DYNAMIC OPTIMIZATION FORMULATIONS FOR PLANT OPERATION UNDER PARTIAL SHUTDOWN CONDITIONS

DYNAMIC OPTIMIZATION FORMULATIONS FOR PLANT OPERATION UNDER SHUTDOWN CONDITIONS

by

ZHIWEN CHONG, B.Eng., M.A.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2012)	McMaster University
(Chemical Engineering)	Hamilton, Ontario, Canada

TITLE:	Dynamic Optimization Formulations for Plant Operation Under Partial Shutdown Conditions
AUTHOR:	Zhiwen Chong B.Eng. (McGill University, Montreal, Quebec, Canada) M.A.Sc. (McMaster University, Hamilton, Ontario, Canada)
SUPERVISOR:	Dr. Christopher L.E. Swartz
NUMBER OF PAGES:	xvi, 194

Abstract

Our research focuses on the development of systematic strategies and formulations for the optimal operation of plants under partial shutdown conditions. A partial shutdown is a type of circumscribed plant unit shutdown that permits the rest of the plant to continue operating to some degree. The goal of a partial shutdown strategy is to be able to manipulate the available degrees-of-freedom in a plant—during and after a shutdown—such that production is restored in a cost optimal fashion while meeting all safety and operational constraints. This can be accomplished through adjustments of production rates, recycles and buffer levels.

In order to compute these adjustments and to arrive at an open-loop advisory policy, we solve a differential-algebraic-equation (DAE) based dynamic optimization problem containing a model of the plant. Within this broad framework, we take a novel multi-tiered dynamic optimization approach that allows us to prioritize multiple competing objectives and specify the trade-offs from one tier to the next. The control trajectories from the solution of the dynamic optimization problem can then be used to inform the formulation of an inventory management policy for operating a plant during partial shutdowns.

In this work, we employ a transient model-based partial shutdown control algorithm for implementing the above-mentioned policy under feedback. This permits the policy to be carried out in the presence of uncertainty and disturbances. An economics-driven Model Predictive Control (MPC) control structure is selected for this purpose.

We also present several parsimonious discrete modeling formulations for handling model discontinuities such as shutdown thresholds, induced shutdowns and minimum shutdown durations. The problem of minimizing the restoration time is also considered. The resulting optimization problem is a mixed-integer dynamic optimization problem (MIDO) that can either be employed in an open-loop advisory capacity or embedded inside a feedback control framework.

In most control applications, some degree of plant-model mismatch will invariably be present. We investigate the use of nonlinear state and parameter estimation algorithms to moderate the adverse effects of mismatch. The estimation algorithms are based on novel configurations of the constrained Unscented Kalman Filter (UKF), a derivative-free observer that received considerable attention in recent years. Constraints on the state and parameter estimates are enforced through a simple projection method that is based on solving a quadratic program.

Plant-model mismatch can often render terminal constraints infeasible. To address this, we employ a form of the multi-tiered optimization problem where the objectives of dynamic feasibility, cost-optimality, deviation from product specifications and input effort are solved in separate tiers. The dynamic feasibility tier ensures that terminal constraints and parameters arriving from a one-step ahead estimator are feasible for the prediction horizon in the control optimization problem.

An accompanying contribution is a proof-of-concept modeling system for prototyping dynamic optimization models known as MLDO (Modeling Language for Dynamic Optimization). The modeling system reads a mathematical description of a problem and automatically generates code in various computer languages for optimization, simulation, visualization and analysis of dynamic optimization problems. Facilities for problem reformulation and transformations are also included.

Acknowledgements

The author wishes to thank his academic advisor, Christopher L. E. Swartz for his support and guidance throughout the course of the author’s graduate studies. This work owes its existence to Dr. Swartz’s direction and vision.

Gratitude is also due to the various institutions that have either directly or indirectly provided financial support for this work, in particular the McMaster Advanced Control Consortium (MACC) and the Department of Chemical Engineering at McMaster University. As well, the author wishes to put on record a heartfelt note of appreciation to his friends at the MACC for their moral support and companionship.

This dissertation is dedicated to the author’s long-suffering parents, whose ardent hope is that one day their son will finally be done with his schooling.

S.D.G.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will acquaint the reader with the core problem that forms the basis of our work, namely the partial shutdown problem. The objectives of our research are stated.

1.1 The Partial Shutdown Problem

In a typical chemical plant, raw materials are transported through a number of processing units to be fashioned into some end-product. Process units are shut down from time to time either for maintenance or due to equipment failure. Shutdowns in intermediate units constitute a disruption in the processing chain, which can adversely affect the plant's ability to continue operating. They frequently also have an adverse effect on the operating economics of a plant due to the attendant loss of production capacity.

From an operations perspective, plant shutdowns can be classified as either *critical* or *partial*, with the former being those that lead to the shutdown of the entire plant and the latter, those that do not. In the case of critical shutdowns, the entire system is invariably forced to shut down, and under such circumstances the usefulness of any optimal control policy is limited.

Under partial shutdown scenarios however, it is frequently possible for an operator to pursue certain courses of action that will permit the unaffected units to continue operating to some degree. Possible remedial actions for handling partial shutdowns include reconfiguring the process pathways, re-routing material streams, slowing down production, making use of buffer capacities and so forth.

In this work, we will confine our attention to the production adjustment and buffer capacity coordination aspects of the remedial process, which involve the management of material flowrates and buffer contents. However, the framework we have adopted is a general one and may be extended to the other types of remedial actions listed above.

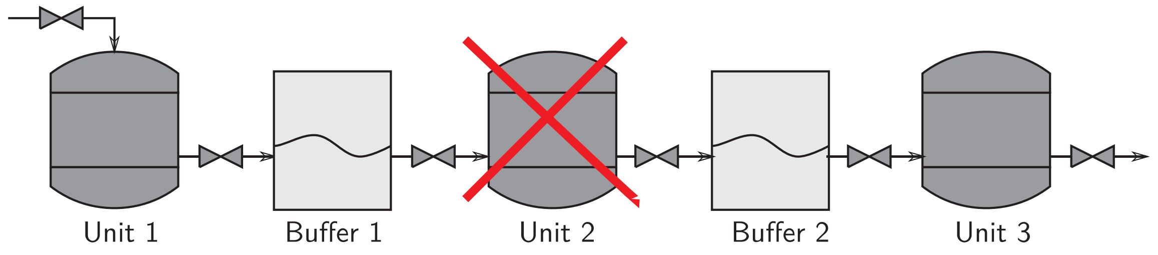


Figure 1.1: Example of partial shutdown in unit 2. Units 1 and 3 are able to continue operating due to the presence of buffer tanks.

In general, there are significant benefits to having judiciously placed buffer capacities in a plant, especially for the mitigation of process variation propagation along a production line. Buffer capacities serve to decouple various segments of a plant. They are not only able to dampen the effects of short term fluctuations, but are also able to deal with larger processing disturbances. In the case of partial shutdowns, they can effectively circumscribe the impact of a unit shutdown by decoupling the offline unit(s) from the rest of the plant.

When a process unit downstream of a buffer tank fails, the tank can hold and accumulate material for a period of time until the process unit is brought back online. Likewise, when a process unit upstream of a buffer tank fails, the material held in the buffer tank can be slowly discharged to the downstream units in order that downstream processing may continue. Fig. 1.1 depicts a series of units, with Unit 2 being in a state of failure. The process is able to continue operating because Unit 1 is allowed to fill Buffer 1, while Unit 3 is allowed to draw material from Buffer 2. In some cases, Units 1 and 3 may have to throttle down production in order to avoid overwhelming or drying out the buffers adjacent to them. The question of how much throttling is required is addressed in this work.

In this simple example, the buffer inventory management strategy is easy to intuit. However in problems where multiple recycles and compositions exist in the context of an integrated plant—a situation that we deal with in this thesis—the problem becomes more complicated. This warrants the use of an optimization framework.

The successful management of buffer inventories, recycles and production rates during the shutdown period is one of the keys to minimizing the losses associated with partial shutdown scenarios. The end-product quality during the shutdown period may violate normal specifications, but in some cases, the material can be recycled and reprocessed. We provide an expanded description of the various approaches to buffer inventory management in Section 2.1.

1.2 Phases of a Partial Shutdown

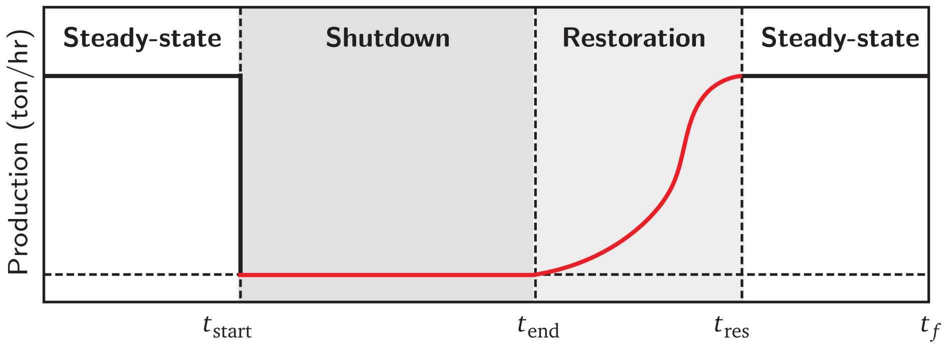


Figure 1.2: Operating phases of a single process unit, under partial shutdown conditions.

Fig. 1.2 shows the different phases of the shutdown process. A process unit initially operates at a certain steady-state point, until a disruption causes it to switch into shutdown mode. The shutdown phase begins at t_{start}, when all input and output flowrates to the process unit (and unbuffered process units adjacent to it) are forced to 0. The shutdown phase proceeds until t_{end}. In this work, we assume that all necessary measures are taken to repair and restart the unit between t_{start} and t_{end}. At time t_{end}, the unit is deemed to be in a state that is ready for operation, and the restoration phase begins. In the restoration phase, control actions are prescribed to the plant to return it to its original steady-state operating point (or to some nearby nominal operating point). The restoration phase terminates at time t_{res}. At this juncture, the plant has been successfully restored to normal operation.

1.3 Optimal Control of Units under Partial Shutdown

In this work, we concern ourselves mainly with the development of optimal control policies for the two middle phases in Fig. 1.2: shutdown and restoration. The problem of determining optimal control inputs for operating the plant is cast as a differential-algebraic-equation (DAE) dynamic optimization problem. The model is optimized with respect to an economics-based objective function. The object of this to restore a plant from a state of partial shutdown in the most profitable (or least unprofitable, as the case may be) fashion, while respecting physical plant constraints.

In this dissertation, we adopt the simultaneous (full-discretization) method for solving the resulting DAE optimization problems. The method used is orthogonal collocation on finite elements. Numerical partial shutdown formulations involving continuous decision variables are presented.

In order to implement the control policies calculated from the DAE optimization problem under uncertainty, a feedback control scheme is proposed in which states (and parameters such as downtime estimates) are used to periodically update the dynamic optimization model. A Model Predictive Control (MPC) based framework is adopted in which a dynamic optimizer is embedded. We also put forward the notion of a user-configurable multi-tiered approach to dynamic optimization that allows us to prioritize multiple competing objectives and specify the trade-offs between the tiers. The mandate of the proposed controller is strictly limited to that of controlling the plant during the shutdown and restoration periods. We envisage a scenario where the operator switches the plant mode from "normal operation" to "abnormal operation" at t_{start}, when the partial shutdown occurs. In this abnormal operation mode, the proposed controller is activated and takes over (from the nominal control system) the task of managing the plant. As soon as the system is restored to its original state (at t_{res}), the shutdown controller then transfers control back to the nominal control system. The estimated duration of the shutdown is a parametric feedback parameter which can be updated by the user.

1.4 Discontinuous Formulations for Partial Shutdowns

Some types of phenomena associated with partial shutdown optimization entail the use of discrete variables. In this dissertation, we present parsimonious and efficient mixed-integer formulations for capturing the real economic cost of induced shutdowns, for imposing minimum shutdown durations and for minimizing the post-shutdown restoration time. In the formulation for minimizing the restoration time, a multi-tiered approach is employed whereby the restoration time is first minimized, and the remaining degrees-of-freedom are subsequently used to optimize the economics and minimize the input effort required. A linear MPC-based shutdown controller (which embeds a multi-tiered Mixed Integer Dynamic Optimzer) is presented for implementing the solutions obtained on plant under uncertainty.

1.5 Nonlinear Parameter and State Estimation of Plant undergoing Partial Shutdowns

In order to moderate the effects of plant-model mismatch, we explore schemes that recursively estimate uncertain parameters and missing states using a Constrained Unscented Kalman Filter (UKF) algorithm. In cases where the set of uncertain parameters are known explicitly, they are estimated directly. In other cases where the sources of the mismatch are unknown (e.g. structural mismatch, persistent stochastic disturbances, etc.), fictitious disturbance terms are estimated to reconcile plant behavior with model predictions.

In the presence of mismatch, the endpoint constraints required to restore the plant from a shutdown state to its nominal steady-state operating point are potentially invalid. An appropriate configuration of a multi-tiered optimization strategy is proposed in which an optimization problem is solved in the first tier to compute a new feasible endpoint, as well as to ensure the feasibility of estimated parameters through the controller's prediction horizon. Once the dynamic feasibility objective is satisfied, the remaining objectives are to use the system's degrees-of-freedom to achieve optimal economics, minimum deviations from product quality targets, and minimum input effort, all in a prioritized fashion. We note that this is one of many possible configurations of the multi-tiered optimization problem.

1.6 Problem Formulation and Solution Techniques

A significant amount know-how in large-scale dynamic optimization was acquired during the course of this work. Optimization formulation and modeling techniques are documented. In the course of this work, a proof-of-concept modeling language named MLDO (Modeling Language for Dynamic Optimization) was developed to handle the generation of code necessary for the solution and analysis of dynamic optimization problems. Functionality for modifying problem formulations is included in this language.

1.7 Research Problem Statement

The objectives of this work are:

- Propose formulations for modeling partial shutdowns in an economics-driven dynamic optimization problem.
- Solve resulting dynamic optimization problems to arrive at a set of optimal control inputs for controlling the process transitions during the shutdown and restoration periods.
- Propose a control framework for implementing the optimization results in a plant in the presence of uncertainties such as plant-model mismatch and disturbances.
- Propose discrete formulations for dealing with discontinuous behaviors associated with partial shutdowns.
- Investigate the use of a state and parameter estimation algorithm to estimate mismatched parameters and unmeasured states, and to moderate the adverse effects of plant-model mismatch.
- Develop relevant know-how in large-scale optimization techniques and build a modeling system for the construction, analysis and solution of large-scale dynamic optimization problems.

1.8 Thesis outline

This thesis is organized according to the following chapters.

In Chapter 2, we review the state-of-the-art in the approaches that have been taken in literature to deal with partial shutdown problems. We also provide an overview of the major concepts and techniques we employ in this thesis, such as dynamic optimization, model predictive control, state estimation and so on.

In Chapter 3, we present a method for formulating a partial shutdown problem as an open-loop economics-driven dynamic optimization problem. We show the use of this method to develop policies for operating a plant under partial shutdown conditions. The Kraft fiber line process was used in the case studies.

In Chapter 4, we pose several efficient discrete optimization formulations for representing certain types of discontinuous behaviors associated with partial shutdowns. These include economic penalties for induced shutdowns and minimum shutdown durations. We also describe a discrete optimization formulation for prioritizing an alternative goal, namely that of minimizing restoration time before maximizing economics.

In Chapter 5, we consider the problem of performing parameter and state estimation on plants undergoing partial shutdown. We explore using estimation techniques (based on the Unscented Kalman Filter) to obtain parameters and states. The controller model predictions are corrected in order to moderate the effects of plant model mismatch. We show case studies in which the algorithm is applied to the aforementioned Kraft fiber line process.

In Chapter 6, we present problem formulation and solution details. Modeling techniques and strategies are covered. We briefly describe the modeling system we developed. Broad concepts related to its design and operation are discussed.

In Chapter 7, we summarize the broad themes covered in this thesis. We present a list of what we consider to be our contributions in this area of study. Future directions are laid out.

Chapter 2

Literature Review

This chapter aims to familiarize the reader with the body of work pertinent to this study. Topics such as buffer inventory management, dynamic optimization, model predictive control, state estimation, and plant-model mismatch will be reviewed.

2.1 The Buffer Inventory Coordination Problem

Many of the approaches for combating the effects of partial shutdowns rely on the use of redundancies in the plant, and buffer tanks provide such a redundancy. The use of buffer inventories to mitigate process variation propagation along a production line is a widely-employed strategy.

In 1969, Pettersson [87] developed a scheme for coordinating production in a pulp and paper mill, which included a systematic approach for managing the buffer tanks. A system comprising nine processing units and ten buffer tanks was considered for optimization, resulting in a production scheme for coordinating the plant in such a way that capacity restrictions were not violated. An example with a shutdown in the evaporator system was shown. In a later publication, Pettersson [88] considered the problem of producing an optimal plantwide production scheme that could account for maintenance shutdowns, limited buffer capacity and steam restrictions. The resulting optimal control problem was solved using Pontryagin's Maximum Principle [89].

Lee and Reklaitis [70] proposed a method for systematically utilizing buffer capacities to decouple upstream equipment failures from downstream processes and vice versa. Using Fourier series constructions, they derived a set of analytical expressions for determining the minimum volume of intermediate storage required as a function of frequency of failure and of failure durations. In a related paper, Lee and Reklaitis [69] noted that overspecified intermediate storage led to the introduction of delays in product change-overs, thus there is considerable economic benefit to specifying only as much capacity as is absolutely required.

Allison [5] used an analytical approach for determining a policy for averaging the loads on a set of surge tanks in series during a transient event (which can be taken to include unit shutdowns). In this scheme, the impact of a surge on a single tank is distributed across the plant to avoid upper and lower level constraints on buffer capacities. Optimal control theory was employed in the solution process.

Huang et. al. [57] advocated the idea of using a dynamic optimization approach for general fault accommodation and control redesign, as opposed to the manual table-lookup approach typically adopted by operators. In a follow-up publication [58], the issue of unit shutdowns—which the authors consider to be a type of fault—in a dynamic optimization context was briefly addressed. With regard to optimizing dynamic models with unit failure representations, the authors suggested first removing all equations related to the unit, followed by the activation of a set of discrete transition equations triggered using integer variables. The authors emphasized that the integer variables only act as transition conditions, and are directly prescribed by a fault detection module, hence the final model to be solved does not contain integer variables.

Dubé [39] investigated a buffer storage operation strategy that minimizes time away from normal operation and prevents departmental shutdowns. This strategy was applied to a highly integrated Kraft pulp mill with the view of maximizing production. The author addressed the issue of determining the longest feasible shutdown time (or “independence” time). The effect of preparation time on the production was also studied and the coordination of buffer capacities for handling planned and unplanned shutdowns was illustrated. A numerical optimization procedure was used in the solution process.

Continuing along the lines of Dubé’s work, Balthazaar [11] considered both a pre-emptive and a reactive response to shutdowns in a Kraft paper mill. In the pre-emptive case, the control problem is solved assuming knowledge of the shutdown ahead of time (as is the case in a scheduled maintenance scenario), thus allowing the plant to take preparatory action in anticipation of the shutdown. In the reactive case, the plant shutdown occurs without warning, and the control system is expected to respond immediately. Balthazaar also examined the problem of determining default steady-state buffer tank levels that are optimal for handling shutdowns, using information on the likelihood of process unit failure. An economics-based objective function was employed in all the case studies.

However, several key limitations exist in the aforementioned approaches. The issue of uncertain downtimes is not treated, as well as the handling of uncertainties due to plant-model mismatch and disturbances in the process. In our work, we address these issues using a multi-tiered economic MPC-type framework, described in Section 3.3 (augmented with state and parameter estimation in Chapter 5). Also, in our framework we go beyond the scope of the surveyed work by considering discrete formulations for assigning an economic cost to shutdowns (and penalizing induced ones) and for minimizing the restoration time.

2.2 Dynamic Optimization

The mathematical technique of Dynamic Optimization forms the basis of our attacks on problems of partial shutdowns. A dynamic optimization problem can be qualitatively described in the follow terms: given a dynamic model of a process, find a set of control actions that will extremize some specified performance criterion. Dynamic chemical processes are often modeled as differential-algebraic-equation (DAE) systems. These DAEs typically comprise conservation laws (mass, energy, momentum etc.), constitutive equations and equilibrium relationships, among other things.

The methods and means for solving such systems are described below. In this work, we are primarily interested in index-1[1] semi-explicit DAE optimization problems, which can be stated as follows:

$$\min_{u(t)} \Phi(x(t), z(t), u(t), p, \theta, t) \tag{2.1}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), z(t), u(t), p, \theta, t), \quad x(0) = x_0 \tag{2.2}$$

$$h(x(t), z(t), u(t), p, \theta, t) = 0 \tag{2.3}$$

$$g(x(t), z(t), u(t), p, \theta, t) \leq 0 \tag{2.4}$$

$$\text{for } t \in [0, t_f]$$

where Φ = objective functional, $x(t) \in \mathbb{R}^{n_x}$ = differential state vector, x_0 = initial state parameter

[1]While higher-index problems are outside the scope of this thesis, many such problems can be reformulated into index-1 systems through index-reduction techniques. This would make them amenable to the methods described here.

vector, $z(t) \in \mathbb{R}^{n_z}$ = algebraic state vector, $u(t) \in \mathbb{R}^{n_u}$ = control input vector, $p \in \mathbb{R}^{n_p}$ = design variable vector (time-invariant variables), $\theta \in \mathbb{R}^{n_\theta}$ = parameter (constant) vector, t_f = final time in prediction horizon. f represents a mapping of differential equations, g is a mapping of inequality constraints and h is a mapping of algebraic equations. We assume that the DAE defined by 2.2 and 2.3 is solvable and of index-1, that is, the partial derivative matrix $\frac{\partial g}{\partial z}$ must be non-singular.

Historically, mathematical techniques (so-called *indirect or variational solution methods*) based on the calculus of variations were employed for treating dynamic optimization problems that embed continuous Ordinary Differential Equations (ODEs). The most notable of these is Pontrayagin's Maximum Principle [89], which uses 1st-order optimality conditions to arrive at the stationary functions of a problem. In unconstrained problems, this leads to a multi-point boundary value problem. For inequality constrained problems however, additional complementarity conditions have to be satisfied and multipliers determined. This often results in a combinatorial problem that is computationally intractable for systems of a nontrivial size.

Therefore, for most problems of practical importance, numerical methods must be employed. *Direct solution methods* seek to transform infinite-dimension problems into finite dimensional nonlinear programs by parameterizing the continuous time profiles. Unlike the indirect methods above, these direct methods admit DAEs as well as ODEs. Direct methods generally fall into three categories, *sequential (single shooting)*, *multiple shooting* and *simultaneous* methods. A brief description is given of each in the following sections.[2]

2.2.1 Sequential (Single-Shooting) Method

The sequential method is a two-layered solution strategy. In this method, only the control input vector $u(t)$ is discretized; this is referred to as "control vector parameterization". The nonlinear optimization and DAE integration steps are performed separately. The control profile $u(t)$ is typically parameterized by first dividing the optimization horizon $[0, t_f]$ into N discrete elements:

$$0 = t_0 < t_1 < t_2 < \ldots < t_N = t_f \tag{2.5}$$

[2]For the sake of completeness, we should mention that there exist other approaches for solving dynamic optimization problems such as dynamic programming, metaheuristic/stochastic algorithms, and others. However, they are outside the scope of this work, and hence will not be discussed in this survey.

On each interval, the control profile is approximated using a basis function, such as a low-order polynomial (often in Lagrange polynomial form) or a piecewise-constant function:

$$u(t) = U_k(t, a_k), \quad t_{k-1} \leq t \leq t_k,\ k = 1 \dots N \tag{2.6}$$

where a_k is a vector of the coefficients of the basis function. For instance, if a piecewise-constant function is chosen, then $U_k(t, a_k) = a_{k,0}$; if a linear function is chosen, then $U_k(t, a_k) = a_{k,0} + a_{k,1}t$; and so on. These become decision variables in the optimization problem.

Given a set of initial values for the DAE and a starting control profile, the DAE model is integrated (using specialized DAE solvers such as DASSL or DASPK [27]) and the objective function value evaluated. Based on the objective value and sensitivity information obtained from the system, the optimizer tries to find a set of control actions that will improve upon the current solution. This new set of control actions are then fixed in the DAE problem, and the integration procedure is repeated with the new inputs. This iterative procedure continues until a set of stipulated termination conditions is met (see Fig. 2.1 for a visual depiction).

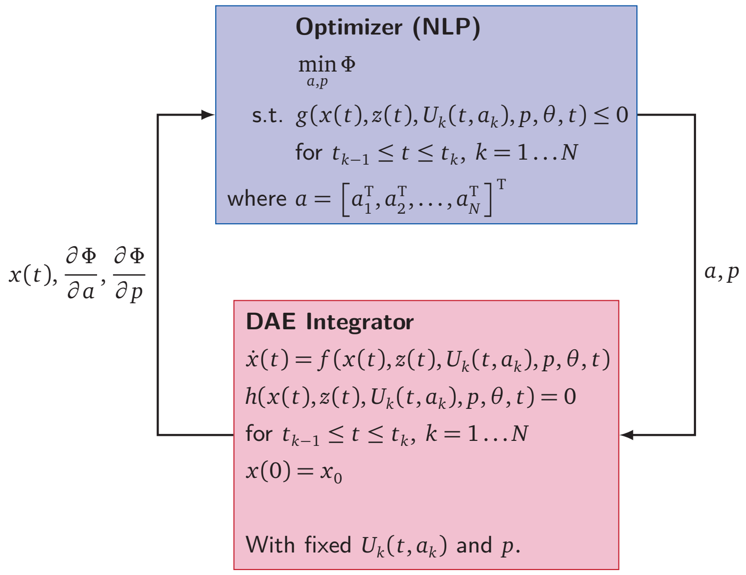

Optimizer (NLP)
$\min_{a,p} \Phi$
s.t. $g(x(t), z(t), U_k(t, a_k), p, \theta, t) \leq 0$
for $t_{k-1} \leq t \leq t_k,\ k = 1 \dots N$
where $a = \left[a_1^T, a_2^T, \dots, a_N^T\right]^T$
$x(t), \frac{\partial \Phi}{\partial a}, \frac{\partial \Phi}{\partial p}$
a, p
DAE Integrator
$\dot{x}(t) = f(x(t), z(t), U_k(t, a_k), p, \theta, t)$
$h(x(t), z(t), U_k(t, a_k), p, \theta, t) = 0$
for $t_{k-1} \leq t \leq t_k,\ k = 1 \dots N$
$x(0) = x_0$
With fixed $U_k(t, a_k)$ and p.

Figure 2.1: Single-shooting method – optimizer and integrator blocks iterate until termination criteria are met.

One attractive advantage of the sequential method is that for problems where number of state variables far exceed the number of control variables ($n_x \gg n_u$, that is, relatively few degrees of freedom are available), the nonlinear programming problem arising is usually manageably small in size. Also, the sequential method typically provides more accurate solutions for the state variables than the simultaneous method, as the DAE solver usually incorporates error-control when performing the integration.

However, in practice, sequential methods often incur significant computational expense from having to integrate a DAE system at every single NLP iterate [1]. A large amount of computational effort may be wasted in trying to obtain DAE solutions when the decision variables are far away from their optimal values. Inefficiencies related to integration are particularly pronounced in stiff systems [106]. Biegler [21] states that the sequential method is only robust for systems with exclusively stable modes (defined as modes in which profiles remain bounded as the time goes to infinity); otherwise, the problem may fail prematurely due to integration failure at unstable intermediate points, even if a stable final solution exists.

Because the state variables only appear in the integration problem and not in the nonlinear program, path constraints involving state variables cannot be directly enforced in the optimization step. Various methods for circumventing this problem have been proposed, ranging from forcing a constraint violation penalty to zero through an endpoint constraint [111] to using initial point solvers that respect path constraints [45].

2.2.2 Multiple Shooting Method

Much like the sequential method, the multiple shooting method (due to Bock and Plitt [23]) also parametrizes the control variables $u(t)$. The primary point of divergence between the former method and the latter is in the treatment of the continuity of the states $x(t)$ at every control stage k. In the multiple shooting method, state continuity is only enforced at the solution of the optimization problem. During the optimization iterates, discontinuities in the states between stages are allowed. In this method, the differential equation is divided into N discrete stages (similar to the control variables):

$$\dot{x}_k(t) = f(x_k(t), z_k(t), U_k(t, a_k), p, \theta, t), \quad t_{k-1} \leq t \leq t_k,\ k = 1 \dots N \tag{2.7}$$

The initial point $x_1(t_0)$ is corresponds to x_0. At each control stage subinterval k, decision variables c_k are used as initial values for the differential equation above:

$$x_k(t_{k-1}) = c_k, \quad k = 2 \dots N \tag{2.8}$$

where c_k are decision variables that are used to enforce continuity in the states.

The continuity in the state profile is enforced by defining a set of equality constraints in the

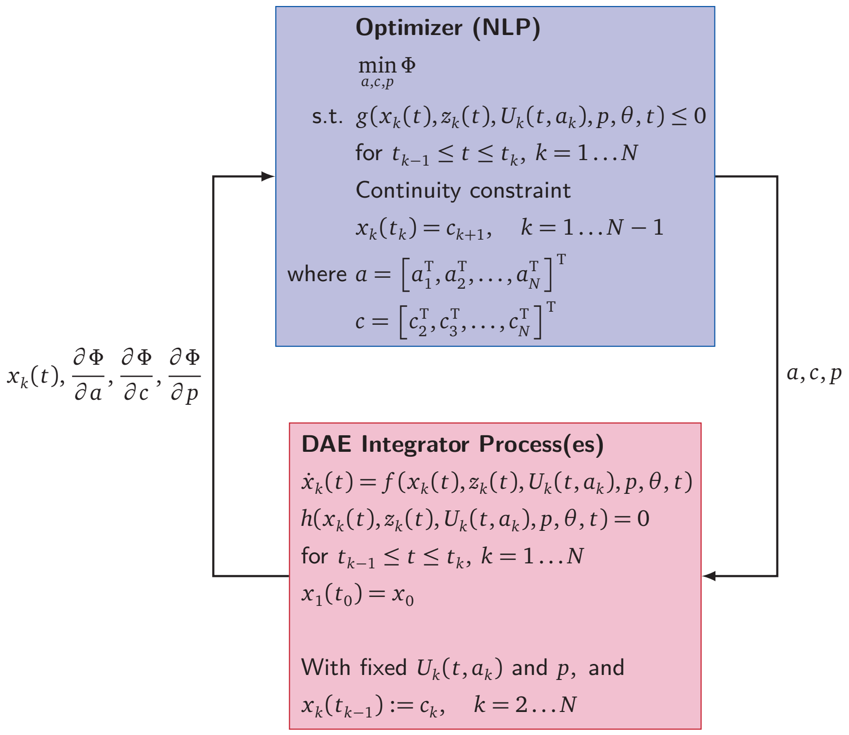

Optimizer (NLP)
$\min_{a,c,p} \Phi$
s.t. $g(x_k(t), z_k(t), U_k(t, a_k), p, \theta, t) \leq 0$
for $t_{k-1} \leq t \leq t_k,\ k = 1 \ldots N$
Continuity constraint
$x_k(t_k) = c_{k+1}, \quad k = 1 \ldots N-1$
where $a = \left[a_1^T, a_2^T, \ldots, a_N^T\right]^T$
$c = \left[c_2^T, c_3^T, \ldots, c_N^T\right]^T$
$x_k(t), \frac{\partial \Phi}{\partial a}, \frac{\partial \Phi}{\partial c}, \frac{\partial \Phi}{\partial p}$
a, c, p
DAE Integrator Process(es)
$\dot{x}_k(t) = f(x_k(t), z_k(t), U_k(t, a_k), p, \theta, t)$
$h(x_k(t), z_k(t), U_k(t, a_k), p, \theta, t) = 0$
for $t_{k-1} \leq t \leq t_k,\ k = 1 \ldots N$
$x_1(t_0) = x_0$
With fixed $U_k(t, a_k)$ and p, and
$x_k(t_{k-1}) := c_k, \quad k = 2 \ldots N$

Figure 2.2: Multiple-shooting method – optimizer and integrator blocks iterate until termination criteria are met.

nonlinear program:

$$x_k(t_k) = c_{k+1}, \quad k = 1 \ldots N-1 \tag{2.9}$$

where $x_k(t_k)$ are the states at point t_k of the integrated solution at subinterval k, obtained from the integrator. Conceptually, this is equivalent to partitioning the DAE into N subintervals (over the time grid), integrating the pieces independently (with some specified initial value in each subinterval), and enforcing continuity between the states in each subinterval upon convergence. The procedure is depicted in Fig. 2.2.

The multiple shooting method lies at the interface between sequential and simultaneous methods, and thus inherits many of their advantages. It gives rise to an NLP whose size lies between those of the two aforementioned methods. As well, it permits the user to employ a DAE integrator to obtain an error-controlled DAE solution. Through the partitioning strategy, multiple-shooting is able to handle unstable modes more reliably than the sequential method. Because each DAE partition can be treated independently (continuity is only enforced at the NLP solution), this also lends the algorithm to easy and natural parallelization with respect to the DAE integration procedure. This enables one to harness the computational power of multiple processors to perform each integration (over the defined subintervals) in a concurrent fashion. The efficacy of the multiple shooting method has been demonstrated in the context of nonlinear model predictive control applications [94]. An extensive treatment of the multiple shooting method for optimal control can be found in Leineweber et al. [71]. A modification of the multiple shooting method that uses collocation on finite elements for integration is presented in Tamini and Li [102].

2.2.3 Simultaneous Method

In the simultaneous method (sometimes known as “full discretization” or “direct transcription”), both the continuous state and control profiles are discretized over a fixed time-grid, resulting in a large scale sparse NLP whose solution usually entails specialized solution strategies [20]. Of note is the fact that no explicit integration step is performed because both the integration and optimization problems are converged simultaneously at the solution point.

A widely-used method for performing full discretization is orthogonal collocation on finite elements (OCFE) [33]. OCFE is an implicit Runge-Kutta method that converts a DAE system into a nonlinear algebraic equation (NLE) system that can be solved with a conventional NLP solver. We have elected to use this method in our work, and it is described in some detail in Section 6.1.1.

Simultaneous methods enjoy several advantages over sequential methods. First, path constraints are handled naturally within the optimization formulation. Further, because the DAE system is only solved once, unstable or nonexistent intermediate solutions are bypassed [20].

There are however disadvantages associated with simultaneous methods. As stated before, large scale NLPs arise from this method and they can be challenging to solve. Reduced space methods have been proposed to alleviate some of the difficulties related to large-scale problems. It is worth noting that large scale interior-point solvers that exploit the sparsity of the problem have been developed [114] and have proven to be successful in finding the solution to these types of problems. The other issue with simultaneous methods is their requirement of a good initial guess for the solution profiles in the NLP. In the sequential and multiple-shooting methods, the profile initializations happen naturally because at each NLP iterate, the optimizer invokes the integration of the DAE system at the current point. The simultaneous method does not utilize such an integration, which accounts for its generally superior speed of convergence, but the downside is a lack of an obvious procedure for warm-starting the NLP. Lang et al. [68]

propose using the solution of a one-time DAE integration at some nominal point as the starting guess.

2.3 Nonlinear Model Predictive Control

In order to implement dynamic optimization solutions on a plant (in the presence of uncertainty), a feedback control scheme is required. Model Predictive Control (MPC) is an umbrella term for a family of feedback control algorithms that exploit a model to predict the future response of a plant given a set of manipulated variable adjustments [91]. It incorporates an optimization routine which computes a sequence of control actions required to drive the plant to some operating point in an optimal fashion. There are a myriad variants of MPC (surveyed in [91]), but for this work our interest lies not with any particular variant, but instead with the feedback architecture that forms the foundation of the MPC framework. We shall attempt to supply the reader with a sketch of this architecture below.

The basic principles of nonlinear MPC can be understood as follows: suppose we are in possession of a discrete (or discretized; see previous section on dynamic optimization) dynamic plant model. This model is typically in state-space form, though many other forms are admissible. It can be stated as follows:

$$x_{k+1} = f(x_k, u_k) \tag{2.10}$$

$$y_k = g(x_k) \tag{2.11}$$

where $x_k \in \mathbb{R}^{n_x}$ are states, $u_k \in \mathbb{R}^{n_u}$ are inputs, and $y_k \in \mathbb{R}^{n_y}$ are measured plant outputs. We can use this model to predict the plant's behavior in response to different values of inputs u_k (in MPC, these are decision variables to be determined by the optimizer).

At time j, the controller solves the open-loop dynamic optimization problem below to obtain the optimal u trajectory that minimizes some objective Φ over a horizon of length p_h:

$$\min_{u_{k\in\mathscr{K}}} \Phi \tag{2.12}$$

$$\text{s.t. } x_k = \hat{x}_j \tag{2.13}$$

$$x_{k+1} = f(x_k, u_k), \quad k \in \mathscr{K} \tag{2.14}$$

$$y_k = g(x_k), \quad k \in \mathscr{K}_f \tag{2.15}$$

where $\mathcal{K} = \{j, j+1, \ldots, j+p_h-1\}$ and $\mathcal{K}_f = \{j, j+1, \ldots, j+p_h\}$. In many standard industrial MPC implementations, the objective function Φ is often a setpoint tracking error plus a move suppression penalty. However, in principle the user can specify any suitable Φ based on the application at hand. As we shall see later, the choice of an economics-based Φ is often a reasonable one. The vector $\hat{x}_j \in \mathbb{R}^{n_x}$ denotes state estimates at MPC iteration j. Suppose that at the current time j, a set of feedback measurements from the plant, $y_j^m \in \mathbb{R}^{n_y}$ is obtained. Typically, a state estimation algorithm (details provided in Section 2.5) can be used to reconstruct the states from these measurements; that is, the state estimate vector $\hat{x}_j$ can be computed from the measurements, y_j^m. In some situations, all the states in the plant are measured, thus the state estimate term is simply $\hat{x}_j = y_j^m$. This case of affairs is known as "full-state feedback"[3], it constitutes the primary means of feedback for the closed-loop framework discussed in Chapters 3 and 4 of this thesis.

The resulting output of optimization problem (2.12–2.15) is a sequence of computed control actions, $u_{k\in\mathcal{K}}$. The first of this sequence, u_j is sent to the plant. At the next sampling instance, the MPC iteration counter is incremented ($j := j+1$) and new feedback measurements y_j^m are read in from plant instruments and sensors. This procedure is repeated at each control interval. Fig. 2.3 provides a broad idea of the MPC scheme in visual terms.

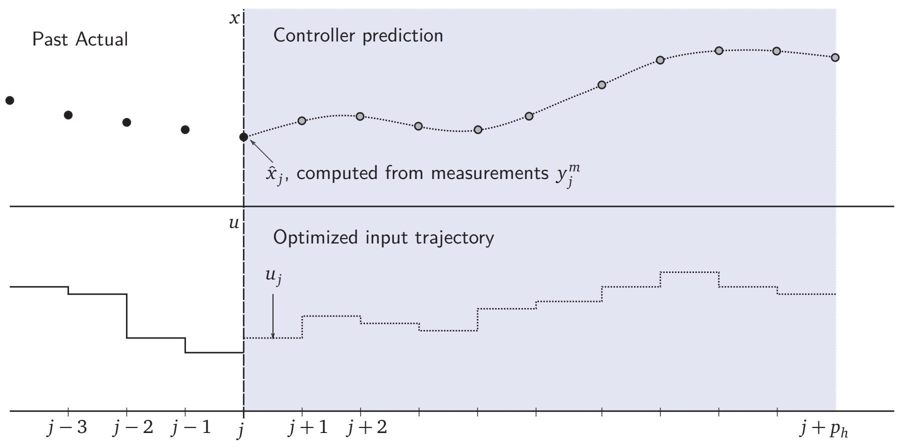


Figure 2.3: MPC summary: (1) Plant measurements at the current time j are used to compute a state estimate $\hat{x}_j$. (2) Based on this, the controller optimizer computes a prediction and optimal input trajectory. (3) Only the first step of the input u_j is implemented on the plant. (4) The MPC counter j is incremented, and the process is repeated.

In most MPC deployments, a *receding* prediction/control horizon is used. That is, as the MPC

[3]In some cases, the states are a subset of the measurement vector (that is, $n_y > n_x$). In such cases, we have $\hat{x}_j = A^x y_j^m$, where A^x is an appropriate matrix that selects the required state variable values from the measurement vector.

marches along, the length of the window p_h remains constant, and shifts along with the current time. In finite time applications such as batch processes, a *shrinking* horizon is sometimes used, where the window decreases in size as the MPC progresses (i.e. as j increments, p_h decrements).

The many favorable properties of MPC have made it one of the most widely-used advanced control schemes today [91]. It is an inherently multivariate controller, therefore interaction effects are handled naturally through the model. In most variants of MPC, operating constraints are handled easily through hard constraints in the optimization problem or soft constraints via penalty terms in the objective function.

One of the earliest algorithms in the MPC family is Dynamic Matrix Control (DMC), a linear MPC algorithm brought to the fore by Cutler and Ramaker [34] in 1979. DMC incorporated finite step response (FSR) models and was capable of handling only linear dynamics. Uncertainties were handled through feedback measurements. Although in real life many chemical processes exhibit inherently nonlinear behavior, linear MPC algorithms continue to be pervasive, especially in continuous processes where the predominant objective is to maintain the plant at certain operating points (i.e. *regulatory control*) [91].

For transient processes however (a class of processes which partial shutdowns fall under), the ability to move rapidly—in a constrained environment—from one operating point to another (i.e. *servo control*) is arguably more important. In such control applications, process nonlinearities can become significant. Nonlinear models are better suited to these applications, thus entailing the use of nonlinear MPC for control.

Much of the work done in the area of applications of MPC to transient processes have been on batch processes, whose many dynamic characteristics are shared with transient processes. Krothapally and Palanki [66] showed an online application of MPC to a batch polymerization process. Nagy and Braatz [83] developed a shrinking-horizon MPC algorithm which uses an economic objective function, and showed an example of an application of their algorithm to a batch crystallization simulation. Hillestad and Anderson [54] applied nonlinear predictive control to the problem of grade transitions in polymerization reactors. The work of Wang et. al. [117] involved the use of nonlinear model predictive control to manage grade transitions on a polyethylene reactor.

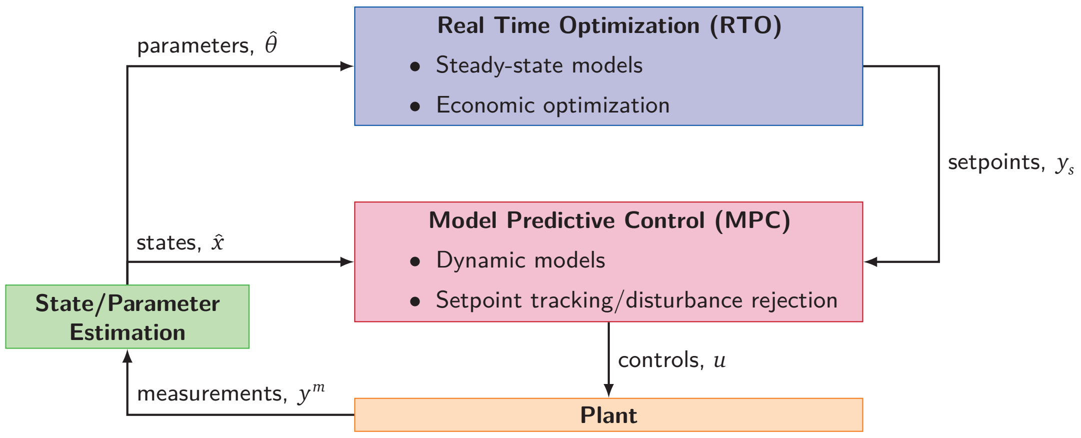

Real Time Optimization (RTO)
• Steady-state models
• Economic optimization
parameters, $\hat{\theta}$
setpoints, y_s
Model Predictive Control (MPC)
• Dynamic models
• Setpoint tracking/disturbance rejection
states, $\hat{x}$
State/Parameter Estimation
controls, u
measurements, y^m
Plant

Figure 2.4: Traditional control hierarchy in a plant. (Note: layers above the RTO (i.e. planning, scheduling) are not shown here.)

2.3.1 Incorporating Economics into the Control Algorithm

In many large oil refineries, economic optimization is typically approached in a hierarchical manner (Fig. 2.4). The control layer that performs the economic optimization is often the Real-Time Optimization (RTO) layer, which computes the optimal operating point of a plant (based on current economics and business decisions) and sends the relevant setpoints to a lower-level multivariable controller, whose duty is to track them. RTO systems are typically based on nonlinear steady-state models and operate at a lower frequency than MPC. The RTO optimization problem can be stated as follows (Marlin & Hrymak [76]):

$$\max_{w} \Phi_{\text{econ}}(w,\hat{\theta}) \tag{2.16}$$

$$\text{s.t. } f_{\text{ss}}(w,\hat{\theta}) = 0 \tag{2.17}$$

$$g_{\text{ss}}(w,\hat{\theta}) \leq 0 \tag{2.18}$$

$$w^L \leq w \leq w^U \tag{2.19}$$

where Φ_{econ} is the economic objective function, $\hat{\theta}$ is the estimated parameter vector, w is vector of optimization variables, and w^L, w^U are its lower and upper bounds respectively. The functions f_{ss} and g_{ss} are the equality and inequality constraints that represent the steady-state plant model. The setpoint vector y_s is typically constructed from a subset of (or is identical to) the optimization variables, hence $y_s = A^{sp}w$ where A^{sp} is an appropriately-sized matrix.

Given y_s, lower level MPC controllers attempt to calculate the controls u that will best track the setpoints. Measurements received from the plant are passed to a state/parameter estimator. The estimated parameters $\hat{\theta}$ are used to update the RTO model, while the estimated states $\hat{x}$ are enter the MPC model to apprise it of the plant's current operating point.

This hierarchical structure works sufficiently well for steady-state operations, but there is some indication that they are inadequate for nonlinear dynamic processes such as grade transitions and batch systems [61] due to the time-scale separation and the potential conflict between the RTO's steady-state models and the MPC's dynamic models. In view of that, there has been a move towards exploring the use of dynamic models in RTO systems and making them computationally tractable [107, 61, 120, 2], culminating in so-called Dynamic RTO (D-RTO) strategies. A comprehensive survey of the issues surrounding the RTO approach can be found in Engell [41].

However, there is a growing level of interest in an alternative approach named *Economics-based MPC*, which involves modifying standard MPC formulations to incorporate profit as a goal and eschewing the RTO layer altogether. The major advantage of this approach is that the economic and control optimization occur at the same frequency, using the same model. Most standard MPC implementations currently available are designed for the process control objectives of setpoint tracking and disturbance rejection. Economics-based MPC proposes to exchange these objectives for a more fundamental objective (economics) which, if posed correctly, naturally subsumes the control objectives.

De Souza et al. [35] and Zanin et al. [122] propose a 1-layer approach where the economic optimization problem is solved together with the MPC optimization problem. This is accomplished by adding an economic function as a penalty term in the objective function of the MPC controller. Their motivation for taking a 1-layer approach as opposed to a traditional 2-layer approach (where economics and dynamics are separated) is to address what is perceived to be a major deficiency associated with traditional layered optimization strategies, that is, because the controller and the optimizer are not dealing with exactly the same pieces of information, conflicts may arise and the predicted optimal operational point may turn out to be suboptimal. Zanin reported success in applying this algorithm to an industrial process. Their 1-layer approach suffers from a few drawbacks, including a lack of separation of disturbance time-scales, and the difficulty in determining the penalty weights for properly trading off economic objectives and process objectives such as product purities. Diehl et al. [36] construct a Lyapunov function for economic MPC and apply it to a particular class of nonlinear systems possessing the property of strong duality in the steady-state problem. This allows for the analysis of the asymptotic stability of the closed-loop system. Heidarinejad et al. [53] apply Lyapunov techniques to the design of a nonlinear Economic MPC. This controller has two operation modes: in the first, the cost function is optimized to its maximum extent, subject to constraints that maintain the closed-loop system states within some predefined stability region. In the second mode, the controller drives the system to some appropriate steady-state. Lyapunov-based constraints ensure that the closed-loop system states are bounded in the predefined stability region and also bounded in a small region containing the origin. Zavala [16] and Huang [125] demonstrate an advanced-step MPC algorithm for solving economic MPC problems where a precalculated optimal solution can be rapidly updated using plant measurements and implemented on the plant, thus avoiding the problem of optimizer-delayed solutions.

2.4 Hybrid (Discrete-Continuous) Dynamic Optimization

A hybrid system is a system containing continuous and discrete elements that need to be considered simultaneously [73]. In chemical process systems, the variables in the plant that evolve continuously (temperature, pressure, level, concentration, flowrates, etc.) have been the primary focus of process control research thus far. The spectrum of hybrid systems is wide, and encompasses classes of systems such as "switched systems", i.e. systems that undergo mode traversals on a macroscopic level. These systems usually transition from one mode to another in response to an event and are frequently managed by supervisors governed by logical rulesets or expert systems.

Broadly, switching behavior can generally be classified as "autonomous" or "controlled". *Autonomous switching* occurs when the states of the system meet a certain condition and switches with no intervention from an external source; for example, a phase change in a multi-phase system. *Controlled switching*, on the other hand, is solely induced by an external supervisory system, e.g. time-switched or logic-based schedules on PLCs (Programmable Logic Controllers) or PACs (Programmable Automation Controllers).

Engell et al. [42] provide a good overview of the hybrid phenomena typically encountered in chemical processing plants. They noted that the most important source of hybrid interaction in a chemical plant typically arises from the interaction of the plant with the control system. On an operational level, switching behavior is commonly observed in batch processes, where processing progresses in distinct phases which begin and end under certain conditions.

There are a diverse number of ways to model hybrid systems, each with a different emphasis. Branicky [26] provides an excellent survey of the issues surrounding various hybrid model structures. The approach in the engineering literature generally begins from continuous-variable description of a dynamic system and seeks to extend it by appending discrete or switching variables. In their paper, Morari and Barić [80] remark that for a large class of hybrid systems (which include chemical process systems), the plant often evolves continuously even though the inputs may be discrete. Xu and Antsaklis [121] further endorse this view by noting that for most chemical process applications, states are often continuous between switches, that is, state jumps usually do not occur. This is an important property that simplifies analysis in many cases.

For the purposes of our work, instead of adopting a full hybrid model approach (surveyed below), we confine our focus to the development of custom MIP-based formulations for the dynamic optimization of partial shutdowns using a MIDO (Mixed Integer Dynamic Optimization) approach. The binary variables in our case are used to determine the shutdown state of each process unit in the plant. This results in an algebraic discrete-continuous dynamic optimization problem that is solvable using any standard mixed-integer solver.

In the next few sections, we will survey a few model representations that are dominant in the discrete-continuous modeling domain within the field of chemical engineering. We note that our approach is somewhat distinct from most of the model-switching approaches covered below, but nevertheless we review them for the sake of completeness.

2.4.1 Linear Hybrid Models

Mixed Logical Dynamical (MLD) Models

Bemporad and Morari [18] describe a modeling paradigm called *Mixed Logical Dynamical* (MLD), designed to describe systems that exhibit discrete-continuous behavior arising from physical laws, logical rules and operating constraints. MLDs comprise linear discrete-time state-space equations with continuous and integer variables. The authors lay out a systematic procedure (inspired by the work done by Raman and Grossmann [93] and Williams [118]) for transforming propositional logic (`IF`, `AND`, `OR`, etc.) statements into efficient and minimal integer formulations for representing discontinuous behavior in dynamic systems. The authors also describe techniques to handle nonlinearities such as bilinear terms, and offer a multi-model approach for more severe nonlinearities. MLDs are solved as MILPs (Mixed Integer Linear Programs) or MIQPs (Mixed Integer Quadratic Programs). The canonical form of an MLD can be stated as follows:

$$x_{k+1} = Ax_k + B_1 u_k + B_2 \delta_k + B_3 z_k \tag{2.20}$$

$$y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k \tag{2.21}$$

$$E_2\delta_k + E_3z_k \leq E_4x_k + E_1u_k + E_5 \tag{2.22}$$

where k = discrete time-instant, x_k = states, u_k = inputs, y_k = outputs, δ_k = binary variables, z_k = auxiliary variables (a logic modeling construct), A, B, C, D, E = model coefficient matrices.

Among the many examples of applications are Feather et al. [44] and their utilization MLD models for grade transition control of a polymerization loop reactor. In Stork et al. [99] a variation of the MLD formulation approach was used to generate a MILP model that describes the operation of a fed-batch emulsification containing discontinuities induced by mode transitions in the capillary number.

Bemporad and Morari [18] propose a MLD-based Mixed Integer Predictive Controller (MIPC), where an MIQP (or MILP) generated from an MLD is solved at every time instance. In the paper, they show that it is possible to track a reference trajectory or stabilize an MLD system to an equilibrium point using the algorithm. In terms of online implementation, Morari and Barić [80] showed a case study of a co-generation power plant where the MLD model generated an MILP with 450 continuous variables, 650 integer variables and 2850 constraints. They reported an average solve time of 100 s, shorter than the 1 hr sampling time, which makes a reasonable case for the use of mixed-integer programming models on (almost) real-time applications. An empirical study of the practical solvability of MILPs for dynamic problems was undertaken in Till et al. [105]. The authors found the number of time-steps to be the dominant source of complexity in a MILP-based dynamic problem, which implies that the MIPC method is necessarily limited in the length of its prediction horizon.

Piecewise-Affine (PWA) Models

The precursor to MLDs is a linear hybrid model form called the *Piecewise Affine* (PWA) model. PWAs were first presented in a seminal 1981 paper by E. D. Sontag [98] as means of modeling nonlinear control problems that contained continuous and discrete dynamics. PWA models are based on the idea of partitioning the state-space into polyhedra. Nonlinearities are addressed by performing local linearizations, giving rise to multiple linear models. Evaluation is done by identifying the polyhedron containing the current state and input vectors, and looking up the corresponding values of the parameters required to fully-determine the system $(A_i, B_i, \ldots)$. Switching is accomplished through rules which determine which polyhedral partition to activate.

This switching behavior requires the use of discrete variables to model. A PWA takes the following general form:

$$x_{k+1} = A_i x_k + B_i u_k + f_i \tag{2.23}$$

$$y_k = C_i x_k + D_i u_k + g_i \tag{2.24}$$

$$\text{for } (x_k, u_k) \in P_i$$

where k = discrete time-instant, i = model index, x_k = states, u_k = inputs, y_k = outputs, P_i = polyhedral partition, f_i, g_i = vectors of real-values. Although it is not obvious in the above representation, any MIP-representable switching behavior—such as event sequences—can be modeled within the PWA framework. This is due to the fact that every MIP-representable system has a feasible set that consists of a union of a finite number of polyhedra that represent a disjunction of linear systems [55].

PWAs can be converted into MLDs under some non-restrictive conditions. The equivalence of MLDs, PWAs, LCPs (Linear Complementarity Problems) and other hybrid structures is proven in Heemels et al. [52]. For the purposes of optimization, PWAs are usually first converted into a form treatable using methods designed for MLDs [18] and potentially, ones for LCPs. The value of PWAs today lies primarily in the fact that they provide an intuitive means for visualizing particular classes of hybrid systems.

2.4.2 Nonlinear Hybrid Models

While linear/multilinear models are favored for their mathematical tractability, the degree of nonlinearity encountered in many chemical process systems may warrant the consideration of nonlinear models for a more accurate representation of the true system. In this section, we discuss nonlinear hybrid models developed for optimal control.

Mixed Integer Dynamic Optimization (MIDO) of ODE/DAE-based Models

In recent years, there has been growing interest in the MIDO approach [3, 4], where the discrete behavior in a continuous ODE/DAE-based model is modeled using integer variables. In their most general form, MIDOs are able to deal with the problem of optimizing the transitions between modes (autonomous switching, triggered by events), the combinatorial problem of event sequencing, and other structural discontinuities and nonsmoothness. MIDO has been used to solve problems in integrated design and control of distillation columns [12]. A general MIDO problem can be formulated as:

$$\min_{u(t)} \Phi(x(t), z(t), u(t), \delta(t), p, \theta, t) \tag{2.25}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), z(t), u(t), \delta(t), p, \theta, t), \quad x(0) = x_0 \tag{2.26}$$

$$g(x(t), z(t), u(t), \delta(t), p, \theta, t) \leq 0 \tag{2.27}$$

$$h(x(t), z(t), u(t), \delta(t), p, \theta, t) = 0 \tag{2.28}$$

$$\text{for } t \in [0, t_f]$$

where Φ = objective function, $x(t) \in \mathbb{R}^{n_x}$ = differential state vector, x_0 = initial state parameter vector, $z(t) \in \mathbb{R}^{n_z}$ = algebraic state vector, $u(t) \in \mathbb{R}^{n_u}$ = control input vector, $p \in \mathbb{R}^{n_p}$ = design variable vector (time-invariant variables), $\theta \in \mathbb{R}^{n_\theta}$ = parameter (constant) vector, t_f = final time in prediction horizon. The vector $\delta(t) \in \{0,1\}^{n_\delta}$ can either be treated as time invariant (for structural discontinuities) or time-varying (for dynamic discontinuities).

Mohideen et al. [79] and Avraam et al. [8] applied the full discretization strategy in their MIDO formulation. In their description of the solution method, Avraam et al. modeled switches with binary variables on a time grid. This is known to generate a fairly large Mixed-Integer Nonlinear Program (MINLP) that is challenging to solve except for problems with very small time horizons. Despite this drawback, Avraam et al. emphasized the generality of their approach, which does not presume knowledge of number of mode switches required.

Mohideen et al. [78] and Bansal et al. [13] both developed a decomposition method based on a modified Generalized Benders Decomposition (GBD) for solving MIDOs in a sequential manner. Bansal proposes an innovation that eschews the solution of an intermediate adjoint problem to obtain the master problem. However, the authors acknowledge that the use of GBD-based methods on nonconvex problems in conjunction with local solvers may potentially cut off the feasible solution space in which the global solution lies. Chachuat et al. [30, 31] address this in proposing a method to find the global solution to nonconvex MIDO problems through a combination of bounds tightening, screening model cuts, and convex underestimation of the subproblems in the context of an Outer Approximation type approach.

At the present time, no closed-loop implementations of nonlinear hybrid control algorithms for chemical processes have been reported, as far as we know. The formidable computational expense of solving a mixed-integer nonlinear (nonconvex) dynamic optimization problem renders the approach intractable for many real-time applications (except when the problem instances are very small). Most nonlinear hybrid control schemes are currently only feasible for open-loop implementation, where optimal trajectories are calculated off-line and implemented on-line with a tracking controller.

2.5 State Estimation

A prerequisite for the successful deployment of an advanced control strategy is the availability of accurate and up-to-date knowledge of the plant states. In a complex plant, the complete set of states in the system is sometimes not directly measurable.

State estimation techniques utilize a model-based approach to reconstructing the current state of the system from available plant measurements observed over time. To illustrate the estimation problem, consider the the nonlinear discrete state-space model defined earlier as Eqns. 2.10 – 2.11, but with additive noise terms:

$$x_{k+1} = f(x_k, u_k) + w_k, \quad x_{k=j} = \hat{x}_j \tag{2.29}$$

$$y_k = g(x_k) + v_k \tag{2.30}$$

where w_k is the noise vector representing unmeasured disturbances to the process that affect the states, and v_k is the noise vector representing the error in the measurements (typically noise/random errors in the instruments). The goal of the state estimator is to use the above model to compute an estimate $\hat{x}_j$, given a set of measurements y_j^m. The counter k indicates the discrete time-steps in the model's prediction horizon, whereas j is the control or closed-loop iterate counter.

The theory for the optimal estimation of linear systems is well-developed. The Kalman Filter (KF) [62] algorithm, introduced in the 1960s, represents a minimum-variance state estimator for continuous linear systems with Gaussian, uncorrelated and white noise structures [96].

In the realm of nonlinear state-estimation, the Extended Kalman Filter (EKF) [82] is a technique that is widely applied in the process industries owing to its relative simplicity. The EKF is essentially an application of the linear Kalman Filtering algorithm to a linearized version of a nonlinear model. In the EKF (as well as most other algorithms in the Kalman family), the noise terms are assumed to have multivariate Gaussian distributions (i.e. $w_k \sim N(0, R_w)$ and $v_k \sim N(0, R_v)$) with a zero mean and covariances R_w and R_v respectively. The covariance of the estimate $\hat{x}_j$ is P_j. The basic EKF equations can be compactly summarized as follows:

$$\text{Model linearization: } A_{j-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{j-1},u_{j-1}}, \; H_j = \left.\frac{\partial g}{\partial x}\right|_{x_j^-} \tag{2.31}$$

$$\text{A priori state prediction: } x_j^- = f(\hat{x}_{j-1}, u_{j-1}) \tag{2.32}$$

$$\text{A priori covariance prediction: } P_j^- = A_{j-1}P_{j-1}A_{j-1}^{\mathrm{T}} + R_w \tag{2.33}$$

$$\text{Kalman gain calculation: } K_j = P_j^- H_j^{\mathrm{T}}(H_j P_j^- H_j^{\mathrm{T}} + R_v)^{-1} \tag{2.34}$$

$$\text{A posteriori state update: } \hat{x}_j = x_j^- + K_j[y_j^m - g(x_j^-)] \tag{2.35}$$

$$\text{A posteriori covariance update: } P_j = (I - K_j H_j)P_j^- \tag{2.36}$$

where x_j^- = a priori state estimate, P_j^- = a priori state estimate covariance, K_j = Kalman gain. In bare terms, given a (known) tuple $(\hat{x}_{j-1}, P_{j-1}, y_j^m, u_{j-1})$, the EKF will return the tuple $(\hat{x}_j, P_j, K_j)$. We note that although this formulation is specific to the ODE-derived state-space representation above, the EKF can be trivially extended to handle DAEs by modifying the model linearization step in Eqn. 2.31 using techniques described in Becerra et al. [17].

The Moving Horizon Estimator (MHE) [77] approach is a general recursive nonlinear state observer that obtains its estimates by solving a nonlinear optimization problem. The fact that the MHE involves an optimization step entails associated advantages and disadvantages. On the positive side, because the estimation problem is posed as an mathematical programming problem, state constraints are easily imposed in the estimation. However, the computational difficulty of solving a nonlinear—and usually nonconvex—optimization problem at every sampling instance (in addition to having to solve a nonlinear MPC optimization problem) may limit its use to applications with relatively long sampling periods. However, recent advances in Advanced Step MHE algorithms [124] may alleviate this problem. These algorithms enable the optimal solution of the MHE estimation problem to be rapidly updated using plant measurements and be put to use while waiting for the solution of the next MHE step.

In this work, we employ a filter known as the Unscented Kalman Filter (UKF), due to Julier and Uhlmann [60]. The UKF is a modern derivative-free Kalman-based nonlinear observer with comparable computational complexity vis-à-vis the EKF, but produces estimates with a higher order of accuracy for nonlinear models by eschewing the latter's model linearization step.

2.5.1 The Unscented Kalman Filter (UKF)

One of the central processes in the Kalman Filter involves the propagation of the probability distribution through the system dynamics. In the EKF, this propagation is done through a linearized model. The Unscented Kalman filter is a derivative-free nonlinear state estimation algorithm first proposed by Julier and Uhlmann [60], and is based on the intuition that it is "easier to approximate a probability distribution than a nonlinear function". This approximation is done using a technique known as an Unscented Transform. This involves generating a deterministic, finite set of sampling points–called sigma points–that capture the mean and covariance of a Gaussian random variable. The sigma points are propagated through a nonlinear function (a central operation in any Kalman-based estimator), and the statistics of the transformed points are used to reconstruct the mean and variance of the transformed Gaussian random variable.

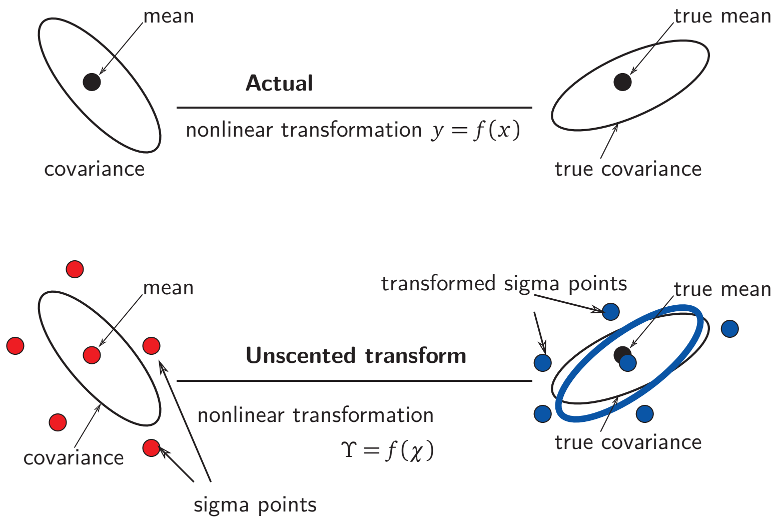


Figure 2.5: The unscented transform for a hypothetical 2-dimensional system. Top figure: nonlinear transformation of random variable x (left to right). Bottom figure: nonlinear transformation of sigma points χ and reconstruction of propagated mean and covariance (left to right).

To illustrate this, consider Fig. 2.5, where a hypothetical variable x (with a certain mean and covariance) undergoes a nonlinear transformation defined by the function $f(x)$. The top figure in Fig. 2.5 shows the true mean and true covariance of x post-transformation. In the bottom figure, a cloud of sigma points χ are scattered around x. These sigma points are carefully chosen such that the moments of x (mean, covariance) can be computed from them. Each individual sigma point then undergoes a transformation f, and the mean and covariance of the transformed points can be used to reconstruct the mean and covariance of the transformed x. The mathematical details of the UKF algorithm can be found in Section 5.4.1.

The UKF enjoys several advantages over the Extended Kalman Filter (EKF). Like the EKF, its computational requirements are modest and it is easy to implement; however, unlike the EKF, it eschews model linearization. This enables the UKF to achieve a higher order of accuracy while propagating the states through nonlinear process dynamics [115]. A concomitant benefit is that no derivatives/Jacobians are required, which makes it a tenable candidate for implementation on systems for which derivatives are difficult to obtain [60]. The UKF treats the underlying model as a black-box, making it suitable for application to DAEs and hybrid dynamic systems (with discrete variables) without any modifications.

In terms of computation, sigma-point transformations are naturally parallelizable tasks (in that each transformation can be performed independently and concurrently), therefore they stand to benefit significantly from modern-day multicore computer processors.

2.5.2 Constrained Kalman Filtering

Being an unconstrained filter, the Kalman filter can potentially give estimates that are physically unrealizable, for instance, negative concentrations in a stream. Fortunately, there exist a number of methods for enforcing state bounds for the estimates arising from Kalman filters, giving rise to a class of filters loosely classified as Constrained Kalman Filters.

An excellent survey of the state-of-the-art can be found in Simon [96]. One of the conclusions was that for linear systems, all of the constrained filters that were reviewed performed identically provided that the constraints are complete and well-posed. For nonlinear systems, no such unification of results exists—the performance of constrained estimation methods can differ from system to system. Kolås et al. proposed a series of modifications to the UKF in which they use information matrices to project the sigma points into the feasible bounded space by solving a QP (or an NLP, depending on the output function). This yields a set of points that obey state constraints. The authors also argue that Constrained Kalman filtering may aid in improving estimation in nonlinear (and non-Gaussian) systems with multimodal probability density functions. This is because state constraints often impose physical meaning on the state estimates, and solutions that lie outside realm of physical realizability are discarded.

2.5.3 Plant-Model Mismatch

In any plant, some degree of plant-model mismatch is inevitable. This can be observed in the deviation between the predictions of the controller model and the observed behavior in the plant. Because plant-model mismatch can cause the incorrect or suboptimal control inputs to be calculated, performing some sort of correction is highly desirable. In control applications, two kinds of mismatch are generally seen:

1. *Parametric Mismatch*
   This occurs the parameters in the model do not match the true (unknown) parameters in a plant. This typically arises due to incorrectly identified parameters, or a parameter whose value has shifted over time.

2. *Structural Mismatch*
   A structural mismatch occurs when the model function is incorrect or is inadequate for modeling the actual behavior in the plant. This can cause a divergence in model predictions and actual plant output.

In the parametric mismatch scenario, parameter estimation can be carried out to update the model parameters using plant measurements. However, if one does not know which parameters are mismatched, alternative compensation schemes are required. One approach is to use time-varying disturbance terms as degrees-of-freedom to adapt the model to match the plant's output. They enter the model as additive noise terms.

Before we discuss the various disturbance estimation methods, it is worth noting that in most nonlinear MPC algorithms, *full-state feedback*(that is, all states in the plant are measured and used to update the controller model) is widely used. It is capable of handling modest amounts of mismatch; however, its only means for model adaptation is through the updating of the states through measurements. Because of this limitation, it is generally incapable of handling drifts and persistent stochastic disturbances due to the lack of integrating disturbances.

State Augmentation with Stochastic States
In one of the early works in joint estimation (described in Section 5.4.1), Kozub and MacGregor [74] propose a method in which the estimation of the disturbances is treated as a parameter estimation problem (where the parameters to be estimated are the disturbances). They do this in the context of an Extended Kalman Filter, where the system is linearized. The augmentation is performed on the state transition matrix and the covariance matrices. The linearized EKF model

is augmented as follows:

$$\begin{aligned} x^d_{k+1} &= A^d x_k + Bu_k + A^s x^s_k, & \hat{x}_{k=j} &= \hat{x}_j \\ x^s_{k+1} &= x^s_k, & \hat{x}^s_{k=j} &= \hat{x}^s_j \end{aligned}$$

where x^d, A^d are the state and state transition matrices of the deterministic states, x^s, A^s are the state and state transition matrices of the stochastic states, and B is the input coefficient matrix. The A^s matrix is tunable matrix that determines the weights of the disturbances entering the system.

