Nonlinear Optimization based frameworks for Model Predictive Control, State-Estimation, Sensitivity Analysis, and Ill-posed Problems

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

CHEMICAL ENGINEERING

by

DAVID MOLINA THIERRY

M.S. CHEMICAL ENGINEERING, UNIVERSIDAD IBEROAMERICANA
B.S. CHEMICAL ENGINEERING, UNIVERSIDAD AUTONOMA DE PUEBLA

CARNEGIE MELLON UNIVERSITY

Pittsburgh, PA

Dec, 2019

Acknowledgments

This work is the result not only certain amount of effort put from my part, but it also has been possible thanks to the contributions, comments and support from several other people. Firstly, I would like to thank anybody taking the time to read this, because despite this being too sentimental at times, these are my sincere thoughts.

I would like to thank the committee members for my defense, Prof. Aaron Johnson, Dr. Bethany Nicholson, Prof. Erik Ydstie, Prof. Nick Sahinidis, and my advisor, Prof. Larry Biegler; for their comments and criticism that has enriched a lot of this manuscript.

Then, I must thank my advisor, Prof. Larry Biegler; for the priceless mentoring that he has offered me throughout the years. I believe that, I am typically a fairly obstinate person; yet he managed to work around this and point me into the right directions. I can only hope for me to be at least half as wise as he is, by the end of my career.

I would like to mention my fellow former and current group members. They have made me grow as a professional and as a person. I would like to mention, John Eason and Wei Wan; for their kindness and providing me with the always valuable voice of experience. Saif Kazi, and Joyce Yu for all the nice times we had in the 3106 office. Can Ekici, Robert Parker, Kuan-Han Li and Vibhav Dabadghao, for I really enjoyed the discussions we had. Also, the postdocs Michael Short, Christina Schenk, and Cornelious Masuku; moreover all the visiting people we had over the years including (but not limited to) Kai Liu, Flemming Holtorf, Karsten Rätze, and Luis Ricardez, must be mentioned.

I would like to acknowledge the CAPD and the PSE community at CMU; including my friends at the Gounaris' group, i.e. Christopher Hanselman, Anirudh Subramanyam, Akang Wang, Natalie Isenberg, Aliakbar Izadkhah, and Xiangyu Yin. Moreover, my friends at Grossmann's group, Qi Chen, Yixin Ye, David Bernal, and Can Li.

I am also thankful to Prof. Flores which is greatly responsible from introducing me to Carnegie Mellon University; and Prof. Ortega and Prof. Galicia for their support and encouragement at early stages of my career. In this note, a special mention must be made to Prof. Grossmann for all his encouragement and inspiration, that eventually made me decide on continuing the path of becoming a PhD student at Carnegie Mellon University.

I greatly acknowledge the funding from NETL and the IDAES project. I must admit that being involved with IDAES was quite an experience, because of all the interactions and collaborations with people from several national laboratories, universities and companies.

Finally, I thank my family for their support, and particularly my Mother Consuelo Thierry for all her sacrifice, since none of this would have been possible otherwise.

David Paul Molina Thierry
Pittsburgh, PA
Dec 2019

Abstract

The development of robust and efficient algorithms for nonlinear optimization is an important matter; because they make possible the solution of models with increasingly size and complexity. In this context, the Interior Point Optimizer (IPOPT) has been proven to be a competitive algorithm, because of its ability to handle large scale problems. However, there are some specific situations in which convergence can not be guaranteed. For instance the violation of a regularity condition given by the Linear Independence Constraint Qualification (LICQ). This property is given in terms of the consistency of local linearizations of the set of constraints, i.e. the gradients. If the set of gradients of equality and active bound inequality are linearly independent, then the LICQ holds. It turns out that if this property is not satisfied, IPOPT would attempt to regularize the system by perturbing it, with the hope to overcome regions of local inconsistency. However, if the LICQ does not hold globally or if the system has severe ill-conditioned, the regularization strategies will fail. Thus, we present the ℓ_1-exact penalty-barrier strategy to deal with this lack of regularity. This works by penalizing the infeasibility with a parameter at the objective; and then by reformulating the problem to obtain a higher dimensional problem, the LICQ property holds at all points for which the new variables are nonzero. Thus achieving higher level of robustness as opposed to the regular IPOPT.

Moreover, an unified framework for nonlinear model-based state-estimation and control with parametric sensitivity is presented. Typically, control strategy of a processes is tied to the ability to accommodate online computations; in this sense nonlinear optimization strategies like Moving Horizon Estimation (MHE) and Model Predictive Control (NMPC) are overlooked. Nevertheless, it has been proposed to use the parametric nature of these optimization problems to create a scheme in which the bulk of computations is partitioned into a *background* phase with all the most computationally expensive steps, and an *online* phase, in which the sensitivity information is used to make inexpensive calculations for the state estimate or controller input. These ideas have been put together into a Python framework, that streamlines the creation of a state-estimation and control strategy with the additional benefit of having parametric sensitivity embedded into it. This framework works by using the Pyomo libraries to create optimization objects and automatically discretize the models given by the user. We demonstrate the effectiveness of the framework in two problems of vastly different complexity. A benchmark distillation case study and a Bubbling Fluidized Bed Reactor (BFB) for the capture of CO_2. The results show that sensitivity is effective for online implementations of the controller and it was seamlessly enabled by our framework.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As in several other engineering disciplines, Chemical Engineering has an element of decision making based on mathematical models, created to reflect its real world counter part. A great deal of the ongoing developments in this area are related to the expansion of the comprehension of processes and phenomena, which are of particular interest to society. Moreover, these processes can have vastly different scales: from molecular to plant; or from milliseconds to years or even decades. Yet, arguably all of these can be represented as variables in equations, drawing relationships between each other. Therefore, it is possible to claim that most developments in Chemical Engineering are linked to a mathematical relationship, or a model.

At the same time as a new development in Chemical Engineering comes to existence there is also new set of possible decisions to make. And, for many years practitioners have pursued systematic ways to perform these decisions with the most confidence. Meanwhile mathematicians have dealt with the optimization problem, which at its basic form seeks to find the *critical* points of a set of variables given an *objective*, which is either *minimized* or *maximized*. From this point in time, the leap towards the modern concept of optimization has been substantial. More importantly, the overlap between engineering as a whole and mathematical optimization has resulted into a long-prolific relationship, that keeps both stakeholders satisfied, and researchers intrigued.

As the relationship of mathematical optimization and Chemical Engineering prospered, there were (and are still ongoing) several significant applications that spawned throughout the years, for example flowsheet synthesis, equipment sizing and costing, process control, etc [1]. The community in charge of these developments largely embraced the state-of-art

of optimization algorithms and software. This is reflected in the strategies to handle the aforementioned applications, e.g. decompose when the problem is intractable, linearize whenever there is strong equivalency of the formulations, etc. Eventually, the capabilities of the algorithms enabled fairly detailed models of processes into optimization. Such is the case of nonlinear models and Nonlinear Programming (NLP).

Nonlinear programs have evolved to the point of providing high detail representations, particularly for smooth and differentiable models. There is also the inclusion of sophisticated techniques into state-of-art solvers that have been able to exploit structure and handle large-scale problems [2], [3], [4], thus, increasing the range of applications of nonlinear models within Chemical Engineering. Nevertheless, there still exist several challenges from the modeling, and from the algorithmic point of view. For instance, the solution of Mathematical Programs with Complementarity Constraints [5], [6], [7], which enables discrete decisions to be modeled entirely with continuous variables, with the compromise of lacking regularity conditions on the system of constraints. Also, related to this problem is the solution of nonlinear programs with inconsistent linearizations of the constraints, i.e. ill-posed NLPs [8], [9], [10]. Moreover, there is the implementation of concurrency at the linear algebra level from the factorizations that take place during the optimization [11], [12], and finally, extensions of parametric sensitivity to address changing active sets and weaker conditions [13]. These are a rather small sample of the challenges found in NLP. Ultimately, the optimization community is actively producing enhancements on the state-of-art solvers, that will generate a more robust and efficient implementations, to solve problems with increasingly more complexity.

Moreover, models that use algebraic representation of differential terms are extremely valuable, since they enable decision making of transient processes. These naturally suggest the possibility of designing optimal controllers by combining the models and the NLP algorithms. This in turn, leads to a rapid growth of the range of dynamic optimization applications. From parameter estimation frameworks [14], nonlinear state-estimation [15],

[16]; and possibly the most popular instance of dynamic optimization problems, i.e. Nonlinear Model Predictive Control, which has seen rapid developments from real-time solution strategies [17], [18] to economic-driven objective functions [19], [20].

1.1 Research Statement

In this dissertation we first explore the ideas of expanding the robustness properties of the interior-point solver IPOPT. Traditionally, the solutions of NLPs are identified by analyzing the relationships between the first and second-order information of the constraints and objective function. This is assumed to be true because the geometry of the feasible set can be captured by linearizations of the system of constraints. Moreover, this assumption is only valid if the gradients of the equality and active inequality constraints are linearly independent; thus, if this is not the case optimality can not be asserted. Within the interior-point framework of IPOPT violation of this last requirement might happen during the solution procedure, and as a result convergence cannot be guaranteed. In both theoretical and practical senses, the posedness of the problem depends significantly on the properties of the local linearizations of the constraints. As a result, IPOPT (and most NLP solvers in general) require strategies to *regularize* the current iterate if such situation arises. Currently, within IPOPT this is covered in an algorithm denominated *Inertia correction*, which seeks to obtain a specific combination of positive, negative and zero eigenvalues of the iteration matrix by perturbing the system with two positive scalars. It turns out that this strategy has proved successful, especially when the case studies are not badly conditioned. Moreover, if the inconsistencies happen globally, i.e. at all points of the searchable space, then the aforementioned strategy will not succeed.

This is the reason why in Chapter 3, the ℓ_1-exact penalty-barrier strategy within IPOPT is presented. As an outline of this strategy, it is possible to construct problems which penalize the equality constraints with a penalty parameter and combine them with the logarithmic barrier strategy from the interior-point framework. Nominally, this problem

is able to find stationary points of the original constrained, if the value of the penalty parameter is above a critical value, which is not known *a priori* [21]. This exemplifies one of the most notorious difficulties with this algorithm. Nevertheless in our implementation we discuss a strategy to update the penalty based on the local curvature and linearizations of the constraints and objective function. Subsequent testing revealed that this strategy is competitive against the nominal version of IPOPT 3.12.9. The most relevant contribution of this section is that a framework can be constructed in which ill-posedness in the constraint sense can be overcome with a variation of the original optimization problem.

The second topic covered in this dissertation is the aggregation of ideas for control, state estimation, and parametric sensitivity into a software framework particularly targeted towards significantly-sized problems in the context of energy production. For this, Model Predictive control (MPC) assumes that there exists a nominal model of a process of interest, which is combined with the current information about its current state; then it is possible to make predictions about optimal sequences of predicted states and inputs by solving an optimization problem. Typically, assumptions are made to reduce the complexity of the associated optimization problem. For example, one rather popular instance of this problem considers linear constraints and quadratic objective function, which is the basis for Linear MPC (LMPC) and requires to solve Quadratic Programs (QPs). This strategy is relevant because the complexity of the optimization problem is in direct conflict with its online implementation, and in this sense solving QPs can be usually more accessible than solving full NLPs. However, if the underlying model of the process is characterized by significant nonlinearities, the LMPC might have reduced performance. Accordingly, a Nonlinear MPC strategy is more appropriate; however for online applications one must overcome the computational complexity of the solution of these problems.

Similar issues are found in the context of Nonlinear state-estimation strategies, which are arguably as important as the control strategy. The core aspect of this problem is that given a model of the process and sequence of measurements generated from its initial

state, the respective sequence of states is to be determined. As an analogy to MPC, this can also be formulated as an optimization problem whose solution represents the optimal sequence of estimated states and disturbances; moreover the optimal *linear* state-estimator is given by the Kalman-Filter; and similar to LMPC its effectiveness depends on the degree of nonlinearities of the process. As an alternative, the Nonlinear state-estimator known as the full-information estimator handles directly the nonlinearities, but for continuous process it becomes increasingly intractable as the size of the problem grows at a constant rate over time. Therefore, the concept of a moving horizon-window applied into this problem gives inception to the Moving Horizon Estimator (MHE), which potentially has the same limitations for online applications given its computational cost.

A strategy to overcome this online computational cost is given by nonlinear parametric sensitivity analysis. This enables the partition of the computation into two parts, online and background, while retaining the properties of the optimal solution, and it is relevant because the online computations are reduced to a set of matrix-vector multiplications, which are typically much less expensive than the solution of an NLP.

All of these ideas were encapsulated into an unified framework for control-estimation-sensitivity, i.e. the Control Adaptation with Predictive Sensitivity Enhancement (CAPRESE). The main motivation towards CAPRESE is to systematically enable the use of nonlinear models for the design of a controller, particularly for challenging energy applications, like CO_2 capture. Moreover, the sensitivity elements of it enable the MHE-NMPC strategies to be computed online, with computations on the order of seconds. CAPRESE was built in Python, which is an accessible programming language for all potential new users. Additionally, it uses Pyomo for the modelling of the process, thus providing access to several state-of-art optimization solvers, and at the same time keeping everything in a single environment. In CAPRESE the workflow starts by the creation of a Differential Algebraic (DAE) model by the user which is automatically discretized by the `pyomo.dae` library. Then CAPRESE creates an optimization object which is able to assemble the rele-

vant optimization problems, as well as the initialization strategies. Therefore, in Chapter 5, the CAPRESE framework is used to control a problem with significant size and nonlinearities in the context of CO_2 capture for energy generation. It is shown that, even in the presence of noise, the framework can generate a fast controller action with sensitivity and be able to attain reasonable performance.

1.2 Dissertation Outline

As previously discussed, this dissertation presents two main issues; one related to nonlinear optimization and the second to a framework for MHE-NMPC with parametric sensitivity analysis. In Chapter 2, the preliminaries for the ℓ_1-exact penalty barrier algorithm in IPOPT are presented, including the origins and relevance of *constraint qualifications*, an outline of the *IPOPT algorithm*, and some useful *post-optimality* analysis issues. Chapter 3 presents the ℓ_1-exact penalty barrier algorithm in IPOPT for the *failure of Constraint Qualifications*. It discusses several details on the underlying linear systems, penalty update, overview of the algorithm, and results in a set of test problems including a modified version of CUTEr test set with degenerate constraints, and a library that contains Mathematical Programs with Complementarity Constraints (MPCCs). Then, Chapter 4 discusses some of the background on how Differential Algebraic Equations (*DAEs*) are handled in this work. Moreover, the underlying *concepts* and *robust stability* properties of MHE and NMPC are discussed. Finally, the *advanced-step* algorithms to enable online computations of input and estimated state are discussed. Chapter 5 discusses the CAPRESE framework and shows an application related to CO_2 capture for energy generation, and presents the performance of the CAPRESE framework alongside a example benchmark case study of a distillation column. Finally the conclusions of this dissertation are laid out in Chapter 6.

Chapter 2

Nonlinear Optimization and Sensitivity Analysis

2.1 Introduction

In this work we will be concerned with Nonlinear Programming (NLP) problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0, \\ & x \geq 0, \end{aligned} \tag{2.1}$$

where it is assumed that $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are C^2 with Lipschitz continuous second derivatives and $n \geq m$. This problem appears in several engineering applications, e.g., as the result of discretization of differential algebraic systems, parameter estimation, etc.

The efficient solution of this problem has been the subject of several studies, that over time have reduced the number of assumptions to be made in order to establish the convergence of algorithms. Nevertheless, a solution of problem (2.1) is characterized by Taylor's series approximations of objective and constraint functions and the feasible set $\mathcal{X}$. For problem (2.1) the feasible set is $\mathcal{X} \coloneqq \{x \in \mathbb{R}^n \mid c(x) = 0, x \geq 0\}$. A point is feasible if $x \in \mathcal{X}$. Moreover, a feasible sequence of points that approaches x is given by $\{y_k\}$ such that $y_k \in \mathcal{X}$ and $\lim_{k\to\infty} y_k = x$.

Before describing what constitutes a solution of (2.1), a fundamental property that relates the feasible set and the constraint functions needs to be mentioned. This is the issue of the

Constraint Qualifications.

2.2 Constraint Qualifications

The Constraint Qualifications (CQs) refer to properties of the algebraic set of constraints, in particular its linearizations at neighborhoods of points. Though there are several CQs with complex relationships among each other, for the purposes of this work, the Linear Independence Constraint Qualification (LICQ) is the most relevant one. In order to define it, the concept of active set needs to be considered.

Definition 1 (Active set and Active bound gradient)**.** *Consider a problem with bounds $x \geq 0$, the active set of bounds for a feasible point x,*

$$\mathcal{A}(x) = \{i \in \{1, 2, \ldots, n\} \mid x_i = 0\}, \tag{2.2}$$

with $n_{\mathcal{A}} = |\mathcal{A}(x)|$. Note that a problem might have variable upper and lower bounds, but without loss of generality it is consider only the later. Then, let a matrix $E_{\mathcal{A}}(x) \in \mathbb{R}^{n \times n_{\mathcal{A}}}$ such that

$$E_{\mathcal{A}}(x)^T x = 0; \tag{2.3}$$

in which the j^{th} column of the matrix ($E_{\mathcal{A},j}$) is given in terms of the columns of an identity matrix of appropriate size, i.e. $E_{\mathcal{A},j}(x) = I_{i(j)}$ and $i(j)$ is the j^{th} element of $\mathcal{A}(x)$.

In other words, the active set for problem (2.1) contains the indices of the variables for which the bounds are active, and there exists a matrix in which each column contains the value of one at the position of a given active variable and zeroes anywhere else. As a consequence, the LICQ can now be defined.

Definition 2 (Linear Independence Constraint Qualification)**.** *Consider a problem with equality constraints $c(x) = 0$ and bounds $x \geq 0$. The linear independence constraint qualification holds at the feasible point x if the combined gradients of equality constraints and active bounds (from definition 1) are linearly independent. In other words, the matrix,*

$$\begin{bmatrix} \nabla c(x) & E_{\mathcal{A}}(x) \end{bmatrix} \tag{2.4}$$

is full column rank.

Alternatively, the Jacobian matrix of the combined active set and equality constraint, i.e. $\mathcal{J} \in \mathbb{R}^{(m+n_{\mathcal{A}})\times n}$ such that $\mathcal{J}(x)^T \coloneqq [\nabla c(x)\, E_{\mathcal{A}}(x)]$, must be full row rank. Satisfaction of the LICQ guarantees that at a given feasible point, the local linearizations successfully describe the geometric features of the feasible set; this is a fundamental feature required to ensure local optimality. To show this, the definition of a limiting direction, tangent cone, and the set of linearized feasible directions must be stated.

Definition 3 (Tangent cone)**.** *Let $\{y_k\}$ be a feasible sequence approaching $x \in \mathcal{X}$ and a sequence of decreasing scalars $\{t_k\}$ (with $\lim_{k\to\infty} t_k = 0$). Then there exist a vector w such that*

$$\lim_{k\to\infty} \frac{y_k - x}{t_k} = w, \tag{2.5}$$

and w is said to be tangent to $\mathcal{X}$. Furthermore, at the local solution x^*, the set of all the tangent vectors is represented by $T_{\mathcal{X}}(x^*)$.

Definition 4 (Set of linearized feasible directions)**.** *Consider a feasible point x and a problem with equality constraints $c(x) = 0$, bounds $x \geq 0$ and active set $\mathcal{A}(x)$. The set of linearized feasible directions $F(x)$ is given as follows:*

$$F(x) = \left\{ p \mid \nabla c_i(x)^T w = 0 \quad \forall i \in \{1,...,m\}, E_{\mathcal{A},i}(x)^T w \geq 0 \quad \forall i \in \mathcal{A}(x) \right\}. \tag{2.6}$$

Thus, it is now possible to show the importance of the LICQ, given by the following theorem.

Theorem 1 (Linearized feasible directions and tangent cone equivalence)**.** *If the LICQ holds at the local solution x^*, then the linearized set of feasible directions and the tangent cone are equivalent.*

Proof. Let $\mathcal{Z}(x^*)$ be a matrix whose columns are a basis for the null space of $\mathcal{J}(x^*)$, i.e. $\mathcal{Z}(x^*) \in \mathbb{R}^{n\times(n-m-n_{\mathcal{A}})}$ such that $\mathcal{J}(x^*)\,\mathcal{Z}(x^*) = 0$, and is full column rank. Suppose there is sequence of decreasing scalars $\{t_k\}$ such that $\lim_{k\to\infty} t_k = 0$. The outline of the proof

is to use the LICQ property and generate a feasible sequence that converges to x^* and has an arbitrary vector w such that $w \in F(x^*)$ and $w \in T_{\mathcal{X}}(x^*)$. First, consider the system of equations

$$\left[\begin{bmatrix} c(y_k) \\ E_{\mathcal{A}}(x^*)^T y_k \end{bmatrix} - t_k \begin{bmatrix} \nabla c(x^*)^T w \\ E_{\mathcal{A}}(x^*)^T w \end{bmatrix}\right] = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ 0 \end{bmatrix}, \tag{2.7}$$

where it is assumed that $w \in F(x^*)$. For a fixed nonnegative value of t_k, the system (2.7) has an unique solution. This is true because the gradient of (2.7) with respect to y_k,

$$\left[\begin{bmatrix} \nabla c(x^*) & E_{\mathcal{A}}(x^*) \end{bmatrix} \mathcal{Z}(x^*)\right],$$

is nonsingular by the LICQ property and the construction of $\mathcal{Z}$. Moreover, since $w \in F(x^*)$, the second bracket in the first row of system (2.7) has the properties of (2.6); which means that y_k is feasible. Thus, the system (2.7) yields a feasible sequence for which $y_k \to x^*$ as $t_k \to 0$. Finally, to prove that $w \in T_{\mathcal{X}}(x^*)$, a first order Taylor's series expansion on bracketed term of the first row of (2.7) yields the following:

$$\begin{aligned} \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ 0 \end{bmatrix} &= \left[\begin{bmatrix} \nabla c(x^*)^T (y_k - x^*) + o(\|y_k - x^*\|) \\ E_{\mathcal{A}}(x^*)^T (y_k - x^*) + o(\|y_k - x^*\|) \end{bmatrix} - t_k \begin{bmatrix} \nabla c(x^*)^T w \\ E_{\mathcal{A}}(x^*)^T w \end{bmatrix}\right], \\ &\qquad \mathcal{Z}(x^*)^T (y_k - x^* - t_k w) \\ &= \begin{bmatrix} \begin{bmatrix} \nabla c(x^*)^T \\ E_{\mathcal{A}}(x^*)^T \end{bmatrix} \\ \mathcal{Z}(x^*)^T \end{bmatrix} (y_k - x^* - t_k w) + o(\|y_k - x^*\|), \end{aligned}$$

in which the matrix in the brackets is nonsingular. With this result and dividing by t_k, the following result is obtained

$$\frac{y_k - x^*}{t_k} = w + o\left(\frac{\|y_k - x^*\|}{t_k}\right).$$

In consequence, $w \in T_{\mathcal{X}}(x^*)$. □

The LICQ property enables the characterization of the tangent cone from first order information of the constraints. This is important when analyzing a necessary condition for optimality, e.g. $\nabla f(x^*)^T w \geq 0 \quad \forall w \in T_{\mathcal{X}}(x^*)$, because if the LICQ holds, then $T_{\mathcal{X}}(x^*) = F(x^*)$, and a compact set of conditions for optimality can be deduced using Taylor's theorem. However, the failure of the LICQ disables such assertions; thus proving optimality becomes difficult.

Finally, it is convenient to define a weaker CQ for later sections of this work, particularly the Mangasarian-Fromovitz Constraint Qualification (MFCQ).

Definition 5 (Mangasarian-Fromovitz Constraint Qualification)**.** *Consider a problem with equality constraints $c(x) = 0$ and bounds $x \geq 0$. The MFCQ holds at the feasible point x if the gradients of the equality constraints at x, $\nabla c_i(x), i \in \{1, \ldots, m\}$ are linearly independent and given an active bound set $\mathcal{A}(x)$, if there exists a nonzero vector w such that,*

$$\begin{aligned} \nabla c_i(x)^T w &= 0 \quad \forall i \in \{1, ..., m\}, \quad \textit{and} \\ E_{\mathcal{A},i}(x)^T w &> 0 \quad \forall i \in \mathcal{A}(x). \end{aligned} \tag{2.8}$$

2.3 First and Second Order Optimality Conditions

After defining the necessary properties of the constraints that enables the use of first order derivative information to delimit a solution of problem (2.1); a rather convenient set of conditions can be used to assert local optimality. Thus, the solution of problem (2.1) is characterized by triplet $(x^*, \lambda^*, z^*) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of primal and dual variables. This set of equations is known as the Karush-Kuhn-Tucker (KKT) conditions, and for a general non-convex feasible set $\mathcal{X}$ and functions is necessary for local-optimality. This is stated in the following definition.

Definition 6 (First Order KKT point)**.** *Assume that the LICQ holds, then vector x^* is a KKT*

point if there exists multipliers z^* and λ^*, such that

$$g(x^*) + A(x^*)\lambda^* - z^* = 0 \qquad \textit{(stationarity)}, \tag{2.9}$$

$$c(x^*) = 0, \quad x^* \geq 0 \qquad \textit{(feasibility)}, \tag{2.10}$$

$$z^* \geq 0 \qquad \textit{(nonnegativity of multipliers)}, \tag{2.11}$$

$$x_i^* \cdot z_i^* = 0 \quad \forall i \in \{1, 2, ..., n\} \qquad \textit{(complementarity)}, \tag{2.12}$$

where the gradients of objective function is denoted by $g(x) \coloneqq \nabla f(x)$ and the transpose of the Jacobian of the constraints is $A(x) \coloneqq \nabla c(x)$.

To further assess the nature of a KKT point, it is also necessary to assess the curvature of the Lagrange function. This is because at such point it might be the case that the directions associated with the KKT conditions are not informative enough to assess whether a perturbation results in an increase or decrease of the objective function $f(x)$. For this consider the Lagrange function $\mathcal{L} : \mathbb{R}^{2n+m} \to \mathbb{R}$,

$$\mathcal{L}(x, \lambda, z) \coloneqq f(x) + c(x)^T \lambda - x^T z. \tag{2.13}$$

This function combines the information of objective and constraints, and it enables a more compact notation of first and second order conditions. The second order conditions are defined with respect to a subset of the optimal linearized feasible directions. For this, the strictly active set is considered, i.e. bounds that are active and have strictly positive multipliers, $i \in \mathcal{A}(x)$ and $z_i > 0$. This is given as follows

$$\begin{aligned} \mathcal{C}(x^*, z^*) = \Big\{ & w \in F(x^*) \mid A(x^*)^T w = 0 \quad \forall i \in \{1, \ldots, m\}, \text{and} \\ & E_{\mathcal{A}}(x^*)^T w = 0 \quad \forall i \in \mathcal{A}(x^*) \text{ with } z_i^* > 0 \Big\} \\ & E_{\mathcal{A}}(x^*)^T w \geq 0 \quad \forall i \in \mathcal{A}(x^*) \text{ with } z_i^* = 0 \Big\}. \end{aligned} \tag{2.14}$$

With this the set of second order necessary conditions can be defined.

Definition 7 (Second Order Necessary Conditions (SONC))**.** *Assume the LICQ holds, then for a local optimal point x^* and multipliers λ^*, z^*. Then, the following holds*

$$w^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*, z^*) w \geq 0 \quad \forall w \in \mathcal{C}(x^*, z^*). \tag{2.15}$$

In other words, at the optimal solution of problem (2.1), the Hessian of the Lagrange function has nonnegative curvature along the feasible directions $w \in \mathcal{C}(x^*, z^*)$. It should be noted that if the inequality of (2.15) is changed to a strict inequality, and $w \neq 0$ then the definitions holds for a strict local solution; these conditions are known as Second Order Sufficient Conditions or (SOSC). Furthermore, it is convenient to define an alternative way of checking second order conditions. This can be done by assembling a projected Hessian matrix, that under certain assumptions does not require checking for all possible directions $w \in \mathcal{C}(x^*, z^*)$. For this purpose, strict complementarity must be defined.

Definition 8 (Strict Complementarity)**.** *Consider a local solution of problem* (2.1)*; strict complementarity holds if*

$$z_i^* > 0 \quad \forall i \in \mathcal{A}(x^*). \tag{2.16}$$

Definition 9 (Reduced Hessian)**.** *Consider a local solution of problem* (2.1)*. If the LICQ and Strict Complementarity hold; then the second order conditions can be verified by checking the properties of the reduced Hessian matrix,*

$$H_R = \mathcal{Z}(x^*)^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*, z^*) \mathcal{Z}(x^*), \tag{2.17}$$

where $\mathcal{Z}(x^*) \in \mathbb{R}^{n \times (n-m-n_{\mathcal{A}})}$ is a matrix whose columns are the basis for the null space of $\mathcal{J}(x^*)$ (the same as in Theorem 1), i.e. $\mathcal{J}(x^*)\mathcal{Z}(x^*) = 0$. Note that under these assumptions any vector $w \in \mathcal{C}(x^*, z^*)$ can be represented by $w = \mathcal{Z}(x^*)u$ for some $u \in \mathbb{R}^{(n-m-n_{\mathcal{A}})}$; in other words $\mathcal{C}(x^*, z^*)$ is equal to the nullspace of $\mathcal{J}(x^*)$. Thus, it suffices to check the positive definiteness of H_R to satisfy SONC and SOSC.

The definitions for first and second order conditions are fundamental for solving NLP problems, because most NLP solvers have been built with the intention to approximate points in which these conditions hold. Nevertheless, generating such points is not a trivial task, and there is still several efforts to generate more efficient and reliable algorithms in the optimization community.

2.4 Interior Point Algorithm - IPOPT

Nonlinear programming theory and algorithms experienced a surge in popularity during the 1960s. Over time two kinds of strategies to handle these problems emerged as the main schools of thought, i.e. active set and interior point strategies. The main difference between them is the way the active set of inequalities (bounds for problem (2.1)) is computed. In particular, interior point approaches the active set of the solution asymptotically as opposed of having to determine it combinatorially. As NLP problems grew in size, interior point strategies became increasingly appealing. Moreover, the development of large-scale linear algebra packages became the landmark moment for the state-of-art interior point solvers. One of the state-of-art solvers that has its roots in this developments is IPOPT. IPOPT (Interior Point OPTimizer) combines the ideas of interior point with a *filter line-search* to efficiently achieve global convergence . The primary idea is to solve sequences of parametrized problems for which the bounds of problem (2.1) are moved into the objective function. This is known as the barrier problem. The main feature of this new problem is that the modified objective function (denominated *barrier function*) will remain bounded as long as the current point resides inside the strict interior of the feasible region given by the bounds. The barrier problem is given as follows,

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & \varphi_{\mu_j}(x) := f(x) - \mu_j \sum_i^n \ln x^{(i)} \\ \text{subject to} \quad & c(x) = 0, \end{aligned} \tag{2.18}$$

where $\mu_j \in \mathbb{R}^1_{>0}$ is a strictly positive scalar denominated as barrier-parameter. The strategy dictates that assuming that the Mangasarian-Fromovitz Constraint Qualification (MFCQ) and SOSC hold for (2.1), solving (2.18) for a strictly decreasing sequence of barrier parameters, i.e. $\{\mu_j\}$ such that $\lim_{j\to 0}\mu_j = 0$, will generate a differentiable trajectory of solutions $(x(\mu_j), \lambda(\mu_j), z(\mu_j))$ that asymptotically approaches (x^*, λ^*, z^*), that is the solution of problem (2.1).

To solve (2.18), a globalized Newton algorithm is used directly on its set of first order

optimality conditions (primal-dual equations) for a fixed value of the barrier parameter

$$\begin{bmatrix} g(x) - z + A(x)\lambda \\ c(x) \\ XZe - \mu_j e \end{bmatrix} = 0, \tag{2.19}$$

where the primal dual bound multiplier is defined as $z(\mu_j) \in \mathbb{R}^n \coloneqq \mu_j X^{-1} e$; and the diagonal matrices for primal variables and bound multipliers are $X \coloneqq \text{diag}(x)$ and $Z \coloneqq \text{diag}(z)$. Moreover, to keep the barrier function bounded it is necessary to enforce strictly positive primal and dual variables $x, z > 0$. Then, a Newton step on the primal-dual equations leads to the following linear system

$$\begin{bmatrix} H_k & A_k & -I \\ A_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \\ d_k^z \end{bmatrix} = - \begin{bmatrix} g_k - z_k + A_k \lambda_k \\ c_k \\ X_k Z_k e - \mu_j e \end{bmatrix}, \tag{2.20}$$

where the subscript notation signals a function evaluated at a particular point, e.g. $g_k \coloneqq g(x_k)$; $H_k \coloneqq \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k, z_k)$ is the Hessian matrix of the Lagrangian, and $\left(d_k^x, d_k^\lambda, d_k^z\right) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ are the search directions. The matrices associated with linear systems like (2.20) are typically *sparse*, this suggest that sparse linear algebra techniques are generally appropriate to deal with these kind of system. Moreover, it is rather desirable to factorize a *symmetric* version of the matrix from (2.20), because indefinite symmetric sparse linear solvers have several useful features, e.g. *inertia* calculation capabilities. The *inertia* of a matrix K is defined as $\text{In}(K) \coloneqq (k_+, k_-, k_0)$, where k_+, k_-, and k_0 are respectively the number of positive, negative and zero eigenvalues of matrix K. This property is essential to determine whenever the search is attempted at points in which SOSC and LICQ do not hold, and convergence can not be guaranteed.

Therefore, to obtain a symmetric form of the matrix in (2.20), one can pivot on the lower-right block of the matrix in (2.20) to obtain the following linear system

$$\begin{bmatrix} W_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix} = - \begin{bmatrix} g_k - \mu_j X_k^{-1} e + A_k \lambda_k \\ c_k \end{bmatrix}, \tag{2.21}$$

which has an indefinite symmetric matrix known as the Augmented KKT matrix, and the augmented primal-dual Hessian matrix is defined by $W_k := H_k + \Sigma_k$ and $\Sigma_k := X_k^{-1} Z_k$. Additionally, the remaining vector is given as

$$d_k^z = \mu_j X_k^{-1} e - z_k - \Sigma_k d_k^x. \tag{2.22}$$

Thus, enabling symmetric linear solvers with inertia computation capabilities allows to correct situations in which the Hessian of the Lagrange has nonnegative curvature along the search directions (violation of SOSC) and the gradients of the equality constraints are linearly dependent (violation of LICQ). This is due to a relationship between the eigenvalues of the augmented KKT matrix, and the Hessian and Jacobian blocks.

Definition 10 (Inertia and SOSC relationship)**.** *Consider definition 9 of the reduced Hessian, and let the augmented KKT matrix*

$$K_k \coloneqq \begin{bmatrix} W_k & A_k \\ A_k^T & 0 \end{bmatrix}. \tag{2.23}$$

Suppose A_k is full column rank. If the reduced Hessian H_R is positive definite, then the inertia $In\,(K_k) = (n, m, 0)$.

Definition 9 tacitly assumes the linear independence of the constraint gradients, otherwise the inertia would have at least one zero eigenvalue or alternatively the matrix can be detected to be singular. In either case, corrections to the inertia can be introduced by perturbing directly the upper left block and the lower right block of K_k. Let $\delta, \delta_c \in \mathbb{R}^1_{\geq 0}$, then the regularized system is shown below,

$$\begin{bmatrix} W_k + \delta I & A_k \\ A_k^T & -\delta_c I \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix} = - \begin{bmatrix} g_k - \mu_j X_k^{-1} e + A_k \lambda_k \\ c_k \end{bmatrix}. \tag{2.24}$$

Therefore, the inertia has to be monitored at every single iteration of IPOPT. Note that if K_k becomes severely ill-conditioned and the perturbations, especially δ_c, might not be

sufficient to correct the inertia of the matrix. The algorithm to compute the search direction with the correct inertia is given in Algorithm 2.1.

Data: Constants $0 < \bar{\delta}_w^{\min} < \bar{\delta}_w^0 < \bar{\delta}_w^{\max}$; $\bar{\delta}_c > 0$; $0 < \kappa_w^- < 1 < \kappa_w^+ < \bar{\kappa}_w^+$; $\kappa_c \geq 0$, and last perturbation made $\delta^{\text{last}} \leftarrow 0$.

IC.1 **Factorize unperturbed matrix from** (2.24). I.e. $\delta = \delta_c = 0$; get `status`;

if `status` $=$ `SymSolverSuccess` ***and*** $In\,(K) = (n, m, 0)$ **then**

  Augmented KKT is nonsingular, ;

  break ;

IC.2 **if** `status` $=$ `SymSolverSingular` ***or*** *nonzero eigenvalues* $k_0 > 0$ **then**

  Iteration matrix might be singular, attempt to regularize ;

  $\delta_c \leftarrow \bar{\delta}_c \mu_j^{\kappa_c}$;

else $\delta_c \leftarrow 0$;

IC.3 **if** $\delta^{last} = 0$ **then** $\delta_w \leftarrow \bar{\delta}_w^0$;

else $\delta \leftarrow \max\left\{\bar{\delta}_w^{\min}, \kappa_w^- \delta^{\text{last}}\right\}$;

while `true` **do**

IC.4 **Factorize perturbed matrix**; get `status` ;

  if `status` $=$ `SymSolverSuccess` ***and*** $In\,(K) = (n, m, 0)$ **then**

    Success; $\delta^{\text{last}} \leftarrow \delta$;

    break ;

IC.5 **if** $\delta^{last} = 0$ **then** $\delta \leftarrow \bar{\kappa}_w^+$;

  else $\delta_w \leftarrow \kappa_w^+ \delta$;

IC.6 **if** $\delta > \bar{\delta}^{\max}$ **then** Abort; start **Feasibility Restoration**. See remark. ;

end

Algorithm 2.1: Inertia correction

Most notably from Algorithm (2.1), the search direction will be accepted as long as the inertia matches the requirement previously described. Then, if the factorization is accepted, a step-size has to be determined particularly for primal variables. To this end, and as a requirement of the interior point algorithm; the iterates must remain strictly inside the

bounds, therefore the maximum step sizes that can be taken are given by the so called fraction to the boundary rule, i.e.

$$\begin{aligned} \alpha_k^{\max} &\coloneqq \max\left\{\alpha \in (0,1] \mid x_k + \alpha d_k^x \geq (1-\tau_j)\, x_k\right\}, \\ \alpha_k^{z} &\coloneqq \max\left\{\alpha \in (0,1] \mid z_k + \alpha d_k^z \geq (1-\tau_j)\, z_k\right\}, \end{aligned} \tag{2.25}$$

where $\tau_j \coloneqq \max\{\tau_{\min}, 1-\mu_j\}$. The determination of the step size α_k is given by the backtracking line-search algorithm.

2.4.1 Filter Line-Search

A key aspect of the Newton algorithm embedded into IPOPT, is the Filter Line-Search. Though the Newton algorithm enables fast convergence to the solution, it requires to limit itself to the neighborhood of the solution. This is one of the reasons why a globalization strategy needs to be added to the solver. In most Newton-based NLP solvers the choice is an instance of a Trust-Region or a Line-Search algorithm. On top of that, such algorithms evaluate the quality of the search directions by using exogenous measures of optimality, e.g. a merit function, for which which historically penalty functions were used. These functions combine objective function and a product of the constraint violation and a penalty parameter, however they require a mechanism to update such parameter which is not a trivial task. As an alternative, IPOPT uses a filter, which a set of pairs of points of barrier objective value and constraint violation that have been historically found through the search. These form a region that is prohibited for exploration, and thus drive the optimization towards more favorable points.

Consider the constraint violation $\theta(x) \coloneqq \|c(x)\|$, then the filter set

$$\mathcal{F}_k \subseteq \left\{\left(\theta_k, \varphi_{\mu_j,k}\right) \in \mathbb{R}^2 \mid \theta_k \geq \theta^{\max}\right\}, \tag{2.26}$$

at iteration k contains a collection of points (θ, φ_μ) that have been previously explored and they define a region of forbidden points.

Then the backtracking line-search procedure attempts to find a step-size $\alpha_{k,l}$, given the current point x_k and search direction d_k^x by evaluating $x_k(\alpha_{k,l}) \coloneqq x_k + \alpha_{k,l} d_k^x$, and checking the following conditions:

- Switching

$$\nabla\varphi_{\mu_j}(x_k)^T d_k^x < 0, \text{ and } \alpha_{k,l}\left[-\nabla\varphi_{\mu_j}(x_k)^T d_k^x\right]^{s_\varphi} > \delta_l\left[\theta(x_k)\right]^{s_\theta}. \tag{2.27}$$

- Armijo Line search

$$\varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) + \eta_\varphi \alpha_{k,l} \nabla\varphi_{\mu_j}(x_k)^T d_k^x. \tag{2.28}$$

- Sufficient decrease

$$\theta(x_k(\alpha_{k,l})) \leq (1-\gamma_\theta)\theta(x_k), \text{ or } \varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) - \gamma_\varphi\theta(x_k). \tag{2.29}$$

These conditions require the following constants $s_\varphi, s_\theta, \delta_l \geq 1$ and $\eta_\varphi, \gamma_\theta, \gamma_\varphi \in (0,1)$. In IPOPT any trial point $(\theta(\alpha_{k,l}), \varphi_\mu(\alpha_{k,l})) \notin \mathcal{F}_k$ is a candidate for testing the conditions (2.27)—(2.29), otherwise the step-size is cut off ($\alpha_{k,l+1} = \frac{1}{2^l}\alpha_{k,l}$). Furthermore, after finding a point acceptable to the filter, if condition (2.28) or (2.29) are not satisfied, the filter for the next iteration is updated as follows as follows

$$\mathcal{F}_{k+1} \coloneqq \mathcal{F}_k \cup \left\{\theta(x_k(\alpha_{k,l})) \leq (1-\gamma_\theta)\theta(x_k), \text{ and } \varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) - \gamma_\varphi\theta(x_k)\right\}. \tag{2.30}$$

For points that are not in the filter, conditions (2.27)-(2.29) have to be checked in a specific succession. This can be summarized in Algorithm 2.2.

