VOT 71910

DEVELOPMENT OF OPTIMIZATION ALGORITHMS FOR UNCERTAIN NONINEAR DYNAMICAL SYSTEM

(PEMBANGUNAN ALGORITMA PENGOPTIMUMAN UNTUK SYSTEM TAK LINEAR TAK PASTI)

MOHD ISMAIL BIN ABD AZIZ

ROHANIN BINTI AHMAD

RESEARCH VOT NO:

71910

Jabatan Matematik

Fakulti Sains

Universiti Teknologi Malaysia

2007

ACKNOWLEDGEMENT

I would like to express my utmost gratitude to Research Management Center, Universiti Teknologi Malaysia for providing the research grant to support this work. Most of the work presented in this report is jointly by the written of Dr. Rohanin Ahmad, my PhD student. Thank you for your hardwork and perservearance in contributing to this research.ssor ouDr. Mohd. Ismail Abd. Aziz for his excellent supervision. My thanks to my MSc project students, Ng Hai Yan, Kek Sie Long and Norliza Abdullah their contribution in testing some features of the algorithms. I would also like to thank Dr. V.M. Becerra (University of Reading, UK) for his idea on using momentum terms and his valuable help to us especially on the understanding of MATLAB. A special thank you for Prof. P.D.Roberts (City University, UK) for his valuable papers, kind words, and suggestions. Last but not least, I would like to convey my most sincere gratitude to my friends and colleagues who have contributed directly or indirectly towards this research.

DEVELOPMENT OF OPTIMIZATION ALGORITHMS FOR UNCERTAIN NONINEAR DYNAMICAL SYSTEM

(*Keywords:Optimal Control,nonlinear, model reality,DISOPE, momentum terms)*

Nonlinear optimal control problems are problems involving real world situations where the objectives are the maximization of the return from, or the minimization of the cost of, the operation of physical, social, and economic processes. Algorithms used to solve these problems are expected to satisfy the objectives consistently and since time translates into cost, must also be fast. An algorithm that definitely can satisfy the objectives is the *Dynamic Integrated Systems Optimization and Parameter Estimation* (DISOPE) algorithm. However, this algorithm has an inherent problem of slow convergence due to its gradient descent type updating mechanism. Hence, the purpose of this study is to overcome this convergence problem by modifying the mechanism. Two approaches were chosen for this purpose. The first is the use of momentum terms and the second is the parallel tangent method. Two new algorithms named DISOPE-MOMENTUM and DISOPE-PARTAN sprouted from these modifications and extensive simulations were performed to observe their performances. To strengthen the findings, theoretical analyses were done on each algorithm. These include optimality, stability, convergence, and the rate of convergence analyses. Based on the results of these simulations, we compared the number of iterations needed by each algorithm to arrive at the optimal solution and the CPU time taken for each algorithm to execute the search. From the theoretical analyses, comparisons were done on the speeds of contraction of the algorithms. Both new algorithms managed to arrive at the optimum in fewer numbers of iterations and in shorter CPU times than DISOPE without compromising on the accuracy of the solutions. The new algorithms also boast faster contractions. Both new algorithms performed better than DISOPE. This study succeeded in overcoming the problem of slow convergence and with the modifications, the new algorithms become more efficient in solving the optimal control problems.

Key researchers:

Asc. Prof. Dr. Mohd Ismail Abd Aziz
Rohanin Ahmad

Email: m_ismail@mel.fs.utm.my
Tel. No: 07-5534231 / 019-7538204
Vote No: 71910

ABSTRAK

Masalah kawalan optimum tak linear adalah masalah dunia nyata yang berobjektifkan pemaksimuman hasil, atau peminimuman kos operasi fizikal, proses-proses sosial atau ekonomi. Algoritma yang ingin digunakan untuk menyelesaikan masalah-masalah ini harus berupaya memenuhi objektif tersebut, dan disebabkan masa boleh ditafsir sebagai kos algoritma ini juga harus pantas. Sebuah algoritma yang berupaya memenuhi objektif di atas ialah algoritma *Pengoptimuman Sistem Bersepadu Dinamik dan Penganggaran Parameter*. Walau bagaimanapun, algoritma ini mempunyai masalah penumpuan lamban yang diwarisi daripada mekanisme pengemaskiniannya yang tergolong ke jenis penurunan gradien. Tujuan kajian ini ialah mengatasi masalah penumpuan di atas dengan cara mengubahsuai mekanisme tersebut. Dua pendekatan telah dipilih untuk tujuan ini. Yang pertama menggunakan sebutan momentum dan yang kedua menggunakan kaedah tangen selari. Dua algoritma baru yang dinamakan DISOPE-MOMENTUM dan DISOPE-PARTAN terhasil daripada ubahsuaian ini dan simulasi secara ekstensif telah dijalankan untuk meninjau prestasi masing-masing. Untuk memampankan hasil penemuan, analisis secara teori telah dilakukan untuk setiap algoritma. Analisis-analisis ini termasuk analisis keoptimuman, kestabilan, penumpuan dan analisis kadar penumpuan. Berdasarkan hasil simulasi, kami bandingkan bilangan lelaran yang diperlukan oleh setiap algoritma untuk mencapai penyelesaian optimum dan juga masa CPU yang diambil oleh setiap algorithma untuk melaksanakan carian. Daripada analisis secara teori, perbandingan telah dibuat terhadap kecekapan, kerumitan dan kepantasan pengecutan algoritma. Kedua-dua algoritma baru ini berupaya mencapai optimum masing-masing dengan bilangan lelaran dan masa CPU kurang daripada DISOPE, tanpa menjejas kejituan penyelesaian. Pengecutan algoritma-algoritma ini juga lebih pantas. Kedua-dua algoritma baru ini melaksanakan tugas lebih baik daripada DISOPE. Kajian ini berjaya mengatasi masalah penumpuan lamban dan dengan pengubahsuaian yang disyorkan, algoritma-algoritma baru ini menjadi lebih cekap dalam menyelesaikan masalah kawalan optimum.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

a_i	-	The input of node i
A	-	A time-invariant state matrix for the system dynamics of an LQR model
$\left.\begin{matrix} A_{mp}, B_{mp}, C_{mp}, \\ D_{mp}, F_{mp}, J_{mp} \end{matrix}\right\}$	-	Constant coefficient matrices of the multipass processes
b^{i+1}	-	The initial conditions, disturbances, and control input effects
B	-	A time-invariant control matrix for the system dynamics of an LQR model
B_0	-	As defined in Equation (3.76)
A^*	-	As defined in Equation 3.22
B^*	-	As defined in Equation 3.22
C	-	As defined in Equation 3.46
C_1	-	As defined in Equation 8.33
C_N	-	The learning rate
C_{2n+m}	-	Bounded mappings
$d^i(t)$	-	Interpass disturbance
D	-	A set of constraints
D_1	-	As defined in Equation 3.79
D_1^M	-	As defined in Equation 7.16
D_3	-	As defined in Equation 8.32
$E(\mathbf{w})$	-	Error function of a back propagation algorithm
$E_y^{(i)}(t)$	-	Error function of DISOPE

E^n	-	The Euclidean space
E_0	-	As defined in Equation 3.83
$\tilde{E}_0$	-	As defined in Equation 3.82
E_1, E_2	-	Matrices representing the contributions from r_1, r_2
E_ς	-	A Banach space
$f(\cdot)$	-	Plant dynamics of the model
$f^*(\cdot)$	-	Plant dynamics of the real process
$f(n), g(n)$	-	Complexity functions, with n input size
$F(\lambda)$	-	As defined in Equation 3.97
$F_P(\lambda)$	-	As defined in Equation 8.31
$g_1(\cdot), g_2(\cdot)$	-	Vectors representing the model reality differences
G	-	The gradient descent direction
$G(t)$	-	A $m \times n$ Kalman gain matrix
h	-	Step size
h_1, h_2, h_3	-	Lipschitz constants
$h_1^{DM}, h_2^{DM}, h_3^{DM}$	-	Lipschitz constants for DISOPE-MOMENTUM
h	-	Input size in time complexity analysis
$H(\cdot)$	-	The Hamiltonian
i, j	-	Indices
I	-	The identity matrix
J	-	The performance index of the LQR model
$\tilde{J}$	-	As defined in Equation 3.65
J^*	-	The real cost functional or performance index
J'	-	The Lagrangian functional
J_{ea}	-	The performance index of the augmented EOP
J_{EOP}	-	The performance index of EOP
J_{MOP}	-	The performance index of MOP
J_{MMOP}	-	The performance index of MMOP
$K(t)$	-	A time-varying $n \times n$ matrix

K_y	-	As defined in Equation 3.41
$\mathrm{l}_1, \mathrm{l}_2$	-	Lines
$L(\cdot)$	-	Performance index of the model
$L^*(\cdot)$	-	Real performance measure function
L_ς	-	A bounded linear operator of E_ς into itself
M	-	The momentum direction
$o(f(n)), O(f(n))$ $\Theta(f(n)), \Omega(f(n))$	-	Sets of complexity functions
$p(t)$	-	The costate vector
p_0, p_1, K	-	Search points
P	-	PARTAN step direction
$P^*(y^{(i)}(t_f), t, t_0)$	-	As defined in Equation 3.52
P_y	-	A matrix of PARTAN parameters
Q	-	Symmetric state weighting matrix for the LQR model
$r(L_\varsigma)$	-	Spectral radius
$r_1,\ r_2$	-	Scalar modification factors
R	-	Symmetric control weighting matrix for the LQR model
$\bar{R}, \bar{Q}$	-	The augmented weighting matrices
S, V	-	Weights for the terminal conditions of the LQR model
$\hat{S}$	-	An arbitrary set in the Euclidean space
S_j	-	The sum of all weighted inputs from node i
t, τ	-	Time
t_0	-	Initial time
t_f, T	-	Terminal time
t_j	-	A set of target outputs
$u(t)$	-	The control vector
u^o	-	Optimal control
$\hat{u}(t),\ \hat{x}(t),\ \hat{p}(t)$	-	Variables used in the optimization step

$v(t), z(t), P(t)$	-	Variables used in the updating step
V^*	-	As defined in Equation 3.22
$\mathbf{w}$	-	A vector of all weights between nodes i and j
W_y	-	A matrix of momentum parameters
W_ς	-	A linear subspace
$x(t)$	-	The state vector
x^o	-	Optimal state
x_0	-	Initial state
x_j	-	Actual output of a network
$y^{(i)}(t)$	-	The process output
$\overset{\prime}{y}$	-	Limit point of the sequence of terms
$Y^i(t)$	-	Pass profile i
Y^∞, y^∞	-	The limit profile
$\alpha(t),\ \gamma(t)$	-	Parameter estimates for the value differences between reality and model
β_s	-	A search parameter
$\Gamma_1^{(i)}$	-	As defined in Equation 3.23
$\Gamma_2^{(i)}$	-	As defined in Equation 3.23
η	-	Step size parameter of the gradient descent algorithm
$\eta(t_f, t, t_0)$	-	As defined in Equation 3.52
η_m	-	Momentum Learning rate of the back propagation
θ	-	A predetermined threshold value
$\vartheta_1, \vartheta_2, \vartheta_3$	-	Contraction coefficients
κ	-	A constant; $\kappa \in [0, \infty)$
$\left.\begin{matrix}\lambda(t), \beta(t), \\ \mu(t), \xi(t)\end{matrix}\right\}$	-	Scalar multipliers
$\bar{\lambda}(t), \bar{\beta}(t)$	-	Augmented scalar multipliers
D	-	Number of instances in time complexity analysis
$\mu\ (t_f, t, t_0)$	-	As defined in Equation 3.52

μ_x, μ_p	-	As defined in Equation 3.50
$\wp_u, \wp_x, \wp_p$	-	PARTAN parameters
ς	-	Pass length
$\sigma(t_0, t_f)$	-	As defined in Equation 3.109
$\phi(\cdot)$	-	Final weighting function – scalar valued
$\phi(t, \tau)$	-	As defined in Equation 3.33
$\bar{\phi}_{21}(t_f, t_0), \bar{\phi}_2(t, \tau)$	-	As defined in Equation 3.36
$\tilde{\phi}_{11}(t_f, t_0), \tilde{\phi}_1(t, \tau)$	-	As defined in Equation 3.37
$\tilde{\phi}_{21}(t_f, t_0), \tilde{\phi}_{21}(t_f, t_0), \tilde{\phi}_2(t_f, \tau)$	-	As defined in Equation 3.38
Φ	-	Symmetric terminal weighting matrix for the LQR model
$\acute{\Phi}$	-	As defined in Equation 3.63
Φ^*	-	Real terminal measure
χ	-	Lagrange multiplier
$\acute{\Psi}$	-	As defined in Equation 3.64
Ψ^*	-	Real terminal constraint vector
$\bar{\Psi}^*$	-	As defined in Equation 3.36
$\tilde{\Psi}^*$	-	As defined in Equation 3.37
$\varpi, \varpi_u, \varpi_x, \varpi_p$	-	Momentum parameters
$\Omega(t_f, t, t_0, \tau)$	-	As defined in Equation 3.53
$\nabla f, g$	-	Gradient of a function
$(\cdot)^{(\infty)}$	-	Limit profiles
$\|\cdot\|_2$	-	The Euclidean norm used in the thesis
DISOPE	-	Dynamic Integrated Systems Optimization and Parameter Estimation
DISOPE-MOMENTUM	-	DISOPE with momentum terms
DISOPE-PARTAN	-	DISOPE with PARTAN step
EOP	-	Expanded Optimal Control Problem
ISOPE	-	Integrated Systems Optimization and Parameter

		Estimation
MMOP	-	Modified Model Based Optimal Control Problem
MOP	-	Model Based Optimal Control Problem
PARTAN	-	Parallel tangent

CHAPTER 1

INTRODUCTION

1.0 The Continuous Optimal Control Problem

Suppose that a plant is described by the time-varying dynamical equation

$$\dot{x} = f^*(x(t), u(t), t) \tag{1.1}$$

where $f^* : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ representing a set of equations describing the process with $x(t) \in \mathbb{R}^n$ as the state vector, $u(t) \in \mathbb{R}^m$ as the control input, and $t \in \mathbb{R}$ as the time. Let the functional

$$J^* = \phi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t), u(t), t)dt \tag{1.2}$$

be the associated cost function or performance index, where $[t_0, t_f]$ is the time interval. In Eq. (1.2) $\phi : \mathbb{R}^n \to \mathbb{R}$ is a scalar valued function called the final weighting function, which depends on the final state and final time. The weighting function $L^* : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ is a continuous function, and it depends on the state and input at intermediate times in $[t_0, t_f]$. In both Eqs. (1.1) and (1.2), $(\cdot)^*$ represents the original problem formulation.

By assuming that the state of the system at initial time is given with value $x(t_0) = x_0$, and with Eq. (1.1) as the only constraint considered on the values of the control and state variables, the optimal control problem can be stated as follows. Find the control input $u^o(t) \in \mathbb{R}^m$ on the time interval $[t_0, t_f]$ that drives the plant (1.1) along a trajectory $x^o(t) \in \mathbb{R}^n$ such that the performance index given by Eq. (1.2) is minimized. Mathematically, the problem can be written as follows.

$$\min_{u(t)} J^* = \phi(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t), u(t), t)dt \tag{1.3}$$

subject to

$$\dot{x} = f^*(x(t), u(t), t) \tag{1.4}$$

$$x(t_0) = x_0 \tag{1.5}$$

In order to emphasize time as the argument of the functions, they are referred to as either the *control trajectory*, which means the time path of the control vector, or the *state trajectory*, which means the time path of the state vector. If time does not enter explicitly as an argument of f^*, we say that the system is autonomous.

1.1 Methods for Solving the Continuous Optimal Control Problem

Two established methods for accomplishing the minimization are the method of Dynamic Programming developed by Bellman (1957) and the variational approach of Pontryagin (Pontryagin *et al.* 1962).

The method of dynamic programming can handle control and state constraints (Becerra, 1994). However, when solving realistic problems, the dynamic programming algorithms face problems referred to as the "curse of dimensionality". This curse causes the algorithms to exceed the memory capacity of computers when the system has more than two or there state variables. Nevertheless, if the state space is limited to a region close to a nominal optimum path, the Dynamic Programming problem can often be well approximated by a linear-quadratic problem, that is a problem with linear (time-varying) dynamics and a quadratic performance index whose (time-varying) weighting matrices are the second derivatives of the states and the controls (Bryson, 1996). This is the classical Accessory Minimum problem, the basic problem for examining the second variation in the calculus of variation. The Accessory Minimum problem can be formulated as a time-varying linear two-point boundary-value problem.

The variational approach of Pontryagin is called the minimum principle. It generalizes the calculus of variations to include problems where optimization is not achieved by calculus. It is an extension of Weierstrass' necessary condition to cases where the optimal functions are bounded. It follows directly from the general continuous-time dynamic programming equations.

The minimum principle deals with one extremal at a time. In optimal control terminology, its states that a minimizing path must satisfy the Euler-Lagrange equations where the optimal controls maximize the Hamiltonian within their bounded region at each point along the path (Hocking, 1991). This transforms the calculus of variation problem to a nonlinear problem at each point along the path.

1.2. The Variational Approach to Optimal Control Law

The algorithms discussed in this research used the variational approach in their course toward finding the solutions of the optimal control problems. Thus we steer our discussion in the direction of this approach.

The optimal control problem defined in Eqs. (1.3), (1.4), and (1.5) in Sec.1.1 is a constrained minimization problem. The variational approach for solving this problem is to eliminate the existence of constraints and turn the problem into an unconstrained problem. Adjoining the constraints to the performance index using Lagrange multipliers does this. Thus a new functional called the *Lagrangian functional* is defined. This approach was introduced by Lagrange (Rao, 1984). According to the Lagrange theory, the minimum of the original problem is achieved by finding the minimum of the new *unconstrained* functional (Lewis and Syrmos, 1995).

Suppose that $p(t)^T \in \mathbb{R}^n$ is the multiplier for the system defined by Eq. (1.4). Then, the new Lagrangian functional to be minimized can be defined as

$$J' = \phi(x(T),T) + \int_{t_0}^{T} [L(x(t),u(t),t) + p(t)^T (f(x(t),u(t),t) - \dot{x}(t))]dt \qquad (1.6)$$

where $p(t)$ is usually referred to as the *costate* function. From Eq. (1.6), if we define a function termed the Hamiltonian, which is

$$H(x(t),u(t),p(t),t) = L(x(t),u(t),t) + p(t)^T f(x(t),u(t),t), \quad (1.7)$$

then Eq. (1.6) can be rewritten as

$$J' = \phi(x(T),T) + \int_{t_0}^{T} [H(x(t),u(t),p(t),t) - p(t)^T \dot{x}(t)]dt \quad (1.8)$$

The problem is now reduced to a problem of finding the minimum of a functional represented by Eq. (1.8). From the calculus of variation, we know that the minima of such functionals happened at points where the gradients have the values of zero. The conditions where the gradients diminished are called the *necessary optimality conditions*. Thus to find the solution of this problem, we have to clearly define these conditions.

1.2.1 The Necessary Optimality Conditions

The necessary optimality conditions for solving Eq. (1.8) are derived by setting all the gradients of the Hamiltonian with respect to $x, u,$ and p, respectively, to zero (Lewis and Syrmos, 1995). The criteria are listed below.

$$\nabla_x H + \dot{p}(t) = 0 \text{ or } -\dot{p}(t) = \nabla_x H \quad (1.9)$$

$$\nabla_u H = 0 \quad (1.10)$$

$$\nabla_p H - \dot{x}(t) = 0 \text{ or } \dot{x}(t) = \nabla_p H \quad (1.11)$$

Eqs. (1.9), (1.10), and (1.11) are also referred to as the *costate* equation, *stationarity* condition, and the *state* equation respectively. Eqs. (1.9) and (1.10) are also called Euler's or Euler-Lagrange equations (Lewis and Syrmos, 1995; Bryson, 1996). The solution of the optimal control problem defined by Eqs. (1.3), (1.4), and (1.5) is arrived at by solving Eqs. (1.9), (1.10), and (1.11) together with the following boundary conditions

$$x(t_0) = x_0 \quad (1.12)$$

and

$$p(t_f) = \nabla_x \phi(x(t))\big|_{t=t_f} \quad (1.13)$$

Even though the manner for finding the solution of the continuous optimal control problem is well outlined, the actual job of finding the solution is by no means effortless. The variational approach to the necessary optimality conditions leads to a nonlinear two-point boundary-value problem (Kirk, 1970) having boundary conditions specified at two separate points in time (Jacob, 1974) given by Eqs. (1.12) and (1.13).

The optimal control law depends on the solution of the nonlinear two-point boundary-value problem, which in general, are difficult problems to solve (Kirk, 1970; Jacob, 1974; Bryson and Ho, 1975; Lewis and Syrmos, 1995) due to the high degree of nonlinearity involved. Analytical solutions are close to impossible; therefore, numerical solutions are usually sought for. Techniques available for solving the nonlinear two-point boundary-value problem include the steepest descent, variations of extremals, and quasilinearization (Kirk, 1970). All these methods begin with the necessary optimality conditions obtained from the application of the minimum principle of Pontryagin.

1.2.2 Difficulties Facing the Nonlinear Optimal Control Problems

Most optimal control problems are very nonlinear. Without a doubt, a ‘good’ model of the real dynamical process must have high degree of nonlinearity. However, using a good model representation of the real problem does not translate into ease of solution. Basically, there are at least three foreboding problems with nonlinear optimal control problem as described above. The first one is due to the variational approach itself, which ends up with a nonlinear two-point boundary-value problem as the necessary optimality conditions. As mentioned earlier, these problems are difficult to solve.

The second problem is associated with having a nonlinear function as the objective function. The objective function has many local minima that are not the global optimum. If the search used is local in nature, the global optimum might not even be located. The third problem is due to the nonlinear set of constraints. The set

of constraints might define a feasible region that is difficult to find (Shang, 1997). Furthermore, even when the feasible region is located, if both the objective functions and the set of constraints are not convex, the convergence to a global optimum is not guaranteed (Stoer and Witzgall, 1970; Koo, 1977; Cesari, 1983; Bunday, 1984; Rao, 1984; Beale, 1988; Shang, 1997).

Hence, a class of methods tries to do away with the 'good' model and instead use 'easy-to-solve' models to approximate the original problems. Here is where the Linear-Quadratic-Regulator (LQR) problem comes in handy.

1.3 LQR Problem as Model

To overcome the difficulties mentioned above, we resort to modeling the original problem defined by Eqs. (1.3) - (1.5), which consisted of both nonlinear system dynamics and performance index with 'simpler' functions. For this purpose, we use the linear-quadratic regulator (LQR) problem. With LQR as model, the system dynamics are represented as linear differential equations and the performance index is a quadratic function in terms of the state and control variables. In LQR problems, the resulting two-point boundary-value problem is linear and is relatively easy to solve (Becerra, 1994), obtaining a linear optimal control law (Kirk, 1970). In LQR problems it was found that it is possible to obtain the optimal law, by numerically integrating a matrix differential equation of the Riccati type.

With the LQR as model, the optimal control problems are defined as follows:

$$\min_{u(t)} J = \frac{1}{2} x(t_f)^T \Phi x(t_f) + \int_{t_0}^{t_f} \frac{1}{2}\left[x(t)^T Q x(t) + u(t)^T R u(t) \right] dt \qquad (1.14)$$

subject to

$$\dot{x} = Ax(t) + Bu(t) \qquad (1.15)$$

$$x(t_0) = x_0 \qquad (1.16)$$

with Φ and Q are symmetric and positive semi-definite, and R symmetric and positive definite weighting matrices having the appropriate dimensions. A and B are

the time-invariant matrices of the system dynamics and control distribution respectively. The Hamiltonian function defined in Eq. (1.7) becomes

$$H = \frac{1}{2}\left(x(t)^T Qx(t) + u(t)^T Ru(t)\right) + p(t)^T \left(Ax(t) + Bu(t)\right) \tag{1.17}$$

with the state equation becoming

$$\dot{x} = Ax(t) + Bu(t) \tag{1.18}$$

and the costate equation

$$-\dot{p}(t) = Qx(t) + A^T p(t) \tag{1.19}$$

The stationarity condition is now

$$0 = Ru(t) + B^T p(t) \tag{1.20}$$

Furthermore, by rearranging Eq. (1.20) we get the basic expression for the control input as

$$u(t) = -R^{-1}B^T p(t) \tag{1.21}$$

The boundary conditions become

$$x(t_0) = x_0 \tag{1.22}$$

and

$$p(t_f) = \nabla_x \phi(x(t))\big|_{t=t_f} = \Phi(t_f)x(t_f) \tag{1.23}$$

1.3.1 Solving the LQR Optimal Control Problem

Kalman showed that the linear quadratic optimal control problem could be solved numerically in an elegant, efficient manner with a "backward sweep" of a matrix Ricatti equation (Bryson, 1996). Jacopo Francesco Ricatti (1676-1754) gave the scalar form of his equation for solving linear second-order two-point boundary-value problems (Bryson, 1996). With LQR as the model, the problem is specified by Eqs. (1.18) and (1.19). To apply the method, we first assume that $x(t)$ and $p(t)$ satisfy a linear relationship in the form of

$$p(t) = K(t)x(t) \tag{1.24}$$

for all $t \in [t_0, t_f]$, where $K(t)$ is a time-varying $n \times n$ yet to be determined matrix function. The following noniterative solution procedure, outlined by Becerra (1994),

is to be followed to get the solution of the LQR optimal control problem using the back-sweep method.

Procedure 1.1: Simple LQR solution

Step 1: Solve backward from t_f to t_0 the following Ricatti differential equation, with terminal condition $K(t_f) = \Phi(t_f)$:

$$\dot{K}(t) = K(t)BR^{-1}B^T K(t) - A^T K(t) - K(t)^T A + Q \qquad (1.25)$$

Step 2: Compute the state $x(t)$, $t \in [t_0, t_f]$ by integrating from the initial condition $x(t_0) = x_0$ the following equation:

$$\dot{x}(t) = \left(A - BG(t)\right) x(t) \qquad (1.26)$$

where $G(t) = R^{-1}B^T K(t)$ is the Kalman gain.

Step 3: Compute the optimal control $u(t)$, $t \in [t_0, t_f]$ from the state feedback control law:

$$u(t) = -G(t)x(t) \qquad (1.27)$$

With LQR as model, the first of the problems listed in Section 1.2.5 above is, without a doubt, solved. The beauty of using LQR as model is that the remaining two problems are unwittingly solved as well. With a quadratic as the objective function, the existence of many local minima is no longer a problem. A quadratic function has only one local minimum. The constraints of LQR problems are linear and thus the feasible region is well defined. One of the attractive properties of a LQR problem is its convexity. The main importance of convexity comes from the following proposition.

Proposition 1.1 (Beale, 1988)

If the region ¡ of the feasible region determined by the constraints of an optimization problem is convex and the objective function $f(x)$ is a convex function in ¡ , then any local minimizer of $f(x)$ in ¡ is also a global minimizer.

