INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9025069

Optimization of multistage systems with nondifferentiable objective functions

Dunatunga, Manimelwadu Samson, Ph.D.

The University of Arizona, 1990

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

OPTIMIZATION OF MULTISTAGE SYSTEMS WITH NONDIFFERENTIABLE OBJECTIVE FUNCTIONS

by

Manimelwadu Samson Dunatunga

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

1990

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Manimelwadu Samson Dunatunga

entitled Optimization of Multistage Systems with Nondifferentiable Objective Functions

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

[signature]	4/15/90 Date
[signature]	4/16/90 Date
[signature]	4/16/90 Date
	Date
	Date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

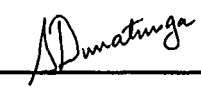
I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

[signature] 4/15/90
Dissertation Director Date

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: 

ACKNOWLEDGEMENT

I would like to thank Dr. Suvrajeet Sen for his advice and patience throughout the course of this research. I would also like to thank the examiners Dr. Geffrey Goldberg and Dr. Julia Higle for their valuable comments.

I would like to thank several friends who have helped me during the course of graduate study: Dr. Wijesena Wathugala, Dr.Tusitha Jayawardana, Dr.Larry Head and others. Lastly, I would like to thank my wife Damayanthi Dassanayake for her patience and support during the course of this research.

TABLE OF CONTENTS

TABLE OF CONTENTS – Continued

TABLE OF CONTENTS – Continued

TABLE OF CONTENTS – Continued

LIST OF TABLES

ABSTRACT

This dissertation is aimed at a class of convex dynamic optimization problems in which the transition functions are twice continuously differentiable and the stagewise objective functions are convex, although not necessarily differentiable. Two basic descent algorithms which use sequential and parallel coordinating techniques are developed. In both algorithms the nondifferentiability of the objective function is accounted for by using subgradient information. The objective of the subproblems generated consists of successive piecewise linear approximations of the stagewise objective function and the value function. In the parallel algorithm, an incentive coordination method is used to coordinate the subproblems. We provide proofs of convergence for these algorithms.

Two variations, namely, subgradient selection and subgradient aggregation, of the basic algorithms are also discussed. In practice while subgradient selection seems to perform well, computational results with subgradient aggregation are rather disappointing.

Computational results of the basic algorithms and variants based on subgradient selection are given. The effect of number of stages on performance of these algorithms is compared with a general nonlinear programming package (NPSOL).

CHAPTER 1

INTRODUCTION

This research is aimed at a class of dynamic optimization problems with the following structure.

$$\begin{aligned} \min \ & \sum_{t=1}^{N} f_t(x_t, u_t) \\ s.t. \ & x_{t+1} = T_t(x_t, u_t), \qquad t = 1, \ldots, N-1 \\ & x_1 \quad \text{is given,} \end{aligned} \tag{1.1}$$

where $f_t(x_t, u_t)$, $t = 1, \ldots, N$ are convex but not necessarily differentiable functions, and $T_t(x_t, u_t)$, $t = 1, \ldots, N-1$ are twice continuously differentiable functions.

For a given t, the function $f_t(x_t, u_t)$ is known as the stagewise objective function of stage t. The function $f_t(x_t, u_t)$ may be such that at some points it is not differentiable. For example, the absolute value function and the maximum of convex functions satisfy the above properties. For each t the function $T_t(x_t, u_t)$ is termed the transition function of stage t.

Often real world problems are constrained problems. Although the focus of the basic algorithmic development is to solve unconstrained problems, constrained dynamic optimization problems can be solved by first transforming them into unconstrained problems. Thus convex constrained dynamic optimization problems can be transformed to the form (1.1) by using penalty function methods. To illustrate, consider the following problem.

$$\begin{aligned} \min \quad & \sum_{t=1}^{N} F_t(x_t, u_t) \\ s.t. \quad & x_{t+1} = T_t(x_t, u_t), \qquad t = 1, \ldots, N-1 \\ & G_{tj}(x_t, u_t) \leq 0 \qquad j = 1, \ldots, m_t \end{aligned} \tag{1.2}$$

where $F_t(x_t, u_t)$ and $G_{tj}(x_t, u_t)$ are convex (not necessarily differentiable), and $T_t(x_t, u_t)$ are twice continuously differentiable functions. The equivalent optimization problem of the form (1.1) using the penalty function method can be written as

$$\begin{aligned} \min \quad & \sum_{t=1}^{N} (F_t(x_t, u_t) + \sigma_t \max \{0, G_{tj}(x_t, u_t)\}) \\ s.t. \quad & x_{t+1} = T_t(x_t, u_t), \qquad t = 1, \ldots, N-1 \end{aligned} \tag{1.3}$$

where $\sigma_t > 0$ are penalty parameters. The presence of the max function in above formulation (1.3) makes the objective function nondifferentiable even if $F_t(x_t, u_t)$ and $G_{tj}(x_t, u_t)$ are differentiable.

The remainder of this chapter is organized as follows. The next section provides both theoretical and practical motivation for the solution of dynamic nondifferentiable optimization problems (DNDO). Throughout this thesis we will utilize some basic concepts from nondifferentiable optimization which are summarized in Section three. Section four summarizes algorithmic concepts that are related to algorithms proposed in this dissertation. Finally in Section five we provide an overview of the developments presented in the following chapters.

Motivation

In the past, process optimization problems were considered difficult to model and cumbersome to solve. These problems were often solved by using over simplified models and applying optimization techniques developed for differentiable dynamic optimization problems. Such over simplification may lead to suboptimal solutions. Our goal here is to develop several techniques that use advances in optimization theory together with advances in computer hardware, to handle more realistic process optimization problems. The techniques we develop are motivated by static nondifferentiable optimization methods and dynamic differentiable optimization methods.

In the past three decades, nonlinear programming algorithms have been developed and applied to a wide range of single stage optimization problems. Most of these techniques are applicable to problems in which the objective function and the constraints are smooth (Fletcher [1981a], Fletcher [1981b]). In the presence of nondifferentiable functions, Lemarechal and Mifflin [1978], Shor [1979], Kiwiel [1985] and others have addressed algorithms for static nondifferentiable optimization (SNDO).

From an algorithmic viewpoint, it is fair to say that static or single stage problems are more easily solved than their dynamic counterparts. Hence when algorithms for static optimization are applied directly to dynamic problems, the results can be disappointing. It is therefore important to develop algorithms that explicitly account for the dynamics encountered in dynamic systems.

In dynamic optimization, the focus has been on smooth optimization problems (Ohno [1978], Turgeon [1980], Yakowitz [1983]). The success of Differential Dynamic Programming (DDP) for smooth dynamic optimization problems motivates the algorithms developed in this dissertation.

In addition to these theoretical motivations, several practical applications motivate the algorithms presented in this dissertation. We will describe numerous applications in detail in chapter 4. These include

(1) Industrial process optimization

(2) Electric power generation optimization

(3) Water resources management.

Several industrial processes such as turning, milling, forging and drawing operations are multistage operations. These processes often can be modeled as dynamic optimization problems with several constraints. The associated objective function of these models are often not differentiable. Also process constraints, such as maximum turning speed, maximum tool force etc. play an important role in the control of these processes. Modeling these processes as dynamic nondifferentiable optimization problems can be done by using penalty function techniques as suggested earlier. As an example, a model of a reversible rolling mill is given in detail in chapter 4.

It is easy to visualize electric power generation as a dynamic system. Demand for power generation varies over time with daily and seasonal variations. The objective of the power company is to maximize the total profit over a certain period of time by maximizing resource utilization. Since the generators have capacity restrictions, the optimization model of the system will have some bounding constraints. Again this leads to a dynamic nondifferentiable optimization problem of the form (1.3).

Water resources management is an extensively analyzed dynamic optimization problem. Several models of this problem can be found in literature (See Murray and Yakowitz [1979], Turgeon [1980], Jamshidi and Wang [1985], Christensen and Soliman [1986]). Some constrained dynamic optimization models can be found

in Murray and Yakowitz [1979]. These models have the form (1.2), and can be solved by the techniques developed in this dissertation.

Definitions in Nonsmooth Analysis

This section summarizes some basic definitions and results for nondifferentiable optimization. These results are used in subsequent chapters. For a thorough exposition of these topics we refer to Clarke [1983].

Convex Set. A set of points $X \subset \Re^n$ is said to be a convex set if and only if $\lambda x^1 + (1-\lambda)x^2 \in X$ for all x^1, $x^2 \in X$ and $\lambda \in [0,1]$.

A convex combination of a set of points $\{x^i\}_{i=1}^k \in \Re^n$ is defined as $\sum_{i=1}^{k} \lambda_i x^i$ for a given set $\{\lambda_i\}_{i=1}^k$ such that $\lambda_i \geq 0 \; \forall i$ and $\sum_{i=1}^{k} \lambda_i = 1$.

The convex hull of a set of points is defined as the smallest convex set that contains the given set of points.

Convex Function. A function $f : \Re^n \rightarrow \Re$ is convex if and only if

$$f(\lambda x^1 + (1-\lambda)x^2) \;\leq\; \lambda f(x^1) + (1-\lambda) f(x^2)$$

for all $\lambda \in [0,1]$ and x^1, $x^2 \in X$. A function is strictly convex if the above inequality is strict.

Differentiability. A function $f : \Re^n \rightarrow \Re$ is differentiable at x^0 if and only if there exists $g \in \Re^n$ such that $f(x) = f(x^0) + g^t(x - x^0) + o(\|x - x^0\|)$ for all x. g is known as the gradient of f at x^0.

The directional derivative of a function f at a point x in the direction d can be written as,

$$f'(x;d) = \lim_{t \rightarrow 0} \frac{(f(x+td) - f(x))}{t}$$

When the gradient exists, using Taylor's theorem, the directional derivative is given by the scalar product of direction d and the gradient g.

Differentiablity of a function is often assumed in mathematical programming. Nevertheless there are numerous applications in which traditional notions of differentiability are inapplicable. To generalize the notion of differentiability, first let us consider lipschitz functions.

Lipschitz function. A function $f : \Re^n \rightarrow \Re$ is lipschitz over a set X if for all $x^1,\ x^2 \in X$, there exists a constant $0 < K < \infty$ such that

$$|f(x^1) - f(x^2)| \leq K|x^1 - x^2|.$$

K is known as the rank of f.

It is easily seen that the set of convex functions is a subset of lipschitz functions. The definition given above suggests that a lipschitz function cannot have any discontinuity over the domain.

The Clarke generalized directional derivative, which is an extension of the directional derivative, for a lipschitz function is defined by Clarke [1983] as follows;

$$f^0(x;d) \ = \ \limsup_{\substack{y \rightarrow x \\ t \downarrow 0}} \frac{(f(y+td) - f(y))}{t}.$$

It is important to note that generalized directional derivative is well defined for any lipschitz function.

Proposition (Clarke [1983])

Let f be lipschitz of rank K near x and d be a feasible direction vector with respect to X. Then

(a) the function $f^0(x;d)$ is finite, positively homogeneous, and subadditive on X, and satisfies

$$|f^0(x;d)| \leq K\|d\|$$

(b) $f^0(x;d)$ is upper semicontinuous as a function of (x,d) and, as a function of d alone, is lipschitz of rank K on X.

(c) $f^0(x;-d) = (-f)^0(x;d)$.

Proof : See Clarke [1983].

The generalized gradient or the subdifferential set of a lipschitz function, which is an extension of the concept of a gradient and is denoted by $\partial f(x)$, is defined as the set

$$\partial f(x^0) = \{\xi : \xi^t d \leq f^0(x^0;d) \ \forall\, d\}.$$

For a convex function f, $\xi \in \partial f(x^0)$ implies

$$f(x) \geq f(x^0) + \xi(x - x^0).$$

When f is continuously differentiable at x, $\partial f(x)$ reduces to the singleton set, the gradient of f at x.

A subgradient of a function at a given point is defined as an element of the subdifferential. For a locally lipschitz function, the subdifferential set is a nonempty convex compact set. When a given point x_0 is stationary for a lipschitz function, the subdifferential set includes zero, i.e. $0 \in \partial f(x^0)$. This leads to the question of how one may recognize a descent direction for a lipschitz function; i.e.,

we need d to satisfy $f(x+td) < f(x)$ for all small $t > 0$. From above definition, if d satisfies

$$\max \{\xi^t d : \xi \in \partial f(x)\} < 0$$

then d is a descent direction for f at x. Next we summarize algorithmic methods.

Related problems and existing solution techniques

There are two optimization problems closely related to dynamic nondifferentiable optimization, namely,

(1) static nondifferentiable optimization problems and

(2) dynamic differentiable optimization problems.

Static nondifferentiable optimization problems share one of the basic concerns with our problem. The objective function, being nondifferentiable, makes techniques that use Taylor's first and second order approximations of the objective function inadequate. Techniques that use subgradient information will therefore be important in our work.

The other issue of concern in this research is the inclusion of dynamics in nonsmooth optimization problems. For smooth problems, Differential Dynamic Programming (DDP) and its variants have been very successful. Though it relies on differentiability of the objective function, DDP has the same dynamic nature as that encountered in DNDO problems. In both cases, the transition relations between successive stages are smooth. In the following section we provide a brief review of static NDO algorithms and dynamic algorithms for smooth discrete time optimization problems.

Static Nondifferentiable Optimization

Algorithms for static nondifferentiable optimization (SNDO) can be broadly classified as descent methods and subgradient methods. All these techniques are iterative procedures which produce a sequence of points that may or may not be feasible to the original problem. Those that produce only feasible solutions are known as feasible direction methods. In addition some algorithms, known as descent algorithms, generate a sequence of points whose objective value descreases monotonically (for minimization problems).

In this section we will first review descent methods, which include some cutting plane algorithms and bundle methods. Subgradient methods will be reviewed subsequently.

Cutting plane methods. These are successive relaxation schemes in which the nonlinear constraints and objective function are approximated by linear inequalities. These methods were first developed for convex programming problems (Kelly [1960]). Generalized cutting plane algorithms have been introduced by Eaves and Zangwill [1971] and these methods have been extended to nonconvex problems by Kiwiel [1985] and others.

Cutting plane algorithms can be described with two major steps. In the first step one solves an LP subproblem to update the iterative point. The second step generates a new cutting plane inequality that updates the piecewise linear approximation. Since each inequality is a support of the objective function, these methods are often called outer linearization methods. The piecewise linear approximation forms an outer envelop and hence provides a lower bound on the optimal value of the original problem. The upper bound is given by the best objective value obtained during the course of the iterations. The algorithm stops when the upper and lower bounds are within a prescribed tolerance.

The cutting plane algorithm described above uses an increasing number of segments of the piecewise approximation as iteration progress. Hence each step becomes increasingly cumbersome. A cut dropping scheme was developed in Eaves and Zangwill [1971] to avoid this explosion while retaining its convergence properties. In this cut dropping method one maintains previous cuts until sufficient reduction is obtained. Once reduction is achieved, all cuts that are nonbinding (at the optimality of the approximation) are dropped. Hence in the nondegenerate case, at most $n + 1$ cuts are retained. There may be more than $n + 1$ cuts active if the LP is degenerate.

Bundle methods. For nondifferentiable optimization, bundle methods were first proposed by Lemarechal [1978] for unconstrained convex problems. Lemarechal, Strodiot and Bihain [1981] extended these methods to nonconvex problems. Two realizations of the basic method, namely, subgradient aggregation and subgradient selection, were proposed by Kiwiel [1985]. Kiwiel [1985] presents subgradient aggregation methods for convex and nonconvex problems for unconstrained as well as constrained problems. An overview of these algorithms is given below. The reader is referred to Kiwiel [1985] for detailed analysis.

Each iteration of a bundle method for unconstrained convex minimization has three major steps, namely, (1) direction finding (2) line search and (3) linearization updating. In the direction finding step one solves a nearest point problem in which we determine a point in the approximate subdifferential that is nearest to the origin. The approximate subdifferential is represented as the convex hull of finitely many subgradients. Hence the solution to this problem is an aggregate subgradient. The search direction to be used in the current iteration is then the direction opposite to the aggregate subgradient. One may think of this scheme as

a generalization of the gradient method for smooth problems, since the aggregate subgradient obtained above represents the direction of steepest descent.

Having obtained a direction, the algorithm now proceeds to the line search step. The line search may produce either a "descent" step or a "null" step. A null step occurs when sufficient objective function reduction is not obtained while maintaining a minimum step size. When a descent step occurs, a new solution is accepted as the next iterate and the algorithm resumes its direction finding computations. In the event of a null step, the algorithm generates a new subgradient at the last point generated during the line search. This subgradient is now used in the polyhedral approximation of the subdifferential, or equivalently, to update the approximation to be used in direction finding. Actually, the process of updating the approximation is slightly more complicated, so as to ensure that when an infinite sequence of null steps occur, the polyhedral approximation of the subdifferential becomes sufficiently similar to the actual subdifferential. We have omitted such details here so as to focus on the essential structure of this class of algorithms.

To keep the size of approximation to a minimum two methods of dropping subgradients were introduced by Kiwiel [1985]. They are (1) the subgradient aggregation method and (2) the subgradient selection method. In the aggregation method, each iteration involves the aggregation of all previous subgradients into a single constraint. This aggregation is based on the Lagrange multipliers obtained in the direction finding subproblem. Such drastic reduction may not be very effective, although partial aggregations may be computationally fruitful. In the subgradient selection method a subset of subgradients are discarded at each iteration. Typically, constraints with positive Lagrange multipliers are preserved

for the next iteration, which limits the size of subproblem to at most $n+1$, n being the number of decision variables.

The above algorithms have been extended to nonconvex functions in Kiwiel [1985] by adding a distance resetting step. This step is used to update the locality measure of subgradients generated. The locality measure is used to weight previous subgradients, so that local subgradients contribute more information to the current direction than do previous ones.

Subgradient optimization. The second category of algorithms for NDO, the subgradient method, is also a generalization of gradient methods. These techniques are closely related to relaxation methods for solving a system of linear inequalities (Agmon [1954] , Motzkin and Shoenberg [1954]). A thorough review of subgradient optimization is given in Shor [1979]. In the following, we will summarize the main points.