The stochastic stationary process states x^s_j add integral action to the estimator. The reason is that the state estimator will continually attempt to converge the state errors to some steady value based on the measurements. If there is a mismatch, a persistent error is created, which the estimator will try to eliminate, thus delivering integral action. A similar approach is discussed in Bequette [19].

Tenny, Rawlings and Wright [103] also advocate an augmentation approach. They show a comparison between input and output disturbance models for handling plant-model mismatch for a nonlinear MPC system. Their nonlinear estimation model can be stated as follows:

$$x_{k+1} = f(x_k, u_k + X_u e_k, w_k) \tag{2.37}$$

$$e_{k+1} = e_k + \xi_k \tag{2.38}$$

$$y_k = h(x_k) + X_y e_k + v_k \tag{2.39}$$

where $x_k \in \mathbb{R}^{n_x}$ are states, $u_k \in \mathbb{R}^{n_u}$ are inputs, and $y_k \in \mathbb{R}^{n_y}$ are outputs. The terms w_k, v_k and ξ_k are stochastic Gaussian zero-mean noise terms. The e_k represents an integrating white noise disturbance term, and X_u, X_y are coefficient matrices that determine how the disturbances enter. In the nonlinear MPC case, if the appropriate number of integrating disturbances are added, offset elimination is either achieved or the closed-loop system is not allowed to reach a steady-state point. This has the effect of removing offset due to unmodeled nonzero mean disturbances and plant-model mismatch. In their work, they show that while an input disturbance model is helpful for offset elimination, there exists a particular class of systems where an input disturbance alone is insufficient to stabilize the system.

The authors point out that the determination of X_u, X_y is an open-question; in practice, these matrices are a complex function of the closed-loop dynamics. To the best of the our knowledge, a constructive method for obtaining them in the general *nonlinear* case has not been devised.

Therefore in most cases they are treated as tuning parameters (similar to the A^s in Kozub and MacGregor [74]) and are chosen by trial-and-error [90].

Moving Horizon Estimation of Disturbances
In Huang [56], state and output disturbances are directly estimated using an advanced-step Moving Horizon Estimator [124]. Input disturbances are not considered. The controller model is as follows:

$$x_{k+1} = f(x_k, u_k) + \eta_j, \quad x_{k=j} = \hat{x}_j \tag{2.40}$$

$$y_k = h(x_k) + \xi_j \tag{2.41}$$

where η_j and ξ_j are disturbance terms state and output disturbance terms respectively. Their values are obtained by solving a maximum likelihood optimization problem in the Moving Horizon Estimator (at every MPC iteration j) with the above perturbed plant model in the NLP constraint set. The parameters η_j and ξ_j are then supplied to MPC controller's prediction model.

2.6 Conclusion

In this chapter, we reviewed the basic problem of buffer inventory coordination. The major solution methods for dynamic optimization problems (simultaneous, sequential and multiple-shooting) were described. An overview of the Model Predictive Control (MPC) framework was offered, with a brief description on the incorporation of economic considerations into such a control framework. This was followed by a discussion of hybrid (discrete-continuous) dynamic optimization and a few dominant model types that are popularly found in the literature. We provide a broad overview of the nonlinear state estimation techniques, with particular emphasis on the Unscented Kalman Filter, and the handling of constraints in such a filter. Various approaches for dealing with plant-model mismatch using disturbance terms were surveyed.

Chapter 3

Dynamic Optimization of Partial Shutdowns

3.1 Introduction

In this chapter, we present several dynamic optimization formulations which we tailor to the problem of partial shutdowns. These formulations are used in the aid of arriving at open-loop advisory policies for operating plants under partial shutdown conditions.

We will also investigate the incorporation of a dynamic optimizer within a nonlinear MPC-type framework for handling partial shutdowns. This model-based partial shutdown controller—which assumes the availability of full-state measurements—provides a means of supplying periodic feedback to the dynamic optimizer.

There are generally two types of control policies for handling process unit shutdowns:

1. **Pre-emptive Policies**, used in situations where a shutdown is known in advance. Pre-emptive policies allow the control system to anticipate a shutdown and to make advance preparations. Example: scheduled maintenance of a reactor.

2. **Reactive Policies**, where a shutdown is unscheduled. Example: equipment failure.

In this thesis, we will confine our attention to reactive shutdown policies. However, we note that our proposed framework is able to accommodate pre-emptive type policies with only very minor extensions.

3.2 Partial Shutdown Optimization Problem

The partial shutdown problem entails finding a set of control trajectories that will restore a plant to some nominal operating state in a cost-optimal fashion. This can be cast in a dynamic optimization problem of the following canonical form:

$$\max \Phi_{\text{econ}}(x(t), z(t), u(t)) \tag{3.1}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), z(t), u(t), \theta), \quad x(0) = x_0 \tag{3.2}$$

$$h(x(t), z(t), u(t), \theta) = 0 \tag{3.3}$$

$$g(x(t), z(t), u(t), \theta) \leq 0 \tag{3.4}$$

$$\text{for } t \in [0, t_f]$$

where Φ_{econ} = economic objective functional, $x(t) \in \mathbb{R}^{n_x}$ = differential state vector, x_0 = initial state parameter vector, $z(t) \in \mathbb{R}^{n_z}$ = algebraic state vector, $u(t) \in \mathbb{R}^{n_u}$ = control input vector, $\theta \in \mathbb{R}^{n_\theta}$ = parameter vector, t_f = final time in prediction horizon. f and h are functions that define a semi-explicit differential-algebraic system, and g is a function that defines the inequality constraints. We assume that the DAE described by 3.2 and 3.3 is of index-1; that is, g is solvable for z (equivalently, we require that g is at least once differentiable, and that its Jacobian with respect to z, or g_z, is nonsingular).

The above DAE optimization problem can be solved in a variety of ways. In this work, we pursue the approach where we discretize $x(t), z(t)$ using the method of orthogonal collocation on finite elements [33] (see Section 6.1.1 for details). This method effectively converts the above DAE system into a finite system of algebraic equations. The resulting equations are then posed as equality constraints in a mathematical programming problem. The mathematical modeling in this work was accomplished using an in-house modeling system, A Modeling Language for Dynamic Optimization (MLDO[1]) [32], which accepts a DAE model as input and generates the requisite discretization constraints in the AMPL mathematical programming language [47].

In the sequel, we will consider the discretized problem directly, where variables $x_k = x(t_k)$, $z_k = z(t_k)$, $u_k = u(t_k)$, for $k = 0, \ldots, N$ where $N = \lfloor t_f / \Delta t \rfloor$ and Δt is the discretized interval length. Time t is partitioned and mapped into N intervals, and $t_{k-1} \leq t \leq t_k$ where $k = 1, \ldots, N$.

[1] See Chapter 6 for details.

3.2.1 Shutdown Constraints (α_{shut}-formulation)

In engineering flowsheet models, each process unit p (where $p \in \mathscr{P}$ = set of all process units) is often modeled individually[2]. Connection equations are used to connect the inputs and outputs of each process unit model. In a chemical engineering context, these connections are often used to couple the material and energy streams of different process units.

In order to represent shutdown behavior in individual process units within the flowsheet, we impose constraints that shut off the material flow F through a particular unit p. Let $\mathscr{I}_p$ denote the index set of all *inlet* flow variables associated with unit p, and $\mathscr{L}_p$, the index set of all variables associated with unit p that *float* when the flowrate is 0. Floating variables include concentrations, compositions, intermediate variables, and other quantities whose values are not defined when the material flow is 0 (see Remark 3.2.4 below). We can now write the following formulation (we which call the α_{shut}-formulation) for representing shutdowns in a process unit p (for time $k = 0, \ldots, N-1$ and $i \in \mathscr{I}_p$, $l \in \mathscr{L}_p$, $p \in \mathscr{P}$):

$$F_{i,k} \geq (1 - \alpha^p_{\text{shut},k}) F^p_{\text{shut}} \tag{3.5}$$

$$F_{i,k} \leq (1 - \alpha^p_{\text{shut},k}) F^U_i \tag{3.6}$$

$$y_{l,k} \geq (1 - \alpha^p_{\text{shut},k}) y^L_l + \alpha^p_{\text{shut},k} y^M_l \tag{3.7}$$

$$y_{l,k} \leq (1 - \alpha^p_{\text{shut},k}) y^U_l + \alpha^p_{\text{shut},k} y^M_l \tag{3.8}$$

where $\alpha^p_{\text{shut},k} \in \{0, 1\}$ = indicator variable representing status of process unit p at time k (1 = shutdown, 0 = normal operation); $y_{l,k}$ = floating variable at time k; y^L_l, y^U_l = lower and upper bound parameters corresponding to y_l; y^M_l = nominal value parameter to which $y_{l,k}$ is pinned to when $\alpha^p_{\text{shut},k} = 1$; $F_{i,k}$ = inlet flowrate variable at time k; F^U_i = upper bound of $F_{i,k}$ for all k; F^p_{shut} = a positive nonzero parameter representing the minimum flowrate threshold before a unit is deemed to have shut down.

One of the consequences of constraints 3.5 – 3.8 is that when $F_{j,k} < F^p_{\text{shut}}$ (that is, when the flowrate falls below a threshold), the unit is deemed to have shut down. For instance, suppose we wish to manually simulate a shutdown in a particular unit p'. This would entail shutdowns in the adjacent units of p' that have no buffer tanks surrounding them. Suppose we denote the set of all shut down units adjacent to p' as $\mathscr{P}_{\text{shut}}$ (where $\mathscr{P}_{\text{shut}} \subset \mathscr{P}$). We can then explicitly

[2]See Appendix A for a flowsheet model description.

impose the following constraint where the values of $\alpha^p_{\text{shut},k}$ are fixed as follows:

$$\alpha^p_{\text{shut},k} = \begin{cases} 1, & p \in \mathscr{P}_{\text{shut}},\ k = k_{\text{start}}, \ldots, k_{\text{end}} \\ 0, & \text{elsewhere} \end{cases} \tag{3.9}$$

where $k_{\text{start}} = \lfloor t_{\text{start}}/\Delta t \rfloor$ = shutdown start time; $k_{\text{end}} = \lceil t_{\text{end}}/\Delta t \rceil$ = shutdown end time (see Fig. 1.2 for the temporal locations of t_{start} and t_{end}). Even though $\alpha^p_{\text{shut},k}$ is a discrete quantity, this explicit fixing of variables (where $\alpha^p_{\text{shut},k}$ are specified parameters in this case) results in the optimization problem being a continuous rather than a discrete one. That said, this α_{shut} formulation lends itself quite naturally to discontinuous (MILP, MINLP) formulations of state-based shutdown triggers by simply redeclaring α_{shut} as binary decision variable vector[3].

Remark 3.2.1. In practice, the value of α_{shut} (as well as an estimate of the value of the expected endpoint of the shutdown, k_{end}) at a given time may be obtained from a plant fault detection module, or set manually by an operator based on past knowledge of similar shutdowns. The variable α_{shut} can be thought of as a flag or a discrete trigger that captures the operational state of the system and allows us to activate the relevant shutdown actions at the right time, such as setting control loops to manual, draining holding vessels etc.

Remark 3.2.2. Constraints 3.5–3.8 give rise to the following mathematical behavior (for a particular unit p):

$$\begin{cases} F_{i,k} \in [F^p_{\text{shut}}, F^U_i], & \text{if } \alpha^p_{\text{shut},k} = 0 \\ F_{i,k} = 0, & \text{if } \alpha^p_{\text{shut},k} = 1 \end{cases}, \ \forall\, i \in \mathscr{I}_p \tag{3.10}$$

$$\text{and,} \begin{cases} y_{l,k} \in [y^L_l, y^U_l], & \text{if } \alpha^p_{\text{shut},k} = 0 \\ y_{l,k} = y^M_l, & \text{if } \alpha^p_{\text{shut},k} = 1 \end{cases}, \ \forall\, l \in \mathscr{L}_p \tag{3.11}$$

Remark 3.2.3. The F^p_{shut} quantity is directly related to a quantity familiar to industrial practitioners, namely the *turndown ratio*, R^p_t, for a unit p:

$$R^p_t = \frac{\text{normal maximum flow}}{\text{minimum controllable flow}} = \frac{F^p_U}{F^p_{\text{shut}}} \tag{3.12}$$

For instance, if due to design reasons a unit is only able to operate at a minimum of 25% of its maximum capacity, the turndown ratio $R_t = 100/25 = 4$. In this context, if the unit is operating

[3]The case in which $\alpha^p_{\text{shut},k}$ is a binary decision variable vector is considered in Chapter 4 of this thesis.

below a threshold F^{p}_{shut} and exceeding the turndown ratio, the unit will need to be shut off.

Remark 3.2.4. Floating variables frequently occur as a result of bilinear formulations. Consider the following stream equations example (where F_P, F_W, F_{DS} are component flows, and x_p is a composition variable):

$$F = F_P + F_w + F_{DS} \tag{3.13}$$

$$F_P = x_P F \tag{3.14}$$

When component flows $F_P = 0$, $F_W = 0$, and $F_{DS} = 0$, it follows that $F = 0$. We can then deduce from Eqn. (3.14) that x_P is allowed to take any value within its given bounds. Physically, x_P is meaningless when $F = 0$ because the composition of a stream with zero flowrate is undefined. Allowing x_p to float in an optimization problem can induce non-unique solutions. It is therefore necessary to pin down the value of x_p when $F_{in} = 0$; this is accomplished through constraints 3.7 – 3.8.

3.2.2 Shutdown Modeling Assumptions

In this work, we assume that the time to restart a failed unit is either: 1) short relative to the overall process dynamics, and is hence neglected, or (2) included in the shutdown duration. The implication of both these assumptions is that a unit is deemed to be immediately ready for operation at the end of a shutdown phase (i.e. at t_{end}; refer to Fig. 1.2). On a similar note, we assume that the shutdown of a unit is perfectly modeled by turning off the inlet/outlet flows to that unit, and that the appropriate shutdown procedure for a unit is carried out. This shutdown procedure itself is not modeled; we assume that a manual or an automated procedure for start-ups and shutdowns is in place [58].

We are disposed to believe that these are reasonable assumptions given that we are essentially solving a macro-level inventory/production problem (that is, we are concerned with the overall ingress and egress of material from process units, as well as the buffer inventories), which is an abstraction level above the actual operating details of the off-line unit.

3.2.3 Restoration constraints

Once the shutdown period is over, the system undergoes a restoration phase where it is brought back to some nominal operating condition. At the end of the restoration phase, we require the independent states and inputs to attain some nominal (steady-state) values. We can express this requirement in the optimization problem by way of restoration constraints, that is, hard terminal constraints imposed on the independent states and inputs post-restoration (during $t \in [t_{\text{res}}, t_f]$ in Fig. 1.2). In this chapter, we assume that the number chosen for the horizon length N is high enough to permit a feasible restoration within its defined time constraints.

Allison [5] demonstrated that in the absence of restoration constraints, the optimal course of action in buffer inventory optimization problems is to drive the storage vessels empty. This is clearly an undesirable outcome in our case as it would deprive the plant of the ability to handle any further shutdowns past the end of the dynamic optimization horizon.

In our partial shutdown problem, we are interested in driving a plant toward a steady-state after the shutdown. Ideally, at the end of the optimization horizon one would like each and every differential state x (including the non-independent states) to return to their exact pre-shutdown values, that is, the original steady-state.

In practice, the non-independent states may not be returnable to their original steady-state values within our finite optimization horizon. Certain states, such as compositions for instance, may take a long time to reach a steady-state point. The time duration required for these states to reach steady-state may lie outside of the time horizon of our dynamic optimizer. We note that one may, at one's discretion, elect to extend the horizon as needed; however, this will incur additional computational cost therefore a balanced approach is advised.

Therefore, instead of attempting to return every state to its pre-shutdown value, we aim to return only the levels of the buffer inventories (independent) to their nominal states at the end of the given horizon, and prescribe a set of inputs that, when held constant, will allow the compositions (non-independent) to eventually settle to their steady-state values. This is done to allow the plant to anticipate further shutdowns.

To steer the system toward a steady-state operating point at the end of the horizon, we employ restoration constraints such as the following in our control optimization: (enforced at the last n_f discrete time points in the prediction horizon)

$$x_{\text{ss}}^{i_x}(1-\epsilon) \le x_k^{i_x} \le x_{\text{ss}}^{i_x}(1+\epsilon), \quad k \in \mathscr{K}_f, \ i_x \in \mathscr{I}_{\text{ss}}^x \tag{3.15}$$

$$u_{\text{ss}}^{i_u}(1-\epsilon) \le u_k^{i_u} \le u_{\text{ss}}^{i_u}(1+\epsilon), \quad k \in \mathscr{K}_f, \ i_u \in \mathscr{I}_{\text{ss}}^u \tag{3.16}$$

where $\mathscr{K}_f = \{N - n_f, \dots, N\}$, $N =$ the horizon length of the controller, $x_k^{i_x}$ and $u_k^{i_u}$ are elements of the state x and input u vectors at time k, indexed by i_x, i_u respectively. The index sets are defined as $\mathscr{I}_{\text{ss}}^{x} = \{i | x_i \in \mathscr{X}_{\text{ss}}\}$ and $\mathscr{I}_{\text{ss}}^{u} = \{i | u_i \in \mathscr{U}_{\text{ss}}\}$, and $\mathscr{X}_{\text{ss}}$, $\mathscr{U}_{\text{ss}}$ are the sets of independent state and inputs variables necessary to specify a steady-state. $\epsilon \geq 0$ is a deviation tolerance percentage parameter, specified as a fraction. $x_{\text{ss}}^{i_x}, u_{\text{ss}}^{i_u}$ are (constant) pre-shutdown values. The time point $(N - n_f)$ is related to restoration endpoint t_{res} as follows:

$$N - n_f = \lfloor t_{\text{res}} / \Delta t \rfloor + 1 \tag{3.17}$$

We wish to emphasize again that only independent states and inputs are to be selected for restoration (a detailed discussion follows in Section 3.2.4 below), for electing to restore all state and input variables to their original values may lead to an over-specified system and may cause convergence difficulties in the presence of plant-model mismatch. In most inventory management problems, typically $\mathscr{X}_{\text{ss}}$ comprises inventory level variables, and $\mathscr{U}_{\text{ss}}$ often contains a subset of the flowrates in the plant.

3.2.4 Determining the subset of states and inputs to specify an endpoint steady-state

A system is said to be at steady-state when its state variables do not change with time. In a dynamic model, this corresponds to the derivatives of its differential states being 0 (that is, $\dot{x} = 0$). A model may have a multiplicity of steady-states.

To illustrate the difference between independent and dependent states and inputs, we present the following example. Consider a dynamic model of the form:

$$\frac{dm}{dt} = F_5 - F_6, \qquad m(0) = m_0 \tag{3.18}$$

$$m\frac{dx^A}{dt} = F_5(x_5^A - x^A), \qquad x^A(0) = x_0^A \tag{3.19}$$

$$F_1 = F_2 + F_3 \tag{3.20}$$

$$F_2 = \alpha F_1 \tag{3.21}$$

$$F_1 x_1^A = F_2 x_2^A + F_3 x_3^A \tag{3.22}$$

$$F_3 + F_4 = F_5 \tag{3.23}$$

$$F_3 x_3^A + F_4 x_4^A = F_5 x_5^A \tag{3.24}$$

where $\alpha, x_1^A, x_2^A, x_4^A$ are constant parameters; m, x^A are differential state variables; F_1, F_4, F_6 are control input variables (decision variables for the optimizer); and $F_2, F_3, F_5, x_3^A, x_5^A$ are algebraic variables. There are 10 unknowns (including the inputs), and 7 equations. Therefore, 3 independent specifications are possible when the system is in transition (that is, non-steady state), and we select the control input variables as the specifications to the system.

At steady-state however, the conditions $dm/dt = 0$ and $dx^A/dt = 0$ impose additional restrictions on the system. They imply that $F_6 = F_5$, therefore F_6 ceases to be an independent input variable at steady-state—its value depends on F_5—and therefore its value cannot be specified arbitrarily. As well, $x^A = x_5^A$, but it can be shown that this can be achieved with any choice of F_1 and F_4 that lie within some feasible operating window, therefore these are independent input variables. Note that the variable m does not appear in the steady-state equations, therefore it may be independently specified.

Through this kind of reasoning[4], we are able to deduce a set of independent states and inputs at steady-state comprising m, F_1 and F_4. If we specify the vectors $x = [m, x^A]^{\mathrm{T}}$ and $u = [F_1, F_4, F_6]^{\mathrm{T}}$, we obtain $\mathscr{X}_{ss} = \{m\}$, $\mathscr{U}_{ss} = \{F_1, F_4\}$, and $\mathscr{I}_{ss}^x = \{1\}$, $\mathscr{I}_{ss}^u = \{1, 2\}$.

If one attempts to specify simultaneously specify F_1, F_4 and F_6 at steady-state, one may either end up with a model whose derivatives dm/dt and dx^A/dt are not exactly zero (as would be the case in the above model), or worse, an overspecified model that is numerically infeasible.

3.2.5 Multi-Tiered Optimization

In order to explain the motivation behind the multi-tiered optimization approach, we first furnish the reader with a brief description of the issues surrounding non-unique solutions in control optimization problems, and the deviation of plant product qualities from their targets during the shutdown and restoration phases.

Input regularization and non-uniqueness

Situations can arise in which the computed optimal input trajectory $u_k \in \mathbb{R}^{n_u}$ (for $k = 0, \ldots, N-1$) is nonunique, that is, there exists more than one input trajectory that corresponds to an identical objective value at the optimum. In mathematical programming parlance, we say that the problem has a set of solutions for which the Sufficient Second-order Optimality Conditions, or SSOCs, does not hold (see Appendix C.2 for a statement of these conditions). When this occurs, a numerical optimization procedure has no recourse for differentiating between the different

[4]In this work, we utilized both a computer algebra system (CAS) and engineering intuition to identify sets $\mathscr{X}_{ss}$, $\mathscr{U}_{ss}$. In principle, these independent sets could conceivably be arrived at through some a graph-theoretic algorithm; however this is outside the scope of our work.

trajectories, and is thus liable to return the first solution trajectory that satisfies some given specified termination condition. Some of these solution trajectories may exhibit a high-frequency chatter-like behavior, while others prescribe large changes in an input variable over a short period of time. Such trajectories, though optimal in a mathematical sense, are undesirable in an operational sense because they give rise to actuator/valve wear and are unappealing to plant operators due their opacity to interpretation.

To avoid the situation, a regularization term is usually added to the objective function to alter the curvature of the objective function (and by extension, the Lagrangian) such that the solution is a unique point. This is typically done through a move suppression penalty on the input trajectory u:

$$\max \Phi_{\text{econ}}(x,z,u) - \sum_{k=1}^{N-1} (u_k - u_{k-1})^{\mathrm{T}} R (u_k - u_{k-1}) \tag{3.25}$$

where $R = \operatorname{diag}[\rho_1, \ldots, \rho_{n_u}]$ is a user-specified penalty weight matrix, and ρ_i (for $i = 1, \ldots, n_u$) are positive-valued penalty weights that are sufficiently-large to create the necessary curvature to obtain a unique solution. The vectors $x = \left[x_0^{\mathrm{T}}, x_1^{\mathrm{T}}, \ldots, x_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = differential states, $z = \left[z_0^{\mathrm{T}}, z_1^{\mathrm{T}}, \ldots, z_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = algebraic states, and $u = \left[u_0^{\mathrm{T}}, u_1^{\mathrm{T}}, \ldots, u_{N-1}^{\mathrm{T}}\right]^{\mathrm{T}}$ = control inputs. The constraints in this optimization problem are assumed to satisfy the Linear Independence Constraint Qualification (LICQ), a regularity property (see Definition C.0.2 in Appendix C).

A concomitant benefit of having this move-suppression penalty term in the objective is that one also obtains smooth trajectories. This technique is often successfully applied in control optimization problems where the object is either to track a setpoint or to regulate a system around an equilibrium point.

However, in our case we have an economic objective function that is denominated in dollar terms. Therefore there is a certain degree of arbitrariness to the selection of the value of the penalty weights, since the penalty term is not denominated in dollar terms and does not represent any manner of economics. An arbitrarily configured move suppression penalty is susceptible to trading-off the original economic objective in ways that are not easy to quantify. Therefore in this work we elect not to use the move suppression penalty approach.

Product Quality Targets during Shutdown and Restoration

Product qualities (e.g. percentage impurity, minimum concentration, etc.) are a set of variables that are closely regulated around some target value during nominal operation. These variables are monitored to ensure that the material at various stages of the plant have the desired properties for processing.

During the shutdown and restoration phases, some of these product quality variables may be rendered incapable of reaching their targets due to the disruption caused during partial shutdowns. In practice, this necessitates the relaxation or release of a subset of the specifications during the transition period. This entails allowing excursions of certain product qualities outside their normal allowable bands while maintaining the process as close to its nominal operating state as possible.

The task of choosing the most suitable set of product quality variables to relax or release often falls on the plant operator. This is a task that calls for either the use of engineering heuristics or personal experience. However, we believe a case can be made for using an optimization technique for selecting trajectories that attempt to meet the targets in the non-shutdown units as best as possible, without sacrificing economics.

Multi-tiered Optimization

In light of the above, we propose an alternative approach to regularizing the trajectories, which takes the form of a hierarchical optimization strategy. In this section, we put forward a multi-tiered optimization approach, which extends the "two-tiered optimization" approach due to Balthazaar and Swartz [101]. This approach directly exploits the non-uniqueness property of the dynamic optimization problem by systematically selecting a solution—within the space of optimal solutions—that meets certain desirable criteria.

The multi-tiered optimization approach allows us to prioritize several competing objectives and specify the trade-offs between them. Separate optimization problems are solved one after the other, with the preceding tiers providing information to the latter tiers. One configuration of a multi-tiered optimization problem is as follows: (in all the tiers described below, we assume that the constraint set includes the dynamic model.)

1. **Tier 1: Obtain trajectory for optimal economics**
   We solve the dynamic optimization problem that maximizes an economic objective function, and record the optimal value of the objective. This value represents an upper-bound on economics achievable by the system.

$$\Phi^*_{\text{econ}} = \max \Phi_{\text{econ}}(x, z, u) \tag{3.26}$$

where $\Phi_{\text{econ}}(x,z,u)$ = economic objective function, $x = \left[x_0^{\mathrm{T}}, x_1^{\mathrm{T}}, \ldots, x_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = differential states, $z = \left[z_0^{\mathrm{T}}, z_1^{\mathrm{T}}, \ldots, z_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = algebraic states, and $u = \left[u_0^{\mathrm{T}}, u_1^{\mathrm{T}}, \ldots, u_{N-1}^{\mathrm{T}}\right]^{\mathrm{T}}$ = control inputs. The optimal value of the economic objective (a scalar value) is stored as $\bar{\Phi}_{\text{econ}} := \Phi^*_{\text{econ}}(x^*, z^*, u^*)$. We note that in this tier, the product quality targets are allowed to vary within their admissible bounds. In the cases considered in this chapter, the quality targets constitute (intermediate) operational targets that are non-critical to the final product quality. In other applications, these targets may be essential, in which case Tiers 1 and 2 may be swapped and their trade-offs appropriately reassigned.

2. **Tier 2: Minimize deviation from product quality targets**
In this tier, we solve an optimization problem to deduce the minimum tolerable deviation of product qualities from their target values during the shutdown and restoration phases, while still remaining feasible. Let s denote the vector of product quality variables in the plant, and $\bar{s}$ their nominal targets.

$$\Phi^*_s = \min \sum_{k=0}^{N} (s_k - \bar{s})^{\mathrm{T}} (s_k - \bar{s}) \tag{3.27}$$

$$\text{s.t. } \Phi_{\text{econ}}(x, z, u) \geq \bar{\Phi}_{\text{econ}}(1 - \varepsilon_e) \tag{3.28}$$

where $\varepsilon_e \geq 0$ is the economic trade-off parameter. It is a tuning parameter representing the maximum fraction of achievable economics (i.e. $\bar{\Phi}_{\text{econ}}$) to trade-off. The squared deviation of the product quality targets at the optimal solution ($\bar{\Phi}_s := \Phi^*_s$) is recorded and enforced in the next tier.

3. **Tier 3: Minimize control effort**
The objective function is replaced with another objective that minimizes control effort, subject to a minimum constraint on the economic performance and minimum deviation from product quality targets.

$$\min \sum_{k=1}^{N-1} (u_k - u_{k-1})^{\mathrm{T}} (u_k - u_{k-1}) \tag{3.29}$$

$$\text{s.t. } \Phi_{\text{econ}}(x, z, u) \geq \bar{\Phi}_{\text{econ}}(1 - \varepsilon_e) \tag{3.30}$$

$$\sum_{k=0}^{N} (s_k - \bar{s})^2 \leq \bar{\Phi}_s(1 + \varepsilon_s) \tag{3.31}$$

where $\epsilon_s \geq 0$ is the product quality deviation trade-off parameter. It is a user-specified tuning parameter which represents the fraction of the attainable target deviation to trade-off in order to achieve minimum control effort. Choosing $\varepsilon_e = 0, \varepsilon_s = 0$ may be feasible, but higher values should deliver smoother control profiles, which may be more desirable from an operations point of view. Positive values of ε_e and ε_s are also favorable from a numerical convergence point of view, as they behave like numerical relaxation factors.

We wish to emphasize that the above set-up represents only one possible configuration of a multi-tiered dynamic optimization problem – many more are possible. Multi-tiered optimization can be thought of as a general technique for systematically prioritizing control objectives (an idea explored in Swartz [100]) without resorting to using arbitrary penalty weights; instead it allows one to specify the trade-offs between one objective and the next. The first tier represents the most important objective and last tier represents the least important. The relaxation factors ε are used to specify the trade-offs in each tier. Therefore the order of the individual optimization problems can be switched around according to their importance in a particular application. Indeed, even the optimization problems in each tier can be replaced or removed as needed. Fig. 3.1 shows a simple schematic of the multi-tiered optimization approach that we have adopted for this chapter of the thesis. We also note that a nice property of this approach is that if the first tier optimization problem is feasible, feasibility is guaranteed for all successive tiers.

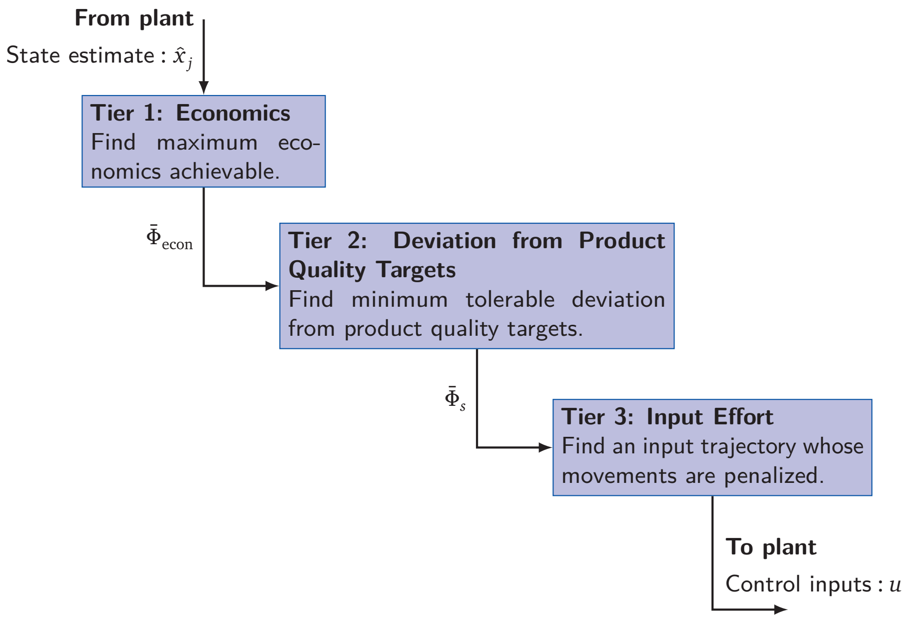

From plant
State estimate : $\hat{x}_j$
Tier 1: Economics
Find maximum economics achievable.
$\bar{\Phi}_{\text{econ}}$
Tier 2: Deviation from Product Quality Targets
Find minimum tolerable deviation from product quality targets.
$\bar{\Phi}_s$
Tier 3: Input Effort
Find an input trajectory whose movements are penalized.
To plant
Control inputs : u

Figure 3.1: Multi-tiered optimization approach.

Another attractive feature of this particular multi-tiered optimization configuration is that unlike the move-suppression penalty approach seen in eqn. 3.25, it permits the user to decide, for example, the exact amount of the economics to trade-off (in dollar terms as opposed to some arbitrarily weighted quantity) in order to obtain smaller product quality deviations or smoother input trajectories. As well, one can specify the percentage of attainable product quality deviation to trade-off in order to obtain smoother trajectories with the minimum input effort.

3.3 Nonlinear MPC for Partial Shutdowns

In previous sections, we have presented dynamic optimization formulations for addressing partial shutdowns in a plant. The dynamic optimization problem can be solved off-line and its solution used as an advisory tool for making plant operation decisions. Plant operators can use dynamic optimization as a tool for assessing what-if scenarios, based on different hypothetical operating strategies. This is a common way of using the results from dynamic optimization problems.

This usage case is akin to the use of a process simulator to simulate different scenarios, except dynamic optimization offers the additional facility of being able to compute *optimal* solutions based on different objective functions. In addition, while process simulation software often include the ability to optimize certain time-invariant plant *parameters*, dynamic optimization techniques offer the ability to optimize time-varying control input *trajectories* that can be made to obey arbitrarily complex constraints (subject to feasibility). As such, open-loop dynamic optimization is a tremendously useful tool for generating plant operating policies and heuristics.

In principle, it may be possible to directly implement the dynamic optimization solution trajectories on the plant. We note that in reality, process disturbances, noise, and plant-model mismatch may invalidate or render infeasible the calculated trajectories. For the implementation of the solution trajectories to be practicable, they need to reflect the current realities in the plant.

One could conceivably update the dynamic optimization problem with the latest information available from the plant and re-solve the problem. This could be done every time an operational decision has to be made. However, if we take the idea one step further and mechanize/algorithmize the procedure, it leads us naturally to the notion of feedback control, where a control algorithm takes an operational decision based on updated plant information at every sampling iteration.

Model Predictive Control (MPC) [91] is a framework that is particularly well-suited to handle this case. Toward this end, we propose a nonlinear MPC-based shutdown controller (with an embedded dynamic optimization problem) as a means of implementing abnormal-situation control on the plant. We selected MPC for its centralized multivariate structure and constraint-handling capabilities. Given that our primary objective is economics-based, the predictive controller here is distinct from a conventional MPC controller in that its objective is not to track given setpoints, but instead to implement manipulated variable adjustments to optimize an economic objective (as well as satisfy lower tier objectives like minimizing deviation of product qualities from their targets, etc.).

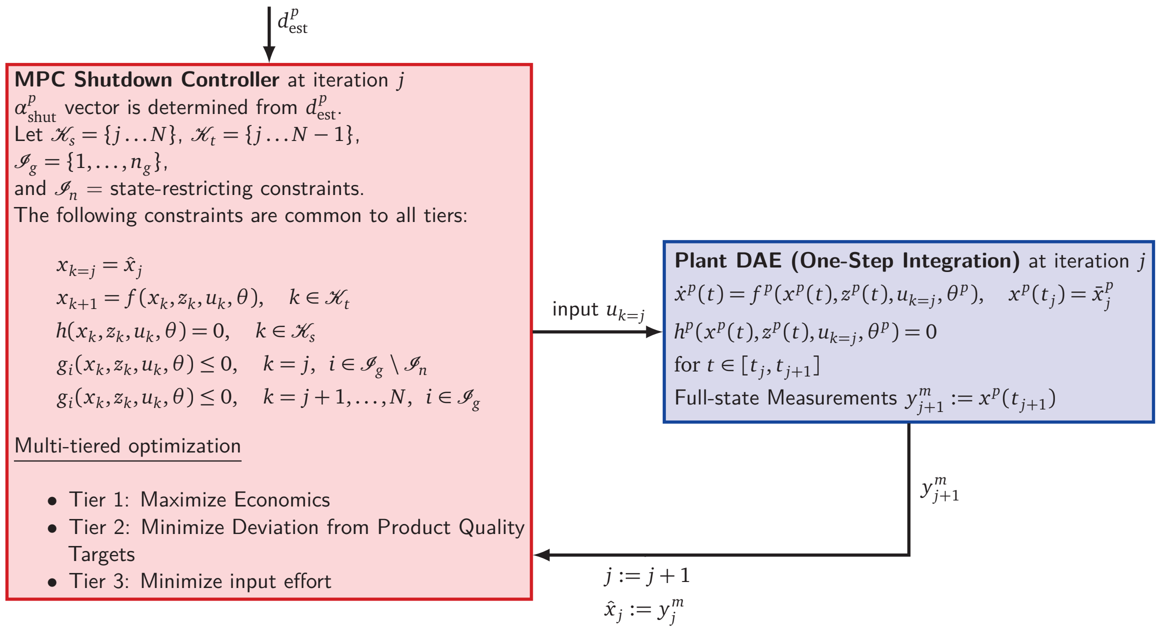

d^p_{est}
MPC Shutdown Controller at iteration j
α^p_{shut} vector is determined from d^p_{est}.
Let $\mathscr{K}_s = \{j \dots N\}$, $\mathscr{K}_t = \{j \dots N-1\}$,
$\mathscr{I}_g = \{1, \dots, n_g\}$,
and $\mathscr{I}_n$ = state-restricting constraints.
The following constraints are common to all tiers:
$x_{k=j} = \hat{x}_j$
$x_{k+1} = f(x_k, z_k, u_k, \theta), \quad k \in \mathscr{K}_t$
$h(x_k, z_k, u_k, \theta) = 0, \quad k \in \mathscr{K}_s$
$g_i(x_k, z_k, u_k, \theta) \leq 0, \quad k = j, \ i \in \mathscr{I}_g \setminus \mathscr{I}_n$
$g_i(x_k, z_k, u_k, \theta) \leq 0, \quad k = j+1, \dots, N, \ i \in \mathscr{I}_g$
Multi-tiered optimization
• Tier 1: Maximize Economics
• Tier 2: Minimize Deviation from Product Quality Targets
• Tier 3: Minimize input effort
input $u_{k=j}$
Plant DAE (One-Step Integration) at iteration j
$\dot{x}^p(t) = f^p(x^p(t), z^p(t), u_{k=j}, \theta^p), \quad x^p(t_j) = \bar{x}^p_j$
$h^p(x^p(t), z^p(t), u_{k=j}, \theta^p) = 0$
for $t \in [t_j, t_{j+1}]$
Full-state Measurements $y^m_{j+1} := x^p(t_{j+1})$
y^m_{j+1}
$j := j+1$
$\hat{x}_j := y^m_j$

Figure 3.2: MPC simulation workflow. Iterations terminate when $j = N$.

3.3.1 Algorithm

In our scheme, the MPC-based shutdown controller is activated at the beginning of a shutdown (t_{start}), and takes over from the nominal control system. We provide a description of the steps carried out by the MPC algorithm below.

MPC Shutdown Controller Algorithm

Fig. 3.2 summarizes the shutdown controller algorithm. Note that it is important to distinguish between the model's discrete time counter k, and the MPC iteration counter j.

1. **Downtime Estimate** (d^p_{est}). At the beginning of the shutdown, a value for the downtime estimate, $d^p_{\text{est}} = t^{\text{est}}_{\text{end}} - t_{\text{start}}$ (where $t^{\text{est}}_{\text{end}}$ is the estimated shutdown end

time) is supplied to the shutdown controller. This downtime estimate parameter can be manually updated during any control interval as new information about the downtime becomes available. This is an instance of manual *parameter* feedback. A detailed description can be found below, in Section 3.4. The values of α^p_{shut} for all process units $p \in \mathscr{P}$ are determined from this parameter.

2. **Open-loop Dynamic Optimization**. The MPC dynamic optimization problem (in the "MPC Shutdown Controller" block) is solved using a multi-tiered optimization approach (described in Section 3.2.5).

3. **Implement Control Inputs on Plant**. Only the current step (time j) of the calculated inputs ($u_{k=j}$) is implemented on the plant.

4. **State Measurements**. Plant state measurements at the next sample-time are taken. In a simulation environment, a 1-step integration is performed from t_j to t_{j+1} (in the "Plant DAE (One-Step Integration)" block); where x^p, z^p = plant model variables; θ^p = plant model parameters; f^p, h^p = plant DAE model functions. The internal states of the plant, $\bar{x}^p_j$ are the initial values for the DAE. The vector y^m_{j+1} is conveyed back to the predictive controller, where it is used as initial values for the next iteration. In this work, we assume full-state feedback. If full state measurements are not available, a state observer is employed.

5. **Iterate**. The counter j is incremented. This process is repeated from Step 1 until the system is restored to its nominal steady state operation (at time t_{res}, see Fig. 1.2), after which control is handed back to the nominal control system (when $j = N$).

The transient processes we are considering (namely partial shutdowns) are finite in duration, therefore the prediction/control horizon length in the controller is finite. Instead of receding prediction and control horizons found in conventional MPC, *shrinking* prediction and control horizons are used. (The prediction and control horizon lengths decrease as the the controller advances toward the end of the time horizon.) Here, a shrinking horizon is selected for reasons of computational efficiency and ease of problem initialization, but in practice, a receding horizon is also admissible.

The horizon length N needs to correspond to the duration of the shutdown and restoration combined. If a shorter length is chosen, suboptimal control may result from the controller not having an adequate picture of the full transient process. Also, in our predictive controller, the control horizon length equals the prediction horizon length. Economics optimization is integrated directly into the predictive controller, which constitutes a 1-layer MPC with economics approach (in contrast to a layered Dynamic RTO approach).

Removal of state-restricting model constraints at the initial point

To ensure model feasibility for closed-loop applications, we require a modification to the original dynamic optimization problem. At every MPC iteration j, inequality model constraints that restrict the value of the initial condition vector $x_{k=j}$ in the MPC model are to be released. The reason for this is that at every iteration, the values of the initial condition vector $x_{k=j}$ are fixed to the current state estimate vector $\hat{x}_j$. This vector is computed from plant measurements, and constitutes a specified input to the MPC dynamic optimization model. It is not a degree-of-freedom in the model, therefore its range of admissible values should not be constrained.

To avoid this situation, at every MPC iteration, we need to release the appropriate state constraints at the initial point. We can do so by partitioning the inequality constraints (and rewriting it for our shrinking horizon case) as follows:

$$g_i(x_k, z_k, u_k, \theta) \leq 0, \qquad k = j, \quad i \in \mathscr{I}_g \setminus \mathscr{I}_n \tag{3.32}$$

$$g_i(x_k, z_k, u_k, \theta) \leq 0, \qquad k = j+1, \dots, N, \quad i \in \mathscr{I}_g \tag{3.33}$$

where $\mathscr{I}_g = \{1, \dots, n_g\}$ is the index set of all inequality constraints, $\mathscr{I}_n \subset \mathscr{I}_g$ is the index set of the model constraints that are restricting the values of the states $x_{k=j}$. This partitioned constraint system implies that at time $k = j$, the state-restricting model constraints are excluded, but at $k = j+1, \dots, N$, all constraints are enforced. Enforcing inequality constraints from $j+1$ onwards is a practice that agrees with the nonlinear MPC formulations in Qin and Badgwell [91], in which inequality constraints on u are enforced from time j whereas inequality constraints in y are enforced from $j+1$.

Remark 3.3.1. Because the values of the algebraic variables z_k are determined from the values of x_k, u_k and θ, it is possible that a given $\hat{x}_j$ at the initial point may cause the corresponding z_j to be in violation of certain constraints that restrict the values of $z_{k=j}$, especially if those constraints are not physical ones. If this situation is encountered, the indices of the offending constraints should be included in the set $\mathscr{I}_n$.

3.4 Parametric Feedback for Addressing Uncertainty in Downtime Estimate, d_{est}

Explicitly known discrete events (such as time and duration of a planned shutdown and restoration) are embedded directly into the prediction model of our proposed MPC controller. These events are specified either through operator intervention, look-up tables, or are automatically prescribed by a fault monitoring module. This form of process event anticipation is distinct from traditional feedforward control in which disturbances are detected solely through plant measurements.

The estimated duration of the shutdown ($d^p_{\text{est}} = t^{\text{est}}_{\text{end}} - t_{\text{start}}$) is a important parameter in calculation of the optimal trajectories because it influences the shape of the resulting trajectories. The downtime estimate would typically be provided by the operator to the control system, based on past operational experience or direct information about the prognosis of the shutdown. In practice, this estimate will not correspond exactly to the actual downtime (d_{act}) for various reasons, not least due to the fact that it may be very difficult in most cases to make an accurate prediction of a unit's downtime.

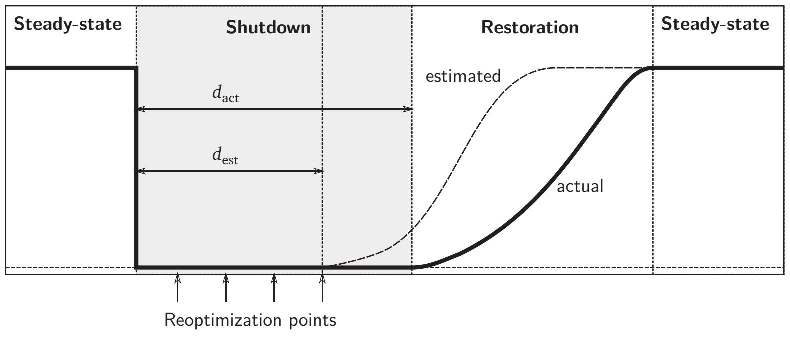


Figure 3.3: Reoptimization at various stages, with $d_{\text{act}} > d_{\text{est}}$.

One approach to addressing this issue is to treat the downtime estimate as a parametric feedback variable. In contrast to other feedback variables, the downtime is a single parameter *manually* supplied by maintenance personnel. By adopting this approach, the operator is allowed to revise any downtime estimates dynamically. At any time during the shutdown, the operator can enter corrected downtime estimates as information arrives and the remainder of the trajectory is re-optimized from the current state of the system, and the controller performs what is essentially a mid-course correction. This is in illustrated in Fig. 3.3, where the estimated downtime d_{est} is shorter than the actual downtime d_{act}. The actual trajectory corresponds to those obtained after reoptimization.

The effectiveness of active reoptimization depends on:

1. the degrees-of-freedom available to the system at the point of reoptimization.
2. how early the reoptimization is done.
3. where the failing unit is situated with respect to buffers.

For uncertainty in the estimated downtime parameter, we have found this feedback approach to possess considerable advantages over a multi-scenario optimization approach (see Section 6.1.4 for a brief description), in that the resulting control trajectories of the former are less conservative than the latter. This observation arises from the fact that the duration of shutdowns is usually inversely proportional to the economics of a plant.

3.5 Application Study

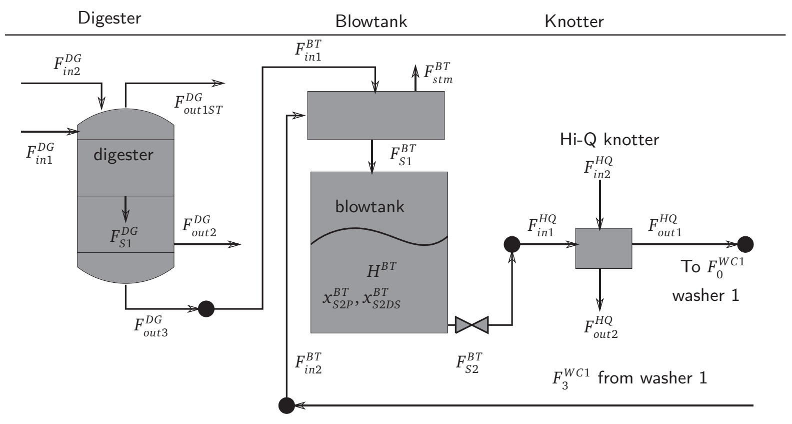
3.5.1 Process Description

The model used for this partial shutdown study is the fiber line of a Kraft pulp and paper plant. This study was inspired by an actual industrial problem in a plant in Ontario, Canada. The mathematical equations of the model and full nomenclature can be found in Appendix A.

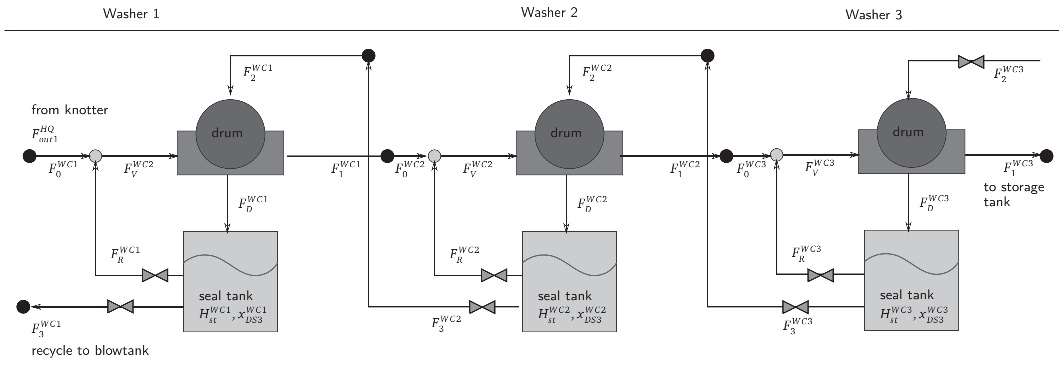
A simplified schematic of the plant in shown in Fig. 3.4. There are two major buffering tanks, the blowtank and the storage tank located before the delignification department. There are also buffering tanks (sealtanks) in the washing department which are used for internal recycles. Each variable name is tagged with a superscript indicating the unit it is associated with. The mapping between the tag names and their corresponding unit names is found in Table 3.1.

Tag	Unit Name	Tag	Unit Name
DG	digester	BT	blowtank
HQ	Hi-Q knotter	WC1	brownstock washer 1
WC2	brownstock washer 2	WC3	brownstock washer 3
ST	storage tank	SR	screeners
OF	O_2 feedpress	MX	mixer
OR	O_2 delignification reactor	PW	Post-O_2 washer

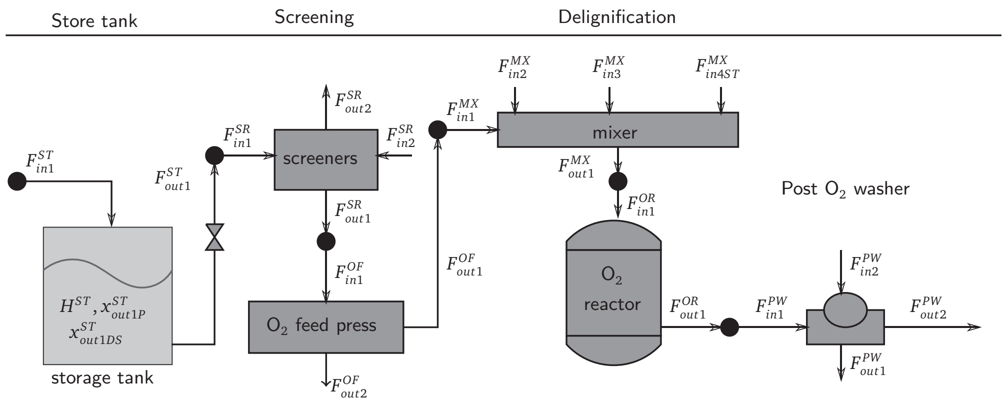
Table 3.1: Variable superscript tag names and corresponding unit names.



(a) Digestion and knotting



(b) Brownstock washing



(c) Screening and delignification

Figure 3.4: Simplified schematic of plant model.

When a process unit shuts down, an entire department is typically usually forced to shut down unless there is internal buffering capacity in the departments. A process unit is shut down and taken off-line for a period of time, and is subsequently restored. Based on an estimate of the downtime (specified by the operator), the proposed control scheme is used to compute and implement a set of optimal control trajectories that accommodates the shutdown in the presence of nonlinear dynamics and recycles.

The manipulated variables and their bounds are given in Table 3.2. We assume full-state measurements are available. The sets of independent states and inputs required to specify the steady-state at the end of the restoration period are:

$$\mathscr{I}^x_{\text{ss}} = \{i | x_i \in \mathscr{X}_{\text{ss}}\}, \qquad \mathscr{X}_{\text{ss}} = \{H^{BT}, H^{WC1}, H^{WC2}, H^{WC3}, H^{ST}\} \tag{3.34}$$

$$\mathscr{I}^u_{\text{ss}} = \{i | u_i \in \mathscr{U}_{\text{ss}}\}, \qquad \mathscr{U}_{\text{ss}} = \{F^{DG}_{in1}, F^{WC1}_{R}, F^{WC1}_{3}, F^{WC1}_{2}, F^{WC2}_{R}, F^{WC3}_{R}, F^{WC3}_{3}\} \tag{3.35}$$

The capacities of the buffer tank inventory units and their nominal levels are given in Table 3.3. Table 3.4 lists product quality variables and their targets. This work uses the simultaneous method for dynamic optimization, where both controls and states are discretized. Orthogonal collocation was used for discretization, resulting in an optimization problem with approximately 18,200 variables, 17,600 equality constraints, 460 inequality constraints, and 53,000 nonzeros in the constraint Jacobian.

Manipulated Variables	**Min** (ton/hr)	**Max** (ton/hr)
F^{DG}_{in1}	0	40
F^{BT}_{S2}	0	600
F^{WC1}_{3}	0	600
$F^{WC1}_{2} = F^{WC2}_{3}$	0	600
F^{WC1}_{R}	0	600
F^{WC1}_{D}	0	600
$F^{WC2}_{2} = F^{WC3}_{3}$	0	600
F^{WC2}_{R}	0	600
F^{WC2}_{D}	0	600
F^{WC3}_{2}	0	600
F^{WC3}_{R}	0	600
F^{WC3}_{D}	0	600
F^{ST}_{out1}	0	600
F^{SR}_{out1}	0	600
F^{OF}_{out1}	0	600
F^{PW}_{out2}	0	600

Table 3.2: Decision variables and their bounds

Tank Name	**Capacity**, V_{tank} (m^3)	**Nom. Inven.** (m^3 / % of capacity)
Blowtank	2050.0	1230.0 / 60%
Sealtank #1	1067.0	248.6 / 23.3%
Sealtank #2	861.0	288.4 / 33.5%
Sealtank #3	683.0	227.4 / 33.3%
Storage tank	1620.0	972.0 / 60%

Table 3.3: Tank names, capacities and nominal inventories

Product Quality Variable	**Description**	**Target Value** (%)
C_M^{SR}	Outlet consistency of screeners	4.5
C_M^{OF}	Outlet consistency of O_2 feedpress	30.0
C_M^{PW}	Outlet consistency of Post-O_2 washer	6.5

Table 3.4: Product quality variables and their targets.

The economic objective function is a linear function of the cumulative profit (or loss) over the horizon of interest:

$$\Phi_{\text{econ}} = \Delta t \sum_{m=1}^{n_m} \left[C_m \left(\sum_{k=0}^{N-1} F_{m,k} \right) \right] \tag{3.36}$$

where Δt is the sample time, F_m is the flowrate (in ton/hr) of some component m, C_m is the price that component (\$/ton), and n_m is the total number of components. The prices C_m can be found in Table 3.5.

Material	**Prices**, C_m (\$/ton)
Pulp	725.00
Chips	-25.00
Chemicals	-100.00
Energy from Black Liquor	0.35

Table 3.5: Average prices of materials [11, 48]

The solvers used in this work were IPOPT 3.8.2 [114] (a large-scale sparse solver that implements a primal-dual interior point method) and CONOPT 3.14G [6] (a feasible path active-set solver). IPOPT was used in the open-loop studies, as well as to generate an initial guess for the first iteration of the shutdown controller. CONOPT was used for closed-loop studies due to its amenability to warm-starts. Its ability to follow a feasible path is also particularly advantageous for efficiently solving multiple multi-tiered optimization problems, as the solution for one tier is a feasible initial point for the next.

3.5.2 Case 1: Nominal Open-loop Optimization of Shutdowns

Description

In this case study, we demonstrate the application of aforementioned methodologies for the dynamic optimization of three separate shutdown scenarios. In Scenario (a), the O_2 Delignifier unit is shut down, and Scenario (b) involves the shutdown of the Hi-Q knotting unit, while in Scenario (c), the digester unit goes offline.

The plant is assumed to be operating at steady-state initially, and the shutdown occurs suddenly. Flowrates into the shut down unit (as well as adjacent quasi-steady-state units) dip to 0. Past the shutdown period, the optimizer responds by prescribing input actions to restore the plant to its steady-state.

In each case below, the simulation horizon is 12 hours and the shutdown duration is 5.5 hours. Here, we assume we know the downtime exactly and that there are no uncertainties. The details are summarized in Table 3.6.

Case 1 Summary (Figs. 3.5)	
Units shut down:	Shutdown scenarios (a) O_2 delignifier; (b) Hi-Q knotter; (c) digester.
Simulation horizon (h):	12
Shutdown start time (h):	0.25
Estimated downtime, d_{est} (h):	5
Actual downtime, d_{act} (h):	5
Reoptimization point (h):	N/A
Disturbances:	None

Table 3.6: Case 1 summary, scenarios (a), (b) and (c)

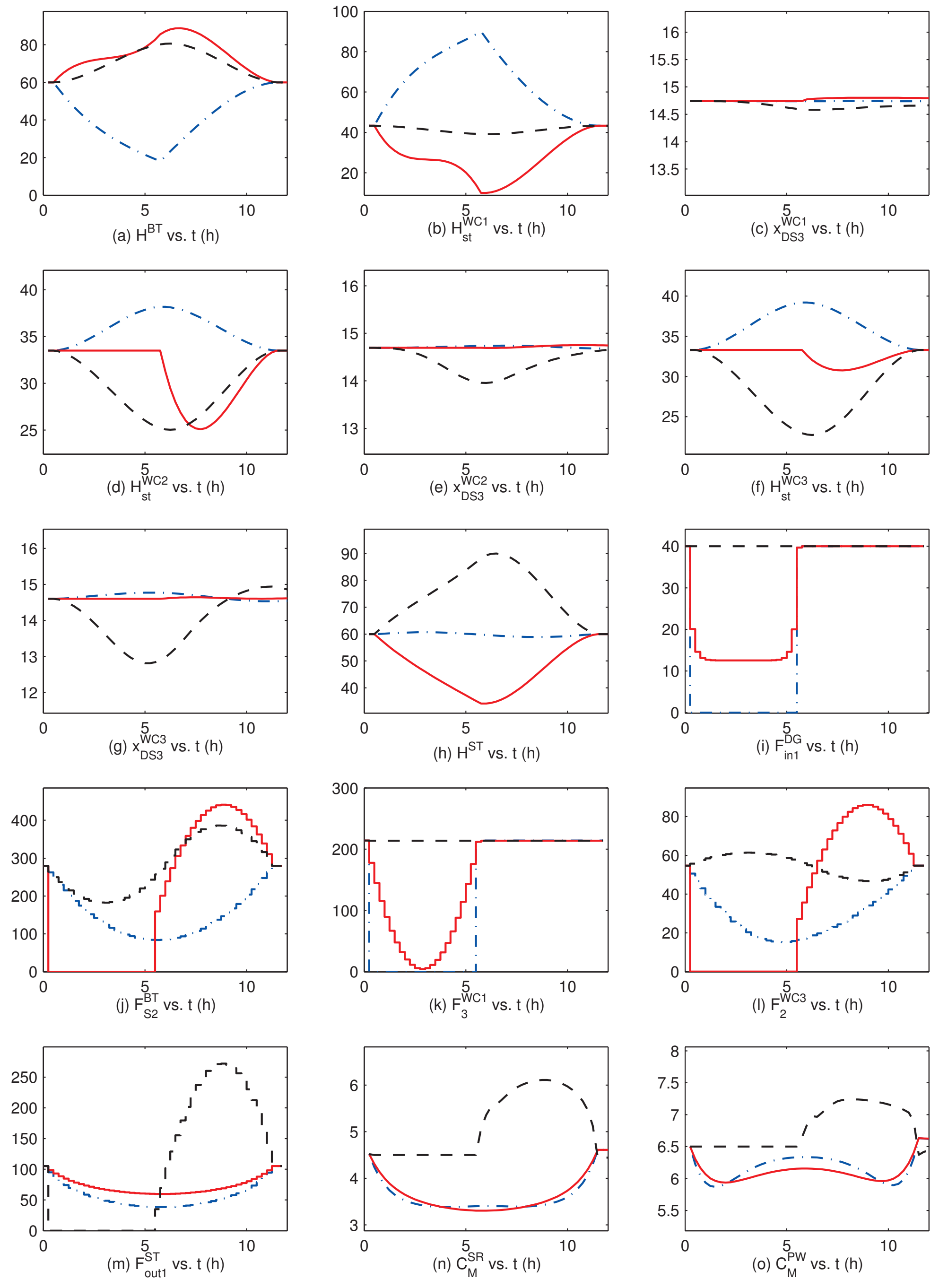


Figure 3.5: *Case 1: Open-loop dynamic optimization.* Solution trajectories for three distinct shutdowns. **Legend**: Dashed lines = Scenario (a) delignifier shutdown. Solid lines = Scenario (b) Hi-Q knotter shutdown. Dotted dashed lines = Scenario (c) digester shutdown.

Results

The results of the three scenarios are as follows:

- *Scenario (a): Shutdown in the O_2 delignifier*

  The overall economics achieved was \$92,963. The trajectories are represented by the dashed lines in Fig. 3.5. The flowrate out the the storage tank, F^{ST}_{out1} (Fig. 3.5m) goes to 0 during the shutdown interval $t = 0.25 - 5.75\,\text{h}$. This has the effect of shutting down the O_2 delignifier and all the surrounding units (up to the screening unit), as there are no buffer capacities in units between it and the storage tank.

  As the shutdown is progressing, we notice a rise in the storage tank level H^{ST} (Fig. 3.5h), due to accumulation of material from upstream units. The digester outlet stream F^{DG}_{in1} (Fig. 3.5i) continues to produce material at its nominal rate, while the outlet flowrate in the blowtank F^{BT}_{S2} (Fig. 3.5j) drops to accommodate the constriction downstream. This causes the a rise in the blowtank level H^{BT} (Fig. 3.5a).

  After the shutdown period is over ($t > 5.5\,\text{h}$), the system attempts to make up for lost production by discharging the contents of the storage tank, as seen in a decline in its level H^{ST}. Two product quality variables C^{SR}_{M} (Fig. 3.5n) and C^{PW}_{M} (Fig. 3.5o) (which are associated with the screening and post-O_2 washing units) can be seen to be deviating from their targets during the restoration period, in response to the material influx during the restoration period. All inputs and states return to their original steady-states post-restoration.

  Because a shutdown in the O_2 delignifier is downstream of all the major production units and its adjacent buffering tank is capable of absorbing the majority of the product being produced upstream, the overall effect of this partial shutdown on the plant is less severe than that of partial shutdowns in units upstream. This is witnessed by the fact that the economics achieved is considerably higher than that of the next two cases.

  We have included upper-bound constraints on the flowrates in this study. However depending on the plant in question, these upper-bounds may be lower than what is specified here. The operating windows of specific units may be accounted for through the specification of the appropriate constraints.

- *Scenario (b): Shutdown in the Hi-Q knotter*

  The overall economics achieved was \$65,292, which is lower than that of a partial shutdown in the downstream delignifier unit above.

The trajectories are represented by the solid lines in Fig. 3.5. This shutdown is indicated by the turning off of flows to all units bounded by the blowtank and the storage tank, that is, from the Hi-Q knotter to Washer 3. The seal tanks of the washers however remain operational as buffer capacities. The blowtank outlet flowrate F_{S2}^{BT} (Fig. 3.5j) is seen to go to 0. All internal recycles in the washing circuit are also at 0 during the shutdown. However, the main recycle stream from Washer 1 seal tank to the blowtank, F_3^{WC1} (Fig. 3.5h) continues to be a degree-of-freedom, therefore the level of the Washer 1 seal tank, H_{st}^{WC1} (Fig. 3.5b) can be seen to be changing during the shutdown.

During the shutdown period, production continues both upstream and downstream of the blowtank and storage tank respectively. This is seen from a rise in H^{BT} (Fig. 3.5a) and a dip in H^{ST} (Fig. 3.5h). The production rate of the digester also drops during the shutdown, as seen in F_{in}^{DG} (Fig. 3.5i). This is to prevent overwhelming the blowtank, whose level H^{BT} has an upper-bound of 90%, with material exiting the digester. The level of the Washer 1 seal tank H_{st}^{WC1} (Fig. 3.5b) also drops during the shutdown period because its contents are recycled to the blowtank, via recycle stream F_3^{WC1} (Fig. 3.5k).

The restoration trajectories for F_{S2}^{BT} (Fig. 3.5j) and F_2^{WC3} (Fig. 3.5l) reveal the general strategy of the optimizer. During the restoration period, the optimizer is pursuing a course of action in which the flowrates are initially increased by a significant amount to increase production. However, in order to meet endpoint constraints–the optimizer quickly ramps down the said flowrates and settles at the steady-state values. As a result, the inventories in the tank are then built back up. The dissolved solids compositions in the washer seal tanks, namely $x_{DS3}^{WC1}, x_{DS3}^{WC2}, x_{DS3}^{WC3}$ (Figs. 3.5c, e, g) do not appear to deviate significantly from their steady-state values, which indicates that the shutdown did not introduce signification variation to the compositions in the counter-current washing circuit.

For this partial shutdown case, the product qualities did not deviate significantly from the target. This is the result of two things: (1) the product quality optimization tier minimizes all deviations from target values; (2) the sizable capacity of the storage tank minimizes the propagation of variation in pulp consistency to downstream units.

- *Scenario (c): Shutdown in the digester*
  The overall economics achieved was $52,286, which is worse than the above two cases. This can be attributed to the fact that the digester is a key unit in the process, and the pulp it generates is the primary component for processing.

  The overall strategy is similar in principle to what is described in the previous case. The trajectories are represented by the dotted-dashed lines in Fig. 3.5. The digester is the first unit in this plant, therefore in this scenario we expect a greater disruption in the plant than the above two cases.

The chip feed to the digester, F_{in1}^{DG} (Fig. 3.5i) is turned off during the shutdown period. We assume that the combined duration of the shutdown and restartup procedures are lumped into this downtime. The strategy is to keep all the units downstream of the blowtank operational while the digester is down. The implementation of this strategy can be see by a decline in the level of H^{BT} (Fig. 3.5a), and a gradual increase during its subsequent restoration.

During the shutdown, the washers continue to operate. However, the liquor recycle to the blowtank from the washing circuits is turned off as seen in the figure for F_3^{WC1} (Fig. 3.5k). The inventories of the 1st and 3rd sealtanks build up as a result, as observed in H_{st}^{WC1} and H_{st}^{WC3} (Figs. 3.5b, f). The pulp consistencies downstream can be seen to drop, particularly in the screeners (C_M^{SR}, Fig. 3.5n) and post-O_2 washers (C_M^{PW}, Fig. 3.5o), due to the loss of pulp output from the digester.

In the restoration phase, the inventories are being balanced in such a way as to enable the system to reach its original steady-state after at the end of the restoration period.

3.5.3 Case 2: Uncertainty in downtime estimate, d_{est} (MPC mode)

Description

This case study demonstrates the use of reoptimization as a feedback mechanism for updating downtime estimates. The downtime estimate is an uncertain quantity that often cannot be accurately determined from the outset. Over the course of time, as more information becomes available, the downtime estimate can be updated and a reoptimization is performed, and a mid-course correction strategy is implemented.