See Beale (1988) or Bunday (1984) for the proof.

With the LQR problem, its feasible region defined by a set of linear constraints is convex (Beale, 1988). Since twice differentiable functions are convex if their Hessian is positive semi-definite, its quadratic objective function is also convex. Hence, the LQR problem satisfies Proposition 1.1. Thus, any solution acquired by a search using it as a model is guaranteed to achieve the global optimum.

Clearly, using LQR as model overcame the problems of the nonlinear optimal control problem in general. However, the usage of LQR has problems of its own. The one glaring problem is the model-reality differences that surfaced when a very nonlinear problem is modeled by 'simple' functions.

1.4 Model-Reality Differences

Since LQR is a simplified model of the original optimal control problem, the matter of model-reality differences cannot be ignored. An algorithm that uses this approach would not be solving the original problem but rather solves the simplified LQR problem. The solution to the LQR problem is hoped to approximate the real solution or in other words to converge to the real solution. Thus, a good algorithm has to take into account the model-reality differences to be successful. One such algorithm is the *Dynamic Integrated Systems Optimization and Parameter Estimation* (DISOPE) algorithm.

1.4.1 DISOPE Algorithm

DISOPE is an algorithm specifically aimed at solving dynamic nonlinear optimal control problems. It was first developed by Roberts (1993) and further improved by Becerra (1994). DISOPE takes into account the model-reality differences in structure and parameters of the problems to be solved. The method is iterative in nature. Repeated solutions of optimization and estimation of parameters

within the model is used for calculating the optimum (Roberts, 1993). An important property of the technique is that the iterations converge to the real optimum in spite of the model-reality differences. An implementable algorithm based on LQR has been designed and implemented in MATLAB by Roberts (1993). For the discussions in this research, the model used will be the LQR model.

The algorithm integrates the information from the real problem and its simplified model by introducing parameters such that the solution of the model provides the control as a function of the current parameter estimates. These estimates in turn are obtained by matching model and real states and performance at the current computed control. In this way, the two problems of optimization and parameter estimation interact. To properly integrate the two problems, different notations for controls are introduced to separate the signals of the optimization problem from the parameter estimation problem and application to reality. The same is done for the states signals of the two problems. A set of additional constraints is defined by matching the different signals from the two problems. The pair of constraints signifies the interconnection between reality and model.

The algorithm uses the back sweep method to generate trial solutions. The solutions are then updated using a mechanism that is based on the gradient descent method. Left as is, the algorithm suffers from the same setbacks as the gradient descent method; that is the slow convergence and the possibility of converging to false minimum. These two problems are well known problems of the gradient descent approach. The problems are so significant that a vast wealth of literature is available on the manners tried and tested to overcome these problems. This research proposed methods of improving DISOPE by modifying its updating mechanism so as to make it resilient in the face of the problems created by the gradient descent part of it.

1.4.2 Problems Faced by Gradient Descent Methods

DISOPE has an updating mechanism that is of the gradient descent type. Gradient descent algorithms are common for having the problem of slow

convergence and false optimum. One of the causes of slow convergence is oscillation of the search algorithm. This usually happens when the iterative computed solution of the algorithm nears an optimum be it local or global. In these vicinities, the surface to be traversed formed ravines. The surfaces of the ravines generate almost perpendicular gradients, making the trial solutions advance in small steps. These oscillations retard the journey of the search to the optimum hence the slow convergence.

For nonlinear problems, where the surface of the objective function contains an abundance of local minima, the solution attained is not guaranteed to be the global optimum. This could easily happen when the ravine in question contains a local optimum. The search algorithm might get trapped without being able to get out and find the global optimum. This case is referred to as convergence to a false optimum. Having the descent method as part of it categorizes DISOPE as a local search procedure. Being a local search algorithm, it does not have the mechanism to escape local minimum once stuck in it.

The other problematic situation faced by the gradient descent methods is when it encounters flat surfaces. Flat surfaces potentially hide both problems mentioned above; slow convergence and false optimum. The problem arises when the flat surface to be negotiated is large. Flat surfaces deliver very small gradients for the search to rapidly move across the area contributing to slow convergence. Occasionally, the gradient is too small such that the norm between two consecutive terms is negligible that the tolerance set for the algorithm to stop is met. When this happens, the solution of the algorithm is also a false optimum.

In short, there are three distinct problems recognized here. The first is the problem of oscillation caused by ravines or flat basins on the surface of the objective function. Second is the convergence to false optimum because the search gets trapped in a local minimum. The third is the convergence to false minimum caused by flat surfaces.

Since DISOPE discussed in this research uses LQR problems as model, the second problem listed here is no longer valid. Thus we are faced with the remaining two problems. These two problems affect the convergence the algorithm, if it converges at all. The physical manifestation of these problems is the slowness of convergence.

1.5 Statement of Problem

DISOPE is a newly developed algorithm (Roberts, 1993; Becerra, 1994) and as such, there are bound to be inherent weaknesses that need to be addressed. From the discussion above, we have identified that a weakness of DISOPE is slow convergence. The slow convergence has been identified to be the result of oscillating search either in the areas of ravines or on flat basins with very small gradients, which has a secondary problem of converging to false optima.

This research aims to improve DISOPE so as to overcome the problem of slow convergence. The research also seeks to provide the relevant theoretical results to support the findings.

1.6 Research Objectives

Specifically, this research addresses the problems of slow convergence of DISOPE. This research goal is overcome this shortcoming and make DISOPE a more attractive algorithm. To achieve the goal we have the following objectives as guideline. The objectives of the research are

- To decompose DISOPE into two distinct maps with one of them being the updating mechanism
- To establish an error function for the updating mechanism
- To establish that gradient descent algorithm is a component of DISOPE

- To establish the convergence rate of DISOPE
- To develop new algorithms by modifying DISOPE according to the chosen improvements
- To carry out simulations on chosen problems to see the effects the modifications have on the convergence speed
- To compare the efficiency of the new algorithms with DISOPE

The products of this research would be new algorithms that can handle the problem inherent to the gradient descent algorithm successfully.

1.7 Scope of Research

For this research, the modifications done to DISOPE are at its updating mechanism. The modifications to the updating mechanism are primarily aimed at improving the performance of the gradient descent algorithm. Alternative methods such as replacing the gradient descent algorithm with other methods were never considered.

On improving the performance of the gradient descent procedure, principally the improvements done to the back propagation algorithm of the neural networks are explored. Out of the numerous improvements documented, two simple and effective modifications are chosen and tried with the updating mechanism of DISOPE. The two are the addition of momentum terms and the use of parallel tangent method.

The time-complexity analyses done in this research are based on the fact that the new algorithms are basically similar to DISOPE with differences only in the updating mechanisms. The optimality, stability, and convergence analyses of the new algorithms follow that of the established DISOPE closely. Primarily the aim of these analyses is for comparison of the performance of the new algorithms with the performance of DISOPE.

1.8 Contributions of the Research

The contributions of this research are primarily to the field of optimal control. The contributions are appreciably in the realization of two new algorithms. The new algorithms are equipped with supporting theoretical analyses to reinforce their appeals. These algorithms are implemented using MATLAB and tested with simulation examples. Further analyses of the original algorithm DISOPE are also provided in the report as enhancements to its theoretical basis.

1.8.1 Contribution to Algorithm Development

Two new algorithms have been developed based on the amendments of the updating mechanism of DISOPE. They are named DISOPE-MOMENTUM and DISOPE-PARTAN. Both of the algorithms maintained the core design of DISOPE. However, the two new algorithms have improved convergence properties due to modifications in the updating mechanisms. Both algorithms have been implemented in software and tested with simulation examples. The two new algorithms proved to be superior in convergence speed over DISOPE.

1.8.2 Contribution to Theoretical Analysis

We added three new theoretical analyses for DISOPE. The first is the verification that DISOPE is a type of a gradient descent method. We used the composite mapping theory to separate the updating mechanism from the rest of the algorithm. The other two are the time-complexity and convergence rate analyses. They are enhancements to DISOPE and act as additional bases for comparison over the efficiency of the new algorithms.

Optimality studies of DISOPE-MOMENTUM and DISOPE-PARTAN are presented in this report. To have sound footing for the new algorithms, the stability and convergence analyses are also offered. These analyses are based on the unit memory repetitive process that falls naturally into the area of 2-D systems. Studies on time-complexity analysis and the individual algorithms' convergence rates have also been derived. All these analyses are compared to same analyses of DISOPE to gauge their efficiency. All these theoretical analyses are important measures of the credibility and merit of the new algorithms. They provide confidence and attractiveness to perspective users.

1.8.3 Contribution to Software Implementation and Algorithm Testing

The two new algorithms have implementations in software using MATLAB as the tool for the programming language. Simulations of several examples with differing levels of difficulties have been carried out with the software. These simulations helped us distinguish and comprehend the effects of the modifications on the original algorithm. They helped us come to a decision that the improvements obtained are worthwhile of the new algorithms. These simulations also permit us to test the new algorithms and make comparison of the results with the performance of the original algorithm DISOPE.

1.8.4 Contribution to the Field of Gradient Descent Algorithm

The subject matter of the research is the gradient descent algorithm of the updating mechanism of DISOPE. The tools used in the research are the momentum terms and the PARTAN algorithm. Both tools improved the performance of the gradient descent algorithm. The problems solved in the simulations also add variety to the cache of problems suitable for the algorithm.

1.9 Outline of Report

The purpose of this research is to establish modifications to DISOPE algorithm that can overcome the problem of slow convergence. The improvements suggested here can significantly reduce the number of iterations needed for convergence. A reduction on the number of iterations means that the number of oscillations is reduced, which in turn, increases the speed of the algorithm.

Chapter 2 gives literature reviews on published relevant works. We begin with reviewing several approaches to solve optimal control problems. Then we present the history of the original algorithm DISOPE. Next, we present the gradient descent algorithm and problems associated with it. Our journey brings us to an established area where the use of this algorithm is well documented, the back propagation algorithm of the neural networks. This algorithm is based on the gradient descent method. Then we review the documented methods done to the back propagation algorithm in order to overcome the problems instigated by the gradient descent part of the algorithm. Also in Chapter 2 we review the basic tools that are used in the convergence and stability analysis of the two new algorithms. These are the theory of multipass processes in the form of 2-D systems representations and limit profiles.

Chapter 3 is a compilation of works done on DISOPE. It describes the algorithm in details. An established convergence and stability analysis is reproduced here. A few numerical examples with varying degrees of complexity are included in this chapter. This chapter acts as a basis for comparison for subsequent chapters when the modifications have been made.

Chapter 4 comprises of further analyses of DISOPE. It deals on the complexity and convergence rate of DISOPE. Besides being a chapter reporting new additional analyses on DISOPE, this chapter also acts as basis for comparison for subsequent chapters.

Chapter 5 reports the first modification done to the updating mechanism of DISOPE. The updating mechanism is added terms of momentum from previous trial solutions, to help overcome the oscillation phenomenon. Two examples were simulated to see the effect the momentum has on the algorithm. The performance of this new algorithm, called DISOPE-MOMENTUM, is then compared to the performance of DISOPE by comparing results from the simulated examples to the examples in Chapter 3.

Chapter 6 reports the second modification done to the updating mechanism. In this chapter, the updating mechanism is altered to have two different updating schemes for two consecutive iterations. The first scheme is the same updating mechanisms of DISOPE retained. This scheme uses the gradient descent algorithm as its updating mechanism. The second scheme uses the parallel tangent algorithm for updating the trial solutions. This new algorithm is called DISOPE-PARTAN algorithm. This chapter ends with two simulated numerical examples. The results of which are again compared to the results of examples in Chapter 3.

Chapter 7 summarizes the work done in Chapters 2 to 8, compares the performance of all the three algorithms, and draws a conclusion on the findings. This chapter also provides suggestions for further research in this area.

1.10 Summary and Conclusion

DISOPE is a newly developed algorithm for solving nonlinear optimal control problem with inherent weaknesses imbedded in it. The weakness has been identified as the slow convergence of the algorithm when the search nears the optimal solution. The goal of this research is to overcome the weakness and make DISOPE more attractive to the end user.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Literature reviews of the subject matters of this research are given in this chapter. Some of the discussions in Chapter 1 are given elaboration here.

First, we review several other approaches in solving nonlinear optimal control problem. These are methods of quasilinearization (Sage and White, 1977), the methods of Hassan and Singh (1976), Teo *et al.* (1981), Shwartz (1996) and Neuro-Dynamic Programming by Bertsekas (2000).

Next, we discuss the history of the conception of DISOPE algorithm beginning with ISOPE algorithm. Next, we discuss the gradient descent algorithm, the basis for the updating mechanism of DISOPE. Here we review the problems relating to the gradient descent algorithms and the problems they caused DISOPE. Some proposed enhancements documented in the literature are related next.

Then we specifically go to the back propagation algorithm of the neural network that uses the gradient descent method as structure. Two improvements found in the literature, namely the inclusion of the momentum and the parallel tangent are looked at closely. These two are the chosen modifications intended for the updating mechanism of DISOPE in a bid to improve its converging behavior.

Next, we review the 2-D presentation of DISOPE in the form of the unit memory repetitive process. This treatment is needed in the theoretical analysis of the new modified algorithms. With the help of limit profiles, local stability of the new algorithms would be established in later chapters. This chapter ends with a summary of the discussions mentioned.

2.2 Other Approaches in Solving Optimal Control Problems

Before discussing the algorithm of concern in this research, we describe a number of other iterative procedures available for solving nonlinear continuous optimal control problems. The first approach is the continuous quasilinearization. It is a technique whereby a nonlinear, multipoint, boundary value problem is transformed into a more readily solvable linear, nonstationary boundary value problem. This technique involves the study of a sequence of vectors, which can be made to approximate the true solution of the nonlinear system (Sage and White, 1977).

The quasilinearization approach to the solution of non-linear dynamic optimization problems is based on solving the non-linear two-point boundary value problem iteratively as a series of linear two-point boundary value problem (Singh and Titli, 1978). These methods are attractive for several reasons. First it is often easier to guess nominal-state-variable histories than nominal-control-variables histories. Second, these methods converge rapidly near the optimum solution (Bryson and Ho, 1975).

One variation of quasilinearization involves choosing nominal functions for $x(t)$ and $\lambda(t)$ that satisfy as many of the boundary conditions as possible. The nominal control vector $u(t)$ is then determined by use of the optimality conditions. The system equations and the influence equations are linearized about the nominal and a succession of nonhomogenous, linear two-point boundary value problems are

solved to modify the solution until it satisfies the system and influence equations to the desired accuracy.

A second method is the method of Hassan and Singh (1976). In their work they developed a two level method for optimization of nonlinear dynamical systems with a quadratic cost function. This approach was based on the possibility to use equilibrium point of the system to expand the dynamic equations in a Taylor series and then fix the second and higher order terms by predicting the states and controls, which arise in these terms. This enables one to decompose the optimization problem into independent 'linear quadratic' sub-problems for given states and controls to be provided by a second level. On the second level a prediction principle type algorithm can be used. The algorithm has the advantage that only linear quadratic problems are solved at the first level and trivial updating is done at the second. There are substantial computational savings compared to the global single level solution, making the method suitable for solving low order nonlinear problems (Singh and Titli, 1978).

Teo *et al.* (1981) considered a class of convex optimal control problems involving a linear hereditary system with a bounded control region. The controls for negative time were treated as a given function rather than as controls. The algorithm was motivated by Barnes (Teo *et al*. 1981).

Schwartz (1996) developed methods that are based on solving a sequence of discrete-time optimal control problems using explicit, fixed step-size Runge-Kutta integration and finite-dimensional B-spline control parameterizations to discretize the optimal control under consideration. This is a group of programs and utilities named RIOTS as a software package. There are limitations to the type of problems that can be solved by the methods; they cannot solve problems with inequality state constraints that require a very high level of discretization. The methods also cannot handle problems with highly unstable nonlinear dynamics and they do not allow for delays in the systems dynamics.

Bertsekas (2000) developed a class of dynamic programming methods for control called the Neuro-Dynamic Programming. It is a relatively new class of methods that have the potential of dealing with problems that for a long time were thought to be intractable due to either a large state space or the lack of accurate model (Bertsekas, 2001). The algorithm combined ideas from the fields of neural networks, artificial intelligence, cognitive science, simulation, and approximation theory.

Neuro-Dynamic Programming is a class of reinforcement learning methods that deals with the curse of dimensionality of Dynamic Programming by using neural network-based approximations of the optimal cost-to-go function. The method has the advantage that it does not require an explicit system model; it uses a simulator, as a model substitute, in order to train the neural network architectures and to obtain sub optimal policies (Bertsekas, 2000).

2.3 Background of DISOPE Algorithm

The algorithm DISOPE on focus in this research is an extension of an earlier algorithm called *Integrated System Optimization and Parameter Estimation* (ISOPE). The underlying principle governing this technique is the consideration of the model-reality differences between real problems and their simplified models. It was originally developed by P.D. Roberts (1979), and Roberts and Williams (1981) for on-line steady state optimization of industrial processes implemented through adjustment of regulatory controller set points. It has been proved to be successful in solving many example problems (Roberts and William, 1981; Ellis and Roberts, 1981; Stevenson *et al.*, 1985; Brdyś *et al.* 1987). Later, Brdyś and Roberts (1987) derived sufficient conditions for convergence of this algorithm.

An essential feature of ISOPE is that the iterations converged to the correct real optimum in spite of the existence of model-reality differences. In order to produce a true optimum regardless of the differences, the interaction between

parameter estimation and model-based optimization is recognized by the algorithm. To match the results of the real and the simplified problems, ISOPE uses Lagrangian techniques to integrate the simplified optimization with parameter estimation. This is achieved by means of Lagrange modifiers introduced into the model-based optimization problem so that the interaction between system optimization and parameter estimation is compensated at the end of the algorithm iterations (Roberts, 1995). This approach was inspired by Haimes and Wismer (1972).

ISOPE was intended for the steady-state optimizing control. Naturally, it was later extended to solve the dynamical optimal control problems. Roberts (1994) extended ISOPE to dynamical problems and it has been termed DISOPE (dynamic ISOPE). The philosophy behind the techniques remains very much the same. As it was originally developed and published, DISOPE addressed continuous-time, unconstrained, centralized and time invariant optimal control problems (Becerra, 1994). Becerra (1994) advanced and improved the existing knowledge on the technique so as to make it attractive for its implementation in the process industry.

DISOPE was initially developed for continuous nonlinear optimal control problems (Roberts, 1993) and then extended to discrete systems (Becerra and Roberts, 1996) and to optimal tracking control problems (Becerra and Roberts, 1994). The technique has also been extended to cope with un-matched terminal constraints (Roberts and Becerra, 1998). A range of applications of DISOPE techniques has also been developed by Becerra, (1994).

The stability and convergence analyses of this algorithm have also been offered by Roberts (1994a, 1994b, 1999, 2000b, 2000c, 2002). The algorithm is considered as a 2-D system and the convergence and stability analyses are based on the multipass process theory in the form of a unit memory repetitive process.

The concern of this research is in the convergence behavior of DISOPE algorithm. This behavior is partly controlled by its updating mechanism. The mechanism is an integral part of DISOPE, however, its importance has been largely

ignored up to now. This research weights heavily on this mechanism and the modifications are tailored specifically for it.

As mentioned in Chapter 1, the updating mechanism of the algorithm is recognized as the gradient descent method and as explained in the same chapter, gradient descent algorithms come with two well-known setbacks; the slow convergence of the algorithm caused by oscillations of the search and the convergence to false optima (Pierre, 1969; Ochiai *et al.*, 1994; Shang, 1997; Qiu *et al.*, 1992; Fukuoka *et al.*, 1998; Qian, 1999). In the next section, we discuss the gradient method and its maladies further.

2.4 Gradient Descent Algorithm

The method of gradient descent is one of the most fundamental procedures for minimizing a differentiable function of several variables, *f*. Common to all gradient search techniques is the use of the gradient $\nabla f = g = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right]^T$. All gradient methods are governed, at least in part, by the following equation:

$$x^{i+1} = x^i - \eta g\,|_{x=x^i} \tag{2.1}$$

where $g = \nabla f$ is the gradient of *f* in column-vector form and η is the step size parameter to be estimated. Gradients methods differ in the way in which η is selected at $x = x^i$ (Pierre, 1969).

In general, the gradient algorithm takes a point $x^i \in \hat{S} \subset E^n$ and computes a new point $x^{i+1} \in \hat{S} \subset E^n$ where $\hat{S}$ represents an arbitrary set and *E* represents the Euclidean space. The new point is defined by Eq. (2.1) where $\eta > 0$ is taken for minimization or $\eta < 0$ for maximization problem. Further, point x^i is the origin of the line, and *g* determines the direction.

Problems faced by gradient descent methods have been outlined in Chapter 1. They are the oscillations of the search and false optima. The two phenomena are discussed in detail in what follows.

The surfaces of nonlinear objective functions are terrains with both hills and valleys and flat basins as depicted in Fig. 2.1 (Shang, 1997). These terrains pose challenges to any search algorithm because of the differing slopes they offered. Steep slopes of tall hills made them difficult to overcome. Hence once a search get stuck in a local minimum, it would be difficult for the search to get out of it and continue to search for the global minimum. Large shallow basins on the other hand, provide little information for search direction, and may take a long time for a search algorithm to pass these regions if the step-size is inappropriately small (Shang, 1997).

Gradient descent methods are classified as local optimization methods (Shang, 1997). They do not have the ability that guarantees the solutions found are global optima. Thus without a doubt, the basic gradient methods are relatively inefficient when ridges or ravines are salient (Pierre, 1969; Qiu *et al.*, 1992; Ochiai *et al.*, 1994) where most optima reside, be it local or global.

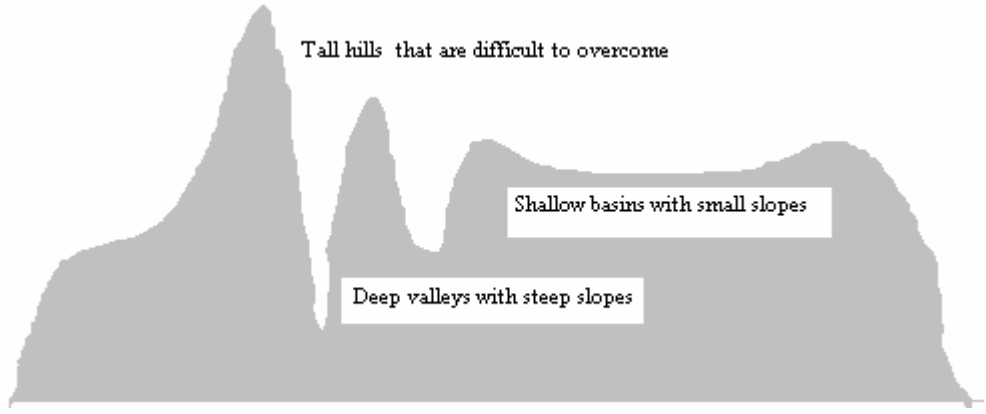


Figure 2.1. The different terrains of the surface of a nonlinear function.

Fig. 2.2 illustrates the locus of the search in the ravine region. As illustrated in the figure, the direction of the gradient is almost perpendicular to the long axis of the valley. At each iteration the locus jumps over the bottom of the ravine. The

search oscillates back and forth in the direction of the short axis, and only moves very slowly along the long axis of the valley (Rumelhart *et al.*, 1986a; Qian, 1999) towards the optimal solution. These oscillations make travel time longer and consequently, the slow convergence.

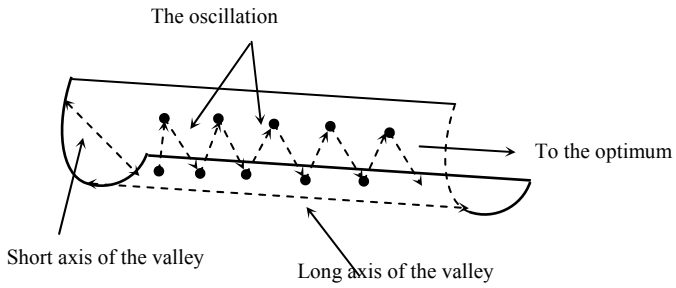


Figure 2.2: The oscillation phenomenon.

Because of this, gradient methods usually show great improvements in the first few iterations but tend to advance slowly as the optimal solution is approached. To further illustrate this phenomenon, consider an objective function with concentric ellipsoidal contours (Ghorbani and Bayat, 2000). If the initial point for a gradient search happens to be precisely on one of the axes of the systems of ellipses, the gradient line will pass right through the optimum and the search will be over in one descent. Otherwise, the search will follow a zigzag course from p_0 to p_1 to p_2 etc. The following Fig. 2.3 illustrates the situation.

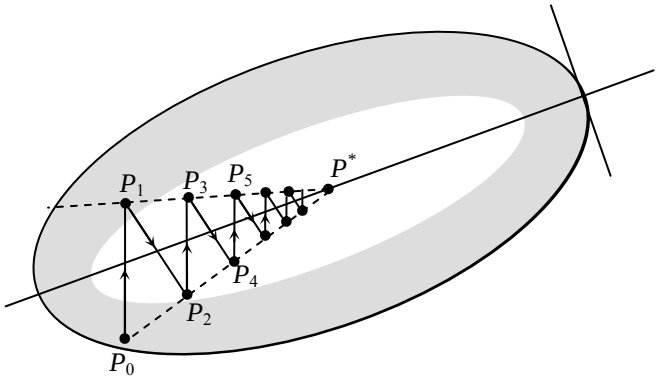


Figure 2.3: The zigzagging phenomenon.

Furthermore, gradient methods are slow to converge when the surface forms a plane or basin with a gentle slope (Fig. 2.1). This results in too small a gradient to move rapidly over the wide flat surface (Ochiai et al, 1994; Qiu et al, 1992). If one such area is encountered, no significant decrease in the error between two consecutive terms occurs for some period of time, hence the slow convergence.

This phenomenon might even be mistakenly interpreted as convergence if the error is too insignificant that it is less than the tolerance specified for the algorithm to terminate. If the algorithm terminates, the point of termination is a false optimum. This phenomenon is called '*premature saturation*' (Fukuoka et al, 1998). However, if the algorithm does go on and the minimizer is still far away, the error will eventually decrease again. In the end, the algorithm will slowly arrive at the true optimum.

Since gradient methods are local optimizers, the possibility that their solutions are local optima is inevitable. Ravines on the surface in all probability contain local minima, hence a local search procedure that enters such a region, will be directed towards that minimum and stop when it reaches it, while it would be desired that it continue towards a global minimum. This is another instant where the gradient descent methods might be giving false optimum as solutions.

In short, the gradient descent method has the problem of slow convergence that might happen in the vicinities of ravines, ridges, and large flat areas of basins. A secondary problem to these terrains is convergence to local or false optima. Fig. 2.4 summarizes these problems. The DISOPE algorithm being a search method of the gradient descent type, is indisputably susceptible to the same problems. Our aim is to propose methods that can overcome the difficulties.