The structure of the methods is extremely straightforward. Consider a convex minimization problem written as follows:

$$\begin{aligned} f^* &= \min\ f(x) \\ s.t.\ \ g(x) &\leq 0 \\ x &\in X \end{aligned}$$

Note that if there are several constraints $g_i(x) \leq 0$, $i \in I$, then one replaces all of these by one constraint $g(x) = \text{Max}\ \{g_i(x), i \in I\}$. If all the constraints are convex, then so is $g(x)$. Typically, one also assumes that X has some special structure; e.g. $X = \{x : 0 \leq x \leq u\}$. The algorithm then generates a sequence of points according to the formula

$$x_{k+1} = P_X(x_k - h_k(x_k).\xi_k(x_k))$$

where $\xi_k(x_k) \in \partial g(x_k)$ if $g(x) > 0$, and $\xi_k(x_k) \in \partial f(x_k)$ otherwise. The operator $P_X(y)$ denotes the projection of the argument y on to the set X. Finally, $h_k(x_k) > 0$ denotes a step size, selected according to some rule, that may depend on x_k.

This method can be shown to converge to an optimal solution under the condition that $h_k > 0$, $h_k \to 0$ as $k \to \infty$ and $\sum_k h_k = \infty$ (Shor [1979]).

Another step length rule given in Poljak [1969] is as follows,

$$h_k = \beta \left(\frac{v(x_k) - v^*}{\|\xi_k\|^2} \right)$$

where

$$v(x_k) = \begin{cases} g(x_k) & \text{if } g(x_k) > 0. \\ f(x_k) & \text{otherwise.} \end{cases}$$

and

$$v^* = \begin{cases} 0 & \text{if } g(x_k) > 0. \\ f^* & \text{otherwise.} \end{cases}$$

Several generalizations of this rule have been found effective in practice (Allen, Helgason, Kennington and Shetty [1987]). Goffin [1977] has studied the rate of convergence of these methods, and has proved a linear convergence rate under the assumption that the problem has a unique solution.

However, traditional subgradient methods have some major handicaps; namely, the absence of a stopping rule, the absence of the descent property and the absence of dual multiplier estimates at optimality. Note that in the absence of a stopping rule, one does not even have a bound on the difference between the value of the last iterate and that of an optimal solution. Furthermore in the absence of dual multiplier estimates, it is difficult to perform any sensitivity analysis. In order to resolve some of these difficulties, Sen and Sherali [1986] have proposed

primal-dual methods, which iterate on the primal and dual problems simultaneously. In this case, the difference between the primal and dual objectives is used as a stopping rule. Since the optimal difference between primal and dual solutions is 0 (i.e. $v^* = 0$), Poljak's step length rule becomes algorithmically implementable. Finally, the rate of convergence of these methods are shown to be linear, under less stringent assumptions.

Stagewise approaches to smooth dynamic optimization.

In this section we will discuss the current status of dynamic differentiable optimization problems. First we describe the fundamental dynamic programming algorithm, and then Differential Dynamic Programming (DDP) and its variations are discussed. The basic dynamic programming algorithm (Bellman [1957]) can be described as follows.

The algorithm consists of two major components, (1) a backward run and (2) a forward run. The backward run starts at the last stage, i.e. at $t = N$, and proceeds recursively with $t = N-1, N-2, \ldots, 1$. At each stage t a subproblem is defined recursively as follows

$$v_t(x_t) = \min_{u_t} (f_t(x_t, u_t) + v_{t+1}(T(x_t, u_t))).$$

Since there is no contribution to the objective function value beyond stage N, $v_{N+1}(x_{N+1}) = 0$ for all real valued x_{N+1}. This subproblem leads to an optimal functional relation of state x_t and control u_t, which may be written as

$$u_t = w_t(x_t).$$

Once the backward run is completed the forward run is started. The forward run is essentially a recursive substitution of state and control. Since the initial

state of the problem is given we can compute the control and then the state of the next stage using the transition function and above equation. Notationally, we may write

$$u_t^* = w_t(x_t^*)$$

$$x_{t+1}^* = T_t(x_t^*, u_t^*)$$

Except for some special problems, it is cumbersome to evaluate the functions $v_t(x_t)$ and $w_t(x_t)$. As shown below differential dynamic programming overcomes some of these difficulties by adopting a successive approximation approach based on Taylor's approximations.

Differential Dynamic Programming (DDP) is a stagewise variation of Newton's method in nonlinear programming. The method, introduced by Mayne [1966], requires the objective function and state transition functions to be smooth (twice continuously differentiable). During the last decade, its numerical properties have been investigated by Murray and Yakowitz [1979, 1981] and several variants have been proposed, as in Ohno [1978] and Sen and Yakowitz [1987]. Let us give a brief overview of the structure of a DDP algorithm. A detailed algorithmic description of DDP is given in Yakowitz and Rutherford [1984].

Each iteration of a DDP algorithm consists of two major components, namely, a backward run and a forward run. These two phases are analogous to the two phases of most nonlinear programming algorithms, namely, direction finding and line search. In the backward run, at each stage one approximates the return from the current stage (denoted f_t) together with a quadratic approximation of the optimal return function associated with all future stages (denoted v_{t+1}). The function, $f_t + v_{t+1}$, is now approximated by a quadratic function, which is denoted v_t. The control strategy for stage t, as a function of the state trajectory, is obtained by requiring that the first order optimality condition be satisfied for the

quadratic function v_t. In this manner, the backward run proceeds recursively until the control strategy for stage 1 is computed. At this point, a forward run begins. The forward run determines the next control policy by retrieving the backward run information, and recursively calculating the control and state of each stage. In order to ensure convergence, a line search ensuring sufficient objective function reduction is usually necessary.

Murray and Yakowitz [1979] have shown that the basic DDP scheme, using second order Taylor's approximations in generating the quadratic function v_t, is locally quadratically convergent. This rate is also borne out by the computational experience reported in Yakowitz and Rutherford [1984]. However, as in Newton's method for nonlinear programming, the basic DDP scheme requires second derivatives. Recently, Sen and Yakowitz [1987] have developed a variation of DDP that does not require the computation of second derivatives. This method, called QDDP, uses first derivative information to approximate the necessary second derivatives. In the spirit of dynamic programming, these approximations are performed in a stagewise manner. This algorithm is shown to be superlinearly convergent. This result has also been computationaly verified using the Broyden rank one update. Recent improvements, using sequential rank-one, rank-two updates, have been reported in Rakshit and Sen [1989].

Stagewise methods for constrained optimal control problems have not been investigated as extensively as have those for the unconstrained problems. However, Yakowitz [1986] introduces stagewise necessary conditions for verification of optimality of stagewise optimization problems. In the same paper, Yakowitz proposes a feasible direction algorithm for linearly constrained optimal control problems. Ohno [1978] has also proposed an algorithm based on solving the Kuhn-Tucker

necessary conditions in a stagewise manner. However, this algorithm is only locally convergent, and furthermore Ohno [1978] does not provide specific rules to update active inequality constraints. As a result, this method loses much of its appeal in the inequality constrained case. We note that both these algorithms also share some of the criticisms of the basic unconstrained DDP algorithm in that second derivatives are necessary for their realization. In the nonlinear programming literature, variable metric algorithms have proven to be the most successful methods to date. However, stagewise variable metric algorithms have not been proposed in the literature.

Parallel decomposition algorithms for dynamic optimization.

Due to increased computational power and parallel processing architecture of recent computers, the feasibility of parallel processing algorithms has been investigated in recent years. We shall briefly review parallel algorithms for dynamic optimization.

Lasserre [1985] proposes an algorithm using a mixed forward-backward method under the assumption that the transition functions are invertible. In this method one solves the first half of the time span by a forward method and the last half by a backward method. Coordination of the two halves is achieved by a higher level program which optimizes the sum of objective functions of two halves. Since the higher level program requires negligible computational effort, each iteration can be performed in half the amount of elapsed time as one iteration of either the backward or forward method.

Another procedure is developed by Scheel and McInnis [1981], in which they combine an interaction-prediction principle and state increment dynamic programming. The method is developed for nonlinear systems with quadratic stagewise

return functions. Their decomposition differ from Lasserre [1985] since it is performed on state variables of the system. The coupling due to the dynamics is removed by introducing a linear term to the objective function of each subproblem. Coordination of the subproblems is achieved by a higher level program. The higher level program minimizes the sum of cost functions of the subproblems to produce new coordination and interaction terms. The algorithm starts with a nominal trajectory and at each iteration each subproblem computes its own state and control variables and passes them to the higher level program. The higher level program adjusts the coordination term and linear interaction terms of each subproblem and passes the new linear terms back to the subprograms. This procedure is repeated until the system settles down to an optimal solution. The algorithm is proven to be convergent given that the state transition matrix is strongly diagonal.

A target coordination scheme for parallel processing of dynamic programming has been developed by Chang, Chang and Luh [1986]. In this method, the original problem is decomposed along the time axis to form several subproblems. The coordination of these subproblems is achieved by controlling initial and final states of each subproblem by the higher level program. The higher level program minimizes the sum of objective functions of each subproblem. It is shown that the decomposed problem and the original problem have the same optimal solution, given that the original cost function is convex and system dynamics are linear. For the unconstrained case the decomposed problem is solved by modified DDP method for lower level subproblems and Newton's method for higher level subproblem. Computational results show that the throughput time decreases but the utilization also decreases as the number of processors increases. However, no convergence results were reported in this paper.

Overview of the upcoming developments

In this section we will briefly describe the descent and parallel algorithms developed in later chapters. In the development of these algorithms we use piecewise linear approximation functions extensively.

Our descent algorithm has a structure similar to that of a DDP algorithm. The major differences lie in the construction of the objective function approximations. Each stagewise objective function and value function are approximated by piecewise linear functions. Each transition function is approximated with a first order Taylor series.

The parallel algorithm is essentially a result of the removal of sequential dependency of adjacent stages of the descent algorithm. This is achieved by a coordinating system which communicates with all subsystems. The subsystems are solved independently using techniques similar to those found in descent algorithms, while the coordinating system supplies the controlling parameters computed by a line search.

Both algorithms use the subdifferential set to identify optimality. Whenever a computed subset contains the zero vector, the control policy is declared optimal.

CHAPTER 2

DESCENT ALGORITHMS FOR CONVEX PROBLEMS

This chapter is devoted to descent algorithms for optimization of unconstrained multistage systems with nondifferentiable objective functions. We will begin this chapter with a brief discussion motivating descent algorithms, and this is followed by a detailed algorithmic development. We establish the convergence of the basic algorithm and suggest two variations of the method. Computational results on several test problems are given in chapter 4.

Recall the problem formulation

$$\begin{aligned} \min \quad & \sum_{t=1}^{N} f_t(x_t, u_t) \\ s.t. \quad & x_{t+1} = T_t(x_t, u_t), \qquad t = 1, \ldots, N-1 \\ & x_1 \quad \text{is given,} \end{aligned} \tag{2.1}$$

where $f_t(x_t, u_t)$ is known as the stagewise objective function in stage t, $t = 1, \ldots, N$, and $T_t(x_t, u_t)$ denotes the transition function in stage t, $t = 1, \ldots, N-1$. It is assumed that $f_t(\cdot)$ are convex and $T_t(\cdot)$ are twice continuously differentiable. A sequence of controls $\{u_t\}_{t=1}^{N}$ is known as a policy. We will refer to a sequence of functions $\{u_t(x_t)\}_{t=1}^{N}$ mapping states to the controls as a strategy. Any valid strategy which generates only a part of the sequence, i.e. $\{u_t(x_t)\}_{t=j}^{N}$, $j > 1$ will be refered as a partial strategy. The sum of future stage returns ($j = t+1, \ldots, N$) is known as the value function of stage t.

Preliminaries

The algorithms developed in this chapter are based on successive approximations of the stagewise objective functions in a sequential manner. Here we will develop a class of algorithms that resemble Differential Dynamic Programming (DDP) for smooth optimization problems. Since our objective functions are not necessarily differentiable we will use piecewise linear approximations to approximate the stagewise objective function together with the value function of future returns. Our basic algorithm consists of (1) an initialization step, (2) the backward run and (3) the forward run. The initialization step is executed only once in the beginning; the backward and forward runs are executed once every iteration.

The algorithm starts with a given nominal policy. The backward run begins at the last stage and proceeds towards the first stage. During the backward run, stagewise optimality conditions for the piecewise linear (PL) approximation are used to obtain a strategy in the form of an affine function. Given this strategy, the value function for stage t, representing returns from stage t through N, is approximated. The backward run then proceeds to stage $t-1$ and one obtains the value function for stage $t-1$ in a similar manner. This process continues until the backward run reaches stage 1. At this point, the backward run is completed and the forward run begins. The affine functions developed during the backward run are used as the strategy functions in the forward run. The strategy functions together with the dynamics of the system are used to obtain a new control policy. If the new policy is better than the nominal one, then it is accepted as the next nominal policy. Otherwise the PL approximation is refined using the new control policy. The first case is referred to as a descent step, while the latter case is termed a null step. In either case the backward run resumes. This procedure is

repeated until no substantial improvement is possible. A detailed development of the algorithm follows.

Derivation of the method

It is assumed that both an initial state and a nominal control policy are given. The state trajectory is computed using the state transition function and the given nominal policy. At each stage, a subgradient and the value of the stagewise objective function are computed. This information is used during the first backward run.

The backward run

Given a feasible control policy, and an initial system state, the backward run attempts to obtain a strategy that reduces the overall objective function. The backward run starts at the last stage and steps back through all stages until the first stage is reached. At each stage t, the backward run

(a) update a PL approximation of the objective function of stage t,

(b) update a PL approximation of the value function of stage t,

(c) solve the corresponding subproblem (QP or LP) and,

(d) compute a strategy relating the control and the state vector of stage t.

These steps are discussed in detail below.

Piecewise linear approximation of stagewise objective functions

The stagewise objective functions we consider are convex but not necessarily differentiable. Hence subgradients are used to construct an outer linearization of the objective function. This PL approximation is updated at each iteration by

using a new set of subgradients. Suppose that at iteration k, the PL approximation of the objective function in stage t, $t \in \{1, 2, .., N\}$ is,

$$\eta_{1t}^k(x_t, u_t) = \max \left\{ g_{1t}^l x_t + h_{1t}^l u_t + f_t^l \quad : l \in I_{1t}^k \right\}, \tag{2.2}$$

where I_{1t}^k denotes the index set of subgradients of the stagewise objective function of stage t (f_t) at iteration k. Also (g_{1t}^l, h_{1t}^l) denotes a subgradient of the stagewise objective function of stage t and f_t^l denotes a scalar. Then the PL approximation at iteration $k+1$ is obtained as follows.

The candidate policy generated at iteration k is denoted by u_t^+ and the corresponding state by x_t^+. Let $(\xi_{tx}^+, \xi_{tu}^+) \in \partial f_t(x_t^+, u_t^+)$, and let $f_t^+ = f_t(x_t^+, u_t^+)$. Then

$$\eta_{1t}^{k+1}(x_t, u_t) = \max \left\{ \begin{array}{ll} g_{1t}^l x_t + h_{1t}^l u_t + f_t^l & : l \in I_{1t}^k \\ \xi_{tx}^+(x_t - x_t^+) + \xi_{tu}^+(u_t - u_t^+) + f_t^+ & \end{array} \right\} \tag{2.3}$$

Piecewise linear approximation of the value function.

With the exception of the last stage, decisions at any stage influence all subsequent stages. Hence the control policy should be determined by considering both the current stage objective function and the objective functions of all future stages. In our algorithm all future stages are accounted for by using a piecewise linear approximation of the value function.

Suppose that at iteration k, the t^{th} stage subproblem is solved. Suppose that the PL approximation of the value function used in stage t is given by

$$\eta_{2t}^k(x_t, u_t) = \max \left\{ g_{2t}^l x_t + h_{2t}^l u_t + v_t^l \quad : l \in I_{2t}^k \right\}, \tag{2.4}$$

where I_{2t}^k denotes the set of indices of subgradients of the value function used in stage t at iteration k. Also (g_{2t}^l, h_{2t}^l) denotes a subgradient of the value function of

stage t and v_t^l denotes a scalar. Then $\eta_{1t}^k + \eta_{2t}^k$ approximates the sum of objective functions of stages t and beyond. Therefore a PL approximation of the value function for stage $t-1$ is given by

$$\eta_{2(t-1)}^k = \eta_{1t}^k + \eta_{2t}^k.$$

That is,

$$\eta_{2(t-1)}^k = \max\{g_{1t}^l x_t + h_{1t}^l u_t + f_t^l : l \in I_{1t}^k\} + \max\{g_{2t}^l x_t + h_{2t}^l u_t + v_t^l : l \in I_{2t}^k\}. \quad (2.5)$$

Note that $\eta_{2(t-1)}^k$ is a function of x_t and u_t. By combining the relationship for optimality of the subproblem for stage t at iteration k, which is derived in a later section, together with a linearization of the transition function, one can rewrite (2.5) only in terms of x_{t-1} and u_{t-1}. This piecewise linear approximation is then used to represent the value function at stage $t-1$.

The linear programming subproblem

At any iteration, the last stage subproblem consists of only the PL approximation of the last stage return. Since we optimize the subproblem for a nominal state vector, the linear programming subproblem may written as

$$\begin{aligned} &\min \ \eta_{1N}^k \\ &s.t. \ \eta_{1N}^k \geq g_{1N}^l x_N^k + h_{1N}^l u_N + f_N^l \quad : l \in I_{1N}^k \end{aligned} \quad (2.6)$$

where the set I_{1N}^k consists of all indices of subgradients of the objective function at stage N, at iteration k.

Linear programming subproblems of all other stages have essentially the same structure. The subproblem in stage t, $t < N$ is given by

$$\begin{aligned} \min\ & \eta_{1t}^k + \eta_{2t}^k \\ s.t.\ & \eta_{1t}^k \geq g_{1t}^l x_t^k + h_{1t}^l u_t + f_t^l \ : \ l \in I_{1t}^k \\ & \eta_{2t}^k \geq g_{2t}^l x_t^k + h_{2t}^l u_t + v_t^l \ : \ l \in I_{2t}^k \end{aligned} \tag{2.7}$$

In (2.7) the sets I_{1t}^k and I_{2t}^k are the same as the sets defined in (2.2) and (2.4). Note that if we do not have an appropriate rule to shrink sets I_{1t}^k and I_{2t}^k from time to time, these subproblems will grow as the iterations proceed. Subgradient dropping schemes are therefore necessary and will be described subsequently.

