Advanced Optimization Strategies for Periodic Adsorption Processes

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Chemical Engineering

Sree Rama Raju Vetukuri
B.Tech., Chemical Engg., J.N.T.U Hyderabad
M.Tech., Chemical Engg., Indian Institute of Technology Madras
M.Ch.E., Chemical Engg., Carnegie Mellon University, Pittsburgh, PA

Carnegie Mellon University
Pittsburgh, PA

January, 2011

Acknowledgement

I feel very privileged to have worked with my advisor, Prof. Larry Biegler. I owe a great debt of gratitude for his guidance, support and patience during the last four years. I am always amazed at the depth and the breadth of his knowledge in a large variety of technical areas and it was a pleasure to build on his extraordinary know-how in the areas of dynamic simulation and optimization. I would also like to thank Prof. Andrea Walther for her valuable comments and suggestions in the development of this work.

I would also like to thank the other members of my thesis committee: Prof. Ignacio Grossmann, Prof. Nick Sahinidis and Prof. Shlomo Ta'asan for their valuable advices, suggestions and time. The financial support from National Energy Technological Laboratory (NETL) is gratefully acknowledged. I thank all the faculty and staff in the Department of Chemical Engineering at CMU for making the four years memorable. In addition, the presence of many visitors to the group during my time here have made it a stimulating place, especially Euclides and Prof. Sachin Patwardhan.

I want to thank all my fellow graduate students and my friends for making grad life enjoyable. Thanks to Ravi for being a nice room-mate (and for the delicious food); Anshul for helping me with PSA optimization studies and for his company during lunch; Parag, Shweta, Mohit and Ram for all the get-togethers and pot-lucks; Manav and Gaurav for all the fun we had in the first year; Sekhar for his regular phone conversations; Jim for

his constant help and for regular converations on optimization; Kexin for helping me with the interior point code; and for all the other friends who have helped me in some way or the other in this thesis.

This thesis would not have been possible without the unconditional love and support from my parents, my aunt, my in-laws, my sister and my brother-in-law. I will be forever indebted to each one of them for helping me become what I am today. Finally, thanks to my wife Deepika. I will always think of the period I spent at CMU as the time where I met the most special person in my life. Deepika's love and patience during this project have kept me going, especially during the most difficult times.

Abstract

Periodic Adsorption Processes (PAPs) have gained increasing commercial importance as an energy-efficient separation technique over the past two decades. Based on fluid-solid interactions, these systems never reach steady state. Instead they operate at cyclic steady state, where the bed conditions at the beginning of the cycle match with those at the end of the cycle. Nevertheless, optimization of these processes remains particularly challenging, because cyclic operation leads to dense Jacobians, whose computation dominates the overall cost of the optimization strategy. In order to efficiently handle these Jacobians during optimization and reduce the computation time, this work presents new composite step trust-region algorithms based on sequential quadratic programming and interior point methods for the solution of minimization problems with both nonlinear equality and inequality constraints. Instead of forming and factoring the dense constraint Jacobian, these algorithms approximate the Jacobian of equality constraints with a specialized quasi-Newton method. Hence, they are well suited to solve optimization problems related to PAPs. In addition to allowing inexactness of the Jacobian and its null-space representation, the algorithm also provides exact second order information in the form of Hessian-vector products to improve the convergence rate. The resulting approach also combines automatic differentiation and more sophisticated integration algorithms to evaluate the direct sensitivity and adjoint sensitivity equations. Numerical performance

results on small scale PAP problems and CUTEr problems show significant reduction in computation time.

Furthermore, we propose a systematic methodology to design PSA cycles using a superstructure based approach. The superstructure is rich enough to predict a number of different PSA operating steps, and their optimal sequence is obtained by solving an optimal control problem. PSA is a potential technology for pre-combustion CO_2 capture because of low operating costs and high performance. We utilize the superstructure approach to synthesize PSA cycles for this purpose which can separate both H_2 and CO_2 at high purity and operate with a low power consumption of 86 kWh/tonne of CO_2 captured.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Synopsis

The main motivation of this thesis arises due to the optimization problems related to Periodic Adsorption Processes (PAPs). PAPs have gained increasing commercial acceptance as an efficient separation technique for a wide range of applications. With increasing industrial applications, there is significant interest for an efficient modeling, simulation and optimization strategy for PAPs. However, despite a vast growth in the practical application of PAPs, the design and optimization of PAPs still largely remains computationally challenging. This chapter gives an overview of PAPs and highlights the computational challenges of the resulting optimization problems. The structure of the thesis is presented in the final section of this chapter.

1.1 Overview of PAPs

Periodic adsorption processes have recently been gaining increasing commercial acceptance as energy efficient alternatives to other gas separation techniques such as cryogenic distillation. Much progress has already been achieved in improving their performance with respect to both the process economics and the attainable recovery or purity of the products [56].

These processes consist of vessels or beds packed with a solid sorbent. The adsorbent is brought in contact with a multi-component feed mixture and the separation of the components is based on the difference in the affinity towards the adsorbent particles. The selectivity between the components to be separated in a PAP process is governed by differences in either adsorption equilibrium or adsorption rates. Typical PAP process involves two important steps [55] as shown in Figure 1.1: (1) *adsorption*, during which the components with higher affinity towards the adsorbent gets adsorbed and the less strongly adsorbed species passes through the adsorbed bed and is recovered as the *raffinate* product. This step is stopped before the bed gets totally saturated; (2) *desorption*, during which the strongly adsorbed species is removed as the *extract* product, thus regenerating the adsorbent for reuse in the next cycle. Typical means of regenerating the adsorbent (Figure 1.2) are:

1. *Pressure Swing Adsorption* (PSA) process: reducing the partial pressure of the adsorbate in the gas phase.

2. *Temperature Swing Adsorption* (TSA) process: increasing the temperature in the adsorbent.

3. *Composition Swing Adsorption* (CSA) process: alter the composition of the fluid

phase so as to control the direction of adsorption.

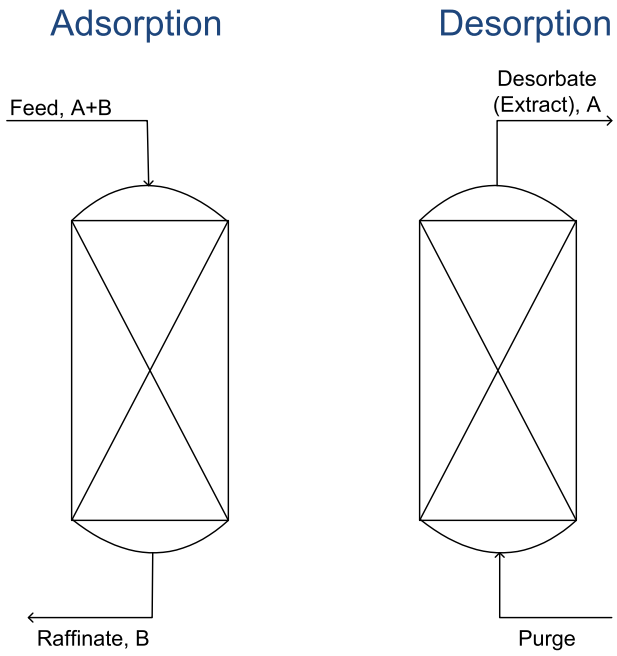

Figure 1.1: Adsorption and desorption steps

All these processes are collectively known as periodic adsorption processes. Figure 1.2 shows the useful capacity of the adsorbent which is the difference in loading between two points, corresponding to the adsorption and regeneration pressures. Besides the adsorption and desorption steps, industrial PAPs operate with a wide variety of steps and also with multiple beds in order to enhance the product purity or recovery and the process performance with respect to the operating costs. Industrial applications of PAPs include vacuum swing adsorption (VSA) to separate oxygen from air, pressure swing adsorption (PSA) to separate hydrogen from hydrocarbons and simulated moving bed chromatography (SMB) to separate enantiomers (e.g., glucose and fructose) in the liquid phase.

Unlike most processes, e.g. distillation which operates at steady state conditions, PAPs operate under periodic transient conditions with each bed repeatedly undergoing a

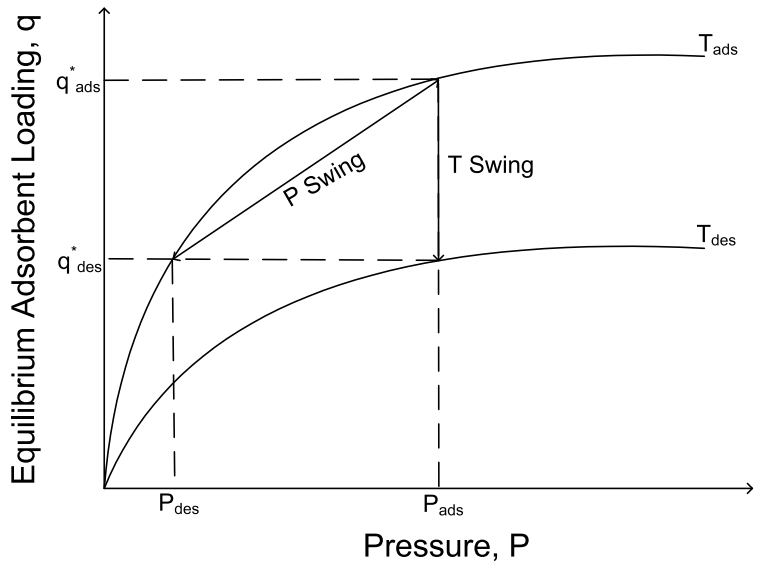

Figure 1.2: Change in equilibrium loading with pressure ($T_{ads} < T_{des}$)

sequence of steps. Because of the periodic operation, these processes reach a cyclic steady state (CSS), where the concentration profiles change dynamically and the profiles at the beginning of each cycle match with those at the end of the cycle. PAPs are practically at CSS for most of their useful operation. Consequently, it is the CSS that determines their technical and economic performance.

The theoretical modeling of PAP systems has been extensively studied to gain a clear understanding of this rather complex process. From the mathematical point of view, periodic adsorption processes can be classified as distributed parameter systems in which properties vary with respect to space dimensions as well as time. Such systems can be described by partial differential equations expressing the physical laws of conservation of mass, energy, and momentum. Additionally, these models may also involve algebraic relations such as equilibrium isotherms and other relationships between variables, such as equations of state and fluid property correlations. Thus, periodic adsorption processes are overall characterized by mixed systems of partial differential and algebraic equations

(PDAEs) in space and time with varying initial and boundary conditions that define the various steps in the process. An exhaustive summary of the bed models in literature is given in Ruthven [56], Nikolič et al. [49] and Kumar et al. [44]. These models differ mainly in the form of the mass transfer rate, the form of the equilibrium isotherm, thermal effects and pressure drop through the bed. These models offer a realistic representation of industrial processes as well as a good compromise between accurate and efficient solutions and can adequately account for all the factors that affect the process performance.

1.2 Computational Challenges

With increasing industrial applications, there is significant interest for an efficient modeling, simulation and optimization strategy for PAPs. Despite a vast growth in the practical application of PAPs, the design and optimization of PAPs still largely remains an experimental effort for several reasons [63]. Firstly, the computational effort required to solve the stiff hyperbolic PDE systems with solution profiles represented by steep adsorption fronts is usually quite expensive and hence time-consuming. Secondly, multi-adsorbent layers, non-isothermal effects and stringent product specifications result in high non-linearities and ill-conditioned matrices which lead to the failure of numerical solvers. Most importantly, the CSS operation yields dense constraint Jacobians, where the time required for the computation of the Jacobian and its factorization dominates the overall optimization process. As a result, the design and optimization of PAP systems presents a significant computational challenge to current optimization methods [9].

Several studies in literature have attempted to optimize the PAP systems. Most of them can be classified either as *Black Box* or *Equation-Oriented* approaches. In the *Black Box* approach, for every experiment the optimizer selects a set of decision variables.

Based on these variables, the bed models are executed until the periodic operation (CSS) is reached. At the end of the runs, the values of objective function and constraints are returned to the optimizer. For instance, Webley [77] uses response surface methodology for this task. The Black Box approach is robust and capable of dealing with very detailed and complicated models. However, since it requires CSS convergence for every single set of decision variables and hundreds of experiments are usually needed, it is extremely time consuming.

On the other hand, Nilchan and Pantelides [50] implement an equation-based approach using gPROMS [2]. The optimizer, CSS condition and bed equations are all put in the same framework and the convergence task is borne by the nonlinear equation solver within gPROMS. This approach is efficient for simple models. However, non-differentiable terms and steep fronts from more complex bed equations can often lead to failure of the Newton solver. This was also observed in [42].

In order to reduce the computational effort and improve the accuracy associated with the temporal integration, Jiang et al. [38] proposed a simultaneous tailored approach for PSA processes. This approach was shown to be more efficient, reliable and accurate for PSA optimization. In this approach, an SQP solver [10] searches for both the optimal operating parameters and concentration profiles simultaneously. Kawajiri et al. [41] and Toumi et al. [68] applied this approach to SMB processes. However, this efficient approach can still be time consuming for large scale systems because of the large set of sensitivity equations that have to be integrated to form the constraint Jacobian. Both these works report that more than 95% of the CPU time was spent by the integrator in forming and factoring the dense constraint Jacobians. Hence, the efficient handling of the constraint Jacobian is essential to evaluate a proposed design, to analyze a process for safety, for

controllability and operability, debottlenecking and retrofitting existing units, and for optimization of new design or existing installations, which is the main objective of this thesis. In this thesis, we focus on developing new cost-efficient and robust optimization methods that overcome the shortcomings of the above approaches.

1.3 Research Problem Statement

Although there have been many sophisticated optimization strategies that have been developed and applied for design and operation of PAP systems, these methods have limitations with respect to convergence, computation time and problem size. Even the most efficient and reliable simultaneous tailored approach [38] for the optimization of moderately sized PAP problems suffers from large computation times arising from the direct sensitivity calculations required to form the constraint Jacobian.

In order to overcome these limitations, this thesis focuses on developing faster algorithms that do not require explicit calculation of the Jacobian through sensitivity calculations. To avoid the bottleneck of sensitivity calculations adjoint approaches are considered and new robust optimization algorithms based on trust-region sequential quadratic programming (SQP) and interior point methods are developed, implemented and tested on PAP applications, which have been shown to be efficient and reliable with desirable convergence properties.

This dissertation also focuses on developing a systematic framework to develop novel PSA cycles for pre-combustion CO_2 capture. We present an optimization-based framework to generate optimal PSA cycles from a 2-bed PSA superstructure. Different PSA operating steps are realized by varying the time-dependent control variables that define the interconnections between the beds. An optimal sequence of operating steps is obtained

by solving an optimal control problem.

1.4 Thesis Outline

To address the challenges outlined in Section 1.2, this study develops novel optimization algorithms for the solution of optimization problems with dense constraint Jacobians. This thesis is divided into seven chapters:

Chapter 2 presents the general dynamic optimization problem formulation of PAPs. The solution strategies for solving the optimization problem are described. Finally, an overview of approaches for the solution of NLP problem is given.

Chapter 3 describes the cost-efficient and accurate sensitivity methods which have been implemented in this thesis to compute the exact Jacobian-vector and Hessian-vector products required by the NLP subproblem.

Chapter 4 proposes an inexact trust region SQP algorithm based on Byrd-Omojokun approach. The performance of the algorithm is tested on equality and bound constrained optimization problems for simulated moving bed and vacuum swing adsorption systems.

In Chapter 5, an inexact interior point trust-region algorithm is developed and analyzed for the solution of minimization problems with nonlinear inequality constraints. Numerical performance of the algorithm is evaluated on small numerical examples.

Chapter 6 introduces a novel two-bed PSA superstructure to determine optimal PSA configurations for pre-combustion CO_2 capture. Solution strategy to solve the PDAE-constrained problem is then discussed. The superstructure approach is illustrated for two different case studies of pre-combustion CO_2 capture.

Chapter 7 summarizes the research and explores future directions for algorithmic improvement and PSA superstructure.

Chapter 2

Background for PAP Optimization

Synopsis

In this chapter we present the general dynamic optimization problem formulation of PAP applications. Efficient solution procedures are required to solve these optimization problems which are often time-consuming and difficult to solve. Two commonly used methodologies, namely, the simultaneous and sequential methods are discussed and compared. Furthermore, an overview of approaches for the solution of nonlinear programming problem is given.

2.1 Optimization Problem

The general PDE constrained optimization problem of PAPs can be expressed as,

$$\min_{y,p} \quad f(z(\tilde{x}, t_f), p) \tag{2.1a}$$

$$s.t. \quad F(\frac{\partial z(\tilde{x}, t)}{\partial t}, \frac{\partial z(\tilde{x}, t)}{\partial \tilde{x}}, z(\tilde{x}, t), t, p) = 0 \quad t \in (t_0, t_f] \tag{2.1b}$$

$$z(\tilde{x}, t_0) = y(\tilde{x}) \tag{2.1c}$$

$$z(\tilde{x}_0, t) = y_B(t) \tag{2.1d}$$

$$w(z(\tilde{x}, t_f), p)) \leq 0 \tag{2.1e}$$

$$c_{ss}(y(\tilde{x}), p) \equiv y(\tilde{x}) - z(t_f, \tilde{x}, p) = 0 \tag{2.1f}$$

$$y_{min} \leq y(\tilde{x}),\ p^L \leq p \leq p^U \tag{2.1g}$$

Here, $F(.)$ is an implicit partial differential equation model with initial and boundary conditions given by (2.1c) and (2.1d) respectively; $z(t)$ are the state variables, that include concentration, loading, temperature etc.; y are the initial conditions for the differential state variables, $z(t)$. These initial conditions are decision variables in the optimization problem along with the design variables, $p \in \mathbb{R}^{n_p}$, that include flow rates, valve constants, bed dimensions, step times etc.; the vector w corresponds to the design constraints that can include purity or pressure specifications; c_{ss} are the CSS conditions which enforce that the bed conditions at the beginning of the cycle match with those at the end of the cycle, t_f; f is the objective function which can be minimizing power consumption or maximizing recovery/profit etc. Typical applications of optimization are cycle design, parameter estimation of adsorbent properties, optimization of operating conditions and online optimization.

The partial differential equations (2.1) in the above optimization problem can be easily transformed into a set of differential equations by discretizing in the spatial domain(s) using method of lines [58]. The discretization schemes can vary from simple finite difference methods (e.g., backward or central difference) to more accurate finite volume methods with flux limiter schemes [46]. The latter scheme is used especially in PSA optimization studies [38] in order to capture the steep adsorption fronts by avoiding numerical smearing and physically unrealistic oscillations. Using the method of lines, the equations in (2.1) are discretized and the following optimization problem is formulated:

$$\min_{y,p} \quad f(z_i, t_f), p) \tag{2.2a}$$

$$s.t. \quad F(z_i(t), dz_i/dt, p, t) = 0 \tag{2.2b}$$

$$z_i(t_0) = y \tag{2.2c}$$

$$z_i(t) = y_B(t) \tag{2.2d}$$

$$w(z_i(t_f), p) \leq 0 \tag{2.2e}$$

$$c_{ss}(y, p) \equiv y - z_i(t_f, p) = 0 \tag{2.2f}$$

$$y_{min} \leq y, \; p^L \leq p \leq p^U \tag{2.2g}$$

where the subscript i indicates the discretized spatial node. The above discretized optimization problem (2.2) is a dynamic optimization problem with n_z differential state variables. It is important to note that initial conditions y are also optimization variables in the above problem, therefore, the optimization problem is at least of size n_z variables. Since PAPs work with multi-component feed mixtures and multiple beds, the resulting optimization problems can be large-scale problems of size $O(10^2 - 10^5)$.

There have been many techniques proposed to solve dynamic optimization problems.

In indirect approaches, the necessary condition of optimality, which is given as a set of algebraic differential equations, is solved [12]. However, treatment of boundary conditions and finding the switching of control variables to handle inequality constraints can be very difficult for large scale problems.

On the other hand, in direct approaches, a nonlinear programming (NLP) solver is applied to the problem directly. In the next section, two techniques of direct approaches, namely, *simultaneous* and *sequential* strategies, are discussed.

2.2 Simultaneous Strategy

The direct simultaneous approach is also referred to as the Direct Transcription approach [8], the complete discretization approach, or the all-at-once approach. Direct transcription of a dynamic optimization problem refers to the procedure of approximating the infinite dimensional problem by a finite dimensional one, which is then solved using a Nonlinear Programming (NLP) solver tailored to large scale problems [8]. In a dynamic optimization problem with differential equations, this approach typically approximates the equations using an appropriate discretization using, for example, an implicit Runge-Kutta method with high order accuracy and stability properties [40]. These methods directly couple the solution of the DAE system with the optimization problem; the DAE system is solved only once, at the optimal point, and therefore can avoid intermediate solutions that may not exist or may require excessive computational effort. This approach generates many equality constraints containing the discretized state equation with almost block-diagonal Jacobian matrix, exploited by sparse linear algebra packages. In PSA optimization studies, Agarwal et al. [3, 4] applied simultaneous approaches to small-scale PAP applications. However, these problems have steep moving fronts and large differences

in timescales, which require too many finite elements resulting in a very large optimization problem and thus may make the simultaneous approach intractable with current methods for optimization of PAP applications.

2.3 Sequential Strategy

In order to overcome the drawbacks of simultaneous approach, in this thesis we use the sequential strategy to solve dynamic optimization problems of PAP applications. In the sequential strategy (also known as control vector parameterization) [70], (2.2) is solved by decomposing it into three components as seen in Figure 2.1: (a) an Initial Value Problem (IVP) which requires numerical solution of the discretized dynamic model to evaluate the state profiles as well as objective and constraint functions from a given set of inputs y and p, (b) a sensitivity component which evaluates first and second order derivatives for the constraint and objective function, and (c) an NLP solver, which accepts function and derivative information from the first two components and updates the values of the input variables y and p. In a gradient based approach, function and derivative evaluations are obtained by combined solution of the IVP and related sensitivity components. In this way, the sequential approach decouples the solution of the optimization problem from solution of the embedded dynamic system, thus exploiting full advantage of state-of-the-art integration and NLP tools. Because of this decoupling, the resulting NLP problem is relatively smaller in size in terms of the optimization variables, y and p compared to the simultaneous approach. Moreover, each iteration is a feasible solution to the differential equations, hence functions are more likely to be well behaved. Furthermore, the stiffness of the differential equations is taken care of by the integrator, and no longer implies ill-conditioning in the resulting NLPs, which is quite important in the optimization of PAP

systems.

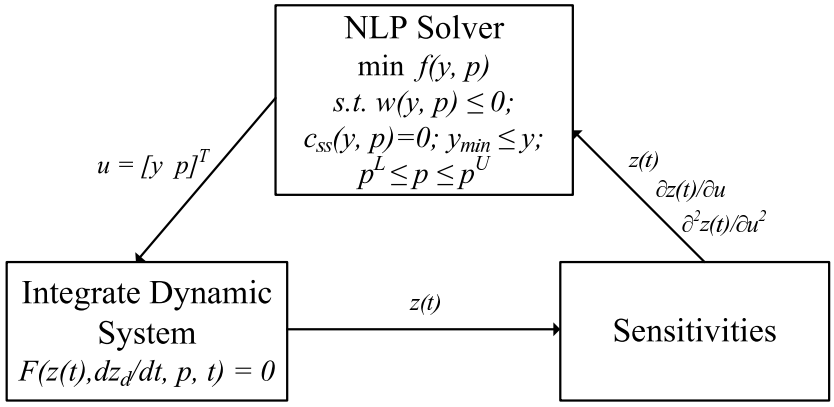

Figure 2.1: Schematic of the sequential approach to dynamic optimization

Gradients of the objective function with respect to the optimization variables are calculated either from direct sensitivity equations of the DAE system or by integration of the adjoint equations. Several efficient codes have been developed for both sensitivity methods [47]. However, the main bottleneck to using this approach for optimization of PAPs is the evaluation of direct sensitivities to obtain the dense constraint Jacobian and Hessian required by the NLP subproblem. For example, the evaluation of an exact Jacobian given by

$$A = \left[I_{n_z} - \frac{\partial z(t_f)}{\partial y} \vdots - \frac{\partial z(t_f)}{\partial p} \right] \tag{2.3}$$

requires integration of $O(n_z(n_z+n_p))$ sensitivities. Furthermore since the PAP differential equations are highly nonlinear, sensitivities with respect to the initial conditions of DAEs are mostly non-zeros. As a result, the integration of large set of sensitivity equations result in *dense constraint Jacobians*. The sparsity plot of constraint Jacobian for a simple PSA cycle is shown in Figure 2.2 where the red dots correspond to the non-zeroes. Moreover for accurate Hessian information that leads to rapidly converging iterative procedures [28], $O(n_z(n_z + n_p)^2)$ sensitivity equations have to be solved. Hence, the computational cost

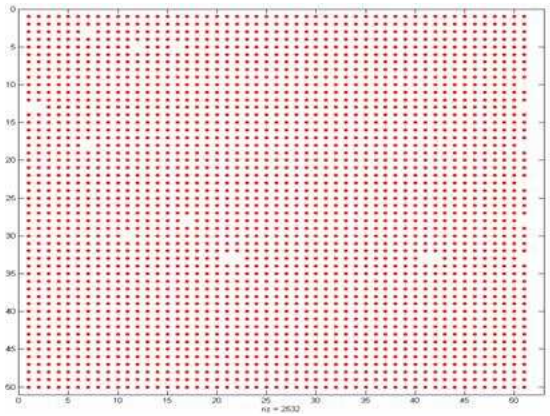

Figure 2.2: Sparsity plot of constraint Jacobian

per NLP iteration is very high [71].

Instead, for this work we consider a novel nonlinear programming algorithm for the sequential approach which overcomes the drawback of large computational times for Jacobian and Hessian evaluations. Here, the NLP algorithm works with exact Jacobian-vector and Hessian-vector products rather than forming and factorizing the dense Jacobian and Hessian matrices at every NLP iteration. These matrix-vector products can be evaluated accurately and efficiently by exploiting the direct/adjoint sensitivity equations and Automatic Differentiation. The derivation and formulation of these sensitivity systems to evaluate the matrix-vector products is explained in detail in Chapter 3.

2.4 Approaches for the Solution of NLP Problem

A general form of NLP can be described as

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x^L \leq x \leq x^U \end{aligned} \tag{2.4}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the nonlinear equality constraints $c : \mathbb{R}^n \to \mathbb{R}^m$ with $m \leq n$ are assumed to be sufficiently smooth and at least twice differentiable functions in x. Problems with general inequality constraints "$d(x) \leq 0$" can be reformulated in the above form by introducing slack variables.

In PAP optimization problems, the degrees of freedom, $(n-m)$ are often fewer than the total number of variables, n. For such problems with few degrees of freedom, specialized reduced Hessian SQP methods have been developed. These methods are often general purpose and often approximate second-order information by quasi-Newton positive definite matrices [10, 17]. These methods rely upon line-searches for global convergence. For difficult NLPs, though, these algorithms are inadequate in that they suffer from line-search failures, converge to saddle points, or have slow progress towards a local solution. Trust-region methods [23] have been derived to overcome these drawbacks. This dissertation focuses on NLP algorithms based on trust region methods, because of several attractive properties: they control the quality of steps even in the presence of ill-conditioned Hessians and Jacobians, and do not require that the reduced Hessian matrix be positive definite.

For the sake of explanation, we consider equality constrained optimization problems, i.e. we ignore bounds in (2.4). Trust region based successive quadratic programming (SQP) algorithms repeatedly solve the following quadratic program for the kth iteration

$$\min_d \quad \nabla_x f(x_k)^\top d + \tfrac{1}{2} d^\top B(x_k) d \tag{2.5a}$$

$$\text{s.t.} \qquad c(x_k) + A(x_k) d = 0 \tag{2.5b}$$

$$||d|| \leq \Delta_k \tag{2.5c}$$

to compute a new step d_k for a given iterate x_k, trust region radius Δ_k and Lagrange multipliers, λ_k. Here $d \in \mathbb{R}^n$; the exact matrix of the dense constraint gradients at x_k is given by,

$$A(x_k) = (\nabla c_1(x_k), ..., \nabla c_m(x_k))^T \in \mathbb{R}^{m \times n}$$

and $B(x_k)$ is the exact Hessian of the Lagrange function defined by

$$L(x, \lambda) = f(x) + \lambda^T c(x) \tag{2.6}$$

Notice that if x_k is far from satisfying one or more of the linearized constraints in (2.5) and Δ_k is small, then the set of constraints may be inconsistent; i.e., there may be no feasible points in the subproblems. This is illustrated in Figure 2.3, a problem in two dimensions and one linear constraint. Clearly, there is no step from x_k that stays within the spherical trust region constraint and satisfies the linear constraint.

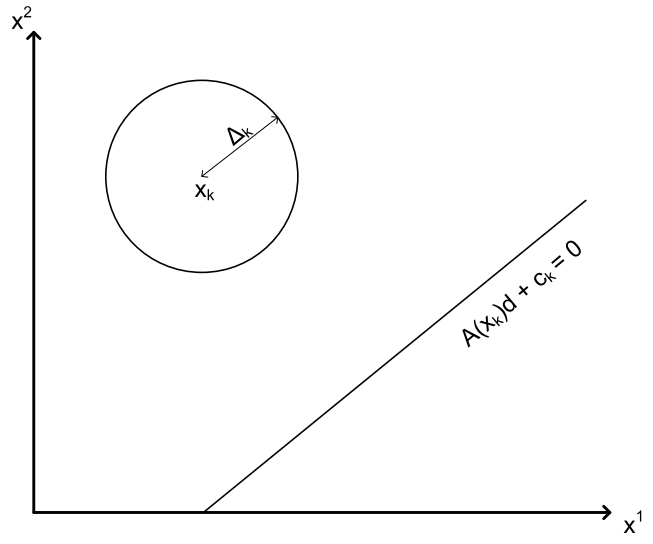

Figure 2.3: Example of inconsistent constraints

Many different ways have been proposed for applying a trust region constraint in a manner that does not generate inconsistent subproblems. Fletcher [27] proposed to

put the linearized equality constraints (2.5b) into the objective (2.5a) in the form of a penalty term, leaving only the trust region constraint (2.5c), thus replacing (2.5) with the subproblem

$$\min_d \quad \nu_k(\nabla_x f(x_k)^\top d + \tfrac{1}{2} d^\top B(x_k) d) + \|c(x_k) + A(x_k)d\|_1$$
$$||d||_\infty \leq \Delta_k$$

The l_∞ norm has the advantage of easily incorporating inequality bound constraints into the subproblem, but the disadvantage of making the exact solution of the subproblem combinatorial. Also, the l_1 term is not continuous at points of feasibility, so that the solution of the subproblem requires reformulation or nonsmooth optimization techniques. Conn, Gould and Toint [21] follow the same approach but instead use the augmented Lagrangian function, solving the subproblems of the form

$$\min_x \quad f(x) + \hat{\lambda}^T c(x) + \tfrac{1}{2\mu_k} c(x)^T c(x)$$
$$||x - x_k||_\infty \leq \Delta_k$$

The objective function is now smooth and can be minimized by standard optimization techniques. However, the trust region constraint still has the combinatorial l_∞ norm and hence not suitable for large problems. The commercial software product LANCELOT [22] is a successful implementation of this approach.