Data: Current iterate x_k and search direction d_k^k, maximum step size $\alpha_k^{\max}$, constants $s_\varphi, s_\theta, \delta_l \geq 1$ and $\eta_\varphi, \gamma_\theta, \gamma_\varphi \in (0,1)$

B.1 **Initialize search**. $\alpha_{k,0} \leftarrow \alpha_k^{\max}$, $l \leftarrow$, `status` $\leftarrow$ `NotAcceptable`, `augmentFilter` $\leftarrow$ `false` ;

while *`status` = `NotAcceptable`* **do**

B.2 **Compute new trial point**. $x(\alpha_{k,l}) \coloneqq x_k + \alpha_{k,l} d_k^x$;

B.3 **Check acceptability to the filter**

if $(\theta(\alpha_{k,l}), \varphi_\mu(\alpha_{k,l})) \in \mathcal{F}_k$ **then**

Reject step size; go to **Second-order Correction**, step B.5 ;

B.4 **Check sufficient decrease**

if *Switching Condition holds,* (2.27) **then**

if *Armijo Condition holds,* (2.28) **then**

accept step size; `status` $\leftarrow$ `Acceptable` ;

break;

else

if *Sufficient Decrease Condition holds,* (2.29) **then**

accept step size; `status` $\leftarrow$ `Acceptable` ;

`augmentFilter` $\leftarrow$ `true` ;

break;

B.5 **Second-order Correction**. Same as in [2] ;

B.6 **Choose new trial step-size**

$\alpha_{k,l+1} = \frac{1}{2^l}\alpha_{k,l}$;

$l \leftarrow l + 1$;

if $\alpha_{k,l} < \alpha_k^{\min}$ **then**

Feasibility Restoration. Same as in [2]

end

Algorithm 2.2: Filter line-search

2.4.2 IPOPT algorithm

Consider the residual norms for *primal*, *dual*, and *complementarity*

$$\begin{aligned} \text{du\_inf} &:= \left\| g(x) + A(x)\lambda - z \right\|_\infty, \quad \text{pr\_inf} := \left\| c(x) \right\|_\infty, \\ \text{cmpl\_inf}_{\mu_j} &:= \left\| XZe - \mu_j e = 0 \right\|_\infty, . \end{aligned} \tag{2.31}$$

The *overall* NLP error is computed using the scaled versions of the previous norms.

$$E_{\mu_j}(x, \lambda, z) := \max \left\{ \frac{\text{du\_inf}}{s_d}, \text{pr\_inf}, \frac{\text{cmpl\_inf}_{\mu_j}}{s_c} \right\}, \tag{2.32}$$

where the scaling factors s_d, s_c are defined below,

$$s_d = \max \left\{ s_{\max}, \frac{\|\lambda\|_1 + \|z\|_1}{m+n} \right\} / s_{\max}, \tag{2.33}$$

$$s_c = \max \left\{ s_{\max}, \frac{\|z\|_1}{n+m} \right\} / s_{\max}. \tag{2.34}$$

and $s_{\max} \geq 1$. Convergence for a particular barrier problem is achieved whenever the overall NLP error is below a factor of the barrier parameter, i.e.

$$E_{\mu_j}(x_k, \lambda_k, z_k) \leq \kappa_\epsilon \mu_j, \tag{2.35}$$

for a positive scalar κ_ϵ. Note that the convergence of the overall problem is checked for (2.35) with $\mu_j = 0$, i.e. $E_0(x_k, \lambda_k, z_k) \leq \kappa_\epsilon \epsilon_{\text{tol}}$. Therefore the overall IPOPT algorithm is given in Algorithm 2.3.

Data: Starting point (x_0, λ_0, z_0) with $x_0, z_0 > 0$; initial value for the barrier parameter $\mu_0 > 0$ and $\delta_w^{last} \leftarrow 0$; constants $\epsilon_{tol}, \kappa_\theta, \kappa_\epsilon > 0$, $s_\theta > 1$, $s_\varphi \geq 1$, $\gamma_\alpha \in (0, 1]$, $\gamma_\varphi, \gamma_\theta \in (0, 1)$, $\eta_\varphi \in (0, \frac{1}{2})$.

A.1 **Initialize.** Initialize the filter $\mathcal{F}_0$, and τ_0. Set iteration counters $j \leftarrow 0$, $k \leftarrow 0$.

`status` $\leftarrow$ `NotConverged`;

while *`status` = `NotConverged`* **do**

A.2 **Check convergence for the overall problem**

if $E_0(x_k, \lambda_k, z_k) \leq \epsilon_{tol}$ **then** `status` $\leftarrow$ `Converged`;

A.3 **Check convergence for the barrier problem**

if $E_{\mu_j}(x_k, \lambda_k, z_z) \leq \kappa_\epsilon \mu_j$ **then**

$\mu_{j+1} \leftarrow \max\left\{\frac{\epsilon_{\text{tol}}}{10}, \min\left\{\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right\}\right\}$; set $j \leftarrow j + 1$;

Re-initialize filter $\mathcal{F}_k$ with (2.26) ;

A.4 **Compute search direction**

Compute $\left(d_k^x, d_k^\lambda, d_k^z\right)$ from (2.24), using Algorithm 2.1 for some δ, δ_c ;

A.5 **Backtracking line search**

Compute step-size $\alpha_{k,l}$ using Algorithm 2.2 ;

A.6 **Accept the trial point**

$\alpha_k \leftarrow \alpha_{k,l}$;

Compute multiplier estimates (λ_{k+1}, z_{k+1})

A.7 **Augment filter**

if *`augmentFilter` = `true`* **then** Update the filter using (2.30);

else $\mathcal{F}_{k+1} := \mathcal{F}_k$;

$k \leftarrow k + 1$

end

Algorithm 2.3: IPOPT main algorithm

Remarks

Feasibility Restoration Phase. The feasibility restoration problem is stated as follows; find a feasible point which is closest to a vector x_R, and is given in the following problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & \|c(x)\|_1 + \frac{\zeta}{2}\|D_R(x - x_R)\|_2^2 \\ \text{subject to} \quad & x \geq 0, \end{aligned} \tag{2.36}$$

where $D_R \in \mathbb{R}^{n\times n}$ is a diagonal matrix constructed in terms of the $i-$th element of x_R. The feasibility restoration problem is important for the convergence of IPOPT because it acts as a work around situations in which the backtracking filter line-search generates step sizes that are bellow the minimum step size. This consideration is key for the convergence proofs of filter line-search [22]. The idea is to generate a point that is either not in the current filter $\mathcal{F}_k$, or is a local minimizer of some measure of the current infeasibility. In this sense, problem (2.36) is a good candidate for such task, though it is not the only possible strategy. In the event of the Restoration Phase being called; an equivalent form of problem (2.36) would be attempted to be solved for a reference vector x_R (which is typically the last iterate x_k, from which Restoration got called), with the same algorithm from IPOPT; that is Algorithm 2.3, for some value of the barrier parameter and (possibly relaxed) tolerances. At every iteration of the restoration phase the current iterate will be tested to verify that $\left(\theta^{\text{resto}}, \varphi^{\text{resto}}\right) \notin \mathcal{F}_k$; if this is not the case, then the iterate will accepted. Then, restoration will end, and the algorithm will revert itself to its normal form. Note that problem (2.36) needs to be formulated in terms of smooth functions (because the ℓ_1 norm is not differentiable at zero), this can be done with the introduction of vectors $p, n \in \mathbb{R}^m$ and the constraint $c(x) = p - n$ as follows,

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & (p+n)^T e + \frac{\zeta}{2}\|D_R(x - x_R)\|_2^2 \\ \text{subject to} \quad & c(x) - p + n = 0, \\ & x \geq 0; p, n \geq 0, \end{aligned} \tag{2.37}$$

where $e \in \mathbb{R}^m$ is a vector of all ones.

2.5 Parametric NLP sensitivity

In this section we are concerned with a problem that contains a *fixed* vector of parameters $p \in \mathbb{R}^{n_p}$. For example,

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x, p) \\ \text{subject to} \quad & c(x, p) = 0, \\ & x \geq 0, \end{aligned} \tag{2.38}$$

where the objective and constraints are defined as $f : \mathbb{R}^n \times \mathbb{R}^{n_p} \to \mathbb{R}$, and $c : \mathbb{R}^n \times \mathbb{R}^{n_p} \to \mathbb{R}^m$ respectively; and are considered to be twice continuous differentiable with bounded second derivatives. Such instances are typical in problems for which there is information that has to be determined *a priori*. Assuming it is possible to solve problem (2.38), we are interested in the properties of the solution $(x(p), \lambda(p), z(p))$ under perturbations of the vector p. Let $s \in \mathbb{R}^{2n+m}$ be the concatenated vector of primal-dual variables, $s^T := \left[x^T, \lambda^T, z^T\right]$. Fiacco [23] establishes the existence of a first-order differentable function $s : \mathcal{N}_p \to \mathcal{N}_s(s^*)$ for some neighborhood of an initial parameter p_0 such that $\mathcal{N}_p(p_0) \subset \mathbb{R}^{n_p}$, and a open set $\mathcal{N}_s \subset \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ containing s^*; by assuming first and second order conditions, and the differentiability of the objective and constraints.

Theorem 2. *For some p_0, if there is a point x^* for which the KKT theorem holds with some multipliers λ^* and z^* (from definition 6); SOSC (strict inequality version of definition 7), strict complementarity (definition 8), and the LICQ (definition 2) hold. Then*

- *For p_0, $x^* = x(p_0)$ is an isolated local minimizer and the multipliers λ^* and z^* are unique, in other words $s^* = s(p_0)$.*
- *In the neighborhood $\mathcal{N}_p(p_0)$, there exist a once continuously differentiable function $s(p)^T = \left[x(p)^T, \lambda(p)^T, z(p)^T\right]$ that satisfies the KKT and SOSC conditions, such that $x(p)$ and an locally unique minimum with multipliers $\lambda(p)$ and $z(p)$ for some $p \in \mathcal{N}_p(p_0)$*
- *The LICQ and strict complementarity hold for $x(p)$ for $p \in \mathcal{N}_p(p_0)$*

Proof. The proof is given in Theorem 2.1 [23]. □

The main result of this theorem is that it sets a framework for approximations of the solution under parametric perturbations. Under the conditions of theorem 2, consider a function that encapsulates the primal-dual equations of a local optimal solution $s^* = s(p_0)$ at p_0,

$$B(x^*, \lambda^*, z^*, p_0) \coloneqq \begin{bmatrix} g(x^*, p_0) - z^* + A(x^*, p_0)\lambda^* \\ c(x^*, p_0) \\ X^* Z^* e \end{bmatrix}, \tag{2.39}$$

since $B(s(p_0), p_0) = 0$; using the implicit function theorem yields the following result

$$\nabla_s B(s^*, p_0)^T \frac{ds^{*T}}{dp} + \nabla_p B(s^*, p_0)^T = 0. \tag{2.40}$$

A closer inspection of the terms in this equation reveals that the matrix $\nabla_s B(s^*, p_0)^T$ is similar to the one from (2.19), i.e. from the interior point primal-dual framework. I.e.

$$\nabla_s B(s^*, p_0)^T \coloneqq \begin{bmatrix} H(s^*, p_0) & A(x^*, p_0) & -I \\ A(x^*, p_0)^T & 0 & 0 \\ Z^* & 0 & X^* \end{bmatrix}, \tag{2.41}$$

additionally the remaining matrices are given as follows

$$\frac{ds^{*T}}{dp} \coloneqq \begin{bmatrix} \frac{dx^{*T}}{dp} \\ \frac{d\lambda^{*T}}{dp} \\ \frac{dz^{*T}}{dp} \end{bmatrix}, \quad \nabla_p B(s^*, p_0)^T \coloneqq \begin{bmatrix} \nabla_{xp}\mathcal{L}(s^*, p_0)^T \\ \nabla_p c(x^*, p_0)^T \\ 0 \end{bmatrix}. \tag{2.42}$$

This suggest that in order to find $\frac{ds^*}{dp}$, one must solve the linear system (2.40) with n_p right-hand-sides ($\nabla_p B(s^*, p_0)^T$). Furthermore, from theorem 2 it is possible to construct a Taylor's series approximation of $s(p)$ for some $p \in \mathcal{N}_p(p_0)$,

$$s(p) = s^* + \frac{ds^{*T}}{dp}(p - p_0) + o\left(\|p - p_0\|^2\right). \tag{2.43}$$

This result suggests that an approximation for optimal solutions can be done using second order information from the objective and constraint functions. However, this calculation requires the factorization of the matrix from (2.41) and n_p backsolves to assemble the first order derivative of s. Moreover, the set of assumptions of Theorem 2 will not hold at points which the active set changes; thus limiting the application of (2.43) to a neighborhood that is unknown a priori. In consequence, there exist weaker assumptions that account for such cases with the trade-off that only directional derivatives can be computed [24].

2.6 Reduced Hessians

Computing reduced Hessians is not a trivial task, because as described in Definition 9, it involves computing a null space basis of the active bound Jacobian. Nevertheless, it is possible to use optimality information to derive a convenient way of computing the reduced Hessian without having to compute a null space basis. Suppose a local solution has been found, and the conditions of Definition 9 hold, i.e. the LICQ and Strict Complementarity. Furthermore, suppose there are no active bounds; then it is possible to partition the variables into dependent and independent (plus bounded, i.e. variables which have an active bound), i.e. $x^{*T} = \left[x_D^T, x_I^T\right]$ such that $x_I \in \mathbb{R}^m$ and $x_D \in \mathbb{R}^{n-m}$ (for $m < n$). Thus the following partitions of Hessian ($H\left(x^*, \lambda^*\right) \in \mathbb{R}^{n \times n}$) and gradients of the constraint ($A\left(x^*\right) \in \mathbb{R}^{n \times m}$) matrices follows

$$A\left(x^*\right) \coloneqq \begin{bmatrix} A_D^T & A_I \end{bmatrix}, \quad H\left(x^*, \lambda^*\right) \coloneqq \begin{bmatrix} H_{DD} & H_{DI} \\ H_{ID} & H_{II} \end{bmatrix}, \tag{2.44}$$

where the blocks are sized accordingly to the partition of dependent (D) and independent (I) variables. Thus the KKT system can be assembled for some search direction

$(d_{x_D}, d_{x_I}, d_\lambda)$ and right-hand-side vectors $(r_{x_D}, r_{x_I}, r_\lambda)$;

$$\begin{bmatrix} H_{DD} & H_{DI} & A_D \\ H_{ID} & H_{II} & A_I \\ A_D^T & A_I^T & 0 \end{bmatrix} \begin{bmatrix} d_{x_D} \\ d_{x_I} \\ d_\lambda \end{bmatrix} = \begin{bmatrix} r_{x_D} \\ r_{x_I} \\ r_\lambda \end{bmatrix}, \tag{2.45}$$

then it can be shown that setting $r_{x_D} = 0$, $r_{x_I} = I \in \mathbb{R}^{(n-m)\times(n-m)}$, and $r_{x_\lambda} = 0$, will result in $d_{x_I} \coloneqq H_R^{-1}$; where the I is the identity matrix of the appropriate size and H_R is the reduced Hessian matrix.

To show this consider the definition of the reduced Hessian matrix $H_R = \mathcal{Z}\left(x^*\right)^T H\left(x^*, \lambda^*\right) \mathcal{Z}\left(x^*\right)$, where $\mathcal{Z}\left(x^*\right) \in \mathbb{R}^{n\times(n-m)}$ is a matrix whose columns form a basis for the null space of the constraint gradients, i.e. $A\left(x^*\right)^T \mathcal{Z}\left(x^*\right)\left(x^*\right) = 0$. Then, by setting the $\mathcal{Z}\left(x^*\right)$ matrix as follows [25],

$$\mathcal{Z}\left(x^*\right) \coloneqq \begin{bmatrix} -A_D^{-T} A_I^T \\ I \end{bmatrix}, \tag{2.46}$$

and using the definition of the reduced Hessian matrix, with (2.44),

$$\mathcal{Z}\left(x^*\right)^T H\left(x^*, \lambda^*\right) \mathcal{Z}\left(x^*\right) = A_I A_D^{-1} H_{DD} A_D^{-T} A_I - A_I A_D^{-1} H_{DI} - H_{ID} A_D^{-1} A_I + H_{II}. \tag{2.47}$$

Furthermore, pivoting on the A_D blocks of the matrix of (2.45), results in the following diagonal matrix

$$\begin{bmatrix} 0 & 0 & I \\ 0 & A_I A_D^{-1} H_{DD} A_D^{-T} A_I - A_I A_D^{-1} H_{DI} - H_{ID} A_D^{-1} A_I + H_{II} & 0 \\ I & 0 & 0 \end{bmatrix}. \tag{2.48}$$

And, by setting $r_{x_D} = 0$, $r_{x_I} = I$, and $r_{x_\lambda} = 0$, it can be seen that the second row contains the desired result.

This property is useful as a post-solution analysis, because it enables a relatively low-cost computation of H_R; as long as the partition $x^{*T} = \left[x_D^T, x_I^T\right]$ is consistent with the vector sizes. Finally, note that if the augmented primal-dual Hessian matrix ($W_k \coloneqq H_k + \Sigma_k$) is used instead of H_k, which is typically the case, then the reduced Hessian will contain *active bound* information.

2.7 k_aug

Both of the last two previous sections discuss post-optimality operations, that imply the factorization on sparse matrices. Moreover, it can be seen the similarities between these systems and the systems that are solved within IPOPT. Part of the performance expected within IPOPT comes from the available sparse-symmetric-linear-solvers. As mentioned in earlier sections these offer additional information with the factorization of the matrices, i.e. the inertia, which is useful for determining the LICQ and SOSC properties of the system. Among other properties, some of them are able to use concurrency to take advantage of multi-core architectures, and thus reduce the amount of time for the factorizations, e.g. MA97, and PARDISO [26]. However, one key difference from the systems used within IPOPT is the presence of multiple right-hand-sides in the linear systems; for instance the system of (2.44), with $(n-m)$ right-hand-sides. In typical linear-algebra fashion, one need only factorize once and re-use the factorization to find the corresponding solution vectors. The aggregate of all these capabilities is reflected into the implementation of `k_aug`, which is shown in Figure 2.1.

Similarly as in the case of IPOPT, The linear systems within `k_aug` are symmetric. For instance, in the case of (2.41) and (2.42); by pivoting on the the block Z^* of the matrix $\nabla_s B\left(s^*, p_0\right)$, the last row can be removed. This leads into the matrices

$$K \coloneqq \begin{bmatrix} H\left(s^*, p_0\right) + \left(X^*\right)^{-1} Z^* & A\left(x^*, p_0\right) \\ A\left(x^*, p_0\right)^T & 0 \end{bmatrix}, \tag{2.49}$$

$$S \coloneqq \begin{bmatrix} \frac{dx^*}{dp}^T \\ \frac{d\lambda^*}{dp}^T \end{bmatrix}, \quad R \coloneqq \begin{bmatrix} \nabla_{xp}\mathcal{L}\left(s^*, p_0\right)^T \\ \nabla_p c\left(x^*, p_0\right)^T \end{bmatrix}; \quad \text{and} \quad \frac{dz^*}{dp}^T = -\left(X^*\right)^{-1} Z^* \frac{dx^*}{dp}^T. \tag{2.50}$$

And, thus the system (2.40) becomes

$$KS = -R, \tag{2.51}$$

where the matrices have the following sizes, $K \in \mathbb{R}^{(n+m)\times(n+m)}$, $S \in \mathbb{R}^{(n+m)\times n_p}$, and $R \in \mathbb{R}^{(n+m)\times n_p}$. It follows that after solving for the matrix S, it is possible to update the

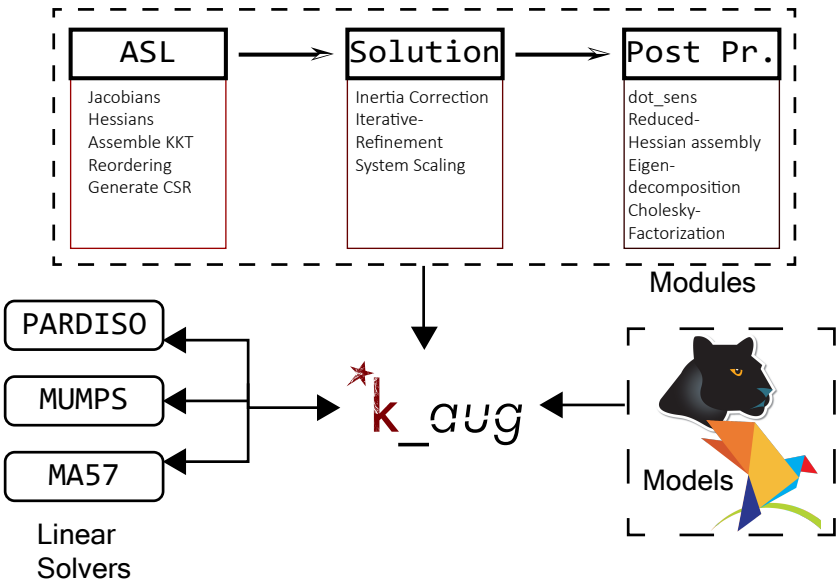

Figure 2.1: k_aug's elements

vector of primal and equality multipliers $\tilde{s} \in \mathbb{R}^{(n+m)}$ such that $\tilde{s}^T := \left[x^T, \lambda^T\right]$, for small parametric perturbations $\Delta p := p - p_0$,

$$\tilde{s}\left(p\right) = \tilde{s}^* + S\Delta p, \tag{2.52}$$

where $\tilde{s}^* := \tilde{s}\left(p_0\right)$, i.e. the optimal solution for p_0.

Moreover, there might be instances in which only a subset of the solution is required. We define the vectors $r\left(p\right) = E^T\tilde{s}\left(p\right)$ and $r^* = E^T\tilde{s}^*$ where $r\left(p\right), r^* \in \mathbb{R}^{n_r}$; and the matrix $E \in \mathbb{R}^{(n+m)\times n_r}$, which consists of concatenated unit vectors with 1's at the positions of $r\left(p\right)$ in $\tilde{s}\left(p\right)$, and zeros otherwise. Depending on the values of n_p and n_r we can exploit the following update cases:

- For $n_p \leq n_r$, application of (2.52) leads to

$$r\left(p\right) = r^* + E^T S\Delta p, \tag{2.53}$$

  which requires n_p backsolves.

- For $n_p > n_r$ an alternative update can be performed. Multiplying (2.52) on both sides by E^T the update becomes: $r(p) = r^* - (E^T K^{-1}) R\Delta p$. Defining $S_r \in \mathbb{R}^{(n+m)\times n_r}$ and symmetry of K allows the solution of the system:

$$KS_r = E, \tag{2.54}$$

and the update of $r(p)$ can be written as:

$$r(p) = r^* - S_r^T R\Delta p, \tag{2.55}$$

which now requires n_r backsolves.

In our implementation `k_aug` computes both S and S_r. To obtain derivative information, `k_aug` uses the AMPL Solver Library (ASL) [27], and a sparse-symmetric-indefinite linear solver, like MUMPS 5.1.2 [28] for the factorization. After computing the sensitivity matrix and the realization of Δp, `dot_sens` an auxiliary module of `k_aug` , performs a computation similar to (2.53) or (2.55), as previously described.

Finally, if the set of active bounds $\mathcal{A}(x^*)$ changes with the perturbation Δp for problem (2.52), then this sensitivity approach needs to be modified. One option is the path-following approach developed in [18], which allows for more general sensitivity perturbations with changing active sets and even weaker regularity conditions. However, this requires the solution of sequences of Quadratic and Linear Programs (QPs and LPs) to compute directional derivatives and active sets. The current implementation of `k_aug` does not implement a strategy for changing active sets; nevertheless future versions of it will take into account this situation. Finally, we note that `k_aug` is able to compute Reduced Hessians in a similar way to the sensitivity matrices. This is done with a variation of the symmetric version of system (2.45).

Chapter 3

ℓ_1-exact Penalty-Barrier Strategy for the Failure of Constraint Qualifications

3.1 Introduction

The state of art tools to solve nonlinear programming problems are mature enough, so that they can handle large-scale instances with reasonable performance. In order to achieve this, these solvers require several assumptions on the properties of the problem, particularly on the consistency of local linearizations (as explained in Section (2.2)). However, it is often difficult to verify beforehand that the problem satisfies these assumptions. Moreover it is possible to reach regions in which small perturbations lead to violations of them [21].

Examples of problems with inconsistent linearizations include problems with bilinear terms, like gasoline blending and network-flow [29]; Mathematical Programs with Complementarity Constraints (MPCCs) [6] and discretized forms of high-index Differential-Algebraic Equation (DAE) systems [30].

Thus, several NLP algorithms have been designed to work under weaker assumptions that attempt to overcome these degeneracies. [10], presented a method that works by computing normal steps towards feasibility by solving a trust-region problem, and then a tangential steps from solving a perturbed primal-dual system with an iterative linear solver. At the limit, even if the linearizations of the constraints are inconsistent, the algorithm might still converge to local stationary points. [31], incorporated a SQP strategy with exact penalty of the constraint violation and a line-search. Provided that the Hessian of the Lagrangian is positive definite, this strategy has inherent regularizing properties. Never-

theless, the algorithm must allocate considerable effort to finding appropriate values of the penalty parameter.

In terms of the typical NLP solvers, IPOPT (Section 2.4)is one of the most competitive solvers available for large-scale optimization. At its core it employs a filter line-search with an interior-point method to converge to the solution. Regarding the loss of regularity, IPOPT uses inertia controlling mechanisms (Algorithm 2.1), which will detect the regions of degeneracy and attempt to regularize the problem. Nevertheless, these mechanisms compete with several factors of the problem that limit their success, like ill-conditioning and numerical instabilities. It has been shown by [8] and [9] that the degeneracies can be removed directly at the linear algebra level by analyzing the pivot sequence and perturbing or removing null rows of the Jacobian block of the KKT matrix. However, the effectiveness of this strategy are linked to the tuning parameters of the sparse-linear algebra libraries.

In this chapter we present a strategy that combines exact penalty and the filter line-search and barrier algorithms within IPOPT to solve problems with inconsistent constraint linearizations. The fundamental penalty-barrier problem will be introduced alongside a strategy to adjust the value of the penalty parameter. Finally, numerical results for degenerate problems and Mathematical Programs with Complementarity Constraints (MPCCs) will be presented.

3.2 Background of the ℓ_1-exact penalty-barrier method

The interior-point method from section 2.4 is effective in problems that attain *regularity* conditions, though it is common to commit modelling errors or formulate problems with pathological constraint inconsistencies. Then, the theoretical convergence properties [22] cannot be guaranteed. In particular, it is required to make assumptions on the consistency of the linearizations of the set of constraints. For instance the linear Independence Constraint Qualification (LICQ from Definition 2) or the Mangasarian-Fromovitz Constraint Qualification (MFCQ from Definition 5).

Inconsistencies of the linearizations that are localized to the neighborhood of certain points are attempted to be regularized with modifications to the search directions computed within IPOPT (this is Algorithm 2.1). However, it is possible that either the modifications are not successful, or a drastic performance loss is attained; therefore, it is desirable to find alternative strategies to overcome the inconsistencies on the set of constraint of problems.

In the spirit of regularizing the problem, it is desirable to formulate the problem such that the resulting problem requires weaker assumptions (e.g. MFCQ) and is steps towards the solution can be readily computed, even at points when (2.1) is degenerate. To start, consider penalizing the ℓ_1-norm of the constraints in the barrier objective function, it follows that this results in an unconstrained penalty-barrier problem,

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) - \mu_j \sum_i^n \ln x^{(i)} + \rho \left\| c(x) \right\|_1 \tag{3.1}$$

where $\rho \in \mathbb{R}^1_{\geq 0}$ is the penalty parameter. Note that even though the ℓ_1-norm has been used, it is possible to formulate this problem with a number of nonsmooth ℓ_p norms. Nevertheless, problem (3.1) can be reformulated as an equivalent smooth constrained problem with the addition of two vectors of penalty-variables p and $n \in \mathbb{R}^m$, that is,

$$\begin{aligned} \underset{x\in\mathbb{R}^n;\, p,n\in\mathbb{R}^m}{\text{minimize}} \quad & f(x) - \mu_j \sum_i^n \ln x^{(i)} + \rho (p+n)^T e \\ \text{subject to} \quad & c(x) - p + n = 0, \\ & p, n \geq 0, \end{aligned} \tag{3.2}$$

where e denotes a vector of all ones of the appropriate dimension. The resulting problem is feasible and satisfies the MFCQ at points in which $p, n > 0$; however it has a higher dimensionality because of the addition of the penalty-variables. Moreover, there is the issue of the value of ρ, which in practice is critical for the convergence of (3.2) to solutions of (2.1).

Thus it might be more suitable for problems where the constraint linearizations are not consistent. However, the value of ρ is critical, because under the MFCQ and the SOSC

(Definition 7) , if $\rho \geq \rho^*$, then a stationary point of (3.2) is also a KKT point of (2.1). Note that the value of ρ^* is not known beforehand; furthermore a large initial guess value of ρ might result in numerical difficulties of the solution strategy, so it might be desireable to gradually increase its value until $\rho \geq \rho^*$. It should be noted that the solution of this problem will require the bounds of the penalty variables to be moved in an additional barrier term in the objective.

Remarks

Linear damping terms for penalties. In the case when variables have have general upper and lower bounds (i.e. $x_{(i)}^{lb} \leq x^{(i)} \leq x_{(i)}^{ub}$, $x_{(i)}^{lb} \in [-\infty, \infty)$, $x_{(i)}^{ub} \in (-\infty, \infty]$, $x_{(i)}^{lb} \leq x_{(i)}^{ub}$), the barrier in the objective will contain the non-negative slacks $\left(x^{(i)} - x_{(i)}^{lb}\right)$ and $\left(x_{(i)}^{ub} - x^{(i)}\right)$, then the logarithmic barrier objective function is written with these terms instead, i.e.

$$\varphi_{\mu_j}(x) := f(x) - \mu_j \sum_{i \in I_{lb}} \ln\left(x^{(i)} - x_{(i)}^{lb}\right) - \mu_j \sum_{i \in I_{ub}} \ln\left(x_{(i)}^{ub} - x^{(i)}\right) + \text{dmp}_{\mu_j}(x),$$

where the term $\text{dmp}_{\mu_j}(x)$ corresponds to the linear damping term. For variables with a single bound (as opposed of having both bounds), the damping term is required, because the barrier terms might become unbounded. These damping terms are defined in terms of the sets of indices of variables with lower and upper bounds I_{lb}, and I_{ub} respectively. Then, the linear damping term is defined as follows

$$\text{dmp}_{\mu_j}(x) := \kappa_d \mu_j \sum_{I_{lb} \setminus I_{ub}} \left(x^{(i)} - x_i^{lb}\right) + \kappa_d \mu_j \sum_{I_{ub} \setminus I_{lb}} \left(x_{(i)}^{ub} - x^{(i)}\right), \tag{3.3}$$

for some positive nonnegative parameter κ_d. For the penalty formulation, clearly the indices of variables p, n are in the set $I_{lb} \setminus I_{ub}$, since they do not have upper bounds. Thus the linear damping term for p, n is required, and it is also multiplied by $1/\rho$ if the inverse mode is used.

3.3 Solution of the penalty-barrier problems

In the *direct* form of the penalty-barrier problem in (3.2), new barrier (and damping) terms need to be created in the objective to take into account the bounds of the penalty-variables. As a result, the set of primal-dual equations include additional penalty-variable multipliers $z_p, z_m \in \mathbb{R}^m$, i.e.,

$$\begin{bmatrix} g(x) - z + A(x)\lambda \\ \rho\, e - z_p - \lambda \\ \rho\, e - z_n + \lambda \\ c(x) - p + n \\ XZe - \mu_j e \\ PZ_p e - \mu_j e \\ NZ_n e - \mu_j e \end{bmatrix} = 0, \tag{3.4}$$

where the additional diagonal matrices are defined for the penalty-variables $P := \text{diag}\,(p)$, $N := \text{diag}\,(n)$, and penalty-bound multipliers $Z_p := \text{diag}\,(z_p)$, $Z_n := \text{diag}\,(z_n)$; with $p, n, z_p, z_n \geq 0$. Moreover, note that the notation of section 2.3 is also being used here, i.e. $g(x) \coloneqq \nabla f(x)$ and $\nabla c(x) \coloneqq A(x)$. By applying Newton's method, the linearization of (3.4) yields the full-space primal-dual linear system,

$$\begin{bmatrix} H_k & 0 & 0 & A_k & -I & 0 & 0 \\ 0 & 0 & 0 & -I & 0 & -I & 0 \\ 0 & 0 & 0 & I & 0 & 0 & -I \\ A_k^T & -I & I & 0 & 0 & 0 & 0 \\ Z_k & 0 & 0 & 0 & X_k & 0 & 0 \\ 0 & Z_{p,k} & 0 & 0 & 0 & P_k & 0 \\ 0 & 0 & Z_{n,k} & 0 & 0 & 0 & N_k \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^p \\ d_k^n \\ d_k^\lambda \\ d_k^z \\ d_k^{z_p} \\ d_k^{z_n} \end{bmatrix} = - \begin{bmatrix} g_k - z_k + A_k\lambda_k \\ \rho e - z_{p,k} - \lambda_k \\ \rho e - z_{n,k} + \lambda_k \\ c_k - p_k + n_k \\ X_k Z_k e - \mu_j e \\ P_k Z_{p,k} e - \mu_j e \\ N_k Z_{n,k} e - \mu_j e \end{bmatrix}. \tag{3.5}$$

The resulting linear system is larger than the system of (2.20), and its conditioning depends on the value of ρ indirectly through the values of the multipliers of the penalty-variables. By performing pivoting on the bottom right blocks and then on the rows that correspond

to the relationship between the penalty parameter and the multipliers, it is possible to decouple some of the rows and create a symmetric matrix that is similar to the matrix in (2.21). For this let the primal-dual term matrices for the penalty-variables be $\Sigma_{p,k} := P_k^{-1} Z_{p,k}$ and $\Sigma_{n,k} := N_k^{-1} Z_{n,k}$, then the augmented primal-dual penalty linear system is,

$$\begin{bmatrix} W_k & A_k \\ A_k^T & -\Sigma_{p,k}^{-1} - \Sigma_{n,k}^{-1} \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix} = - \begin{bmatrix} g_k - \mu_j X_k^{-1} e + A_k \lambda_k \\ c_k - p_k + n_k + Z_{p,k}^{-1} \left[\rho P_k - \mu_j e\right] + Z_{n,k}^{-1} \left[-\rho N_k + \mu_j e\right] - \lambda_k \left[\Sigma_{p,k} + \Sigma_{n,k}\right] \end{bmatrix}, \quad (3.6)$$

and the remaining search directions are defined in terms of $\lambda_k^+ := \lambda_k + d_k^\lambda$ as follows,

$$\begin{aligned} d_k^p = Z_{p,k}^{-1} \left[\mu_j e + P_k \lambda_k^+ - \rho p_k\right], \quad d_k^n = Z_{n,k}^{-1} \left[\mu_j e - N_k \lambda_k^+ - \rho n_k\right], \\ d_k^{z_p} = \mu_j P_k^{-1} e - z_{p,k} - \Sigma_{p,k} d_k^p, \quad d_k^{z_n} = \mu_j N_k^{-1} e - z_{n,k} - \Sigma_{n,k} d_k^n. \end{aligned} \quad (3.7)$$

The linear system of (3.6) is convenient due to its similar nonzero structure to (2.21). Furthermore, it can be inferred that if the MFCQ hold for problem (2.1), as $p_k, n_k \to 0$, (3.6) reduces itself to the same terms of system (2.21), on the other hand, if $p_k, n_k > 0$, the lower-right block of the matrix has a similar effect as the δ_c term in system (2.21). Thus, steps on (3.6) can be taken as long as the problem remains infeasible, and regularization (δ) would only be needed if the Newton step is in a direction of negative curvature.

Nevertheless, as feasible points are approached, the multipliers values will grow at the same rate as the values of the penalty parameter; and there is the possibility of having an unbounded penalty parameter if the MFCQ does not hold. Then every other quantity that is either directly multiplied by the penalty parameter or involves the multipliers will cause numerical difficulties. Therefore, an alternative strategy is to divide the penalty-barrier problem by ρ with the expectation of attaining better numerical properties. In other

words,

$$\begin{aligned} \underset{x\in\mathbb{R}^n;\, p,n\in\mathbb{R}^m}{\text{minimize}} \quad & \frac{1}{\rho}\left[f(x)-\mu_j\sum_i^n \ln x^{(i)}\right]+(p+n)^T e \\ \text{subject to} \quad & c(x)-p+n=0, \\ & p,n\geq 0. \end{aligned} \tag{3.8}$$

This form of the penalty-barrier problem was designated as *inverse*. The idea of this form is to reduce the effects of the link between multipliers and penalty parameter. At the limit if the penalty becomes unbounded, IPOPT will solve for a stationary point (not necessarily feasible) of problem (2.1) without bounds on x. Otherwise if the problem converges to a stationary point for which the penalty-variables are zero and the penalty-parameter is bounded (KKT point of (2.1)). This effect can be seen in the primal-dual system of problem,

$$\begin{bmatrix} \rho^{-1}\left[g(x)-z\right]+A\lambda \\ e-z_p-\lambda \\ e-z_n+\lambda \\ c(x)-p+n \\ XZe-\mu_j e \\ PZ_p e-\mu_j e \\ NZ_n e-\mu_j e \end{bmatrix}=0, \tag{3.9}$$

where the bracketed term in the first equation vanishes as $\rho\to\infty$. The augmented linear system will also experience a similar effects on the KKT matrix and right-hand-side. For the *inverse* form of the penalty-barrier problem, let the augmented Hessian matrix be $\overline{W}_k := \rho^{-1}\left[\nabla^2_{xx} f(x_k)+\Sigma_k\right]+\sum_i^n \nabla^2_{xx} c_i(x_k)\,\lambda_k^{(i)}$, so the Newton step on the set of primal-dual equations is given as follows,

$$\begin{aligned} &\begin{bmatrix} \overline{W}_{\rho,k} & A_k \\ A_k^T & -\Sigma_{p,k}^{-1}-\Sigma_{n,k}^{-1} \end{bmatrix}\begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix}= \\ &\quad -\begin{bmatrix} \rho^{-1}\left[g_k-\mu_j X_k^{-1}e\right]+A_k\lambda_k \\ c_k-p_k+n_k+Z_{p,k}^{-1}\left[P_k-\mu_j e\right]+Z_{n,k}^{-1}\left[-N_k+\mu_j e\right]-\lambda_k\left[\Sigma_{p,k}+\Sigma_{n,k}\right] \end{bmatrix}, \end{aligned} \tag{3.10}$$

in which the structure remains almost the same as (3.6); however $\overline{W}_k = \sum_i^n \nabla^2_{xx} c_i\left(x_k\right) \lambda_k^{(i)}$ as $\rho \to \infty$.

$$\begin{aligned} d_k^p &= Z_{p,k}^{-1}\left[\mu_j e + P_k \lambda_k^+ - p_k\right], \quad d_k^n = Z_{n,k}^{-1}\left[\mu_j e - N_k \lambda_k^+ - n_k\right], \\ d_k^{z_p} &= \mu_j P_k^{-1} e - z_{p,k} - \Sigma_{p,k} d_k^p, \quad d_k^{z_n} = \mu_j N_k^{-1} e - z_{n,k} - \Sigma_{n,k} d_k^n, \end{aligned} \tag{3.11}$$

and

$$d_k^z = \mu_j X_k^{-1} e - z_k - \Sigma_k d_k^x. \tag{3.12}$$

Remarks

LICQ of penalty-barrier problem. Consider a point x for which $c\left(x\right) \neq 0$ (i.e. $p, n > 0$), then the LICQ for problem (3.2)(or (3.8)) holds. To see this, it can be noted that the gradients of the constraints of (3.2)(or (3.8)) is given as follows

$$\begin{bmatrix} A\left(x\right) & E_{\mathcal{A}(x)} \\ -I & 0 \\ I & 0 \end{bmatrix}. \tag{3.13}$$

In this matrix $E_{\mathcal{A}(x)}$ is always full-rank, and even if $A\left(x\right)$ is rank deficient, the whole matrix remains full column rank.

Inertia of penalty-barrier KKT matrices. Despite the linear independence of the gradients of the constraints in the penalty-barrier problems, it might be the case in which the inertia of the associated KKT matrices is not *correct* (i.e. $(n, m, 0)$). Thus, the parameters $\delta, \delta_c \in \mathbb{R}_{\geq 0}$ are still necessary to perturb the KKT matrix. Moreover, this perturbation is done in the same way of the inertia correction algorithm 2.1.

3.4 Penalty update

The penalty parameter ρ plays a critical role in this strategy as it characterizes a solution and it defines the search direction. There will be a compromise between the quality of the

solution and the numerical performance of the algorithm, so an appropriate update rule must be selected. On the other hand, it is difficult to develop generalized update rules for all kinds of problems, so a combination between heuristics and analytical rules will be used.

The methodology of [32] was adapted to the form of the computed search directions in this work. First, at step k consider a linearized model of the infeasibility of the original problem,

$$m_k\left(d^x\right) := \left\|A_k^T d^x + c_k\right\|_1, \tag{3.14}$$

the strategy is based on selecting a value of the penalty parameter so the predicted reduction of a quadratic model of the penalty-barrier objective function for a search direction d^x, is commensurate to the current norm of the infeasibility. For this the quadratic model of the penalty-barrier objective function is given by,

$$q_{\rho,k}\left(d^x\right) := \varphi_{\mu_j}\left(x_k\right) + \nabla\varphi_{\mu_j}\left(x_k\right)^T d^x + d^{xT} W_k d^x + \rho\, m_k\left(d^x\right), \tag{3.15}$$

so that the predicted reduction quantified with the quadratic model function in terms of a search direction d_k^x, for a particular search direction, i.e. $\text{pred}_{\rho,k}\left(d\right) := q_{\rho,k}\left(0\right) - q_{\rho,k}\left(d\right)$. Thus, in order for a penalty parameter to be acceptable, the reduction has to be proportional to the current infeasibility; in other words,

$$\text{pred}_{\rho,k}\left(d_k^x\right) \geq \rho\kappa_\rho \left\|c_k\right\|_1, \tag{3.16}$$

for some scalar $\kappa_\rho \in (0,1)$. This rule can be combined with the relations for the linearized KKT conditions previously described. In particular, using the fourth row of (3.5) yields a lower bound on the penalty parameter,

$$\rho \geq \frac{\nabla\varphi\left(x_k\right)^T d_k^x + \dfrac{1}{2} d_k^{xT} W_k d_k^x}{\left(1-\kappa_\rho\right)\left\|c_k\right\|_1 - \left(p_k^+ + n_k^+\right)^T e}, \tag{3.17}$$

with the vectors $p_k^+ = p_k + d_k^p$ and $n_k^+ = n_k + d_k^n$. Given this result, an update of the penalty

parameter can be done by checking if $\rho \geq \rho_{\text{trial}}$ and setting $\rho = \rho_{\text{trial}} + \epsilon_\rho$ otherwise, where

$$\rho_{\text{trial}} := \frac{\nabla\varphi\left(x_k\right)^T d_k^x + \frac{\gamma_\rho}{2} d_k^{xT} W_k d_k^x}{\left(1-\kappa_\rho\right)\left\|c_k\right\| - \left(p_k^+ + n_k^+\right)^T e}. \tag{3.18}$$

Note that the parameter $\gamma_\rho \in \{0,1\}$ was added so that whenever $d_k^{xT} W_k d_k^x < 0$, $\gamma_\rho = 0$ otherwise $\gamma_\rho = 1$.

This rule defines the update of the penalty parameter based on the current local information. However, it is possible that for some problems the rate at which the penalty is updated is not large enough. Therefore, an additional heuristic was imposed in order to improve the rate of growth of the penalty parameter; and this is to *double* its value if there is not sufficient progress towards feasibility measured by using the denominator of equation (3.17),

$$\rho = \begin{cases} 2\,\rho & \text{if} \quad \left(1-\kappa_\rho\right)\left\|c_k\right\| - \left(p_k^+ + n_k^+\right)^T e < 0 \\ & \text{unchanged} \quad \text{otherwise} \end{cases}. \tag{3.19}$$

This update will be checked whenever (3.18) is satisfied. Moreover, it was found that it is desirable to limit this kind of update to certain range of values of the penalty parameter to ensure numerical stability of the factorizations. Therefore this particular heuristic is only performed if $\rho \leq \overline{\rho}$, which is determined by the user.

3.5 ℓ_1-IPOPT Phase algorithm

The ℓ_1-penalty strategies were implemented as an alternative to the *feasibility restoration* algorithm of IPOPT. To enable this, it was required to change some of the modules within IPOPT, mostly to include the effect the penalty in several of the quantities required during the solution algorithm. Moreover, several options were included to change the behaviour of the penalty updates.