One of the most prolific algorithms that are based on the gradient descent method is the back propagation algorithm of the neural networks. The area of research that is aimed at improving the convergence speed of the back propagation is enormous with numerous methods proposed to do just that. We take advantage of

this vast volume of knowledge to improve the convergence behavior of DISOPE based on the improvements done to the back propagation algorithm.

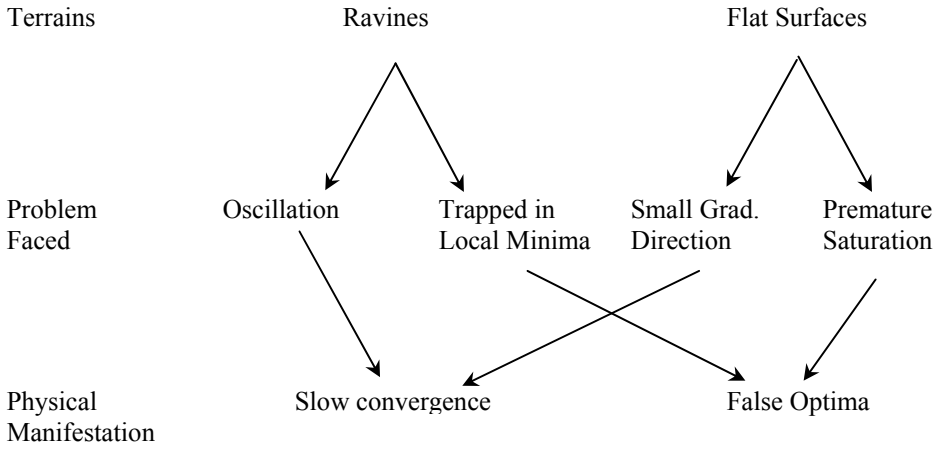


Figure 2.4: The schematics of the problems faced by gradient descent algorithms.

2.5 Back Propagation Algorithm

The back propagation algorithm of Rumelhart and McClelland (1986b) is an iterative gradient descent algorithm designed to train multi-layer feed forward networks of sigmoid nodes by minimizing the mean square error between the actual output of the network and the desired output.

Before discussing the back propagation algorithm in detail, some basics of the neural networks are in order. We begin with the description of a perceptron.

2.5.1 A Perceptron

A perceptron is a connected network. The basic perceptron is composed of an input layer and an output layer of nodes as in Fig. 2.5. Each input node is

connected to every output node and vice versa, but there is no connection between nodes in the same layer. Assigned to each connection is a weight. When the first layer sends a signal to the second layer the associated weights are applied on the inputs. The receiving nodes then sum up the incoming values. If the sum exceeds a predetermined threshold, the receiving nodes will fire output signals.

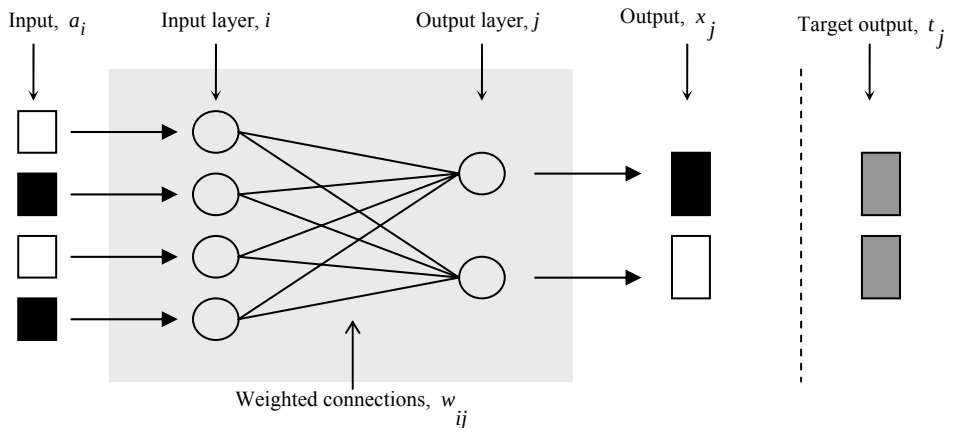


Figure 2.5: A single layer perceptron.

The input of a node i is represented by a_i. The input of a node j, S_j, is the sum of all the weighted inputs from node i. The output of a node j is determined by the following rule.

$$\left.\begin{aligned} &S_j = \sum_{i=0}^{n} a_i \mathbf{w}_{ij} \\ &\text{If } S_j > \theta \text{ then } x_j = 1 \\ &\text{If } S_j \le \theta \text{ then } x_j = 0 \end{aligned}\right\} \qquad (2.2)$$

where **w** is a vector representing all the weights between nodes i and j, n is the number of nodes in the input layer, and θ is a predetermined threshold value.

The weights can be adjusted so that the network produces a desired output given a set of inputs. The adjusting of weights to produce a particular output is called the training of the network. It is a mechanism that allows the network to learn. The training is accomplished by comparing the actual output x_j of the network with a

set of target outputs t_j. If there is a difference between the two, the weights are adjusted to produce a set of outputs closer to the target values. New weights are determined by adding an error correction value to the old weights. The amount of correction is determined by a multiple of the difference between the actual and the target outputs. The multiplier is a constant called a learning rate. The calculation of the new weights can be summed up as follows.

$$\mathbf{w}_{ij}(\text{new}) = \mathbf{w}_{ij}(\text{old}) + C_N(t_j - x_j)a_i \qquad (2.3)$$

where C_N is the learning rate. The training procedure is repeated until the performance no longer improves, theoretically when $t_j = x_j$.

2.5.2 Multilayer Perceptron

A multilayer perceptron network is a net with one or more layers of nodes between the input and the output units. These extra layers are called the hidden layers. The outputs of one layer are fed-forward as inputs to the next layer. Fig. 2.6 illustrates the architecture of a simple three-layer perceptron. The multilayer perceptron can solve more complicated problems compared to the single layer perceptron although training may be difficult. The multilayer perceptrons are typically trained using a supervised training algorithm known as the back propagation algorithm.

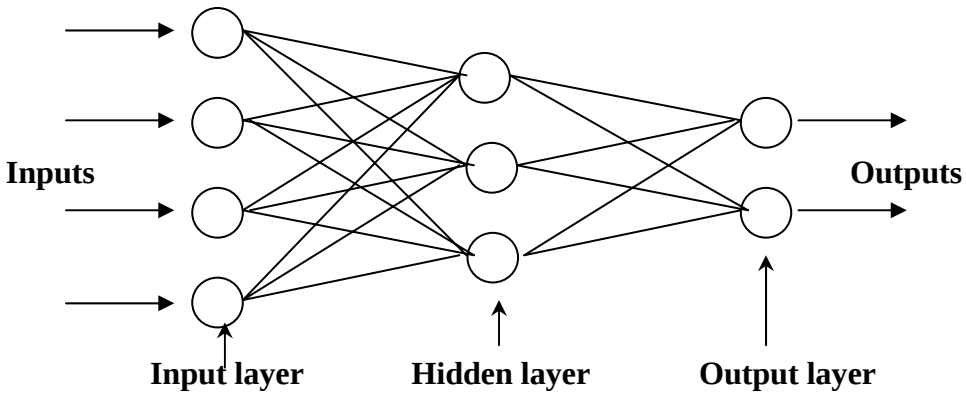


Figure 2.6: A simple three-layer perceptron.

2.5.3 Representation of the Back Propagation Algorithm

As mentioned earlier, the back propagation algorithm is a gradient descent method aimed at minimizing the total squared error of the output computed by the net. The error function is taken to be the least squares error function below.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{p}(t_j - x_j)^2 \tag{2.4}$$

where *p* is the number of nodes in the output layer. While in training, the weights in each iteration are changed according to the following rule (Rumelhart and McClelland, 1986).

$$\Delta \mathbf{w}_{ij} = -\eta_m \frac{\partial E}{\partial \mathbf{w}_{ij}} \tag{2.5}$$

From (2.3) and (2.5) the back propagation algorithm is defined as

$$\mathbf{w}_{ij}(n+1) = \mathbf{w}_{ij}(n) - \eta_m \frac{\partial E}{\partial \mathbf{w}_{ij}} \tag{2.6}$$

where η_m is a small positive number known as the learning rate and *n* is the iteration index.

The standard back propagation algorithm inherits the problems of the gradient descent methods. It shows very slow convergence (Moreira and Fiesler, 1995; Fukuoka *et al.*, 1998) and has the tendency to converge to a false local minimum (Fukuoka *et al.*, 1998). Over the years, many acceleration techniques have been developed to speed up the convergence of this algorithm. In the next section, we review some of the better-known approaches that have been documented.

2.5.4 Approaches to Overcome the Slow Convergence

The most popular learning acceleration approach is the inclusion of the momentum term introduced by Rumelhart and McClelland (1986b). With this approach, the weight changes in a direction that is a combination of the current gradient and the previous gradient. In order to use momentum, weights from

previous training patterns must be saved. Convergence is faster when a momentum term is added to the weight update formula with appropriate learning rate.

There are methods that are able to generate noninterfering directions and can be used to overcome the difficulties of oscillations, by deflecting the gradient. Rather than moving along $-\nabla f(x)$, for example, one can move along $-H\nabla f(x)$ or along $-\nabla f(x)+v$, where H and v are appropriate matrix and vector respectively (Ghorbani and Bhavsar, 1993). The method of Newton uses the first one and deflects the gradient descent direction by premultiplying it by the inverse of the Hessian matrix. However, the calculation of Hessian matrix is a complex, error-prone and an expensive task.

Amongst the methods with the aim of improving the learning capability are from the field of optimal filtering, the extended Kalman algorithm (Singhal &Wu, 1989). From the field of numerical analysis, the second order and line-search (Becker & le Cun, 1988), the conjugate gradient (Kramer & Sangiovanni-Vincentelli, 1988), and quasi-Newton (Watrous, (1987) methods. Conjugate gradient techniques are known to be the most effective minimization methods that use only the first derivative information (Polak, 1997). They basically compute the new search direction by using the gradient direction and the previous search direction. Their advantage over the basic gradient descent algorithm is a faster convergence rate near an optimum point (Ghorbani and Bhavsar, 1993). However the conjugate method requires more storage of intermediate results than the momentum method. It is also less robust than the momentum method when the error surface is relatively flat (Qian, 1999).

In some cases it is more advantageous to accumulate the weight correction terms for several patterns and make a single weight adjustments equal to the average of the weight correction terms, for each weight rather than updating the weights after each pattern is presented. This procedure is called batch updating has a smoothing effect on the correction terms. In some cases, however, this smoothing may increase the chances of convergence to a local minimum.

Adaptive learning rate is one class of techniques that was found to be successful in overcoming the difficulties. Basically, these methods improve the speed of training by changing the learning rate during training. The amount of weight update can be allowed to adapt to the shape of the error surface at each particular situation. This approach eliminates the trial and error search pattern for the best values of the learning rate parameters.

Some adjustable learning rate algorithms are designed for specific problems, such as classification problems in which there are significantly fewer training patterns from some classes than from others. If the traditional approach, duplication or creating noisy copies of the training patterns from the underrepresented classes, is not practical, the learning rate may be increased when training patterns from the underrepresented classes are presented. (DeRouin *et al.*, 1991; Moreira and Fiesler, 1995).

Another type of adjustable learning rate is based on the determination of the maximum safe step size at each stage of training (Weir, 1991). It provides protection against the overshoot of the minimum error that can occur in other form of back propagation. This algorithm however requires additional computations of gradients that are not calculated in standard back propagation.

Another class of adjustable learning rate algorithm involves risk taking. Algorithm of this type have been developed by many researcher, among them Cater (1987), Fahlman (1988), and Silva and Almeida (1990). Heuristic optimization techniques perform a search in the weight space as an alternative to the back propagation. The method of delta-bar-delta (Jacobs, 1988), and the quickprop (Fahlman, 1988) are examples of this kind of algorithm.

Other techniques include the global learning rate adaptation with different variations, where proper values for learning rates and momentum factors are chosen to optimize the algorithm (Fausett, 1994). Local learning rates adaptations on the other hand are techniques wherein independent learning rates are used for every

connection and optimal learning rates for every weight is found. One variation is to set up schedule to change the step length as the network is learning (Jacobs, 1988).

Yu and Chen (1997) suggested back-propagation learning by using simultaneously the optimized learning rate and the momentum factor. Qiu *et al.* (1992), suggested accelerating the training of back propagation networks by using adaptive selection of momentum values. The authors used an optimization technique based on the parallel tangents methods to compute the values of the momentum parameter.

Mohd (1996) proposed the use of interval arithmetic together with the methods of Moore, Hansen or Alefeld to determine the optimal value of the learning rate. Once the optimal learning rate is found it is then applied to the gradient search to find the solution of the optimization problem.

Van Ooyen and Nienhuis (1992) proposed a new error function to be used instead of the usual least squares error function. They proposed a function in the form of the squares of the differences between the actual and target values, summed over the output units and all cases.

Kamarthi and Pittner (1999) proposed a universal acceleration technique for the back propagation algorithm based on extrapolation of each individual interconnection weight. The procedure is activated a few times in between iterations of the conventional back propagation. It minimally alters the computational structure of the back propagation algorithm.

Among these multitudes of learning algorithms, back propagation with momentum acceleration remains one of the most popular learning paradigms, mainly because of its faster convergence than the back propagation method in a variety of problem and because of its computational simplicity. The incorporation of momentum in the back propagation algorithm has been extensively studied, especially from an experimental point of view (Fahlman, 1988; Tesauro and

Janssens, 1988; Jacobs, 1988; Tollenaere, 1990). Sato (1991) and Hagiwara and Sato (1995) provided some theoretical backgrounds. Perantonis and Karras (1995) establish a link between the use of momentum in multilayer feedforward neural networks learning and constrained optimization learning techniques.

The momentum also reduces the likelihood that the net will find weights that are a local, but not global, minimum (Fausett, 1994). When using momentum, the net is proceeding not in the direction of the gradient but in the direction of a combination of the current gradient and the previous direction of weight correction. Qian (1999) established that the momentum method is stable in the continuous time case.

Another technique that is known for its simplicity is the parallel tangent (PARTAN) method. It is another technique reported to be able to overcome the difficulties of oscillation. It uses deflecting gradient technique and may be considered as a special case of the conjugate gradient method (Ghorbani and Bhavsar, 1993). It comprises of two phases namely climbing through gradient and accelerating through parallel tangent. PARTAN overcomes the inefficiency of zigzagging in the conventional back-propagation learning algorithm by deflecting the gradient through acceleration phase. Regardless of the degree of the complexity of the problem used, the PARTAN back propagation algorithm shows at least two times faster convergence to the solution than the conventional back propagation alone (Ghorbani and Bayat, 2000). The gradient PARTAN algorithm is also a global error adaptation technique (Ghorbani and Bayat, 2000); hence besides overcoming the oscillation problem, it is also capable of overcoming the problem of convergence to local minima.

The list of techniques mentioned above is by no means exhaustive. From all the techniques mention above, we settled on the techniques of momentum and PARTAN for the updating mechanism of DISOPE. There are two reasons for choosing these techniques. One is the techniques need no new information; they use the readily available information from DISOPE. The other is based on the

effectiveness and simplicity of the methods. In the following two sections we detailed the methods.

2.6 Momentum Term

The standard back propagation algorithm has been known to show very slow rate of convergence (Moreira and Fiesler, 1995). Furthermore, the shape of multidimensional error function for the majority of the applications usually presents irregularities. As mentioned these irregularities could be in the forms of convergence to local minimum and false minimum. It was discovered, however, that the appropriate manipulation of the learning rate during the training process could lead to very good results.

Over the years, many new acceleration techniques have been developed to speed up the convergence in this algorithm (Kamarthi and Pittner, 1999; Ochiai et al, 1994; Solomon and van Hemmen, 1996). The most popular of these strategies is to include a momentum term in the weight-updating phase (Baldi, 1995). A momentum term added to the original back propagation has been widely used because of its simplicity and effectiveness (Hagiwara, 1995). This fact has been taken into consideration in improving the updating procedure of DISOPE.

The inclusion of the momentum term transforms Eq. (2.6) into the following:

$$\mathbf{w}_{ij}(n+1) = \mathbf{w}_{ij}(n) - \eta^m \frac{\partial E}{\partial \mathbf{w}_{ij}} + \varpi(\mathbf{w}_{ij}(n) - \mathbf{w}_{ij}(n-1)) \qquad (2.7)$$

where ϖ is the momentum parameter. That is the modification of the weight vector at the current time step depends on both the current gradient and the weight change of the previous step (Qian, 1999). The rationale for using the momentum term is that it helps average out the oscillation along the short axis (Rumelhart *et al*., 1986a). The values of ϖ are constrained to be in the range of (0,1] (Fausett, 1994). It is desirable to use small learning rates to avoid major disruptions of the direction of

learning. Fig. 2.7 below explains the effect of the momentum term on the ravine phenomenon explained in Section 2.4.

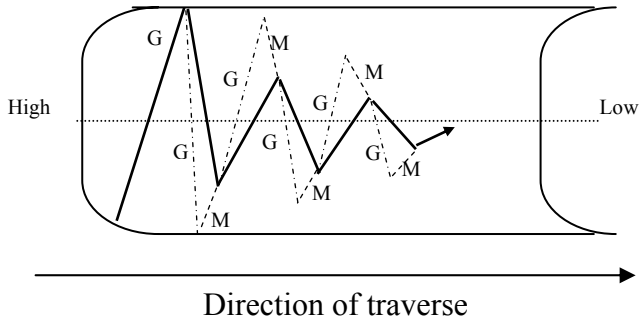


Figure 2.7: The locus of weights with the momentum terms.

Fig. 2.7 shows the locus of weights in the ravine region with the addition of the momentum term (Rohanin *et. al*, 2002; Rohanin and Mohd_Ismail, 2003d). G stands for the gradient descent direction and M stands for the deflection caused by the momentum term. The solid zigzagging line is the combined effect of both G and M. The momentum term works at reducing the oscillation when the weights jump over the bottom of the ravine. Note that after the weight crosses the ravine, the momentum vector corrected the steepest descent vector by deflecting it further down the line. As a result, the direction of the amassed vector is in the downward position.

The deflection by the momentum could also remedy the problem caused by plateaus. The inclusion of the momentum could increase the stride size made by the algorithm hence the error between two consecutive steps is made bigger and the time taken to traverse such area is reduced. Hence the momentum is not only capable of increasing the speed of convergence dramatically (Qian, 1999; Qiu et al, 1992), but it also is capable of avoiding false minima and reducing the likelihood that the procedure will find solutions that are local, not global, minimum (Fausett, 1994).

2.7 Parallel Tangent

Pierre (1969) reported that in 1951, Forsythe and Motzkin advanced a procedure, called acceleration-step search, for expediting the rate of convergence of the best-step version of the steepest ascent for a function of two variables $x = \{x_1, x_2\}$. Surprisingly, not only using the acceleration-step search reduces the number of required iterations in general, but also the gradient need not be computed at the start of each search. If $f(x_1, x_2)$ is a quadratic function with a well-defined minimum, the procedure requires the use of three searches to locate the optimum exactly; thus, the procedure is quadratic convergent.

One version of the method is as follows. Starting at an initial point $x = p_0$, the gradient $g(p_0)$ is evaluated, and a search for a minimum is conducted in the negative direction of the gradient. At the winning point p_1, the gradient $g(p_1)$ is evaluated and a second search is conducted, as before. The winner of this second search is designated p_2. At this stage of the process, $g(p_2)$ is not computed, rather, a search is conducted along the straight line which connects the initial point p_0 and the point p_2. This acceleration-step search is conducted along the line $x = \beta_s p_2 + (1 - \beta_s) p_0$ where β is the search parameter. If $f(x_1, x_2)$ is quadratic, this third search results in the optimal value $f(x_1, x_2)$ as illustrated in Fig. 2.8. In the case where $f(x_1, x_2)$ is not quadratic, the acceleration-step search requires use of additional iterations.

In the *n*-dimensional case, the acceleration step discussed above is known as the methods of parallel tangent (PARTAN). The generation of search generated by the procedure is "conjugate directions" with which the minimum of a quadratic can be located in a finite number of iterations (Pierre, 1969).

The name PARTAN has no significance as far as the mechanics of the search procedure are concerned; however, the name has an interesting geometrical origin,

which is shown in the two-dimensional case of Fig. 2.8. The line labeled l_0 is tangent to an equimagnitude contour of $f(x)$ at the original search point p_0; the line labeled l_2 is tangent to an equimagnitude contour at the search point p_2; and the line labeled l_1 is perpendicular to both l_0 and l_2; thus, lines l_0 and l_2 are *parallel tangents*. Note that the acceleration step from p_0 through p_2 to p_3 is taken through the two points p_0 and p_2 at which the two parallel lines l_0 and l_2 are tangent to equimagnitude contours. This feature is common to all PARTAN methods.

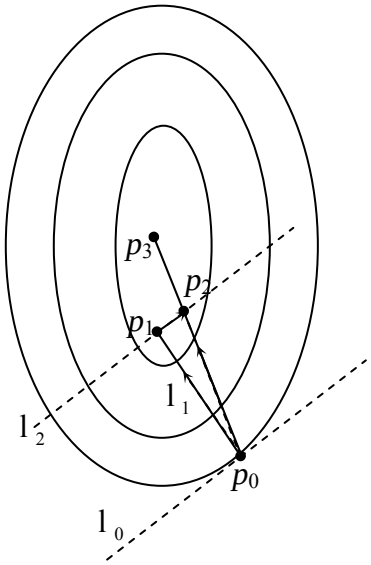


Figure 2.8. Locus of the search for a quadratic function.

The strong point common to all PARTAN methods, is that the acceleration step from p_0 through p_2 to p_3 is taken through the two points p_0 and p_2 at which the two parallel lines l_0 and l_2 are tangent to the equimagnitude contours. This feature enables us to traverse swiftly along *straight* and *narrow* ravines (ridges). To see this consider any two lines in the x_1x_2 plane which are parallel and which intersect a straight ravine of $f(x_1, x_2)$ (Fig. 2.9). Observed that the point of tangency defines a line, which parallels the ravine. Hence, by searching along the parallel ravine-line, we effectively follow the ridge (Pierre, 1969). For curved ravines, PARTAN search is not quite so efficient, but it invariably much better than gradient search alone (Pierre, 1969).

The general procedure for PARTAN search is as follow:

1. Starting at the initial point p_0, search in the direction defined by $\nabla f(p_0)$ until extremal point p_1 is found;
2. Search for the extremal point p_2 which lies along the line defined by p_1 and $\nabla f(p_1)$;
3. Search for the extremal point p_3 which corresponds to the optimum of $f\left[p_2 + \beta_s(p_2 - p_0)\right]$ with respect to β_s;
4. Alternate between gradient search and acceleration steps.

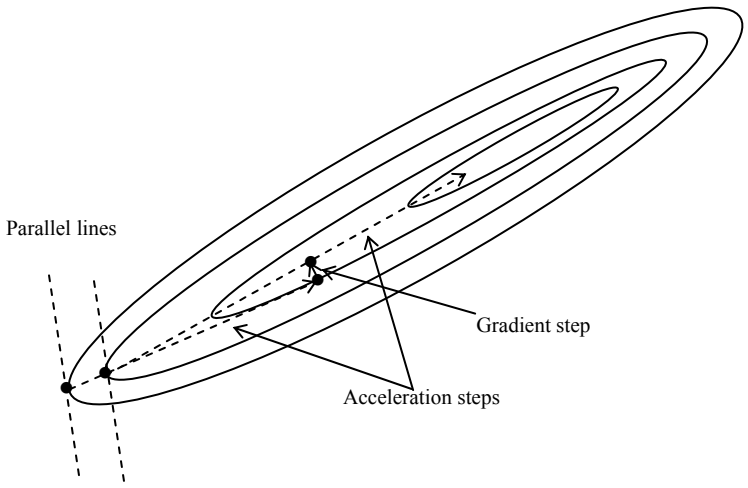


Figure 2.9. The points of tangency of the two parallel lines define a line that parallels the ravine.

The above procedure is called the *gradient*-PARTAN search. The gradient descent searches are used to find $p_1, p_2, p_4, p_6, \ldots$ and acceleration steps are used to locate $p_3, p_5, p_7, p_9, \ldots$. With PARTAN, the acceleration steps are conducted through the following pairs of points:

(p_0, p_2), (p_1, p_4), (p_3, p_6), … ,(p_{2k-3}, p_{2k}), … .

The locus of the gradient-PARTAN search would look as depicted in Fig. 2.10 below. In this figure G stands for the gradient direction and P stands for the acceleration by the PARTAN step. If $f(x)$ is quadratic with a well-defined

optimum, the exact optimal point is located after 2*n*-1 searches, except for round off error.

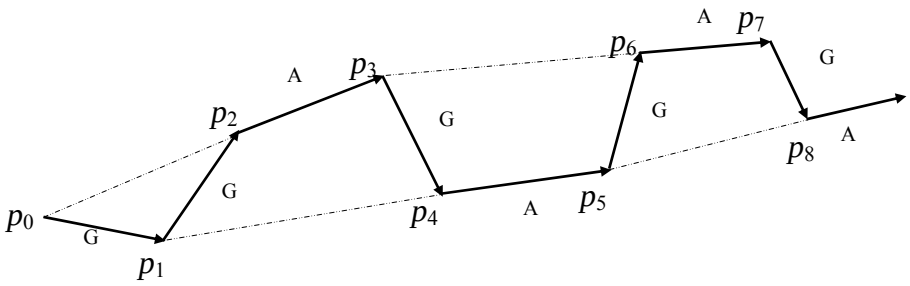


Figure 2.10: The path taken by the gradient-PARTAN.

Although conjugate gradient methods and second-order minimization methods converge faster than gradient-PARTAN method, the simplicity and ravine following properties of gradient-PARTAN make it attractive. The -PARTAN technique combines many desirable properties of the simple gradient method. It has many forms and gradient PARTAN is one form, which amounts to a multidimensional extension of the accelerated gradient method (Ghorbani and Bayat, 2000). This technique represents a distinct improvement over the method of steepest descent. It is an alternative method of improvement to that of the momentum term in the back propagation algorithm.

Two new algorithms are proposed in this research.

2.8 Multipass Processes

The term 'multipass process' is one which is used to describe a class of systems that possess two distinct properties (Edwards and Owens, 1982):

(i) Repetitive operation and

(ii) Interaction between the state-and/or-output functions generated during successive cycles of operation.

Each individual cycle is termed ‘pass’ and through the repetition of these passes as time proceeds we get the ‘multipass process’ term.

The modeling aspect of the process originates from the work of Edwards in the late 60’s and early 70’s from his work on the longwall coal-cutting machine (Edwards and Owens, 1982). The common feature is shown to be the dependence of present-pass behavior on the behavior of the process produced on one or more previous passes. The property is referred to as the *interpass interaction.* The variables generated in the course of pass *i* of the process, not at only particular points but along the entire length of the process, affect the outputs of process *i* +1.