The other general approach for overcoming inconsistency is to relax the linearized equality constraints when necessary to allow feasible subproblem solutions. This idea is illustrated in Figure 2.4 for the simple problem considered in Figure 2.3. If an appropriate vector π_k is chosen for this problem and the linear equality constraint is changed to

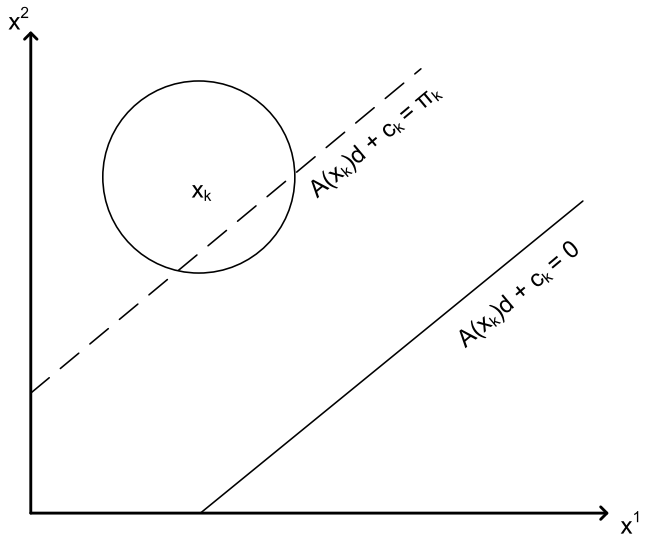

Figure 2.4: Example of relaxed constraints becoming consistent

$A(x_k)d + c_k = \pi_k$ (shown as the dashed line), then the relaxed constraint falls inside the trust region constraint. Thus, the new subproblem

$$\min_d \quad \nabla_x f(x_k)^\top d + \tfrac{1}{2} d^\top B(x_k) d \tag{2.8a}$$

$$\text{s.t.} \qquad c(x_k) + A(x_k)d = \pi_k \tag{2.8b}$$

$$||d||_2 \leq \Delta_k \tag{2.8c}$$

admits solutions d that lie on the relaxed constraint (2.8b). Both Vardi [69] and Byrd, Schnabel and Schultz [14] proposed subproblem formulation (2.8), and both selected $\pi_k = -\alpha_k c_k$ for some $\alpha_k \in [0, 1)$. Byrd, Schnabel and Schultz [14] solve (2.8) by decomposing the step d into two parts that can be solved separately. They write $d = \alpha_k n_k + Z(x_k)p_Z$, where $n_k = -A(x_k)^T(A(x_k)A(x_k)^T)^{-1}c_k$ and u solves the reduced subproblem

$$\min_{p_Z \in \mathbb{R}^{n-m}} \quad (\nabla_x f(x_k) + \alpha_k B(x_k) n_k)^\top Z(x_k) p_Z + \tfrac{1}{2} p_Z^\top Z(x_k)^T B(x_k) Z(x_k) p_Z \tag{2.9a}$$

$$||Z(x_k)p_Z||_2 \leq \sqrt{\Delta_k^2 - \alpha_k ||n_k||^2} \tag{2.9b}$$

which results from substitution of the formula for d into (2.8). Z_k is null space matrix such that $A(x_k)Z(x_k) = 0$. The main difficulty of this algorithm is choosing α_k intelligently. The choice of α_k strongly influences algorithm performance because it determines the relative sizes of $\|n_k\|$ and $\|Z_k p_Z\|$, and hence controls whether d tends more to satisfy constraint feasibility or to minimize the objective function.

2.4.1 Byrd-Omojokun Algorithm

The formulation of Byrd and Omojokun [52] is used to make the constraints (2.5) consistent. They first compute a step n that satisfies a certain relaxation of the linearized equality constraints defined by the subproblem

$$\min_{n\in\mathbb{R}^n} \| c(x_k) + A(x_k)n \|^2, \qquad s.t. \quad \| n \|_2 \leq \zeta\Delta_k \tag{2.10}$$

where $\zeta \in (0, 1)$. Problem (2.10) is called the *normal subproblem*, and its solution n_k is called the *normal step* at x_k. Now the original subproblem (2.5) is reformulated as

$$\min_d \quad \nabla_x f(x_k)^\top d + \tfrac{1}{2} d^\top B(x_k) d \tag{2.11a}$$

$$\text{s.t.} \qquad A(x_k)d = A(x_k)n_k \tag{2.11b}$$

$$||d||_2 \leq \Delta_k \tag{2.11c}$$

The new formulation is obviously consistent because the choice $d = n_k$ satisfies both the constraints (2.11b)-(2.11c). An interesting feature of this formulation is that when $m = n$, i.e., when the problem reduces to that of finding the root of a system of nonlinear equations, then only the normal subproblem remains, and now the Byrd-Omojokun method coincides with the well-known Levenberg-Marquardt method. The normal step

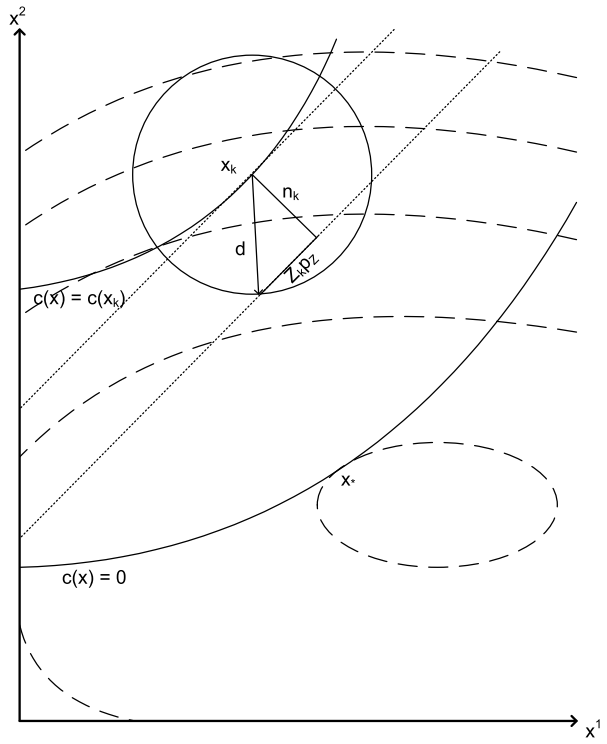

Figure 2.5: Byrd-Omojokun Composite Step

n is restricted to lie in the range space of $A(x_k)^T$, and the other part of d is confined to the null space spanned by the matrix $Z(x_k)$; thus $d = n + Z(x_k)p_Z$. Substituting d into (2.11), results in

$$\min_{p_Z \in \mathbb{R}^{n-m}} \quad (\nabla_x f(x_k) + B_k n_k)^\top Z(x_k) p_Z + \tfrac{1}{2} p_Z^\top Z(x_k)^T B(x_k) Z(x_k) p_Z \tag{2.12a}$$

$$||Z(x_k)p_Z||_2 \leq \sqrt{\Delta_k^2 - \|n_k\|^2} \tag{2.12b}$$

This is called the *tangential subproblem*, and the component $Z(x_k)p_Z$ found from it is called the *tangential step* at x_k. Figure 2.5 illustrates the normal and tangential steps for a simple problem with two unknowns and one nonlinear equality constraint. The dashed elliptical lines represent level curves of the objective $f(x)$ with the minimum in the lower right part of the picture. The solid line is the equality constraint, and the broken circle is

the trust region constraint. The null manifold is shown as a dotted line through the point x_k, and the normal step runs perpendicular to it. This same manifold translated to the point $x_k + n_k$ defines the possible set of tangential steps, as shown by the lower dotted line. The final step d in this case reaches all the way to the trust region constraint.

The NLP algorithms developed in this thesis are based on the efficient Byrd-Omojokun method because it breaks the original equality constrained problem (2.5) into two smaller, simpler subproblems, each with a single trust region constraint, and because its performance does not hinge on extraneous algorithm parameters. The outline of Byrd-Omojokun algorithm is given below

Algorithm 2.4.1. *General Description of Byrd-Omojokun Algorithm*

Constant $\epsilon > 0$

Initialization: Choose any x_0 *and* Δ_0

LOOP, starting with $k = 0$

1. *Compute* c_k, ∇f_k *and* $A(x_k)$
2. *Compute Lagrange multipliers* λ_k
3. *IF* $\|\nabla f_k - A(x_k)\lambda_k\|_\infty < \epsilon$ *AND* $\|c_k\|_\infty < \epsilon$ *THEN STOP*
4. *Solve the normal problem* (2.10) *for* n_k
5. *Compute* $Z(x_k)$ *such that* $A(x_k)Z(x_k) = 0$
6. *Compute Hessian,* $B(x_k)$ *or an approximation to it*
7. *Solve the tangential subproblem* (2.12) *for* $p_Z(x_k)$
8. *Compute the overall step,* $d_k = n_k + Z(x_k)p_Z(x_k)$

9. Update trust-region radius

10. CONTINUE, after incrementing k

Implementations of the Byrd-Omojokun approach for equality constrained NLPs are described in [45] and reviewed extensively in [23] and [51]. Moreover, [5, 19, 16] uses a related approach that deals with handling inequality constraints and/or bounds.

2.5 Byrd-Omojokun Algorithm for Dense Jacobians

In this dissertation we show how the Byrd-Omojokun algorithm is tailored to handle dense constraint Jacobians. As mentioned in the previous section, the normal subproblem requires the solution of a linear solution of the form

$$A(x_k)A(x_k)^T v = b$$

where $A(x)$ is exact matrix of constraint gradient at x. Furthermore, a representation $Z(x)$ of null space of $A(x)$ is needed frequently for the computation of the next step. For these reasons, the explicit forming and factoring of the constraint Jacobian $A(x)$ provides an efficient step calculation if $A(x)$ is sparse and well structured, see, e.g.,[5]. However, this approach may result in very time-consuming computations, especially if the Jacobian of the constraints is dense or unstructured, for example, in the case of optimization problems related to PAP applications. In such problems, the run-time needed for optimization process is dominated significantly by the computation of the dense Jacobian and its factorization. In order to circumvent these dense calculations, we present in this thesis a specialized Byrd-Omojokun trust-region algorithm that does not require the exact evaluation of the constraint Jacobian.

2.6 Concluding Remarks

A general dynamic optimization problem of periodic adsorption processes is presented. Both the simultaneous and sequential approaches are briefly described. It is shown that sequential approach is preferred over simultaneous approach for the optimization of PAPs. However, the main bottleneck to using this approach is the evaluation of direct sensitivities to obtain the dense constraint Jacobian and Hessian required by the NLP subproblem. This issue is addressed in the next chapter which describes the computation of Jacobian-vector and Hessian-vector products using adjoint sensitivities.

An overview of approaches for solving the NLP subproblem based on trust-region SQP methods is presented. Byrd-Omojokun method is known to be efficient because it breaks the original equality constrained problem into two smaller and simpler subproblems. In this thesis, the Byrd-Omojokun algorithm is tailored to handle dense constraint Jacobians and the specialized algorithm works with exact second order information in the form of Hessian-vector products.

Chapter 3

Sensitivity Equations for Gradient Evaluations

Synopsis

The sequential approach for numerical solution of dynamic optimization problems requires sensitivity information with respect to hundreds of optimization variables. Inexpensive calculation of sensitivities is particularly important in PAP applications because the systems of interest are typically modeled with $10^2 - 10^5$ equations. The NLP algorithms developed in this thesis work with exact Jacobian-vector and Hessian-vector products instead of full Jacobian and Hessian. This chapter describes the cost-efficient and accurate sensitivity methods which have been implemented in this thesis to compute these matrix-vector products required by the NLP subproblem.

3.1 Requirements of Gradient based NLP Algorithms

For the solution of generally constrained NLP subproblem in (2.2) (Figure 2.1) SQP and interior point methods have proven to be mostly efficient. Both methods require the gradients of the objective function and the gradients of the constraint Jacobian with respect to the unknowns $\tilde{p}^\top = [y^\top \; p^\top] \in \mathbb{R}^{n_z+n_p}$. In case of (2.2), these are

$$\frac{df(z(t_f,p))}{d\tilde{p}}, \; \frac{dw(z(t_f,p)}{d\tilde{p}}, \; \frac{dc_{ss}(z(t_f,p)}{d\tilde{p}} \tag{3.1}$$

Furthermore, these methods require an approximation of the Hessian of the Lagrangian (2.6) with respect to $\tilde{p}$:

$$L_{pp}(p,\lambda) = \frac{d^2 L(\tilde{p},\lambda)}{d\tilde{p}^2} \tag{3.2}$$

Optimization algorithms show an improved robustness and less iterations, if the exact Hessian instead of an approximation is employed, as shown by Vassiliadis et al. [71]. However, computation of full Hessian requires integrating $O(n_z(n_z+n_p)^2)$ sensitivity equations which is computationally expensive. Vassiliadis et al. use second-order forward sensitivity equations to obtain the Hessian and therefore have the drawback of a large computational effort. The number of iterations decreases, but the overall computational time increases. This issue is dealt in the Section 3.4.2.

In this work, we use "differentiate then discretize" strategy as opposed to the "discretize then differentiate". Optimization approaches are frequently called discretize-then-differentiate methods when using the discrete derivatives provided for example by Automatic Differentiation (AD) and optimize-then-differentiate methods when the derivatives are based on the continuous formulation. Our "differentiate then discretize" approach uses Automatic Differentiation (AD) to construct the relevant system of equations required by

the integration subroutines rather than discretizing the state system before applying AD directly. In this way, we avoid the differentiation of a large and complex integration code with corrector iterations and error control, and we exploit efficient integration procedures for the sensitivity system and this strategy results in accurate derivatives.

Forward sensitivity and adjoint equations are the two options for gradient evaluation based on continuous derivative formulation. All the sensitivity equations in this chapter are derived for the PAP optimization problems as in (2.2). We start by considering an ordinary differential equation (ODE) model:

$$z' \equiv \frac{dz}{dt} = \tilde{F}(z,p), \;\; t \in (t_0, t_f] \qquad z(t_0) = y \tag{3.3}$$

Now we show how the first-order derivatives i.e., gradient of objective function and the gradient of constraint Jacobians are computed using the direct and adjoint sensitivities.

3.2 Forward Sensitivity Equations

The direct sensitivity equations are obtained by differentiating the original model equations (3.3) with respect to the parameters, y and p. Defining, $s_p(t) = \frac{\partial z(t)}{\partial p}$ and $s_y(t) = \frac{\partial z(t)}{\partial y}$, formal differentiation of the differential equations results in,

$$\dot{s}_y = \tilde{F}_z s_y \qquad s_y(t_0) = I_{n_z} \tag{3.4a}$$

$$\dot{s}_p = \tilde{F}_z s_p + \tilde{F}_p \qquad s_p(t_0) = 0 \tag{3.4b}$$

where

$$\tilde{F}_y = \begin{pmatrix} \frac{\partial \tilde{F}_1}{\partial z_1} & \cdots & \frac{\partial \tilde{F}_1}{\partial z_{n_z}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{F}_{n_z}}{\partial z_1} & \cdots & \frac{\partial \tilde{F}_{n_z}}{\partial z_{n_z}} \end{pmatrix} \qquad \tilde{F}_p = \begin{pmatrix} \frac{\partial \tilde{F}_1}{\partial p_1} & \cdots & \frac{\partial \tilde{F}_1}{\partial p_{n_p}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{F}_{n_z}}{\partial p_1} & \cdots & \frac{\partial \tilde{F}_{n_p}}{\partial p_{n_p}} \end{pmatrix}$$

The sensitivity equations (3.4) depend on states $z(t)$ and can be solved simultaneously with the state system in (3.3). The structure of the linear sensitivity equations has been exploited in DAE/ODE solvers by using methods like staggered-corrector techniques since the state and the direct sensitivity system share the same Jacobian, $\tilde{F}_y$ [26, 47].

In this work, we consider the direct sensitivity equations to evaluate the exact Jacobian of constraints given by

$$A \equiv \nabla c_{ss}^T = \left[I_{n_z} - \frac{\partial z(t_f)}{\partial y} \quad \vdots \quad - \frac{\partial z(t_f)}{\partial p} \right] \tag{3.5}$$

which requires integrating $O(n_z(n_z+n_p))$ differential equations. This can be prohibitively time consuming, particularly, for large problems. For example, optimization of 5-bed 11-step PSA process with 1000 state variables [39] took 2 CPU h to form each Jacobian matrix at every iteration on a 2.4 GHz processor using forward sensitivities. Consequently, they must be adapted to reduce computational cost.

3.3 First Order Adjoint Sensitivities

We demonstrate the application of the adjoint sensitivity analysis with a point-form functional $\tilde{\varphi}(z(t_f), \tilde{p})$ in order to determine the gradient of objective function i.e., $\tilde{\varphi} \equiv f$

$$\frac{d\tilde{\varphi}(z(t_f, \tilde{p}))}{d\tilde{p}} \tag{3.6}$$

Instead of using the chain rule, the gradients are computed by introducing the so-called adjoint variables $\tilde{\lambda}(t) \in \mathbb{R}^{n_z}$ (Cao et al. [18]). The adjoint variables are computed by the integration of the first-order adjoint equations:

$$\begin{aligned} \dot{\tilde{\lambda}}^T + \tilde{\lambda}^T \tilde{F}_z = 0 \;\; t \in [t_0, t_f] \\ \tilde{\lambda}^T(t_f) = \tilde{\varphi}_z(t_f) \end{aligned} \tag{3.7}$$

The gradients are computed by the formula

$$\frac{\partial \tilde{\varphi}}{\partial y} = \tilde{\varphi}_y(t_f) + \tilde{\lambda}(t_0) \tag{3.8}$$

and

$$\frac{\partial \tilde{\varphi}}{\partial p} = \tilde{\varphi}_p(t_f) + \int_{t_0}^{t_f} \tilde{\lambda}^T \tilde{F}_p dt \tag{3.9}$$

The adjoint method is preferred when there is a single functional and many parameters because only one adjoint system is solved compared to $n_z + n_p$ direct sensitivity systems. The implementation of the adjoint sensitivity method consists of three major steps. First, solve the original different equations (3.3) forward till final time, t_f. Second, at t_f we compute the initial conditions of the adjoint system (3.7). Finally, we solve the adjoint system backward to the start point, and calculate the gradients (3.8)-(3.9). The backward integration of the adjoint equations requires the information of state formation, $z(t)$ which have to be stored during the forward integration phase. Efficient integration codes [59] employ *check-pointing schemes* to balance between storage space and execution time.

3.4 Jacobian-vector and Hessian-vector Products

As pointed out in the previous chapter, the optimization algorithms developed in this thesis avoid forming the computationally expensive constraint Jacobian by working with Jacobian-vector products. Moreover, the NLP algorithms work with exact second-order information in the form of Hessian-vector products. The forward sensitivity equation for exact Jacobian evaluation (3.4) depends on the number of parameters, (n_z+n_p) in the state equations. Since, the initial conditions are also parameters in the optimization problems related to PAPs, the parameter space in these problems is usually large. Hence, evaluating the matrix-vector products using direct sensitivities is computationally expensive. To evaluate the matrix-vector products, we solve sensitivity systems for a given direction in the parameter space so that the dependence on the number of parameters can be significantly reduced.

3.4.1 Jacobian-vector products, Av and A^Tw

Let, $v = [v_y \quad \vdots \quad v_p]^T, \quad v_y \in \mathbb{R}^{n_z}$ and $v_p \in \mathbb{R}^{n_p}$ be the direction along which the sensitivities have to be solved. The direction, v can appear in the NLP step-computation as in,

$$y_{k+1} = y_k + \alpha v_y \qquad p_{k+1} = p_k + \alpha v_p \tag{3.10}$$

where α is the step size along search directions.

Jacobian-vector product, Av:

The equations used in evaluating the Jacobian-vector product, Av are given by *directional*

first order sensitivities for the state system (3.3),

$$\dot{s} = \tilde{F}_z s + \tilde{F}_p v_p \qquad s(t_0) = y_{\tilde{p}} v = v_y \tag{3.11}$$

where $s(t) = \frac{\partial z(t)}{\partial \tilde{p}} v \equiv z_{\tilde{p}} v \in \mathbb{R}^{n_z}$. The Jacobian-vector product, Av given by

$$\begin{aligned} Av &= v_y - \frac{\partial z(t_f)}{\partial y} v_y - \frac{\partial z(t_f)}{\partial p} v_p \\ &= v_y - z_{\tilde{p}}(t_f) v \\ &= v_y - s(t_f) \end{aligned} \tag{3.12}$$

is computed by integrating both the state equations (3.3) and directional forward sensitivity equations (3.11) simultaneously forward in time. It is important to note that directional forward sensitivity equations are of size n_z which is equal to the number of states in the model.

vector-Jacobian product, $A^T w$:

The vector-Jacobian products is computed using the first-order adjoint sensitivities (3.7) with the point-form functional chosen as,

$$\tilde{\varphi}(z(t_f), \tilde{p}) = w^T (y - z(t_f)) \tag{3.13}$$

where $w \in \mathbb{R}^m$. The forward integration of the state equation (3.3) followed by the backward integration of (3.7) with the integrals (3.8)-(3.9) updated on the way results in the vector-Jacobian product.

3.4.2 Hessian-vector products, Bv

Exact second-order derivative information in the form of Hessian-vector product is computed using the *directional second order adjoint sensitivities* [53]. The second-order adjoint equations are obtained by differentiating the first-order adjoint equations with respect to $\tilde{p}$:

$$\dot{\tilde{\lambda}}_{\tilde{p}} + \tilde{F}_z^T \tilde{\lambda}_{\tilde{p}} + \tilde{\lambda}^T (\tilde{F}_{z\tilde{p}} + \tilde{F}_{zz} z_{\tilde{p}}) = 0 \;\; t \in [t_0, t_f] \tag{3.14}$$

$$\tilde{\lambda}_{\tilde{p}}(t_f) = \tilde{\varphi}_{zz}(t_f) z_{\tilde{p}}(t_f) + \tilde{\varphi}_{z\tilde{p}}(t_f) \tag{3.15}$$

Instead of solving an $n_z \times n_p$ system of ODEs in (3.14), we can post-multiply it by a direction vector $v \in \mathbb{R}^{n_y+n_p}$ to obtain the *directional second-order adjoint sensitivities* (dSOA)

$$\dot{\tilde{\lambda}}_{\tilde{p}} v + \tilde{F}_z^T \tilde{\lambda}_{\tilde{p}} v + \tilde{\lambda}^T (\tilde{F}_{z\tilde{p}} v + \tilde{F}_{zz} z_{\tilde{p}} v) = 0 \;\; t \in [t_0, t_f] \tag{3.16}$$

$$\tilde{\lambda}_{\tilde{p}v}(t_f) = \tilde{\varphi}_{zz}(t_f) z_{\tilde{p}}(t_f) v + \tilde{\varphi}_{z\tilde{p}}(t_f) v \tag{3.17}$$

Now by defining $\mu \equiv \lambda_{\tilde{p}} v$, the dSOA equations become

$$\dot{\mu} + \tilde{F}_z^T \mu + \tilde{\lambda}^T (\tilde{F}_{z\tilde{p}} v + \tilde{F}_{zz} s) = 0 \;\; t \in [t_0, t_f] \tag{3.18}$$

$$\mu(t_f) = \tilde{\varphi}_{zz}(t_f) s(t_f) + \tilde{\varphi}_{z\tilde{p}}(t_f) v \tag{3.19}$$

where $s \equiv z_{\tilde{p}} v$ is obtained by solving the directional version of the first order forward sensitivity equations (3.11). The directional second order derivative of $\tilde{\varphi}(\tilde{p})$ is obtained

from

$$\frac{\partial^2 \tilde{\varphi}}{\partial \tilde{p}^2} v = \tilde{\varphi}_{\tilde{p}\tilde{p}}(t_f)v + \tilde{\varphi}_{\tilde{p}\tilde{z}}(t_f)s(t_f) + [(\tilde{\lambda}^\top z_{\tilde{p}\tilde{p}}v + z_{\tilde{p}}^\top \mu]_{t=t_0} + \int_{t_0}^{t_f} \tilde{F}_{\tilde{p}}^\top \mu + \tilde{\lambda}^\top (\tilde{F}_{\tilde{p}\tilde{p}}v + \tilde{F}_{\tilde{p}z}s)dt \tag{3.20}$$

For Hessian-vector product evaluation, the point-form function, $\tilde{\varphi}$ is chosen as the product of Lagrange function (for example, (2.6)) and an arbitrary vector, v. The advantage of using this kind of a formulation is that the directional direct sensitivity systems (3.11), first-order adjoint sensitivities (3.7) and the directional second-order adjoint sensitivities are all of dimension n_z and are weakly dependent on the number of parameters $n_z + n_p$. Moreover, the cost ratio to compute an Hessian-vector product compared to a simulation is just between 2 to 4 and the obtained derivatives are more accurate than finite-difference methods.

The formulations of the RHS of sensitivity, adjoint and the quadrature systems contain several vector-matrix, matrix-vector, vector-matrix-vector products i.e. $\tilde{F}_z s + \tilde{F}_p v_p$, $\lambda^T \tilde{F}_z$, $\tilde{\lambda}^T \tilde{F}_p$, $\tilde{F}_z^T \tilde{\lambda}_{\tilde{p}} + \tilde{\lambda}^T (\tilde{F}_{z\tilde{p}} + \tilde{F}_{zz} z_{\tilde{p}})$, $\tilde{F}_z$ etc. These terms should be calculated accurately and efficiently for accurate and cheap sensitivities. Automatic differentiation (AD) allows the computation of exact derivatives for functions given as computer programs [31]. AD is applied to calculate the above products of derivative matrices and vectors since the corresponding computational effort is bounded above by a small multiple of the computational effort to evaluate the function itself. Also, since the model equations (3.3) and the directional direct sensitivity equations (3.11) are initial value problems and the first order adjoint (3.7) and dSOA (3.18) systems are terminal value problems, an IVP Sensitivity solver with forward and backward integration capability is required.

Implementation Overview

The embedded dynamic system in (3.3) is integrated forward in time along with directional first order sensitivity equations (3.11) and $z(t)$ and $s(t)$ are stored at checkpoints in time. Then the final values for both the adjoint systems (3.7) and (3.18) are calculated. The first order adjoints equations (3.7) are integrated backward in time along with directional second order adjoint equations (3.18) with the desired integrals (3.20) updated on the way to compute the gradients.

3.5 Concluding Remarks

We present a continuous derivative formulation to compute the necessary derivative information required by the NLP subproblem. Efficient directional forward and adjoint equations are discussed to compute accurate Jacobian-vector and Hessian-vector products. The methodology relies heavily on automatic differentiation, since AD is the only reliable and efficient technology for evaluation of the RHS of the sensitivity systems.

Given that we can compute accurate and efficient Jacobian-vector and Hessian-vector products, the next two chapters focus on the novel nonlinear programming algorithms developed in this work for handling dense constraint Jacobians. In order to efficiently handle these Jacobians during optimization and reduce the computation time, this work presents new composite step trust-region algorithms for the solution of minimization problems with both nonlinear equality and inequality constraints. Furthermore, numerical performance results for these algorithms are presented.

Chapter 4

SQP Trust-Region Algorithm with Inexact Jacobians

Synopsis

This chapter presents a new SQP trust-region algorithm for the solution of minimization problems with both nonlinear equality and inequality constraints. Instead of forming and factoring the dense constraint Jacobian, this algorithm approximates the Jacobian of equality constraints with a specialized quasi-Newton method. Hence it is well suited to solve PAP optimization problems. In addition to allowing inexactness of the Jacobian and its null-space representation, this algorithm also provides exact Hessian-vector products to improve the convergence rate. A five-fold reduction in computation is demonstrated with this approach for two PAP optimization problems.

4.1 Optimization Strategy

We discuss a trust region based sequential quadratic programming (SQP) algorithm that forms the basis of this chapter. This algorithm is formulated for the solution of NLP problems of the form

$$\begin{aligned} \min_{\{x_1,x_2\}} \quad & f(x_1,x_2) \\ \text{s.t.} \quad & c(x_1,x_2)=0 \\ & x_1^L \le x_1 \le x_1^U \end{aligned} \tag{4.1}$$

where the objective function $f:\mathbb{R}^n \to \mathbb{R}$ and the nonlinear equality constraints $c:\mathbb{R}^n \to \mathbb{R}^m$ with $m \le n$ are assumed to be sufficiently smooth and at least twice differentiable functions in $x^T=[x_1^T,x_2^T]$. with $x_1 \in \mathbb{R}^{n-m}$. Note that only bounds on x_1 are considered in this work, and, as will be seen later, this allows us to take advantage of robust and efficient trust region-based optimization methods. Section 4.1.1 describes a work around to handle general inequality constraints and bounds.