In this section we discuss some of the highlights of the changes within the IPOPT framework.

3.5.1 Convergence test

The *overall* NLP error in IPOPT is given in terms of the infinity norms of the residuals of dual infeasibility (stationarity), primal infeasibility, and complementarity. For the ℓ_1-penalty-barrier function; the residuals also contain the penalty-variables and their respective multipliers, i.e.

$$\begin{aligned} \text{du\_inf} &:= \left\| \begin{bmatrix} g(x) + A(x)\lambda - z \\ \rho - e - z_p \\ \rho + e - z_n \end{bmatrix} \right\|_\infty, \quad \text{pr\_inf} := \|c(x) - p + n\|_\infty, \\ \text{cmpl\_inf}_{\mu_j} &:= \left\| \begin{bmatrix} XZe - \mu_j e = 0 \\ PZ_p e - \mu_j e = 0 \\ NZ_n e - \mu_j e = 0 \end{bmatrix} \right\|_\infty, \end{aligned} \tag{3.20}$$

then the overall NLP error is based on the maximum value of scaled versions of these quantities,

$$E_{\mu_j}(x, \lambda, z, p, z_p, n, z_n) := \max\left\{ \frac{\text{du\_inf}}{s_d}, \text{pr\_inf}, \frac{\text{cmpl\_inf}_{\mu_j}}{s_c} \right\}, \tag{3.21}$$

and the scaling factors s_d, s_c are defined below,

$$s_d = \max\left\{ s_{\max}, \frac{\|\lambda\|_1 + \|z\|_1 + \|z_p\|_1 + \|z_n\|_1}{3m + n} \right\} / s_{\max}, \tag{3.22}$$

$$s_c = \max\left\{ s_{\max}, \frac{\|z\|_1 + \|z_p\|_1 + \|z_n\|}{n + 2m} \right\} / s_{\max}. \tag{3.23}$$

for some $s_{\max} \geq 1$. With these quantities, convergence for a particular barrier problem is attained whenever the overall NLP error is below a factor of the barrier parameter, in other words

$$E_{\mu_j}(x_k, \lambda_k, z_k, p_k, z_{p,k}, n_k, z_{n,k}) \leq \kappa_\epsilon \mu_j, \tag{3.24}$$

where κ_ϵ is a positive constant. If such test is true, then the barrier parameter must be

decreased. Moreover, to determine convergence of the overall NLP problem, the test $E_0\left(x_k, \lambda_k, z_k, p_k, z_{p,k}, n_k, z_{n,k}\right) \leq \epsilon_{\text{tol}}$ has to be executed.

3.5.2 Filter line-search

As mentioned in the section 2.4.1 Newton based solvers attain fast convergence at the neighborhood of the solution. However, starting in these neighborhoods is typically not the case, thus an algorithm like the filter line-search is required. In this algorithm, the quality of the step sizes is determined by evaluating the properties of linearizations of the barrier function, and its acceptability to the filter set. It should be noted that an alternative strategy to this is using a *merit function*. This scheme was used commonly in several non-linear optimization strategies, and attempts to achieve a measure of combined optimality and feasibility. Moreover, the typical merit function resembles the ℓ_1-penalty-barrier function from this work.

In the sense of the interior point method used here, the bounds of the penalty-variables p and n need to be part of an additional barrier term, thus this leads to the *strict*[1] ℓ_1-penalty-barrier function, i.e.

$$\Phi^{\rho}_{\mu_j}\left(x, p, n\right) := \left[f\left(x\right) - \mu_j \sum_i^n \ln x^{(i)}\right] + \rho\left(p+n\right)^T e - \mu_j \sum_i^m \left(\ln p^{(i)} + \ln n^{(i)}\right). \quad (3.25)$$

Moreover, throughout the search the equality constraints in (3.2) are not necessarily equal to zero. Thus, the measure of the penalty-infeasibility can be given as

$$\Theta\left(x, p, n\right) := \left\|c\left(x\right) + p - n\right\| \quad (3.26)$$

Thus, the filter for the ℓ_1-penalty-barrier phase is defined in terms of these quantities,

$$\mathcal{F}_k \subseteq \left\{\left(\Theta_k, \Phi^{\rho}_{\mu_j,k}\right) \in \mathbb{R}^2 \mid \Theta_k \geq \Theta^{\max}\right\}. \quad (3.27)$$

With this definition and using definitions of α_k^{max} for the variables p and n as in (2.25), the Filter line-search Algorithm 2.2 remains unchanged; except that now it depends on

[1] *strict* in the sense of having the logarithmic barrier terms for p and n

the value of ρ. This in turn implies that in the same way as the *barrier* parameter update (μ_j), the filter need to be reset whenever a new value of ρ is assumed. This is necessary to maintain the convergence properties of the Filter line-search as discussed in [22].

3.5.3 ℓ_1-exact penalty-barrier IPOPT algorithm

The new algorithm adds steps for the computations of the new penalty parameter in step ℓA.5 and the subsequent update ℓA.9. Also; whenever the penalty parameter changes, it also resets the filter.

Data: Starting point $(x_0, \lambda_0, z_0, p_0, z_{p,0}, n_0, z_{n,0})$ with $x_0, z_0, p_0, z_{p,0}, n_0, z_{n,0} > 0$;

$\mu_0 > 0$, $\rho_0 > 0$, and $\delta_w^{last} \leftarrow 0$; same constants as Alg. 2.3, $\kappa_\rho, \overline{\rho} > 0$

ℓA.1 **Initialize.** Initialize the filter $\mathcal{F}_0$, and τ_0. Set iteration counters $j \leftarrow 0$, $k \leftarrow 0$.

`status` $\leftarrow$ `NotConverged` ;

while *`status` = `NotConverged`* **do**

ℓA.2 **Check convergence for the overall problem**

if $E_0\left(x_k, \lambda_k, z_k, p_k, z_{p,k}, n_k, z_{n,k}\right) \leq \epsilon_{tol}$ **then** `status` $\leftarrow$ `Converged`;

ℓA.3 **Check convergence for the barrier problem**

if $E_{\mu_j}\left(x_k, \lambda_k, z_k, p_k, z_{p,k}, n_k, z_{n,k}\right) \leq \kappa_\epsilon \mu_j$ **then**

Update μ_{j+1}; set $j \leftarrow j+1$; Re-initialize filter $\mathcal{F}_k$ with (3.27) ;

ℓA.4 **Compute search direction**

Compute $\left(d_k^x, d_k^\lambda, d_k^z, d_k^p, d_k^{z_p}, d_k^n, d_k^{z_n}\right)$ from (3.6) or (3.10) accordingly, using Algorithm 2.1 for some δ, δ_c ;

ℓA.5 **Compute penalty parameter trial.** I.e. ρ_{trial} using (3.18) ;

ℓA.6 **Backtracking line search**

Compute step-size $\alpha_{k,l}$ using Algorithm 2.2 ;

ℓA.7 **Accept the trial point**

$\alpha_k \leftarrow \alpha_{k,l}$; Compute multiplier estimates $(\lambda_{k+1}, z_{k+1}, z_{p,k+1}, z_{n,k+1})$

ℓA.8 **Augment filter**

if *`augmentFilter` = `true`* **then** Update the filter using (2.30) (With p, n quantities);

else $\mathcal{F}_{k+1} \coloneqq \mathcal{F}_k$;

ℓA.9 **Update the penalty parameter**

if $\rho \leq \rho_{trial}$ **then** $\rho \leftarrow \rho_{\text{trial}}$; Re-initialize $\mathcal{F}_k$ with (3.27);

else if $\rho \leq \overline{\rho}$ **then** Use (3.19) accordingly; Re-initialize $\mathcal{F}_k$ with (3.27);

;

$k \leftarrow k+1$;

end

Algorithm 3.1: ℓ_1-IPOPT algorithm

Remarks

Global convergence properties For a fixed value of the penalty parameter, the convergence of this problem should follow from the proofs in section 4.3 of [22]. In this work it is assumed boundedness and continuity of the functions, uniform positive definiteness of the Reduced Hessian; and a lower bound on the minimum singular value of the matrix and linear independence of columns of the matrix of constraint gradients. If this is satisfied, then there exists a sequence of iterates that will eventually generate iterates that are feasible and optimal. Moreover, this assumes that the filter is reset as μ_j is changed. By analogy the same should be apply for ρ_0.

3.6 Numerical Results

The ℓ_1–exact penalty strategies implemented within IPOPT and were compared against the version of IPOPT as described in Section 2.4, that was designated as *vanilla*. For this, the main ℓ_1-penalty strategies were the *direct* from eq. (3.2) and the *inverse* from eq. (3.8) forms of the penalty-barrier objective function. Each of these strategies were tested with different penalty parameter update rules, these include, *quadratic* (normal), *linear*, pure Hessian (no Σ term in the augmented Hessian), and *fixed* update rules. In summary, the ℓ_1-penalty phase was set-up to use 8 different kinds of update of the penalty parameter and were tested against the *vanilla* IPOPT. The ℓ_1-penalty phase update modes were:

- *direct*-quadratic (ρ)
- *direct*-quadratic-no-Σ_k
- *direct*-linear (ρ_L, $\gamma_\rho = 0$)
- *direct*-fixed (ρ_0)
- *inverse*-quadratic ($1/\rho$)
- *inverse*-quadratic-no-Σ_k
- *inverse*-linear ($1/\rho_L$, γ_ρ)

- *inverse*-fixed ($1/\rho_0$)

Rather than attempting to show that the proposed strategies outperform the original version of IPOPT, the objective was to show the effect on the robustness of the algorithm.

All of the used IPOPT libraries were compiled with GCC and GFortran 7.4.0. For the BLAS and LAPACK routines, the sequential version of the Intel MKL library v2018.1-163 [33] was used alongside MA57 [34] as the linear solver. All the relevant options for IPOPT were set to their default values and all of the problems were compiled using AMPL [35] in a Linux machine with a Intel(R) Xeon(R) E5-2440 CPU.

The first numerical case study is based on the CUTEr [36] set of problems. In this particular test, since it has been reported that IPOPT performs competitively, an augmented version of IPOPT that switches from the regular to the ℓ_1-penalty strategy was tested. For this, the regular IPOPT algorithm is applied at the beginning, and then it switches to the ℓ_1-penalty strategy only after the first *feasibility restoration* is triggered. This provides a reference study in which the posedness of the problem is not known beforehand. To compare the strategies, performance plots were constructed. The plots show the probability of a particular strategy's cumulative performance ratio based on iterations, i.e.

$$P(\tau) = \frac{1}{N}\text{size}\left\{p \in \text{Problems} : \left(\frac{i_{p,s}}{\min\{i_{p,s} : s \in \text{strategy}\}} \leq \tau\right)\right\},$$

being at least τ of the best possible ratio [Dolanmore].

The results for the original CUTEr set are shown on Figure 3.1. From these results, it can be noted that some of the ℓ_1-penalty strategies produce an increase of robustness, except for the fixed penalty strategies, which performs worse than the *vanilla* strategy. This effect can be associated with the capability of the ℓ_1-penalty strategies to solve some problems like *steenbrg* and *steenbrc*, in which IPOPT detects degenerate Jacobians. As a consequence *vanilla* IPOPT sets δ_c to a nonzero value, however several multipliers of the constraints grow quickly to large values and then solver stops at feasible points with a failure in the computation of the line-search step. The penalty strategies, on the other hand; are able to compute steps that converge to stationary points consistently. The results

of the original CUTEr set show that the ℓ_1-penalty strategies require more iterations as opposed to the *vanilla* solver. Nevertheless the ℓ_1-penalty strategies can solve more than 99% of the problems.

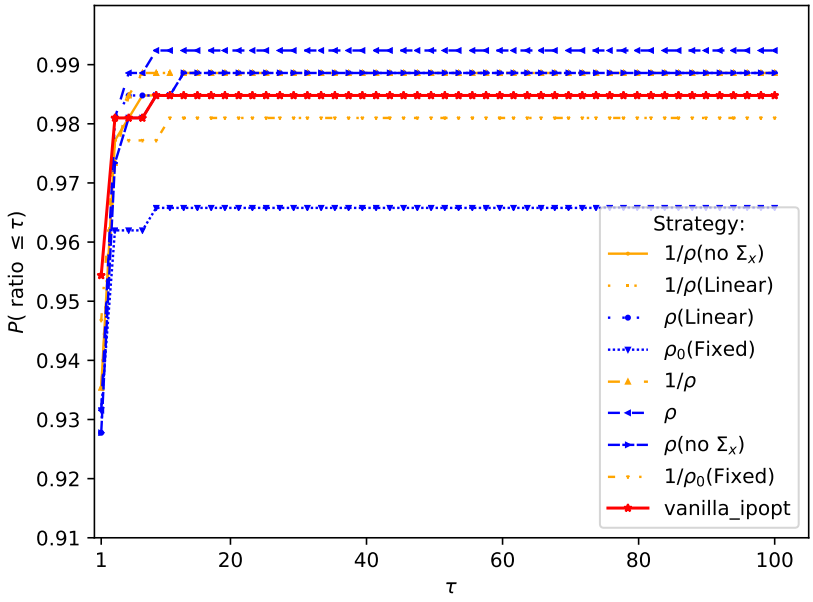

Figure 3.1: Performance profiles for unmodified CUTEr test set

To further test the effect of degeneracies, a modified CUTEr set as proposed in [10] was also tested. This can be constructed by splitting a single constraint from each one of the base problems into two, which results in a feasible but degenerate set of problems, i.e. a base constraint $c_1(x) = 0$ is replaced by,

$$c_1(x) = 0 \quad \text{, and} \quad c_1(x) - c_1^2(x) = 0. \tag{3.28}$$

Then the modified CUTEr set was tested for all the strategies. For the following case studies, the ℓ_1–exact penalty strategies were used immediately from the start of the optimization rather than engaging them after feasibility restoration is triggered. This is because the problems will be inconsistent everywhere and it will ensure that the advantages

of the regularizing effect are fully exploited during the execution of the optimization algorithm. Additionally, because the constraints (3.28) were selected and appended arbitrarily, the positions of the variables and constraints at the internal level of the problems were permuted to diminish the potential effects during the symbolic factorization stages within IPOPT by using AMPL's *nl* writer options. The results for the modified CUTEr unpermuted and permuted (problems for which the order of the constraints and has been shuffled) set are shown on Figures 3.2 and 3.3 respectively. Both plots show consistent results when the strategies are compared among each other. The permuted tests show a marginal increase of robustness of the ℓ_1-penalty strategies. On the other hand, both fixed penalty approaches perform worse than any of the other strategies including *vanilla*. Moreover, the *direct* (ρ, ρ_L and ρ_{Σ_x}) strategies are able to solve more problems that any other solver in the tests, particularly the *direct* strategy with the linear update rule of the penalty parameter. Approximately 95% of the problems can be solved with this strategy. Some example problems that fail with the *vanilla* approach but are solved with the *direct* penalty approach include *hs047*, *ssnlbeam*, *eigenc2* and the *brainpc-x* problems. At the same time, problems like *qpnboei1*, *steenbrf* and *gridnetc* converge to feasible points at which the penalty parameter grows indefinitely with the *direct* penalty approaches; however, this is not experienced by the *inverse* approaches, which are able to solve these problems to an acceptable level.

The results given by the modified CUTEr set suggest that redundant constraints can be handled, as long as the ℓ_1–exact penalty strategy is used from the start. However, there are situations in which the degeneracies arise as result of an intrinsic property of the problem to be solved and they might not be captured by the previous test. With this in mind, a set of Mathematical Programs with Equilibrium constraints (MPEC) was tested [37].

An MPEC is an optimization problem that involves variational inequalities, and they appear in several kinds of applications, most notably bilevel optimization problems. MPECs can be reformulated into a Mathematical Program with Complementarity Constraints (MPCC),

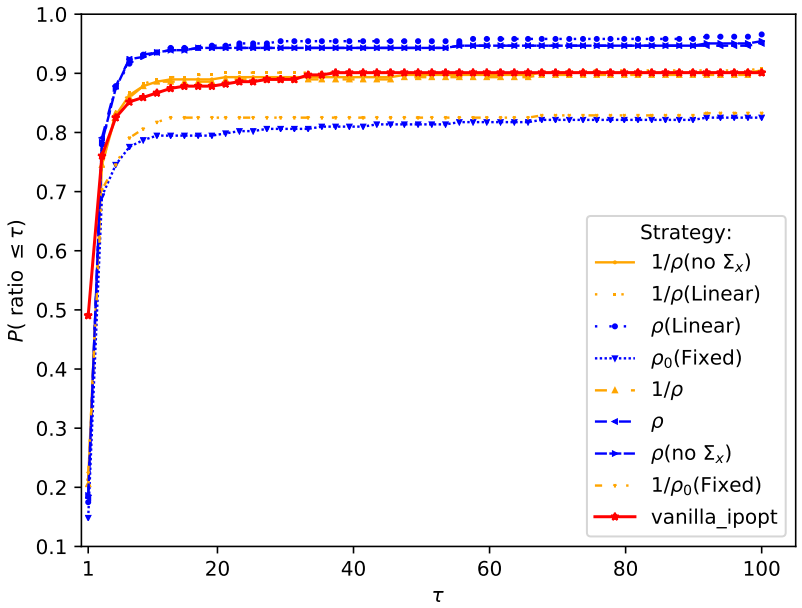

Figure 3.2: Performance profiles for CUTEr test set with a degenerate constraint

$$
\begin{aligned}
&\text{MPCC}: && && \text{RegEq}\,(0): \\
&\min_{x,y,z} && f\,(x,y,z) && \min_{x,y,z} && f\,(x,y,z) \\
&\text{s.t.} && h\,(x,y,z)=0, && \text{s.t.} && h\,(x,y,z)=0, \\
& && g\,(x,y,z)\geq 0, && \iff && g\,(x,y,z)\geq 0, \\
& && 0\leq x_i \perp y_i \geq 0 \quad \forall i, && && x_i y_i = 0 \quad \forall i, \\
& && && && x\geq 0 \\
& && && && y\geq 0
\end{aligned}
\tag{3.29}
$$

where the last constraint of the MPCC implies that $x_i = 0$ or $y_i = 0$ or both $\forall i$, and $x \geq 0, y \geq 0$. This kind of problem does not satisfy constraint qualifications, e.g. MFCQ at all feasible points, so the corresponding multipliers might become unbounded [38]. This creates a particularly difficult situation for standard NLP solvers. Under several assump-

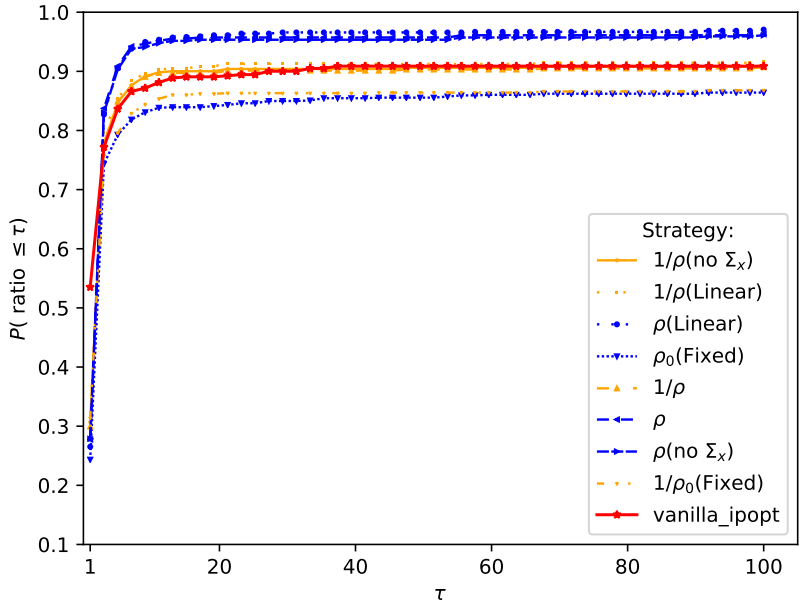

Figure 3.3: Performance profiles for *permuted* CUTEr test set with a degenerate constraint

tions [6], the MPCCs can be attempted to be solved as a reformulated NLP, e.g. using equality constraints to replace the complementarities and then coupling them with a relaxation parameter ϵ (RegEq(ϵ)), or by penalizing the constraint in the objective function. Still, as the number of complementarities increases, the capability to solve these problems becomes hindered. Therefore, the algorithms presented here might improve the chances of dealing with difficult instances of MPCCs as they typically display improved robustness. The problems from the MacMPEC library [ref] were used. For this, a reformulation of the complementarity constraints with equalities without relaxation (RegEq(0)) was tested with all the ℓ_1-penalty strategies previously described, starting directly into ℓ_1-penalty mode; and then, performance profiles were constructed in Figure 3.4. In this test set, the *vanilla* version of the algorithm is at the bottom in terms of robustness, then it is followed by the fixed penalty strategies, which do not have an active penalty update rule, therefore

hindering their performance.

Furthermore, this test library shows that both the *inverse* quadratic and linear update of the penalty parameter solve the most problems. On the other hand, the *direct* strategies result in large values of the penalty parameter and thus the values of the multipliers of bounds and equality constraints are also large, then the line-search experiences difficulties at almost feasible points. This situation is more favorable in the *inverse* mode, which has values close to 1 for the equality multipliers; however, some problems experience similar behavior with the multipliers of the bounds and the algorithm will eventually fail. Examples of this last situation is found in problems like *pack-comp1c-16* and *liswet1-inv-200*. The

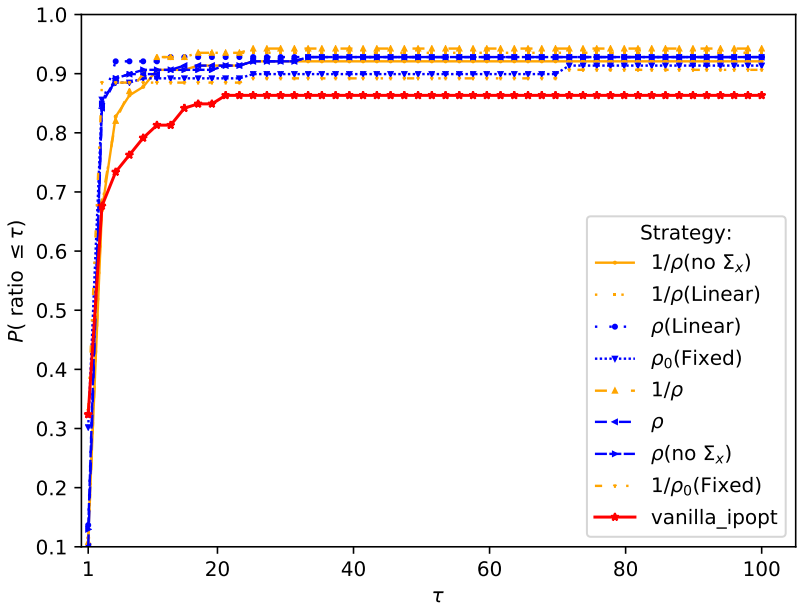

Figure 3.4: Performance profiles for Mathematical Programs with Equilibrium Constraints (MacMPEC) test set

results from the MacMPEC tests illustrate some of the limitations in terms of the numerical issues that arise from the violation of the MFCQ.

Finally, motivated by the results of the structured strategies presented in [9], and [8], two blending problems were also solved. The premise of a blending problem is that given a set of feeds with several qualifications, a blend with a set of specifications that minimizes cost has to be created. This problem is an instance of the pooling problem which has more complex forms and usually involve integer variables. A complication with the blending problem formulation is that it involves bilinear terms that at certain points, creates a rank-deficient Jacobian when the flows are zero. For instance, for a set of feed sources $i \in I$, tanks $j \in J$, products $k \in K$, and time horizon $t \in \{1, ..., N_T\}$; the mass and the quality(property) balance is written as follows,

$$\sum_{k \in K} s_{t,jk} - \sum_{i \in I} s_{t,ij} + v_{t+1,j} = v_{t,j}, \quad t \in \{1, ..., N_T\}, j \in J \tag{3.30}$$

$$\sum_{k \in K} q_{t,j} s_{t,jk} - \sum_{i \in I} q_{t,i} s_{t,ij} + q_{t+1,j} v_{t+1,j} = q_{t,j} v_{t,j}, \quad t \in \{1, ..., N_T\}, j \in J, \tag{3.31}$$

where $s_{t,lm}$ represents the stream flow between nodes l and m (e.g. tank j to product stream m), $q_{t,j}$ and $v_{t,j}$ are the stream qualities and tank inventories at period t. Points at which the flow becomes zero are typically degenerate.

IPOPT and the ℓ_1-penalty strategies were tested in two instances of this kind of blending problem. The results for the blending problems are shown on Tables 3.1 and 3.2. For both problems it can be concluded that *vanilla* IPOPT is not as competitive as the rest of the ℓ_1-penalty strategies because for no stationary point is found for either problem. Moreover, the default regularization (δ based) allows to compute search directions, but the progress towards stationary points is rather slow as the regularized Newton steps no longer attain super-linear convergence. For these problems the *direct* penalty with either quadratic and linear update rule require fewer iterations overall. The strategies with the fixed penalty parameter can solve the problems but they require more iterations than the other strategies.

Strategy	Iter	CPUs	Objective	Status
vanilla	18519	182	-5.22E+02	restoration failure
ρ	135	1.38	-5.22E+02	optimal
ρ_{Σ_x}	135	1.37	-5.22E+02	acceptable
ρ_0	138	1.63	-5.22E+02	optimal
ρ_L	135	1.36	-5.22E+02	optimal
$1/\rho$	677	8.62	-5.18E+02	optimal
$1/\rho_{\Sigma_x}$	677	8.72	-5.18E+02	acceptable
$1/\rho_0$	348	4.19	-5.18E+02	acceptable
$1/\rho_L$	677	8.61	-5.18E+02	optimal

Table 3.1: Results for Blend 1 (SP-RCSA) n =827, m =766

Strategy	Iterations	CPUs	Objective	Status
vanilla	37232	4460	-1.09E+02	restoration failure
ρ	1239	215	-8.96E+01	acceptable
ρ_{Σ_x}	1239	215	-8.96E+01	optimal
ρ_0	18685	4260	-1.08E+02	acceptable
ρ_L	1239	215	-8.96E+01	acceptable
$1/\rho$	2512	310	-6.19E+01	acceptable
$1/\rho_{\Sigma_x}$	2512	310	-6.19E+01	acceptable
$1/\rho_0$	2492	293	-1.07E+02	acceptable
$1/\rho_L$	2512	305	-6.19E+01	acceptable

Table 3.2: Results Blend 2 (SP-GRB) n =5142, m =4668

3.7 Conclusion

Ill-posed problems lack regularity of local linearizations of the set of constraints. This situation curtails the performance and robustness of state-of-the-art solvers like IPOPT. The proposed ℓ_1-exact penalty-barrier approach has the advantage of satisfying the LICQ until feasible points are reached. Moreover, combined with the update rule, it is possible enhance the robustness properties of IPOPT.

Numerical testing has revealed that there is an increase of performance, though it is possible to construct problems with pathological behaviour that can lead to failures. Finally, MPCCs were solved directly as NLP without introducing additional strategies to handle the complementarities. This demonstrated that the presented algorithm can be useful for degenerate problems.

Chapter 4

Nonlinear Model Predictive Control and State-Estimation

4.1 Introduction

In Chemical Engineering process models can typically be constructed from fundamental understanding of the physics of the system of interest. This is what is commonly referred as first-principles. Their importance is reflected in the fact that there are several useful applications that combine these models and optimization, e.g. process design [39]. Moreover, transitioning to models that take into account the dynamic details of the processes is key for predicting the behaviour of the process, and therefore opening the possibility of using optimization to determine the best operational actions.

The availability of dynamic models comes with an increase of complexity, in terms of the theoretical and practical aspects that have to be consider to handle them. Typically, models are described by combinations of Differential and Algebraic Equations (DAE), for instance the following semi-explicit DAE model

$$\begin{aligned} \frac{d\zeta}{dt} &= \boldsymbol{F}\left(\zeta, \chi\right), \\ 0 &= \boldsymbol{G}\left(\zeta, \chi\right), \\ \zeta\left(0\right) &= \zeta_0, \end{aligned} \tag{4.1}$$

where two vectors of variables are considered, differential $\zeta \in \mathbb{R}^{n_\zeta}$ and algebraic $\chi \in \mathbb{R}^{n_\chi}$. Moreover, the right-hand-sides involved are the functions $\boldsymbol{F} : \mathbb{R}^{n_\zeta} \times \mathbb{R}^{n_\chi} \to \mathbb{R}^{n_\zeta}$ and $\boldsymbol{G} : \mathbb{R}^{n_\zeta} \times \mathbb{R}^{n_\chi} \to \mathbb{R}^{n_\chi}$; where it is assumed that $\nabla_\chi \boldsymbol{G}\left(\zeta, \chi\right)$ is nonsingular for some

ζ contained in an open set $\mathcal{N}_\zeta \subset \mathbb{R}^{n_\zeta}$. As a consequence of embedding a model in the form of (4.1) into an optimization problem, the continuous nature of the DAE model will result into an infinite dimensional problem. Instead of using a variational approach, for the rest of this work we consider discretizing the time domain. For this chapter we first present a full-discretization approach in which all of the variables are considered, then the background information for Model Predictive Control and State-Estimation, and finally an implementation of an unified framework for them.

4.2 Direct Transcription

It is possible to discretize system (4.1) by approximating the variable vectors ζ and χ as polynomials at specific points in time or collocation points. Consider partitioning the time domain into N_f finite elements and N_c collocation points, additionally for a given finite element i the Lagrange interpolating polynomial basis for the differential variable ζ has the form

$$\zeta(t) = \sum_{j=0}^{N_c} l_j(\tau)\,\zeta_{ij}, \quad l_j(\tau) = \prod_{k=0,\neq j}^{N_c} \frac{(\tau - \tau_k)}{(\tau_j - \tau_k)}, \qquad t \in [t_{i-1}, t_i], \tag{4.2}$$

for some $\tau \in [0,1]$, and collocation points $\tau_0 = 0, \tau_j < \tau_{j+1}$ for $j = 0, \ldots, N_c - 1$. Assuming that the polynomial approximation is exact at the j-th collocation points, then $\zeta(t_{ij}) = \zeta_{ij}$ for some $t_{ij} = t_{i-1} + \tau_j h$; moreover considering τ_j to have the value of the roots of an N_c-th order polynomial, the integration of the differential term becomes exact. Furthermore, since similar ideas can be used for the algebraic variable χ, then the DAE system can be represented as follows,

$$\begin{aligned}
\sum_{j=0}^{N_c} \zeta_{ij} \frac{dl_j(\tau_k)}{d\tau} &= \boldsymbol{F}(\zeta_{ik}, \chi_{ik}) \qquad i \in \{1, \ldots, N_f\}, k \in \{1, \ldots, N_c\}, \\
0 &= \boldsymbol{G}(\zeta_{ik}, \chi_{ik}) \qquad i \in \{1, \ldots, N_f\}, k \in \{1, \ldots, N_c\}, \\
\zeta_{i+1,0} &= \sum_{j=0}^{N_c} l_j(1) \zeta_{ij} \qquad i \in \{1, \ldots, N_f - 1\}, \\
\zeta_{1,0} &= \zeta_0.
\end{aligned} \tag{4.3}$$

It can be noted the system of equations of (4.3) is an algebraic approximation of the original DAE system, moreover optimization case studies with such systems embedded can now be handled with traditional NLP solvers. Moreover, we present an optimal control strategy that uses a DAE model as the basis for the formulation of NLPs that estimate the current estate and control action for a process. To simplify notation though the rest of this Chapter, we will consider that the discretized DAE (4.3) can be represented as a *discrete*-time system, and the set of differential and algebraic variables are condensed into a single vector of states. The first issue that will be covered is State-Estimation.

4.3 Moving Horizon Estimation

Dynamic processes rarely have states that can be measured directly. In reality, most of the systems of interest only generate a set of measurements over a period of time. Thus, the first step towards the creation of an optimal control strategy is the determination of the states of the system given a set of measurements.

Consider a *discrete*-time model of a process,

$$\begin{aligned}
x(k+1) &= f(x(k), w(k)), \\
y(k) &= h(z(k)) + v(k),
\end{aligned} \tag{4.4}$$

where the maps are the process model $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, and the measurement function $h : \mathbb{R}^n \to \mathbb{R}^{n_y}$. At some time $k \in \mathbb{Z}_{\geq 0}$, the model generates measurements in the form of a vector $y \in \mathbb{R}^{n_y}$, for some unknown state and disturbance vectors $x \in \mathbb{R}^{n_x}$ and $w \in \mathbb{R}^{n_w}$

respectively. Moreover, it is considered that the measurements have noise, represented by the vector $v \in \mathbb{R}^{n_y}$. Assuming that both the model and the set of measurements is known, it is natural to create a problem that minimizes the discrepancies between them; in other words *fits* the data to the model. This is the primary idea of the full-information estimator, which attempts to minimize the objective function in terms of the disturbance and noise;

$$\mathcal{V}_N^{\text{full}}\left(z_{0|N}, \boldsymbol{\omega}_N\right) \coloneqq \ell_x\left(z_{0|N} - \overline{z}_0\right) + \sum_{i=0}^{N} \ell_i\left(\omega_{i|N}, \nu_{i|N}\right) \tag{4.5}$$

for some initial and $i-$th stage cost *continuous* functions $\ell_x : \mathbb{R}^{n_x} \to \mathbb{R}$ and $\ell_i : \mathbb{R}^{n_w} \times \mathbb{R}^{n_y} \to \mathbb{R}$ respectively; and initial state $\overline{z}_0 \in \mathbb{X}$, where $\mathbb{X}$ is an invariant set. It can be noted that this objective is formulated for N stages of time, starting from 0; where the variables of the problem $z \in \mathbb{R}^{n_x}$, $\omega \in \mathbb{R}^{n_w}$, and $\nu \in \mathbb{R}^{n_v}$ are used in place of the variables of the system x, w and v respectively. Furthermore, the bold symbols represent a sequence of vectors, e.g. $\mathbf{w} \coloneqq \{w(k)\}$ for some $k \in \mathbb{Z}_{\geq 0}$. Combining the objective and model (4.4), yields the optimization problem for the full-information estimator of size N

$$\begin{aligned}
\min_{z_{0|N}, \boldsymbol{\omega}_N} \mathcal{V}_N^{\text{full}}\left(z_{0|N}, \boldsymbol{\omega}_N\right) &\coloneqq \ell_x\left(z_{0|N} - \overline{z}_0\right) + \sum_{i=0}^{N} \ell_i\left(\omega_{i|N}, \nu_{i|N}\right) \\
\text{s.t.} \quad z_{l+1|N} &= f\left(z_{l|N}, \omega_{l|N}\right) \quad l \in \{0, \ldots, N\}, \\
y(l) &= h\left(z_{l|N}\right) + \nu_{l|N} \quad l \in \{0, \ldots, N\}, \\
z_{l|N} &\in \mathbb{X} \quad l \in \{0, \ldots, N\}, \\
\omega_{l|N} &\in \mathbb{W}, \quad l \in \{0, \ldots, N\},
\end{aligned} \tag{4.6}$$

where $\mathbb{W}$ is an invariant set for the disturbances. The name full-information reflects the fact that the problem takes the *full* sequence of measurements from time $k = 0$. Though this problem grows in size as k increases to the point of becoming impractical, it is conceptually important, because it sets the foundation for the stability properties of more tractable estimators; i.e. the Moving Horizon Estimator (MHE).

The stability of the full-information estimator primarily assumes that the disturbances and noise sequences satisfy a property of convergence. Then it is possible to show that the

state estimate will approach to the real state asymptotically.

To discuss these properties, it is necessary to consider functions that have special properties as they are sampled over time.

Definition 11 ($\mathcal{K}, \mathcal{K}_\infty$ and $\mathcal{KL}$ functions)**.** *A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}$ function if it is continuous, strictly increasing, and zero when evaluated at zero. A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}_\infty$ function if it is class $\mathcal{K}$ and it grows unboundedly, i.e. $\alpha(s) \to \infty$ as $s \to \infty$. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{KL}$ function if it is continuous, and $\beta(\cdot, k)$ is a $\mathcal{K}$ function for every $k \in \mathbb{Z}_{\geq 0}$ and for every $s \geq 0$, $\beta(s, \cdot)$ is nonincreasing and $\lim_{k\to\infty} \beta(s, k) = 0$.*

These functions will be later useful to define the properties of the optimal control scheme in the next section. Furthermore, the following notation will be used. The sup norm over a sequence $\|\mathbf{w}\| = \sup\{\|w(k)\| \mid k \in \mathbb{Z}_{\geq 0}\}$; and $h(\mathbf{x}) \coloneqq \{h(x(k))\}$ for some $k \in \mathbb{Z}_{\geq 0}$.

The following analysis of stability for the full-information problem has been adapted from [40]. To start the description of the properties of the system, it is required to assume that perturbations to the system result in significant responses. This is related to the ability to determine current and future sequences of states after the occurrence of given state, in other words to assess the detectability of the system. There is not an unified definition of *detectability* across the literature. In this work it is assumed that two different sequences of states of the system are the result of the discrepancies of disturbances and noise besides their initial state, whose influence decays over time. This is denominated as the incrementally Input/Output-to-State Stability.

Definition 12 (Incrementally Input/Output-to-State Stability (i-IOSS))**.** *Consider the system* (4.4)*; two initial states z_1 and z_2, and two disturbance sequences $\mathbf{w}_1$ and $\mathbf{w}_2$ that generate state sequences $\mathbf{x}_1(z_1, \mathbf{w}_1)$ and $\mathbf{x}_2(z_2, \mathbf{w}_2)$. The system is Incrementally Input/Output-to-State Stable (i-IOSS) if there exist functions $\beta(\cdot) \in \mathcal{KL}$ and $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}$, and for all $k \in \mathbb{Z}_{\geq 0}$, the system satisfies*

$$\|x_1(k; z_1, \mathbf{w}_1) - x_2(k; z_2, \mathbf{w}_2)\| \leq \beta_1(\|z_1 - z_2\|, k) + \alpha_1(\|\mathbf{w}_1 - \mathbf{w}_2\|) + \alpha_2(\|h(\mathbf{x}_1) - h(\mathbf{x}_2)\|), \tag{4.7}$$

where $x_i(k; z_i, \mathbf{w})$ *denotes a state that satisfies* $x_i(k) = f(x_i(k-1), w_i(k-1))$ *with* $x_i(0) = z_i$*, and* $\mathbf{w}_i = \{w_i(0), w_i(1), \ldots, w_i(k-1)\}$ *for* $i \in \{1, 2\}$*, and* $k \in \mathbb{Z}_{\geq 0}$*.*

Then, as mentioned earlier, the ability of converge to the true state as more data becomes available regardless of the initial reference state, serves as the primary definition of stability of the full-information problem. Additionally, the disturbance and noise have to be considered into this definition; this is summarized into the Robust Global Asymptotic Stability definition.

Definition 13 (Robust Global Asymptotic Stability (RGAS))**.** *Consider a noisy measurement sequence given by* $\mathbf{y} = h(\mathbf{x}) + \mathbf{v}$*. An estimate is considered to be Robust Global Asymptotic Stable (RGAS) if for initial state* $\overline{z}_0$*, and bounded disturbance and noise sequences* $(\mathbf{w}, \mathbf{v})$*, there exists functions* $\delta_w(\cdot)\,.\delta_v(\cdot) \in \mathcal{K}$ *such that for* $k \in \mathbb{Z}_{\geq 0}$ *the following holds*

$$\|x(k; x_0, \mathbf{w}) - z_{k|N}\| \leq \beta_x\left(\|z_{0|N} - \overline{z}_0\|, k\right) + \delta_w(\|\mathbf{w}\|) + \delta_v(\|\mathbf{v}\|), \tag{4.8}$$

where $z_{k|N}$*, and its associated variables* $\boldsymbol{\omega}_{k|N}$ *and* $z_{0|N}$ *are given by the full-information problem with some prior state information* $\overline{z}_0$ *for a window of size* N*.*

What follows are definitions to set up the set of assumptions for the RGAS estimator; these are useful to characterize the evolution of the disturbance sequence over time.

Definition 14 (Bounded sequence, $\mathbb{B}$)**.** *A sequence* $\mathbf{w} = \{w(k)\}$*, for* $k \in \mathbb{Z}_{\geq 0}$ *is bounded if* $\|\mathbf{w}\|$ *is finite. Moreover, the set of all the bounded sequences is given by* $\mathbb{B}$*.*

Definition 15 (Convergent sequence, $\mathbb{C}$)**.** *A bounded sequence* $\mathbf{w} = \{w(k)\}$*, for* $k \in \mathbb{Z}_{\geq 0}$ *is regarded convergent if* $\|w(k)\| \to 0$ *as* $k \to \infty$*. Moreover, the set of all convergent sequences is given by* $\mathbb{C}$*.*

Definition 16 ($\beta-$convergent sequence, $\mathbb{C}_\beta$)**.** *Consider a function* $\beta(\cdot) \in \mathcal{KL}$*, a bounded sequence* $\boldsymbol{w} = \{w(k)\}$*, for* $k \in \mathbb{Z}_{\geq 0}$ *is regarded as* $\beta-$*convergent if*

$$\|w(k)\| \leq \beta(\|\mathbf{w}\|, k) \quad \forall k \in \mathbb{Z}_{\geq 0}. \tag{4.9}$$

Furthermore, the set of all $\beta-$*convergent sequences is given by* $\mathbb{C}_\beta$*.*

In order to attain an RGAS estimate; it is assumed that both sequences of disturbance and noise belong to a class of convergent sequences.

Assumption 1 ($\beta-$convergent disturbances)**.** *The sequences of disturbance and measurement noise* $\mathbf{w}, \mathbf{v} \in \mathbb{C}_\beta$ *for some* $\beta(\cdot) \in \mathcal{KL}$*.*

Moreover, the stage cost of the objective function (4.5) is assumed to be positive definite in the dynamic sense, by being bounded by $\mathcal{K}_\infty$ functions. As a consequence of these two assumptions and the i-IOSS property of the system the subsequent lemma is useful to prove the RGAS of the full-information estimator.

Assumption 2 (Positive definite stage cost)**.** *Let the functions* $\underline{\gamma}_x, \underline{\gamma}_w, \underline{\gamma}_\nu, \gamma_w, \gamma_w, \gamma_\nu \in \mathcal{K}_\infty$*. All the stage costs are continuous and they satisfy the following inequalities*

$$\begin{aligned} \underline{\gamma}_x\left(\|z_{0|N} - \overline{z}_0\|\right) \leq \ell_x\left(z_{0|N} - \overline{z}_0\right) \leq \gamma_x\left(\|z_{0|N} - \overline{z}_0\|\right), \\ \underline{\gamma}_w\left(\|\omega_{k|N}\|\right) + \underline{\gamma}_\nu\left(\|\nu_{k|N}\|\right) \leq \ell_i\left(\omega_{k|N}, \nu_{k|N}\right) \leq \gamma_w\left(\|\omega_{k|N}\|\right) + \gamma_\nu\left(\|\nu_{k|N}\|\right) \quad \forall k \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{4.10}$$

Lemma 1 ([40])**.** *Consider an i-IOSS system following* (4.4)*, that satisfies Assumptions 1 and 2. Then the full-information estimator satisfies the following properties.*

1. *For functions* $\pi_x, \pi_w, \pi_v \in \mathcal{K}$

$$\|x(k; x_0, \mathbf{w}) - z_{k|N}\| \leq \pi_x\left(\|z_{0|N} - \overline{z}_0\|\right) + \pi_w\left(\|\mathbf{w}\|\right) + \pi_v\left(\|\mathbf{v}\|\right),$$

   for all $k \in \mathbb{Z}_{\geq 0}$

2. *For* $k \to \infty$*,*

$$\|x(k; x_0, \mathbf{w}) - z_{k|N}\| \to 0.$$

Thus, it is possible to assert the RGAS of the full-information estimator with the folloing theorem.

