar{x} + h) = I(\bar{x}) + A(h) + \epsilon(h)\|h\|, \tag{357}$$

and the error functional $\epsilon(h) \to 0$ as $\|h\| \to 0$. We write $A = I'(\bar{x})$, where $I'(\bar{x})$ is called the Fréchet derivative of I at $\bar{x}$. Since $I'(\bar{x})$ is a continuous linear operator we can write

$$A(h) = [I'(\bar{x})](h) = I'(\bar{x})h \tag{358}$$

The latter form is preferred for readability in large equations.

In addition, if I is Fréchet differentiable for every function $x \in X$, then the functional I is Fréchet differentiable.

The Fréchet derivative is unique for a given function x, as shown in the following theorem.

Theorem 9. *The Fréchet derivative of a Fréchet differentiable functional $I : X \to \mathbb{R}$ at the point $\bar{x} \in X$ is unique.*

Proof. Let $L : X \to \mathbb{R}$ be a linear functional, if

$$\frac{L(h)}{\|h\|} \to 0 \text{ as } \|h\| \to 0 \tag{359}$$

then $L(\cdot) = 0$. By contradiction, let $L(h_0) \neq 0$ for some nonzero $h_0 \in X$. Assuming $h_n = \frac{1}{n}h_0$, note that if $n \to \infty$ the norm $\|h\| \to 0$, however using the linearity of L and the homogeneity of the absolute, we have

$$\lim_{n\to\infty} \frac{L(h_n)}{\|h_n\|} = \lim_{n\to\infty} \frac{\frac{1}{n}L(h_0)}{|\frac{1}{n}|\|h_0\|} = \frac{L(h_0)}{\|h_0\|} \neq 0 \tag{360}$$

that contradicts the assumption (359), therefore the conclusion that $L(\cdot) = 0$ holds.

Let $L_1 : X \to \mathbb{R}$ and $L_2 : X \to \mathbb{R}$ be continuous linear functionals such that

$$I(\bar{x} + h) = I(\bar{x}) + L_1(h) + \epsilon_1(h)\|h\|, \text{ for all } h \in X \tag{361a}$$

$$I(\bar{x} + h) = I(\bar{x}) + L_2(h) + \epsilon_2(h)\|h\|, \text{ for all } h \in X \tag{361b}$$

with $\epsilon_1 \to 0$ and $\epsilon_2 \to 0$ as $\|h\| \to 0$. Subtracting both equations

$$\frac{(L_1 - L_2)}{\|h\|} = (\epsilon_1 - \epsilon_2)(h) \to 0 \text{ as } \|h\| \to 0 \tag{362}$$

therefore $L_1 = L_2$. □

If a function f has an extremum point (maximum or minimum) in z^*, its derivative at z^*, $\frac{df}{dz}(z^*)$ is 0. In the same way, If x^* is an extremum of the functional I, then its Fréchet derivative is $I'(x^*) = 0$, in the same manner that if a function f has a local extremum at the point z^*, then $\frac{df}{dz}(z^*) = 0$.

Theorem 10. *Let X be a normed linear space and $I : X \to \mathbb{R}$ be a Fréchet differentiable functional in $x^* \in X$. If I has a local extremum at x^*, then $I'(x^*) = 0$.*

Proof. By definition if $x^* \in X$ is a minimum of $I : X \to \mathbb{R}$, then there is a $r > 0$ such that $I(x^* + h) \geq I(x^*)$ for all h in a ball $\|h\| < r$.