For the majority of the trust-region SQP algorithms, the computation of the next iterate requires the solution of a linear system of the form

$$A(x^k)A(x^k)^\top v = b$$

where

$$A(x) = (\nabla c_1(x), ..., \nabla c_m(x))^\top \in \mathbb{R}^{m\times n}$$

is the exact matrix of constraint gradient at x. Furthermore, a representation $Z(x)$ of null

space of $A(x)$ is needed frequently for the computation of the next step. For these reasons, the explicit forming and factoring of the constraint Jacobian $A(x)$ provides an efficient step calculation if $A(x)$ is sparse and well structured, see, e.g.,[5]. As an alternative, one may use iterative system solves up to a certain accuracy, for example, Krylov subspace [15] or multigrid methods, for efficient step computation in each iteration. However, these iterative methods assume that the Jacobian can be obtained at low cost and focus only on the efficient solution of the linear system. However, this approach may result in very time-consuming computations, especially if the Jacobian of the constraints is dense or unstructured, for example, in the case of optimization problems related to PAP applications. In such problems, the run-time needed for optimization process is dominated significantly by the computation of the dense Jacobian and its factorization. For example, optimization of a 5-bed 11-step PSA process with 1000 state variables [39] using a reduced Hessian SQP approach [10] took almost 200 CPU h to converge and the computation of a single Jacobian took 2 h which was calculated from the direct sensitivity approach applied to the PDAE system. In short, the current NLP algorithms are prohibitively time consuming and memory intensive for the optimization of PAP problems with only a few hundred variables. Therefore, the bottleneck to efficient calculations is the evaluation of direct sensitivities to obtain the dense Jacobian matrix. In order to circumvent these dense calculations, we present a class of trust-region SQP algorithms that does not require the exact evaluation of the constraint Jacobian or an iterative solution of a linear system with a system matrix that involves the constraint Jacobian.

4.1.1 Handling General Inequalities and Bounds

We now consider the reformulation of (2.2) to (4.1) and carefully choose the inequality constraints and bounds on variables y and p. First, we note that without bounds on the state variables, $y \geq y_{min}$ the DAE model may not be solvable over the course of the optimization (e.g. if the states are negative). However, since these bounds are unlikely to be active at the solution, we do not enforce them explicitly as variable bounds, but treat them with a simple constraint aggregation approach. Our constraint aggregation method uses the KS function [11] to lump all the bound constraints into a simple composite function given by

$$KS(y, \rho) = y_{min} - \frac{1}{\rho} ln \left(\sum_i exp(-\rho(y_i - y_{min})) \right) \geq 0 \tag{4.2}$$

where y_{min} is the lower bound of y_i which keeps $exp(-\rho(y_i - y_{min})) \leq 1$ and the KS function well-behaved. For active bounds, y_i becomes active, $y_i \to y_{min}$, $exp(-\rho(y_i - y_{min})) \to 1$ and these terms dominate in the KS function. Otherwise as ρ approaches infinity, we have $exp(-\rho(y_i - y_{min})) \approx 0$ for $y_i \gg y_{min}$.

With the KS function added to the inequality constraints, the NLP from (2.2), with DAEs solved implicitly for $z(t)$, is now given by:

$$\begin{aligned} \min_{\{p,y\}} \quad & f(p, y) \\ \text{s.t.} \quad & h(p, y) = 0 \\ & g(p, y) \leq 0 \\ & p^L \leq p \leq p^U \end{aligned} \tag{4.3}$$

which we convert to (4.1) by adding slack variables to the inequalities (e.g. purity and recovery constraints, KS function), i.e.,

$$g(p, y) + \hat{s} = 0 \tag{4.4}$$

with non-negative bounds on the slack variables, $\hat{s}$. The augmented set of equality constraints now includes the original set of equality constraints, h, and (4.4).

Finally, special care has to be taken on the bounds of the slack variables $\hat{s}$ since only $(n - m)$ decision variables can be bounded in (4.1). Here we choose a subset of decision variables, $\bar{p}$, that would normally be used to solve the equations of a *design specification problem*, which arises frequently in the modeling of PAPs. For example, a valve constant can be chosen for a pressure specification, or adjustment of a flow can be used to meet a purity specification. This design problem is posed as,

$$\begin{aligned} h(\bar{p}, y) &= 0 \\ g(\bar{p}, y) + \hat{s} &= 0 \end{aligned} \tag{4.5}$$

and it is expected that the Jacobian of these equations is always nonsingular with respect to y and $\bar{p}$. In this way, we can bound the rest of the free variables, $\hat{p}$ and the slack variables, $\hat{s}$. The variables and constraints in problem (4.3) are now reassigned to conform to problem (4.1) with

$$x_1 := \begin{bmatrix} \hat{p} \\ \\ \hat{s} \end{bmatrix}, \quad x_2 := \begin{bmatrix} y \\ \\ \bar{p} \end{bmatrix} \text{ and } \quad c = \begin{bmatrix} h \\ \\ g + \hat{s} \end{bmatrix}. \tag{4.6}$$

4.1.2 Solution Strategy

Problem (4.1) is solved by a successive quadratic programming (SQP) trust-region algorithm that repeatedly solves the following quadratic program for the kth iteration:

$$\begin{aligned} \min_{d} \quad & \nabla_x f(x^k)^\top d + \tfrac{1}{2} d^\top B(x^k) d \\ \text{s.t.} \quad & c(x^k) + A(x^k) d = 0 \\ & x_1^L \leq x_1^k + d_1 \leq x_1^U \\ & ||d|| \leq \Delta^k \end{aligned} \tag{4.7}$$

in order to compute a new step $d^k = [d_1^T, d_2^T]^T$ for a given iterate x^k, trust region radius Δ^k and Lagrange multipliers, λ^k. Here the exact matrix of the dense constraint gradients at x^k is given by,

$$A(x^k) = (\nabla c_1(x^k), ..., \nabla c_m(x^k))^T \in \mathbb{R}^{m \times n}$$

and $B(x^k)$ is the exact Hessian of the Lagrange function defined by

$$L(x, \lambda) = f(x) + \lambda^T c(x) \tag{4.8}$$

Furthermore, $||.||$ denotes the Euclidean norm $||.||_2$. Since problem (4.7) may not have a feasible solution, we follow a composite step approach proposed by Byrd and Omojokun [52]. For efficient step computation of dynamic optimization problems, the proposed algorithm exploits direct sensitivity equations and/or adjoint sensitivity equations to evaluate the products of exact Jacobian $A(x)$ and a given vector v, $A(x)v$ and $A(x)^T v$ and the product of exact Hessian $B(x)$ and a given vector v, $B(x)v$. Efficient computation of these matrix-vector products can also be computed by applying Automatic Differentiation

(AD). However, the repeated computation of the full matrix, $A(x)$, which requires many Jacobian-vector products, may be prohibitively time consuming. The nonlinear programming algorithm presented in this work uses exact Jacobian-vector and vector-Jacobian products and avoids forming the full Jacobian matrix. Instead of forming the full Jacobian matrix, $A(x^k)$ is approximated using a specialized quasi-Newton method. Also, in order to improve the convergence rate the algorithm uses exact Hessian-vector products.

Exhaustive literature on composite step trust region methods that employ exact derivative information can be found in [23]. Effects of inexact derivative information on the global convergence is studied in [15, 35] where the analysis is focused on inexactness arising due to iterative system solves. However, the inexactness in this work arises due to the Jacobian approximation using a quasi-Newton update. The convergence analysis and performance results focusing on equality constrained optimization problems are presented in [74, 75]. In this work, we extend the algorithm to general nonlinear programming problems with both equality and inequality constraints of the form (4.1).

We now consider an algorithm for solving the nonlinear programming problem of the form (4.1) which does not require construction of the exact Jacobian $A(x)$. The chapter is divided into two parts. In the first part, we will be focus on the inexact trust region algorithm for solving nonlinear equality constrained optimization problems i.e. we ignore bounds in (4.1). In the second part, we extend the inexact algorithm for the nonlinear equality constrained optimization problems to general nonlinearly constrained optimization problem of the from (4.1). Numerical performance results for both the algorithms are presented thereafter.

We use the following notation for the rest of the chapter. Let A^k be the approximation of the exact Jacobian, $A(x^k)$ for each iteration k. We suppose that an approximation

$Z^k \in \mathbb{R}^{n\times(n-m)}$ of an exact null space basis $Z(x^k)$ can be constructed from the approximate Jacobian A^k such that $A^k Z^k = 0$ and $A(x^k)Z(x^k) = 0$.

4.2 Jacobian Approximation via TR1 Update

For PDAE-constrained optimization problems, the Jacobian of constraints is obtained by either numerical differentiation or analytical methods such as direct sensitivity equations. However, both approaches may result in very time-consuming computations, especially if the Jacobian of constraints is dense and unstructured. In order to avoid forming the dense Jacobian at very step, we apply the two-sided rank one (TR1) update to generate an approximation A_k of the exact constraint Jacobian, $A(x_k)$ as proposed in [33]. The TR1 update is defined by,

$$A^{k+1} = A^k + \frac{(q_k - A^k\delta_k)(\mu_k^T - \sigma_k^T A^k)}{(\mu_k^T - \sigma_k^T A^k)\delta_k} \tag{4.9}$$

where,

$$q_k \equiv c(x^k + \delta_k) - c(x^k), \quad \mu_k^T \equiv \nabla_x L^T(x^k + \delta_k, \lambda^k + \sigma_k) - \nabla_x L^T(x^k + \delta_k, \lambda^k)$$

$$\sigma_k \equiv \lambda_{k+1} - \lambda^k \in \mathbb{R}^m \qquad \delta_k = x^{k+1} - x^k$$

One important feature of this update is that it is possible to reconstruct the exact Jacobian, $A(x^k)$ with at most m TR1 updates for a fixed iterate x^k [75]. The matrix vector product, $A^k\delta_k$ and the vector matrix product, $\sigma_k^T A^k$ in the rank-one term of (4.9) can be computed efficiently and accurately using sensitivities and Automatic Differentiation. Also, the update uses the information of the objective function at the current iteration,

x^k unlike other quasi-Newton Jacobian updates.

Factorization of a dense matrix A_k has an algebraic complexity of $O(nm^2)$ operations in general. The optimization algorithm requires solution of linear systems that involve the dense matrix A_k at every step. Instead of factorizing A_k at every optimization step, it is better to factorize A_k only once at the beginning of the optimization procedure and maintain a factorized null space representation Z^k of the approximated derivative information during the whole optimization procedure with just an $O(mn)$ cost of operations [34, 65].

4.3 Equality Constrained Optimization

The proposed SQP algorithm uses a trust-region globalization method to force convergence from distant starting points. Trust region SQP methods have several attractive properties because they control the quality of steps even in the presence of ill-conditioned Hessians and Jacobians, and do not require that the reduced Hessian matrix be positive definite. The use of the trust region constraint in (4.7): $||d|| \leq \Delta^k$ guarantees boundedness of the computed step d^k.

A globally convergent SQP algorithm usually defines a merit function, $\phi(x;\nu)$ which determines how accurately the model problem approximates the original problem at the new point $x^k + d^k$, and allows a decision to accept or reject the computed step d^k to be made. The proposed algorithm uses the non-differentiable l_2 merit function

$$\phi(x;\nu) = f(x) + \nu||c(x)|| \tag{4.10}$$

with the penalty parameter $\nu > 0$. The merit function is approximated at $x^k + d^k$ using

the same approximation that determines the model problem from the original problem. In our case, a model of ϕ around the current iterate x^k is given by

$$m^k(d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T B(x^k) d + \nu^k \parallel c(x^k) + A(x^k) d \parallel \tag{4.11}$$

Then with $\text{ared}^k(d)$ defined by

$$\text{ared}^{\text{k}}(d) = \phi(x^k; \nu^k) - \phi(x^k + d; \nu^k) \tag{4.12}$$

as the actual reduction in the merit function caused by a step d^k, and $\text{pred}^k(d)$ defined by,

$$\begin{aligned} \text{pred}^{\text{k}}(d) &= m^k(0) - m^k(d) \qquad (4.13) \\ &= \underbrace{-\nabla f(x^k)^T d - \frac{1}{2} d^T B(x^k) d}_{term1} + \underbrace{\nu^k(\parallel c(x^k) \parallel - \parallel c(x^k) + A(x^k) d \parallel)}_{term2} \end{aligned}$$

as the predicted reduction in the model of the merit function caused by a step d^k, the model is said to be accurate enough if

$$\frac{ared^k(d)}{pred^k(d)} \geq \eta, \text{where } 0 < \eta \leq 1 \tag{4.14}$$

Note that $\text{pred}^k(d)$ is positive for a nonzero step d^k and the sufficient decrease in merit function is obtained if condition (4.14) is satisfied.

To describe the approach we first focus on equality constrained optimization problems, i.e. we ignore bounds in (4.7). Computation of the step d^k requires the solution of an equality-constrained subproblem (ECSP). One potential difficulty that can arise when

solving ECSP is that the linearized constraint and the trust region constraint may not intersect if Δ^k is too small. Consequently, there will be no feasible region that satisfies both constraints and ECSP will not have a solution in the trust region. In order to overcome this difficulty, we follow the approach of Byrd and Omojokun [52] and decompose the ECSP into two trust-region subproblems. The two underbraced terms in (4.13) correspond to these trust-region subproblems. The first term is minimized in the *tangential subproblem* while the second term is minimized in the *normal subproblem*. This allows us to compute the composite step,

$$d^k = n^k + t^k,$$

a combination of the normal step, n^k and the tangential step, t^k. The normal sub-problem aims to attain feasibility of the constraints while the tangential sub-problem aims to reduce the quadratic objective function in ECSP by maintaining linearized feasibility. By splitting the step taken by ECSP, we write the resulting sub-problems as follows:

4.3.1 The Normal Subproblem

At each step of the algorithm we first solve the normal step n^k which reduces the infeasibility of the linearized constraints by solving the following *normal subproblem* at the current iterate x^k

$$\min_{n\in\mathbb{R}^n} \parallel c(x^k) + A(x^k)n \parallel^2, \qquad s.t. \quad \parallel n \parallel \leq \Delta_1^k \tag{4.15}$$

This subproblem has a spherical trust region in $\mathbb{R}^n$ with a quadratic objective function. The exact Cauchy step for (4.15) is given by

$$n^C(x^k) = -\alpha^C(x^k)A(x^k)^T c(x^k) \tag{4.16}$$

where α_k^C is the optimal solution of the problem

$$\begin{aligned} &\min_{\alpha \geq 0} \parallel c(x^k) - \alpha A(x^k)A(x^k)^T c(x^k) \parallel^2 \\ &s.t. \quad \parallel \alpha A(x^k)^T c(x^k) \parallel \leq \Delta_1^k \end{aligned} \tag{4.17}$$

Since we can evaluate $A(x^k)v$ and $A(x^k)^T w$ for given v and w exactly, we are able to compute the exact Cauchy step without evaluating the complete Jacobian, $A(x^k)$. However, using the Cauchy step results in slow convergence [51]. To accelerate the convergence rate, we also use the Newton step. The exact Newton step of (4.15) is given by

$$n^N(x^k) = -A(x^k)^+ c(x^k) = -A(x^k)^T (A(x^k)A(x^k)^T)^{-1} c(x^k) \tag{4.18}$$

However, it is expensive to compute $(A(x^k)A(x^k)^T)^{-1}c(x^k)$ exactly. Using the QR factorization and its orthonormal properties, an approximation

$$n^N(x^k) = -A(x^k)^T (A^k(A^k)^T)^{-1} c(x^k) \tag{4.19}$$

of the exact Newton step (4.18) can be computed easily. In combination with the exact Cauchy step, we define an inexact dogleg step by setting

$$n^D(x^k) = \tilde{\eta} n^N(x^k) + (1 - \tilde{\eta}) n^C(x^k) \tag{4.20}$$

with $\tilde{\eta} = 1$ if $\| n \| \leq \Delta_1^k$. Otherwise $\tilde{\eta} \in [0, 1]$ would be adjusted such that the length of n^D is equal to Δ_1^k. The choice n^k is governed by

$$n^k = \begin{cases} n^D(x^k) & \text{if } \mathrm{npred^k}(n^D(x^k)) \geq \gamma_n \mathrm{npred^k}(n^C(x^k)) \\ n^C(x^k) & \text{otherwise} \end{cases} \tag{4.21}$$

for some constant $\gamma_n > 0$ and the normal predicted reduction is given by,

$$\mathrm{npred^k}(n) = \| c(x^k) \| - \| c(x^k) + A(x^k)n \| \tag{4.22}$$

The normal step n^k provides sufficient normal Cauchy decrease (i.e., one may use the exact Cauchy step as a normal step) and also fulfills the range space condition i.e., the normal steps n^D, n^C and a linear combination of n^D and n^C are of the form $A^T(x^k)v^k$. Hence, working with the inexact constraint Jacobian, A^k does not hinder the computation of a reasonable normal step [74].

4.3.2 The Tangential Subproblem

Given the normal step, the tangential step is used to reduce a suitable model within the trust region by maintaining linearized infeasibility. The tangential step, $t^k = Z^k p_Z$ is obtained by solving the following subproblem

$$\begin{aligned} \min_{p_Z \in \mathbb{R}^{n-m}} \quad & (\nabla f(x^k) + B(x^k)n^k)^T Z^k p_Z + \frac{1}{2} p_Z^T (Z^k)^T B(x^k) Z^k p_Z \\ s.t. \quad & \| Z^k p_Z \| \leq \Delta_2^k \end{aligned} \tag{4.23}$$

This subproblem is a small NLP of size $(n - m)$ with a quadratic objective function and ellipsoidal trust-region constraint. The exact null space basis $Z(x^k)$ can be constructed from the exact Jacobian $A(x^k)$ such that $A(x^k)Z(x^k) = 0$. This can be done by performing QR factorization on $A(x^k)$, which may however be expensive for large-scale problems. The steepest descent direction for this problem is given by

$$p_Z^C(x^k) = -(Z^k)^T(\nabla f(x^k) + B(x^k)n^k) \tag{4.24}$$

However, the steepest descent step for the tangential problem is inexact due to the inexact null space term, Z^k.

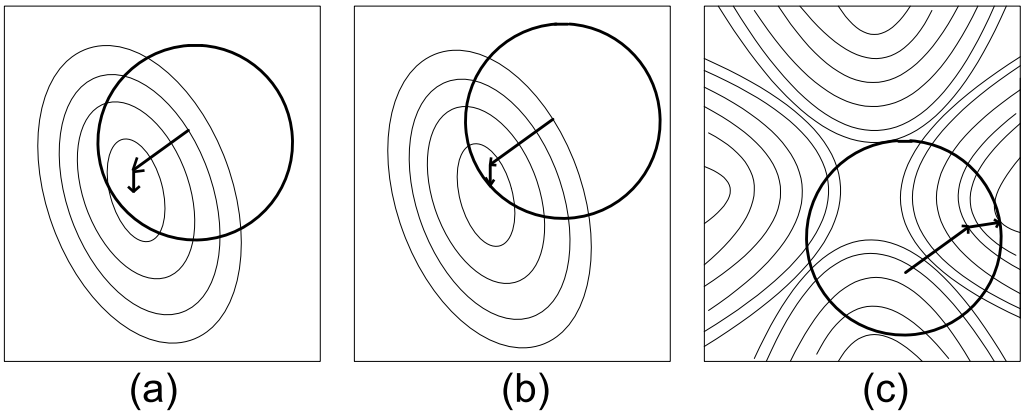

Figure 4.1: Steihaug CG algorithm: Handling convex/non-convex QP

Since the steepest descent results in only a linear convergence rate, we apply the Steihaug conjugate gradient (CG) algorithm [66] to accelerate the convergence rate. Three possibilities might arise for the solution of (4.23) which are handled efficiently in the Steihaug algorithm as shown in Figure 4.1. First, if $B(x_k)$ is positive definite and the trust region constraint is inactive, this algorithm reduces to a CG computation of the Newton step and converges to a convex interior solution (Figure 4.1(a)). Secondly, if negative curvature is encountered, the algorithm finds a useful point on the trust region

boundary (Figure 4.1(c)). Finally, if the model is convex and the solution lies outside the trust region, the algorithm terminates where the path crosses the trust region (Figure 4.1(b)). Steihaug proved that the objective function of the subproblem (4.23) decreases monotonically with each step and that each step moves farther away from the initial point, in the sense that $\| p_Z^{CG}(x^{j+1}) \| > \| p_Z^{CG}(x^j) \|$. These two properties imply that stopping the iteration as soon as the trust region is encountered is a sensible strategy. The Hessian-vector product in the Cauchy step (4.24) and the conjugate step computations is computed exactly using sensitivities and AD.

The composite trust-region step from (4.15) and (4.23) is:

$$d^k = n^k + Z^k p_Z \tag{4.25}$$

and we set

$$x^{k+1} = x^k + d^k$$

if we obtain a sufficient reduction (4.14) in the merit function (4.10). Otherwise, the trust region is reduced and a new trial step is computed.

The inexact algorithm represents a Byrd-Omojokun trust-region approach that takes the inexactness of the Jacobian and its null-space representation into account. The global convergence of the algorithm to first-order critical points is ensured by the theoretical results presented in [74]. In the inexact setting, the normal step (4.15) and the tangential step (4.23) are calculated based on the approximate Jacobian A^k and its null space Z^k (with $(A^k)^T Z^k = 0$). The matrix Z^k only approximates the null space $Z(x^k)$ of the exact Jacobian, $A(x^k)$. Therefore, in the inexact setting the tangential step may not maintain the feasibility of the combined step, $d^k = n^k + t^k$ with respect to the linearized constraints

and hence the identity $A(x^k)d^k = A(x^k)n^k$ is not necessarily valid. The resulting effects on the composite step, d^k are illustrated in Figure 4.2.

$Z(x^k)$ = exact null space $\qquad$ Z^k = inexact null space

$A(x^k)(n^k + t^k) = A(x^k)n^k$ $\qquad$ $A(x^k)(n^k + t^k) = A(x^k)n^k$+?

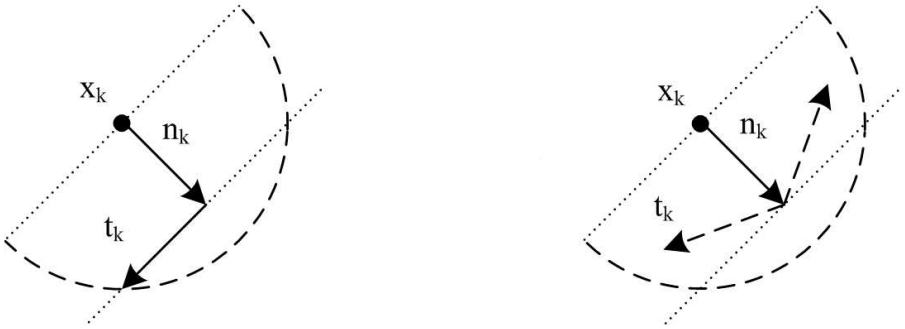

Figure 4.2: Computation of tangential step for exact and inexact null space representations

Hence, one has to safeguard the approximation Z^k by controlling the inexactness. The quality of the approximated constraint Jacobian can be adjusted by bounding the influence of the error in Z^k on the predicted reduction of the function m^k given by

$$\begin{aligned} \text{pred}^k(d^k) &= -\nabla f(x^k)^T(n^k + t^k) - \frac{1}{2}(n^k + t^k)^T B(x^k)(n^k + t^k) \\ &\quad +\nu^k(\| c(x^k) \| - \| c(x^k) + A(x^k)(n^k + t^k) \|) \\ &= \text{tpred}^k(t^k) + \nu^k \text{npred}^k(n^k) + \chi^k + \text{err}^k(d^k, \nu^k), \end{aligned}$$

where

$$\chi^k = -\nabla f(x^k)^T n^k - \frac{1}{2}(n^k)^T B(x^k)n^k$$

$$\mathrm{err}^k(d^k, \nu^k) = \nu^k(\parallel c(x^k) + A(x^k)n^k \parallel - \parallel c(x^k) + A(x^k)d^k \parallel).$$

To ensure that the trust-region algorithm is globally convergent, it has been shown in [74] that if

$$-\mathrm{err}^{\mathrm{k}}(d^k, \nu^k) < \left(\frac{1-\bar{\eta}}{2}\right) \mathrm{ipred}^{\mathrm{k}}(d^k) \tag{4.26}$$

where

$$\mathrm{ipred}^{\mathrm{k}}(d^k) = \mathrm{tpred}^{\mathrm{k}}(t^k) + \nu^k \mathrm{npred}^{\mathrm{k}}(n^k) + \chi^k$$

holds for a constant $\bar{\eta} \in (0, 1)$, then we have $\mathrm{pred}^k(d^k)$ non-negative as long as the inexact predicted reduction, $\mathrm{ipred}^k(d^k) \geq 0$. As can be seen, $err^k(d^k)$ is a measure of the error in Z^k and it becomes zero if we work with exact null space $Z(x^k)$. In the inexact setting, this inequality can be easily tested by evaluating two Jacobian-vector products during the optimization process. If it is not satisfied at the current step, the inexact null space Z^k needs to be updated and the normal step, n^k and the tangential step, t^k has to be recomputed. As a result, the inexact algorithm avoids forming an exact Jacobian at every iteration and hence reduces the computation time for optimization, as compared to the exact Byrd-Omojokun algorithm.

4.4 Treatment of Bounds

To handle bounds on the decision variables, x_1 in (4.7), we calculate the inexact null-space Z^k based on coordinate decomposition [10] instead of the QR decomposition. Here we partition the approximate constraint Jacobian into sub-matrices with m and $n - m$ columns, respectively, $A^k = [A_1 \mid A_2]$ and define $Y^k = [0 \mid I]^T$ and $Z^k = [I \mid -A_1^T(A_2)^{-T}]^T$. The coordinate basis is less expensive to compute and allows the normal

step to be calculated independently of the bound constraints.

In the presence of bounds, the inexact tangential problem now becomes,

$$\begin{aligned} \min_{p_Z \in \mathbb{R}^{n-m}} \quad & (\nabla f(x^k) + B(x^k)n^k)^T Z^k p_Z + \frac{1}{2} p_Z^T (Z^k)^T B(x^k) Z^k) p_Z \\ s.t. \quad & \| Z^k p_Z \| \leq \Delta_2^k \\ & x_1^L \leq x_1^k + p_Z \leq x_1^U \end{aligned} \tag{4.27}$$

Assuming inactive bounds, the first-order Karush-Kuhn-Tucker (KKT) conditions for (4.27) are:

$$((Z^k)^\top B(x^k) Z^k + \alpha^k I) p_Z = -(Z^k)^\top (\nabla f(x^k) + B(x^k) n^k), \tag{4.28}$$

$$\alpha^k = 0 \text{ if } ||Z^k p_Z|| < \Delta_2^k \text{ and } \alpha^k > 0 \text{ if } ||Z^k p_Z|| = \Delta_2^k. \tag{4.29}$$

Here α^k is the Marquardt parameter, which is just the Lagrange multiplier associated with the trust-region constraint. This algorithm provides an attractive way to deal with negative curvature. If the QP (4.27) is convex, the Newton step (4.28) may either lie within the trust-region ($\alpha^k = 0$) or on the boundary of the trust-region ($\alpha^k > 0$) and if the QP is nonconvex, the Newton step must lie on the trust-region boundary ($\alpha^k > 0$). In the case when α^k is non-zero, α^k is calculated by solving:

$$\frac{1}{||((Z^k)^\top B(x^k) Z^k + \alpha^k I)^{-1} ((Z^k)^\top (\nabla f^k + B(x^k) n^k))} - \frac{1}{\Delta_2^k} = 0, \tag{4.30}$$

This equation is solved by Newton's method. For a more detailed discussion on the solution to this problem, refer to [29]. The step p_Z calculated above may not satisfy the bounds. It is thus adjusted it to satisfy the bounds in a manner analogous to [30]. The

most violated bound is identified and made active. Then, a sub-space minimization of the objective function of (4.27) with respect to the remaining components of p_Z is carried out. After each such minimization, the most violated bound is identified, made active, and a sub-space minimization is carried out again until the final step lies within the bounds or exceeds the radius of the trust-region in length when the step calculated previously is sufficiently smaller than the trust-region radius. For a more detailed discussion on handling bounds, the interested reader is referred to [30, 5] whose codes have been used for the solution of (4.27). Similar to the equality constrained case, the accuracy of the tangential step is checked by verifying inequality (4.26).

4.5 Calculation of Lagrange multipliers

Lagrange multipliers are only used to evaluate the exact Hessian of the Lagrangian (4.8) times an arbitrary vector. An estimate of the Lagrange multipliers associated with the nonlinear equality constraints in (4.1) at every new iterate x_{k+1} is given by the solution of the linear system

$$(A^{k+1}Y^{k+1})^T\lambda^{k+1} = -(Y^{k+1})^T\nabla f^{k+1} \tag{4.31}$$

Since it is expensive to evaluate the exact Jacobian $A(x^{k+1})$, this linear system is solved using the factorization of the inexact Jacobian A^{k+1}.

In order to limit the inexactness of Z^{k+1} to a certain amount in the direction λ^{k+1} i.e. maintain sufficient quality of the Lagrange multipliers, we use the test from [75] given by

$$\| ((Z^{k+1})^T A(x^{k+1})^T\lambda^{k+1})_I \| \leq \omega \| ((Z^{k+1})^T\nabla f^{k+1})_I \| \quad \omega \in (0, \frac{1}{2}) \tag{4.32}$$

to check whether the approximation of the null space Z^{k+1} is good enough. We use the

subscript I to indicate subvectors with elements that correspond to x^{k+1} strictly between its bounds. As can be seen, this test is satisfied if we work with exact null space, $Z(x^k)$. In the inexact setting, this inequality can be easily verified by evaluating an exact vector-Jacobian product at every iteration. If this test fails at the current step, the inexact null space Z^{k+1} needs to be updated.

4.6 Update of Penalty Parameter and TR Radius

In this trust region algorithm, the goodness of the computed step d^k is measured by the l_2 merit function (4.10). A step d^k is accepted if $x^k + d^k$ yields a reasonable decrease in merit function. Then, the step d^k is accepted if

$$\frac{ared^k}{ipred^k} > \eta, \text{where } 0 \leq \eta \leq 1 \tag{4.33}$$

where is the $ared^k$ actual reduction in merit function from x^k to $x^k + d^k$ which is given by (4.12). Since this algorithm deals with inexact Jacobian A^k, the usual identity of the predicted reduction is not valid, we use the inexact predicted reduction (4.27) by omitting the error term in (4.13). $ipred^k$ is the inexact predicted reduction in the quadratic model which is a sum of two terms: the predicted reduction in the model for the tangential step (*tpred*) and the predicted reduction in the model for the normal step (*npred*).