We will consider two scenarios. In Scenario (1), the actual downtime d_{act} is shorter than the initial estimate d_{est}. In Scenario (2), the actual downtime d_{act} is longer than the initial estimate d_{est}. In each scenario, we investigate the effect of updating the downtime estimate at different times, that is, we assume that at a chosen reoptimization point, the operator receives an corrected downtime estimate from the maintenance personnel. The estimate is entered into the control system, and the MPC algorithm reoptimizes the remainder of the trajectories based on this new estimate. In this study, we only update the downtime estimate once per scenario,

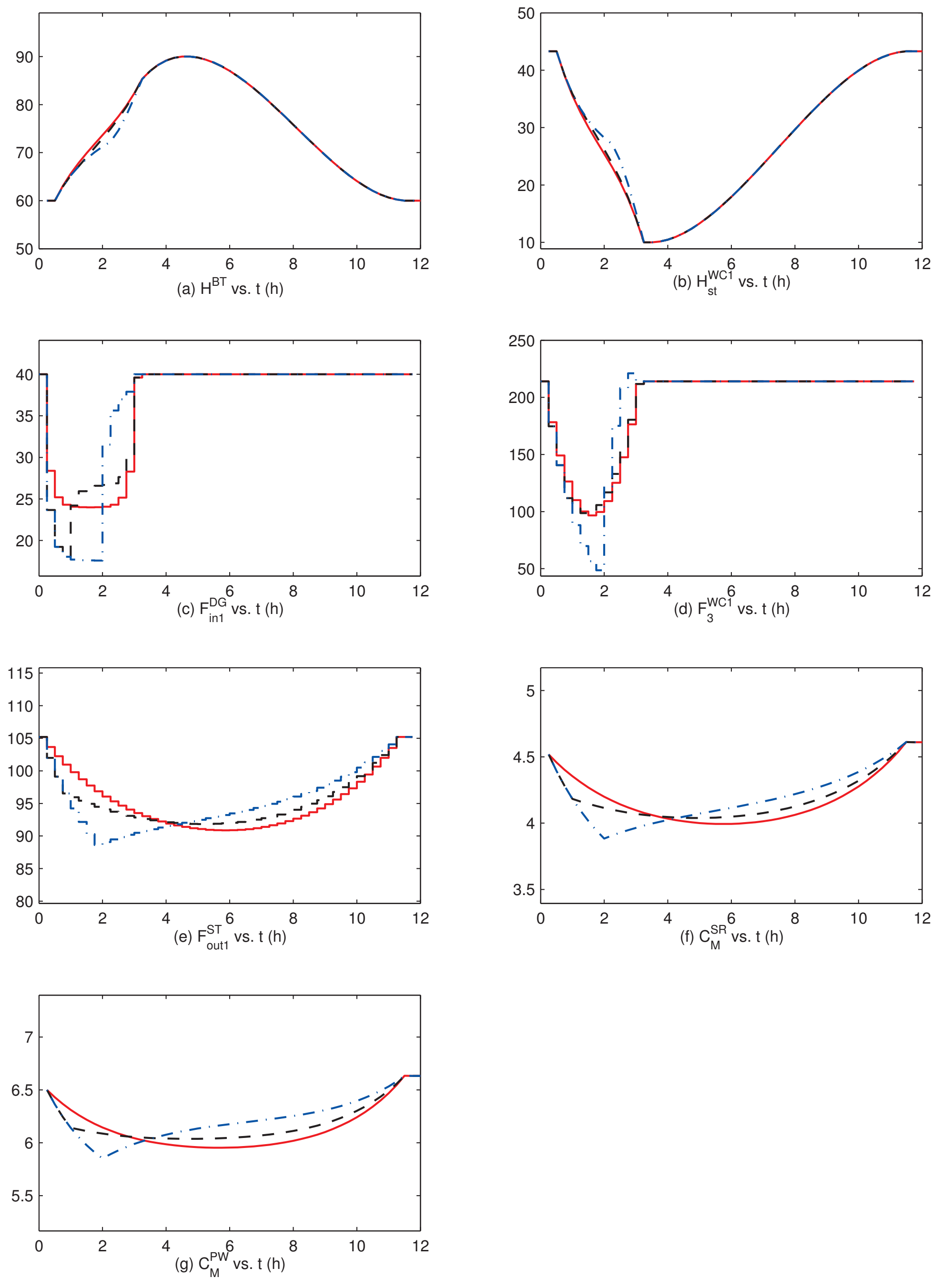


Figure 3.6: Closed-loop results. *Scenario (1), Downtime estimate reoptimization.* Hi-Q knotter shutdown (duration $d_{\text{act}} = 2.5\,\text{h} < d_{\text{est}} = 3.5\,\text{h}$). Reoptimization at $t = 0.5, 1.25, 2.25\,\text{h}$.
Legend: Solid lines = (1a) reoptimized at 0.25 h. Dashed lines = (1b) reoptimized at 1.25 h. Dotted dashed lines = (1c) reoptimized at 2.25 h.

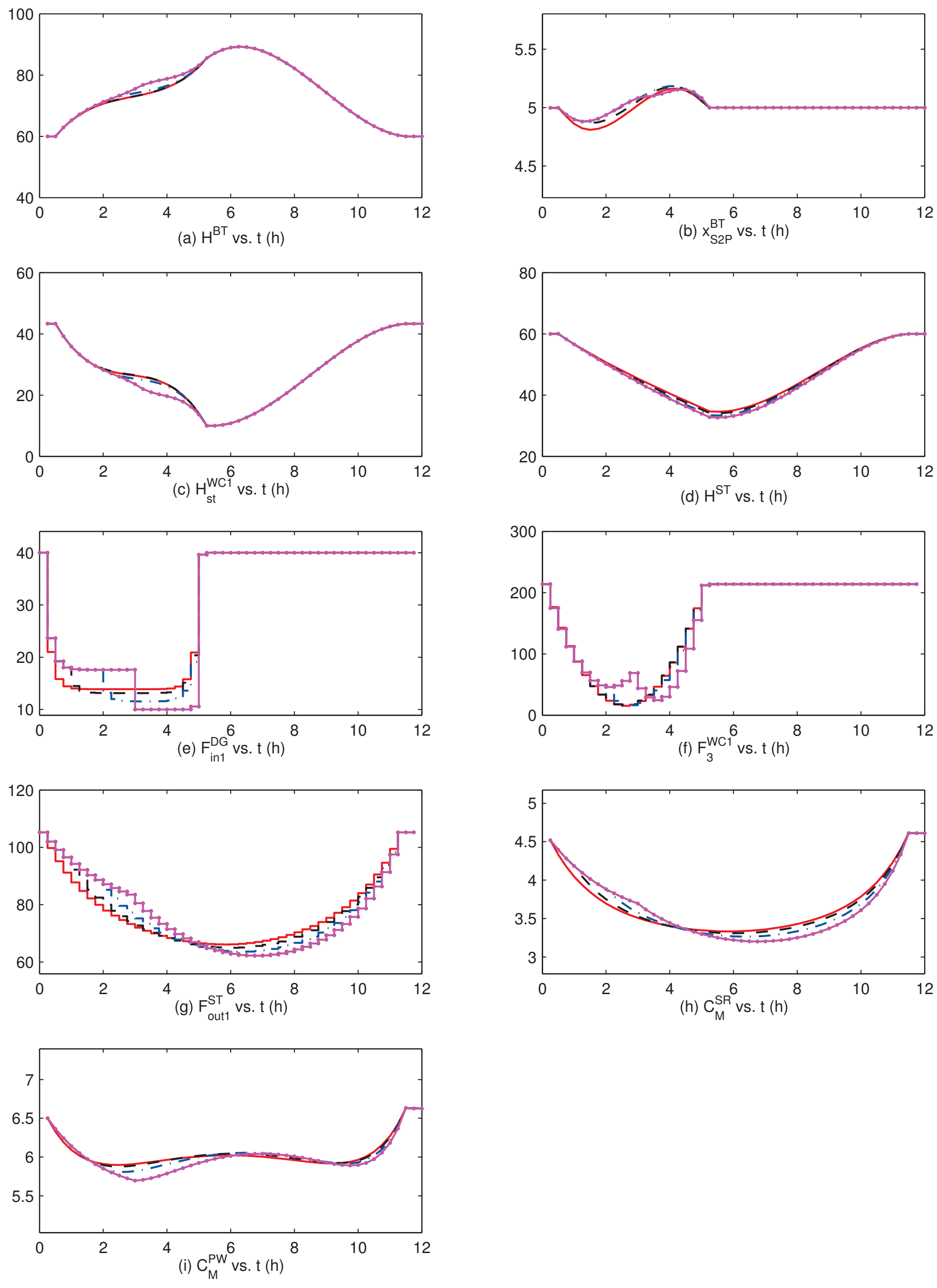


Figure 3.7: Closed-loop results. *Scenario (2), Downtime estimate reoptimization.* Hi-Q knotter shutdown (duration $d_{\text{act}} = 4.5\,\text{h} > d_{\text{est}} = 3.5\,\text{h}$). Reoptimization at $t = 0.5, 1.25, 2.25, 2.75\,\text{h}$. **Legend**: Solid lines = (2a) reoptimized at 0.25 h. dashed lines = (2b) reoptimized at 1.25 h. Dotted dashed lines = (2c) reoptimized at 2.25 h. Dotted lines = (2d) reoptimized at 2.75 h.

but in principle, it can be done as many times as required. The attributes of the two scenarios are summarized in Table 3.7. Given that there are 3 reoptimization points in Scenario (1), and 4 reoptimization points in Scenario (2), 7 independent sub-cases are carried out.

Case 2 Summary		
	Scenario (1), $d_{est} > d_{act}$, Fig. 3.6	Scenario (2), $d_{est} < d_{act}$, Fig. 3.7
Units shut down:	Hi-Q knotter	Hi-Q knotter
Simulation horizon (h):	12	12
Shutdown start time (h):	0.25	0.25
Estimated downtime, d_{est} (h):	3.5	3.5
Actual downtime, d_{act} (h):	2.5	4.5
Reoptimization points (h):	0.25, 1.25, 2.25 h	0.25, 1.25, 2.25, 2.75 h
Disturbances:	None	None

Table 3.7: Case 2 summary

Results

The economics obtained in each scenario sub-case (at each reoptimization point) are listed in Table 3.8.

Scenario sub-cases	d_{est}	d_{act}	**Reopt time (h)**	**Economics, Φ^*_{econ} ($)**
Scenario (1a), $d_{est} > d_{act}$ (ideal)	3.5	2.5	0.25	84,590
Scenario (1b), $d_{est} > d_{act}$	3.5	2.5	1.25	84,588
Scenario (1c), $d_{est} > d_{act}$	3.5	2.5	2.25	84,574
Scenario (2a), $d_{est} < d_{act}$ (ideal)	3.5	4.5	0.25	69,158
Scenario (2b), $d_{est} < d_{act}$	3.5	4.5	1.25	69,155
Scenario (2c), $d_{est} < d_{act}$	3.5	4.5	2.25	69,153
Scenario (2d), $d_{est} < d_{act}$	3.5	4.5	2.75	69,150

Table 3.8: Economics of scenarios (a) and (b). The ideal case is when the actual downtime is known at the start of the shutdown (at $t = 0.25$ h), and a reoptimization is performed at that point.

Several observations may be gleaned from the results. The ideal results correspond to the subcases in each scenario where the reoptimization time is $t = 0.25$ h. This is because the shutdown starts at $t = 0.25$ h, therefore if the correct downtime is given to the optimizer at this point, it can take the necessary corrective actions without any hindrances.

First of all, it should be obvious that a longer downtime generally corresponds to poorer economic outcomes. Secondly, one observes that the economic outcomes generally deteriorates with reoptimization points that occur later. This is an eminently intuitive result, because the outcome of a reoptimization depends on the degrees-of-freedom available at the point of reoptimization. It stands to reason that if the plant had been operated up to that point based on an incorrect downtime, the plant may have moved to a point that is disadvantaged vis-à-vis a point that was based on a correct downtime. Further, owing to the fact that our controller has a finite-shrinking horizon, a later reoptimization step finds itself with few remaining degrees-of-freedom to adjust.

Fig. 3.6 shows the trajectories obtained from the sub-cases of Scenario (1). Due to the size of the buffers, the differences in the states H^{BT} and H_{st}^{WC1} (Figs. 3.6a, b) for the three sub-cases are visible, though almost imperceptible despite the difference in reoptimization times. However in Scenarios (1b) and (1c), several trajectories depart visibly from the previous two scenarios, notably F_{in1}^{DG}, F_3^{WC1} and F_{out1}^{ST} (Figs. 3.6c, d, e). This is because the reoptimization point in Scenario (1c) occurs much later, therefore trajectories based on an incorrect (longer) downtime have already been implemented up to that point. Since the controller originally assumed that the downtime was longer than it actually is, F_{in1}^{DG}, F_3^{WC1} and F_{out1}^{ST} (Figs. 3.6c, d, e) were adjusted in such a way as to simultaneously decrease the amount of material entering the blowtank (to prevent it from hitting its upper level bound) and slowing down the rate of the amount of material being discharged from the storage tank.

In Scenario (2), depicted in Fig. 3.7, analogous adjustments are made—prior to reoptimization—based on an incorrect (shorter) downtime. Effects of this can be seen the trajectories for F_{in1}^{DG}, F_3^{WC1} and F_{out1}^{ST} (Figs. 3.7e, f, g). If we compare the trajectories for (2a) and (2d), we observe that in the latter case—in which reoptimization occurs later—the recycle F_3^{WC1} (Fig. 3.7f) to the blowtank is throttled down to avoid overwhelming the blowtank, and the storage tank discharge F_{out1}^{ST} (Figs. 3.7g) rate drops more, in order to avoid running out of material to feed downstream units. The state trajectories for H^{BT}, x_{S2P}^{BT}, H_{st}^{WC1} and H^{ST} (Figs. 3.7a – d) can all be seen to evolve slightly differently in the different sub-cases, as do the product quality variables C_M^{SR} and C_M^{PW} (Figs. 3.7h, i).

In general, parametric feedback of the downtime estimate allows the operators to be less conservative in their original estimates, which in turn leads to smaller losses in economics owing to the way the plant is operated during a shutdown.

3.5.4 Case 3: Closed-loop Study with Unmeasured Disturbances

Description

In this case study, we compare the implementation of an open-loop dynamic optimization solution (with no feedback) with that of a closed-loop solution (with feedback) in the presence of uncertainty. In the former case, we assume that an open-loop dynamic optimization tool was used to generate the desired input trajectories, and that the trajectories are used as a recipe for plant operation, and the implementation is devoid of any closed-loop correction mechanism. The purpose of this study is to compare the performance of recipe-driven operation with an actual feedback control system on a plant.

A shutdown controller (Section 3.3) with full-state measurements is used to compute and implement the closed-loop solution. An unmeasured step disturbance in the shower stream of Washer 3, F_2^{WC3} enters in plant between t=6.25 – 8.25 hrs. During this period, the actual flowrate of the shower stream that enters Washer 3 is 30% higher than modeled. No noise or plant-model mismatch (except for the disturbance) is present. Table 3.9 below summarizes this case.

Case 3 Summary (Fig. 3.8)

Units shut down:	Hi-Q knotter
Simulation horizon (h):	12
Shutdown start time (h):	0.25
Estimated downtime, d_{est} (h):	5
Actual downtime, d_{act} (h):	5
Disturbances:	Actual F_2^{WC3} stream flowrate in plant is 30% higher than in model, between t=6.25 – 8.25 hrs.

Table 3.9: Case 3 summary

Results

Fig. 3.8 shows a comparison between the open-loop operation (dashed-lines) and closed-loop operation (solid lines). The plots show the *actual* plant trajectories that result from the implementation of the two different policies. We will compare the results obtained for a system with a disturbance in the shower water flowrate.

The controller has no direct means of measuring the *actual* shower water flowrate F_2^{WC3} that actually enters the plant; it merely assumes that the F_2^{WC3} input trajectory it prescribes is actually

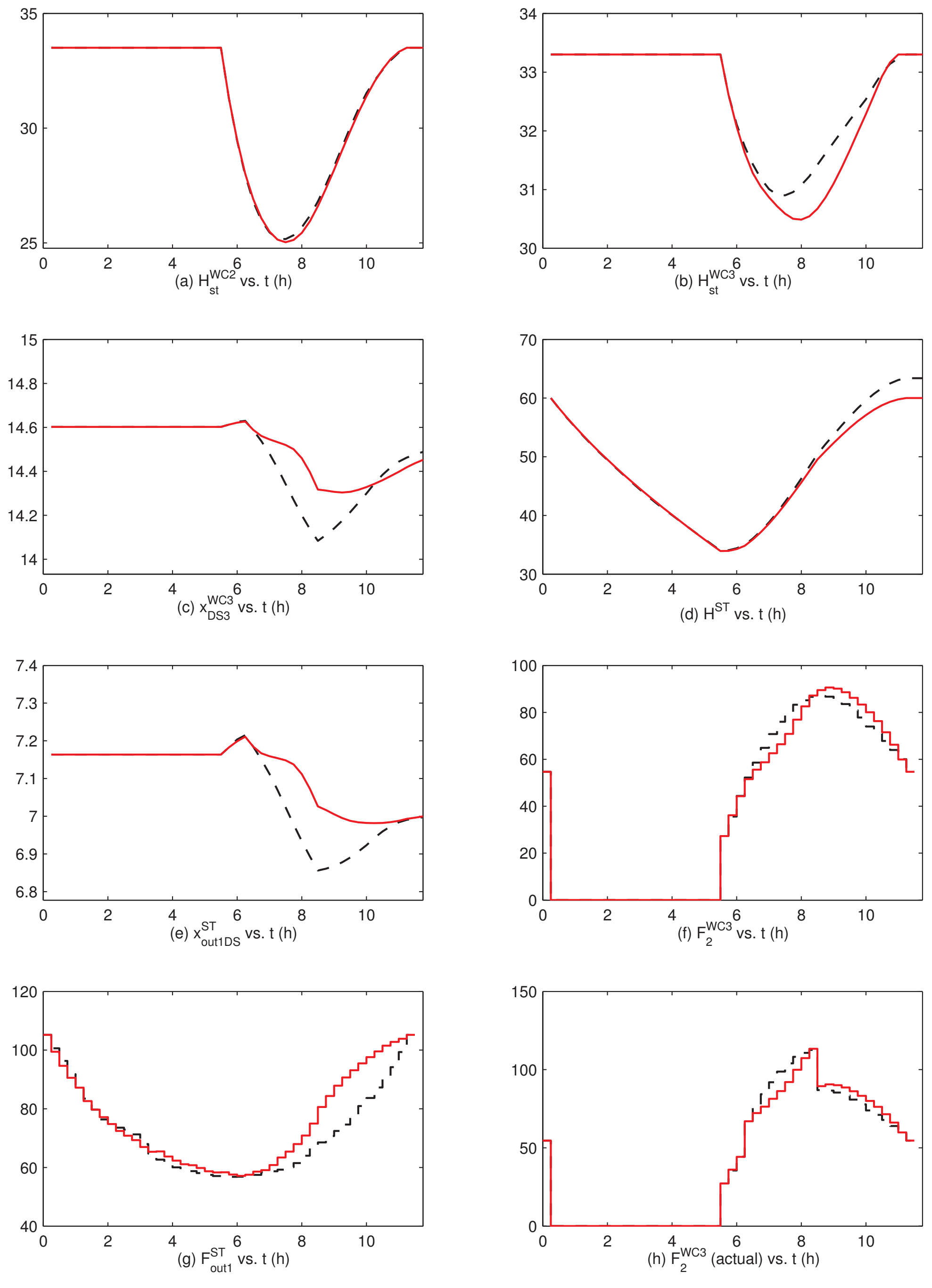


Figure 3.8: Closed-loop results with a disturbance entering the Washer 3 shower stream F_2^{WC3} between t=6.25 – 8.25 hrs.
Legend: Dashed lines = plant trajectories from open-loop solution (no feedback). Solid lines = plant trajectories from closed-loop solution (with feedback).

implemented as is in the plant. Fig. 3.8f shows the shower water F_2^{WC3} trajectory that the controller "thinks" it has implemented on the plant. For illustration purposes, we also show the trajectory for "F_2^{WC3} (actual)" (Fig. 3.8h), which depicts the *actual* shower water flowrate that enters the 3rd washer. The disturbance can be observed to occur between t=6.25 – 8.25 hrs, where actual F_2^{WC3} (Fig. 3.8h) is 30% higher than the presumed F_2^{WC3} (Fig. 3.8f). As mentioned, the controller is not cognizant of the actual input trajectory.

We observe the that the open-loop and closed-loop results are identical up to the point when the disturbance enters (at t=6.25 h). The reader may notice a slight discrepancy between the two results in F_{out1}^{ST} (Fig. 3.8g); this can be ascribed to the fact that a small positive value (1×10^{-4}) for the multi-tier relaxation factors ϵ_e and ϵ_s were used for numerical expediency. The unmeasured disturbance results in an influx of material into the storage tank, which causes its level H^{ST} (Fig. 3.8d) to rise. Through the state measurements, the shutdown controller is able to detect this, and hence computes a change of course in the storage tank outlet flowrate, F_{out1}^{ST} (Fig. 3.8g), where the excess material is discharged in order to compensate for the influx. Note also that the H^{ST} returns to nominal levels at the end of the restoration, as required.

In contrast, in the open-loop situation no feedback mechanism is present, therefore the disturbance in the shower water flowrate is not detected. The F_{out1}^{ST} (Fig. 3.8g) trajectory continues to operate as if no disturbance had occurred. As a result, the tank level H^{ST} (Fig. 3.8d) does not return to its intended restoration target value at the end of the restoration, which is an undesirable result as it deprives the tank from anticipating further shutdowns. Other transient discrepancies between the open-loop and the closed-loop are seen in H_{st}^{WC3}, x_{DS3}^{WC3}, and x_{out1DS}^{ST} (Figs. 3.8b, c, e).

The overall economics obtained in closed-loop ($\Phi_{\text{econ}}^* = \$66,915$) is higher than that obtained by implementing the open-loop solution on the plant ($\Phi_{\text{econ}}^* = \$63,368$). This demonstrates the advantage of using a closed-loop framework for implementing dynamic optimization solutions under uncertainty. In Chapter 5, this framework is enlarged and extended to handle cases where some state measurements are missing and where there exists significant plant-model mismatch.

3.6 Conclusion

The primary focus of this work has been on the use of dynamic optimization for coordinating the buffer capacities in a plant in order to mitigate the effects of a unit shutdown. The model employed was a large-scale differential algebraic equation system which was solved using the simultaneous method. Uncertainty in the downtime estimate parameter, a parameter which has a significant effect on the optimality of the problem, was considered. Methods to handle this uncertainty were proposed and compared. An MPC-based control framework for implementing the above optimal control policies was tendered for the detection and correction of disturbances and/or plant-model mismatch. Several case studies illustrating its application were presented.

Chapter 4

Discontinuous Dynamic Optimization Formulations for Handling Partial Shutdowns

In this chapter, we will investigate specific discontinuous phenomena that arise in shutdown modeling, such as shutdown thresholds, induced shutdowns, discontinuous costs, and minimum shutdown durations. A formulation for minimizing the restoration time instead of maximizing economics is proposed. The optimization formulations are embedded within a feedback control framework that is based on model predictive control (MPC). The open-loop problem is cast as a Mixed-Integer Dynamic Optimization (MIDO).

4.1 Introduction

The starting point of the formulations presented in this chapter is identical to that found in Section 3.2, except that binary variables are present in the current formulation. We will re-state the system found in Section 3.2 in the form of a Mixed-Integer Dynamic Optimization (MIDO) problem here:

$$\max_{u(t)} \Phi_{\text{econ}}(x(t), z(t), u(t)) \tag{4.1}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), z(t), u(t), \delta(t), \theta), \quad x(0) = x_0 \tag{4.2}$$

$$h(x(t), z(t), u(t), \delta(t), \theta) = 0 \tag{4.3}$$

$$g(x(t), z(t), u(t), \delta(t), \theta) \leq 0 \tag{4.4}$$

$$\text{for } t \in [0, t_f]$$

where Φ_{econ} = economic objective function, $x(t) \in \mathbb{R}^{n_x}$ = differential state vector, x_0 = initial state parameter vector, $z(t) \in \mathbb{R}^{n_z}$ = algebraic state vector, $u(t) \in \mathbb{R}^{n_u}$ = control input vector, $\delta(t) \in \{0,1\}^{n_\delta}$ = binary variable vector, $\theta \in \mathbb{R}^{n_\theta}$ = parameter vector, t_f = final time in prediction horizon. f and h are functions that define a semi-explicit differential-algebraic system, and g is a function that defines the inequality constraints.

We take a similar approach to that of Section 3.2 in the discretization of the above problem. In the rest of this chapter, we will focus our discussion on the discretized system where variables $x_k = x(t_k)$, $z_k = z(t_k)$, $u_k = u(t_k)$, $\delta_k = \delta(t_k)$ for $k = 0 \ldots N$ where $N = \lfloor t_f / \Delta t \rfloor$ and Δt is the discretized interval length. Time t is partitioned and mapped into N intervals, and $t_{k-1} \leq t \leq t_k$ where $k = 1, \ldots, N$. The resulting optimization problem structure is a Mixed-Integer Program (MIP), either of the linear or nonlinear variety.

4.1.1 Shutdown Formulation

The α_{shut} formulation in Section 3.2.1 is similarly adopted here as well. However, the difference here is that the $\alpha^p_{\text{shut},k}$ variables are now binary *decision variables* as opposed merely to being fixed parameters, as they were in Section 3.2.1. We will restate the constraints for the reader's convenience: (for time $k = 0, \ldots, N-1$ and $i \in \mathscr{I}_p$, $l \in \mathscr{L}_p$)

$$F_{i,k} \geq (1 - \alpha^p_{\text{shut},k}) F^p_{\text{shut}} \tag{4.5}$$

$$F_{i,k} \leq (1 - \alpha^p_{\text{shut},k}) F^U_i \tag{4.6}$$

$$y_{l,k} \geq (1 - \alpha^p_{\text{shut},k}) y^L_l + \alpha^p_{\text{shut},k} y^M_l \tag{4.7}$$

$$y_{l,k} \leq (1 - \alpha^p_{\text{shut},k}) y^U_l + \alpha^p_{\text{shut},k} y^M_l \tag{4.8}$$

where p = identifier index of a process unit; $\mathscr{P}$ = set of all process units; $\mathscr{I}_p$ = index set of all *inlet* flow variables associated with unit p; $\mathscr{L}_p$ = index set of all variables associated with unit p that *float* when the flowrate is 0; $\alpha^p_{\text{shut},k} \in \{0,1\}$ = binary indicator variable representing status of process unit p at time k (1 = shutdown, 0 = normal operation); $y_{l,k}$ = floating variable at time k; y^L_l, y^U_l = lower and upper bound parameters corresponding to y_l; y^M_l = nominal value parameter to which $y_{l,k}$ is pinned to when $\alpha^p_{\text{shut},k} = 1$; $F_{i,k}$ = inlet flowrate variable at time k; F_i^U = upper bound of $F_{i,k}$ for all k; F_{shut}^p = a positive nonzero parameter representing the minimum flowrate threshold before a unit is deemed to have shut down.

In order to manually simulate a user-specified *known* shutdown in units $p \in \mathscr{P}_{\text{shut}}$ ($\mathscr{P}_{\text{shut}}$ is the set of known shut-down units), the following constraint is explicitly imposed:

$$\alpha_{\text{shut},k}^p = 1, \quad p \in \mathscr{P}_{\text{shut}},\ k = k_{\text{start}}, \ldots, k_{\text{end}} \tag{4.9}$$

where $k_{\text{start}} = \lfloor t_{\text{start}}/\Delta t \rfloor$ = shutdown start time; $k_{\text{end}} = \lceil t_{\text{end}}/\Delta t \rceil$ = shutdown end time. These $\alpha_{\text{shut},k}^p$ binary variables that are fixed above are essentially transformed into constant values; however, unlike the approach taken in Section 3.2.1, the $\alpha_{\text{shut},k}^p$ of the remaining units are not forced to 0. This means the optimizer is free to decide the course of action (whether to shut down other units or not), given that the units in $p \in \mathscr{P}_{\text{shut}}$ have already shut down.

One of the consequences of constraints 4.7 – 4.8 is that when $F_{j,k} < F_{\text{shut}}^p$ (that is, when the flowrate falls below a threshold), the unit is deemed to have shut down, and $\alpha_{\text{shut},k}^p$ = 1 is automatically (as opposed to manually) triggered by the optimizer to remain feasible. This phenomenon we refer to as an *induced shutdown*, and it is described in the Section 4.2 below.

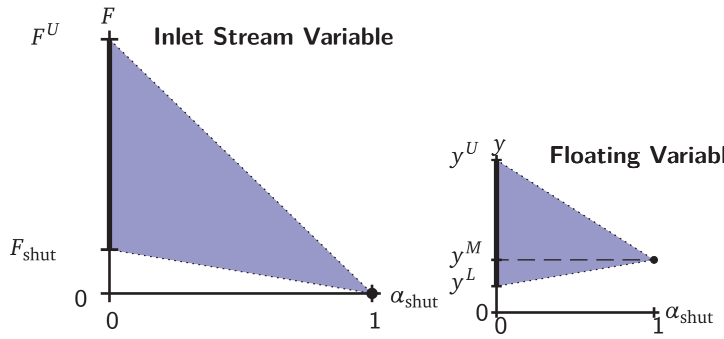


Figure 4.1: Feasible region of the α_{shut} formulation, both discrete and relaxed.

Remark 4.1.1. We note that $F_{i,k}$ belongs to a class of semi-continuous variables (variables whose range belongs to the set $\{0\} \cup [L, U]$, where L, U are positive lower and upper bounds respectively). The feasible region obtained by the integer-relaxation of constraints 4.5–4.8 corresponds to their convex hull. This is a favorable property that gives rise to tight bounds for the branch-and-bound solution procedure (which is used for solving integer programs), potentially leading to faster solutions. Fig. 4.1 shows two diagrams depicting the feasible regions of particular inlet stream and floating variables. When α_{shut} is 0 (non-shutdown mode), the individual feasible regions are $F \in [F_{\text{shut}}, F^U]$ and $y \in [y^L, y^U]$. In shutdown mode, α_{shut} takes a value of 1, and the feasible regions reduce to single points, $F = 0$ and $y = y^M$. In

MIP algorithms, relaxed problems are solved in which the binary variables are converted into continuous variables with [0,1] bounds. One can easily see that when $\alpha_{\text{shut}} \in [0,1]$, the regions induced (shaded in the figures) form the convex hulls of the two disparate feasible regions corresponding to $\alpha_{\text{shut}} = 0$ and $\alpha_{\text{shut}} = 1$ in each individual case.

4.2 Induced Shutdowns

The optimizer will occasionally produce an operating policy that prescribes the shutting off of material flows to upstream/downstream units *across the boundary of a buffer capacity* (effectively shutting them down) to accommodate the original shutdown. Henceforth we shall denote this type of shutdown an “induced shutdown”. Note that induced shutdowns are caused by, but are distinct from the original shutdown.

Induced shutdowns usually occur in response to long shutdown durations. A single buffer tank has the capacity to buffer shutdowns for a limited time before it overflows or empties. Therefore, when a long shutdown occurs, it is sometimes necessary to throttle down the production upstream (or downstream, in the case of a recycle) in order to keep from violating the level constraints in the buffer tanks.

However, instead of gradually reducing material flows over time, the optimizer may prescribe flowrate trajectories that force some process units—apart from the already shut down units—to operate below their minimum threshold F^p_{shut}, that is, the flowrates F^p_{shut}, $\forall p \in \mathscr{P} \setminus \mathscr{P}_{\text{shut}}$ are run below F^p_{shut}. This effectively causes an induced shutdown on those units.

Induced shutdowns are undesirable because each shutdown incurs both a fixed cost (in terms of the manpower, time and resources required to start a unit up again) and variable cost (in the form of lost production over time). In order to force the optimizer to avoid this scenario, it is necessary to properly characterize the cost of induced shutdowns, and to penalize it the objective function.

4.2.1 Economic Penalty of a Shutdown

Because our optimization formulation uses an economic objective function, it is fitting to have a shutdown penalty term that is economic in nature, instead of using an arbitrarily weighted penalty function. There are two components to the cost of a shutdown: a fixed cost and a variable cost. Fixed costs may include costs associated with equipment replacement/repairs, inspections, testing, and other fees that are incurred per shutdown. Variable costs comprise expenses such as hourly wages for third-party repair personnel or reassigned/diverted maintenance personnel, lost production, etc.

In order to assign a cost to shutdowns, we need to count the number of shutdowns and calculate their durations:

$$N^p_{\text{shut}} = \sum_{k=0}^{N-1} D^p_k \tag{4.10}$$

$$W^p_{\text{tot}} = \Delta t \sum_{k=0}^{N-1} \alpha^p_{\text{shut},k} \tag{4.11}$$

where N^p_{shut} = the number of shutdowns occurring in unit p over the horizon; D^p_k = entry point marker of each shutdown (1 when true, 0 otherwise) – obtained using the formulation below; W^p_{tot} = total downtime experienced by unit p.

Due to the worst-case exponential solution complexity associated with binary variables, we endeavor to be as parsimonious as possible with their use in our proposed discrete formulations. We employ the following constraint formulation from Kelly and Zyngier[63] to deduce useful time-markers (used in our formulations) from $\alpha^p_{\text{shut},k}$ without introducing any new binary variables: (for time $k = 1, \ldots, N-1$)

$$\alpha^p_{\text{shut},k} - \alpha^p_{\text{shut},k-1} = D^p_k - U^p_k \tag{4.12}$$

$$\alpha^p_{\text{shut},k} + \alpha^p_{\text{shut},k-1} = D^p_k + U^p_k + 2Q^p_k \tag{4.13}$$

$$D^p_k + U^p_k + Q^p_k \leq 1 \tag{4.14}$$

where $D^p_k \in [0,1]$; $U^p_k \in [0,1]$ = exit point marker of each shutdown (1 when true, 0 otherwise); and, $Q^p_k \in [0,1]$ = auxiliary variable. See Fig. 4.2 for a graphical explanation. These time marker variables are used in several of the formulations that follow.

A useful property of the formulation defined by constraints 4.12 – 4.14 is that its feasible region guarantees the integrality of the continuous variables D^p_k, U^p_k, and Q^p_k. Therefore, while the overall shutdown cost characterization formulation is technically discontinuous, we are able to avoid introducing new integer variables by effectively piggybacking on the existing ones introduced earlier in the Section 4.1.1.

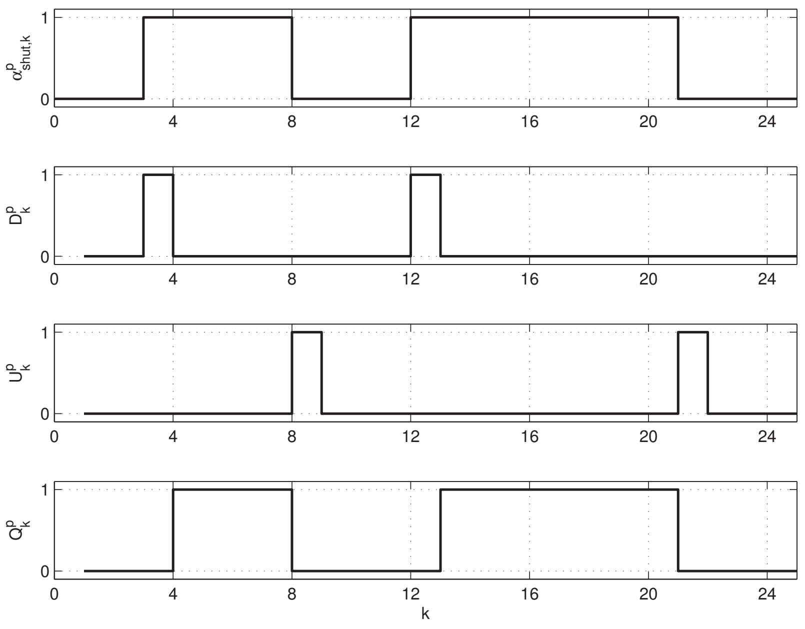


Figure 4.2: Example of shutdown time-points for a unit p. Binary markers: $\alpha^p_{\text{shut},k}$ = system shutdown state, D^p_k = shutdown entry, and U^p_k = shutdown exit.

The total cost of induced shutdowns can then be accounted for in the objective function:

$$\max \Phi_{\text{econ}} = \Phi_{\text{prod}} - \sum_{p\in\mathscr{P}} C^p_{\text{fix}} N^p_{\text{shut}} - \sum_{p\in\mathscr{P}} C^p_{\text{var}} W^p_{\text{tot}} \tag{4.15}$$

and the production economics term Φ_{prod} is defined:

$$\Phi_{\text{prod}} = \Delta t \sum_{m\in\mathscr{M}} \left[C_m \left(\sum_{k=0}^{N-1} F_{m,k} \right) \right] \tag{4.16}$$

where Δt = sampling time, C^p_{fix} = estimated fixed costs expressed as dollars per shutdown; C^p_{var} = estimated variable costs expressed as dollars per time, m = materials produced or consumed, $\mathscr{M}$ = set of materials consumed/produced, C_m = price of material m (positive if produced, negative if consumed), and $F_{m,k}$ = flowrate of material m at time k.

This has the effect of penalizing both the number of shutdowns and their durations on the basis of their economics. The optimizer will then attempt to find a control policy that avoids induced shutdowns unless they are absolutely necessary for maintaining feasibility. In some cases, despite the costs incurred in an induced shutdown, an economic case can be made for proceeding with one. For instance, when the raw material or energy spot prices are such that it is disadvantageous to run the plant, the optimal course of action may be to shut down the plant until it becomes economically viable to operate again.

4.3 Minimum Shutdown Duration

The partial shutdown optimization problem can be thought of as a hybrid optimization problem with mode switches (with modes corresponding to the on-line or offline status the units). In the case of induced shutdowns, it is possible for the optimizer to trigger shutdowns of unrealizable durations, for instance, 1 sample time. We can prevent this behavior by enforcing a minimum downtime in the optimization. When a unit shuts down, there is usually a minimum contiguous duration for which it has to physically stay offline. The following formulation ensures that shutdown events for process unit p have a minimum contiguous duration of T^p discrete periods:

$$S_k^p = \sum_{i=k}^{k+T^p-1} \alpha_{\text{shut},i}^p, \quad k = 1, \ldots, N - T^p \tag{4.17}$$

$$S_k^p \geq T^p D_k^p, \quad k = 1, \ldots, N - T^p \tag{4.18}$$

$$D_k^p = 0, \quad k = N - T^p + 1, \ldots, N - 1 \tag{4.19}$$

where S_k^p = a T^p-step ahead summation of $\alpha_{\text{shut},k}^p$. We wish to emphasize that no new binary variables have been introduced here.

We note that this is an aggregation-type approach which also serves to reduce the integer problem space. A higher value of T^p reduces the number of branches that need to be considered during the integer solution process, which often translates into faster solve times.

4.4 Alternative: Optimal Restoration Time

Thus far, we have assumed that the objective of interest is monetary in nature, that is, we desire to operate the plant in a such a way as to maximize profit (or minimize economic losses, as the case may be). However, in some circumstances, a dynamic performance objective may be more appropriate. For instance, if a large disturbance is imminent, an operator may wish to restore the plant as rapidly as possible (at the expense of economics if necessary) in order to anticipate the disturbances that may occur.

In this section, we propose an alternative optimization formulation whose first goal is to calculate trajectories to restore the plant to its original operating state as quickly as possible. Also, in our experience, there are typically many degrees-of-freedom in a large scale optimization problem. In order to exploit this fact, we propose a multi-tiered approach to computing the optimal restoration strategy. In this approach, the economics of the restoration are not disregarded entirely, but merely relegated to the position of a secondary objective in relation to the restoration time. Because we are working with a discrete representation of time, binary variables are necessary for formulating an optimization problem involving time as a decision variable.

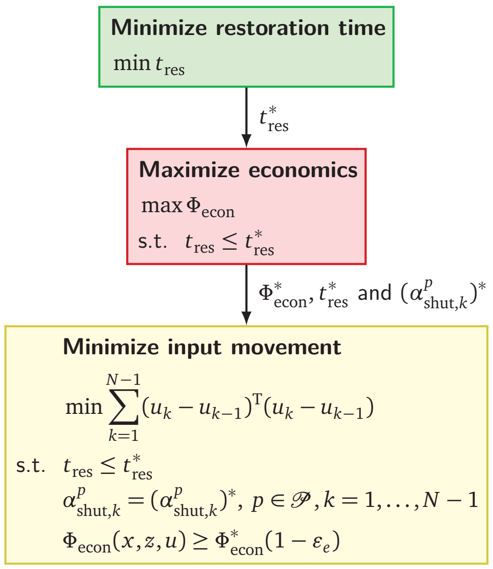

Minimize restoration time
$\min t_{\text{res}}$
t^*_{res}
Maximize economics
$\max \Phi_{\text{econ}}$
s.t. $t_{\text{res}} \leq t^*_{\text{res}}$
$\Phi^*_{\text{econ}}, t^*_{\text{res}}$ and $(\alpha^p_{\text{shut},k})^*$
Minimize input movement
$\min \sum_{k=1}^{N-1} (u_k - u_{k-1})^{\mathrm{T}}(u_k - u_{k-1})$
s.t. $t_{\text{res}} \leq t^*_{\text{res}}$
$\alpha^p_{\text{shut},k} = (\alpha^p_{\text{shut},k})^*,\ p \in \mathscr{P}, k = 1, \ldots, N-1$
$\Phi_{\text{econ}}(x,z,u) \geq \Phi^*_{\text{econ}}(1-\varepsilon_e)$

Figure 4.3: Restoration time minimization using a multi-tiered approach.

Multi-Tiered Optimization of Restoration Time

Fig. 4.3 summarizes the steps required to perform a restoration time optimization. In Tier 1, the restoration time endpoint t_{res} is minimized:

$$\min t_{\text{res}} \tag{4.20}$$

Eqn. 4.31 provides the mathematical formulation for computing t_{res}.

In Tier 2, the restoration time endpoint is locked to the optimal value t^*_{res} obtained in the first tier. An economic objective function Φ_{econ} is then maximized.

$$\max \Phi_{\text{econ}}(x,z,u) \tag{4.21}$$

$$s.t.\ t_{\text{res}} \leq t^*_{\text{res}} \tag{4.22}$$

where the vectors $x = \left[x_0^{\mathrm{T}}, x_1^{\mathrm{T}}, \ldots, x_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = differential states, $z = \left[z_0^{\mathrm{T}}, z_1^{\mathrm{T}}, \ldots, z_N^{\mathrm{T}}\right]^{\mathrm{T}}$ = algebraic states, and $u = \left[u_0^{\mathrm{T}}, u_1^{\mathrm{T}}, \ldots, u_{N-1}^{\mathrm{T}}\right]^{\mathrm{T}}$ = control inputs. Note that no binary variables are fixed in this tier, therefore this is still a mixed-integer programming problem. This is necessary to allow the optimizer enough degrees-of-freedom to find an appropriate solution trajectories that will maximize economics.

Finally, in Tier 3, we minimize the control effort by penalizing movement in the trajectories. The terms t_{res} and Φ_{econ} are locked to the optimal values obtained in previous tiers. Also, all binary variables are fixed in this tier (to the values obtained in the previous tier); the problem reduces to a continuous optimization problem.

$$\min \sum_{k=1}^{N-1} (u_k - u_{k-1})^{\mathrm{T}} (u_k - u_{k-1}) \tag{4.23}$$

$$s.t. \quad t_{\mathrm{res}} \leq t_{\mathrm{res}}^* \tag{4.24}$$

$$\alpha_{\mathrm{shut},k}^p = (\alpha_{\mathrm{shut},k}^p)^*, \; p \in \mathscr{P}, k = 1, \ldots, N-1 \tag{4.25}$$

$$\Phi_{\mathrm{econ}}(x, z, u) \geq \Phi_{\mathrm{econ}}^*(1 - \varepsilon_e) \tag{4.26}$$

where $(\alpha_{\mathrm{shut},k}^p)^*$ is the optimal sequence obtained in the previous tier, and $\varepsilon_e \geq 0$ is an economic trade-off parameter representing the maximum fraction of achievable economics (i.e. Φ_{econ}^*) to trade-off.

Remark 4.4.1. To decrease the time necessary for restoration, the controller typically needs to prescribe more drastic/aggressive control actions. While eqn. 4.20 minimizes the restoration time, no cost is assigned to induced shutdowns, therefore the optimization problem can sometimes produce solutions that cause induced shutdowns in other units. This partly defeats the purpose of obtaining time-optimal performance (due to the re-startups that have to be performed in the other units). In practice, induced shutdowns are indirectly penalized in the objective function in eqn. 4.20 as follows:

$$\min t_{\mathrm{res}} + \sum_{p \in \mathscr{P}} W_{\mathrm{tot}}^p \tag{4.27}$$

where W_{tot}^p = total downtime experienced by unit p. The 2nd term in the objective function (representing total system downtime, defined in Eqn. 4.11) disincentivizes solution trajectories that produce undesired induced shutdowns. We elected to use a time-based penalty in order ensure that the entire objective function is denominated in time units—this is done for consistency reasons.

Remark 4.4.2. Economics and speed of restoration may sometimes constitute complementary goals. If the cost of a slow restoration can be accurately quantified in economic terms, the minimization of restoration time may be subsumed by the economic optimization tier (Tier 2), and Tier 1 may be removed. In our case, the two are competing goals and the cost associated with a longer restoration time is difficult to quantify, therefore it is natural to solve them in separate tiers.

MIP Formulation for Restoration Time

In order to represent the system-wide restoration endpoint time in the optimization problem, we introduce the following constraints:

$$\Delta\alpha_{\text{res},k} = \alpha_{\text{res},k} - \alpha_{\text{res},k-1}, \quad k = 1, \ldots, N-1 \tag{4.28}$$

$$\sum_{k=1}^{N-1} \Delta\alpha_{\text{res},k} = 1 \tag{4.29}$$

$$\alpha_{\text{res},0} = 0 \tag{4.30}$$

where $\alpha_{\text{res},k} \in [0,1]$ = restoration end-point marker and $\Delta\alpha_{\text{res},k} \in \{0,1\}$. Note that by declaring $\Delta\alpha_{\text{res},k}$ binary rather than $\alpha_{\text{res},k}$, we require 1 fewer binary variable. Also, we wish to point out that $\Delta\alpha_{\text{res},k}$ in constraint 4.29 can be declared as a Special Ordered Set Type 1, a structure with properties exploitable in MIP codes for increased solution efficiency, provided a natural ordering of weights are known [15]. The restoration endpoint can thus be written as follows:

$$t_{\text{res}} = \sum_{k=1}^{N-1} (\Delta\alpha_{\text{res},k})(k\Delta t) \tag{4.31}$$

Restoration constraints, which force the independent subset of state and input variables $(x_k^{i_x}, u_k^{i_u})$ to return to their original steady-state values $(x_{ss}^{i_x}, u_{ss}^{i_u})$, are given below: (for $k = 0, \ldots, N-1$)

$$x_k^{i_x} \geq \alpha_{\text{res},k}(x_{ss}^{i_x} - \epsilon) + (1 - \alpha_{\text{res},k})x_L^{i_x}, \qquad i_x \in \mathscr{I}_{\text{ss}}^x \tag{4.32}$$

$$x_k^{i_x} \leq \alpha_{\text{res},k}(x_{ss}^{i_x} + \epsilon) + (1 - \alpha_{\text{res},k})x_U^{i_x}, \qquad i_x \in \mathscr{I}_{\text{ss}}^x \tag{4.33}$$

$$u_k^{i_u} \geq \alpha_{\text{res},k}(u_{ss}^{i_u} - \epsilon) + (1 - \alpha_{\text{res},k})u_L^{i_u}, \qquad i_u \in \mathscr{I}_{\text{ss}}^u \tag{4.34}$$

$$u_k^{i_u} \leq \alpha_{\text{res},k}(u_{ss}^{i_u} + \epsilon) + (1 - \alpha_{\text{res},k})u_U^{i_u}, \qquad i_u \in \mathscr{I}_{\text{ss}}^u \tag{4.35}$$

The index sets are defined as $\mathscr{I}_{\text{ss}}^x = \{i | x_i \in \mathscr{X}_{\text{ss}}\}$ and $\mathscr{I}_{\text{ss}}^u = \{i | u_i \in \mathscr{U}_{\text{ss}}\}$, and $\mathscr{X}_{\text{ss}}, \mathscr{U}_{\text{ss}}$ are the sets of independent state and inputs variables necessary to specify a steady-state. $\epsilon > 0$ is a deviation tolerance percentage parameter, specified as a fraction. The parameter vectors $x_{\text{ss}}^{i_x}$ and $u_{\text{ss}}^{i_u}$ are pre-shutdown values. The above constraints are equivalent to:

$$\begin{cases} x_k^{i_x} \in [x_{ss}^{i_x} - \epsilon, x_{ss}^{i_x} + \epsilon], i_x \in \mathscr{I}_{\text{ss}}^x, & \text{if } t_k \geq t_{\text{res}} \\ x_k^{i_x} \in [x_L^{i_x}, x_U^{i_x}], i_x \in \mathscr{I}_{\text{ss}}^x, & \text{if } t_k < t_{\text{res}} \end{cases} \tag{4.36}$$

$$\begin{cases} u_k^{i_u} \in [u_{ss}^{i_u} - \epsilon, u_{ss}^{i_u} + \epsilon], i_u \in \mathscr{I}_{\text{ss}}^u, & \text{if } t_k \geq t_{\text{res}} \\ u_k^{i_u} \in [u_L^{i_u}, u_U^{i_u}], i_u \in \mathscr{I}_{\text{ss}}^u, & \text{if } t_k < t_{\text{res}} \end{cases} \tag{4.37}$$

where $x_L^{i_x}, x_U^{i_x}$ = upper and lower bounds of the independent states x^{i_x}; ϵ = numerical relaxation factor.

In order to further tighten our integer formulation in Eqns. 4.28–4.30, we require a good lower bound on the variable t_{res}. We can achieve this by introducing an MIP cut, based on the characteristics of our problem. Temporally, the restoration endpoint cannot occur before the endpoint of the last shutdown occurring in *any* process unit in the plant. Also, we may want to declare a minimum duration of T_R periods (after the endpoint of the last shutdown) for the restoration. These requirements can be enforced via the following additional set of constraints:

$$t_{\text{res}} \geq U_k^p \cdot (k + T_R) \cdot \Delta t, \qquad \forall p \in \mathscr{P}, k = 0, \ldots, N-1 \tag{4.38}$$

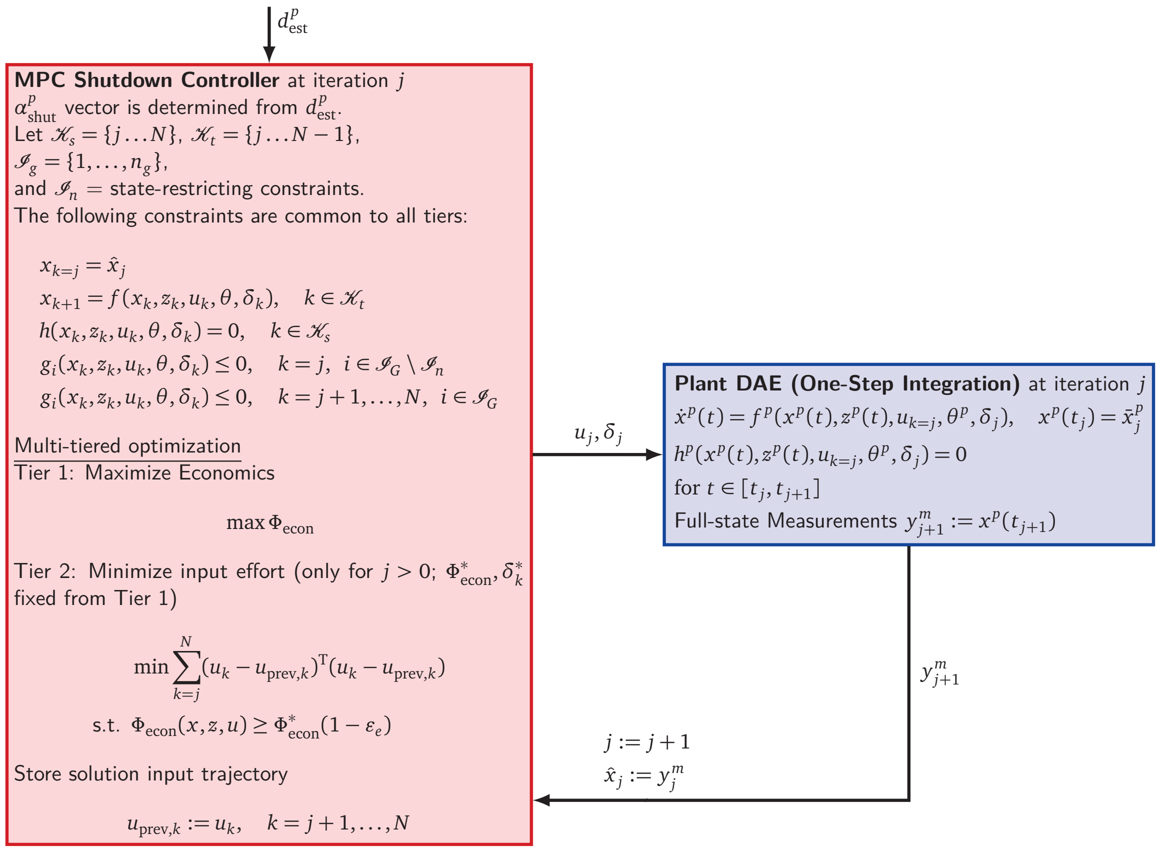
$$\alpha_{\text{res},k} = 0, \qquad k = N - 1 - T_R, \ldots, N-1 \tag{4.39}$$

Typically, $T_R \geq 1$ because it is almost always impossible for the restoration process to reach completion immediately after the last shutdown. Constraint 4.38 forces t_{res} to be greater than all shutdown endpoints (U_k^p), plus a minimum duration. Constraint 4.38 further shrinks the integer search space by specifying that no restorations can occur in the period where there are less than T_R periods left before the endpoint is reached (since T_R signifies a minimum restoration duration).

4.5 Model-based Shutdown Controller

The MPC algorithm here is based on the one described in Section 3.3, with binary variables δ_k being the major extension. This results in a Mixed-Integer Dynamic Optimization problem being embedded in the MPC superstructure. Fig. 4.4 depicts a visual notion of the algorithm.

The multi-tiered approach described here comprises an economics maximization tier, a product quality deviation minimization tier, and an input effort minimization tier. In order to avoid solving multiple mixed-integer problems while applying the multi-tiered method, the binary values obtained from the solution of Tier 1, δ_k^* is used to fix the δ_k vector in subsequent tiers. For simplicity, we will eschew the product quality tier (described in Section 3.2.5) here.



MPC Shutdown Controller at iteration j
α^p_{shut} vector is determined from d^p_{est}.
Let $\mathscr{K}_s = \{j \ldots N\}$, $\mathscr{K}_t = \{j \ldots N-1\}$,
$\mathscr{I}_g = \{1, \ldots, n_g\}$,
and $\mathscr{I}_n$ = state-restricting constraints.
The following constraints are common to all tiers:

$$
\begin{aligned}
& x_{k=j} = \hat{x}_j \\
& x_{k+1} = f(x_k, z_k, u_k, \theta, \delta_k), \quad k \in \mathscr{K}_t \\
& h(x_k, z_k, u_k, \theta, \delta_k) = 0, \quad k \in \mathscr{K}_s \\
& g_i(x_k, z_k, u_k, \theta, \delta_k) \leq 0, \quad k = j, \; i \in \mathscr{I}_G \setminus \mathscr{I}_n \\
& g_i(x_k, z_k, u_k, \theta, \delta_k) \leq 0, \quad k = j+1, \ldots, N, \; i \in \mathscr{I}_G
\end{aligned}
$$

Multi-tiered optimization
Tier 1: Maximize Economics

$$\max \Phi_{\text{econ}}$$

Tier 2: Minimize input effort (only for $j > 0$; $\Phi^*_{\text{econ}}, \delta^*_k$ fixed from Tier 1)

$$\min \sum_{k=j}^{N} (u_k - u_{\text{prev},k})^{\mathrm{T}} (u_k - u_{\text{prev},k})$$

$$\text{s.t.} \;\; \Phi_{\text{econ}}(x, z, u) \geq \Phi^*_{\text{econ}}(1 - \varepsilon_e)$$

Store solution input trajectory

$$u_{\text{prev},k} := u_k, \quad k = j+1, \ldots, N$$

Plant DAE (One-Step Integration) at iteration j
$\dot{x}^p(t) = f^p(x^p(t), z^p(t), u_{k=j}, \theta^p, \delta_j), \quad x^p(t_j) = \bar{x}^p_j$
$h^p(x^p(t), z^p(t), u_{k=j}, \theta^p, \delta_j) = 0$
for $t \in [t_j, t_{j+1}]$
Full-state Measurements $y^m_{j+1} := x^p(t_{j+1})$

Figure 4.4: MPC simulation workflow.

4.5.1 Multi-tiered optimization in Closed-Loop

In a closed-loop scenario, we use the multi-tiered approach (as described in Eqns. 3.26, 3.29 and 3.30, omitting the product quality tier) in the *first* iteration of the MPC algorithm to select a set of smooth open-loop trajectories to initialize the MPC open-loop dynamic optimization problem.

It is not used in subsequent iterations for when the multi-tiered method is applied successively, as one may obtain qualitatively different future trajectories from one iteration to the next. Input trajectories whose shapes are very different from previously calculated ones (which is a possibility when the problem has non-unique solutions) are not appealing to operators, and may be difficult to interpret intuitively. Therefore, we propose a modification to the two-tiered method when doing closed-loop control whereby the objective function in the second tier (Eqn. 3.29) is replaced with the following: (see Fig. 4.4)

$$\min \sum_{k=j}^{N} (u_k - u_{\text{prev},k})^{\mathrm{T}} (u_k - u_{\text{prev},k}) \tag{4.40}$$

where j = current MPC iteration, k = time counter in controller model, u = input trajectories (decision variables) at current MPC iteration, and u_{prev} = input trajectories calculated in previous MPC iteration $k-1$ (constants).

This multi-tiered approach produces trajectories that have a higher likelihood of having a qualitative resemblance to the trajectories in the previous step. By penalizing the deviation of the current input trajectory from the previous input trajectory, there is a possibility that the disturbance rejection abilities of the controller may be rendered more sluggish. We argue however that this issue is moderated by the fact that: (1) the penalty only occurs in the 2nd tier, so the system is remains economically optimal; (2) dynamic performance is not the primary or ultimate control objective here, and if there is an economic incentive for high dynamic performance, it will captured in the Tier 1 objective.

4.6 Case Studies

4.6.1 Open-loop Mixed-Integer Nonlinear Dynamic Optimization

Background

In this section, we study the use of several of the proposed formulations (α_{shut}, induced shutdowns, minimum shutdown duration) on the fiber line of a Kraft pulp and paper plant. Model equations for the Kraft fiber line are listed in Appendix A. Figs. A.1, A.4, A.5 show the plant topology. The problem is cast as Mixed-Integer Dynamic Optimization (MIDO) problem and discretized into a nonconvex Mixed-Integer Nonlinear Program (MINLP). At the present time, the high computational expense of solving an MINLP—as well as the unpredictability of its solve time—limit its utility in real-time applications such as closed-loop control. Thus we will only consider the open-loop control case in this *nonlinear* study. The closed-loop Mixed Integer Linear Program (MILP) *linear* control case will be considered in a later section, Section 4.6.3.

In this study, the prediction horizon length is $t_f = 12$ hrs, with a sampling time of $\Delta t = 15$ minutes. The minimum shutdown duration was $T^p = 8$ periods (or 2 hrs).

A multi-tiered solution approach (where Tier 1: economics, Tier 2: input move penalization and all binary variables fixed) was used for regularization. We solved the Tier 1 MINLP using Bonmin 1.4 [24], with the Branch-and-Bound algorithm selected. The Tier 2 NLP was solved using IPOPT 3.8.2[114]. All computations were carried out on a Core 2 Duo 3 GHz machine with 3GB RAM. For this model, the collocated problem had approximately 16,800 variables (of which 83 are binary), 19,500 constraints, and 54,870 Jacobian nonzeros after pre-solve. The two-tiered optimization approach (described in Section 3.2.5) was used, with the binary variables fixed in the second tier.

As a point of rigor, we also acknowledge that for a nonconvex problem, the bounds obtained by Bonmin via the solution of relaxed NLP subproblems using a local solver may not represent rigorous bounds, therefore the final solution obtained may be a local optimum or merely a feasible solution. As of the time of writing, there are few reliable MINLP codes that are capable of solving large-scale nonconvex problems to global optimality in a reasonable amount of time. However, for the purposes of our application, obtaining the global optimum is not strictly necessary; we submit that a feasible (or locally optimal) solution that obeys all the given constraints and that improves upon the nominal is often sufficient to aid the formulation of a partial shutdown policy.

Failure in the Hi-Q department, with induced shutdown

In this study, we shall consider a partial shutdown in the knotting department. In order to induce a shutdown (for the purposes of demonstration), we assume that the initial level of the blowtank, H^{BT} has been shifted from a nominal level of 60% to 65%. The duration of the failure in the knotter (the primary shutdown) was 7.75 hours. The long shutdown duration and the shifted levels are contrived to give rise to induced shutdowns in other units because the blowtank level is running close to its upper operating bound of 90%, and therefore the optimizer is disposed to pick solution trajectories that avoid the violation of this upper bound constraint.

Two MINLP cases are run. In the first instance, the economic cost of induced shutdowns was not considered in the problem. In the second, we use the formulation in Section 4.2 to properly account for the cost of induced shutdowns. In this latter case, induced shutdowns are indirectly

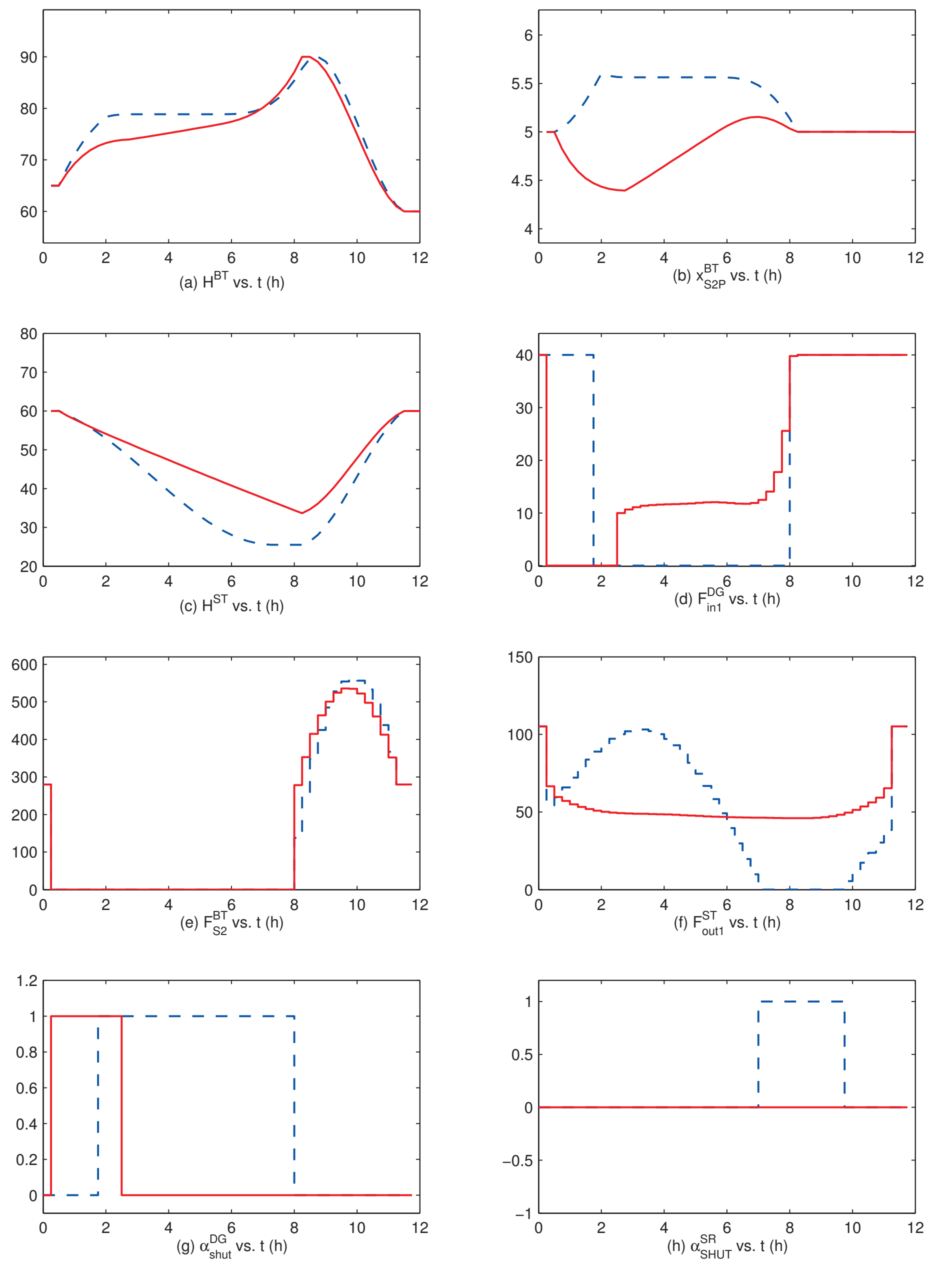


Figure 4.5: *Induced Shutdowns.* A comparison between a formulations that properly accounts for induced shutdown costs and one that does not.
Legend: Dashed lines = no induced shutdown cost. Solid lines = with induced shutdowns cost accounted for.

penalized, but are permitted if problem feasibility or overall economic considerations demand it.

Fig. 4.5 shows the results of two cases. The nominal shutdown in the knotter department is indicated by the blowtank outlet flowrate, $F_{S2}^{BT} = 0$ (Fig. 4.5e) during the shutdown period, $t = 0.25 - 8$ h.

In the first case (with no induced shutdown cost), the economics achieved was $\Phi^{*}_{\text{econ}} = \$47{,}236$ (shutdown costs were not accounted for) and the MINLP solve time was 10 hrs. In this instance, the optimizer prescribes an induced shutdown in the digester (as witnessed by $\alpha^{DG}_{\text{shut}} = 1$, Fig. 4.5g, between t=1.75 – 8 hr) and in the delignification department (as indicated by $\alpha^{SR}_{\text{shut}} = 1$, Fig. 4.5h, between t=6.5 – 9.75 hr). The production in the digester (F^{DG}_{in1}, Fig. 4.5d) is turned off in an effort to avoid overwhelming the blowtank with an influx of material that may cause its level upper-bound to be violated. In the case of the delignification department, the induced shutdown was driven by a economic motivation, which is to discharge as much of the material in the storage tank as possible, as seen in F^{ST}_{out1}, Fig. 4.5f. This is because: (1) as mentioned in Allison [5], an optimal course of action is often drive the storage vessels empty. (2) since there is no economic disincentive for shutdowns, the optimizer will pick a solution that will quickly discharge the tank contents and the shut off the outlet flow, which in turn causes induced shutdowns downstream of the buffer tank. This case shows the undesirable effects of not accounting for the costs induced shutdowns, as witnessed by the two additional shutdowns induced by the knotter shutdown.

In the second case (with induced shutdown cost), the cost of each induced shutdown—if they should occur—would be properly accounted for. The economics achieved was $\Phi^{*}_{\text{econ}} = \$48{,}346$ (with shutdown costs accounted for) and the MINLP solve time was 7 hrs 36 mins. If we examine the trajectories for $\alpha^{DG}_{\text{shut}}$ and $\alpha^{SR}_{\text{shut}}$ (Figs. 4.5g, h), we observe that as opposed to the previous case where induced shutdown costs were not accounted for, an induced shutdown of a much shorter duration occurs in the digester in the present case, and no induced shutdown happens in the delignification department downstream. We deduce that the induced shutdown in the digester in the present case was needed in order for the problem to be feasible, while the induced shutdown in the delignification department was found to be unnecessary. With this, we show that by properly accounting for the economics of shutdowns, the optimizer produces operating policies reflect the proper economic trade-offs in a partial shutdown situation.

Accounting for the economic costs of induced shutdowns may also be beneficial from a computational point of view, especially for non-unique problems. The induced shutdown term in the objective function can potentially help the optimizer differentiate between similar solutions (by acting as a "tie-breaker" for solutions that lead to the same nominal economics), which can in turn help reduce the branch-and-bound search space. An indication of this may be seen in the above two cases, where the solve time of the case with induced shutdown characterization is appreciably shorter than the other case.

For highly integrated systems with recycles, it can be difficult to predict which units are most critical to performance and profit. A model-based approach such as the one proposed here can aid in determining these critical units, and additional capital investment for increasing reliability can then be concentrated on them.

4.6.2 Minimization of Restoration Time with Multi-tiered Optimization

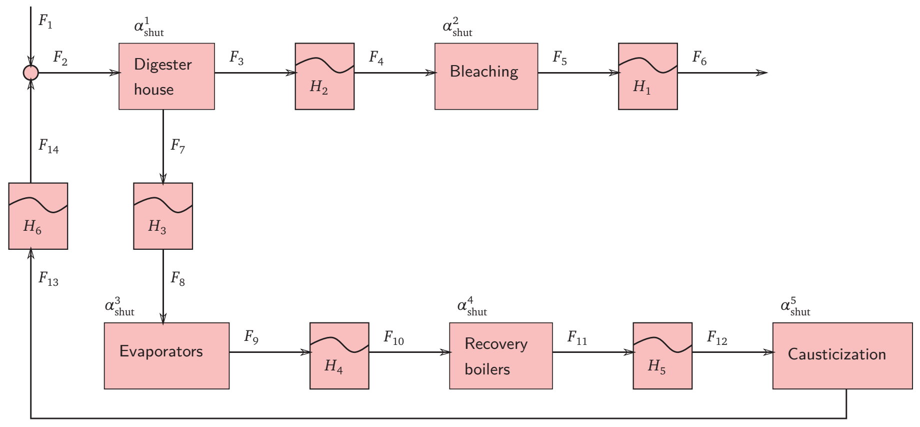


Figure 4.6: Leiviskä et al. pulp mill, plant topology.

Background

In this case study, we will demonstrate our restoration time minimization formulations (in section 4.4) on a pulp-and-paper plant model adapted from Leiviskä et al. [72]. The plant model is linear, and its topology is shown in Fig. 4.6. Only inventories are tracked; no compositions are considered. The chip feed stream F_1 is mixed with white liquor from stream F_{14} and conveyed to the digester house model, where it is cooked and reacted into pulp. The pulp is then bleached, and sent downstream for further processing. In the digester, spent liquor is produced in stream F_7, and is passed through a series of evaporators, boilers and causticizers in order to be regenerated. The model equations are found in Appendix B.