A multipass process shares a number of characteristics in common with conventional linear dynamic processes (Edwards and Owens, 1982). Thus, a general model for a linear multipass process would be:

$$\left\{\begin{array}{l} \dfrac{d}{dt}x^i(t) = A_{mp}x^i(t) + B_{mp}d^i(t) + C_{mp}u^i(t) \\ y^i(t) = D_{mp}x^i(t) + F_{mp}d^i(t) + J_{mp}u^i(t) \end{array}\right\} \qquad (2.8)$$

where $A_{mp}, B_{mp}, C_{mp}, D_{mp}, F_{mp},$ and J_{mp} are constant coefficient matrices and $0 \le t \le \varsigma$, is the pass length. In Eq. (2.8), the vector $y^i(t)$ is the general representation of the process output, $x^i(t)$ is the state vector and $u^i(t)$ is the control vector. A vector $d^i(t)$ is required to denote the interpass disturbance variables in general since more than a single variable may produce the interaction. It is necessary to use two coordinates to specify a variable in a repetitive process. These two coordinates are the pass number and the position along this pass. Hence in Eq. (2.8) i denotes the pass number and *t* measures the distance along each pass from its starting point.

2.8.1 Link to 2-D Systems

Repetitive processes have strong structural links with two dimensional, or 2-D, systems (Edwards and Owens, 1982). These systems propagate information in two separate directions that can be considered as the two distinct dimensions. From

this standpoint, multipass processes fall naturally into the area of the 2-D systems with i, being one dimension and t, the other.

This link is useful in the sense that the 2-D systems theory can be used to analyze the local stability and convergence behavior of multipass processes. Roberts (1994a) used linear 2-D system theory techniques to analyze local stability and convergence behavior of DISOPE. The 2-D analysis is based on the theory of unit memory repetitive processes developed principally by Edwards and Owens (1982).

2.8.2 Abstract Model of Multipass Processes

Multipass processes have a number of common features that distinguish them from the more familiar dynamic processes. These are (Edwards and Owens, 1982):

(i) A number of passes through a known set of dynamics

(ii) Each pass is characterized by a pass length $\varsigma_i > 0$ and a pass profile $y^i(t)$ defined on $0 \le t \le \varsigma_i$. The pass profile need not be a scalar quantity.

(iii) An initial pass profile $y^0(t)$ defined on $0 \le t \le \varsigma_0$, where ς_0 is the initial pass length. The function y^0 plays the role of an initial condition for the process.

(iv) Each pass will be subject to its own boundary conditions, disturbance inputs and control inputs.

(v) The process has a unit memory property, i.e. the dynamics of pass *i* depends only upon the independent inputs to that pass and the pass profile on pass $i-1$.

To obtain an abstract setting for the consideration of multipass stability theory we can regard the pass profile $y^i(t)$ on pass *i* as a point in a suitably chosen function space. More precisely, the pass profile y^i is regarded as a point in a Banach space E^i_ς, i.e.

$$y^i \in E^i_\varsigma, \quad i \ge 0 \tag{2.9}$$

A general abstract model of multipass systems dynamics then has the structure of a recursion relation

$$y^{i+1} = f^{i+1}(y^i), \quad i \geq 0 \tag{2.10}$$

where f^i is an abstract mapping of E_ς^i into E_ς^{i+1}. Eq. (2.10) is in the form of a unit memory repetitive process.

The unit memory property assumed in the above discussion is not as restrictive as might initially appear to be. It can be extended to describe multipass systems with any finite length memory property (Edwards and Owen, 1982) where the dynamics of pass *i* depends only upon the independent inputs to that pass and the pass profiles on pass $i-1, i-2, ..., i-M$. Formally

$$y^{i+1} = f^{i+1}(y^i, y^{i-1}, ..., y^{i+1-M}), \quad i \geq 0 \tag{2.11}$$

2.8.3 An Abstract Model of the Linear Multipass Process of Constant Pass Length in the Form of a Unit Memory Repetitive Process

An analysis of the general abstract model given above is out of the scope of this research. However, in order to gain insight into the multipass processes and coincidental with the analysis being done in this research, the discussion is restricted to the processes having linearity properties and a constant pass length. The following definition from Edwards and Owens (1982) is of a general nature and describes many processes of physical interest.

Definition 2.1

A linear unit memory repetitive process $S(E_\varsigma, W_\varsigma, L_\varsigma)$ of constant pass length $\varsigma > 0$ consists of a Banach space E_ς, a linear subspace W_ς and a bounded linear operator L_ς of E_ς into itself. The systems dynamics are described by linear recursion relations of the form

$$y^{i+1} = L_\varsigma y^i + b^{i+1}, i \geq 0 \tag{2.12}$$

where $y^i \in E_\varsigma$ is the pass profile of pass *i* and $b^{i+1} \in W_\varsigma, i \geq 0$. The term $L_\varsigma y^i$ represents the contribution from the *i*th pass to the *i*+1th pass profile and b^{i+1} represents the initial conditions, disturbances and control input effects.

To say that L_ς is linear implies that

$$L_\varsigma(\lambda_1 x_1 + \lambda_2 x_2) = \lambda_1 L_\varsigma x_1 + \lambda_2 L_\varsigma x_2 \tag{2.13}$$

for all $x_1, x_2 \in E_\varsigma$ and scalars λ_1, λ_2. If L_ς is bounded then its norm

$$\left\| L_\varsigma \right\|_\varsigma \ @\sup_{\|y\| \leq 1} \frac{\left\| L_\varsigma y \right\|_\varsigma}{\left\| y \right\|_\varsigma} < +\infty$$

where $\|\cdot\|_\varsigma$ denotes both the norm on E_ς and the induced operator norm.

A particular example is the following unit memory linear repetitive process without disturbances and control inputs.

$$\left.\begin{aligned} &\frac{d}{dt} X^i(t) = A_o X^i(t) + B_o Y^i(t);\ X^i(0) = d^i \\ &Y^i(t) = C X^i(t) + D_1 Y^i(t);\ t \in [0, T] \end{aligned}\right\} \tag{2.14}$$

where the pass length $\varsigma = T$; the pass profile Y^i is $Y^i(t)$; A_o, B_o, C and D_1 are constant matrices of appropriate dimensions, and d^i is a vector of initial conditions which can change from pass to pass.

2.8.4 Properties of the Linear Unit Memory Repetitive Processes

Two important properties of linear repetitive processes are stability and the limit profile (Rogers and Owens, 1992). In the following discussion, the definitions of asymptotically stable and limit profile are given as in Edwards and Owens (1982). Two related theorems are stated without proof for use in later chapters.

Definition 2.2 - Asymptotically Stable

A linear multi-pass process $S(E_\varsigma, W_\varsigma, L_\varsigma)$ of constant pass length $\varsigma > 0$ is said to be asymptotically stable if there exists a real scalar $\delta > 0$, such that, given any initial profile y_0 and any strongly convergent (i.e. convergent in norm) disturbance sequence $\{b^i\}_{i\geq 1}$, the sequence $\{y^i\}_{i\geq 1}$ generated by the perturbed process

$$y^{i+1} = (L_\varsigma + \gamma) y^i + b^{i+1}, \quad i \geq 0 \tag{2.15}$$

where γ is the model perturbation, converges strongly to a 'limit profile' $y^\infty \in E_\varsigma$ whenever $\|\gamma\|_\varsigma \leq \delta$.

A sufficient and necessary condition for asymptotic stability is provided by the following theorem: (see Edwards and Owen, (1982) for the proof).

Theorem 2.1

The linear repetitive process defined by Eq. (2.12) of constant path length $\varsigma > 0$ is asymptotically stable if and only if the spectral radius $r(L_\varsigma) < 1$.

Application of this theorem to the unit memory linear repetitive process described by Eq. (2.14), for the particular case where the initial condition d^i is independent of Y^i, provides the interesting result, that asymptotic stability then is achieved if and only if all eigenvalues of the matrix D_1 lie in the open unit circle in the complex plane (Rogers and Owens, 1992). However, this simple result does not apply when d^i is dependent on Y^i when it will be found that asymptotic stability also depends upon the matrices $A_o, B_o,$ and C.

Definition 2.3 - Limit Profile

If the linear multi-pass process of Eq. (2.12) of constant pass length $\varsigma > 0$ is asymptotically stable and $\{b^i\}_{i\geq 1}$ is a disturbance sequence that converges strongly to a disturbance b^∞ then the strong limit

$$y^{\infty} \; @\lim_{i\to\infty} y^{i} \tag{2.16}$$

of the pass profiles is termed the *limit profile* corresponding to the disturbance sequence $\{b^1, b^2, b^3, ...\}$.

Theorem 2.2

If the linear multi-pass process $S(E_{\varsigma}, W_{\varsigma}, L_{\varsigma})$ is asymptotically stable and $\{b^i\}_{i\geq 1}$ is a disturbance sequence converging strongly to a disturbance b^{∞}, then the limit profile corresponding to this disturbance sequence is the unique solution of the linear equation

$$y^{\infty} = L_{\varsigma} y^{\infty} + b^{\infty} \tag{2.17}$$

Proof (see Edwards and Owens (1982)).

The limit profile y^{∞} is the final converged solution of the unit memory repetitive process, assuming stability and uniqueness. That is

$$y^{\infty} = \lim_{i\to\infty} y^{\infty} \tag{2.18}$$

If we assume that the disturbance sequence $\{b^i\}_{i\geq 1}$ converges strongly to a disturbance b^{∞} then the corresponding limit profile of the process described by Eq. (2.12) is given by

$$y^{\infty} = (I - L_{\varsigma})^{-1} b^{\infty} \tag{2.19}$$

For the example of Eq. (2.14), the limit profile is described by

$$\left.\begin{aligned} \frac{d}{dt} X^{\infty}(t) &= \left(A_o + B_o \left(I - D_1\right)^{-1} C\right) X^{\infty}(t);\; X^{\infty}(0) = d^{\infty} \\ Y^{\infty}(t) &= \left(I - D_1\right)^{-1} C X^{\infty}(t) \end{aligned}\right\} \tag{2.20}$$

where it is assumed that the initial condition converges strongly to d^{∞}.

2.8.5 DISOPE as 2-D System

Iterative algorithms naturally fall into the area of 2D systems (Fornasini and Marchesini, 1978; Roesser, 1975) where one dimension is the time horizon of the

dynamic system under investigation and the other is the progress of the iterations. Roberts (1994) used linear 2-D system theory techniques to analyze local stability and convergence behavior of DISOPE. The 2-D analysis is based on the theory of unit memory repetitive processes developed principally by Edwards and Owens (1982).

Roberts (1994) stated that the nonlinear DISOPE algorithm can be formulated as a nonlinear unit memory repetitive process. In order to gain insight into local convergence properties of the technique, his initial work investigated the special case of linear real and model-based problem with quadratic performance indices. Using this theory, local convergence behavior of DISOPE has been analyzed and associated stability theorems have been obtained and studied (Roberts, 2000b).

An important observation from the analyses is that, the resulting 2-D system contains initial conditions whose values depend upon the output solution of the previous iteration. This results in a difficult complex eigenvalue type problem (Roberts, 2000b). The difficulties arise from the use of fixed time horizon. Based on the analyses and results of DISOPE, the two new algorithms will be analyzed accordingly. In the next section, some definitions and theorems for comparing efficiencies of algorithms are presented. Efficiencies are best described by complexity analysis.

2.9 Summary and Conclusion

In this chapter, we reviewed the literature on related topics touched by the research. These reviews present the underlying theme encompassing the whole research. All the topics reviewed here will later be used in the chapters that follow.

The discussions on ISOPE and DISOPE algorithms are the basis upon which the report stand on. The reviews on the gradient descent algorithms and the problems related to them are the motivating factors of the research. The numerous documented modifications on the gradient descent algorithms are reported as well.

The literature reviews are then tailored to the tools used in the back propagation algorithms of the neural networks in overcoming the gradient descent problems.

The major contribution of this research is the development and implementation of two new algorithms. The theoretical analyses of these new algorithms are achieved through the use of the concepts of 2-D presentation of the algorithms in the form of a unit memory repetitive processes and the limit profile. The preliminary discussions on these topics are also presented in this chapter. In the following chapter we proceed with the discussion of DISOPE in detail. The chapter is a compilation of work done by Roberts (1979,1993, 1994a, 1994b, 1999, 2000b, 2000c, and 2002) and Becerra (1994) on DISOPE.

CHAPTER 3

DISOPE AS AN ALGORITHM FOR SOLUTION OF NONLINEAR OPTIMAL CONTROL

3.1 Introduction

This chapter is a compilation of works done by Roberts (1993, 2000b, 2000c, 2001, 2002) and Becerra (1994). It describes DISOPE algorithm in detail and forms a basis for the work in subsequent chapters. The chapter starts with the formulation of problems and goes on to describe the algorithm. Next, we present the algorithm mapping of DISOPE. Roberts (2000b, 2000c, 2002) used 2-D systems theory (Edwards and Owens, 1982) in the form of unit memory repetitive process to analyze the optimality, stability, and convergence behavior of the algorithm. All the analyses are presented here. The purpose of this chapter is to be a reference point for comparing with results from two new algorithms in Chapters 5 and 6. Three simulation examples with varying degrees of nonlinearity are included in this chapter.

3.2 Problem Formulation

DISOPE is a technique for solving nonlinear optimal control problems subject to model-reality differences (Roberts, 1993). In the continuous version of the method, it strives to find the solution of the following Real Optimal Control Problem (ROP):

$$\left.\begin{aligned} &\min_{u(t)} J^* = \left\{ \Phi^*(x(t_f)) + \int_{t_0}^{t_f} L^*(x(t),u(t),t)dt \right\} \\ &\text{subject to} \\ &\quad \dot{x}(t) = f^*(x(t),u(t),t); \quad x(t_0) = x_0 \\ &\quad \Psi^*(x(t_f)) = 0_{q,1} \end{aligned}\right\} \tag{3.1}$$

with terminal conditions and constraints. ROP is defined over the fixed time horizon $t \in [t_0, t_f]$, where $u(t) \in \mathbb{R}^m$ and $x(t) \in \mathbb{R}^n$ are the continuous control and state vectors respectively, $\Phi^* : \mathbb{R}^n \to \mathbb{R}$ is the real terminal measure, dependent on the final state and time. $L^* : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ is the real performance measure function, dependent on the state and input at intermediate times $[t_0, t_f]$. $f^* : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ represents the real nonlinear dynamical system, and $\Psi^* : \mathbb{R}^n \to \mathbb{R}^q$ is the real terminal constraint vector.

However, since a good representation of reality does not always translate into ease of solution, DISOPE does not work directly on ROP. The technique modeled the reality with a *manageable* equivalent problem, called the Model Based Optimal Control Problem (MOP).

$$\left.\begin{aligned} &\min_{u(t)} J_{MOP} = \left\{ \Phi^*(x(t_f)) + \int_{t_0}^{t_f} \left[L(x(t),u(t),\gamma(t)) \right] dt \right\} \\ &\text{subject to} \\ &\quad \dot{x}(t) = f(x(t),u(t),\alpha(t)); \quad x(t_0) = x_0 \\ &\quad \Psi^*(x(t_f)) = 0_{q,1} \end{aligned}\right\} \tag{3.2}$$

In an attempt of not totally forsaking reality, the formulation of MOP includes parameter estimates $\gamma(t) \in \mathbb{R}$ and $\alpha(t) \in \mathbb{R}^r$. These two estimates *take account* of the *value differences* between reality and model. In Eq. (3.2) $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ and $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}^n$ are respectively, the performance index and the plant dynamics of the model. Both are approximates of the real performance index L^* and the plant dynamics f^*.

MOP is then expanded into another optimal control problem, which is equivalent to ROP called the Expanded Optimal Control Problem (EOP). The role of EOP is to tie both ROP and MOP together. Including the equality expressions of

state functions and performance indices from both ROP and MOP as constraints does this. EOP is defined as follows:

$$\left.\begin{aligned}
&\min_{u(t)} J_{EOP} = \left\{ \Phi^*(x(t_f)) + \int_{t_0}^{t_f} \left[L(x(t), u(t), \gamma(t)) \right.\right. \\
&\qquad \left.\left. + \frac{1}{2} r_1 \left\| u(t) - \hat{u}(t) \right\|^2 + \frac{1}{2} r_2 \left\| x(t) - \hat{x}(t) \right\|^2 \right] dt \right\} \\
&\text{subject to} \\
&\qquad \dot{x}(t) = f(x(t), u(t), \alpha(t)); \quad x(t_0) = x_0 \\
&\qquad \Psi^*(x(t_f)) = 0_{q,1}
\end{aligned}\right\} \tag{3.3}$$

with the new additional constraints defined as

$$\left.\begin{aligned}
f(\hat{x}(t), \hat{u}(t), \alpha(t)) &= f^*(\hat{x}(t), \hat{u}(t), t) \\
L(\hat{x}(t), \hat{u}(t), \gamma(t)) &= L^*(\hat{x}(t), \hat{u}(t), t) \\
u(t) &= \hat{u}(t) \\
x(t) &= \hat{x}(t)
\end{aligned}\right\} \tag{3.4}$$

In Eqs. (3.3) and (3.4), $\hat{u}(t) \in \mathbb{R}^m$ and $\hat{x}(t) \in \mathbb{R}^n$ are introduced as the state and control variables used in the optimization step. This is to distinguish them from the variables used in the parameter estimation step. $\left\| u(t) - \hat{u}(t) \right\|$ and $\left\| x(t) - \hat{x}(t) \right\|$ are convexification terms introduced in the performance index to aid convergence. r_1 and r_2 are given scalar convexification factors, which are adjustable to provide a facility for regulating convergence (Roberts, 2002). EOP is used as basis for solving ROP. All the necessary optimality conditions needed for solving ROP are derived from it.

In the present and subsequent chapters, the norm that we will be using is the Euclidean norm $\left\| x \right\|_p = \left(\sum \left\| x_i \right\|^p \right)^{\frac{1}{p}}$. The default value would be the same as those used by MATLAB which is $p = 2$, corresponding to the Euclidean length.

Following the tradition of the variational approach to solving constrained optimization problems, the constraints in Eqs. (3.3) and (3.4) are adjoined to the performance index with Lagrange multipliers to produce an augmented problem. To

reduce the likelihood of confusion, the time index *t* has been dropped from the following formulation of the augmented performance index, where the subscript *ea* stands for 'augmented EOP'.

$$\begin{aligned}\min_{u(t)} J_{ea} = \Phi^*(x(t_f)) + \int_{t_0}^{t_f} \big[& L(x,u,\gamma) + p^T(f(x,u,\alpha) - \dot{x}) \\ & + \lambda^T(\hat{u}-u) + \beta^T(\hat{x}-x) + \mu^T(f^*(\hat{x},\hat{u},t) - f(\hat{x},\hat{u},\alpha)) \\ & + \xi^T(L^*(\hat{x},\hat{u},t) - L(\hat{x},\hat{u},\gamma)) + \frac{1}{2} r_1 \|u-\hat{u}\|^2 + \frac{1}{2} r_2 \|x-\hat{x}\|^2 \big] dt\end{aligned} \tag{3.5}$$

where $p(t) \in \mathbb{R}^n$ is the costate vector and $\lambda(t) \in \mathbb{R}^m$, $\beta(t) \in \mathbb{R}^n$, $\mu(t) \in \mathbb{R}^m$, and $\xi(t) \in \mathbb{R}$ are the multipliers. From Eq. (3.5) we define a Hamiltonian *H* as:

$$H(x,u,\gamma,p,\alpha) = L(x,u,\gamma) + p^T f(x,u,\alpha) - \lambda^T u - \beta^T x \tag{3.6}$$

with which the performance index of Eq. (3.5) becomes

$$\begin{aligned}\min_{u(t)} J_{ea} = \Phi^*(x(t_f)) + \int_{t_0}^{t_f} \big[& H - p^T \dot{x} + \lambda^T \hat{u} + \beta^T \hat{x} \\ & + \mu^T(f^*(\hat{x},\hat{u},t) - f(\hat{x},\hat{u},\alpha)) + \xi^T(L^*(\hat{x},\hat{u},t) - L(\hat{x},\hat{u},\gamma)) \\ & + \tfrac{1}{2} r_1 \|u-\hat{u}\|^2 + \tfrac{1}{2} r_2 \|x-\hat{x}\|^2 \big] dt\end{aligned} \tag{3.7}$$

With reference to the Lagrange theory, the minimum of EOP is achieved by finding the minimum of the unconstrained J_{ea}. This happened when the first variation of J_{ea} is set to zero, that is $\delta J_{ea} = 0$. With t_0, t_f, and x_0 fixed, the first variation of J_{ea} is as follows:

$$\begin{aligned}\delta J_{ea} = \nabla_x \Phi^{*T} \delta x\big|_{t=t_f} + \int_{t_0}^{t_f} \Big\{ & \left(\nabla_u H + r_1 \|u-\hat{u}\|\right)^T \delta u + \left(\nabla_x H + r_2 \|x-\hat{x}\|\right)^T \delta x \\ & - p^T \delta \dot{x} + [\nabla_p H - \dot{x}]^T \delta p \\ & + [\lambda + (f_{\hat{u}}^* - f_{\hat{u}})^T \mu + (\nabla_{\hat{u}} L^* - \nabla_{\hat{u}} L)^T \xi - r_1 \|u-\hat{u}\|]^T \delta \hat{u} \\ & + [\beta + (f_{\hat{x}}^* - f_{\hat{x}})^T \mu + (\nabla_{\hat{x}} L^* - \nabla_{\hat{x}} L)^T \xi - r_2 \|x-\hat{x}\|]^T \delta \hat{x} \\ & + [\nabla_\alpha H - f_\alpha^T \mu]^T \delta\alpha + [\nabla_\gamma H - \nabla_\gamma L^T \xi]^T \delta\gamma \Big\} dt\end{aligned} \tag{3.8}$$

To eliminate the variation in $\dot{x}$, the expression $-p^T \delta \dot{x}$ in Eq. (3.8) is integrated by parts (Lewis and Syrmos, 1995) producing the following expression for the first variation of J_{ea}.

$$\begin{aligned}\delta J_{ea} = &[\nabla_x \Phi^* - p]^T \delta x\big|_{t=t_f} + p^T \delta x\big|_{t=t_0} + \int_{t_0}^{t_f} \Big\{ \left(\nabla_u H + r_1 \|u-\hat{u}\|\right)^T \delta u \\ &+ \left(\nabla_x H + r_2 \|x-\hat{x}\| + \dot{p}\right)^T \delta x + [\nabla_p H - \dot{x}]^T \delta p \\ &+ [\lambda + (f_{\hat{u}}^* - f_{\hat{u}})^T \mu + (\nabla_{\hat{u}} L^* - \nabla_{\hat{u}} L)^T \xi - r_1 \|u-\hat{u}\|]^T \delta \hat{u} \\ &+ [\beta + (f_{\hat{x}}^* - f_{\hat{x}})^T \mu + (\nabla_{\hat{x}} L^* - \nabla_{\hat{x}} L)^T \xi - r_2 \|x-\hat{x}\|]^T \delta \hat{x} \\ &+ [\nabla_\alpha H - f_\alpha^T \mu]^T \delta\alpha + [\nabla_\gamma H - \nabla_\gamma L^T \xi]^T \delta\gamma \Big\} dt\end{aligned} \tag{3.9}$$

To get the necessary optimality conditions for the optimization, the coefficients of the independent increments of $u, x,$ and p in Eq. (3.9) are set to zero. These conditions are

$$\left.\begin{aligned}\nabla_u H + r_1(u(t)-\hat{u}(t)) &= 0 \\ \nabla_x H + \dot{p}(t) + r_2(x(t)-\hat{x}(t)) &= 0 \\ \nabla_p H - \dot{x}(t) &= 0\end{aligned}\right\} \tag{3.10}$$

Also, from Eq. (3.9), the expression for the Lagrangian multipliers λ and β are obtained by setting to zero the coefficients of the increments of $\hat{u}$ and $\hat{x}$ respectively, while taking the values of the other two multipliers μ and ξ to be $\mu = \hat{p}$ and $\xi = 1$ (Becerra, 1994) and from Eq. (3.4), $u - \hat{u} = 0$ and $x - \hat{x} = 0$. The expressions are

$$\left.\begin{aligned}\lambda(t) &= \left[\frac{\partial f}{\partial \hat{u}}(.) - \frac{\partial f^*}{\partial \hat{u}}(.)\right]^T \hat{p}(t) + \left(\nabla_{\hat{u}} L^*(.) - \nabla_{\hat{u}} L(.)\right) \\ \beta(t) &= \left[\frac{\partial f}{\partial \hat{x}}(.) - \frac{\partial f^*}{\partial \hat{x}}(.)\right]^T \hat{p}(t) + \left(\nabla_{\hat{x}} L^*(.) - \nabla_{\hat{x}} L(.)\right)\end{aligned}\right\} \tag{3.11}$$

From the constants at the beginning of the expression of Eq. (3.9), we get the terminal condition for EOP as

$$p(t_f) = \nabla_x \Phi^*(x(t))\big|_{t=t_f} \tag{3.12}$$

Thus, the solution of the optimal control problem EOP defined by Eqs. (3.3) and (3.4) is arrived at by solving Eqs. (3.10) together with the boundary conditions $x(t_0) = x_0$ and $p(t_f) = \nabla_x \Phi^*(x(t))\big|_{t=t_f}$, and optimality conditions

$$\left.\begin{aligned} f(\hat{x}(t),\hat{u}(t),\alpha(t)) - f^*(\hat{x}(t),\hat{u}(t),t) = 0 \\ L(\hat{x}(t),\hat{u}(t),\gamma(t)) - L^*(\hat{x}(t),\hat{u}(t),t) = 0 \end{aligned}\right\} \tag{3.13}$$

and

$$\left.\begin{aligned} u(t) = \hat{u}(t) \\ x(t) = \hat{x}(t) \\ p(t) = \hat{p}(t) \end{aligned}\right\} \tag{3.14}$$

The pair of equations in Eq. (3.13) defines the parameter estimation problem and the ones in Eq. (3.14) define the interconnections between the parameter estimation problem and the optimization problem. Since EOP is equivalent to ROP, the solution of ROP is also gained.