In order to ensure that the inexact predicted reduction, $ipred^k$ due to a nonzero step d^k is positive and the penalty parameter in merit function (4.10) is monotonically increasing i.e. $\nu^{k+1} \geq \nu^k$ for all k, ν is updated based on the following rule:

$$\nu^{k+1} = \max\left[\nu^k, \frac{-\chi^k}{(1-\rho)npred^k}\right] \tag{4.34}$$

where ν^k is the penalty parameter at the previous iteration and $\rho \in (0, 1)$.

Also, it may become necessary to increase or decrease the radius of trust region at some iteration k. An increase in trust region radius may be beneficial if the current step d^k yields a sufficient reduction in the merit function and trust region constraint is nearly active i.e. η is close to unity. On the other hand, the trust region radius is decreased if the model subproblem does not model the original problem accurately enough i.e. η is close to zero. In this work, the expansion and contraction of the trust region is based on the updating rules in [5].

Furthermore, we need to adjust trust-region radius of both the subproblems to enforce $\|Z^k p_Z + n^k\| \leq \Delta^k$ where Δ^k is the size of the overall trust-region. For the solution of the overall problem (4.1), we use coordinate basis to compute Z^k and Y^k, the normal step n^k and the tangential step $Z^k p_Z$ are non-orthogonal. As a consequence, this may allow the normal step to significantly increase the objective function and hence the merit function (4.10). In order to allow the tangential step to cut back this significant increase in objective function, we impose $\Delta_2 = M\Delta_1, M > 2$.

4.7 Termination Criteria

The algorithm is terminated if the first order necessary conditions for problem (4.1) are satisfied. We define the following termination conditions depending on if bounds are present or not. If bounds are absent in constrained problems or they are inactive at the solution, then the problem is said to have converged if

$$\max\{||(Z^k)^\top \nabla f_k||_2, ||c_k||_2\} \leq \epsilon \tag{4.35}$$

and for constrained problems with bounds, the convergence criteria is as follows:

$$((Z^k)^\top \nabla f^k)_i = \begin{cases} \geq 0 & \text{if } x^i = x^L \\ \leq 0 & \text{if } x^i = x^U \quad \text{and} \quad \|c^k\|_2 \leq \epsilon \\ = 0 & \text{otherwise} \end{cases} \tag{4.36}$$

where $0 < \epsilon << 1$ is some tolerance defined by the user and $(Z^k)^T \nabla f^k$ is the reduced gradient at iteration k.

4.8 Overall Optimization Algorithm

The previous discussion has presented the Byrd-Omojokun algorithm to solve general nonlinear programming problems of the form (4.1) with inexactness in Jacobian accounted for. Below is the step by step description of the overall algorithm:

Composite step trust region algorithm with inexact Jacobians:

Start: Set initial values $x^0, \lambda^0, \nu^{-1}, A^0, Z^0, \Delta^0, \vartheta \in (0,1), \eta \in (0,1), \epsilon > 0$ and $\omega \in (0, \frac{1}{2})$.

for $k = 0, 1...$

1. Compute a normal step, n^k according to (4.15).
2. Compute a tangential step, t^k according to (4.27) and the total step, $d^k = n^k + t^k$.
3. Compute the smallest value $\tilde{\nu}$ such that

$$\text{ipred}^k(d^k) = \text{tpred}^k(t^k) + \tilde{\nu}\text{npred}^k(n^k) + \chi^k \geq \vartheta\tilde{\nu}\text{npred}^k(n^k).$$

If $\tilde{\nu} \leq \nu^{k-1}$, set $\nu^k = \nu^{k-1}$. Otherwise set $\nu^k = \max\{\tilde{\nu}, 1.5\nu^{k-1}\}$.

4. If (4.26) is not fulfilled, update A^k and Z^k and go to step 1.

5. If

$$\text{ared}^{\text{k}}(d^k) < \eta \, \text{ipred}^{\text{k}}(d^k) \tag{4.37}$$

decrease Δ^k by a constant factor and go to step 1.

6. Set $x^{k+1} = x^k + d^k$ and choose Δ^{k+1} such that $\Delta^{k+1} \geq \Delta^k$

7. Update A^{k+1}, Z^{k+1}, and compute Lagrange multipliers according to (4.31)

8. If (4.32) is not fulfilled, update A^k and Z^k and go to step 7.

9. If the KKT conditions are satisfied to a given tolerance (Section 4.7), stop. Else set $k = k + 1$.

As can be seen, this algorithm reduces to the algorithm in [5] if an exact Jacobian and its null space is computed at every iteration.

4.9 Implementation and Numerical Results

In order to demonstrate the performance of the algorithm, we present and analyze numerous numerical results in this section. We performed numerical tests on dynamic optimization problems related to PAPs, namely the simulated moving bed (SMB) and vacuum swing adsorption (VSA) systems.

The nonlinear programming algorithm described in the previous section has been implemented in C/C++. In all of the results presented below, the Jacobian A^0 at the initial guess x^0 is computed exactly. Also, if either of the tests (4.26) or (4.32) fails, we update A^k and Z^k by generating the exact Jacobian $A(x^k)$ and an exact null space representation

$Z(x^k)$ and then we set $A^k = A(x^k), Z^k = Z(x^k)$. Routines from LAPACK are used to generate the LU factors and the corresponding orthonormal factors are calculated based on QR factorization. All the numerical experiments are performed on a Pentium IV/2.80 GHz machine with 1 GB memory.

To judge the performance of the algorithm, optimization was also performed with an exact Jacobian computed at each optimization iteration. All the runs were also performed using the reduced Hessian SQP approach [10] as in [38, 39]. This rSQP code is based on line-search globalization strategy and works with approximate second-order information using BFGS quasi-Newton method. For the inexact approach, we report the number of failures of the criteria (4.26) and (4.32). The total number of Jacobian evaluations of the inexact algorithm is equal to the number of failures due to (4.26) in Step 4 and (4.32) in Step 8, plus the one due to initialization with the exact Jacobian.

4.9.1 Dynamic Optimization Problems

We consider the dynamic optimization problems of two periodic adsorption processes, namely, a simple six bed simulated moving bed process and a single bed 3-step O_2 VSA cycle. In this work, we solve the dynamic optimization problems using the sequential approach shown in Figure 2.1. In this approach, the function evaluation requires a forward simulation of the differential equations and the necessary derivative information in the form of exact Jacobian-vector and Hessian-vector products (and any Jacobian required by the NLP algorithm) is computed using sensitivity calculations. The continuous derivative formulation uses the Automatic Differentiation (AD) tool, ADOL-C [32] to construct the relevant system of equations required by the integration subroutines and we exploit the efficient integration tool CVODES [59] to integrate the state and sensitivity systems.

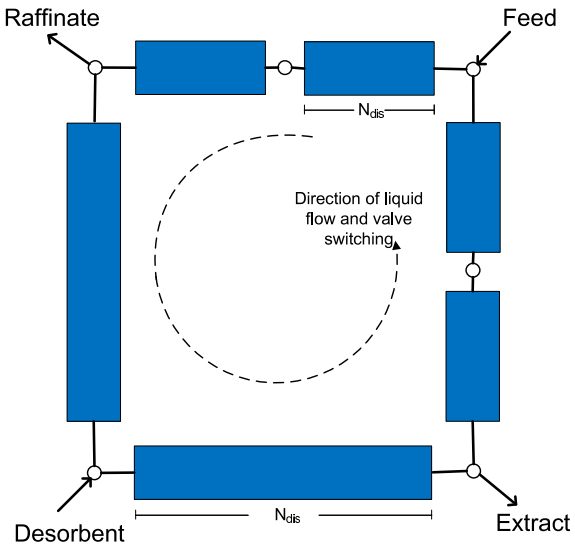

Figure 4.3: SMB Process

4.9.1.1 Case Study 1: SMB Process

An SMB system consists of multiple columns connected to each other, making a circulation loop (Fig. 4.3). SMB system consists of four zones, each fulfilling different functions. The feed and desorbent are supplied continuously through the inlet ports and at the same time extract and raffinate are withdrawn at the outlet ports. These six inlet/outlet ports are switched in the direction of liquid flow at a regular interval, T. The same switching operation is repeated for $N_{col} = 6$ steps which constitutes a cycle. Since SMB repeats the same operation for all columns, this symmetric operation can be exploited to reduce the problem size by a single-step optimization formulation [25, 41]. Here, operation is considered over only one step where the profiles at the beginning of a step are identical to those of the downstream adjacent column at the end of the step. This ensures a CSS condition at the end of a cycle. In this case study, we solve the small problem considered in [25, 74].

The behavior of the chromatographic columns, identified by index $n = 1 \dots N_{col}$ is

described through an equilibrium assumption between the solid and liquid phases along with a simple spatial discretization. Here, the mass balance in the liquid phase over time t and space l is given by,

$$\epsilon_B \frac{\partial C_{n,i}(l,t)}{\partial t} + (1-\epsilon_B)\frac{\partial q_{n,i}(l,t)}{\partial t} + \frac{Q_i(t)}{S_i}\frac{\partial C_{n,i}(l,t)}{\partial l} = 0 \qquad i = A, B \tag{4.38}$$

The equilibrium relationship between the solid and liquid phases is given by,

$$\frac{\partial q_{n,i}(l,t)}{\partial t} = K_i(C_{n,i})C_{n,i}(l,t) \tag{4.39}$$

where $C_{n,i}$ is the concentration of component i in column n in the liquid phase; $q_{n,i}$ is the concentration of component i in column n in the solid phase; Q_n is the volumetric flow rate in each zone n; S is the cross-sectional area of the bed and ϵ_B is the bed voidage. We assume that the zone velocities are constant during a step and all the inlet/outlet ports are switched simultaneously. Dividing the column into N_{dis} compartments and applying a simple backward difference results in,

$$\frac{dC_{n,i,j}}{dt} = K_i(C_{n,i,j})N_{dis}[C_{n,i,j-1} - C_{n,i,j}] \tag{4.40}$$

for $j = 1 \ldots N_{dis}$; $i = A, B$ (binary mixture); $n = 1 \ldots, N_{col}$. We define the state variables for this system, $C_{n,i,j} = z_m(t), m = 1 \ldots 12N_{dis}$ as the concentrations of A and B in the j^{th} compartment for the six columns where the index is ordered as: $m = j + (n-1)N_{dis}$ for component A and $m = j + 6N_{dis} + (n-1)N_{dis}$ for component B. After spatial discretization, the dynamic SMB model along with cyclic steady state conditions can be

described by the following DAEs,

$$\begin{aligned} &\frac{dz_m}{dt} = g_m(z_m, t) \quad m = 1, ...6N_{dis} \\ &z_m(0) - z_{m+N_{dis}}(T) = 0 \quad m = 1, ...5N_{dis} \\ &z_m(0) - z_{m-5N_{dis}}(T) = 0 \quad m = 5N_{dis}+1, ...6N_{dis} \end{aligned} \tag{4.41}$$

The port switching time, T and the constant flows, $[Q_I, Q_{De}, Q_{Ex}, Q_{Fe}]$ as independent decision variables while the remaining flows $Q_{II}, Q_{III}, Q_{IV}, Q_{Ra}$ are determined from a linear mass balance. $z_m(T)$ is obtained from the forward simulation of the ODE system in (4.41) with $12N_{dis}$ states. In this case study, we consider the adsorption isotherm

$$K_A = \frac{4(1+\bar{\alpha}C_A^2)}{1+(1+\bar{\alpha}C_A)^2}, \quad K_B = \frac{2(1+\bar{\alpha}C_B^2)}{1+(1+\bar{\alpha}C_B)^2}$$

where $\bar{\alpha} = 0$ reduces the nonlinear isotherm to a linear isotherm.

Equality Constrained Case

The following equality constrained problem is considered in order to determine the decision variables that lead to the desired profiles,

$$\min_{z_m(0),q,T} \quad f = \sum_{i=1}^{12N_{dis}} (z_m(0) - \bar{z}_m(0))^2 \qquad s.t. \quad \text{CSS constraints in (4.41)} \tag{4.42}$$

The data $\bar{z}_m(0)$ are from the optimal solution obtained by solving a maximum throughput problem with 95% purities of A and B in the raffinate and extract streams, respectively. The problem has $12N_{dis}$+5 variables and $12N_{dis}$ equality constraints. For all the numerical tests, we choose the initial state to be equal to zero and the termination criteria set to

$|f| < 10^{-5}$ and $\| c(x) \| < 10^{-5}$. For this objective function, it can be seen that these criteria are compatible with the KKT conditions in step 9.

We present optimization results using the orthogonal (QR) and coordinate basis for the linear ($\bar{\alpha} = 0$) and nonlinear isotherm with $\bar{\alpha} = 1$. Firstly, we show the results in Table 4.1 in which the derivatives are obtained by the "discretize then differentiate" strategy. Table 4.1 show the corresponding iteration counts and the number of exact Jacobian evaluations for the inexact and exact approaches and also the number of failures in (4.26) and (4.32) for the inexact case. In this case, an explicit Runge-Kutta method is used to discretize the state equations and Automatic Differentiation is applied to the discretized equations to obtain the discrete derivative information.

We then compare the above results with those in Tables 4.2 and 4.3 obtained by the "differentiate then discretize" strategy as described in Chapter 3. Table 4.2 and 4.3 show the corresponding iteration counts and the number of exact Jacobian evaluations for the inexact, exact and rSQP approaches and also the number of failures in (4.26) and (4.32) for the inexact case. We observe that the number of iterations and the number of Jacobian

Table 4.1: Optimization results for equality constrained SMB problem with orthogonal null space: discrete derivatives

						evaluation of full $A(x_k)$			
						inexact			exact
				iteration count		fail. in	fail. in		
N_{dis}	$\bar{\alpha}$	n	m	inexact	exact	(4.26)	(4.32)	total	total
10	0	125	120	19	15	7	2	10	13
20	0	245	240	24	11	13	2	16	12
30	0	365	360	11	11	4	3	8	11
40	0	485	480	9	15	0	3	4	12
10	1	125	120	5	15	0	3	4	14
20	1	245	240	18	5	1	9	11	6
30	1	365	360	7	21	0	5	6	17
40	1	485	480	13	21	5	5	11	19

Table 4.2: Optimization results for equality constrained SMB problem with orthogonal null space

							evaluation of full $A(x^k)$				
								inexact		exact	rSQP
				iteration count (CPU min.)			fail. in	fail. in			
N_{dis}	$\bar{\alpha}$	n	m	inexact	exact	rSQP	(4.26)	(4.32)	total	total	total
10	0	125	120	7 (0.7)	6 (1.2)	28 (4.4)	0	3	**4**	**7**	**28**
20	0	245	240	6 (6.3)	6 (7.5)	23 (27)	1	3	**5**	**7**	**23**
30	0	365	360	8 (15)	7 (29.3)	33 (122.2)	0	3	**4**	**8**	**33**
40	0	485	480	7 (28)	6 (67.5)	35 (314.5)	0	2	**3**	**7**	**35**
10	1	125	120	8 (2.1)	6 (1.7)	27 (8.6)	0	3	**4**	**7**	**27**
20	1	245	240	7 (14)	6 (12.4)	32 (58.7)	0	4	**5**	**7**	**32**
30	1	365	360	9 (24.1)	6 (29.3)	32 (131.3)	1	3	**5**	**7**	**32**
40	1	485	480	7 (37)	7 (70.5)	34 (295.8)	1	2	**4**	**8**	**34**

Table 4.3: Optimization results for equality constrained SMB problem with coordinate basis

							evaluation of full $A(x^k)$				
								inexact		exact	rSQP
				iteration count (CPU min.)			fail. in	fail. in			
N_{dis}	$\bar{\alpha}$	n	m	inexact	exact	rSQP	(4.26)	(4.32)	total	total	total
10	0	125	120	6 (0.54)	6 (0.86)	28 (4.4)	0	4	**5**	**7**	**28**
20	0	245	240	8 (7.2)	7 (8.5)	23 (27)	0	5	**6**	**8**	**23**
30	0	365	360	7 (44)	7 (57.6)	33 (122.2)	0	5	**6**	**8**	**33**
40	0	485	480	8 (64)	8 (82)	35 (314.5)	0	6	**7**	**9**	**35**
10	1	125	120	6 (1.8)	6 (2.5)	27 (8.6)	0	4	**5**	**7**	**27**
20	1	245	240	7 (8.2)	7 (15.4)	32 (58.7)	0	4	**5**	**8**	**32**
30	1	365	360	8 (25.1)	7 (36)	32 (131.3)	0	5	**6**	**8**	**32**
40	1	485	480	7 (41)	7 (65.4)	34 (295.8)	0	5	**6**	**8**	**34**

evaluations in "differentiate then discretize" are fewer compared to the "discretize then differentiate" strategy.

As can be seen in Tables 4.2 and 4.3, the inexact and exact methods take far fewer iterations than rSQP and the dominant computational cost (shown in bold) is significantly lower. The inexact trust region approach is at least 3 times faster than rSQP and up to 2.4 times faster than the exact approach. This performance improvement is due to providing exact second order information in the proposed algorithm, as opposed to the approximate second order information provided by BFGS [51] updates in rSQP. On the other hand, rSQP computes an exact Jacobian at every NLP iteration and hence, takes many more

Jacobian evaluations than the inexact and exact methods. Even though the inexact method takes almost the same number of iterations as the exact method in most of the cases, the number of exact Jacobian evaluations, which is the dominant computational cost during the entire optimization process, is less than with the exact method. Hence we obtain significant savings in CPU time over the exact and rSQP methods. Also, the algorithm converges to local solution even from poor starting points i.e. with an initial guess far away from the optimum.

Bound Constrained Case

We now solve the following optimization problem in order to maximize the feed throughput subject to specifications on the purities of species A and B in the raffinate and extract streams, respectively.

$$\begin{aligned} \max_{z_m(0),q,T} \quad & Q_{Fe} \\ \text{s.t.} \quad & (\tfrac{dz_m}{dt} = g_m(z_m,t) \quad m = 1,...6N_{dis}, \text{ solved implicitly.}) \\ & z_m(0) - z_{m+N_{dis}}(T) = 0 \quad m = 1,...5N_{dis} \\ & z_m(0) - z_{m-5N_{dis}}(T) = 0 \quad m = 5N_{dis}+1,...6N_{dis} \\ & \text{Purity}_{Ra} \geq 0.95 \\ & \text{Purity}_{Ex} \geq 0.95 \\ & 0 \leq q \leq Q_{max} \end{aligned} \tag{4.43}$$

The problem has $12N_{dis} + 8$ variables, $12N_{dis}$ equality constraints, two inequality constraints and upper and lower bounds on feed flow-rates in each section of SMB. In this case study, we set the bound on flow rate $Q_{max} = 2$. In order to avoid concentrations

Table 4.4: Optimization results for general constrained SMB problem with coordinate basis

							evaluation of full $A(x^k)$				
								inexact		exact	rSQP
				iteration count (CPU min.)			fail. in	fail. in			
N_{dis}	$\bar{\alpha}$	n	m	inexact	exact	rSQP	(4.26)	(4.32)	total	total	total
10	0	128	123	15 (1.8)	12 (2.75)	22 (6.5)	2	8	**11**	**13**	**22**
20	0	248	243	17 (14.4)	24 (28.5)	26 (34.3)	0	11	**12**	**25**	**26**
30	0	368	363	15 (32.8)	14 (61.5)	30 (127.7)	0	9	**10**	**15**	**30**
10	0.1	128	123	18 (1.2)	20 (2)	25 (5.3)	0	13	**14**	**21**	**25**
10	1	128	123	14 (2)	10 (1.2)	27 (12)	0	10	**11**	**11**	**27**
20	0.1	248	243	16 (14)	18 (22.1)	30 (33.4)	0	9	**10**	**19**	**30**
20	1	248	243	15 (26)	20 (38.5)	33 (56.3)	0	11	**12**	**21**	**33**

becoming negative, which would result in failure of the state and sensitivity integration, we lump all the constraints into one inequality constraint using the KS function (4.2). To accommodate the non-negative bounds on slack variables resulting from the two purity constraints and the KS function, we move the feed flow rate, Q_{Fe}, extract flow rate, Q_{Ex} and the step time, T into the set of dependent variables. This choice has been made by solving the design problem by fixing the independent variables as described in Section 4.1.1

Table 4.4 presents results of the general constrained case with both the linear and nonlinear isotherm. As in the equality constrained case, the inexact and exact methods take fewer iterations compared to rSQP and the inexact method requires fewer Jacobian evaluations to converge to the same solution. Here the dominant computational cost (shown in bold) is significantly lower. In the table we see that the inexact trust region approach is up to six times as fast as rSQP and up to twice as fast as the exact approach.

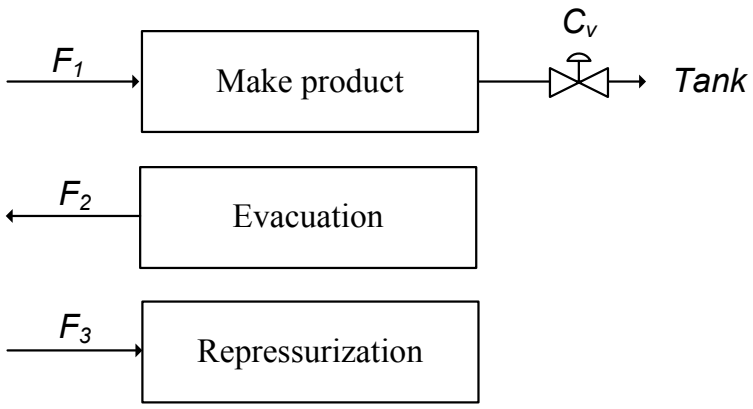

Figure 4.4: VSA Process

4.9.1.2 Case Study 2: VSA Process

In this case study, we consider a single-bed three step non-isothermal cycle as shown in Figure 4.4. This process is used to separate O_2 from air using a zeolite molecular sieve as the adsorbent. The cycle is operated in three steps: (1) adsorption (2) evacuation and (3) re-pressurization. In contrast to the SMB case study, in this example the PDAE model is given by the following equations:

Component mass balance

$$\epsilon_B \frac{\partial \rho_i}{\partial t} + \rho_s \frac{\partial n_i}{\partial t} + \frac{\partial (\rho_i \nu)}{\partial l} = 0 \qquad i = 1, 2, ..., N_c$$

Linear driving-force equation

$$\frac{\partial n_i}{\partial t} = \frac{k_i}{R_{PV} T} (P_i - P_i^*) \qquad i = 1, 2, ..., N_c$$

Energy balance

$$\left(\epsilon_B \sum_{i=1}^{N_c} \rho_i (c_p^i - R_E) + \rho_s c_s + \rho_s \sum_{i=1}^{N_c} n_i c_p^i \right) \frac{\partial T}{\partial t} - \rho_s \sum_{i=1}^{N_c} q_i \frac{\partial n_i}{\partial t} + \frac{\partial (\nu h)}{\partial l} = 0$$

Steady-state momentum balance (Ergun equation)

$$-\frac{\partial P}{\partial l} = 150\frac{\mu\nu}{d_p^2}\frac{(1-\epsilon_B)^2}{\epsilon_B^3} + 1.75\frac{\rho M\nu^2}{d_p}\frac{1-\epsilon_B}{\epsilon_B^3}$$

Adsorption isotherm

$$n_i = m_1\frac{b_i P_i^*}{1+\sum_{i=1}^{N_c} b_j P_j^*} \qquad i = 1, 2, ..., N_c$$

Ideal gas law

$$\rho = \frac{P}{R_{PV}T}$$

Enthalpy

$$h = \sum_{i=1}^{N_c} \rho_i (aT + bT^2 + cT^3 + dT^4)$$

Molecular weight

$$M = \sum_{i=1}^{N_c} y_i M_{0,i} \quad \text{where} \quad y_i = \rho_i/\rho$$

Fluid viscosity

$$\mu = \sum_{i=1}^{N_c} \mu_i y_i \quad \text{where} \quad \mu_i = \mu_{0,i} + \mu_{1,i}T$$

Here, P_i is the component partial pressure in the gas phase; P_i^* is the component equilibrium partial pressure in the gas phase; n_i is the solid phase loading; n_i^* is the component equilibrium loading; y_i is the mole fraction of the component in gas phase; ρ_i is the component molar density; k_i is the component mass transfer coefficient; c_p^i is the component heat capacity; T is the temperature; P is the total pressure in the bed; h is the mixture enthalpy; M is the molecular weight; b is the affinity parameter of the adsorbent; ν is the

superficial velocity of the gas; μ is the viscosity of the gas; c_s is the heat capacity of the adsorbent; d_p is the particle diameter; m_i is the saturation capacity of the adsorbent; q_i is the component heat of adsorption; ϵ_B is the inter-particle void volume of the adsorbent; ϵ_T is the total void volume of the adsorbent; ρ_s is the bulk density of the adsorbent. The equation for flow through the valve is given by

$$\nu = \begin{cases} CvP_H\sqrt{\frac{1-(P_L/P_H)^2}{MT}} & \text{if } P_H/P_L < P_{crit} \\ CvP_H\sqrt{\frac{1-P_{crit}^{-2}}{MT}} & \text{if otherwise} \end{cases} \tag{4.44}$$

Here, ν is the flow rate, Cv is the valve constant, M is the molecular weight of the gas mixture, γ is the ratio of constant pressure and constant volume heat capacities (e.g. $\gamma = 1.4$ for ideal gas, P_H is the inlet pressure, P_L is the outlet pressure from the valve and $P_{crit} = (\frac{2}{1+\gamma})^{\frac{\gamma}{1-\gamma}}$ is the critical pressure ratio.

We use the method of lines to spatially discretize the PDAEs into a set of differential algebraic equations. Because first and second-order finite difference or finite-element methods in space introduce oscillations near the steep adsorption fronts and also do not preserve the mass and energy balance in the spatial direction, we apply a high resolution finite-volume method along with the flux limiter scheme outlined in [38]. The resulting DAEs are then integrated using an ODE solver (CVODES) with the algebraic equations solved using a Newton solver in an inner loop. In addition, due to non-isothermal and steep adsorption fronts, the bed model can be so nonlinear and ill-conditioned that the integration may fail. In this work, we decide by inspection which variables need to rescaled to avoid these failures.

The adsorption bed is discretized into N_{dis} nodes and each node has five differential equations, which are densities and loadings for two components and one temperature

Table 4.5: Boundary conditions for the PSA cycle

Step	Beginning of bed	End of bed
Step 1	Constant flow	Determined by valve equation
Step 2	Constant flow (reverse direction)	Velocity is zero
Step 3	Constant flow	Velocity is zero

variable. Therefore, the number of state variables for the DAE model is $5N_{dis}$. The boundary conditions are given in Table 4.5. The data for the physical parameters are taken from Jiang et al. [38].

In this case study, we solve the following general constrained problem to maximize O_2 recovery subject to the cyclic steady state constraints and design specification on the purity. Here, we adjust the step times of the evacuation and re-pressurization steps (T_2, T_3) with the valve constant (Cv) as the dependent variable to accommodate the purity constraint.

$$\begin{aligned} \max_{C_v, T_2, T_3} \quad & O_2 \text{ Recovery} \\ \text{s.t.} \quad & (\tfrac{dz_m}{dt} = g_m(z_m, t) \quad m = 1, ...5N_{dis}, \text{ solved implicitly.}) \\ & z_m(0) - z_{N_{dis}}(T) = 0 \quad m = 1, ...5N_{dis} \\ & O_2 \text{ Purity} = 0.35 \\ & 0 \leq T_2 \leq 100 \\ & 0 \leq T_3 \leq 150 \end{aligned} \tag{4.45}$$

A better assessment of the algorithm's performance can be clearly seen in this case study (Table 4.6). The total number of iterations are significantly reduced with the proposed inexact algorithm. Furthermore, the savings in the number of times an exact

Table 4.6: Optimization results for the VSA problem with coordinate basis

						evaluation of full $A(x^k)$				
						inexact			exact	rSQP
			iteration count (CPU time, min)			fail. in	fail. in			
N_{dis}	n	m	inexact	exact	rSQP	(4.26)	(4.32)	total	total	total
10	53	51	11 (8.7)	16 (21.5)	46 (55.2)	3	1	**5**	**14**	**46**
20	123	121	12 (42)	24 (127)	32 (213)	2	2	**5**	**21**	**32**
30	153	151	22 (212)	24 (541)	53 (1227)	1	8	**10**	**25**	**53**

Table 4.7: Optimization variables and objective at the solution of the VSA problem

	$T_2(s)/T_3(s)/Cv/f$		
N_{dis}	inexact	exact	rSQP
10	100/86.17/1079.516/0.912	100/86.18/1079.516/0.912	100/86.22/1079.83/0.912
20	100/106.738/869.685/0.9118	100/106.738/869.685/0.9118	100/106.740/869.72/0.9118
30	100/98.28/912.746/0.9124	100/98.28/912.735/0.9124	100/98.26/912.786/0.9124

Jacobian is computed, which is the dominant computational cost (shown in bold), are also significant. Here the inexact trust region approach is at least 5 times faster than rSQP and up to 3 times as fast as the exact approach.

Figure 4.5 shows the O_2 mole fraction profiles of the adsorption step at the optimum of the inexact method for $N_{dis} = 10$. The O_2 mole fraction profiles of the exact and the rSQP method are very similar to the profiles of the inexact method which means that all three methods converged to essentially the same solution for a given value of N_{dis}. This can also be seen from the results in Table 4.7, where the objective function achieves a 91.2 % O_2 recovery at the required purity for all cases in Table 4.6.