To prove by contradiction, we suppose that $I'(x^*)h_0 \neq 0$ for some $h_0 \in X$. Assuming

$$h_n = -\frac{1}{n}\frac{|I'(x^*)h_0|}{I'(x^*)h_0}h_0 \tag{363}$$

as $n \to \infty$ we have $\|h_n\| \to 0$, and with a sufficiently large N we have $\|h_n\| < r$ for all $n > N$. By definition of derivative

$$\frac{I(x^* + h_n) - I(\bar{x})}{\|h_n\|} = \frac{I'(x^*)h_0}{\|h_n\|} + \epsilon(h_n) \tag{364}$$

we know that

$$\frac{I'(x^*)h_n}{\|h_n\|} = \frac{-\frac{1}{n}\frac{|I'(x^*)h_0|}{I'(x^*)h_0}I'(x^*)h_0}{\left|-\frac{1}{n}\frac{|I'(x^*)h_0|}{I'(x^*)h_0}\right| \|h_0\|} = -\frac{|I'(x^*)h_0|}{\|h_0\|} \tag{365}$$

For all $n > N$ we have

$$-\frac{|I'(x^*)h_0|}{\|h_0\|} + \epsilon(h_n) = \frac{I(x^* + h_n) - I(x^*)}{\|h_n\|} \geq 0 \tag{366}$$

taking the limit $n \to \infty$, we obtain that $-|\dot{I}_{x^*}(h_0)| \geq 0$ which contradicts the assumptions. Therefore we conclude that $I'(x^*) = 0$. □

The Fréchet derivative has good properties however its definition using limits is difficult to work. We need a way to obtain the derivative that relies on the already known and well-developed tools. For this matter, we can use a different notion of differentiability. The Gateaux derivative of a functional is similar to the directional derivative for functions. However, instead of taking the derivative in the direction of a vector, we take the derivative in the direction of a function $h \in X$.

Definition 6 (Gateaux Derivative or First Variation)**.** *The functional $I : X \to \mathbb{R}$ at the point $\bar{x} \in X$ has the Gateaux derivative (also known as first variation) in the direction $h \in X$ defined by*

$$\delta I(\bar{x}, h) \equiv \lim_{\xi \to 0} \frac{I(\bar{x} + \xi h) - I(\bar{x})}{\xi} = \frac{dI}{d\xi}(\bar{x} + \xi h)\bigg|_{\xi=0} \tag{367}$$

if the limit exists. If the Gateaux derivative exists for all directions $h \in X$, then I is Gateaux differentiable.

The notion of differentiability defined by Gateaux is weaker than the one defined by Fréchet, since $\delta I(\bar{x}, h)$ might be neither linear nor continuous with respect to h. However for some cases, the Fréchet and the Gateaux derivative can be related:

Theorem 11. *Let functional $I : X \to \mathbb{R}$ be Fréchet differentiable. Then I is also Gateaux differentiable. Furthermore, the Gateaux and Fréchet derivative agree,*

$$I'(\bar{x})h = \delta I(\bar{x}, h) \tag{368}$$

for all $h \in X$.

Proof. Assuming $h = \xi\hat{h}$, the definition of Fréchet derivative gives

$$I(\bar{x} + \xi\hat{h}) = I(\bar{x}) + I'(\bar{x})[\xi\hat{h}] + \epsilon(\xi\hat{h})\|\xi\hat{h}\|. \tag{369}$$

Using the linearity of the Fréchet derivative and the homogeneous property of the norm, we have

$$\frac{I(\bar{x} + \xi\hat{h}) - I(\bar{x})}{\xi} = I'(\bar{x})\hat{h} + \frac{|\xi|}{\xi}\|\hat{h}\|\epsilon(\xi\hat{h}) \tag{370}$$

As $\xi \to 0$, $\xi h \to 0$ and $\epsilon(\xi h) \to 0$, while $\frac{|\xi|}{\xi} = \pm 1$. Therefore,

$$\delta I(\bar{x}, \hat{h}) = \lim_{\xi \to 0} \left[\frac{I(\bar{x} + \xi\hat{h}) - I(\bar{x})}{\xi} - \frac{|\xi|}{\xi}\|\hat{h}\|\epsilon(\xi\hat{h}) \right] = I'(\bar{x})\hat{h} \tag{371}$$

□

Note that the converse is not true, Gateaux differentiability does not imply Fréchet differentiability.

One of the important outcomes of Theorem 11 is that we are able to compute the Fréchet derivative applied to h by differentiating with respect to ξ at $\xi = 0$, as defined by Gateaux derivative. Meaning that,

$$I'(\bar{x})h = \frac{d}{d\xi} I(\bar{x} + \xi h)\bigg|_{\xi=0}. \tag{372}$$

Some rules that are valid for the classical notion of derivatives carry to Fréchet derivative (BERGER, 1977), for instance, the chain rule and the product rule.