Looking back at the Hamiltonian in Eq. (3.6), the necessary optimality conditions in Eq. (3.10), and border conditions $x(t_0) = x_0$ and $p(t_f) = \nabla_x \Phi^*(x(t))\big|_{t=t_f}$, we can easily see that an optimal control problem defined as

$$\left.\begin{aligned} &\min_{u(t)} J_{MMOP} = \Big\{\Phi^*(x(t_f)) + \int_{t_0}^{t_f} \big[L(x(t),u(t),\gamma(t)) - \lambda(t)^T u(t) \\ &\quad - \beta(t)^T x(t) + \tfrac{1}{2} r_1 \|u(t) - \hat{u}(t)\|^2 + \tfrac{1}{2} r_2 \|x(t) - \hat{x}(t)\|^2 + \tfrac{1}{2} r_3 \|p(t) - \hat{p}(t)\|^2 \big] dt \Big\} \\ &\text{subject to} \\ &\quad \dot{x}(t) = f(x(t),u(t),\alpha(t)); \quad x(t_0) = x_0 \\ &\quad \Psi^*(x(t_f)) = 0_{q,1} \end{aligned}\right\} \tag{3.15}$$

also satisfies them. This problem is simpler than EOP above. Thus, instead of using EOP as the model to be solved, we will use this problem called the Modified Model Based Problem (MMOP) as the model used in finding the solution of ROP. If with given values of $\hat{u}(t), \hat{x}(t)$ and $\hat{p}(t)$ we compute the functions of $\lambda(t)$, $\alpha(t)$, $\gamma(t)$, and $\beta(t)$ using Eqs. (3.11) and (3.13), and if the solutions obtained satisfy conditions in Eq. (3.14), then that solution is also the solution to ROP (Becerra, 1994). Notice, in Eq. (3.15), we add another convexification term $\|p(t) - \hat{p}(t)\|^2$ in order to aid the convexification of the costate. To satisfy the optimality conditions of (3.10), we can give the value of zero to r_3 since in problems involving Lagrange multipliers, the values of the multipliers are not essential in solving the problems.

3.3 DISOPE Algorithm

The previous reasoning leads to the following algorithm with MMOP as model.

Algorithm 3.1

Data $f,\ L,\ \varphi,\ x_0,\ t_0,\ t_f$ and means for calculating f^* and L^*.

Step 0 Compute or choose a nominal solution $u^0(t), x^0(t),$ and $p^0(t)$. Set $i = 0, u^0(t) = \hat{u}^0(t),\ x^0(t) = \hat{x}^0(t),\ p^0(t) = \hat{p}^0(t),\ t \in [t_0, t_f]$.

Step 1 Compute the parameters $\alpha^{(i)}(t),\ \gamma^{(i)}(t)$ using (3.13). This is called the *parameter estimation* step.

S*tep 2* Compute the multipliers $\lambda^{(i)}(t)$ and $\beta^{(i)}(t)$ from Eq. (3.11).

Step 3 With specified $\alpha^{(i)}(t),\ \gamma^{(i)}(t),\ \lambda^{(i)}(t)$, and $\beta^{(i)}(t)$ solve MMOP to obtain $\hat{u}^{(i)}(t),\ \hat{x}^{(i)}(t)$, and $\hat{p}^{(i)}(t)$. This is called the *system optimization* step.

Step 4 This step is the updating mechanism, testing the convergence and updates the estimates for the solution of ROP.

$$\left.\begin{aligned} u^{(i+1)}(t) &= u^{(i)}(t) + k_u(\hat{u}^{(i)}(t) - u^{(i)}(t)) \\ x^{(i+1)}(t) &= x^{(i)}(t) + k_x(\hat{x}^{(i)}(t) - x^{(i)}(t)) \\ p^{(i+1)}(t) &= p^{(i)}(t) + k_p(\hat{p}^{(i)}(t) - p^{(i)}(t)) \end{aligned}\right\} \tag{3.16}$$

where $k_u, k_x, k_p \in (0,1]$ are scalar gains. If $\left\|u^{(i+1)}(t) - u^{(i)}(t)\right\| \le \varepsilon$, ε a given tolerance, stop, else set $i = i + 1$ and continue from step 1.

Figure 3.1 is the schematic representation of DISOPE in the form of a flow chart.

3.4 DISOPE With LQR as Model

For the purpose of this research, only LQR problems will be considered as

model whilst DISOPE is used as the search algorithm for finding the optimal solution of the optimal control problems presented. Consider MMOP as defined in Eq. (3.15) , which is equivalent to ROP as defined in Eq. (3.1). For the benefit of subsequent sections, MMOP will be formulated as an LQR model at the *i*th iteration in the following fashion.

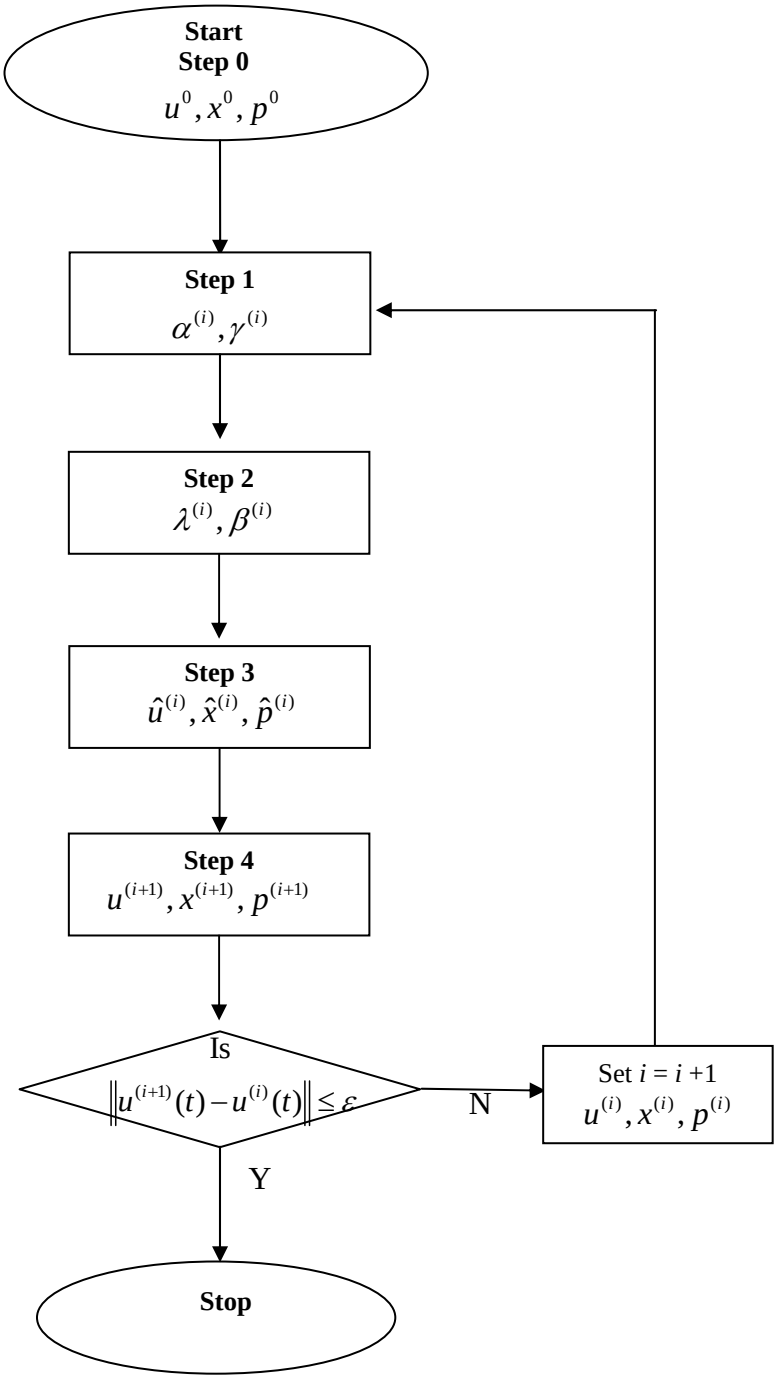


Figure 3.1: The flow chart of DISOPE algorithm.

$$\min_{u(t)} J = \left\{ \begin{array}{l} \frac{1}{2}\left(x(t_f)^T S x(t_f) + \Gamma_1^{(i)T} x(t_f) \right) + \int_{t_0}^{t_f} \left\{ \frac{1}{2}(x(t)^T Q x(t) + u(t)^T R u(t)) \right. \\ \qquad - \lambda^{(i)}(t)^T u(t) - \beta^{(i)}(t)^T x(t) \\ \qquad \left. + \frac{1}{2} r_1 \left\| u(t) - u^{(i)}(t) \right\|^2 + \frac{1}{2} r_2 \left\| x(t) - x^{(i)}(t) \right\|^2 \right\} dt \\ \text{subject to: } \dot{x}(t) = Ax(t) + Bu(t) + \alpha^{(i)}(t);\ x(t_0) = x_0 \\ \qquad Vx(t_f) + b + \gamma_3^{(i)} = 0_{q,1} \end{array} \right\} \quad (3.17)$$

The solution of MMOP is obtained by solving the appropriate optimality conditions (Roberts, 2002), which provide estimates of the optimal control, state, and costate signals as:

$$\left. \begin{array}{l} \hat{u}^{(i)}(t) = \bar{R}^{-1}\left(-B^T \hat{p}^{(i)}(t) + \lambda^{(i)}(t) + r_1 u^{(i)}(t) \right) \\ \frac{d}{dt}\hat{x}^{(i)}(t) = A\hat{x}^{(i)}(t) + B\hat{u}^{(i)}(t) + \alpha^{(i)}(t) \\ \frac{d}{dt}\hat{p}^{(i)}(t) = -\bar{Q}\hat{x}^{(i)}(t) - A^T \hat{p}^{(i)}(t) + \beta^{(i)}(t) + r_2 x^{(i)}(t) \end{array} \right\} \quad (3.18)$$

where $\bar{R} = R + r_1 I_m$, $\bar{Q} = R + r_2 I_n$ and $\hat{p}^{(i)}(t) \in \mathbb{R}^n$ is the vector of costate variables. The corresponding mixed boundary conditions are:

$$\left. \begin{array}{l} \hat{x}^{(i)}(t_0) = x_0 \\ \hat{p}^{(i)}(t_f) = S\hat{x}^{(i)}(T) + \Gamma_1^{(i)} + \left[V + \Gamma_2^{(i)} \right]^T \chi \\ V\hat{x}^{(i)}(t_f) + b + \gamma_3^{(i)} = 0_{q,1} \end{array} \right\} \quad (3.19)$$

In Eqs. (3.17), (3.18), and (3.19), $S \in \mathbb{R}^{n \times n}$ and $Q \in \mathbb{R}^{n \times n} \geq 0$ are symmetric and positive semidefinite matrices. $R \in \mathbb{R}^{m \times m} \geq 0$ is symmetric positive definite. $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $V \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^{q}$, χ is a Lagrange multiplier. $(\hat{\cdot})$ represents the current solution and $(.)^{(i)}$ represents the variable value at iteration *i*. The model parameters $\alpha(t) \in \mathbb{R}^n$, and $\gamma_3 \in \mathbb{R}^q$, together with modifiers $\lambda(t) \in \mathbb{R}^m$, $\beta(t) \in \mathbb{R}^n$, $\Gamma_1 \in \mathbb{R}^n$, and $\Gamma_2 \in \mathbb{R}^{q \times n}$, are calculated using the relationships:

$$\left. \begin{array}{l} \alpha^{(i)}(t) = f^*(x^{(i)}(t), u^{(i)}(t)) - Ax^{(i)}(t) - Bu^{(i)}(t) \\ \gamma_3^{(i)} = \Psi^*(x^{(i)}(t_f)) - Vx^{(i)}(t_f) - b \end{array} \right\} \quad (3.20)$$

and

$$\left.\begin{aligned}
\lambda^{(i)}(t) &= -\left[\frac{\partial f^*(\cdot)}{\partial u} - B\right]^T p^{(i)}(t) - \left[\nabla_u L^*(\cdot)\right] - Ru^{(i)}(t) \\
\beta^{(i)}(t) &= -\left[\frac{\partial f^*(\cdot)}{\partial x} - A\right]^T p^{(i)}(t) - \left[\nabla_x L^*(\cdot)\right] - Qx^{(i)}(t) \\
\Gamma_1^{(i)} &= \nabla_x \Phi^*(x^{(i)}(t_f)) - Sx^{(i)}(t_f) \\
\Gamma_2^{(i)} &= \frac{\partial \Psi^*(x^{(i)}(t_f))}{\partial x} - V
\end{aligned}\right\} \qquad (3.21)$$

In addition to using scalars convexification factors r_1 and r_2, convergence and stability is also regulated by use of the relaxation scheme given by Eq. (3.16) (Roberts, 2002).

3.5 The Algorithm Mapping of DISOPE

To gain an insight into the inner process of DISOPE; we developed the following algorithm mapping based on Roberts (1994a, 1994b, 2000a, 2000b, 2002). Here DISOPE is characterized as a non linear mapping that defines a recursive relationship for the control $u^{(i)}(t)$, state $x^{(i)}(t)$, and costate $p^{(i)}(t)$ in terms of their values at the previous iteration. The mapping shows how the algorithm solution develops from iteration to iteration and will be the basis for subsequent 2-D interpretation, optimality, stability, and convergence analyses. For the purpose of simplification, we define the following notations for use in the analyses.

$$\left.\begin{aligned}
&B^*(x,u,t) = \frac{\partial f^*(x(t),u(t))}{\partial u},\ A^*(x,u,t) = \frac{\partial f^*(x(t),u(t))}{\partial x} \\
&L_u^*(x,u,t) = \nabla_u L^*(x(t),u(t)),\ L_x^*(x,u,t) = \nabla_x L^*(x(t),u(t)) \\
&\Phi_x^*(x(t_f)) = \nabla_x \Phi^*(x(t_f)), \qquad V^*(x^{(i)}(t_f)) = \frac{\partial \Psi^*(x^{(i)}(t_f))}{\partial x}
\end{aligned}\right\} \qquad (3.22)$$

With these notations, the transition from iteration *i* to iteration *i*+1 given by Eqs. (3.16), (3.18)-(3.21), can be expressed as

$$\left.\begin{aligned}
\bar{\lambda}^{(i)}(t) &= \left[B - B^*(x^{(i)},u^{(i)},t)\right]p^{(i)}(t) + \bar{R}u^{(i)}(t) - L_u^*(x^{(i)},u^{(i)},t)\\
\bar{\beta}^{(i)}(t) &= \left[A - A^*(x^{(i)},u^{(i)},t)\right]p^{(i)}(t) + \bar{Q}x^{(i)}(t) - L_x^*(x^{(i)},u^{(i)},t)\\
\Gamma_1^{(i)} &= -\left[Sx^{(i)}(t_f) - \Phi_x^*(x^{(i)}(t_f))\right]\\
\Gamma_2^{(i)} &= -\left[V - V^*(x^{(i)}(t_f))\right]\\
\alpha^{(i)}(t) &= f^*(x^{(i)},u^{(i)},t) - Ax^{(i)}(t) - Bu^{(i)}(t)\\
\gamma_3^{(i)} &= \Psi^*(x^{(i)}(t_f)) - Vx^{(i)}(t_f) - b
\end{aligned}\right\} \tag{3.23}$$

$$\left.\begin{aligned}
\hat{u}^{(i)}(t) &= \bar{R}^{-1}(-B^T\hat{p}^{(i)}(t) + \bar{\lambda}^{(i)}(t)\\
\frac{d}{dt}\hat{x}^{(i)}(t) &= A\hat{x}^{(i)}(t) - B\bar{R}^{-1}B^T\hat{p}^{(i)}(t) + \alpha^{(i)}(t) + B\bar{R}^{-1}\bar{\lambda}^{(i)}(t)\\
\frac{d}{dt}\hat{p}^{(i)}(t) &= -\bar{Q}\hat{x}^{(i)}(t) - A^T\hat{p}^{(i)}(t) + \bar{\beta}^{(i)}(t)
\end{aligned}\right\} \tag{3.24}$$

$$\left.\begin{aligned}
\hat{p}^{(i)}(t_f) &= S\hat{x}^{(i)}(t_f) + \Gamma_1^{(i)} + \left[V + \Gamma_2^{(i)}\right]\chi^{(i)}\\
V\hat{x}^{(i)}(t_f) + b + \gamma_3^{(i)} &= 0_{q,1}
\end{aligned}\right\} \tag{3.25}$$

where $\bar{\lambda}^{(i)}(t) = \lambda^{(i)}(t) + r_1u^{(i)}(t)$ and $\bar{\beta}^{(i)}(t) = \beta^{(i)}(t) + r_2x^{(i)}(t)$. The updating mechanism given by Eq. (3.16) stays as is which is

$$\left.\begin{aligned}
u^{(i+1)}(t) &= u^{(i)}(t) + k_u(\hat{u}^{(i)}(t) - u^{(i)}(t))\\
x^{(i+1)}(t) &= x^{(i)}(t) + k_x(\hat{x}^{(i)}(t) - x^{(i)}(t))\\
p^{(i+1)}(t) &= p^{(i)}(t) + k_p(\hat{p}^{(i)}(t) - p^{(i)}(t))
\end{aligned}\right\} \tag{3.26}$$

Similar to the general expressions in Section 2.2, Eq. (3.23) represents the computation of modifiers and parameters, Eq. (3.24) represents the solution of the MMOP subject to terminal conditions defined by Eq. (3.25), and Eq. (3.26) represents the update of control, state, and costate estimates.

With the use of Eq. (3.23), Eq. (3.24), can be simplified into

$$\begin{bmatrix}\frac{d}{dt}\hat{x}^{(i)}(t)\\ \frac{d}{dt}\hat{p}^{(i)}(t)\end{bmatrix} = H\begin{bmatrix}\hat{x}^{(i)}(t)\\ \hat{p}^{(i)}(t)\end{bmatrix} + E_1y^{(i)}(t) + g_1(y^{(i)}(t)) \tag{3.27}$$

where

$$y^{(i)}(t) = \left[u^{(i)}(t)^T \quad x^{(i)}(t)^T \quad p^{(i)}(t)^T\right]^T \tag{3.28}$$

and

$$H=\begin{bmatrix} A & -B\bar{R}^{-1}B^T \\ -\bar{Q} & -A^T \end{bmatrix},\ E_1=\begin{bmatrix} r_1 B\bar{R}^{-1} & O_n & O_n \\ O_{n,m} & r_2 I_n & O_n \end{bmatrix},\ g_1(y^{(i)}(t))=\begin{bmatrix} g_{11}(y^{(i)}(t)) \\ g_{21}(y^{(i)}(t)) \end{bmatrix} \Bigg\} \tag{3.29}$$

with

$$\left.\begin{aligned} g_{11}(y^{(i)}(t)) &= B\bar{R}^{-1}\left(\left[B-B^*(x^{(i)},u^{(i)},t)\right]^T \hat{p}^{(i)}(t)+Ru^{(i)}(t)-L_u^*(x^{(i)},u^{(i)},t)\right) \\ &\quad + f(x^{(i)},u^{(i)},t)-Ax^{(i)}(t)-Bu^{(i)}(t) \\ g_{12}(y^{(i)}(t)) &= \left(\left[A-A^*(x^{(i)},u^{(i)},t)\right]^T \hat{p}^{(i)}(t)+Qx^{(i)}(t)-L_x^*(x^{(i)},u^{(i)},t)\right) \end{aligned}\right\} \tag{3.30}$$

In Eq. (3.29), the Hamiltonian matrix H is a transition matrix; E_1 represents contributions from r_1 and r_2, and $g_1(\cdot)$ represents the model reality differences. The terminal conditions become:

$$\left.\begin{aligned} &\hat{p}^{(i)}(t_f)=\Phi_x^*\left(x^{(i)}(t_f)\right)+\left[V^*\left(x^{(i)}(t_f)\right)\right]^T \chi^{(i)}+S\left(\hat{x}^{(i)}(t_f)-x^{(i)}(t_f)\right) \\ &\Psi^*\left(x^{(i)}(t_f)\right)+V\left(\hat{x}^{(i)}(t_f)-x^{(i)}(t_f)\right)=O_{q,1} \end{aligned}\right\} \tag{3.31}$$

The solution of Eq. (3.27) becomes

$$\begin{bmatrix} \hat{x}^{(i)}(t) \\ \hat{p}^{(i)}(t) \end{bmatrix}=\phi(t,t_0)\begin{bmatrix} x_0 \\ \hat{p}^{(i)}(t_0) \end{bmatrix}+\int_{t_0}^{t}\phi(t,\tau)\left[E_1 y^{(i)}(\tau)+g_1\left(y^{(i)}(\tau)\right)\right]d\tau \tag{3.32}$$

where

$$\phi(t,\tau)=e^{H(t-\tau)} \tag{3.33}$$

By writing

$$\phi(t,\tau)=\begin{bmatrix} \phi_1(t,\tau) \\ \phi_2(t,\tau) \end{bmatrix}=\begin{bmatrix} \phi_{11}(t,\tau) & \phi_{12}(t,\tau) \\ \phi_{21}(t,\tau) & \phi_{22}(t,\tau) \end{bmatrix} \tag{3.34}$$

and substituting it into Eq. (3.31), the elimination of $\chi^{(i)}$, produces

$$\begin{aligned} \hat{p}^{(i)}(t) = -\bar{\phi}_{22}\left(t_f,t_0\right)^{-1}\Big\{&\bar{\phi}_{21}\left(t_f,t_0\right)x_0+\bar{\Psi}^*\left(y^{(i)}(t_f)\right) \\ &+\int_{t_0}^{t_f}\bar{\phi}_2\left(t_f,\tau\right)\left[E_1 y^{(i)}(\tau)+g\left(y(\tau)\right)\right]d\tau\Big\} \end{aligned} \tag{3.35}$$

where

$$\left.\begin{aligned}
\bar{\phi}_{21}(t_f,t_0) &= \tilde{\phi}_{21}(t_f,t_0)+\left[V^*\left(x^{(i)}(t_f)\right)\right]^T\sum\left(x^{(i)}(t_f)\right)^{-1}V\tilde{\phi}_{11}(t_f,t_0)\\
\bar{\phi}_{2}(t_f,\tau) &= \tilde{\phi}_{2}(t_f,\tau)+\left[V^*\left(x^{(i)}(t_f)\right)\right]^T\sum\left(x^{(i)}(t_f)\right)^{-1}V\tilde{\phi}_{1}(t_f,\tau)\\
\bar{\Psi}^*\left(y^{(i)}(t_f)\right) &= Sx^{(i)}(t_f)-\Phi_x^*\left(x^{(i)}(t_f)\right)\\
&\quad+\left[V^*\left(x^{(i)}(t_f)\right)\right]^T\sum\left(x^{(i)}(t_f)\right)^{-1}\tilde{\Psi}^*\left(x^{(i)}(t_f)\right)
\end{aligned}\right\} \quad (3.36)$$

$$\left.\begin{aligned}
\tilde{\phi}_{11}(t_f,t_0) &= \phi_{11}(t_f,t_0)-\phi_{12}(t_f,t_0)\tilde{\phi}_{22}(t_f,t_0)^{-1}\tilde{\phi}_{21}(t_f,t_0)\\
\tilde{\phi}_{1}(t_f,\tau) &= \phi_{1}(t_f,\tau)-\phi_{12}(t_f,t_0)\tilde{\phi}_{22}(t_f,t_0)^{-1}\tilde{\phi}_{2}(t_f,\tau)\\
\tilde{\Psi}^*\left(x^{(i)}(t_f)\right) &= \Psi^*\left(x^{(i)}(t_f)\right)\\
&\quad-Vx^{(i)}(t_f)+V\phi_{12}(t_f,t_0)\tilde{\phi}_{22}(t_f,t_0)^{-1}\left(\Phi_x^*\left(x^{(i)}(t_f)\right)-Sx^{(i)}(t_f)\right)\\
\sum\left(x^{(i)}(t_f)\right) &= V\phi_{12}(t_f,t_0)\tilde{\phi}_{22}(t_f,t_0)^{-1}\left[V^*\left(x^{(i)}(t_f)\right)\right]^T
\end{aligned}\right\} \quad (3.37)$$

and

$$\left.\begin{aligned}
\tilde{\phi}_{22}(t_f,t_0) &= \phi_{22}(t_f,t_0)-S\phi_{12}(t_f,t_0)\\
\tilde{\phi}_{21}(t_f,t_0) &= \phi_{21}(t_f,t_0)-S\phi_{1}(t_f,t_0)\\
\tilde{\phi}_{2}(t_f,\tau) &= \phi_{2}(t_f,\tau)-S\phi_{1}(t_f,\tau)
\end{aligned}\right\} \quad (3.38)$$

Furthermore, using Eq. (3.23) the optimal control estimates in Eq. (3.24), can be written as

$$\begin{aligned}
\hat{u}^{(i)}(t) &= -\bar{R}^{-1}B^T\hat{p}^{(i)}(t)+r_1\bar{R}^{-1}u^{(i)}(t)+\bar{R}^{-1}\left[B-B^*(x^{(i)},u^{(i)},t)\right]^T p^{(i)}(t)\\
&\quad+\bar{R}^{-1}\left[Ru^{(i)}(t)-L_u^*(x^{(i)},u^{(i)},t)\right]
\end{aligned} \quad (3.39)$$

The updating mechanism represented by Eq. (3.26) can be written as

$$y^{(i+1)}(t) = y^{(i)}(t)+K_y\begin{bmatrix}\hat{u}^{(i)}(t)\\ \hat{x}^{(i)}(t)\\ \hat{p}^{(i)}(t)\end{bmatrix}-K_y y^{(i)}(t) \quad (3.40)$$

where

$$K_y = \begin{bmatrix}k_u I_m & O_{m,n} & O_{m,n}\\ O_{n,m} & k_x I_n & O_n\\ O_{n,m} & O_n & k_p I_n\end{bmatrix} \quad (3.41)$$

Let

$$\hat{y}^{(i)}(t) = [\hat{u}^{(i)}(t)^T \quad \hat{x}^{(i)}(t)^T \quad \hat{p}^{(i)}(t)^T]^T \tag{3.42}$$

then (3.40) can be written as

$$y^{(i+1)}(t) = y^{(i)}(t) + K_y\left[\hat{y}^{(i)}(t) - y^{(i)}(t)\right] \tag{3.43}$$

which can be expressed as

$$y^{(i+1)}(t) = K_y\hat{y}^{(i)}(t) + \left[I_{2n+m} - K_y\right]y^{(i)}(t) \tag{3.44}$$

The combination of Eqs. (3.39) and (3.44) produces

$$y^{(i+1)}(t) = C\begin{bmatrix}\hat{x}^{(i)}\\ \hat{p}^{(i)}\end{bmatrix} + \left[I_{2n+m} - K_y\right]y^{(i)}(t) + E_2 y^{(i)}(t) + g_2\left(y^{(i)}(t)\right) \tag{3.45}$$

where

$$C = \begin{bmatrix} O_{m,n} & -k_u\bar{R}^{-1}B^T \\ k_x I_n & O_n \\ O_n & k_p I_n \end{bmatrix}, \quad E_2 = \begin{bmatrix} r_1 k_u\bar{R}^{-1} & O_{m,n} & O_{m,n} \\ O_{n,m} & O_n & O_n \\ O_{n,m} & O_n & O_n \end{bmatrix}, \tag{3.46}$$

$$g_2\left(y^{(i)}(t)\right) = \begin{bmatrix} g_{21}\left(y^{(i)}(t)\right) \\ O_{n,2n+m} \\ O_{n,2n+m} \end{bmatrix} \tag{3.47}$$

with

$$g_{21}\left(y^{(i)}(t)\right) = k_u\bar{R}^{-1}\left\{\left[B - B^*(x^{(i)},u^{(i)},t)\right]^T p^{(i)}(t) + \left[Ru^{(i)}(t) - L_u^*(x^{(i)},u^{(i)},t)\right]\right\} \tag{3.48}$$

We note that here E_2 provides a contribution from the convexification coefficient r_1 and $g_2(\cdot)$ represents model-reality differences.