4.10 Concluding Remarks

In this chapter, we focus on the solution of dynamic optimization problems related to periodic adsorption processes. Because these result in dense constraint Jacobians, the time required for the computation of the Jacobian and its factorization dominates the

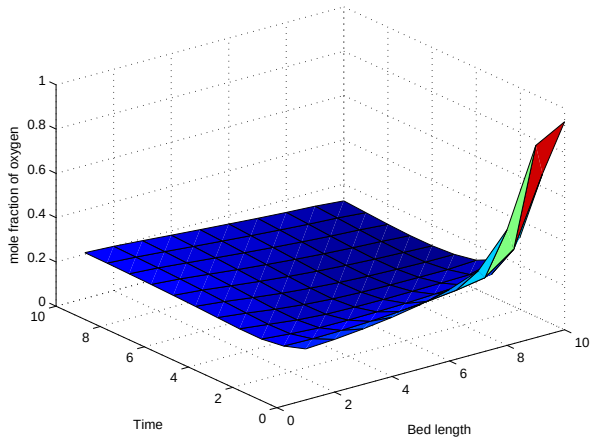

Figure 4.5: O_2 mole fraction profiles of the adsorption step at the solution

overall optimization process. To avoid these dense calculations, we developed and implemented an inexact trust region SQP algorithm based on Byrd-Omojokun approach that approximates the constraint Jacobian using TR1 updates. Furthermore, the specialized algorithm works with exact first and second order derivative information in the form of exact Jacobian-vector, vector-Jacobian and Hessian-vector which are efficiently calculated using sensitivities and Automatic Differentiation.

The performance of the algorithm is tested on equality and bound constrained optimization problems for simulated moving bed and vacuum swing adsorption systems. In comparisons with rSQP and exact trust region methods, the inexact method generally requires far fewer Jacobian evaluations. Since Jacobian evaluation is the dominant computational cost, we obtain significant savings in CPU time.

The proposed algorithm has a limitation of handling general inequality constraints i.e. only $(n - m)$ variables can have bounds. In order to overcome this drawback, the next chapter focuses on handling inequality constraints using an interior point trust-region method.

Chapter 5

Interior Point Trust-Region Algorithm with Inexact Jacobians

Synopsis

In this chapter, a class of trust-region algorithms is developed and analyzed for the solution of minimization problems with nonlinear inequality constraints. Based on composite-step trust region methods with barrier functions, the resulting algorithm also does not require the computation of exact Jacobians; only Jacobian vector products are used along with approximate Jacobian matrices. As demonstrated on small numerical examples, this feature has significant potential benefits for problems where Jacobian calculations are expensive.

5.1 Introduction

We consider an inequality constrained nonlinear program given by:

$$\min_{x \in \mathbb{R}^N} f(x) \qquad \text{subject to } \; g(x) \leq 0, \tag{5.1}$$

where the objective $f : \mathbb{R}^N \to \mathbb{R}$ and the vector of the constraints $g : \mathbb{R}^N \to \mathbb{R}^M$ with $N \geq M$ are given smooth functions. Following the strategy of interior point methods, we introduce nonnegative slack variables $s \in \mathbb{R}^M$ yielding the modified constraint $g(x) + s = 0, s \geq 0$, and form the barrier problem from (5.1) in x and s given by

$$\begin{aligned} &\min_{x \in \mathbb{R}^N} f(x) - \mu \sum_{i=1}^{M} \ln s^i \quad \text{subject to} \\ &g(x) + s = 0 \quad (s > 0) \end{aligned} \tag{5.2}$$

with $s \in \mathbb{R}^M$ implicitly assumed to be positive and a barrier parameter $\mu > 0$.

We define the constraint Jacobians as

$$A(x) = (\nabla g_1(x), \ldots, \nabla g_M(x))^\top \in \mathbb{R}^{M \times N}$$

and allow these matrices $A(x)$ to be approximated in the algorithm. Hence, the algorithm that we propose, allows inexactness in the Jacobian itself. Note that the constraint Jacobian may be dense, i.e., we do not assume that $A(x)$ has any structure. There are several papers dealing with inexactness in the equality-constrained case, see, e.g., [35, 74, 72]. However, in the case of inequality constraints only a few results are available, see, e.g., [24, 72]. Curtis et. al. [24] focused on inexactness arising due to iterative system solves. They assume that the Jacobian can be obtained at low cost and focus

only on the efficient solution of the linear system. More recently, Vetukuri et al. [72] developed an inexact SQP algorithm for solving general optimization problems which avoids forming and factoring the dense Jacobian by working with exact Jacobian-vector and Hessian-vector products. A five-fold reduction in computation is demonstrated with our algorithm for two PAP optimization problems. However, the algorithm can handle bounds only on $(N - M)$ optimization variables.

To begin, we note that the first order optimality condition for the barrier problem (5.2) are given by:

$$\begin{aligned}
\mathcal{L}(x, s, \lambda) &= f(x) - \mu \sum_{i=1}^{M} \ln s^i + \lambda(g(x) + s) \\
\nabla_x \mathcal{L}(x, s, \lambda) &= \nabla f(x) + A(x)^\top \lambda = 0 \\
\nabla_s \mathcal{L}(x, s, \lambda) &= -\mu S^{-1} e + \lambda = 0 \\
\nabla_\lambda \mathcal{L}(x, s, \lambda) &= g(x) + s = 0
\end{aligned}$$

where

$$S = diag(s_1, \ldots, s_M) \in \mathbb{R}^{M \times M} \quad \text{and} \quad e = (1, \ldots, 1)^\top \in \mathbb{R}^M.$$

As can be seen from (5.2), the solution of this barrier problem coincides with a solution of the original problem (5.1) if $\mu = 0$. Therefore, we compute a solution of the original optimization problem (5.1) in a nested way. An outer algorithm reduces the barrier parameter μ step by step in an appropriate way to ensure convergence as discussed later in Sec. 5.6. An inner algorithm is used to compute a solution for the current barrier problem. This will be done by a composite-step trust-region method analogous to the equality constrained algorithm in Chapter 4, the convergence of which is the subject of

the Secs. 5.3–5.5. Hence, the overall layout of the proposed solution procedure is given by

Algorithm 5.1.1.

1. *Outer iteration: Stopping criterion*

   If (5.1) is solved to the required accuracy, stop.

2. *Outer iteration: Solution of barrier problem*

   (a) *Initialization:* $\Delta_0, \xi \in (0,1), \delta > 0$

   (b) *Inner iteration:*

      i. *Stopping criterion: If* (5.2) *is solved to the required accuracy, stop.*

      ii. *Computation of normal step*

      iii. *Computation of tangential step*

      iv. *Tests on inexactness and termination*

3. *Outer iteration: Reduction of barrier parameter* μ*, go to 2*

5.2 Notation and Assumptions

For the analysis of the inner iteration, we define the following terms:

$$z = \begin{pmatrix} x \\ \\ s \end{pmatrix}, \quad \varphi(z) = f(x) - \mu \sum_{i=1}^{M} \ln s^i, \quad c(z) = g(x) + s, \quad (\text{with } s > 0).$$

Then the barrier problem (5.2) becomes:

$$\min_{z \in \mathbb{R}^{N+M}} \varphi(z) \quad \text{subject to } c(z) = 0 \tag{5.3}$$

with $z^\top = [x^\top s^\top]$. To solve the barrier problem we apply an SQP trust-region algorithm and approximately solve the trust region subproblem at the kth iteration:

$$\begin{aligned} \min_{d \in \mathbb{R}^{N+M}} \quad & \nabla\varphi(z_k)^\top d + \tfrac{1}{2} d^\top \begin{bmatrix} B_k & 0 \\ 0 & \mu S_k^2 \end{bmatrix} d \\ \text{subject to} \quad & g(x_k) + s_k + A(x_k) d_x + d_s = 0 \\ & \|d_x, S_k^{-1} d_s\| \leq \Delta_k, d_s \geq -\tau s_k \end{aligned} \tag{5.4}$$

to compute a new step d_k for a given iterate z_k and Lagrange multipliers λ_k. For this purpose, we will use a trust region algorithm discussed in detail below. Here, we introduce the scaling S_k^{-1} in the trust region constraint in (5.4) that penalizes d_s near the boundary of the feasible region and hence avoids the slack variables approach zero prematurely. The additional parameter $\tau \in (0, 1)$ is chosen close to 1 to impose the well-known fraction to the boundary rule [79] which ensures that the slack variables remain positive. As B_k one may use the exact second-order information $\nabla^2_{xx}\mathcal{L}(x_k, s_k, \lambda_k)$. Then, the functions $f(\cdot)$ and $c(\cdot)$ are assumed to be twice continuously differentiable. Alternatively, one may employ a symmetric matrix approximating the Hessian information. Finally, $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$.

Since the trust region problem (5.4) may have no feasible solution, we consider a composite-step method. Following the approach of Byrd and Omojokun, we define the merit function

$$\phi(z; \nu) = f(x) - \mu \sum_{i=1}^{M} \ln s^i + \nu \|g(x) + s\| \equiv \varphi(z) + \nu \|g(x) + s\|$$

with the penalty parameter $\nu > 0$ to judge the progress toward the solution. This merit

function is exact but non-differentiable due to the Euclidean norm in the second term. A model of $\phi(\cdot;\nu_k)$ around an iterate x_k is given by the function

$$\begin{aligned} m_k(d) =& f(x_k) + \nabla f(x_k)^\top d_x + \frac{1}{2} d_x^\top B_k d_x + \nu_k \|g(x_k) + s_k + A(x_k) d_x + d_s\| \\ & - \mu \left(\sum_{i=0}^{M} \ln s_k^i + e^\top S_k^{-1} d_s - \frac{1}{2} d_s^\top S_k^{-2} d_s \right). \end{aligned}$$

For measuring the progress of our algorithm, we define, for a given iterate z_k and a step d, the actual reduction in the merit function as

$$\operatorname{ared}_k(d) = \phi(z_k;\nu_k) - \phi(z_k + d;\nu_k). \tag{5.5}$$

The predicted reduction in the merit function is defined as the change of the model m_k caused by a step d, i.e.,

$$\begin{aligned} \operatorname{pred}_k(d) &= m_k(0) - m_k(d) \\ &= -\nabla f(x_k)^\top d_x - \frac{1}{2} d_x^\top B_k d_x \\ &\quad + \nu_k (\|g(x_k) + s_k\| - \|g(x_k) + s_k + A(x_k) d_x + d_s\|) \\ &\quad + \mu \left(e^\top S_k^{-1} d_s - \frac{1}{2} d_s^\top S_k^{-2} d_s \right). \end{aligned} \tag{5.6}$$

We suppose that for each iteration k one can provide an approximation A_k of the exact Jacobian $A(x_k)$ of the inequality constraints. Furthermore, we define an exact null space basis of $\hat{A}(z) = [A(x_k)\ I]$ as

$$Z(x_k)^\top = [I\ -A(x_k)^\top] \tag{5.7}$$

with $\hat{A}(z_k)Z(x_k) = 0$ and $\hat{A}_k Z_k = 0$ as the corresponding approximation with $Z_k^\top = [I \; -A_k^\top]$.

The approximation of the derivative matrices using quasi-Newton update formulas fits into this setting. For this purpose, one may employ the well-known symmetric rank one (SR1) update formula to approximate the Hessian $\nabla^2_{xx}\mathcal{L}(x_k, s_k, \lambda_k)$. This approach is examined for unconstrained optimization in [20]. The two-sided rank one (TR1) update formula (4.9) as proposed in [33] can be used to approximate the constraint Jacobian. It has been studied for equality constrained problems in [75, 72]. Another possibility would be to compute the required Hessian-vector products exactly employing, for example, automatic differentiation (AD). For the first-order information, the exact information $A(x_k)$ could be computed for the iterate k and used for the following iterates as long as the restrictions on the inexactness are fulfilled. This is a promising approach since the iterates converge frequently in a tangential way toward the optimal solution. This observation holds when the Hessian is approximated for example by a quasi-Newton formula and the exact Jacobian of the constraints is used.

To prove the convergence results presented, we make the following assumptions:

(AS1) The functions $f(\cdot)$ and $c(\cdot)$ are twice continuously differentiable on an open convex set $\mathcal{X}$ containing all iterates.

(AS2) The sequence $\{f(x_k)\}$ is bounded below. The sequences $\{\nabla f(x_k)\}$, $\{c(x_k)\}$, $\{A(x_k)\}$, and $\{B_k\}$ are bounded.

(AS3) The functions $\nabla f(\cdot)$, $c(\cdot)$, and $A(\cdot)$ are Lipschitz continuous on an open convex set $\mathcal{X}$ containing all iterates.

(AS4) The gradients $\nabla f(x)$, $\nabla_x \mathcal{L}(x, s, \lambda)$, the gradient-vector product $\nabla f(x)^\top d$ and the products $A(x)v$, $w^\top A(x)$ can be evaluated exactly.

(AS5) For fixed x_k, the approximation A_k can be improved in a finite number of steps such that an exact Jacobian $A(x_k)$ is obtained.

Assumptions (AS1) - (AS3) are standard assumptions required also for the global convergence analysis in other papers.

To provide the exact gradients, Jacobian×vector and vector×Jacobian products required in (AS4), one may apply for example automatic differentiation [31] or even direct sensitivity and adjoint differential equations if the NLP description (5.1) is based on differential equations (Chapter 3).

In (AS5), we assume that we can improve the approximation A_k such that it becomes $A(x_k)$ after a finite number of improvement steps. This is possible, for example, for the TR1 approach by performing M rank one updates without changing the current iterate x_k since the TR1 update procedure yields the exact Jacobian $A(x_k)$ for fixed x_k after at most M updates as shown in [75]. If one freezes the Jacobian and null space information as proposed above, one can evaluate new exact Jacobian information if the restrictions on the inexactness are not valid anymore. This approach ensures that assumption (AS5) holds. Hence, one can use the approximation $Z_k = Z_{k-1}$ and improve the approximation

of the null space if required.

5.3 A Jacobian-free Trust-Region Method

To apply a composite step trust-region method as proposed by Byrd and Omojokun, we first compute a normal step n that lies well inside the trust-region radius and that attempts to satisfy the linear constraints in (5.4). Subsequently, we take a tangential step t toward optimality. Putting both steps together, we obtain the total step $d = n + t$.

5.3.1 The Normal Subproblem

For the current iterate $z_k^\top = [x_k^\top, s_k^\top]$, we compute a normal step $n_k = (n_{x,k}, n_{s,k})$ that best satisfies the linearized constraints by solving the *normal subproblem*

$$\begin{aligned} &\min_n \ \|g(x_k) + s_k + A(x_k)n_x + n_s\|^2 \\ &\text{subject to } \|(n_x, S_k^{-1}n_s)\| \leq \tilde{\Delta}_k \\ &n_s \geq -\xi\tau s_k \end{aligned} \tag{5.8}$$

with $\tilde{\Delta}_k = \xi\Delta_k$ and $\xi \in (0, 1)$. This optimization problem may have infinitely many solutions. To simplify the constraints we define $\tilde{n} = (n_x, \tilde{n}_s) = (n_x, S^{-1}n_s)$ and $\hat{A}(x_k) = [A(x_k)\ S_k]$ The exact steepest descent direction for the unconstrained problem of (5.8) is given by (see [13])

$$\tilde{n}_k^C = -\begin{pmatrix} A(x_k)^\top \\ S_k \end{pmatrix}(g(x_k) + s_k). = \hat{A}(x_k)^\top(g(x_k) + s_k) \tag{5.9}$$

Note that with assumption (AS4) we can evaluate $A(x_k)v$ and $A(x_k)^\top w$ for given v and w exactly, so we are able to compute the exact steepest descent direction. Then, the exact Cauchy step for (5.8) is given by $\alpha_k^C \tilde{n}_k^C$, where α_k^C is the optimal solution of the problem:

$$\begin{aligned} &\min_{\alpha \geq 0} \ \|g(x_k) + s_k + \alpha(A(x_k)n_{x,k}^C + S\tilde{n}_{s,k}^C)\|^2 \\ &\text{subject to } \|\alpha(n_{x,k}^C, \tilde{n}_{s,k}^C)\| \leq \tilde{\Delta}_k. \\ &\alpha\tilde{n}_{s,k}^C \geq -\xi\tau \end{aligned} \tag{5.10}$$

Nevertheless, employing only the Cauchy step may yield very slow convergence [51]. To accelerate the optimization process, one could use in addition also the exact Newton step or approximations thereof. The exact Newton step of (4.15) is given by

$$\tilde{n}^N(x_k) = -\hat{A}(x_k)^+(g(x_k) + s_k) = -\hat{A}(x_k)^T(\hat{A}(x_k)\hat{A}(x_k)^T)^{-1}(g(x_k) + s_k) \tag{5.11}$$

However, it is expensive to compute $(\hat{A}(x_k)\hat{A}(x_k)^T)^{-1}(g(x_k) + s_k)$ exactly. Using the QR factorization and its orthonormal properties, an approximation

$$\tilde{n}^N(x_k) = -\hat{A}(x_k)^T(\hat{A}_k(\hat{A}_k)^T)^{-1}(g(x_k) + s_k) \tag{5.12}$$

of the exact Newton step (5.11) can be computed easily. In combination with the exact Cauchy step, we define an inexact dogleg step by setting

$$\tilde{n}^D(x_k) = \tilde{\eta}\tilde{n}^N(x_k) + (1 - \tilde{\eta})\tilde{n}^C(x_k) \tag{5.13}$$

with $\tilde{\eta} = 1$ if $\| \tilde{n}^N \| \leq \tilde{\Delta}_k$. Otherwise $\tilde{\eta} \in [0, 1]$ would be adjusted such that the length of $\tilde{n}^D$ is equal to $\tilde{\Delta}_k$. Because of the bounds on the slack variables in (5.8), the Cauchy and

Newton steps are truncated such that $\tilde{\theta}^C\tilde{n}^C$ and $\tilde{\theta}^N\tilde{n}^N$ are feasible where $\tilde{\theta}^C, \tilde{\theta}^N \in (0, 1]$. The choice $\tilde{n}_k$ is governed by

$$\tilde{n}_k = \begin{cases} \tilde{n}^D(x_k) & \text{if } \mathrm{npred_k}(\tilde{n}^D(x_k)) \geq \gamma_n \mathrm{npred_k}(\tilde{\theta}^C\tilde{n}^C(x_k)) \\ \tilde{\theta}^C\tilde{n}^C(x_k) & \text{otherwise} \end{cases} \tag{5.14}$$

for some constant $\gamma_n > 0$, and the normal predicted reduction is given by,

$$\mathrm{npred}_k(\tilde{n}) = \|g(x_k) + s_k\| - \|g(x_k) + s_k + A(x_k)n_x + S\tilde{n}_s\| \tag{5.15}$$

Finally, we transform $\tilde{n}$ into the original space of variable to obtain the normal step $n = (n_x, S\tilde{n}_s)$.

To allow more freedom in the choice of the direction guaranteeing convergence, one has to analyze the reduction in the linearized constraints caused by the normal step. For that purpose, we define the *normal predicted reduction* for a vector n as

$$\mathrm{npred}_k(n) = \|g(x_k) + s_k\| - \|g(x_k) + s_k + A(x_k)n_x + n_s\| \tag{5.16}$$

and require that the normal step n_k computed in the kth iteration satisfies the following condition:

Normal Cauchy Decrease Condition. *An approximate solution n_k of the normal subproblem* (5.8) *must satisfy*

$$\mathrm{npred}_k(n_k) \geq \gamma_n \mathrm{npred}_k(\alpha_k^C n_k^C), \tag{5.17}$$

for some constant $\gamma_n > 0$.

For our convergence analysis, the normal steps n_k additionally have to fulfill the *range space condition*

$$\exists\, v_k \in \mathbb{R}^M \quad \text{such that} \quad n_k = \begin{pmatrix} A(x_k)^\top \\ S_k^2 \end{pmatrix} v_k. \tag{5.18}$$

Note that the Cauchy step n_k^C is of this form.

Since $\alpha = 0$ is feasible for the optimization problem (5.10), it follows from (5.17) that

$$\text{npred}_k(n_k) \geq 0 \tag{5.19}$$

holds. One can improve the bound on the normal predicted reduction as shown for example in [23, Lemma 15.4.17] and [13, Lemma 2]. The main ingredient of the proofs is the normal Cauchy decrease condition. Since the inexactness of the Jacobian does not influence the derivation of the result, we skip the proof of the following lemma. It can be proved along the lines of Lemma 2 in [13].

Lemma 5.3.1. *Suppose that $s_k > 0$ and that n_k be an approximate solution of the normal subproblem* (5.8) *such that* (5.17) *holds. Then*

$$\begin{aligned} &\|g(x_k) + s_k\| npred_k(n_k) \geq \\ &\frac{\gamma_n}{2} \left\| \begin{pmatrix} A(x_k)^\top \\ S_k \end{pmatrix} (g(x_k)+s_k) \right\| \min \left\{ \tilde{\Delta}_k, \xi\tau, \frac{\left\| \begin{pmatrix} A(x_k)^\top \\ S_k \end{pmatrix} (g(x_k) + s_k) \right\|}{\left\| \begin{pmatrix} A(x_k)^\top \\ S_k \end{pmatrix} \right\|^2} \right\}. \end{aligned} \tag{5.20}$$

5.3.2 The Tangential Subproblem

Given a current iterate z_k, next we compute the tangential step towards optimality. Usually, one tries to maintain linearized feasibility, i.e., the exact tangential step $t(z_k) = Z(z_k)p_k$ should be in the exact null space of the constraints. Since we may be using an approximation Z_k of the exact null space $Z(x_k)$, we will have to safeguard the computation of the tangential step $t_k = Z_k p_k = [I \ -A_k]^\top p$ by limiting the amount of inexactness. On the other hand, since we have $Z(x_k)^\top = [I \ -A(x_k)^\top]$, we will be able to compute at least an exact tangential Cauchy step.

We first concentrate on computing an approximate solution of the inexact *tangential subproblem*

$$\begin{aligned} \min_p \ & (\nabla f(x_k) + B_k n_k)^\top Z_{x,k} p + \frac{1}{2} p^\top Z_{x,k}^\top B_k Z_{x,k} p \\ & \qquad -\mu \left(e^\top S_k^{-1} Z_{s,k} p - n_{s,k}^\top S_k^{-2} Z_{s,k} p - \frac{1}{2} p^\top Z_{s,k}^\top S_k^{-2} Z_{s,k} p \right) \\ & \text{subject to } \|(Z_{x,k} p, S_k^{-1} Z_{s,k} p)\| \le \hat{\Delta}_k \\ & \qquad S_k^{-1}(n_s + Z_{s,k} p) \ge -\tau e \end{aligned} \tag{5.21}$$

with $\hat{\Delta}_k = \left(\Delta_k^2 - \|(n_{x,k}, S_k^{-1} n_{s,k})\|^2\right)^{1/2}$.

The steepest descent direction in the null space basis variables for this optimization problem is given by

$$p_k^C = -(\nabla f(x_k) + B_k n_{x,k}) - \mu A(x_k)^\top (S_k^{-1} e - S_k^{-2} n_{s,k}), \tag{5.22}$$

see, e.g., [13, 35]. In the inequality-constrained case, one can evaluate the product $Z(x_k)p_k^C$ exactly due to the representation (5.7) of the null space and Assumption (AS4). This dif-

fers significantly from the equality-constrained case and simplifies the proof of convergence considerably.

Once more, the pure steepest descent direction p_k^C may yield poor convergence. For this reason, also alternative tangential steps may be employed. To evaluate the improvement provided by the tangential step, we define the *tangential predicted reduction* produced by a tangential step $t = Z_k p$ as change in the objective function of the tangential subproblem. Hence, we have

$$\begin{aligned}\mathrm{tpred}_k(t) = &- \left(\nabla f(x_k) + B_k n_{x,k}\right)^\top t_x - \frac{1}{2} t_x^\top B_k t_x \\ &+ \mu \left(e^\top S_k^{-1} t_s - n_{s,k} S_k^{-2} t_s - \frac{1}{2} t_s^\top S_k^{-2} t_s \right).\end{aligned}$$

To ensure global convergence of the trust-region algorithm applied in the inner iteration, we will impose the following condition on the tangential step.

Tangential Cauchy Decrease Condition. *An approximate solution $t_k = Z_k p_k$ of the tangential subproblem (5.21) must satisfy*

$$tpred_k(t_k) \geq \gamma_t \, tpred_k(\theta_k^C Z(x_k) p_k^C), \tag{5.23}$$

for some constant $\gamma_t > 0$, where θ_k^C solves the problem

$$\begin{aligned}&\min_{\theta \geq 0} \left[-\mathrm{tpred}_k(\theta Z(x_k) p_k^C) \right] \quad \text{subject to} \\ &\|\theta(p_k^C, -S_k^{-1} A(x_k) p_k^C)\| \leq \hat{\Delta}_k \qquad S_k^{-1}(n_{s,k} - A(x_k) p_k^C) \geq -\tau e.\end{aligned} \tag{5.24}$$

Since $\theta = 0$ is feasible for the optimization problem (5.24), it follows that

$$\text{tpred}_k(t_k) \geq 0. \tag{5.25}$$

As with the normal step, we use the tangential Cauchy decrease condition to provide a bound on sufficient decrease for the solution of the tangential subproblem. As long as this condition is satisfied, this does not require an exact null space representation for the tangential subproblem (5.21). Lemma 5.3.2 provides this condition and is the tangential analog of Lemma 5.3.1 for the normal step.

Lemma 5.3.2. *Suppose that $s_k > 0$ and let t_k be an approximate solution of the tangential subproblem* (5.21) *that satisfies* (5.23)*. Then*

$$\begin{aligned} &\textit{tpred}_k(t_k) \geq \frac{\gamma_t}{2}\|p_k^C\|\hat{M} \qquad \textit{with} \\ &\hat{M} = \min\left\{ \frac{\min\{\hat{\Delta}_k, (1-\xi)\tau\}}{\|I + A(x_k)^\top S_k^{-2} A(x_k)\|^{\frac{1}{2}}}, \frac{\|p_k^C\|}{\|B_k + \mu A(x_k)^\top S_k^{-2} A(x_k)\|^{\frac{1}{2}}} \right\}. \end{aligned} \tag{5.26}$$

The proof of this lemma follows exactly along the lines of Lemma 3 in [13]. It is based on the Cauchy decrease for $\text{tpred}(\theta^C Z(x_k) p^C)$ and the result follows from application of (5.23).

To accelerate the convergence rate, one should not use the steepest descent direction given by (5.22) but an approximation of the Newton step. For this purpose, we can apply the Steihaug CG algorithm, see, e.g., [23, 51], or a direct solve if the corresponding system is small enough as long as (5.23) is fulfilled for the tangential step t_k.

Nevertheless, we note that the matrix A_k only approximates $A(x_k)$, so additional care is needed to make sure that (5.23) holds. Fortunately, with sufficient updates of A_k, Assumption (AS5) guarantees that this condition is satisfied.

On the other hand, one has for the combined step $d_k = n_k + t_k$ that the identity $\hat{A}(z_k)d_k = \hat{A}(z_k)n_k$ is not necessarily valid. Therefore, we obtain for the predicted reduction (5.6) of the function m_k the equation

$$\begin{aligned}\mathrm{pred}_k(d_k) = & -\nabla f(x_k)^\top (n_{x,k} + t_{x,k}) - \frac{1}{2}(n_{x,k} + t_{x,k})^\top B_k (n_{x,k} + t_{x,k}) \\ & + \nu_k (\|g(x_k) + s_k\| \\ & \qquad - \|g(x_k) + s_k + A(x_k)(n_{x,k} + t_{x,k}) + (n_{s,k} + t_{s,k})\|) \\ & + \mu \left(e^\top S_k^{-1}(n_{s,k} + t_{s,k}) - \frac{1}{2}(n_{s,k} + t_{s,k})^\top S_k^{-2}(n_{s,k} + t_{s,k}) \right). \\ = & \ \mathrm{tpred}(t_k) + \nu_k \mathrm{npred}(n_k) + \chi_k + \mathrm{err}_k(d_k),\end{aligned}$$

where

$$\chi_k = -\nabla f(x_k)^\top n_{x,k} - \frac{1}{2} n_{x,k}^\top B_k n_{x,k} + \mu \left(e^\top S_k^{-1} n_{s,k} - \frac{1}{2} n_{s,k}^\top S_k^{-2} n_{s,k} \right) \tag{5.27}$$

as defined in [13], and the remaining error terms:

$$\begin{aligned}\mathrm{err}_k(d_k) = \nu_k (&\|g(x_k) + s_k + A(x_k)(n_{x,k} + t_{x,k}) + (n_{s,k} + t_{s,k})\| \\ &- \|g(x_k) + s_k + A(x_k) n_{x,k} + n_{s,k}\|).\end{aligned}$$

As can be seen, $\mathrm{err}_k(d_k)$ is a measure for the error in Z_k in approximating $Z(x_k)$. Since the usual identity for the predicted reduction is not valid, we define as in [74] an *inexact predicted reduction*:

$$\mathrm{ipred}_k(d_k) = \mathrm{tpred}(t_k) + \nu_k \mathrm{npred}(n_k) + \chi_k \tag{5.28}$$

that omits the error term. However, to ensure well-posedness and convergence of our

trust-region method, we impose a bound on the error term $\mathrm{err}_k(d_k)$. Obviously, one can derive that

$$\begin{aligned}|\mathrm{err}_k(d_k)| &= \nu_k \big| \|g(x_k) + s_k + A(x_k)n_{x,k} + n_{s,k}\| \\ &\qquad - \|g(x_k) + s_k + A(x_k)d_{x,k} + d_{s,k}\| \big| \\ &\leq \nu_k \|A(x_k)t_{x,k} + t_{s,k}\| \leq \nu_k\, \gamma \Delta_k^2.\end{aligned}$$

Hence, one may use a criterion like

$$\|A(x_k)t_{x,k} + t_{s,k}\| \leq \gamma \Delta_k^2 \tag{5.29}$$

for a constant $\gamma > 0$ to bound the inexactness that is due to the tangential step. This inequality can be easily verified by evaluating one Jacobian-vector product. However, with (5.29) $\mathrm{pred}_k(d_k)$ may still become negative if $\mathrm{err}_k(d_k)$ is large relative to $\mathrm{ipred}_k(d_k)$. For this reason, we will use the direct criterion:

$$-\mathrm{err}_k(d_k) \leq \frac{(1-\eta)}{2}\, \mathrm{ipred}_k(d_k) \tag{5.30}$$

for a constant $\eta \in (0, 1)$ similar to the equality constraint case studied in [74]. This inequality can be used to control the error in the inexact predicted reduction and therefore helps to ensure well-posedness of the algorithm. Note that one only has to bound a negative $\mathrm{err}_k(d_k)$ since a positive error leads to an even larger $\mathrm{pred}_k(d_k)$. If (5.30) holds,

one has

$$\begin{aligned} \text{pred}_k(d_k) &= \text{ipred}_k(d_k) + \text{err}_k(d_k) \\ &\geq \text{ipred}_k(d_k) - \frac{(1-\eta)}{2}\,\text{ipred}_k(d_k) \\ &\geq \frac{(\eta+1)}{2}\,\text{ipred}_k(d_k) \geq 0 \end{aligned} \tag{5.31}$$

if $\text{ipred}_k(d_k) \geq 0$. Once more, (5.30) can be easily verified by evaluating two Jacobian-vector products.