The quadratic programming subproblem

The quadratic programming subproblem has essentially all the elements of the linear programming subproblem. In addition, the objective function consists of a quadratic term representing the Euclidean distance from the current control. Thus the QP subproblem for the last stage (N) is,

$$\begin{aligned} \min\ & \eta_{1N}^k + \frac{1}{2}\|u_N - u_N^k\|^2 \\ s.t.\ & \eta_{1N}^k \geq g_{1N}^l x_N^k + h_{1N}^l u_N + f_N^l \ : \ l \in I_{1N}^k \end{aligned} \tag{2.8}$$

Similarly, the QP subproblem of stage t, $t < N$ is,

$$\begin{aligned} \min\ & \eta_{1t}^k + \eta_{2t}^k + \frac{1}{2}\|u_t - u_t^k\|^2 \\ s.t.\ & \eta_{1t}^k \geq g_{1t}^l x_t^k + h_{1t}^l u_t + f_t^l \ : \ l \in I_{1t}^k \\ & \eta_{2t}^k \geq g_{2t}^l x_t^k + h_{2t}^l u_t + v_t^l \ : \ l \in I_{2t}^k \end{aligned} \tag{2.9}$$

The function $\frac{1}{2}\|u_t - u_t^k\|^2$ represents the Euclidean distance between u_t and u_t^k. Although the quadratic programming subproblem may yield better results than the linear programming subproblem, it needs more effort to solve. Thus one subproblem may be better in one situation and the other may be better in another situation.

Computation of relationship of state and control vector perturbations

Once the subproblem corresponding to stage t ($t > 1$) is solved the binding constraints can be used in the computation of the optimal control of the subproblem (u_t) for a given initial state (x_t). Define u_t^+, η_{1t}^+ and η_{2t}^+ as the optimal values of the corresponding variables of the appropriate subproblem (2.6), (2.7), (2.8) or (2.9). Also define λ_{1t}^l and λ_{2t}^l as the corresponding dual variables of the subproblem constraints. Then

$$(\eta_{1t} - \eta_{1t}^+) - g_{1t}^l(x_t - x_t^k) - h_{1t}^l(u_t - u_t^+) = 0, \quad l \in I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\}$$

and

$$(\eta_{2t} - \eta_{2t}^+) - g_{2t}^l(x_t - x_t^k) - h_{2t}^l(u_t - u_t^+) = 0, \quad l \in I_{2t}^k \cap \{i : \lambda_{2t}^i > 0\}$$

which yields,

$$\eta_{1t} - h_{1t}^l u_t = g_{1t}^l x_t + (\eta_{1t}^+ - g_{1t}^l x_t^k - h_{1t}^l u_t^+), \quad l \in I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\} \qquad (2.10)$$

$$\eta_{2t} - h_{2t}^l u_t = g_{2t}^l x_t + (\eta_{2t}^+ - g_{2t}^l x_t^k - h_{2t}^l u_t^+), \quad l \in I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\} \qquad (2.11)$$

Next we define the matrices H_{1t}, H_{2t}, G_{1t} and G_{2t} as follows.

$$H_{1t} = \begin{pmatrix} h_{1t}^{l_1} \\ \vdots \\ h_{1t}^{l_p} \end{pmatrix}, \quad G_{1t} = \begin{pmatrix} g_{1t}^{l_1} \\ \vdots \\ g_{1t}^{l_p} \end{pmatrix}, \quad \{l_1, \ldots, l_p\} \equiv I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\}$$

$$H_{2t} = \begin{pmatrix} h_{2t}^{l_1} \\ \vdots \\ h_{2t}^{l_p} \end{pmatrix}, \quad G_{2t} = \begin{pmatrix} g_{2t}^{l_1} \\ \vdots \\ g_{2t}^{l_p} \end{pmatrix}, \quad \{l_1, \ldots, l_p\} \equiv I_{2t}^k \cap \{i : \lambda_{2t}^i > 0\}$$

Also denote $\mathbf{1}$ as a vector of ones and $\mathbf{0}$ as a vector of zeros of appropriate length. Then assuming that the matrix $\begin{pmatrix} 1 & 0 & -H_{1t} \\ 0 & 1 & -H_{2t} \end{pmatrix}$ is invertible, (i.e. the appropriate subproblem satisfies the Haar condition at optimality, Madsen and Jacobsen [1978]), we have

$$\begin{aligned} \eta_{1t} &= A^k_{\eta_{1t}} x_t + B^k_{\eta_{1t}} \\ \eta_{2t} &= A^k_{\eta_{2t}} x_t + B^k_{\eta_{2t}} \\ u_t &= A^k_{u_t} x_t + B^k_{u_t} \end{aligned} \tag{2.12}$$

where

$$\begin{pmatrix} A^k_{\eta_{1t}} \\ A^k_{\eta_{2t}} \\ A^k_{u_t} \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & -H_{1t} \\ 0 & 1 & -H_{2t} \end{pmatrix}^{-1} \begin{pmatrix} G_{1t} \\ G_{2t} \end{pmatrix}$$

and

$$\begin{pmatrix} B^k_{\eta_{1t}} \\ B^k_{\eta_{2t}} \\ B^k_{du_t} \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & -H_{1t} \\ 0 & 1 & -H_{2t} \end{pmatrix}^{-1} \begin{pmatrix} (\eta^+_{1t} - G_{1t}.x^k_t - H_{1t}.u^+_t) \\ (\eta^+_{2t} - G_{2t}.x^k_t - H_{2t}.u^+_t) \end{pmatrix}$$

Thus, the backward run generates the strategies (2.12) for $t = N, N-1, \ldots, 1$.

The forward run

Once the backward run is completed, the forward run can be started from stage 1. The forward run involves a line search procedure, in which we choose a line search parameter, compute a new policy and test for acceptability. We will use (2.12) and the state transition function recursively to compute the state trajectory x_t and controls u_t. The new policy generated by the forward run is tested for a sufficient reduction condition, which we will discuss next.

Sufficient reduction condition

It is usually necessary to monitor the actual improvement of the objective function value to guarantee the convergence of descent algorithms. Suppose that the predicted reduction of the objective function value at iteration k is ρ_k. In our algorithm, the predicted objective function value at a given iteration is $\eta_{11} + \eta_{21}$. Therefore the predicted reduction is given by,

$$\rho_k = \eta_{11}^+ + \eta_{21}^+ - \eta_{11}^k(x_1^k, u_1^k) - \eta_{21}^k(x_1^k, u_1^k) \tag{2.13}$$

where η_{11}^+ and η_{21}^+ are the values of η_{11} and η_{21} respectively at the optimality of subproblem of stage 1. If the actual reduction is r_k, then for the acceptance it should satisfy $r_k \geq \rho_k \mu_l$. where $0 < \mu_l \leq 1$ is the line search parameter. The choice of the line search parameter is arbitrary. For our prototype, we fixed this parameter to 1.

The basic algorithm

Before proving convergence of the algorithm, we summarize the basic scheme. The piecewise linear approximation of the stagewise objective function is updated using (2.3), and the piecewise linear approximation of the value function is updated using (2.5). Since the basic algorithm uses all previously generated subgradient inequalities to form the subproblem, the size of the subproblems grows linearly.

Algorithm 2.1: Basic algorithm

Step 0. (Initialization)

Given a starting point x_0 and a nominal control policy $u^0 = (u_0^0, \ldots, u_N^0)$,

choose μ_l (line search parameter): $0 < \mu_l \leq 1$,

compute state trajectory and objective function value for (x_0, u^0).

Step 1. (Backward run)

Repeat for all stages; stage $N, N-1, \ldots, 1$. {

Update piecewise linear approximations using (2.3) and (2.5).

Prepare and solve corresponding subproblem (LP or QP).

Compute affine strategy function. }

Step 2. (Forward run)

If $u_t^+ = u_t^k$ for all t then (optimum) stop.

Otherwise compute new trajectory and objective function value.

If $r_k > \rho_k \mu_l$ then accept new iterate and goto step 1.

Otherwise accept null step and goto step 1.

Convergence properties of basic algorithm

The analysis of the convergence properties of the basic algorithm is given in this section. Note that due to the dynamic nature of our cutting plane algorithm, standard proofs of cutting plane algorithms cannot be used. In the following proof we assume that there exists a stationary point for the problem 2.1. Upon executing the algorithm one of the following outcomes is possible.

1. The algorithm stops at iteration K with (x^K, u^K).
2. The algorithm generates an infinite sequence (x^k, u^k), $k \in 1, \ldots, \infty$
3. The algorithm generates an infinite sequence of null steps after iteration K.

The algorithmic properties are revealed by analyzing each of these cases separately.

Case 1. Finite terminating sequence

In this section we will prove that if the algorithm terminates at iteration K after generating a finite number of points, then (x^K, u^K) is stationary for the overall problem. For convex problems this ensures global optimality of (x^K, u^K).

The algorithm stops at iteration K implies that, at iteration K we have $u_t^+ = u_t^K$ for all t. For $j = 1, \ldots, N-1$, let us define

$$\underline{U}_j(x_j, u_j) \equiv (U_{j+1}(x_j, u_j), \ldots, U_N(x_j, u_j))$$

and

$$\underline{X}_j(x_j, u_j) \equiv (X_{j+1}(x_j, u_j), \ldots, X_N(x_j, u_j))$$

where the components $X_t(x_j, u_j)$ and $U_t(x_j, u_j)$ are defined in a recursive manner, as follows.

$$\begin{aligned} X_{j+1}(x_j, u_j) &= T_j(x_j, u_j) \\ X_{t+1}(x_j, u_j) &= T_t(X_t(x_j, u_j), U_t(x_j, u_j)) \quad \text{for } t = j+1, \ldots, N-1 \\ U_t(x_j, u_j) &= A_{u_t}^k X_t(x_j, u_j) + B_{u_t}^k \quad \text{for } t = j+1, \ldots, N. \end{aligned} \tag{2.14}$$

Note that $\underline{U}_j(x_j, u_j)$ is a partial strategy which may be generated by the forward run at iteration k, given that the state and control vectors at stage j are x_j and u_j respectively.

Also define $\hat{X}_{j+1}(x_j, u_j)$ and $\hat{U}_{j+1}(x_j, u_j)$ as

$$\begin{aligned} \hat{X}_{j+1}(x_j, u_j) &= x_{j+1}^k + J_{u_j}(u_j - u_j^k) + J_{x_j}(x_j - x_j^k) \\ \hat{U}_{j+1}(x_j, u_j) &= A_{u_{j+1}}^k \hat{X}_{j+1}(x_j, u_j) + B_{u_{j+1}}^k \end{aligned} \tag{2.15}$$

where J_{u_j} and J_{x_j} are the Jacobian matrices of the transition relation $T_j(x_j, u_j)$ at (x_j^k, u_j^k) with respect to u_j and x_j. $A_{u_{j+1}}^k$ and $B_{u_{j+1}}^k$ are as defined in (2.12) with $t = j + 1$. It is important to distinguish between the functions $X_{j+1}(x_j, u_j)$, $U_{j+1}(x_j, u_j)$ and the functions $\hat{X}_{j+1}(x_j, u_j)$ and $\hat{U}_{j+1}(x_j, u_j)$. The former uses the exact transition relation, where as the latter uses a linearization via the Jacobian matrices J_{u_j} and J_{x_j}.

In the following analysis we will use the notation

$$\left(\sum_{t=p}^{N} f_t\right)(x_j, u_j)$$

to represent the function

$$\sum_{t=p}^{N} f_t(X_t(x_j, u_j), U_t(x_j, u_j))$$

where $p \geq j$. Since this function is well defined, appropriate operations can be performed.

In lemmas 2.1 and 2.2, we will show that the piecewise linear approximations generated are such that at any given point the subdifferential of the PL approximation is a subset of the subdifferential of the actual value function. This result together with lemma 2.3 is then used to show that (x^K, u^K) is stationary to the overall problem.

Lemma 2.1 Suppose that $\eta_{1(N-1)}^k(x_{N-1}^k, u_{N-1})$ and $\eta_{2(N-1)}^k(x_{N-1}^k, u_{N-1})$ are defined as in (2.2) and (2.4) respectively, and that (2.3) and (2.5) were used for updating them. Then at iteration k,

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})(x^k_{N-1}, u_{N-1}) \leq \left(\sum_{t=N-1}^{N} f_t\right)(x^k_{N-1}, u_{N-1}) + o(\epsilon)$$

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})(x^k_{N-1}, u^k_{N-1}) = \left(\sum_{t=N-1}^{N} f_t\right)(x^k_{N-1}, u^k_{N-1})$$

where $\epsilon = \|u_{N-1} - u^k_{N-1}\|$.

Proof: Since $f_{N-1}(x_{N-1}, u_{N-1})$ is a convex function and $\eta^k_{1(N-1)}(x_{N-1}, u_{N-1})$ is an outer envelop generated by subgradients of $f_{N-1}(x_{N-1}, u_{N-1})$,

$$\eta^k_{1(N-1)}(x^k_{N-1}, u_{N-1}) \leq f_{N-1}(x^k_{N-1}, u_{N-1}), \tag{2.16}$$

and furthermore since $\eta^k_{1(N-1)}$ contains a subgradient generated at (x^k_{N-1}, u^k_{N-1}),

$$\eta^k_{1(N-1)}(x^k_{N-1}, u^k_{N-1}) = f_{N-1}(x^k_{N-1}, u^k_{N-1}). \tag{2.17}$$

Also using (2.5) and $x^k_N = T_{N-1}(x^k_{N-1}, u^k_{N-1})$, since $I^k_{2N} \equiv \emptyset$ we have

$$\eta^k_{2(N-1)}(x^k_{N-1}, u^k_{N-1}) = \eta_{1N}(x^k_N, u^k_N) = f_N(x^k_N, u^k_N). \tag{2.18}$$

Furthermore,

$$\begin{aligned} \eta^k_{2(N-1)}(x^k_{N-1}, u_{N-1}) &= \eta_{1N}(\hat{X}_N(x^k_{N-1}, u_{N-1}), \hat{U}_N(x^k_{N-1}, u_{N-1})) \\ &\leq f_N(\hat{X}_N(x^k_{N-1}, u_{N-1}), \hat{U}_N(x^k_{N-1}, u_{N-1})) \end{aligned} \tag{2.19}$$

From (2.14)

$$\begin{aligned} X_N(x^k_{N-1}, u_{N-1}) &= T_{N-1}(x^k_{N-1}, u_{N-1}) \\ U_N(x^k_{N-1}, u_{N-1}) &= A^k_{u_N} X_N(x^k_{N-1}, u_{N-1}) + B^k_{u_N}, \end{aligned}$$

and from (2.15)

$$\begin{aligned} \hat{X}_N(x^k_{N-1}, u_{N-1}) &= x^k_N + J_{u_{N-1}}(u_{N-1} - u^k_{N-1}) \\ \hat{U}_N(x^k_{N-1}, u_{N-1}) &= A^k_{u_N} \hat{X}_N(x^k_{N-1}, u_{N-1}) + B^k_{u_N}. \end{aligned}$$

Then since the transition function $T_{N-1}(x_{N-1}, u_{N-1})$ is twice continuously differentiable, we get

$$\begin{aligned} \hat{X}_N(x^k_{N-1}, u_{N-1}) &= X_N(x^k_{N-1}, u_{N-1}) + o(\epsilon) \\ \hat{U}_N(x^k_{N-1}, u_{N-1}) &= U_N(x^k_{N-1}, u_{N-1}) + o(\epsilon), \end{aligned}$$

where $\epsilon = \|u_{N-1} - u^k_{N-1}\|$.

Therefore

$$\begin{aligned} & f_N(\hat{X}_N(x^k_{N-1}, u_{N-1}), \hat{U}_N(x^k_{N-1}, u_{N-1})) \\ &= f_N(X_N(x^k_{N-1}, u_{N-1}) + o(\epsilon), U_N(x^k_{N-1}, u_{N-1}) + o(\epsilon)). \end{aligned} \tag{2.20}$$

Since functions $f_t(\cdot)$ are locally lipschitz, (2.19) and (2.20) implies the following.

$$\eta^k_{2(N-1)}(x^k_{N-1}, u_{N-1}) \leq f_N(X_N(x^k_{N-1}, u_{N-1}), U_N(x^k_{N-1}, u_{N-1})) + o(\epsilon) \tag{2.21}$$

Therefore (2.16) and (2.21) leads to,

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})(x^k_{N-1}, u_{N-1}) \leq \left(\sum_{t=N-1}^{N} f_t \right)(x^k_{N-1}, u_{N-1}) + o(\epsilon).$$

From (2.17) and (2.18),

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})(x^k_{N-1}, u^k_{N-1}) = \left(\sum_{t=N-1}^{N} f_t \right)(x^k_{N-1}, u^k_{N-1}). \blacksquare$$

The above lemma result in the following Corollary.

Corollary 2.1 The directional derivative

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})'(x^k_{N-1}, u^k_{N-1}; 0, y) \leq \left(\sum_{t=N-1}^{N} f_t \right)'(x^k_{N-1}, u^k_{N-1}; 0, y)$$

for all directions y. Furthermore

$$\partial(\eta^k_{1(N-1)} + \eta^k_{2(N-1)}) \subseteq \partial \left(\sum_{j=N-1}^{N} f_j \right) \quad \text{at} \quad (x^k_{N-1}, u^k_{N-1}).$$

Proof: Computing the directional derivative of $(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})$ at (x^k_{N-1}, u^k_{N-1}) in any direction $(0, y)$, from lemma 2.1 we have,

$$\lim_{\epsilon \to 0} \frac{(\eta^k_{1(N-1)}+\eta^k_{2(N-1)})(x^k_{N-1}, u^k_{N-1}+\epsilon y)-(\eta^k_{1(N-1)}+\eta^k_{2(N-1)})(x^k_{N-1}, u^k_{N-1})}{\epsilon}$$

$$\leq \lim_{\epsilon \to 0} \frac{\left(\sum_{t=N-1}^{N} f_t\right)(x^k_{N-1}, u^k_{N-1}+\epsilon y) + o(\epsilon) - \left(\sum_{t=N-1}^{N} f_t\right)(x^k_{N-1}, u^k_{N-1})}{\epsilon}.$$

Therefore,

$$(\eta^k_{1(N-1)} + \eta^k_{2(N-1)})'(x^k_{N-1}, u^k_{N-1}; 0, y) \leq \left(\sum_{t=N-1}^{N} f_t\right)'(x^k_{N-1}, u^k_{N-1}; 0, y)$$

for all directions y.