The definition of the Fréchet derivative also allows expanding functionals using Taylor series (ERNZERHOF, 1994). Given a functional $I : X \to \mathbb{R}$, it can be expanded in the following form

$$I(x + h) = I(x) + I'(x)h + \text{higher order terms.} \tag{373}$$

In the case that the higher order terms are truncated the approximation error $\epsilon(h) \to 0$ as $\|h\| \to 0$.

This notion leads to an important tool for the calculus of variations, the first variation. The first variation does not differ from the definition of the Fréchet derivative obtained using the Gateaux derivative. However, it can be understood with a different intuition.

Let I be a Fréchet differentiable functional, $\bar{x}$ be a function in the space X, and δx be a function close enough to 0 that can assume any shape given that it satisfies the regularity condition of the space X. The variation δI on the functional I is caused by "perturbing" the function $\bar{x}$ with δx. The variation δI can be obtained using the Gateaux derivative,

$$\delta I(x, \delta x) = \frac{d}{\delta \xi} I(\bar{x} + \xi \delta x)\bigg|_{\xi=0} \tag{374}$$

Since the target functional has the form (345),

$$I(x) = \int_{t_0}^{t_f} F(x, \dot{x}, t)\, dt, \tag{375}$$

we can develop a generic form for the first variation of this functional.

Theorem 12 (First Variation)**.** *Given a Fréchet differentiable functional $I : X \to \mathbb{R}$ defined by*

$$I(x) = \int_{t_0}^{t_f} F(x, \dot{x}, t)\, dt, \tag{376}$$

where F is a continuously differentiable function with respect to x, $\dot{x}$, and t.

The first variation of I, namely δI, with a perturbation δx is given by

$$\delta I(x, \delta x) = \int_{t_0}^{t_f} \left[\frac{\partial F}{\partial x}(x, \dot{x}, t)\delta x + \frac{\partial F}{\partial \dot{x}}(x, \dot{x}, t)\delta \dot{x} \right] dt. \tag{377}$$

Proof. If the functional I is Fréchet differentiable, then the definition of the Gateaux and Fréchet differentiations are interchangeable. Therefore

$$\delta I(x, \delta x) = I'(x)\delta x = \frac{d}{d\xi} I(x + \xi \delta x)\bigg|_{\xi=0}. \tag{378}$$

Replacing $I(x + \xi \delta x)$ with its definition results in

$$\delta I(x, \delta x) = \frac{d}{d\xi} \int_{t_0}^{t_f} F(x + \xi \delta x, \dot{x} + \xi \delta \dot{x}, t)\, dt \bigg|_{\xi=0}. \tag{379}$$

Let us define an auxiliary variable $y = x + \xi \delta x$, with the first derivative with respect to t given by $\dot{y} = \dot{x} + \xi \delta \dot{x}$. By replacing y in the definition of $I(x + \xi \delta x)$, we obtain

$$\delta I(x, \delta x) = \frac{d}{d\xi} \int_{t_0}^{t_f} F(y, \dot{y}, t)\, dt, \bigg|_{\xi=0}. \tag{380}$$

Since F is continuously differentiable with respect to its arguments, we can move the differentiation to inside the integral,

$$\delta I(x, \delta x) = \int_{t_0}^{t_f} \frac{dF}{d\xi}(y, \dot{y}, t)\bigg|_{\xi=0} dt, \tag{381}$$

and applying the total derivative rule we obtain

$$\delta I(x, \delta x) = \int_{t_0}^{t_f} \left[\frac{\partial F}{\partial y}(y, \dot{y}, t) \frac{dy}{d\xi} + \frac{\partial F}{\partial \dot{y}}(y, \dot{y}, t) \frac{d\dot{y}}{d\xi} \right] \Bigg|_{\xi=0} dt. \tag{382}$$