Theorem 3 (RGAS full-information estimator)**.** *Consider a detectable system (i-IOSS property), that follows* (4.4)*. If it satisfies Assumptions 1 and 2, i.e. convergent disturbances and positive definite stage costs respectively; then the Full-information estimator is Robust Global Asymptotic Stable (RGAS).*

Proof. As a consequence of both propositions of Lemma 1, it can be noted that a $\overline{\beta}(\cdot) \in \mathcal{KL}$ function can be constructed such that

$$\|x(k; x_0, \mathbf{w}) - z_{k|N}\| \leq \overline{\beta}\left(\pi_x\left(\|z_{0|N} - \overline{z}_0\|\right) + \pi_w(\|\mathbf{w}\|) + \pi_v(\|\mathbf{v}\|), k\right). \tag{4.11}$$

Moreover, it is possible to show that any function $\beta(\cdot) \in \mathcal{KL}$ has the property $\beta\left(\sum_i^n a_i, k\right) \leq \sum_i^n \beta(na_i, k)$ for some nonnegative coefficients a_i. Thus, using this property on $\overline{\beta}(\cdot)$; yields the following result

$$\|x(k; x_0, \mathbf{w}) - z_{k|N}\| \leq \alpha_x\left(\|z_{0|N} - \overline{z}_0\|, k\right) + \alpha_w(\|\mathbf{w}\|, k) + \alpha_v(\|\mathbf{v}\|, k),$$

where $\alpha_x, \alpha_w, \alpha_v \in \mathcal{KL}$. Finally, without lost of generality, setting $\beta_x(\cdot) \coloneqq \alpha_x(\cdot), \delta_w(\cdot) \coloneqq \alpha_w(\cdot, 0)$, and $\delta_v(\cdot) \coloneqq \alpha_v(\cdot, 0)$ gives the desired result, i.e. Eq. (4.8). □

Though the assumption of convergent disturbance is restrictive, there are recent developments that indicate that a slight variation of the full-information objective and the i-IOSS property can relax the assumption of convergent disturbance and noise [41]. Nevertheless, with this properties it is possible to create practical versions of state estimators.

Regarding the stage costs, it is convenient to choose quadratic forms as follows,

$$\begin{aligned} \ell_x(z_0 - \overline{z}_0) &\coloneqq (z_0 - \overline{z}_0)^T \Pi_0^{-1} (z_0 - \overline{z}_0), \\ \ell_i(\omega_i, \nu_i) &\coloneqq \omega_i^T \mathcal{Q}_i^{-1} \omega_i + \nu_i^T \mathcal{R}_i^{-1} \nu_i, \end{aligned} \tag{4.12}$$

where the matrices $\Pi_0 \in \mathbb{R}^{n_x \times n_x}$, $\mathcal{Q}_i \in \mathbb{R}^{n_w \times n_w}$, and $\mathcal{R}_i \in \mathbb{R}^{n_y \times n_y}$ are chosen to be positive definite. With these functions the full-information for an $N-$th sized sequence of

measurements $\{y(0), \ldots, y(N)\}$ and initial estimate $\overline{z}_0$ estimator problem is formulated

$$\begin{aligned}
\min_{z_{0|N}, \boldsymbol{\omega}_N} \quad & \mathcal{V}_N\left(z_{0|N}, \boldsymbol{\omega}_N\right) \coloneqq \left(z_{0|N} - \overline{z}_0\right)^T \Pi_0^{-1} \left(z_{0|N} - \overline{z}_0\right) + \sum_{i=0}^{N} \left[\omega_{i|N}^T \mathcal{Q}_i^{-1} \omega_{i|N} + \nu_{i|N}^T \mathcal{R}_i^{-1} \nu_{i|N}\right] \\
\text{s.t.} \quad & z_{l+1|N} = f\left(z_{l|N}, \omega_{l|N}\right) \quad l \in \{0, \ldots, N\}, \\
& y(l) = h\left(z_{l|N}\right) + \nu_{l|N} \quad l \in \{0, \ldots, N\}, \\
& z_{l|N} \in \mathbb{X} \quad l \in \{0, \ldots, N\}, \\
& \omega_{l|N} \in \mathbb{W} \quad l \in \{0, \ldots, N\}.
\end{aligned} \tag{4.13}$$

In practice, the full-information problem quickly becomes intractable, therefore online implementations of this estimator in continuous processes is not possible. Instead, the concept of a constant *horizon window* is used. The horizon window is the set of points contained from the current time to $\mathcal{N}$ steps in the past, i.e. $\{k - \mathcal{N}, \ldots, k\}$ where k is the current time of the process. Thus, the estimator is specified to only include measurements inside the current horizon window $\mathbf{y}_k \coloneqq \{y(k - \mathcal{N}), \ldots, y(k)\}$, and it is denominated the Moving Horizon Estimator (MHE); which is written as follows

$$\begin{aligned}
\min_{z_{k-\mathcal{N}|k}, \boldsymbol{\omega}_k} \quad & \mathcal{V}_{\mathcal{N}}\left(z_{k-\mathcal{N}|k}, \boldsymbol{\omega}_k\right) \coloneqq \left(z_{k-\mathcal{N}|k} - z_{k-\mathcal{N}|k-1}\right)^T \Pi_{k-\mathcal{N}|k-1}^{-1} \left(z_{k-\mathcal{N}|k} - z_{k-\mathcal{N}|k-1}\right) + \\
& \qquad \sum_{i=-\mathcal{N}}^{-1} \left[\omega_{k+i|k}^T \mathcal{Q}_i^{-1} \omega_{k+i|k} + \nu_{k+i|k}^T \mathcal{R}_i^{-1} \nu_{k+i|k}\right] + \nu_{k|k}^T \mathcal{R}_0^{-1} \nu_{k|k} \\
\text{s.t.} \quad & z_{k+l+1|k} = f\left(z_{k+l|k}, \omega_{k+l|k}\right) \quad l \in \{-\mathcal{N}, -\mathcal{N}+1, \ldots, -1\}, \\
& y(k+l) = h\left(z_{k+l|k}\right) + \nu_{k+l|k} \quad l \in \{-\mathcal{N}, -\mathcal{N}+1, \ldots, 0\}, \\
& z_{k+l|k} \in \mathbb{X} \quad l \in \{-\mathcal{N}, -\mathcal{N}+1, \ldots, 0\}, \\
& \omega_{k+l|k} \in \mathbb{W} \quad l \in \{-\mathcal{N}, -\mathcal{N}+1, \ldots, -1\},
\end{aligned} \tag{4.14}$$

for some *prior* (from time $k-1$) state-estimate $z_{k-\mathcal{N}|k-1}$ and covariance $\Pi_{k-\mathcal{N}|k-1}$. The horizon window is shifted ahead every step in time, e.g. at time $k+1$ the horizon window is $\{k+1-\mathcal{N}, \ldots, k+1\}$, such that the problem keeps a *fixed* size. This means that every

subsequent problem should have similar complexity, however the compromise is that the a horizon window ignores all past information (all data from $\{0, \ldots, k-\mathcal{N}-1\}$). Therefore, the term from the objective which is denominated as the *arrival cost*,

$$\varphi_a\left(z_{k-\mathcal{N}|k} - z_{k-\mathcal{N}|k-1}, \Pi^{-1}_{k-\mathcal{N}|k-1}\right) \coloneqq \left(z_{k-\mathcal{N}|k} - z_{k-\mathcal{N}|k-1}\right)^T \Pi_{k-\mathcal{N}|k-1} \left(z_{k-\mathcal{N}|k} - z_{k-\mathcal{N}|k-1}\right),$$

attempts to under-bound the contribution of all the stage costs in the objective function of the full-information problem up to the point $k-\mathcal{N}-1$, and it becomes the primary consideration for attaining stability of the MHE.

Finally, note that the MHE problem (4.14) is formulated so that the resulting sequence of states includes the most recent state $z_{k|k}$, which ultimately can serve as the initial basis for a Model Predictive Control scheme for computing inputs to the process.

4.4 Nonlinear Model Predictive Control

Typical Nonlinear Model Predictive Control (NMPC) formulations consider the existence of a discrete-time model of a process that has unmeasured disturbances or model inaccuracies summarized by a vector $w(k) \in \mathbb{R}^{n_w}$ (as in the MHE section), and an input vector $u(k) \in \mathbb{R}^{n_u}$ for some $k \in \mathbb{Z}_{\geq 0}$ as follows,

$$x(k+1) = f_w(x(k), u(k), w(k)), \tag{4.15}$$

where $x(k) \in \mathbb{R}^{n_x}$, and the mapping $f_w : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ is the model for the uncertain plant. Moreover, it is assumed that this model can be split into two terms, a term for the nominal plant and a term that aggregates the disturbances as follows

$$x(k+1) = f_u(x(k), u(k)) + g_w(x(k), u(k), w(k)), \tag{4.16}$$

where the mappings are defined as follows, $f_u : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $g_w : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$.

Given a state at a particular time k, it is possible to use the nominal model of the plant to generate predictions of *future* states, e.g. for $N-1$ periods into the future

$$\begin{aligned} z_{l+1|k} &= f_u\left(z_{l|k}, \upsilon_{l|k}\right) \quad \forall l \in \{0,\ldots,N-1\}, \\ z_{0|k} &= x\left(k\right), \end{aligned} \tag{4.17}$$

where the predicted state and input are $z_l \in \mathbb{R}^{n_x}$ and $\upsilon_l \in \mathbb{R}^{n_u}$ respectively. In the NMPC framework, sequences of predicted states and inputs can be found such that they minimize an objective function in terms of stage-cost functions for N periods of time. In other words,

$$J_N\left(\mathbf{z}\right) \coloneqq \varphi_f\left(z_N\right) + \sum_{i=0}^{N-1} L\left(z_i, \upsilon_i\right), \tag{4.18}$$

for some stage-cost function $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and a terminal cost $\varphi_f : \mathbb{R}^{n_x} \to \mathbb{R}$. Furthermore, if invariant sets $\mathbb{X}_f \subseteq \mathbb{X} \subseteq R^{n_x}$, and $\mathbb{U} \subseteq R^{n_u}$; the NMPC optimization problem can be formulated as follows,

$$\begin{aligned} \min_{\boldsymbol{\upsilon}_k} \; & J_N\left(\mathbf{z}_k\right) \coloneqq \varphi_f\left(z_{N|k}\right) + \sum_{i=0}^{N-1} L\left(z_{i|k}, \upsilon_{i|k}\right) \\ \text{s.t.} \quad & z_{l+1|k} = f_u\left(z_{l|k}, \upsilon_{l|k}\right) \quad \forall l \in \{0,\ldots,N-1\}, \\ & z_{0|k} = x\left(k\right), \\ & z_{l+1|k} \in \mathbb{X} \quad l \in \{0,\ldots,N-1\}, z_{N|k} \in \mathbb{X}_f, \\ & \upsilon_{l+1|k} \in \mathbb{U} \quad l \in \{0,\ldots,N-1\}. \end{aligned} \tag{4.19}$$

Solving of problem (4.19) for a state $x\left(k\right)$ will generate a sequence of predicted controller inputs $\boldsymbol{\upsilon}_k$. Thus, it is possible to set $u\left(k\right) \coloneqq \upsilon_{0|k}$. At the next sampling time, after the solution of (4.19) for $x\left(k+1\right)$, the input of the process can be set in similar manner. This effectively generates a feedback law in the form $u\left(k\right) = h_u\left(x\left(k\right)\right)$.

The stability properties of the NMPC have been established by several authors, e.g. [42], [43], [17]. One key assumption is that problem (4.19) can be solved instantaneously, which is restrictive in the sense that most practical applications have models that require a non-negligible amount of time to be handled. Nevertheless, conceptually, we will assume that this is the case; to construct the set of properties of such problem.

Consider the same definitions of the $\mathcal{K}, \mathcal{K}_\infty$ and $\mathcal{KL}$ classes of functions from Definition 11 of the MHE section. Furthermore, any of the functions in these classes defined here will be supposed as not related to the ones from the MHE section.

To assess the nominal stability of the NMPC, it suffices to show that the objective function using its associated feedback law is a Lyapunov function.

Definition 17 (Lyapunov Function)**.** *A function $V : \mathbb{R}^n \to \mathbb{R}$ is denominated Lyapunov function for system* (4.17) *with control law $u(k) = h_u(x(k))$, if there exists functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot) \in \mathcal{K}$, such that for every $x \in \mathbb{X}$*

$$\begin{aligned} V(x) &\geq \alpha_1(\|x\|), \\ V(x) &\leq \alpha_2(\|x\|), \\ \Delta V(x) = V(f_u(x, h_u(x))) - V(x) &\leq -\alpha_3(\|x\|). \end{aligned} \tag{4.20}$$

Then, the assumptions required for the nominal stability are related to the properties of the terminal and stage-cost functions of (4.19); these being decaying and bounded respectively, when the closed-loop system is considered. This is the way, in which it can be proved that in the absence of disturbance, the closed-loop system generates a Lyapunov function.

Assumption 3 (Nominal Stability Assumptions)**.** *Consider the following requirements.*

1. *The terminal cost $\varphi_f(z)$ is strictly positive for all $z \in \mathbb{X} \setminus \{0\}$*
2. *There exists a local control law $u = h_f(z)$ such that $f_u(z, h_f(z)) \in \mathbb{X}_f$, and*

$$\varphi_f(f_u(z, h_f(z)) - \varphi_f(z)) \leq -L(z, h_f(z)) \quad \forall z \in \mathbb{X}_f.$$

3. *For $\underline{\alpha}_x(\cdot), \alpha_x(\cdot) \in \mathcal{K}$; the stage cost $L(z, h_f(z))$ satisfies*

$$\underline{\alpha}_x(\|z\|) \leq L(z, v) \leq \alpha_x(\|z\|).$$

Theorem 4 (Nominal Stability of NMPC)**.** *Consider problem* (4.19)*, with its associated control law $u(k) = h_u(x(k))$, and suppose that Assumptions 3 hold. Then, its objective $J_N(\mathbf{x})$ is a Lyapunov function.*

Proof. As given in [17]. □

Furthermore, to extend the stability proofs to the case of the system with disturbances, the definition of Input-to-State Stability (ISS) are required; which is related to the property of a system to generate state sequences that are increasingly bounded by their respective disturbances. Then, the respective ISS-Lyapunov counterpart adds a term to account for the disturbances.

Definition 18 (Input-to-State Stability (ISS))**.** *Consider the system of* (4.15) *with* $u(k) = h_u(x(k))$*, and* $x(0) = x_0$ *for* $x_0 \in \mathbb{X}$*. The system is said to be Input-to-State Stable (ISS), if there exists functions* $\beta(\cdot, k) \in \mathcal{KL}$*, and* $\gamma(\cdot) \in \mathcal{K}$ *such that for all* $k \in \mathbb{Z}_{\geq 0}$*, and* $w(k) \in \mathbb{W}$*, the following holds*

$$\|x(k)\| \leq \beta(\|x_0\|, k) + \gamma(\|w(k)\|). \tag{4.21}$$

Definition 19 (ISS-Lyapunov Function)**.** *A function* $V : \mathbb{R}^n \to \mathbb{R}$ *is denominated ISS-Lyapunov function for system* (4.15) *with control law* $u(k) = h_u(x(k))$*, if there exists functions* $\alpha_1, \alpha_2, \alpha_3, \sigma \in \mathcal{K}$*, such that for every* $x \in \mathbb{X}$*,* $w \in \mathbb{W}$

$$\begin{aligned} V(x) &\geq \alpha_1(\|x\|), \\ V(x) &\leq \alpha_2(\|x\|), \\ \Delta V(x) = V(f_w(x, h_u(x), w)) - V(x) &\leq -\alpha_3(\|x\|) + \sigma(\|w\|). \end{aligned} \tag{4.22}$$

To account for the mismatch between the objective functions of the nominal and disturbed models, the following term is defined

$$\epsilon(x(k+1)) \coloneqq J_N(x(k+1)) - J_N(z(k+1)), \tag{4.23}$$

it follows that this term is bounded by the mismatch term of equation (4.16); and this is described in the following assumption

Assumption 4 (Upper bound on mismatch)**.**

$$|\epsilon(x(k+1))| \leq K_J \|g_w(x(k), u(k), w(k)).\|$$

Further assumptions are made regarding the bounding of the mismatch function of (4.4), and the stage-cost under disturbances. With these it follows that the system is ISS.

Assumption 5 (Robust Stability Assumptions)**.** *For a nominal control law $u = h_u^{id}(x)$ and for all $k \in \mathbb{Z}_{\geq 0}$:*

1. *There exists functions $\alpha_g(\cdot), \sigma_g(\cdot) \in \mathcal{K}$ such that*

$$\|g_w(x(k), u(k), w(k))\| \leq \alpha_g(\|x(k)\|) + \sigma_g(\|w(k)\|).$$

2. *There exists a function $\alpha_4(\cdot) \in \mathcal{K}$, and a positive constant M such that*

$$-L(x(k), u(k)) + M(\alpha_g(\|x(k)\|) + \sigma_g(\|w(k)\|)) \leq -\alpha_4(\|x(k)\|) + \sigma_{g1}(\|w(k)\|).$$

Theorem 5 (Robust ISS Stability of NMPC [17])**.** *Consider problem* (4.19)*, with its associated control law $u(k) = h_u(x(k))$, and suppose that Assumptions 3−5 hold. Moreover, let $M \geq K_J$. Then, its objective $J_N(\mathbf{x})$ is a ISS-Lyapunov function.*

Proof. As shown in [17]. □

This implies that by attaining certain properties of the mismatch term, the closed loop system will be robust and stable. In practical terms, the terminal and stage-cost are defined in terms of quadratic functions with positive definite matrices $P, Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u}$, i.e.

$$\begin{aligned} \varphi_f(z) &\coloneqq z^T P z, \\ L_i(z_i, \upsilon_i) &\coloneqq z_i^T Q_i z_i + \upsilon_i^T R_i \upsilon_i. \end{aligned} \tag{4.24}$$

These terms form the NMPC problem for a particular state of the plant $x(k)$,

$$
\begin{aligned}
\min_{\boldsymbol{\upsilon}_k} \; & J_N\left(\mathbf{z}_k\right) \coloneqq z_{N|k}^T P z_{N|k} + \sum_{i=0}^{N-1} \left[z_{i|k}^T Q_i z_{i|k} + \upsilon_{i|k}^T R_i \upsilon_{i|k} \right] \\
\text{s.t.} \quad & z_{l+1|k} = f_u\left(z_{l|k}, \upsilon_{l|k}\right) \quad \forall l \in \{0, \ldots, N-1\}, \\
& z_{0|k} = x\left(k\right), \\
& z_{l+1|k} \in \mathbb{X} \quad l \in \{0, \ldots, N-1\}, z_{N|k} \in \mathbb{X}_f, \\
& \upsilon_{l+1|k} \in \mathbb{U} \quad l \in \{0, \ldots, N-1\}.
\end{aligned}
\tag{4.25}
$$

The remaining issue is the computational delays incurred by the solution of the NLPs for MHE and NMPC. One way to address this is using the underlying properties of NLPs. In particular, the properties with respect to parametric sensitivity.

4.5 Advanced-Step: MHE & NMPC

To deal with the computational delays of the solution of the NLPs associated with MHE and NMPC it is possible to partition the computations into online and background; moreover it is desired to minimize the amount of time online, so that the control action is computed close to instantaneously. To do this, two main ideas have to be discussed, i.e. the parametric nature of these problems and the predictions in the background.

Previously, in section 2.5, it is mentioned that any parametric information within the NLP has to be set before the solution procedure. Moreover this parametric information influences the solution in a specific way. And this functional dependency can be obtained in an analytical form, as long as it satisfies certain assumptions about the first and second order information of the problem, and the existence of a neighborhood for which the active set remains constant. This is useful to make approximations of solutions for which the values of the set of parameter has changed; in other words an initial solution can be further analyzed to approximate perturbed solutions. In the context of MHE and NMPC, such parametric nature is not ambiguous. Let us denote the MHE problem at time k as

$\mathcal{E}\left(y\left(k-\mathcal{N}\right),\ldots,y\left(k\right),\Pi_{k-\mathcal{N}|k-1},x_{k-\mathcal{N}|k-1}\right)$, where the terms inside the parentheses are the set of *fixed* information, i.e. parameters, from which it is assumed that $y\left(k\right)$ is the most relevant.For NMPC, the problem is denoted $\mathcal{C}\left(x\left(k\right)\right)$, where the only parametric information is the set of states from the process. Furthermore, let the parameter vectors be defined as follows $p_k^{\text{mhe}} \coloneqq y\left(k\right)$ and $p_k^{\text{nmpc}} \coloneqq x\left(k\right)$.

Suppose one wants to solve problems (4.14) and (4.25) *beforehand* (any time immediately before k but after $k-1$), then the pieces of unknown information at this point are given by p_k^{mhe} and p_k^{nmpc}. One way to generate an approximation of that information is by using predictions given the state and input vectors for $k-1$ and the nominal model of the plant. E.g.

$$\begin{aligned} z_{k|k-1} &= f_u\left(x\left(k-1\right),u\left(k-1\right)\right), \\ y_{k|k-1} &= h\left(z_{k|k-1}\right), \end{aligned} \tag{4.26}$$

and set the respective parameter vectors to the value of the predictions $p_{k|k-1}^{\text{mhe}} \leftarrow y_{k|k-1}$, and $p_{k|k-1}^{\text{nmpc}} \leftarrow z_{k|k-1}$. The NLPs can be attempted to be solved for these new values by the time the process evolves to k. Then, the relevant sensitivity information can be computed, i.e. the sensitivity matrices from (2.40), $S_{k|k-1}$. The aggregate of all these operations is said to be the background calculations, i.e. everyone by the time the process reaches time k. At time k, the real information is revealed and one can set $p_{k|k}^{\text{mhe}}$, and $p_{k|k}^{\text{nmpc}}$ to the real measurement and state[1] and perform an update given the sensitivity information and the realization of parameters in the form of equation (2.43). For example, let $s_{k|k}$ be the solution vector (primal-dual variables) of the NLP of interest, then the update can be done as follows

$$s_{k|k} = s_{k|k-1} + \left(S_{k|k-1}\right)^T\left(p_{k|k} - p_{k|k-1}\right) + o\left(\|p_{k|k} - p_{k|k-1}\|^2\right), \tag{4.27}$$

for the respective problems, i.e. either MHE or NMPC (e.g. $s_{k|k}^{\text{mhe}}$). The vectors are solution s and parameters p; and sensitivity matrices S. This last calculation constitutes the entirety

[1]Though the real state of the plant is not typically known as described in section 4.3, one can set this vector as the estimation from the solution of the MHE problem *after* it has been computed i.e. $p_{k|k}^{\text{nmpc}} \coloneqq z_{k|k}$.

of the *online* computations, and because it is based on matrix-vector multiplications, it is considered to be fairly inexpensive.

This scheme is denominated as *Advanced Step*, since it attempts to solve the NLPs ahead of time; furthermore, the MHE and NMPC schemes that use that concept are denoted as *asMHE* and *asNMPC*.

In practical sense, the sensitivity information and updates can be computed as part of the post-optimality analysis done in Section 2.7 with `k_aug` . In this context, we re-iterate the fact that the sensitivity update can be limited to a subset of the solution vector of the NLP. For this analysis let $r\left(p\right), r^{*} \in \mathbb{R}^{n_r} \subseteq \mathbb{R}^{n_s}$ denote the vector of relevant variables. Then, it can be noted that for the asMHE the relevant variables corresponds to updates of the estimated-state, i.e. $r^{\text{mhe}} \coloneqq z_{k|k}$; while for asNMPC it corresponds to updates of the predicted input $r^{\text{nmpc}} \coloneqq u_{k|k}$. It follows that for asMHE, the size of the measurement vector is larger than the state vector, and in consequence the dimension of the parameter vector of asMHE is less than the one of the relevant vector, i.e. $n_y \leq n_x \implies n_p \leq n_r$ and the update (2.53), i.e.

$$r^{\text{mhe}}_{k|k} = r^{\text{mhe}}_{k|k-1} + E^{T}_{\text{mhe}} S^{\text{mhe}}_{k|k-1} \Delta p^{\text{mhe}}_{k|k}, \tag{4.28}$$

is usually sufficient. Here n_y *offline* backsolves to assemble $S^{\text{mhe}}_{k|k-1}$ are required with only n_x *online* vector-vector products. On the other hand, for asNMPC typically the size of the state vector is larger than the input; thus $n_x > n_u \implies n_p > n_r$ and (2.55) should be used, this is

$$r^{\text{nmpc}}_{k|k} = r^{\text{nmpc}}_{k|k} - \left(S^{\text{nmpc}}_{r,k|k-1}\right)^{T} R^{\text{nmpc}}_{k|k-1} \Delta p^{\text{nmpc}}_{k|k}. \tag{4.29}$$

Here n_u *offline* backsolves are required for $S^{\text{mhe}}_{r,k|k-1}$ with only n_u *online* vector-vector products. On a final note, these update strategies are typical but not necessarily true all the time, i.e. it might be required to shift between each other depending the size of the measurement, state, and input vectors.

Finally, we present the step-by-step algorithms for the MHE and NMPC schemes for computing the feedback for a process. Two variations of the MHE-NMPC strategies are

presented:

- The idealized (no computational delay) version, i.e. ideal MHE (*iMHE*) and ideal NMPC (*iNMPC*).
- The advanced-step (sensitivity-based) version, i.e. advanced-step MHE (*asMHE*) and advanced-step NMPC (*asNMPC*).

The algorithms for the ideal step MHE and NMPC are described in Algorithms 4.1 (*iMHE*) and 4.3 (*iNMPC*) and their advanced step counterparts are described in Algorithms 4.2 (*asMHE*) and 4.4 (*asNMPC*). Moreover, a diagram of the relationship between the horizon and prediction window between the two problems is showed in Figure 4.1, and a block diagram showing the plant-controller layout is also shown in Figure 4.2.

For the advanced step algorithms 4.2 and 4.4, their particular differences are due to the steps involved in the solution of the NLPs (4.14) and (4.25) for the predicted measurement and state, and the computation of sensitivity matrices; which happen in the background blocks. It is clear that the advanced step methods require more computations than the ideal ones (mainly because of the sensitivity calculations); however most of these are carried out in background, in advance of the current sampling time. The actual online computations are related exclusively to the fast sensitivity updates (online blocks).

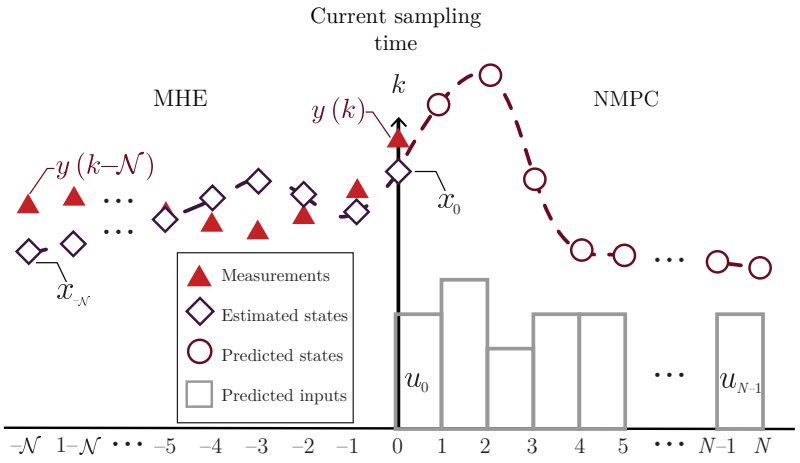

Figure 4.1: MHE-NMPC coupling

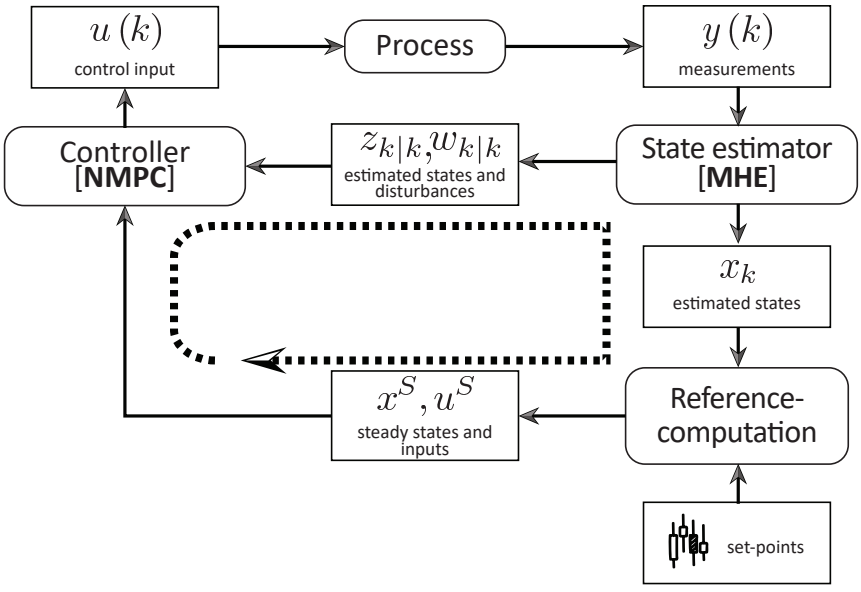

Figure 4.2: MHE-NMPC controller scheme

Data: At sampling time k, given horizon length $\mathcal{N}$, measurement and noise covariance matrices $\mathcal{R}$ and $\mathcal{Q}$ and a nominal "discrete" model $f(\cdot)$.

iM.1 At state k, with $\{y(k-\mathcal{N}), \dots y(k-1)\}$ measurement sequence, and prior information $z_{k-\mathcal{N}|k-1}$ and $\Pi_{k-\mathcal{N}|k-1}$;

iM.2 Get $y(k)$;

iM.3 Solve NLP (4.14) with the measurement sequence and prior information , $\mathcal{E}\left(y(k-\mathcal{N}), \dots y(k), \Pi_{k-\mathcal{N}|k-1}, z_{k-\mathcal{N}|k-1}\right)$;

iM.4 Compute covariance $\Pi_{k+1-\mathcal{N}|k}$, keep estimate $z_{k+1-\mathcal{N}|k}$;

iM.5 Set $x(k) \leftarrow z_{k|k}$;

iM.6 Finish current sampling time's computation, set $k \leftarrow k+1$;

iM.7 Shift horizon (i.e. ignore oldest data point). Go to iM.1 ;

Algorithm 4.1: Ideal Step MHE (*iMHE*)

Data: Given horizon length $\mathcal{N}$, measurement and noise covariance matrices $\mathcal{R}$ and $\mathcal{Q}$ and a "discrete" nominal model $f(\cdot)$.

Background computations

asM.1 Ready with $\{y(k-\mathcal{N}), \dots y(k-1)\}$ measurement sequence, prior information $z_{k-\mathcal{N}|k-1}$ and $\Pi_{k-\mathcal{N}|k-1}$, and state-input $\left(z_{k-1|k-1}, u(k-1)\right)$;

asM.2 *Predict measurement* $y_{k|k-1}$ with nominal model (4.26) using previous state-input tuplet $\left(z_{k-1|k-1}, u(k-1)\right)$;

asM.3 Solve NLP (4.14) with predicted measurement, $\mathcal{E}\left(y(k-\mathcal{N}), \dots y_{k|k-1}, \Pi_{-\mathcal{N}|k-1}, x_{-\mathcal{N}|k-1}\right)$;

asM.4 Compute sensitivity matrix, i.e. $E_x^T S_{k|k-1}^{\text{mhe}} = \left(\frac{dz}{dy}\right)_{k|k-1}$ with (2.51) ;

`/* Using optimal solution, with` E_x `defining the appropriate subset of` $\tilde{s}$ `*/`

asM.5 Compute covariance $\Pi_{k-\mathcal{N}|k-1}$, keep estimate $z_{k-\mathcal{N}|k-1}$;

Online computations

asM.6 At sampling time k ;

asM.7 Get "real" measurement from the plant $y(k)$;

asM.8 Using $\Delta p_{k|k}^{\text{mhe}} := (y(k) - y_{k|k-1})$, compute the updated state estimate $z_{k|k}$ from (4.28) ;

asM.9 Set $x(k) \leftarrow z_{k|k}$;

asM.10 Finish current sampling time's computation, $k \leftarrow k+1$;

asM.11 Shift MHE horizon. Go to asM.1 ;

`/* We assume it is possible to get` $u(k)$ `*/`

Algorithm 4.2: Advanced Step MHE (*asMHE*)

Data: Given steady states x^s and u^s, horizon length N, Q and R matrices, and a "discrete" nominal model $f(\cdot)$.

iN.1 At state k ;

iN.2 Get $x(k)$(Get $z_{k|k}$) ;

iN.3 Solve NLP (4.25) for curent (estimated)state, $\mathcal{C}(x(k))$ ($\mathcal{C}\left(z_{k|k}\right)$) ;

iN.4 Set $u(k) \leftarrow u_{k|k}$ and inject into the plant ;

iN.5 Finish current sampling time's computation, set $k \leftarrow k+1$;

iN.6 Go to iN.1 ;

Algorithm 4.3: Ideal Step NMPC (*iNMPC*)

Data: Given steady states x^s and u^s, horizon length N, Q and R matrices, and a "discrete" nominal model $f(\cdot)$.

Background computations

asN.1 Ready with state-input tuplet $\left(z_{k-1|k-1}, u(k-1)\right)$;

asN.2 *Predict state* $z_{k|k-1}$ for k with the model of (4.26) and $\left(z_{k-1|k-1}, u(k-1)\right)$;

asN.3 Solve NLP (4.25) for predicted state, $\mathcal{C}\left(z_{k|k-1}\right)$;

asN.4 Compute sensitivity matrix $S^{\text{nmpc}}_{r,k|k-1} = \left(\dfrac{du}{dz}\right)_{k|k-1}$, i.e. solve (2.54) ;

Online computations

asN.5 At sampling time k ;

asN.6 Get "real"(estimated) state from the plant $x(k)$($z_{k|k}$) ;

asN.7 Compute input with sensitivity $u_{k|k}$ with (4.29) ;

asN.8 Finish current sampling time's computation, set $k \leftarrow k+1$;

asN.9 Set $u(k) \leftarrow u_{k|k}$ into the plant. Go to asN.1 ;

Algorithm 4.4: Advanced Step NMPC (*asNMPC*)

These algorithms are designed to minimize the online computations; and their implementation should requires bookkeeping of the information being generated from the process and the subsequent updates. Furthermore, these algorithms were put into an unified framework that includes data-management and sensitivity computations. As described in the following Chapter.

Chapter 5

CAPRESE: Framework and Case studies

5.1 Introduction

In this chapter, an unified framework for Nonlinear Model Predictive Control and State-Estimation is presented. Moreover, problems with nontrivial dynamics are used as examples for control, in the context of CO_2 capture from power generation processes. It can be noted that the interest on simulating and controlling such processes is related to the concerns of potential adverse situations, that are originated with the release of combustion gases to the atmosphere, particularly CO_2. High levels of CO_2 in the atmosphere have been associated with global warming effects [44]; therefore there have been efforts to decrease CO_2 emissions by several countries. This can be seen particularly for power generation, as it is considered as the largest source of emissions globally [45]. As a consequence, post-combustion carbon capture techniques are attractive, because they can potentially be implemented in already existing sites, and they typically are more energy efficient than other methods [46].

Within the scope of power-plant operations, the application of amine-based technology can be useful to capture CO_2 considering its range of concentration and the widespread commercial applications of amines [47]. Nevertheless, CO_2 capture systems incur an increase in the price of production of electricity, so the optimization of CO_2 capture technology is essential. Furthermore, recent studies show that solid sorbents are attractive for removing CO_2 from gas streams, because of reduced energy requirements for regeneration and improved adsorption/desorption properties due to development of new materials [48]. Consequently, efficient gas-solid contact operation using fluidized beds is desir-

able [49] for several applications, including combustion, polymerization, gasification, etc. However, fundamental design and optimization of these systems requires advanced computational tools especially because of the complex hydrodynamics of the gas-solid flow [50].

Early work by Kunii and Levenspiel [51] led to an approximate fundamental model for Bubbling Fluidized Beds (BFB). In their model, the bed is effectively separated into regions on which the hydrodynamic behavior is described by empirical correlations that lead to workable designs. A more detailed model of the BFB for CO_2 capture, based on Kunii's assumptions, was described by Lee et al. [52]. Their model considers axial variations, detailed kinetics and thermodynamics, and it was implemented within an Equation Oriented (EO) modeling environment. Additionally, transient models of fluidized bed reactors have also been developed to aid in the creation of process control systems [53] [54]. Based on the work of Lee et al. [52], Modekurti et al. [55] developed and validated a dynamic version of the BFB model. This allowed various feedback control schemes to be compared. Most notably a Linear Model Predictive Controller (LMPC) showed better performance under disturbances than a regular PID controller.

In this context, LMPC has been shown to be relatively straightforward to implement and effective for feedback control. However, as the BFB system is inherently nonlinear and needs to operate to tight specifications over a range of input conditions and disturbances, LMPC might not have adequate performance [42]. Instead, control strategies that rely on nonlinear dynamics are needed. For this the optimization strategies discussed in the previous section are possible candidates, as their fundamental properties like robust stability are well established. Another issue that is generated from the implementation of fully nonlinear models are the computational delays associated with the solution of the nonlinear optimization problems. Nevertheless, it is also been proposed to incorporate parametric sensitivity analysis as a way to drastically to minimize the delays from the computations of the controller input. Furthermore, there have been recent studies for

CO_2 capture have addressed similar computational challenges, by means of model reduction as well as advanced-step nonlinear control algorithms. Yu et al. [56] compared several model reduction techniques for both temporal and spatial domains of the same BFB model for CO_2 capture introduced by Lee. Using collocation on finite elements for the spatial discretization leads to significant reduction in simulation time while providing accurate results. Further reduction by means of a (temporal) eigenvalue analysis reveals that the system states have large differences in time scales. In particular, the dynamics of the states associated with the gas phase are faster than the solid phase. This last result is significant because several differential states can be approximated as algebraic states (i.e., in quasi-steady state), thus simplifying the discretized structure in the NMPC and MHE optimization problems. Moreover, Yu and Biegler [57] recently extended this model reduction approach to economic NMPC, and demonstrated its effectiveness on a related BFB carbon capture system.

The aggregation of the concepts for modeling, design, and control of CO_2 capture processes have motivated the creation of a general modeling framework intrinsically linked to optimization, along with features to address uncertainties in modeling, operations and the market. Consequently, NETL's Institute for the Design of Advanced Energy Systems (IDAES) [58] is developing new open source tools modeling based on Python and Pyomo [59] that enable optimization with efficient and robust state-of-art solvers. Pyomo provides a flexible environment for optimization with inherent object-oriented aspects, along with special tools for automatic discretization of general Differential-Algebraic Equation (DAE) optimization. These advances also lead to a generic implementation of NLP strategies for state estimation and control.

In this chapter, we introduce *CAPRESE* (Control and Adaptation with Predictive Sensitivity Enhancements), a nonlinear optimization-based framework in Python for sensitivity-based NMPC and MHE strategies for a general Pyomo DAE model. Following the framwork, we introduce the BFB model for CO_2 capture. Then, the CAPRESE implementations

for two case studies, an initial benchmark distillation case study example and CO_2 capture through BFB control, are presented in the last section.

5.2 Control Adaptation with Predictive Sensitivity Enhancement

The Control Adaptation with Predictive Sensitivity Enhancement (CAPRESE) framework was created with the intention of streamline the development of an MHE-NMPC controller strategy for a generic process model. The overall estimation and control strategy and algorithmic framework for CAPRESE is described in Section 4.5. CAPRESE takes into account the difficulties that result from using Nonlinear Models vis-à-vis the potential computational delays, with the incorporation of parametric sensitivity strategies that minimize the amount of online computations to only consider matrix-vector multiplications. Moreover, it relais on state-of-art NLP solvers like IPOPT to carry out the solution of the resulting NLP prolems; and then it takes advantage of the `k_aug` framework to handle the required sensitivity computations.

The CAPRESE environment is founded on the Python language, and Pyomo as the modeling framework which handles the optimization related objects that are not directly involved with the solution algorithms. As a result, it is also possible to enable automatic discretization of differential algebraic models by means of the current dynamic optimization tools within. These and the main classes within CAPRESE are summarized in Figure 5.1.

The model used by CAPRESE is required to be written with the automatic discretization syntax expected by `pyomo.dae` [60]. Moreover, the user is expected to provide an initial guess for the steady-state model. This initial guess is essential, as it is taken as the reference point for the solution of most subproblems inside the framework.

In order to create a controller object, all the state, input and measurement information of the model must first be declared explicitly and then passed to the object constructor. For example, a simple CSTR [61] requires the declaration of states, controls, and bounds shown

Listing 5.1: Declaration of model states, measurements, etc.

```
states = ["Ca", "T", "Tj"]  #: Concentration, Temperature, Cooling
measurements = ['T']  #: Reactor temperature
controls = ["u1"]  #: Cooling inlet Temperature
u_bounds = {"u1": (200, 1000)}
state_bounds = {"Ca": (0.0, None),
                "T": (2.0E+02, None),
                "Tj": (2.0E+02, None)}
ref_state = {("Ca", (0,)): 0.010}
```

in Listing 5.2.

These lists and dictionaries are then passed to the MHE-NMPC object with the model and the desired sampling time, as shown in Listing 5.2.

Listing 5.2: MHE - NMPC object creation.

```
e = MheGen_DAE(mod,
               t_sample,
               states,
               controls,
               states,
               measurements,
               u_bounds=u_bounds,
               ref_state=ref_state,
               var_bounds=state_bounds)
```

At this point, when an initial guess for the reference steady-state problem is given, the CAPRESE framework computes the subsequent initialization points for the larger NLP problems to be solved. This is a key feature of CAPRESE, since the solution of sequences of NLPs can not be done effectively if no appropriate initial guesses are given. This also implies that a careful data collection and communication has to be done, because there are

several models working at once to generate the controller input computation.

As a whole, CAPRESE attempts to unified the framerowks for model based control with parametric sensitivity. It's ongoing development is focussed on creating an user friendly approach for modeling while keeping the elements of the underlying Differential Algebraic Models clearly laid out. In the next section, we discuss example case studies of the CAPRESE framework, which have vastly different levels of complexity; nevertheless the framework is able to compute the controller inputs while attaining reasonable performance.

Case Study Results

To demonstrate the CAPRESE framework with both ideal and advanced step strategies, we consider two case studies. The first tests the behavior of the advanced-step strategies on a benchmark case study that involves a high-purity binary distillation column. This is a typical process application that shows how this MHE-NMPC framework performs with large, first principle DAE models. The second case study considers the CO_2 capture process with the BFB model described in the second section. We note that for the ideal strategies both NLPs (4.14) and (4.25) need to solved sequentially on-line. In contrast the advanced step strategy solves NLP problems (4.14) and (4.25) strategy simultaneously in background between sampling points. Also, sensitivity-based updates in Algorithms $asMHE$ and $asNMPC$ are performed with negligible computation time. For the case studies the specified sampling times (60s for Case Study 1, and 120s for Case Study 2) provided ample time to solve these NLPs, as shown in Table 1. For applications where additional CPU time may be needed, the advanced step approach has also been extended to background solutions over multiple sampling times [18].