Which implies

$$\partial(\eta^k_{1(N-1)} + \eta^k_{2(N-1)}) \subseteq \partial\left(\sum_{j=N-1}^{N} f_j\right) \quad \text{at } (x^k_{N-1}, u^k_{N-1}). \blacksquare$$

Lemma 2.2 If $\eta^k_{1(t)}$ and $\eta^k_{2(t)}$ are updated using (2.3) and (2.5) respectively, and

$$(\eta^k_{1(t+1)} + \eta^k_{2(t+1)})(x^k_{t+1}, u_{t+1}) \leq \left(\sum_{j=t+1}^{N} f_j\right)(x^k_{t+1}, u_{t+1}) + o(\epsilon),$$

$$(\eta^k_{1(t+1)} + \eta^k_{2(t+1)})(x^k_{t+1}, u^k_{t+1}) = \left(\sum_{j=t+1}^{N} f_j\right)(x^k_{t+1}, u^k_{t+1})$$

then

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u_t) \leq \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u_t) + o(\epsilon)$$

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u^k_t) = \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u^k_t).$$

Proof: Since $f_t(x_t, u_t)$ is convex and $\eta^k_{1(t)}$ is a support of $f_t(x_t, u_t)$,

$$\eta^k_{1(t)}(x^k_t, u_t) \leq f_t(x^k_t, u_t) \tag{2.22}$$

and since $\eta^k_{1(t)}$ consists of a subgradient inequality generated at (x^k_t, u^k_t),

$$\eta^k_{1(t)}(x^k_t, u^k_t) = f_t(x^k_t, u^k_t) \tag{2.23}$$

By definition,

$$\eta^k_{2(t)}(x^k_t, u_t) = (\eta^k_{1(t+1)} + \eta^k_{2(t+1)})(\hat{X}_{t+1}(x^k_t, u_t), \hat{U}_{t+1}(x^k_t, u_t)) \tag{2.24}$$

and

$$\begin{aligned} \eta^k_{2(t)}(x^k_t, u^k_t) &= (\eta^k_{1(t+1)} + \eta^k_{2(t+1)})(\hat{X}_{t+1}(x^k_t, u^k_t), \hat{U}_{t+1}(x^k_t, u^k_t)) \\ &= \left(\sum_{j=t+1}^{N} f_j\right)(x^k_t, u^k_t) \end{aligned} \tag{2.25}$$

Again from (2.14) and (2.15)

$$\begin{aligned} \hat{X}_{t+1}(x^k_t, u_t) &= X_{t+1}(x^k_t, u_t) + o(\epsilon) \\ \hat{U}_{t+1}(x^k_t, u_t) &= U_{t+1}(x^k_t, u_t) + o(\epsilon) \end{aligned}$$

Then

$$\begin{aligned} \eta_{2(t)}(x^k_t, u_t) &= (\eta_{1(t+1)} + \eta_{2(t+1)})(X_{t+1}(x^k_t, u_t), U_{t+1}(x^k_t, u_t)) + o(\epsilon) \\ &\leq \left(\sum_{j=t+1}^{N} f_j\right)(x^k_t, u_t) + o(\epsilon). \end{aligned} \tag{2.26}$$

From (2.22) and (2.26),

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u_t) \leq \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u_t) + o(\epsilon).$$

From (2.23) and (2.25),

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u^k_t) \;=\; \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u^k_t). \blacksquare$$

The above lemma proves that the approximate objective function of stage t subproblem is a lower support of the collective objective function of stages t to N in the neighbourhood of (x^k_t, u^k_t). This result leads to the following corollary.

Corollary 2.2

$$\partial(\eta^k_{1(1)} + \eta^k_{2(1)}) \;\subseteq\; \partial\left(\sum_{j=1}^{N} f_j\right) \quad \text{at} \;\; (x^k_1, u^k_1).$$

Proof: Omitted. ■

Lemma 2.3 Suppose that the basic algorithm terminates at iteration K with (x^K, u^K). Then (x^K, u^K) is stationary for problem 2.1.

Proof: Since $u^+_t = u^K_t$ for all t,

$$0 \;\in\; \partial(\eta^K_{1(1)} + \eta^K_{2(1)}) \quad \text{at} \;\; (x^K_1, u^K_1)$$

But corollary 2.1 implies that

$$\partial(\eta^K_{1(1)} + \eta^K_{2(1)}) \;\subseteq\; \partial\left(\sum_{j=1}^{N} f_j\right) \quad \text{at} \;\; (x^K_1, u^K_1).$$

Therefore

$$0 \;\in\; \partial\left(\sum_{j=1}^{N} f_j\right) \quad \text{at} \;\; (x^K_1, u^K_1).$$

and it follows that (x^K, u^K) is stationary. ■

Next we consider the cases in which the algorithm does not stop in finitely many iterations. The following two cases show that any accumulation point of the sequence generated by the basic algorithm is a stationary point of problem 2.1.

Case 2. Infinite sequence with descent steps

Here we will prove that for any infinite sequence of points generated by algorithm 2.1 all subsequences converge to stationary points. To start with let us first define the index set κ as

$$\kappa = \{k : (x^k, u^k) \neq (x^{k+1}, u^{k+1})\}$$

i.e. the set κ includes all the iteration numbers at which there is a descent.

Lemma 2.4 Assume that problem 2.1 has a finite minimum. Then any infinite sequence $(\eta^k_{1(1)} + \eta^k_{2(1)})(x^k_1, u^k_1)$ $k = 1, \cdots, \infty$ generated by the algorithm 2.1 has exactly one accumulation point.

Proof: By construction

$$(\eta^k_{1(1)} + \eta^k_{2(1)})(x^k_1, u^k_1) = \left(\sum_{j=1}^{N} f_j\right)(x^k_1, u^k_1)$$

and

$$\left(\sum_{j=1}^{N} f_j\right)(x^k_1, u^k_1) \leq \left(\sum_{j=1}^{N} f_j\right)(x^{k-1}_1, u^{k-1}_1).$$

Then since $\left(\sum_{j=1}^{N} f_j\right)(x_1, u_1)$ is bounded from below, $\left(\sum_{j=1}^{N} f_j\right)(x^k_1, u^k_1)$ $k = 1, \cdots, \infty$ has exactly one accumulation point. ■

Lemma 2.5 Suppose that for some index set M, $\lim_{\substack{k_l \in \kappa \\ l \in M}} (x^{k_l}_1, u^{k_l}_1) = (\bar{x}_1, \bar{u}_1)$, $\lim_{\substack{k_l \in \kappa \\ l \in M}} (x^{k_l+1}_1, u^{k_l+1}_1) = (\tilde{x}_1, \tilde{u}_1) \neq (\bar{x}_1, \bar{u}_1)$, and $0 \notin \partial\left(\sum_{j=1}^{N} f_j\right)(\bar{x}_1, \bar{u}_1)$ then $\left(\sum_{j=1}^{N} f_j\right)(\bar{x}_1, \bar{u}_1) > \left(\sum_{j=1}^{N} f_j\right)(\tilde{x}_1, \tilde{u}_1)$.

Proof: Since $0 \notin \partial \left(\sum_{j=1}^{N} f_j \right) (\bar{x}_1, \bar{u}_1)$ there exists $(\hat{x}_1, \hat{u}_1)$ such that

$$\left(\sum_{j=1}^{N} f_j \right) (\bar{x}_1, \bar{u}_1) > \left(\sum_{j=1}^{N} f_j \right) (\hat{x}_1, \hat{u}_1).$$

Since $(x_1^{k_l+1}, u_1^{k_l+1})$ minimizes $(\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x, u)$, there exists $\lambda_{k_l} > 0$ such that

$$\begin{aligned} (\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l+1}, u_1^{k_l+1}) < & (\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \\ & + \lambda \max \left\{ \xi' d^{k_l} | \xi \in \partial(\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \right\} \\ & \forall \lambda \in [0, \lambda_{k_l}], \end{aligned} \tag{2.27}$$

where $d^{k_l} = (\hat{x}_1, \hat{u}_1) - (x_1^{k_l}, u_1^{k_l})$.

Then the sufficient reduction requirement of the algorithm implies the following.

$$\begin{aligned} & (\eta_{1(1)}^{k_l+1} + \eta_{2(1)}^{k_l+1})(x_1^{k_l+1}, u_1^{k_l+1}) - (\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \\ & \leq \mu_l \left((\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l+1}, u_1^{k_l+1}) - (\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \right) \\ & < \mu_l \lambda \max \left\{ \xi' d^{k_l} | \xi \in \partial(\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \right\}, \quad \forall \lambda \in [0, \lambda_{k_l}]. \end{aligned}$$

Hence

$$\begin{aligned} & \lim_{\substack{k_l \in \kappa \\ l \in M}} \left((\eta_{1(1)}^{k_l+1} + \eta_{2(1)}^{k_l+1})(x_1^{k_l+1}, u_1^{k_l+1}) - (\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \right) \\ & < \liminf_{\substack{k_l \in \kappa \\ l \in M}} \left(\mu_l \lambda_{k_l} \max \left\{ \xi' d^{k_l} | \xi \in \partial(\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})(x_1^{k_l}, u_1^{k_l}) \right\} \right). \end{aligned}$$

Since

$$(\eta_{1(1)}^{k_l} + \eta_{2(1)}^{k_l})'(x_1^{k_l}, u_1^{k_l}; d) \leq \left(\sum_{j=1}^{N} f_j \right)' (x_1^{k_l}, u_1^{k_l}; d),$$

and

$$\limsup \left(\sum_{j=1}^{N} f_j \right)' (x_1^{k_l}, u_1^{k_l}; d^{k_l}) \leq \left(\sum_{j=1}^{N} f_j \right)' (\bar{x}_1, \bar{u}_1; d^*) < 0,$$

where $d^* = (\hat{x}_1, \hat{u}_1) - (\bar{x}_1, \bar{u}_1)$,

it follows that $\liminf \lambda_{k_l} > 0$.

Since $(x_1^{k_l+1}, u_1^{k_l+1}) \equiv (x_1^{k_l+1}, u_1^{k_l+1})$, and choosing $\lambda = \liminf \lambda_{k_l}$, we have

$$\left(\sum_{j=1}^{N} f_j\right)(\tilde{x}_1, \tilde{u}_1) - \left(\sum_{j=1}^{N} f_j\right)(\bar{x}_1, \bar{u}_1)$$

$$< \left(\mu_l \lambda \max\left\{\xi' d^* | \xi \in \partial(\sum_{j=1}^{N} f_j)(\bar{x}_1, \bar{u}_1)\right\}\right).$$

This implies that $\left(\sum_{j=1}^{N} f_j\right)(\bar{x}_1, \bar{u}_1) > \left(\sum_{j=1}^{N} f_j\right)(\tilde{x}_1, \tilde{u}_1)$. ■

Lemma 2.6 Every accumulation point generated by algorithm 2.1 is stationary for problem 2.1.

Proof: Let $(\bar{x}, \bar{u})$ be an accumulation point of (x^k, u^k) $k \in \kappa$, and for some index set M, let $\lim_{l \in M}(x^{k_l}, u^{k_l}) = (\bar{x}, \bar{u})$. Let $(\tilde{x}, \tilde{u})$ denote an accumulation point of (x^{k_l+1}, u^{k_l+1}). Suppose that $(\bar{x}, \bar{u})$ is not stationary. Then from Lemma 2.5,

$$(\sum_{j=1}^{N} f_j)(\bar{x}_1, \bar{u}_1) > (\sum_{j=1}^{N} f_j)(\tilde{x}_1, \tilde{u}_1).$$

But from Lemma 2.4,

$$(\sum_{j=1}^{N} f_j)(\bar{x}_1, \bar{u}_1) = (\sum_{j=1}^{N} f_j)(\tilde{x}_1, \tilde{u}_1)$$

which is a contradiction. Therefore the result. ■

Case 3. Infinite sequence of null steps

Given that the algorithm 2.1 generates an infinite sequence of null steps after iteration K, here we will first show that the sequence of predicted objective function value reductions, that is, ρ_k converges to zero (see (2.13)). Here (x_1^{k+}, u_1^{k+}) will be used to represent the candidate policy generated at iteration k, L will denote the rank of lipschitz function $\left(\sum_{t=1}^{N} f_t\right)(x_1, u_1)$.

Lemma 2.7 Given that the algorithm 2.1 generates an infinite sequence of null steps, the candidate policy (x_1^{k+}, u_1^{k+}) at iteration $k > K$ cannot be found in $\bigcup_{i\in\{1,\ldots,k-1\}} \mathcal{B}\left((x_1^{i+}, u_1^{i+}), \frac{(1-\mu_l)}{2L}\rho_i\right)$, where $\mathcal{B}\left((x_1^{i+}, u_1^{i+}), \frac{(1-\mu_l)}{2L}\rho_i\right)$ represents a ball of radius $\frac{(1-\mu_l)}{2L}\rho_i$ centered at (x_1^{i+}, u_1^{i+}).

Proof: Note that (x_1^{i+}, u_1^{i+}) is the candidate policy at iteration i implies that, at any iteration k such that $k > K$ and $k > i$,

$$(\eta_{1(1)}^k + \eta_{2(1)}^k)(x_1^{i+}, u_1^{i+}) = \left(\sum_{t=1}^{N} f_t\right)(x_1^{i+}, u_1^{i+})$$

and $\forall\, (x_1, u_1) \in \mathcal{B}\left((x_1^{i+}, u_1^{i+}), \epsilon\right)$

$$\left|(\eta_{1(1)}^k + \eta_{2(1)}^k)(x_1, u_1) - \left(\sum_{t=1}^{N} f_t\right)(x_1, u_1)\right| \leq 2L\epsilon.$$

Suppose that the candidate policy $(x_1^{k+}, u_1^{k+}) \in \mathcal{B}\left((x_1^{i+}, u_1^{i+}), \epsilon\right)$. Then the actual objective function value reduction, denoted r_k satisfies

$$|\rho_k - r_k| \leq 2L\epsilon.$$

Therefore

$$\rho_k - 2L\epsilon \leq r_k.$$

Since (x_1^{k+}, u_1^{k+}) produces a null step

$$r_k \;<\; \rho_k \mu_l.$$

Therefore

$$\rho_k - 2L\epsilon \;\leq\; r_k \;<\; \rho_k \mu_l,$$

which implies that

$$2L\epsilon \;>\; \rho_k - \rho_k \mu_l,$$

or

$$\epsilon \;>\; \frac{(1-\mu_l)}{2L}\rho_k.$$

Therefore

$$(x_1^{k+}, u_1^{k+}) \notin \mathcal{B}\left((x_1^{i+}, u_1^{i+}), \frac{(1-\mu_l)}{2L}\rho_i\right) \qquad \forall\, i < k.$$

Hence

$$(x_1^{k+}, u_1^{k+}) \notin \bigcup_{i\in\{1,\ldots,k-1\}} \mathcal{B}\left((x_1^{i+}, u_1^{i+}), \frac{(1-\mu_l)}{2L}\rho_i\right). \blacksquare$$

Corollary 2.3 Assume that $\|(x_1, u_1)\| \to \infty \Rightarrow \left(\sum_{t=1}^{N} f_t\right)(x_1, u_1) \to \infty$. If algorithm 2.1 generates only null steps after iteration K, then

$$\lim_{k\to\infty} \rho_k \;=\; 0.$$

Proof: Under the hypothesis, the sequence (x_1^{k+}, u_1^{k+}) is bounded. Hence lemma 2.7 implies that

$$\lim_{k\to\infty} \frac{(1-\mu_l)}{2L}\rho_k = 0$$

Therefore

$$\lim_{k\to\infty} \rho_k = 0. \blacksquare$$

Lemma 2.8 Suppose that the algorithm 2.1 generates an infinite sequence of null points after generating (x^K, u^K). Let the hypothesis of Corollary 2.3 hold. Then (x^K, u^K) is stationary to the problem 2.1.

Proof: Suppose that predicted objective function value drop at iteration k is ρ_k. Then from Corollary 2.3,

$$\lim_{k \to \infty} \rho_k \to 0.$$

Since algorithm generates only null points after iteration K, for all $k \geq K$,

$$(\eta^k_{1(1)} + \eta^k_{2(1)})(x^K_1, u^K_1) = \left(\sum_{j=1}^{N} f_j \right)(x^K_1, u^K_1),$$

and since $(\eta^k_{1(1)} + \eta^k_{2(1)})(x_1, u_1)$ is a lower support of $\left(\sum_{j=1}^{N} f_j \right)(x_1, u_1)$,

$$(\eta^k_{1(1)} + \eta^k_{2(1)})(x_1, u_1) \leq \left(\sum_{j=1}^{N} f_j \right)(x_1, u_1)$$

Also the fact that the predicted reduction at iteration k is ρ_k implies that, $\forall\, (x_1, u_1)$,

$$\left(\sum_{j=1}^{N} f_j \right)(x^K_1, u^K_1) - \rho_k \leq (\eta^k_{1(1)} + \eta^k_{2(1)})(x_1, u_1)$$

$$\leq \left(\sum_{j=1}^{N} f_j \right)(x_1, u_1).$$

Thus as $k \to \infty$,

$$\lim_{k \to \infty} \left(\sum_{j=1}^{N} f_j \right)(x^K_1, u^K_1) \leq \lim_{k \to \infty} \left(\sum_{j=1}^{N} f_j \right)(x_1, u_1).$$

Therefore (x^K, u^K) is stationary. ■

Theorem 2.1 : Under the assumptions (1) Transition functions are twice continuously differentiable, (2) Stagewise objective functions are convex and (3) A finite solution exists, every convergent subsequence generated by Algorithm 2.1 converges to a stationary point of problem 2.1.

Proof : Under the assumptions made above algorithm will generate a sequence of points that falls into one of the classes discussed above. According to lemma 2.3, 2.6 and 2.8 all convergent subsequences converges to a stationary point. Thus the result. ■

The algorithm with subgradient selection

In this section we will describe a variation of algorithm 2.1 which implements subgradient selection techniques. The main motivation for the use of subgradient selection technique is quite clear. In the basic algorithm, since we keep all the information generated, the subproblem becomes increasingly difficult to solve as the iteration progress. In the subgradient selection technique we eliminate, to a certain extent, some of the redundant information without loosing much of the actual convergence properties of the algorithm. Selection of the subgradient inequalities is done at the step of preparing subproblems of each stage. At any iteration only those subgradient inequalities with positive dual variables are retained in preparing subproblems of the following iteration. This helps to carry necessary information from iteration to iteration as well as from stage to stage while keeping the size of the subproblems small.