From Eqs. (3.32) and (3.35) we obtain

$$\begin{aligned}\begin{bmatrix}\hat{x}^{(i)}(t)\\ \hat{p}^{(i)}(t)\end{bmatrix} = &\begin{bmatrix}\mu_x\left(t_f,t,t_0\right)\\ \mu_p\left(t_f,t,t_0\right)\end{bmatrix}x_0 + \int_{t_0}^{t}\phi(t,\tau)\left[E_1 y^{(i)}(\tau) + g_1\left(y^{(i)}(\tau)\right)\right]d\tau \\ &-\begin{bmatrix}\phi_{12}(t,t_0)\\ \phi_{22}(t,t_0)\end{bmatrix}\%_{22}\left(t_f,t_0\right)^{-1}\left\{\int_{t_0}^{t_f}\bar{\phi}_2(t_f,\tau)\left[E_1 y^{(i)}(\tau) + g_1\left(y^{(i)}(\tau)\right)\right]d\tau \right. \\ &\left. + \bar{\Psi}^*\left(y^{(i)}(t_f)\right)\right\}\end{aligned} \tag{3.49}$$

where

$$\left.\begin{aligned}\mu_x\left(t_f,t,t_0\right) &= \phi_{11}(t,t_0) - \phi_{12}(t,t_0)\%_{22}(t_f,t_0)^{-1}\bar{\phi}_{21}(t_f,t_0)\\ \mu_p\left(t_f,t,t_0\right) &= \phi_{21}(t,t_0) - \phi_{22}(t,t_0)\%_{22}(t_f,t_0)^{-1}\bar{\phi}_{21}(t_f,t_0)\end{aligned}\right\} \tag{3.50}$$

Substituting Eq. (3.49) into Eq. (3.45) then gives

$$\begin{aligned} y^{(i+1)}(t) &= \left[E_2 + I_{2n+m} - K_y \right] y^{(i)}(t) + C\int_{t_0}^{t_f} \Omega\left(t_f, t, t_0, \tau\right) E_1 y^{(i)}(\tau) d\tau \\ &+ C\mu\left(t_f, t, t_0\right) x_0 + g_2\left(y^{(i)}(t)\right) + C\int_{t_0}^{t_f} \Omega\left(t_f, t, t_0, \tau\right) g_1\left(y^{(i)}(\tau)\right) d\tau \\ &+ CP^*\left(y^{(i)}\left(t_f\right), t, t_0\right) \end{aligned} \tag{3.51}$$

Eq. (3.51) is what we are looking for, the definition of a recursive relationship for $y^{(i+1)}(t)$ in terms of $y^{(i)}(t)$ where

$$\begin{aligned} &\mu\left(t_f, t, t_0\right) = \begin{bmatrix} \mu_x\left(t_f, t, t_0\right) \\ \mu_p\left(t_f, t, t_0\right) \end{bmatrix}, \quad \eta\left(t_f, t, t_0\right) = \begin{bmatrix} \phi_{12}(t, t_0) \\ \phi_{22}(t, t_0) \end{bmatrix} \bar{\phi}_{22}\left(t_f, t_0\right)^{-1} \\ &P^*\left(y^{(i)}(t_f), t, t_0\right) = -\eta\left(t_f, t, t_0\right) \bar{\Psi}^*\left(y^{(i)}(t_f)\right) \end{aligned} \tag{3.52}$$

and

$$\Omega(t_f, t, t_0, \tau) = \begin{cases} \phi(t, \tau) - \eta\left(t_f, t, t_0\right) \bar{\phi}_2(t_f, \tau), & t_0 \le \tau \le t \\ -\eta\left(t_f, t, t_0\right) \bar{\phi}_2(t_f, \tau), & t \le \tau \le t_f \end{cases} \tag{3.53}$$

The mapping in Eq. (3.51) can be decomposed into

$$y^{(i+1)}(t) = \bar{y}^{(i+1)}(t) + \tilde{y}^{(i+1)}(t) \tag{3.54}$$

where

$$\bar{y}^{(i+1)}(t) = \left[E_2 + I - K_y \right] y^{(i)}(t) + C\int_{t_0}^{t_f} \Omega(t_f, t, t_0, \tau) E_1 y^{(i)}(\tau) d\tau + C\mu(t_f, t, t_0) x_0 \tag{3.55}$$

is a linear term, and

$$\tilde{y}^{(i+1)}(t) = g_2(y^{(i)}(t)) + C\int_{t_0}^{t_f} \Omega(t_f, t, t_0, \tau) g_1\left(y^{(i)}(\tau)\right) d\tau + CP^*(y^{(i)}(t_f), t, t_0) \tag{3.56}$$

represents a non-linear contribution.

3.6 The Optimality Analysis

To do this analysis, the algorithm mapping given by Eq. (3.45) is interpreted as a unit memory repetitive process. A limit profile is taken of the unit memory repetitive process represented by a non-linear differential form. An optimality

theorem is proven based on the limit profile.

3.6.1 A Unit Memory Repetitive Process Interpretation

A unit memory repetitive process can clearly represent the algorithm mapping defined by Eq. (3.51). As described in Chapter 2, it falls naturally into the area of 2-D systems where the pass profile $Y^{(i)}(t)$ is $Y^{(i)}(t) = \left[u^{(i)}(t)^T \quad x^{(i)}(t)^T \quad p^{(i)}(t)^T \right]^T$ consisting of the control, state and co-state signal estimates at each iteration. The pass length ς is the time horizon $t_f - t_0$, and the boundary conditions are x_0 and $y^{(i)}(t_f)$. Here the Banach space $E_\varsigma = C_{2n+m}(t_0, t_f)$ of bounded mappings of the interval $t_0 \le t \le t_f$ into the vector space ¡ $^{2n+m}$ with norm

$$\left\| Y^{(i+1)} \right\| = \sup_{t_0 \le t \le t_f} \left\| Y(t) \right\|_{2n+m} \tag{3.57}$$

where $\|\cdot\|_{2n+m}$ is any convenient norm in ¡ $^{2n+m}$. This link to 2-D systems are crucial since as mentioned in Subsection 2.8.1, the 2-D systems theory can be used to analyze the local stability and convergence behavior of the multipass process.

By looking at Eqs. (3.51), (3.55), and (3.56) we can see that the algorithm mapping can be written in the form:

$$Y^{(i+1)} = \phi_1^{(i+1)} Y^{(i)} + \phi_2^{(i+1)} \left(Y^{(i)} \right), i \ge 0 \tag{3.58}$$

with

$$\left.\begin{aligned} \phi_1^{(i+1)} Y^{(i)} &= \left[E_2 + I_{2n+m} - K_y \right] Y^{(i)}(t) + C \int_{t_0}^{t_f} \Omega\left(t_f, t, t_0, \tau\right) E_1 Y^{(i)}(\tau) d\tau + C\mu\left(t_f, t, t_0\right) x_0 \\ \phi_2^{(i+1)} \left(Y^{(i)} \right) &= g_2\left(Y^{(i)}(t) \right) + C \int_{t_0}^{t_f} \Omega\left(t_f, t, t_0, \tau\right) g_1\left(Y^{(i)}(\tau) \right) d\tau + CP^*\left(Y^{(i)}(t_f), t, t_0 \right) x_0 \end{aligned}\right\} \tag{3.59}$$

where $\phi_1^{(i+1)} Y^{(i)}$ consists of linear terms and $\phi_2^{(i+1)} \left(Y^{(i)} \right)$ is a nonlinear contribution.

The unit memory repetitive process representation of the algorithm can also be written in a non-linear differential form by defining

$X^{(i)}(t) = \left[\hat{x}^{(i)}(t)^T \quad \hat{p}^{(i)}(t)^T \right]^T$. Then Eqs. (3.27) and (3.45) provide

$$\left.\begin{aligned} \frac{d}{dt} X^{(i)}(t) &= HX^{(i)}(t) + E_1 Y^{(i)}(t) + g_1\left(Y^{(i)}(t)\right) \\ Y^{(i+1)}(t) &= CX^{(i)}(t) + \left[E_2 + I_{2n+m} - K_y \right] Y^{(i)}(t) + g_2\left(Y^{(i)}(t)\right) \end{aligned}\right\} \tag{3.60}$$

where the initial condition $X^{(i)}(t_0) = \left[\hat{x}^{(i)}(t_0)^T \quad \hat{p}^{(i)}(t_0)^T \right]^T = \left[x_0^T \quad \hat{p}^{(i)}(t_0)^T \right]^T$

with $\hat{p}^{(i)}(t_0)$ given by

$$\begin{aligned} \hat{p}^{(i)}(t_0) = -\%_{22}(t_f, t_0)^{-1} \{ & \bar{\phi}_{21}(t_f, t_0) x_0 + \bar{\Psi}^*(Y^{(i)}(t_f)) \\ & + \int_{t_0}^{t_f} \bar{\phi}_2(t_f, t_0) \left[E_1 Y^{(i)}(\tau) + g_1\left(Y^{(i)}(\tau)\right) \right] d\tau \} \end{aligned} \tag{3.61}$$

This is an output dependent initial condition and can be written in the form of

$$X^{(i)}(t_0) = \Phi\left(t_f, t_0\right) x_0 + \Psi\left(Y^{(i)}(t_f)\right) + \int_{t_0}^{t_f} J\left(Y^{(i)}(\tau)\right) d\tau \tag{3.62}$$

where

$$\Phi\left(t_f, t_0\right) = \begin{bmatrix} I_n \\ -\%_{22}\left(t_f, t_0\right)^{-1} \bar{\phi}_{21}\left(t_f, t_0\right) \end{bmatrix} \tag{3.63}$$

$$\Psi\left(Y^{(i)}(t_f)\right) = \begin{bmatrix} O_n \\ -\%_{22}\left(t_f, t_0\right)^{-1} \bar{\Psi}^*\left(Y^{(i)}(t_f)\right) \end{bmatrix} \tag{3.64}$$

and

$$J\left(Y^{(i)}(\tau)\right) = \begin{bmatrix} O_n \\ -\%_{22}(t_f, t_0)^{-1} \bar{\phi}_2(t_f, t_0) \left[E_1 Y^{(i)}(\tau) + g_1\left(Y^{(i)}(\tau)\right) \right] \end{bmatrix} \tag{3.65}$$

where $\bar{\Psi}^*(Y^{(i)}(t_f))$ is defined in Eq. (3.36). Note that the initial condition is a nonlinear function of the output $Y^{(i)}(t)$. Note also that the non-linear contribution to (3.60) and the initial condition of (3.62) are completely contained in the components $g_1\left(Y^{(i)}(t)\right)$, $g_2\left(Y^{(i)}(t)\right)$ and $\bar{\Psi}^*\left(Y^{(i)}(t_f)\right)$.

3.6.2 DISOPE as a Linear Multipass Process

Linear multipass processes have properties that are desirable to our analyses. Thus we proceed with transforming the nonlinear process described above into a linear process with the help of LQR model. The original non-linear optimal control problem ROP may be considered as linear with a quadratic performance index and a linear terminal constraint when LQR is used as model. Thus in Eq. (3.1) we let

$$\left.\begin{aligned} &\Phi^*(x(t_f)) = \tfrac{1}{2}x(t_f)S^*x(t_f) \\ &L^*(x(t),u(t)) = \tfrac{1}{2}\left(x(t)^T Q^* x(t) + u(t)^T R^* u(t)\right) \\ &f^*(x(t),u(t)) = A^*x(t) + B^*u(t) \\ &\Psi^*(x(t_f)) = V^*x(t_f) + b^* \end{aligned}\right\} \tag{3.66}$$

Without loss of generality, we will consider a zero initial condition x_0, and in Eq. (3.15) and (3.66) $b = O_{q,1}$ and $b^* = O_{q,1}$.

The expressions $g_1\left(Y^{(i)}(t)\right)$ and $g_2\left(Y^{(i)}(t)\right)$, which represent the non-linear contribution to Eq. (3.60) can be transformed into linear expressions as follows. Using Eq. (3.22), (3.29), (3.30), (3.47), and (3.48), together with (3.66), $g_1\left(Y^{(i)}(t)\right)$ given by Eq. (3.29) as

$$g_1\left(Y^{(i)}(t)\right) = \begin{bmatrix} g_{11}\left(y^{(i)}(t)\right) \\ g_{21}\left(y^{(i)}(t)\right) \end{bmatrix} \tag{3.67}$$

where

$$\left.\begin{aligned} g_{11}\left(y^{(i)}(t)\right) &= B\bar{R}^{-1}\left([B - B^*(x^{(i)},u^{(i)},t)]^T \hat{p}^{(i)}(t) + Ru^{(i)}(t) - L_u^*(x^{(i)},u^{(i)},t)\right) \\ &\quad + f(x^{(i)},u^{(i)},t) - Ax^{(i)}(t) - Bu^{(i)}(t) \\ g_{21}\left(y^{(i)}(t)\right) &= \left([A - A^*(x^{(i)},u^{(i)},t)]^T \hat{p}^{(i)}(t) + Qx^{(i)}(t) - L_u^*(x^{(i)},u^{(i)},t)\right) \end{aligned}\right\} \tag{3.68}$$

becomes

$$\left.\begin{aligned} g_{11}\left(y^{(i)}(t)\right) &= B\bar{R}^{-1}\left([B - B^*]^T \hat{p}^{(i)}(t) + Ru^{(i)}(t) - R^*u^{(i)}(t)\right) \\ &\quad + A^*x^{(i)}(t) + B^*u^{(i)}(t) - Ax^{(i)}(t) - Bu^{(i)}(t) \\ g_{21}\left(y^{(i)}(t)\right) &= \left([A - A^*]^T \hat{p}^{(i)}(t) + Qx^{(i)}(t) - Q^*x^{(i)}(t)\right) \end{aligned}\right\} \tag{3.69}$$

In other words, Eq. (3.67) is transformed into

$$g_1\left(Y^{(i)}(t)\right)=\begin{bmatrix}\left(\left(B^*-B\right)+B\bar{R}^{-1}(R-R^*)\right) & \left(A^*-A\right) & B\bar{R}^{-1}(B-B^*)^T\\ O_{n,m} & \left(Q-Q^*\right) & \left(A-A^*\right)^T\end{bmatrix}y^{(i)}(t) \qquad (3.70)$$

which is a linear expression.

For the expression of $g_2\left(Y^{(i)}(t)\right)$, which from Eq. (3.47) is

$$g_2(Y^{(i)}(t))=\begin{bmatrix}g_{21}\left(Y^{(i)}(t)\right)\\ O_{n,2n+m}\\ O_{n,2n+m}\end{bmatrix} \qquad (3.71)$$

with $g_{21}\left(Y^{(i)}(t)\right)$ as

$$g_{21}\left(Y^{(i)}(t)\right)=k_u\bar{R}^{-1}\left\{\left[B-B^*(x^{(i)},u^{(i)},t)\right]^T\ p^{(i)}(t)+\left[Ru^{(i)}(t)-L_u^*(x^{(i)},u^{(i)},t)\right]\right\} \qquad (3.72)$$

the expression becomes

$$g_{21}=k_u\bar{R}^{-1}\left\{\left(B-B^*\right)^T\ p^{(i)}(t)+\left(Ru^{(i)}(t)-R^*u^{(i)}(t)\right)\right\} \qquad (3.73)$$

Hence with (3.73), $g_2\left(Y^{(i)}(t)\right)$ can be rewritten as a linear expression of

$$g_2(Y^{(i)}(t))=\begin{bmatrix}k_u\bar{R}^{-1}\left(R-R^*\right) & O_{m,n} & k_u\bar{R}^{-1}\left(B-B^*\right)^T\\ O_{n,m} & O_n & O_n\\ O_{n,m} & O_n & O_n\end{bmatrix}Y^{(i)}(t) \qquad (3.74)$$

Consequently, Eq. (3.60) can be written in the *linear* unit memory repetitive process form

$$\left.\begin{aligned}\tfrac{d}{dt}X^{(i)}(t)&=HX^{(i)}(t)+B_0Y^{(i)}(t)\\ Y^{(i+1)}(t)&=CX^{(i)}(t)+D_1Y^{(i)}(t)\end{aligned}\right\} \qquad (3.75)$$

where from (3.60) $B_0Y^{(i)}(t)=E_1Y^{(i)}(t)+g_1\left(Y^{(i)}(t)\right)$ or alternatively,

$$B_0=\begin{bmatrix}r_1B\bar{R}^{-1}+\left(B^*-B\right)+B\bar{R}^{-1}\left(R-R^*\right) & A^*-A & B\bar{R}^{-1}\left(B-B^*\right)^T\\ O_{n,m} & r_2I_n+\left(Q-Q^*\right) & \left(A-A^*\right)^T\end{bmatrix} \qquad (3.76)$$

This simplifies into

$$B_0=\begin{bmatrix}\left(B^*-B\bar{R}^{-1}R^*\right) & \left(A^*-A\right) & B\bar{R}^{-1}(B-B^*)^T\\ O_{n,m} & \left(\bar{Q}-Q^*\right) & \left(A-A^*\right)^T\end{bmatrix} \qquad (3.77)$$

And $D_1 Y^{(i)}(t) = \left[E_2 + I_{2n+m} - K_y \right] Y^{(i)}(t) + g_2\left(Y^{(i)}(t)\right)$ is now given by

$$D_1 = \begin{bmatrix} r_1 k_u \bar{R}^{-1} + I_m - k_u I_m + k_u \bar{R}^{-1}\left(R - R^*\right) & O_{m,n} & k_u \bar{R}^{-1}\left(B - B^*\right)^T \\ O_{n,m} & I_n - k_x I_n & O_n \\ O_{n,m} & O_n & I_n - k_p I_n \end{bmatrix} \quad (3.78)$$

which simplifies into

$$D_1 = \begin{bmatrix} I_m - k_u \bar{R}^{-1} R^* & O_{m,n} & k_u \bar{R}^{-1}(B - B^*)^T \\ O_{n,m} & (1 - k_x) I_n & O_n \\ O_{n,m} & O_n & (1 - k_p) I_n \end{bmatrix} \quad (3.79)$$

H and *C* are as defined in (3.29) and (3.46) respectively.

Using (3.64) and (3.65), together with (3.36), the initial condition, Eq. (3.62) can be written as

$$X^{(i)}(t_0) = \tilde{E}_0 Y^{(i)}(t_f) + \int_{t_0}^{t_f} J(t_f, t_0, \tau) Y^{(i)}(\tau) d\tau \quad (3.80)$$

where

$$J(t_f, t_0, \tau) = \begin{bmatrix} O_{n,2n+m} \\ -\tilde{\phi}_{22}(t_f, t_0)^{-1} \bar{\phi}_2(t_f, \tau) B_0 \end{bmatrix} \quad (3.81)$$

and

$$\tilde{E}_0 = \begin{bmatrix} O_{n,2n+m} \\ -\tilde{\phi}_{22}(t_f, t_0)^{-1} E_0 \end{bmatrix} \quad (3.82)$$

where

$$E_0 = \begin{bmatrix} O_{n,m} & \left(-\left(S^* - S\right) + V^{*T} \sum (t_f, t_0)^{-1} \bar{V}(t_f, t_0)\right) & O_n \end{bmatrix} \quad (3.83)$$

with

$$\sum (t_f, t_0) = V \phi_{12}(t_f, t_0) \tilde{\phi}_{22}(t_f, t_0)^{-1} V^{*T} \quad (3.84)$$

$$\bar{V}(t_f, t_0) = V^* - V + V \phi_{12}(t_f, t_0) \tilde{\phi}_{22}(t_f, t_0)^{-1} \left(S^* - S\right) \quad (3.85)$$

Eq. (3.75) is a linear unit memory repetitive process with output dependent initial condition given by Eq. (3.80).

3.6.3 The Optimality Conditions of ROP and Its Linear Unit Memory Repetitive Process Form

In Subsection 3.6.2, we transformed the nonlinear unit memory repetitive process form of ROP into a linear form. It is crucial that we ensure the linear form satisfy all the necessary optimality conditions of ROP before we proceed. We achieve this by using the property of limit profile of the linear unit memory repetitive process. In the following analysis, we assumed the existence of the limit point of the sequence of terms generated by DISOPE for the limit profile to be valid.

From Subsection 2.8.4, we note that two important properties of linear repetitive processes are stability and the limit profile. By assuming that Theorem 2.1 is satisfied, that is the linear repetitive process defined by Eq. (3.75) is asymptotically stable, Definition 2.3 give the limit profile as

$$Y^{(\infty)} = \lim_{i \to \infty} Y^{(i)} \tag{3.86}$$

and by Theorem 2.2, the limit profile is the unique solution to Eq. (3.75). In this particular situation, for a stable system, the limit profile is $X^{(\infty)} = O_{2n,1}$ and $Y^{(\infty)} = O_{2n+m,1},\ t \in \left[t_0, t_f \right]$ and Eq. (3.66) represents a local linear-quadratic perturbation analysis of Eq. (3.1) about this zero point.

Application to (3.60) shows that $X^{(\infty)}(t)$ and $Y^{(\infty)}(t)$ satisfy the relationships:

$$\tfrac{d}{dt} X^{(\infty)}(t) = HX^{(\infty)}(t) + E_1 Y^{(\infty)}(t) + g_1\left(Y^{(\infty)}(t)\right) \tag{3.87}$$

$$CX^{(\infty)}(t) + \left[E_2 - K_y \right] Y^{(\infty)}(t) + g_2\left(Y^{(\infty)}(t)\right) = 0 \tag{3.88}$$

with Eq. (3.87) being the differential equation portion of the limit profile and Eq. (3.88) is the algebraic portion. This set of two equations is called the limit profile of the linear unit memory repetitive process representation of DISOPE. The initial conditions previously given by (3.62) is now defined as

$$X^{(\infty)}(t_0) = \Phi\left(t_f, t_0\right) x_0 + \Psi\left(Y^{(\infty)}(t_f)\right) + \int_{t_0}^{t_f} J\left(Y^{(\infty)}(\tau)\right) d\tau \tag{3.89}$$

The boundary conditions on Eqs. (3.87) and (3.88) can be written in the alternative two-point form as follows

$$\left.\begin{aligned} & X^{(\infty)}(0) = x_0 \\ & \hat{p}^{(\infty)}(t_f) = \Phi_x^*(x^{(\infty)}(t_f)) + \left[V^*(x^{(\infty)}(t_f))\right]^T \chi^{(\infty)} + S\left(\hat{x}^{(\infty)}(t_f) - x^{(\infty)}(t_f)\right) \\ & \Psi^*\left(x^{(\infty)}(t_f)\right) + V\left(\hat{x}^{(\infty)}(t_f) - x^{(\infty)}(t_f)\right) = O_{2n+m,1} \end{aligned}\right\} \quad (3.90)$$

The following theorem on optimality (Roberts, 2002) verifies that the limit profile of the linear unit memory repetitive process representation of DISOPE satisfies all the necessary optimality conditions of ROP.

Theorem 3.1 – Optimality

Let Eq. (3.75) be the linear unit memory repetitive process representation of DISOPE algorithm. If it is asymptotically stable then its limit profile given by Eqs. (3.87) and (3.88) satisfy the necessary optimality conditions of the nonlinear optimal control problem ROP defined by Eq. (3.1).

Proof

Using the notation of Eq. (3.22), the optimal solution $u^o(t)$, $x^o(t)$, $p^o(t)$, of ROP, with Lagrange multiplier χ^o, satisfies the necessary optimal conditions (Lewis and Syrmos, 1995) as

$$\left.\begin{aligned} & B^*(x^o,u^o,t)^T p^o(t) + L_u^*(x^o,u^o,t) = O_m \\ & \tfrac{d}{dt} x^o(t) = f^*(x^o,u^o,t) \\ & \tfrac{d}{dt} p^o(t) = -A^*(z^o,v^o,t)^T p^o(t) - L_x^*(x^o,u^o,t) \end{aligned}\right\} \quad (3.91)$$

with boundary conditions

$$\left.\begin{aligned} & x^{(\infty)}(t_0) = x_0 \\ & p^{(\infty)}(t_f) = \Phi_x^*\left(x^{(\infty)}(t_f)\right) + \left[V^*\left(x^{(\infty)}(t_f)\right)\right]^T \chi^{(\infty)} \\ & \Psi^*\left(x^{(\infty)}(t_f)\right) = O_{q,1} \end{aligned}\right\} \quad (3.92)$$

We first analyze the algebraic portion of the limit profile that is Eq. (3.88). From the limit profile point of view, with $X^{(\infty)}(t) = \left[\hat{x}^{(\infty)}(t)^T \quad \hat{p}^{(\infty)}(t)^T\right]^T$,

$Y^{(\infty)}(t) = \left[u^{(\infty)}(t)^T \quad x^{(\infty)}(t)^T \quad p^{(\infty)}(t)^T \right]^T$ and Eqs. (3.35) and (3.40) it is can be shown that as $i \to \infty$, $x^{(\infty)}(t)^T = \hat{x}^{(\infty)}(t)^T$, and $p^{(\infty)}(t)^T = \hat{p}^{(\infty)}(t)^T$. Furthermore,

$$B^*(x^{(\infty)}, u^{(\infty)}, t)^T \, p^{(\infty)}(t) + L_u^*(x^{(\infty)}, u^{(\infty)}, t) = O_{m,1}. \tag{3.93}$$

Hence the first part of Eq. (3.91) is satisfied.

To see if the next two equations in (3.91) are also satisfied, we analyze the differential equation portion of the limit profile in Eq. (3.87). Using the form of Eq. (3.27) together with Eqs. (3.29) and the notations of (3.22) we get

$$\left.\begin{array}{l} \frac{d}{dt} x^{(\infty)}(t) = f^*(x^{(\infty)}, u^{(\infty)}, t) \\ \frac{d}{dt} p^{(\infty)}(t) = -A^*(z^{(\infty)}, v^{(\infty)}, t)^T \, p^{(\infty)}(t) - L_x^*(x^{(\infty)}, u^{(\infty)}, t) \end{array}\right\}. \tag{3.94}$$

From Eqs. (3.93) and (3.94) we see that the optimality conditions of Eq. (3.91) are satisfied.