Replacing y with $x + \xi\delta x$, $\dot{y}$ with $\dot{x} + \xi\delta\dot{x}$, and applying $\xi = 0$, we obtain

$$\delta I(x, \delta x) = \int_{t_0}^{t_f} \left[\frac{\partial F}{\partial x}(x, \dot{x}, t)\delta x + \frac{\partial F}{\partial \dot{x}}(x, \dot{x}, t)\delta\dot{x} \right] dt. \tag{383}$$

□

Sometimes the first variation is written as $\delta I(x)$, without exposing the dependence on the perturbation variable δx, for the sake of simplicity and readability. Alternatively, the dependence on x can also be suppressed, representing only δI.

The first perturbation can be applied to multivariate functionals. For instance given the functional

$$I(x, y) = \int_{t_0}^{t_f} F(x, y, \dot{x}, \dot{y}, t)\, dt, \tag{384}$$

the first variational $\delta I(x, y, \delta x, \delta y)$ is given by the sum of the partial derivatives multiplied by the perturbation,

$$\delta I(x, y) = \int_{t_0}^{t_f} \left[\frac{\partial}{\partial x} F(x, y, t)\delta x + \frac{\partial}{\partial \dot{x}} F(x, y, t)\delta\dot{x} + \frac{\partial}{\partial y} F(x, y, t)\delta y + \frac{\partial}{\partial \dot{y}} F(x, y, t)\delta\dot{y} \right] dt. \tag{385}$$

A.2.1 Euler-Lagrange equation

In Theorem 10 we have seen that a function x^* is an extremum with respect to a functional I if the Fréchet derivative is zero at x^*. The Euler-Lagrange equation gives a more tangible *necessary* condition for the optimality of functionals with integrals.

Theorem 13 (Euler-Lagrange Equation)**.** *Let I be a Fréchet differentiable functional of the form*

$$I(x) = \int_{t_0}^{t_f} F\left(x(t), \dot{x}(t), t\right)\, dt, \tag{386}$$

where the function F is differentiable with respect to x, $\dot{x}$, and t. The function $x(t) \in \mathcal{C}^1[t_0, t_f]$ passes through the points $x(t_0) = x_0$ and $x(t_f) = x_f$. If I has an extremum at x^*, then x^* satisfies the Euler-Lagrange equation:

$$\frac{\partial F}{\partial x}\left(x^*(t), \dot{x}^*(t), t\right) - \frac{d}{dt}\frac{\partial F}{\partial \dot{x}}\left(x^*(t), \dot{x}^*(t), t\right) = 0 \tag{387}$$

for all t in $[t_0, t_f]$. Alternatively, in a more compact format:

$$\frac{\partial F}{\partial x} - \frac{d}{dt}\frac{\partial F}{\partial \dot{x}} = 0. \tag{388}$$

Proof. The proof follows by showing that the Euler-Lagrange is an implication of Theorem 10. However, note that the set of curves in $\mathcal{C}^1[t_0, t_f]$ that meet the conditions $x(t_0) = x_0$ and $x(t_f) = x_f$ does not form a linear space, so Theorem 10 does not apply directly. Therefore, let us define the linear space H given by

$$H = \left\{ \delta x \in \mathcal{C}^1[t_0, t_f] \,\middle|\, \delta x(t_0) = \delta x(t_f) = 0 \right\}, \tag{389}$$

which has the property that for all $\delta x \in H$, $x^*(t_0) + \delta x(t_0) = x_0$ and $x^*(t_f) + \delta x(t_f) = x_f$. Which means that δx is allowed to perturbate at all t except the integral limits, otherwise the sum of $x^* + \delta x$ will not pass through (t_0, x_0) and (t_f, x_f).

Call $J : H \to \mathbb{R}$ a functional defined by

$$J(\delta x) = I(x^* + \delta x) \tag{390}$$

for all $\delta x \in H$. Notice that J has an extremum at $J(0) = I(x^*)$.