Suppose that the candidate policy generated at iteration k is denoted by u_t^+ and the corresponding state by x_t^+. Let $(\xi_{tx}^+, \xi_{tu}^+) \in \partial f_t(x_t^+, u_t^+)$, and let $f_t^+ = f_t(x_t^+, u_t^+)$. Also suppose that the iteration k generates a null step. Then

piecewise linear (PL) approximation of the stagewise objective function is updated as follows.

$$\eta_{1t}^{k+1}(x_t, u_t) = \max \begin{Bmatrix} g_{1t}^l x_t + h_{1t}^l u_t + f_t^l & : l \in I_{1t}^k \\ \xi_{tx}^+(x_t - x_t^+) + \xi_{tu}^+(u_t - u_t^+) + f_t^+ \end{Bmatrix} \tag{2.30}$$

In case of a descent step at iteration k, the following is used for update.

$$\eta_{1t}^{k+1}(x_t, u_t) = \max \begin{Bmatrix} g_{1t}^l x_t + h_{1t}^l u_t + f_t^l & : l \in I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\} \\ \xi_{tx}^+(x_t - x_t^+) + \xi_{tu}^+(u_t - u_t^+) + f_t^+ \end{Bmatrix} \tag{2.31}$$

The PL approximation of the value function is updated as follows regardless of the outcome of iteration k.

$$\eta_{2(t-1)}^k = \eta_{1t}^k(x_t, u_t) + \sum_{l \in I_{2t}^k} \lambda_{2t}^l (g_{2t}^l . x_t + h_{2t}^l . u_t + v_t^l) \tag{2.32}$$

Algorithm 2.2 : Algorithm with subgradient selection

Step 0. (Initialization)

Given a starting point x_0 and a nominal control policy $u^0 = (u_0^0, \ldots, u_N^0)$,

choose μ_l (line search parameter): $0 < \mu_l \leq 1$,

compute state trajectory and objective function value for (x_0, u^0).

Step 1. (Backward run)

Repeat for all stages; stage $N, N-1, \ldots, 1$. {

Update piecewise linear approximations using (2.30) (2.31) and (2.32).

Prepare and solve corresponding subproblem (LP or QP).

Compute affine strategy function. }

Step 2. (Forward run)

If $u_t^+ = u_t^k$ for all t then (optimum) stop.

Otherwise compute new trajectory and objective function value.

If $r_k > \rho_k \mu_l$ then accept new iterate and goto step 1.

Otherwise accept null step and goto step 1.

Convergence properties of algorithm with subgradient selection

For the analysis of convergence properties of the subgradient selection algorithm we may use the same classification of sequences as in the analysis of the basic algorithm. Recall the three classes of sequences

1. A finite sequence terminating at (x^K, u^K).
2. An infinite sequence (x^k, u^k), $k \in 1, \ldots, \infty$ with descent steps.
3. An infinite sequence with null steps after (x^K, u^K).

Case 1. Finite terminating sequence

For the proof of stationarity of terminating point (x^K, u^K) of a finite sequence generated by algorithm 2.2 we will make use of Lemma 2.1. Let us first show that the PL approximation of stagewise objective function and of value function used in algorithm 2.2 satisfy the criteria used in Lemma 2.1.

Lemma 2.9 Suppose that $\eta^k_{1(N-1)}(x^k_{N-1}, u_{N-1})$ and $\eta^k_{2(N-1)}(x^k_{N-1}, u_{N-1})$ are defined as in (2.2) and (2.4) respectively with $t = N-1$ and $x_{N-1} = x^k_{N-1}$, and updated using (2.30) (2.31) and (2.32). Then

$$\begin{aligned}
\eta^k_{1(N-1)}(x^k_{N-1}, u^k_{N-1}) &= f_{N-1}(x^k_N, u^k_N) \\
\eta^k_{1(N-1)}(x^k_{N-1}, u_{N-1}) &\leq f_{N-1}(x^k_{N-1}, u_{N-1}) \\
\eta^k_{2(N-1)}(x^k_{N-1}, u^k_{N-1}) &= f_N(\hat{X}_N(x^k_{N-1}, u^k_{N-1}), \hat{U}_N(x^k_{N-1}, u^k_{N-1})) \\
\eta^k_{2(N-1)}(x^k_{N-1}, u_{N-1}) &\leq f_N(\hat{X}_N(x^k_{N-1}, u_{N-1}), \hat{U}_N(x^k_{N-1}, u_{N-1})).
\end{aligned}$$

Proof: Since $\eta^k_{1(N-1)}$ is an outer envelop of $f_{N-1}(x_{N-1}, u_{N-1})$

$$\eta^k_{1(N-1)}(x^k_{N-1}, u_{N-1}) \leq f_{N-1}(x^k_{N-1}, u_{N-1}),$$

and majorizes a subgradient generated at (x^k_{N-1}, u^k_{N-1})

$$\eta^k_{1(N-1)}(x^k_{N-1}, u^k_{N-1}) = f_{N-1}(x^k_{N-1}, u^k_{N-1}).$$

From (2.32)

$$\eta^k_{2(N-1)}(x^k_{N-1}, u^k_{N-1}) = \eta^k_{1N}(x^k_N, u^k_N)$$

Therefore

$$\eta^k_{2(N-1)}(x^k_{N-1}, u^k_{N-1}) = f_N(x^k_N, u^k_N) = f_N(\hat{X}_N(x^k_{N-1}, u^k_{N-1}), \hat{U}_N(x^k_{N-1}, u^k_{N-1}))$$

and

$$\begin{aligned}\eta^k_{2(N-1)}(x^k_{N-1},u_{N-1}) &\leq \eta^k_{1N}(\hat{X}_N(x^k_{N-1},u_{N-1}),\hat{U}_N(x^k_{N-1},u_{N-1}))\\ &\leq f_N(\hat{X}_N(x^k_{N-1},u_{N-1}),\hat{U}_N(x^k_{N-1},u_{N-1}))\end{aligned}$$

Therefore the result. ■

Lemma 2.10 Given that $\eta^k_{1(t)}$, $\eta^k_{2(t)}$ are updated using (2.30), (2.31) and (2.32), and

$$\begin{aligned}(\eta^k_{1(t+1)}+\eta^k_{2(t+1)})(x^k_{t+1},u_{t+1}) &\leq \left(\sum_{j=t+1}^{N} f_j\right)(x^k_{t+1},u_{t+1})+o(\epsilon)\\ (\eta^k_{1(t+1)}+\eta^k_{2(t+1)})(x^k_{t+1},u^k_{t+1}) &= \left(\sum_{j=t+1}^{N} f_j\right)(x^k_{t+1},u^k_{t+1})\end{aligned}$$

then

$$\begin{aligned}(\eta^k_{1(t)}+\eta^k_{2(t)})(x^k_t,u_t) &\leq \left(\sum_{j=t}^{N} f_j\right)(x^k_t,u_t)+o(\epsilon)\\ (\eta^k_{1(t)}+\eta^k_{2(t)})(x^k_t,u^k_t) &= \left(\sum_{j=t}^{N} f_j\right)(x^k_t,u^k_t)\end{aligned}$$

Proof: Since $f_t(x_t,u_t)$ is convex,

$$\eta^k_{1(t)}(x^k_t,u_t) \leq f_t(x^k_t,u_t) \tag{2.33}$$

From (2.32),

$$\eta^k_{2(t)}(x^k_t,u_t) \leq (\eta^k_{1(t+1)}+\eta^k_{2(t+1)})(\hat{X}_{t+1}(x^k_t,u_t),\hat{U}_{t+1}(x^k_t,u_t)) \tag{2.34}$$

Then

$$\begin{aligned}\eta^k_{2(t)}(x^k_t,u_t) &\leq (\eta^k_{1(t+1)}+\eta^k_{2(t+1)})(X_{t+1}(x^k_t,u_t),U_{t+1}(x^k_t,u_t))+o(\epsilon)\\ &= \left(\sum_{j=t+1}^{N} f_j\right)(x^k_t,u_t)+o(\epsilon).\end{aligned} \tag{2.35}$$

From (2.33) and (2.35),

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u_t) \leq \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u_t) + o(\epsilon).$$

Since $\eta^k_{1(t)}$ consists of a subgradient inequality generated at (x^k_t, u^k_t)

$$\eta^k_{1(t)}(x^k_t, u^k_t) = f_t(x^k_t, u^k_t). \tag{2.36}$$

From (2.32)

$$\begin{aligned} \eta^k_{2(t)}(x^k_t, u^k_t) &= (\eta^k_{1(t+1)} + \eta^k_{2(t+1)})(X_{t+1}(x^k_t, u^k_t), U_{t+1}(x^k_t, u^k_t)) \\ &= \left(\sum_{j=t+1}^{N} f_j\right)(x^k_t, u^k_t). \end{aligned} \tag{2.37}$$

Then

$$(\eta^k_{1(t)} + \eta^k_{2(t)})(x^k_t, u^k_t) = \left(\sum_{j=t}^{N} f_j\right)(x^k_t, u^k_t) \blacksquare$$

Lemma 2.11 Suppose that the algorithm with subgradient aggregation generate a finite sequence of points and stops at iteration K generating (x^K, u^K). Then (x^K, u^K) is stationary to the problem 2.1.

Proof: Follows lemma 2.1, 2.2 and 2.3. ■

Case 2. Infinite sequence of descent steps

Lemma 2.12 Suppose that the algorithm 2.2 generates an infinite sequence of points (x^k, u^k) such that a subsequence of (x^k, u^k); $(x^{l(k)}, u^{l(k)}) \rightarrow (x^*, u^*)$ as $k \rightarrow \infty$. Then (x^*, u^*) is stationary to the problem 2.1.

Proof: Follows lemmas 2.4, 2.5 and 2.6. ■

Case 3. Infinite sequence of null steps

Lemma 2.13 Suppose that the algorithm 2.2 generates an infinite sequence of null points after generating (x^K, u^K). Then (x^K, u^K) is stationary to the problem 2.1.

Proof: Follows lemma 2.8. ■

Theorem 2.2 Under the assumptions (1) Transition functions are twice continuously differentiable (2) Stagewise objective functions are convex and (3) A finite solution exists, every convergent subsequence generated by Algorithm 2.2 converges to a stationary point of problem 2.1.

Proof: Under the assumptions made above algorithm will generate a sequence of points that falls into one of the classes discussed above. According to lemmas 2.11, 2.12, and 2.13 all convergent subsequences converges to a stationary point. Thus the result. ■

The algorithm with subgradient aggregation

In this section we will describe an implementation of a subgradient aggregation technique. The subgradient aggregation takes place when preparing subproblems at each stage. At any iteration previous subgradients are aggregated to produce a single support whenever a new stratergy is accepted for the next iterate. Otherwise the same procedure is used as in basic algorithm. In the above process dual variables are used as weights for aggregation.

Suppose that the iteration k generates a null step. Then the piecewise linear approximation of stagewise objective function is updated as follows.

$$\eta_{1t}^{k+1}(x_t, u_t) = \max \begin{Bmatrix} g_{1t}^l x_t + h_{1t}^l u_t + f_t^l & : l \in I_{1t}^k \\ \xi_{tx}^+(x_t - x_t^+) + \xi_{tu}^+(u_t - u_t^+) + f_t^+ \end{Bmatrix} \tag{2.38}$$

In case of descent step, update uses the following formula.

$$\eta_{1t}^{k+1} = \max \begin{Bmatrix} \sum_{l \in I_{1t}^k \cap \{i:\lambda_{1t}^i > 0\}} \lambda_{1t}^l (g_{1t}^l x_t + h_{1t}^l u_t + f_t^l) \\ \xi_{tx}^+(x_t - x_t^+) + \xi_{tu}^+(u_t - u_t^+) + f_t^+ \end{Bmatrix} \tag{2.39}$$

The piecewise linear approximation of future return is updated using the following regardless of the outcome of iteration k.

$$\eta_{2(t-1)}^k = \eta_{1t}^k(x_t, u_t) + \sum_{l \in I_{2t}^k} \lambda_{2t}^l (g_{2t}^l x_t + h_{2t}^l u_t + v_t^l) \tag{2.40}$$

In (2.38) and (2.39) λ_{1t}^l represent the lagrange coefficient corresponding to l^{th} segment of η_{1t}^k, and λ_{2t}^l of (2.40) represent the lagrangian coefficient of l^{th} segment of η_{2t}^k.

Algorithm 2.3 : Algorithm with subgradient aggregation

Step 0. (Initialization)

Given a starting point x_0 and a nominal control policy $u^0 = (u_0^0, \ldots, u_N^0)$,

choose μ_l (line search parameter): $0 < \mu_l \leq 1$,

compute state trajectory and objective function value for (x_0, u^0).

Step 1. (Backward run)

Repeat for all stages; stage $N, N-1, \ldots, 1$. {

Update piecewise linear approximations using (2.38) (2.39) and (2.40).

Prepare and solve corresponding subproblem (LP or QP).

Compute affine strategy function. }

Step 2. (Forward run)

If $u_t^+ = u_t^k$ for all t then (optimum) stop.

Otherwise compute new trajectory and objective function value.

If $r_k > \rho_k \mu_1$ accept new iterate and goto step 1.

Otherwise accept null step and goto step 1.

Convergence properties of algorithm with subgradient aggregation

The difference between the algorithms with subgradient selection and subgradient aggregation lies in the approximation updating step. With subgradient selection we use a set of subgradient inequalities to approximate a convex function. With subgradient aggregation, we use the Lagrange function of a similar set of subgradient inequalities. This implies both PL functions have the same properties, i.e. (1) at any iteration k for all $t \in 1, \ldots, N$, PL approximations at (x_t^k, u_t^k) have the same value as the corresponding function f_t. (2) both PL functions are

lower support functions of f_t. Similarities between these two methods leads to the same convergence analysis. Therefore we will omit the proof of convergence for the algorithm with subgradient aggregation.

CHAPTER 3

PARALLEL ALGORITHMS

This chapter is devoted to the development of a parallel algorithm for the unconstrained DNDO problem. There are several key issues motivating the development of parallel computational algorithms. First of all real time control of large interconnected systems may necessitate such algorithms. The computational power available today may limit the sequential computation of optimal control of such a system. Further, sequential processing may lead to vast amount of data transmission between subsystems and huge data base requirements for keeping track of the system status. In contrast distributed processing requires little interconnection between subsystems and very little data transmission. Since distributed processing leads to localized controlling of subsystems they have better data privacy and are more immune to intruders. We begin with some preliminaries and then the complete algorithm is presented. Computational results are given in Chapter 4.

Preliminaries

The algorithm developed here is based on decomposition and coordination of interconnected stages. It is an iterative optimization procedure, in which each iteration consists of (1) solving low level subsystems and (2) solving a high level coordinating system. The low level subsystems are solved in parallel with each other on separate processors and in series with the coordinating system.

The parallel algorithm starts with an initial policy. First, the coordinating system determines a set of parameters and initial states for each subsystem and

dispatches them to the appropriate subsystem. Then each subsystem acts according to the initial state and the coordinating parameters transmitted to them by the coordinator. The subsystems return their new controls, their sensitivity with respect to the input state, the objective function value and its sensitivity with respect to the input state, back to the coordinator. Once the coordinating system obtains the feedback of each subsystem, it constructs a new set of coordinating parameters and initial states of the subsystems using their reactions to the previous sets of parameters. At this point the coordinator computes the overall objective function value and updates the system parameters. This procedure is repeated until the realization of overall optimality by the coordinator.

We start the detailed description by deriving a parallel method for unconstrained DNDO in the next section, and thereafter we investigate its convergence. The application of subgradient aggregation and subgradient selection techniques for the parallel algorithm are discussed in last two sections.

Derivation of basic parallel algorithm

This section presents the derivation of the basic parallel algorithm. First the subsystem and the coordinating system are described and then two major steps of the basic algorithm (1) solving subsystems and (2) solving coordinating system are described in detail.

An obvious way to decompose a sequential dynamic system is by assigning each system stage to an individual subsystem. Since each subsystem is solved on separate processors, this decomposition technique requires as many processors as number of stages in the original system. Alternatively, one can decompose the sequential system so as to use fewer processors by assigning more system

stages to an individual subsystem. Without loss of generality, we will use the first decomposition technique in the convergence analysis.

The subsystem

The goal of a particular subsystem is to make local decisions that are confined to the given subsystem and optimize the cost of its own operations plus a penalty arising from any deviation from the final state pre-determined by the coordinator. Thus the subsystem objective function consists of the sum of stagewise objective functions of its stages and a coordinating function. Next we define a subsystem.

Case 1. One stage per subsystem

Suppose that the objective function of stage i of DNDO is $f_i(x_i, u_i)$, the transition function from stage i to stage $i+1$ is $T_i(x_i, u_i)$. Also assume that the initial state and the desired final state set forth by the coordinating system for the subsystem t are x_t^k and z_t^k respectively. The coordinating parameters associated with subsystem t are α_t^k and β_t^k which are set by the coordinator at each iteration. Note that $\alpha_t^k = (\alpha_t^{k1}, \ldots, \alpha_t^{km})$ and $\beta_t^k = (\beta_t^{k1}, \ldots, \beta_t^{km})$, where m is the number of linear segments of the coordinating function. Each element of α_t^k is a scalar and each element of β_t^k is a vector. Then, the subsystem t can be described as follows.

$$\begin{aligned} \min \;\; & f_t(x_t^k, u_t) + \max_l \{\alpha_t^{kl} + \beta_t^{kl}(y_t - z_t^k)\} \\ s.t. \;\; & y_t = T_t(x_t^k, u_t) \end{aligned} \tag{3.1}$$

Case 2. Several stages per subsystem

Let us assume that the stage i of DNDO has the parameters as defined in case 1. Also assume that the stages $S_t + 1, \ldots, S_{t+1}$ belong to the subsystem t. Then, the subsystem t can be described as follows.

$$\begin{aligned} \min \quad & \sum_{j=S_t+1}^{S_{t+1}} f_j(x_j, u_j) + \max_l \{\alpha_t^{kl} + \beta_t^{kl}(y_t - z_t^k)\} \\ s.t. \quad & x_{j+1} = T_j(x_j, u_j) \quad \text{for } j = S_t + 1, \ldots, S_{t+1} \\ & y_t = x_{S_{t+1}+1} \\ & x_{S_t+1} = x_{S_t+1}^k, \end{aligned} \tag{3.2}$$

where $x_{S_t+1}^k$ is the initial state passed to the subsystem from the coordinating system. The coordinating parameters α_t^k and β_t^k respectively represent the predicted value and its variation with respect to z_t^k, of the value function of future stages. These parameters are computed by the coordinating system as described in next section.

After subsystem optimization, each subsystem passes back some information to the coordinating system. For the subsystem t, these are (1) affine relationships of initial state and control of each stage and (2) a set of subgradients of the subsystem objective function value. The state, control vector relations are used by the coordinating system for computation of the next set of initial states, and the subgradient inequalities are used for coordinating parameter computations.

The coordinating system

The purpose of the coordinating system is to generate a feasible set of controls. It also computes a set of initial states and coordinating parameters that coordinates all subsystems.

Solving the subsystem

As mentioned earlier each subsystem has its own control and objective function which are transparent to the others. Given the initial state and the coordinating parameters each subsystem computes its own control in the following manner.

Consider the subsystem t which consists of only one stage. Suppose that the initial state and final state passed to the subsystem from coordinating system are x_t^k and z_t^k, and the coordinating vectors are α_t^k and β_t^k. For these parameters the subsystem optimization problem is given by (3.1).