5.2.1 Distillation column benchmark case study.

For this case study, the distillation model is a dynamic MESH model presented in [62] and later modified in [63] to index 1. The model considers the separation of a binary mixture of methanol and n-propanol, with tray-by-tray mass and energy balances, Raoult's Law for thermodynamic equilibrium, and liquid flows from the trays determined by the Francis weir formula. For a column with 40 trays, the DAE model has 84 differential equations. It is assumed that the temperatures are measured for each tray and they have diagonal covariance matrices with $0.0625I$. As shown in Figure 5.2, the control problem has set-points on the temperatures of tray 14 and 28 with diagonal weight matrices $q_{ii} = 1 \times 10^{-5}$ and 1.0, respectively. The manipulated variables are the reboiler heat duty and the reflux ratio. Here, $n_u = 2$ (inputs), $n_x = 84$ (states), $n_y = 40$ (measurements), and because $n_y \leq n_x$, (4.28) is used in Step 8 in Algorithm 4.2.

The continuous time DAE model is transformed into a discrete time model using 3 point Radau collocation. MHE and NMPC were run with the advanced step and the ideal strategies (assuming no computational delay), with ten-step horizons ($N = \mathcal{N} = 10$) and 60s sampling times. These problems have 20672 variables and 19792 equations in the case of MHE, and 19036 variables and 19016 equations for NMPC. Finally, random noise was introduced on all the states at the plant with variance $\sigma = 10^{-4}$. The results for this case study are shown on Figure 5.2.

For state estimation the maximum relative error considers all the states of the problem. It can be seen, that these errors remain small for most of the simulation. However, at the setpoint change at sampling time 350, one of the states almost loses observability and the relative error grows very quickly. Nevertheless, the advanced-step MHE follows similar behavior as ideal MHE with only a small difference in performance. NMPC experiences reasonably good performance, even under the presence of noise. Moreover, the advanced-step NMPC follows the ideal case almost exactly. The average timings for this problem are displayed in Table 5.1. Because the on-line computations require much less time than the

offline costs, the total computational delay for both asMHE and asNMPC amounts to only 1.19 CPU seconds. This is considerably less than the 57.29 CPU seconds required for the ideal strategies. In future work we plan to further reduce the offline computational cost by coupling the sensitivity matrix step with the reduced Hessian calculation.

5.2.2 Bubbling Fluidized Beds

The BFB model used here stems from the derivation presented in [56]. This model contains mass and heat balances for the gas and solid phase inside the emulsion, cloud wake and bubble regions shown in Figure 5.3. The modeling equations comprise a total of twenty PDEs with time t and axial coordinate x as independent variables and several algebraic equations, which describe the thermodynamics, kinetics, and hydrodynamics of the fluidized bed. The main differences from the model of [56] are the inclusion of a momentum balance for the gas at the bubble region, i.e.:

$$\frac{\partial m_g}{\partial t} = -2v_g\frac{\partial v_g}{\partial x}\sum_j M_j c_{b,j} - v_g^2\frac{\partial \rho_g}{\partial x} - \mu_g\frac{\partial^2 v_g}{\partial x^2} - \frac{\partial P}{\partial x} - (1-e)\,\rho_s g_c \tag{5.1}$$

and the expansion of the PDEs of the gas phase at the bubble region to separate the v_g partial derivative terms [1]. The resulting system of Partial Differential Algebraic Equations (PDAEs) can be written concisely as follows:

$$\frac{\partial \tilde{\zeta}}{\partial t} = \frac{\partial \tilde{\zeta}}{\partial x} + \tilde{F}\left(\tilde{\zeta}, \tilde{\chi}\right), \tag{5.2a}$$

$$0 = \tilde{G}\left(\tilde{\zeta}, \tilde{\chi}\right), \tag{5.2b}$$

$$\tilde{\zeta}(0,x) = \tilde{\mathbf{z}}_0, \tag{5.2c}$$

where the variables $\tilde{\zeta}\,(t,x) \in \mathbb{R}^{n_{\tilde{\zeta}}}$ and $\tilde{\chi}\,(t,x) \in \mathbb{R}^{n_{\tilde{\chi}}}$ represent the set of differential and algebraic states from the model. Similarly $\tilde{F} : \mathbb{R}^{n_{\tilde{\zeta}}} \times \mathbb{R}^{n_{\tilde{\chi}}} \rightarrow: \mathbb{R}^{n_{\tilde{\zeta}}}$ and $\tilde{G} : \mathbb{R}^{n_{\tilde{\zeta}}} \times \mathbb{R}^{n_{\tilde{\chi}}} \rightarrow: \mathbb{R}^{n_{\tilde{\chi}}}$ represent the right hand side of the differential and algebraic equations. Finally, (5.2c) represents the initial condition of the system. Note that the spatial boundary conditions for the system of equations (5.2) are contained in the set of algebraic equations (5.2b).

[1] For the detailed model, please refer to the Appendix (Section .3) of this manuscript.

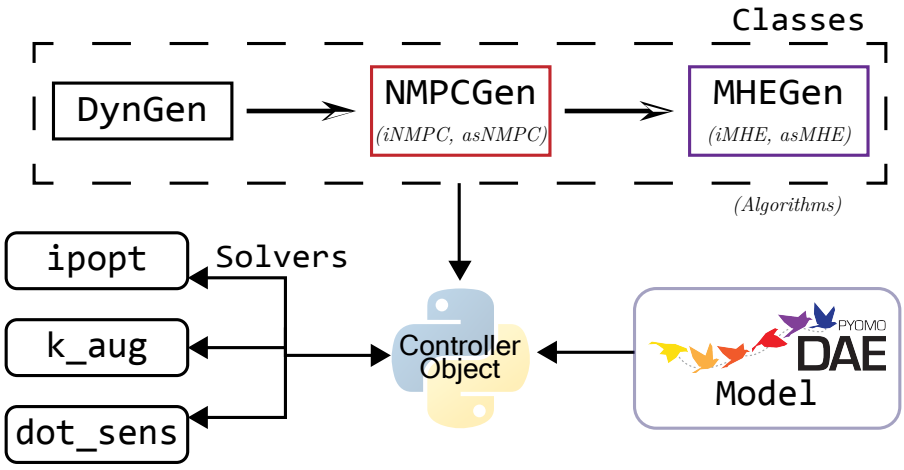

Figure 5.1: Diagram of MHE-NMPC classes and methods in CAPRESE.

		Distillation		BFB CO_2	
		NMPC	MHE	NMPC	MHE
Offline	**NLP (Ipopt)**	**11.00**	**8.50**	**6.83**	**9.45**
	Red. Hessian (k_aug)	-	**3.54**	-	**13.63**
	Sens. Matrix (k_aug)	1.87	2.58	12.49	7.94
Online	*Sens. step (dot_sens)*	*0.10*	*0.49*	*0.13*	*0.52*

Table 5.1: Timings for the optimization and sensitivity in CPU seconds, using Ipopt 3.12 and Intel i7-6700 CPUs. The bold letters are associated with the ideal strategies required timings

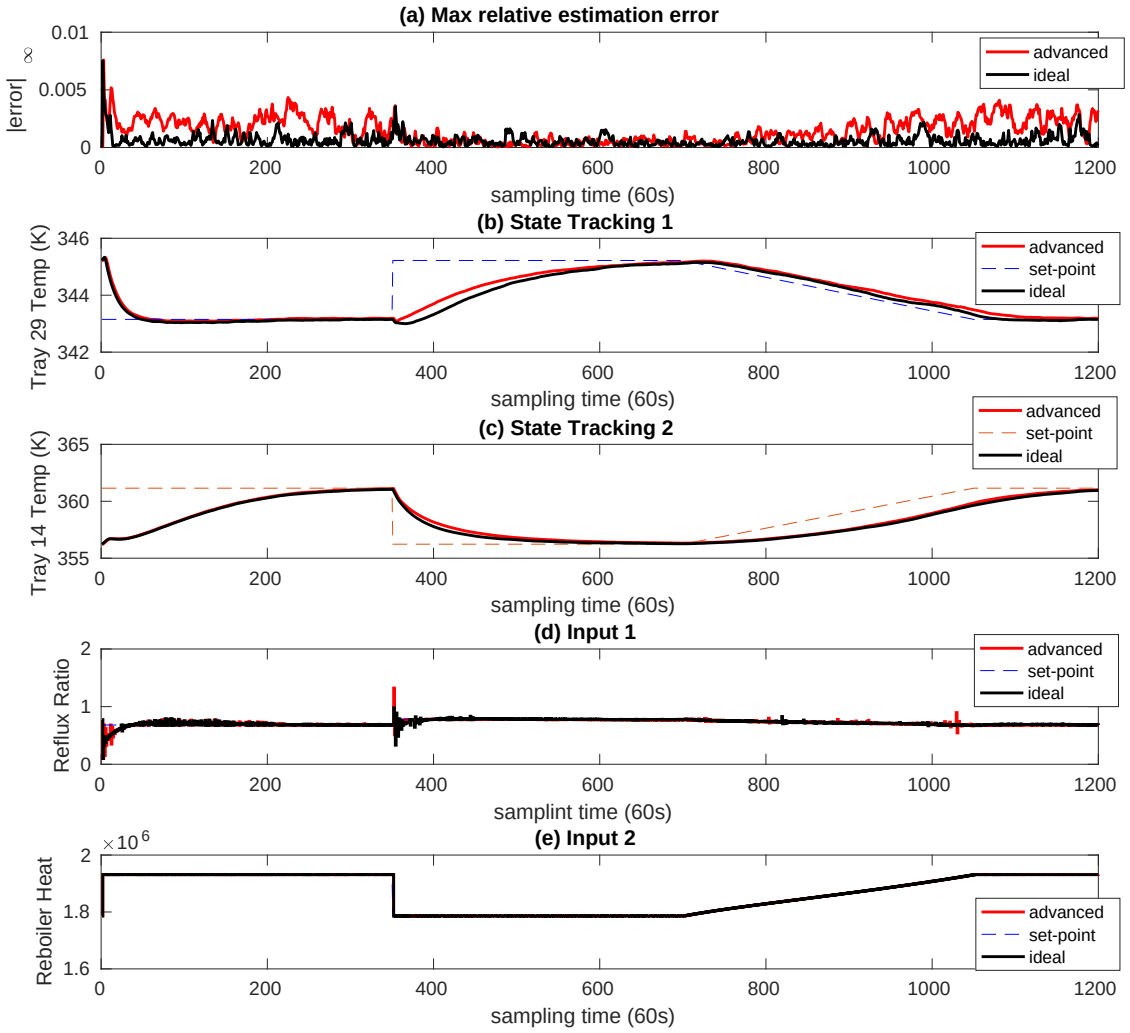

Figure 5.2: Control input and state-tracking results for distillation case study.

For the optimization problems defined by the NMPC and MHE, the simultaneous solution approach requires a fully discretized form of the model. Therefore, by first discretizing equation (5.2) in space, the system collapses to a DAE system in time, i.e.:

$$\frac{d\zeta}{dt} = \mathbf{F}(\zeta, \upsilon) \tag{5.3a}$$

$$0 = \mathbf{G}(\zeta, \upsilon), \tag{5.3b}$$

$$\zeta(0) = \mathbf{z}_0, \tag{5.3c}$$

where the new variables $\zeta(t)$ and $q(t)$ correspond to the spatially discretized versions of the original differential and algebraic equations (e.g. $\zeta(t) = \tilde{\zeta}_{ij}(t)$ and $\chi(t) = \tilde{\chi}_{ij}(t)$ for the i-th finite element and j-th collocation point in space), the functions $\mathbf{F}(\cdot)$ and $\mathbf{G}(\cdot)$ map the newly discretized variables into corresponding right-hand-sides. Also, $\mathbf{G}(\cdot)$ contains additional equations from the discretization (e.g., spatial finite-element continuity for orthogonal collocation). Lastly, $\mathbf{z}_0$ is the initial condition for the spatially discretized model. Subsequently, an additional temporal discretization of equation (5.3) results in a fully algebraic representation of the DAE system, which has a larger number of equations and variables.

Yu and Biegler [56] considered different time-scales among the differential states of the BFB. In particular, the fast states of the model were associated with the states in the gas phase. This was due to a combination of bubble phase hydrodynamics and the kinetics associated with physisorption. Further identification of the time-scales enables a reduction of the model by creating a partition of slow and fast differential states $\zeta^T = \left[\zeta_s^T \ \zeta_f^T\right]$. Then the time derivative term of the fast differential states ($d\zeta_f/dt$) in the system of equation (5.3) is set to zero, ζ_f become algebraic states and equation (5.3) becomes:

$$\frac{\zeta_s}{dt} = \mathbf{F}(\zeta_s, \zeta_f, \chi) \tag{5.4a}$$

$$0 = \mathbf{G}(\zeta_s, \zeta_f, \chi), \tag{5.4b}$$

$$\zeta_s(0) = \hat{\mathbf{z}}_0, \tag{5.4c}$$

where $\hat{\mathbf{z}}_0$ contains the initial conditions of the reduced (slow) set of states ζ_s. Assuming that the matrix $\begin{bmatrix}\nabla^T_{\zeta_f}\mathbf{G} & \nabla_\chi \mathbf{G}^T\end{bmatrix}$ is non-singular, the system of eq. (5.4) will approximate the dynamics of eq. (5.3). For this work, we tested the assumption of fast differential states based on the observations from [56], in order to have the most robust working model.

BFB CO_2 Capture: Results

For this case study, the BFB reactor was used to capture CO_2 for different set-points. The model for this problem posed several difficulties during the optimization stages of the control framework. The is mostly due to the ill-conditioning of the working matrices as the differential states of this model have significant differences in time scales. Approximating some of the faster differential states as algebraic states helps to collapse these time scales and create a simpler model at the possible expense of modeling accuracy. Inspired by the observed results of [56], in which the states associated with the molar balances in the gas region were found to have separate time-scales, two reduced models were created:

- *rm1* converts the differential states associated with molar fluxes in the gas phase to algebraic states.
- *rm2* converts the molar fluxes and also the enthalpy and momentum fluxes of the gas-bubble region to algebraic states.

Model *rm1* achieves a 7.83% reduction of model size with respect to the original model. A comparison with respect to the original model was analyzed by simulations in which large step changes of the input over time were made at the 200th and 400th sampling time respectively. At the given sampling times, the maximum relative errors between *rm1* and the original model are shown in Figure 5.4. We observe that the relative error remains less than 5×10^{-3} throughout the simulation.

In [56] it was observed that the dynamics of the energy and momentum balances for the gas-bubble region are also fast, so these were also converted to algebraic states in *rm2*. Model *rm2* achieves a model reduction of 9.57% with respect to the original model. As

shown in Figure 5.4 it also displays a relative error as high as 6.5×10^{-3}. In spite of the slight error increase, *rm2* yielded the least amount of failures during optimization of all the tested models, which made it the selected working model for control and state estimation.

Once the reduced model was identified, an closed-loop MHE-NMPC study was set up and solved. In this problem the CO_2 capture percentage is used as reference to compute steady-state trajectories of all differential states, such that full state tracking is performed by the NMPC. For this, the amount of gas flowing into the bed was manipulated, and for the state-estimation problem, we assume that the internal axial temperature and gas velocity can be measured. Random noise was added to all states in the plant with relative variance $\sigma = 10^{-4}$. Also the corresponding covariance matrices for MHE are shown in a table in Supplemental Information. The state tracking problem was solved using both the advanced and the ideal strategies for MHE and NMPC, with ten-step horizons ($N = \mathcal{N} = 10$) and 120s sampling times. These problems were fully discretized; Legendre collocation was used in space with 5 finite elements and 3 collocation points, and Radau collocation was used in time with 10 finite elements and 1 collocation point. The resulting NLPs have 32810 variables and 31300 equations for MHE, and 31160 variables and 31150 equations for NMPC. Here, $n_u = 2$ (inputs), $n_x = 150$ (states), $n_y = 31$ (measurements), and because $n_y \leq n_x$, (4.28) is used in Step 8 in Algorithm 4.2. The results and statistics for the framework with the BFB for CO_2 capture are displayed in Figure 5.5 and Table 5.1 respectively.

For state estimation the advanced-step MHE experiences larger amplitude error spikes than ideal MHE. Nevertheless, these differences do not affect the performance of the overall controller, as the relative estimation error remains bounded during the simulation and the noise on the states was handled effectively. The results show that both the ideal and advanced step NMPC track the input with little difference in performance. Moreover, the advanced-step schemes are able to reach the set-points with only small computational delays. As displayed in Table 5.1, only 0.52 CPU seconds are required for MHE and 0.13 CPU seconds for NMPC, with an overall 0.65 CPU seconds online cost. This is considerably less

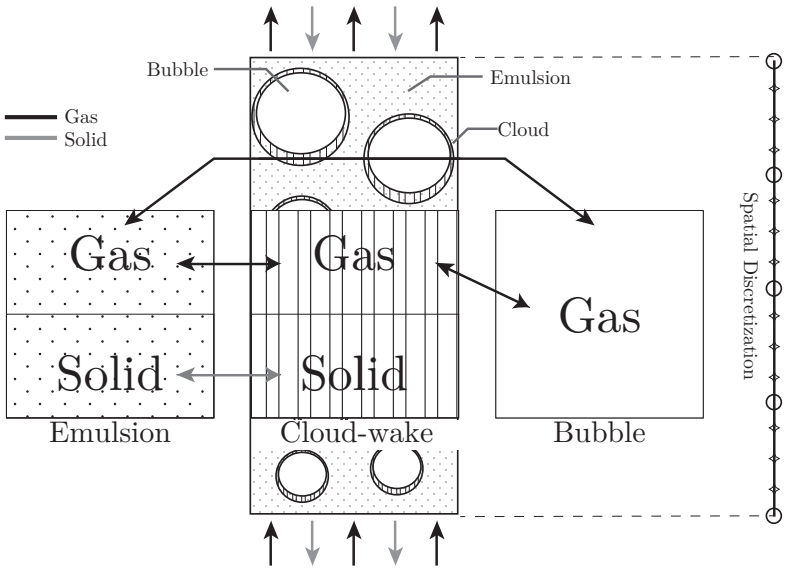

Figure 5.3: BFB reactor, regions and discretization

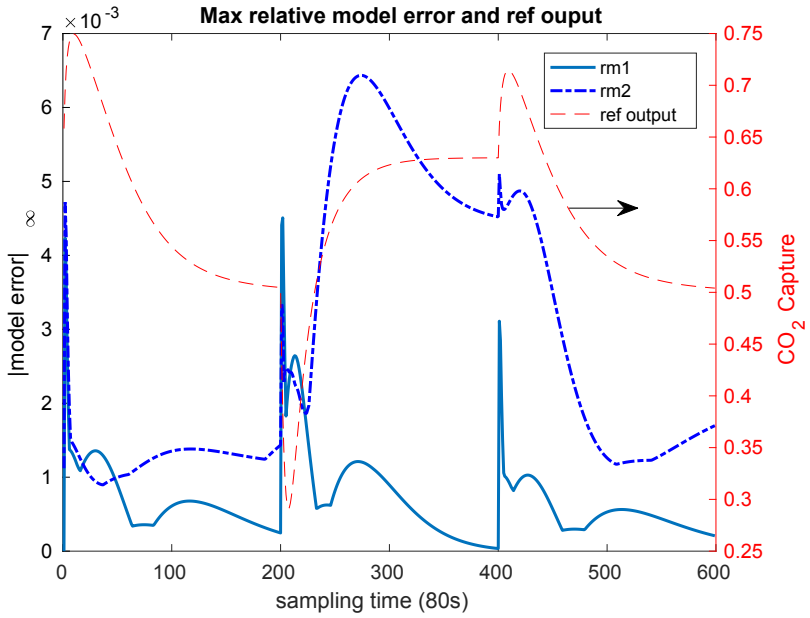

Figure 5.4: Approximation errors from model reduction: maximum relative state error of the two proposed reduced models (right) and output response of the reference model (red line left axis, CO_2 capture).

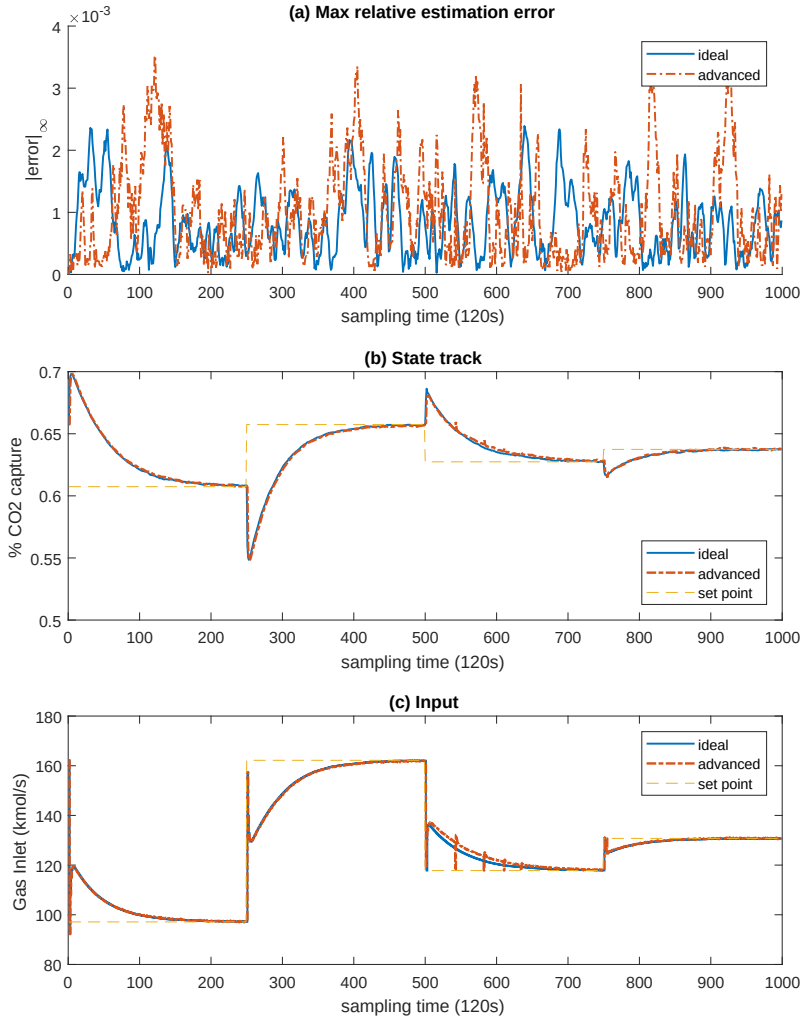

Figure 5.5: Max. relative estimation error, control input, state-tracking results.

than the 29.91 CPU seconds required for the ideal strategies. This case study suggests that sensitivity-based MHE and NMPC can be very efficient in significantly reducing computational delay, with little performance loss.

5.3 Conclusions

This study demonstrates a general framework for the implementation of Nonlinear Model Predictive Control (NMPC) and Moving Horizon Estimation (MHE) using Pyomo for a detailed Bubbling Fluidized Bed (BFB) reactor for CO_2 capture. The BFB system is characterized by a large PDAE model. Based on the observations of Yu et al. [56], it contains several states with different time-scales. Thus, our proposed estimation/control framework was applied and tested with a modified BFB model. Furthermore, the optimization-based framework implements computations with minimal time delays, by means of the advanced-step strategies [17]. This framework maintains MHE+NMPC delays to less than a few CPU seconds with a negligible performance loss for the controller and state-estimator, compared to the ideal strategies. Future work will take into account strategies for sensitivity computations with changing active sets; and in increasing the performance within the interior point optimizer, using parallel decomposition of the factorization steps.

Chapter 6

Conclusion

6.1 Summary and Contributions

In this dissertation, two main topics are covered. The first one is concerned with a strategy within the NLP solver IPOPT, to handle problems that do not satisfy Constraint Qualifications (CQs). In this context, CQs are define the well posedness of the problem. And, though there exist several kinds of CQs, the most relevant one is concerned with the linear independence of the linearization of the set of equality and active inequality (bounds) constraints (LICQ). If such regularity condition is not satisfied, optimality as given by the set of first-order optimality conditions can not be asserted. Moreover, in the practical sense, in IPOPT global convergence can not be guaranteed, and any attempts to solve such problem has a high chance of failure. This is the reason why IPOPT has built-in strategies that try to overcome such possible situations in the local sense, i.e. the *inertia* correction. In this strategy, perturbations are done to the iteration matrix such that the number of positive, negative and zero eigenvalues (i.e. the inertia) is changed in an attempt to achieve the specific value of it that guarantees the satisfaction of the LICQ. Despite this, there might be instances in which such perturbations are effective; and, it can also be the case that the gradients are globally dependent, i.e. dependent at all points of the search space. Thus this strategy might not be sufficient by itself. To overcome this situations, the ℓ_1-exact penalty-barrier strategy in IPOPT was introduced. In this strategy, the equality constraints are penalized into the objective function with a penalty parameter. This form of the problem has a guarantee of attaining a stationary points of the original barrier problem, as long as an sufficiently large value of the penalty parameter achieved [21]. Moreover, after re-

formulating the exact penalty function to obtain a differentiable version of the problem; two new vectors of variables are obtained. This problem has the remarkable guarantee of satisfying the LICQ for all points for which the penalty variables are positive. Therefore the complexity of the lack of regularity is traded for a problem with more variables and a penalty parameter to be determined, but has a guaranteed constraint qualification. A strategy to update the penalty parameter based on local curvature and linearized infeasibility was presented, and the overall ℓ_1-exact penalty-barrier formulation was implemented into IPOPT 3.12.9. Then, this strategy was tested with the CUTEr test set, for which degeneracies were introduced. And a test set that contains Mathematical Programs with Complementarity Constraints (MPCCs). From the results it is possible to conclude to main points:

a. In order to achieve optimality, it is critical to have an adaptive value of the penalty parameter.

b. The presented strategies here grant an increase of robustness of the overall algorithm, in other words more degenerate problems were solved as opposed to normal IPOPT.

c. In the case of MPCCs, the ℓ_1-exact penalty-barrier strategies perform *better* than normal IPOPT.

In addition to these results, we note that there are still case studies in which the penalty formulation failed. These can be summarized as follows: Inadequate value of the penalty parameter, unbounded value of the penalty parameter; and for a small value of the barrier parameter, the existence of an excess of *nearly* active bounds, i.e. violation of the MFCQ. Nevertheless, we highlight that the main contribution of this part we have presented a strategy for interior-point that is able to attain better robustness properties, especially for problems that do not attain certain CQs.

In the second part of this dissertation, we discuss the properties of nonlinear model based state-estimation and control, these are the Moving Horizon Estimator (MHE) and Nonlinear Model Predictive Control (NMPC); particularly with the intention of creating an

unified software implementation for the design and testing of such controller. Moreover, we show that by making use of parametric sensitivity, it is possible to enable the online implementation of the controller that requires the solution fully nonlinear optimization problems. Thus, the Control Adaptation with Predictive Sensitivity Enhancement (CAPRESE) was created. CAPRESE is a software implementation that encapsulates the computations for control-estimation-sensitivity, using fully nonlinear optimization. This was done using a combination of Python and Pyomo from which the optimization objects that represent the MHE-NMPC tasks are created. This has the advantage that all the scripting and meta-algorithms can be kept in Python; and users can create models that can be automatically discretized by means of `pyomo.dae`. Moreover, we introduce the post-optimality software `k_aug` to handle computations of sensitivity matrices and Reduced Hessians, which are critical for the online updates of MHE-NMPC; and is integrated into CAPRESE to achieve a significant increase of performance, as opposed to the online solution of the underlying NLPs. Then, the CAPRESE framework is used to control two case studies. A benchmark distillation column model, which provides reasonable nonlinearities; and a Bubbling Fluidized Bed reactor for CO_2 capture, in which a large model with significant nonlinearities has to be controlled. The results are summarized as follows;

a. The combination of MHE and NMPC is able to produce a controller input that can handle cases with significant noise.

b. Fully nonlinear optimization strategies can be enabled online by means of parametric sensitivity with an almost insignificant impact of performance of the estimation and control.

Finally, we can state that the main contribution of this section is that the foundation has been laid out for an unified implementation of MHE-NMPC for state-estimation and control that can handle large-scale nonlinear model in online applications.

6.2 Recommendations for future work

IPOPT for failure of CQs.

a. **Convergence analysis of the ℓ_1-exact penalty-barrier strategy.** Though preliminary results suggest that its convergence follows from the regular proofs for the filter line search. A formal statement needs to be made, including a full set of assumptions; and a study of situations in which the penalty parameter becomes unbounded.

b. **Violation of CQs because of active variable bounds.** This is one situation that might be found in situations for which the barrier parameter is small enough. It is characterized for having terms in the KKT matrix that start growing rapidly, and even with added benefits of the ℓ_1-exact penalty strategy, successful solutions might not be guaranteed.

c. **Combination with Structured Regularization techniques.** It might be possible to attain more successes in solving locally degeneracies if the structured regularization strategies from [9] are triggered whenever the barrier parameter is close to a certain minimum threshold. This would generate an increase of performance in the already discussed strategies.

d. $\rho - \mu$ **strategy.** In [38] and [64], a similar approach is suggested; in this way the penalty parameter is related to the barrier parameter as follows

$$\rho = \frac{1}{2\mu}.$$

In our framework, μ is updated following a monotonic update rule, which will ensure that ρ grows monotonically as well. However, the numerical performance of this strategy is yet to be analyzed.

IPOPT (General)

a. **Linear algebra structure exploitation.** Several works have suggested and implemented variations of this idea, e.g. [11], [65], and [12]. We suggest, implementing a strategy that is able to take advantage of the blocked diagonal structure from dynamic optimization problems, so that multiprocessor architectures can be used and thus provide faster solution times.

CAPRESE framework

a. **Incorporation of the computation of terminal regions and penalties.** For completeness of the NMPC strategies a reliable computation of the terms for the terminal function need to be included into the framework.

b. **Adaptive Horizon.** Another important aspect that would ensure a better performance of the controller strategy is the choice of a horizon length N, which was not analyzed in this work. However, there are systematic ways of calculating and adapting the appropriate horizon length, as the NMPC evolves over time [66].

c. **Improved dynamic simulation strategies.** It is not uncommon to formulate problems with high levels of variation of the time-scales. Which in turn results in stiffness and difficulties during integration. Therefore, CAPRESE requires more sophisticated strategies to attempt to adjust the strategy of simulation or provide diagnostics of the kinds of potential issues with the models of interest.

d. **Hybrid dynamics.** The introduction of nonsmooth dynamics is of great value, because several industrial level applications have points where the derivatives are not smooth. To handle these with reasonable performance Mathematical Programs with Complementarity Constraints can be used in combination of a robust NLP solver, like IPOPT. Nevertheless, there are several potential problematic issues found with these kinds of dynamic complementarity problems. From the perspective of implementation, CAPRESE might be an effective place to start.

Bibliography

[1] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg, "Systematic methods for chemical process design," 1997.

[2] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[3] R. H. Byrd, J. Nocedal, and R. A. Waltz, "K nitro: An integrated package for nonlinear optimization," in *Large-scale nonlinear optimization*, pp. 35–59, Springer, 2006.

[4] N. Chiang, C. G. Petra, and V. M. Zavala, "Structured nonconvex optimization of large-scale energy systems using pips-nlp," in *2014 Power Systems Computation Conference*, pp. 1–7, IEEE, 2014.

[5] A. U. Raghunathan and L. T. Biegler, "An interior point method for mathematical programs with complementarity constraints (mpccs)," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 720–750, 2005.

[6] B. Baumrucker, J. Renfro, and L. T. Biegler, "Mpec problem formulations and solution strategies with chemical engineering applications," *Computers & Chemical Engineering*, vol. 32, no. 12, pp. 2903–2913, 2008.

[7] B. Baumrucker and L. Biegler, "Mpec strategies for optimization of a class of hybrid dynamic systems," *Journal of Process Control*, vol. 19, no. 8, pp. 1248–1256, 2009.

[8] K. Wang, Z. Shao, Y. Lang, J. Qian, and L. T. Biegler, "Barrier nlp methods with structured regularization for optimization of degenerate optimization problems," *Computers & Chemical Engineering*, vol. 57, pp. 24–29, 2013.

[9] W. Wan and L. T. Biegler, "Structured regularization for barrier nlp solvers," *Computational Optimization and Applications*, vol. 66, no. 3, pp. 401–424, 2017.

[10] F. E. Curtis, J. Nocedal, and A. Wächter, "A matrix-free algorithm for equality constrained optimization problems with rank-deficient jacobians," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1224–1249, 2009.

[11] J. Kang, Y. Cao, D. P. Word, and C. D. Laird, "An interior-point method for efficient solution of block-structured nlp problems using an implicit schur-complement decomposition," *Computers & Chemical Engineering*, vol. 71, pp. 563–573, 2014.

[12] W. Wan, J. P. Eason, B. Nicholson, and L. T. Biegler, "Parallel cyclic reduction decomposition for dynamic optimization problems," *Computers & Chemical Engineering*, vol. 120, pp. 54–69, 2019.

[13] V. Kungurtsev and J. Jaschke, "A predictor-corrector path-following algorithm for dual-degenerate parametric optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 538–564, 2017.

[14] M. Short, C. Schenk, D. Thierry, J. S. Rodriguez, L. T. Biegler, and S. Garcia-Muñoz, "Kipet–an open-source kinetic parameter estimation toolkit," in *Computer Aided Chemical Engineering*, vol. 47, pp. 299–304, Elsevier, 2019.

[15] R. López-Negrete, S. C. Patwardhan, and L. T. Biegler, "Constrained particle filter approach to approximate the arrival cost in moving horizon estimation," *Journal of Process Control*, vol. 21, no. 6, pp. 909–919, 2011.

[16] B. Nicholson, R. López-Negrete, and L. T. Biegler, "On-line state estimation of nonlinear dynamic systems with gross errors," *Computers & Chemical Engineering*, vol. 70, pp. 149–159, 2014.

[17] V. M. Zavala and L. T. Biegler, "The advanced-step nmpc controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[18] J. Jäschke, X. Yang, and L. T. Biegler, "Fast economic model predictive control based on NLP-sensitivities," *Journal of Process Control*, vol. 24, no. 8, pp. 1260–1272, 2014.

[19] D. W. Griffith, V. M. Zavala, and L. T. Biegler, "Robustly stable economic nmpc for non-dissipative stage costs," *Journal of Process Control*, vol. 57, pp. 116–126, 2017.

[20] M. Yu and L. T. Biegler, "Economic nmpc strategies for solid sorbent-based co2 capture," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 103–108, 2018.

[21] S.-P. Han and O. L. Mangasarian, "Exact penalty functions in nonlinear programming," *Mathematical programming*, vol. 17, no. 1, pp. 251–269, 1979.

[22] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 1–31, 2005.

[23] A. V. Fiacco and Y. Ishizuka, "Sensitivity and stability analysis for nonlinear programming," *Annals of Operations Research*, vol. 27, no. 1, pp. 215–235, 1990.

[24] D. Ralph and S. Dempe, "Directional derivatives of the solution of a parametric nonlinear program," *Mathematical programming*, vol. 70, no. 1-3, pp. 159–172, 1995.

[25] L. T. Biegler, J. Nocedal, and C. Schmid, "A reduced hessian method for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 5, no. 2, pp. 314–347, 1995.

[26] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with pardiso," *Future Generation Computer Systems*, vol. 20, no. 3, pp. 475–487, 2004.

[27] D. M. Gay, "Hooking your solver to AMPL," tech. rep., Citeseer, 1997.

[28] P. R. Amestoy, I. S. Duff, and J.-Y. L'excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers," *Computer methods in applied mechanics and engineering*, vol. 184, no. 2-4, pp. 501–520, 2000.

[29] I. Quesada and I. E. Grossmann, "Global optimization of bilinear process networks with multicomponent flows," *Computers & Chemical Engineering*, vol. 19, no. 12, pp. 1219–1242, 1995.

[30] S. Kameswaran and L. T. Biegler, "Simultaneous dynamic optimization strategies: Recent advances and challenges," *Computers & Chemical Engineering*, vol. 30, no. 10-12, pp. 1560–1575, 2006.

[31] R. H. Byrd, G. Lopez-Calva, and J. Nocedal, "A line search exact penalty method using steering rules," *Mathematical Programming*, vol. 133, no. 1-2, pp. 39–73, 2012.

[32] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Steering exact penalty methods for nonlinear programming," *Optimization Methods and Software*, vol. 23, no. 2, pp. 197–213, 2008.

[33] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Y. Wang, "Intel math kernel library," in *High-Performance Computing on the Intel® Xeon Phi™*, pp. 167–188, Springer, 2014.

[34] I. S. Duff, "MA57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 118–144, 2004.

[35] R. Fourer, D. M. Gay, and B. W. Kernighan, "Ampl. a modeling language for mathematical programming," 1993.

[36] N. I. Gould, D. Orban, and P. L. Toint, "Cuter and sifdec: A constrained and unconstrained testing environment, revisited," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 4, pp. 373–394, 2003.

[37] S. Leyffer, "Macmpec: Ampl collection of mpecs," *Argonne National Laboratory*, 2000.

[38] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM review*, vol. 44, no. 4, pp. 525–597, 2002.

[39] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, vol. 10. Siam, 2010.

[40] J. B. Rawlings and L. Ji, "Optimization-based state estimation: Current status and some new results," *Journal of Process Control*, vol. 22, no. 8, pp. 1439–1444, 2012.

[41] L. Ji, J. B. Rawlings, W. Hu, A. Wynn, and M. Diehl, "Robust stability of moving horizon estimation under bounded disturbances," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3509–3514, 2016.

[42] F. Allgiower and A. Zheng, "Nonlinear model predictive control," *Progress in Systems and Control Theory*, vol. 26, 2000.

[43] D. Mayne, "Nonlinear model predictive control: Challenges and opportunities," in *Nonlinear model predictive control*, pp. 23–44, Springer, 2000.

[44] K. Caldeira and M. R. Rampino, "The mid-cretaceous super plume, carbon dioxide, and global warming," *Geophysical Research Letters*, vol. 18, no. 6, pp. 987–990, 1991.

[45] J. G. Olivier, J. A. Peters, and G. Janssens-Maenhout, "Trends in global CO2 emissions 2012 report," 2012.

[46] F. A. Rahman, M. M. A. Aziz, R. Saidur, W. A. W. A. Bakar, M. Hainin, R. Putrajaya, and N. A. Hassan, "Pollution to solution: Capture and sequestration of carbon dioxide (co2) and its utilization as a renewable energy source for a sustainable future," *Renewable and Sustainable Energy Reviews*, vol. 71, pp. 112–126, 2017.

[47] A. B. Rao and E. S. Rubin, "A technical, economic, and environmental assessment of amine-based CO2 capture technology for power plant greenhouse gas control," *Environmental science & technology*, vol. 36, no. 20, pp. 4467–4475, 2002.

[48] A. Samanta, A. Zhao, G. K. Shimizu, P. Sarkar, and R. Gupta, "Post-combustion co2

capture using solid sorbents: a review," *Industrial & Engineering Chemistry Research*, vol. 51, no. 4, pp. 1438–1463, 2011.

[49] F. Engstrom, J. M. Isaksson, and R. Kuivalainen, "Fluidized bed reactor," Sept. 12 1989. US Patent 4,864,944.

[50] F. Taghipour, N. Ellis, and C. Wong, "Experimental and computational study of gas–solid fluidized bed hydrodynamics," *Chemical Engineering Science*, vol. 60, no. 24, pp. 6857–6867, 2005.

[51] D. Kunii and O. Levenspiel, "Bubbling bed model. model for flow of gas through a fluidized bed," *Industrial & Engineering Chemistry Fundamentals*, vol. 7, no. 3, pp. 446–452, 1968.

[52] A. Lee and D. C. Miller, "A one-dimensional (1-d) three-region model for a bubbling fluidized-bed adsorber," *Industrial & Engineering Chemistry Research*, vol. 52, no. 1, pp. 469–484, 2012.

[53] K.-Y. Choi and W. H. Ray, "The dynamic behaviour of fluidized bed reactors for solid catalysed gas phase olefin polymerization," *Chemical Engineering Science*, vol. 40, no. 12, pp. 2261–2279, 1985.

[54] J. R. Muir, C. Brereton, J. R. Grace, and C. J. Lim, "Dynamic modeling for simulation and control of a circulating fluidized-bed combustor," *AIChE Journal*, vol. 43, no. 5, pp. 1141–1152, 1997.

[55] S. Modekurti, D. Bhattacharyya, and S. E. Zitney, "Dynamic modeling and control studies of a two-stage bubbling fluidized bed adsorber-reactor for solid–sorbent CO2 capture," *Industrial & Engineering Chemistry Research*, vol. 52, no. 30, pp. 10250–10260, 2013.

[56] M. Yu, D. C. Miller, and L. T. Biegler, "Dynamic reduced order models for simulating

bubbling fluidized bed adsorbers," *Industrial & Engineering Chemistry Research*, vol. 54, no. 27, pp. 6959–6974, 2015.

[57] M. Yu and L. T. Biegler, "Economic NMPC strategies for solid sorbent-based CO2 capture," *10th IFAC International Symposium on Advanced Control of Chemical Processes (ADCHEM 2018)*, no. 2, pp. 103–108, 2018.

[58] "IDAES website." `https://idaes.org/about/overview/`.

[59] D. Woodruff, G. Hackebeil, C. D. Laird, B. L. Nicholson, W. E. Hart, J. D. Siirola, and J.-P. Watson, "Pyomo v5. 0," tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2017.

[60] B. Nicholson, J. D. Siirola, J.-P. Watson, V. M. Zavala, and L. T. Biegler, "pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations," *Mathematical Programming Computation*, vol. 10, no. 2, pp. 187–223, 2018.

[61] R. López-Negrete, "Nonlinear programming sensitivity based methods for constrained state estimation," *PhD thesis*, 2011.

[62] M. Diehl, *Real-time optimization for large scale nonlinear processes*. PhD thesis, 2001.

[63] R. Lopez-Negrete, F. J. D'Amato, L. T. Biegler, and A. Kumar, "Fast nonlinear model predictive control: Formulation and industrial process applications," *Computers & Chemical Engineering*, vol. 51, pp. 55–64, 2013.

[64] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques*, vol. 4. Siam, 1990.

[65] V. M. Zavala, C. D. Laird, and L. T. Biegler, "Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems," *Chemical Engineering Science*, vol. 63, no. 19, pp. 4834–4845, 2008.

[66] D. W. Griffith, S. C. Patwardhan, and L. T. Biegler, "Quasi-infinite adaptive horizon nonlinear model predictive control," 2018.

Appendix

.1 CUTEr test set

The list of problems taken into account for this work are presented on Tables 1 through 3. The results for the degenerate CUTEr set are presented on the Tables 4 through 30. The results were generated triggering the ℓ_1-exact penalty mode when restoration phase is called, note that the status code corresponds to the same codes from IPOPT.

.2 Example CAPRESE snippet

To illustrate the way the framework interacts with the user, this section contains example code inside main loop of the controller(Python).