This lower bound (5.31) allows us now to work directly with $\text{ipred}_k(d_k)$ (and $\text{npred}_k(d_k)$ and $\text{tpred}_k(d_k)$) instead of with pred_k, which contains the additional error terms.

5.3.3 The Trust-Region Algorithm

After specifying the computation of the normal and tangential step, we can now state the algorithm for solving the barrier problem (5.2) for a fixed value of barrier parameter μ:

Algorithm 5.3.3 (Inner trust-region algorithm)**.**

Start: *Set initial values* $z_0^\top = [x_0^\top, s_0^\top]$ *with* $s_0 > 0$, λ_0, $\nu_{-1} > 0$, A_0, Δ_0, $\rho \in (0,1)$, $\eta \in (0,1)$

for $k = 0, 1, \ldots$

1. *Compute a normal step* n_k *such that* (5.17) *and* (5.18) *hold.*
2. *Compute a tangential step* t_k *such that* (5.23) *holds.*

   Compute the total step $d_k = n_k + t_k$.
3. *Compute the smallest value* $\tilde{\nu}_k$ *such that*

$$ipred_k(d_k) = tpred(t_k) + \tilde{\nu}_k npred(n_k) + \chi_k \geq \rho\,\tilde{\nu}_k\, npred_k(n_k). \tag{5.32}$$

If $\tilde{\nu}_k \leq \nu_{k-1}$, *set* $\nu_k = \nu_{k-1}$, *otherwise set* $\nu_k = \max\{\tilde{\nu}_k, 1.5\nu_{k-1}\}$.

4. *If* (5.30) *does not hold, update* A_k *and go to step 1.*

5. *If*

$$ared_k(d_k) < \eta\, ipred_k(d_k)$$

decrease Δ_k *by a constant factor and go to 1.*

6. *Set* $x_{k+1} = x_k + d_{x,k}$, $s_{k+1} = \max\{s_k + d_{s,k}, -g(x_{k+1})\}$, *and choose* Δ_{k+1} *such that* $\Delta_{k+1} \geq \Delta_k$

7. *Compute new* A_{k+1} *and Lagrange multipliers* $\lambda_{k+1} = \mu S_{k+1}^{-1} e$.

8. *If* $\nabla f(x_{k+1}) + A(x_{k+1})^\top \lambda_{k+1} = 0$ *and* $g(x_{k+1}) \leq 0$, *stop.*
Else increase k *by 1 and go to 1.*

Algorithm 5.3.3 represents a Byrd-Omojokun trust-region algorithm that takes the inexactness of the Jacobian and its null space representation into account. Here, the computation of a normal direction in Step 1 is identical to a standard approach where the normal Cauchy decrease condition has to be fulfilled. Note that the inexactness of the Jacobian may enter into the normal direction due to the choice of the normal step. The tangential direction computed in Step 2 has to fulfill the tangential Cauchy decrease condition. Both are standard requirements for the Byrd-Omojokun algorithm.

In Step 3, χ_k can be of any sign. Furthermore, we have that $\mathrm{npred}_k(n_k)$ and $\mathrm{tpred}_k(t_k)$ are nonnegative due to (5.19) and (5.25). Hence, if $\mathrm{npred}_k(n_k) > 0$ holds it follows that $\mathrm{ipred}_k(d_k) \geq \rho\, \nu_k\, \mathrm{npred}_k(n_k)$ is valid when

$$\nu_k \geq \frac{-\chi_k}{(1-\rho)\mathrm{npred}_k(n_k)}.$$

This lower bound is a sufficient condition, but not necessary, as condition (5.32) may hold also for smaller values of ν_k. If $\text{npred}_k(n_k) = 0$ one can conclude from Lemma 5.3.1 and $s_k > 0$ that $g(x_k) + s_k = 0$. Therefore, $n = 0$ solves the normal subproblem (5.8). Since $s_k > 0$, the squared objective of the normal subproblem (5.8) is positive definite on the range space of $\hat{A}(x_k)$. Hence, $n = 0$ is the unique minimizer in the range space. It follows for $\text{npred}_k(n_k) = 0$ that $n_k = 0$, $\chi_k = 0$, and that (5.32) is satisfied for any value of ν_k.

The additional test on (5.30) in Step 4 ensures that the inexactness of the Jacobian and its null space representation does not harm the tangential direction too much. Due to assumption (AS5), we need only a finite number of improvement steps for fixed x_k to obtain an exact $Z_k = Z(x_k)$ such that (5.30) is fulfilled.

Steps 5 and 6 are standard update procedures of a trust-region algorithm, except that $\text{ipred}(d_k)$ is not equal to the predicted reduction $\text{pred}(d_k)$ due to the inexactness.

In Step 7, we compute an approximation A_{k+1} to the Jacobian as well as multipliers λ_{k+1} based on the values of the slack variables. Finally, in Step 8 we check whether x_{k+1} is a stationary point of the inexact problem, i.e., whether the equations

$$\nabla f(x_{k+1}) + \mu A(x_{k+1})^\top S_{k+1}^{-1} e = 0, \qquad g(x_{k+1}) \leq 0$$

hold. This can easily be checked through a vector×Jacobian product.

5.4 Well-posedness of Algorithm 5.3.3

An important property of a trust-region algorithm is well-posedness. Here, one has to show that the trust-region radius cannot shrink to zero if an iterate x_k is not a stationary point of the NLP (5.1). For this purpose, we analyze the relation of the actual and

predicted reduction. We will employ ideas used in the proof of Lemma 4 in [13]. In addition, we must take the inexactness of the Jacobian and its null space representation into account. That is, we have to ensure that the error term $\mathrm{err}_k(d_k)$ does not dominate the model. In Step 4 of Algorithm 5.3.3, we require that (5.30) holds. Employing this inequality, we can prove the following result that is related to Lemma 4 in [13].

Lemma 5.4.1. *Let the assumptions (AS2), (AS3), and (AS5) hold on the open convex set $\mathcal{X}$ containing all iterates. Then there exists a positive constant ζ such that for any iterate z_k and any step $d_k = n_k + t_k$ generated by Algorithm 5.3.3 with $[x_k, x_k + d_{x,k}] \subset \mathcal{X}$, $d_{s,k} \geq -\tau s_k$, and $ared_k(d_k) \leq \eta\, ipred_k(d_k)$, it follows that*

$$0 \leq \eta ipred_k(d_k) - ared_k(d_k) \leq \zeta(1 + \nu_k)\Delta_k^2 \tag{5.33}$$

Proof. The proof is adapted from Lemma 4 in [13] and is given in Walther et al. [76]. □

Next, we show that Algorithm 5.3.3 can not generate an infinite cycling between Steps 1 and 5.

Lemma 5.4.2. *Let the assumptions (AS2), (AS3), and (AS5) hold on the open convex set $\mathcal{X}$ containing all iterates. Suppose that $s_k > 0$ and that (x_k, s_k) is not a stationary point of the barrier problem (5.2). Then there exists a Δ_k^0 such that*

$$ared_k(d_k) \geq \eta\, ipred_k(d_k)$$

for any $\Delta \in (0, \Delta_k^0)$.

Proof. This proof follows along the lines of the proof of Proposition in [13] and is given in [76]. □

5.5 Convergence Analysis

We define the scaled merit function

$$\tilde{\phi}(z;\nu) \equiv \frac{1}{\nu}\phi(z;\mu) = \frac{1}{\nu}\left(f(x) - \mu\sum_{i=1}^{M}\ln s^i\right) + \|g(x)+s\|.$$

Then, we can deduce from Step 4 of Algorithm 5.3.3 that

$$\begin{aligned}\tilde{\phi}(z_{k-1},\nu_{k-1}) - & \frac{\eta\,\mathrm{ipred}_{k-1}(d_{k-1})}{\nu_{k-1}} \\ & \geq \tilde{\phi}(z_k,\nu_{k-1}) \\ & = \tilde{\phi}(z_k,\nu_k) + \left(\frac{1}{\nu_{k-1}} - \frac{1}{\nu_k}\right)\left(f(x_k) - \mu\sum_{i=1}^{M}\ln s_k^i\right) \\ & \quad - \frac{\eta\,\mathrm{ipred}_{k-1}(d_{k-1})}{\nu_{k-1}}.\end{aligned}$$

For the derivation of the next result, it is not required to take the inexactness of A_k into account. It can be proved exactly along the lines of [13, Lemma 5 and Lemma 6] by substituting pred by ipred. Therefore, we skip the proofs here.

Lemma 5.5.1 (Boundedness of the slack variables and the steps)**.** *Assume that (AS1) and (AS2) hold. Then one has that:*

- *The sequence $\{s_k\}$ is bounded.*
- $\|z_{k+1} - z_k\| \leq \gamma_0\Delta_k$

This lemma also shows that the modified merit function $\tilde{\phi}(z_{k-1},\nu_{k-1})$ is bounded below, a result that is required to prove Lemma 5.5.2. For the derivation of this result, it is again not required to handle the inexactness of A_k directly. The inexact first-order

information are taken into account by Lemma 5.4.1 which is used to prove lemma. The proof is adapted from [13, Lemma 7] and is given in [76].

Lemma 5.5.2 (Stationarity of all limit points)**.** *Assume that (AS1), (AS2), and (AS3) hold. Then, we have*

$$\lim_{k\to\infty} \begin{pmatrix} A_k \\ S_k \\ s \end{pmatrix} (g(x_k) + s_k) = 0.$$

Lemma 5.5.2 shows that $A(x_k)(g(x_k)+s_k) \to 0$ and $S_k(x_k)(g(x_k)+s_k) \to 0$. Hence, if $[A(x_k)\ S_k]$ has full rank, which has not been assumed so far, it follows that $g(x_k)+s_k \to 0$, i.e., the iterates converge to a feasible point. If the full rank property is not fulfilled, then $[A(x_k)\ S_k]$ converges to a degenerate matrix. Due to Step 6. in Algorithm 5.3.3, one has

$$g(x_k) + s_k \geq g(x_k)^+ \geq 0,$$

where $g(x_k)^+ = (\max\{g(x_k)^i, 0\})$. Using this property it is possible to show the following result exactly along the lines of [13, Lemma 8], for which we skip the proof.

Lemma 5.5.3. *Assume that (AS1), (AS2), and (AS3) hold. Then, we have*

$$A(x_k)^\top g(x_k)^+ \to 0 \qquad k \to \infty$$

Furthermore, if $g(x_k)^+ \not\to 0$ then the penalty parameters ν_k tend to infinity.

From now on, we assume that the matrix $(A(x_k)\ S_k)$ and its approximation $(A_k\ S_k)$ have full rank:

(AS6) $(A(x_k) \;\; S_k)$ has full row rank for all iterates z_k with $\sigma_D((A(x_k) \;\; S_k)) \geq \hat{\sigma} > 0$, where $\sigma_D((A(x_k) \;\; S_k))$ denotes the smallest singular value of $A(x_k)$.

(AS7) $(A_k \;\; S_k)$ has full row rank for all iterations with $\sigma_D((A_k \; S_k)) \geq \tilde{\sigma} > 0$.

With Assumption (AS6), we obtain from Lemma 5.2, that all limit points are feasible. To prove also the first-order optimality of all limit points, we need that the normal step can be bounded by the normal predicted reduction and that the penalty factor ν_k eventually becomes constant. For that purpose, we state the next two lemmas. For the proofs of the following two results, it is not necessary to handle the inexactness of A_k directly, so the proofs are given in [13].

Lemma 5.5.4 (Upper bound on normal step)**.** *Let assumptions (AS1) – (AS7) hold. Then there exist positive constants γ_1 and γ_2 such that if $\|g(x_k) + s_k\| \leq \gamma_1$ then*

$$\|(n_{x,k}, S_k^{-1} n_{s,k})\| \leq \gamma_2 \, npred_k(n_k) \tag{5.34}$$

Lemma 5.5.5 (Bound on tpred and constant ν_k for $k \geq k_1$)**.** *Suppose that the assumptions (AS1) – (AS7) are satisfied. Then the sequence of penalty parameters $\{\nu_k\}$ is bounded. Furthermore, there exist an index k_1 and positive constants $\bar{\nu}$ and γ_3, such that $\nu_k = \bar{\nu}$ holds for all $k \geq k_1$ and*

$$ipred_k(d_k) \geq \gamma_3 tpred_k(t_k). \tag{5.35}$$

The main result of this chapter deals with global convergence for the inner trust-region method given by Algorithm 5.3.3.

Theorem 5.5.6 (All limit points are first-order optimal)**.** *Suppose that (AS1) – (AS7) hold. Then, it follows that*

$$\begin{aligned}\lim_{k\to\infty} \nabla_x \mathcal{L}(x_k, \lambda_k) &= \lim_{k\to\infty} (\nabla f(x_k) + A(x_k)^\top \lambda_k) \\ &= \lim_{k\to\infty} (\nabla f(x_k) + \mu A(x_k)^\top S_k^{-1} e) = 0,\end{aligned}$$

$\{s_k\}$ is bounded away from zero and $g(x_k)$ is negative for all large k.

Proof. Step 7 of Algorithm 5.3.3 ensures that

$$\begin{aligned}&\text{tpred}_k(t_k) \geq \frac{\gamma_t}{2} \|p_k^C\| \hat{M} \qquad \text{with} \\ &\hat{M} = \min\left\{ \frac{\min\{\hat{\Delta}_k, (1-\xi)\tau\}}{\|I + A(x_k)^\top S_k^{-2} A(x_k)\|^{\frac{1}{2}}}, \frac{\|p_k^C\|}{\|B_k + \mu A(x_k)^\top S_k^{-2} A(x_k)\|^{\frac{1}{2}}} \right\}.\end{aligned}$$

for $\gamma_t > 0$ and where $p_k^C = -(\nabla f(x_k) + B_k n_{x,k}) + \mu A(x,k)^\top (S_k^{-1} e - S_k^{-2} n_{s,k})$. Because the sequences $\{A(x_k)\}$ and B_k are bounded, tpred can be rewritten as:

$$\text{tpred}_k(t_k) \geq \gamma_1' \|p_k^C\| \min(1, \tilde{\Delta}_k, \|p_k^C\|). \tag{5.36}$$

for some positive constant γ_1'. We further define the quantity

$$q_k = \nabla f(x_k) + \mu A(x_k)^\top S_k^{-1} e \tag{5.37}$$

and by the boundedness of $A(x_k)$ we can define:

$$\|p_k^C\| \geq \gamma_2' \|q_k\| - \gamma_3' \|n_k\|. \tag{5.38}$$

The rest of the proof follows as in Lemma 12 in [13]. It shows, by contradiction that the sequence $\{q_k\} \to 0$ under the above assumptions and the fact that $n_k \to 0$. □

5.6 Convergence of the Barrier Algorithm

For overall convergence, the convergence of the outer barrier algorithm must be shown. For this purpose, we state the algorithm in more detail:

Algorithm 5.6.1 (Outer Barrier Algorithm)**.**

Start: *Set initial values x_0, s_0 a barrier parameter $\mu_0 > 0$, a reduction factor $\alpha \in (0,1)$ and a sequence $\{\varepsilon_l\}$ that tends to zero*

for $l = 0, 1, \ldots$

1. *Apply Algorithm 5.3.3 to compute a point (x_l, s_l) for the given (x_{l-1}, s_{l-1}) such that*

$$\|g(x_l) + s_l\| \leq \varepsilon_l \tag{5.39}$$

$$\|\nabla f(x_l) + \lambda_l^\top A(x_l)\| \leq \varepsilon_l \tag{5.40}$$

where $\lambda_l = \mu_l S_{k+1}^{-1} e$

2. *Choose $\mu_{l+1} \in (0, \alpha\mu_l)$*

Hence, the (x_l, s_l) denote only the iterates of the outer iteration. With respect to convergence, one can show the following results:

Theorem 5.6.2 (Convergence of the outer barrier algorithm)**.** *Let the iterates (x_l, s_l) be generated by Algorithm 5.6.1. Suppose that the assumptions (AS1) – (AS7) hold. Then it*

follows that each inner iteration finds a (x_l, s_l) *satisfying* (5.39) *and* (5.40)*. Furthermore the first order optimality conditions hold at each limit point* x_* *of* $\{x_l, s_l\}$ *that satisfies the linear independence constraint qualification. That is, there exists* $\lambda \in \mathbb{R}^M$ *such that*

$$\nabla f(x_*) + \lambda_*^\top A(x_*) = 0, \quad g(x_*) \leq 0, \quad \lambda_* \geq 0, \quad g(x_*)^\top \lambda_* = 0.$$

Proof. Due to the above assumptions, Algorithm 5.3.3 at iteration l always converges to a limit point $\{x_l, s_l\}$ such that (5.39) and (5.40) hold as shown in Sec. 5.5. Let $\{x_{l_k}\}$ be a subsequence that converges to x_*. One has that $g(x_{k_l}) + s_{k_l} \to 0$ and $s_{k_l} > 0$. Therefore x_* is feasible. Furthermore, x_* satisfies the linear independence constraint qualification. With $I = \{i : g_i(x_*) = 0\}$, it follows for $i \notin I$ that $g_i(x_*) < 0$ and $s_{i,k_l} > 0$. This yields $\lambda_{i,k_l} = \mu_{k_l}/s_{i,k_l} \to 0$ due to the definition of λ in Algorithm 5.3.3. Combining this with $\nabla f(x_{k_l}) + \lambda_l^\top A(x_{k_l}) \to 0$, one obtains

$$\nabla f(x_{k_l}) + \sum_{i \in I} \lambda_{i,k_l} \nabla g_i(x_{k_l}) \to 0.$$

Since the vectors $\{\nabla g_i(x_*)\}$ are linearly independent, the positive sequence $\{\lambda_{k_l}\}$ converges to a limit point $\lambda_* \geq 0$. Taking the limit yields $\nabla f(x_*) + \lambda_*^\top A(x_*) = 0$ and $g(x_*)^\top \lambda_* = 0$. □

5.7 Numerical Results

The barrier algorithm with inexact Jacobians was implemented in C, and the AD tool ADOL-C [32] provided the required exact matrix-vector products of the first and second derivatives. The Jacobian is approximated using the Two-sided Rank One (TR1) up-

date. The initial Jacobian, A_0 at the initial guess, x_0 is computed exactly and an exact Jacobian is also computed using AD when (5.30) fails. In addition to the Cauchy step computations in the inner normal and tangential problems, we also compute the dogleg step for the normal subproblem and use the preconditioned Steihaug conjugate gradient method for the tangential problem in order to accelerate the convergence rate. Also, for the outer iterations, we use the same barrier update strategy and tuning parameters that are reported in [73].

A comprehensive numerical study is beyond the scope of this chapter, as there are many aspects of a barrier code (including update of the barrier parameter, various approximation and restarting strategies, and a number of tuning parameters) that require a separate study. Instead, we focus on the performance of the inexact algorithm and compare it to its exact Jacobian counterpart. Here, we report the iteration counts and exact Jacobian evaluations for a set of small problems from the CUTEr collection as shown in Table 5.1. This set contains problems with linear or nonlinear objective functions and inequality constraints which are linear, nonlinear or just simple bounds. In this table N denotes the number of variables and M denotes the total number of constraints, including general inequalities and bounds.

To judge the performance of the inexact algorithm, optimization was also performed with an exact Jacobian computed in each optimization step with AD. Table 5.1 states the iteration counts of the outer problem and the number of Jacobian evaluations for the optimization runs. For the inexact approach, we also report the number of failures of (5.30). The total number of Jacobian evaluations for the inexact problem is equal to the number of failures in (5.30) plus one due to the initialization of A_0 with the exact Jacobian. The number of Jacobian evaluations for the exact problem gives an estimate of

the number of iterations for the inner trust-region problem. It is important to note that the results presented in the Table 5.1 can be improved by tuning the parameters of the algorithm like initial trust-region radius, penalty parameter, updating barrier parameter etc. To keep it consistent, we just compare the results of the inexact algorithm with exact method.

As can be seen in Table 5.1, the inexact algorithm compares very well to its exact counterpart and the iteration count is reasonable for all problems. Moreover, the number of times the test (5.30) fails is relatively rare and hence a new exact Jacobian computation is seldom required. In particular, comparing the evaluation count for full Jacobians in the next to last column of Table 5.1 with the full Jacobian evaluations for the exact approach as given in the last column, one observes significant savings can be obtained for this collection of test problems.

5.8 Concluding Remarks

In this chapter, we have proposed and analyzed a class of trust-region methods for inequality constrained nonlinear optimization problems based only on inexact information on the constraint Jacobian, without any assumption on the method to approximate these matrices. We prove global first-order convergence for the presented algorithm under mild conditions. Moreover, numerical results on small examples demonstrate significant potential advantages of this approach.

Table 5.1: Optimization details for CUTEr problems

					evaluation of full $A(x_k)$		
					inexact		exact
			Outer iterations		fail. in		
problem	N	M	inexact	exact	(5.30)	total	total
twobars	2	6	5	5	123	124	232
cb2	3	3	6	6	111	112	162
camel6	2	4	5	5	0	1	7
cantilvr	5	6	6	6	1	2	51
hatfldb	4	5	6	5	0	1	18
hatfldc	4	8	5	5	0	1	20
hs001	2	1	4	4	0	1	39
hs002	2	1	5	5	0	1	48
hs003	2	1	4	4	0	1	27
hs005	2	4	6	6	0	1	15
hs010	2	1	6	5	5	6	125
hs011	2	1	5	5	7	8	28
hs012	2	1	5	5	9	10	59
hs022	2	2	5	5	0	1	53
hs033	3	6	6	6	185	186	267
hs043	4	3	6	6	123	125	183
hs064	3	4	5	5	15	16	42
prob1	2	2	6	6	6	7	191
womflet	3	3	6	6	423	424	983

Chapter 6

PSA Superstructure for Pre-combustion CO_2 Capture

Synopsis

PSA is a potential capture technology candidate for CO_2 capture from effluent stream of shift converter. However, most of the PSA cycles available in literature are based on purifying the light component, H_2 and consider the heavy product, CO_2 as a waste stream. In this chapter, we present a systematic optimization-based formulation for the synthesis and design of novel PSA cycles for pre-combustion CO_2 capture which can simultaneously produce H_2 and CO_2 at a high purity. Here, we use a superstructure-based approach to simultaneously determine optimal cycle configurations and design parameters for PSA units. This approach is illustrated for two case studies. The first case study deals with obtaining optimal PSA cycle which maximizes CO_2 recovery by maintaining a desired purity level of CO_2 and H_2 purity. The second case study deals with minimizing the power consumption by maintaining a desired purity and recovery level of CO_2.

6.1 Introduction

Almost one-third of industrial CO_2 emissions are from energy conversion [36]. Since CO_2 accumulation of green-house gases may seriously affect the global climate, efficient capture and storage of carbon dioxide is very important. The purpose of CO_2 capture is to produce a concentrated stream to around 90 to 95 vol% that can be readily transported to a CO_2 storage site. One of the potential capture systems that has gained popularity in recent years is the pre-combustion capture system. Pre-combustion capture involves partial oxidation (gasification) of coal to produce syngas (or fuel gas) composed mainly of carbon monoxide and hydrogen. The carbon monoxide is reacted in a shift converter to increase carbon dioxide and hydrogen yield. CO_2 is then concentrated from this H_2/CO_2 mixture, resulting in a hydrogen-rich fuel and a CO_2-rich stream available for storage. Compared to post-combustion capture, a pre-combustion system is preferable for CO_2 capture because the fuel gas from the shift converter has a higher CO_2 concentration in the range 30-60%, and is also typically at a higher pressure, thus offering cost-effective means for CO_2 capture [60].

Among the capture technology candidates like absorption, adsorption, cryogenic distillation and membrane technology, pressure/vacuum swing adsorption (PSA/VSA) technology offers significant advantages for pre-combustion CO_2 capture in terms of performance, low energy requirements and operating costs. PSA processes achieve gas separation using principles of adsorption and desorption, and therefore operate at near ambient conditions in a solvent-free manner. Moreover, only feed compression and vacuum generation constitute the key energy requirements for the process, which can be substantially lower for pre-combustion capture because feed is available for separation at high pressure; this leads to reduced compression requirements. This is a well established technology for not

only removing trace components from a mixture, but also for extracting bulk products at extremely high purity [67, 56, 82, 63].

Most of the commercial PSA cycles have been developed [7, 81, 78, 39, 37, 54] to recover H_2 (light-product) at an extremely high purity, and do not focus on enriching the strongly adsorbed CO_2 (heavy-product). For example, Jiang et al. [39] studied a 5-bed 11-step PSA cycle to obtain a high purity H_2 from reformer gas. However, heavy product CO_2 is considered as a waste product. Thus, a major limitation exists with the use of these conventional PSA cycles for high purity CO_2 capture because the light-product purge step (or the light reflux step) in these cycles uses a portion of the light-product for purge. This necessarily dilutes the heavy component in the heavy-product stream. Therefore, a pure light component is easy to attain from such cycles, but not a pure heavy component. However, for CO_2 sequestration, it is necessary to concentrate CO_2 to a high purity. Very few cycles are available in literature that are designed to concentrate CO_2. The 5-bed 5-step PSA process developed in [64, 62] extracts methane and carbon dioxide both at a high purity from a feed mixture having 40-60% CO_2 and CH_4. A pure CO_2 rinse step was used in the process to obtain a CO_2 product containing 99.8-99% CO_2. Schell et al. [57] suggested a dual-reflux PSA process with a stripping and a rectifying section to obtain both light and heavy product at high purities. Xiao et al. [80] studied single-stage and dual-stage 2-bed 8-step VSA processes which could recover more than 90% of CO_2, at 95% purity, from a feed mixture having 21.5% CO_2 and 76.8% H_2. Air Products and Chemicals, Inc. have developed the Gemini process to simultaneously produce H_2 and CO_2 at high purities and recoveries [61]. Most of these PSA cycle configurations are based on heuristics and it is not clear why a particular configuration is preferred over others. In this chapter, we address the challenge of systematically synthesizing novel PSA cycles

and obtaining an optimal sequence of operating steps without making any assumption on the kinds of steps that should be included within the cycle.

In this chapter, we propose a systematic optimization-based framework to design and optimize new PSA cycles, and apply this approach to obtain PSA cycles for pre-combustion capture that produce both H_2 and CO_2 at a high purity and recovery. As discussed in the next section, optimal designs are generated from a 2-bed PSA superstructure after solving an optimal control problem. We consider a detailed partial differential algebraic equation (PDAE) based mathematical model, with the cyclic steady state condition, for the optimal control problem (OCP). This study is an extension to the earlier work by Agarwal et al. [3]. Agarwal et al. solved the OCP using a full discretization, where the PDAE model is fully discretized in the spatial and temporal domain resulting in a large number of algebraic constraints. Since the current implementation of the simultaneous approach has difficulties with solving large PSA problems, we use a partial discretization approach, that discretizes the PDAE model only in the spatial domain and the periodic operation of the PSA process is enforced as a set of equality constraints.

6.2 PSA Superstructure

The 2-bed PSA superstructure shown in Figure 6.1 consists of two beds, a co-current bed (CoB) and a counter-current bed (CnB) that determine co-current and counter-current operating steps in the cycle, respectively. The superstructure is designed with only two beds for two main reasons. Firstly, it maintains the flow to be always uni-directional i.e., co-current for CoB and counter-current for CnB. Secondly, a 2-bed formulation suffices since most PSA cycles in literature have at most 2 beds interacting at any given time.