This optimization problem is approximately solved by using a piecewise linearization of $f_t(x_t, u_t)$ and $T_t(x_t, u_t)$ at (x_t^k, u_t^k). The piecewise linear approximation of $f_t(x_t, u_t)$ denoted $\eta_t^k(x_t, u_t)$ consists of subgradient inequalities generated thus far plus a new subgradient inequality generated at (x_t^k, u_t^k). Then $\eta_t^k(x_t, u_t)$ can be expressed as

$$\eta_t^k(x_t, u_t) = \max_{l \in I_{1t}^k} \{g_{1t}^l x_t + h_{1t}^l u_t + f_t^l\} \tag{3.3}$$

The coordinating function is represented by $\pi_t^k(x_t, u_t)$, a piecewise linear function resulting from a linearization of the transition function. Thus,

$$\pi_t^k(x_t, u_t) = \max_{l \in I_{2t}^k} \{\alpha_t^{kl} + \beta_t^{kl}(T_t(x_t^k, u_t^k) + J_{u_t}(u_t - u_t^k) + J_{x_t}(x_t - x_t^k) - z_t^k)\}.$$

Since $T_t(x_t^k, u_t^k) = z_t^k$,

$$\pi_t^k(x_t, u_t) = \max_{l \in I_{2t}^k} \{\alpha_t^{kl} + \beta_t^{kl}(J_{u_t}(u_t - u_t^k) + J_{x_t}(x_t - x_t^k))\} \tag{3.4}$$

This leads to the following linear programming subproblem.

$$\begin{aligned} \min_{u_t} \ & \eta_t^k + \pi_t^k \\ s.\,to \ & \eta_t^k \geq g_{1t}^l x_t^k + h_{1t}^l u_t + f_t^l \ : \ l \in I_{1t}^k \\ & \pi_t^k \geq \alpha_t^{kl} + \beta_t^{kl} J_{u_t}(u_t - u_t^k) \ : \ l \in I_{2t}^k \end{aligned} \tag{3.5}$$

The set I_{1t}^k contains the indices of the subgradient inequalities of f_t at iteration k and I_{2t}^k consists of the indices of coordinating vector pairs α_t^k and β_t^k.

The solution of the subproblem (3.5) leads to an affine relationship between state vector x_t and control vector u_t. Following the method of analysis introduced in chapter 2, we define matrices H_{1t}, H_{2t}, G_{1t} and G_{2t} as

$$H_{1t} = \begin{pmatrix} h_{1t}^{l_1} \\ \vdots \\ h_{1t}^{l_p} \end{pmatrix}, \quad G_{1t} = \begin{pmatrix} g_{1t}^{l_1} \\ \vdots \\ g_{1t}^{l_p} \end{pmatrix}, \quad \{l_1, \ldots, l_p\} \equiv I_{1t}^k \cap \{i : \lambda_{1t}^i > 0\}$$

$$H_{2t} = \begin{pmatrix} \beta_t^{kl_1} J_{u_t} \\ \vdots \\ \beta_t^{kl_p} J_{u_t} \end{pmatrix}, \quad G_{2t} = \begin{pmatrix} \beta_t^{kl_1} J_{x_t} \\ \vdots \\ \beta_t^{kl_p} J_{x_t} \end{pmatrix}, \quad \{l_1, \ldots, l_p\} \equiv I_{2t}^k \cap \{i : \lambda_{2t}^i > 0\}.$$

Then, we get

$$\begin{aligned} \eta_t &= A_{\eta_t}^k x_t + B_{\eta_t}^k \\ \pi_t &= A_{\pi_t}^k x_t + B_{\pi_t}^k \\ u_t &= A_{u_t}^k x_t + B_{u_t}^k \end{aligned} \tag{3.6}$$

where

$$\begin{pmatrix} A_{\eta_t}^k \\ A_{\pi_t}^k \\ A_{u_t}^k \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & -H_{1t} \\ 0 & 1 & -H_{2t} \end{pmatrix}^{-1} \begin{pmatrix} G_{1t} \\ G_{2t} \end{pmatrix}$$

and

$$\begin{pmatrix} B_{\eta_t}^k \\ B_{\pi_t}^k \\ B_{du_t}^k \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & -H_{1t} \\ 0 & 1 & -H_{2t} \end{pmatrix}^{-1} \begin{pmatrix} (\eta_t^+ - G_{1t} x_t^k - H_{1t} u_t^+) \\ (\pi_t^+ - G_{2t} x_t^k - H_{2t} u_t^+) \end{pmatrix}$$

These relations are used in computing a support of the subsystem objective function for a given initial state as illustrated below.

First the set of subgradients of subsystem objective function is easily extracted from the subsystem LP approximation. Since the objective function of the subsystem is given by

$$v_t^k(x_t, u_t) = \eta_t^k(x_t, u_t) + \pi_t^k(x_t, u_t), \tag{3.7}$$

which is

$$\begin{aligned} v_t^k(x_t, u_t) = & \max_{l_1 \in I_{1t}^k} \{g_{1t}^{l_1} x_t + h_{1t}^{l_1} u_t + f_t^{l_1}\} \\ & + \max_{l_2 \in I_{2t}^k} \{\alpha_t^{kl_2} + \beta_t^{kl_2}(J_{u_t}(u_t - u_t^k) + J_{x_t}(x_t - x_t^k))\} \end{aligned}$$

a subgradient inequality can be written as

$$w_t^k \geq g_{1t}^{l_1} x_t + h_{1t}^{l_1} u_t + f_t^{l_1} + \alpha_t^{kl_2} + \beta_t^{kl_2}(J_{u_t}(u_t - u_t^k) + J_{x_t}(x_t - x_t^k))$$

By substituting $A_{u_t}^k x_t + B_{u_t}^k$ for u_t the subgradient inequality is obtained as a function of initial state of the subsystem.

$$\begin{aligned} w_t^k \geq & (g_{1t}^{l_1} + h_{1t}^{l_1} A_{u_t}^k + \beta_t^{kl_2}(J_{u_t} A_{u_t}^k + J_{x_t}))x_t \\ & + f_t^{l_1} + h_{1t}^{l_1} B_{u_t}^k + \alpha_t^{kl_2} + \beta_t^{kl_2}(J_{u_t}(B_{u_t}^k - u_t^k) - J_{x_t} x_t^k) \end{aligned} \tag{3.8}$$

Denoting the components of w_t^k as $\omega_{x_t}^k$ and ω_t^k we get

$$w_t^k \geq \omega_{x_t}^k x_t + \omega_t^k \tag{3.9}$$

The matrices A^k, B^k and ω^k are transmitted to the coordinating system.

In case of subsystems containing several stages the procedure varies slightly. For this case each subsystem is solved as a DNDO of reduced dimension. Algorithms described in chapter two can be used for solving these subsystems. Note that these algorithms produce the necessary affine relationships of the low level system stages. Since an optimum of these subsystems is not necessary for the overall convergence, a few iterations of innerloop may be recommended.

Solving the coordinating system

The coordinating system is essentially similar to a forward pass of the descent method developed in chapter 2. The coordinating system computes overall system states and controls by using the stage transition functions $T_t(x_t, u_t)$ and the affine relations between state and control vectors developed at the subsystems. We recursively compute

$$u_t^+ = A_{u_t}^k x_t^+ + B_{u_t}^k \quad \text{for } t = 1, \ldots, N$$

$$x_{t+1}^+ = T_t(x_t^+, u_t^+) \quad \text{for } t = 1, \ldots, N-1$$

The new state - control vector pair, which is always feasible to the overall system, is then tested for an improvement over the previous iteration. As in descent algorithms, if the new solution is better, it is accepted as the new iterate with a "descent step" , i.e., $z_t^{k+1} = x_t^+$, otherwise rejected with a "null step", $z_t^{k+1} = z_t^k$.

The coordinating parameters α_t^{k+1} and β_t^{k+1} of subsystems are computed using $\omega_{x_t}^k$ and ω_t^k (as defined in (3.9)) and z_t^{k+1} as

$$\begin{aligned} \alpha_t^{k+1} &= \omega_{x_t}^k z_t^{k+1} + \omega_t^k \\ \beta_t^{k+1} &= \omega_{x_t}^k \end{aligned} \tag{3.10}$$

The Basic algorithm

In this section we will describe the basic parallel programming algorithm.

Algorithm 3.1 : Basic parallel algorithm

Step 0. (Initialization)

Given a starting point x_0 and a nominal control policy $u^0 = (u_0^0, \ldots, u_N^0)$,

Compute state trajectory and objective function value for (x_0, u^0).

Prepare subsystem optimization problems.

Step 1. (Subsystem optimization)

Coprocess for all stages; stage $1, \ldots, N$. {

Prepare and solve corresponding sub problem LP.

Compute parameters to return to coordinating system. }

Step 2. (Coordination)

Compute new trajectory (x^+, u^+) and objective function f^+ value.

If $(x^+, u^+) = (x^k, u^k)$ (optimum) stop.

Otherwise check sufficient reduction condition.

If satisfied take "descent step" otherwise "null step".

Goto step 1.

Convergence Properties of basic algorithm

Convergence properties of the basic parallel algorithm are analyzed in this section. We will analyze three classes of sequences that may be generated by the algorithm. These are,

1. A finite sequence terminating at (x^K, u^K).
2. An infinite sequence (x^k, u^k), $k \in 1, \ldots, \infty$ with descent steps.
3. An infinite sequence with null steps after (x^K, u^K).

Note that any sequence of points generated by the algorithm belongs to one of the above three classes. Therefore if these three classes converge to a stationary point then so does the algorithm. To show that all classes converge to a stationary point let us first start with case 1.

Case 1. Finite terminating sequence

The optimality test used in our basic parallel algorithm implies that each subsystem must be optimal at a control that satisfies both end states, initial and final, of its own for overall system optimality. Thus it is necessary that for the parameters and initial states set by the coordinating system, each subsystem reach the optimality and produce the final state set by the coordinating system.

Now at any iteration, from the parallel algorithm, the set of initial and final states generated by coordinating system is feasible for each stage control vector. This implies that if the control vector is unchanged at any iteration, both end states are satisfied regardless of the other parameters.

The stopping rule of basic algorithm 3.1 suggests that it stops only when all subproblems are optimal at the current solution. Thus the algorithm stops at iteration K implies that the solution $u_t^+ = u_t^K$ for all t.

Let us define $\hat{X}_{t+1}(x_t, u_t)$ and $\hat{U}_{t+1}(x_t, u_t)$ as

$$\begin{aligned} \hat{X}_{t+1}(x_t, u_t) &= x_{t+1}^k + J_{u_t}(u_t - u_t^k) + J_{x_t}(x_t - x_t^k) \\ \hat{U}_{t+1}(x_t, u_t) &= A_{u_{t+1}}^k \hat{X}_{t+1}(x_t, u_t) + B_{u_{t+1}}^k \end{aligned}$$

Where J_{u_t} and J_{x_t} are the Jacobian matrices of transition function $T_t(x_t, u_t)$ at (x_t^k, u_t^k) with respect to u_t and x_t. $A_{u_{t+1}}^k$ and $B_{u_{t+1}}^k$ are as defined in (3.6).

Lemma 3.1 Suppose that algorithm 3.1 generates a finite sequence of points and stops at iteration K generating (x^K, u^K). Also, suppose that $\eta_{N-1}^K(x_{N-1}^K, u_{N-1})$

and $\pi^K_{N-1}(x^K_{N-1}, u_{N-1})$ are defined as in (3.3) and (3.4) respectively with $t = N-1$ and $x_{N-1} = x^K_{N-1}$. Then

$$\pi^K_{N-1}(x^K_{N-1}, u^K_{N-1}) \;=\; f_N(x^K_N, u^K_N)$$

$$\pi^K_{N-1}(x^K_{N-1}, u_{N-1}) \;\leq\; f_N(\hat{X}_N(x^K_{N-1}, u_{N-1}), \hat{U}_N(x^K_{N-1}, u_{N-1}))$$

and

$$\pi^K_t(x^K_t, u_t) \;=\; (\eta^K_{t+1} + \pi^K_{t+1})(\hat{X}_{t+1}(x^K_t, u_t), \hat{U}_{t+1}(x^K_t, u_t)).$$

Proof: Since $f_N(x_N, u_N)$ is a convex function and $\eta^k_N(x_N, u_N)$ is an outer envelop generated by a set of subgradient of $f_N(x_N, u_N)$,

$$\eta^k_N(x_N, u_N) \;\leq\; f_N(x_N, u_N) \tag{3.11}$$

and since η^k_N majorizes a subgradient generated at (x^k_N, u^k_N)

$$\eta^k_N(x^k_N, u^k_N) \;=\; f_N(x^k_N, u^k_N). \tag{3.12}$$

Since I^k_{2N} is an empty set, from (3.7)

$$v^k_N(x^k_N, u^k_N) \;=\; \eta^k_N(x^k_N, u^k_N) \tag{3.13}$$

At iteration K the algorithm stops implies $v^K_N = v^{K-1}_N$. Therefore from (3.12) and (3.13)

$$v^{K-1}_N(x^K_N, u^K_N) \;=\; f_N(x^K_N, u^K_N)$$

but since $v^{j-1}_N(x^j_N, u^j_N) = \pi^j_{N-1}(x^j_{N-1}, u^j_{N-1})$ for all $j > 1$

$$\pi^K_{N-1}(x^K_{N-1}, u^K_{N-1}) \;=\; f_N(x^K_N, u^K_N). \tag{3.14}$$

To prove the inequality let us follow the same technique. From (3.7) and (3.11)

$$v_N^K(x_N, u_N) \leq f_N(x_N, u_N)$$

Then again using the same argument, since for $j > 1$

$$v_N^{j-1}(\hat{X}_N(x_{N-1}^j, u_{N-1}), \hat{U}_N(x_{N-1}^j, u_{N-1})) = \pi_{N-1}^j(x_{N-1}^j, u_{N-1}),$$

$$\pi_{N-1}^K(x_{N-1}^K, u_{N-1}) \leq f_N(\hat{X}_N(x_{N-1}^K, u_{N-1}), \hat{U}_N(x_{N-1}^K, u_{N-1})). \qquad (3.15)$$

To prove the equality between stage t value function and stage $t+1$ total objective function value we will use a similar argument. From (3.7) at iteration K,

$$v_{t+1}^K(x_{t+1}, u_{t+1}) = \eta_{t+1}^K(x_{t+1}, u_{t+1}) + \pi_{t+1}^K(x_{t+1}, u_{t+1})$$

Then since $v_t^K(x_t, u_t) = v_t^{K-1}(x_t, u_t)$

$$v_{t+1}^{K-1}(\hat{X}_{t+1}(x_t^K, u_t), \hat{U}_{t+1}(x_t^K, u_t)) = \pi_t^K(x_t^K, u_t)$$

Therefore

$$\pi_t^K(x_t^K, u_t) = (\eta_{t+1}^K + \pi_{t+1}^K)(\hat{X}_{t+1}(x_t^K, u_t), \hat{U}_{t+1}(x_t^K, u_t)). \blacksquare$$

Lemma 3.2 Suppose that the basic parallel algorithm generates a finite sequence of points and stops at iteration K generating (x^K, u^K). Then (x^K, u^K) is stationary to the problem 1.1.

Proof: From lemma 3.1, the function π satisfies the necessary conditions of η_2 of Chapter 2. Also, the function η satisfy the necessary conditions of η_1. Then proof follows lemma 2.1, 2.2 and 2.3. ■

Case 2. Infinite sequence with descent steps

In here we will consider the convergence of infinite sequence of points with descent steps that can be generated by the algorithm. We will omit the proof whenever the proof is similar to that present for the descent method in chapter 2.

Lemma 3.4 Suppose that the algorithm 3.1 generates an infinite sequence of points (x^k, u^k). Then $(\eta_1^k + \pi_1^k)(x_1^k, u_1^k)$ $k = 1, \cdots, \infty$ has exactly one accumulation point.

Proof: Follows from the same arguments as in lemma 2.4. ■

Lemma 3.5 Suppose for some index set M, $\lim_{\substack{k_l \in \kappa \\ l \in M}} (x_1^{k_l}, u_1^{k_l}) = (\bar{x}_1, \bar{u}_1)$, $\lim_{\substack{k_l \in \kappa \\ l \in M}} (x_1^{k_l+1}, u_1^{k_l+1}) = (\tilde{x}_1, \tilde{u}_1) \neq (\bar{x}_1, \bar{u}_1)$, and $0 \notin \partial \left(\sum_{j=1}^{N} f_j \right) (\bar{x}_1, \bar{u}_1)$ then $\left(\sum_{j=1}^{N} f_j \right) (\bar{x}_1, \bar{u}_1) > \left(\sum_{j=1}^{N} f_j \right) (\tilde{x}_1, \tilde{u}_1)$.

Proof: Follows from the same arguments as in lemma 2.5. ■

Lemma 3.6 Every accumulation point generated by the algorithm 3.1 is stationary.

Proof: From Lemma 3.4 and 3.5 proof follows lemma 2.6. ■

Case 3. Infinite sequence with null steps

Suppose that the algorithm 3.1 generates an infinite sequence of points with null steps after iteration K. By construction of the algorithm, we have

$$(\eta_1^k + \pi_1^k)(x_1^k, u_1^k) = \left(\sum_{j=1}^{N} f_j \right) (x_1^k, u_1^k).$$

Also note that the relationship of predicted objective function values derived in Corollary 2.3 holds for the algorithm 3.1. Therefore the following lemma holds.

Lemma 3.7 : Suppose that the algorithm 3.1 generates an infinite sequence of null points after generating (x^K, u^K). Then (x^K, u^K) is stationary to the problem 1.1.

Proof : Omitted. ■

Theorem 3.1 : Under the assumptions (1) Transition functions are twice continuously differentiable, (2) Stagewise objective functions are convex and (3) A finite solution exists, every convergent subsequence generated by Algorithm 3.1 converges to a stationary point of problem 1.1.

Proof : Follows lemma 3.2, 3.6 and 3.7. ■

In the next section we will discuss extension of algorithm 3.1 which will reduce the size of the subproblems at each iteration.