The state variables H_j $(j = 1, \ldots, 6)$ represent the tank levels in terms of percentages. The input variables F_i ($i = 1, \ldots, 14$) are stream flowrates in tons/hr. The variables α^p_{shut} ($p = 1, \ldots, 5$) are shutdown state indicators of each process unit p. The prediction horizon length is $t_f = 12$ hrs, with a sampling time of $\Delta t = 10$ minutes. The minimum shutdown duration was $T^p = 18$ periods (or 3 hrs). Shutdown thresholds correspond to a turndown ratio of 4. The cumulative amount of material exiting the plant from stream F_6 is defined as $I_{out}(t) = \left(\int_0^t F_6(\bar{t})\, d\bar{t} \right)$. The total shutdown time was penalized in the first tier to avoid induced shutdowns.

In this case study, the evaporators are shut down for $d_{est} = 4$ hours 30 minutes. The problems in Tier 1 (MILP) and Tier 2 (MIQP) have 331 binary variables, 5,525 linear variables, and 7,441 constraints. The solver used was CPLEX 12.2, and the total solve time (all 2 tiers) was 0.62 s. For comparison purposes, a nominal case is solved in which economics are maximized.

Discussion

The resulting optimal trajectories are shown in Fig. 4.7, with the solid lines representing the trajectories that are minimizing the restoration time, and dashed lines representing the maximum economics strategy. The shutdown is seen in the F_8 trajectory, where the material flow rate into the evaporators go to 0 during the shutdown period. The last shutdown ended at $t = 5.67$ hrs, and the minimum restoration time was at $t^*_{\text{res}} = 7.33$ hrs.

In some cases, faster restoration times may sometimes involve economic trade-offs, depending on how the economics is represented. We point out that in the case under consideration, by opting for a faster restoration, the overall production in the plant is diminished, as witnessed by the reduced throughput in I_{out}, a quantity on which the economics of this plant is based. In order to perform a rapid restoration, the minimum restoration time solution pursues a strategy in which flowrates $F_i, i = \{1, 2, 4, 6, 12, 14\}$ throughout the plant are throttled down rapidly to a low level during the shutdown and then throttled up again rapidly after the shutdown to effect the restoration as early as possible. This course of action causes a drastic drop in the overall throughput in the entire plant during the shutdown period.

If one were to compare the trajectories in Fig. 4.7 corresponding to the maximum economics and minimum restoration time cases respectively, one is likely to notice that in the former case, the input variables (the flowrates F_j) are adjusted in a more gradual fashion than in the latter. As it turns out, for the selected horizon of interest, the economic maximization policy produces a higher overall throughput (area under the F_6 graph). In the economic maximization policy, the level H_3 corresponding to the buffer tank prior to the shut down unit is allowed to accumulate material, and H_5 is seen to be gradually discharging material. This means the buffer capacities are being used more effectively in combating the partial shutdown than in the

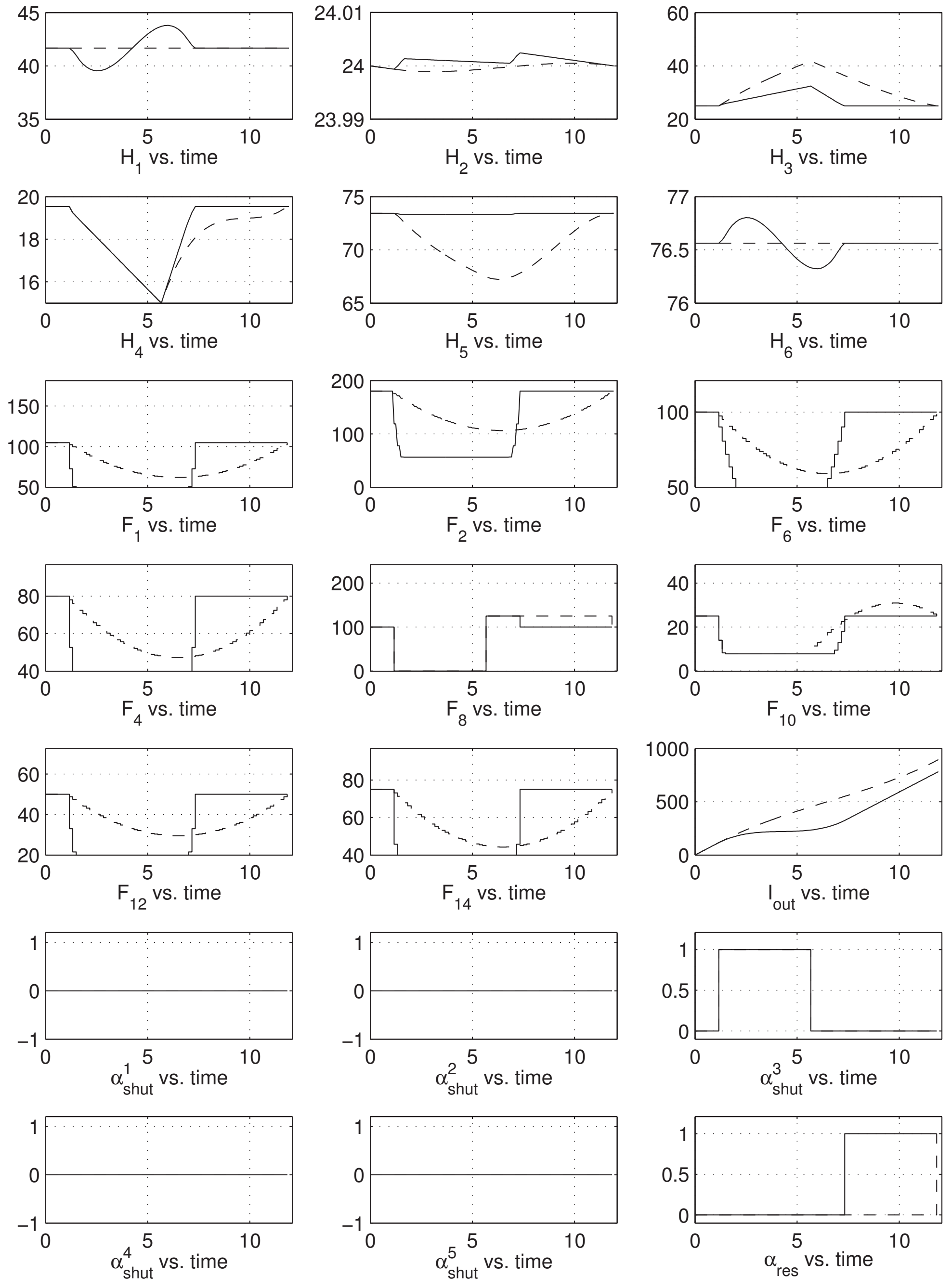


Figure 4.7: Minimization of restoration time in Leiviskä plant ($t^*_{\text{res}} = 7.33$ hrs, $\Phi^*_{\text{econ}} = \$260{,}340$). Legend: *solid lines* = multi-tiered minimization of restoration time. *dashed lines* = nominal case, maximizing economics. All flowrates F_j are tons/hr, levels H_i and I_{out} is in tons)

minimum restoration time strategy.

In the multi-tiered restoration time minimization case, the optimal restoration time and economics are $t^*_{\text{res}} = 7.33$ hrs and $\Phi^*_{\text{econ}} = \$260{,}340$ respectively. In nominal economic maximization case, they are $t^*_{\text{res}} = 11.83$ hrs and $\Phi^*_{\text{econ}} = \$396{,}420$. Therefore in this particular scenario, one is incurring a 34% loss in profits in order to obtain a restoration duration that is 38% shorter. An optimization-based tool allows decision-makers and operators to decide on the trade-off between performance and economics.

As an aside, in this minimum restoration formulation the total shutdown time was penalized in the first tier to avoid induced shutdowns, therefore no induced shutdowns occur. A case was run without this penalty, and several undesirable induced shutdowns are observed because their costs were not accounted for in the first tier.

4.6.3 Closed-loop Mixed-Integer Linear Controller

Background

In this case study, we will apply the model-based shutdown controller to the Leiviskä et al. [72] pulp mill described in the previous section. We will consider a case in which the evaporators are shut down for $d_{est} = 4$ hours 20 minutes. The initial prediction horizon length is $t_f = 12$ hrs, with a sampling time of $\Delta t = 10$ minutes. The minimum shutdown duration was $T^p = 18$ periods (or 3 hrs). Shutdown thresholds correspond to a turndown ratio of 4. A plant-model mismatch was introduced in the model parameters $\gamma_{3,2}$, $\gamma_{9,8}$, $\gamma_{13,12}$, where the values in the plant were 15% higher than in the controller model.

In each MPC iteration, an MILP with 5591 variables (292 binaries) was solved in Tier 1, while a QP was solved in Tier 2. The average solve time per MPC iteration (both tiers inclusive) with CPLEX 12.2 was 0.84 s.

Discussion

In Fig. 4.8, the evaporator shutdown is seen in the trajectory of $F_8 = 0$ from $t = 1.33$ to 5.33 hrs (effected by setting $\alpha^3_{\text{shut}} = 1$ for that period). The shutdown controller prescribes trajectories that balance the inventories in the various buffer tanks, accounting for the recycle loop. In order to keep the plant running upstream and downstream, the levels rise and fall in H_3 and H_4 respectively, which indicates an accumulation and discharge of material in those buffer tanks.

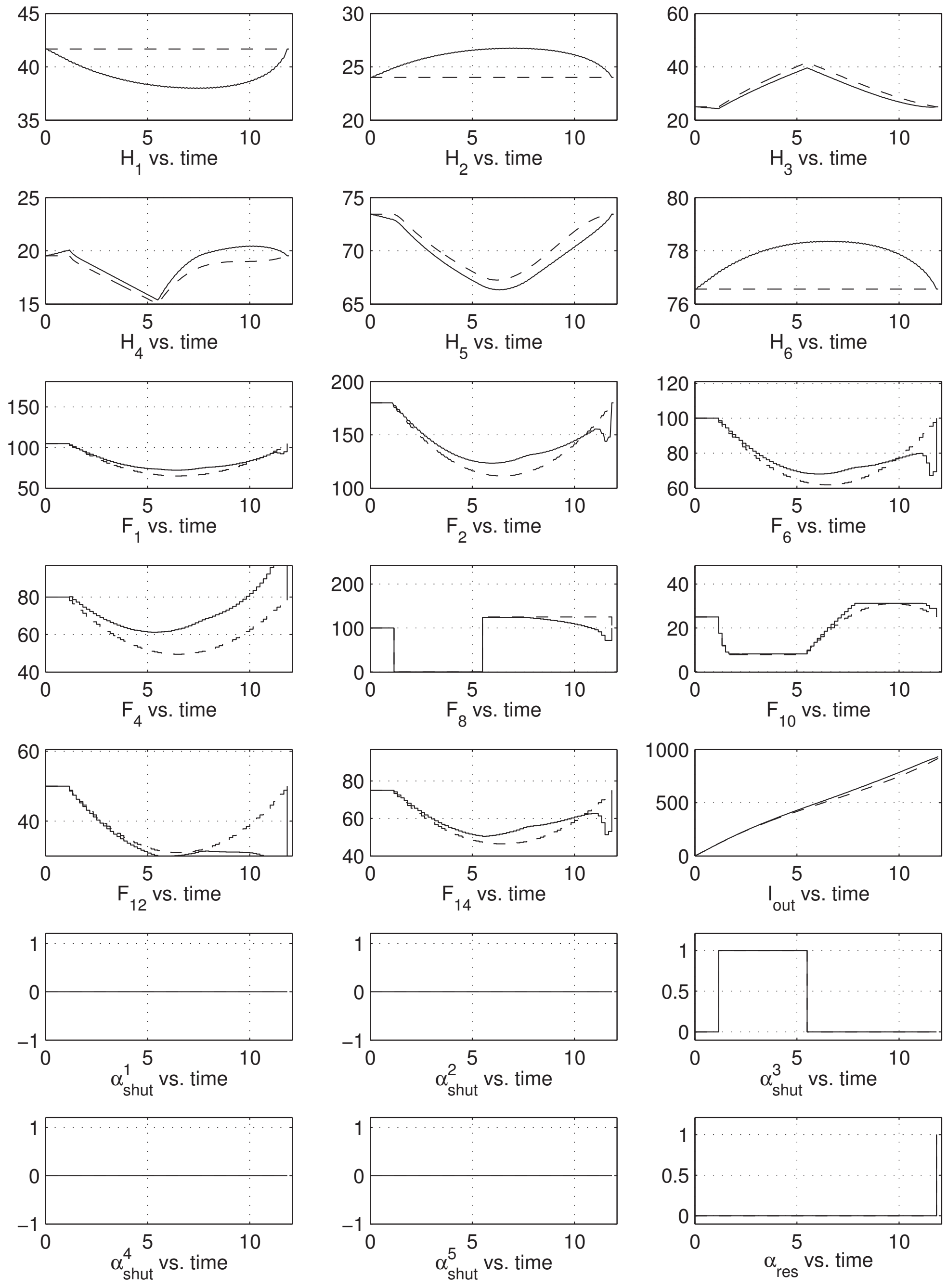


Figure 4.8: Shutdown in evaporators in Leiviskä plant, with plant-model mismatch.
Legend: *dashed lines* = nominal open-loop trajectories. *solid lines* = closed-loop trajectories. All flowrates F_j are tons/hr, levels H_i and I_{out} is in tons)

We also observe in Fig. 4.8 the difference between the trajectories actually implemented in closed-loop mode and the open-loop trajectories. This difference is due to the controller compensating for plant-model mismatch. This underlines the importance of applying feedback when implementing control policies in the presence of uncertainty. However, we also note that while the two sets of trajectories differ in their actual values, their qualitative behavior (i.e. general shape) remain similar as a result of our applying the two-tiered optimization approach. No induced shutdowns were observed to have occurred here; however, we can show that the recovery boilers will succumb to an induced shutdown (with a downtime of 3 hrs) if the evaporator shutdown were to be 20 minutes longer.

4.7 Conclusion

In this work, we outlined a strategy for obtaining optimal operating policies for partial shutdowns. In particular, we presented several efficient mixed-integer linear formulations that have tight continuous relaxations and are parsimonious in the number of binary variables they require. The only binary variables that appear in this chapter are in the sequence $\alpha^{p}_{\text{shut},k}$ $(p \in \mathscr{P}, k = 0, \ldots, N-1)$, and if restoration time minimization if desired, $\alpha_{\text{res},k}$ $(k = 0, \ldots, N-1)$.

Induced shutdowns are penalized in the objective by capturing the true economic cost of shutdowns using a discontinuous formulation. Rapid switching between shutdown and non-shutdown modes are dampened by restricting shutdowns to a given minimum duration.

As an alternative to economics maximization, we described a discontinuous optimization strategy for minimizing the restoration time. This formulation uses a multi-tiered approach whereby the restoration time is first minimized; the remaining degrees-of-freedom are then used to optimize the economics and minimize the input effort required.

An economics-driven MPC shutdown controller is put forward as a means of implementing the operating policies on a plant in the presence of uncertainties such as plant-model mismatch and disturbances. Downtime estimates can be updated manually through parametric feedback. In order to address the problem of solution non-uniqueness, a regularization method using multi-tiered optimization is presented.

Chapter 5

State and Parameter Estimation applied to Partial Shutdowns

5.1 Introduction

The implementation of a successful partial shutdown control policy (or any control policy for that matter) entails accurate tacit and explicit knowledge of the evolution of a plant, given certain inputs. This knowledge is frequently obtained through experience and from first principles. A large part of this knowledge can often be codified into the form of a mathematical model. In chemical engineering, mathematical models are usually constructed on the basis of historical data, experimental design or through the careful application of physical laws such as the conservation of mass, energy and momentum.

Very often, despite much care taken during the modeling process, predictions made using such models do not match the exact behavior of the plant. There are a host of possible reasons for this. For instance, the parameters identified during the model-building process may have shifted over the course of the plant's operation. Furthermore, it is frequently the case that the true model structure is not known exactly, or the selected model structure was perhaps (over)simplified for ease of computation. To address the discrepancy between the plant and the model, the mechanisms of feedback and estimation are commonly employed to continually correct the model's notion of the plant by updating it with new plant measurements at each sampling time.

A series of such measurements taken over time can be used to reconstruct the states and parameters of a model. In this work, we employ a modern technique—belonging to the Kalman family of filters—known as the Unscented Kalman Filter (UKF), due to Julier and Uhlmann [60] to perform this estimation online in a recursive fashion. UKFs are ordinarily used for state estimation, but they can be easily extended to accommodate both the estimation of states and uncertain parameters, as shown by Wan and Van der Merwe [109]. UKFs possess a number of advantages over traditional estimators like the Extended Kalman Filter (EKF), among them being the ability to handle arbitrary nonlinear—and possibly even discontinuous or nonsmooth—models, a more accurate propagation of states through nonlinear process dynamics by eschewing model linearization, and amenability to parallelization, which makes them computationally attractive for implementation on modern multicore processors.

State estimation is an important tool in predictive control deployments because in order for a model's predictions into future to be accurate, the controller needs to be in possession of accurate and up-to-date model states, including the values of the states that are unmeasured (and hence must be estimated). In control deployments where no state estimation scheme is employed, only a subset of states in process model are updated using plant measurements. This can lead to incorrect model predictions, which in turn may lead to potentially deleterious control actions being prescribed. Having incorrect values of model parameters can also beget similar consequences.

We propose to moderate these plant-model mismatch effects by continually correcting our controller model by estimating a selected set of parameters, as well as missing state measurements. In cases where the exact set of mismatched parameters is known, we can estimate them directly. For cases in which the sources of the mismatch are unknown (e.g. structural mismatch, persistent stochastic disturbances, etc.), fictitious disturbance terms are recursively estimated. In this work, we will investigate the use of UKF algorithms (with bound constraint handling) for performing state and parameter estimation on a differential-algebraic equation (DAE) system. In particular, these algorithms will be applied to systems under partial shutdown conditions.

The existence of plant-model mismatch also entails a modified approach to the implementation of an optimal control strategy, in our case, an Economics-based Model Predictive Control (MPC). In our MPC framework for partial shutdown handling, we typically solve an economics optimization problem in which hard terminal restoration constraints are used to drive the plant back to its original operating point post-shutdown. However, if a sufficiently large disturbance enters during the period of transience, it may render such terminal constraints infeasible. We attempt to address this problem by employing a multi-tiered dynamic optimization approach. Multi-tiered dynamic optimization permits the prioritization of several different objectives in order of decreasing importance. In the first tier, a dynamic feasibility problem is solved to compute a feasible terminal operating point. This is intended to enable the controller to move toward the next best feasible terminal operating point in the event that the originally prescribed terminal operating point becomes infeasible. A parameter relaxation is also performed in this tier to ensure that the estimated parameters arriving from the parameter estimator remain feasible throughout the prediction horizon. In the subsequent tier, an economic objective is maximized in order to find the most profitable way to control the plant. Within the space of economically optimal solutions, the next optimization tier attempts to maintain product quality targets as close to their desired values as possible. In the last optimization tier, the optimizer attempts to find a set of control trajectories that satisfy all the preceding objectives and can be implemented on the plant with the least control effort.

In the following sections, we will describe in detail the partial shutdown problem, the effects of plant-model mismatch, two state and parameter estimation algorithms based on the UKF, and finally, a multi-tiered optimization approach for performing optimization under plant-model mismatch conditions.

5.2 Partial Shutdown Problem

The partial shutdown formulation used in this chapter is identical to the one described in Section 3.2 for the continuous case, and Section 4.1 for the discrete-continuous case. In order to avoid repetition, we urge the reader to consult those sections for a statement of the canonical forms.

As a matter of convention, we will adopt in next several sections a nonlinear discrete state-space representation of the DAE system described in Eqn. 3.2 and Eqn. 3.4. This is done to simplify discussions on control and estimation. This state-space representation can be obtained from the aforementioned DAE form by means of a few simplifying statements. We assume that z_k is uniquely determined by u_k and x_k. We neglect the inequality constraints (for the sake of simplifying the discussion, though the inequality constraints will be present in the final optimization problem). A set of output variables y is computed from the differential (x) and algebraic (z) states by way of the matrices H^x and H^z respectively. A nonlinear state-space (M1) of the controller at any feedback iteration j (which is to be distinguished[1] from the model's

[1]Throughout this chapter, we will be using j to denote the feedback iteration counter, and k to denote the model's prediction horizon counter.

counter k) can then be written as follows:

$$\text{Model (M1)}: \quad x_{k+1} = f(x_k, z_k, u_k, \theta), \quad x_k = \hat{x}_j, \qquad k = j, \ldots, N-1 \tag{5.1}$$

$$h(x_k, z_k, u_k, \theta) = 0, \qquad k = j, \ldots, N \tag{5.2}$$

$$y_k = H^x x_k + H^z z_k, \qquad k = j, \ldots, N \tag{5.3}$$

where $k =$ the counter for the model prediction horizon, $N =$ the horizon length or endpoint, $\hat{x}_j \in \mathbb{R}^{n_x}$ is state estimate of the initial value at j, $z_k \in \mathbb{R}^{n_z}$ = algebraic states, and $y_k \in \mathbb{R}^{n_y}$ = output variables. Note that because this state-space model is derived from a DAE, we include Eqn. 5.2, which is an implicit relationship for resolving the values of the algebraic states z_k. We further require the Jacobian of h with respect to z, that is, h_z to be nonsingular. $H^x \in \mathbb{R}^{n_y \times n_x}$ and $H^z \in \mathbb{R}^{n_y \times n_z}$ are matrices which provide the output relationship.

In our control algorithm, we utilize a shrinking prediction horizon (as opposed to the more common receding horizon), therefore the endpoint N remains constant as j is incremented. The diagram shown in Fig. 5.5 may be helpful for visualizing M1 within its correct context, that is, inside a feedback loop.

5.3 Plant-Model Mismatch

The controller's notion of the plant (represented by M1) constitutes an approximation to reality. In a real plant, there exist uncertainties which M1 may not entirely capture. These uncertainties usually take one or more of following forms [75]:

1. Plant-model mismatch
2. Transient disturbances
   (a) Fast - detectable variations in the feed, conditions etc.
   (b) Slow - process drifts
3. Unknown initial conditions
4. Measurement Noise

Many of these forms of uncertainties, notably forms 2, 3 and 4, have traditionally been handled through the mechanisms of feedback and state estimation (MacGregor et al. [74]).

Under plant-model mismatch, the behavior of a controller model is expected to deviate from the observed behavior in the plant as time goes by. This manifests itself in a continual pattern of divergence between the value of the controller predicted variable $x_{j|j-1}$ (that is, predicted value of the states at j, given information at $j-1$) and the current state estimate, $\hat{x}_j$ (which is calculated from the current measurements). Intuitively, this indicates that the controller's future predictions are progressively getting farther and farther away from the behavior indicated by the measurements.

The aim of any mismatch-compensation scheme then is to adapt the controller model M1 to match the behavior of plant based on available information. In standard control situations, there are two primary types of mismatch, each requiring a different tack to address:

1. *Parametric Mismatch*
   This occurs when the model parameters θ do not match that of the plant. This typically arises due to incorrectly identified parameters in the model, or a parameter whose value has shifted over time (e.g. fouling changes the heat transfer coefficients in heat exchangers).

2. *Structural Mismatch*
   A structural mismatch occurs when the system is incorrectly or incompletely modeled.

In most situations, the overall plant-model mismatch is caused by a combination of the above. In order to account for the mismatch between the controller model and the plant, we can elect to perform a closed-loop adaptation of M1 to fit the measurement data arriving from the plant.

If the cause of the mismatch is primarily parametric and the model structure is fairly accurate, a direct estimation of the mismatched parameters θ can often yield good results. This was demonstrated in the work of Huang [56].

In our work, we also attempt to estimate fictitious disturbance terms to adapt the controller model to the plant. These terms (i.e. biases) essentially behave like regression fitting parameters for reconciling the observed behavior of the plant (through measurements) with the model's predictions. We denote the following disturbance-perturbed controller model as (M2):

$$\text{Model (M2):}\quad x_{k+1} = f(x_k, z_k, u_k + d_j^u, \theta, d_j^z) + d_j^x, \quad x_{k=j} = \hat{x}_j, \qquad k = j, \ldots, N-1 \quad (5.4)$$

$$h(x_k, z_k, u_k + d_j^u, \theta, d_j^z) = 0, \qquad k = j, \ldots, N \quad (5.5)$$

$$y_k = H^x x_k + H^z z_k + d_j^y, \qquad k = j, \ldots, N \quad (5.6)$$

where $d_j^x \in \mathbb{R}^{n_x}$ = differential state disturbance, $d_j^z \in \mathbb{R}^{n_z}$ = algebraic state disturbance, $d_j^u \in \mathbb{R}^{n_u}$ = input disturbance, and $d_j^y \in \mathbb{R}^{n_y}$ = output disturbance, all at iteration j. In a nominal model, all disturbance terms are 0 ($d_j^x = d_j^z = d_j^u = d_j^y = 0$), therefore when no disturbances are to be estimated, model M2 reduces to model M1.

All of these quantities can be estimated by treating them as stochastic white noise processes in the estimation problem. For the purposes of parameter estimation, these disturbances may be subsumed under the vector θ and treated using standard methods described in Sections 5.4.1 and 5.4.2. We wish to emphasize that as a rule of thumb, the number of disturbances that can be estimated is limited to the number of measurements[2] available [90]. Therefore typically not all disturbance terms are selected. The new θ vector may include both uncertain plant parameters as well as the selected fictitious disturbances:

$$\theta \triangleq B^t \theta_t + B^x d_j^x + B^z d_j^z + B^u d_j^u + B^y d_j^y \tag{5.7}$$

where θ_t is original vector of uncertain physical/model parameters, and B^t, B^x, B^z, B^y, B^y are matrices used to incorporate selected disturbance terms.

A survey of the various disturbance estimation methods is to be found in the literature review in Section 2.5.3. In the literature, many strategies for performing closed-loop adaptations (on a state-space type model) are based on estimating one or more combinations of the above-mentioned disturbances.

5.4 State and Parameter Estimation

Before we delve into the details of the estimation algorithms, we should mention the two conceptual strategies behind the state and parameter estimation algorithms we are investigating. They are known as *joint (state and parameter) estimation* and simple *parameter estimation*, and will be described at some length below. These strategies will be compared and contrasted in the case studies.

[2]This rule is based on the global observability property for linear systems and can be verified using observability tests. However, for general nonlinear systems, a certificate of global observability is often difficult to obtain, therefore this is considered a local property. This does not necessarily diminish its utility as a general rule of thumb.

5.4.1 Joint (State and Parameter) Estimation

State Augmentation

In joint estimation (also known as the *state augmentation* approach [74, 90]), the UKF algorithm is used to recursively compute both a state and parameter estimate $(\hat{x}_j, \hat{\theta}_j)$ at iterate j simultaneously. In order to achieve this, a transformation on the original problem is carried out where the parameters to be estimated are treated as stationary states. This additional fictitious stationary state appears as Eqn. 5.9 [82] via the relationship defined in Eqn. 5.7. The estimation model can be written as follows:

$$x_j = f(x_{j-1}, z_{j-1}, u_{j-1}, \theta_{j-1}) + w_{j-1}, \qquad x_{j-1} = \hat{x}_{j-1} \tag{5.8}$$

$$\theta_j = \theta_{j-1} + w_{\theta,j-1}, \qquad \theta_{j-1} = \hat{\theta}_{j-1} \tag{5.9}$$

$$h(x_k, z_k, u_k, \theta_k) = 0, \qquad k = j-1, \ldots, j \tag{5.10}$$

$$y_j = H^x x_j + H^z z_j + v_j \tag{5.11}$$

where $w_{j-1} \sim N(0, R_w)$ (process noise), $v_j \sim N(0, R_v)$ (measurement noise) and $w_{\theta,j-1} \sim N(0, R_\theta)$ (stochastic noise) are normally distributed random variables. The model functions f, h and matrices H^x, H^z used in estimation are identical to those of the controller model's (M1).

In the state estimation problem, x_{j-1} and θ_{j-1} are assumed to be a Gaussian random variables whose means are $\hat{x}_{j-1} = E[x_{j-1}]$ and $\hat{\theta}_{j-1} = E[\theta_{j-1}]$ respectively. Their error covariance matrices are defined as $P_{j-1} = E[(x_{j-1} - \hat{x}_{j-1})(x_{j-1} - \hat{x}_{j-1})^{\mathrm{T}}]$ and $P_{\theta,j-1} = E[(\theta_{j-1} - \hat{\theta}_{j-1})(\theta_{j-1} - \hat{\theta}_{j-1})^{\mathrm{T}}]$ respectively. We assume a system covariance R_θ for the parameter vector θ. These matrices are converted into the augmented form below.

By introducing definitions for the augmented quantities, we can rewrite the above system into an augmented form, whose form resembles a standard nonlinear state space model (we remind the reader that Eqn. 5.13 is simply an implicit function for computing the values of z_k). This makes it amenable to any standard state-estimation algorithm. We present the augmented model (M3) below (where the superscript "a" denotes augmented quantities):

$$\text{Model (M3)}: \quad x^{\mathrm{a}}_j = f^{\mathrm{a}}(x^{\mathrm{a}}_{j-1}, z_{j-1}, u_{j-1}), \qquad x^{\mathrm{a}}_{j-1} = \hat{x}^{\mathrm{a}}_{j-1} \tag{5.12}$$

$$h^{\mathrm{a}}(x^{a}_k, z_k, u_k) = 0, \qquad k = j-1, \ldots, j \tag{5.13}$$

$$y_j = H^{x,a} x^{\mathrm{a}}_j + H^z z_j \tag{5.14}$$

where

$$
\begin{aligned}
& x_k^{\mathrm{a}} \triangleq \begin{bmatrix} x_k \\ \theta_k \end{bmatrix}, \quad \hat{x}_k^{\mathrm{a}} \triangleq \begin{bmatrix} \hat{x}_k \\ \hat{\theta}_k \end{bmatrix}, \; k = j-1, \ldots, j \\
& f^{\mathrm{a}}(x_{j-1}^{a}, z_{j-1}, u_{j-1}) \triangleq \begin{bmatrix} f(x_{j-1}, z_{j-1}, u_{j-1}, \theta_{j-1}) \\ \theta_{j-1} \end{bmatrix} \\
& h^{\mathrm{a}}(x_k^{a}, z_k, u_k) \triangleq h(x_k, z_k, u_k, \theta_k), \; k = j-1, \ldots, j \\
& H^{x,a} \triangleq \begin{bmatrix} H^x & 0 \\ 0 & 0 \end{bmatrix} \\
& P_{j-1}^{\mathrm{a}} \triangleq \begin{bmatrix} P_{j-1} & 0 \\ 0 & P_{\theta,j-1} \end{bmatrix}, \quad R_w^{\mathrm{a}} \triangleq \begin{bmatrix} R_w & 0 \\ 0 & R_\theta \end{bmatrix}, \\
& R_v^{\mathrm{a}} \triangleq R_v
\end{aligned}
$$

If there are bound constraints in the states and parameters, they can be similarly augmented:

$$
x_L^{\mathrm{a}} \leq x_j^{\mathrm{a}} \leq x_U^{\mathrm{a}}, \quad \text{where } x_L^{\mathrm{a}} \triangleq \begin{bmatrix} x_L \\ \theta_L \end{bmatrix}, x_U^{\mathrm{a}} \triangleq \begin{bmatrix} x_U \\ \theta_U \end{bmatrix}
$$

These augmented quantities (along with Eqns. 5.12 – 5.14) can be fed into a standard Unscented Kalman-type state estimator framework, such as the one described in Section 5.4.1, to perform simultaneous state and parameter estimation with no modifications to the filter itself. Fig. 5.1 depicts the process of state augmentation and its place in the state / parameter estimation procedure. The Constrained UKF state estimator is not explicitly aware of the augmentation; it handles the augmented problem as it would a standard *state* estimation problem, that is, a separate parameter estimation algorithm is not required.

This strategy is useful in situations where there is a significant amount of interaction between the states and the parameters, and the user desires to estimate them simultaneously. The disadvantage is that the state estimation framework has to deal with a larger system of equations, which can increase the computational demands especially in an algorithm like the UKF whose complexity scales with the number of states.

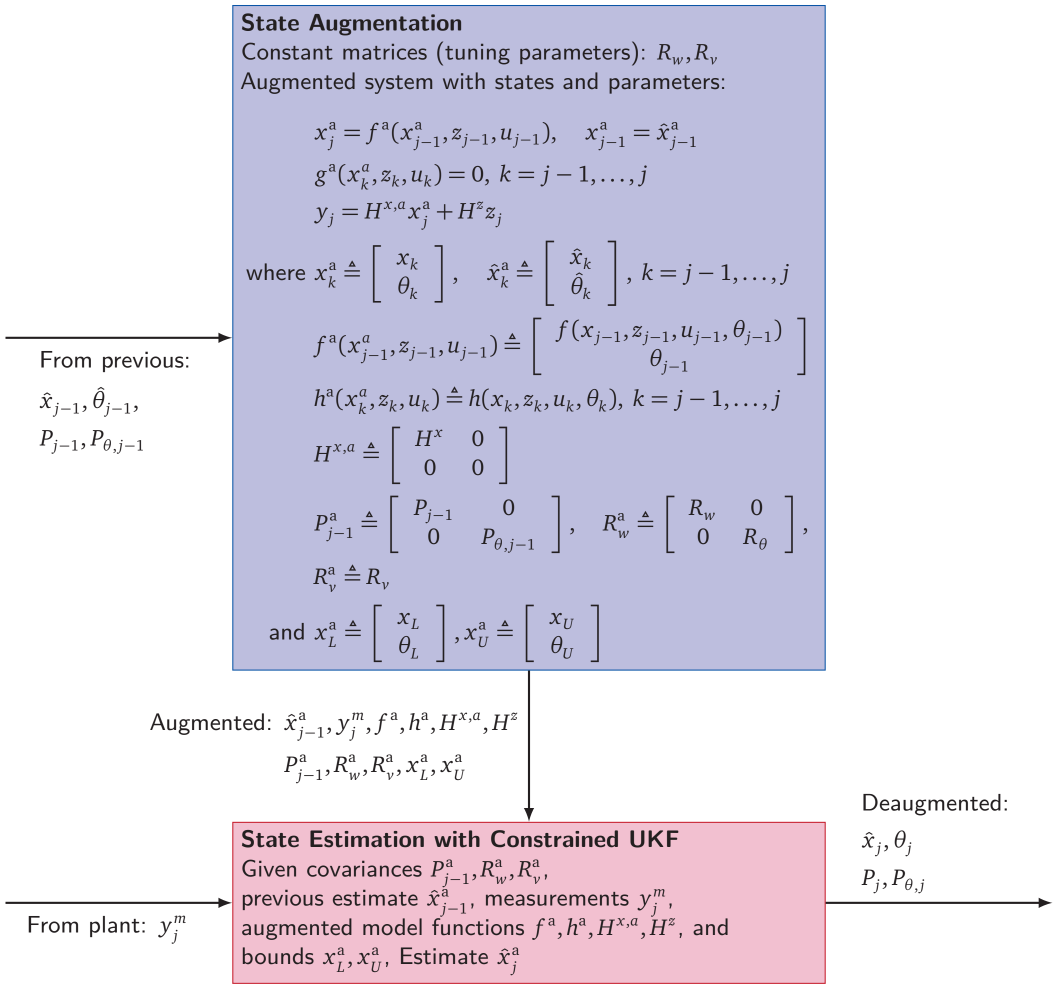

State Augmentation
Constant matrices (tuning parameters): R_w, R_v
Augmented system with states and parameters:
$x^a_j = f^a(x^a_{j-1}, z_{j-1}, u_{j-1}), \quad x^a_{j-1} = \hat{x}^a_{j-1}$
$g^a(x^a_k, z_k, u_k) = 0,\ k = j-1, \ldots, j$
$y_j = H^{x,a} x^a_j + H^z z_j$
where $x^a_k \triangleq \begin{bmatrix} x_k \\ \theta_k \end{bmatrix}, \quad \hat{x}^a_k \triangleq \begin{bmatrix} \hat{x}_k \\ \hat{\theta}_k \end{bmatrix}, k = j-1, \ldots, j$
$f^a(x^a_{j-1}, z_{j-1}, u_{j-1}) \triangleq \begin{bmatrix} f(x_{j-1}, z_{j-1}, u_{j-1}, \theta_{j-1}) \\ \theta_{j-1} \end{bmatrix}$
$h^a(x^a_k, z_k, u_k) \triangleq h(x_k, z_k, u_k, \theta_k),\ k = j-1, \ldots, j$
$H^{x,a} \triangleq \begin{bmatrix} H^x & 0 \\ 0 & 0 \end{bmatrix}$
$P^a_{j-1} \triangleq \begin{bmatrix} P_{j-1} & 0 \\ 0 & P_{\theta,j-1} \end{bmatrix}, \quad R^a_w \triangleq \begin{bmatrix} R_w & 0 \\ 0 & R_\theta \end{bmatrix},$
$R^a_v \triangleq R_v$
and $x^a_L \triangleq \begin{bmatrix} x_L \\ \theta_L \end{bmatrix}, x^a_U \triangleq \begin{bmatrix} x_U \\ \theta_U \end{bmatrix}$
From previous: $\hat{x}_{j-1}, \hat{\theta}_{j-1}, P_{j-1}, P_{\theta,j-1}$
Augmented: $\hat{x}^a_{j-1}, y^m_j, f^a, h^a, H^{x,a}, H^z$ $P^a_{j-1}, R^a_w, R^a_v, x^a_L, x^a_U$
Deaugmented: $\hat{x}_j, \theta_j$ $P_j, P_{\theta,j}$
From plant: y^m_j
State Estimation with Constrained UKF
Given covariances P^a_{j-1}, R^a_w, R^a_v,
previous estimate $\hat{x}^a_{j-1}$, measurements y^m_j,
augmented model functions $f^a, h^a, H^{x,a}, H^z$, and
bounds x^a_L, x^a_U, Estimate $\hat{x}^a_j$

Figure 5.1: Joint (state and parameter) Estimation Problem.

The Constrained Unscented Kalman Filter (UKF) for Nonlinear State Estimation

We present here the standard form of the UKF that is used for joint state and parameter estimation. In our UKF, we introduce a simple means for constraint handling that works on the basis of a projection of the states calculated during certain steps in the UKF into the feasible constrained region by solving a quadratic program (details to follow).

For convenience of abstraction, we can represent the model in Eqns. 5.12 – 5.14 as a mapping. We define the estimation model (E1) as follows:

$$\text{Model (E1)}: \quad f_{est} : (x^a_{j-1}, u_{j-1}) \mapsto (x^a_j, y_j) \tag{5.15}$$

The constrained UKF algorithm which we have synthesized from Wan and van der Merwe [115] and Kolås et al. [65] appears below.

UKF Algorithm Initialization *(prior to starting the filter algorithm)*

1. Select UKF tuning parameters α, β, κ.

   - $\alpha \in [1 \times 10^{-4}, 1]$ determines the spread of sigma points from the origin (a larger value produces a wider spread of points).

   - β is set to 2 for Gaussian systems. No guidance is available for non-Gaussian systems. However, the assumption is usually made that the system is Gaussian.

   - κ is generally set to 0, although Van der Merwe and Wan [109] suggest that $\kappa = 3 - n_x$ may be ideal for Gaussian systems. Kolås et al. [65] note that $\kappa < 0$ may lead to negative definite covariance calculations and care is needed when selecting this parameter. In our experience, 0 was a good choice for maintaining the positive-definiteness of the covariance matrix.

   Note: α, β, κ represent only one possible selection of tuning parameters. There exist other ways of tuning a UKF, each with their own individual properties [60, 65, 115]. As far as is known, no systematic evidence exists that any one method of tuning is superior to another owing to the effects of nonlinearity and stochasticity.

2. Using the tunings above, calculate weight terms and other algebraic parameters used in the UKF. The dimension of the augmented state vector is $n_a = n_x + n_\theta$.

$$\lambda = \alpha^2(n_a + \kappa) - n_a \tag{5.16}$$

$$\Gamma = n_a + \lambda \tag{5.17}$$

$$W_m^0 = \frac{\lambda}{n_a + \lambda} \tag{5.18}$$

$$W_c^0 = \frac{\lambda}{n_a + \lambda} + (1 - \alpha^2 + \beta) \tag{5.19}$$

$$W_m^i = W_c^i = \frac{0.5}{n_a + \lambda}, \quad i = 1, \ldots, 2n_a \tag{5.20}$$

3. Initialize the initial state error covariance matrix with variances of each state σ_x^2 on the diagonals, $P_0 = \text{diag}[\sigma_{x,1}^2, \ldots, \sigma_{x,n_x}^2]$. These initial variances can either be obtained from historical data, or arbitrarily selected based on some intuitive notion of the individual variances. Initialize $\hat{x}_0$ with the best available information on the initial state values.

4. Initialize the initial parameter error covariance matrix with variances of each parameter σ_θ^2 on the diagonals, $P_{\theta,0} = \mathrm{diag}[\sigma_{\theta,1}^2, \ldots, \sigma_{\theta,n_\theta}^2]$. Again, these initial variances can either be obtained from historical data, or arbitrarily selected based on how aggressive one desires the estimation process to be. Also, initialize $\hat{\theta}_0$ with the best available information.

5. Combine P_0 and $P_{\theta,0}$ into an augmented error covariance matrix, P_0^{a}. Also, combine $\hat{x}_0$ and $\hat{\theta}_0$ into an augmented state vector $\hat{x}_0^{\mathrm{a}}$

Recursive State Estimation using the UKF

1. A total number of $2n_a + 1$ sigma points, χ_{j-1}^i (where $i = 0, \ldots, 2n_a$) are generated around the last available estimate $\hat{x}_{j-1}^{\mathrm{a}}$ (at the first iteration, this would be $\hat{x}_0^{\mathrm{a}}$).

$$\chi_{j-1}^0 = \hat{x}_{j-1}^{\mathrm{a}} \tag{5.21}$$

$$\chi_{j-1}^i = \hat{x}_{j-1}^{\mathrm{a}} + \Gamma(\sqrt{P_{j-1}^{\mathrm{a}}})_i, \qquad i = 1, \ldots, n_a \tag{5.22}$$

$$\chi_{j-1}^{n_a+i} = \hat{x}_{j-1}^{\mathrm{a}} - \Gamma(\sqrt{P_{j-1}^{\mathrm{a}}})_i, \qquad i = 1, \ldots, n_a \tag{5.23}$$

Note: the notation $\hat{x}_{j-1}^{\mathrm{a}} \pm \Gamma(\sqrt{P_{j-1}^{\mathrm{a}}})_i$ denotes a vector $\hat{x}_{j-1}^{\mathrm{a}}$ being added or subtracted to column i of the $\Gamma(\sqrt{P_{j-1}^{\mathrm{a}}})$ matrix. The square-root of the matrix can be obtained either by through a Cholesky factorization or a symmetric matrix square root algorithm. The latter has favorable properties [65] and will be used in this work.

2. Each sigma point $\chi_{j-1}^i (i = 0, \ldots, 2n_a)$ undergoes the nonlinear transformation defined in (5.15). This yields a cloud of transformed sigma points for the predicted states $\chi_j^{i,-}$ and predicted outputs Υ_j^i at the current time step, as follows[3]:

$$f_{est} : (\chi_{j-1}^i, u_{j-1}) \mapsto (\chi_j^{i,-}, \Upsilon_j^i), \quad i = 0, \ldots, 2n_a \tag{5.24}$$

The statistics of the Gaussian random variable are estimated by computing the mean and variance of the transformed sigma points.

3. The predicted (*a priori*) states $\hat{x}_j^-$ (mean) are calculated by taking a weighted combination of the sigma points:

$$\hat{x}_j^- = \sum_{i=0}^{2n_a} W_m^i \chi_j^{i,-} \tag{5.25}$$

[3]The superscript-minus ($^-$) notation is used to denote an *a priori* predicted variable.

Similarly, the predicted (*a priori*) covariance matrix P_j^- is computed from the transformed sigma points:

$$P_j^- = \sum_{i=0}^{2n_a} W_c^i (\chi_j^{i,-} - \hat{x}_j^-)(\chi_j^{i,-} - \hat{x}_j^-)^{\mathrm{T}} + R_w^{\mathrm{a}} \tag{5.26}$$

where W_m^i and W_c^i are determined in eqns. 5.18 – 5.20.

4. The predicted outputs are calculated by taking a weighted combination of the transformed output sigma points:

$$\hat{y}_j = \sum_{i=0}^{2n_a} W_m^i \Upsilon_j^i \tag{5.27}$$

A covariance matrix for the outputs P_y is calculated:

$$P_y = \sum_{i=0}^{2n_a} W_c^i (\Upsilon_j^i - \hat{y}_j)(\Upsilon_j^i - \hat{y}_j)^{\mathrm{T}} + R_v \tag{5.28}$$

5. **[Constraint handling step]** The results of the previous step are projected into the feasible space by means of the QP problem defined in Eqn. 5.37.

$$\hat{x}_j^- := Q_{\mathrm{opt}}(\hat{x}_j^-, P_j^-, x_L, x_U) \tag{5.29}$$

6. A matrix representing the cross-covariance between x and y is calculated:

$$P_{xy} = \sum_{i=0}^{2n_a} W_c^i (\chi_j^{i,-} - \hat{x}_j^-)(\Upsilon_j^i - \hat{y}_j)^{\mathrm{T}} \tag{5.30}$$

The Kalman gain K_j is then constructed as follows:

$$K_j = P_{xy} P_y^{-1} \tag{5.31}$$

7. Using the current measurement y_j^m, the predicted states $\hat{x}_j^-$ are updated to produce a state estimate $\hat{x}_j^{\mathrm{a}}$:

$$\hat{x}_j^{\mathrm{a}} = \hat{x}_j^- + K_j(y_j^m - \hat{y}_j) \tag{5.32}$$

The updated (*a posteriori*) covariance matrix P_j^{a} is computed:

$$P_j^{\text{a}} = P_j^{-} - K_j P_y K_j^{\text{T}} \tag{5.33}$$

8. **[Constraint handling step]** The results of the previous step are projected into the feasible space by means of the QP problem defined in Eqn. 5.37.

$$\hat{x}_j^{\text{a}} := Q_{\text{opt}}(\hat{x}_j^{\text{a}}, P_j^{\text{a}}, x_L^{\text{a}}, x_U^{\text{a}}) \tag{5.34}$$

9. Deaugment x_j^{a} into $\hat{x}_j, \theta_j$, and P_j^{a} into $P_j, P_{\theta,j}$. Return to Step 1 for the next iteration.

QP Projection for Bound Constraint Handling in the UKF

A general projection quadratic program (QP) can be written as follows:

$$\min_x (x-\bar{x})^{\text{T}} \bar{P}^{-1} (x-\bar{x}) \tag{5.35}$$

$$\text{s.t. } x_L \leq x \leq x_U \tag{5.36}$$

where $\bar{x}$ and $\bar{P}$ are generic arguments for some state vector and covariance matrix, respectively. Simon [96] notes that by choosing the weighting matrix as the inverse of a covariance $\bar{P}^{-1}$ (otherwise known as an "information" matrix), a maximum probability estimate of the state (under constraints) is obtained. If identity matrix I was chosen as the weighting matrix, one obtains a least-squares estimate of the constrained states. In this work, we elect to use an inverse covariance weighting to preserve the statistics of the estimate.

For practical computation, the above problem is often rewritten into standard quadratic programming form:

$$Q_{\text{opt}}(\bar{x}, \bar{P}, x_L, x_U) : \begin{cases} & \min_x \frac{1}{2} x^{\text{T}} \bar{Q} x + \bar{f}^{\text{T}} x \\ \text{s.t.} & x_L \leq x \leq x_U \\ \text{where} & \bar{Q} = \frac{1}{2}\left[\bar{P}^{-1} + (\bar{P}^{-1})^{\text{T}}\right] \\ & \bar{f} = -\bar{P}^{-1}\bar{x} \end{cases} \tag{5.37}$$

where $\bar{Q} = \frac{1}{2}\left[\bar{P}^{-1} + (\bar{P}^{-1})^{\text{T}}\right]$ is a symmetrized version of the information matrix $\bar{P}^{-1}$. We have found this symmetrization to be necessary because although $\bar{P}^{-1}$ is symmetric in theory, small amounts of numerical noise arising during the estimation may render the matrix non-symmetric in practice. For the same reason, while the covariance matrix $\bar{P}$

is theoretically always invertible (in our algorithm, the covariance matrices are positive-definite, and all positive-definite matrices are invertible), it may be necessary in practice to use a Moore-Penrose pseudo-inverse (obtained through a least squares solution of a linear system) as accumulated numerical noise during the iterations can introduce ill-conditioning to the matrix.

In this work, we have chosen to limit the type of constraints in our UKF estimator to simple state (and parameter) bounds because we believe it represents a choice that is congruent with the complexity of the constraints arising in our application. In principle, one can handle arbitrary nonlinear constraints by rewriting the problem as a nonlinear program [65].

5.4.2 Simple Parameter Estimation

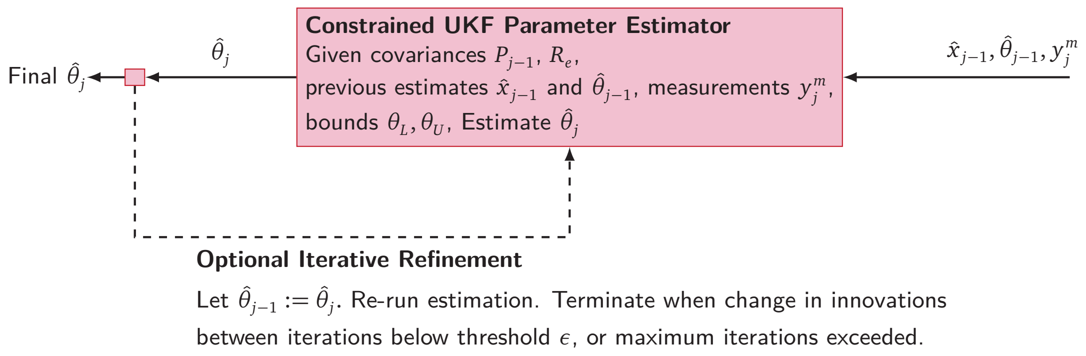

Constrained UKF Parameter Estimator
Given covariances P_{j-1}, R_e,
previous estimates $\hat{x}_{j-1}$ and $\hat{\theta}_{j-1}$, measurements y_j^m,
bounds θ_L, θ_U, Estimate $\hat{\theta}_j$
$\hat{x}_{j-1}, \hat{\theta}_{j-1}, y_j^m$
$\hat{\theta}_j$
Final $\hat{\theta}_j$
Optional Iterative Refinement
Let $\hat{\theta}_{j-1} := \hat{\theta}_j$. Re-run estimation. Terminate when change in innovations
between iterations below threshold ϵ, or maximum iterations exceeded.

Figure 5.2: Simple Parameter Estimation.

In situations where only parameter estimation capabilities are desired, the UKF estimation can be modified to perform parameter estimation exclusively. In a sense, the resulting algorithm resembles a standard Recursive Least Squares (RLS)[51] filter. However our algorithm offers several advantages. By adopting a constrained UKF-based framework, it inherits all the advantages of the UKF algorithm (no linearization step, arbitrary nonlinear model, easy parallelization, etc.) and also adds simple bound constraint handling.

This constrained UKF parameter estimation algorithm is well-suited to situation where full-state measurements are available, and only estimations of parameters are required. If full-state measurements are not available, the states can be obtained from a separate state-estimator. When the parameter estimator is paired with a state estimator, it is known as the *dual estimation* approach. This decoupling of the state estimator from the parameter estimator offers several benefits. Firstly, the parameter estimator algorithm is often simpler (in terms of the number of steps that need to be carried out) and potentially computationally less intensive to run than a corresponding joint estimation algorithm. If there are a large number of parameters, this may offer a speed advantage over a joint estimation scheme in which the parameters are subject to the computational demands exerted by the full state estimator.

Also, because states and parameters usually evolve with different dynamics, the separation of state and parameter estimators provides a finer level of control over the parameter estimation process, especially through additional tuning parameters such as the exponential forgetting factor λ_f, which may improve tracking performance in some situations.

We have discovered that the parameter estimation can be improved through a numerical refinement procedure (see Fig. 5.2 for a visual depiction). We offer the following (optional) iterative refinement procedure:

1. Denote the iteration count as i, and the parameter estimate vector $\hat{\theta}_j$ at iteration i as $\hat{\theta}_j^i$. In the first iteration, the parameter estimate vector $\hat{\theta}_j^1$ is obtained from the parameter estimator. Then, let $i = 2$.

2. The vector $\hat{\theta}_j^{i-1}$ is fed into the parameter estimator as an input ($\hat{\theta}_{j-1}$), and the estimation is re-run. A new (refined) $\hat{\theta}_j^i$ is obtained.

3. Terminate and return the current estimate $\hat{\theta}_j^i$ if one of the following conditions are met:

   (a) The relative change in the innovations is less than a specified numerical threshold (ϵ):
   $$\frac{||(y_j^m - y_j)_i - (y_j^m - y_j)_{i-1}||}{||(y_j^m - y_j)_{i-1}||} \leq \epsilon$$
   where $(y_j^m - y_j)^i$ is the innovations at iteration i.

   (b) The maximum number of iterations (a user-specified parameter) is exceeded.

   Otherwise, let $i := i + 1$ and return to step 2.

We consider this an optional procedure because the computational demands of repeating a full parameter estimation may be quite substantial in larger problems. However, in our experience, iterative refinement can lead to a significant acceleration in estimation convergence.

At this point, we remark that in a process model the uncertain parameters of interest can either be time-invariant or time-varying. It could be argued that if the desired parameters were time-invariant, the estimation of parameters can be performed offline exclusively using only historical data. While this is true, we believe that a reasonably good case could be made for using a recursive online estimation algorithm such as the UKF, as a complement to (instead of a replacement for) offline estimation. In practice, the true values of the parameters in the plant may drift over the course of a plant's operation, therefore the parameters obtained using offline estimation of historical data may not necessarily be up to date. Recursive estimation algorithms on the other hand use live data and provide the most up to date estimates. We see the two as being complementary, because the parameters that obtained through offline estimation can be used as initial guesses in the online estimation. Starting with parameter values close to the actual values in the plant can accelerate the convergence of the online estimation considerably.

We note also that a drawback of the exclusive offline estimation approach in our context is the limited availability of historical plant data under partial shutdowns (especially during the restoration phase). In many plants, partial shutdowns do not usually happen very frequently, therefore there may be a dearth of plant data under such conditions. Partial shutdowns also usually push the plant outside of their usual ranges of operation, and nominal plant operation data may be unsuitable for the purposes of offline parameter estimation for partial shutdowns. These factors provide the motivation for using a online parameter estimator.

The Constrained Unscented Kalman Filter for Nonlinear Parameter Estimation

As mentioned before, although the UKF algorithm is primarily designed for state estimation, it can easily be used for the purposes of parameter estimation. In this section, we employ a modified UKF that is used to *exclusively* estimate parameters, inspired primarily by work done by Van der Merwe and Wan [109] who outlined a modification that operates on principles similar to a weighted recursive least squares (W-RLS) algorithm. In this work, we extend the aforementioned work of Van der Merwe and Wan [109] by including iterative refinement and constraint handling (similar in vein to the method described in the constrained UKF section).

At this juncture, we remind the reader that the number of parameters that can be estimated is typically limited to the number of measurements available, $n_\theta \leq n_y$. Attempting to estimate more parameters than available measurements can potentially lead to nonunique estimates, which may in some cases cause deterioration in the quality of the estimation. For convenience of abstraction, we can represent the model in Eqns. 5.12 – 5.14 as a mapping. We define the estimation model (E2) as follows:

$$\text{Model (E2)}: \quad f_{p,est} : (x_{j-1}, u_{j-1}, \theta_{j-1}) \mapsto y_j \tag{5.38}$$

We describe the algorithm below:

Recursive Parameter Estimation using a constrained UKF algorithm

1. At a time step j, we carry out the prediction step using the following equations:

$$\hat{\theta}_j^- = \hat{\theta}_{j-1} \tag{5.39}$$

$$P_{\theta,j}^- = \frac{1}{\lambda_f} P_{\theta,j-1} \tag{5.40}$$

   where λ_f is the exponential forgetting factor, often chosen to be 1 or less (see discussion in box below).

2. Sigma points Θ_{j-1}^i (where $i = 0 \dots 2n_\theta$) are generated around the prediction $\hat{\theta}_j^-$.

$$\Theta_j^0 = \hat{\theta}_j^- \tag{5.41}$$

$$\Theta_j^i = \hat{\theta}_j^- + \Gamma(\sqrt{P_{\theta,j}^-})_i, \qquad i = 1 \dots n_\theta \tag{5.42}$$

$$\Theta_j^{n_\theta+i} = \hat{\theta}_j^- - \Gamma(\sqrt{P_{\theta,j}^-})_i, \qquad i = 1 \dots n_\theta \tag{5.43}$$

   Note that the notation $\hat{\theta}_j^- \pm \Gamma(\sqrt{P_{\theta,j}^-})_i$ denotes a vector $\hat{\theta}_j$ being added or subtracted to column i of the $\Gamma(\sqrt{P_{\theta,j}^-})$ matrix.

3. Each sigma point Θ_{j-1}^i undergoes the nonlinear transformation defined by Model E2 (eqn. 5.38). For a dynamic system, the state-space differential equation is our transformation of interest. This yields a cloud of transformed sigma points:

$$f_{p,est} : (\hat{x}_{j-1}, u_{j-1}, \Theta_j^i) \mapsto Y_j^i, \qquad i = 0 \dots 2n_\theta \tag{5.44}$$

   We assume that the state estimate $\hat{x}_{j-1}$ here is obtained using some state estimator (e.g. the standard UKF) or measurements (in the case of full-state feedback). The transformed sigma points of the output variable, Y_j^i can be thought of as an output estimate resulting from the perturbation of the parameter vector θ, and hence conveys and implicit notion of the sensitivity of the output variables with respect to the parameters.

The predicted outputs $\hat{y}_j$ (mean) are calculated by taking a weighted combination of the sigma points:

$$\hat{y}_j = \sum_{i=0}^{2n_\theta} W_m^i Y_j^i \tag{5.45}$$

Similarly, the predicted (*a priori*) output covariance matrix P_y is computed from the transformed sigma points:

$$P_y = \sum_{i=0}^{2n_\theta} W_c^i (Y_j^i - \hat{y}_j)(Y_j^i - \hat{y}_j)^{\mathrm{T}} + R_e \tag{5.46}$$

where R_e is a tuning matrix which influences the aggressiveness of the estimator (by virtue of its effect on the Kalman gain). A diagonal matrix whose diagonal values lie between 0.01 to 5 is typically chosen, with smaller values leading to higher Kalman gains, and hence more aggressive estimation.

4. **[Constraint handling step]** The results of the previous step are projected into the feasible space by means of the QP problem defined in Eqn. 5.37.

$$\hat{y}_j := Q_{\mathrm{opt}}(\hat{y}_j, P_y, y_L, y_U) \tag{5.47}$$

5. A matrix representing the cross-covariance between θ and y is calculated:

$$P_{\theta y} = \sum_{i=0}^{2n_\theta} W_c^i (\Theta_j^i - \hat{\theta}_j^-)(Y_j^i - \hat{y}_j)^{\mathrm{T}} \tag{5.48}$$

The Kalman gain $K_{\theta,j}$ is then constructed as follows:

$$K_{\theta,j} = P_{\theta y} P_y^{-1} \tag{5.49}$$

6. Using the current measurement y_j^m, the predicted parameters $\hat{\theta}_j^-$ are updated to produce a parameter estimate $\hat{\theta}_j$:

$$\hat{\theta}_j = \hat{\theta}_j^- + K_{\theta,j}(y_j^m - \hat{y}_j) \tag{5.50}$$

The updated (*a posteriori*) covariance matrix P_j is computed:

$$P_{\theta,j} = P^{-}_{\theta,j} - K_{\theta,j} P_y K^{\mathrm{T}}_{\theta,j} \tag{5.51}$$

7. **[Constraint handling step]** The results of the previous step are projected into the feasible space by means of the QP problem defined in Eqn. 5.37.

$$\theta_j := Q_{\text{opt}}(\hat{\theta}_j, P_{\theta,j}, \theta_L, \theta_U) \tag{5.52}$$

Exponential Forgetting Factor
In an recursive least-squares context, an exponential forgetting factor $\lambda_f \in (0,1]$ is often used as a means of prioritizing current information over older information. Common values of λ_f are usually between 0.950 and 0.999. A lower value for the forgetting factor can improve tracking speed, which aids convergence in situations where the parameter being estimated is static or drifts very slowly relative to the system dynamics. In regulatory control, the forgetting factor can also help avoid a "parameter burst" situation when the level of persistent excitation is small — however, in our particular case this is less of an issue as the system is in dynamic transience.

The forgetting factor is related to quantity known as the *asymptotic memory length* (n_A), which is the length of past steps to remember. It is defined as follows:

$$n_A = \frac{\lambda_f}{1 - \lambda_f} \tag{5.53}$$

When $\lambda_f \to 1$ (the default case), $n_A \to \infty$ which implies all past information is retained.

5.5 Estimation during Shutdowns

When a unit (or set of units) $p \in \mathscr{P}_{shut}$ undergoes a shutdown, the controller loses the measurements for those particular units. As well, their states and parameters cease to exist.

In the controller model, by way of the α_{shut} formulation, the inputs and outputs of these units are set to 0, and the floating variables (algebraic states such as composition, concentration etc.) are fixed at some nominal value. This is essentially equivalent to removing the model equations and variables for the shutdown units from the optimization problem. The α_{shut} approach permits the algorithm to continue solving the controller optimization problem (to determine control actions for the rest of the plant) without fundamentally altering the structure of the model.

From a state and parameter estimation point of view however, the loss of states, parameters and measurements from the shutdown units entails a change in the estimation problem during the shutdown period. That is, a reduced estimator (with shrunken covariance, measurement and state matrices to account for the loss of states, parameters and measurements) is to be used during the shutdown duration as opposed to the nominal estimator used the rest of the time.

To demonstrate how the matrices and vectors are reduced during the shutdown period, consider an arbitrary covariance matrix C and state vector v that is used during nominal operation. We also derive a corresponding matrix I_C and a vector I_v containing their indices (this will be used to map values between the full and reduced forms).

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad I_C = \begin{bmatrix} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \end{bmatrix}, \quad I_v = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Suppose the second state v_2 ceases to exist during the shutdown. We shrink the C and v quantities (and their corresponding index matrix I_C and vector I_v) accordingly, that is, the second row and column of the two matrices, and second row of the two vectors, are eliminated:

$$C' = \begin{bmatrix} c_{11} & c_{13} \\ c_{31} & c_{33} \end{bmatrix}, \quad v' = \begin{bmatrix} v_1 \\ v_3 \end{bmatrix}, \quad I'_C = \begin{bmatrix} (1,1) & (1,3) \\ (3,1) & (3,3) \end{bmatrix}, \quad I'_v = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

These reduced quantities C', v' are sent to the estimator during the shutdown period. The estimator therefore sees a smaller estimation problem, but it treats it like any other estimation problem and produces the corresponding estimates ($\hat{v}'$) and covariance update ($\hat{C}'$) with conforming dimensions:

$$\hat{C}' = \begin{bmatrix} \hat{c}_{11} & \hat{c}_{13} \\ \hat{c}_{31} & \hat{c}_{33} \end{bmatrix}, \quad \hat{v}' = \begin{bmatrix} \hat{v}_1 \\ \hat{v}_3 \end{bmatrix},$$

As the estimator exits the shutdown period, the above quantities can be mapped back to their original dimensions using the indices I'_C and I'_v:

$$C(I'_C) := \hat{C}' \rightarrow C = \begin{bmatrix} \hat{c}_{11} & c_{12} & \hat{c}_{13} \\ c_{21} & c_{22} & c_{23} \\ \hat{c}_{31} & c_{32} & \hat{c}_{33} \end{bmatrix}, \qquad v(I'_v) := \hat{v}' \rightarrow v = \begin{bmatrix} \hat{v}_1 \\ v_2 \\ \hat{v}_3 \end{bmatrix}$$

The nominal estimator then resumes estimation with the full matrices and vectors. Note that the quantities that did not exist during the shutdown are simply resumed with their pre-shutdown values. The relevant matrices and vectors that are to be transformed into their reduced forms during shutdown are the following:

- Joint (state and parameter) estimation: $P, R_w, R_v, \hat{x}_j, y_j^m, \theta_j$
  The model function f_{est} has to be adjusted to accommodate the new dimensions of the matrices and vectors above.

- Parameter estimation: $P, R_w, R_e, \hat{x}_j, y_j^m, \theta_j$
  The model function $f_{p,est}$ has to be adjusted accordingly.

5.6 Multi-tiered dynamic optimization

In this section, we describe a multi-tiered approach for performing control optimization that we have embedded within the open-loop dynamic optimization portion of our MPC framework. The general idea behind multi-tiered optimization is that at each tier, some threshold quantity is computed using an optimization routine and is enforced in subsequent tiers. This approach represents a means of systematically prioritizing competing objectives in a problem with non-unique solutions. Arguably, each of the above tiers can be integrated into a single tier with appropriate weights on their respective objectives (De Souza et al. [35] and Zanin et al. [122] show an example where Tiers 2 and 4 are merged into a weighted objective), but the choice of such weights can be arbitrary. We believe that a multitiered approach provides a clean and natural separation of the objectives based on their priorities.

In Tier 1, we compute feasible restoration targets as well as perform a relaxation of the parameter estimate throughout the horizon. In previous chapters, we have established that in a partial shutdown situation, the controller's optimal course of action from an economic standpoint is to run down the tanks to their minimum levels at the end of the control horizon. Given a finite prediction horizon, this will render the plant incapable of anticipating the next shutdown,

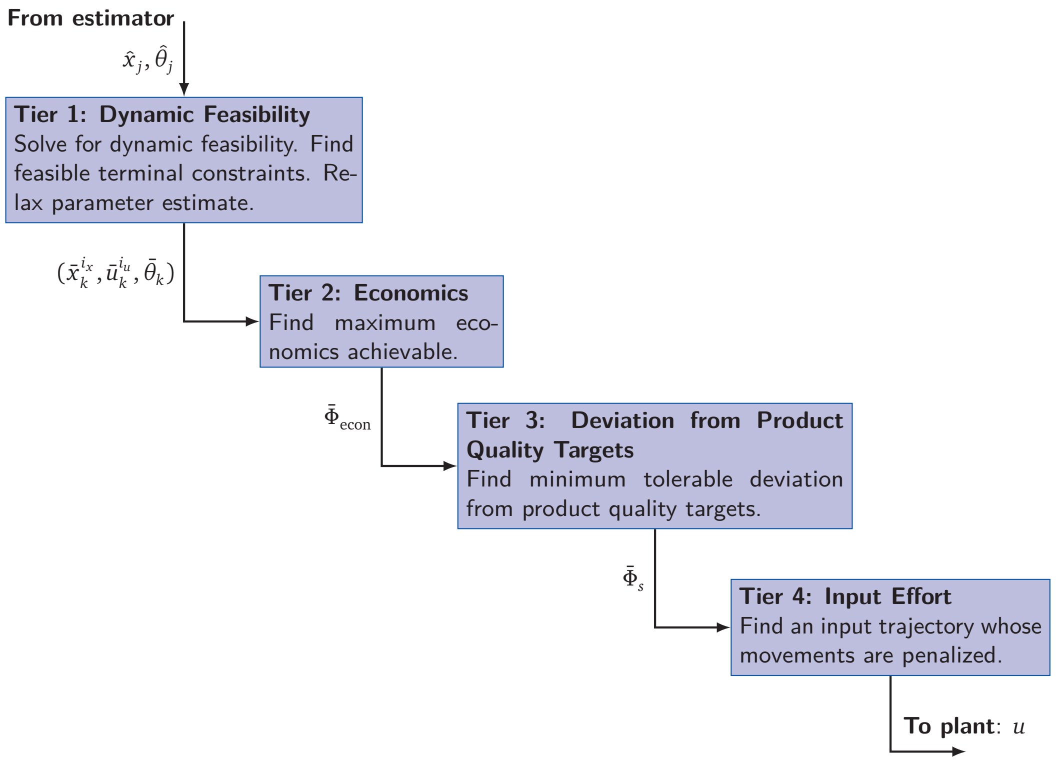

From estimator
$\hat{x}_j, \hat{\theta}_j$
Tier 1: Dynamic Feasibility
Solve for dynamic feasibility. Find feasible terminal constraints. Relax parameter estimate.
$(\bar{x}_k^{i_x}, \bar{u}_k^{i_u}, \bar{\theta}_k)$
Tier 2: Economics
Find maximum economics achievable.
$\bar{\Phi}_{\text{econ}}$
Tier 3: Deviation from Product Quality Targets
Find minimum tolerable deviation from product quality targets.
$\bar{\Phi}_s$
Tier 4: Input Effort
Find an input trajectory whose movements are penalized.
To plant: u

Figure 5.3: Multi-tiered optimization strategy.

therefore this is an unacceptable outcome. We prevent this from happening by requiring the controller to move the system back to its original operating point (x_{ss}, u_{ss}) in the optimization problem. The following endpoint constraints are enforced at the last n_f discrete time points in the prediction horizon:

$$\bar{x}_{\text{ss}}^{i_x}(1-\epsilon) \le x_k^{i_x} \le \bar{x}_{\text{ss}}^{i_x}(1+\epsilon), \quad k \in \mathcal{K}_f, i_x \in \mathcal{I}_{\text{ss}}^x \tag{5.54}$$

$$\bar{u}_{\text{ss}}^{i_u}(1-\epsilon) \le u_k^{i_u} \le \bar{u}_{\text{ss}}^{i_u}(1+\epsilon), \quad k \in \mathcal{K}_f, i_u \in \mathcal{I}_{\text{ss}}^u \tag{5.55}$$

where $\mathcal{K}_f = \{N - n_f, \ldots, N\}$, $N =$ the horizon length of the controller, $x_k^{i_x}$ and $u_k^{i_u}$ are elements of the state x and input u vectors at time k, indexed by i_x, i_u respectively. The index sets are defined as $\mathcal{I}_{\text{ss}}^x = \{i | x_i \in \mathcal{X}_{\text{ss}}\}$ and $\mathcal{I}_{\text{ss}}^u = \{i | u_i \in \mathcal{U}_{\text{ss}}\}$, and $\mathcal{X}_{\text{ss}}, \mathcal{U}_{\text{ss}}$ are the sets of independent state and inputs variables necessary to specify a steady-state. $\epsilon \ge 0$ is a deviation tolerance percentage parameter, specified as a fraction. $\bar{x}_{\text{ss}}^{i_x}, \bar{u}_{\text{ss}}^{i_u}$ are (constant) pre-shutdown steady-state values.