Finally, the limit profile boundary conditions defined by Eq. (3.90) can be written as

$$\left\{\begin{array}{l} x^{(\infty)}(0) = x_0 \\ p^{(\infty)}(t_f) = \Phi_x^*\left(x^{(\infty)}(t_f)\right) + \left[V^*\left(x^{(\infty)}(t_f)\right)\right]^T \chi^{(\infty)} \\ \Psi^*\left(x^{(\infty)}(t_f)\right) = O_{q,1} \end{array}\right\}. \tag{3.95}$$

Thus by comparing Eq. (3.95) with Eq. (3.92), we conclude that the limit profile solution $u^{(\infty)}(t), x^{(\infty)}(t), p^{(\infty)}(t)$ with Lagrange multiplier $\chi^{(\infty)}$, satisfies all the necessary optimality conditions of ROP. □

Thus with this theorem the algorithm is guaranteed to be able to search and arrived at the optimal solution of the optimal control problem.

3.7 Stability and Convergence Analyses of DISOPE

Stability is a property of the numerical scheme that has nothing to do with the scheme's approximation power. It characterizes the robustness of the scheme with respect to small perturbations. Stability yields convergence of the numerical solution to the true solution (Gautschi, 1997).

As mentioned earlier, 2-D systems theory can be used to analyze the stability of multipass processes (Rogers and Owen, 1992). Definition 2.2 and Theorem 2.1 are the specific properties of asymptotic stability in multipass processes. In the next subsection we will use this theory to analyze this property of DISOPE.

3.7.1 Stability Analysis

The linear unit memory repetitive process defined by Eq. (3.75) with output dependent initial condition given by Eq. (3.80) is assumed to be asymptotically stable in the analysis of Section 3.6 above. For linear constant coefficient systems, stability depends only on the locations of the roots of the closed-loop characteristic equation (Brogan, 1991). In this subsection, we give a theorem (Roberts, 2002) that asserts the stability according to Theorem 2.1 in Chapter 2.

Theorem 3.2 –Stability (Asymptotic)

Let Eq. (3.75) be the linear unit memory repetitive process representation of DISOPE algorithm. If Eq. (3.80) is its output dependent initial condition then its sufficient and necessary condition for asymptotic stability is that all solutions for λ in the relationship:

$$\left| I_{2n} - E_0 F(\lambda) \exp\left\{ \left(H + B_0 F(\lambda) \right)(t_f - t_0) \right\} - \int_{t_0}^{t_f} J(t_f, t_0, \tau) F(\lambda) \exp\left\{ \left(H + B_0 F(\lambda) \right)\tau \right\} d\tau \right| = 0 \tag{3.96}$$

lie in the unit circle in the complex plane, where

$$F(\lambda) = \left[\lambda I_{2n+m} - D_1\right]^{-1} C \tag{3.97}$$

Proof

Roberts (2000) proved the theorem and the proof is restated here for convenience.

In bounded linear operator form, we can define:

$$\left.\begin{aligned} \dot{x}(t) &= Hx(t) + B_0 y(t); x(t_0) = E_0 y(t_f) + \int_{t_0}^{t_f} J(t_f, t_0, \tau) y(\tau) d\tau \\ \lambda y(t) &= Cx(t) + D_1 y(t) \end{aligned}\right\} \tag{3.98}$$

Then

$$\left.\begin{aligned} y(t) &= [\lambda I_{2n+m} - D_1]^{-1} C x(t) \\ \dot{x}(t) &= [H + B_0 [\lambda I_{2n+m} - D_1]^{-1} C] x(t) = [H + B_0 F(\lambda)] x(t) \end{aligned}\right\} \tag{3.99}$$

giving

$$\left.\begin{aligned} x(t) &= \exp\left((H + B_0 F(\lambda))t\right) x(0) \\ y(t) &= F(\lambda) \exp\left((H + B_0 F(\lambda))t\right) x(0) \end{aligned}\right\} \tag{3.100}$$

Hence from (3.80) and (3.100)

$$\begin{aligned} x_0 = {} & E_0 F(\lambda) \exp\left((H + B_0 F(\lambda))t_f\right) x(0) \\ & + \int_{t_0}^{t_f} J(t_f, t_0, \tau) F(\lambda) \exp\left((H + B_0 F(\lambda))\tau\right) d\tau x(0) \end{aligned} \tag{3.101}$$

That is by rearranging,

$$\begin{aligned} \Big(I_{2n} - {} & E_0 F(\lambda) \exp\left((H + B_0 F(\lambda))t_f\right) \\ & - \int_{t_0}^{t_f} J(t_f, t_0, \tau) F(\lambda) \exp\left((H + B_0 F(\lambda))\tau\right) d\tau \Big) x(0) = 0 \end{aligned} \tag{3.102}$$

Thus the spectral values of λ are given by

$$\left| \begin{aligned} I_{2n} - {} & E_0 F(\lambda) \exp\left((H + B_0 F(\lambda))t_f\right) \\ & - \int_{t_0}^{t_f} J(t_f, t_0, \tau) F(\lambda) \exp\left((H + B_0 F(\lambda))\tau\right) d\tau \end{aligned} \right| = 0 \tag{3.103}$$

Hence, by Theorem 2.1, asymptotic stability occurs if and only if the solutions of λ in Eq. (3.103) lie in the unit circle in the complex plane. □

Thus the algorithm of DISOPE in the form of a linear unit memory repetitive

process is asymptotically stable whenever the condition in Theorem 3.2 is satisfied.

Stability is a powerful concept; it implies almost immediately convergence. With the property of stability at hand, the convergence of the algorithm follows. In the next subsection we give the convergence analysis of the algorithm.

3.7.2 Convergence Analysis

The convergence behavior of DISOPE may be investigated from the nonlinear unit memory representation of the algorithm (Roberts, 2002). Before going any further, we first state the following definition of contraction mapping for use later in the convergence theorem.

Definition 3.1 – Contraction Mapping

Consider the fixed-point iteration

$$y^{(i+1)} = \phi(y^{(i)}),\ i = 0,1,2,\mathrm{L}\ . \tag{3.104}$$

We say that $\phi : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ is a contraction map (or contractive) on a set $\mathcal{D} \subseteq \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ if there exist a constant θ with $0 < \theta < 1$ such that, in some appropriate vector norm

$$\left\| \phi(y) - \phi(y^*) \right\| \le \theta \left\| y - y^* \right\| \text{ for all } y, y^* \in \mathcal{D} \tag{3.105}$$

Next we state the theorem stating the contraction mapping principle, in which the existence of a limit point and hence convergence of an algorithm is implied.

Theorem 3.3 – Contraction Mapping Principle

Let $\mathcal{D} \subseteq \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ be a complete subset of $\mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$. If $\phi : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$ is contractive in the sense of (3.105) and maps $\mathcal{D}$ into $\mathcal{D}$, then the iteration in (3.104) is well defined and converges to a unique limit point $\hat{y} \in \mathcal{D}$, such that $\lim_{i \to \infty} y^{(i)} = \hat{y}$.

Proof (See Gautschi (1997))

Based on Definition 3.1 and Theorem 3.3, we use Eq. (3.51) to investigate the convergence behavior of DISOPE. From Eq. (3.51), successive terms are used to determine a contraction condition in the form of

$$\left\| y^{(i+1)}(t) - y^{(i)}(t) \right\| \le \theta_1 \left\| y^{(i)}(t) - y^{(i-1)}(t) \right\| \tag{3.106}$$

such that (3.105) is satisfied, where

$$\left\| y(t) \right\| = \sup_t \left\| y(t) \right\|_W , \; t \in \left[t_0, t_f \right] \tag{3.107}$$

with

$$\left\| y(t) \right\|_W = \left(\left| y_1(t) \right|^W + \left| y_2(t) \right|^W \mathrm{L} \; + \left| y_{2n+m}(t) \right|^W \right)^{1/W} , \; W \in [1, \infty] \tag{3.108}$$

together with the following Lipschitz continuity assumptions

$$\left.\begin{aligned} &\left\| g_1(y^{(i)}(t)) - g_1(y^{(i-1)}(t)) \right\| \le h_1 \left\| y^{(i)}(t) - y^{(i-1)}(t) \right\| \\ &\left\| g_2(y^{(i)}(t)) - g_2(y^{(i-1)}(t)) \right\| \le h_2 \left\| y^{(i)}(t) - y^{(i-1)}(t) \right\| \\ &\left\| P^*\left(y^{(i)}(t_f), t, t_0 \right) - P^*\left(y^{(i-1)}(t_f), t, t_0 \right) \right\| \le h_3 \left\| y^{(i)}(t) - y^{(i-1)}(t) \right\| \end{aligned}\right\} \tag{3.109}$$

where h_1, h_2 and h_3 are the Lipschitz constants. The convergence conditions are then given by the following theorem.

Theorem 3.4 – Convergence

A sufficient condition for the algorithm mapping represented by Eq. (3.51) to exhibit asymptotic convergence according to (3.106) for every iteration $i > 1$ is given by the expression

$$\theta_1 = \left\| E_2 + I_{2n+m} - K_y \right\| + h_2 + \left(h_3 + \left(\left\| E_1 \right\| + h_1 \right) \sigma\left(t_0, t_f \right) \right) \left\| C \right\| \le 1 \tag{3.110}$$

where

$$\sigma\left(t_0, t_f \right) = \sup_{t \in [t_0, t_f]} \int_{t_0}^{t_f} \left\| \Omega\left(t_f, t, t_0, \tau \right) \right\| d\tau \tag{3.111}$$

Proof

Taking the norm of two successive iterations from Eq. (3.51) and taking the norm produces

$$\begin{aligned}
&\left\|y^{(i+1)}(t)-y^{(i)}(t)\right\| \le \left\|E_2+I_{2n+m}-K_y\right\|\left\|y^{(i)}(t)-y^{(i-1)}(t)\right\| \\
&+\|C\|\|E_1\|\int_{t_0}^{t_f}\left\|\Omega\left(t_f,t,t_0,\tau\right)\right\|\left\|y^{(i)}(\tau)-y^{(i-1)}(\tau)\right\|d\tau+\left\|g_2(y^{(i)}(t))-g_2(y^{(i-1)}(t))\right\| \\
&+\|C\|\int_{t_0}^{t_f}\left\|\Omega\left(t_f,t,t_0,\tau\right)\right\|\left\|g_1(y^{(i)}(\tau))-g_1(y^{(i-1)}(\tau))\right\|d\tau \\
&+\|C\|\left\|P^*\left(y^{(i)}(t_f),t,t_0\right)-P^*\left(y^{(i-1)}(t_f),t,t_0\right)\right\|
\end{aligned} \tag{3.112}$$

By assuming Lipschitz continuity as defined by Eq. (3.109) we obtain

$$\begin{aligned}
&\left\|y^{(i+1)}(t)-y^{(i)}(t)\right\| \le \left(\left\|E_2+I_{2n+m}-K_y\right\|+h_2+h_3\|C\|\right)\left\|y^{(i)}(t)-y^{(i-1)}(t)\right\| \\
&+\left(\|E_1\|+h_1\right)\|C\|\left\|y^{(i)}(t)-y^{(i-1)}(t)\right\|\int_{t_0}^{t_f}\left\|\Omega\left(t_f,t,t_0,\tau\right)\right\|d\tau
\end{aligned} \tag{3.113}$$

By using (3.111) then gives the contraction equation

$$\begin{aligned}
&\left\|y^{(i+1)}(t)-y^{(i)}(t)\right\| \le \left(\left\|E_2+I_{2n+m}-K_y\right\|+h_2\right. \\
&\left.+\left(h_3+\left(\|E_1\|+h_1\right)\sigma\left(t_0,t_f\right)\right)\|C\|\right)\left\|y^{(i)}(t)-y^{(i-1)}(t)\right\|
\end{aligned} \tag{3.114}$$

and, hence, the iterations will contract asymptotically according to (3.106) if (3.110) holds.□

It is important to note that Theorem 3.4 provides a sufficient condition only. It is not necessary to satisfy the inequality given by (3.110) for the algorithm to converge. However, the iterations are guaranteed to contract if the condition is satisfied (Roberts, 2002).

Hence with Theorems 3.2 and 3.4 the stability and convergence of DISOPE are established. In the next section we present three numerical examples solved using DISOPE algorithm. All the simulations in this chapter and subsequent chapters were done on a Mobile Intel Pentium 4-M computer with a 2.00 GHz CPU and 192 MB of RAM. MATLAB 6.0 was used as the programming language.

3.8 Numerical Examples

The unique feature of DISOPE is its integration of parameters from both the real problem and its model before arriving at the optimal solution. In view of this,

DISOPE converges in just one iteration whenever no model-reality difference is introduced. When the real problems are modeled as LQR problem, it is customary to use the appropriate identity matrices for *Q* and *R*; the weights for the performance index, although other choices of *Q* and *R* are also acceptable.

DISOPE requires an initial solution to start the iterations. A recommended one is the solution of the relaxed MMOP with the continuous parameter estimate $\alpha(t) = 0$ and the given scalar convexification factors $r_1 = r_2 = 0$. It is based on the steady state solution of the LQR problem. It solves the algebraic Riccati equation instead of the differential Riccati equation for continuous optimization problem. The initial solution is also called a nominal solution. With this nominal solution, DISOPE uses repeated solutions of optimization and estimation of parameters within the model for calculating the optimum.

The relaxation gains $k_v, k_z,$ and k_p and the convexification factors r_1 and r_2 are provided to regulate stability and convergence. In the initial application of the algorithm to a given problem, the relaxation gains $k_v, k_z,$ and k_p would be set to unity and r_1 and r_2 would both be set to zero. These parameters would only be adjusted if convergence difficulties arise (Roberts, 1993).

The following examples are chosen to demonstrate the outcome of DISOPE with different choices of input to the weights and parameters mentioned above. These examples have different levels of difficulties and nonlinearities. These results will later be used as bases for comparisons for the two new algorithms in Chapters 5 and 6.

Example 3.1- Continuous Stirred Tank

Consider the continuous stirred tank reactor problem taken from Kirk (1970). The real optimization problem (ROP) is as follows:

$$\min_{u(x)} \; J^* = \int_0^{0.78} (x_1^2 + x_2^2 + 0.1u^2)dt$$

subject to

$$\dot{x}_1 = -(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - (1+u)(x_1 + 0.25)$$

$$\dot{x}_1 = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right)$$

$$x(0) = [0.05 \;\; 0]^T$$

The modified model (MOP) used in DISOPE algorithm is:

$$\min_{u(x)} \; J = \int_0^{0.78} (x^T Q x + u^T R u + \gamma(t))dt$$

subject to

$$\dot{x} = \begin{bmatrix} 4.25 & 1 \\ -6.25 & -2 \end{bmatrix} x(t) + \begin{bmatrix} -0.25 \\ 0 \end{bmatrix} u(t) + \alpha(t)$$

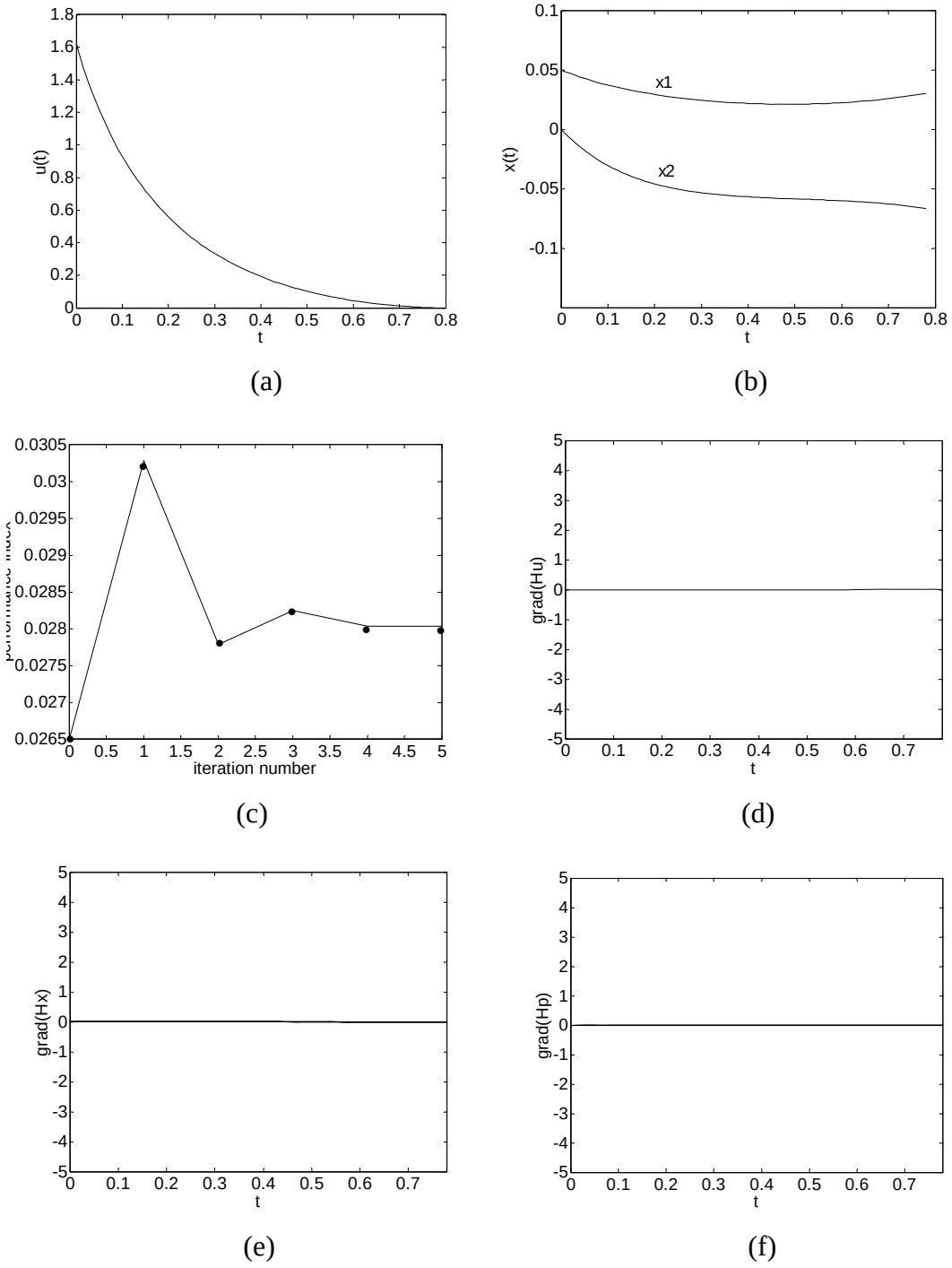
$$x(0) = [0.05 \;\; 0]^T$$

where $x(t) \in ¡^{2}$, $u(t) \in ¡$, $\gamma(t) \in ¡$, and $\alpha(t) \in ¡^{2}$. In this example, $Q = 2I_2$ and $R = 0.2$. The numerical integration step used was h = 0.01 and a tolerance of 0.01 was specified for convergence. The results of this simulation are given in Table 3.1. Table 3.1 showcases the performance of the algorithm with different values chosen for the relaxation gains $k_v, k_z,$ and k_p and the convexification factors r_1 and r_2.

Table 3.1: The results of the algorithm's performance with different values of the relaxation gains and convexification factors.

case	r_1	r_2	k_v	$k_z = k_p$	No. of iterations	J^*	CPU Time (s)
i	0	0	1	1	10	0.028	1.832
ii	0	0	0.3	1	9	0.028	1.683
iii	0.5	0	1	1	9	0.028	1.733
iv	1	0	1	1	12	0.028	2.173
v	1	0	0.8	1	13	0.028	2.444
vi	0	0	0.8	1	9	0.028	1.762
vii	0.1	0	0.8	1	5	0.028	0.991

Figure 3.2 below illustrates the final responses of DISOPE for case (vii). Figs. 3.2(a), (b), and (c) graph the trajectories of the optimal control, the states, and the performance index, respectively.



(a) (b) (c) (d) (e) (f)

Figure 3.2: The final responses of DISOPE for Example 3.1(vii).

The remaining three figures, Figs. 3.2(d), (e), and (f), illustrate the optimality conditions with respect to control $u(t)$, the states $x(t)$, and the costates $p(t)$. It can be seen from the last three figures that the solution satisfies all three of the optimality conditions given by Eq. (3.10).

Example 3.2 - The third order nonlinear system

Consider the third order nonlinear system problem taken from Becerra (1994). The real optimization problem (ROP) is as follows:

$$\min_{u(x)} \; J^* = \int_0^2 (x_1^4 + x_2^2 + x_3^2 + u_1^2 + u_2^6)dt$$

subject to

$$\dot{x}_1 = -x_1 + x_1 x_2 + u_1$$

$$\dot{x}_2 = x_1 - 2x_2 + x_3^3$$

$$\dot{x}_3 = -3x_3 + x_2 + \sin(u_2)$$

$$x(0) = [1.2 \; 0.0 \; 1.0]^{\mathrm{T}}; \quad x_1(2) = 0$$

The modified model (MOP) used in the DISOPE algorithm follows:

$$\min_{u(x)} \; J = \int_0^2 (x_1^2 + x_2^2 + x_3^2 + u_1^2 + u_2^2)dt$$

subject to

$$\dot{x} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -2 & 3 \\ 0 & 1 & -3 \end{bmatrix} x(t) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} u(t) + \alpha(t)$$

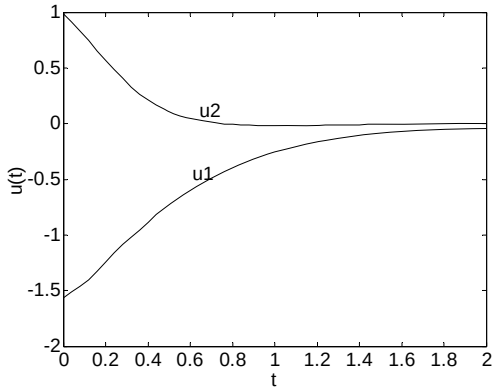
$$x(0) = [1.2 \; 0.0 \; 1.0]^{\mathrm{T}}; \quad x_1(2) = 0$$

The values for Q and R are taken to be $2I_3$ and $2I_2$ respectively. The numerical integration step used was h = 0.04 and a tolerance of 0.01 was specified for convergence. The results of this simulation are given in Table 3.2 below.

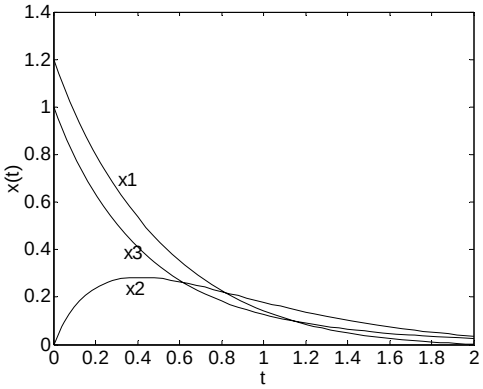
Table 3.2 The algorithm's performance for Example 3.2.

case	r_1	r_2	k_u	k_x	k_p	No. of iterations	J^*	CPU Time (s)
i	0	1	1	0.3	1	12	0.663	2.794
ii	0	1	1	0.4	1	10	0.660	2.344
iii	0	1	1	0.5	1	8	0.659	1.973
iv	0	1	1	0.7	1	8	0.656	1.912
v.	0	1	1	0.8	1	9	0.656	2.244

Figure 3.3 below demonstrates the final responses of Example 3.2 in case (iv). Figs. 3.3(a), (b), and (c) graph the trajectories of the optimal control, the states, and the performance index, respectively. Fig. 3.3 (b) indicates that the solution satisfies the terminal condition of $x_1(2) = 0$. The remaining three figures, Figs. 3.3(d), (e), and (f), illustrate the optimality conditions with respect to control $u(t)$, the states $x(t)$, and the costates $p(t)$. It can be seen from the last three figures that the solution satisfies all three of the optimality conditions given by Eq. (3.10). In this example specifically, which has higher degree of nonlinearity than the first example, these observations are crucial. This is so because some of the solutions found in the simulations have the tendency of not satisfying either the terminal condition or the optimality conditions. These are examples of instances of false optima.



(a)



(b)

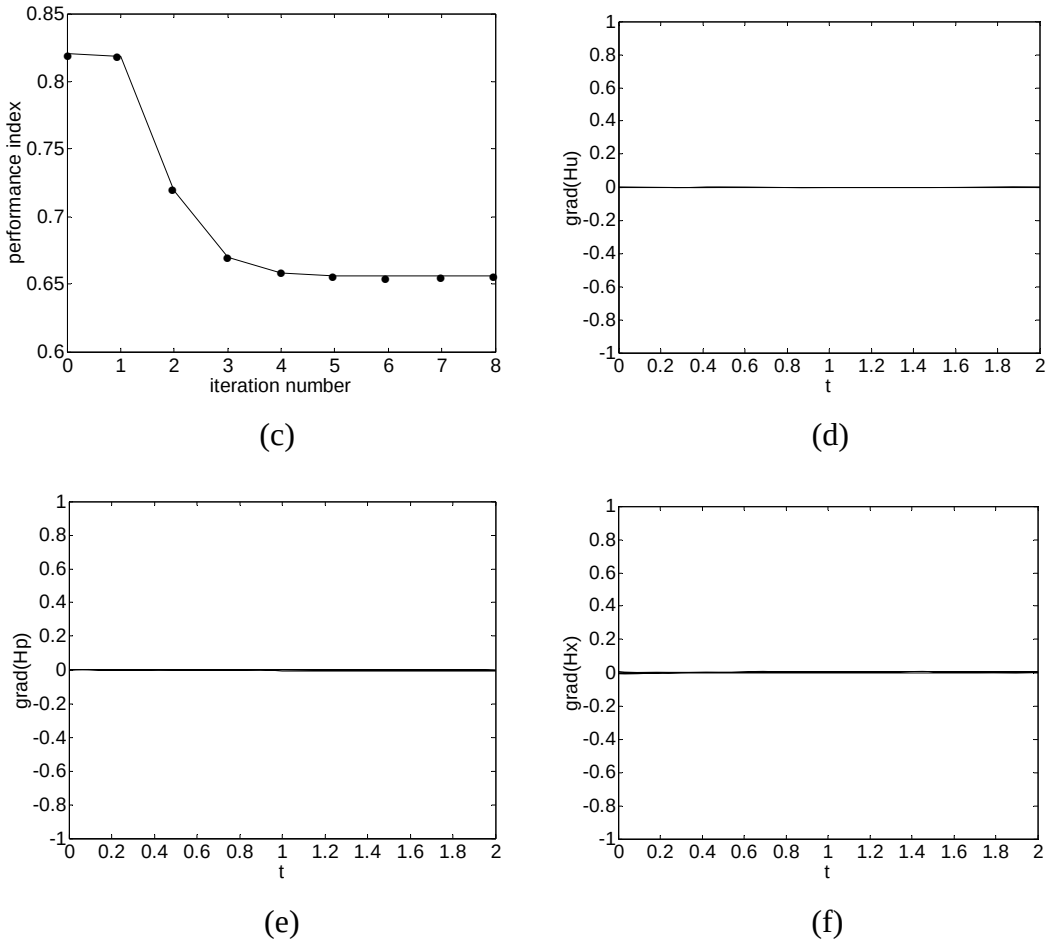


Figure 3.3: The final responses of DISOPE for Example 3.2(iv).