Listing 1: Iteration from the controller.

```
e.solve_dyn(e.PlantSample)  #: Plant

e.update_state_real()  # Update the current state
e.update_soi_sp_nmpc()  #: To keep track of the state of interest.

e.update_measurement()  # Update the current measurement
e.compute_y_offset()  #: Get the offset for y

if i > 1:
e.sens_dot_mhe()
e.update_state_mhe(as_nmpc_mhe_strategy=True)
```

Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con
gilbert	1000	2	0	swopf	83	79	14	smmpsf	720	241	23	hs042	4	3	0
bt11	5	4	0	dallasm	196	152	0	hs100lnp	7	3	0	optmass	66	45	11
hs046	5	3	0	himmelbj	45	17	0	gridnetf	7565	3845	0	sseblin	192	49	24
hs007	2	2	0	aug3d	3873	1001	0	lsnnodoc	5	5	0	hs028	3	2	0
hs114	10	4	8	bt5	3	3	0	bt12	5	4	0	batch	46	13	61
extrasim	2	2	0	allinitc	4	2	3	loadbal	31	12	20	hs050	5	4	0
model	1831	325	15	bt9	4	3	0	hs081	5	4	0	hs087	9	5	0
disc2	28	18	6	qpnboei1	384	10	339	dtoc1na	1485	991	0	hs080	5	4	0
eigenbco	110	56	0	trimloss	142	21	55	genhs28	10	9	0	hs062	3	2	0
catenary	496	167	0	dtoc1nd	735	491	0	hs071	4	2	1	trainh	20000	10003	0
tame	2	2	0	minc44	303	263	0	hs078	5	4	0	sawpath	593	591	196
dtoc1l	14985	9991	0	hs073	4	2	2	ncvxqp1	1000	501	0	steenbrb	468	109	0
fccu	19	9	0	hs039	4	3	0	avion2	49	16	0	gouldqp2	699	350	0
hager4	10000	5001	0	catena	32	12	0	gridnetc	7564	3845	0	zigzag	58	41	10
hs056	7	5	0	hs053	5	4	0	hs041	4	2	4	maratos	2	2	0
robot	14	10	0	ncvxqp4	1000	251	0	hs109	9	7	4	aljazzaf	3	2	0
bt13	5	2	1	ncvxqp7	1000	751	0	orthregb	27	7	0	reading3	202	102	1
optcdeg2	1199	801	0	hs006	2	2	0	prodpl0	60	21	9	orthrdm2	4003	2001	0
hs027	3	2	0	qpcstair	385	210	147	blockqp2	2005	1001	1	eigenaco	110	56	0
portfl2	12	2	0	reading1	10001	5001	0	hvycrash	201	151	0	dtoc5	9998	5000	0
blockqp4	2005	1001	1	spanhyd	81	34	0	linspanh	81	34	0	hs047	5	4	0
qpcboei1	384	10	339	aug3dc	3873	1001	0	eigena2	110	56	0	steenbrf	468	109	0
hs111lnp	10	4	0	portfl3	12	2	0	hs061	3	3	0	degenlpa	20	15	0
hs032	3	2	1	hs040	4	4	0	hs051	5	4	0	launch	25	10	20
mwright	5	4	0	sosqp1	20000	10002	0	ncvxqp8	1000	751	0	hs060	3	2	0

Table 1: CUTEr set, part 1.

Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con
dualc5	8	2	277	dixchlng	10	6	0	bt4	3	3	0	smbank	117	65	0
orthrege	36	21	0	aug2d	20192	9997	0	hs99exp	28	22	0	bloweyc	2002	1003	0
hs111	10	4	0	gouldqp3	699	350	0	dixchlnv	100	51	0	hs112	10	4	0
try-b	2	2	0	sosqp2	20000	10002	0	dual3	111	2	0	hadamard	65	65	192
ssnlbeam	31	21	0	gridneth	61	37	0	dtoc1nc	1485	991	0	degenlpb	20	16	0
portfl6	12	2	0	optctrl3	119	81	1	hs048	5	3	0	himmelbk	24	15	0
hager2	10000	5001	0	hs079	5	4	0	ncvxqp2	1000	501	0	optcdeg3	1199	801	0
qpnstair	385	210	147	brainpc2	13805	13801	0	dual4	75	2	0	aug2dcqp	20200	9997	198
hs099	23	19	0	orthregc	10005	5001	0	steenbrg	540	127	0	deconvc	51	2	0
eigenc2	462	232	0	brainpc6	6905	6901	0	dtoc6	10000	5001	0	bt1	2	2	0
hs077	5	3	0	brainpc7	6905	6901	0	byrdsphr	3	3	0	hs107	9	7	8
blockqp3	2005	1001	1	hues-mod	10000	3	0	hager3	10000	5001	0	hager1	10001	5002	0
lch	600	2	0	orthrgds	10003	5001	0	bt2	3	2	0	hs35mod	3	2	1
csfi2	5	3	2	brainpc8	6905	6901	0	gridneti	61	37	0	gridnete	7565	3845	0
corkscrw	8997	6001	1000	brainpc0	6905	6901	0	eigenb2	110	56	0	blockqp5	2005	1001	1
ubh5	19997	14001	0	brainpc3	6905	6901	0	dtoc4	14997	9999	0	gausselm	1495	1241	2722
ncvxqp6	1000	251	0	cvxqp3	10000	7501	0	dtoc3	14997	9999	0	eg3	101	2	199
lotschd	12	8	0	brainpc5	6905	6901	0	gridnetd	5145	3845	0	aug2dc	20200	9997	198
dualc2	7	2	228	cvxqp2	10000	2501	0	hs119	16	9	0	ncvxqp3	1000	501	0
minperm	1113	1034	0	huestis	10000	3	0	portfl4	12	2	0	yao	2002	3	2001
aug3dcqp	3873	1001	0	brainpc1	6905	6901	0	cvxqp1	1000	501	0	hs026	3	2	0
bt8	5	3	0	brainpc4	6905	6901	0	bt3	5	4	0	bloweyb	2002	1003	0
qpcboei2	143	5	136	orthrgdm	10003	5001	0	steenbra	432	109	0	static3	434	97	0
kissing	127	43	861	orthregd	10003	5001	0	optctrl6	119	81	1	qpnboei2	143	5	136
alsotame	2	2	2	brainpc9	6905	6901	0	reading2	15003	10003	0	orthrega	517	257	0

Table 2: CUTEr set, part 2.

Problem	n_var	eq_con	in_con	Problem	n_var	eq_con	in_con
hs052	5	4	0	dtoc2	5994	3997	0
steenbrd	468	109	0	orthrds2	203	101	0
bloweya	2002	1003	0	eigencco	30	16	0
dualc1	9	2	214	lakes	90	79	0
dualc8	8	2	502	ncvxqp5	1000	251	0
portfl1	12	2	0	dual1	85	2	0
hs074	4	4	1	odfits	10	7	0
trainf	20000	10003	0	bt6	5	3	0
csfi1	5	3	2	clnlbeam	1499	1001	0
dual2	96	2	0	bt7	5	4	0
gridnetg	47	37	0	rk23	17	12	0
hong	4	2	0	hs054	6	2	0
dtoc1nb	1485	991	0	gridnetb	13284	6725	0
hs014	2	2	1	ubh1	17997	12001	0
sreadin3	10001	5002	0	dittert	327	265	0
optcntrl	29	21	1	blockqp1	2005	1001	1
steenbre	540	127	0	ncvxqp9	1000	751	0
dnieper	57	25	0	hs063	3	3	0
steenbrc	540	127	0	hs009	2	2	0
gridneta	8964	6725	0	hs067	10	8	14
prodpl1	60	21	9	ssebnln	192	73	24
aug3dqp	3873	1001	0	hs075	4	4	1
hs049	5	3	0				
aug2dqp	20192	9997	198				
fletcher	4	2	4				

Table 3: CUTEr set, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	77	1.67E+01	2	swopf	23	4.73E-02	0	smmpsf	674	1.83E+00	0	hs042	10	1.51E-02	0
bt11	68	8.37E-02	0	dallasm	36	7.27E-02	1	hs100lnp	196	3.36E-01	0	optmass	23	3.72E-02	0
hs046	45	7.98E-02	1	himmelbj	3000	3.75E+02		gridnetf	32	1.44E+00	0	sseblin	81	2.01E-01	0
hs007	79	1.80E-01	0	aug3d	12	9.04E-01	0	lsnnodoc	1	3.96E-03	0	hs028	1	3.46E-03	0
hs114	141	2.33E-01	0	bt5	53	6.88E-02	0	bt12	37	4.84E-02	0	batch	33	5.33E-02	0
extrasim	11	1.83E-02	0	allinitc	163	2.12E-01	1	loadbal	15	2.32E-02	0	hs050	9	1.13E-02	0
model	35	1.61E-01	0	bt9	128	4.07E-01	0	hs081	23	4.17E-02	0	hs087	76	1.53E-01	0
disc2	3000	8.57E+00	-1	qpnboei1	331	1.65E+00	1	dtoc1na	6	4.11E-01	0	hs080	26	4.70E-02	0
eigenbco	495	2.45E+00	0	trimloss	67	2.20E-01	1	genhs28	1	3.79E-03	0	hs062	7	7.43E-03	0
catenary	282	2.55E+00	-2	dtoc1nd	22	5.41E-01	0	hs071	74	1.13E-01	0	trainh	615	5.03E+01	0
tame	9	1.36E-02	0	minc44	389	5.05E+00	0	hs078	24	3.29E-02	0	sawpath	3000	2.48E+02	-1
dtoc1l	6	3.24E-01	0	hs073	3000	4.34E+00	-1	ncvxqp1	197	1.05E+01	0	steenbrb	49	4.94E-01	0
fccu	8	2.18E-02	0	hs039	128	2.83E-01	0	avion2	93	1.58E-01	0	gouldqp2	15	4.75E-02	0
hager4	19	7.86E-01	0	catena	149	2.09E-01	0	gridnetc	33	9.14E-01	0	zigzag	31	5.11E-02	0
hs056	15	1.86E-02	0	hs053	13	2.76E-02	0	hs041	16	3.01E-02	0	maratos	19	3.78E-02	0
robot	127	1.69E-01	0	ncvxqp4	349	9.95E+00	0	hs109	103	2.34E-01	0	aljazzaf	29	3.97E-02	0
bt13	33	5.95E-02	0	ncvxqp7	233	1.45E+01	0	orthregb	67	1.28E-01	0	reading3	29	6.37E-02	0
optcdeg2	34	1.76E-01	0	hs006	2045	2.85E+00	1	prodpl0	17	2.69E-02	0	orthrdm2	14	3.64E+00	0
hs027	87	1.12E-01	0	qpcstair	205	1.02E+00	0	blockqp2	13	9.30E-02	0	eigenaco	5	2.18E-02	0
portfl2	8	1.35E-02	0	reading1	1235	9.08E+01	1	hvycrash	194	7.36E-01	0	dtoc5	7	1.63E-01	0
blockqp4	11	1.41E-01	0	spanhyd	20	3.80E-02	0	linspanh	15	2.66E-02	0	hs047	25	6.32E-02	-2
qpcboei1	178	7.75E-01	0	aug3dc	11	1.27E-01	0	eigena2	3	1.38E-02	0	steenbrf	992	2.82E+00	0
hs111lnp	87	3.29E-01	2	portfl3	10	1.65E-02	0	hs061	20	4.84E-02	0	degenlpa	73	1.33E-01	1
hs032	16	2.36E-02	0	hs040	7	1.18E-02	0	hs051	1	3.50E-03	0	launch	143	4.51E-01	2
mwright	156	2.99E-01	0	sosqp1	45	3.84E+00	0	ncvxqp8	210	1.15E+01	0	hs060	75	1.28E-01	0

Table 4: CUTEr set results, vanilla, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	3.66E-02	0	dixchlng	24	3.40E-02	0	bt4	27	4.77E-02	0	smbank	17	2.83E-02	0
orthrege	85	2.79E-01	-2	aug2d	11	6.81E-01	0	hs99exp	53	7.99E-02	0	bloweyc	44	1.16E+01	0
hs111	638	1.33E+00	0	gouldqp3	18	1.00E-01	0	dixchlnv	72	2.38E-01	0	hs112	17	2.46E-02	0
try-b	46	1.04E-01	0	sosqp2	58	7.92E+00	0	dual3	19	2.00E-01	0	hadamard	6	2.03E-02	0
ssnlbeam	3000	4.32E+00	-1	gridneth	12	2.49E-02	0	dtoc1nc	17	8.39E-01	0	degenlpb	53	7.48E-02	0
portfl6	8	9.29E-03	0	optctrl3	79	1.83E-01	0	hs048	1	1.62E-03	0	himmelbk	214	5.48E-01	3
hager2	10	2.25E-01	0	hs079	159	1.99E-01	0	ncvxqp2	564	4.17E+01	0	optcdeg3	28	1.16E-01	0
qpnstair	288	1.54E+00	0	brainpc2	358	1.90E+04	-4	dual4	16	6.79E-02	0	aug2dcqp	32	1.93E+00	0
hs099	15	3.29E-02	0	orthregc	110	2.65E+03	0	steenbrg	159	2.61E+00	0	deconvc	166	5.46E-01	0
eigenc2	13	1.03E+00	-2	brainpc6	215	1.04E+01	-2	dtoc6	31	9.62E-01	0	bt1	48	6.06E-02	0
hs077	44	8.79E-02	0	brainpc7	242	7.08E+02	-2	byrdsphr	108	3.79E-01	0	hs107	92	2.10E-01	0
blockqp3	81	8.28E-01	0	hues-mod	85	3.61E+03	-4	hager3	10	3.89E-01	0	hager1	1	3.23E-02	0
lch	352	1.04E+01	0	orthrgds	158	3.64E+02	0	bt2	16	2.93E-02	0	hs35mod	15	2.02E-02	0
csfi2	102	1.74E-01	0	brainpc8	258	1.60E+03	-2	gridneti	18	3.93E-02	0	gridnete	12	7.49E-01	0
corkscrw	474	1.21E+01	0	brainpc0	15	3.85E+03	-4	eigenb2	4	1.57E-02	0	blockqp5	50	5.25E-01	0
ubh5	12	1.04E+00	0	brainpc3	302	2.29E+01	-2	dtoc4	3	1.92E-01	0	gausselm	1343	2.79E+01	0
ncvxqp6	350	8.48E+00	0	cvxqp3	58	5.46E+02	0	dtoc3	10	4.31E-01	0	eg3	19	4.80E-02	0
lotschd	33	5.04E-02	0	brainpc5	659	7.38E+02	-2	gridnetd	23	8.64E-01	0	aug2dc	17	7.53E-01	0
dualc2	19	3.88E-02	0	cvxqp2	32	2.99E+01	0	hs119	48	9.25E-02	0	ncvxqp3	455	2.82E+01	0
minperm	3000	9.40E+02	-1	huestis	95	3.62E+03	-4	portfl4	9	1.25E-02	0	yao	486	5.29E+00	0
aug3dcqp	24	3.85E-01	0	brainpc1	3000	1.42E+02	-1	cvxqp1	167	5.29E+00	0	hs026	33	4.79E-02	0
bt8	20	3.76E-02	0	brainpc4	274	1.50E+01	-2	bt3	12	3.24E-02	0	bloweyb	7	1.94E+00	0
qpcboei2	85	1.90E-01	0	orthrgdm	19	2.03E+00	0	steenbra	23	1.53E-01	0	static3	73	2.36E-01	4
kissing	483	7.21E+00	0	orthregd	18	2.02E+00	0	optctrl6	79	1.96E-01	0	qpnboei2	157	4.32E-01	0
alsotame	9	9.15E-03	0	brainpc9	319	2.85E+03	-2	reading2	15	8.09E-01	0	orthrega	123	6.51E-01	0

Table 5: CUTEr set results, vanilla, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	10	1.94E-02	0	dtoc2	86	2.46E+00	0
steenbrd	121	1.81E+00	0	orthrds2	78	2.32E-01	0
bloweya	169	2.86E+01	0	eigencco	30	4.69E-02	0
dualc1	24	6.53E-02	0	lakes	364	1.10E+00	0
dualc8	19	5.95E-02	0	ncvxqp5	386	7.93E+00	0
portfl1	9	1.36E-02	0	dual1	18	8.90E-02	0
hs074	12	3.19E-02	0	odfits	17	2.47E-02	0
trainf	53	2.20E+00	0	bt6	37	5.10E-02	0
csfi1	33	5.14E-02	0	clnlbeam	187	1.06E+00	0
dual2	15	6.83E-02	0	bt7	89	1.34E-01	0
gridnetg	25	5.40E-02	1	rk23	9	1.17E-02	0
hong	24	3.24E-02	0	hs054	9	1.36E-02	0
dtoc1nb	6	2.29E-01	0	gridnetb	9	7.02E-01	0
hs014	10	1.10E-02	0	ubh1	12	1.00E+00	0
sreadin3	12	5.49E-01	0	dittert	91	1.09E+00	0
optcntrl	44	6.64E-02	0	blockqp1	27	4.38E-01	0
steenbre	1636	3.80E+01	1	ncvxqp9	268	1.66E+01	0
dnieper	44	7.24E-02	0	hs063	17	2.15E-02	0
steenbrc	289	7.59E-01	-2	hs009	0	2.14E-03	0
gridneta	26	7.58E-01	0	hs067	11	1.45E-02	0
prodpl1	15	1.53E-02	0	ssebnln	150	6.81E-01	-2
aug3dqp	23	3.06E-01	0	hs075	13	3.19E-02	0
hs049	19	2.51E-02	0				
aug2dqp	33	1.96E+00	0				
fletcher	72	1.08E-01	0				

Table 6: CUTEr set results, vanilla, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	77	1.68E+01	0	swopf	20	6.84E-02	0	smmpsf	674	1.92E+00	0	hs042	10	1.53E-02	0
bt11	68	7.63E-02	0	dallasm	36	7.98E-02	1	hs100lnp	172	3.04E-01	0	optmass	23	3.68E-02	0
hs046	31	8.09E-02	0	himmelbj	3000	1.46E+01	-1	gridnetf	38	2.09E+00	0	sseblin	113	4.01E-01	0
hs007	66	1.35E-01	0	aug3d	12	6.35E-01	0	lsnnodoc	1	5.00E-03	0	hs028	1	4.01E-03	0
hs114	100	2.40E-01	0	bt5	53	7.20E-02	0	bt12	37	6.61E-02	0	batch	33	5.71E-02	0
extrasim	11	1.67E-02	0	allinitc	45	1.06E-01	0	loadbal	15	2.38E-02	0	hs050	9	1.40E-02	0
model	35	1.70E-01	0	bt9	70	1.77E-01	0	hs081	24	7.19E-02	0	hs087	76	1.17E-01	0
disc2	45	1.31E-01	0	qpnboei1	3000	5.39E+01	-1	dtoc1na	6	2.77E-01	0	hs080	16	4.04E-02	0
eigenbco	495	2.30E+00	0	trimloss	67	2.61E-01	0	genhs28	1	4.74E-03	0	hs062	7	8.94E-03	0
catenary	3000	1.22E+01	-1	dtoc1nd	22	3.77E-01	0	hs071	47	1.39E-01	0	trainh	100	8.26E+00	0
tame	9	5.44E-03	0	minc44	51	5.31E-01	0	hs078	24	3.80E-02	0	sawpath	3000	3.19E+02	-1
dtoc1l	6	3.06E-01	0	hs073	3000	4.44E+00	-1	ncvxqp1	312	1.40E+01	1	steenbrb	49	5.31E-01	0
fccu	13	3.88E-02	0	hs039	70	1.79E-01	0	avion2	93	1.71E-01	0	gouldqp2	15	4.79E-02	0
hager4	19	5.79E-01	0	catena	149	2.16E-01	0	gridnetc	39	4.17E+00	0	zigzag	31	5.64E-02	0
hs056	15	1.75E-02	0	hs053	17	4.30E-02	0	hs041	21	5.96E-02	0	maratos	78	2.10E-01	0
robot	115	2.93E-01	0	ncvxqp4	568	1.40E+01	0	hs109	60	1.66E-01	0	aljazzaf	29	3.48E-02	0
bt13	50	1.38E-01	0	ncvxqp7	262	1.60E+01	0	orthregb	47	1.33E-01	0	reading3	29	7.53E-02	0
optcdeg2	47	3.35E-01	0	hs006	2045	2.62E+00	1	prodpl0	17	2.46E-02	0	orthrdm2	14	3.77E+00	0
hs027	87	1.05E-01	0	qpcstair	813	5.47E+00	1	blockqp2	13	1.05E-01	0	eigenaco	5	2.81E-02	0
portfl2	8	1.30E-02	0	reading1	91	7.18E+00	0	hvycrash	616	2.76E+00	1	dtoc5	18	5.56E-01	0
blockqp4	11	8.77E-02	0	spanhyd	20	3.74E-02	0	linspanh	15	2.99E-02	0	hs047	31	8.14E-02	0
qpcboei1	286	1.51E+00	0	aug3dc	11	1.16E-01	0	eigena2	3	1.58E-02	0	steenbrf	326	1.94E+00	1
hs111lnp	97	2.26E-01	2	portfl3	10	1.42E-02	0	hs061	27	6.40E-02	0	degenlpa	84	2.08E-01	0
hs032	16	1.93E-02	0	hs040	7	9.98E-03	0	hs051	1	3.39E-03	0	launch	3000	3.23E+01	-1
mwright	125	1.92E-01	0	sosqp1	37	3.35E+00	0	ncvxqp8	367	3.21E+01	1	hs060	72	1.48E-01	0

Table 7: CUTEr set results, penalty mode ρ, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	4.18E-02	0	dixchlng	24	3.06E-02	0	bt4	261	7.49E-01	0	smbank	17	2.61E-02	0
orthrege	189	6.51E-01	1	aug2d	11	6.81E-01	0	hs99exp	91	2.41E-01	0	bloweyc	44	1.20E+01	0
hs111	3000	8.98E+00	-1	gouldqp3	21	1.56E-01	0	dixchlnv	72	2.67E-01	0	hs112	17	2.04E-02	0
try-b	30	6.94E-02	0	sosqp2	28	2.53E+00	0	dual3	23	3.01E-01	0	hadamard	6	2.89E-02	0
ssnlbeam	3000	4.41E+00	-1	gridneth	24	8.18E-02	0	dtoc1nc	17	8.63E-01	0	degenlpb	68	1.52E-01	0
portfl6	8	6.66E-03	0	optctrl3	79	2.09E-01	0	hs048	1	2.94E-04	0	himmelbk	1097	4.21E+00	0
hager2	10	2.46E-01	0	hs079	159	2.06E-01	0	ncvxqp2	447	2.35E+01	0	optcdeg3	43	3.01E-01	0
qpnstair	1300	1.29E+01	1	brainpc2	690	2.91E+02	1	dual4	16	7.32E-02	0	aug2dcqp	32	1.83E+00	0
hs099	24	6.90E-02	0	orthregc	110	3.48E+03	0	steenbrg	159	2.81E+00	0	deconvc	166	3.55E-01	0
eigenc2	45	1.96E+00	0	brainpc6	334	3.60E+01	1	dtoc6	58	1.86E+00	0	bt1	48	6.62E-02	0
hs077	97	2.55E-01	0	brainpc7	3000	6.96E+02	-1	byrdsphr	61	1.64E-01	0	hs107	3000	8.31E+00	-1
blockqp3	49	9.18E-01	1	hues-mod	105	4.95E+03	0	hager3	10	3.89E-01	0	hager1	1	3.16E-02	0
lch	695	2.21E+01	1	orthrgds	158	3.79E+02	0	bt2	23	6.46E-02	0	hs35mod	15	2.38E-02	0
csfi2	68	2.01E-01	0	brainpc8	544	6.31E+01	1	gridneti	90	3.68E-01	0	gridnete	31	1.67E+00	0
corkscrw	241	9.24E+00	0	brainpc0	1159	2.64E+03	1	eigenb2	4	2.24E-02	0	blockqp5	50	4.55E-01	0
ubh5	19	1.95E+00	0	brainpc3	2052	4.57E+02	1	dtoc4	3	1.49E-01	0	gausselm	1343	3.02E+01	0
ncvxqp6	867	3.93E+01	1	cvxqp3	58	7.06E+02	0	dtoc3	21	1.06E+00	0	eg3	19	5.61E-02	0
lotschd	39	8.57E-02	0	brainpc5	3000	4.91E+02	-1	gridnetd	50	2.04E+00	0	aug2dc	17	8.06E-01	0
dualc2	19	4.63E-02	0	cvxqp2	40	5.16E+01	0	hs119	50	1.37E-01	0	ncvxqp3	468	3.00E+01	0
minperm	85	2.16E+01	0	huestis	84	3.74E+03	2	portfl4	9	1.16E-02	0	yao	3000	4.17E+01	-1
aug3dcqp	24	3.15E-01	0	brainpc1	3000	2.16E+02	-1	cvxqp1	67	1.41E+00	0	hs026	33	5.49E-02	0
bt8	17	4.65E-02	0	brainpc4	295	2.84E+01	1	bt3	18	5.72E-02	0	bloweyb	7	1.73E+00	0
qpcboei2	85	2.00E-01	0	orthrgdm	19	2.48E+00	0	steenbra	23	1.68E-01	0	static3	73	2.72E-01	4
kissing	229	3.03E+00	0	orthregd	18	2.14E+00	0	optctrl6	79	2.06E-01	0	qpnboei2	456	2.25E+00	0
alsotame	9	1.49E-02	0	brainpc9	3000	4.47E+02	-1	reading2	15	8.80E-01	0	orthrega	92	6.18E-01	0

Table 8: CUTEr set results, penalty mode ρ, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	17	3.44E-02	0	dtoc2	86	2.61E+00	0
steenbrd	121	1.70E+00	0	orthrds2	962	4.31E+00	1
bloweya	169	2.97E+01	0	eigencco	30	5.44E-02	0
dualc1	24	6.47E-02	0	lakes	435	1.36E+00	0
dualc8	19	6.76E-02	0	ncvxqp5	386	8.53E+00	0
portfl1	9	1.63E-02	0	dual1	18	1.06E-01	0
hs074	24	8.57E-02	0	odfits	17	2.17E-02	0
trainf	53	2.31E+00	0	bt6	26	6.59E-02	0
csfi1	46	1.19E-01	0	clnlbeam	187	1.19E+00	0
dual2	15	1.26E-01	0	bt7	45	1.11E-01	0
gridnetg	31	9.53E-02	0	rk23	9	1.64E-02	0
hong	24	3.63E-02	0	hs054	9	1.08E-02	0
dtoc1nb	6	2.59E-01	0	gridnetb	36	1.97E+00	0
hs014	10	1.75E-02	0	ubh1	20	1.52E+00	0
sreadin3	12	6.02E-01	0	dittert	91	9.55E-01	0
optcntrl	44	1.37E-01	0	blockqp1	34	5.91E-01	0
steenbre	1636	4.01E+01	1	ncvxqp9	327	2.46E+01	0
dnieper	44	8.02E-02	0	hs063	17	3.22E-02	0
steenbrc	97	5.49E-01	1	hs009	0	0.00E+00	0
gridneta	35	1.22E+00	0	hs067	11	2.09E-02	0
prodpl1	15	2.11E-02	0	ssebnln	3000	2.63E+01	-1
aug3dqp	23	2.83E-01	0	hs075	35	8.41E-02	0
hs049	19	2.89E-02	0				
aug2dqp	33	1.88E+00	0				
fletcher	61	1.98E-01	2				

Table 9: CUTEr set results, penalty mode ρ, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	77	1.70E+01	0	swopf	20	6.65E-02	0	smmpsf	674	1.89E+00	0	hs042	10	1.57E-02	0
bt11	68	8.91E-02	0	dallasm	36	7.89E-02	1	hs100lnp	172	3.03E-01	0	optmass	23	4.09E-02	0
hs046	31	7.56E-02	0	himmelbj	3000	1.48E+01	-1	gridnetf	38	1.80E+00	0	sseblin	117	4.10E-01	0
hs007	66	1.52E-01	0	aug3d	12	6.26E-01	0	lsnnodoc	1	2.41E-03	0	hs028	1	4.09E-03	0
hs114	118	2.93E-01	0	bt5	53	7.48E-02	0	bt12	37	6.31E-02	0	batch	33	5.98E-02	0
extrasim	11	1.20E-02	0	allinitc	41	9.39E-02	0	loadbal	15	2.62E-02	0	hs050	9	1.00E-02	0
model	35	1.78E-01	0	bt9	70	1.61E-01	0	hs081	24	6.71E-02	0	hs087	76	9.70E-02	0
disc2	45	1.29E-01	0	qpnboei1	1815	1.21E+01	0	dtoc1na	6	2.52E-01	0	hs080	16	3.93E-02	0
eigenbco	495	2.25E+00	0	trimloss	63	2.45E-01	0	genhs28	1	4.33E-03	0	hs062	7	1.33E-02	0
catenary	3000	1.20E+01	-1	dtoc1nd	22	3.35E-01	0	hs071	60	1.53E-01	0	trainh	100	8.26E+00	0
tame	9	9.74E-03	0	minc44	51	5.31E-01	0	hs078	24	3.22E-02	0	sawpath	3000	7.20E+01	-1
dtoc1l	6	3.02E-01	0	hs073	3000	4.20E+00	-1	ncvxqp1	312	1.37E+01	1	steenbrb	49	5.13E-01	0
fccu	13	3.52E-02	0	hs039	70	1.81E-01	0	avion2	93	1.50E-01	0	gouldqp2	15	5.23E-02	0
hager4	19	5.74E-01	0	catena	149	2.07E-01	0	gridnetc	39	4.10E+00	0	zigzag	31	5.48E-02	0
hs056	15	1.83E-02	0	hs053	17	4.58E-02	0	hs041	22	5.13E-02	0	maratos	78	2.03E-01	0
robot	115	3.01E-01	0	ncvxqp4	568	1.38E+01	0	hs109	60	1.66E-01	0	aljazzaf	29	4.24E-02	0
bt13	50	1.33E-01	0	ncvxqp7	262	1.55E+01	0	orthregb	47	1.23E-01	0	reading3	29	7.34E-02	0
optcdeg2	47	3.33E-01	0	hs006	2045	2.61E+00	1	prodpl0	17	2.80E-02	0	orthrdm2	14	3.88E+00	0
hs027	87	1.15E-01	0	qpcstair	702	4.04E+00	0	blockqp2	13	1.06E-01	0	eigenaco	5	2.80E-02	0
portfl2	8	1.16E-02	0	reading1	91	6.30E+00	0	hvycrash	616	2.67E+00	1	dtoc5	18	5.64E-01	0
blockqp4	11	8.85E-02	0	spanhyd	20	4.33E-02	0	linspanh	15	2.34E-02	0	hs047	31	8.15E-02	0
qpcboei1	280	1.48E+00	0	aug3dc	11	1.10E-01	0	eigena2	3	9.34E-03	0	steenbrf	326	1.91E+00	1
hs111lnp	97	2.17E-01	2	portfl3	10	1.26E-02	0	hs061	27	6.52E-02	0	degenlpa	84	2.26E-01	0
hs032	16	1.90E-02	0	hs040	7	1.23E-02	0	hs051	1	1.20E-03	0	launch	3000	1.60E+01	-1
mwright	125	2.08E-01	0	sosqp1	37	3.19E+00	0	ncvxqp8	367	3.21E+01	1	hs060	72	1.26E-01	0

Table 10: CUTEr set results, penalty mode ρ no Σ, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	3.71E-02	0	dixchlng	24	3.64E-02	0	bt4	261	7.31E-01	0	smbank	17	4.10E-02	0
orthrege	189	6.15E-01	1	aug2d	11	6.69E-01	0	hs99exp	91	2.33E-01	0	bloweyc	44	1.22E+01	0
hs111	3000	8.98E+00	-1	gouldqp3	21	1.53E-01	0	dixchlnv	72	2.85E-01	0	hs112	17	2.60E-02	0
try-b	30	5.18E-02	0	sosqp2	28	2.50E+00	0	dual3	23	3.05E-01	0	hadamard	6	2.18E-02	0
ssnlbeam	3000	4.30E+00	-1	gridneth	24	8.01E-02	0	dtoc1nc	17	8.63E-01	0	degenlpb	68	1.58E-01	0
portfl6	8	1.47E-02	0	optctrl3	79	1.84E-01	0	hs048	1	4.22E-03	0	himmelbk	1097	4.47E+00	0
hager2	10	2.40E-01	0	hs079	159	1.85E-01	0	ncvxqp2	447	2.40E+01	0	optcdeg3	43	2.88E-01	0
qpnstair	703	4.48E+00	0	brainpc2	690	2.80E+02	1	dual4	16	7.58E-02	0	aug2dcqp	32	1.85E+00	0
hs099	24	6.44E-02	0	orthregc	110	3.42E+03	0	steenbrg	159	2.85E+00	0	deconvc	166	3.50E-01	0
eigenc2	45	1.86E+00	0	brainpc6	334	3.86E+01	1	dtoc6	58	1.96E+00	0	bt1	48	6.26E-02	0
hs077	97	2.56E-01	0	brainpc7	3000	6.69E+02	-1	byrdsphr	61	1.78E-01	0	hs107	53	1.61E-01	0
blockqp3	49	8.81E-01	1	hues-mod	105	5.00E+03	0	hager3	10	3.71E-01	0	hager1	1	2.70E-02	0
lch	695	2.14E+01	1	orthrgds	158	3.98E+02	0	bt2	23	6.29E-02	0	hs35mod	15	2.09E-02	0
csfi2	68	1.95E-01	0	brainpc8	544	6.58E+01	1	gridneti	90	4.17E-01	0	gridnete	31	1.55E+00	0
corkscrw	221	7.99E+00	0	brainpc0	1159	2.63E+03	1	eigenb2	4	1.59E-02	0	blockqp5	50	4.59E-01	0
ubh5	19	1.87E+00	0	brainpc3	2052	4.55E+02	1	dtoc4	3	1.66E-01	0	gausselm	1343	3.07E+01	0
ncvxqp6	867	3.84E+01	1	cvxqp3	58	6.44E+02	0	dtoc3	21	1.09E+00	0	eg3	19	5.74E-02	0
lotschd	39	7.62E-02	0	brainpc5	3000	5.40E+02	-1	gridnetd	50	2.14E+00	0	aug2dc	17	9.09E-01	0
dualc2	19	4.79E-02	0	cvxqp2	40	4.67E+01	0	hs119	50	1.33E-01	0	ncvxqp3	468	3.05E+01	0
minperm	85	2.14E+01	0	huestis	84	3.69E+03	2	portfl4	9	1.69E-02	0	yao	3000	4.20E+01	-1
aug3dcqp	24	3.19E-01	0	brainpc1	3000	2.40E+02	-1	cvxqp1	67	1.41E+00	0	hs026	33	4.49E-02	0
bt8	17	3.89E-02	0	brainpc4	295	3.09E+01	1	bt3	18	5.68E-02	0	bloweyb	7	1.80E+00	0
qpcboei2	85	2.03E-01	0	orthrgdm	19	2.77E+00	0	steenbra	23	1.65E-01	0	static3	73	2.72E-01	4
kissing	157	2.17E+00	0	orthregd	18	2.51E+00	0	optctrl6	79	2.05E-01	0	qpnboei2	456	2.13E+00	0
alsotame	9	1.51E-02	0	brainpc9	3000	4.62E+02	-1	reading2	15	8.72E-01	0	orthrega	92	6.09E-01	0

Table 11: CUTEr set results, penalty mode ρ no Σ, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	17	4.24E-02	0	dtoc2	86	2.93E+00	0
steenbrd	121	1.74E+00	0	orthrds2	962	4.51E+00	1
bloweya	169	3.01E+01	0	eigencco	30	5.03E-02	0
dualc1	24	6.92E-02	0	lakes	435	1.43E+00	0
dualc8	19	5.64E-02	0	ncvxqp5	386	8.65E+00	0
portfl1	9	1.93E-02	0	dual1	18	1.14E-01	0
hs074	24	7.87E-02	0	odfits	17	2.80E-02	0
trainf	53	2.31E+00	0	bt6	26	7.68E-02	0
csfi1	48	1.36E-01	0	clnlbeam	187	1.26E+00	0
dual2	15	9.45E-02	0	bt7	45	1.12E-01	0
gridnetg	31	8.40E-02	0	rk23	9	1.71E-02	0
hong	24	2.49E-02	0	hs054	9	1.83E-02	0
dtoc1nb	6	2.70E-01	0	gridnetb	36	2.11E+00	0
hs014	10	1.64E-02	0	ubh1	20	1.60E+00	0
sreadin3	12	6.04E-01	0	dittert	91	9.47E-01	0
optcntrl	43	1.10E-01	0	blockqp1	34	6.38E-01	0
steenbre	1636	4.09E+01	1	ncvxqp9	327	2.50E+01	0
dnieper	44	8.20E-02	0	hs063	17	2.50E-02	0
steenbrc	97	5.84E-01	1	hs009	0	2.80E-05	0
gridneta	35	1.54E+00	0	hs067	11	1.96E-02	0
prodpl1	15	2.41E-02	0	ssebnln	3000	2.56E+01	-1
aug3dqp	23	3.02E-01	0	hs075	34	9.50E-02	0
hs049	19	2.30E-02	0				
aug2dqp	33	1.91E+00	0				
fletcher	48	1.56E-01	2				

Table 12: CUTEr set results, penalty mode ρ no Σ, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	96	2.11E+01	0	swopf	20	5.29E-02	0	smmpsf	674	1.85E+00	0	hs042	10	1.11E-02	0
bt11	68	8.47E-02	0	dallasm	36	7.68E-02	1	hs100lnp	170	2.84E-01	0	optmass	23	4.25E-02	0
hs046	34	9.07E-02	0	himmelbj	3000	1.43E+01	-1	gridnetf	38	1.76E+00	0	sseblin	117	4.02E-01	0
hs007	67	1.41E-01	0	aug3d	12	6.06E-01	0	lsnnodoc	1	3.89E-03	0	hs028	1	3.92E-03	0
hs114	118	2.88E-01	0	bt5	53	7.35E-02	0	bt12	37	6.27E-02	0	batch	33	6.23E-02	0
extrasim	11	1.53E-02	0	allinitc	41	9.17E-02	0	loadbal	15	2.32E-02	0	hs050	9	1.45E-02	0
model	35	1.69E-01	0	bt9	54	1.29E-01	0	hs081	24	6.62E-02	0	hs087	76	1.05E-01	0
disc2	45	1.32E-01	0	qpnboei1	1635	1.37E+01	1	dtoc1na	6	2.50E-01	0	hs080	16	3.76E-02	0
eigenbco	495	2.17E+00	0	trimloss	63	2.43E-01	0	genhs28	1	3.80E-03	0	hs062	7	1.32E-02	0
catenary	3000	1.14E+01	-1	dtoc1nd	22	3.40E-01	0	hs071	60	1.64E-01	0	trainh	100	8.79E+00	0
tame	9	1.39E-02	0	minc44	53	5.77E-01	0	hs078	24	2.44E-02	0	sawpath	3000	9.25E+01	-1
dtoc1l	6	3.06E-01	0	hs073	3000	4.29E+00	-1	ncvxqp1	312	1.36E+01	1	steenbrb	49	5.08E-01	0
fccu	13	2.94E-02	0	hs039	54	1.21E-01	0	avion2	93	1.69E-01	0	gouldqp2	15	4.92E-02	0
hager4	19	5.43E-01	0	catena	149	2.10E-01	0	gridnetc	43	5.42E+00	0	zigzag	31	4.58E-02	0
hs056	15	1.85E-02	0	hs053	15	3.42E-02	0	hs041	22	5.72E-02	0	maratos	74	1.70E-01	0
robot	122	3.24E-01	0	ncvxqp4	568	1.37E+01	0	hs109	60	1.47E-01	0	aljazzaf	29	4.15E-02	0
bt13	53	1.33E-01	0	ncvxqp7	262	1.55E+01	0	orthregb	47	1.14E-01	0	reading3	29	6.43E-02	0
optcdeg2	46	3.24E-01	0	hs006	2045	2.59E+00	1	prodpl0	17	1.93E-02	0	orthrdm2	14	3.60E+00	0
hs027	87	1.19E-01	0	qpcstair	642	3.53E+00	0	blockqp2	13	1.02E-01	0	eigenaco	5	2.69E-02	0
portfl2	8	1.37E-02	0	reading1	91	6.45E+00	0	hvycrash	616	2.52E+00	1	dtoc5	18	4.96E-01	0
blockqp4	11	8.82E-02	0	spanhyd	20	3.90E-02	0	linspanh	15	2.47E-02	0	hs047	31	6.76E-02	0
qpcboei1	264	1.36E+00	0	aug3dc	11	1.09E-01	0	eigena2	3	1.41E-02	0	steenbrf	94	4.36E-01	1
hs111lnp	97	2.13E-01	2	portfl3	10	1.55E-02	0	hs061	24	6.38E-02	0	degenlpa	88	1.98E-01	0
hs032	16	2.40E-02	0	hs040	7	1.16E-02	0	hs051	1	6.43E-04	0	launch	3000	1.53E+01	-1
mwright	133	2.23E-01	0	sosqp1	36	3.30E+00	1	ncvxqp8	367	3.27E+01	1	hs060	72	1.35E-01	0

Table 13: CUTEr set results, penalty mode ρ_L (linear), part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	3.60E-02	0	dixchlng	24	3.44E-02	0	bt4	261	6.87E-01	0	smbank	17	2.54E-02	0
orthrege	200	6.52E-01	0	aug2d	11	6.79E-01	0	hs99exp	91	2.28E-01	0	bloweyc	44	1.17E+01	0
hs111	1271	4.10E+00	0	gouldqp3	26	1.77E-01	0	dixchlnv	72	2.61E-01	0	hs112	17	2.05E-02	0
try-b	30	6.30E-02	0	sosqp2	32	2.85E+00	0	dual3	23	2.78E-01	0	hadamard	6	2.24E-02	0
ssnlbeam	3000	4.24E+00	-1	gridneth	25	7.50E-02	0	dtoc1nc	17	7.21E-01	0	degenlpb	68	1.54E-01	0
portfl6	8	1.27E-02	0	optctrl3	79	1.93E-01	0	hs048	1	3.86E-03	0	himmelbk	1097	4.18E+00	0
hager2	10	2.52E-01	0	hs079	159	2.01E-01	0	ncvxqp2	447	2.33E+01	0	optcdeg3	43	2.99E-01	0
qpnstair	724	4.52E+00	0	brainpc2	690	2.71E+02	1	dual4	16	6.28E-02	0	aug2dcqp	32	1.76E+00	0
hs099	24	5.57E-02	0	orthregc	110	3.44E+03	0	steenbrg	159	2.73E+00	0	deconvc	166	3.58E-01	0
eigenc2	48	2.04E+00	0	brainpc6	334	3.69E+01	1	dtoc6	52	1.59E+00	0	bt1	48	6.56E-02	0
hs077	30	9.09E-02	0	brainpc7	3000	3.56E+02	-1	byrdsphr	61	1.60E-01	0	hs107	3000	8.17E+00	-1
blockqp3	49	9.12E-01	1	hues-mod	102	4.83E+03	0	hager3	10	3.51E-01	0	hager1	1	2.90E-02	0
lch	3000	1.20E+02	-1	orthrgds	158	4.18E+02	0	bt2	30	7.88E-02	0	hs35mod	15	1.92E-02	0
csfi2	68	2.00E-01	0	brainpc8	544	7.52E+01	1	gridneti	20	6.71E-02	0	gridnete	31	1.58E+00	0
corkscrw	259	9.66E+00	0	brainpc0	3000	3.09E+03	-1	eigenb2	4	2.59E-02	0	blockqp5	50	4.59E-01	0
ubh5	19	1.93E+00	0	brainpc3	2052	4.60E+02	1	dtoc4	3	1.42E-01	0	gausselm	1343	2.98E+01	0
ncvxqp6	867	3.90E+01	1	cvxqp3	58	7.43E+02	0	dtoc3	17	8.45E-01	0	eg3	19	4.05E-02	0
lotschd	38	7.50E-02	0	brainpc5	3000	4.90E+02	-1	gridnetd	41	1.68E+00	0	aug2dc	17	7.71E-01	0
dualc2	19	4.69E-02	0	cvxqp2	39	5.10E+01	0	hs119	52	1.37E-01	0	ncvxqp3	468	3.07E+01	0
minperm	85	2.15E+01	0	huestis	117	4.98E+03	2	portfl4	9	8.97E-03	0	yao	3000	4.03E+01	-1
aug3dcqp	24	3.25E-01	0	brainpc1	3000	1.69E+02	-1	cvxqp1	67	1.39E+00	0	hs026	33	4.30E-02	0
bt8	17	4.55E-02	0	brainpc4	295	2.67E+01	1	bt3	18	4.88E-02	0	bloweyb	7	1.67E+00	0
qpcboei2	85	2.07E-01	0	orthrgdm	19	2.32E+00	0	steenbra	23	1.64E-01	0	static3	73	2.30E-01	4
kissing	157	2.13E+00	0	orthregd	18	1.96E+00	0	optctrl6	79	2.01E-01	0	qpnboei2	533	2.71E+00	1
alsotame	9	1.44E-02	0	brainpc9	3000	3.85E+02	-1	reading2	15	8.65E-01	0	orthrega	126	7.89E-01	0