The restoration constraints laid out hitherto (Eqns. 5.54 – 5.55) are usually feasible under mild plant-model mismatch conditions, where there are enough degrees-of-freedom left to adjust for the effects of mismatch. However, in a highly constrained system, they are liable become infeasible in the optimization problem, particularly in the presence of large disturbances or significant mismatch situations. One recourse is to find a new feasible operating point in the neighborhood of the original steady-state point that is as close as possible to the original, which is the goal of Tier 1. In Tier 2, we attempt to find the economic upper-limit for the system. In Tier 3, we compute the maximum tolerable deviation from product quality specifications. Finally, in Tier 4, we compute a input profile that is capable of achieving the desired control with the least effort.

We remark that from a computational standpoint, a warm-startable feasible-path optimization solver often gives good performance on these types of problems because the solution of each tier constitutes a good initial guess for the next tier.

Fig. 5.3 depicts the multi-tiered optimization strategy. Separate optimization problems are solved one after the other, with the preceding tiers providing information to the latter tiers. We wish to emphasize that unlike the separation of concerns in the control hierarchy, this strategy is essentially a technique to arrive at a single set of inputs u that satisfy (as best as they can) the objectives in each tier, in the order of their priorities. Each of these optimization tiers is considered in detail below.

Preparation

In the closed-loop algorithm, the original endpoint constraints (specified in eqns. 5.54 – 5.55) from the problem are dropped, and the time-invariant parameter vector θ is converted into a time-varying algebraic vector θ_k.

Tier 1: Solve for Dynamic Feasibility

For feasibility reasons, at each MPC iteration all inequality constraints solely involving state variables x_j at the current time j (and only at the current time j) are dropped. The value of x_j is fixed to the estimated value $\hat{x}_j$, a value that may violate the constraints. See Section 3.3.1 for details.

At MPC iterate j, we solve the original DAE optimization problem with this objective function:

$$\min \left[\underbrace{\sum_{k\in\mathscr{K}_c, i_x\in\mathscr{I}_{\text{ss}}^x} (x_k^{i_x} - \bar{x}_{\text{ss}}^{i_x})^{\text{T}} A^x (x_k^{i_x} - \bar{x}_{\text{ss}}) + \sum_{k\in\mathscr{K}_f, i_u\in\mathscr{I}_{\text{ss}}^u} (u_k^{i_u} - \bar{u}_{\text{ss}}^{i_u})^{\text{T}} A^u (u_k^{i_u} - \bar{u}_{\text{ss}})}_{\text{① endpoint deviation}} \right.$$

$$\left. + \underbrace{\sum_{k=j+1}^{N} (\theta_k - \hat{\theta}_j)^{\mathrm{T}} A^{\theta} (\theta_k - \hat{\theta}_j)}_{\text{② parameter relaxation}} \right] \tag{5.56}$$

$$\text{s.t. } \theta_j = \hat{\theta}_j \tag{5.57}$$

where A^x, A^u and A^θ are user-tunable weighting matrices (often set to the identity matrix I), $\hat{\theta}_j$ is the parameter estimate vector for iteration j, $x_k^{i_x}, u_k^{i_u}$ are the subset of independent states and inputs that define a steady-state, and $\bar{x}_{\mathrm{ss}}^{i_x}, \bar{u}_{\mathrm{ss}}^{i_u}$ are their (constant-valued) targets — which are set to the nominal pre-shutdown steady-state values, $\mathcal{K}_c = \{\max(N - n_f, j+1), \ldots, N\}$ and $\mathcal{K}_f = \{N - n_f, \ldots, N\}$, where n_f is the number of periods before the horizon endpoint N that marks the post-restoration phase.

The intent of the two main parts of the objective function are as follows:

① The *endpoint deviation* part of the objective aims to find a feasible independent tuple $(x_k^{i_x}, u_k^{i_u})$ to be used as restoration targets that are as close as possible to the pre-shutdown steady-state operating point $(\bar{x}_{\mathrm{ss}}, \bar{u}_{\mathrm{ss}})$. As mentioned earlier, large disturbances or plant-model mismatch may render the original endpoint infeasible, therefore this procedure allows us to recompute a feasible endpoint at every MPC iteration. It is worth noting that the endpoint deviation formulation represents a convenient means for computing (and enforcing, as is done in the next tier) terminal constraints in general MPC problems.

Also, in previous chapters, we have assumed that the controller horizon length (N) is fixed to a value that is sufficiently large to permit a feasible restoration. In practice, it is difficult to know *a priori* the exact restoration time required for a given partial shutdown, therefore one needs choose a conservatively large N that will allow for a restoration to be carried out within the confines of the controller's mandate. If however the length of the chosen horizon is too short, that is, that no feasible restoration path can be found within the given horizon length, one may elect to lengthen the horizon at the expense of increasing the size of the optimization problem.

This dynamic feasibility tier provides an alternative (albeit perhaps suboptimal) approach that does not entail an increase in the problem size. The endpoint deviation term causes the controller to continually drive the system toward the desired endpoint $(\bar{x}_{\mathrm{ss}}, \bar{u}_{\mathrm{ss}})$ if it is feasible, or to the closest feasible operating point if it is not, thus circumventing difficulties caused infeasibilities due to a horizon length that is too short.

Finally, we note that if $(\bar{x}_{\text{ss}}, \bar{u}_{\text{ss}})$ are achievable at the endpoint, this part of the objective function goes to zero and the problem reduces to the standard case of affairs where endpoint constraints are strictly enforced.

② The *parameter relaxation* term in the objective is used to aid feasibility. When performing parameter estimation using one-step-ahead estimator like a Kalman filter, it is sometimes possible for the estimator to return a set of parameter values that are only feasible for the next time period, but not for the entire horizon[4]. This is because the Kalman family of filters typically only has a one-step-ahead view of the system. In our experience, a constant parameter held throughout the horizon can sometimes cause state variables to exceed bounds, especially when the estimator is tuned aggressively. Estimated disturbances and biases, when held constant throughout the prediction horizon, are particularly prone to causing infeasibilities.

In our approach, we relax the normally time-invariant parameter vector θ into a time-varying algebraic variable θ_k. At the current time j, θ_k is set to the estimated value given by the parameter estimator, $\hat{\theta}_j$ (eqn. 5.57); this quantity is always feasible because the estimator has a one-step-ahead view of the system, and it uses the same model as the controller. From period $j+1$ onwards however, we wish to permit the value of θ to deviate from the estimate at time j, but we penalize this deviation. This represents a kind of "soft constraint" on the parameter estimate, and can be seen as a means for remaining feasible in the optimization problem.

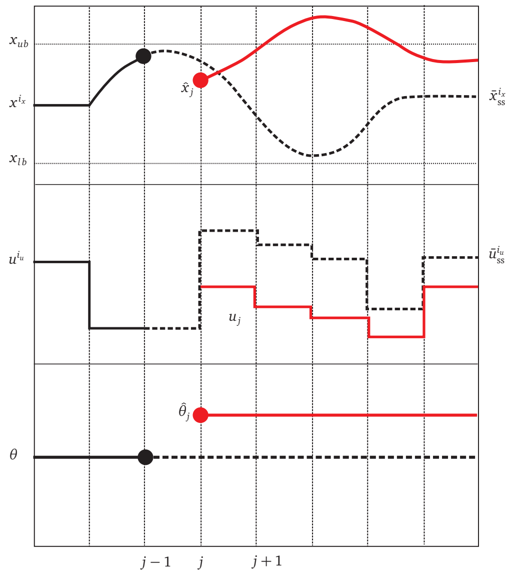
Note that if the parameter estimate $\hat{\theta}_j$ is indeed feasible for the entire prediction horizon, the parameter relaxation term will be equal to zero at the optimum, and the problem reduces to the ordinary case of affairs in which the parameter relaxation term does not exist.

The result of this optimization tier is a set of targets that are feasible for the current problem in iterate j. These targets are recorded as follows:

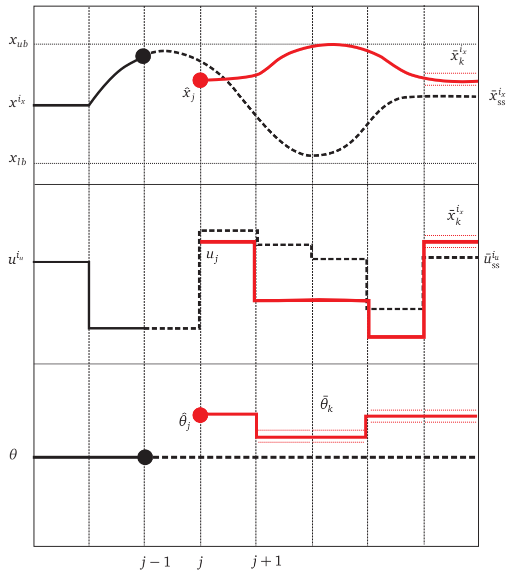
$$\bar{x}_k^{i_x} := x_k^{*,i_x}, \qquad k \in \mathscr{K}_c, i_x \in \mathscr{I}_{\text{ss}}^x \tag{5.58}$$
$$\bar{u}_k^{i_u} := u_k^{*,i_u}, \qquad k \in \mathscr{K}_f, i_x \in \mathscr{I}_{\text{ss}}^x \tag{5.59}$$
$$\bar{\theta}_k := \theta_k^*, \qquad k = j+1, \ldots, N \tag{5.60}$$

[4]There exists a class of estimators, namely the Moving Horizon Estimator (MHE) family, that is insusceptible to this. MHEs solve a constrained optimization problem typically containing a multi-step prediction horizon, therefore they are cognizant of constraints throughout the horizon. However the computational cost of solving an MHE problem may be significant.

where $(\bar{x}_k^{i_x}, \bar{u}_k^{i_u}, \bar{\theta}_k)$ are (constant-valued) targets to be fixed in the next tier. The * superscript attached to each variable denotes its value at the optimum.



(a) No dynamic feasibility tier.



(b) With dynamic feasibility tier.

Figure 5.4: Under plant-model mismatch conditions, the dynamic feasibility tier is used to recompute the endpoint targets and relax the parameters received from an estimator. The figures show the trajectories computed in the controller model.

To illustrate the utility of the dynamic feasibility tier, let us suppose we have a hypothetical controller affected by plant-model mismatch. In Fig. 5.4(a) (where no dynamic feasibility tier is applied), the dashed lines represent the predicted trajectories at time $j-1$. At time j, a new state estimate $\hat{x}_j$ and a parameter estimate $\hat{\theta}_j$ (which has not converged to its true value) is received. The estimate $\hat{\theta}_j$ is held constant through the prediction horizon. Note that these new values for the states and parameters do not match the predictions made at time $j-1$ due to plant-model mismatch. The controller proceeds to recalculate a set of u_j with $\hat{x}_j$ and $\hat{\theta}_j$ and finds that the state $x_k^{i_x}$ is unable to satisfy the the terminal constraint that forces it to go to $\bar{x}_{ss}^{i_x}$ at the endpoint. Furthermore, holding the current unconverged value of the parameter $\hat{\theta}_j$ constant through the horizon causes $x_k^{i_x}$ to exceed its upper-bound, x_{ub}. This is because in our case, $\hat{\theta}_j$ is obtained from a state-estimator that only has a one-step-ahead view of the model, and hence returns estimates that are only guaranteed to be feasible between the current and next sample times. The controller optimization problem is thus infeasible and the controller cannot proceed.

Fig. 5.4(b) shows the results obtained with a dynamic feasibility tier. Instead of holding constant an unconverged parameter estimate $\hat{\theta}_j$, the parameter vector is converted into a time-varying variable. The terminal constraints that force $x_k^{i_x} = \bar{x}_{\text{ss}}^{i_x}$ and $u_k^{i_u} = \bar{u}_{\text{ss}}^{i_u}$ at the endpoints are relaxed. The dynamic feasibility tier then solves a feasibility problem in order to compute new endpoint targets $(\bar{x}_k^{i_x}, \bar{u}_k^{i_u})$ as well as relaxed parameter target trajectory $\bar{\theta}_k$ that are proximate to the original target values. Within the solution space of these feasible targets, the controller is then able to proceed to find a u_j trajectory that optimizes lower-tier objectives.

Tier 2: Maximize Economics

In the economics optimization tier, we enforce new restoration constraints using the target tuple obtained in the previous tier. We solve the following economic optimality problem:

$$\max \Phi_{\text{econ}}(x, z, u) \tag{5.61}$$

$$\text{s.t. } \bar{x}_k^{i_x}(1-\epsilon) \le x_k^{i_x} \le \bar{x}_k^{i_x}(1+\epsilon), \qquad k \in \mathscr{K}_c, i_x \in \mathscr{I}_{\text{ss}}^x \tag{5.62}$$

$$\bar{u}_k^{i_u}(1-\epsilon) \le u_k^{i_u} \le \bar{u}_k^{i_u}(1+\epsilon), \qquad k \in \mathscr{K}_f, i_u \in \mathscr{I}_{\text{ss}}^u \tag{5.63}$$

$$\bar{\theta}_k(1-\epsilon) \le \theta_k \le \bar{\theta}_k(1+\epsilon), \qquad k = j+1, \ldots, N \tag{5.64}$$

$$\theta_j = \hat{\theta}_j \tag{5.65}$$

where $\epsilon \ge 0$ is a numerical tolerance tuning parameter. The set of counters $\mathscr{K}_c$ ensures that as the controller enters the post-restoration period, the terminal constraints are not enforced at the current and past $x_k^{i_x}$ (whose values are obtained from the state estimator, and fixed in problem instance j), but only on their future values. Failure to do this may result in an infeasible optimization problem.

Notice that the endpoint targets and parameter deviations are enforced as constraints in this tier (relaxed by ϵ). From this tier, we obtain a (constant) value for target economics, $\bar{\Phi}_{\text{econ}} := \Phi^*_{\text{econ}}(x^*, z^*, u^*)$, which is enforced in the next tier.

We point out that at this tier, the product quality targets are allowed to vary within their admissible bounds. In the cases we are considering, the quality targets are (intermediate) operational targets that are non-critical to the final product quality. In other applications, these quality targets may be essential, in which case Tiers 1 and 2 may be swapped and their trade-offs appropriately reassigned.

Tier 3: Minimize deviation from Product Quality Targets

In this tier, we perform an optimization to ensure that product quality targets are as close to their nominal values as possible.

$$\min \Phi_s = \sum_{k=j}^{N} (s_k - \bar{s})^{\mathrm{T}} (s_k - \bar{s}) \tag{5.66}$$

$$\text{s.t. } \Phi_{\text{econ}}(x, z, u) \geq (1 - \varepsilon_e) \bar{\Phi}_{\text{econ}} \tag{5.67}$$

$$\bar{x}_k^{i_x}(1-\epsilon) \leq x_k^{i_x} \leq \bar{x}_k^{i_x}(1+\epsilon), \quad k \in \mathscr{K}_c, i_x \in \mathscr{I}_{\text{ss}}^x \tag{5.68}$$

$$\bar{u}_k^{i_u}(1-\epsilon) \leq u_k^{i_u} \leq \bar{u}_k^{i_u}(1+\epsilon), \quad k \in \mathscr{K}_f, i_u \in \mathscr{I}_{\text{ss}}^u \tag{5.69}$$

$$\bar{\theta}_k(1-\epsilon) \leq \theta_k \leq \bar{\theta}_k(1+\epsilon), \quad k = j+1, \ldots, N \tag{5.70}$$

$$\theta_j = \hat{\theta}_j \tag{5.71}$$

where s_k is a vector of product quality variables, $\bar{s}$ are their targets, ε_e is trade-off parameter specifying the allowable fraction of the optimal economics to trade-off.

In this tier, the economics is fixed to the optimal value obtained in the previous tier. The relaxation parameter ε allows the user to specific the exact percentage of the dollar amount to trade off in order to the achieve the objectives in the current and subsequent tiers. The squared deviation of the product quality targets at the optimum ($\bar{\Phi}_s := \Phi_s^*$) is recorded and enforced in the next tier.

Tier 4: Minimize input effort

In the final tier, and the input movement is minimized. This represents a type of regularization on the inputs.

$$\min \sum_{k=j+1}^{N} (u_k - u_{k-1})^{\mathrm{T}} (u_k - u_{k-1}) \tag{5.72}$$

$$\text{s.t. } \sum_{k=j}^{N} (s_k - \bar{s})^{\mathrm{T}} (s_k - \bar{s}) \leq \bar{\Phi}_s (1 + \varepsilon_s) \tag{5.73}$$

$$\Phi_{\text{econ}}(x, z, u) \geq (1 - \varepsilon_e) \bar{\Phi}_{\text{econ}} \tag{5.74}$$

$$\bar{x}_k^{i_x}(1-\epsilon) \leq x_k^{i_x} \leq \bar{x}_k^{i_x}(1+\epsilon), \quad k \in \mathscr{K}_c, i_x \in \mathscr{I}_{\text{ss}}^x \tag{5.75}$$

$$\bar{u}_k^{i_u}(1-\epsilon) \leq u_k^{i_u} \leq \bar{u}_k^{i_u}(1+\epsilon), \quad k \in \mathscr{K}_f, i_u \in \mathscr{I}_{\text{ss}}^u \tag{5.76}$$

$$\bar{\theta}_k(1-\epsilon) \leq \theta_k \leq \bar{\theta}_k(1+\epsilon), \quad k = j+1, \ldots, N \tag{5.77}$$

$$\theta_j = \hat{\theta}_j \tag{5.78}$$

where ε_s is the fraction allowable trade-off for the deviation of product qualities from their targets. Fig. 5.5 shows the multi-tiered optimization existing within a closed-loop feedback framework.

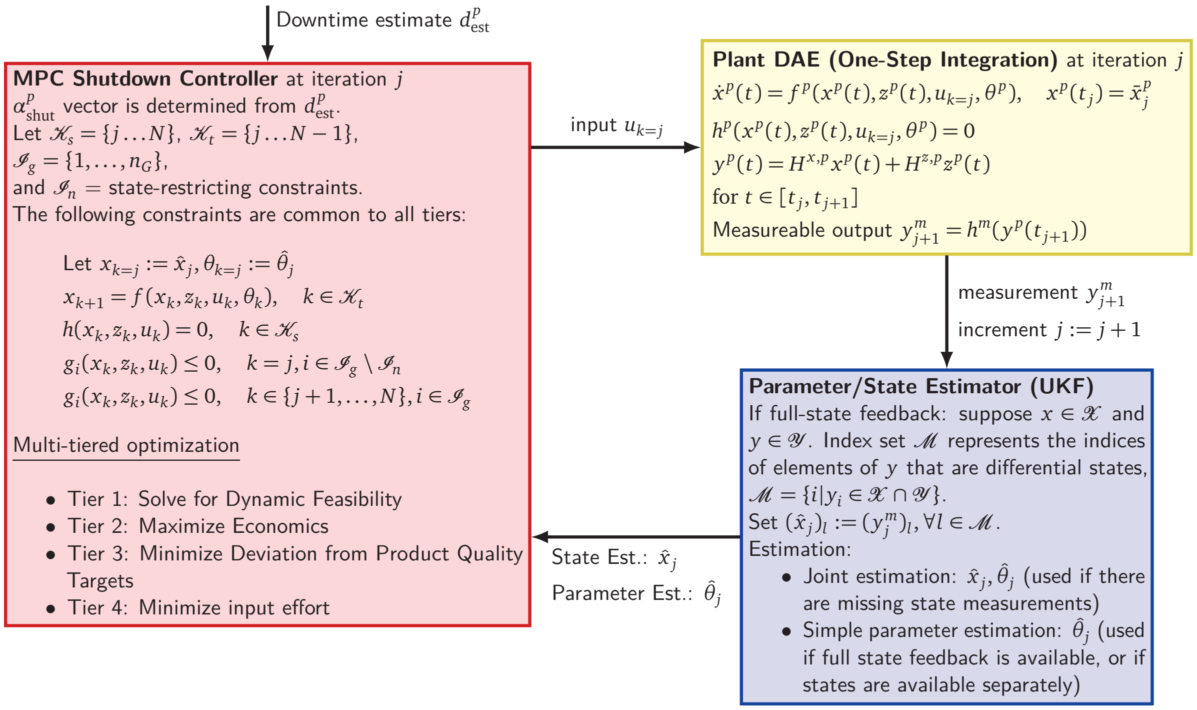

Downtime estimate d^p_{est}
MPC Shutdown Controller at iteration j
α^p_{shut} vector is determined from d^p_{est}.
Let $\mathcal{K}_s = \{j \dots N\}$, $\mathcal{K}_t = \{j \dots N-1\}$,
$\mathcal{I}_g = \{1, \dots, n_G\}$,
and $\mathcal{I}_n$ = state-restricting constraints.
The following constraints are common to all tiers:
Let $x_{k=j} := \hat{x}_j, \theta_{k=j} := \hat{\theta}_j$
$x_{k+1} = f(x_k, z_k, u_k, \theta_k), \quad k \in \mathcal{K}_t$
$h(x_k, z_k, u_k) = 0, \quad k \in \mathcal{K}_s$
$g_i(x_k, z_k, u_k) \le 0, \quad k = j, i \in \mathcal{I}_g \setminus \mathcal{I}_n$
$g_i(x_k, z_k, u_k) \le 0, \quad k \in \{j+1, \dots, N\}, i \in \mathcal{I}_g$
Multi-tiered optimization
• Tier 1: Solve for Dynamic Feasibility
• Tier 2: Maximize Economics
• Tier 3: Minimize Deviation from Product Quality Targets
• Tier 4: Minimize input effort
input $u_{k=j}$
Plant DAE (One-Step Integration) at iteration j
$\dot{x}^p(t) = f^p(x^p(t), z^p(t), u_{k=j}, \theta^p), \quad x^p(t_j) = \bar{x}^p_j$
$h^p(x^p(t), z^p(t), u_{k=j}, \theta^p) = 0$
$y^p(t) = H^{x,p} x^p(t) + H^{z,p} z^p(t)$
for $t \in [t_j, t_{j+1}]$
Measureable output $y^m_{j+1} = h^m(y^p(t_{j+1}))$
measurement y^m_{j+1}
increment $j := j+1$
Parameter/State Estimator (UKF)
If full-state feedback: suppose $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Index set $\mathcal{M}$ represents the indices of elements of y that are differential states, $\mathcal{M} = \{i | y_i \in \mathcal{X} \cap \mathcal{Y}\}$.
Set $(\hat{x}_j)_l := (y^m_j)_l, \forall l \in \mathcal{M}$.
Estimation:
• Joint estimation: $\hat{x}_j, \hat{\theta}_j$ (used if there are missing state measurements)
• Simple parameter estimation: $\hat{\theta}_j$ (used if full state feedback is available, or if states are available separately)
State Est.: $\hat{x}_j$
Parameter Est.: $\hat{\theta}_j$

Figure 5.5: Nonlinear MPC feedback loop, with multi-tiered controller, plant and parameter/state estimator

5.7 Case Studies

5.7.1 Preliminaries

In this section, we will present several cases to compare and contrast the various approaches to using state and parameter estimation to address plant-model mismatch. In order to compare the different approaches, we use a few metrics derived from the mean-square error (MSE). For any arbitrary time-varying variable or parameter b, the mean-squared error (and a related quantity, the root mean squared error, or RMSE) can be stated as follows:

$$\text{MSE}(b) = \frac{1}{N} \sum_{i=1}^{N} (b_k - \hat{b}_k)^2 \tag{5.79}$$

$$\text{RMSE}(b) = \sqrt{MSE(b)} \tag{5.80}$$

where $k = 1, \dots, N$ is a time counter, b_k is the true value of a variable (presumed to be known in the context of a simulation, though not in practice) and $\hat{b}_k$ is its estimate.

To compare results in between different case studies, we take the liberty of defining another metric to capture to overall error in a particular case: the sum of mean-squared-errors (SMSE). The SMSE is a composite metric that represents the summation of all the MSEs of either the states or the parameters in a particular run or simulation. In the definitions below, we distinguish between the MSE summations of differential state variables (SMSE_x), algebraic state variables (SMSE_z) and parameters (SMSE_θ):

$$\text{SMSE}_x = \sum_{i=1}^{n_x} \text{MSE}(x_i) \tag{5.81}$$

$$\text{SMSE}_z = \sum_{i=1}^{n_z} \text{MSE}(z_i) \tag{5.82}$$

$$\text{SMSE}_\theta = \sum_{i=1}^{n_\theta} \text{MSE}(\theta_i) \tag{5.83}$$

where x_i, z_i, θ_i are the ith elements in the set of state and parameter variable sets, respectively.

For the case studies, we will be considering two different plants: the full Kraft fiber line process, whose model equations can be found in Appendix A, and a simpler subset of the Kraft plant with only three units (the digester, blowtank and Hi-Q knotter) which we shall call the Mini-Kraft process.

5.7.2 The effect of tuning parameters

The UKF estimation algorithms rely on a few basic tuning parameters for their operation. Apart from the usual covariance matrix tunings found on any standard Kalman filter, the UKF also requires tunings that determine the spread and weight of the sigma points, (α, β, κ). Additionally, the UKF parameter estimator makes use of a forgetting factor λ_f. Kolås et al. [65] made the observation that a commonly used UKF tuning for tuple (α, β, κ) is (1,2,0). The parameter β and κ are usually based on whether the random variable being estimated is Gaussian and are ordinarily set to their default recommended values.

The parameter α however is proportional to the spread of the sigma points, and is a tuning parameter that influences the estimation in an interpretable way. Table 5.1 shows the effect

Noise %	λ_f	α	SMSE_θ
0.00	1.00	0.10	0.000464
0.00	1.00	0.50	0.000466
0.00	1.00	1.00	0.000473
1.00	1.00	0.10	0.033677
1.00	1.00	0.50	0.045771
1.00	1.00	1.00	0.063470
2.00	1.00	0.10	0.100438
2.00	1.00	0.50	0.341262
2.00	1.00	1.00	0.753859

Table 5.1: Effect of UKF tuning parameter α vs. noise and resulting MSE for simple parameter estimation of Mini-Kraft system. (Smaller SMSE_θ is better.)

of α on the overall parameter estimation error, SMSE_θ of a simple estimation performed on the Mini-Kraft model (shown in the next section). The cases with $\alpha \in \{0.10, 0.50, 1.00\}$ were repeated for three noise cases, that is, assuming 0%, 1%, 2% noise on the measurements. Interestingly, the results seem to indicate that a smaller spread produces smaller estimation errors in all the noise cases.

A larger spread tends to capture nonlinearity and variability better, but in this case, since the model variability was fairly low, a lower spread appears to be the better choice. In our case studies, we have elected to use $\alpha = 0.1$ across all cases to ensure a certain degree of comparability between them.

Noise %	λ_f	α	SMSE_θ
0.00	1.00	0.10	0.000464
0.00	0.99	0.10	0.000349
0.00	0.95	0.10	0.000143
1.00	1.00	0.10	0.033677
1.00	0.99	0.10	0.048992
1.00	0.95	0.10	0.109231
2.00	1.00	0.10	0.100438
2.00	0.99	0.10	0.230509
2.00	0.95	0.10	0.577144

Table 5.2: Effect of forgetting factor vs. noise % and resulting MSE for simple parameter estimation of Mini-Kraft system. (Smaller SMSE_θ is better.)

Table 5.2 shows the effect of the forgetting factors $\lambda_f \in \{1.00, 0.99, 0.95\}$ on the parameter estimation error SMSE_θ. The case $\lambda_f = 1.00$ represents the nominal case, in which there is no forgetting at all, because the asymptotic memory length $n_A \to \infty$. The results indicate that when the amount of measurement noise is 0%, the smaller forgetting factor is better. However, when the amount of measurement noise is 1% or 2%, a larger forgetting factor gives more favorable results.

This outcome can be understood in terms of the role λ_f plays in the estimation. The forgetting factor λ_f is often used to prioritize current information over the previous. In the parameter UKF, it enters in the covariance prediction step as a reciprocal, $P^-_{\theta,j} = \lambda_f^{-1} P_{\theta,j-1}$ (eqn. 5.40). A lower value of λ_f in this case not only corresponds to a shorter asymptotic memory length, it also increases the magnitude of the elements in the predicted covariance matrix. If we follow the algorithm's logic down to the Kalman gain computation step, we will see that this results in a boost in the Kalman gain. Therefore, by choosing a lower value of λ_f, we expect to see an improvement in tracking speed because the estimator is made more aggressive owing to the increased Kalman gain.

When there is very little excitation in the system, the elements of the error covariance matrix are almost 0. Therefore a low λ_f causes the Kalman filter to behave more aggressively (through the agency of the covariance matrix), and better tracking behavior is obtained. However, when there exists sufficient excitation in the system (from noise), a higher λ_f may give rise to better tracking, possibly by not amplifying the noise in the system.

5.7.3 Case studies on Mini-Kraft process

Description

The Mini-Kraft process (Fig. 5.6) is used to illustrate the concepts and techniques discussed on a smaller, recycle-free (and minimally interacting) model, which makes the results easier to interpret and explain. The Mini-Kraft process is a subset of the Kraft fiber line, with only the digester, blowtank and Hi-Q knotter units as its consituents. The reader is referred to Appendix A for the nomenclature and model equations.

A shutdown occurs in the Hi-Q knotter unit between t=1 – 4 hrs. Two parameters in the model are mismatched: the liquor-to-wood ratio in the digester (r^{DG}_{lw}, value in model = 3.6, value in plant = 3.06) and the pulp shrinkage factor term (β^{DG}_{as1}, value in model = 0, value in plant = -9.5). These discrepancies cause the model to predict a lower pulp concentration, higher dissolved solids concentration and higher flowrate exiting the digester than is the case in reality. The following are the differential states x, inputs u and measurements y in this problem:

$$x = [H^{BT}, x^{BT}_{S2P}, x^{BT}_{S2DS}]^{\mathrm{T}}$$

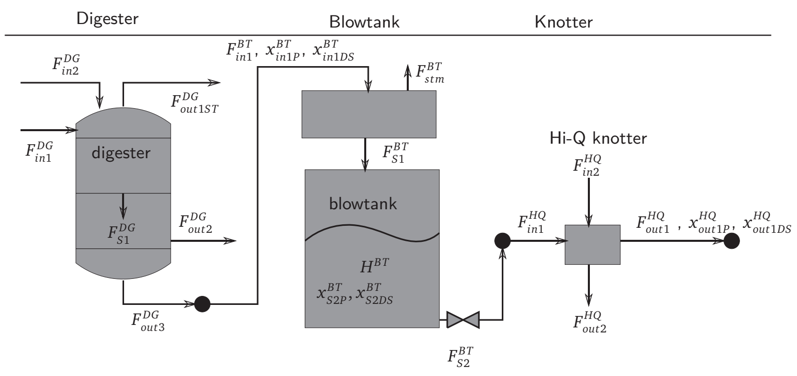


Figure 5.6: The Mini-Kraft process.

$$
\begin{aligned}
u &= [F_{in1}^{DG}, F_{S2}^{BT}]^{\mathrm{T}} \\
y^m &= [x^{\mathrm{T}}, x_{in1P}^{BT}, x_{in1DS}^{BT}, x_{out1P}^{HQ}, x_{out1DS}^{HQ}]^{\mathrm{T}}
\end{aligned}
$$

where y^m is the vector of measured variables in the plant. The subscripts P and DS refer to pulp and dissolved solid components, respectively.

Since this a subset of the full plant, our economic objective function necessarily only captures some of the streams with economic significance. As a result, one may not get a true economic picture of the full process, as the economics may be affected by downstream processes. Also, the prices available are for finished products, therefore profits for unfinished products must be discounted. In light of this, we have chosen to use a crude method for quantifying the plant subset's economics by calculating the profits and losses downstream, by taking the Hi-Q pulp output F_{out1P}^{HQ} as a basis.

The main source of downstream costs is the cost of the chemicals used to delignify the pulp. During nominal operation, at a production rate of $F_{out1P}^{HQ} = 4.64$ ton/hr of pulp, $F_{in1P}^{MX} = 4.41$ ton/hr of pulp enters the delignification unit. From the model, we know that 0.25 tons/hr of chemicals is used per ton/hr of pulp, and that the chemicals cost \$100/ton. Using dimensional analysis, we can compute the cost of chemicals used for Y ton/hr of pulp exiting the Hi-Q knotter (F_{out1P}^{HQ}):

$$
M_{\text{chem}} = \frac{4.41}{4.64}\left[\frac{\text{ton/hr } F_{in1P}^{MX}}{\text{ton/hr } F_{out1P}^{HQ}}\right] 0.25 \left[\frac{\text{ton/hr chem}}{\text{ton/hr } F_{in1P}^{MX}}\right] = 0.2376 \left[\frac{\text{ton/hr chem}}{\text{ton/hr } F_{out1P}^{HQ}}\right] \tag{5.84}
$$

$$
\Phi_{\text{chem}} = -100\left[\frac{\$}{\text{ton chem}}\right]\int_0^{t_f}\left(0.2376\left[\frac{\text{ton/hr chem}}{\text{ton/hr } F_{out1P}^{HQ}}\right]\cdot Y[\text{ton/hr } F_{out1P}^{HQ}]\right) dt \tag{5.85}
$$

where M_{chem} represents the nominal tons of chemicals used per ton of Hi-Q pulp and Φ_{chem} is the total cost of chemicals that would have been consumed downstream of our Mini-Kraft plant subset, given a Hi-Q pulp output flowrate of Y.

Similarly, we know that processed pulp (at the end of the Post-O_2 washer unit, F^{PW}_{out2P}, see Fig. 3.4(c)) has an economic value of \$725/ton. We also know that during nominal operation, 4.31 ton/hr of processed pulp is obtained at a production rate of 4.64 ton/hr of Hi-Q pulp.

$$M_{\text{pulp}} = \frac{4.31}{4.64}\left[\frac{\text{ton/hr } F^{PW}_{out2P}}{\text{ton/hr } F^{HQ}_{out1P}}\right] = 0.9289\left[\frac{\text{ton/hr } F^{PW}_{out2P}}{\text{ton/hr } F^{HQ}_{out1P}}\right] \tag{5.86}$$

$$\Phi_{\text{pulp}} = 725\left[\frac{\$}{\text{ton } F^{PW}_{out2P}}\right]\int_0^{t_f}\left(0.9289\left[\frac{\text{ton/hr } F^{PW}_{out2P}}{\text{ton/hr } F^{HQ}_{out1P}}\right]\cdot Y\left[\text{ton/hr } F^{HQ}_{out1P}\right]\right)dt \tag{5.87}$$

where M_{pulp} represents the nominal tons of processed pulp per ton of Hi-Q pulp and Φ_{pulp} is the total price of the processed pulp that would have been produced downstream of our Mini-Kraft plant subset, given a Hi-Q pulp output flowrate of Y. All other the other economically significant streams that exist in the Mini-Kraft subset are computed directly, and the total economics is obtained through a summation of all the cost terms. Through this exercise, we are able to approximate the economics of a plant subset.

We acknowledge that this method may not reflect the true absolute economics of this subset (indeed, quantifying the true economics of a plant subset is not an easy task in general). However, it captures the trade-offs in a way that is adequate for the purposes of our case studies, which is to compare the economics obtained in each of the cases relative to another.

For the following case studies, we elect to use a forgetting factor $\lambda_f = 0.995$ (which corresponds to a finite memory length of 200) as a compromise between noisy and noise-free cases, as well as a balance between having no forgetting at all and an overly aggressive forgetting factor. The UKF tuning parameters are set to $(\alpha, \beta, \kappa) = (0.1, 2, 0)$. For all simple parameter estimation cases, the Iterative Refinement step[5] is carried out and full-state feedback is assumed. Multi-tiered optimization is used in all cases. State and parameter constraints are imposed. No measurement noise was assumed in any of the cases. During the shutdown, a reduced estimator is used. We shall consider several case studies below in which the joint and simple estimation strategies are contrasted with the base case full-state feedback strategy.

Case 1a (Full-state feedback, no estimation): This represents the baseline case in which full-state measurements (that is, all x variables) are available, and no estimation is performed.

[5]Extensive testing reveals that turning on iterative refinement invariably leads to faster convergence to the correct parameters in almost all cases, at the expense of solving the estimation problem several times per feedback iteration.

Case 1b (Simple parameter estimation, with full state measurements, direct parameter estimation): in this scenario, We assume that we know the exact set of parameters that are mismatched between the model and the plant (i.e. r_{lw}^{DG} and β_{as1}^{DG}), and use a simple parameter estimator to directly estimate these parameters from the measurements.

Case 1c (Joint estimation): This case extends Case 1b by assuming that some state measurements are missing from y^m, namely x_{S2P}^{BT} and x_{S2DS}^{BT}. The estimator has to estimate both the full state vector and the mismatched parameters r_{lw}^{DG} and β_{as1}^{DG}.

Case 1d (Simple parameter estimation, with full state measurements, disturbance estimation): we consider here a case in which we assume we do not know the exact set of mismatched parameters. However, the gap between the sensors and the model predictions allow us make an educated guess as to which quantities need to be corrected. In this case, due to the divergence between model predictions and plant measurements in x_{in1P}^{BT} and x_{in1DS}^{BT} (pulp and dissolved solid concentrations at the inlet of the blowtank), we can deduce the outlet concentrations of those compounds from the digester are incorrectly captured in the model. Hence we can try to correct for this discrepancy by adding disturbance bias terms for the component flowrates of pulp and dissolved solids at the outlet of our digester model as follows:

$$F_{in1P}^{BT} = F_{out3P}^{DG}(1+\beta_{out3P}^{DG}) \tag{5.88}$$

$$F_{in1DS}^{BT} = F_{out3DS}^{DG}(1+\beta_{out3Ds}^{DG}) \tag{5.89}$$

where β_{out3P}^{DG} and β_{out3Ds}^{DG} are disturbance terms to be estimated.

Results

With respect to the figures that are to follow, in the case of the x and z state variables, the dashed lines represent controller model's prediction of the current step given information in past step, i.e. $\hat{x}_{j|j-1}$, $\hat{z}_{j|j-1}$. In the case of parameter variables, the dashed lines represent the parameter estimates $\hat{\theta}_{j|j-1}$. The solid lines in all cases represent the true values in the plant. Table 5.3 summarizes the above cases and their results. Note that an SMSE$_\theta$ is not defined for Case 1d because β_{out3P}^{DG} and β_{out3DS}^{DG} are disturbance terms with no “true” values.

- In Case 1a (Fig. 5.7), we see that during the shutdown in the Hi-Q knotter, the outlet of the blowtank (F_{S2}^{BT}) is shut off and the inventory in the blowtank continues to increase (H^{BT}). As soon as the shutdown is over, this inventory is discharged. Due to the persistent mismatch in r_{lw}^{DG} and β_{as1}^{DG}, the digester outlet concentrations is being predicted incorrectly, as witnessed by the gaps between the predicted and the actual in the x_{in1P}^{BT} and x_{in1DS}^{BT} trajectories. These also give rise to a small but perceptible gaps in the differential states

Case	**Mode**	**Estimated θ**	**Missing States**	**Obj ($)**	SMSE$_x$	SMSE$_z$	SMSE$_\theta$
1a	F	-	-	29,644	0.0106	21.383	-
1b	P, F	r_{lw}^{DG}, β_{as1}^{DG}	-	30,257	0.000182	0.455	5×10^{-6}
1c	J	r_{lw}^{DG}, β_{as1}^{DG}	x_{S2P}^{BT}, x_{S2DS}^{BT}	30,178	0.000236	0.455	2×10^{-5}
1d	P, F	β_{out3P}^{DG}, β_{out3DS}^{DG}	-	30,401	0.00930	0.455	-

[1]**Mode:** F = full state feedback, P = simple parameter estimation, J = joint estimation.

Table 5.3: Summary of Mini-Kraft case studies. Mismatch in r_{lw}^{DG}, β_{as1}^{DG} in all cases. (Smaller SMSEs are better.)

$(H^{BT}, x_{S2P}^{BT}, x_{S2DS}^{BT})$ as witnessed by their RMSEs. A standard full-state feedback scheme has no recourse mechanism for shrinking this gap.

- In Case 1b (Fig. 5.8), the causes of the abovementioned gaps are directly addressed by correcting the mismatched parameters r_{lw}^{DG} and β_{as1}^{DG} using a simple parameter estimation scheme. The trajectories for r_{lw}^{DG} and β_{as1}^{DG} show that their parameter estimates converging to the true values in short order. In turn, this closes the gap in the x_{in1P}^{BT} and x_{in1DS}^{BT} trajectories. If we were to compare this result to the results obtained in Case 1a, we would notice a significant qualitative difference in the control actions prescribed in both cases. Instead of immediately discharging the contents of the blowtank, the controller continues to accumulate material in the tank and elects to discharge it toward the end of the restoration period. The overall errors (SMSE$_x$, SMSE$_z$) are significantly smaller than that of Case 1a, and the objective value is marginally improved.

- Case 1c (Fig. 5.9) depicts a scenario in which two of the differential states, x_{S2P}^{BT} and x_{S2DS}^{BT} are unmeasured and have to estimated along with r_{lw}^{DG} and β_{as1}^{DG} through a joint estimation scheme. The trajectories for x_{S2P}^{BT} and x_{S2DS}^{BT} indicate that the computed state estimates track reality very well, and the estimates of the mismatched parameters converge to within a tolerance of their true values. It is within our expectation that the state estimation should perform well in this case because pulp and dissolved solid concentrations were measured before and after the blowtanks, and as well, the mismatched parameters which have a strong influence on the differential states were corrected by way of the parameter estimates.

- Finally, in Case 1d (Fig. 5.10), we demonstrate that disturbance terms are potentially good candidates for estimation when the true source of mismatch in a plant is unknown (which is often the case). Because the mismatched parameters r_{lw}^{DG} and β_{as1}^{DG} in the digester directly affect the pulp and dissolved solid concentrations, we can compensate for their effects by adjusting the component flows of the aforementioned compounds in the digester outlet. The trajectories for r_{lw}^{DG} and β_{as1}^{DG} show a quick convergence to the correct values with only a minimal disturbance-based adjustment, as seen in β_{out3P}^{DG} and β_{out3DS}^{DG}.

Referring to Table 5.3, we see evidence that by performing state/parameter estimation in Cases 1b, 1c and 1d, the model prediction errors are moderated and improvements in the objective are observed vis-à-vis the base Case 1a. Comparable performance was obtained in Cases 1b, 1c and 1d with respect to each other. The qualitative behavior of the control trajectories in Cases 1b, 1c and 1d (where the model had a corrected notion of the plant) were similar to each other, but differed visibly from those in Case 1a (where the model had an incorrect notion of the plant). This provides a strong motivation for the (judicious) use of an estimator in plant-model mismatch situations.

5.7.4 Case studies on full Kraft fiber line process

Description

In this section, we will demonstrate and compare applications of the simple and joint estimation strategies on the full Kraft fiber line process. The reader is referred to Appendix A for the nomenclature, topology diagrams (Figs. A.1, A.4, A.5), and model equations. These cases are similar to the previous Mini-Kraft in that a shutdown occurs in the Hi-Q knotter unit. However, this larger model contains several recycle streams, more buffer tanks and more processing units.

Two parameters in the model are mismatched in Cases 2a-2d below: the liquor-to-wood ratio in the digester (r_{lw}^{DG}, value in model = 3.6, value in plant = 3.06) and the pulp shrinkage factor term (β_{as1}^{DG}, value in model = 0, value in plant = −9.5). The shutdown occurs between t=0.25 – 5.25 hrs. In Case 2e, an additional parameter, namely the displacement ratio bias in Washer 1 is mismatched (β_{DR}^{WC1}, value in model = 0, value in plant = 0.45). The shutdown in this case occurs between t=0.25 – 2.25 hrs.

The following are the differential states x, inputs u and measurements y in this problem:

$$x = [H^{BT}, x_{S2P}^{BT}, x_{S2DS}^{BT}, H_{st}^{WC1}, x_{DS3}^{WC1}, H_{st}^{WC2}, x_{DS3}^{WC2}, H_{st}^{WC3}, x_{DS3}^{WC3}, H^{ST}, x_{out1P}^{ST}, x_{out1DS}^{ST}]^{\mathrm{T}}$$
$$u = [F_{in1}^{DG}, F_{S2}^{BT}, F_{3}^{WC1}, F_{R}^{WC1}, F_{2}^{WC1}, F_{R}^{WC2}, F_{3}^{WC3}, F_{R}^{WC3}, F_{2}^{WC3}, F_{out1}^{ST}]^{\mathrm{T}}$$
$$y^{m} = [x^{\mathrm{T}}, x_{in1P}^{BT}, x_{in1DS}^{BT}, x_{out1P}^{HQ}, x_{out1DS}^{HQ}]^{\mathrm{T}}$$

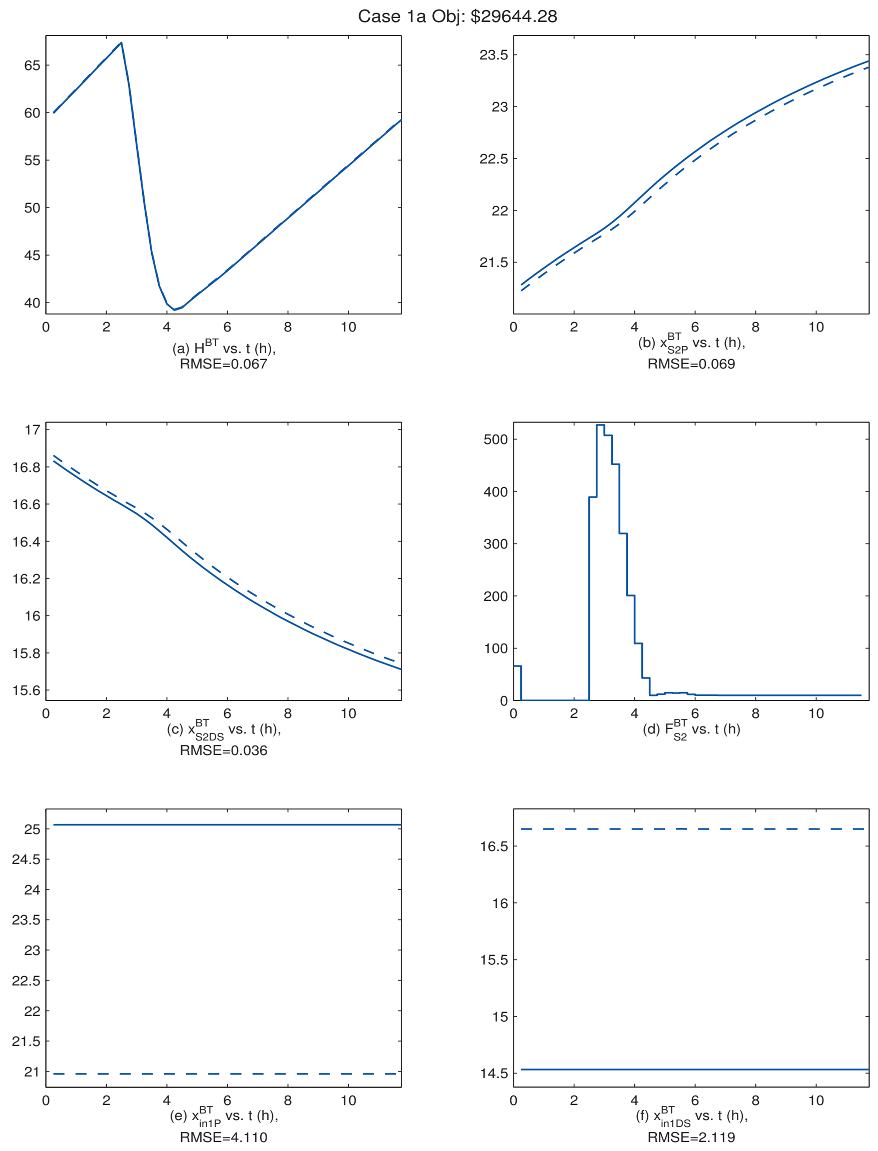


Figure 5.7: Case 1a: Mini-kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using full-state feedback. **Legend**: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

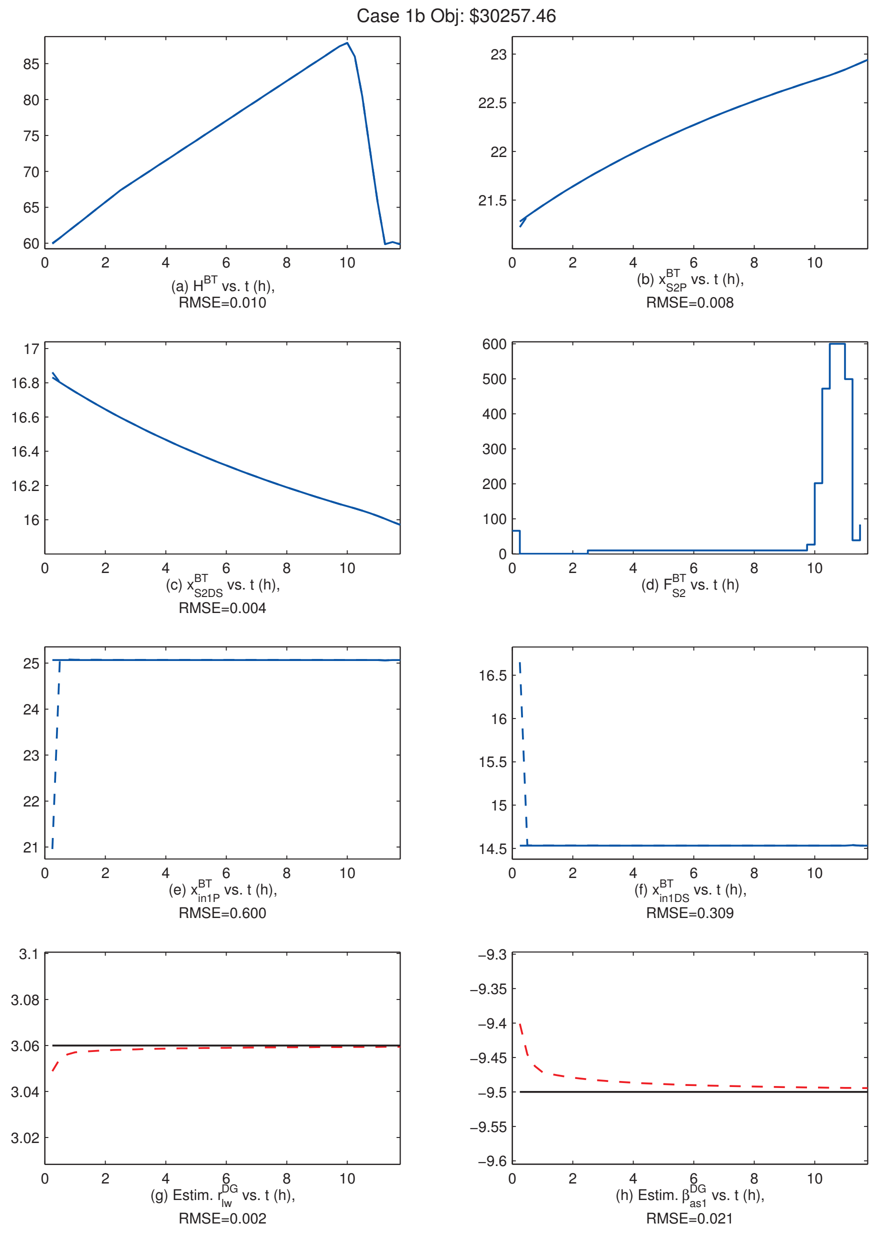


Figure 5.8: Case 1b: Mini-kraft model, with mismatch in (r_{lw}^{DG}, β_{as1}^{DG}), and using simple parameter estimation to estimate parameters (r_{lw}^{DG}, β_{as1}^{DG}). States obtained through full-state feedback. **Legend**: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

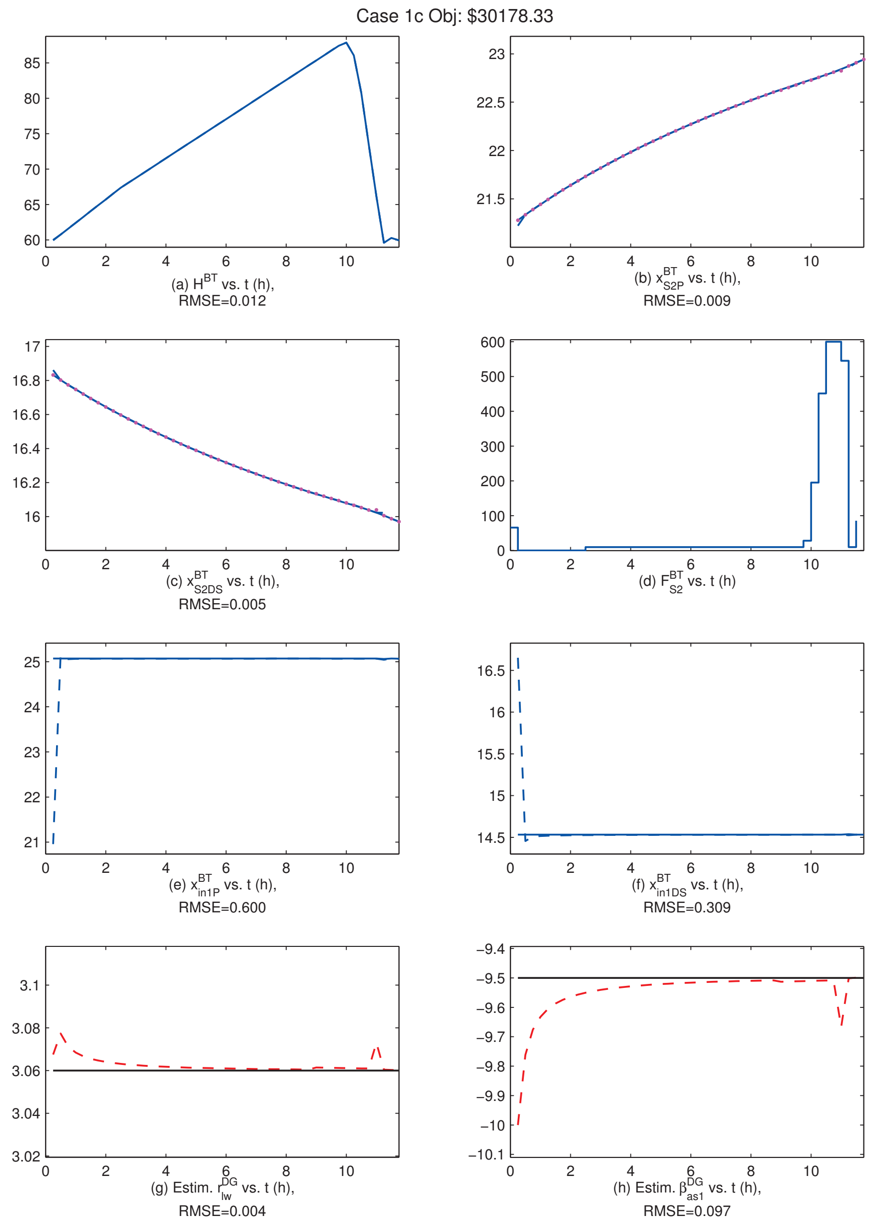


Figure 5.9: Case 1c: Mini-kraft model, with mismatch in (r_{lw}^{DG}, β_{as1}^{DG}), and using joint state and parameter estimation to estimate mismatched parameters (r_{lw}^{DG}, β_{as1}^{DG}) and missing differential states (x_{S2P}^{BT}, x_{S2DS}).
Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories, magenta dots = the estimates of missing state measurements.

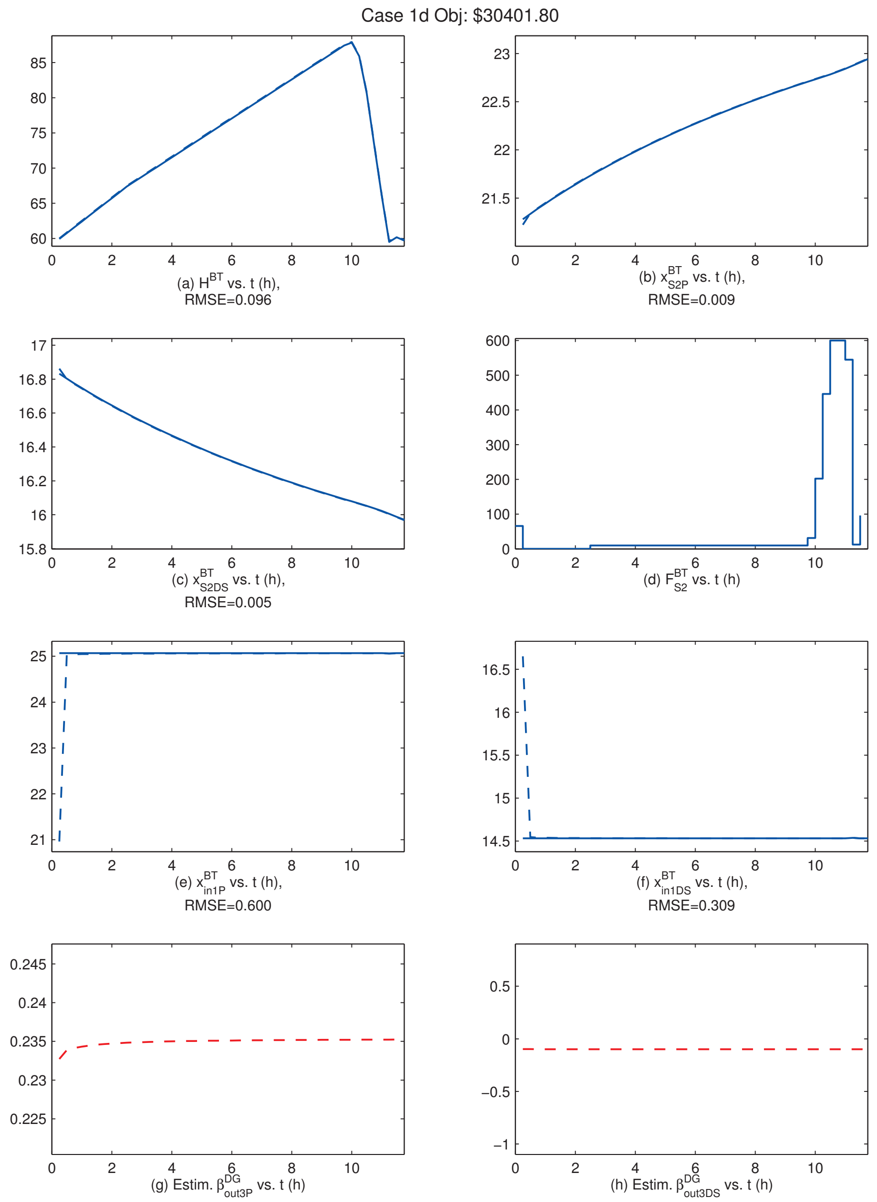


Figure 5.10: Case 1d: Mini-kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using simple parameter estimation to estimate algebraic biases (β^{DG}_{out3P}, β^{DG}_{out3DS}). States obtained through full-state feedback.

Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

where y^m is the vector of measured variables in the plant.

Product quality targets are also considered in this model. The vectors of quality variables s and their targets $\bar{s}$ are:

$$\begin{aligned} s &= [C_M^{SR}, C_M^{OF}, C_M^{PW}]^{\mathrm{T}} \\ \bar{s} &= [4.5, 30, 6.5]^{\mathrm{T}} \end{aligned}$$

where C_M^{SR}, C_M^{OF}, and C_M^{PW} are the outlet consistencies (in percentages) of the screeners, O_2 feedpress and post-O_2 washers respectively.

The UKF tuning parameters are set at $(\alpha, \beta, \kappa) = (0.1, 2, 0)$ and $\lambda_f = 0.995$. For all the simple parameter estimation cases, the Iterative Refinement step is carried out and full-state feedback is assumed. Multi-tiered optimization is used in all cases. State and parameter constraints are imposed. During the shutdown, a reduced estimator is used. Several case studies are run in which the joint and simple estimation strategies are contrasted with the full-state feedback strategy.

Case 2a (Full-state feedback, no estimation): The controller is run with full-state (all x variables) measurements are available, and no estimation is performed.

Case 2b (Simple parameter estimation, with full state measurements, direct parameter estimation): we assume that we know the exact set of mismatched parameters (i.e. r_{lw}^{DG} and β_{as1}^{DG}), and estimate them directly using a simple parameter estimator.

Case 2c (Joint estimation, missing state measurements): We extend Case 2b here by dropping measurements for the states x_{S2P}^{BT} and x_{S2DS}^{BT}, thus entailing the use of the joint estimation to perform both state estimation and parameter estimation of r_{lw}^{DG} and β_{as1}^{DG}.

Case 2d (Simple parameter estimation, with full state measurements, disturbance estimation): in this scenario, the exact set of mismatched parameters is assumed to be unknown. Like in Case 1d, we correct for this by adding disturbance bias terms for the component flowrates of pulp and dissolved solids at the outlet of our digester model as follows:

$$F_{in1P}^{BT} = F_{out3P}^{DG}(1 + \beta_{out3P}^{DG}) \tag{5.90}$$

$$F_{in1DS}^{BT} = F_{out3DS}^{DG}(1 + \beta_{out3Ds}^{DG}) \tag{5.91}$$

where β_{out3P}^{DG} and β_{out3Ds}^{DG} are disturbance terms to be estimated.

Case 2e (noise, joint estimation, missing states, additional mismatch, state disturbances) represents a distinct case (from all the other cases) in which elements of all the above cases are brought together, and an additional parameter mismatch in β_{DR}^{WC1} (a parameter involved in the calculation of Displacement Ratio (DR) in washer 1) is considered. The measurements are corrupted with Gaussian white noise that is 1% of the measurement values. Because β_{DR}^{WC1} affects the outlet dissolved solid concentration in the washers (and the washers are intricately linked in a counter-current configuration, we extend our measurement vector by measuring the dissolved solid concentrating out of washers 1, 2 and 3—namely x_{DS1}^{WC1}, x_{DS1}^{WC2}, and x_{DS1}^{WC3}—by assuming the appropriate sensors are added to the F_1 stream of all washer units. We assume that we do not know that there is a mismatch in β_{DR}^{WC1}; however, we assume that the operator has managed to deduce from model prediction errors in the dissolved solid concentrations of the washers (x_{DS1}^{WC1}, x_{DS1}^{WC2}, and x_{DS1}^{WC3}) that there is a mismatch in model related to dissolved solids. To compensate for that, state disturbances on dissolved solid concentrations in all washer seal tanks (β_{xDS3}^{WC1}, β_{xDS3}^{WC2}, β_{xDS3}^{WC3}) are estimated. These disturbances enter the differential equations for computing the washer dissolved solid concentrations (refer to eqn. A.86 in Appendix A) in the following way:

$$\frac{dx_{DS3}^{WC1}}{dt} = \frac{100F_D^{WC1}}{V_{st}^{WC1}\cdot\rho\cdot H_{st}^{WC1}}\cdot(x_{DSD}^{WC1} - x_{DS3}^{WC1}) + \beta_{xDS3}^{WC1}, \qquad x_{DS3}^{WC1}(0) = x_{DSD}^{WC1}(0) \tag{5.92}$$

$$\frac{dx_{DS3}^{WC2}}{dt} = \frac{100F_D^{WC2}}{V_{st}^{WC2}\cdot\rho\cdot H_{st}^{WC2}}\cdot(x_{DSD}^{WC2} - x_{DS3}^{WC2}) + \beta_{xDS3}^{WC2}, \qquad x_{DS3}^{WC2}(0) = x_{DSD}^{WC2}(0) \tag{5.93}$$

$$\frac{dx_{DS3}^{WC3}}{dt} = \frac{100F_D^{WC3}}{V_{st}^{WC3}\cdot\rho\cdot H_{st}^{WC3}}\cdot(x_{DSD}^{WC3} - x_{DS3}^{WC3}) + \beta_{xDS3}^{WC3}, \qquad x_{DS3}^{WC3}(0) = x_{DSD}^{WC3}(0) \tag{5.94}$$

Also, measurements for x_{S2P}^{BT} and x_{S2DS}^{BT} are assumed to be unavailable and must hence be estimated. A joint estimation is used to perform both state and parameter estimation.

Results

Table 5.4 summarizes the above cases on the full Kraft fiber line model and their results. Note that an SMSE$_\theta$ is not defined for Case 2d because β_{out3P}^{DG} and β_{out3DS}^{DG} are disturbance terms with no "true" values. In Case 2e, β_{xDS3}^{WC1}, β_{xDS3}^{WC2}, and β_{xDS3}^{WC3} are disturbance terms but r_{lw}^{DG} and β_{as1}^{DG} are actual uncertain parameters, hence the SMSE$_\theta$ is computed based only on the latter two actual parameters.