The superstructure is designed to get the light product from the upper end (light

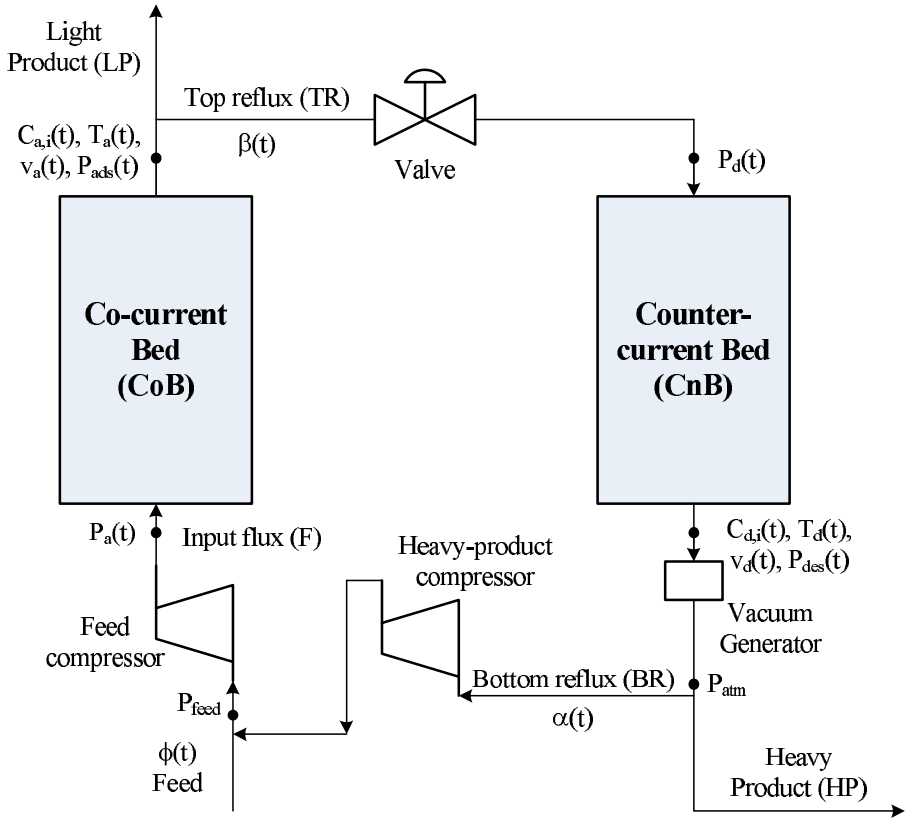

Figure 6.1: A 2-bed PSA Superstructure [3]

end) of CoB and heavy product from the lower end (heavy end) of CnB. The beds are connected with time dependent variables, top reflux $\beta(t)$ and a bottom reflux stream, $\alpha(t)$. β(t) and α(t) determine the fraction of the light product and the heavy product streams that go in the top and the bottom reflux, respectively. In addition, the time dependent feed fraction, ϕ(t), determines the fraction of feed that is fed to CoB. The fuel gas at pressure P_{feed}, is further compressed, if required, and is then fed to the co-current bed. For CoB, pressure is specified at the light end by P_{ads}, while the pressure at the other end of the bed P_a is determined from the pressure drop in this bed. The velocity v_a, concentration for ith component $C_{a,i}$, and temperature T_a at the light end of the co-current bed are determined from the outlet flux. Similarly for CnB bed, pressure is specified at the heavy end by P_{des}, while $C_{d,i}$, T_d and v_d at this end are obtained from the output flux of this bed. P_d at the other end is obtained from the pressure drop in the counter-

current bed. The superstructure also incorporates compressors and valves to account for different pressure levels in the beds. The heavy reflux is compressed to the inlet pressure P_{feed} with the help of the heavy reflux compressor, while the feed and the heavy reflux together are then compressed by the feed compressor. Since the main focus of this work is to develop cycles that concentrate heavy product, a vacuum generator is also included in the superstructure to extract the strongly-adsorbed component with a high recovery. Furthermore, this helps to avoid discrete variables and uses only continuous variables for the optimization problem

The superstructure generates a wide variety of different operating steps of a PSA process by varying the control variables α(t), β(t), ϕ(t), P_{ads}(t) and P_{des}(t). Since we have two beds in the superstructure, half of the operating steps are realized from the co-current bed CoB and half from the counter-current bed CnB. In an actual 2-bed PSA unit, the co-current bed after performing its steps will follow the steps of the counter-current bed and vice-versa. However, in the mathematical framework, this is realized by giving final conditions of the co-current bed as the initial conditions for the counter-current bed and vice-versa, thus modeling the true 2-bed behavior. This multi-bed approach is described in detail in Jiang et al. [39].

Although the optimal PSA cycle configurations based on the 2-bed superstructure are constructed from the solution of the control profiles of α(t), β(t), ϕ(t), P_{ads}(t) and P_{des}(t), multi-bed cycles follow immediately from these solutions. These are generated by stacking the optimal 2-bed configurations in parallel to ensure a continuous product withdrawal or feed step happens at all points in time.

6.3 Optimization Problem

We obtain an optimal PSA configuration by solving an optimal control problem for a specified objective function. The optimal profiles for the control variables α(t), β(t), ϕ(t), P_{ads}(t) and P_{des}(t) along with other decision variables, individual step times and bed dimensions, can be obtained by solving the following optimization problem:

$$
\begin{aligned}
\min \quad & \Phi(z(x,t_f), u(t_f), z_0, p) \\
\text{s.t.} \quad & F\left(\frac{\partial z}{\partial t}, \frac{\partial z}{\partial x}, z(x,t), u(t), z_0, p\right) = 0 \\
& z_{CoB}(x,0) = z_{CnB}(x,t_f) \\
& z_{CnB}(x,0) = z_{CoB}(x,t_f) \\
& g(z(x,t), u(t), p) \leq 0 \\
& 0 \leq (\alpha(t), \beta(t), \phi(t)) \leq 1 \\
& b_L \leq (P_{ads}(t), P_{des}(t), p) \leq b_U \\
& u(t) = [\alpha(t), \beta(t), \phi(t), P_{ads}(t), P_{des}(t)]
\end{aligned}
$$

Here Φ is the objective function which can be overall power consumption, product purity or recovery. $F(.)$ corresponds to the partial different equation based model for the PSA system with initial conditions z_0. The second and third equality constraints correspond to the cyclic steady state (CSS) condition which is enforced by giving final conditions of the co-current bed as the initial condition for the counter-current bed and vice versa. Inequalities constraints g can correspond to maintain the product purity or recovery. Finally, we impose bounds on the control variables, $u(t)$ and the design parameter, p. The control variables α(t), β(t) and ϕ(t) are fractions bounded between 0 and 1. Other control variables, P_{ads}(t) and P_{des}(t), and decision variables p are bounded between their respective bounds b_L and b_U.

6.4 PSA model and Solution Strategies

6.4.1 Model Equations

The conservation equations and models for the isotherm, equations of state, equilibrium, thermodynamic and transport properties, bed connection equations in the superstructure are listed in Table 6.1. The model is based on the following assumptions:

1. Gases follows the ideal gas behavior.
2. There are no radial variations in temperature, pressure and concentrations of the gases in the solid and the gas phase.
3. Pressure drop along the bed is calculated by the Ergun equation.
4. Adsorption equilibrium is described by dual site Langmuir isotherm.
5. The adsorption rate is approximated by the linear driving force (LDF) expression.

. The following equations are used to evaluate the performance variables,

$$\text{purity}_L = \frac{\int (1-\beta(t)) v_a(t) C_{a,L}(t)\, \mathrm{d}t}{\int (1-\beta(t)) v_a(t) \sum_i C_{a,i}(t)\, \mathrm{d}t} \tag{6.1a}$$

$$\text{purity}_H = \frac{\int (1-\alpha(t))(-v_d(t)) C_{d,H}(t)\, \mathrm{d}t}{\int (1-\alpha(t))(-v_d(t)) \sum_i C_{d,i}(t)\, \mathrm{d}t} \tag{6.1b}$$

$$\text{recovery}_L = \frac{\int (1-\beta(t)) v_a(t) C_{a,L}(t)\, \mathrm{d}t}{\mathrm{Q}_{feed,L}} \tag{6.1c}$$

$$\text{recovery}_H = \frac{\int (1-\alpha(t))(-v_d(t)) C_{d,H}(t)\, \mathrm{d}t}{\mathrm{Q}_{feed,H}} \tag{6.1d}$$

$$\mathrm{Q}_{feed,i} = \int \phi(t) v_{feed} C_{feed,i}\, \mathrm{d}t \qquad i \in \{\mathrm{L},\, \mathrm{H}\} \tag{6.1e}$$

Here Q_{feed} is the feed flux. The total power consumption, given by the following equations, is the sum of the work done by the compressors and the vacuum generator.

$$\begin{aligned} \mathrm{W}_{total} &= \int \frac{\gamma R T_d}{\gamma - 1} \left[\frac{\sum_i \left(\phi(t) v_{feed} C_{feed,i} + \alpha(t)(-v_d) C_{d,i}\right)}{\eta_c} \left(\left(\frac{P_a}{P_{feed}} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right) \right. \\ &\quad + \frac{\sum_i \alpha(t)(-v_d) C_{d,i}}{\eta_h} \left(\min \left\{ \left(\frac{P_{feed}}{P_{atm}} \right)^{\frac{\gamma-1}{\gamma}}, \left(\frac{P_{feed}}{P_{des}} \right)^{\frac{\gamma-1}{\gamma}} \right\} - 1 \right) \\ &\quad \left. + \frac{\sum_i (-v_d) C_{d,i}}{\eta_v} \max \left\{ 0, \left(\left(\frac{P_{atm}}{P_{des}} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right) \right\} \right] \\ &\quad + \frac{\gamma}{\gamma - 1} \frac{\phi(t) v_{feed} P_{feed}}{\eta_{fg}} \left(\left(\frac{P_{feed}}{P_{inlet}} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right) \mathrm{d}t \end{aligned} \tag{6.2a}$$

$$\text{Power/mole } \mathrm{CO_2} = \frac{\mathrm{W}_{total}}{\int (1 - \alpha(t)) v_d(t) C_{d,H}(t) \, \mathrm{d}t} \tag{6.2b}$$

Here, the *max* function ensures that the work done by the vacuum generator is zero when P_{des} is more than the atmospheric pressure P_{atm}. Similarly, since the vacuum generator discharges heavy reflux at P_{atm}, the *min* function ensures a proper upstream pressure for the heavy product compressor. Since *min* and *max* functions introduce non-differentiability, the following smoothing approximations are adopted [6]. A value of 0.01 is used for ϵ in the following equations.

$$\min(f_1(x), f_2(x)) = f_1(x) - \max(0, f_1(x) - f_2(x)) \tag{6.3a}$$

$$\max(0, f(x)) = 0.5 \left(f(x) + \sqrt{f(x)^2 + \epsilon^2} \right) \tag{6.3b}$$

6.4.2 Solution Methodology

In a previous study [3], (6.1) is solved using a complete discretization approach where the PDAEs are turned into a set of algebraic equations by discretizing the state and the

Table 6.1: PSA Model Equations [3]

Component mass balance

$$\epsilon_b \frac{\partial C_i}{\partial t} + (1-\epsilon_b)\rho_s \frac{\partial q_i}{\partial t} + \frac{\partial (vC_i)}{\partial x} = 0 \qquad i \in \{\text{L, H}\} \tag{6.4}$$

LDF equation

$$\frac{\partial q_i}{\partial t} = k_i(q_i^* - q_i) \qquad i \in \{\text{L, H}\} \tag{6.5}$$

Energy balance

$$\left(\epsilon_t \sum_i C_i(C_{pg}^i - R) + \rho_s C_{ps}\right)\frac{\partial T}{\partial t} - \rho_s \sum_i \Delta H_i^{ads} \frac{\partial q_i}{\partial t} + \frac{\partial (vh)}{\partial x} + U_A(T - T_w) = 0 \tag{6.6}$$

$$C_{pg}^i = a_c^i + b_c^i T + c_c^i T^2 + d_c^i T^3 \qquad i \in \{\text{L, H}\}$$

$$h = \sum_i \left(C_i \int C_{pg}^i \, \mathrm{d}T\right)$$

Dual-site Langmuir Isotherm

$$q_i^* = \frac{q_{1i}^s b_{1i} C_i RT}{1 + \sum_j b_{1j} C_j RT} + \frac{q_{2i}^s b_{2i} C_i RT}{1 + \sum_j b_{2j} C_j RT} \qquad i \in \{\text{L, H}\} \tag{6.7}$$

$$q_{mi}^s = k_{mi}^1 + k_{mi}^2 T \qquad b_{mi} = k_{mi}^3 exp(\frac{k_{mi}^4}{T}) \qquad i \in \{\text{L, H}\} \qquad m = 1, 2$$

Ergun equation

$$-\frac{\partial P}{\partial x} = \frac{150\mu(1-\epsilon_b)^2}{d_p^2 \epsilon_b^3} v + \frac{1.75}{d_p}\left(\frac{1-\epsilon_b}{\epsilon_b^3}\right)\left(\sum_i M_w^i C_i\right) v|v| \tag{6.8}$$

Ideal gas equation

$$P = RT \sum_i C_i \tag{6.9}$$

Bed connection equations (see Figure 6.1)

$$F_i(t) = \phi(t) v_{feed} C_{feed,i} + \alpha(t)(-v_d(t)) C_{i,d}(t) \qquad i \in \{\text{L, H}\} \tag{6.10}$$

$$TR_i(t) = \beta(t) v_a(t) C_{a,i}(t) \qquad i \in \{\text{L, H}\} \tag{6.11}$$

$$LP_i(t) = (1-\beta(t)) v_a(t) C_{a,i}(t) \qquad i \in \{\text{L, H}\} \tag{6.12}$$

$$BR_i(t) = \alpha(t)(-v_d(t)) C_{d,i}(t) \qquad i \in \{\text{L, H}\} \tag{6.13}$$

$$HP_i(t) = (1-\alpha(t))(-v_d(t)) C_{d,i}(t) \qquad i \in \{\text{L, H}\} \tag{6.14}$$

control variables both in space and time. As a result, the PDAE-constrained optimal control problem (Equation (6.1)) gets converted into a large-scale nonlinear programming (NLP) problem. The authors use an efficient NLP solver, IPOPT [73] to solve the resulting large scale NLP problem. However, a complete discretization scheme can lead to a very large set of algebraic equations, which can be expensive to solve. Hence, the work was limited to only 10 spatial nodes for the optimization problem and the accuracy of the solution is checked by performing dynamic simulations in MATLAB [1] at the optimal values of the decision variables obtained from IPOPT, and further compare the profiles and the performance variables.

The drawbacks of the complete-discretization approach is that if the concentration profiles are steep, a finer finite element discretization is required, resulting in creating large number of variables in the optimization problem. The number may blow up when the superstructure contains more beds and multicomponent feed mixtures. Ko et al. [43] reported that in their optimization scheme of PSA processes using a complete-discretization approach, the solution is dependent on the number of spatial nodes. In order to overcome these drawbacks, we use a partial discretization approach (see Figure 2.1) to solve (6.1). In this approach, the PDAEs are discretized only in spatial domain resulting in a set of differential-algebraic equations (DAEs). Here, the solution of the optimization problem is decoupled from the solution of the embedded dynamic system thus exploiting the full advantages of the state-of-the-art integration and NLP tools. As a result, the partially discretized PDAEs are integrated outside the optimization problem using sophisticated DAE solvers which are able to capture the steep temporal adsorption fronts with high accuracy and also the optimization problem is relatively smaller in size.

In this work, the PDAE model is discretized in the spatial domain using a conservative

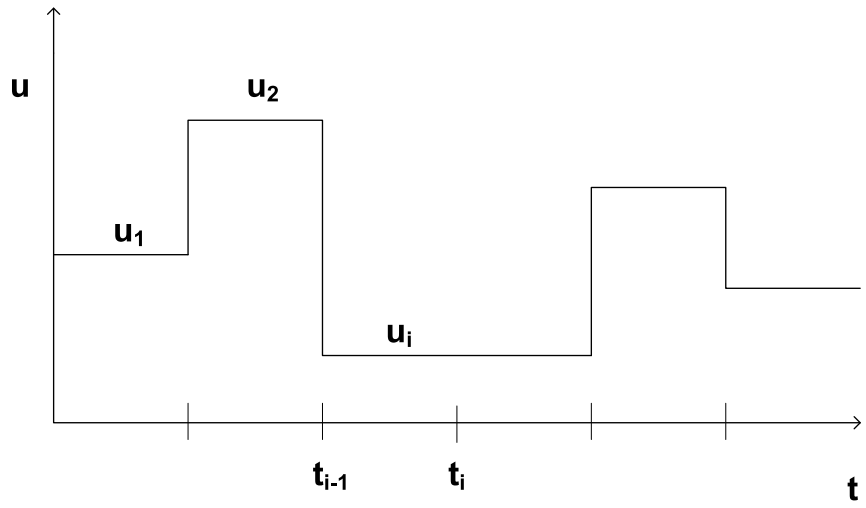

Figure 6.2: Discretization of control variable

finite volume method which results in a system of differential-algebraic equations (DAEs). The modified second-order Van Leer flux limiter[46] is used to mitigate numerical error and avoid physically unrealistic smearing oscillations near the adsorption fronts. The discretized model is solved repeatedly by varying the initial and boundary conditions that define the various steps in the 2-bed PSA superstructure. We use a reduced Sequential Quadratic Programming (rSQP [10]) algorithm as Non-linear programming (NLP) solver. The solver searches for both the optimal operating parameters and concentration profiles simultaneously. The control variables α(t), β(t), ϕ(t), P_{ads}(t), and P_{des}(t) are discretized in a piecewise constant manner and added as parameters in the optimization problem. The discretized concentration nodes at the beginning of a cycle are treated as part of decision variables. For the computation of the first derivatives with respect to the decision variables, sensitivity equations are integrated along with DAEs by using a variable-step variable-order integrator, CVODES [59] and letting the integrator adapt the step-size in the temporal domain. Since the DAEs are integrated at every NLP iteration, the accuracy test of the results at the solution used in [3] can be skipped.

As mentioned earlier, the control variables $u(t)$ are discretized in a piecewise constant

manner (Figure 6.2). Some of the cycles reported [3] have steps with a very short duration when feed is supplied. Such small step times can lead to a large number of parallel beds in the realization of continuous operation, which is expensive to implement in practice. To avoid these kinds of steps, the step times can be constrained to avoid an overly complicated cycle. To handle this, in this work we use a slot-based formulation in which we allow the step times to be integer multiples of each other.

6.5 Case Studies

In the case studies, we consider the feed to be a syngas mixture having 55% H_2 and 45% CO_2, coming at a temperature of 310 K after a single shift conversion in an IGCC [48]. The feed mixture also consists of negligible amounts of CO, CH_4, Ar and N_2, besides H_2 and CO_2. However, hydrogen and carbon dioxide together constitute around 97-99% of the mixture [48]. Therefore, we consider a binary feed mixture for the case studies. The bed length is fixed and is assumed to be 10 meters. For all the case studies, we also assume an efficiency of 72% for all compressors and the vacuum generator in the superstructure. Activated carbon is chosen as the adsorbent, especially to extract CO_2. The adsorbent properties and other model parameters are taken from [39, 3]. In this work, we consider two different case studies. The first case involves superstructure optimization to obtain an optimal PSA configuration which maximizes CO_2 recovery for a given lower bound on both CO_2 and H_2 purity, while the second case involves generating optimal cycle that minimizes overall power consumption for a given lower bound on CO_2 purity and recovery.

6.5.1 Case I: Cycle synthesis to maximize CO_2 recovery

In this case study, we solve the discretized form of the optimal control problem (6.1) to obtain an optimal cycle that maximizes CO_2 recovery subject to a lower bound of 90% on both H_2 and CO_2 purity. Optimal control problem after spatial discretization is given below

$$\begin{aligned} \max \quad & CO_2 \text{ recovery} \\ \text{s.t.} \quad & (\tfrac{dz_m}{dt} = f_m(z_m, t) \quad m = 1, ..10N_{dis}, \text{ solved implicitly.}) \\ & c_m^{ss} = 0 \quad m = 1, ...10N_{dis} \\ & H_2 \text{ Purity} \geq 0.9 \\ & CO_2 \text{ Purity} \geq 0.9 \\ & LB \leq u_i, t_{slot} \leq UB \quad i = 1, ...N_{slot} \end{aligned} \tag{6.15}$$

where N_{dis} is the number of finite volume and N_{slot} is number of equally spaced control discretization elements. In this case study, we choose $N_{dis} = 20$ and $N_{slot} = 10$. The optimization problem is solved using rSQP solver. At the solution, the cycle time is given by $t_{cycle} = t_{slot} N_{slot}$.

The optimal profiles for the control variables α(t), β(t), ϕ(t), P_{ads}(t) and P_{des}(t) are shown in Figure 6.3. They are drawn against the cycle time normalized between 0 and 1. These profiles translate into a 2-bed 8-step cycle as illustrated in Figure 6.4. The cycle starts with $\alpha = 1$, $\beta = 0$ and $\phi = 1$ which suggests that there is heavy reflux stream, no light reflux and feed streams. The CoB undergoes the heavy reflux and the CnB undergoes desorption at 200 kPa. The heavy reflux enriches the CO_2 concentrations towards the heavy end of CoB with the CO_2 coming from the CnB. In the next two steps,

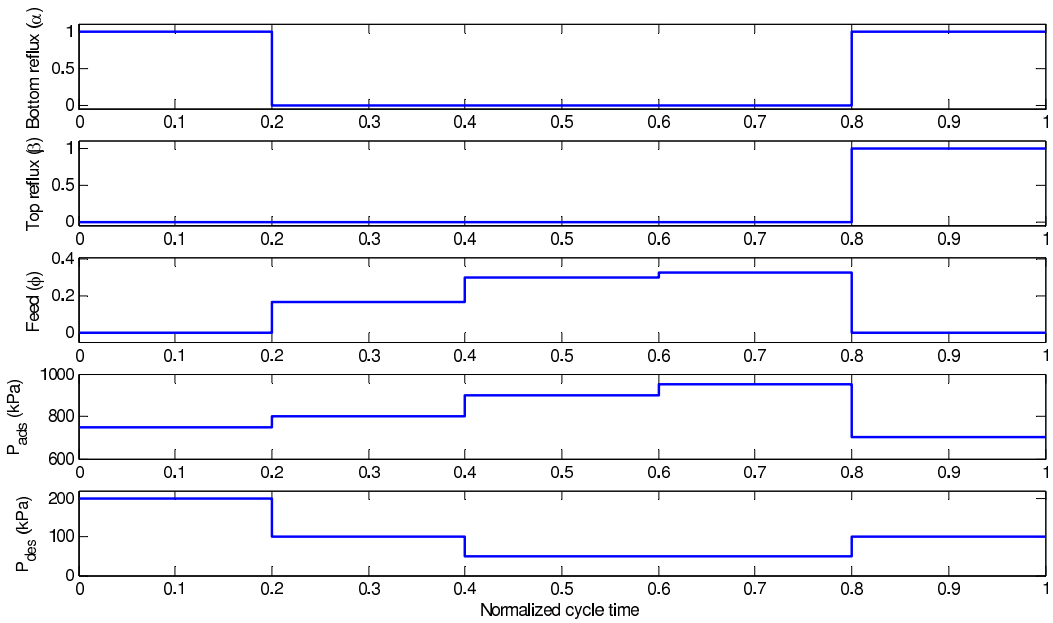

Figure 6.3: Case I: Optimal control profiles

the CoB undergoes feed pressurization with adsorption as can be seen from the pressure profiles and the CnB undergoes desorption at atmospheric and vacuum. During these two steps, pure H_2 and CO_2 are collected simultaneously from CoB and CnB respectively. In the last step, there is no feed to CoB and a recirculation of components occurs within the system. We call this the total reflux step. During this step, the heavy end of the CoB gets enriched with CO_2 and the light end of CnB with H_2 which results in high purity of both the components. Figure 6.5 shows the gas-phase CO_2 concentration profiles for the 4 steps of the CoB. As can be seen, the heavy end of the CoB gets enriched with CO_2 and the cycle stops just before CO_2 breaks through the CoB.

The optimization results at the solution are given in Table 6.2. The problem has 200 differential equations and 21 degrees of freedom which took around 2 hours to converge on an Intel 2.8 GHz machine with 8 GB RAM. An optimum CO_2 recovery of 94% at a purity of 90% was obtained. Also, a reasonably high hydrogen purity of 95% and a recovery of 91% was obtained simultaneously with a power consumption of 135.53 kWh/tonne CO_2

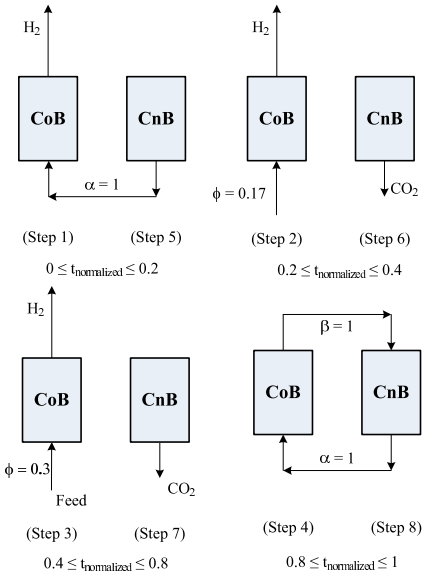

Figure 6.4: Case I: Optimal configuration

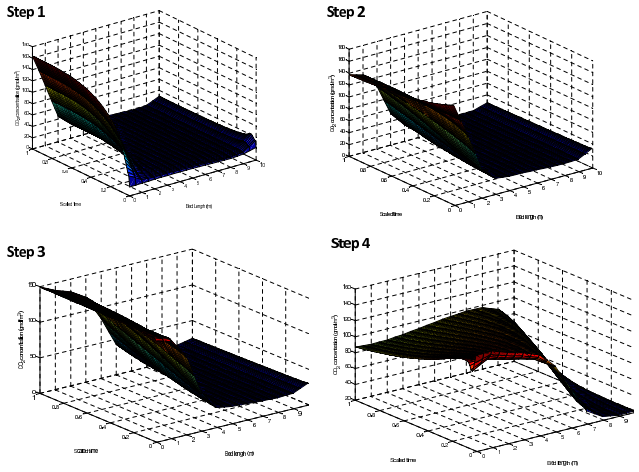

Figure 6.5: Case I: Gas-phase CO_2 concentration profiles of CoB

captured.

Figure 6.6 shows a trade-off curve between CO_2 purity and recovery. The curve is constructed by varying the lower bound on CO_2 purity and solving the superstructure NLP repeatedly. The curve shows that a high CO_2 recovery can be obtained at the expense of low CO_2 purity for the given process conditions and with activated carbon as the adsorbent.

Table 6.2: Case I: Optimization results

No. of state variables	200
Degrees of freedom	51
CPU time	2 h
Optimal cycle time	150.48 sec
CO_2 recovery	93.91%
CO_2 purity	90%
H_2 recovery	91.46%
H_2 purity	94.84%
Feed flux	52.65 gmol m^{-2} h^{-1}
Power consumption	135.53 kWh/tonne CO_2 captured

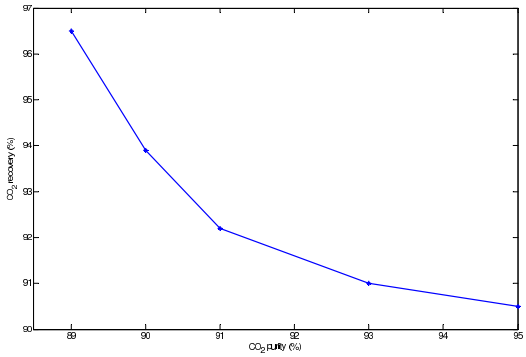

Figure 6.6: Case I: Purity-recovery trade-off curve

6.5.2 Case II: Cycle synthesis to minimize power consumption

Here, we minimize the power consumption subject to a lower bound of 90% on CO_2 purity and recovery. The optimal control problem after spatial discretization is given below

$$\begin{aligned} \min \quad & \text{Power/mole of } CO_2 \\ \text{s.t.} \quad & (\tfrac{dz_m}{dt} = f_m(z_m, t) \quad m = 1, ..10N_{dis}, \text{ solved implicitly.}) \\ & c_m^{ss} = 0 \quad m = 1, ...10N_{dis} \\ & CO_2 \text{ recovery} \geq 0.9 \\ & CO_2 \text{ purity} \geq 0.9 \\ & LB \leq u_i, t_{slot} \leq UB \quad i = 1, ...N_{slot} \end{aligned} \tag{6.16}$$

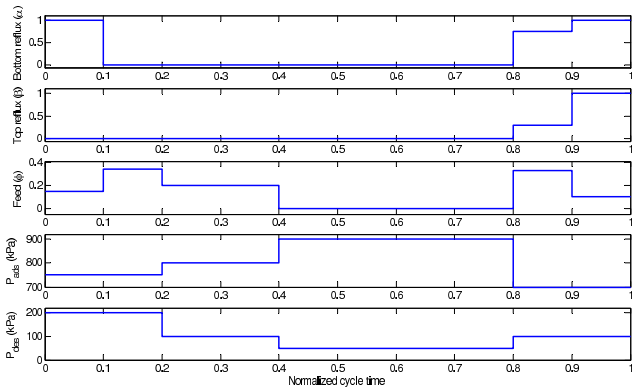

Figure 6.7: Case II: Optimal control profiles

As in the previous case study, we choose $N_{dis} = 20$ and $N_{slot} = 10$ and solve the optimization problem using rSQP solver.