By Theorem 10, if 0 is an extremum of J then $J'(0) = 0$. So if the first variation is applied with a perturbation $\delta x \in H$,

$$\delta J(\delta x) = \int_{t_0}^{t_f} \left[\frac{\partial F}{\partial x}(x^*, \dot{x}^*, t)\delta x + \frac{\partial F}{\partial \dot{x}}(x^*, \dot{x}^*, t)\delta \dot{x} \right] dt = 0, \tag{391}$$

integrating by parts the second term, results in

$$\left[\frac{\partial F}{\partial \dot{x}}(x^*, \dot{x}^*, t)\delta x(t) \right] \Bigg|_{t_0}^{t_f} + \int_{t_0}^{t_f} \left[\frac{\partial F}{\partial x}(x^*, \dot{x}^*, t) - \frac{d}{dt}\frac{\partial F}{\partial \dot{x}}(x^*, \dot{x}^*, t) \right] \delta x \, dt = 0. \tag{392}$$

Notice that $\delta x(t_0) = \delta x(t_f) = 0$, therefore the term outside the integral vanishes. The remaining terms are

$$\int_{t_0}^{t_f} \left[\frac{\partial}{\partial x} F(x^*, \dot{x}^*, t) - \frac{d}{dt}\frac{\partial}{\partial \dot{x}} F(x^*, \dot{x}^*, t) \right] \delta x \, dt = 0, \tag{393}$$

as the function δx can be any function in H, we use the Lemma 2, which is stated in the following, to conclude that

$$\frac{\partial}{\partial x} F(x^*, \dot{x}^*, t) - \frac{d}{dt}\frac{\partial}{\partial \dot{x}} F(x^*, \dot{x}^*, t) = 0. \tag{394}$$

□

In the following we give the fundamental lemma for the calculus of variations, which supports the condition established by the Euler-Lagrange equation.

Lemma 2 (Fundamental Lemma)**.** *Given a particular function $f \in \mathcal{C}[t_0, t_f]$, if*

$$\int_{t_0}^{t_f} f(t)h(t) \, dt = 0 \tag{395}$$

for every $h \in \mathcal{C}^1[t_0, t_f]$ *such that*

$$h(t_0) = h(t_f) = 0 \tag{396}$$

we conclude that

$$f(t) = 0 \qquad \forall t \in [t_0, t_f]. \tag{397}$$

Proof. We will develop a proof by contradiction. Assume that there exists an interval $[t_1, t_2] \subset [t_0, t_f]$ in which $f(t) > 0$ (an equivalent demonstration can be made for the assumption $f(t) < 0$).

As (395) is valid for every h, we choose h as

$$h(t) = \begin{cases} 0 & \text{for } t \in [t_0, t_1) \\ (t - t_1)^2(t - t_2)^2 & \text{for } t \in [t_1, t_2] \\ 0 & \text{for } t \in (t_2, t_f] \end{cases} \tag{398}$$

Notice that the chosen h is continuously differentiable, even at the points t_1 and t_2.

The integral (395) becomes

$$\int_{t_0}^{t_f} f(t)h(t)\,dt = \int_{t_0}^{t_1} f(t)h(t)\,dt + \int_{t_1}^{t_2} f(t)h(t)\,dt + \int_{t_2}^{t_f} f(t)h(t)\,dt \tag{399a}$$

$$= \int_{t_1}^{t_2} f(t)h(t)\,dt \tag{399b}$$

$$= \int_{t_1}^{t_2} f(t)(t - t_1)^2(t - t_2)^2\,dt > 0 \tag{399c}$$

The inequality (399c) results from the assumption that $f(t) > 0$, and, at the same time, h only takes nonnegative values in this interval. Since (399c) contradicts (395), we conclude that

$$f(t) = 0 \tag{400}$$

must hold for all $t \in [t_0, t_f]$. □

Remark. *Lemma 2 can be modified to hold for a more restrictive condition. For instance the case where* $h \in \mathcal{C}^p[t_0, t_f]$*. The proof follows in the same way as the proof given in the lemma, however one must choose* $h = (t - t_1)^{p+1}(t - t_2)^{p+1}$ *for the interval* $[t_1, t_2]$*, which guarantees that the picked h is p-differentiable.*