- Case 2a (Fig. 5.11) is the vanilla full-state feedback case. The shutdown in the Hi-Q knotter unit causes all the washers to shut down as well. The washers are down, but the seal tank 1 remains functional to provide liquor recycle to the blowtank through stream F_3^{WC1} (Fig. 5.11f). Our multi-tiered optimization formulation attempts to keep the product

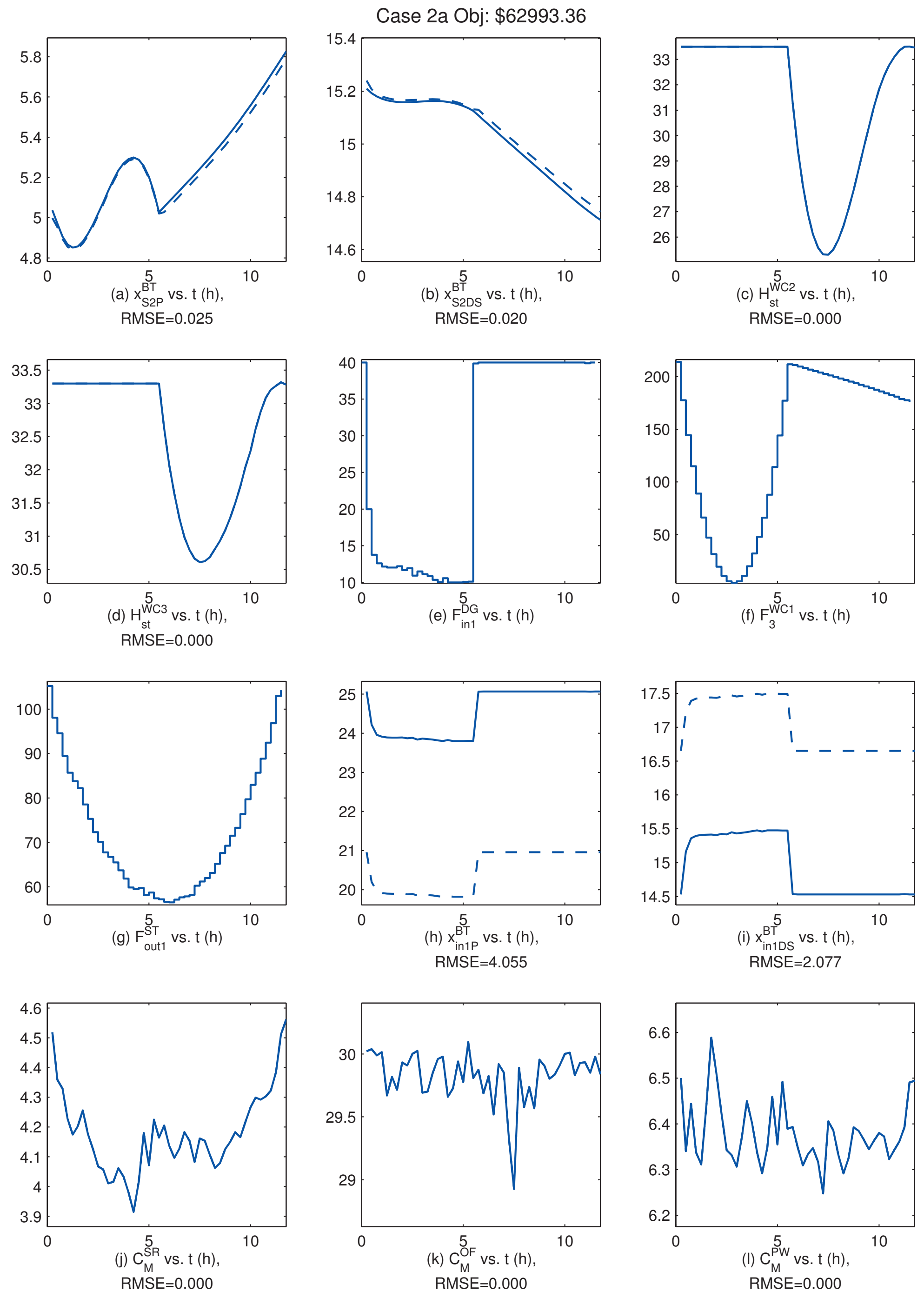


Figure 5.11: Case 2a: Kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using full-state feedback.
Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

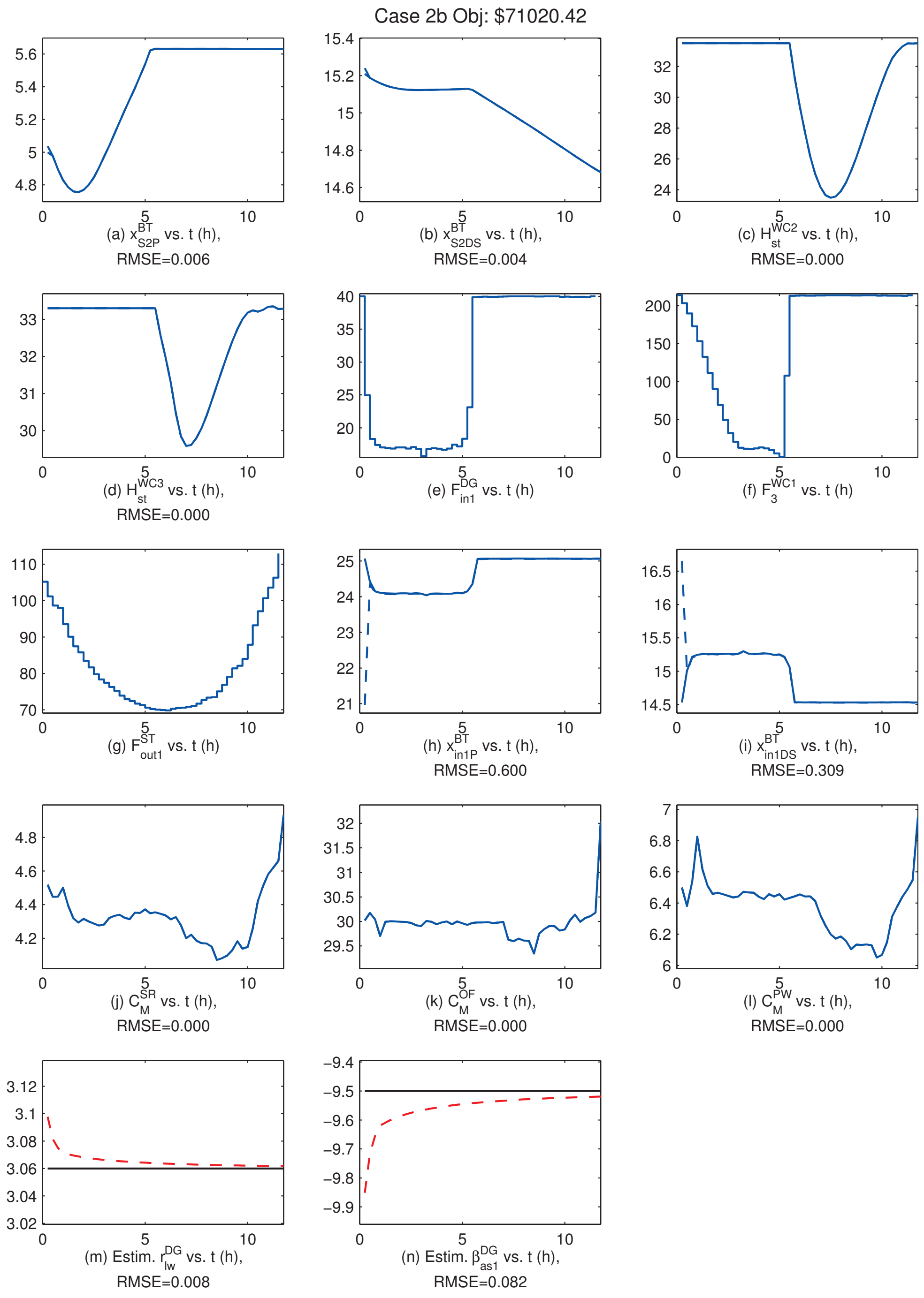


Figure 5.12: Case 2b: Kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using simple parameter estimation to estimate parameters (r^{DG}_{lw}, β^{DG}_{as1}). States obtained through full-state feedback.
Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

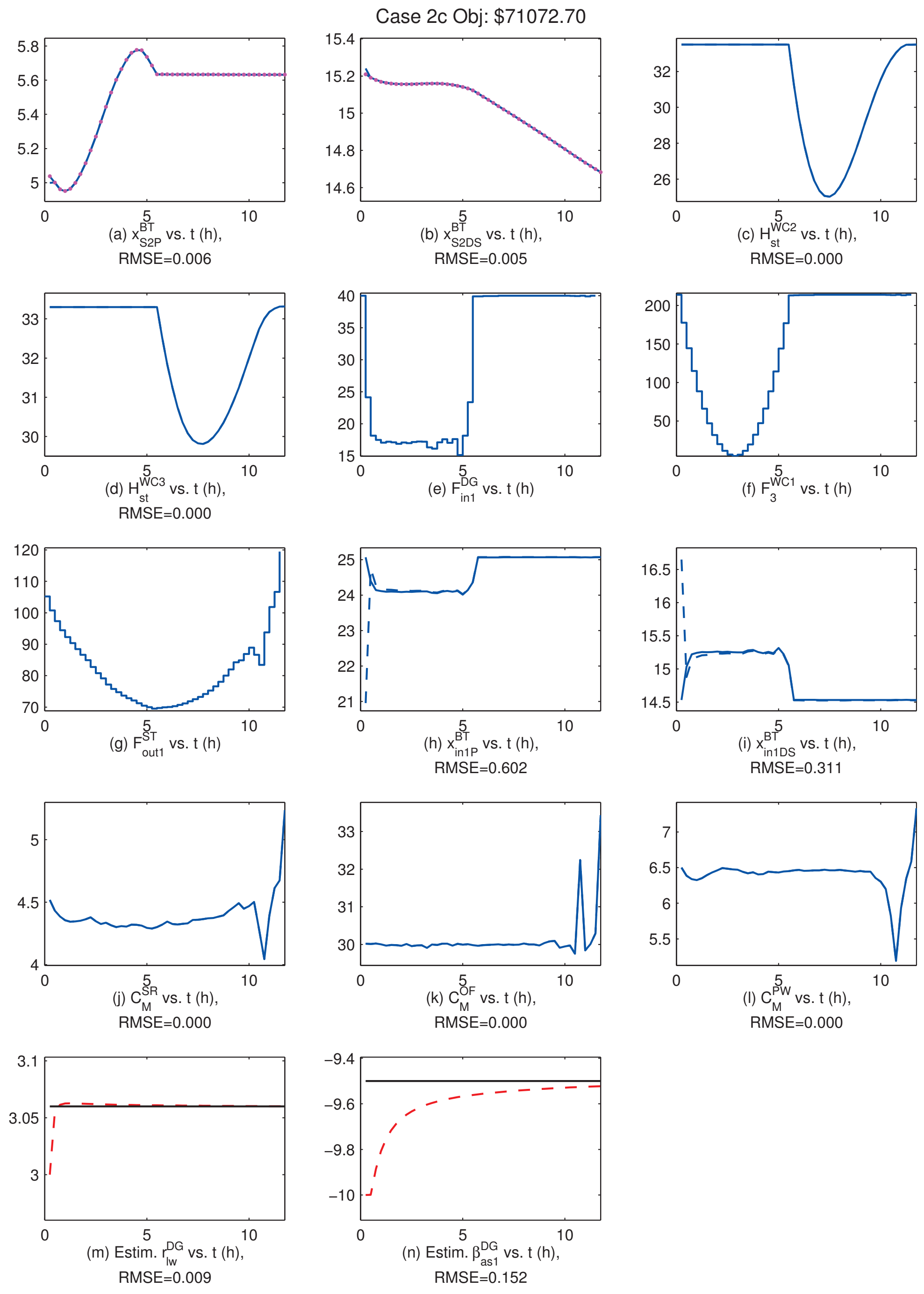


Figure 5.13: Case 2c: Kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using joint state and parameter estimation to estimate mismatched parameters (r^{DG}_{lw}, β^{DG}_{as1}) and missing differential states (x^{BT}_{S2P}, x_{S2DS}).
Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories, magenta dots = the estimates of missing state measurements.

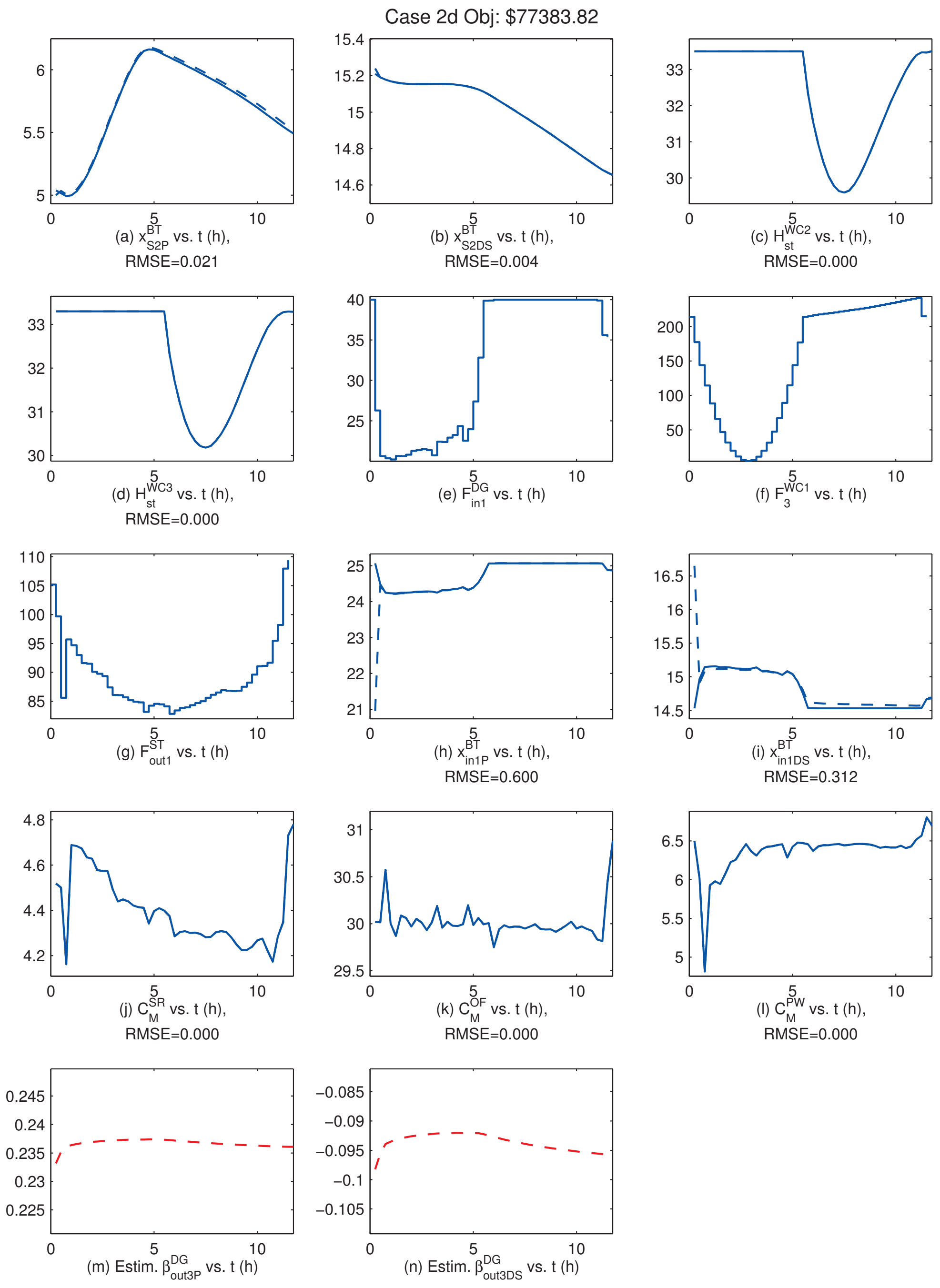


Figure 5.14: Case 2d: Kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}), and using simple parameter estimation to estimate algebraic biases (β^{DG}_{out3P}, β^{DG}_{out3DS}). States obtained through full-state feedback. **Legend**: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

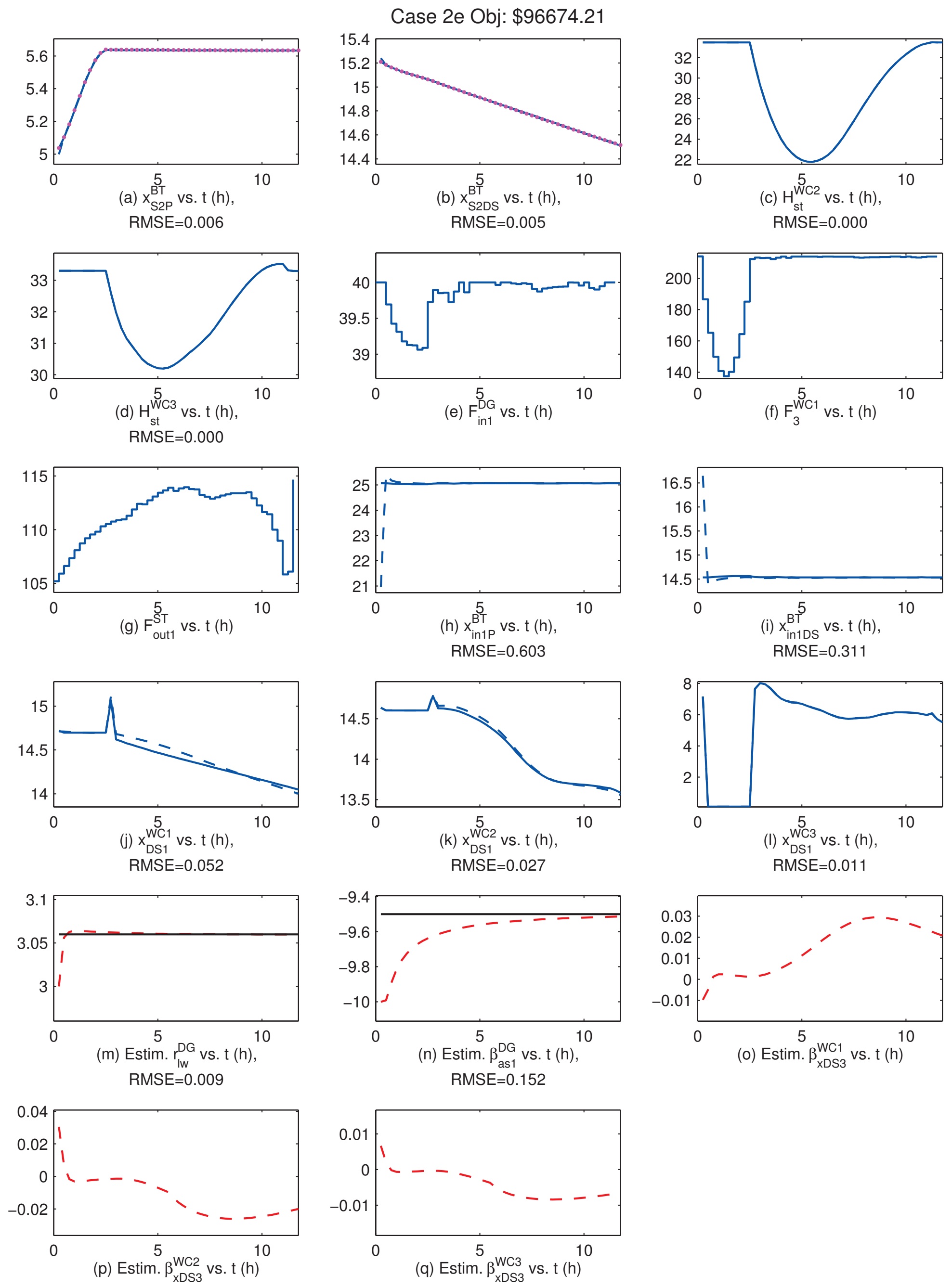


Figure 5.15: Case 2e: Kraft model, with mismatch in (r^{DG}_{lw}, β^{DG}_{as1}, β^{WC1}_{DR}), and using simple parameter estimation to estimate algebraic biases (r^{DG}_{lw}, β^{DG}_{as1}, β^{WC1}_{xDS3}, β^{WC2}_{xDS3}, β^{WC3}_{xDS3}). States obtained through full-state feedback.

Legend: Dashed lines = controller predicted trajectories ($\hat{x}_{j|j-1}$,$\hat{z}_{j|j-1}$,$\hat{\theta}_{j|j-1}$), solid lines = actual plant trajectories.

Case	**Mode**[1]	**Estimated** θ	**Missing States**	**Obj ($)**	SMSE$_x$	SMSE$_z$	SMSE$_\theta$
2a	F	-	-	62,993	1.03×10^{-3}	20.8	-
2b	P, F	r_{lw}^{DG}, β_{as1}^{DG}	-	71,020	5.00×10^{-5}	0.455	0.00679
2c	J	r_{lw}^{DG}, β_{as1}^{DG}	x_{S2P}^{BT}, x_{S2DS}^{BT}	71,072	5.00×10^{-5}	0.460	0.0232
2d	P, F	β_{out3P}^{DG}, β_{out3DS}^{DG}	-	77,383	4.80×10^{-4}	0.457	-
2e	J	r_{lw}^{DG}, β_{as1}^{DG}, β_{xDS3}^{WC1}, β_{xDS3}^{WC2}, β_{xDS3}^{WC3}	x_{S2P}^{BT}, x_{S2DS}^{BT}	96,674	6.00×10^{-5}	0.463	0.0232

[1]**Mode:** F = full state feedback, P = simple parameter estimation, J = joint estimation.

Table 5.4: Summary of Kraft case studies. Mismatch in r_{lw}^{DG}, β_{as1}^{DG} in cases 2a-2d. Case 2e is distinct from the others: (1) shorter downtime; (2) an additional mismatch in β_{DR}^{WC1}. (Smaller SMSEs are better.)

qualities $(C_M^{SR}, C_M^{OF}, C_M^{PW})$ (Fig. 5.11j, k, l) as close to their targets (4.5, 30.0, 6.5) as possible. We observe that due to the mismatch, there is a persistent gap between the model prediction and the plant values in the trajectories for x_{in1P}^{BT} and x_{in1DS}^{BT} (Fig. 5.11h, i). There are also signs of mismatch in trajectories for x_{S2P}^{BT} and x_{S2DS}^{BT} (Fig. 5.11a, b), though they are almost negligibly small compared to the magnitude of the variables. Full-state feedback, owing to the fact that it can only update the initial differential states $x_j = \hat{x}_j$ at every iteration, is unable to close the prediction gap in x_{in1P}^{BT} and x_{in1DS}^{BT} (Fig. 5.11h, i).

- In Case 2b (Fig. 5.12), a simple parameter estimation of the mismatched parameters r_{lw}^{DG} and β_{as1}^{DG} (Fig. 5.12m, n) converges fairly quickly, and the prediction gap in x_{in1P}^{BT} and x_{in1DS}^{BT} (Fig. 5.12h, i) is rapidly closed. If we compare the trajectories of this case with that of the previous (Case 2a), we notice several things. We note that qualitative behavior of the input trajectories are markedly different in the two cases, with the inputs of Case 2b being more conservative and move-suppressed. The overall shapes of input trajectories in Case 2b of F_{in1}^{DG}, F_3^{WC1} and F_{out1}^{ST} (Figs. 5.12e, f, g) differ significantly from the full-state feedback Case 2a (Figs. 5.11e, f, g). The product quality targets can be seen to fluctuate less in Case 2b. The overall prediction errors in x, z and θ (as witnessed by the values of SMSE$_x$, SMSE$_z$ and SMSE$_\theta$) are also significantly lower in Case 2b, and the objective is appreciably better. This provides support to the premise that an economic improvement may be had by making corrections to a model.

- In Case 2c (Fig. 5.13), we repeat Case 2b by making unavailable state measurements x_{S2P}^{BT} and x_{S2DS}^{BT}, thus entailing the use of a joint state and parameter estimation strategy.

The trajectories for x_{S2P}^{BT} and x_{S2DS}^{BT} (Fig. 5.13a, b) indicate that the state estimation was successful in tracking the true states of the plant. The parameters r_{lw}^{DG} and β_{as1}^{DG} (Fig. 5.13m, n) were also found to converge rapidly to their true values. It is interesting to note that although the SMSE$_x$, SMSE$_z$ and SMSE$_\theta$ for Case 2c are higher than for Case 2b, the objective function of the former is actually marginally improved over the latter. However, the difference is insignificant, and for most purposes we can consider the controller performance in Cases 2b and 2c to be almost identical.

- In Case 2d (Fig. 5.14), although we estimate the disturbances instead of the mismatched parameters, we find the results to be very similar to that of Case 2b, because finding appropriate values for the chosen disturbances has an equivalent effect to correcting the actual mismatched parameters. The prediction errors (SMSE$_x$, SMSE$_z$) in this case are higher than that of Case 2b, but the objective value is perceptibly improved in this case.

- Case 2e (Fig. 5.15) is an attempt to investigate the performance of a joint estimation strategy in the presence of various effects such as noise, missing state measurements, and additional mismatch. In this case, we estimate states, state disturbances and mismatched parameters. The trajectories of the states with missing measurements x_{S2P}^{BT} and x_{S2DS}^{BT} (Fig. 5.15a, b) indicate that the estimated states closely track reality. As with the other estimation cases, the estimator manages to quickly converge to the correct value of r_{lw}^{DG} and β_{as1}^{DG} (Fig. 5.15m, n).

  In this case, we have chosen to estimate state disturbances (β_{xDS3}^{WC1}, β_{xDS3}^{WC2}, β_{xDS3}^{WC3}, Figs. 5.15o, p, q) to compensate for a mismatch in a parameter used to calculate Displacement Ratio (DR), β_{DR}^{WC1}. A mismatch in DR would normally cause a persistent mismatch in x_{DS1}^{WC1} which would propagate to other washer models due to the highly interacting counter-current configuration of the washers. In this case, the gap is very slight but visible in the trajectories for x_{DS1}^{WC1} and x_{DS1}^{WC2} (Figs. 5.15j, k). The state disturbance trajectories (Figs. 5.15o, p, q) can be seen to be attempting to to compensate for this. Because the state disturbances correct for the velocity at which differential variables change, they can compensate for integrating errors affecting differential state variables, which in this case are the sealtank compositions $x_{DS3}^{WC,i}(i = 1, \ldots, 3)$. Their effect on the washer outlet compositions, $x_{DS1}^{WC,i}(i = 1, \ldots, 3)$—the mismatched variables—however, is indirect. In this particular case, the state disturbances perform acceptably in closing the prediction gap, despite the fact that they are only able to indirectly affect the variables of interest. In general, state disturbances alone are insufficient for correcting mismatch in algebraic variables, and therefore it is important to utilize model knowledge in the selection of disturbances to estimate.

We wish to emphasize that although we have shown that economic improvements can be realized by correcting model parameters and compensating for model mismatch, it would be imprudent for us to make the broad assertion that that this is always the case. In some scenarios, it is possible for an incorrect model to prescribe control actions that give rise to better performance. In reality, owing to the diverse interacting effects of nonlinearity, stochasticity, etc. it is sometimes possible to blunder into the right with the wrong model. We see hints of this in the Kraft case studies, where a case with a higher prediction error (SMSE_x, SMSE_z, SMSE_θ) can sometimes bring about a better objective value. Therefore we offer these estimation algorithms as a tool for situations where they are able to add value.

5.8 Conclusion

In this chapter, we applied state and parameter estimation strategies in the context of a Economic MPC control framework applied to plants in partial shutdown conditions. Novel configurations of a bound-constrained UKF algorithms were used for state and parameter estimation of a large-scale differential algebraic system. In this work, we adopted two main strategies: simple parameter estimation and joint estimation. An adaptation of the estimation algorithm was necessary to accommodate partial shutdowns. This entailed solving a reduced estimation problem during the shutdown period to account for the fact that states, parameters and measurements are missing from offline units. The motivation for the performing estimation is to obtain estimates for unmeasured states as well as to moderate the effects of plant-model mismatch by correcting incorrect model parameters, with the ultimate aim of correcting the model's predictions. Apart from simply estimating mismatched parameters, we also demonstrate cases in which fictitious disturbance terms are estimated to compensate for plant-model mismatch.

In terms of algorithmic contributions, we proposed an iterative refinement feature to a constrained UKF parameter estimation algorithm to improve estimator performance at every feedback iteration. Our MPC for handling partial shutdowns uses a multi-tiered optimization formulation that computes a set of input trajectories that are the result of prioritizing several competing objectives. In the first tier, a dynamic feasibility ensures that terminal constraints and parameter estimates from a one-step estimator are feasible throughout the model prediction horizon. We believe that the utility of this simple technique reaches beyond this work, because it represents a practical method for ensuring the feasibility of terminal constraints and parameter values obtained from estimators whose prediction horizon is shorter than that of the control optimization problem. An economics tier is subsequently solved to determine the maximum economics obtainable from a given state. Within the solution space of economically optimal trajectories, the third tier optimization attempts to find a solution that delivers the minimum possible deviation from product quality targets during the shutdown and restoration periods. The final tier returns a solution that satisfies all the above objectives and corresponds to minimum input effort.

Chapter 6

Problem Formulation and Solution Techniques

In the course of producing this thesis, we developed a software modeling system for representing dynamic optimization problems. We present the reader with an overview of the conceptual ideas behind the language, and a general description of the code generation process that is a pivotal component of the modeling system. Several germane mathematical transformations are discussed. In the second part of this chapter, we discuss modeling formulation and solution techniques used in this work. General formulations and numerical experiences are recorded.

6.1 Modeling Language for Dynamic Optimization (MLDO)

One of the contributions of this thesis to the corpus of modeling tools is a Modeling Language for Dynamic Optimization, MLDO [32]. MLDO is a domain-specific language for describing DAE-based dynamic optimization problems. Its syntax is minimalist and math-like by design, and only constructs related to dynamic optimization are included in its feature set. It admits problems of the following form:

$$\min_{u(t)} \Phi(x(t), z(t), u(t), \delta(t), p, \theta) \tag{6.1}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), z(t), u(t), \delta(t), p, \theta), \quad x(0) = x_0 \tag{6.2}$$

$$g(x(t), z(t), u(t), \delta(t), p, \theta) \leq 0 \tag{6.3}$$

$$h(x(t), z(t), u(t), \delta(t), p, \theta) = 0 \tag{6.4}$$
$$\text{for } t \in [0, t_f]$$

where Φ = objective function, $x(t) \in \mathbb{R}^{n_x}$ = differential state vector, x_0 = initial state parameter vector, $z(t) \in \mathbb{R}^{n_z}$ = algebraic state vector, $u(t) \in \mathbb{R}^{n_u}$ = control input vector, $p \in \mathbb{R}^{n_p}$ = design variable vector (non-time-varying variables), $\delta(t) \in \{0,1\}^{n_\delta}$ = binary (logical) variable vector, $\theta \in \mathbb{R}^{n_\theta}$ = parameter vector (constant), t_f = final time in prediction horizon.

At the core of the modeling system is a software program that parses a description of a problem (in the MLDO language) and automatically produces code in various computer languages. This component of the modeling system we refer to as the "MLDO code generator". Depending on the target language selected, the output code can be variously used for simulation, optimization, visualization, analysis or documentation. Support for various target languages is an important feature, because for non-trivial nonlinear modeling/optimization problems, a single software tool is rarely sufficient for performing all the tasks necessary to obtain a satisfactory solution. Frequently, an entire ecosystem of software programs and libraries are required.

The MLDO code generator is built on the concept of "metaprogramming", a technique in which code in one programming language is used to generate code in another[1]. There has been some interest in the process systems engineering community in employing these techniques for improving the quality and speed of software development. An example of a metaprogramming application is ControlH [43], a control specifications language developed at Honeywell that is capable of generating C or Ada code. Pantelides [86], in his description of the Speedup process simulation tool, shows the use of the Pascal programming language to generate FORTRAN code for calculating derivatives.

Metaprogramming is particularly suited to situations where large amounts of complicated—but repeated—computer code is required to solve a problem, and where such code is susceptible to human error if written manually. The metaprogramming technique proposes to transfer the task of writing the code to the computer. The user is only required to specify the particular problem in a specially defined language (a "domain-specific language" or DSL in software parlance). A code generator then analyzes the structure of the specified problem and generates the requisite programming code for finding a solution.

[1]In languages with strong metaprogramming traditions such as Lisp, metaprogramming is done by extending the language through meta-linguistic abstractions—this results in what is known as an Internal DSL (domain specific language). In our metaprogramming approach, we do not extend the syntax of an existing language, but instead construct a new one, resulting in an External DSL.

In contrast to conventional programming languages like FORTRAN or C, modeling languages like MLDO are usually *declarative* rather than *imperative*; that is, they contain a *problem definition* rather than the explicit computer instructions required to solve the problem. This is an important attribute that enables MLDO to output code that is clean, concise and readable.

By adopting a one-to-many (as in one model definition, many target languages) approach, MLDO represents a single entry point to process modeling. From a business perspective, this approach allows organizations to protect their investments in modeling effort and time—and to some extent, prevent vendor lock-in—by enabling them to switch to newer modeling platforms as they emerge. With the code processing and generation infrastructure already present in MLDO, new target languages can be added to the MLDO code generator's repertoire with relatively low effort. This flexibility leads to easier migrations, without entailing the manual re-development of an entire model on the new platform of choice (which, in the context of very large models, may require significant effort and is prone to human errors). Any transformations done at the mathematical level (e.g. manual variable/equation scaling, algebraic transformations, etc.) are preserved, which is also an advantage from a maintenance and sustainability point of view.

In short, MLDO can serve as a platform that allows for experimentation with various numerical software configurations and formulation ideas. It enables the user to focus on modeling aspects without having to repeatedly deal with the lower-level boilerplate code. MLDO also has a extensible add-on architecture for applying custom mathematical transformations on MLDO models. It can be used as a reformulation engine to alter model formulations. As of writing, the software implementation of MLDO is able to generate code in the following languages (for specific tasks):

- *Dynamic optimization*
  AMPL, FORTRAN/C++ (using DAEOC libraries), MATLAB (using INTLAB for automatic differentiation)

- *Simulation/DAE integration*
  gPROMS, MATLAB (plain .m, SUNDIALS-interfaced .m), C (`mex` code), AMPL

- *Symbolic analysis*
  Maple

- *Documentation*
  $\LaTeX$

- *Visualization*

MATLAB (.m plotfile), HTML (Sparklines), Python (Matplotlib)

- *Solution Storage*
  JSON[2]

A variety of dynamic optimization software exists in the market (ACADO, DAETools, PROPT, APMonitor, gPROMS, JModelica, DAEOC, TACO etc.), but MLDO is currently unique in its aim to be a modeling system that targets other modeling systems. The MLDO modeling system is written entirely in Python, a multi-paradigm programming language.

In the following sections, we describe some of the techniques employed in MLDO. The intention of this is not to showcase MLDO itself, but rather to use MLDO as a platform for discussing the generic ideas and concepts behind the creation of a domain-specific modeling system. These concepts can be extended to other usage cases and applications.

6.1.1 Simultaneous Dynamic Optimization: Orthogonal Collocation on Finite Elements (OCFE)

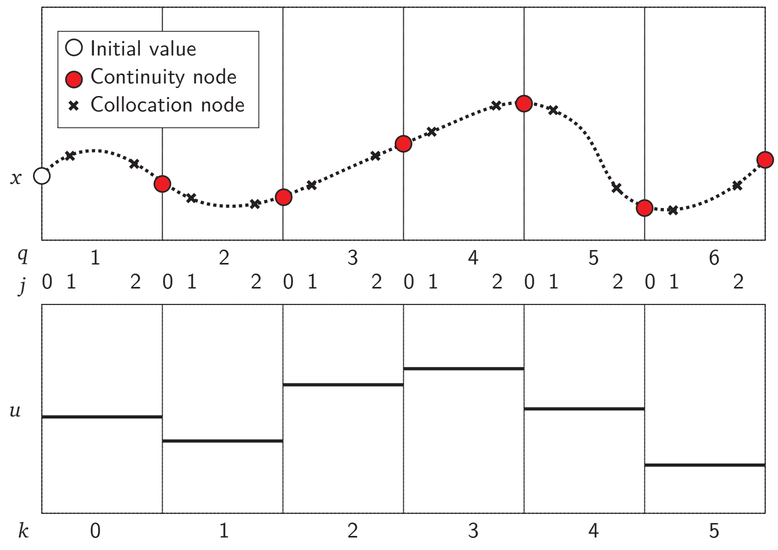


Figure 6.1: Example of OCFE on $\dot{x} = f(x,u)$ for $n_{\mathrm{FE}} = 6$ (number of finite elements) and $n_{\mathrm{COL}} = 2$ (collocation points per element), where q = finite element counter, j = collocation point counter within each finite element, k = discrete sample time.

In this work, we employ the simultaneous method for performing dynamic optimization, which

[2]After each optimization run, the solution point and post-optimality information (Lagrange multipliers, deduced bounds, etc.) are encoded in this data exchange format. This facilitates importation into other software for post-optimality analysis.

involves solving a discretized DAE optimization problem. Orthogonal Collocation on Finite Elements (OCFE) is a weighted-residual method for discretizing a DAE system into a set of algebraic equations that are admissible as constraints in an optimization problem. One of the advantages of this is that it makes the dynamic optimization problem amenable to analysis using mostly optimization theory.

It has been demonstrated that OCFE essentially corresponds to a well-conditioned implicit Runge-Kutta method for integrating DAEs [33]. Implicit Runge-Kutta methods have been shown to have good numerical properties such as having matrices that possess smaller condition numbers and are less susceptible to rounding errors [7]. Given an adequate number of finite elements (vis-à-vis the dynamics of the model), two or three collocation points per element are generally sufficient for a reasonably accurate approximation (Vasantharajan and Biegler [110]). Approximation error can be further controlled using adaptive knot placement techniques [33].

The basic idea behind OCFE is the segmentation of the continuous-time solution profile of the DAE system into components called finite elements. Within each finite element, a residual function containing a polynomial approximation of the solution is forced to be exactly zero at collocation nodes (Figure 6.1). These collocation nodes are chosen as roots of orthogonal polynomial family, such as the Jacobi polynomials. Function (and in some cases, derivative) continuity is imposed at the boundaries of finite elements (continuity nodes, Figure 6.1). Details of this method can be found in Cuthrell and Biegler [33].

To illustrate the method, we will derive the OCFE equations from a DAE system of the following form:

$$\dot{x}(t) = f(x(t), z(t), u(t)), \quad x(0) = x_0 \tag{6.5}$$

$$h(x(t), z(t), u(t)) = 0 \tag{6.6}$$

$$\text{for } t \in [0, t_f]$$

where $x(t)$ = vector of differential variables, $z(t)$ = vector of algebraic variables, $u(t)$ = vector of control input variables (decision variables in an optimization problem). The vector $u(t)$ is assumed to be constant valued at every sample-time (zero-order hold); that is, the input profile is piecewise-constant.

The time horizon $[0, t_f]$ is partitioned into n_{FE} finite elements (see Figure 6.1 for an example). Within each finite element q, we introduce $\tilde{x}(t)$ and $\tilde{z}(t)$, which represent polynomial approximations of the differential and algebraic state profiles respectively. The input profile $\tilde{u}(t)$ is taken to be a piecewise-constant trajectory—whose values are typically decision variables in an optimization problem—that are prescribed to the model. We state these as follows:

$$\tilde{x}(t) = x^{(q)}(\tau) = \sum_{j=0}^{n_{\mathrm{COL}}} x_j^{(q)} \phi_j(\tau), \qquad q = 1, \ldots, n_{\mathrm{FE}} \tag{6.7}$$

$$\tilde{z}(t) = z^{(q)}(\tau) = \sum_{j=0}^{n_{\mathrm{COL}}} z_j^{(q)} \phi_j(\tau), \qquad q = 1, \ldots, n_{\mathrm{FE}} \tag{6.8}$$

$$\tilde{u}(t) = u^{(q)}(\tau) = u_k, \qquad q = 1, \ldots, n_{\mathrm{FE}} \tag{6.9}$$

where τ_i is a rescaled time variable at collocation point i, n_{COL} is the number of collocation points per finite element, $x_j^{(q)}$ and $z_j^{(q)}$ are coefficients to be determined, k is related to q via eqn. 6.21, and $\phi_j(\tau)$ are Lagrange polynomial basis functions, defined as follows:

$$\phi_j(\tau) = \prod_{\substack{l=0 \\ l \neq j}}^{n_{\mathrm{COL}}} \frac{(\tau - \tau_l)}{(\tau_j - \tau_l)} \tag{6.10}$$

The complete collocation procedure is described in detail below.

1. **Time scaling equation**
   For convenience, the time ordinate is rescaled into a [0,1] interval in each finite element q using the following equation:

$$\tau = \frac{t - t_0^{(q)}}{\delta} \tag{6.11}$$

   where q = finite element counter, $t_0^{(q)}$ = left-hand side boundary of the current finite element q, and δ = finite element length (assumed uniform). At specific collocation nodes τ_i within each finite element q, the above equation can be written thus:

$$\tau_i = \frac{t_i^{(q)} - t_0^{(q)}}{\delta}, \qquad i = 0, \ldots, n_{\mathrm{COL}} \tag{6.12}$$

   where $t_i^{(q)}$ = time at finite element q and collocation counter i, n_{COL} = total number of collocation points per finite element. In practice, each τ_i within a finite element corresponds the root of an orthogonal polynomial family [33] over an interval $[0, 1]$ and are constants in the problem. The Jacobi family of orthogonal polynomials is commonly used. Efficient FORTRAN routines for generating Jacobi roots can be found in Villadsen and Michelsen [112]. In the software implementation, we use a table of pre-generated Jacobi roots for collocation. (All references to temporal variables henceforth will be made in terms of the variable τ_i.)

2. **Method of Ordinates: Lagrange Interpolation Polynomials**
Each differential and algebraic variable is approximated with a polynomial. In lieu of the standard coeffients-based representation for polynomials, $P(\tau) = a_0 + a_1\tau + \ldots + a_n\tau^n$ (method of coefficients), Lagrange interpolation polynomials are employed in what is referred to as the method of ordinates [112], where a polynomial is represented thus, $P(\tau) = P(\tau_0)\phi_0(\tau) + P(\tau_1)\phi_1(\tau) + \ldots + P(\tau_n)\phi_n(\tau)$ where ϕ_j are Lagrange basis functions, $\tau_0, \ldots, \tau_n$ are $n+1$ distinct points, and $P(\tau_j)$ are the values of the polynomial evaluated at those points. The variables at the collocation nodes τ_i are represented thus:

$$x^{(q)}(\tau_i) = \sum_{j=0}^{n_{\text{COL}}} x_j^{(q)} \phi_j(\tau_i) \tag{6.13}$$

$$\frac{dx^{(q)}(\tau_i)}{d\tau} = \sum_{j=0}^{n_{\text{COL}}} x_j^{(q)} \frac{d\phi_j(\tau_i)}{d\tau} \tag{6.14}$$

$$z^{(q)}(\tau_i) = \sum_{j=0}^{n_{\text{COL}}} z_j^{(q)} \phi_j(\tau_i) \tag{6.15}$$

and,

$$\phi_j(\tau_i) = \prod_{\substack{l=0 \\ l\neq j}}^{n_{\text{COL}}} \frac{(\tau_i - \tau_l)}{(\tau_j - \tau_l)} \tag{6.16}$$

$$\frac{d\phi_j(\tau_i)}{d\tau} = \left(\prod_{\substack{l=0 \\ l\neq j}}^{n_{\text{COL}}} \frac{1}{(\tau_j - \tau_l)} \right) \cdot \left(\sum_{\substack{k=0 \\ k\neq j}}^{n_{\text{COL}}} \prod_{\substack{l=0 \\ l\neq j, l\neq k}}^{n_{\text{COL}}} (\tau_i - \tau_l) \right) \tag{6.17}$$

Observe from equation (6.16) that:

$$\phi_j(\tau_i) = \begin{cases} 1, & \text{when } i = j \quad (6.18) \\ 0, & \text{when } i \neq j \quad (6.19) \end{cases}$$

As a consequence, it follows that the polynomial coefficients of $x^{(q)}(\tau_i)$ and $z^{(q)}(\tau_i)$ simply correspond to the values of the differential and algebraic states at the collocation points (i.e. $x_i^{(q)}$ and $z_i^{(q)}$, respectively), thus equations (6.13, 6.15, and 6.16) can be dropped from the problem.

3. **Control Input Vector**

   The control input vector is represented by the following equation:

$$u^{(q)}(\tau_i) = u_k, \qquad q = 1, \dots, n_{\text{FE}},\ i = 0, \dots, n_{\text{COL}} \tag{6.20}$$

$$k = \lfloor \frac{q-1}{\eta} \rfloor \tag{6.21}$$

   where u_k is the control input to the plant with a zero-order hold (piecewise constant) and η is the number of finite elements per sample. The temporal ordinate of the control inputs is measured in terms of *sample units* (k). In this implementation [9], multiple finite elements per sample may be specified (through the η parameter) for the purpose of improving accuracy. In a standard OCFE implementation, the length of one sample time corresponds exactly to the length of one finite element, therefore $\eta = 1$.

4. **Continuity equation**

   Continuity between finite elements is enforced in the differential state variables by using the polynomial approximation derived for the previous finite element to calculate the value of the differential variable at the left-hand side boundary of the current finite element (at the continuity node, refer to Figure 6.1).

$$x_0^{(q)} = \sum_{j=0}^{n_{\text{COL}}} x_j^{(q-1)} \phi_j(1) \qquad \text{for } q = 2, \dots, n_{\text{FE}} \tag{6.22}$$

5. **Residual equations**

   The differential algebraic equation system can be represented in residual form. To obtain the residual equations, we replace the differential and algebraic state variables and the inputs in (6.5–6.6) with the following discretized approximations (expressions 6.23–6.26).

$$\dot{\tilde{x}}(t) = \left[\frac{dx^{(q)}(\tau)}{d\tau}\right] \frac{d\tau}{dt}\bigg|_{\tau=\tau} = \left[\sum_{j=0}^{n_{\text{COL}}} x_j^{(q)} \frac{d\phi_j(\tau)}{d\tau}\right] \frac{1}{\delta} \tag{6.23}$$

$$\tilde{x}(t) = x^{(q)}(\tau) \tag{6.24}$$

$$\tilde{z}(t) = z^{(q)}(\tau) \tag{6.25}$$

$$\tilde{u}(t) = u^{(q)}(\tau) = u_k \tag{6.26}$$

   Therefore,

$$R_1 = \dot{\tilde{x}}(t) - f(\tilde{x}(t), \tilde{z}(t), \tilde{u}(t)), \qquad \tilde{x}(0) = x_0 \tag{6.27}$$

$$R_2 = h(\tilde{x}(t), \tilde{z}(t), \tilde{u}(t)) \tag{6.28}$$

where R_1, R_2 are residual equations. At the collocation nodes, we require the residuals to be zero, therefore we can rewrite the above as follows:

$$\frac{1}{\delta}\sum_{j=0}^{n_{\mathrm{COL}}} x_j^{(q)} \frac{d\phi_j(\tau_i)}{d\tau} - f(x_i^{(q)}, z_i^{(q)}, u_k) = 0, \qquad q = 1, \ldots, n_{\mathrm{FE}},\ i = 1\ldots n_{\mathrm{COL}} \tag{6.29}$$

$$h(x_i^{(q)}, z_i^{(q)}, u_k) = 0, \qquad q = 1, \ldots, n_{\mathrm{FE}},\ i = 1\ldots n_{\mathrm{COL}} \tag{6.30}$$

$$x_0^{(1)} = x_0 \tag{6.31}$$

These residual equations at collocation nodes are algebraic equations that can be enforced as equality constraints in an optimization problem.

For a compact form of a fully-collocated dynamic optimization problem, we refer the reader to Section 6.1.2.

6.1.2 Model Discretization for DAE Optimization in MLDO

One of the primary uses of the MLDO code generator in this thesis is for generating DAE optimization code (simultaneous approach) that is solvable using any number of commercial/open-source numerical optimization solvers.

To achieve this goal in the least amount of time, we elected to leverage an intermediary modeling language, AMPL [47]. AMPL is a popular commercial mathematical programming system with a standardized software interface to optimization solvers of various types[3]. This design decision helped us to avoid having to write specific interface code for each individual solver. It also allowed us to take advantage of a host of important features in AMPL such as automatic differentiation (exact 1st and 2nd derivatives), sparsity structure generation, bounds checking, model pre-processing/presolve and others.

We note that AMPL was designed as a purely algebraic modeling system, therefore in order to pose a dynamic optimization problem in it, the problem has to first be converted into algebraic form through discretization. Using a direct transcription (simultaneous) approach where both controls and states are discretized, the MLDO code generator transforms the DAE model into a set of nonlinear algebraic equations via a discretization process (either orthogonal collocation

[3]This not only includes a variety of nonlinear solvers (IPOPT, CONOPT, KNITRO, etc.), but also solvers for different types of optimization problems, i.e. continuous/discrete, convex/nonconvex, linear/quadratic/general nonlinear

[see Section 6.1.1] or backward/forward Euler). This equation system can then be posed as constraints in a conventional nonlinear program (NLP) in AMPL.

The system in Eqns. 6.1 – 6.4 can be transformed into various (discretized) algebraic forms. The following are canonical forms of the discretization methods supported by MLDO:

Orthogonal Collocation on Finite Elements

$$\min \Phi(x_i^{(q)}, z_i^{(q)}, u_k, \delta_k, p, \theta) \tag{6.32}$$

$$\text{s.t. } D_{x,i}^{(q)} \triangleq \frac{1}{\Delta t} \sum_{j=0}^{n_{\text{COL}}} \left[x_j^{(q)} \frac{d\phi_j(\tau_i)}{d\tau} \right], \qquad q = 1 \dots n_{\text{FE}},\ i = 1 \dots n_{\text{COL}} \tag{6.33}$$

$$D_{x,i}^{(q)} = f(x_i^{(q)}, z_i^{(q)}, u_k, \delta_k, p, \theta), \qquad q = 1 \dots n_{\text{FE}},\ i = 1 \dots n_{\text{COL}} \tag{6.34}$$

$$g(x_i^{(q)}, z_i^{(q)}, u_k, \delta_k, p, \theta) \leq 0, \qquad q = 1, \dots, n_{\text{FE}},\ i = 1, \dots, n_{\text{COL}} \tag{6.35}$$

$$h(x_i^{(q)}, z_i^{(q)}, u_k, \delta_k, p, \theta) = 0, \qquad q = 1 \dots n_{\text{FE}},\ i = 1, \dots, n_{\text{COL}} \tag{6.36}$$

$$x_0^{(q)} = \sum_{j=0}^{n_{\text{COL}}} \left[x_j^{(q-1)} \phi_j(1) \right], \qquad q = 2 \dots n_{\text{FE}} \tag{6.37}$$

$$x_0^{(1)} = x_0 \tag{6.38}$$

where the following terms are pre-computed constant parameters:

$$\phi_j(1) = \prod_{\substack{l=0 \\ l \neq j}}^{n_{\text{COL}}} \frac{(1 - \tau_l)}{(\tau_j - \tau_l)}, \qquad \frac{d\phi_j(\tau_i)}{d\tau} = \left(\prod_{\substack{l=0 \\ l \neq j}}^{n_{\text{COL}}} \frac{1}{(\tau_j - \tau_l)} \right) \cdot \left(\sum_{\substack{r=0 \\ r \neq j}}^{n_{\text{COL}}} \prod_{\substack{l=0 \\ l \neq j, l \neq r}}^{n_{\text{COL}}} (\tau_i - \tau_l) \right)$$

$$k = \lfloor \frac{q-1}{\eta} \rfloor, \quad \tau_i = i\text{'th root of a Jacobi polynomial of degree } n_{\text{COL}}$$

Backward Euler

$$\min \Phi(x_k, z_k, u_k, \delta_k, p, \theta) \tag{6.39}$$

$$\text{s.t. } \frac{x_k - x_{k-1}}{\Delta t} = f(x_k, z_k, u_{k-1}, \delta_k, p, \theta), \qquad k = 1 \dots N \tag{6.40}$$

$$g(x_k, z_k, u_k, \delta_k, p, \theta) \leq 0, \qquad k = 0 \dots N \tag{6.41}$$

$$h(x_k, z_k, u_k, \delta_k, p, \theta) = 0, \qquad k = 0 \dots N \tag{6.42}$$

Forward Euler

$$\min \Phi(x_k, z_k, u_k, \delta_k, p, \theta) \tag{6.43}$$

$$\text{s.t.} \quad \frac{x_{k+1}-x_k}{\Delta t} = f(x_k, z_k, u_k, \delta_k, p, \theta), \qquad k = 0 \ldots N-1 \tag{6.44}$$

$$g(x_k, z_k, u_k, \delta_k, p, \theta) \leq 0, \qquad k = 0 \ldots N \tag{6.45}$$

$$h(x_k, z_k, u_k, \delta_k, p, \theta) = 0, \qquad k = 0 \ldots N \tag{6.46}$$

In most situations, we would recommend using either orthogonal collocation or backward Euler due to their superior numerical stability properties. Forward Euler is merely provided for completeness.

6.1.3 The MLDO Code Generation Process

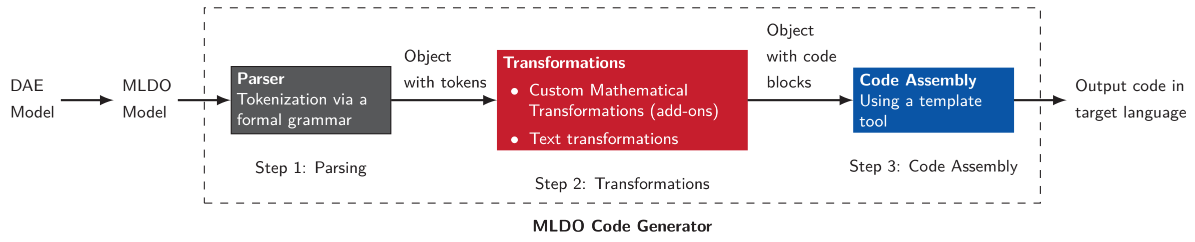


Figure 6.2: Simplified code generation workflow for MLDO.

The MLDO code generation process follows the general workflow shown in Figure 6.2. This workflow figure has been vastly simplified, but it conveys the key operations performed by the code generator. A DAE model (in mathematical form) is the starting point. It is transcribed into an MLDO model by the user. The MLDO Code Generator program reads in the MLDO and outputs code in the target languages of choice. Within the code generator, the major procedures are carried out in 3 steps. We describe them in detail below.

Step 1: Parsing

An MLDO model file is decomposed into meaningful elements. A *parser* is a module within the code generator that carries out the procedure of breaking down a plain-text description of model into its constituent elements via a process known as *tokenization*. It does so by utilizing a set of rules called a *formal grammar*. In human languages, the syntax of a language (English, French, etc.) is defined by their grammars. Analogously, a formal grammar is a set of mathematical/programmatic rules that define the syntax of a computer language. To employ a very simplified example, consider this self-explanatory grammar for a Canadian postal address (notated in Backus-Naur Form, BNF[4]):

[4]BNF is one of several notation methods for formal grammars

```
<postaladdress> ::= <name> <EOL> <streetaddress> <EOL>
<name>          ::= <title> <firstname> <middleinitial> <lastname>
<streetaddress> ::= <housenumber> <streetname> <streettype> <EOL>
                    <townname> <province> <postalcode>
<streettype>    ::= "St." | "Ave." | "Blvd."
<province>      ::= "ON" | "QC" | "AB" | "BC" | "MB" | "SK" | "NL" | "NS"
```

where `<token>` are tokens, `"string"` are strings, `EOL` is the end-of-line, and the pipe symbol | represents the OR operator. Notice that tokens can be broken down into other tokens; for instance, the `<name>` token in the `<postaladdress>` definition is itself defined as the combination of several other tokens, namely `<title>`, `<firstname>`, `<middleinitial>`, and `<lastname>`. Suppose we are given the below address as an input string to the parser:

```
Mr John R Doe
1 Main St.
Hamilton ON L8S 4L7
```

Using the grammar defined above, a parser would decompose the address into tokens, generating the following mapping:

Token	**String corresponding to the token**
postaladdress	Mr John R Doe 1 Main St. Hamilton ON L8S 4L7
name	Mr John R Doe
title	Mr
firstname	John
middleinitial	R
lastname	Doe
streetaddress	1 Main St. Hamilton ON L8S 4L7
housenumber	1
streetname	Main
streettype	St.
townname	Hamilton
province	ON
postalcode	L8S 4L7

Table 6.1: Parsed results of a Canadian postal address

The grammar[5] that defines the MLDO language is considerably more complex than the grammar

[5]The MLDO parser relies on a Python library called `pyparsing`.

in the example above, but it essentially operates on the same principles.

A dynamic optimization problem has a well-structured canonical form, which lends itself to a precise grammar. An MLDO model is tokenized into, among other things, declarations of the objective functional Φ, variables, DAE functions $\{f(\cdot), h(\cdot)\}$, and constraints $\{g(\cdot)\}$. Furthermore, variables are differentiated into 4 types: differential (x), algebraic (z), input (u) or design (p). Note that the discrete variables (δ) do not constitute a distinct type by themselves—they are obtained by declaring variables belonging to any of the previously described types as having the quality of being binary. Time-invariant parameters (θ) are not variables, but they also considered a type. In MLDO, the user is required to explicitly declare each variable's type in the optimization problem. The mapping of variables to their correct types ensures that the different elements in an equation receive the appropriate treatment during the transformation stage. The MLDO parser produces an "Object with Tokens" similar in form to that of Table 6.1, which can then be manipulated and transformed to obtain the desired result.

We note that to add a new target language, a developer only needs to write code for Steps 2 and 3 of the code-generation process. Step 1 is common to all target languages.

Step 2: Transformations

The Object with Tokens generated in the previous step contains a generic description of the dynamic optimization problem that has not yet been customized for any particular target language. In this step, we carry out transformations to the object that will give results that are particular to the desired target language. Custom mathematical transformations—implemented as add-ons—are also applied at this stage (see Section 6.1.4 for a sampling of the types of transformations that can be performed).

Text processing techniques are used to rewrite individual strings in Object with Tokens into the form acceptable in the target language. Consider the following (simplified) example of the transformation of several equations and constraints in MLDO into AMPL (with orthogonal collocation code contrasted with Backward Euler):

Refering to Table 6.2, we notice that array indices (e.g. `[FE_,COL_]`, `[k_]`, etc.) in the AMPL code differ in the collocation and Euler cases. They also vary with different variable types (e.g. `alpha` is a parameter, `x` is a differential variable, `u` is a piecewise constant input variable) because each variable has a different time representation (but which all map to a common time grid). MLDO makes use of all the information contained in Object with Tokens to differentiate the variable types and perform the correct mapping for each element in the model.

MLDO Statement	AMPL Statement (orthogonal collocation)
`$x = -alpha * x;`	`Derivative_x_[FE_,COL_] = -alpha * x[FE_,COL_];`
`x(0) = 0.0;`	`x[1,0] = 0.0;`
`-1 <= x <= y;`	`-1 <= x[FE_,COL_] <= y[FE_,COL_];`
`u >= 0;`	`u[fe2samp_[FE_]] >= 0;`
MLDO Statement	**AMPL Statement (Backward Euler)**
`$x = -alpha * x;`	`Derivative_x_[k_] = -alpha * x[k_];`
`x(0) = 0.0;`	`x[0] = 0.0;`
`-1 <= x <= y;`	`-1 <= x[k_] <= y[k_];`
`u >= 0;`	`u[k_] >= 0;`

Table 6.2: Text transformations from MLDO to AMPL (collocation and Backward Euler)

We utilize a combination of several mechanisms to perform these text transformations: sub-grammars, regular expressions and simple text substitutions. Sub-grammars are simply grammars defined at the equation or constraint level to enable us to parse strings at a finer level. While the grammar in the parsing step decomposed the model into equations, constraints and variables, the sub-grammars defined in this step decompose equations and constraints into their individual variables and operations (e.g. `=,<=,>=,+,-,*,/,^` and so on). Regular expressions are codified expressions used for matching specific patterns in a set of strings. They allow one to find complex textual patterns in a string, and substitute them with a custom replacement string (functionally, they can be thought of as an advanced version of the Find and Replace text feature found in all word processors). Finally, simple text substitutions are simply fast Find and Replace operations.

Sections	Code Blocks (transformed)
`minimize`	$\Phi(\cdot)$
`statevars`	Declarations for x
`controlvars`	Declarations for u
`othervars`	Declarations for z
`designvars`	Declarations for p
`params`	Declarations for θ
`dae`	$f(\cdot)$ and $h(\cdot)$
`init`	$x(0)$
`constraints`	$g(\cdot)$
`ocfe`	Problem-specific parameters, e.g. t_f, n_{FE}, n_{COL}, etc.

Table 6.3: Major sections in Object with Code Blocks

The results of these transformations are code blocks in the target language, which we neatly classify into different sections (see Table 6.3) in an object called "Object with Code Blocks". The various pieces contained in this object can then be used to assemble the final output file.

Step 3: Code Assembly

The chief means of assembling disparate pieces of code blocks into a coherent output file is through the use of a template tool[6]. A *template* is a text file with placeholders for various code blocks, and a template tool fills in the placeholders with the relevant code block according to certain pre-specified rules. Below is a vastly simplified example of an AMPL template with placeholders for various code blocks denoted `${codeblock}`:

```
# === AMPL file generated from MLDO ===
# --- Variable and Parameter Declarations ---
${params}
${statevars}
${controlvars}
${othervars}
${designvars}
# --- Optimization Problem ----
minimize obj: ${minimize};
subject to
${dae}
${init}
${constraints}
# --- Solution ----
${ocfe}
solve;
```

The template tool outputs a text file with the various codeblocks contained in the Object with Code Blocks substituted into the placeholders' positions. The model invariant code segments (that is, codes that are not model-specific) are hard-coded into the template. Each individual target language requires its own customized template. The file generated at the end of this process can either be subject to further post-processing, or run in the target environment.

Example of MLDO code generation

Consider the following dynamic optimization example problem (due to Tjoa and Biegler

[6]MLDO uses the Python template library, `Cheetah`.

[22]):

$$\min_{u(t)} -[1 - x_a(t_f) - x_b(t_f)] \tag{6.47}$$

$$\text{s.t. } \dot{x}_a(t) = u(t) \cdot [10 \cdot x_b(t) - x_a(t)], \qquad x_a(0) = 1 \tag{6.48}$$

$$\dot{x}_b(t) = u(t) \cdot [x_a(t) - 10 \cdot x_b(t)] - [1 - u(t)] \cdot x_b(t), \qquad x_b(0) = 0 \tag{6.49}$$

$$0 \leq u(t) \leq 1 \tag{6.50}$$

$$t \in [0, 0.4]$$

where $u(t)$ = control input variable, $x_a(t), x_b(t)$ = differential variables, t_f = terminal time. In order to solve this problem in AMPL using Backward Euler (with $n_{\text{FE}} = 16$ finite elements), we transcribe it in MLDO syntax as follows:

```
# --- MLDO Model file ---
minimize:           -(1 - xa(tf_) - xb(tf_));
statevars:          xa; xb;
controlvars:        u;
dae:                $xa = u*(10*xb - xa);
                    $xb = u*(xa - 10*xb) - (1 - u)*xb;
init:               xa(0) = 1;
                    xb(0) = 0;
constraints:        0 <= u <= 1;
ocfe:               tf = 0.4; nFE = 16; mode = euler; submode = bkwd;
```

Any text following a hash symbol (#) is treated as a comment. The objective function resides in the `minimize` section. Differential/control input variables are declared in the `statevars` and `controlvars` section respectively. The `dae` section contains the differential algebraic equation (DAE) system, and `init` section, their initial values. The `$` sign denotes the differential operator. Inequality constraints are written in the `constraints` section. The `ocfe` section contains information specific to this problem instance, such as final time, no. of finite elements, etc. Each section name is terminated by a colon (`:`), while each statement within a section is terminated with a semicolon (`;`).

AMPL Output

A section of the AMPL code that is generated from the above MLDO model appears below (extraneous code has been omitted for the sake of brevity):

```
# === AMPL file generated from MLDO ===
# ======== Euler Parameters ========
param tf_;
param nFEperSample_;
param sampleSize_;
param nSamples_ := tf_/sampleSize_;            # number of samples
param nk_ := round(nFEperSample_ * nSamples_); # number of finite elements
param delta_ := tf_/nk_;                       # size of finite element
set kSet_ ordered := 0 .. nk_-1;
# --- Variable and Parameter Declarations ---
var xa{k_ in kSet_};
var Derivative_xa_{k_ in 0..last(kSet_)-1};
var xb{k_ in kSet_};
var Derivative_xb_{k_ in 0..last(kSet_)-1};
var u{k_ in kSet_};
# --- Optimization Problem ----
minimize obj: -(1 - xa[last(kSet_)] - xb[last(kSet_)]);
subject to
# ==== Trial Function Definitions ===
TrialFunction1_ {k_ in 1..last(kSet_)}:
  Derivative_xa_[k_] = (xa[k_] - xa[k_-1])/delta_;
TrialFunction2_ {k_ in 1..last(kSet_)}:
  Derivative_xb_[k_] = (xb[k_] - xb[k_-1])/delta_;
# ======== Differential Algebraic System - Residuals ========
DAEdiff3_ {k_ in 1..last(kSet_)}:
  Derivative_xa_[k_] = u[k_]*(10*xb[k_] - xa[k_]);
DAEdiff4_ {k_ in 1..last(kSet_)}:
  Derivative_xb_[k_] = u[k_]*(xa[k_] - 10*xb[k_]) - (1 - u[k_-1])*xb[k_];
bkwdeuler5_: u[last(kSet_)] = u[last(kSet_)-1];
# ======== DAE Initial Conditions ========
DAEinit6_: xa[0] = 1;
DAEinit7_: xb[0] = 0;
# ======== Operating Constraints ========
OperatingCons8_ {k_ in kSet_}: 0 <= u[k_] <= 1;
# ============ Model Data =============
data;
# ====== OCFE parameters ======
let tf_ := 0.4;
let nFEperSample_ := 1;
let sampleSize_ := 0.025;
# --- Solution ---
option solver ipopt; # Default solver for dynamic optimization problems
solve;
```

gPROMS Output

From the same MLDO code, the following gPROMS simulation code is generated:

```
    {  gPROMS file generated by MLDO }
    # --- gPROMS type declarations ----
    DECLARE TYPE # modelfile:unsignedtype
        unsignedtype = 0.5 : -1E+30 : 1E+30
    END
    # --- gPROMS model ----
    MODEL example
    VARIABLE
        xa AS unsignedtype
        xb AS unsignedtype
        u AS unsignedtype
    EQUATION
        $xa = u*(10*xb - xa);
        $xb = u*(xa - 10*xb) - (1 - u)*xb;
    END
    # --- gPROMS process ----
    PROCESS example_process
        UNIT
            mdl AS example
        EQUATION
           WITHIN mdl DO
               u = 0; # default value, please change.
           END
        INITIAL
            WITHIN mdl DO
                xa = 1;
                xb = 0;
            END
        SCHEDULE
            CONTINUE FOR 0.4
    END
```

When there is insufficient information to generate a working output file, MLDO generally makes assumptions in the code output and informs the user of them. For instance, gPROMS—being simulation language—requires the inputs in this case for the model to be fully specified, but the input trajectories $u(t)$ are typically unknown prior to optimization. In this case, MLDO sets the input trajectory to a constant $u(t) = 0$ in the output and advises the user to supply an appropriate $u(t)$ trajectory. Note also that variable bounds are omitted in simulation mode.

6.1.4 Custom Mathematical Transformations in MLDO

Below is a small sampling of the built-in mathematical transformations that can be applied to an MLDO-based dynamic optimization problem. The advantage of specifying these transformations at the MLDO level is that the user is abstracted from the actual set of constraints generated to deliver the required results. As these custom transformations do not modify the original model, they provide a layer of separation between the mathematical description of a model and its solution strategy. As a convenience feature, they also allow the user to effortlessly switch between different formulations to compare results without re-writing large amounts of code.

Complementarity Constraints

Complementarity constraints are constructs used to formulate non-smooth operators such as $\max(z_1, z_2), \min(z_1, z_2), |z|$, and sgn(z) functions [14]. They are typically written as:

$$0 \leq x \perp y \geq 0 \tag{6.51}$$

where x, y are vectors with non-negative elements, and the $\perp$ operator denotes an operation in which the products of their individual elements are zero (that is, $x_i y_i = 0$). These gives rise to a type of optimization problem known as Mathematical Programs with Complementarity Constraints (MPCC).

These constraints violate regularity properties when posed as-is in a standard nonlinear program (NLP), and must thus be handled specially. There are many different ways of reformulating them in order to make them amenable to solution as NLPs. In MLDO, one can specify the complementarity constraints as an AMPL construct (and use a tailor-made complementarity solver such as IPOPT-C or KNITRO to solve the resulting problem), or allow MLDO to automatically reformulate them into the penalty formulation, which will make them amenable to any nonlinear solver [92],

$$\min \Phi + \sum_i \rho_i (x_i y_i) \tag{6.52}$$

$$\text{s.t. } x \geq 0, y \geq 0 \tag{6.53}$$

where Φ is the original objective, and ρ_i is a sufficiently large user-specified penalty term. MLDO permits the user to toggle between the two options with a single directive in the MLDO code.

Using these complementarity constraints, it is easy to build libraries of formulations that can be accessed with simple MLDO directives. As alluded to above, $\max(z_1, z_2), \min(z_1, z_2), |z|$, and sgn(z) are among the non-smooth functions can be modeled using complementarity constraints.

We have implemented one particular complementarity formulation in MLDO — the Inclusive-OR switching formulation. Suppose we have a time-varying quantity p_k and a time-varying indicator variable $y_k \in [0,1]$. The non-smooth Inclusive-OR switching behavior on the left-hand-side below can be exactly reproduced using the complementarity formulation on the right-hand-side (modified from Baumrucker et al. [14]):

$$\left\{ \begin{array}{ll} y_k = 1, & p_k < \bar{p} \\ y_k = 0, & p_k > \bar{p} \\ y_k \in [0,1], & p_k = \bar{p} \end{array} \right\} \equiv \left\{ \begin{array}{l} 0 \leq y_k \leq 1 \\ p_k - \bar{p} = s_k^0 - s_k^1 \\ 0 \leq s_k^0 \perp y_k \geq 0 \\ 0 \leq s_k^1 \perp (1 - y_k) \geq 0 \\ s_k^0, s_k^1 \geq 0 \end{array} \right\} \tag{6.54}$$

for $k = 0 \ldots k_f$. This formulation can be invoked in MLDO using a single line of code.

Complementarity constraints also allow us to model linear MPC controllers that are embedded within a large-scale plant model. One method of modeling the effects of MPC controllers on a plant is to embed the first-order optimality (Karush-Kuhn Tucker) conditions of the quadratic programs of linear MPCs into a dynamic plant model. MLDO has the ability to assemble the requisite code for these controller equations, a process that includes the linearization of nonlinear dynamic plant into state-space form and the generation of the KKT equations. MLDO was used this capacity in Baker et al. [10].

Hard and Soft Constraints

In control applications, it is often desirable to constrain the rate-of-change of the inputs (u_k, with $k = 0, \ldots, N-1$) to within certain rate bounds. In MLDO, hard-constraints and soft constraints are conveniently specified in a single directive, resulting in:

$$\min \Phi + \rho \underbrace{\sum_{k=0}^{N-2} (u_{k+1} - u_k)}_{\text{soft constraints}} \tag{6.55}$$

$$\text{s.t. } \underbrace{\Delta u_L \le u_{k+1} - u_k \le \Delta u_U}_{\text{hard constraints}}, \quad k = 0 \ldots, N-2 \tag{6.56}$$

where Φ is the original objective, ρ is the move-suppression penalty term, $\Delta u_L, \Delta u_U$ are hard rate-limits on the inputs.

Multi-Scenario Method (Optimization Under Uncertainty)

Suppose there exists a model parameter vector $\tilde{\theta} \in \mathbb{R}^{n_{\tilde{\theta}}}$ whose values are uncertain but bounded, that is, $\tilde{\theta}_L \le \tilde{\theta} \le \tilde{\theta}_U$. In the multi-scenario method, the goal is to solve for a trajectory $u(t)$ that is robust to all realizations of this uncertain parameter vector. In order to do this, we can create discrete realizations of each element of $\tilde{\theta}$ by choosing n_{disc} number of parameters values that lie between their upper bounds ($\tilde{\theta}_U$) and lower bounds ($\tilde{\theta}_L$). This results in a set of scenarios θ_s, where $s = 0 \ldots n_{\text{scen}}$ and the total number of scenarios is $n_{\text{scen}} = n_{\tilde{\theta}} \times n_{\text{disc}}$. A set of parallel models—comprising unique combinations of each uncertain parameter and their discretized values—is solved simultaneously in a single NLP problem to find a single set of inputs $u(t)$ that are feasible for all the realizations of the parametric uncertainty. The multiscenario problem can be stated as follows:

$$\min_{u(t)} \Phi(x_0(t), z_0(t), u(t), \delta_0(t), p_0, \theta_0) \tag{6.57}$$

$$\text{s.t. } \dot{x}_s(t) = f(x_s(t), z_s(t), u(t), \delta_s(t), p_s, \theta_s), \quad x_s(0) = \bar{x}_0 \tag{6.58}$$

$$g(x_s(t), z_s(t), u(t), \delta_s(t), p_s, \theta_s) \leq 0 \tag{6.59}$$

$$h(x_s(t), z_s(t), u(t), \delta_s(t), p_s, \theta_s) = 0 \tag{6.60}$$

$$\text{for } t \in [0, t_f] \text{ and } s = 0 \ldots n_{\text{scen}}$$

Note that only the input trajectory $u(t)$ and the initial value $\bar{x}_0$ are common through all scenarios. We select Scenario 0 (that is, $s = 0$) to represent the nominal case, where the parameters are at their nominal values. These parallel model descriptions typically result in a high-dimensional optimization problem. However, because the all the equations and constraints are identical (apart from the parameters) and repeated, this problem possesses a special structure that can be exploited using specialized linear algebra techniques [67]. The ability to automatically generate these parallel models is built into MLDO.