Table 14: CUTEr set results, penalty mode ρ_L (linear), part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	18	4.91E-02	0	dtoc2	86	2.59E+00	0
steenbrd	121	1.65E+00	0	orthrds2	512	2.21E+00	1
bloweya	169	3.06E+01	0	eigencco	30	5.30E-02	0
dualc1	24	6.97E-02	0	lakes	530	1.61E+00	0
dualc8	19	6.88E-02	0	ncvxqp5	386	8.48E+00	0
portfl1	9	9.95E-03	0	dual1	18	1.05E-01	0
hs074	24	6.77E-02	0	odfits	17	2.70E-02	0
trainf	53	2.31E+00	0	bt6	21	5.17E-02	0
csfi1	43	1.38E-01	0	clnlbeam	187	1.21E+00	0
dual2	15	9.63E-02	0	bt7	51	1.15E-01	0
gridnetg	28	7.75E-02	0	rk23	9	1.74E-02	0
hong	24	3.51E-02	0	hs054	9	1.49E-02	0
dtoc1nb	6	2.39E-01	0	gridnetb	36	2.03E+00	0
hs014	10	1.54E-02	0	ubh1	20	1.58E+00	0
sreadin3	12	6.30E-01	0	dittert	91	8.75E-01	0
optcntrl	49	1.37E-01	0	blockqp1	34	6.06E-01	0
steenbre	1636	4.06E+01	1	ncvxqp9	326	2.38E+01	0
dnieper	44	8.02E-02	0	hs063	17	2.77E-02	0
steenbrc	96	4.68E-01	1	hs009	0	1.63E-03	0
gridneta	40	1.48E+00	0	hs067	11	1.92E-02	0
prodpl1	15	2.06E-02	0	ssebnln	3000	2.51E+01	-1
aug3dqp	23	2.80E-01	0	hs075	37	9.29E-02	0
hs049	19	2.18E-02	0				
aug2dqp	33	1.80E+00	0				
fletcher	48	1.33E-01	2				

Table 15: CUTEr set results, penalty mode ρ_L (linear), part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	56	1.11E+01	0	swopf	22	6.43E-02	0	smmpsf	674	2.03E+00	0	hs042	10	1.41E-02	0
bt11	68	8.48E-02	0	dallasm	36	8.38E-02	1	hs100lnp	175	3.11E-01	0	optmass	23	4.32E-02	0
hs046	61	1.84E-01	0	himmelbj	3000	1.59E+01	-1	gridnetf	35	1.94E+00	0	sseblin	149	5.13E-01	0
hs007	52	1.31E-01	0	aug3d	12	6.93E-01	0	lsnnodoc	1	5.42E-03	0	hs028	1	6.03E-03	0
hs114	98	2.47E-01	0	bt5	53	8.38E-02	0	bt12	37	6.89E-02	0	batch	33	6.13E-02	0
extrasim	11	2.35E-02	0	allinitc	29	5.91E-02	2	loadbal	15	2.52E-02	0	hs050	9	1.47E-02	0
model	35	1.82E-01	0	bt9	149	4.29E-01	0	hs081	43	1.22E-01	0	hs087	76	1.07E-01	0
disc2	142	4.56E-01	0	qpnboei1	920	6.84E+00	2	dtoc1na	6	2.94E-01	0	hs080	28	6.76E-02	0
eigenbco	495	2.58E+00	0	trimloss	128	5.78E-01	0	genhs28	1	1.40E-05	0	hs062	7	1.45E-02	0
catenary	108	6.19E-01	2	dtoc1nd	22	5.67E-01	0	hs071	44	1.19E-01	0	trainh	367	3.45E+01	0
tame	9	1.07E-02	0	minc44	43	5.18E-01	0	hs078	24	2.99E-02	0	sawpath	3000	3.20E+02	-1
dtoc1l	6	3.56E-01	0	hs073	3000	4.45E+00	-1	ncvxqp1	3000	9.56E+01	-1	steenbrb	49	5.37E-01	0
fccu	10	3.39E-02	0	hs039	149	4.24E-01	0	avion2	93	1.77E-01	0	gouldqp2	15	5.01E-02	0
hager4	19	7.65E-01	0	catena	149	2.29E-01	0	gridnetc	35	1.18E+00	0	zigzag	31	5.16E-02	0
hs056	15	2.75E-02	0	hs053	14	3.96E-02	0	hs041	19	5.22E-02	0	maratos	146	3.73E-01	0
robot	105	2.90E-01	0	ncvxqp4	207	4.04E+00	0	hs109	74	1.90E-01	0	aljazzaf	29	4.78E-02	0
bt13	41	1.12E-01	0	ncvxqp7	240	1.06E+01	2	orthregb	50	1.35E-01	0	reading3	29	8.14E-02	0
optcdeg2	43	3.58E-01	0	hs006	2045	2.75E+00	1	prodpl0	17	2.59E-02	0	orthrdm2	14	4.02E+00	0
hs027	87	1.33E-01	0	qpcstair	258	1.63E+00	2	blockqp2	13	1.01E-01	0	eigenaco	5	2.28E-02	0
portfl2	8	1.44E-02	0	reading1	84	6.94E+00	0	hvycrash	250	9.88E-01	1	dtoc5	14	4.57E-01	0
blockqp4	11	9.84E-02	0	spanhyd	20	4.59E-02	0	linspanh	15	2.31E-02	0	hs047	130	3.76E-01	0
qpcboei1	264	1.56E+00	2	aug3dc	11	1.28E-01	0	eigena2	3	9.60E-03	0	steenbrf	382	1.78E+00	0
hs111lnp	78	1.73E-01	2	portfl3	10	2.21E-02	0	hs061	40	1.03E-01	0	degenlpa	87	1.97E-01	2
hs032	16	2.43E-02	0	hs040	7	1.26E-02	0	hs051	1	9.40E-05	0	launch	123	4.18E-01	2
mwright	145	2.74E-01	0	sosqp1	33	3.01E+00	0	ncvxqp8	1920	1.35E+02	2	hs060	70	1.34E-01	0

Table 16: CUTEr set results, penalty mode ρ_0 (fixed), part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	5.12E-02	0	dixchlng	24	3.70E-02	0	bt4	102	3.27E-01	2	smbank	17	3.55E-02	0
orthrege	3000	1.75E+01	-1	aug2d	11	7.25E-01	0	hs99exp	156	4.58E-01	-2	bloweyc	44	1.21E+01	0
hs111	229	7.28E-01	0	gouldqp3	20	1.56E-01	0	dixchlnv	72	2.65E-01	0	hs112	17	2.01E-02	0
try-b	35	7.98E-02	0	sosqp2	37	3.39E+00	0	dual3	22	2.16E-01	0	hadamard	6	3.32E-02	0
ssnlbeam	3000	4.72E+00	-1	gridneth	17	5.40E-02	0	dtoc1nc	17	8.23E-01	0	degenlpb	56	1.28E-01	2
portfl6	8	1.45E-02	0	optctrl3	79	2.07E-01	0	hs048	1	3.33E-03	0	himmelbk	150	5.44E-01	0
hager2	10	2.66E-01	0	hs079	159	2.19E-01	0	ncvxqp2	268	7.26E+00	2	optcdeg3	35	2.60E-01	0
qpnstair	274	1.76E+00	2	brainpc2	279	6.34E+01	1	dual4	16	7.33E-02	0	aug2dcqp	32	1.87E+00	0
hs099	17	4.69E-02	0	orthregc	110	3.34E+03	0	steenbrg	159	2.81E+00	0	deconvc	166	3.71E-01	0
eigenc2	47	3.19E+00	0	brainpc6	3000	4.27E+02	-1	dtoc6	30	9.82E-01	2	bt1	48	6.81E-02	0
hs077	41	1.17E-01	0	brainpc7	3000	5.42E+02	-1	byrdsphr	86	2.21E-01	0	hs107	35	9.78E-02	2
blockqp3	185	2.84E+00	0	hues-mod	120	5.36E+03	2	hager3	10	3.87E-01	0	hager1	1	3.31E-02	0
lch	785	2.32E+01	0	orthrgds	158	3.88E+02	0	bt2	27	6.96E-02	0	hs35mod	15	2.03E-02	0
csfi2	68	2.28E-01	0	brainpc8	208	1.34E+01	1	gridneti	18	5.54E-02	0	gridnete	19	1.05E+00	0
corkscrw	1009	4.97E+01	0	brainpc0	3000	2.85E+03	-1	eigenb2	4	2.12E-02	0	blockqp5	50	4.64E-01	0
ubh5	15	1.67E+00	0	brainpc3	287	3.12E+01	1	dtoc4	3	1.61E-01	0	gausselm	1343	3.02E+01	0
ncvxqp6	316	5.96E+00	0	cvxqp3	58	6.96E+02	0	dtoc3	12	6.51E-01	0	eg3	19	5.13E-02	0
lotschd	32	6.00E-02	0	brainpc5	567	6.70E+01	1	gridnetd	26	1.16E+00	0	aug2dc	17	8.46E-01	0
dualc2	19	4.92E-02	0	cvxqp2	34	4.30E+01	0	hs119	65	1.74E-01	0	ncvxqp3	3000	1.04E+02	-1
minperm	92	2.36E+01	0	huestis	74	3.19E+03	2	portfl4	9	1.50E-02	0	yao	1386	1.99E+01	2
aug3dcqp	24	3.18E-01	0	brainpc1	3000	2.03E+02	-1	cvxqp1	508	1.27E+01	2	hs026	33	4.75E-02	0
bt8	14	4.11E-02	0	brainpc4	672	8.14E+01	1	bt3	14	4.43E-02	0	bloweyb	7	1.76E+00	0
qpcboei2	85	2.31E-01	0	orthrgdm	19	2.43E+00	0	steenbra	23	1.88E-01	0	static3	73	2.39E-01	4
kissing	187	2.69E+00	0	orthregd	18	2.27E+00	0	optctrl6	79	2.22E-01	0	qpnboei2	391	1.82E+00	2
alsotame	9	1.50E-02	0	brainpc9	1283	1.42E+02	1	reading2	15	9.85E-01	0	orthrega	105	7.12E-01	0

Table 17: CUTEr set results, penalty mode ρ_0 (fixed), part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	13	3.83E-02	0	dtoc2	86	2.65E+00	0
steenbrd	121	1.70E+00	0	orthrds2	1159	5.01E+00	2
bloweya	169	2.99E+01	0	eigencco	30	4.78E-02	0
dualc1	24	7.73E-02	0	lakes	754	2.33E+00	0
dualc8	19	6.95E-02	0	ncvxqp5	386	8.48E+00	0
portfl1	9	1.65E-02	0	dual1	18	1.01E-01	0
hs074	18	5.57E-02	0	odfits	17	2.07E-02	0
trainf	53	2.34E+00	0	bt6	26	6.82E-02	0
csfi1	31	7.10E-02	0	clnlbeam	187	1.21E+00	0
dual2	15	1.00E-01	0	bt7	75	1.89E-01	0
gridnetg	27	7.67E-02	0	rk23	9	1.07E-02	0
hong	24	3.74E-02	0	hs054	9	1.61E-02	0
dtoc1nb	6	2.68E-01	0	gridnetb	19	1.03E+00	0
hs014	10	1.70E-02	0	ubh1	15	1.24E+00	0
sreadin3	12	6.45E-01	0	dittert	91	9.46E-01	0
optcntrl	54	1.44E-01	0	blockqp1	31	5.76E-01	0
steenbre	1636	3.97E+01	1	ncvxqp9	519	2.28E+01	2
dnieper	44	7.96E-02	0	hs063	17	3.04E-02	0
steenbrc	3000	3.36E+01	-1	hs009	0	3.72E-03	0
gridneta	29	9.50E-01	0	hs067	11	2.07E-02	0
prodpl1	15	2.72E-02	0	ssebnln	149	5.23E-01	0
aug3dqp	23	2.87E-01	0	hs075	27	7.55E-02	2
hs049	19	2.51E-02	0				
aug2dqp	33	1.90E+00	0				
fletcher	21	6.18E-02	2				

Table 18: CUTEr set results, penalty mode ρ_0 (fixed), part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	97	2.05E+01	0	swopf	31	1.03E-01	0	smmpsf	674	2.14E+00	0	hs042	10	1.84E-02	0
bt11	68	8.34E-02	0	dallasm	36	7.52E-02	1	hs100lnp	169	3.16E-01	0	optmass	23	4.28E-02	0
hs046	30	7.97E-02	0	himmelbj	3000	1.54E+01	-1	gridnetf	52	4.63E+00	1	sseblin	73	3.02E-01	0
hs007	47	1.05E-01	0	aug3d	12	6.28E-01	0	lsnnodoc	1	2.57E-03	0	hs028	1	4.84E-03	0
hs114	129	3.65E-01	0	bt5	53	7.30E-02	0	bt12	37	6.32E-02	0	batch	33	7.00E-02	0
extrasim	11	1.58E-02	0	allinitc	46	1.05E-01	0	loadbal	15	3.01E-02	0	hs050	9	1.18E-02	0
model	35	1.70E-01	0	bt9	70	1.92E-01	0	hs081	28	9.94E-02	0	hs087	76	1.16E-01	0
disc2	54	1.61E-01	0	qpnboei1	1526	1.32E+01	0	dtoc1na	6	4.05E-01	0	hs080	16	4.26E-02	0
eigenbco	495	2.22E+00	0	trimloss	118	5.04E-01	2	genhs28	1	5.77E-03	0	hs062	7	1.52E-02	0
catenary	739	3.47E+00	0	dtoc1nd	22	4.59E-01	0	hs071	87	3.41E-01	0	trainh	3000	3.62E+02	-1
tame	9	1.09E-02	0	minc44	32	4.14E-01	0	hs078	24	3.85E-02	0	sawpath	3000	2.13E+02	-1
dtoc1l	6	3.02E-01	0	hs073	3000	4.32E+00	-1	ncvxqp1	234	8.41E+00	0	steenbrb	49	6.00E-01	0
fccu	12	3.70E-02	0	hs039	70	1.80E-01	0	avion2	93	1.80E-01	0	gouldqp2	15	5.13E-02	0
hager4	19	5.61E-01	0	catena	149	2.09E-01	0	gridnetc	49	1.83E+00	1	zigzag	31	6.15E-02	0
hs056	15	1.45E-02	0	hs053	14	3.63E-02	0	hs041	20	5.35E-02	0	maratos	48	1.41E-01	0
robot	725	3.40E+00	0	ncvxqp4	601	1.18E+01	0	hs109	62	1.93E-01	0	aljazzaf	29	4.57E-02	0
bt13	63	1.73E-01	0	ncvxqp7	665	3.01E+01	0	orthregb	38	9.32E-02	0	reading3	29	8.14E-02	0
optcdeg2	390	3.02E+00	0	hs006	2045	2.74E+00	1	prodpl0	17	1.62E-02	0	orthrdm2	14	3.84E+00	0
hs027	87	1.18E-01	0	qpcstair	537	3.91E+00	0	blockqp2	13	1.11E-01	0	eigenaco	5	2.34E-02	0
portfl2	8	1.38E-02	0	reading1	98	8.22E+03	-4	hvycrash	1129	6.55E+00	0	dtoc5	22	7.54E-01	0
blockqp4	11	8.65E-02	0	spanhyd	20	3.93E-02	0	linspanh	15	2.88E-02	0	hs047	59	2.10E-01	0
qpcboei1	965	7.73E+00	0	aug3dc	11	1.13E-01	0	eigena2	3	1.63E-02	0	steenbrf	74	4.09E-01	1
hs111lnp	112	2.67E-01	2	portfl3	10	1.61E-02	0	hs061	21	6.33E-02	0	degenlpa	87	2.56E-01	0
hs032	16	2.00E-02	0	hs040	7	1.23E-02	0	hs051	1	4.70E-03	0	launch	3000	3.33E+01	-1
mwright	143	2.62E-01	0	sosqp1	36	3.85E+00	0	ncvxqp8	1558	6.35E+01	0	hs060	68	1.39E-01	0

Table 19: CUTEr set results, penalty mode $\frac{1}{\rho}$, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	3.79E-02	0	dixchlng	24	3.29E-02	0	bt4	81	2.89E-01	0	smbank	17	3.75E-02	0
orthrege	3000	1.64E+01	-1	aug2d	11	7.26E-01	0	hs99exp	56	2.05E-01	0	bloweyc	44	1.85E+01	0
hs111	126	3.69E-01	0	gouldqp3	30	3.49E-01	0	dixchlnv	72	4.62E-01	0	hs112	17	4.13E-02	0
try-b	32	7.64E-02	0	sosqp2	39	6.71E+00	0	dual3	25	9.22E-01	0	hadamard	6	4.53E-02	0
ssnlbeam	3000	4.40E+00	-1	gridneth	19	6.88E-02	0	dtoc1nc	17	1.44E+00	0	degenlpb	58	2.10E-01	0
portfl6	8	1.06E-02	0	optctrl3	79	2.17E-01	0	hs048	1	1.14E-02	0	himmelbk	161	1.41E+00	1
hager2	10	2.81E-01	0	hs079	159	2.25E-01	0	ncvxqp2	959	6.71E+01	0	optcdeg3	80	1.15E+00	1
qpnstair	594	4.55E+00	1	brainpc2	190	2.50E+01	1	dual4	16	9.47E-02	0	aug2dcqp	32	3.09E+00	0
hs099	21	6.42E-02	0	orthregc	110	3.09E+03	0	steenbrg	159	3.46E+00	0	deconvc	166	7.13E-01	0
eigenc2	29	1.66E+00	0	brainpc6	193	2.36E+01	1	dtoc6	40	1.84E+00	0	bt1	48	9.69E-02	0
hs077	25	7.14E-02	0	brainpc7	107	6.00E+00	1	byrdsphr	67	2.49E-01	0	hs107	3000	1.37E+01	-1
blockqp3	354	1.20E+01	0	hues-mod	123	5.71E+03	0	hager3	10	5.30E-01	0	hager1	1	5.38E-02	0
lch	918	2.85E+01	0	orthrgds	158	4.48E+02	0	bt2	22	6.79E-02	0	hs35mod	15	3.16E-02	0
csfi2	60	1.75E-01	0	brainpc8	160	1.06E+01	1	gridneti	27	1.24E-01	0	gridnete	26	2.68E+00	0
corkscrw	3000	2.56E+02	-1	brainpc0	36	2.44E+03	1	eigenb2	4	2.54E-02	0	blockqp5	50	6.36E-01	0
ubh5	18	2.10E+00	0	brainpc3	260	1.62E+01	1	dtoc4	3	2.06E-01	0	gausselm	1343	3.99E+01	0
ncvxqp6	780	1.71E+01	0	cvxqp3	58	5.68E+02	0	dtoc3	18	1.46E+00	0	eg3	19	8.70E-02	0
lotschd	37	7.46E-02	0	brainpc5	576	2.10E+03	1	gridnetd	47	4.19E+00	0	aug2dc	17	1.27E+00	0
dualc2	19	4.36E-02	0	cvxqp2	49	4.22E+01	0	hs119	42	1.86E-01	0	ncvxqp3	1011	8.00E+01	0
minperm	33	7.74E+00	0	huestis	3000	1.34E+05	-1	portfl4	9	2.72E-02	0	yao	362	8.03E+00	0
aug3dcqp	24	3.40E-01	0	brainpc1	3000	3.24E+02	-1	cvxqp1	187	4.17E+00	0	hs026	33	6.84E-02	0
bt8	17	4.80E-02	0	brainpc4	312	4.04E+01	1	bt3	17	5.81E-02	0	bloweyb	7	3.20E+00	0
qpcboei2	85	2.68E-01	0	orthrgdm	19	3.84E+00	0	steenbra	23	1.85E-01	0	static3	73	4.72E-01	4
kissing	1123	1.99E+01	0	orthregd	18	3.69E+00	0	optctrl6	79	2.56E-01	0	qpnboei2	3000	2.94E+01	-1
alsotame	9	1.75E-02	0	brainpc9	122	1.15E+01	0	reading2	15	1.09E+00	0	orthrega	136	1.64E+00	0

Table 20: CUTEr set results, penalty mode $\frac{1}{\rho}$, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	14	4.93E-02	0	dtoc2	86	3.49E+00	0
steenbrd	121	2.25E+00	0	orthrds2	3000	3.09E+01	-1
bloweya	169	4.03E+01	0	eigencco	30	7.95E-02	0
dualc1	24	8.53E-02	0	lakes	457	2.30E+00	0
dualc8	19	8.81E-02	0	ncvxqp5	386	1.07E+01	0
portfl1	9	1.28E-02	0	dual1	18	2.18E-01	0
hs074	21	8.00E-02	0	odfits	17	3.39E-02	0
trainf	53	2.94E+00	0	bt6	23	9.27E-02	0
csfi1	34	1.29E-01	0	clnlbeam	187	1.88E+00	0
dual2	15	1.54E-01	0	bt7	64	2.51E-01	0
gridnetg	38	2.26E-01	0	rk23	9	2.99E-02	0
hong	24	4.17E-02	0	hs054	9	1.60E-02	0
dtoc1nb	6	4.33E-01	0	gridnetb	24	2.20E+00	0
hs014	10	1.31E-02	0	ubh1	16	1.89E+00	0
sreadin3	12	8.57E-01	0	dittert	91	1.23E+00	0
optcntrl	43	1.73E-01	0	blockqp1	35	8.07E-01	0
steenbre	1636	5.12E+01	1	ncvxqp9	3000	1.87E+02	-1
dnieper	44	1.34E-01	0	hs063	17	3.65E-02	0
steenbrc	76	6.71E-01	1	hs009	0	4.12E-03	0
gridneta	42	4.40E+00	0	hs067	11	2.66E-02	0
prodpl1	15	3.74E-02	0	ssebnln	746	5.35E+00	0
aug3dqp	23	5.11E-01	0	hs075	30	1.11E-01	0
hs049	19	3.67E-02	0				
aug2dqp	33	3.03E+00	0				
fletcher	189	8.89E-01	0				

Table 21: CUTEr set results, penalty mode $\frac{1}{\rho}$, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	97	2.08E+01	0	swopf	25	8.74E-02	0	smmpsf	674	2.14E+00	0	hs042	10	1.56E-02	0
bt11	68	8.02E-02	0	dallasm	36	7.69E-02	1	hs100lnp	169	3.23E-01	0	optmass	23	3.48E-02	0
hs046	30	7.94E-02	0	himmelbj	3000	1.48E+01	-1	gridnetf	52	5.23E+00	1	sseblin	102	4.79E-01	0
hs007	47	1.05E-01	0	aug3d	12	6.31E-01	0	lsnnodoc	1	8.40E-05	0	hs028	1	5.33E-03	0
hs114	118	2.76E-01	0	bt5	53	7.00E-02	0	bt12	37	6.90E-02	0	batch	33	5.96E-02	0
extrasim	11	1.48E-02	0	allinitc	45	1.07E-01	0	loadbal	15	2.94E-02	0	hs050	9	1.06E-02	0
model	35	1.68E-01	0	bt9	70	1.85E-01	0	hs081	28	9.65E-02	0	hs087	76	1.18E-01	0
disc2	54	1.55E-01	0	qpnboei1	1531	1.27E+01	0	dtoc1na	6	3.61E-01	0	hs080	16	4.21E-02	0
eigenbco	495	2.17E+00	0	trimloss	314	1.49E+00	0	genhs28	1	5.17E-04	0	hs062	7	1.37E-02	0
catenary	739	3.46E+00	0	dtoc1nd	22	3.78E-01	0	hs071	115	4.71E-01	0	trainh	3000	4.19E+02	-1
tame	9	1.40E-02	0	minc44	32	4.04E-01	0	hs078	24	4.86E-02	0	sawpath	3000	1.97E+02	-1
dtoc1l	6	3.10E-01	0	hs073	3000	4.27E+00	-1	ncvxqp1	234	8.57E+00	0	steenbrb	49	5.70E-01	0
fccu	12	2.85E-02	0	hs039	70	1.91E-01	0	avion2	93	1.68E-01	0	gouldqp2	15	5.03E-02	0
hager4	19	5.70E-01	0	catena	149	2.13E-01	0	gridnetc	49	1.81E+00	1	zigzag	31	5.45E-02	0
hs056	15	2.26E-02	0	hs053	14	3.58E-02	0	hs041	20	5.67E-02	0	maratos	48	1.16E-01	0
robot	725	3.46E+00	0	ncvxqp4	601	1.18E+01	0	hs109	62	1.77E-01	0	aljazzaf	29	4.24E-02	0
bt13	78	1.99E-01	0	ncvxqp7	665	3.02E+01	0	orthregb	38	8.69E-02	0	reading3	29	7.80E-02	0
optcdeg2	390	3.09E+00	0	hs006	2045	2.52E+00	1	prodpl0	17	2.76E-02	0	orthrdm2	14	4.36E+00	0
hs027	87	1.16E-01	0	qpcstair	537	3.98E+00	0	blockqp2	13	1.02E-01	0	eigenaco	5	3.80E-02	0
portfl2	8	1.36E-02	0	reading1	98	8.20E+03	-4	hvycrash	1129	6.43E+00	0	dtoc5	22	9.89E-01	0
blockqp4	11	8.84E-02	0	spanhyd	20	4.52E-02	0	linspanh	15	3.04E-02	0	hs047	59	2.23E-01	0
qpcboei1	1065	7.94E+00	0	aug3dc	11	1.43E-01	0	eigena2	3	1.56E-02	0	steenbrf	74	4.57E-01	1
hs111lnp	112	2.62E-01	2	portfl3	10	2.59E-02	0	hs061	21	4.94E-02	0	degenlpa	87	3.18E-01	0
hs032	16	2.23E-02	0	hs040	7	1.23E-02	0	hs051	1	3.23E-03	0	launch	3000	3.27E+01	-1
mwright	143	2.66E-01	0	sosqp1	36	3.98E+00	0	ncvxqp8	1558	6.26E+01	0	hs060	68	1.26E-01	0

Table 22: CUTEr set results, penalty mode $\frac{1}{\rho}$ no Σ, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	4.23E-02	0	dixchlng	24	3.77E-02	0	bt4	81	2.90E-01	0	smbank	17	3.66E-02	0
orthrege	3000	1.71E+01	-1	aug2d	11	7.11E-01	0	hs99exp	56	1.94E-01	0	bloweyc	44	1.72E+01	0
hs111	126	4.09E-01	0	gouldqp3	30	3.63E-01	0	dixchlnv	72	4.64E-01	0	hs112	17	1.43E-02	0
try-b	32	8.20E-02	0	sosqp2	39	6.70E+00	0	dual3	25	7.62E-01	0	hadamard	6	3.01E-02	0
ssnlbeam	3000	4.50E+00	-1	gridneth	19	5.30E-02	0	dtoc1nc	17	1.58E+00	0	degenlpb	58	1.60E-01	0
portfl6	8	1.65E-02	0	optctrl3	79	2.10E-01	0	hs048	1	6.83E-03	0	himmelbk	161	9.37E-01	1
hager2	10	2.70E-01	0	hs079	159	2.22E-01	0	ncvxqp2	959	6.62E+01	0	optcdeg3	80	9.82E-01	1
qpnstair	531	4.17E+00	1	brainpc2	190	2.39E+01	1	dual4	16	1.13E-01	0	aug2dcqp	32	2.68E+00	0
hs099	21	5.20E-02	0	orthregc	110	3.11E+03	0	steenbrg	159	3.45E+00	0	deconvc	166	5.06E-01	0
eigenc2	29	1.89E+00	0	brainpc6	193	2.37E+01	1	dtoc6	40	1.75E+00	0	bt1	48	7.65E-02	0
hs077	25	9.13E-02	0	brainpc7	107	6.07E+00	1	byrdsphr	67	2.10E-01	0	hs107	3000	1.11E+01	-1
blockqp3	354	1.18E+01	0	hues-mod	123	5.72E+03	0	hager3	10	4.82E-01	0	hager1	1	4.31E-02	0
lch	918	2.79E+01	0	orthrgds	158	4.45E+02	0	bt2	22	6.81E-02	0	hs35mod	15	4.21E-02	0
csfi2	60	1.86E-01	0	brainpc8	160	1.04E+01	1	gridneti	27	1.22E-01	0	gridnete	26	2.19E+00	0
corkscrw	3000	2.38E+02	-1	brainpc0	36	2.36E+03	1	eigenb2	4	2.53E-02	0	blockqp5	50	5.35E-01	0
ubh5	18	2.04E+00	0	brainpc3	260	1.58E+01	1	dtoc4	3	2.06E-01	0	gausselm	1343	3.95E+01	0
ncvxqp6	780	1.72E+01	0	cvxqp3	58	5.68E+02	0	dtoc3	18	1.21E+00	0	eg3	19	5.52E-02	0
lotschd	37	8.10E-02	0	brainpc5	576	2.05E+03	1	gridnetd	47	4.51E+00	0	aug2dc	17	8.80E-01	0
dualc2	19	5.36E-02	0	cvxqp2	49	4.58E+01	0	hs119	42	1.27E-01	0	ncvxqp3	1011	7.55E+01	0
minperm	33	7.66E+00	0	huestis	3000	1.35E+05	-1	portfl4	9	1.68E-02	0	yao	363	7.08E+00	0
aug3dcqp	24	3.50E-01	0	brainpc1	3000	3.37E+02	-1	cvxqp1	187	3.94E+00	0	hs026	33	7.37E-02	0
bt8	17	4.75E-02	0	brainpc4	312	4.10E+01	1	bt3	17	8.06E-02	0	bloweyb	7	2.62E+00	0
qpcboei2	85	2.15E-01	0	orthrgdm	19	4.24E+00	0	steenbra	23	2.12E-01	0	static3	73	4.15E-01	4
kissing	495	8.71E+00	1	orthregd	18	4.01E+00	0	optctrl6	79	2.54E-01	0	qpnboei2	3000	2.68E+01	-1
alsotame	9	1.05E-02	0	brainpc9	122	1.22E+01	0	reading2	15	1.18E+00	0	orthrega	136	1.51E+00	0

Table 23: CUTEr set results, penalty mode $\frac{1}{\rho}$ no Σ, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	14	5.06E-02	0	dtoc2	86	4.03E+00	0
steenbrd	121	2.21E+00	0	orthrds2	3000	3.12E+01	-1
bloweya	169	4.01E+01	0	eigencco	30	7.79E-02	0
dualc1	24	9.67E-02	0	lakes	457	2.48E+00	0
dualc8	19	1.01E-01	0	ncvxqp5	386	1.10E+01	0
portfl1	9	2.00E-02	0	dual1	18	1.90E-01	0
hs074	21	9.20E-02	0	odfits	17	3.38E-02	0
trainf	53	3.43E+00	0	bt6	23	7.52E-02	0
csfi1	34	1.30E-01	0	clnlbeam	187	1.83E+00	0
dual2	15	2.38E-01	0	bt7	64	2.25E-01	0
gridnetg	38	2.46E-01	0	rk23	9	1.67E-02	0
hong	24	6.03E-02	0	hs054	9	2.41E-02	0
dtoc1nb	6	5.37E-01	0	gridnetb	24	2.13E+00	0
hs014	10	3.01E-02	0	ubh1	16	1.84E+00	0
sreadin3	12	1.03E+00	0	dittert	91	1.19E+00	0
optcntrl	59	3.39E-01	0	blockqp1	35	8.92E-01	0
steenbre	1636	4.90E+01	1	ncvxqp9	3000	1.89E+02	-1
dnieper	44	1.25E-01	0	hs063	17	3.11E-02	0
steenbrc	76	6.07E-01	1	hs009	0	6.82E-03	0
gridneta	42	3.50E+00	0	hs067	11	2.82E-02	0
prodpl1	15	3.15E-02	0	ssebnln	780	5.83E+00	0
aug3dqp	23	4.10E-01	0	hs075	30	1.06E-01	0
hs049	19	3.75E-02	0				
aug2dqp	33	2.68E+00	0				
fletcher	284	1.37E+00	0				

Table 24: CUTEr set results, penalty mode $\frac{1}{\rho}$ no Σ, part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	97	1.97E+01	0	swopf	25	8.55E-02	0	smmpsf	674	2.00E+00	0	hs042	10	1.85E-02	0
bt11	68	8.92E-02	0	dallasm	36	7.34E-02	1	hs100lnp	169	3.28E-01	0	optmass	23	4.28E-02	0
hs046	20	5.70E-02	0	himmelbj	3000	1.47E+01	-1	gridnetf	43	2.86E+00	0	sseblin	102	5.32E-01	0
hs007	47	1.05E-01	0	aug3d	12	6.09E-01	0	lsnnodoc	1	4.63E-03	0	hs028	1	1.56E-03	0
hs114	118	3.00E-01	0	bt5	53	7.67E-02	0	bt12	37	6.34E-02	0	batch	33	6.77E-02	0
extrasim	11	1.70E-02	0	allinitc	45	1.05E-01	0	loadbal	15	2.99E-02	0	hs050	9	2.02E-02	0
model	35	1.70E-01	0	bt9	52	1.35E-01	0	hs081	28	7.88E-02	0	hs087	76	1.23E-01	0
disc2	54	1.61E-01	0	qpnboei1	1466	1.24E+01	0	dtoc1na	6	3.08E-01	0	hs080	16	4.81E-02	0
eigenbco	495	2.18E+00	0	trimloss	201	8.89E-01	0	genhs28	1	2.47E-03	0	hs062	7	1.31E-02	0
catenary	352	1.79E+00	0	dtoc1nd	22	4.01E-01	0	hs071	115	4.48E-01	0	trainh	3000	4.05E+02	-1
tame	9	1.38E-02	0	minc44	32	3.98E-01	0	hs078	24	3.63E-02	0	sawpath	3000	2.10E+02	-1
dtoc1l	6	3.02E-01	0	hs073	3000	4.23E+00	-1	ncvxqp1	234	8.10E+00	0	steenbrb	49	6.11E-01	0
fccu	12	3.08E-02	0	hs039	52	1.36E-01	0	avion2	93	1.75E-01	0	gouldqp2	15	5.45E-02	0
hager4	19	5.72E-01	0	catena	149	2.13E-01	0	gridnetc	58	3.78E+00	0	zigzag	31	6.77E-02	0
hs056	15	1.83E-02	0	hs053	15	4.23E-02	0	hs041	20	5.67E-02	0	maratos	48	1.45E-01	0
robot	121	3.38E-01	0	ncvxqp4	601	1.16E+01	0	hs109	62	1.91E-01	0	aljazzaf	29	4.75E-02	0
bt13	79	2.13E-01	1	ncvxqp7	665	2.98E+01	0	orthregb	38	8.91E-02	0	reading3	29	9.50E-02	0
optcdeg2	190	1.58E+00	0	hs006	2045	2.61E+00	1	prodpl0	17	2.16E-02	0	orthrdm2	14	4.28E+00	0
hs027	87	1.18E-01	0	qpcstair	561	3.48E+00	0	blockqp2	13	1.12E-01	0	eigenaco	5	2.67E-02	0
portfl2	8	1.42E-02	0	reading1	98	8.25E+03	-4	hvycrash	1129	6.59E+00	0	dtoc5	22	8.01E-01	0
blockqp4	11	8.95E-02	0	spanhyd	20	5.35E-02	0	linspanh	15	2.75E-02	0	hs047	44	1.47E-01	0
qpcboei1	783	5.87E+00	0	aug3dc	11	1.43E-01	0	eigena2	3	1.63E-02	0	steenbrf	74	4.43E-01	1
hs111lnp	112	2.57E-01	2	portfl3	10	1.45E-02	0	hs061	21	5.05E-02	0	degenlpa	87	2.74E-01	0
hs032	16	2.37E-02	0	hs040	7	1.10E-02	0	hs051	1	1.52E-03	0	launch	3000	3.35E+01	-1
mwright	132	2.26E-01	0	sosqp1	36	4.31E+00	0	ncvxqp8	1558	6.53E+01	0	hs060	68	1.43E-01	0

Table 25: CUTEr set results, penalty mode $\frac{1}{\rho_L}$(linear), part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	4.05E-02	0	dixchlng	24	4.31E-02	0	bt4	81	1.92E-01	0	smbank	17	2.95E-02	0
orthrege	3000	1.76E+01	-1	aug2d	11	7.56E-01	0	hs99exp	56	1.22E-01	0	bloweyc	44	1.19E+01	0
hs111	3000	1.46E+01	-1	gouldqp3	28	2.58E-01	0	dixchlnv	72	2.35E-01	0	hs112	17	2.47E-02	0
try-b	32	8.20E-02	0	sosqp2	39	7.01E+00	0	dual3	26	4.01E-01	0	hadamard	6	2.62E-02	0
ssnlbeam	3000	4.80E+00	-1	gridneth	19	7.08E-02	0	dtoc1nc	17	6.74E-01	0	degenlpb	58	1.30E-01	0
portfl6	8	1.22E-02	0	optctrl3	79	2.13E-01	0	hs048	1	4.10E-03	0	himmelbk	161	8.49E-01	1
hager2	10	2.76E-01	0	hs079	159	2.09E-01	0	ncvxqp2	959	5.09E+01	0	optcdeg3	62	5.95E-01	0
qpnstair	653	5.29E+00	1	brainpc2	190	2.74E+01	1	dual4	16	7.37E-02	0	aug2dcqp	32	1.77E+00	0
hs099	21	7.35E-02	0	orthregc	110	3.05E+03	0	steenbrg	159	2.77E+00	0	deconvc	166	3.45E-01	0
eigenc2	30	1.62E+00	0	brainpc6	193	2.50E+01	1	dtoc6	23	7.62E-01	0	bt1	48	4.98E-02	0
hs077	27	1.02E-01	0	brainpc7	110	7.10E+00	0	byrdsphr	63	1.68E-01	0	hs107	3000	7.96E+00	-1
blockqp3	354	1.14E+01	0	hues-mod	117	5.34E+03	1	hager3	10	3.48E-01	0	hager1	1	3.40E-02	0
lch	918	2.72E+01	0	orthrgds	158	4.65E+02	0	bt2	22	5.49E-02	0	hs35mod	15	1.90E-02	0
csfi2	60	1.63E-01	0	brainpc8	160	1.18E+01	1	gridneti	27	1.01E-01	0	gridnete	26	1.36E+00	0
corkscrw	3000	2.53E+02	-1	brainpc0	36	2.51E+03	1	eigenb2	4	1.28E-02	0	blockqp5	50	4.32E-01	0
ubh5	18	2.23E+00	0	brainpc3	260	1.56E+01	1	dtoc4	3	1.60E-01	0	gausselm	1343	2.94E+01	0
ncvxqp6	780	1.87E+01	0	cvxqp3	58	5.51E+02	0	dtoc3	16	8.02E-01	0	eg3	19	4.97E-02	0
lotschd	39	9.09E-02	0	brainpc5	576	2.12E+03	1	gridnetd	40	1.68E+00	0	aug2dc	17	7.92E-01	0
dualc2	19	5.28E-02	0	cvxqp2	49	4.39E+01	0	hs119	43	1.15E-01	0	ncvxqp3	1011	5.67E+01	0
minperm	33	8.25E+00	0	huestis	195	7.46E+03	2	portfl4	9	1.25E-02	0	yao	106	1.35E+00	0
aug3dcqp	24	3.78E-01	0	brainpc1	3000	1.73E+02	-1	cvxqp1	161	2.56E+00	0	hs026	33	5.12E-02	0
bt8	17	5.94E-02	0	brainpc4	312	2.64E+01	1	bt3	17	4.36E-02	0	bloweyb	7	1.70E+00	0
qpcboei2	85	2.42E-01	0	orthrgdm	19	2.29E+00	0	steenbra	23	1.62E-01	0	static3	73	2.37E-01	4
kissing	495	9.40E+00	1	orthregd	18	2.02E+00	0	optctrl6	79	1.76E-01	0	qpnboei2	3000	1.91E+01	-1
alsotame	9	1.62E-02	0	brainpc9	122	7.95E+00	0	reading2	15	8.69E-01	0	orthrega	142	9.60E-01	0

Table 26: CUTEr set results, penalty mode $\frac{1}{\rho_L}$(linear), part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	15	3.84E-02	0	dtoc2	86	2.52E+00	0
steenbrd	121	1.67E+00	0	orthrds2	270	1.08E+00	1
bloweya	169	2.93E+01	0	eigencco	30	5.36E-02	0
dualc1	24	6.31E-02	0	lakes	586	2.19E+00	0
dualc8	19	5.70E-02	0	ncvxqp5	386	8.34E+00	0
portfl1	9	7.00E-03	0	dual1	18	8.72E-02	0
hs074	20	4.42E-02	0	odfits	17	2.41E-02	0
trainf	53	2.27E+00	0	bt6	23	6.17E-02	0
csfi1	34	8.48E-02	0	clnlbeam	187	1.07E+00	0
dual2	15	8.99E-02	0	bt7	55	1.31E-01	0
gridnetg	38	1.52E-01	0	rk23	9	1.13E-02	0
hong	24	2.93E-02	0	hs054	9	1.43E-02	0
dtoc1nb	6	2.56E-01	0	gridnetb	24	1.33E+00	0
hs014	10	1.59E-02	0	ubh1	16	1.33E+00	0
sreadin3	12	6.05E-01	0	dittert	91	8.61E-01	0
optcntrl	57	1.95E-01	0	blockqp1	35	5.92E-01	0
steenbre	1636	3.92E+01	1	ncvxqp9	3000	1.50E+02	-1
dnieper	44	7.44E-02	0	hs063	17	2.78E-02	0
steenbrc	76	4.04E-01	1	hs009	0	1.64E-03	0
gridneta	38	1.97E+00	0	hs067	11	1.98E-02	0
prodpl1	15	2.02E-02	0	ssebnln	779	3.86E+00	0
aug3dqp	23	2.68E-01	0	hs075	26	6.07E-02	0
hs049	19	2.72E-02	0				
aug2dqp	33	1.81E+00	0				
fletcher	173	5.96E-01	0				