The optimal control profiles obtained for α(t), β(t), ϕ(t), P_{ads}(t) and P_{des}(t) are shown in Figure 6.7. These profiles translate to a 2-bed 10-step cycle as shown in Figure 6.8. In this cycle, the CoB starts with adsorption at feed pressure with heavy reflux and the CnB undergoes counter-current depressurization. During this step (Step 2 and 7), CO_2 from the CnB enriches the heavy end of CoB. In the next step, CoB undergoes feed pressurization along with adsorption and the CnB undergoes desorption at atmospheric pressure. In Step 8, more CO_2 is removed with CnB undergoing desorption at vacuum. There is continuous removal of both pure H_2 and CO_2 during steps 2 and 7 and steps 3 and 8. In steps 4 and 9, the pressure in the CoB decreases to feed pressure and CnB increases to atmospheric pressure. This step is called the pressure equalization step, which is responsible for decreasing the power consumption. The last step is more like the total reflux step, when the heavy end of CoB gets enriched with CO_2 and the light end of CnB gets enriched with H_2. As a result, during the next half of the cycle high purity of both components are obtained. Figure 6.9 shows the gas-phase CO_2 concentration profiles for the 5 steps of the CoB. As can be seen, CO_2 gets enriched at the heavy end of CoB and

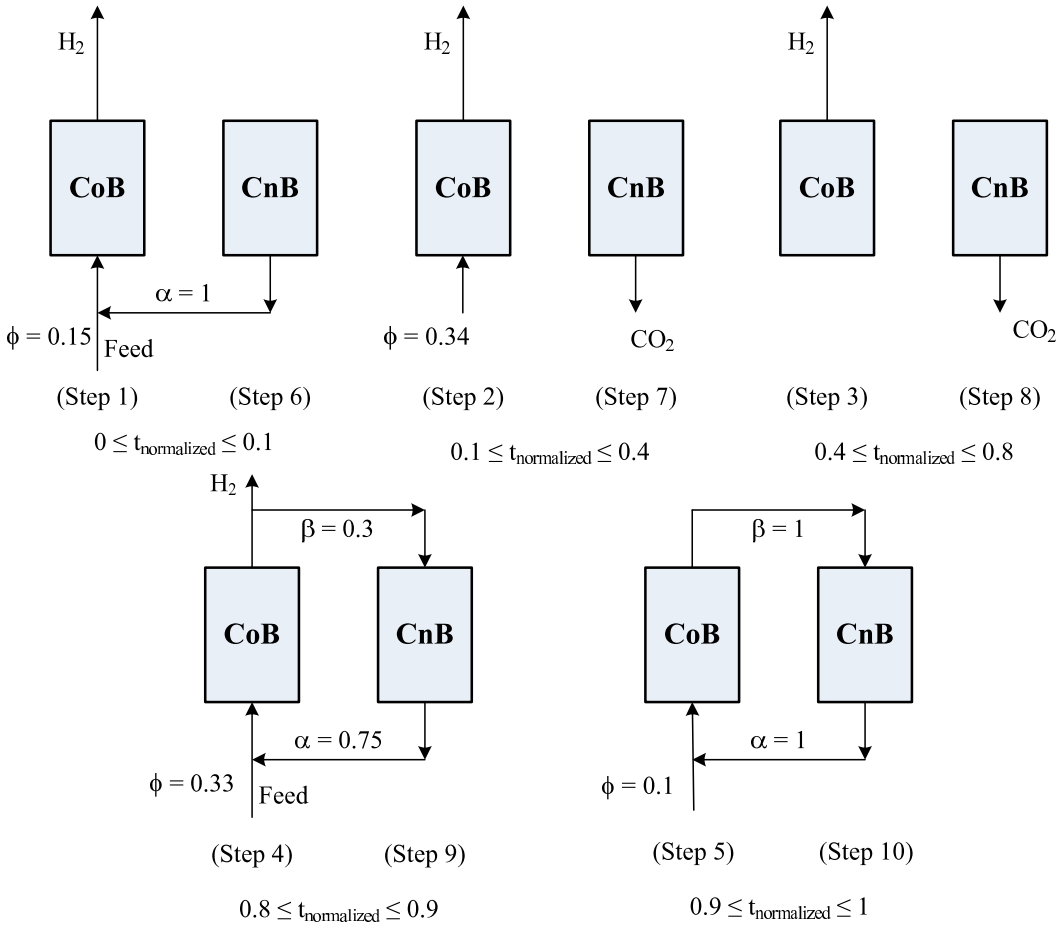

Figure 6.8: Case II: Optimal configuration

in step 5 CO_2 almost breaks through the bed when the cycle is stopped. After step 5, the cycle is repeated with the CoB undergoing the steps of CnB and vice-versa.

The optimization results at the solution are given in Table 6.3. As in the previous case study, this problem also has 200 differential equations and 21 degrees of freedom which took 8.5 hours to converge on an Intel 2.8 GHz machine with 8 GB RAM. An optimum CO_2 recovery of 92% at a purity of 90% was obtained. Also, a reasonably high hydrogen purity of 93.4% and a recovery of 92% was obtained simultaneously with a lower power consumption of 85.79 kWh/tonne CO_2 captured. The reason for lower power consumption compared to the previous case study is mainly because of the pressure equalization step.

Figure 6.10 shows a trade-off curve between CO_2 purity and power consumption per

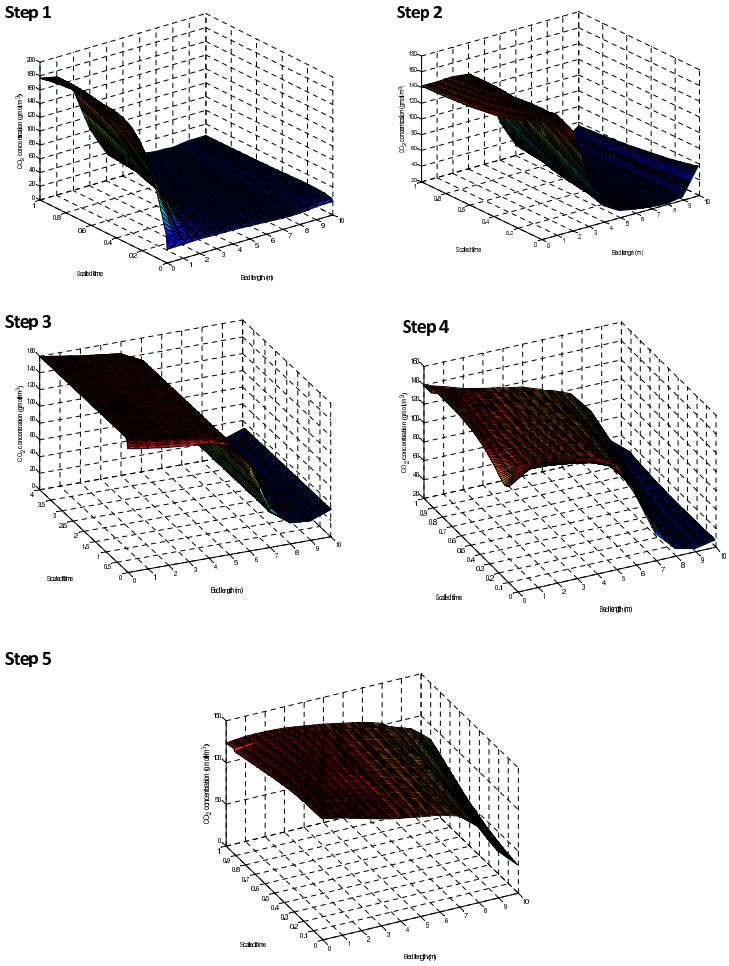

Figure 6.9: Case II: Gas-phase CO_2 concentration profiles of CoB

tonne of CO_2 captured. The curve is constructed by varying the lower bound on CO_2 purity and solving the superstructure NLP repeatedly. The curve shows that a high CO_2 purity can be obtained at the expense of low CO_2 purity for the given process conditions and with activated carbon as the adsorbent. The power consumption for obtaining CO_2 at a high purity of 96 % is lower than the power required for the optimal cycle in the max. recovery case study.

Table 6.3: Case II: Optimization results

No. of state variables	200
Degrees of freedom	51
CPU time	8.5 h
Optimal cycle time	225 sec
CO_2 recovery	92.08%
CO_2 purity	90%
H_2 recovery	91.6%
H_2 purity	93.4%
Feed flux	43.46 gmol m^{-2} h^{-1}
Power consumption	85.79 kWh/tonne CO_2 captured

6.6 Concluding Remarks

Most conventional PSA cycles for high purity CO_2 capture are designed to recover only the light product, H_2 at an extremely high purity, and consider the heavy product, CO_2 as a waste stream. In this work, we propose a systematic optimization-based methodology to design PSA cycles which simultaneously produce H_2 and CO_2 at a high purity. We formulate a 2-bed PSA superstructure which can predict a wide variety of different PSA operating steps. Different operating steps are realized by manipulating the bed connections with the help of time dependent control variables such as fractions of feed stream, top and bottom reflux, and bed pressures.

This approach is illustrated for two different case studies of pre-combustion CO_2 capture using only activated carbon as the sorbent. The first case study deals with obtaining optimal PSA cycle which maximizes CO_2 recovery by maintaining a desired purity level of CO_2 and H_2 purity. Superstructure optimization for this case results in a 2-bed 8-step cycle which can produce both H_2 and CO_2 at a substantially high purity of 95% and 90%, respectively with a significantly high CO_2 recovery of 94% is achieved. The second case study deals with minimizing the power consumption by maintaining a desired purity and

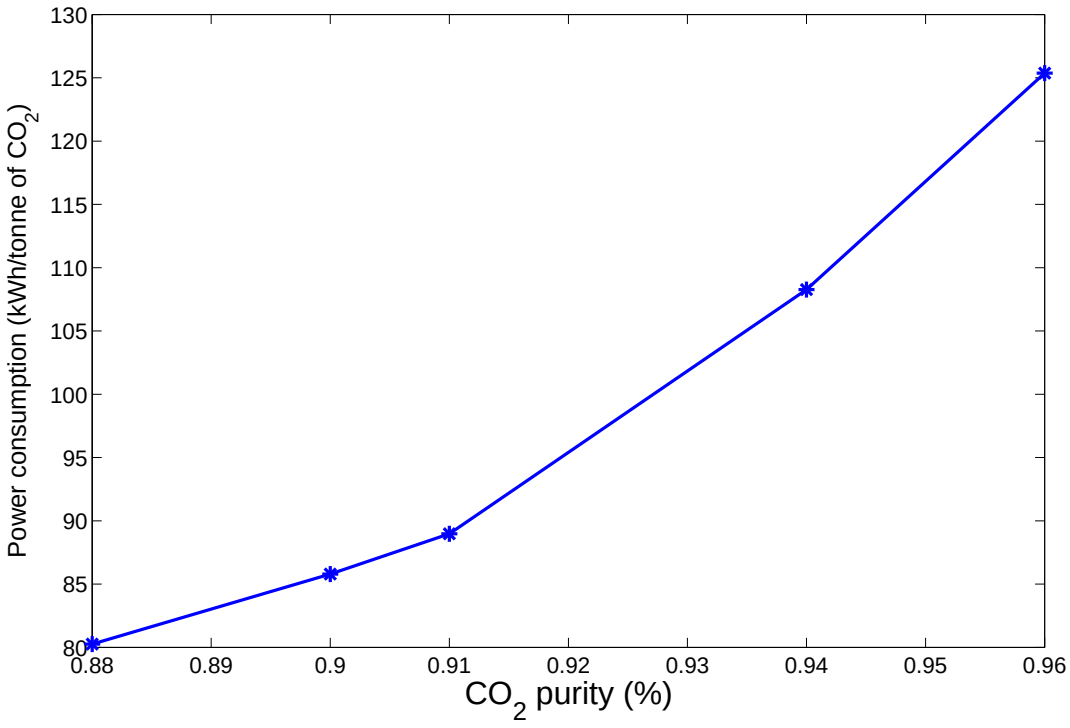

Figure 6.10: Case II: Purity-power consumption trade-off curve

recovery level of CO_2. This case study results in a 2-bed 10-step cycle which can produce CO_2 at a purity of 90% and a recovery of 92% with a low power consumption of 85.79 kWh/tonne CO_2 captured.

Our superstructure based methodology, is quite generic and can be extended to many other PSA applications. Moreover, the superstructure can also be used to evaluate different kinds of adsorbents for the same feedstock and process conditions.

Chapter 7

Conclusions and Future Work

Synopsis

Periodic Adsorption Processes have gained increasing commercial acceptance as an efficient separation technique for a wide range of applications. However, despite a vast growth in the practical application of PAPs, the design and optimization of PAPs still largely remains computationally challenging with the current methods. In order to address this problem, this work focuses on developing new optimization algorithms based on trust-region methods. Moreover, this thesis also presents a systematic methodology using the superstructure approach to synthesize novel PSA cycles for CO_2 capture. This chapter summarizes all these contributions and directions for future work.

7.1 Thesis Summary and Contributions

The main motivation of this thesis arises due to the optimization problems related to Periodic Adsorption Processes (PAPs). Although there have been many sophisticated optimization strategies that have been developed and applied for design and operation of PAP systems, these methods have limitations with respect to convergence, computation time and problem size. Even the most efficient and reliable simultaneous tailored approach for the optimization of moderately sized PAP problems suffers from large computation times arising from the direct sensitivity calculations required to form the constraint Jacobian. In order to overcome these limitations, this thesis focuses on developing faster algorithms that do not require explicit calculation of the Jacobian through sensitivity calculations. Furthermore, we also explore a superstructure approach for optimal design of PSA cycles for pre-combustion CO_2 capture. We discuss our contributions separately for both these works in the subsequent sections.

Inexact NLP algorithms

In Chapter 4, we develop a SQP trust-region algorithm for the solution of minimization problems with both nonlinear equality and inequality constraints. Instead of forming and factoring the dense constraint Jacobian, this algorithm approximates the Jacobian of equality constraints with a specialized quasi-Newton method. Moreover, the algorithm uses exact second order information and hence results in faster convergence. This algorithm is tested on small scale PAP applications which resulted in a five-fold reduction in computation time compared to exact methods.

In Chapter 5, we develop a composite-step trust region methods with barrier functions for the solution of minimization problems with nonlinear inequality constraints. The re-

sulting algorithm also does not require the computation of exact Jacobians; only Jacobian vector products are used along with approximate Jacobian matrices. As demonstrated on small numerical examples, this feature has significant potential benefits for problems where Jacobian calculations are expensive.

Specific contributions for this part of the dissertation:

- Developed SQP and interior point trust region algorithms for solving general optimization problems with dense constraint Jacobians.
- Quasi-Newton approximation of the Jacobian avoids the high computation cost associated with the calculation of Jacobian.
- Implemented the algorithm in C programming language and interfaced it with standard integrators and automatic differentiation tools
- Evaluation of exact first-order and second-order information calculation in the form of Jacobian-vector and Hessian-vector products using adjoint sensitivities and Automatic Differentiation. As a result, the integration of direct sensitivity equations to form the dense Jacobian is avoided.
- The results presented in Chapters 4 and 5 are promising and encouraging to use the algorithms for large scale problems.

PSA Superstructure

In Chapter 6, we present a new PSA superstructure idea to design new PSA cycles for pre-combustion CO_2 capture. Different operating steps are realized by varying the time-dependent control variables that define the interconnections between the beds. An optimal control problem is solved using the partial discretization approach to obtain the optimal

profile of the control variables and design parameters. To avoid steps with short duration, we use a slot-based formulation for discretizing the control variables.

This approach is illustrated for two different case studies. When CO_2 recovery is maximized, superstructure optimization results in a 2-bed 8-step cycle which can produce both H_2 and CO_2 at a substantially high purity of 95% and 90% respectively, with a significantly high CO_2 recovery of 94%. When power is minimized, superstructure optimization results in a 2-bed 10-step cycle which can produce CO_2 at a purity of 90% and a recovery of 92% with a low power consumption of 85.79 kWh/tonne CO_2 captured.

To summarize our contributions for this part of the dissertation:

- PSA cycle synthesis for pre-combustion CO_2 capture. The new cycles designed obtain CO_2 and H_2 at high purity and recovery with low power consumption. This superstructure idea suggests novel operating steps for high purity CO_2 capture.
- The proposed superstructure framework is quite generic and can be extended to many other PSA applications. Besides developing optimal cycles, the framework can be used to evaluate different kinds of adsorbents for the same feedstock and process conditions.

7.2 Directions for Future Work

Algorithmic Improvements

- **Better recovery strategies**

  In the inexact algorithms, we evaluate an exact Jacobian when the tests on the quality of Jacobian fail. However for large-scale problems, the computation of a single Jacobian may take as long as 5 h or more. These time-consuming evaluations

of Jacobian and null-space matrices may be a bottle-neck when the process is to be optimized in real time. Therefore, better recovery strategies can be developed to avoid forming the Jacobian when the tests fail. For example, we aim to exploit the special feature of the two-sided rank-one update. The exact Jacobian, $A(x_k)$ can be reconstructed with up to m TR1 updates along independent vectors for a fixed iterate x_k [75]. The computational cost involved is relatively cheap since the update requires only matrix-vector and vector-matrix products which can be obtained efficiently using sensitivities and Automatic Differentiation.

- **Extending interior point algorithm to general optimization problems**

  The inexact interior point method developed in this work can handle only inequality constrained problems. In the future, the algorithm can be extended to solve general optimization problems and development of the corresponding theory. Furthermore, algorithmic parameters should be well-tuned for better performance.

PSA Superstructure

- The superstructure can be modified to include products tanks and flow valves. Product tanks can help obtain operating steps that involve a pure product purge, which is not possible with the current superstructure.

- In addition to incorporating valves in the superstructure, the PDE models have to be modified to handle dynamics in the valve equations. This is important because valves would naturally have limits on how fast they can be tuned.

- The partial discretization approach can be extended to applications that involve multi-component feed mixtures. Moreover, multiple layers of adsorbents can also

be incorporated in CoB and CnB for efficient separation [39].

- In the preliminary numerical tests using the inexact algorithm, we have considered small scale simulated moving bed and pressure swing adsorption processes for the separation of binary feed mixtures. With the proposed algorithm, significant savings in computational time are obtained due to fewer Jacobian evaluations during the optimization process. The proposed algorithms can be extended to solve optimization problems of large-Scale PSA processes with multi-component feed mixtures.

Bibliography

[1] *MATLAB User's Guide*. The Mathworks, Inc., 1994-2005.

[2] gPROMS User's Manual. PSE Limited, London, UK, 2000.

[3] A. Agarwal, L. T. Biegler, and S. E. Zitney. Superstructure-Based Optimal Synthesis of Pressure Swing Adsorption Cycles for Precombustion CO2 Capture. *Ind. Eng. Chem. Res.*, 49(11):5066–5079, 2010.

[4] A. Agarwal, L. T. Biegler, and S. E. Zitney. Superstructure-Based Optimal Synthesis of PSA cycles for Post-Combustion CO_2 Capture. *AIChE J.*, 56(7):1813–1828, 2010.

[5] N. Arora and L. T. Biegler. A Trust Region SQP Algorithm for Equality Constrained Parameter Estimation with Simple Parameter Bounds. *Comput. Optim. Appl.*, 28(1):51–86, 2004.

[6] S. Balakrishna and L. T. Biegler. Targeting Strategies for the Synthesis and Energy Integration of Nonisothermal Reactor Networks. *Ind. Eng. Chem. Res.*, 31(9):2152–2164, 1992.

[7] C. Benkmann. System for Treatment of Plural Crude Gases in Single Adsorption Plant. US Patent 4402712, 1983.

[8] J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming, Ad- vances in Design and Control*. SIAM, 2001.

[9] L. T. Biegler, L. Jiang, and V. G. Fox. Recent Advances in Simulation and Optimal Design of Pressure Swing Adsorption Systems. *Sep. Purif. Rev.*, 33(1):1–39, 2005.

[10] L. T. Biegler, J. Nocedal, and C. Schmid. A Reduced Hessian Method for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 5:314–347, 1995.

[11] K. F. Bloss, L. T. Biegler, and W. E. Schiesser. Dynamic Process Optimization through Adjoint Formulations and Constraint Aggregation. *Ind. Eng. Chem. Res.*, 38(2):421 – 432, 1999.

[12] A. E. Bryson and Y. C. Ho. *Applied Optimal Control*. John Wiley & Sons, 1975.

[13] R. Byrd, J. Gilbert, and J. Nocedal. A Trust Region Method based on Interior Point Techniques for Nonlinear Programming. *Math. Program.*, 89A:149–185, 2000.

[14] R. Byrd, R. B. Schnabel, and G. Schultz. A Trust Region Algorithm for Nonlinearly Constrained Optimization. *SIAM J. Numer. Anal.*, 24:1152–1170, 1987.

[15] R. H. Byrd, F. E. Curtis, and J. Nocedal. An Inexact SQP Method for Equality Constrained Optimization. *SIAM J. on Optimization*, 19(1):351–369, 2008.

[16] R. H. Byrd, M. E. Hribar, and J. Nocedal. An Interior Point Algorithm for Large Scale Nonlinear Programming. Technical report, Optimization Technology Center, Northwestern University, 1997.

[17] R. H. Byrd and J. Nocedal. An Analysis of Reduced Hessian Methods for Constrained Optimization. Technical report, University of Colorado at Boulder, Department of Computer Science, 1988.

[18] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint Sensitivity Analysis for Differential Algebraic Equations: The Adjoint DAE System and its Numerical Solution. *SIAM J. Sci. Comput*, 24:1076–1089, 2000.

[19] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Optim.*, 6:418, 1996.

[20] A. Conn, N. Gould, and P. Toint. Convergence of quasi-Newton Matrices Generated by the Symmetric Rank One Update. *Math. Program.*, 50A(2):177–196, 1991.

[21] A. R. Conn, N. I. M. Gould, and P. L. Toint. A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds. *SIAM J. Numer. Anal.*, 28:545–572, 1991.

[22] A. R. Conn, N. I. M. Gould, and P. L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Springer Series in Computational Mathematics 17, Berlin, 1992.

[23] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[24] F. Curtis, O. Schenk, and A. Wächter. An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations. SIAM Journal on Scientific Computing, 2010.

[25] M. Diehl and A. Walther. A test problem for Periodic Optimal Control problems. Technical report, TU Dresden, 2006.

[26] W. F. Feehery, J. E. Tolsma, and P. I. Barton. Efficient Sensitivity Analysis of Large-Scale Differential-Algebraic Systems. *Appl. Numer. Math.*, 25(1):41–54, 1997.

[27] R. Fletcher. *Numerical Experiments with an Exact Penalty l_1 Function Method*, pages 99–129. New York: Academic Press, 1981.

[28] R. Fletcher. *Practical Methods of Optimization*. Wiley Publications, 1987.

[29] D. M. Gay. Computing Optimal Locally Constrained Steps. *SIAM J. Sci. Stat. Comput.*, 2(2):186, 1981.

[30] D. M. Gay. A Trust-Region Approach to Linearly Constrained Optimization. *Numerical Analysis Proceedings (Dundee, 1983), D.F. Griffiths, Ed*, Springer-Verlag:72, 1983.

[31] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, 2000.

[32] A. Griewank, D. Juedes, and J. Utke. ADOL-C: A Package for the Automatic Differentiation of Algorithms written in C/C++. *ACM Trans. Math. Software*, 22:131–167, 1996.

[33] A. Griewank and A. Walther. On Constrained Optimization by Adjoint-based Quasi-Newton methods. *Optim. Methods. Softw.*, 17:869–889, 2002.

[34] A. Griewank, A. Walther, and M. Korzec. Maintaining factorized KKT Systems subject to Rank-one Updates of Hessians and Jacobians. *Optimization Methods and Software*, 22(2):279 – 295, 2007.

[35] M. Heinkenschloss and L. N. Vicente. Analysis of Inexact Trust-Region SQP Algorithms. *SIAM J. on Optimization*, 12(2):283–302, 2002.

[36] IEA/WEO. World Energy Outlook 2006. Technical report, International Energy Agency, Paris, France, 2006.

[37] J.-G. Jee, M.-B. Kim, and C.-H. Lee. Adsorption Characteristics of Hydrogen Mixtures in a Layered Bed: Binary, Ternary, and Five-Component Mixtures. *Ind. Eng. Chem. Res.*, 40(3):868–878, 2001.

[38] L. Jiang, L. T. Biegler, and V. G. Fox. Simulation and Optimization of Pressure-Swing Adsorption Systems for Air Separation. *AIChE Journal*, 49(5):1140–1157, 2003.

[39] L. Jiang, V. G. Fox, and L. T. Biegler. Simulation and Optimal Design of Multiple-Bed Pressure Swing Adsorption Systems. *AIChE Journal*, 50(11):2904–2917, 2004.

[40] S. Kameswaran and L. T. Biegler. Convergence Rates for Direct Transcription of Optimal Control Problems using Collocation at Radau points. *Comput. Optim. Appl.*, 41(1):81–126, 2008.

[41] Y. Kawajiri and L. T. Biegler. Optimization Strategies for Simulated Moving Bed and PowerFeed Processes. *AIChE Journal*, 52(4):1343–1350, 2006.

[42] D. Ko and I. L. Moon. Multiobjective Optimization of Cyclic Adsorption Processes. *Ind. Eng. Chem. Res.*, 41(1):93–104, 2002.

[43] D. Ko, R. Siriwardane, and L. T. Biegler. Optimization of Pressure Swing Adsorption Process using Zeolite 13X for CO_2 Sequestration. *Ind. Eng. Chem. Res.*, 42(2):339–348, 2003.

[44] R. Kumar, V. G. Fox, D. Hartzog, R. E. Larson, C. Y. C., P. Houghton, and T. Naheiri. A Versatile Process Simulator for Adsorptive Separations. *Chem. Eng. Sci.*, 49:3115–3125, 1994.

[45] M. Lalee, J. Nocedal, and T. Plantenga. On The Implementation Of An Algorithm For Large-Scale Equality Constrained Optimization. *SIAM Journal on Optimization*, 8:682–706, 1998.

[46] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser, 1992.

[47] S. Li, L. Petzold, and W. Zhu. Sensitivity Analysis of DifferentialAlgebraic equations: A Comparison of Methods on a Special Problem. *Applied Numerical Mathematics*, 32:161–174, 2000.

[48] NETL/DOE. The Cost and Performance Baseline for Fossil Energy Power Plants study, Volume 1: Bituminous Coal and Natural Gas to Electricity. Technical report, National Energy Technology Laboratory, Department of Energy, USA, May, 2007.

[49] D. Nikolič, A. Giovanoglou, M. C. Georgiadis, and E. S. Kikkinides. Generic Modeling Framework for Gas Separations Using Multibed Pressure Swing Adsorption Processes. *Ind. Eng. Chem. Res.*, 47(9):3156–3169, 2008.

[50] S. Nilchan and C. C. Pantelides. On the Optimisation of Periodic Adsorption Processes. *Adsorption*, 4(2):113–147, 1998.

[51] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series, 1999.

[52] E. O. Omojokun. *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*. PhD thesis, University of Colorado, Boulder, 1989.

[53] D. B. Özyurt and P. I. Barton. Cheap Second Order Directional Derivatives of Stiff ODE Embedded Functionals. *SIAM J. Sci. Comput.*, 26(5):1725–1743, 2005.

[54] J. A. Ritter and A. D. Ebner. State-of-the-Art Adsorption and Membrane Separation Processes for Hydrogen Production in the Chemical and Petrochemical Industries. *Separ. Sci. Technol.*, 42(6):1123–1193, 2007.

[55] D. M. Ruthven. *Principles of Adsorption and Adsorption Processes*. John Wiley-Interscience: New York, NY, 1984.

[56] D. M. Ruthven, S. Farooq, and K. S. Knaebel. *Pressure Swing Adsorption*. VCH Publishers: New York, NY, 1994.

[57] J. Schell, N. Casas, and M. Mazzotti. Pre-combustion CO_2 Capture for IGCC Plants by an Adsorption Process. *Energ. Procedia*, 1:655–660, 2009.

[58] W. E. Schiesser. *The Numerical Method of Lines Integration of Partial Differential Equations*. Academic Press: San Diego, CA, 1991.

[59] R. Serban and A. Hindmarsh. CVODES: the Sensitivity-Enabled ODE Solver in SUNDIALS. *Proceedings of IDETC/CIE*, 2005.

[60] R. E. H. Sims, H.-H. Rogner, and K. Gregory. Carbon emission and mitigation cost comparisons between fossil fuel, nuclear and renewable energy resources for electricity generation. *Energ. Policy*, 31(13):1315–1326, 2003.

[61] S. Sircar. Separation of Multicomponent Gas Mixtures. US Patent 4171206, 1979.

[62] S. Sircar. Separation of Methane and Carbon Dioxide Gas Mixtures by Pressure Swing Adsorption. *Separ. Sci. Technol.*, 23(6):519–529, 1988.

[63] S. Sircar. Pressure Swing Adsorption: Commentaries. *Ind. Eng. Chem. Res.*, 41(6):1389–1392, 2002.

[64] S. Sircar and J. W. Zondlo. Hydrogen Purification by Selective Adsorption. US Patent 4077779, 1978.

[65] P. Stange, A. Griewank, and M. Bollhöfer. On the Efficient Update of Rectangular LU-factorizations subject to Low Rank Modifications. *Electronic Trans. on Numerical Analysis*, 26:161–177, 2007.

[66] T. Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[67] M. Suzuki. *Adsorption Engineering*. Kondasha Ltd.: Tokyo, and Elsevier Science Publishers: Amsterdam, 1990.

[68] A. Toumi, F. Hanisch, and S. Engell. Optimal Operation of Continuous Chromatographic Processes: Mathematical Optimization of the VARICOL Process. *IECR Journal*, (5):4328–4337, 2002.

[69] A. Vardi. A Trust Region Algorithm for Equality Constrained Minimization: Convergence Properties and Implementation. *SIAM J. Numer. Anal.*, 22:575–591, 1985.

[70] V. S. Vassiliadis. *Computational Solution of Dynamic Optimization Problems with general Differential-Algebraic Constraints*. PhD thesis, University of London, London, 1993.

[71] V. S. Vassiliadis, E. B. Canto, and J. R. Banga. Second-Order Sensitivities of General Dynamic Systems with Application to Optimal Control Problems. *Chemical Engineering Science*, 54(17):3851 – 3860, 1999.

[72] S. R. R. Vetukuri, L. T. Biegler, and A. Walther. An Inexact Trust-Region Algorithm for the Optimization of Periodic Adsorption Processes. *Industrial & Engineering Chemistry Research*, 49(23):12004–12013, 2010.

[73] A. Wächter and L. Biegler. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Math. Program.*, 106 (1):25–57, 2006.

[74] A. Walther. A First-Order Convergence Analysis of Trust-Region Methods with Inexact Jacobians. *SIAM J. on Optimization*, 19(1):307–325, 2008.

[75] A. Walther and L. T. Biegler. Numerical Experiments with an Inexact Jacobian Trust-Region Algorithm. *Computational Optimization and Applications*, published on-line(DOI 10.1007/s10589-009-9247-4), 2009.

[76] A. Walther, S. R. R. Vetukuri, and L. T. Biegler. A First-Order Convergence Analysis of Trust-Region Methods with Inexact Jacobians and Inequality Constraints . *Optimization Methods and Software*, Submitted, 2010.

[77] P. A. Webley. Optimization of PSA Systems for Air Separation. *Fundamentals of Adsorption Conference*, 1998.

[78] M. Whysall and L. J. M. Wagemans. Very Large-scale Pressure Swing Adsorption Processes. US Patent 6210466, 2001.

[79] M. H. Wright. Interior Methods for Constrained Optimization. *Acta Numerica*, pages 341–407, 1992.

[80] P. Xiao, S. Wilson, G. Xiao, R. Singh, and P. Webley. Novel Adsorption Processes for Carbon Dioxide Capture within an IGCC Process. *Energy Procedia*, 1:631–638, 2009.

[81] T. Yamaguchi and K. Yasushi. Gas Separation Process. US Patent 5250088, 1993.

[82] R. T. Yang. *Gas Separation by Adsorption Processes*. Butterworths: Boston, MA, 1997.