Subgradient selection and aggregation

Similarities between the two basic algorithms, sequential descent algorithm and parallel algorithm, leads us to use the same enhancement techniques for both algorithms. The basic idea of subgradient selection technique is to update PL approximations generated at each iteration more efficiently, using a subset of subgradient inequality constraints which are active or near active at a particular iteration. In our algorithm we used all available subgradients if the iteration produced a null step, otherwise only the active set is selected as in followings.

$$\eta_t^{k+1} = \max \begin{Bmatrix} g_{1t}^l.x_t + h_{1t}^l.u_t + f_t^l \quad : l \in I_{1t}^k \cap \{i \ : \lambda_{1t}^i > 0\} \\ \xi_{tx}(x_t - x_t^{k+1}) + \xi_{tu}(u_t - u_t^{k+1}) + f_t^+ \end{Bmatrix} \tag{3.16}$$

$$\pi_{t-1}^k = \eta_t^k + \sum_{l \in I_{2t}^k} \lambda_{2t}^l(\alpha_t^{kl} + \beta_t^{kl}(J_{ut}(u_t - u_t^k) + J_{xt}(x_t - x_t^k)) \tag{3.17}$$

For the subgradient aggregation technique, whenever there is an improvement over the past the subgradient inequalities which are inactive are dropped and the active ones are aggregated.

In the parallel processing algorithm these updates are done at the subsystem. The PL approximations of stagewise objective function and the subsystem objective function are updated using following equations.

$$\eta_t^{k+1} = \max \left\{ \begin{matrix} \sum_{l \in I_{1t}^k \cap \{i \ : \lambda_{1t}^i > 0\}} \lambda_{1t}^l.(g_{1t}^l.x_t + h_{1t}^l.u_t + f_t^l) \\ \xi_{tx}(x_t - x_t^{k+1}) + \xi_{tu}(u_t - u_t^{k+1}) + f_t^+ \end{matrix} \right\} \tag{3.18}$$

$$\pi_{t-1}^k = \eta_t^k + \sum_{l \in I_{2t}^k} \lambda_{2t}^l(\alpha_t^{kl} + \beta_t^{kl}(J_{ut}(u_t - u_t^k) + J_{xt}(x_t - x_t^k)) \tag{3.19}$$

Here λ_{1t}^l represent the lagrange coefficient corresponding to l^{th} segment of η_t^k, and λ_{2t}^l represent the lagrangian coefficient of l^{th} segment of π_t^k.

The convergence proof follow the same path as the proof of sequential descent algorithm with minor modifications.

CHAPTER 4

APPLICATIONS AND COMPUTATIONAL RESULTS

This chapter describes the application areas of nondifferentiable dynamic optimization techniques and reports our computational experience. First we verify that the algorithms discussed in the previous chapters do provide an optimal solution to a test problem. This verification is carried out by comparing our solutions with those obtained from a standard nonlinear programming package (NPSOL). We also apply our methods to constrained nonlinear programs and minimax optimization problems. Finally, some applications in optimal control of electric power generating systems and optimal control of production processes are described.

A test problem.

The test problem developed for testing the proposed algorithms consists of a linear transition function and a convex, nondifferentiable objective function.

$$\min \sum_{t=1}^{N} \max \left\{ x_t^2 + \sum_{i=1}^{4} (x_t - u_{ti})^2, x_t + u_{t1} - 2u_{t2} + 3u_{t3} - 4u_{t4} + 2 \right\}$$

$$s.t.\ x_{t+1} = 0.7x_t + 0.2u_{t1} + 0.3u_{t2} - 0.2u_{t3} + u_{t4} + 1$$

$$x_1 = 0$$

The results for 3, 7 and 10 stage problems are given in table 4.1a, 4.1b and 4.1c respectively.

Algorithm	No.of iterations	Solution
NPSOL	16	3.76395
Sequential without cut dropping	72	3.76395
Parallel without cut dropping	83	3.76395
Sequential with selection	155	3.76395
Parallel with selection	167	3.76395

Table 4.1a - Results of test problem - 3 stages

Algorithm	No.of iterations	Solution
NPSOL	25	10.17052
Sequential without cut dropping	77	10.17052
Parallel without cut dropping	89	10.17052
Sequential with selection	188	10.17052
Parallel with selection	203	10.17052

Table 4.1b - Results of test problem - 7 stages

Algorithm	No.of iterations	Solution
NPSOL	26	14.09264
Sequential without cut dropping	77	14.09264
Parallel without cut dropping	89	14.09264
Sequential with selection	207	14.09264
Parallel with selection	235	14.09264

Table 4.1c - Results of test problem - 10 stages

The results show that both sequential and parallel algorithms and their subgradient selection variants converge to the optimum solution. Also the test problems was solved by NPSOL with fewer number of iterations than any of the algorithms proposed in this dissertation. This indicates that when a NDO problem can be written as a smooth constrained optimization problem, standard NLP algorithms may be more appropriate.

Applications in constrained nonlinear programming

It is often possible to reformulate a constrained nonlinear optimization problem as an unconstrained DNDO problem. Exact penalty function methods can be used in this reformulation. For this reformulation some form of separability of problem variables is required. Reformulation of two nonlinear programming problems are given below.

Example 4.1 Rosen-Suzuki problem

The original problem, which has three constraints and four variables, is given below.

$$\begin{aligned} \min\ & y_1^2 + y_2^2 + 2y_3^2 + y_4^2 - 5y_1 - 2y_2 - 21y_3 + 7y_4 \\ s.t.\ & y_1^2 + y_2^2 + y_3^2 + y_4^2 + y_1 - y_2 + y_3 - y_4 \ \leq\ 8 \\ & y_1^2 + 2y_2^2 + y_3^2 + 2y_4^2 - y_1 - y_4 \ \leq\ 10 \\ & 2y_1^2 + y_2^2 + y_3^2 + 2y_1 - y_2 - y_4 \ \leq\ 5 \end{aligned}$$

Let us formulate this problem as a multistage optimization problem by associating each variable with a unique stage. Thus stage one consists of variable y_1 as its control variable u_{11} and stage two consists of variable y_2 as its control variable u_{21} etc. The stages are interconnected using contributions of all preceeding stages to the original objective function value and the constraints, making the state variable a 4-tuple. i.e. state variable of stage two consists of $(y_1^2 - 5y_1, y_1^2 + y_1, y_1^2 - y_1, 2y_1^2 + 2y_1)$. Then equivalent DNDO optimization problem is as follows.

$$\begin{aligned} \min\ & \sum_{t=1}^{4} f_t(x_t, u_t) \\ s.t.\ & x_{t+1} = T_t(x_t, u_t) \end{aligned}$$

where

$$f_t(x_t, u_t) = 0 \quad \text{for } t = 1, 2, 3$$

$$f_4(x_4, u_4) = x_{41} + u_{41}^2 + 7u_{41} + \sigma \max\{0, x_{42} + u_{41}^2 - u_{41} - 8\}$$
$$+ \sigma \max\{0, x_{43} + 2u_{41}^2 - u_{41} - 10\}$$
$$+ \sigma \max\{0, x_{44} - u_{41} - 5\}$$

and

$$T_1 = \begin{pmatrix} u_{11}^2 - 5u_{11} \\ u_{11}^2 + u_{11} \\ u_{11}^2 - u_{11} \\ 2u_{11}^2 + 2u_{11} \end{pmatrix} \quad T_2 = \begin{pmatrix} x_{21} + u_{21}^2 - 5u_{21} \\ x_{22} + u_{21}^2 - u_{21} \\ x_{23} + 2u_{21}^2 \\ x_{24} + u_{21}^2 - u_{21} \end{pmatrix} \quad T_3 = \begin{pmatrix} x_{31} + 2u_{31}^2 - 21u_{31} \\ x_{32} + u_{31}^2 + u_{31} \\ x_{33} + u_{31}^2 \\ x_{34} + u_{31}^2 \end{pmatrix}.$$

In the above reformulation, the penalty parameter is σ. Also it consists of four stages in which the first three have no contribution to the objective function.

This problem is solved using $(y_1 = 0, y_2 = 0, y_3 = 0, y_4 = 0)$ as the starting point at which objective function value is zero. Initially the penalty parameter σ is set to 1. Since the overall problem is strictly convex, both algorithms converge to the only stationary point which is the global optimum. The results are given in table 4.2. Since the resulting solution is feasible, there is no need to increase the penalty parameter, and the algorithm terminates.

Iteration	Objective function value			
	Without cut dropping		With cut dropping	
	Sequential	Parallel	Sequential	Parallel
5	-36.2579	-29.2532	-36.5309	-28.8037
10	-42.6905	-39.9631	-42.8984	-39.6085
15	-43.9946	-43.7950	-43.8979	-43.4816
20	-43.9983	-43.8183	-43.9887	-43.5984
23	-44.0000	-43.9908	-43.9980	-43.9499
28		-44.0000	-43.9997	-43.9878
32			-44.0000	-43.9912
39				-44.0000

Table 4.2 - Results - Rosen-Suzuki problem

Using the same technique as in the above example, one can transform Ohno's problem (Ohno [1978]) to a DNDO. The original problem, a constrained nonlinear programming problem, is the following.

$$\min\ y_1^2 + y_2^2 + y_3^2$$

$$s.t.\ y_1^2 + y_1 - 4y_2 - y_3 + 3 \leq 0$$

The transformed problem has two stages, and satisfy our test problem requirements. Define the following variables

$$u_{11} = y_1, \quad u_{21} = y_2, \quad u_{22} = y_3.$$

Then two stage formulation is,

$$\min\ f_1 + f_2$$

$$s.t.\ x_2 = u_{11}^2 + u_{11}$$

$$f_1 = u_{11}^2$$

$$f_2 = u_{21}^2 + u_{22}^2 + \sigma \max\{0, x_2 - 4u_{21} - u_{22} + 3\}$$

The starting point is $(y_1 = 1, y_2 = -2, y_3 = -3)$, which is infeasible to the original problem. Once again we put $\sigma = 1$, and the sequence of objective values obtained by the various methods are reported in table 4.3. The solution obtained by the methods was feasible and hence optimum.

Iteration	Objective function value			
	Without cut dropping		With cut dropping	
	Sequential	Parallel	Sequential	Parallel
5	0.610094	0.938565	0.798035	1.281768
10	0.507916	0.510740	0.507410	0.543276
15	0.507161	0.507151	0.507190	0.507391
20	0.507133	0.507133	0.507144	0.507151
27			0.507133	0.507140
30				0.507133

Table 4.3 - Results - Constrained quadratic problem

Applications in Minimax Optimization

Several minimax optimization problems can be reformulated as multistage nondifferentiable optimization problems. Several minimax optimization problems were reformulated and solved with the algorithms developed chapter 2 and 3. The following example is from Shor [1979].

Example 4.2 Shor's problem

The original problem is a minimization of the maximum of ten quadratic functions of five variables. The multistage formulation has five stages with one control variable and ten state variables for each stage. Form of the original problem is,

$$\min_x \max_i \left\{ b_i \sum_{j=1}^{5} (x_j - a_{ij})^2 \ : \ i = 1, \ldots, 10 \right\}$$

where transpose of A, and b are given as,

$$A^T = \begin{pmatrix} 0 & 2 & 1 & 1 & 3 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 2 & 4 & 2 & 2 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 2 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 3 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

and

$$b = (1 \ \ 5 \ \ 10 \ \ 2 \ \ 4 \ \ 3 \ \ 1.7 \ \ 2.5 \ \ 6 \ \ 3.5).$$

The multistage formulation is,

$$\min \sum_{t=1}^{5} f_t(x)$$

$$s.t. \ x_{(t+1)j} = x_{tj} + (u_{t1} - a_{jt})^2 \ : \ t = 1, \ldots, 5 \ ; \ \ j = 1, \ldots, 10$$

$$f_t(x) = 0 \ : \ t = 1, \ldots, 4$$

$$f_5(x) = \max_j \left\{ b_j (x_{5j} + (u_{51} - a_{j5})^2) \right\}.$$

Note that above formulation is essentially similar to the formulation encountered in Rosen-Suzuki problem. In Rosen-Suzuki problem we explicitly created the minmax formulation while it is given for the Shor's problem. Results obtained by using $(x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0, x_5 = 1)$ starting point at which objective function value is 80.0, is given in table 4.4.

Iteration	Objective function value			
	Without cut dropping		With cut dropping	
	Sequential	Parallel	Sequential	Parallel
15	22.62020	22.97790	22.65150	24.67420
20	22.60180	22.61860	22.60400	23.21590
25	22.60050	22.60840	22.60206	22.68832
30	22.60017	22.60230	22.60125	22.60847
36	22.60016	22.60060	22.60032	22.60421
40		22.60030	22.60022	22.60140
45		22.60023	22.60016	22.60072
50		22.60019		22.60032
55		22.60017		22.60026
65		22.60016		22.60019
77				22.60016

Table 4.4 - Results - Shor's problem

Optimal control of electrical power generating systems

There are at least two kinds of dynamic optimization problems that arise in such systems. The first one is a system wide problem, where as several generators are to be controlled in order to meet continuously varing electricity demand. Note that each generator has restrictions such as minimum up time, maximum uptime, minimum down time and maximum load. This poses a dynamic optimization problem that has a constrained nonlinear objective function. Another type of optimization problem arises in the context of controlling a particular generator. The question here is how a generator ought to be used so as to produce electrical

energy that has the highest present worth over its life span. Again this poses a nonlinear dynamic optimization problem in which objective function may not necessarily be differentiable.

Example 4.3 Optimal control of a Nuclear Power Plant

This example, taken from Overton and Wilson [1980], models the energy transfer of a nuclear power plant from the nuclear reactor core to the turbine. Energy is transfered using a coolant gas that has three primary controllable properties namely, gas temperature (T), gas pressure (P) and the water content (W). Higher gas temperatures result in higher efficiency in energy transfer and higher gas pressures result in a higher energy transfer rate, both leading to increased power transfer from the core. Although the water content of the coolant does not directly affect the energy transfer from core, it plays an important role in controlling the system state.

The system state of stage t is a 2-tuple $(s_{1(t)}, s_{2(t)})$ which must be monitored during its operation. These two state variables always increase with system evolution, but their rate of increase can be controlled by selecting proper coolant temperature, pressure and the water content.

Lifetime of the fusion system is divided equally into 2 year periods. Each two year period is considered as a single stage of the dynamic system. The objective function of a given stage t is the present worth of the profit generated in that period which is given by,

$$f_t = a_t \left(\frac{P_t}{215}\right)^{0.6667} (955 + \min\{6.5(T_t - 643), 7.5(T_t - 643)\})$$

where a_t is the present worth factor of corresponding stage profit.

The system transition function, that is the relationship of the state of a given stage to the state and controls of the preceding stage is given by,

$$s_{1(t+1)} = s_{1(t)} + 2\left(\frac{W_t}{3}\right)^{0.25}\left(\frac{P_t}{215}\right)^{0.5}\exp\left(-15500\left(\frac{1}{T_t}-\frac{1}{633}\right)\right)$$

$$s_{2(t+1)} = s_{2(t)} + 0.44\left(\frac{P_t}{215}\right)^{1.6667}(1.451 - 0.53\log_{10} W_t)$$

In addition to the above properties the system state variables and controls have the following bounding constraints.

$$0 \leq s_{1(t)} \leq 50$$

$$0 \leq s_{2(t)} \leq 15.66$$

$$215 \leq P_t \leq 400$$

$$603 \leq T_t \leq 663$$

$$3 \leq W_t \leq 30$$

For the formulation of nuclear power generating system as a multistage constrained optimization problem, define the following variables.

$$u_{1(t)} = \left(\frac{P_t}{215}\right)^{0.667}$$

$$u_{2(t)} = T_t$$

$$u_{3(t)} = \log_{10} W_t$$

Then the multistage constrained optimization problem formulation is as follows.

$$\begin{aligned}
\min \; & -\sum_{t=1}^{N} a_t u_{1(t)} \left(955 + \min\left\{6.5(u_{2(t)} - 643), 7.5(u_{2(t)} - 643)\right\}\right) \\
s.t. \; s_{1(t+1)} &= s_{1(t)} + 1.5197 u_{1(t)}^{0.75} \exp\left(\frac{u_{3(t)}}{4} - 15500\left(\frac{1}{u_{2(t)}} - \frac{1}{633}\right)\right) \\
s_{2(t+1)} &= s_{2(t)} + u_{1(t)}^{2.5}\left(0.6384 - 0.2332 u_{3(t)}\right) \\
& 1.0 \leq u_{1(t)} \leq 1.5127 \\
& 603 \leq u_{2(t)} \leq 663 \\
& 0.4771 \leq u_{3(t)} \leq 1.4771 \\
& 0.0 \leq s_{1(t)} \leq 50 \\
& 0.0 \leq s_{2(t)} \leq 15.66
\end{aligned}$$

Note that the change of variables to $u_{1(t)}, u_{2(t)}, u_{3(t)}$ leads to a convex stage-wise objective function. Upon including the bounding constraints into the objective function, via an exact penalty function, the DNDO formulation reduces to

$$\begin{aligned}
\min \; & -\sum_{t=1}^{N} a_t u_{1(t)} \left(955 + \min\left\{6.5(u_{2(t)} - 643), 7.5(u_{2(t)} - 643)\right\}\right) \\
& + \sigma \max\left\{0, 1.0 - u_{1(t)}, u_{1(t)} - 1.5127\right\} \\
& + \sigma \max\left\{0, 603 - u_{2(t)}, u_{2(t)} - 663\right\} \\
& + \sigma \max\left\{0, 0.4771 - u_{3(t)}, u_{3(t)} - 1.4771\right\} \\
& + \sigma \max\left\{0, s_{1(t)} - 50.00, s_{2(t)} - 15.66\right\} \\
s.t. \; s_{1(t+1)} &= s_{1(t)} + 1.5197 u_{1(t)}^{0.75} \exp\left(\frac{u_{3(t)}}{4} - 15500\left(\frac{1}{u_{2(t)}} - \frac{1}{633}\right)\right) \\
s_{2(t+1)} &= s_{2(t)} + u_{1(t)}^{2.5}\left(0.6384 - 0.2332 u_{3(t)}\right).
\end{aligned}$$

Here σ is the penalty parameter associated with the constraints. The present worth factors of each stage a_i are given in table 4.5.

Stage (t)	Present worth factor (a_t)	Stage (t)	Present worth factor(a_t)
1	4.000	8	1.200
2	3.900	9	0.800
3	3.600	10	0.600
4	2.900	11	0.500
5	2.200	12	0.400
6	1.800	13	0.300
7	1.400	14	0.250

Table 4.5 - Parameters - Nuclear plant

This problem is solved using ($P_t = 215, T_t = 603, W_t = 3.162$) for each stage t, which is feasible, as the starting point. The initial objective function value is 1.6402×10^4. Results of the descent and parallel algorithms are as follows.

Iteration	Objective function value ($\times 10^4$)	
	Sequentail	Parallel
25	-2.4904	-2.4854
30	-2.6897	-2.6998
35	-2.8968	-2.9232
40	-3.1115	-3.1196
45	-3.1518	-3.1553
50	-3.2575	

Table 4.6 - Results - Nuclear plant

Note that the transition functions of the above problem are nonconvex. Therefore the overall objective function is nonconvex leading to possibly several stationary points. Since our algorithm only seeks stationary points this nonconvexity explains the differences in solutions obtained by the two methods.