We wish to point out that because an MLDO model is decoupled from the underlying solution process, one can potentially write code generator that uses the same MLDO model to automatically generate for instance, a multistage stochastic program and/or employ different formulations of the Generalized Benders Decomposition on the stochastic program. This is one of the conveniences of having a model form that is abstracted from the solution formulation.

6.1.5 Miscellaneous MLDO features

Apart from the above, MLDO also includes implementations of ideas that, in our experience, are capable of providing considerable aid in the solution and analysis of dynamic optimization problems.

Closed-Loop Simulations
MLDO can be used to generate code necessary to perform closed-loop (MPC) simulations and to manage the message-passing between non-homogeneous environments.

Because the input/output between the different environments is sequential, that is, a program has to wait for the previous program to complete before it can begin processing, we can employ a very simple message-passing scheme. Currently, a file-based polling scheme (see Fig. 6.3) is used to exchange data between different environments (in our case, AMPL and MATLAB). In this method, all the programs that participate in the simulation are kept open at the same time. They each wait (by continually polling the filesystem[7]) for a specific file to exist. The file is then read into memory and promptly deleted; the information that is read is then processed. The results are then written into a file, which is to be read by the next program in the processing chain.

This file-based polling scheme is distinct from the standard approach in which one program calls another, and reads in the output of the called program that terminates. Although the former requires more memory (since all the programs are held in memory and do not exit at every iteration), it avoids the latter's overhead of having to start various programs repeatedly. The file-based polling scheme is programatically easier to implement. For instance, the solution of the optimization in the current iteration is often a very good initial guess for the optimization in the next iteration. In our scheme, the solution of each optimization problem is simply held in memory. In the standard approach, one would have needed to write a considerable amount of code to properly store the solution in a file[8], and even more code to retrieve it.

The file-based polling scheme also allows new programs to be added to the processing chain very easily. It has proven to be a remarkably simple, intuitive and robust way of structuring a closed-loop simulation that comprise multiple software participants (controller, plant, state estimator, etc.). We acknowledge the file-based polling scheme is not the most efficient means of passing information (because disk I/O can be slow relative to memory I/O); there exists more sophisticated way of passing data through pointers and sockets. However, not all software have an interface for dealing with these, therefore we resort to the lowest common denominator of a file. In our context, the disk access times were negligible compared to the time required for solving the optimization problem, therefore this trade-off was acceptable. Furthermore, the use of files for I/O makes errors reproducible, protects from memory-related crashes, and is portable across different operating systems [37]. File-based I/O is the primary I/O model used in commercial modeling languages such as AMPL and GAMS.

Apart from the message passing, MLDO also generates auxiliary code in the controller model to perform pre- and post-optimal processing, tasks which are often crucial for solving a large-scale nonlinear programming program—an enterprise that straddles the realms of science and art—successfully. These include exception handling and recovery, modifications to the initial guess (back-offs and perturbations), and other practical tasks.

[7]This can be done using a fairly simple and unsophisticated polling method, i.e. entering into an infinite `while` loop in which the termination condition is the successful reading of a file.

[8]Although this may at first appear to be a trivial task, it is not the case for a large-scale optimization problem with a shrinking horizon that affects the variable indices in the problem

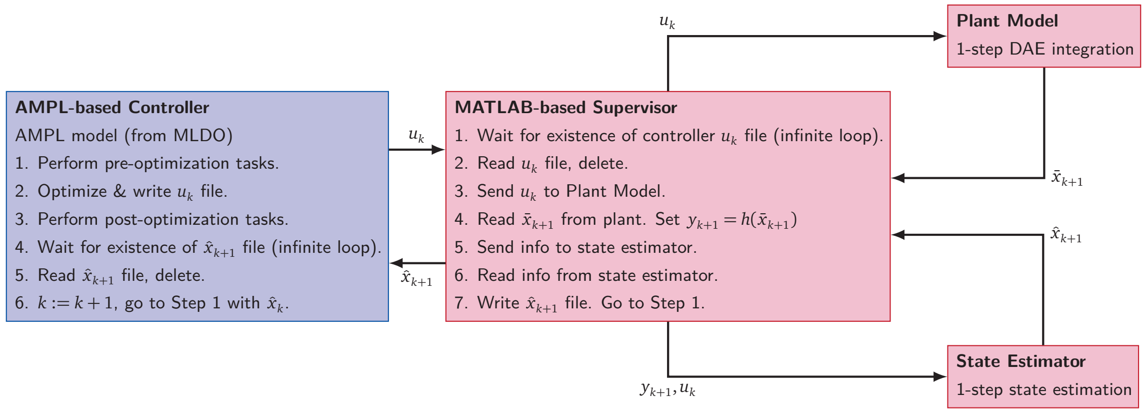

AMPL-based Controller
AMPL model (from MLDO)
1. Perform pre-optimization tasks.
2. Optimize & write u_k file.
3. Perform post-optimization tasks.
4. Wait for existence of $\hat{x}_{k+1}$ file (infinite loop).
5. Read $\hat{x}_{k+1}$ file, delete.
6. $k := k+1$, go to Step 1 with $\hat{x}_k$.
MATLAB-based Supervisor
1. Wait for existence of controller u_k file (infinite loop).
2. Read u_k file, delete.
3. Send u_k to Plant Model.
4. Read $\bar{x}_{k+1}$ from plant. Set $y_{k+1} = h(\bar{x}_{k+1})$
5. Send info to state estimator.
6. Read info from state estimator.
7. Write $\hat{x}_{k+1}$ file. Go to Step 1.
Plant Model
1-step DAE integration
State Estimator
1-step state estimation
u_k
u_k
$\hat{x}_{k+1}$
$\bar{x}_{k+1}$
$\hat{x}_{k+1}$
y_{k+1}, u_k

Figure 6.3: Closed-Loop Simulation Schematic

Visualization with Sparklines

In many large-scale dynamic optimization problems, the number of solution trajectories may range in the hundreds. Frequently, only a few trajectories are of interest, and most users will elect to plot and examine those. There exist situations however where it may be helpful to visualize every single trajectory in a solution set as there may be aberrant behavior in certain unexpected variables. While plotting and examining hundreds of variables is not usually thought of as a practical proposition, there exists a compact visualization method that will enable users to quickly examine trends for a large number of trajectories, whether on the computer screen or on the printed page. Statistician Edward Tufte proposed using compact word-sized graphics called “Sparklines”[108] as an economical and high-data-density means of conveying trend information[9]. MLDO has the ability to generate Sparklines[10] (Fig. 6.4) for solution trajectories. There are many variants of Sparklines. In our implementation, there are 5 basic elements in the representation of a time-varying variable (see Fig. 6.4): 1) The starting value; 2) The pictorial evolution of the trajectory over time; 3) The final value; 4) the minimum value attained in the horizon of interest; and 5) the maximum value attained. In MLDO, the generated Sparklines

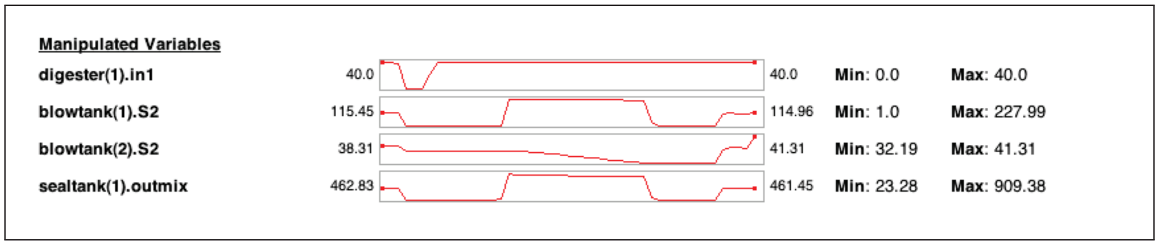


Figure 6.4: Use of Sparklines as compact way for displaying a large no. of trajectories.

[9]It is worth noting that Sparklines support was added to Microsoft’s ubiquitous Excel spreadsheet software in 2010.

[10]MLDO uses the `PIL` (Python Imaging Library) and `matplotlib` libraries to generate Sparklines.

are embedded in a standard HTML document that can be served from any web server. When a mouse pointer is hovered over a Sparkline trajectory, a detailed zoomed floating image of the trajectory appears as a transient pop-up. In this thesis, we used successfully used Sparklines as a means to visualize 200-300 trajectories and to identify issues in the results of dynamic optimization problems.

Automatic Generation of Documentation
Because MLDO code very closely resembles mathematical notation, it can easily be converted to a professionally typeset mathematical document through the intermediary of a typesetting language, $\LaTeX$. This essentially makes MLDO code self-documenting, in that human-readable documentation can be automatically generated from model code. This functionality facilitates the exchange of mathematical models with parties who have no knowledge of MLDO syntax. This also preserves the model for posterity, because although new modeling environments (with new syntax) emerge every few years, the mathematics of a model remain unchanged. All transformations at the mathematical level (such as numerical scaling) are preserved in the documentation generation.

6.2 Nonlinear Modeling Techniques for Dynamic Optimization

Large-scale nonlinear modeling is a challenging endeavor that is usually aided not only by mathematical know-how, but also by engineering intuition. This is particularly true when modeling for optimization applications, where the problem search space can be very large, and numerical pitfalls abound owing to nonlinearity, non-uniqueness and non-convexity. Furthermore, the problem becomes considerably more complicated when dynamics are involved.

In nonlinear optimization modeling, certain formulations are known to have more favorable numerical properties than others in a given context. Knowledge of such formulations is often crucial to finding a solution for a nonlinear dynamic model. In this section, we describe several general formulation techniques (often unrecorded in the literature) used during the model development phase, as well as rules-of-thumb for nonlinear modeling.

6.2.1 Variable and Equation Scaling

The task of scaling in mathematical modeling is to ensure that the absolute values of its equations, constraints, and variables (and when possible, their non-zero derivatives as well) are roughly on the same order of magnitude. This is done to avoid ill-conditioning and to ensure that the solver's numerical tolerances can be reasonably met. This task not only requires intimate knowledge of the ranges of values that are admissible, but also the average values that are typically encountered. For nonlinear models especially, scaling is neither a straightforward nor a well-defined exercise, because it can be difficult to have foreknowledge on the evolution of and interaction between the variables, equations and constraints during the solution process. Owing to this, scaling algorithms that attempt to automatically determine good scaling factors for a nonlinear system are heuristics at best. In many instances, a modeler's engineering insights may yield more appropriate scaling factors and/or reformulations that yield better scaled models. While it is difficult to define what constitutes good scaling (especially for nonlinear nonconvex problems), rules of thumb are offered in Drud [38] and Wächter [113]. Here are examples of the types of scaling methods we employed during the modeling process:

1. *Equation/Constraint scaling*
   This is the simplest method of scaling and is often the first recourse. In this work, our goal is to get the LHS or RHS of every equation and constraint into the range of roughly $[0, 100]$ by multiplying both sides of the equation by a constant scaling factor. For instance, consider the following example in which y is in the order of magnitude of 1×10^6:

   $y = x^2 + z^3$ (unscaled; LHS, RHS magnitude in the order of 1×10^6)

   $\downarrow$

   $(1 \times 10^{-4})y = 1 \times 10^{-4}(x^2 + z^3)$ (scaled, LHS, RHS magnitude in the order of 1×10^2)

2. *Variable scaling*
   The scaling of variables, though sometimes neglected, is as important as equation scaling. For instance, suppose we had a variable $x \in [a, b]$, and the nonlinear equation:

   $$y = f(x)$$

   Suppose $x \in [a, b]$ is deemed to be poorly scaled and we wish work in terms of a new variable $\bar{x}$ whose domain is [c,d]. The following simple formula can be used:

   $$x = \frac{(b-a)(\bar{x}-c)}{d-c} + a$$

Therefore, the original equation now becomes:

$$y = f\left(\frac{(b-a)(\bar{x}-c)}{d-c} + a\right)$$

where all instances of x are replaced by the scaling expression. As an example, in our work we commonly encounter variables that represent fractions, compositions or other quantities that lie in the domain [0,1]. It is frequently helpful to transform their domains to [0,100] (and treat them as percentages). Consider this familiar chemical engineering component balance:

$$\text{(unscaled)} \begin{cases} F_1x_1 + F_2x_2 = F_3x_3 \\ x_1, x_2, x_3 \in [0,1] \end{cases} \rightarrow \text{(scaled)} \begin{cases} F_1(0.01\bar{x}_1) + F_2(0.01\bar{x}_2) = F_3(0.01\bar{x}_3) \\ \bar{x}_1, \bar{x}_2, \bar{x}_3 \in [0,100] \end{cases}$$

In the scaled case, the $\bar{x}$ variables vary in the interval [0,100] during optimization. This type of simple scaling has been found to be beneficial to the conditioning of the problem, especially when some of the x variables tend toward small values. After the solution is found, the reported results (which are in terms of $\bar{x}$) can be transformed back into x values. In some cases where it is undesirable for the independent variable x to numerically cross 0, this type of scaling can be used to project its domain into an interval where only positive or negative values are admissible.

3. *Derivative scaling*

Derivative scaling in ODEs/DAEs is somewhat more subtle. For example, suppose there is a derivative term dx/dt whose values lie in the range $[0.001, 0.008]$ s^{-1} (a tiny and potentially numerically problematic range). A rescale is possible by applying the chain rule to change the time units of the derivative. One can pre-multiply the derivative (and only the derivative) with a scaling factor, as follows:

$$\text{(unscaled)} \begin{cases} \frac{dx}{dt} = f(x) \\ \frac{dx}{dt} \in [0.001, 0.008] \end{cases} \quad \rightarrow \quad \text{(scaled)} \begin{cases} 3600\frac{dx}{dt} = f(x) \\ \frac{dx}{dt} \in [3.6, 28.8] \end{cases}$$

Note that the right-hand-side of the scaled equation has not changed, merely the magnitude of the derivative. The units for dx/dt is now h^{-1} (3600 s = 1 h), and it evolves within the more reasonable range of $[3.6, 28.8]$.

4. *Objective function scaling*
   If the average magnitude of the objective function is known, it is frequently helpful to perform a manual scaling of the objective. In a constrained optimization problem, the objective function enters as a term in the Lagrangian expression, whose derivative is used for obtaining first order optimality conditions. The goal therefore is to select a scaling factor so that the objective value will have numerically the same approximate magnitude as the constraints. Table 6.4 compares the Lagrangian of an unscaled objective and a scaled one. Most modern solvers will attempt to perform some form of automatic objective

	Unscaled	**Scaled**
Problem	$\min \Phi(x)$ s.t. $h(x) = 0, g(x) \leq 0$	$\min \rho\Phi(x)$ s.t. $h(x) = 0, g(x) \leq 0$
Lagrangian	$L = \Phi(x) + \sum_i \lambda_i h_i(x) + \sum_i \mu_j g_j(x)$	$L = \rho\Phi(x) + \sum_i \lambda_i h_i(x) + \sum_i \mu_j g_j(x)$

Table 6.4: Unscaled and scaled objective functions, where ρ is a constant objective scaling factor, and μ, λ are KKT multipliers.

   scaling based on heuristics, but modelers can often pick a more suitable scaling factor ρ if they have physical or numerical insights into the model.

5. *State and parameter scaling (for estimation)*
   In order to have a well-conditioned covariance matrix for estimation, it is important to scale the states x and the parameters θ (normally constant, but are being updated during estimation), such that all the variances are approximately of the same magnitude.

6.2.2 Numerical Formulations

We present here a list of formulations and state the advantages of particular formulations in terms of their amenability to numerical optimization.

1. *Products preferred to quotients*
   In general, division by a variable are unfavorable because in the course of the solution iterations, a division by 0 may occur. Table 6.5 shows an example of a product vs. a quotient formulation and the properties of each. While both formulations are mathematically equivalent, the product formulation is preferred because it does not admit undefined values during iteration.

   With respect to the quotient formulation, one can arguably write a constraint $x \geq \epsilon$ to avoid the singularity. However, the question of how small ϵ ought to be is not always

	Quotient	**Product**
Equation	$y = \frac{1}{x}$	$xy = 1$
Expression	$f(w) = y - \frac{1}{x}$ where $w = [x, y]^{\mathrm{T}}$	$f(w) = xy - 1$ where $w = [x, y]^{\mathrm{T}}$
Gradient	$\nabla f_w = [\frac{1}{x^2}, 1]^{\mathrm{T}}$	$\nabla f_w = [y, x]^{\mathrm{T}}$
Notes	• Expression and gradient are undefined at $x = 0$.	• Expression and gradient do not succumb to division by 0 errors.

Table 6.5: Example of product formulation (preferred) vs. quotient formulation

straightforward and has to be decided on a case-by-case basis. In our model, many of the expressions involving compositions and ratios are converted from quotient to product form in our model. For instance, in the computer codes we rewrite eqn. A.60 (restated below) in the knotter model as follows:

$$x_w = \frac{F_{in2W}}{F_{in2}} \text{ (quotient) } \rightarrow F_{in2} \cdot x_w = F_{in2W} \text{ (product)} \tag{6.61}$$

2. *Sparse formulations preferred to dense*

   Sparser formulations are generally preferred to dense ones when: (1) the solution algorithm is designed to accommodate/exploit the sparsity structure, and; (2) proper care is taken to avoid introducing linearly dependent equations while reformulating the system for added sparsity. For instance, this expression:

$$f = \frac{2}{x^2} + 5(\exp(y) + \log(z))^2 - 3 = 0 \tag{6.62}$$

can be equivalently expressed as:

$$f_1 = r_1 x^2 - 2 = 0 \tag{6.63}$$

$$f_2 = r_2 - \exp(y) = 0 \tag{6.64}$$

$$f_3 = \exp(r_3) - z = 0 \tag{6.65}$$

$$f_4 = r_4 - (r_2 + r_3) = 0 \tag{6.66}$$

$$f_5 = r_5 - 5r_4^2 = 0 \tag{6.67}$$

$$f_6 = r_1 + r_5 - 3 = 0 \tag{6.68}$$

where $r_i, i = 1\ldots5$ are dummy variables. Although the latter system of equations (6.63 – 6.68) is much larger than 6.62, its derivatives are simpler and more mildly nonlinear.

Furthermore, a sparser formulation permits the modeler to scale and add bounds to *individual* constituent expressions (or intermediate variables) in order to obtain a better conditioned system. This is immensely useful when the variables/expressions in an equation are of vastly different magnitudes. Sparse formulations often lead to sparser Jacobian and Hessian matrices. Many specialized linear solvers (especially those used in optimization solvers) are often able to take advantage of the sparse structure of these matrices[11].

The above reformulations and rules-of-thumb were borne out of our numerical experience with nonlinear models. However, there may exist exceptional situations (for instance, numerical effects peculiar to a solver's algorithm) where the reformulations may give poorer than expected performance. In such situations, the modeler's intuition and judgment may be called upon to select an appropriate alternate formulation for the problem at hand.

6.2.3 Modeling techniques specific to chemical engineering models

Material balances: component flow rates preferred to mass fractions

In certain situations, a component flowrate formulation may have advantages over mass fraction formulations, particularly when the mass fraction *is not required* to calculate another quantity (because the former allows us to write linear equations rather than bilinear ones). Consider the following examples:

$$\text{(mass fractions)}\begin{cases} F_1 = F_2 + F_3 \\ F_1 x_1^A = F_2 x_2^A + F_3 x_3^A \\ x_1^A + x_1^B = 1 \\ x_2^A + x_2^B = 1 \\ x_3^A + x_3^B = 1 \end{cases} \rightarrow \text{(component flows)}\begin{cases} F_1^A = F_2^A + F_3^A \\ F_1^B = F_2^B + F_3^B \end{cases}$$

where F_i^j = component flowrate of stream $i = 1,2,3$ and component $j = \{A,B\}$, F_i = flowrate of stream i, and x_i^j = mass fraction of stream $i = 1,2,3$ and component $j = \{A,B\}$. In the

[11]In IPOPT, a sparse linear solver (in our case, HSL MA57 [40]) is used which exploits the structure of large-scale sparse symmetric linear systems.

component flows formulation, the system of equations is linear. In the mass fraction formulation, there exist bilinear terms, and some of the mass fractions could potentially take very small values, which can be numerically problematic (though this problem can be partially ameliorated using the scaling techniques described prior). In the component flowrate formulation, if a particular mass fraction quantity is required, it can be calculated as follows:

$$x^i = \frac{F^i}{\sum_{i=1}^{n_c} F^i} \tag{6.69}$$

where x^i is the mass fraction of component i, F^i is the flowrate of component i, and n_c is the number of components in the stream.

Tank modeling

Instead of modeling tank levels in terms of changes in scale-dependent properties (height, volume, or mass) as is typically done, tracking the level changes in terms of percentages can bring about several advantages. With a percentage model, one obtains a well-scaled derivative, scale invariance in the differential state variable (it varies in the [0,100] range regardless of tank size). Consider the following example, where the level of a tank is allowed to vary between 30"–270":

$$\text{(textbook method)} \begin{cases} \rho A \frac{dh}{dt} = F_{in} - F_{out} \\ 30 \leq h \leq 270 \end{cases} \quad \text{(percentage based)} \begin{cases} \rho A h_{total} \frac{1}{100} \frac{dH}{dt} = F_{in} - F_{out} \\ 10\% \leq H \leq 90\% \\ h_{total} = 300 \end{cases}$$

where ρ = density, A = area of tank base, F_{in}, F_{out} = inlet and outlet flowrates respectively, h = tank content levels, H = level as a percentage, h_{total} = maximum level. If the values of h (the tank level) span a large range, the advantages of the percentage-based formulation will be particularly pronounced as because the numerical variation of H is always in the [0,100] range. Also, the maximum and minimum level constraints can be conveniently specified as percentages instead of in absolute height/volume/mass numbers.

6.2.4 Understanding solver algorithms

A good understanding of the solvers in one's toolkit is crucial for the successful solution of large-scale nonlinear dynamic optimization models. Different solvers implement different algorithms, and it is frequently necessary to exploit the peculiarities of a particular solver to one's advantage.

In this work, we used several optimization solvers. For the continuous optimization problems, IPOPT [114] and CONOPT [6] were employed. For the mixed-integer (discrete) optimization problems, CPLEX [59] and Bonmin [24] were used. IPOPT implements a primal-dual interior-point method for nonlinear optimization. CPLEX is a well-established solver for linear and convex programs (both continuous and discrete), while Bonmin implements several algorithms for mixed-integer nonlinear programs. Bonmin was used in branch-and-bound mode, with IPOPT as a subsolver for the relaxed nonlinear programs.

In this section, we will present some general insights pertaining to the use of two major classes of algorithms for solving nonlinear programs: active-set (e.g. GRG, as implemented in CONOPT) and interior point (e.g. IPM, as implemented in IPOPT).

1. *Improving performance by adding bounds and simple constraints*
   Active-set methods rely on enumerating inequality constraints in order to identify the active-set, and the complexity of a problem grows exponentially with the number of inequality constraints (see Appendix C.3). When modeling for an active-set solver, a modeler will want to be parsimonious with the inequality constraints.

   Interior point methods however have polynomial complexity and in our experience, they appear to benefit from having more inequality constraints than fewer. The inequality constraints are translated into bounds, which are subsequently incorporated into the barrier term. This aids the interior point optimizer in narrowing down the feasible region and converging to the solution.

2. *Feasibility of path*
   Feasible-path algorithms follow a path where all the intermediate iterates are feasible points to the original problem, once a feasible point is found. In this work, we chose to use CONOPT for MPC studies owing to the fact that it is a feasible-path solver. At every MPC iteration, a multi-tiered optimization problem is solved. For multi-tiered optimization, the advantages of a feasible path solver are particularly pronounced because the algorithm exploits the fact that the solution of every optimization tier is a feasible initial guess for the next tier. Once the solution of Tier 1 is found, CONOPT is able to converge very quickly to the solutions in subsequent tiers.

   IPOPT on the other hand implements an interior-point method which always ensures feasibility with respect to inequality constraints, but does not preserve feasibility with respect to equality constraints during iteration. This means that when given a feasible/optimal initial guess, it can potentially follow a path which leads it to a region where there are no feasible solutions satisfying the equality constraints. We can discourage excursions from feasibility by setting an upper bound for the allowable constraint violation during iterations (though the `theta_max_fact` option).

Constraint Violation Penalty Trick. In both the CONOPT and IPOPT solvers, an effective means for deterring excursions from a feasible initial guess is to add a constraint violation penalty in the objective (Chachuat [29]). An easy reformulation can carried out as follows:

$$\begin{array}{rl} & \text{(Original)} \\ & \min_x f(x) \\ \text{s.t.} & h(x) = 0 \\ & g(x) \le 0 \end{array} \quad \longrightarrow \quad \begin{array}{rl} & \text{(Reformulated)} \\ & \min_{x,r} f(x) + ||r||_2^2 \\ \text{s.t.} & h(x) = r \\ & r = 0 \\ & g(x) \le 0 \end{array}$$

where $x \in \mathbb{R}^{n_x}$, $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}$, $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_h}$, and $g : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_g}$. The vector $r \in \mathbb{R}^{n_h}$ contains the residuals of the equality constraints $h(x)$. We note that at the solution point, the residual vector $r = 0$, therefore the solution of the original problem is not altered by this reformulation. The constraint violation penalty ($||r||_2^2$) merely provides the optimizer a disincentive from taking large steps that increase the amount of constraint violation in the equality constraints.

3. *Warm-starting in MPC applications*

   In MPC-based applications, the solution of one iterate usually provides a good initial guess to the next. It is desirable for the optimizer to take advantage of this as much as possible [25, 125]. Interior-point methods often have difficulty exploiting a good initial guess that is close to the solution [49, 46]. In this respect, active-set methods such as GRG in CONOPT usually do much better when given an initial guess close to the solution, provided that the active-set of the guess and the actual solution do not differ substantially [123].

   That said, IPOPT can be made more amenable to warm starts. By default, IPOPT perturbs the initial guess so that it is sufficiently in the interior defined by the bounds. If a good initial guess is given, the magnitude of this perturbation can be reduced (through `bound_frac` and `bound_push` options). As well, the initial size of the barrier parameter μ can be made more conservative (by reducing `mu_init`).

4. *Non-unique optimization problems*
   The problems we deal with in this work tend to have non-unique solutions. To tackle this, we use a hierarchical optimization approach (by way of our multi-tiered optimization procedures) to regularize the problem toward a unique solution. Nevertheless, the problem in the first-tier tends not satisfy Sufficient Second-order Optimality Conditions (SSOCs). One of the indicators of this is that the reduced Hessian of the problem at the solution point is non-positive definite (in the case of IPOPT—which we use to solve to open-loop problems and to arrive at a good initial guess for the closed-loop ones—this corresponds to projection of the augmented Hessian of the Lagrangian $W_k + R_k$ to the constraint Jacobian J_k not having the property of positive definiteness; see eqn. C.33 in Appendix C.3).

   Given this scenario, IPOPT employs a regularization approach in which the matrix in the augmented KKT system is perturbed by a constant value $\delta_x > 0$ along its diagonal (using a trial-and-error inertia correction heuristic) until the desired matrix inertia[12] is achieved [114]. This permits IPOPT to iterate toward a point that satisfies KKT conditions.

   The implication of this is that for a non-unique problem (e.g. Tier 1 of a multi-tiered optimization problem), IPOPT may still converge to a solution despite the lack of SSOC fulfillment. However, it may terminate at a critical point that may not be an optimum, but still satisfies the KKT (first-order optimality) conditions up to some numerical tolerance[13]. We would argue that given the state-of-the-art, this is an acceptable situation in practice. Further work is required to develop methods for handling non-unique optimization problems. (Note: one may argue for a regularization penalty term to be added to the objective. However in our experience, for this to work, it is sometimes necessary to make the penalty weight sufficiently large to induce a unique optimum. Unfortunately a very large penalty can easily overwhelm the original objective function, which distorts the meaning of the original optimization problem.)

   Non-unique problems can cause IPOPT to struggle to identify the solution. It can slow convergence considerably because IPOPT has to perform frequent regularization. If we know *a priori* that a problem is non-unique and that the solution obtained will be a KKT point, we can reduce the number of iterations required to reach the optimality tolerances by scaling down the objective function with a small scaling factor.

[12]The inertia of a symmetric matrix is defined as $I = (i_p, i_n, i_z)$ where each element of the tuple denotes the number of postive, negative and zero eigenvalues respectively. The matrix is positive definite when $i_p = i_n$.

[13]We can easily inspect IPOPT's output to verify that this is the case: if the log regularization term `lg(rg)` (that is, $\log_{10} \delta_x$; see [114]) is not null at the solution point, then the solution may not be true minimizer.

CONOPT however converges easily even when Tier 1 is non-unique. This can perhaps be attributed to the fact that the GRG algorithm works in reduced (though not necessarily orthogonal) space, where the effects of non-uniqueness are moderated.

5. *Heuristic for recovering from solver failures*
When performing successive nonconvex dynamic optimization in shrinking horizon MPC mode, the numerical properties (Jacobian conditioning, matrix structures, problem size, problem structure after presolve etc.) of the problem instance at each iteration changes. The numerical reality is that non-feasible path optimizers can occasionally terminate at an infeasible point. We found the following heuristic procedure (for randomly perturbing the current initial guess) to be effective when convergence difficulties are encountered [29, 24]. Suppose we wished to perturb the current initial guess of the following NLP upon convergence failure:

$$x = \arg\min_{x}\{f(x)|h(x) = 0, g(x) \leq 0, x \in \mathbb{R}^{n_x}\} \tag{6.70}$$

The perturbation procedure is as follows:

(a) Let initial guess be x_{init} (in MPC, this is typically the solution from the previous iteration).

(b) Solve optimization problem in eqn. 6.70.

(c) **if** problem fails to converge,

i. Let $x_{\text{new}} := x_{\text{init}} + h \cdot N(0,1) \cdot x_{\text{init}}$

ii. **for** $i = 1 \ldots n_x$, **if** $x^i_{\text{new}} > x^i_{\text{ub}}$, then $x^i_{\text{new}} := x^i_{\text{ub}} - \epsilon$

iii. **for** $i = 1 \ldots n_x$, **if** $x^i_{\text{new}} < x^i_{\text{lb}}$ then $x^i_{\text{new}} := x^i_{\text{lb}} + \epsilon$

iv. Let $x_{\text{init}} := x_{\text{new}}$. Go to Step 2.

else return solution.

where i = vector element, $x_{\text{new}} \in \mathbb{R}^{n_x}$ = new initial guess; $x_{\text{init}} \in \mathbb{R}^{n_x}$ = initial guess in current iteration; h = perturbation magnitude (e.g. 1×10^{-5}); $N(0,1)$ = normally distributed random number with mean 0 and variance 1; $x_{\text{ub}}, x_{\text{lb}} \in \mathbb{R}^{n_x}$ = upper and lower bounds of x – these can typically be obtained automatically through the modeling language's presolve phase; ϵ = a backoff factor, typically set to approximately equal to the constraint feasibility tolerance on the optimizer (e.g. 1×10^{-6}).

6.3 Conclusion

In this chapter, we provided an overview of the broad concepts behind the design of a Modeling Language for Dynamic Optimization, MLDO. The multi-part code generation process, which is central to MLDO's operation, was outlined with examples illustrating the principles of parsing, transforming and assembling code from a canonical description of a mathematical problem. To provide a flavor of the type of transformations that can be carried out with a code generation infrastructure, we listed several notable mathematical transformations that have been implemented in MLDO. Finally, some auxiliary helper features of MLDO were described. We also discussed numerical modeling formulations and solution techniques used in this work, and numerical experiences with various formulations are documented.

Chapter 7

Conclusion

The aim of this work is to develop a suite of mathematical formulations for computing and implementing control policies on plants under partial shutdown. To reach this goal, we employed a number of techniques such as dynamic optimization, model predictive control, state/parameter estimation and model code generation.

The chief contributions of this work are as follows:

1. **Dynamic optimization of partial shutdowns**
   A configurable multi-tiered economics-driven dynamic optimization formulation was presented. The purpose of this formulation is to compute the requisite optimal control inputs to drive the process back to nominal operation after the occurrence of an unplanned partial shutdown. This procedure involves the coordination of production flowrates, recycles and inventories. The problem was cast as DAE optimization problem and solved using the simultaneous (full-discretization) method. The multi-tiered approachs permits multiple competing objectives to be prioritized with specified trade-offs.

2. **A model-based control framework for partial shutdowns**
   In order to implement partial shutdown policies in presence of uncertainty (e.g. plant-model mismatch, disturbances, noise) in the plant, a feedback control framework was adopted. The proposed controller is a transient one and has a limited mandate of managing the plant during and after partial shutdowns; once a plant is restored from a partial shutdown to its nominal operating point, plant control is handed back to the nominal control system. In the design of this controller, we have adopted the Model Predictive Control (MPC) paradigm, where an open-loop dynamic optimization problem is solved at every

sampling instance. This controller also permits parametric feedback, that is, the updating of model parameters such as "estimated downtime", which readjusts the shutdown period in the optimization problem, and allows the trajectories to be re-optimized in the next time step.

3. **Discrete formulations for induced shutdowns**
   We present efficient discrete formulations for modeling discontinuities associated with partial shutdowns. These formulations are parsimonious with respect to the number of binary variables required. Formulations are created for modeling shutdown thresholds, capturing the fixed and variable costs of a shutdown, and to enforce minimum shutdown durations. These are used in concert to handle induced shutdowns, which is a type of shutdown resulting from another shutdown. The final problem takes the form of a Mixed-Integer Dynamic Optimization (MIDO) problem.

4. **Discrete formulations to determine the minimum restoration time**
   A multi-tiered MIP-based minimum restoration time formulation attempts to compute the least possible it time it takes for a system under partial shutdown to be restored to its nominal state, while obeying all constraints. In this formulation, the restoration duration is optimized first. If there are degrees-of-freedom remaining, the economics and input movements of the control trajectories are optimized. Simple MIP cuts are given for improving the solution efficiency.

5. **State and Parameter Estimation using a Constrained Unscented Kalman Filter**
   Novel configurations of Constrained Unscented Kalman Filter algorithms were used for state and parameter estimation of a large-scale differential algebraic system. Two main strategies were employed: simple parameter estimation and joint estimation. An iterative refinement procedure was proposed in the simple parameter estimation strategy to improve estimator performance.

   The state and parameter estimation algorithms were applied in the context of performing optimal control on a plant undergoing partial shutdowns. A reduced estimation problem was solved during the shutdown period to account for the missing states, parameters and measurements corresponding to the offline units. Mismatched parameters and fictitious disturbances are recursively estimated and updated in the controller model's in order to correct its predictions.

   The control algorithm incorporates a multi-tiered dynamic optimization problem. A dynamic feasibility tier ensures that in the event of large disruption in the process (such as a shutdown), the terminal constraints and parameter estimates arriving from a one-step estimator are feasible throughout the model prediction horizon. This represents a general technique for handling terminal constraints and one-step ahead estimates in MPC control optimization problems.

6. **A Modeling Language for Dynamic Optimization, MLDO**
   An in-house modeling language was created to aid the modeling, solution, and analysis of dynamic optimization problems. The MLDO system takes a canonical mathematical description of a dynamic optimization problem and generates code in other programming/modeling languages. As well, it generates the requisite constraints and equations for orthogonal collocation on finite elements and for the implicit Euler method, and supports other manners of mathematical transformations. This permits the user to do rapid prototyping and experiment with different formulations without writing an excessive amount of boilerplate code.

7. **Modeling of Kraft fiber line**
   The algorithms and formulations in this thesis were tested on a model of the Kraft pulp and paper process, in particular the fiber line subset of the full plant. The model captures details related to buffer levels and compositions, which are the quantities of interest for inventory optimization.

7.1 Recommendations for Further Work

Several broad candidate areas for further exploration were identified. They are:

1. **Energy optimization**
   Our primary concern in this work was optimal inventory and production management, therefore our models are driven by material balances. The framework we have proposed, however, is general and may be used to include energy considerations, which may influence the economics of a partial shutdown restoration considerably.

2. **Inclusion of startup and shutdown dynamics**
   We have considered the macro-view of a partial shutdown in which the actual startup and shutdown procedure of a unit is abstracted. An extension to this work would be one that takes the micro-view and investigates the explicit procedure for unit startups and shutdowns. The procedure usually takes the form of a sequence of manual actions

prescribed by the operator based on experience or some standard written procedure. If the procedure is a simple one with time-based switching and no logical paths, it can be modeled using a multi-stage DAE model. However, if the procedure relies on state-based switching with proposition `IF-THEN-ELSE` logic dictating the mode transitions, then it can be modeled using any number of standard hybrid dynamic model paradigms. Such a model can then be cast into an optimization framework to obtain optimal control inputs for performing the startup and shutdown.

In principle, the startup and shutdown problem can be decoupled from the macro problem that we are considering in this thesis. The main piece of information required in the macro-view model is the shutdown estimate, which the micro-view model can provide. The micro-view model on the other hand only requires the inlet flowrate and composition information prior to shutdown. In some cases, when the startup and shutdown procedure interacts significantly with the inventories, then iterating between the macro and micro model may be called for.

3. **Dynamic optimization using a multiple shooting method**
   One of the biggest challenges in the use of a full-discretization (simultaneous) method for dynamic optimization is finding a suitable initial guess for the resulting nonlinear program. Partial shutdown problems in particular contain trajectories that change drastically, therefore it is difficult to make any prognostications about the shape of the state profile at the optimal solution point. We believe that the multiple shooting method may be a good candidate for addressing this issue. It inherits many of the advantageous properties of the simultaneous and sequential methods, and its incorporation of a DAE solver for each discrete element may reduce the need for a good initial guess for the entire trajectory profile. Its computational demands are potentially higher due to the need to invoke a DAE solver at every optimization iteration, but this may be offset by the fact that it is an inherently parallelizable algorithm and stands to benefit from multiprocessor machines (as opposed to the simultaneous method which can only use a single processor, unless the solver is modified to use parallelized linear algebra routines customized for the problem structure at hand).

4. **Regularization of non-unique optimization problems**
   In this work, we presented a multi-tiered approach for systematically prioritizing objectives in a non-unique optimization problem. This is done by performing multiple optimization operations that successively “choose” specific solutions within the space of non-unique optimal solutions based on some criteria. However, the first tier problem, being non-

unique, does not fulfill Sufficient Second-order Optimality Conditions (SSOCs). This can cause the solver to struggle in identifying a solution. We believe that reduced methods that work in a well-conditioned subspace (perhaps based on QR factorization or Singular Value Decomposition) may yield elegant solution methods for dealing with this situation.

List of References

[1] O. Abel and W. Marquardt. Scenario-integrated modeling and optimization of dynamic systems. *AIChE Journal*, 46(4):803 – 823, 2000.

[2] V. Adetola and M. Guay. Integration of real-time optimization and model predictive control. *Journal of Process Control*, 20(2):125 – 33, 2010.

[3] R. J. Allgor and P. I. Barton. Mixed-integer dynamic optimization. *Computers and Chemical Engineering*, 21(SUPPL. 1):S451 – S456, 1997.

[4] R. J. Allgor and P. I. Barton. Mixed-integer dynamic optimization i: Problem formulation. *Computers and Chemical Engineering*, 23(4-5):567 – 584, 1999.

[5] B. Allison. Averaging Level control for Surge Tanks in Series. In *Control Systems Conference*, pages 159–167. Technical Association of the Pulp and Paper Industry Press, June 2004.

[6] ARKI Consulting and Development. *Documentation for GAMS/CONOPT, available at* http://www.gams.com/docs/solver/conopt.pdf, 2002.

[7] U. M. Ascher and L. R. Petzold. Projected implicit Runge-Kutta methods for differential-algebraic equations. *SIAM Journal on Numerical Analysis*, 28(4):1097 – 1120, 1991.

[8] M. P. Avraam, N. Shah, and C. C. Pantelides. Modelling and optimisation of general hybrid systems in the continuous time domain. *Comput. Chem. Eng. (UK)*, 22:221 – 8, 1998.

[9] R. Baker and C. L. E. Swartz. Rigorous handling of input saturation in the design of dynamically operable plants. *Industrial and Engineering Chemistry Research*, 43(18):5880 – 5887, 2004.

[10] R. Baker, C. L. E. Swartz, and Z. Chong. Integrated Design and Control with Constrained Predictive Control – an MPCC formulation. *(manuscript complete)*, 2008.

[11] A. K. S. Balthazaar. Dynamic optimization of multi-unit systems under failure conditions. Master's thesis, McMaster University, 2005.

[12] V. Bansal, J. D. Perkins, and E. N. Pistikopoulos. A Case Study in Siumltaneous Design and Control Using Rigorous, Mixed-Integer Dynamic Optimization Models. *Industrial and Engineering Chemistry Research*, 41(4):760–778, February 2002.

[13] V. Bansal, V. Sakizlis, R. Ross, J. D. Perkins, and E. N. Pisikopoulos. New algorithms for mixed-integer dynamic optimization. *Computers and Chemical Engineering*, 27:647–668, 2003.

[14] B. T. Baumrucker, J. G. Renfro, and L. T. Biegler. MPEC problem formulations and solution strategies with chemical engineering applications. *Comput. Chem. Eng. (UK)*, 32(12):2903 – 13, 2008.

[15] E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In J. Laurence, editor, *Proceeedings of 5th intl. conf. on operational research*, pages 447–454. Tavistock Publications, London, 1970.

[16] V. M. Becerra, Z. H. Abu-El-Zeet, and P. D. Roberts. Integrating predictive control and economic optimisation. *IEE Colloquium (Digest)*, 10(96):11 – 15, 1999.

[17] V. M. Becerra, P. D. Roberts, and G. W. Griffiths. Applying the extended kalman filter to systems described by nonlinear differential-algebraic equations. *Control Engineering Practice*, 9(3):267 – 281, 2001.

[18] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica (UK)*, 35(3):407 – 27, 1999.

[19] B. W. Bequette. Non-linear model predictive control: A personal retrospective. *Canadian Journal of Chemical Engineering*, 85(4):408 – 415, 2007.

[20] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575 – 593, 2002.

[21] L. T. Biegler and I. E. Grossmann. Retrospective on optimization. *Computers and Chemical Engineering*, 28(8):1169 – 1192, 2004.

[22] L. T. Biegler and I. B. Tjoa. Catalyst mixing for packed bed reactor. In M. Morari and I. E. Grossmann, editors, *CACHE Process Design Case Studies, vol. 6*. CACHE, 1991.

[23] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress, Budapest.*, 1984.

[24] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuejols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. Technical report, IBM Research Division, 2005.

[25] D. Bonvin. Optimal operation of batch reactors - a personal view. *Journal of Process Control*, 8(5-6):355 – 368, 1998.

[26] M. S. Branicky. Issues and subtleties in hybrid control systems. In *IEEE International Symposium on Intelligent Control - Proceedings*, pages 336 – 341, 1996.

[27] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comput. (USA)*, 15(6):1467 – 88, 1994.

[28] J. J. Castro and F. J. Doyle III. Plantwide control of the fiber line in a pulp mill. *Industrial and Engineering Chemistry Research*, 41(5):1310 – 1320, 2002.

[29] B. Chachuat. Personal correspondence.

[30] B. Chachuat, A. B. Singer, and P. I. Barton. Global mixed-integer dynamic optimization. *AIChE Journal*, 51(8):2235 – 2253, 2005.

[31] B. Chachuat, A. B. Singer, and P. I. Barton. Global methods for dynamic optimization and mixed-integer dynamic optimization. *Ind. Eng. Chem. Res. (USA)*, 45:8373–8392, 2006.

[32] Z. Chong and C. L. E. Swartz. A modeling language for dynamic optimization. In *56th CSChE Canadian Chemical Engineering Conference, Sherbrooke QC*, 2006.

[33] J. E. Cuthrell and L. T. Biegler. On the optimization of differential-algebraic process systems. *AIChE J. (USA)*, 33(8):1257 – 70, 1987.

[34] C. R. Cutler and B. L. Ramaker. Dynamic matrix control-a computer control algorithm. *1980 Joint Automatic Control Conference*, I:WP5–B/6 pp. –, 1980.

[35] G. De Souza, D. Odloak, and A. C. Zanin. Real time optimization (RTO) with model predictive control (MPC). *Computers and Chemical Engineering*, 34(12):1999 – 2006, 2010.

[36] M. Diehl, R. Amrit, and J. B. Rawlings. A lyapunov function for economic optimizing model predictive control. *Automatic Control, IEEE Transactions on*, 56(3):703 –707, 2011.

[37] A. S. Drud. Replaceable solvers in GAMS.

[38] A. S. Drud. *GAMS: The Solver Manuals (CONOPT)*. GAMS Development Corporation, 2004.

[39] J. F. H. Dubé. Pulp mill scheduling: Optimal use of storage volumes to maximize production. Master's thesis, McMaster University, 2000.

[40] I. S. Duff. MA57 – A new code for the solution of sparse Symmetric Definite And indefinite Systems, 2002.

[41] S. Engell. Feedback control for optimal process operation. *Journal of Process Control*, 17(3):203 – 19, 2007.

[42] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. Continuous-discrete interactions in chemical processing plants. *Proc. IEEE (USA)*, 88(7):1050 – 68, 2000.

[43] M. Englehart. High quality automatic code generation for control applications. In *Proceedings., IEEE/IFAC Joint Symposium on 7-9 March 1994*, pages 363 – 367, Tucson, AZ, USA, 1994.

[44] D. Feather, D. Harrell, R. Lieberman, and F. J. Doyle III. Hybrid approach to polymer grade transition control. *AIChE Journal*, 50(10):2502 – 2513, 2004.

[45] W. F. Feehery and P. I. Barton. Dynamic optimization with state variable path constraints. *Comput. Chem. Eng. (UK)*, 22(9):1241 – 56, 1998.

[46] A. Forsgren. On warm starts for interior methods (report TRITA-MAT-2005-OS4). Technical report, Royal Institute of Technology, Sweden, 2005.

[47] R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Manage. Sci. (USA)*, 36(5):519 – 54, 1990.

[48] R. Gebart, L. Westerlund, and I. Landalv. Black liquor gasification–the fast lane to the biorefinery. In *Energy Conference (Risø National Laboratory)*, 2005.

[49] J. Gondzio and A. Grothey. A new unblocking technique to warmstart interior point methods based on sensitivity analysis. *SIAM J. Optim. (USA)*, 19:1184–1210, 2008.

[50] T. M. Grace, B. Leopold, and E. W. Malcolm. *Pulp and Paper Manufacture - Alkaline Pulping*, volume 5. The Joint Texbook Committee of the Paper Industry, Montreal, Quebec, Canada, third edition, 1989.

[51] S. Haykin. *Adaptive filter theory*. Prentice Hall, Upper Saddle River, N.J, 2002.

[52] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol.1:364 – 9, 2001.

[53] M. Heidarinejad, J. Liu, and P. D. Christofides. Economic model predictive control of nonlinear process systems using Lyapunov techniques (pre-print). *AIChE Journal*, 2011.

[54] M. Hillestad and K. S. Andersen. Model predictive control for grade transitions of a polypropylene reactor. *Fourth European Symposium on Computer Aided Process Engineering. ESCAPE 4*, page 41.6, 1994.

[55] J. N. Hooker. A principled approach to mixed integer linear problem formulation. *Tepper School of Business. Paper 146.*, 2008.

[56] R. Huang. *Nonlinear Model Predictive Control and Dynamic Real Time Optimization for Large-Scale Processes*. PhD thesis, Carnegie-Mellon University, 2010.

[57] Y. J. Huang, G. V. Reklaitis, and V. Venkatasubramanian. Dynamic optimization based fault accommodation. *Computers and Chemical Engineering*, 24(2):439 – 444, 2000.

[58] Y. J. Huang, V. Venkatasubramanian, and G. V. Reklaitis. A model-based fault accommodation system. *Industrial and Engineering Chemistry Research*, 41(16):3806 – 3821, 2002.

[59] IBM Corporation. IBM CPLEX 12.1 User's Guide, 2009.

[60] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman Filter to nonlinear systems. In *AeroSense: the 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[61] J. V. Kadam, M. Schlegel, W. Marquardt, R. L. Toussain, D. H. van Hessem, J. van den Berg, and O. H. Bosgra. A two-level strategy of integrated dynamic optimization and control of industrial processes - a case study. Technical report, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1999.

[62] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, pages 33–45, 1960.

[63] J. D. Kelly and D. Zyngier. An improved MILP modeling of sequence-dependent switchovers for discrete-time scheduling problems. *Industrial and Engineering Chemistry Research*, 46:4964–4974, 2007.

[64] T. K. Kelly and M. Kupferschmid. Numerical verification of second-order sufficiency conditions for nonlinear programming. *SIAM Review*, 40(2):310 – 314, 1998.

[65] S. Kolas, B. A. Foss, and T. S. Schei. Constrained nonlinear state estimation based on the UKF approach. *Computers & Chemical Engineering*, 33(8):1386 – 1401, 2009.

[66] M. Krothapally and S. Palanki. On-line optimization of batch polymerization processes. *Proceedings of the American Control Conference*, 2:1187 – 1190, 1997.

[67] C. D. Laird and L. T. Biegler. IPOPT, An Interior-Point Solver for Exploiting Structure in Large-Scale Nonlinear Programs. In *INFORMS Annual Meeting, Pittsburgh, PA*, 2006.

[68] Y. D. Lang, A. Cervantes, and L. T. Biegler. DynoPC: A Dynamic Optimization Tool for Process Engineering. In *INFORMS National Meeting, Seattle, WA*, 1998.

[69] E. S. Lee and G. V. Reklaitis. Intermediate storage and operation of batch processes under batch failure. *Computers and Chemical Engineering*, 13(4-5):491 – 498, 1989.

[70] E. S. Lee and G. V. Reklaitis. Intermediate storage and the operation of periodic processes under equipment failure. *Comput. Chem. Eng. (UK)*, 13(11-12):1235 – 43, 1989.

[71] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. part 1: theoretical aspects. *Computers & Chemical Engineering*, 27(2):157 – 166, 2003.

[72] K. Leiviskä, E. Jutila, P. Uronen, and S. Heikkilä. Production Control of Complex Integrated Mills. *Computers In Industry*, 1(4):225–233, 1980.

[73] J. Lunze. *Modeling, analysis and design of hybrid systems*, chapter What is a hybrid system? Springer Verlag, 2002.

[74] J. F. MacGregor, D. J. Kozub, A. Penlidis, and A. E. Hamielec. State estimation for polymerization reactors. In *Dynamics and Control of Chemical Reactors and Distillation Columns. Selected Papers from the IFAC Symposium,*, pages 147 – 52, Oxford, UK, 1988.

[75] R. D. M. MacRosty. *Modeling, Optimization and Control of an Electric Arc Furnace*. PhD thesis, McMaster University, 2005.

[76] T. E. Marlin and H. N. Hrymak. Real-time operations optimization of continuous processes. In *5th International Conference on Chemical Process Control (CPC-5)*, pages 156–165, 1996.

[77] H. Michalska and D. Q. Mayne. Moving horizon observers and observer-based control. *IEEE Trans. Autom. Control (USA)*, 42:2209–2224, 1995.

[78] M. J. Mohideen, J. D. Perkins, and E. N. Pistikopoulos. Optimal Design of Dynamic Systems under Uncertainty. *AIChE Journal*, 42(8):2251–2272, 1996.

[79] M. J. Mohideen, J. D. Perkins, and E. N. Pistikopoulos. Towards an efficient numerical procedure for mixed integer optimal control. In *6th International Symposium on Process Systems Engineering and 30th European Symposium on Computer Aided Process Engineering*, volume 21, pages 457 – 62, UK, 1997.

[80] M. Morari and M. Baric. Recent developments in the control of constrained hybrid systems. *Computers and Chemical Engineering*, 30(10-12):1619 – 1631, 2006.

[81] W. R. Morrow. Hessian-free methods for checking the scecond-order sufficient conditions in equality constrained optimization and equilibrium problems. (arXiv:1106.0899v1 [math.OC] 5 Jun 2011).

[82] K. R. Muske and T. F. Edgar. *Nonlinear Process Control*, chapter Nonlinear State Estimation, pages 311–370. Prentice Hall PTR, 1996.

[83] Z. K. Nagy and R. D. Braatz. Robust nonlinear model predictive control of batch processes. *AIChE Journal*, 49(7):1776 – 1786, 2003.

[84] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[85] A. Noël, M. Savoie, H. Budman, and L. Lafontaine. Advanced brownstock washer control: Successful industrial implementation at James Mclaren. In *2nd IEEE Conference on Control Applications, Vancouver B. C.*, 1993.

[86] C. C. Pantelides. SpeedUp – recent advances in process simulation. *Comput. Chem. Eng. (UK)*, 12(7):745 – 55, 1988.

[87] B. Pettersson. Production control of a complex integrated pulp and paper mill. *TAPPI Journal*, 52(11):2155–2159, 1969.

[88] B. Pettersson. Optimal production schemes coordinating subprocesses in a complex integrated pulp and paper mill. *Pulp and Paper Canada*, 71(5):59–63, 1970.

[89] L. S. Pontryagin, V. Boltyanskii, R. Gamkrelidge, and E. Mishchenko. *The mathematical theory of optimal processes*. Interscience, NY, 1962.

[90] J. Prakash, S. C. Patwardhan, and S. L. Shah. State estimation and nonlinear predictive control of autonomous hybrid system using derivative free state estimators. *Journal of Process Control*, 20(7):787 – 99, 2010.

[91] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733 – 764, 2003.

[92] D. Ralph and S. J. Wright. Some properties of regularization and penalization schemes for mpecs. *Optimization Methods and Software*, 19(5 SPEC ISS):527 – 556, 2004.

[93] R. Raman and I. E. Grossmann. Relation between MILP modelling and logical inference for chemical process synthesis. *Computers and Chemical Engineering*, 15(2):73 – 84, 1991.

[94] A. Schafer, P. Kuhl, M. Diehl, J. Schloder, and H. G. Bock. Fast reduced multiple shooting methods for nonlinear model predictive control. *Chemical Engineering and Processing: Process Intensification*, 46(11):1200 – 14, 2007.

[95] C. A. Schweiger and C. A. Floudas. Interactions of Design and Control: Optimization with Dynamic Models. In *Optimal Control: Theory, Algorithms, and Applications*, pages 388–435. Kluwer Academic Publishers, 1998.

[96] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory and Applications*, 4(8):1303 – 18, 2010.

[97] G. A. Smook. *Handbook for Pulp and Paper Technologists*. Angus Wilde Publications, 2nd edition, 1999.

[98] E. D. Sontag. Nonlinear regulation: the piecewise linear approach. *IEEE Trans. Autom. Control (USA)*, AC-26(2):346 – 58, 1981.

[99] M. Stork, R. L. Tousain, J. A. Wieringa, and O. H. Bosgra. A milp approach to the optimization of the operation procedure of a fed-batch emulsification process in a stirred vessel. *Computers & Chemical Engineering*, 27(11):1681–1691, 2003.

[100] C. L. E. Swartz. Algorithm for hierarchical supervisory control. *Computers and Chemical Engineering*, 19(11):1173 – 1180, 1995.

[101] C. L. E. Swartz and A. K. S. Balthazaar. Dynamic optimization of an integrated multi-unit system under failure conditions. *AIChE Annual Meeting, Conference Proceedings*, pages 7532 – 7533, 2005.

[102] J. Tamini and P. Li. Nonlinear model predictive control using multiple shooting combined with collocation on finite elements. In *International Symposium on Advanced Control of Chemical Processes (ADCHEM), Istanbul, Turkey*, 2009.

[103] M. J. Tenny, J. B. Rawlings, and S. J. Wright. Closed-loop behavior of nonlinear model predictive control. *AIChE Journal*, 50(9):2142 – 2154, 2004.

[104] Thermo Electron Corporation. Gamma based consistency measurement in the pulp and paper industry (http://www.thermo.com/eThermo/CMA/PDFs/Product/productPDF_14149.pdf). Technical report, Thermo Electron Corporation, 2006.

[105] J. Till, S. Engell, S. Panek, and O. Stursberg. Applied hybrid system optimization: an empirical investigation of complexity. *Control Eng. Pract. (UK)*, 12(10):1291 – 303, 2004.

[106] I. B. Tjoa and L. T. Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial and Engineering Chemistry Research*, 30(2):376 – 385, 1991.

[107] T. Tosukhowong, J. M. Lee, J. H. Lee, and J. Lu. An introduction to a dynamic plant-wide optimization strategy for an integrated plant. *Computers and Chemical Engineering*, 29(1):199 – 208, 2004.

[108] E. Tufte. Sparklines: theory and practice (http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR&topic_id=1). Technical report, -, 2006.

[109] R. Van der Merwe and E. A. Wan. The square-root unscented kalman filter for state and parameter-estimation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume vol.6, pages 3461 – 4, Piscataway, NJ, USA, 2001.

[110] S. Vasantharajan and L. T. Biegler. Simultaneous strategies for optimization of differential-algebraic systems with enforcement of error criteria. *Computers and Chemical Engineering*, 14(10):1083 – 1100, 1990.

[111] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems. problems without path constraints. *Industrial and Engineering Chemistry Research*, 33(9):2111 – 2122, 1994.

[112] J. Villadsen and M. Michelsen. *Solution of differential equation models by polynomial approximation (Prentice-Hall international series in the physical and chemical engineering sciences)*. Prentice-Hall, 1978.

[113] A. Wächter. Short tutorial: getting started with IPOPT in 90 minutes. Technical report, IBM T. J. Watson Research Center., 2009.

[114] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25 – 57, 2006.

[115] E. A. Wan and R. V. D. Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC.*, pages 153–158. IEEE, 2000.

[116] R. Wang. Dynamic simulation of brownstock washers and bleach plants. Master's thesis, University of British Columbia, 1993.

[117] Y. Wang, H. Seki, S. Ooyama, K. Akamatsu, M. Ogawa, and M. Ohshima. Nonlinear predictive control for optimal grade transition of polymerization reactors. *Advanced Control of Chemical Processes 2000 (ADCHEM 2000). Proceedings volume from the IFAC Symposium*, 3:701 – 6, 2001.

[118] H. P. Williams. *Model building in mathematical programming*. Wiley and Sons, New York, 3rd edition, 1993.

[119] P. A. Wisnewski, F. J. Doyle III, and F. Kayihan. Fundamental continuous-pulp-digester model for simulation and control. *AIChE Journal*, 43(12):3175 – 3192, 1997.

[120] L. Wurth, R. Hannemann, and W. Marquardt. Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control*, 19(8):1277 – 1288, 2009.

[121] X. Xu and P. J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. *Hybrid Systems: Computation and Control. 6th International Workshop, HSCC 2003. Proceedings (Lecture Notes in Computer Science Vol. 2623)*, pages 540 – 55, 2003.

[122] A. C. Zanin, M. T. de Gouvêa, and D. Odloak. Integrating real-time optimization into the model predictive controller of the FCC system. *Control Engineering Practice*, 10:819–831, 2002.

[123] V. M. Zavala. Warm-Starting Interior Point Methods (CMU PSE Seminar, Nov 3, 2006). 2006.

[124] V. M. Zavala. *Computational Strategies for the Optimal Operation of Large-Scale Chemical Processes*. PhD thesis, Carnegie Mellon University, 2008.

[125] V. M. Zavala, C. D. Laird, and L. T. Biegler. Fast implementations and rigorous models: Can both be accommodated in NMPC? *International Journal of Robust and Nonlinear Control*, 18(8):800 – 815, 2008.

Appendix A

Process Description - Kraft Mill Fiber Line

A.1 Process Overview

The Kraft pulping process is composed of various production departments that are separated by buffer capacities. The major departments considered in this work are digestion, knotting, washing, screening and delignification.

The Kraft process begins with digestion. The digester vessel is filled with wood chips and white liquor (a concentrated solution of Na_2S and NaOH) and heated according to a predetermined schedule in a process known as “cooking”. The wood chips eventually disintegrate into fibers, forming pulp. Lignin, the organic component that holds cellulose fibers in wood together, reacts with the white liquor and is solubilized. The contents in the vessel are maintained at approximately 170°C for 2 hours, and then discharged into an adjacent tank. During this reaction, the white liquor turns into black liquor. The black liquor produced is channeled to a chemical recovery system which regenerates a part of it back into white liquor [97] and uses the rest as fuel for producing steam. The off-vapors are sent to a heat exchanger where it used to heat water for pulp washing. The cooked pulp is then subjected to various physical and chemical separation processes designed to remove unprocessable wood, residual black liquor and lignin.

The knotting department consists of knotter machines that function to remove undigested chips and ill-sized wood pieces known as “knots” which hinder downstream processing. Rejected chips are recycled to the digester for further cooking.

Following that, the pulp stream is directed to the washing department and undergoes a process known as brownstock washing (or simply, washing), where the residual black liquor is separated from the pulp in a carefully controlled process. This involves feeding the pulp into a series of counter-current vacuum drum washers, in which black liquor is displaced.

The washed pulp is then conveyed to the screening department for removal of “shives”, that is, wood pieces whose sizes lie between that of processable pulp and knots. Typically, vibrating pressure screens are employed for this task.

At this stage, the pulp stream would have been adequately prepared for delignification–the process by which the remaining lignin in the pulp is removed. The pulp stream is first mixed with chemical streams containing caustic soda and magnesium sulfate. This mixture is then fed in a counter-current direction with respect to an oxygen stream running within an O_2-delignification reactor. A reaction occurs in which the lignin separates from the cellulose fibers in the pulp stream. The delignified pulp stream is then ready for bleaching, where the pulp is whitened in a chlorination reactor.

In the following sections, we present the reader with the mathematical details of the process described above.

A.2 Mathematical Model Overview

The mathematical model in this thesis is based heavily on prior work done by Balthazaar [11] and Dubé [39], with some extensions based on various literature sources. We have chosen to model only a subset of the entire Kraft Paper mill (the fiber line) in order to focus our efforts on analyzing the issues surrounding the optimization and control of the process under shutdown conditions. Most units are modeled with quasi-steady-state equations. The only dynamics assumed in the system are in the tanks.

The level of modeling detail was chosen to ensure that the macro-effects germane to our application were adequately captured (inventories and compositions). The reader is urged to consult Castro and Doyle [28] for a detailed PDE-based model of the Kraft process.

A.2.1 Conventions and Nomenclature

Many of the flowrate and mass fraction variable names used in this model are codified[1]. They are constructed by concatenating three pieces of information:

$$F_{[w][i][j]}, \quad x_{[i][j]}$$

where the first letter F denotes a flowrate (in tons/hr), and x denotes a mass fraction. Furthermore, w = type of variable (in = inlet stream flowrate, out = outlet stream flowrate), i = stream number ($i = 1, 2, \ldots$), j = component in stream (P = Pulp, DS = Dissolved solids, W = water). In some units, a ST subscript refers to steam (water in vapor form) — this is done to differentiate the stream from streams containing liquid water. If the stream number is preceded by an S, it refers to an internal stream that does not interact directly with adjacent models. For clarity, refer to the following examples:

$$\begin{aligned} F_{out3DS} &= \text{component flowrate of outlet stream 3, dissolved solids component} \\ F_{in4ST} &= \text{steam flowrate in stream 4} \\ F_{S1P} &= \text{component flowrate of internal stream 1, pulp component} \\ x_{1W} &= \text{mass fraction of stream 1, water component} \end{aligned}$$

All flowrates are specified in tons/hour and compositions are given in mass fractions. Only three types of component streams are considered in the model: pulp (P), dissolved solids (DS) and water (W) (with steam (ST) being water in vapor form). The pulp component consists of celullose, bound lignin, knots and shives. The dissolved solids component contains organic and inorganic subcomponents.

The figures (Figs. A.1–A.5) show individual units with major streams labeled. The labeled variable names in the figures are assumed to belong to the individual model's namespace, e.g. the digester's F_{in1} is distinct from the blowtank's F_{in1}. A round black circle (•) denotes a connection point between units.

Each variable name is tagged with a superscript indicating the unit it is associated with. The mapping between the tag names and their corresponding unit names can be found in Table A.1 The tags are omitted from the variable names when describing the equations for individual units (in order to avoid clutter), but are used for disambiguation when multiple units are being

[1]The brownstock washing models are the exception. The nomenclature there follows an intuitive convention that is congruent with its literature source.

Tag	Unit Name	Tag	Unit Name
DG	digester	BT	blowtank
HQ	Hi-Q knotter	WC1	brownstock washer 1
WC2	brownstock washer 2	WC3	brownstock washer 3
ST	storage tank	SR	screeners
OF	O_2 feedpress	MX	mixer
OR	O_2 delignification reactor	PW	Post-O_2 washer

Table A.1: Variable superscript tag names and corresponding unit names.

described.

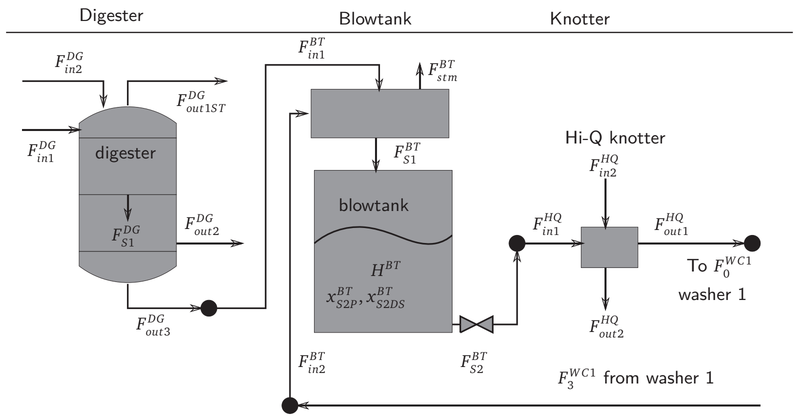


Figure A.1: Digester and knotter departments, with separating blowtank.

A.3 Digester (DG)

The continuous Kraft digester is the single most important unit in the Kraft process. Its primary function is to convert solid wood chips (obtained by debarking and chipping softwood logs) into a pulp stream. It accomplishes this by cooking the chips and reacting them with white liquor (a concentrated solution of Na_2S and NaOH). Cooking involves heating the mix from 70°C to 170°C and maintaining it at the latter temperature. During cooking, the organic component, lignin (the glue that bonds cellulose fibers together) is partially removed. The major streams in this model are shown in Fig. A.1.

Parameters

$a_{ST} = 0.04$ (split ratio), $r_{\mathrm{lw}} = 3.6$ (liquor to wood flow ratio), $x_{1DS} = 0.04$ (mass fraction), $x_{1P} = 0.43$ (mass fraction), $x_{w,blow} = 0.62$ (water fraction), $\beta_{as1} = 0$ (bias parameter),

$\beta_{as2} = 0$ (bias parameter), $F_{\max} = 40$ (maximum inlet pulp flowrate)

Variables (Algebraic States)

F_{S1DS} (flowrate), F_{S1P} (flowrate), F_{S1W} (flowrate), F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{out1ST} (flowrate), F_{out2DS} (flowrate), F_{out2W} (flowrate), F_{out3DS} (flowrate), F_{out3P} (flowrate), F_{out3W} (flowrate), F_{out3} (flowrate), $\alpha_{\text{shrink},1}$ (shrinkage factor 1), $\alpha_{\text{shrink},2}$ (shrinkage factor 2), ζ (production factor)

Variables (Control)

F_1 (pulp inlet flowrate)

Equations

The top section of our digester model is based on the model developed by Wisnewski and Doyle [119]. As lignin is removed from the cellulose fibers in the cooking process, there is a loss in pulp mass that needs to be accounted for. This loss is captured using a defined quantity known as “pulp shrinkage factor”. The pulp shrinkage factor is related to another quantity ζ, the production factor. The cubic relationship between these two variables was developed in Balthazaar’s [11] work. In our plant, the shrinkage factors were almost exactly linear in the range of feasible operation, therefore we present the linearized versions of the equations here. The pulp shrinkage regression equations are given below:

$$\zeta = \frac{F_1}{F_{\max}} \tag{A.1}$$

$$\alpha_{\text{shrink},1} = \frac{-3.94 \cdot \zeta + 14.59 + \beta_{as1}}{100} \tag{A.2}$$

$$\alpha_{\text{shrink},2} = \frac{-3.21 \cdot \zeta + 12.15 + \beta_{as2}}{100} \tag{A.3}$$

where $\alpha_{\text{shrink},1}$ represents the pulp shrinkage occurring in the top portion of the digester, while $\alpha_{\text{shrink},2}$ is a similar quantity for the bottom part of the digester.