Table 27: CUTEr set results, penalty mode $\frac{1}{\rho_L}$(linear), part 3.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
gilbert	85	1.89E+01	0	swopf	32	1.20E-01	0	smmpsf	674	2.04E+00	0	hs042	10	1.14E-02	0
bt11	68	9.66E-02	0	dallasm	36	7.92E-02	1	hs100lnp	184	3.70E-01	0	optmass	23	4.08E-02	0
hs046	42	1.10E-01	0	himmelbj	3000	1.62E+01	-1	gridnetf	54	4.22E+00	0	sseblin	55	2.17E-01	0
hs007	35	8.68E-02	0	aug3d	12	7.21E-01	0	lsnnodoc	1	4.90E-03	0	hs028	1	4.15E-03	0
hs114	80	1.98E-01	0	bt5	53	7.86E-02	0	bt12	37	5.65E-02	0	batch	33	4.88E-02	0
extrasim	11	1.83E-02	0	allinitc	36	9.05E-02	2	loadbal	15	3.19E-02	0	hs050	9	1.45E-02	0
model	35	1.87E-01	0	bt9	94	2.93E-01	0	hs081	62	1.97E-01	0	hs087	76	1.14E-01	0
disc2	106	3.64E-01	0	qpnboei1	464	3.43E+00	2	dtoc1na	6	3.07E-01	0	hs080	22	5.73E-02	0
eigenbco	495	2.56E+00	0	trimloss	153	8.20E-01	0	genhs28	1	4.56E-03	0	hs062	7	1.33E-02	0
catenary	225	1.32E+00	2	dtoc1nd	22	6.00E-01	0	hs071	36	1.21E-01	0	trainh	536	4.52E+01	0
tame	9	9.47E-03	0	minc44	22	2.66E-01	0	hs078	24	3.79E-02	0	sawpath	3000	2.64E+02	-1
dtoc1l	6	3.44E-01	0	hs073	3000	4.64E+00	-1	ncvxqp1	219	6.75E+00	2	steenbrb	49	5.39E-01	0
fccu	12	4.07E-02	0	hs039	94	2.99E-01	0	avion2	93	1.63E-01	0	gouldqp2	15	5.53E-02	0
hager4	19	7.26E-01	0	catena	149	2.50E-01	0	gridnetc	43	2.21E+00	0	zigzag	31	5.89E-02	0
hs056	15	1.88E-02	0	hs053	13	3.77E-02	0	hs041	36	1.09E-01	0	maratos	61	1.67E-01	0
robot	108	3.15E-01	0	ncvxqp4	261	5.77E+00	0	hs109	100	3.03E-01	0	aljazzaf	29	4.10E-02	0
bt13	31	9.12E-02	0	ncvxqp7	173	8.71E+00	2	orthregb	41	1.13E-01	0	reading3	29	8.51E-02	0
optcdeg2	40	3.32E-01	0	hs006	2045	2.71E+00	1	prodpl0	17	2.21E-02	0	orthrdm2	14	4.19E+00	0
hs027	87	1.20E-01	0	qpcstair	198	1.27E+00	2	blockqp2	13	1.12E-01	0	eigenaco	5	3.17E-02	0
portfl2	8	1.23E-02	0	reading1	92	8.53E+00	0	hvycrash	324	1.33E+00	1	dtoc5	13	4.39E-01	0
blockqp4	11	9.05E-02	0	spanhyd	20	4.09E-02	0	linspanh	15	3.12E-02	0	hs047	38	1.16E-01	0
qpcboei1	3000	1.95E+01	-1	aug3dc	11	1.24E-01	0	eigena2	3	1.32E-02	0	steenbrf	442	2.72E+00	1
hs111lnp	108	2.56E-01	2	portfl3	10	1.24E-02	0	hs061	32	9.06E-02	0	degenlpa	111	3.70E-01	2
hs032	16	1.35E-02	0	hs040	7	1.50E-02	0	hs051	1	4.23E-03	0	launch	571	3.49E+00	2
mwright	135	2.53E-01	0	sosqp1	33	3.34E+00	1	ncvxqp8	255	9.14E+00	2	hs060	70	1.48E-01	0

Table 28: CUTEr set results, penalty mode $\frac{1}{\rho_0}$, part 1.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
dualc5	14	3.89E-02	0	dixchlng	24	3.52E-02	0	bt4	106	3.04E-01	0	smbank	17	2.40E-02	0
orthrege	160	6.72E-01	0	aug2d	11	7.50E-01	0	hs99exp	113	2.93E-01	-2	bloweyc	44	1.13E+01	0
hs111	3000	1.55E+01	-1	gouldqp3	29	2.81E-01	0	dixchlnv	72	2.44E-01	0	hs112	17	2.18E-02	0
try-b	33	7.74E-02	0	sosqp2	43	1.19E+01	0	dual3	32	6.33E-01	0	hadamard	6	2.36E-02	0
ssnlbeam	3000	4.85E+00	-1	gridneth	21	6.24E-02	0	dtoc1nc	17	6.82E-01	0	degenlpb	72	1.98E-01	2
portfl6	8	1.23E-02	0	optctrl3	79	1.99E-01	0	hs048	1	4.32E-03	0	himmelbk	172	6.81E-01	0
hager2	10	2.87E-01	0	hs079	159	2.11E-01	0	ncvxqp2	427	1.35E+01	2	optcdeg3	45	3.38E-01	1
qpnstair	232	1.49E+00	2	brainpc2	182	2.34E+01	1	dual4	16	6.79E-02	0	aug2dcqp	32	1.74E+00	0
hs099	20	5.48E-02	0	orthregc	110	3.51E+03	0	steenbrg	159	2.71E+00	0	deconvc	166	3.51E-01	0
eigenc2	58	3.64E+00	0	brainpc6	185	1.17E+01	1	dtoc6	56	1.82E+00	2	bt1	48	6.51E-02	0
hs077	34	1.26E-01	0	brainpc7	103	6.21E+00	1	byrdsphr	108	2.94E-01	0	hs107	29	8.63E-02	0
blockqp3	340	9.72E+00	0	hues-mod	104	4.61E+03	2	hager3	10	3.33E-01	0	hager1	1	3.26E-02	0
lch	685	2.14E+01	0	orthrgds	158	4.08E+02	0	bt2	22	5.57E-02	0	hs35mod	15	2.01E-02	0
csfi2	49	1.37E-01	0	brainpc8	171	1.09E+01	1	gridneti	27	1.19E-01	0	gridnete	33	1.56E+00	0
corkscrw	2092	1.22E+02	0	brainpc0	885	2.52E+03	1	eigenb2	4	2.04E-02	0	blockqp5	50	4.34E-01	0
ubh5	16	1.82E+00	0	brainpc3	253	1.67E+01	1	dtoc4	3	1.49E-01	0	gausselm	1343	2.91E+01	0
ncvxqp6	419	9.13E+00	0	cvxqp3	58	6.34E+02	0	dtoc3	12	6.19E-01	0	eg3	19	5.28E-02	0
lotschd	32	6.83E-02	0	brainpc5	533	4.68E+03	-4	gridnetd	29	1.03E+00	0	aug2dc	17	7.67E-01	0
dualc2	19	4.16E-02	0	cvxqp2	45	5.70E+01	0	hs119	48	1.20E-01	0	ncvxqp3	2228	6.57E+01	2
minperm	41	9.95E+00	1	huestis	86	3.85E+03	2	portfl4	9	1.55E-02	0	yao	73	9.33E-01	2
aug3dcqp	24	3.67E-01	0	brainpc1	3000	1.70E+02	-1	cvxqp1	59	1.19E+00	2	hs026	33	4.95E-02	0
bt8	17	4.96E-02	0	brainpc4	126	6.10E+00	1	bt3	16	4.00E-02	0	bloweyb	7	1.68E+00	0
qpcboei2	85	2.15E-01	0	orthrgdm	19	2.11E+00	0	steenbra	23	1.65E-01	0	static3	73	2.28E-01	4
kissing	281	4.40E+00	0	orthregd	18	1.90E+00	0	optctrl6	79	1.97E-01	0	qpnboei2	271	1.17E+00	2
alsotame	9	1.68E-02	0	brainpc9	111	5.27E+00	1	reading2	15	8.55E-01	0	orthrega	85	5.58E-01	0

Table 29: CUTEr set results, penalty mode $\frac{1}{\rho_0}$, part 2.

Problem	iter	CPUs	STAT	Problem	iter	CPUs	STAT
hs052	13	3.15E-02	0	dtoc2	86	2.48E+00	0
steenbrd	121	1.65E+00	0	orthrds2	3000	2.27E+01	-1
bloweya	169	2.87E+01	0	eigencco	30	5.19E-02	0
dualc1	24	6.38E-02	0	lakes	418	1.21E+00	0
dualc8	19	5.94E-02	0	ncvxqp5	386	8.15E+00	0
portfl1	9	1.06E-02	0	dual1	18	9.42E-02	0
hs074	17	4.37E-02	0	odfits	17	2.50E-02	0
trainf	53	2.18E+00	0	bt6	23	5.95E-02	0
csfi1	38	8.15E-02	0	clnlbeam	187	1.08E+00	0
dual2	15	9.19E-02	0	bt7	51	1.25E-01	0
gridnetg	25	7.82E-02	0	rk23	9	1.54E-02	0
hong	24	3.40E-02	0	hs054	9	1.42E-02	0
dtoc1nb	6	2.33E-01	0	gridnetb	29	1.55E+00	0
hs014	10	1.56E-02	0	ubh1	16	1.23E+00	0
sreadin3	12	5.93E-01	0	dittert	91	8.63E-01	0
optcntrl	58	1.78E-01	0	blockqp1	26	4.98E-01	0
steenbre	1636	3.89E+01	1	ncvxqp9	317	1.33E+01	2
dnieper	44	7.52E-02	0	hs063	17	2.78E-02	0
steenbrc	577	3.36E+00	0	hs009	0	2.57E-03	0
gridneta	37	1.56E+00	0	hs067	11	1.81E-02	0
prodpl1	15	2.48E-02	0	ssebnln	283	1.79E+00	0
aug3dqp	23	2.73E-01	0	hs075	25	6.46E-02	2
hs049	19	2.17E-02	0				
aug2dqp	33	1.79E+00	0				
fletcher	60	1.89E-01	0				

Table 30: CUTEr set results, penalty mode $\frac{1}{\rho_0}$, part 3.

```
if i > 1:
e.sens_dot_nmpc()
e.update_u(e.olnmpc)

e.print_r_mhe()  #: Print results MHE-sens
e.print_r_dyn()  #: Print results NMPC-sens
#: State-estimation MHE
e.preparation_phase_mhe(as_strategy=True)
stat = e.solve_dyn(e.lsmhe,      tag="lsmhe")

if stat != 0:
sys.exit()
e.sens_k_aug_mhe()  #: Sensitivity matrix
e.prior_phase()          #: Prior-phase and arrival cost

#: Control NMPC
e.preparation_phase_nmpc(as_strategy=True, make_prediction=False)
stat_nmpc = e.solve_dyn(e.olnmpc, tag="olnmpc")
if stat_nmpc != 0:
sys.exit()

e.sens_k_aug_nmpc()  #: Sensitivity matrix
e.print_r_nmpc()  #: Print to NMPC results file
e.cycleSamPlant(plant_step=True)  #: Plant cycle
e.plant_uinject(e.PlantSample, src_kind="dict", skip_homotopy=True)
e.noisy_plant_manager(sigma=0.0015, action="apply", update_level=True
```

.3 The BFB model

This section contains a brief introduction to the Bubbling Fluidized Bed (BFB) model for CO_2 capture. The model used in this work is derived from the work of Yu and Biegler [56], in which a one-dimensional BFB model is used to test model reduction techniques. The CO_2 capture is performed with an Amine impregnated solid sorbent, in which the the following reversible reactions take place:

$$H_2O_{(g)} \rightleftharpoons H_2O_{(phys)}$$
$$2R_2NH + CO_{2,(g)} \rightleftharpoons R_2NH_2^+ + R_2NCO_2^-$$
$$R_2NH + CO_{2,(g)} + H_2O_{(phys)} \rightleftharpoons R_2NH_2^+ + HCO_3^-.$$

The BFB model consist mainly of algebraic equations for kinetics, thermodynamics, hydrodynamic correlations, and a set of PDEs for the mass and energy balances. These PDEs contain differential terms in space and time. For example, the molar flux in the bubble, $N_{gb,i} = \delta c_{b,i}$ has the following PDE:

$$\frac{\partial N_{gb,i}}{\partial t} = -\left(v_g \frac{\partial c_{b,i}}{\partial x} + c_{b,i}\frac{\partial v_g}{\partial x}\right) - D_i \frac{\partial^2 c_{b,i}}{\partial x^2} - \delta K_{bc}\left(c_{b,i} - c_{c,i}\right) + K_{b,bulk}/A_x, \quad i \in \{\text{species}\} \tag{1}$$

where most notably, the dispersion term $\partial^2 c_{b,i}/\partial x^2$ was added from the previous version. A similar modification was done to the energy balance , with the enthalpy flux term $H_{gb} = T_{gb} c_{pg} \delta \left(\sum_i c_{b,i}\right)$ and the following PDE:

$$\frac{\partial H_{gb}}{\partial t} = -c_{pg}\left(v_g \sum_i c_{b,i}\frac{\partial T_{gb}}{\partial x} + T_{gb}\left(v_{gb}\sum_i \frac{\partial c_{b,i}}{\partial x} + \sum_i c_{b,i}\frac{\partial v_g}{\partial x}\right)\right) - k_g \frac{\partial^2 T_{g,b}}{\partial x^2} - \delta H_{bc}\left(T_{g,b} - T_{g,c}\right) + H_{g,bulk}/A_x, \tag{2}$$

with the inclusion of the second partial derivative term and an expansion of the convective term. Further PDEs include mass and energy balances for the remaining phases at each region; i.e. gas-cloud, solid-cloud, gas-emulsion and solid-emulsion. Some of these equations and their differential terms are listed as follows:

Gas-cloud mass balance, $N_{gc,i} = c_{c,i} f_{cw} \delta e_d$:

$$\frac{\partial N_{gc,i}}{\partial t} = \delta K_{bc,i}(c_{b,i} - c_{c,i}) - \delta K_{ce,i}(c_{c,i} - c_{e,i}) - \delta f_{cw}(1 - e_d) r_{gc,i}, \quad i \in \{\text{species}\} \tag{3}$$

Gas-cloud enthalpy balance, $H_{gc} = T_{gc} f_{cw} e_d c_{pg} \delta \left(\sum_i c_{c,i}\right)$:

$$\begin{aligned}\frac{\partial H_{gc}}{\partial t} = {} & \delta H_{bc}(T_{gb} - T_{gc}) - \delta H_{ce}(T_{gc} - T_{ge}) \\ & - \delta f_{cw}(1 - e_d)\rho_s a_p h_p (T_{gc} - T_{sc}) - \delta f_{cw}(1 - e_d) \sum_j r_{gc,j} c_{p,g,c,j} (T_{gc} - T_{ref})\end{aligned} \tag{4}$$

Solid-cloud mass balance, $N_{sc,i} = n_{c,i} f_{cw} \delta \left(1 - e_d\right) \rho_s$:

$$\frac{\partial N_{sc,i}}{\partial t} = -\frac{\partial J_c n_{c,i}}{\partial x} - K_{s,bulk,i} - \delta \rho_s K_{ce,bs}(n_{c,i} - n_{e,i}) + f_{cw}\delta(1 - e_d) r_{sc,1}, \quad i \in \{\text{species}\} \tag{5}$$

Solid-cloud energy balance, $H_{sc} = T_{sc} f_{cw} \delta \left(1 - e_d\right) \rho_s c_{ps}$:

$$\begin{aligned}\frac{\partial H_{sc}}{\partial t} = {} & -\frac{\partial J_c h_{sc}}{\partial x} - H_{s,bulk} - \delta \rho_s K_{ce,bs}(h_{sc} - h_{se}) \\ & + f_{cw}\delta(1 - e_d) \sum_j \left(r_{gc,j} c_{p,g,c,j}\right) (T_{gc} - T_{ref}) + f_{cw}\delta(1 - e_d)\rho_s a_p h_p (T_{gc} - T_{sc})\end{aligned} \tag{6}$$

Gas-emulsion mass balance, $N_{ge,i} = c_{e,i} \left(1 - f_{cw}\delta - \delta\right) e_d$:

$$\frac{\partial N_{ge,i}}{\partial t} = \delta K_{ce,i}(c_{c,i} - c_{e,i}) - (1 - f_{cw}\delta - \delta)(1 - e_d) r_{ge,i} - K_{g,bulk,i}, \quad i \in \{\text{species}\} \tag{7}$$

Gas-emulsion energy balance, $H_{ge} = T_{ge} c_{pg} \left(1 - f_{cw}\delta - \delta\right) e_d \left(\sum_i c_{e,i}\right)$:

$$\begin{aligned}\frac{\partial H_{ge}}{\partial t} = {} & \delta H_{ce}(T_{gc} - T_{ge}) - (1 - f_{cw}\delta - \delta)(1 - e_d)\rho_s a_p h_p (T_{ge} - T_{se}) \\ & - H_{g,bulk} - (1 - f_{cw}\delta - \delta)(1 - e_d) \sum_j r_{ge,j} c_{p,g,e,j} (T_{ge} - T_{ref})\end{aligned} \tag{8}$$

Solid-emulsion mass balance, $N_{se,i} = n_{e,i} \left(1 - f_{cw}\delta - \delta\right) \left(1 - e_d\right) \rho_s$:

$$\frac{\partial N_{se,i}}{\partial t} = \frac{\partial J_e n_{e,i}}{\partial x} + K_{s,bulk,i} + \delta \rho_s K_{ce,bs}(n_{c,i} - n_{e,i}) + (1 - f_{cw}\delta - \delta)(1 - e_d) r_{se,1}, \quad i \in \{\text{species}\} \tag{9}$$

Solid-emulsion energy balance, $H_{se} = T_{se}\left(1 - f_{cw}\delta - \delta\right)\left(1 - e_d\right)\rho_s c_{cps}$:

$$\frac{\partial H_{se}}{\partial t} = \frac{\partial J_e h_{se}}{\partial x} + H_{s,bulk} + \delta\rho_s K_{ce,bs}(h_{sc} - h_{se})$$
$$+ (1 - f_{cw}\delta - \delta)(1 - e_d)\sum_j r_{ge,j} c_{p,g,e,j}(T_{ge} - T_{ref}) + (1 - f_{cw}\delta - \delta)(1 - e_d)\rho_s a_p h_p (T_{ge} - T_{se})$$
$$+ \pi d_x h_t \Delta T_{hx} N_x C_r. \quad (10)$$

At the reactor inlet ($x = 0$), the boundary conditions are taken from the gas introduced into the bed; and at the outlet ($x = L$) the first derivatives related to dispersion terms in the bubble are equal to zero:

$$c_{b,i}\Big|_{x=0} = z_{g,in,i}\frac{P_{in}}{R_0\left(T_{g,in} + 273.15\right)}, \quad i \in \{\text{species}\} \quad (11)$$
$$v_g\Big|_{x=0} = \frac{F_{g,in}}{3600A_x}\frac{P_{in}}{R_0\left(T_{g,in} + 273.15\right)} \quad (12)$$
$$T_{gb}\Big|_{x=0} = T_{g,in} \quad (13)$$
$$P\Big|_{x=0} = P_{in} \quad (14)$$
$$\frac{\partial T_{gb}}{\partial x}\Big|_{x=L} = 0 \quad (15)$$
$$\frac{\partial c_{b,i}}{\partial x}\Big|_{x=L} = 0, \quad i \in \{\text{species}\} \quad (16)$$
$$\frac{\partial v_b}{\partial x}\Big|_{x=L} = 0. \quad (17)$$

Moreover, the mass and energy balances at the bottom of the bed ($x = 0$) generate another set of boundary conditions for the sorbent loading and enthalpy.

$$J_e n_{e,j}\Big|_{x=0} = J_c n_{c,j}\Big|_{x=0}, \quad j \in \{\text{species}\} \quad (18)$$
$$J_e h_{se,j}\Big|_{x=0} = J_c h_{sc,j}\Big|_{x=0}, \quad j \in \{\text{species}\}. \quad (19)$$

Furthermore, with the over-flow configuration of the bed, the mass and energy balances at the top ($x = L$) dictate another set of boundary conditions; these involve the fluxes of

solid J_c and J_e, and the inlet conditions to the bed $F_{s,in}$, $z_{s,in,j}$ and $h_{s,in}$:

$$J_e n_{e,j}\Big|_{x=0} = J_c n_{c,j}\Big|_{x=0}, \quad j \in \{\text{species}\} \tag{20}$$

$$J_e h_{se,j}\Big|_{x=0} = J_c h_{sc,j}\Big|_{x=0}, \quad j \in \{\text{species}\} \tag{21}$$

$$J_e\Big|_{x=L} n_{e,j}\Big|_{x=L} A_x + F_{s,out} z_{s,out,j} = J_c\Big|_{x=L} n_{c,j}\Big|_{x=L} A_x + F_{s,in} z_{s,in,j}, \quad j \in \{\text{species}\} \tag{22}$$

$$J_e\Big|_{x=L} h_{se,j}\Big|_{x=L} A_x + F_{s,out} h_{s,out} = J_c\Big|_{x=L} h_{sc,j}\Big|_{x=L} A_x + F_{s,in} h_{s,in}, \quad j \in \{\text{species}\} \tag{23}$$

The remaining algebraic equations account for the following aspects:

- Reaction Kinetics
- Hydrodynamic Empirical Correlations
- Gas properties
- Heat exchanger correlations

It should be noted that most of these equations must be written for each discretization point in time and space. Thus the final size of the algebraic model passed to the solver has a considerable size.

.3.1 Reaction Kinetics

The following equations describe the reaction rates found in the physisorption reactions:

$$r_{1c} = k_{1c} * (P * y_{c,c} * 10^5 - \frac{n_{c,h}}{K_{e1c}}) \tag{24}$$

$$r_{2c} = k_{2c} * ((1 - \frac{2n_{c,n} + n_{c,c}}{nv}) * n_{c,h} * P * y_{c,c} * 10^5 - (\frac{1}{K_{e2c}} \frac{n_{c,n} + n_{c,c}}{nv}) * n_{c,c}) \tag{25}$$

$$r_{3c} = k_{3c} * ((1 - \frac{2n_{c,n} + n_{c,c}}{nv})^2 * (P * y_{c,c} * 10^5)^{m_1} - \frac{1}{K_{e3c}} \frac{n_{c,n} + n_{c,c}}{nv} * \frac{n_{c,n}}{nv}) \tag{26}$$

$$r_{1e} = k_{1e} * (P * y_{e,h} * 10^5 - \frac{n_{e,h}}{K_{e1e}}); \tag{27}$$

$$r_{2e} = k_{2e} * (\frac{1 - 2n_{e,n} - n_{e,c}}{nv} * n_{e,h} * (P * y_{e,c} * 10^5) - \frac{1}{K_{e2e}} n_{e,c} \frac{n_{e,n} + n_{e,c}}{nv}) \tag{28}$$

$$r_{3e} = k_{3e} * ((1 - \frac{2n_{e,n} + n_{e,c}}{nv})^2 * (P * y_{e,c} * 10^5)^{m_1} - \frac{1}{K_{e3e}} \frac{n_{e,n}}{nv} \frac{n_{e,c} + n_{e,n}}{nv}) \tag{29}$$

Then, the overall reaction rates are:

$$r_{gc,c} = \frac{nv * r_{3c} + r_{2c}}{1000} \tag{30}$$

$$r_{ge,c} = \frac{nv * r_{3e} + r_{2e}}{1000} \tag{31}$$

$$r_{sc,c} = r_{2c} \tag{32}$$

$$r_{se,c} = r_{2e} \tag{33}$$

$$r_{gc,h} = r_{1c}/1000 \tag{34}$$

$$r_{ge,h} = r_{1e}/1000 \tag{35}$$

$$r_{sc,h} = r_{1c} - r_{2c} \tag{36}$$

$$r_{se,h} = r_{1e} - r_{2e} \tag{37}$$

$$r_{gc,n} = 0 \tag{38}$$

$$r_{ge,n} = 0 \tag{39}$$

$$r_{sc,n} = nv * r_{3c} \tag{40}$$

$$r_{se,n} = nv * r_{3e} \tag{41}$$

Then the corresponding equilibrium constants as functions of the temperature and pressure are:

$$K_{e1c} * P * 10^5 = \exp\left(\frac{-dH_1}{R_0(T_{sc} + 273.15)} + \frac{dS_1}{R_0}\right) \tag{42}$$

$$K_{e2c} * P * 10^5 = \exp\left(\frac{-dH_2}{R_0(T_{sc} + 273.15)} + \frac{dS_2}{R_0}\right) \tag{43}$$

$$K_{e3c} * P * 10^5 = \exp\left(\frac{-dH_3}{R_0(T_{sc} + 273.15)} + \frac{dS_3}{R_0}\right) \tag{44}$$

$$K_{e1e} * P * 10^5 = \exp\left(\frac{-dH_1}{R_0(T_{se} + 273.15)} + \frac{dS_1}{R_0}\right) \tag{45}$$

$$K_{e2e} * P * 10^5 = \exp\left(\frac{-dH_2}{R_0(T_{se} + 273.15)} + \frac{dS_2}{R_0}\right) \tag{46}$$

$$K_{e3e} * P * 10^5 = \exp\left(\frac{-dH_3}{R_0(T_{se} + 273.15)} + \frac{dS_3}{R_0}\right) \tag{47}$$

And the kinetic constants:

$$k_{1c} = A_1 * (T_{sc} + 273.15) * \exp\left(\frac{-E_1}{R_0(T_{sc} + 273.15)}\right) \tag{48}$$

$$k_{2c} = A_2 * (T_{sc} + 273.15) * \exp\left(\frac{-E_2}{R_0(T_{sc} + 273.15)}\right) \tag{49}$$

$$k_{3c} = A_3 * (T_{sc} + 273.15) * \exp\left(\frac{-E_3}{R_0(T_{sc} + 273.15)}\right) \tag{50}$$

$$k_{1e} = A_1 * (T_{se} + 273.15) * \exp\left(\frac{-E_1}{R_0(T_{se} + 273.15)}\right) \tag{51}$$

$$k_{2e} = A_2 * (T_{se} + 273.15) * \exp\left(\frac{-E_2}{R_0(T_{se} + 273.15)}\right) \tag{52}$$

$$k_{3e} = A_3 * (T_{se} + 273.15) * \exp\left(\frac{-E_3}{R_0(T_{se} + 273.15)}\right) \tag{53}$$

.3.2 Hydrodynamic Empirical Correlations

The following equations encapsulate the behavior that governs the fluidized bed. These equations comprise a more elaborated version of the work of Kunii and Levenspiel [52]:

$$(1 - e) = (1 - e_d) * (1 - \delta) \tag{54}$$

Fluidization velocity

$$\frac{10 * 1.75 * (d_p v_{mf} \rho_{g,0})^2}{\phi_s e_{mf}^3 \mu_{g,0}^2} + \frac{10 * 150 * (1 - e_{mf}) d_p v_{mf} \rho_{g,0}}{\phi_s^2 * e_{mf}^3 \mu_{g,0}} = \frac{10 * d_p^3 \rho_{g,0} (\rho_s - \rho_{g,0}) g_c}{\mu_{g,0}^2} \tag{55}$$

Reynolds, Nusselt and Archimides numbers:

$$\text{Ar} = \frac{d_p^3 * \rho_g * (\rho_s - \rho_g) * gc}{\mu_g^2} \tag{56}$$

$$\text{Nu}_p = 0.03 \text{Re}_d^{1.3} \tag{57}$$

$$\text{Re}_d = \frac{v_e d_p \rho_g}{\mu_g} \tag{58}$$

Size and velocity of bubbles:

$$v_{br} = 0.711 * \sqrt{gc * d_b} \tag{59}$$

$$d_{b0} = 1.38 * gc^{-0.2} * ((v_g(0) - v_e(0)) * A_0)^{0.4} \tag{60}$$

$$d_{be} = \frac{D_t}{4} * (-g_1 + g_3)^2 \tag{61}$$

$$d_{bm} = 2.59 * (gc^{-0.2}) * ((v_g - v_e) * A_x)^{0.4} \tag{62}$$

$$g_1 = 2.56 \times 10^{-2} * \frac{\sqrt{\frac{D_t}{gc}}}{v_{mf}} \tag{63}$$

$$g_2 = \frac{D_t * (g_1 + g_3)^2}{4} \tag{64}$$

$$g_3 = \sqrt{g_1^2 + \frac{4 * d_{bm}}{D_t}} \tag{65}$$

$$(\frac{\sqrt{d_{bu}} - \sqrt{d_{be}}}{\sqrt{d_{b0}} - \sqrt{d_{be}}})^{1-\frac{g_1}{g_3}} * (\frac{\sqrt{d_{bu}} - \sqrt{g_2}}{\sqrt{d_{b0}} - \sqrt{g_2}})^{1+\frac{g_1}{g_3}} = \exp\left(-0.3\frac{x}{D_t}\right) \tag{66}$$

$$d_b = d_{bu} \tag{67}$$

$$f_c = \frac{3\frac{v_{mf}}{e_{mf}}}{v_{br} - \frac{v_{mf}}{e_{mf}}} \tag{68}$$

$$f_{cw} = f_c + f_w \tag{69}$$

$$v_b = 1.55 * ((v_g - v_{mf}) + 14.1 * (d_b + 0.005)) * (D_{te}^{0.32}) + v_{br} \tag{70}$$

$$(1 - e_{mf}) d_p^{0.1} gc^{0.118} (\rho_s - \rho_g)^{0.118} x^{0.043} = 2.54 \rho_g^{0.016} \mu_g^{0.066} e^{0.090*F} (1 - e_d) \tag{71}$$

$$v_e * d_p^{0.568} * gc^{0.663} * (\rho_s - \rho_g)^{0.663} * x^{0.244} = v_{mf} * 188 * \rho_g^{0.089} * \mu_g^{0.371} * \exp(0.508 * F) \tag{72}$$

There is also the upgraded empirical correlations that compute the mass and heat transfer coefficients for the different regions:

$$K_{bc,j} = 1.32 * 4.5 * \frac{v_{mf}}{d_b} + 5.85 * \frac{(D_j \times 10^{-4})^{0.5} gc^{0.25}}{d_b^{5/4}}, \quad j \in \{\text{species}\} \tag{73}$$

$$K_{ce,j} = 6.77 * \sqrt{\frac{e_d * (D_j \times 10^{-4}) * v_{br}}{d_b^3}}, \quad j \in \{\text{species}\} \tag{74}$$

$$H_{bc} = 1.32 * 4.5\frac{v_{mf}c_{bt}c_{pg}}{db} + 5.85\frac{\sqrt{k_g/1000 * c_{bt} * c_{pg}} * gc^{0.25}}{d_b^{5/4}} \tag{75}$$

$$H_{ce} = 6.78 * \sqrt{\frac{e_d v_b k_g * c_{ct} * c_{pg}}{1000 d_b^3}} \tag{76}$$

$$K_{ce,bs} = 3 * \frac{(1-e_d)v_e}{(1-\delta)e_d d_b} \tag{77}$$

$$\mathrm{Nu}_p = \frac{1000 * h_p * d_p}{k_g} \tag{78}$$

$$K_{g,bulk,k} = K_d \left(\sum_i c_{e,i} - \sum_i c_{b,i} \right) y_{b,k}, \quad k \in \{\text{species}\} \tag{79}$$

$$H_{g,bulk} = K_d \left(\sum_i c_{e,i} - \sum_i c_{b,i} \right) c_{pg} T_{gb} \tag{80}$$

$$K_{s,bulk,k} = -A_x \frac{\partial J_c}{\partial x} n_{e,k}, \quad k \in \{\text{species}\} \tag{81}$$

$$H_{s,bulk,k} = -A_x \frac{\partial J_c}{\partial x} h_{se} \tag{82}$$

.3.3 Gas Phase Properties

The original model of [52] used Aspen properties for the computation of transport and thermodynamic properties. Then, in the work of [56] these were either approximated with regressions or simplifications. For example, the gas properties followed the ideal gas law, and the gas viscosity and thermal conductivity were assumed to be constant. In this work we follow the same methodology as the latter. Most notably, the gas diffusion coefficient

are given by the following relationships:

$$D_c = [\ 0.1593 - 0.1282\,(P-1.4) + 0.001\,(T_{ge}-60) + 0.0964\,(P-1.4)^2 \tag{83}$$

$$- 0.0006921\,(P-1.4) * (T_{ge}-60) - 3.3532 \times 10^{-6}\,(T_{ge}-60)^2\ ]\,\frac{y_{e,h}}{y_{e,h}+y_{e,n}} + \tag{84}$$

$$[\ 0.1495 - 0.1204 * (P-1.4) + 0.0008896 * (T_{ge}-60) + 0.0906(P-1.4)^2 \tag{85}$$

$$- 0.0005857(P-1.4)(T_{ge}-60) - 3.559 \times 10^{-6} * (T_{ge}-60)^2\ ]\,\frac{y_{e,n}}{y_{e,h}+y_{e,n}} \tag{86}$$

$$D_h = (0.1593 - 0.1282(P-1.4) + 0.001(T_{ge}-60) + 0.0964(P-1.4)^2 \tag{87}$$

$$- 0.0006921(P-1.4)(T_{ge}-60) - 3.3532 \times 10^{-6}(T_{ge}-60)^2)\frac{y_{e,c}}{y_{e,c}+y_{e,n}} + \tag{88}$$

$$(0.2165 - 0.1743 * (P-1.4) + 0.001377 * (T_{ge}-60) + 0.13109(P-1.4)^2 \tag{89}$$

$$- 0.0009115 * (P-1.4) * (T_{ge}-60) - 4.8394 \times 10^{-6}(T_{ge}-60)^2) * \frac{y_{e,n}}{y_{e,c}+y_{e,n}} \tag{90}$$

$$D_n = (0.1495 - 0.1204(P-1.4) + 0.0008896(T_{ge}-60) + 0.0906(P-1.4)^2 \tag{91}$$

$$- 0.0005857(P-1.4)(T_{ge}-60) - 3.559 \times 10^{-6} * (T_{ge}-60)^2)\frac{y_{e,c}}{y_{e,h}+y_{e,h}} + \tag{92}$$

$$(0.2165 - 0.1743(P-1.4) + 0.001377(T_{ge}-60) + 0.13109(P-1.4)^2 \tag{93}$$

$$- 0.0009115(P-1.4)(T_{ge}-60) - 4.8394 \times 10^{-6} * (T_{ge}-60)^2) * \frac{y_{e,h}}{y_{e,h}+y_{e,c}} \tag{94}$$

Considering constant gas heat capacities; the enthalpies are:

$$h_{sc} = 10^{-3}\,[(n_{c,h}+n_{c,c})\,(c_{p,g,c,h}T_{sc} + dH_1) + n_{c,c}\,(c_{p,g,c,c}T_{sc} + dH_2) + n_{c,n}\,(c_{p,g,c,c}T_{sc} + dH_3)] + c_{ps}T_{sc} \tag{95}$$

$$h_{se} = 10^{-3}\,[(n_{e,h}+n_{e,c})\,(c_{p,g,e,h}T_{se} + dH_1) + n_{e,c}\,(c_{p,g,c,c}T_{sc} + dH_2) + n_{e,n}\,(c_{p,g,c,c}T_{sc} + dH_3)] + c_{ps}T_{se} \tag{96}$$

And gas molar flow:

$$G_b = v_g * A_x * \sum_i c_{b,i} \tag{97}$$

.3.4 Heat exchanger correlations

The terms in the energy balance related to the heat exchanger are computed using empirical correlations for the heat transfer coefficients. Once again these depend on the hydrodynamic properties of the bubble and the solid.

$$k_{pa} = (3.58 - 2.5 * e_d) * k_g \frac{k_p}{k_g}^{0.46-0.46*e_d} \tag{98}$$

$$\tau = 0.44 * \frac{d_p g_c}{v_{mf}^2 (f_n - a_h)^2}^{0.14} \frac{d_p}{d_x}^{0.225} \tag{99}$$

$$f_n = \frac{v_g}{v_{mf}} \tag{100}$$

$$f_b = 0.33 * (\frac{v_{mf}^2 * (f_n - a_h)^2}{d_p * gc})^{0.14} \tag{101}$$

$$h_d = 2\sqrt{\frac{k_{pa}\rho_s c_{ps}(1 - e_d)}{1000\Pi\tau}} \tag{102}$$

$$Pr = 1000\frac{c_{pg,m}\mu_g}{k_g} \tag{103}$$

$$\mathrm{Nu}_h = 0.009 * \mathrm{Ar}^{0.5} * Pr^{0.33} \tag{104}$$

$$\mathrm{Nu}_h = \frac{1000 * h_l * d_p}{k_g} \tag{105}$$

$$h_t = f_b * h_d + (1 - f_b) * h_l \tag{106}$$

$$h_{hx,in} = -0.2831 - 2.9863 \times 10^{-6} * (P_{hx,in} - 1.3) + 7.3855 \times 10^{-5} * (T_{hx,in} - 60) \tag{107}$$

$$\Delta T_{hx} = T_{tube} - T_{se} \tag{108}$$

$$h_t * \Delta T_{hx}\mathrm{Cr} = h_w(T_{hx} - T_{tube}) \tag{109}$$

$$T_{hx} = 33.2104 + 14170.15(h_{xh} + 0.285) \tag{110}$$

$$\rho_{hx} = 959.5222 + 0.002877(P_{hx} - 1.3) - 1.0044496(T_{hx} - 60) \tag{111}$$

Finally, the pressure drop inside the heat-exchanger tubes is given by a differential equation in the axial direction:

$$\begin{aligned} \frac{\partial P_{hx}}{\partial x} &= \Delta P_{hx} + \rho_{hx} * 10^{-5} \\ P_{hx,N}\Big|_{x=0} &= P_{hx,in} \end{aligned} \tag{112}$$

.4 BFB Nomenclature

A_x: Total bed cross-sectional area (m^2)

a_p: Particle specific surface area (m^2/kg)

C_r: Average correction factor for heat exchanger tubes

$c_{b,t}$: Bubble region gas total concentration ($kmol/m^3$)

$c_{c,t}$: Cloud-Wake region gas total concentration ($kmol/m^3$)

$c_{e,t}$: Emulsion region gas total concentration ($kmol/m^3$)

$c_{b,j}$: Bubble region gas concentration for component j ($kmol/m^3$)

$c_{c,j}$: Cloud-Wake region gas concentration for component j ($kmol/m^3$)

$c_{e,j}$: Emulsion region gas concentration for component j ($kmol/m^3$)

c_{ps}: Particle heat capacity (kJ/(kg K))

c_{pg}: Bubble region gas molar specific heat capacity (kJ/(kmol K))

$c_{p,g,e,j}$: Gas phase pure component molar specific heat capacity in Emulsion region (kJ/(kmol K))

$c_{p,g,c,j}$: Gas phase pure component molar specific heat capacity in Cloud-Wake region (kJ/(kmol K))

D_j: Gas species diffusivity (m/s^2)

d_x: Heat exchanger tube diameter (m)

e_d: voidage of emulsion region

$F_{g,in}$: Input flue gas flow (kmol/s)

$F_{s,in}$: Input solid sorbent flow (mol/s)

$F_{s,in}$: Output solid sorbent flow (mol/s)

f_{cw}: Cloud-Wake to bubble region volume ratio

G_b: molar flowrate of gas in bubbles (kmol/s)

$H_{g,bulk}$: Gas phase bulk flow heat transfer (kJ/(s m))

$H_{s,bulk}$: Solid phase bulk flow heat transfer (kJ/(s m))

H_{bc}: Bubble to Cloud-Wake gas heat transfer coefficient (kJ/(m^3 K s))

H_{ce}: Cloud-Wake to Emulsion gas heat transfer coefficient (kJ/(m^3 K s))

H_{gb}: Gas-bubble enthalpy flux (kJ/(m^3))

H_{gc}: Gas-cloud enthalpy flux (kJ/(m^3))

H_{ge}: Gas-emulsion enthalpy flux (kJ/(m^3))

H_{sc}: Solid-bubble enthalpy flux (kJ/(m^3))

H_{se}: Solid-bubble enthalpy flux (kJ/(m^3))

h_{se}: Sorbent specific enthalpy in Emulsion region (kJ/kg)

h_{sc}: Sorbent specific enthalpy in Cloud-Wake region (kJ/kg)

h_t: Overall heat transfer coefficient (kJ/(m^2 K s))

h_p: Convective heat transfer coefficient (kJ/(m^2 K s))

$h_{s,in}$: Output solid sorbent specific enthalpy (kJ/kg)

$h_{s,out}$: Output solid sorbent specific enthalpy (kJ/kg)

J_c: Cloud-Wake region solids flux (kg/(m^2 s))

J_e: Emulsion region solids flux (kg/(m^2 s))

$K_{ce,bs}$: Cloud-Wake to Emulsion solids mass transfer coefficient (s^{-1})

$K_{ce,1}$: Cloud-Wake to Emulsion gas mass transfer coefficient (s^{-1})

$K_{bc,1}$: Bubble to Cloud-Wake gas mass transfer coefficient (s^{-1})

$K_{g,bulk,j}$: Gas phase bulk flow mass transfer for component j (kmol/(m s))

$K_{s,bulk,j}$: Solid phase bulk flow mass transfer for component j (kmol/(m s))

$K_{ce,bs}$: Cloud-Wake to Emulsion solids mass transfer coefficient (s^{-1})

L: Reactor length (m)

$n_{c,j}$: Cloud-Wake region adsorbed species concentrations for component j (mol/kg)

$n_{e,j}$: Emulsion region adsorbed species concentrations for component j (mol/kg)

$N_{gb,i}$: Gas-bubble molar flux (mol/m^3)

$N_{gc,i}$: Gas-cloud molar flux (mol/m^3)

$N_{ge,i}$: Gas-emulsion molar flux (mol/m^3)

$N_{sc,i}$: Solid-cloud molar flux (mol/m^3)

$N_{se,i}$: Solid-emulsion molar flux (mol/m^3)

N_x: Number of tubes in heat exchanger

P: Bed pressure (bar)

P_{in}: Bed inlet pressure (bar)

$r_{se,j}$: Emulsion region adsorbed species reaction rates (mol/(m^3 s))

$r_{sc,j}$: Cloud-Wake region adsorbed species reaction rates (mol/(m^3 s))

$r_{gc,j}$: Cloud-Wake region gas component reaction rates (kmol/(m^3 s))

$r_{ge,j}$: Emulsion region gas component reaction rates (kmol/(m^3 s))

T_{se}: Emulsion region solids temperature (K)

T_{sc}: Cloud-Wake region solids temperature (K)

T_{gb}: Bubble region gas temperature (K)

T_{ge}: Emulsion region gas temperature (K)

T_{gc}: Cloud-Wake region gas temperature (K)

$T_{g,in}$: Input flue gas temperature (K)

x: Axial position inside the bed (m)

$y_{c,j}$: Cloud-wake region gas mole fraction for component j

$y_{b,j}$: Bubble region gas mole fraction for component j

$y_{e,j}$: Emulsion region gas mole fraction for component j

$z_{g,in,j}$: Input flue gas mole fraction for component j

$z_{s,in,j}$: Input solid sorbent mole fraction for component j

$z_{s,out,j}$: Output solid sorbent mole fraction for component j

δ: Volume fraction of bubbles in bed

ρ_g: Gas density (kg/m^3)

ρ_s: Solid density (kg/m^3)

ΔT_{hx}: Heat exchanger temperature difference (K)