Optimal control of Production Processes

There are several industrial processes that can be modeled as multi-stage systems. These production processes are often subject to constraints that play a key role at each stage. Unless carefully monitored, these constraints may be violated at some time during the process, and may lead to reduced productivity, shorter machine lifetimes and in some cases, to accidents. Most of the time, the goal of these production processes is to increase the profit generated which depend on several parameters such as elapsed time of overall process, total raw material used and total labor used. Often these goals (objective functions) are nondifferentiable thus making them more difficult to analyze. These processes can be modeled and controlled more accurately by using the algorithms discussed in this thesis. As an example, consider the following rolling mill scheduling problem.

Example 4.4 Optimal control of a reversible rolling mill

This example is discussed in depth by Ray and Szekely [1973]. The rolling mill considered in the above paper is a reversing hot strip mill which processes hot metal strips to a desired final thickness. The mill consists of two driven rolls with the top roll adjustable in the vertical plane to allow variation of the roll gap. The rolls are reversible to allow feeding unfinished metal strips in any of the two directions. The mill is used to process metal ingots that have a given fixed initial thickness, width and length. During the process the initial thickness is reduced to a desired thickness, while the width of the workpiece remains the same. The workpiece is allowed to passthrough the mill odd number of times since it should leave in the same direction as it entered the mill.

The objective of the process is to achieve the desired final thickness in smallest possible time, without overloading the mill. The mill has several capacity

constraints on its operating speed, roll force, draft and the bite angle. The reader is refered to Ray and Szekely[1973] for more detailed analysis of this process.

Let N(odd) denote the number of passes needed to process an ingot with initial thickness x_1 to a final thickness x_N. The primary control variable of the process is the draft, the reduction of thickness during any given pass, which governs the bite angle, the maximum velocity and the total process time. For a mill with roll diameter of 36 inch, the bite angle constraint is given by

$$v_t \leq 400.25 - 2500\,(d_t - 2.0)^2 \qquad \text{for } 2.0 \leq d_t \leq 2.4$$

where v_t is the roll speed in fpm and d_t is the draft in inches.

The relationship between draft and maximum mill velocity is given by the following empirical formula.

$$v_t \leq \frac{200}{d_t} + 300 \qquad \text{for } 0 \leq d_t \leq 2.0$$

The maximum force required for the process, denoted F_t, depends on the draft d_t and the reduction ratio $r_t = \dfrac{x_t}{x_t - d_t}$, where x_t is the entry thickness of the ingot. The following empricial formula gives the maximum force (in 10^6 lbf) required.

$$F_t = 2d_t r_t - 0.7r_t - 2d_t + 3.15$$

The maximum allowable force for safety reasons for the particular example is 3.6×10^6 lbf. Then the force constraint is

$$3.6 \geq 2d_t r_t - 0.7r_t - 2d_t + 3.15,$$

which simplifies to

$$0 \geq d_t^2 + 0.225d_t - 0.575x_t.$$

In addition to these constraints, there are some constraints related to the power requirement. For simplicity we will assume the total power available is unconstrained.

The objective function of the overall process is to minimize the total elapsed time which is the sum of

1. loading time $T_f = \sum_{t=1}^{N} T_{ft}$ and,
2. processing time $T_c = \sum_{t=1}^{N} T_{ct}$,

where T_{ft} is a constant and the T_{ct} can be given by,

$$T_{ct} = 2\left(\frac{\sqrt{u^2 + as_t} - u}{a}\right) + \frac{\left(\max\left\{0, \sqrt{u^2 + as_t} - v_t\right\}\right)^2}{av_t}.$$

Here s_t is the entry length of the workpiece and v_t is the maximum workpiece travel speed in the mill. u denotes the workpiece speed at the entry and the exit, and a denotes the constant acceleration (deceleration). Then the process optimization problem can be formulated as,

$$\min \sum_{t=1}^{N} \left\{ T_{ft} + 2\left(\frac{\sqrt{u^2 + as_t} - u}{a}\right) + \frac{\left(\max\left\{0, \sqrt{u^2 + as_t} - v_t\right\}\right)^2}{av_t} \right\}$$

s.t.

$$v_t \leq \begin{cases} 400.25 - 2500\,(d_t - 2.0)^2 & \text{if } 2.0 \leq d_t < 2.4 \\ \frac{200}{d_t} + 300 & \text{if } 0.0 \leq d_t < 2.0 \end{cases}$$

$$0 \geq d_t^2 + 0.225 d_t - 0.575 x_t$$

$$s_t = \frac{x_1 s_1}{x_t}$$

$$x_{t+1} = x_t - d_t$$

$$x_N,\ x_1 \text{ given}$$

This problem is reformulated as an unconstrained DNDO problem below. The convexity of the stagewise objective function and first order smoothness of

the transition function is maintained by making v_t a control variable and removing d_t from the control variable list. The maximum force constraint is added to the objective function as a penalty function. The new formulation is as follows.

$$\min \sum_{t=1}^{N} \left\{ T_{ft} + 2\left(\frac{\sqrt{u^2+as_t}-u}{a}\right) + \frac{\left(\max\left\{0, \sqrt{u^2+as_t}-v_t\right\}\right)^2}{av_t} \right\}$$
$$+ \sigma \sum_{t=1}^{N} \max\{0, d_t^2 + 0.225d_t - 0.575x_t\}$$

s.t.

$$d_t = \begin{cases} 1.99 + 0.02\sqrt{400.25 - v_t} & \text{if } v_t \leq 400 \\ \frac{200}{v_t - 300} & \text{otherwise} \end{cases}$$

$$s_t = \frac{x_1 s_1}{x_t}$$

$$x_{t+1} = x_t - d_t$$

Note that many of the emprical formulas given here were originally presented in Ray and Szekely [1973] as a set of graphical representations. Due to this, some inaccuracy of the solution may be expected.

This problem is solved using ($v_t = 400$) for each t as the starting point. Values of the other parameters are given in table 4.7 and the results of two methods are given in table 4.8.

Parameter	Value
T_{ft}	0.033 minutes
s_1	4 ft
x_1	24 inch
x_2	1.125 inch
a	21000 ft/min/min
u	350 ft/min
N	13

Table 4.7 - Parameters - Rolling mill

Iteration	Objective function value	
	Sequential	Parallel
25	95.4819	95.8419
30	82.6454	82.6454
35	69.6380	69.6380
40	58.8388	58.8388
45	50.6059	50.6059
50	50.4190	50.2646
55	50.2377	49.0620
60	50.2165	48.9200
65	50.0379	

Table 4.8 - Results - Rolling mill

Once again nonconvexity of the overall objective function explains the difference in solutions obtained by two methods.

Conclusions

This dissertation discusses the fundamentals, enhancements and implementation of two new classes of algorithms for dynamic nondifferentiable optimization problems (DNDO). The algorithms presented in this dissertation illustrate the use of subgradient information for DNDO. Subgradient aggregation and selection concepts are also discussed. Use of decomposition-coordination in this environment are discussed in relation with the development of parallel algorithms. These developments are supported by the theoretical convergence properties and numerical examples. We establish that these algorithms converge to a global optimum for convex problems. Computational results shows that both parallel and sequential algorithms take approximately the same order of iterations to reach a certain accuracy. This leads to the conclusion that for problems with large number of stages, elapsed processing time can be considerably reduced by implementing the parallel algorithm on several processors.

Future Directions

Basic developments of this dissertation was aimed at a class of unconstrained convex multistage problems with nondifferentiable stagewise objective functions. These developments could be extended in several directions. Firstly the use of piecewise linear approximations could be replaced either partially or fully with piecewise quadratic functions. These piecewise quadratic functions could be generated using either value or subgradient information. Secondly these developments could be extended for constrained problems. Currently, constraints are added to the objective function as a penalty function. But introducing explicit constraints as first order approximations to the subproblems could lead to better and faster approximations.

LIST OF SYMBOLS

$A^k_{u_t}$: Coefficient of x_t of state, control linear relationship generated by the forward run at iteration k.

$A^k_{\eta_{1t}}$: Coefficient of x_t of state, stagewise objective function linear relationship generated by the forward run at iteration k.

$A^k_{\eta_{2t}}$: Coefficient of x_t of state, value function linear relationship generated by the forward run at iteration k.

$B^k_{u_t}$: Constant offset of state, control linear relationship generated by the forward run at iteration k.

$B^k_{\eta_{1t}}$: Constant offset of state, stagewise objective function linear relationship generated by forward run at iteration k.

$B^k_{\eta_{2t}}$: Constant offset of state, value function linear relationship generated by forward run at iteration k.

$f_t(x_t, u_t)$: Stagewise objective function of stage t.

g^l_{1t} : Coefficient of x_t of a subgradient inequality of stage t objective function.

g^l_{2t} : Coefficient of x_t of a subgradient inequality of stage t value function.

h^l_{1t} : Coefficient of u_t of a subgradient inequality of stage t objective function.

h^l_{2t} : Coefficient of u_t of a subgradient inequality of stage t value function.

I_{1t} : Index set of subgradient inequalities of stage t objective function.

I_{2t} : Index set of subgradient inequalities of stage t value function.

J_{u_t} : Jacobian matrix of transfer function $T_t(x_t, u_t)$ wrt u_t.

J_{x_t} : Jacobian matrix of transfer function $T_t(x_t, u_t)$ wrt x_t.

k : Iteration counter.

N : Number of stages.

r_k : Actual objective function value reduction at iteration k.

Superscript ^{k} : Corresponding elements of iteration k.

Superscript $^+$: Corresponding elements of Candidate policy.

S_t : Last stage of subsystem t.

t : Time or stage.

$T_t(x_t, u_t)$: State transition function of stage t to $t+1$.

u_t : Control variable of stage t.

$U_t(x_j, u_j)$: Control vector of stage t of a partial strategy generated by the forward run using (x_j, u_j) and exact transition function.

$\hat{U}_t(x_j, u_j)$: Control vector of stage t of a partial strategy generated by the forward run using (x_j, u_j) and a linearization of transition function.

$v_t(x_t, u_t)$: Subsystem t objective function.

x_t : State variable of stage t.

$X_t(x_j, u_j)$: State vector of stage t corresponding to $U_t(x_j, u_j)$.

$\hat{X}_t(x_j, u_j)$: State vector of stage t corresponding to $\hat{U}_t(x_j, u_j)$.

z_t : Final state of subsystem t set forth by the coordinating system.

α_t^k : Constant coordinating parameter of subsystem t at iteration k.

β_t^k : Coefficient of variable coordinating parameter of subsystem t at iteration k.

η_{1t}^k : Piecewise linear approximation of stage t objective function.

η_{2t}^k : Piecewise linear approximation of stage t value function.

κ : Set of iterations in which there is a descent.

λ_{1t}^l : Dual variable of constraints of set I_{1t}.

λ_{2t}^{l} : Dual variable of constraints of set I_{2t}.

μ_l : Line search parameter.

ξ_{tx}, ξ_{tu} : An element of subdifferential set of $f_t(x_t, u_t)$.

$\pi_t(x_t, u_t)$: Coordinating function of subsystem t.

ρ_k : Predicted objective function value reduction at iteration k.

ω_t : Intersection of a linear lower support of $v_t(x_t, u_t)$.

ω_{x_t} : Gradient of a linear lower support of $v_t(x_t, u_t)$.

$\partial f_t(x_t, u_t)$: The subdifferential set of $f_t(x_t, u_t)$.

LIST OF REFERENCES

Agmon S. [1954] "The relaxation method for linear inequalities." Canadian J.of Math. vol 6. pp 382-392

Allen E., Helgason R., Kennington J. and Shetty B. [1987] "A generalization of Polyak's convergence result for subgradient optimization." Math. Programming 37. pp 309-317

Bellman R. [1957] Dynamic Programming. Princeton University Press.

Bertskeas D.P., Lauer G.S., Sandell N.R. and Posbergh T.A. [1983] "Optimal short-term scheduling of large-scale power systems." IEEE Transactions on Automatic Control AC-28, No.1. pp 1-11

Bertsekas D.P. and Mitter S.K. [1973] "A Descent Numerical Method for Optimization Problems with Nondifferentiable Cost Functions." SIAM J. Control, Vol.11, No.4. pp 637- 652

Chang S.C., Chang T.S. and Luh P.B. [1986] "A hierarchical decomposition for large scale optimal control problems with parallel processing structure." Proceedings of the 1986 American Control Conference, Seattle Washington. pp 1995-2000

Christensen G.S and Soliman S.A. [1986] " Long term optimal operation of a parallel multireservoir power system." Journal of Optimization Theory and Applications vol 50 no 3. pp 383-395

Clarke F.H. [1976] "A new approach to Lagrange Multipliers." Mathematics of Operations Research Vol. 1 No. 2. pp 165-174

Clarke F.H. [1983] "Optimization and Nonsmooth Analysis." John Wiley & Sons. New York.

Dantzig G.B and Wolfe P. [1960] "Decomposition principle of linear programs." Operations research 8. pp 101-110

Eaves, B.C. & Zangwill, W.I. [1971] "Generalized cutting plane algorithms." SIAM J. Control, Vol 9. pp 529-542

Fletcher, R. [1981a] "Practical Methods of Optimization. Vol 1." John Wiley & Sons. New York.

Fletcher, R. [1981b] "Practical Methods of Optimization. Vol 2." John Wiley & Sons. New York.

Goffin J.L. [1977] "On convergence rates of subgradient optimization methods" Math. Programming 13. pp 329-347

Ho J.K. and Loute E. [1979] "A comparative study of two methods for staircase linear programs." ACM tran. on Math. Software V.15. pp 17-30

Ho J.K. and Manne A.S. [1974] "Nested decomposition for dynamic models." Math. programming 6. pp 121-140

Jamshidi M. and Wang C.M. [1985] "Hierarchical optimization of large-scale water resources systems." Real time Control of Large scale Systems. Schmidt G., Singh M., Titli A. and Tzafestas S. edt. Springer Verlag. Berlin.

Kelley,J.E. [1960] "The Cutting Plane Method for solving Convex Programs." J.SIAM, Vol.8. pp 703-712

Kiwiel,K.C. [1985] "Methods of Descent for Nondifferentiable Optimization." Lecture notes in mathematics. Dold,A. and Eckmann,B. edt., Springer-Verlag.

Lasdon L.S. [1970] "Optimization theory for large systems." McMillan Publishing Company.

Lasserre J.B. [1985] "A mixed forward-backward dynamic programming method using parallel computation." Jour. of Optimization Theory & applications. 45. pp 165-168

Lemarechal C. [1978] "Bundle methods in nonsmooth optimization." Nonsmooth optimization. Lemarechal,C. & Mifflin,R. eds. IIASA. proceedings 3, Pergamon, Oxford. pp. 79-102

Lemarechal C. and Mifflin R. [1978] "Nonsmooth optimization." IIASA. proceedings 3, Pergamon, Oxford.

Madsen K. and Jacobsen H. [1978] "Linearly constrained minimax optimization." Mathematical Programming 14. pp 208-223

Mayne D. [1966] "A second order gradient method for determining optimal trajectories for nonlinear discrete time systems." Int. J. Control 3. pp 85-95

Motzkin T.S. and Shoenberg I.J. [1954] "The relaxation method for linear inequalities." Canadian J.of Math. vol 6. pp 393-404

Murray D.M. and Yakowitz S.J. [1979] "Constrained differential dynamic programming and its application to multireservoir control." Water resources research 15. pp 1017-1027

Murray D.M. and Yakowitz S.J. [1981] "The application of optimal control methodology to nonlinear programming problems." Mathematical programming 21. pp 331-347

Noel M.C. and Smeers Y. [1986] "Nested Decomposition of Multi stage nonlinear programs with recourse." Math. Programming 34. pp 131-151

Ohno K. [1978] "Differential dynamic programming and separable programs." Journal of optimization theory and applications vol 24. pp 617-637

Overton,R.S. and Wilson,G.T.[1980] "A Comparison of the Minimum Principle and Differential Dynamic Programming." Numerical optimization of dynamic systems. Dixon,L.C.W. and Szego,G.P. edt., North-Holland Publishing Company. pp. 75-100

Poljak B.T. [1969] "Minimization of unsmooth functionals." USSR computational mathematics and mathematical physics 9. pp 14-29

Rakshit A. and Sen S. [1989] "Sequential rank-one/rank-two update for quasi-Newton differential dynamic programming." Optimal Control Applications and Methods 11 pp. 95-101

Ray W.H. and Szekely J. [1973] "Process Optimization, with Applications in Metallurgy and Chemical Engineering." John Wiley & Sons. New York.

Scheel C. and McInnis B. [1981] "Parallel Processing of Optimal-Control Problems by Dynamic Programming." Information Sciences 25. pp 85-114

Sen S. and Sherali H.D. [1986] "A class of convergent primal dual subgradient algorithms for nondifferentiable optimization." Math. programming 35 pp. 279-297

Sen S. and Yakowitz S.J. [1987] "A quazi-newton differential dynamic programming algorithm for discrete-time optimal control." Automatica to appear.

Serling M.J.H. and Irving M.R. [1985] "Distributed Computation for Real time Control of Electric Power Systems." Real time Control of Large scale Systems. Schmidt G., Singh M., Titli A. and Tzafestas S. edt. Springer Verlag. Berlin.

Shor N.Z. [1979] "Minimization methods for Nondifferentiable Functions." Springer-Verlag Berlin.

Shor N.Z., Shabashova L.P. [1972] "Solution of Minimix Problems by the Method of Generalized Gradient Descent with Dialation of the Space." Kibernetika No.1. pp 82-86.

Soliman S.A., Christensen G.S. and Abdel-Halim M.A. [1986] "Optimal operation of Multi Reservoir Power Systems Using Functional Analysis." Journal of Optimization Theory and Applications vol 49 no 3. pp 448-461

Turgeon A. [1980] "Decomposition methods for the Long-Term Scheduling of Reservoirs in Series." Water resources Research vol 17. pp 1565-1570

Wittrock R.J. [1985] "Dual nested decomposition of staircase linear programs." Math programming study 24. pp 65-86

Yakowitz S.J. [1983] "Convergence rate analysis of the state increment dynamic programming method." Automatica vol 19. pp 53-60

Yakowitz S.J. [1986] "The stagewise Kuhn-Tucker conditions and differential dynamic programming." IEEE Trans. Automatic control AC-31. pp 25-30

Yakowitz S.J. and Rutherford B. [1984] "Computational aspects of discrete time optimal control." Appl. Math. computation 15. pp 29-45