The mass fraction composition of softwood are $x_{1P} = 0.43$, $x_{1DS} = 0.04$ and $x_{1W} = 0.53$ (Grace [50]). The inlet component streams are defined as follows:

$$F_{in1P} = F_1 \cdot x_{1P} \tag{A.4}$$

$$F_{in1W} = F_1 \cdot x_{1W} \tag{A.5}$$

$$F_{in1DS} = F_1 \cdot x_{1DS} \tag{A.6}$$

$$x_{1P} + x_{1W} + x_{1DS} = 1 \tag{A.7}$$

A simple mass balance (with shrinkages) governs the material flow in the top portion of the digester.

$$F_{S1P} = (1 - \alpha_{\text{shrink},1}) \cdot F_{in1P} \tag{A.8}$$

$$F_{S1DS} = F_{in1DS} + F_{in2DS} + \alpha_{\text{shrink},1} \cdot F_{in1P} \tag{A.9}$$

$$F_{S1W} = F_{in1W} + F_{in2W} - F_{out1ST} \tag{A.10}$$

A pressure relief valve at the top of the digester releases steam during the cooking process. The amount of steam vented from the digester, F_{out1ST} is related to the inlet water flow (Dubé [39]) by a proportionality factor, $a_{ST} = 0.04$.

$$F_{out1ST} = a_{ST} \cdot (F_{in1W} + F_{in2W}) \tag{A.11}$$

Because the complexities within a digester are not fully understood, it is a common practice in industry to maintain a constant liquor-to-wood flowrate ratio [50]. The r_{lw} (liquor-to-wood ratio) parameter, which regulates the proportion of liquor used relative to the amount of wood chips fed, is assumed to be 3.6 by mass (Dubé [39], Grace [50]).

$$F_{in2} = r_{\text{lw}} \cdot F_{in1P} \tag{A.12}$$

The mass fraction of dissolved solids in the liquor stream entering the digester was obtained by a stoichiometric unit conversion on the concentration of its constituents, 1.0 M NaOH and 0.8M Na_2S (Grace [50]). The dissolved solids mass fraction works out to 0.212, which means that the water fraction in the stream is (1 - 0.212) = 0.788.

$$F_{in2W} = 0.788 \cdot F_{in2} \tag{A.13}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.14}$$

The material transfer in the bottom part of the digester is modeled by mass balances (accounting for shrinkage). A splitter at the bottom is assumed.

$$F_{out3W} = x_{w,blow} \cdot F_{S1W} \tag{A.15}$$

$$F_{out2DS} \cdot F_{out3W} = F_{out2W} \cdot F_{out3DS} \tag{A.16}$$

$$F_{out3} = F_{out3P} + F_{out3W} + F_{out3DS} \tag{A.17}$$

The $x_{w,blow}$ (blowline water fraction) quantity has a value of 0.62, based on industrial data (Balthazaar, [11]). The shrinkage occurring at the bottom of the digester is given by:

$$F_{out3P} = (1 - \alpha_{\text{shrink},2}) \cdot F_{S1P} \tag{A.18}$$

$$F_{out3DS} = F_{S1DS} + \alpha_{\text{shrink},2} \cdot F_{S1P} - F_{out2DS} \tag{A.19}$$

$$F_{out3W} + F_{out2W} = F_{S1W} \tag{A.20}$$

Similar to Balthazaar [11] and Dubé [39], we assume quasi-steady dynamics for this unit in order to simplify the modeling and focus on the macro effects. In the event of a shutdown, we assume the outlet stream is instantaneously redirected to a dump tank, and that the shutdown time fully encompasses the time it takes to drain the digester and start it back up. At the end of a shutdown, we assume the digester is in a restoration-ready state.

Typically, there exist dynamics in the compositions. In his work, Dubé [39] reported an average digester residence time of $\tau = 1.56$ hours. However, given the data and the configuration of this particular model, we have observed that large changes in the input (chip feed flowrate) within its feasible range of operation only alter the compositions very slightly. Therefore we are able to neglect the effects of transport delay in composition with no significant loss of accuracy[2].

Owing to the pressure in the digester, its output is periodically expelled (or "blown") into an adjacent tank called a "blowtank"—described in the next section.

Inequality Constraints

$$0 \le \zeta \le 1 \tag{A.21}$$

$$0 \le F_1 \le F_{\max} \tag{A.22}$$

All other variables are constrained to be non-negative.

[2]This fact can be demonstrated in a simulation model by adding dynamic lags (see Section A.10) that correspond to the residence time given above. In our experiments, the compositions vary within a sufficiently narrow range that their dynamic effects may be safely neglected. This is partly attributable to the fact that the main chip feed stream is assumed to have a constant composition, and the incoming liquor flow is regulated according to a strict ratio to the chip feed flowrate.

A.4 Blowtank (BT)

In this section, we will lay down the equations that define the operation of a buffer tank. This generic buffer tank is modeled after an integrated blowtank unit. The top of the tank, where steam is vented (in a blowtank-type configuration), is assumed to be steady-state. The bottom portion of the tank is agitated, therefore a perfect mixing assumption is appropriate here. The major streams in this model are shown in Fig. A.1.

Parameters

$F_{S2,max} = 600$ (maximum flowrate of F_{S2} stream), $H_0 = 60$ (percentage, initial height of tank contents), $\rho = 0.900$ ton/m^3 (average density of contents), $V_{tank} = 2050$ m^3 (total tank capacity, volumetric), $M_{tank} = 0.900 \cdot 2050$ (total tank capacity, in tons), $x_{stm} = 0.02$ (steam fraction).

Variables (Differential States)

H (tank level, %), x_{S2DS} (mass fraction), x_{S2P} (mass fraction)

Variables (Algebraic States)

F_{S1DS} (flowrate), F_{S1P} (flowrate), F_{S1W} (flowrate), F_{S1} (flowrate), F_{S2DS} (flowrate), F_{S2P} (flowrate), F_{S2W} (flowrate), F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{stm} (flowrate), x_{1DS} (mass fraction), x_{1P} (mass fraction), x_{1W} (mass fraction), x_{2DS} (mass fraction), x_{2W} (mass fraction), x_{S1DS} (mass fraction), x_{S1P} (mass fraction), x_{S1W} (mass fraction), x_{S2W} (mass fraction)

Variables (Control)

F_{S2}

Equations

The following equations characterize inlet streams 1 and 2: (Note: in the computer code, all quotients are written as products to avoid division by 0 errors)

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.23}$$

$$x_{1P} = \frac{F_{in1P}}{F_{in1}} \tag{A.24}$$

$$x_{1DS} = \frac{F_{in1DS}}{F_{in1}} \tag{A.25}$$

$$x_{1P} + x_{1DS} + x_{1W} = 1 \tag{A.26}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.27}$$

$$x_{2DS} = \frac{F_{in2DS}}{F_{in2}} \tag{A.28}$$

$$x_{2DS} + x_{2W} = 1 \tag{A.29}$$

The compositions of the internal streams S1 and S2 are characterized by the below equations:

$$x_{S1P} = \frac{F_{S1P}}{F_{S1}} \tag{A.30}$$

$$x_{S1DS} = \frac{F_{S1DS}}{F_{S1}} \tag{A.31}$$

$$x_{S1W} = \frac{F_{S1W}}{F_{S1}} \tag{A.32}$$

$$x_{S1P} + x_{S1DS} + x_{S1W} = 1 \tag{A.33}$$

$$x_{S2P} = \frac{F_{S2P}}{F_{S2}} \tag{A.34}$$

$$x_{S2W} = \frac{F_{S2W}}{F_{S2}} \tag{A.35}$$

$$x_{S2DS} = \frac{F_{S2DS}}{F_{S2}} \tag{A.36}$$

$$x_{S2P} + x_{S2W} + x_{S2DS} = 1 \tag{A.37}$$

The amount of steam vented from the tank (for tank pressure relief) is proportional to the mass fraction of water in the stream entering the tank, and the proportionality factor x_{stm} is 0.02 (Dubé [39]). The material balance equations for the top part of the tank is as follows:

$$F_{in1} + F_{in2} = F_{S1} + F_{stm} \tag{A.38}$$

$$F_{in1DS} + F_{in2DS} = F_{S1DS} \tag{A.39}$$

$$F_{in1W} + F_{in2W} = F_{S1W} + F_{stm} \tag{A.40}$$

$$F_{stm} = x_{stm} \cdot F_{in1W} \tag{A.41}$$

The dynamic portion of the tank is governed by the equations below. This part of the tank represents the dynamic behavior of the buffer capacities.

$$M_{tank} \cdot 0.01 \cdot \frac{dH}{dt} = F_{S1} - F_{S2}, \qquad H(0) = H_0 \tag{A.42}$$

$$M_{tank} \cdot 0.01 \cdot H \cdot \frac{dx_{S2P}}{dt} = F_{S1P} - F_{S1} \cdot x_{S2P}, \qquad x_{S2P}(0) = x_{S1P}(0) \tag{A.43}$$

$$M_{tank} \cdot 0.01 \cdot H \cdot \frac{dx_{S2DS}}{dt} = F_{S1DS} - F_{S1} \cdot x_{S2DS}, \qquad x_{S2DS}(0) = x_{S1DS}(0) \tag{A.44}$$

The assumption of constant density is implicit in the above equations. Strictly speaking, the overall density of materials in the tank depends on instantaneous dynamic compositions, but the simplifying assumption of the materials having constant density is reasonable because the compositions changes in the tank occur fairly slowly.

It should be noted that in the pulp and paper industry, concentration quantities like x_{S2P} (mass fraction of pulp in the tank) and x_{S2DS} (mass fraction of dissolved solids) are directly measurable. The former can be measured on-line using a range of methods [104], including:

- pulp slurry conductivity
- pressure drop cause by flow through a fixed length of pipe
- intensity of transmitted microwaves/ultrasonic waves/reflected light
- load on a motor operating an agitator mixer
- head needed to maintain consistent flow through viscosity tube

The latter is typically measured using conductivity measurements or measurements taken with auto-titrators [85]. With x_{S2P} and x_{S2DS} known, x_{S2W} is trivially obtained.

Inequality Constraints

$$0 \leq F_{S2} \leq F_{S2,max} \tag{A.45}$$

$$10 \leq H \leq 90 \tag{A.46}$$

$$0 \leq x_{2DS} \leq 1 \tag{A.47}$$

$$0 \leq x_{2W} \leq 1 \tag{A.48}$$

$$0 \leq x_{1DS} \leq 1 \tag{A.49}$$

$$0 \leq x_{1P} \leq 1 \tag{A.50}$$

$$0 \leq x_{1W} \leq 1 \tag{A.51}$$

A.5 Hi-Q knotter (HQ)

Knotters are machines designed to screen and remove large undigested chips and wood knots. In our model, we consider two knotters operating in series, a Hi-Q knotter (manufactured by companies such as Ingersoll-Rand and GL&V) and a Jonsson knotter (manufactured by companies like Bird and Lamort). The pulp entering the Hi-Q knotter is mixed with wash liquor and the coarse, undigested chips are screened from the pulp stream. Accepted pulp is channeled to the brownstock washing department, while rejects are typically sent to the second (Jonsson) knotter for further screening (which we do not consider here). The major streams in this model are shown in Fig. A.1. Because the Jonsson knotter is functionally the same as the Hi-Q knotter, and it does not participate in this section of the plant, we will confine our description to the Hi-Q knotter equations.

Parameters

$\alpha_{dil1} = 0.05$ (dilution fraction), $\alpha_{knotrej1} = 0.10$ (reject fraction), $x_w = 0.95$ (water fraction)

Variables (Algebraic States)

F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), F_{out2DS} (flowrate), F_{out2P} (flowrate), F_{out2W} (flowrate), F_{out1} (flowrate), F_{out2} (flowrate), x_{out1P} (mass fraction), x_{out1DS} (mass fraction), x_{out1W} (mass fraction)

Equations

The Hi-Q knotter streams are defined as follows:

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.52}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.53}$$

$$F_{out2} = F_{out2P} + F_{out2W} + F_{out2DS} \tag{A.54}$$

$$F_{out1} = F_{out1P} + F_{out1W} + F_{out1DS} \tag{A.55}$$

The Hi-Q knotter is governed by the following overall and component mass balances:

$$F_{in1P} = F_{out1P} + F_{out2P} \tag{A.56}$$

$$F_{in1W} + F_{in2W} = F_{out1W} + F_{out2W} \tag{A.57}$$

$$F_{in1DS} + F_{in2DS} = F_{out1DS} + F_{out2DS} \tag{A.58}$$

The ratio of the F_{in2} stream to the F_{in1} stream is measured by the dilution factor, α_{dil1} (Dubé, [39]). Stream F_{in2} is assumed to be made up of 95% water. The pulp loss is represented using a knot rejection factor, $\alpha_{knotrej1}$ (Grace, [50]).

$$F_{in2} = \alpha_{dil1} \cdot F_{in1} \tag{A.59}$$

$$x_w = \frac{F_{in2W}}{F_{in2}} \tag{A.60}$$

$$F_{out2P} = \alpha_{knotrej1} \cdot F_{in1P} \tag{A.61}$$

The dilution stream F_{in2} is used to control the consistency of the rejects stream F_{out2} (Dubé [39]). The pulp/water and water/dissolved solids ratios are assumed to be equal in streams F_{out1} and F_{out2}, and this is enforced using two bilinear equations.

$$F_{out1W} \cdot F_{out2P} = F_{out1P} \cdot F_{out2W} \tag{A.62}$$

$$F_{out2W} \cdot F_{out1DS} = F_{out2DS} \cdot F_{out1W} \tag{A.63}$$

The outlet component flowrates are calculated as follows:

$$F_{out1} \cdot x_{out1P} = F_{out1P} \tag{A.64}$$

$$F_{out1} \cdot x_{out1DS} = F_{out1DS} \tag{A.65}$$

$$x_{out1P} + x_{out1W} + x_{out1DS} = 1 \tag{A.66}$$

Inequality Constraints

$$0 \le F_{in1} \tag{A.67}$$

$$0 \le F_{out1DS} \le 100 \tag{A.68}$$

$$0 \le F_{out1W} \le 600 \tag{A.69}$$

All other variables are constrained to be non-negative.

A.6 Brownstock Washing Circuits (WC1, WC2, WC3)

Washing units (also known as brownstock washers) serve to remove residual lignin in the pulp stream. The washing department is composed of three major process units: a header box, a vacuum drum washer and a seal tank. The header box mixes the contents of the incoming pulp

(F_0) and recycled liquor stream (F_R) and channels it to the vacuum drum washer for washing. The washer sprays the slurry stream (F_V) with wash liquor (F_2) and rejects are pushed to the seal tank. The contents of the seal tank are recycled to the header box and in some cases, conveyed to the agitated bottoms section of the blowtank as well. Washing circuits/stages are usually set up in a counter-current configuration. In a normal pulp mill, there may exist one or more washing circuits (in our model, there are three circuits). For simplicity, we will confine ourselves to a description of a single circuit of the washing department. The brownstock washing circuit equations in this thesis are modified from the work by Wang [116]. A schematic of a single washing circuit is shown in Fig. A.3.

Parameters for Washer i

$\rho = 0.998$ (average density of tank contents), and $a_1 = 0.133333$, $a_2 =$ -7, $a_3 = 1.316667$ are regression coefficients for the displacement ratio. $\beta_{DR} = 0$ is bias parameter for the displacement ratio.

Variables (Differential States) for Washer i

H_{st} (tank level, %), x_{DS3} (mass fraction).

Variables (Algebraic States) for Washer i

C_{M0} (inlet consistency, pulp-free basis), C_{M1} (outlet consistency, pulp-free basis), D_R (displacement ratio), F_0 (flowrate), F_1 (flowrate), F_V (flowrate), R_W (washing liquor ratio), x_{DS0} (mass fraction), x_{DS1} (mass fraction), x_{DSV} (mass fraction), x_{DS2} (mass fraction), x_{DSD} (mass fraction), r_{lv} (ratio of F_V over F_D).

Variables (Control) for Washer i

F_3 (flowrate), F_2 (flowrate), F_D (flowrate), F_R (flowrate)

Variables (Algebraic States) for terminal units only

F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1DS} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), F_{rec1DS} (recycle flowrate), F_{rec1W} (recycle flowrate).

Specifications for Washer 1

$H_{st0}^{WC1} = 43.3\%$ (initial tank level, as percentage), $V_{st}^{WC1} = 1067$ (m^3, tank volume)

Specifications for Washer 2

$H_{st0}^{WC2} = 33.5\%$ (initial tank level, as percentage), $V_{st}^{WC2} = 861$ (m^3, tank volume)

Specifications for Washer 3

$H_{st0}^{WC3} = 33.3\%$ (initial tank level, as percentage), $V_{st}^{WC3} = 683$ (m^3, tank volume)
$x_{DS3}^{WC3} = 0.01$ (dissolved solids fraction for shower water in last unit).

Circuit Connection Equations and Degrees-of-Freedom

In our model, there are 3 circuits, and 5 connection equations for each circuit.

$$\begin{aligned} F_0^{WC,i} &= F_1^{WC,i-1} \\ x_{DS0}^{WC,i} &= x_{DS1}^{WC,i-1} \\ C_{M0}^{WC,i} &= C_{M1}^{WC,i-1} \\ F_3^{WC,i} &= F_2^{WC,i-1} \\ x_{DS3}^{WC,i} &= x_{DS2}^{WC,i-1} \end{aligned}$$

where i is the index for the current circuit. A special note about the degrees-of-freedom: this is a counter-current system in which not all variables available for control are not independent because of variable dependencies.

Terminal Unit Equations

The following equations are stream definitions in Washer 1 (i.e. the first washer in the series) only. They convert component flowrates from the unit preceding the first washer (the Hi-Q knotter) into a mass fraction formulation. We use a mass fraction formulation for washers due to computations required on the compositions. These equations function as an interface between the washer models and the rest of the plant. Fig. A.2 shows the terminal unit variables in their context.

$$F_0^{WC1} = F_{in1P}^{WC1} + F_{in1W}^{WC1} + F_{in1DS}^{WC1} \tag{A.70}$$

$$x_{DS0}^{WC1} = \frac{F_{in1DS}^{WC1}}{F_0^{WC1}} \tag{A.71}$$

$$F_3^{WC1} = F_{rec1W}^{WC1} + F_{rec1DS}^{WC1} \tag{A.72}$$

$$x_{DS3}^{WC1} = \frac{F_{rec1DS}^{WC1}}{F_3^{WC1}} \tag{A.73}$$

$$C_{M0}^{WC1} = \frac{F_{in1P}^{WC1}}{F_0^{WC1} - F_{in1P}^{WC1}} \cdot 100 \tag{A.74}$$

These equations are for washer 3 (the last washer) only. They are used to convert from mass fractions back to component flows in order to interface with the rest of the plant. We compute the consistencies in the washers on a pulp-free basis to be consistent with product quality targets that are specified on those terms. In the final unit, the outlet consistency C_{M1} is fixed to a target value of 13.6.

$$F_1^{WC3} = F_{out1P}^{WC3} + F_{out1W}^{WC3} + F_{out1DS}^{WC3} \tag{A.75}$$

$$C_{M1}^{WC3} = \frac{F_{out1P}^{WC3}}{F_1^{WC3} - F_{out1P}^{WC3}} \cdot 100 \tag{A.76}$$

$$x_{DS1}^{WC3} = \frac{F_{out1DS}^{WC3}}{F_1^{WC3}} \tag{A.77}$$

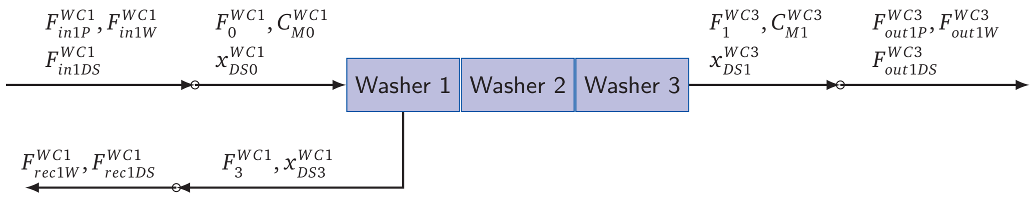

$F_{in1P}^{WC1}, F_{in1W}^{WC1}$
F_{in1DS}^{WC1}
F_0^{WC1}, C_{M0}^{WC1}
x_{DS0}^{WC1}
Washer 1
Washer 2
Washer 3
F_1^{WC3}, C_{M1}^{WC3}
x_{DS1}^{WC3}
$F_{out1P}^{WC3}, F_{out1W}^{WC3}$
F_{out1DS}^{WC3}
$F_{rec1W}^{WC1}, F_{rec1DS}^{WC1}$
F_3^{WC1}, x_{DS3}^{WC1}

Figure A.2: Terminal unit connections.

Equations

Generic stream and composition balance equations, for all washers (see Fig, A.3).

$$F_V = r_{lv} \cdot F_D \tag{A.78}$$

$$F_V = F_0 + F_R \tag{A.79}$$

$$F_V + F_2 = F_1 + F_D \tag{A.80}$$

$$F_V \cdot x_{DSV} = F_0 \cdot x_{DS0} + F_R \cdot x_{DS3} \tag{A.81}$$

$$F_V \cdot x_{DSV} + F_2 \cdot x_{DS2} = F_1 \cdot x_{DS1} + F_D \cdot x_{DSD} \tag{A.82}$$

The definitions of the Displacement Ratio (D_R) and Wash Liquor Ratio (R_W) are stated below. The displacement ratio (D_R) measures the washing efficiency of the solid in terms of percent dissolved solids removal (Noël et. al., [85]). This ratio easily be obtained by collecting samples on the washers. The displacement ratio is regressed against R_W using data from Grace et al. [50]. (Note: in the computer code, all quotients are written as products to avoid division by 0 errors)

$$D_R = \frac{x_{DS1} - x_{DS2}}{x_{DSV} - x_{DS2}} = a_1 R_W^3 + (0.1)a_2 R_W^2 + a_3 R_W + \beta_{DR} \tag{A.83}$$

$$R_W = \frac{F_2}{F_1} \tag{A.84}$$

The seal tank serves to create sufficient vacuum through the drop leg for proper washer operation. It is also used to store filtrate necessary for startup and functions as a buffer as well. The seal tank can be modeled simply as a continuously stirred tank.

$$V_{st} \cdot \rho \cdot 0.01 \cdot \frac{d}{dt} H_{st} = F_D - F_3 - F_R, \qquad H_{st}(0) = H_{st0} \tag{A.85}$$

$$V_{st} \cdot \rho \cdot 0.01 \cdot H_{st} \cdot \frac{d}{dt} x_{DS3} = F_D \cdot (x_{DSD} - x_{DS3}), \qquad x_{DS3}(0) = x_{DSD}(0) \tag{A.86}$$

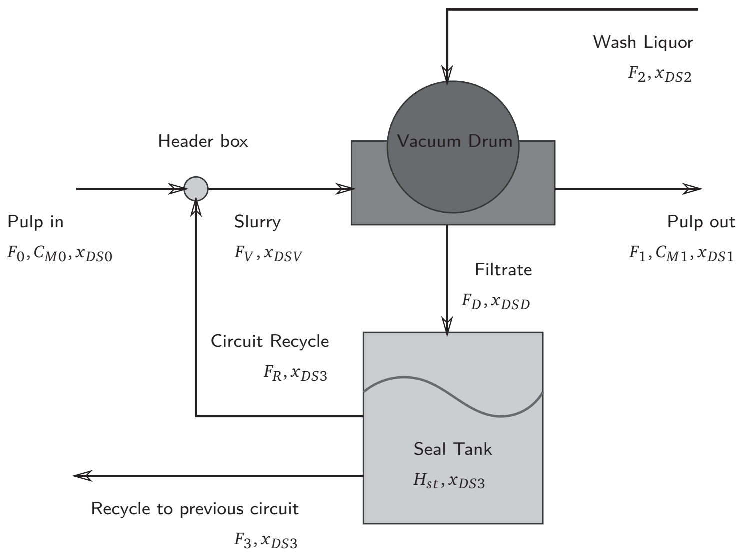


Figure A.3: A single brownstock washing circuit

Inequality Constraints

$$0 \le F_0 \le 600 \tag{A.87}$$

$$0 \le F_2 \le 600 \tag{A.88}$$

$$0 \le F_3 \le 600 \tag{A.89}$$

$$0 \le F_R \le 600 \tag{A.90}$$

$$0 \le F_V \le 600 \tag{A.91}$$

$$0 \le F_D \le 600 \tag{A.92}$$

$$0 \le R_W \le 10 \tag{A.93}$$

$$0 \leq C_{M1} \leq 100 \tag{A.94}$$
$$0 \leq C_{M0} \leq 100 \tag{A.95}$$
$$10 \leq H_{st} \leq 90 \tag{A.96}$$
$$0 \leq x_{DS3} \leq 1 \tag{A.97}$$
$$0 \leq x_{DSD} \leq 1 \tag{A.98}$$
$$0 \leq x_{DS0} \leq 1 \tag{A.99}$$
$$0 \leq x_{DSV} \leq 1 \tag{A.100}$$
$$0 \leq x_{DS1} \leq 1 \tag{A.101}$$
$$0 \leq x_{DS2} \leq 1 \tag{A.102}$$

All other variables were constrained to non-negative values. All 3 brownstock washers are connected in countercurrent fashion, shown in Fig. A.4.

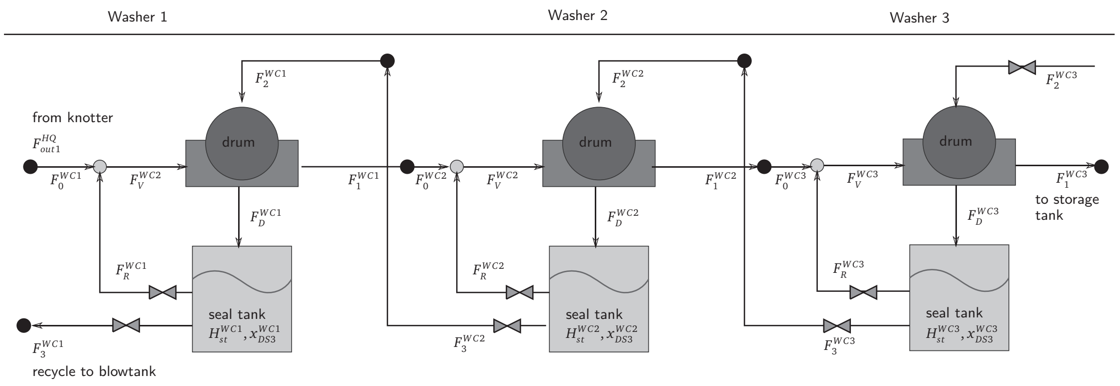


Figure A.4: Brownstock washing department, washing circuits in series.

A.7 Storage tank (ST)

The storage tank is a stirred tank buffer capacity between the brownstock washers and the screening units. Similar to the blowtank, it is modeled with a set of stirred tank equations (with assumptions of perfect mixing). The major streams in this model are shown in Fig. A.5.

Parameters

$F_{out1max} = 600$ (maximum outlet flowrate), $H_0 = 60$ (initial tank height, %), $M_{tank} = 0.900 \cdot 1620 = 1458$ tons (tank capacity), $V_{tank} = 1620$ m^3 (tank capacity, volumetric)

Variables (Differential States)

H (tank level, %), x_{out1DS} (mass fraction), x_{out1P} (mass fraction)

Variables (Algebraic States)

F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), x_{in1DS} (mass fraction), x_{in1P} (mass fraction), x_{in1W} (mass fraction), x_{out1W} (mass fraction)

Variables (Control)

F_{out1} (flowrate)

Equations

The following equations characterize the inlet and outlet streams of the tank:

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.103}$$

$$x_{in1P} = \frac{F_{in1P}}{F_{in1}} \tag{A.104}$$

$$x_{in1DS} = \frac{F_{in1DS}}{F_{in1}} \tag{A.105}$$

$$x_{in1P} + x_{in1DS} + x_{in1W} = 1 \tag{A.106}$$

$$F_{out1P} = x_{out1P} \cdot F_{out1} \tag{A.107}$$

$$F_{out1W} = x_{out1W} \cdot F_{out1} \tag{A.108}$$

$$F_{out1DS} = x_{out1DS} \cdot F_{out1} \tag{A.109}$$

$$x_{out1P} + x_{out1W} + x_{out1DS} = 1 \tag{A.110}$$

The dynamic equations with respect to the mass and compositions are as follows:

$$M_{tank} \cdot 0.01 \cdot \frac{d}{dt} H = F_{in1} - F_{out1}, \qquad H(0) = H_0 \tag{A.111}$$

$$M_{tank} \cdot 0.01 \cdot H \cdot \frac{d}{dt} x_{out1P} = F_{in1P} - F_{in1} \cdot x_{out1P}, \qquad x_{out1P}(0) = x_{in1P}(0) \tag{A.112}$$

$$M_{tank} \cdot 0.01 \cdot H \cdot \frac{d}{dt} x_{out1DS} = F_{in1DS} - F_{in1} \cdot x_{out1DS}, \qquad x_{out1DS}(0) = x_{in1DS}(0) \tag{A.113}$$

Inequality Constraints

$$0 \leq F_{out1} \leq F_{out1max} \tag{A.114}$$

$$10 \leq H \leq 90 \tag{A.115}$$

All other variables are constrained to non-negative values.

A.8 Screening and Delignification Departments

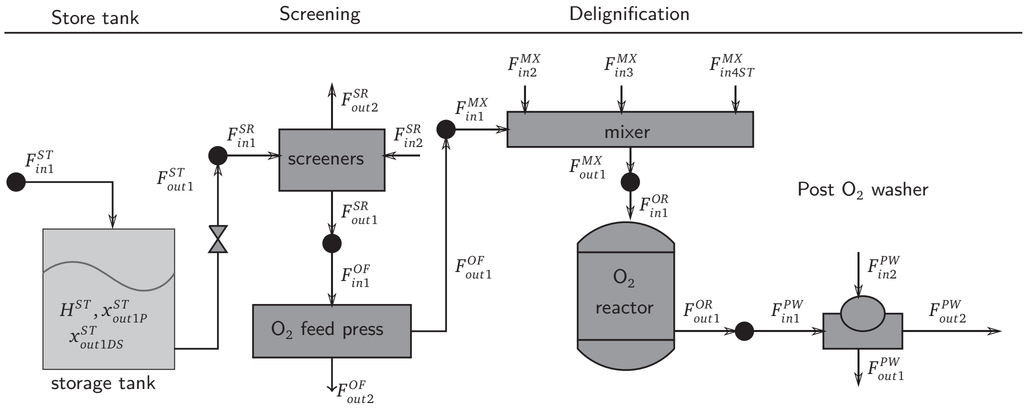


Figure A.5: Screening and delignification departments

A.8.1 Screening (SR)

Shives are a type of wood whose size is between that of processable pulp and knots. They hinder the bleaching process and cause deformities in formed sheets, and therefore need to be removed. A simplified pressure screen model is used. The function of pressure screens is to filter shives from the process stream. Balthazaar [11] offers a three-step screen model, which we have simplified into a one stage model while retaining the macro-effects and properties of the multi-stage process. The major streams in this model is shown in Fig. A.5.

Parameters

$c_{ploss} = 0.95$ (pulp loss coefficient), $x_w = 0.8$ (water fraction)

Variables (Control)

F_{out1} (flowrate)

Variables (Algebraic States)

F_{out2} (flowrate), F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate),

F_{out1W} (flowrate), F_{out2DS} (flowrate), F_{out2P} (flowrate), F_{out2W} (flowrate), C_M (consistency, %)

Equations

The stream definitions in this units are:

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.116}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.117}$$

$$F_{out1} = F_{out1P} + F_{out1W} + F_{out1DS} \tag{A.118}$$

$$F_{out2} = F_{out2W} + F_{out2DS} \tag{A.119}$$

A component mass balance relates the component streams:

$$F_{in1P} = F_{out1P} + F_{out2P} \tag{A.120}$$

$$F_{in1W} + F_{in2W} = F_{out1W} + F_{out2W} \tag{A.121}$$

$$F_{in1DS} + F_{in2DS} = F_{out1DS} + F_{out2DS} \tag{A.122}$$

Composition ratios are assumed to be equal across streams 1 and 2: (Note: in the computer code, all quotients are written as products to avoid division by 0 errors)

$$\frac{F_{out1W}}{F_{out1P}} = \frac{F_{out2W}}{F_{out2P}} \tag{A.123}$$

$$\frac{F_{out1P}}{F_{out1DS}} = \frac{F_{out2P}}{F_{out2DS}} \tag{A.124}$$

5% of pulp is lost in a screening unit due to the extraction of shives, therefore the pulp loss coefficient, $c_{ploss} = 0.95$. The target output consistency for downstream processing in O_2-delignification unit is 4.5%, so $C_M = 4.5$ (however, this value is relaxed during a partial shutdown and enforced during non-shutdown periods). Note that the consistency of pulp here is computed on a mass percent basis, as opposed to the pulp-free basis in the brownstock washers—this is done to be consistent with the specifications. The dilution stream is taken to have a water content of 80%, which means $x_w = 0.8$.

$$F_{out1P} = c_{ploss} \cdot F_{in1P} \tag{A.125}$$

$$F_{out1P} = \frac{C_M}{100} \cdot (F_{out1P} + F_{out1W} + F_{out1DS}) \tag{A.126}$$

$$F_{in2W} = x_w \cdot (F_{in2W} + F_{in2DS}) \tag{A.127}$$

The rejects from the screening department are discarded.

Inequality Constraints

$$0 \leq F_{out1} \leq 600 \tag{A.128}$$

$$0 \leq C_M \leq 100 \tag{A.129}$$

All other variables are constrained to non-negative values.

A.8.2 Delignification Department

At this stage of the process, the pulp is typically still dark in color due to the bound lignin and will be required to undergo bleaching. Unfortunately bleaching entails the use of chlorine, the emissions of which have a deleterious effect on the environment. In order to reduce the utilization of chlorine in bleaching, delignification is performed to remove as much as of the residual lignin as possible. This has the favorable effect of reducing the amount of chlorine required by the chlorination reactor in the bleaching department downstream.

In delignification, the pulp stream is heated with steam and mixed with caustic and then run down the O_2 reactor in a counter-current fashion. In this process, the impregnated pulp streams reacts with oxygen and a separation of lignin from the cellulose fibers occurs. It should be mentioned that if this were to be carried out too far, carbohydrate degradation also occurs, which results in an undesirable drop in viscosity. Process conditions must therefore be tightly controlled to preserve the integrity of the pulp.

The O_2 delignification department is made up of several units. The major ones are the O_2 feed presses, mixer, reactor and the post-O_2 washer. The O_2 feed press functions to raise the consistency of the pulp from 4.5% to 30% and is modeled by a simple mass balance. The mixer mixes the pulp with steam and caustic, while the reactor is the main vehicle for performing the delignification. The post-O_2 washer removes remaining dissolved solids and is modeled by a straightforward pass-through balance.

A.8.3 O_2 feed press (OF)

Variables (Control)

F_{out1} (flowrate)

Variables (Algebraic States)

F_{out2} (flowrate), F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), F_{out2DS} (flowrate), F_{out2W} (flowrate), C_M (outlet consistency, %), C_{M0} (inlet consistency, %)

Equations

The feed press equations consistent of simple mass balances and splits. Refer to Fig. A.5. (Note: in the actual model, all quotients are written as products to avoid division by 0 errors)

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.130}$$

$$F_{out1} = F_{out1P} + F_{out1W} + F_{out1DS} \tag{A.131}$$

$$F_{out2} = F_{out2W} + F_{out2DS} \tag{A.132}$$

$$F_{in1P} = F_{out1P} \tag{A.133}$$

$$F_{in1W} = F_{out1W} + F_{out2W} \tag{A.134}$$

$$F_{in1DS} = F_{out1DS} + F_{out2DS} \tag{A.135}$$

$$\frac{C_M}{100} \cdot F_{out1} = F_{out1P} \tag{A.136}$$

$$\frac{C_{M0}}{100} F_{in1} = F_{in1P} \tag{A.137}$$

$$\frac{F_{out1W}}{F_{out1DS}} = \frac{F_{out2W}}{F_{out2DS}} \tag{A.138}$$

The target output consistency for this unit is 30%, so $C_M = 30$ (however, this value is relaxed during a partial shutdown and enforced during non-shutdown periods). Note that the consistency of pulp here is computed on a mass percent basis, as opposed to the pulp-free basis in the brownstock washers—this is done to be consistent with the specifications.

Inequality Constraints

$$0 \leq C_M \leq 100 \tag{A.139}$$

$$C_{M0} \leq C_M \tag{A.140}$$

$$0 \leq F_{out1} \leq 600 \tag{A.141}$$

All other variables are constrained to non-negative values.

A.8.4 Delignification Mixer (MX)

Parameters

$C_{p,pulp1} = 3.972$ MJ/ton·K (specific heat of pulp), $H_{ref} = 2547.3$ MJ/ton (reference enthalpy, saturated steam at 298K), $H_{stm} = 3267.5$ MJ/ton (enthalpy of medium pressure steam (49 bar, 415°C), $T_{in,chem} = 25$°C (temperature of incoming chemicals), $T_{in,pulp} = 25$°C (temperature of incoming pulp), $T_{ref} = 25$°C (reference temperature), $T_{set} = 100$°C (setpoint temperature), $g_{MgSO4} = 0.002$ (magnesium sulfate dosage, fraction), $g_{NaOH} = 0.02$ (caustic dosage, fraction)

Variables (Algebraic States)

F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{in3DS} (flowrate), F_{in3W} (flowrate), F_{in3} (flowrate), F_{in4ST} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), F_{out1} (flowrate)

Equations

The delignification mixer is modeled as a perfect steady-state mixer [11]. A mass balance around the mixer yields the following equations: (Refer to Fig. A.5).

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.142}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.143}$$

$$F_{in3} = F_{in3W} + F_{in3DS} \tag{A.144}$$

$$F_{out1} = F_{out1P} + F_{out1W} + F_{out1DS} \tag{A.145}$$

$$F_{in1P} = F_{out1P} \tag{A.146}$$

$$F_{in1DS} + F_{in2DS} + F_{in3DS} = F_{out1DS} \tag{A.147}$$

$$F_{in1W} + F_{in2W} + F_{in3W} + F_{in4ST} = F_{out1W} \tag{A.148}$$

The pulp stream (F_{in1}) enters the mixer at 25°C; likewise for the caustic (F_{in2}) and magnesium sulfate (F_{in3}) streams. Magnesium sulfate is added for pulp protection. F_{in4ST} is the stream containing medium-pressure steam. The energy balance is used to calculate the steam consumption of the system.

$$F_{in4ST}(H_{stm} - H_{ref}) = F_{out1} \cdot C_{p,pulp1} \cdot (T_{set} - T_{ref}) \tag{A.149}$$

Using a reference temperature of $T_{ref} = 25°\text{C}$, the reference enthalpy of saturated steam is $H_{ref} = 2547.3$ MJ/ton. The enthalpy of medium-pressure steam, $H_{stm} = 3267.5$ MJ/ton at 49 bar and 415°C. This steam is used to heat the mixture to the required temperature of a setpoint temperature of $T_{set} = 100°\text{C}$. $C_{p,pulp1}$ is the average heat capacity of the exiting pulp, whose value is 3.972 MJ/ton·K. The terms representing the other streams are canceled out due to our choice of reference temperature.

The chemical dosage equations which determine the composition of the inlet streams are as follows:

$$F_{in2DS} = g_{NaOH} \cdot F_{in1P} \tag{A.150}$$

$$F_{in3DS} = g_{MgSO4} \cdot F_{in1P} \tag{A.151}$$

$$0.92 \cdot F_{in2W} = 0.08 \cdot F_{in2DS} \tag{A.152}$$

$$0.955 \cdot F_{in3W} = 0.045 \cdot F_{in1DS} \tag{A.153}$$

where $g_{NaOH} = 0.02$ ton NaOH/ton pulp and $g_{MgSO4} = 0.002$ ton $MgSO_4$/ton pulp [50]. The concentrations of the NaOH and $MgSO_4$ streams are 8% and 4.5% respectively, and their ratios are be directly calculated using the equations above.

Inequality Constraints

All variables are constrained to non-negative values.

A.8.5 O_2 delignification reactor (OR)

Parameters

$a_{O2} = 2.301$ (shrinkage factor)

Variables (Algebraic States)

F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{out1DS} (flowrate), F_{out1P} (flowrate), F_{out1W} (flowrate), ζ (production factor)

Equations

The O_2 delignification reactor is a counter-current reactor where the lignin in the pulp stream reacted with oxygen. The model presented here is regression-based model that is greatly simplified. The major streams in this model is shown in Fig. A.5. The following equations are derived from a mass balance in the reactor.

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.154}$$

$$F_{out1P} = (1 - 1\times 10^{-2}\cdot a_{O2})\cdot F_{in1P} \tag{A.155}$$

$$F_{out1DS} = F_{in1DS} + 1\times 10^{-2}\cdot a_{O2}\cdot F_{in1P} \tag{A.156}$$

$$F_{out1W} = F_{in1W} \tag{A.157}$$

where a_{O2} is a shrinkage factor to account for the fact that pulp shrinkage occurs in the reactor [11]. This factor is related to the production factor, ζ (see eqn. A.1 for the definition) by nonlinear function obtained by performing regression on industrial data [11]:

$$a_{O2} = a_0 + a_1\zeta + a_2\zeta^2 + a_3\zeta^3 \tag{A.158}$$

The values of the coefficients are $a_0 = 2.301$, $a_1 = -0.0022$, $a_2 = 0.0116$, $a_3 = -0.0113$. However, we found that in operating range of this plant ($\zeta \in [0,1]$), the variation in the value of a_{O2} was negligible, therefore we elected to substitute the above cubic equation with a constant (with $a_{O2} = 2.301$) to remove the nonlinearity from the constraint set.

Inequality Constraints
All variables are constrained to non-negative values.

A.8.6 Post-O_2 Washers (PW)

Parameters
$a_{dil} = 0.05$ (dilution factor), $x_w = 0.98$ (water fraction), $a_1 = 0.133333$, $a_2 =$ -7, $a_3 =$ 1.316667 are regression coefficients for the displacement ratio.

Variables (Control)
F_{out2} (flowrate)

Variables (Algebraic States)
F_{in1DS} (flowrate), F_{in1P} (flowrate), F_{in1W} (flowrate), F_{in1} (flowrate), F_{in2DS} (flowrate), F_{in2W} (flowrate), F_{in2} (flowrate), F_{out1DS} (flowrate), F_{out1W} (flowrate), F_{out2P} (flowrate), F_{out2W} (flowrate), D_R (displacement ratio) R_W (wash liquor ratio)

Equations

The equations for the post-O_2 washers consist mostly of mass balances and splits: (see Fig. A.5). (Note: in the actual model, all quotients are written as products to avoid division by 0 errors)

$$F_{in1} = F_{in1P} + F_{in1W} + F_{in1DS} \tag{A.159}$$

$$F_{in2} = F_{in2W} + F_{in2DS} \tag{A.160}$$

$$F_{out2} = F_{out2P} + F_{out2W} + F_{out2DS} \tag{A.161}$$

$$F_{in1P} = F_{out2P} \tag{A.162}$$

$$F_{in1W} + F_{in2W} = F_{out1W} + F_{out2W} \tag{A.163}$$

$$F_{in1DS} + F_{in2DS} = F_{out1DS} + F_{out2DS} \tag{A.164}$$

$$F_{out2P} = \frac{C_M}{100} \cdot F_{out2} \tag{A.165}$$

$$F_{in2W} + F_{in2DS} = a_{dil} \cdot F_{in1} \tag{A.166}$$

$$F_{in2W} = x_w \cdot F_{in2} \tag{A.167}$$

The target output consistency for this unit is 6.5%, so $C_M = 6.5$ (however, this value is relaxed during a partial shutdown and enforced during non-shutdown periods). Note that the consistency of pulp here is computed on a mass percent basis, as opposed to the pulp-free basis in the brownstock washers—this is done to be consistent with the specifications.

To calculate the dissolved solids exiting the washer, we employ the following equations:

$$R_W = \frac{F_{in2}}{F_{out2}} \tag{A.168}$$

$$D_R = a_1 R_W^3 + (0.1) a_2 R_W^2 + a_3 R_W \tag{A.169}$$

$$\frac{F_{out2DS}}{F_{out2}} = D_R \left[\frac{F_{in1DS}}{F_{in1}} - (1 - x_w) \right] + (1 - x_w) \tag{A.170}$$

Inequality Constraints

$$0 \leq F_{in2} \leq 600 \tag{A.171}$$

$$0 \leq F_{out2} \leq 600 \tag{A.172}$$

All variables are constrained to non-negative values.

A.9 Composite Model and Connectivity

The individual models above are connected by component flowrates to form a composite model of the Kraft plant. These connections occur at points marked by a round black circle (•) in Figs. A.1, A.4 and A.5. Connectivity equations are posed as equality constraints that equate stream components. Consider, for instance, the connection between the digester and the blowtank in Fig. A.1. They can be written as follows:

$$F^{DG}_{out3P} = F^{BT}_{in1P} \tag{A.173}$$

$$F^{DG}_{out3W} = F^{BT}_{in1W} \tag{A.174}$$

$$F^{DG}_{out3DS} = F^{BT}_{in1DS} \tag{A.175}$$

where F are component flowrates, and the superscripts DG and BT are used to refer to the digester and blowtank's variables respectively. Other connections in the composite model are made in a similar fashion.

A.10 Virtual Dynamic Lags

Most process units in this work are modeled with the assumptions of quasi-steady-state operation due to their fast dynamics. In some units, the composition dynamics may be significant. Dynamics often occur in the form of a delay or temporal lag in a variable. If an approximation these dynamics is desired, one can employ a mathematical construction that involves writing the equations for a train of virtual mixing models, and appending them to the outlet of quasi-steady-state unit (Schweiger and Floudas [95]).

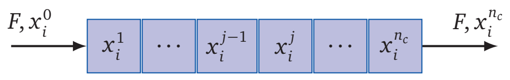


Figure A.6: Virtual dynamic lags used to model a transport delay.

The following set of coupled differential equations serve to approximate a transport delay in the compositions of a stream:

$$\frac{dx_i^j}{dt} = \frac{F}{M}(x_i^{j-1} - x_i^j), \quad i = 1 \dots n_c,\ j = 1 \dots n_j \tag{A.176}$$

where x_i is the mass fraction of component i, F is the stream flowrate, M is mass of the virtual tank's contents, $x_{i,in}$ is the inlet mass fraction of component i, and n_c is the number of components. The values of M and n_j are tuning parameters that can be adjusted to mimic the time constant/dead time behavior in the actual system. As n_j increases, the response becomes more sluggish, while M influences the shape of the lagged response. A visual depiction of this is shown in Fig. A.6.

Example. Consider a quasi-steady state model in the Kraft process, with outlet flow F, and compositions x_P, x_W and x_{DS}. Suppose we wish to lag these compositions to reflect the delay present in the system when the compositions change. For $n_j = 3$ and $M = 10$, we can write:

$$\frac{dx_P^1}{dt} = \frac{F}{10}(x_P - x_P^1) \tag{A.177}$$

$$\frac{dx_W^1}{dt} = \frac{F}{10}(x_W - x_W^1) \tag{A.178}$$

$$\frac{dx_P^2}{dt} = \frac{F}{10}(x_P^1 - x_P^2) \tag{A.179}$$

$$\frac{dx_W^2}{dt} = \frac{F}{10}(x_W^1 - x_W^2) \tag{A.180}$$

$$\frac{dx_P^3}{dt} = \frac{F}{10}(x_P^2 - x_P^3) \tag{A.181}$$

$$\frac{dx_W^3}{dt} = \frac{F}{10}(x_W^2 - x_W^3) \tag{A.182}$$

$$x_P^3 + x_W^3 + x_{DS}^3 = 1 \tag{A.183}$$

where x_P^3, x_W^3 are the lagged mass fractions. The mass fraction x_{DS}^3, also lagged, is computed via a difference.

Appendix B

Pulp and Paper Mill Inventory Model

Parameters
Flowrate bounds (tons/hr): $F_1^U = 131.25$, $F_2^U = 225$, $F_3^U = 100$, $F_4^U = 100$, $F_5^U = 125$, $F_6^U = 125$, $F_7^U = 125$, $F_8^U = 125$, $F_9^U = 31.25$, $F_{10}^U = 31.25$, $F_{11}^U = 62.5$, $F_{12}^U = 62.5$, $F_{13}^U = 93.75$, $F_{14}^U = 93.75$

Initial mass in tanks (tons): $M_{init,1} = 625$, $M_{init,2} = 360$, $M_{init,3} = 500$, $M_{init,4} = 156.25$, $M_{init,5} = 1175$, $M_{init,6} = 1531.25$

Tank capacity limits (tons): $M_{max,1} = 1500$, $M_{max,2} = 1500$, $M_{max,3} = 2000$, $M_{max,4} = 800$, $M_{max,5} = 1600$, $M_{max,6} = 2000$

Flow ratios: $\gamma_{11,10} = 2.0$, $\gamma_{13,12} = 1.5$, $\gamma_{3,2} = 0.4444$, $\gamma_{5,4} = 1.25$, $\gamma_{9,8} = 0.25$

Variables (Differential States)
H_1, H_2, H_3, H_4, H_5, H_6 (tank levels, %)

Variables (Algebraic States)
F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}, F_{11}, F_{12}, F_{13}, F_{14} (flowrates, tons/hr)

Variables (Control)
F_1, F_4, F_6, F_8, F_{10}, F_{12}, F_{14} (flowrates, tons/hr)

Equations
In this section, we describe a linear model of a pulp and paper mill adapted from work by

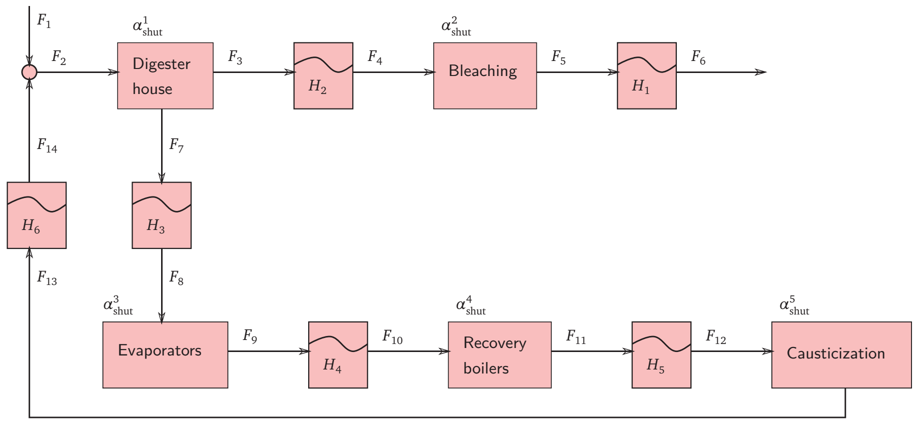


Figure B.1: Leiviskä et al. pulp mill, plant topology.

Leiviskä et al. [72]. The plant topology is shown in Fig. B.1. Only inventories are tracked. No compositions are considered. The resulting model is linear. The chip feed stream F_1 is mixed with white liquor from stream F_{14} and conveyed to the digester house, where it is cooked and reacted into pulp. The pulp is then bleached, and sent downstream for further processing. In the digester, spent liquor is produced in stream F_7, and is passed through a series of evaporators, boilers and causticizers in order to be regenerated.

The inventory equations for each tank are:

$$M_{max,1} \cdot 0.01 \cdot \frac{dH_1}{dt} = F_5 - F_6 \tag{B.1}$$

$$M_{max,2} \cdot 0.01 \cdot \frac{dH_2}{dt} = F_3 - F_4 \tag{B.2}$$

$$M_{max,3} \cdot 0.01 \cdot \frac{dH_3}{dt} = F_7 - F_8 \tag{B.3}$$

$$M_{max,4} \cdot 0.01 \cdot \frac{dH_4}{dt} = F_9 - F_{10} \tag{B.4}$$

$$M_{max,5} \cdot 0.01 \cdot \frac{dH_5}{dt} = F_{11} - F_{12} \tag{B.5}$$

$$M_{max,6} \cdot 0.01 \cdot \frac{dH_6}{dt} = F_{13} - F_{14} \tag{B.6}$$

The initial conditions for the above differential equations are:

$$H_j(0) = \frac{M_{init,j}}{M_{max,j}} \cdot 100, \qquad j = 1, \ldots, 6 \tag{B.7}$$

Simple flow ratios are used to model the relationship between the outlet and and inlet flowrates of individual processing units:

$$F_2 = F_3 + F_7 \tag{B.8}$$

$$F_3 = \gamma_{3,2} \cdot F_2 \tag{B.9}$$

$$F_5 = \gamma_{5,4} \cdot F_4 \tag{B.10}$$

$$F_9 = \gamma_{9,8} \cdot F_8 \tag{B.11}$$

$$F_{11} = \gamma_{11,10} \cdot F_{10} \tag{B.12}$$

$$F_{13} = \gamma_{13,12} \cdot F_{12} \tag{B.13}$$

The mixing point before the digester is modeled as follows:

$$F_2 = F_1 + F_{14} \tag{B.14}$$

Inequality Constraints

The flowrates are allowed to vary between defined bounds, and the tank levels may fluctuate between 15 – 85% of their maximum capacities.

$$0 \le F_i \le F_i^U, \qquad i = 1, \ldots, 14 \tag{B.15}$$

$$15 \le H_j \le 85, \qquad j = 1, \ldots, 6 \tag{B.16}$$

Appendix C

Basic results in continuous nonlinear optimization

A continuous nonlinear constrained optimization problem (P) can be expressed in the following canonical form:

$$(P) \qquad \min f(x) \tag{C.1}$$

$$\text{s.t. } g_i(x) \leq 0, \qquad i = 1, \dots, n_g \tag{C.2}$$

$$h_j(x) = 0, \qquad j = 1, \dots, n_h \tag{C.3}$$

where $x \in \mathscr{X} \subseteq \mathbb{R}^{n_x}$, $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ is the objective function, $g_i : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ are inequality constraints, and $h_j : \mathbb{R}^{n_x} \mapsto \mathbb{R}$ are equality constraints. The functions f, g_i and h_j are assumed to be twice-continuously differentiable.

Definition C.0.1 (The active set)**.** The active set is the index set of inequality constraint expressions whose values are *at the constraint boundaries* (that is to say, they are “active”) at some feasible point x. We define the active-set as follows:

$$A(x) = \{i | g_i(x) = 0\} \tag{C.4}$$

Definition C.0.2 (Linear independence constraint qualification, LICQ)**.** Given a point x, we say that the LICQ holds if the union of the sets of equality and active inequality constraint gradients, that is, $\{\nabla_x h_j(x) | j = 1, \dots, n_h\} \cup \{\nabla_x g_i(x) | i \in A(x)\}$, is linearly independent.

Definition C.0.3 (The Lagrangian and its Hessian)**.** The Lagrangian expression is defined as follows:

$$L(x,\mu,\lambda) = f(x) + \sum_{i=1}^{n_g} \mu_i g_i(x) + \sum_{j=1}^{n_h} \lambda_j h_j(x) \tag{C.5}$$

where $\mu = [\mu_i, \dots, \mu_{n_g}]^{\mathrm{T}}$, $\lambda = [\lambda_1, \dots, \lambda_{n_h}]^{\mathrm{T}}$ are KKT multipliers (see KKT section below). The Hessian of the Lagrangian is:

$$H(x,\mu,\lambda) = \nabla_{xx} L(x,\mu,\lambda) = \nabla_{xx} f(x) + \sum_{i=1}^{n_g} \mu_i \nabla_{xx} g_i(x) + \sum_{j=1}^{n_h} \lambda_j \nabla_{xx} h_j(x) \tag{C.6}$$

Many favorable properties accrue from the convexity of a problem. The following theorem states the criteria for convexity.

Theorem C.0.1 (Convexity)**.** *The problem is convex if and only if (1) f and every g_i are convex; (2) every h_j is affine; and (3) the set $\mathcal{X}$ is a convex set.*

C.1 First-Order Optimality Conditions

The first-order necessary conditions for optimality (also known as KKT conditions) gives the conditions under which a point x^* may be considered to be a stationary point.

Theorem C.1.1 (Karush-Kuhn-Tucker (KKT) conditions / First-order Necessary Conditions)**.** *Suppose that x^* is a local solution of problem P. Let us also suppose that: (1) f, each g_i and each h_j are continuously differentiable at x^*; and (2) x^* satisfies a constraint qualification (e.g. LICQ). Then there exists a set of constants μ_i and λ_j that fulfills the following criteria:*

$$\textit{Stationarity condition: } \nabla_x L = \nabla_x f(x^*) + \sum_{i\in A(x^*)} \mu_i \nabla_x g_i(x^*) + \sum_{j=1}^{n_h} \lambda_j \nabla_x h_j(x^*) = 0 \tag{C.7}$$

$$\textit{Primal feasibility: } g_i(x^*) \leq 0, \qquad i = 1,\dots,n_g \tag{C.8}$$

$$\textit{Primal feasibility: } h_j(x^*) = 0, \qquad j = 1,\dots,n_h \tag{C.9}$$

$$\textit{Dual feasibility: } \mu_i \geq 0, \qquad i = 1,\dots,n_g \tag{C.10}$$

$$\textit{Complementarity condition: } \mu_i g_i(x^*) = 0, \qquad i = 1,\dots,n_g \tag{C.11}$$

Remarks on KKT conditions:

1. If the problem is convex, then the local minimum x^* is also a global minimum and the above necessary conditions are also sufficient conditions for optimality.

2. When LICQ holds, the values of μ_i and λ_j at x^* are unique.

C.2 Second-Order Optimality Conditions

Definition C.2.1 (Linearized feasible direction set, $\mathscr{F}(x)$)**.** Given a feasible point x and active set $A(x)$, the linearized feasible direction set may be defined as follows:

$$\mathscr{F}(x) = \{d | d^{\mathrm{T}} \nabla_x h_j(x) = 0 \text{ for all } j,\ d^{\mathrm{T}} \nabla_x g_i(x) \leq 0 \text{ for all } i \in A(x)\} \tag{C.12}$$

Definition C.2.2 (Critical cone, $\mathscr{C}(x^*, \mu^*)$)**.** Given a linearized feasible direction set at x^*, $\mathscr{F}(x^*)$ and a KKT multiplier vector at x^*, μ^*, we define the critical cone as follows:

$$\mathscr{C}(x^*, \mu^*) = \{w \in \mathscr{F}(x^*) | \nabla_x g_i(x^*)^{\mathrm{T}} w = 0, \text{ for all } i \in A(x^*) \text{ with } \mu_i^* > 0\} \tag{C.13}$$

Theorem C.2.1 (Sufficient Second-Order Optimality Conditions, SSOC)**.** *Suppose x^* is a point that satisfies the KKT conditions for problem P, and the corresponding KKT multipliers are λ^*, μ^*. The point x^* is a strict local solution if:*

$$w^{\mathrm{T}} H(x^*, \lambda^*, \mu^*) w > 0, \textit{ for all } w \in \mathscr{C}(\mu^*) \setminus 0 \tag{C.14}$$

where $H(x^*, \lambda^*, \mu^*)$ is the Hessian of the Lagrangian at x^*.

The above description of the SSOC however does not provide any practical means for checking SSOC for a particular problem. The reader is advised to consult Kelly and Kupferschmid [64], Nocedal and Wright [84] and Morrow [81] for numerical means of verifying the SSOC.

C.3 The IPOPT algorithm and active-set methods

In order to provide some mathematical context for some of the arguments related to active-set methods and IPOPT's interior-point algorithm, we will supply a brief sketch of the two strategies.

Consider the following optimization problem:

$$\min f(x) \tag{C.15}$$
$$\text{s.t. } c(x) = 0 \tag{C.16}$$
$$x \geq 0 \tag{C.17}$$

where $x \in \mathbb{R}^{n_x}$, $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}$, $c : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_c}$. We note that general inequality constraints can be trivially converted to the above form by introducing bounded slack variables. The first-order optimality conditions (KKT conditions) for the above problem can be written as follows:

$$\nabla_x f(x) + \nabla_x c(x)\lambda - \nu = 0 \tag{C.18}$$
$$c(x) = 0 \tag{C.19}$$
$$XVe = 0 \tag{C.20}$$
$$x, \nu \geq 0 \tag{C.21}$$

where $\lambda \in \mathbb{R}^{n_\lambda}, \nu \in \mathbb{R}^{n_\nu}$ are KKT multipliers, $X = \text{diag}(x)$, $V = \text{diag}(\nu)$ and $e = [1, \ldots, 1]^{\text{T}}$. This result assumes that constraint qualifications are satisfied.

The active-set method. In the active-set method, the goal of algorithm is to determine the active-set on which the solution lies. A simplified version of the steps involved is described below (adapted from [123]). We define the element j of a vector x as x_j. The active-set is the subset of the inequality constraints that are active. In our case, because $x \geq 0$ defines our inequality constraint set, the elements of x that lie at 0 define the active set.

1. Start with a trial guess for active-set, $A(x) = \{j | x_j = 0\}$.

2. Solve the optimization problem, with the trial active-set enforced.

$$\begin{aligned} &\min f(x) \\ \text{s.t. } &c(x) = 0 \\ &x_j = 0, \quad j \in A(x) \end{aligned}$$

   Check to see if problem converges to a point that satisfies the KKT system defined in C.18 – C.21.

3. If the solution is a KKT point, accept the point and terminate. We denote the solution s^* as a tuple $s^* = (x^*, \lambda^*, \nu^*)$, where the * superscript indicates an optimal point. The solution

active-set is $A(x^*) = \{j | x_j^* = 0\}$.

Otherwise, proceed with another trial guess for $A(x)$ and return to Step 2.

There exist various methods and heuristics for arriving at a trial guess quickly, and for discarding poor guesses quickly. However, in general the problem of selecting the correct active-set is a combinatorial one and has a worst-case exponential complexity with respect to the number of inequality constraints.

The interior-point method. IPOPT eschews the combinatorial problem of active-set determination by adopting an interior-point (also known as a barrier) approach. The problem in C.15 – C.17 can be transformed into the following barrier problem [114]:

$$\min \phi(x;\mu) = \left(f(x) - \mu \sum_{j=1}^{n_x} ln(x_j) \right) \tag{C.22}$$

$$\text{s.t. } c(x) = 0 \tag{C.23}$$

where μ is a barrier parameter and $\phi(x;\mu)$ is the barrier objective function. Notice that for any $\mu > 0$, the log barrier term causes the barrier objective $\phi(x;\mu)$ to approach ∞ if any of the x_j terms approaches 0. The basic idea is to solve a series of barrier problems for decreasing values of the parameter μ, with μ eventually converging to 0. This can be thought of as a homotopy method applied to the following primal-dual system modified from C.18 – C.21:

$$\nabla_x f(x) + \nabla_x c(x)\lambda - v = 0 \tag{C.24}$$

$$c(x) = 0 \tag{C.25}$$

$$XVe = \mu e \tag{C.26}$$

where the complementarity constraints $XVe = 0$ have been relaxed to $XVe = \mu e$, and the explicit condition $x, v \geq 0$ removed. As we will see later, the latter condition is enforced implicitly as the algorithm proceeds in such as way as to always satisfy $x, v \geq 0$ during the iteration process. At the solution point, $\mu = 0$ and $x, v \geq 0$, therefore the original KKT system in C.18 – C.21 is recovered.

An issue to note here is that the KKT system only provides conditions to determine a stationary point. In this problem, a local minimum is a stationary point, but so are local maxima and saddle points. To address this issue, IPOPT incorporates a Hessian regularization scheme that attempts to avoid maxima and saddle points.

To explain how this regularization works, let us suppose that we are solving the barrier problem in C.22 – C.23, and our current iterate is given by the tuple (x_k, λ_k, ν_k). As mentioned before, the iterates are selected in such a way as to always strictly satisfy $x_k, \nu_k \geq 0$. Let us denote the Hessian of the Lagrangian (eqn. C.24) as W_k, and proceed the give the following definition:

$$W_k \triangleq \nabla_{xx} f(x_k) + \sum_{i=1}^{n_c} \lambda_i \nabla_{xx} c_i(x_k) \tag{C.27}$$

where $c_i(x_k)$ is the i'th equality constraint in $c(x_k)$, and λ_i is the i'th element of the KKT multiplier λ. Let us also define the Jacobian J (of constraint system $c(x)$) at iteration k as:

$$J_k \triangleq \nabla_x c(x_k) \tag{C.28}$$

For a fixed current value of the barrier parameter $\mu = \bar{\mu}$, we use a Newton step to compute the search direction. The Newton step equation for C.24 – C.26 can be written as a linear system:

$$\begin{bmatrix} W_k & J_k & -I \\ J_k^{\mathrm{T}} & 0 & 0 \\ V_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \\ \Delta \nu_k \end{bmatrix} = - \begin{bmatrix} \nabla_x f(x_k) + \nabla_x c(x_k) \lambda_k - \nu_k \\ c(x_k) \\ X_k V_k e - \bar{\mu} e \end{bmatrix} \tag{C.29}$$

However, the above linear system is unsymmetric, which is unappealing from a numerical linear algebra perspective. In order to exploit the efficiencies of symmetric linear solvers, the authors of IPOPT propose solving an equivalent but smaller—and symmetric—linear system (obtained by eliminating the last block row in C.29):

$$\begin{bmatrix} W_k + R_k & J_k \\ J_k^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla_x \phi(x_k; \bar{\mu}) + J_k \lambda_k \\ c(x_k) \end{bmatrix} \tag{C.30}$$

where $R_k \triangleq X_k^{-1} V_k$. Note that $\Delta \nu_k$ can now be obtained explicitly through:

$$\Delta \nu_k = \bar{\mu} X_k^{-1} e - \nu_k - R_k \Delta x_k \tag{C.31}$$

In order to guarantee selected descent properties, we need to ensure that $W_k + R_k$, when projected to the null-space of the constraint Jacobian J_k, is positive definite. More explicitly, we require the reduced barrier Hessian:

$$Z_k^{\mathrm{T}} W_k Z_k \succ 0 \tag{C.32}$$

where $Z_k \in \mathbb{R}^{n_x \times (n_x - n_\lambda)}$ is a matrix whose columns form the null-space of J_k. We use the notation $Y \succ 0$ to indicate that some matrix Y is positive-definite. We emphasize that this reduced barrier Hessian term is never actually formed in IPOPT, therefore an indirect check is done by way of the inertia[1] of the augmented matrix. As well, a solution to C.30 may not exist if J_k is non-full-rank. Therefore, a further modification to C.30 is required, which results in:

$$\underbrace{\begin{bmatrix} W_k + R_k + \delta_w I & J_k \\ J_k^{\mathrm{T}} & -\delta_c I \end{bmatrix}}_{K_k} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla_x \phi(x_k; \bar{\mu}) + J_k \lambda_k \\ c(x_k) \end{bmatrix} \tag{C.33}$$

where $\delta_w, \delta_c \geq 0$ are perturbation parameters chosen using a heuristic method (inertia correction procedure; see [114]). The negative sign is attached to δ_c in the matrix to avoid generating too many positive eigenvalues. Various values of δ_w, δ_c are tried until the inertia of the augmented matrix K_k indicates that it is positive definite.

For problems that do not satisfy Sufficient Second Order Conditions (i.e. many large-scale DAEs, ill-posed formulations, or problems with non-unique solutions, in our case), the value of $\delta_w > 0$ at the solution point, therefore the reported value for $\log_{10} \delta_w$ (`lg(rg)` in IPOPT's output) will be nonzero.

Once the Newton step $(\Delta x_k, \Delta \lambda_k, \Delta \nu_k)$ is obtained, a line-search routine is run to compute step sizes. A fraction-to-boundary rule is used to calculate maximum step sizes that will strictly satisfy the condition $x, \nu \geq 0$. The fraction-to-boundary parameter is $\tau = \min(0.99, \bar{\mu})$. The maximum step sizes $\alpha^x_{\max,k}, \alpha^\nu_{\max,k}$ are determined by:

$$\alpha^x_{\max,k} = \max\{a \in (0,1] | x_k + a\Delta x_k \geq (1-\tau) x_k\} \tag{C.34}$$
$$\alpha^\nu_{\max,k} = \max\{a \in (0,1] | \nu_k + a\Delta \nu_k \geq (1-\tau) \nu_k\} \tag{C.35}$$

Then, a filter is applied to a sequence of trial step-sizes for x. The sequence is $\alpha^x_{l,k} = 2^{-l} \alpha^x_{\max,k}$ for $l = 0, 1, \ldots, n_l$. A trial step-size is deemed acceptable under the filter method if it sufficiently decreases either the objective function $\phi(x; \mu)$, or the l_1-norm of the constraint violation, $||c(x)||_1$. If an acceptable step-size is found, the step is taken and a new set of iterate values is

[1]The inertia of a symmetric matrix is defined as $I = (i_p, i_n, i_z)$ where each element of the tuple denotes the number of postive, negative and zero eigenvalues respectively. The matrix is positive definite when $i_p = i_n$.

obtained:

$$x_{k+1} = x_k + \alpha_{l,k}^{x} \Delta x_k \tag{C.36}$$

$$\lambda_{k+1} = \lambda_k + \alpha_{l,k}^{x} \Delta \lambda_k \tag{C.37}$$

$$\nu_{k+1} = x_k + \alpha_{\max,k}^{\nu} \Delta \nu_k \tag{C.38}$$

We draw the reader's attention to the fact that a different step-size is used for ν_k (i.e. $\alpha_{\max,k}^{\nu}$) vis-à-vis the other variables ($\alpha_{l,k}^{x}$). The authors claim that this is done for efficiency purposes, in order to not restrict the steps.

The above procedure is repeated until the iterates converge to the solution. Note that many details have been omitted in this simplified description of the primal-dual interior-point algorithm used in IPOPT. The reader is urged to consult [114] for full implementation information.