Example 3.3 -Robotic

Consider a fourth order non-linear system representing a horizontal planar revolute/prismatic two degrees of freedom robot manipulator originally from Craig (1989) and modeled in the state-space form by Roberts (1993). ROP is defined as:

$$\min_{u(t)} \tfrac{1}{2}\int_{t_0}^{4}[x_1^2 + x_2^4 + x_3^2 + x_4^4 + u_1^2 + 0.1u_2^4]dt$$

subject to

$$\dot{x}_1 = x_2; x_1(0) = 2, x_1(4) = 0$$
$$\dot{x}_2 = \frac{u_1 - 4x_2x_4(x_3 + 0.5)}{1 + 2(x_3 + 0.5)}; x_2(0) = 0, x_2(4) = 0$$
$$\dot{x}_3 = x_4; x_3(0) = 1, x_3(4) = 0$$
$$\dot{x}_4 = (x_3 + 0.5)x_2^2 + 0.5u_2; x_4(0) = 0, x_4(4) = 0$$

where $x_1(t)$ and $x_2(t)$ are the angular position and velocity of link 1, $x_3(t)$ and $x_4(t)$

are the angular position and velocity of the prismatic link 2. $u_1(t)$ and $u_2(t)$ are the driving torque and force of the two links.

MOP is taken as a linear quadratic model representing small perturbations about the equilibrium point at the origin:

$$\min_{u(t)} \tfrac{1}{2}\int_0^4 [x^T Qx + u^T Ru + \gamma(t)]dt$$

subject to

$$\dot{x} = Ax + Bu + \alpha(t)$$

$$x(0) = [2 \quad 0 \quad 1 \quad 0]^T, x(4) = [0 \quad 0 \quad 0 \quad 0]^T$$

with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0.5 \end{bmatrix}$$

where $x(t) \in ¡^4$, $u(t) \in ¡^2$, $\gamma(t) \in ¡$, and $\alpha(t) \in ¡^4$. In this example the value of the weighting matrix Q is kept constant at

$$Q = \begin{bmatrix} 0.015 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } R = I_2$$

The numerical integration step used was h = 0.05 and a tolerance of 0.01 was specified for convergence. The simulation results are tabulated in the following table.

Table 3.3: The simulation results of Example 3.3.

case	r_1	r_2	$k_v = k_z$	k_p	No. of iterations	J^*	CPU Time (s)
i	1	0	0.30	1	179	6.462	138.459
ii	1	0	0.35	1	698	6.462	558.253
iii	1	0	0.20	1	87	6.452	66.255
iv	1	0	0.25	1	91	6.460	67.577

Figure 3.4 below shows the graphs of the final responses of Example 3.3(iii). Figs. 3.4(a), (b), and (c) graph the trajectories of the optimal control, the states, and the performance index, respectively. The remaining three figures, Fig. 3.4(d), (e), and (f), illustrate the optimality conditions with respect to control $u(t)$, the states $x(t)$, and the costates $p(t)$. It can be seen from the last three figures that the solution satisfies all three of the optimality conditions given by Eq. (3.10).

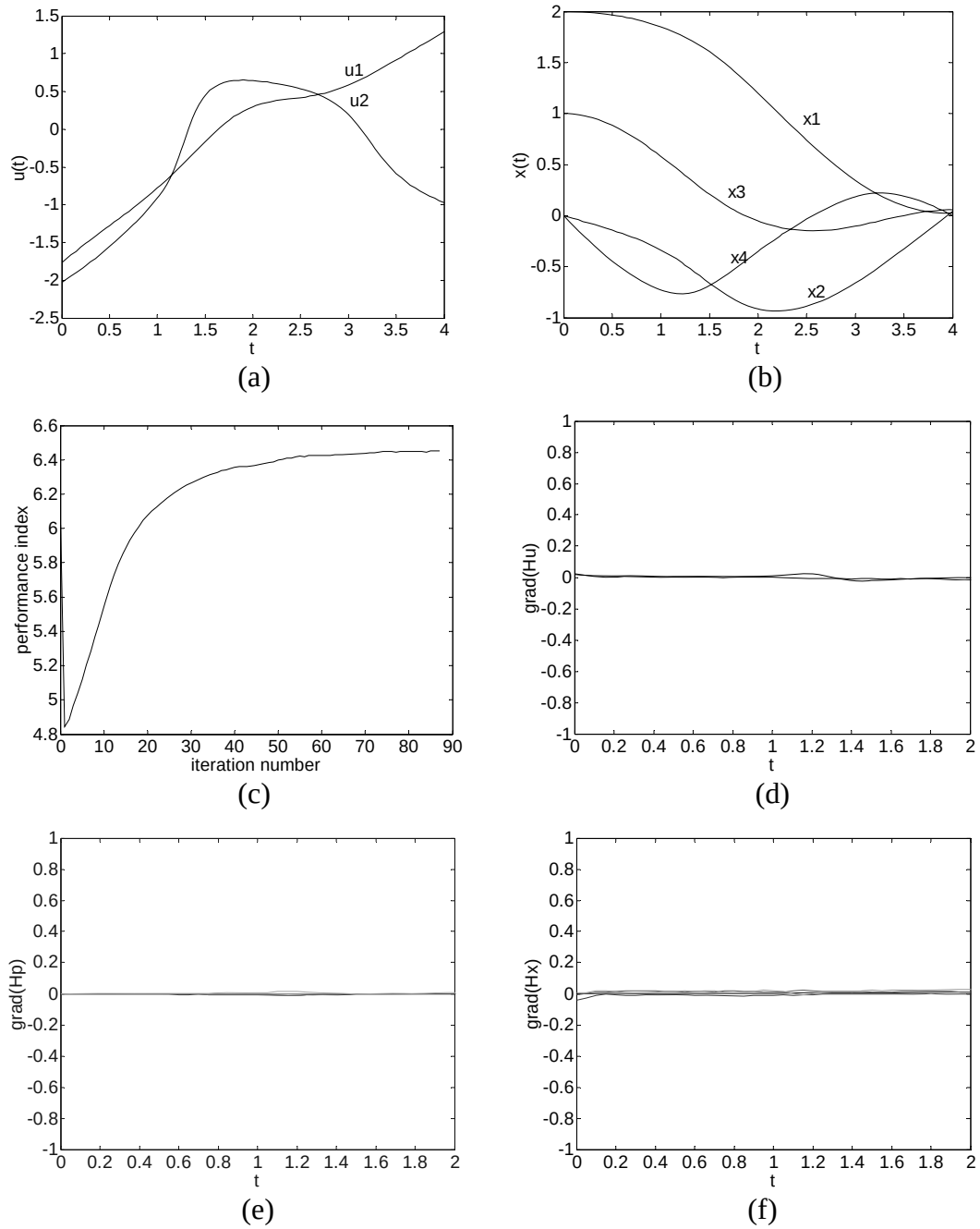


Figure 3.4: The final responses of DISOPE for Example 3.3(iii).

3.9 Summary and Conclusion

A detailed analysis of DISOPE has been presented in this chapter. It begins with the formulation of problem where the real problem ROP is transformed into a simpler problem MOP. MOP is formulated such that the 'reality' of ROP is represented by appropriate parameters. ROP and MOP are then tied together by another problem; the EOP. From EOP the necessary optimality conditions of the problem to be solved are derived. Another problem, MMOP that satisfies the same optimality conditions is formed. Both EOP and MMOP are equivalent to ROP, thus the solution of either of them is also the solution of ROP. MMOP is a simpler problem compared to EOP. Hence it is MMOP that is used by the algorithm when finding the optimal solution of ROP. DISOPE uses the back sweep method of Bryson and Ho (1975) to find the trial solutions of MMOP. This is done in the System Optimization step of the algorithm. The trial solutions are then updated by the updating mechanism and then tested for convergence. If the conditions for convergence are met, the algorithm will stop, otherwise the algorithm will iterate another trial solution. The process would go on until the time when the convergence conditions are met.

DISOPE that is analyzed here uses the LQR problem as model. Using this model, an algorithm mapping to trace the transition of the search from iteration i to iteration $i + 1$ was developed. Using this mapping, the analyses of optimality, stability and convergence of the algorithm were established. These analyses were based on the 2-D systems theory in the form of unit memory repetitive processes which used the concept of limit profiles to analyzed the algorithm's local convergence and stability. Also included is the analysis of the global convergence behavior of the algorithm for regions far from the limit profiles. Because the purpose of the chapter is to become a platform for comparing results from subsequent chapters, numerical examples with differing levels of complexity have also been included at the end of the chapter showing the responses of the algorithm at different situations. To complete the theoretical analysis of DISOPE, in the following chapter, among other things, discussions on convergence rate of DISOPE will be presented.

CHAPTER 4

FURTHER ANALYSES OF DISOPE ALGORITHM MAPPING AND ITS CONVERGENCE

4.1 Introduction

This chapter presents further theoretical analyses of DISOPE concerning its justification as a gradient descent algorithm and convergence rate.

We begin this chapter with a decomposition of DISOPE. The algorithm is decomposed into two distinct maps, Map B and Map C. This is done in order to highlight the updating mechanism, a form of a gradient descent algorithm, where the major part of the work done in the research takes place. The analysis goes on to show that the algorithm has the basic characteristics of a gradient descent method. Numerical simulation examples are presented to illustrate the property.

Next, an analysis on the rate of convergence for DISOPE is presented. The analysis begins with the establishment of a limit point to the sequences generated by the iterative procedure. Lastly, a complexity analysis of DISOPE is included here for the purpose of comparing the algorithm's efficiency with the two modified algorithms that will be introduced in Chapters 5 and 6.

4.2 Decomposition of DISOPE

DISOPE algorithm is made up of a four-step process that is repeated several times until a predetermined terminating criterion is met. Figure 3.1 suggests that the algorithm could be thought of as comprising of several possible individual maps indicated by each step in the figure. Upon close inspection, we concur that Steps 1 to 3 are doing basically the job of a calculator, producing outputs to be evaluated by Step 4. Step 4 on the other hand is doing the vital job of determining whether the solutions produced by the accumulative actions of Steps 1 to 3 should be accepted or rejected. If they were accepted then the search terminates, otherwise, the whole process would be repeated. Thus it is imperative that in the following analysis, DISOPE is decomposed into two distinct maps, with Steps 1 to 3 lumped together as one map and Step 4 is the other. Definition 4.1 that follows explicates the intention.

Definition 4.1

Let DISOPE algorithm be decomposed into two distinct maps; Map *B* and Map *C*. Map *B* is Steps 1 to 3 of Algorithm 3.1 and Map *C* is Step 4 of the algorithm given by Eq. (3.16). We say $A = BC$ is a composite map for DISOPE.

Definition 4.2 that follows defines composite mapping and Figure 4.1 gives the illustrations.

Definition 4.2

Let $Y, \hat{Y}, Z \in ¡^{m} \times ¡^{n} \times ¡^{n}$ be nonempty closed sets with $Y^{(i)}(t) = [u^{(i)}(t)^T \quad x^{(i)}(t)^T \quad p^{(i)}(t)^T]^T$, $\hat{Y}^{(i)}(t) = [\hat{u}^{(i)}(t)^T \quad \hat{x}^{(i)}(t)^T \quad \hat{p}^{(i)}(t)^T]^T$, and $Z^{(i)}(t) = [u^{(i-1)}(t)^T \quad x^{(i-1)}(t)^T \quad p^{(i-1)}(t)^T]^T$. Let $B: Z \to \hat{Y}$ and $C: \hat{Y} \to Y$ be point-to-set maps. The composite map $A = CB$ is defined as the point-to-set map $A: Z \to Y$ with $A(z) = \cup\{C(\hat{y}) : \hat{y} \in B(z)\}$.

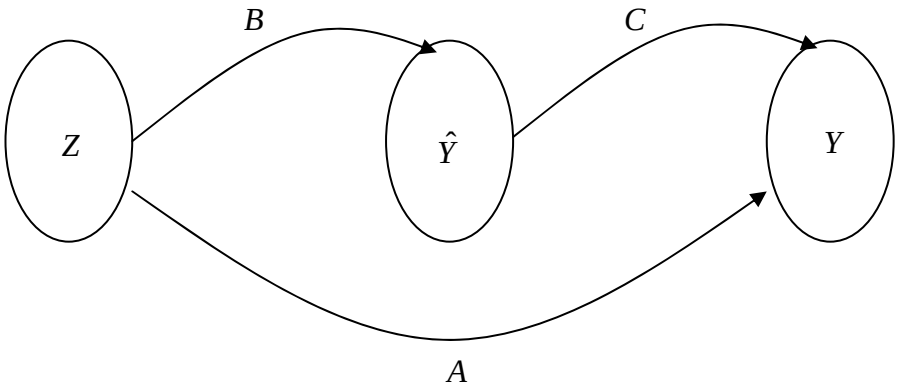


Figure 4.1: Composite map of DISOPE.

By viewing the algorithm as the composite Map *CB* (Rohanin and Mohd_Ismail, 2002), where *B* is known to be convergent and *C* corresponds to the set of intermediate steps of the complex algorithm, the overall convergence of such a scheme would be established (Bazaraa and Sherali, 1993).

Roberts (1993) and Becerra (1994) conjectured that DISOPE is a gradient descent algorithm. We intent to justify that the inference is valid since the discussion in this research revolves around this idea. From the algorithm decomposition of this section, clearly Map *C* is the part of DISOPE worthy of analysis to prove this conjecture. In the next section we carry out this justification.

4.3 Map C As a Gradient Descent Algorithm

Map C, defined by Eq. (3.16) could be written as

$$\left.\begin{aligned} u^{(i+1)}(t) &= u^{(i)}(t) - k_u(u^{(i)}(t) - \hat{u}^{(i)}(t)) \\ x^{(i+1)}(t) &= x^{(i)}(t) - k_x(x^{(i)}(t) - \hat{x}^{(i)}(t)) \\ p^{(i+1)}(t) &= p^{(i)}(t) - k_p(p^{(i)}(t) - \hat{p}^{(i)}(t)) \end{aligned}\right\} \tag{4.1}$$

By letting

$$y(t) = \left[u^{(i)}(t)^T \quad x^{(i)}(t)^T \quad p^{(i)}(t)^T \right]^T \tag{4.2}$$

Eq. (4.1) can be written in the form of

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right] \tag{4.3}$$

with K_y given by Eq. (3.41). Looking at Eq. (2.1), we identify that Eq. (4.3) is in a form of a gradient-like method, with K_y being the step size parameter to be estimated and the term $\left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]$ as the gradient of some function $E_y^i(t)$.

We can confirmed that Map *C* would be some form of a gradient descent method if we could deliver a function $E_y^i(t)$ associated with the optimal control problem such that its gradient is $\left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]$. In the following analysis we present such a function hence forth to be referred to as *the error function.*

4.3.1 Generating the Error Function

The formulation of MMOP in Eq. (3.15), contains an expression of terms called the convexification factors as follows

$$\tfrac{1}{2} r_1 \left\| u(t) - \hat{u}(t) \right\|^2 + \tfrac{1}{2} r_2 \left\| x(t) - \hat{x}(t) \right\|^2 + \tfrac{1}{2} r_3 \left\| p(t) - \hat{p}(t) \right\|^2 \tag{4.4}$$

These terms were put there, to augment the performance index in order to aid with the convergence. By looking at the optimality condition given by Eq. (3.14), we see that these factors work by matching the signals from reality and the parameter estimation problem.

The condition given by Eq. (3.14) is re-expressed here as

$$y^{(i)}(t) = \hat{y}^{(i)}(t) \tag{4.5}$$

Since DISOPE solves the optimal control problem numerically, Condition (4.5) is satisfied instead as

$$\left\| y^{(i)}(t) - \hat{y}^{(i)}(t) \right\| \to 0 \tag{4.6}$$

Expression (4.6) implies that DISOPE strives to eliminate the discrepancies between $y^{(i)}(t)$ and $\hat{y}^{(i)}(t)$. If we consider $\left\| y^{(i)}(t) - \hat{y}^{(i)}(t) \right\|$ to be an *error*, the expression for the convexification factors can then be viewed as a function of errors. In other words

the convexification factors work by driving each normed term in the expression to zero.

From the discussion above, if we take a similar expression for the error function $E_y^{(i)}(t)$ such that its gradient is the vector $\left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]$, then we succeeded in showing that Map *C* is a gradient descent method. In what follows we proposed the error function.

Proposition 4.1

Let DISOPE algorithm be partitioned into two distinct maps as in Definition 4.1 with the assumption that Map *B* converges. Consider the function $E_y^{(i)}(t) \in ¡^m \times ¡^n \times ¡^n$ defined as

$$E_y^{(i)}(t) = \tfrac{1}{2}\left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]^T \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right] \tag{4.7}$$

If the vector $\left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]$ is its gradient then $E_y^{(i)}(t)$ is an error function for Map C such that Map C is justified as a gradient descent method.

Proof

It is obvious from Eq. (4.7) that the gradient of $E_y^{(i)}(t)$ is given by

$$\left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] = y^{(i)}(t) - \hat{y}^{(i)}(t) \tag{4.8}$$

Substituting Eq. (4.8) into Eq. (4.3) gives us

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y \left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] \tag{4.9}$$

Comparing Eq. (4.9) with Eq. (2.1), we see that this is a form of a gradient descent method where

$$\left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] \tag{4.10}$$

is the gradient of some function to be minimized by the method. Since Eq. (4.9) is another form of Map C, then Map C is vindicated as a gradient descent method with $E_y^{(i)}(t)$ as its error function. This completes the proof. □

With Map C established as a gradient method. DISOPE in turn is a gradient descent algorithm. The conjecture is thus justified.

As a gradient descent algorithm, the basic characteristics inherent to all gradient descent methods are without a doubt intrinsic to DISOPE. The following section analyzes these properties.

4.4 DISOPE and the Basic Characteristics of the Gradient Descent Method

As mentioned in Section 1.5, one of the attractive features of this algorithm is the integration of parameters from both the real problem and its model. In view of this, DISOPE converges in just one iteration whenever no model-reality differences is introduced. The absence of model-reality differences is synonymous to having the gradient discussed in Section 4.3 as zero. In general, a zero gradient translates into the point of optimality in an optimization. If a gradient descent method were used in finding the optimal solution, this would mean that the initial guess is precisely the optimum. When this is the case, the gradient descent method would converge in just one iteration. Hence this is one characteristic of DISOPE that is in accord with the gradient descent method.

The other characteristic that is well known with the gradient descent method is the distance of the initial guesses to the optimal solution influences its speed of convergence. The closer the initial guess the faster the convergence.

DISOPE requires an initial solution signified by Step 0 in Algorithm 3.1 to start its iterations. A recommended one is the solution of the relaxed MMOP with $\alpha(t) = 0,\ r_1 = r_2 = 0$. Since DISOPE is composed of Maps *B* and *C*, with Map *B* supplying the input for Map *C*, the initial solution is for Map *B***.** However, the initial solution of *B* affects its output and hence the input for Map *C*. Thus it is imperative that a good initial solution to Map *B* would translate into a good input to Map *C*.

In our discussion of DISOPE, the real problems are always modeled as linear quadratic regulators as in Eq. (3.17). In gearing up to furnish an initial solution to Map *B, Q* and *R*; weights for the performance index, play a big hand in generating the solution. It is customary to use the identity matrices for *Q* and *R*. It is noticed however that other choices of *Q* and *R* have different effects on the speed of convergence of the algorithm. In fact the right choice of weights can tremendously cut down on the number of iterations as shown in the numerical examples below.

In the simulation below, examples using DISOPE are evaluated with different initial guesses. The distances between the initial guesses and the optimal solutions are then gauged and the numbers of iterations are then compared based on the distance recorded.

4.4.1 Numerical Examples

Each of the two examples below is simulated with two different initial solutions by giving different values to the weighting matrices of the performance index.

Example 4.1

Consider the problem discussed in Example 3.1. Two different initial solutions are used in the simulations of this problem. The first uses the value of the weighting matrix $Q = 2I_2$. The second initial solution uses the value of $Q = \begin{bmatrix} 22.40 & 4.480 \\ 4.480 & 0.896 \end{bmatrix}$.

In both cases the value of R is kept constant at $R = 0.2$. During the iterations of DISOPE, no tuning was done to the values of the parameters r_1 and, r_2 and k_v, k_z and k_p, that is, r_1 and, r_2 are set to zero and k_v, k_z and k_p are set to one. The integration step taken is $h = 0.01$, and the tolerance considered for the convergence is $tol = 0.01$. The 2-norms between the initial solutions and the optimal solution are calculated to gauge the ‘closeness’ of the initial guess to the converged solution. The results of the

simulations are summarized in Table 4.1 below. Figures 4.2 (a) and (b) below show the related results.

Table 4.1: Results of simulations with different values of Q.

	No. of iter.	$\|x_{init} - x_{opt}\|$	Values of J
$Q = 2I_2$	10	0.198	0.028
$Q = \begin{bmatrix} 22.40 & 4.480 \\ 4.480 & 0.896 \end{bmatrix}$	4	0.033	0.028

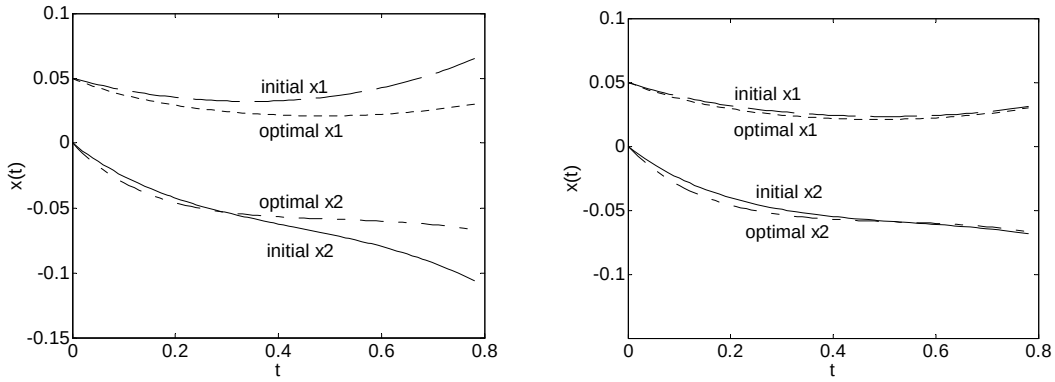


Figure 4.2: Comparisons of closeness between two different initial solutions and the optimal solution (a) Result for $Q = 2I_2$; (b) Result for Q = [22.40 4.480; 4.480 0.896].

It can be clearly seen from the results in Table 4.1 and Figures 4.2 (a) and (b) that a closer initial solution would translate into faster convergence for the algorithm. With an appropriate value of Q the number of iterations were reduced to more than half the original number recorded.

Example 4.2

We reconsider the problem from Example 3.3 here. In this example the value of the weighting matrix Q is kept constant at

$$Q = \begin{bmatrix} 0.015 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The weighting matrix R on the other hand is given two different values; $R = I_2$ and $R = 1.5I_2$. The numerical integration step used was $h = 0.05$ and a tolerance of 0.01 was specified for convergence.

With the choices of Q and $R = I_2$ together with $r_1 = 1, r_2 = 0$, $k_v = k_z = 0.25$, and $k_p = 1$ DISOPE converged in 91 iterations. To see the effect of the different initial solutions on the convergence speed of the algorithm, we simulated next $R = 1.5I_2$. The result is a faster convergence. In fact with this value of R, we did not have to resort to using the convexification terms to aid with convergence. We were able to keep the values of r_1 and r_2 at zero. The algorithm converged in 77 iterations. The 2-norms between the two initial solutions and the optimal solution are calculated and summarized in Table 4.2 below.

Table 4.2: Results of simulations with different values of R.

	No. of iter.	$\|x_{init} - x_{opt}\|$	Value of J
$R = I_2$	91	4.674	6.379
$R = 1.5I_2$	77	3.743	6.379

The two examples above showed that the choice of weights and hence the initial solution influence the speed of convergence. The closer the initial guess the faster the convergence (Rohanin and Mohd_Ismail, 2003d). Clearly, the second characteristic of the gradient descent method mentioned above is also inherent in DISOPE (Rohanin and Mohd_Ismail, 2003b, 2003c). These results verify both the stability and convergence analyses carried out for DISOPE in Chapter 3. In Theorem 3.2 for stability, both Q and R are present in the H and B_0 matrices located in the

characteristic equation that needed to be solved for the spectral values of λ in Eq. (3.96). Further more, in the sufficient condition for convergence given by Theorem 3.4, Q and R are present in the Lipschitz continuity assumptions in $g_1(y^{(i)}(t))$ and $g_2(y^{(i)}(t))$. Hence the variations of values for Q and R influence the convergence behavior of the algorithm.

4.5 Analysis on the Rate of Convergence

In this section we analyze the rate of convergence of DISOPE. The analysis presented here is based on the assumption that the problems to be solved by this algorithm are modeled as LQR problems. We begin with the basic definitions and theorems necessary for determining the rate of convergence.

Definition 4.3 (Polak, 1997)

Let Y be a real normed space.

(a) A sequence $\{y^{(i)}(t)\}_{i=0}^{\infty}$ in Y is said to converge to a point $\hat{y}$, indicated by $y^{(i)} \to \hat{y}$, as $i \to \infty$, if $\lim_{i\to\infty} \|y^{(i)} - \hat{y}\| = 0$. The point $\hat{y}$ is called the limit point of $\{y^{(i)}(t)\}_{i=0}^{\infty}$.

(b) A point $\hat{y}$ is said to be an accumulation point of a sequence $\{y^{(i)}\}_{i\in¥}$ in Y, if there exists an infinite subset $K \subset ¥$ such that for the subsequence $\{y^{(i)}\}_{i\in K}$

$$\lim_{\substack{i\to\infty \\ i\in K}} \|y^{(i)} - \hat{y}\| = 0, \tag{4.11}$$

denoted by $y^{(i)} \to^{K} \hat{y}$ as $i \to \infty$ or $\lim_{K} y^{(i)} = \hat{y}$.

(c) A sequence $\{y^{(i)}(t)\}_{i=0}^{\infty}$ in Y is said to be Cauchy if for any $\delta > 0$ there exists an i_δ such that, if $i \geq i_\delta$ and $j \geq i_\delta$, then $\|y^{(i)} - y^{(j)}\| < \delta$.

It can be shown that any sequence that converges is Cauchy and that, if a Cauchy sequence has accumulation points, it must converge.

Theorem 4.2 (Polak, 1997)

Let $\{y^i\}_{i=0}^{\infty}$ be a sequence in ¡ $^{m\times n\times n}$. If there exists a $c \in (0,1)$ and $i_0 \in ¥$ such that

$$\left\| y^{(i+1)} - y^{(i)} \right\| \le c \left\| y^{(i)} - y^{(i-1)} \right\|, \ \forall\, i \ge i_{0+1} \tag{4.12}$$

then there exist a $\hat{y} \in$ ¡ ^{n} such that $y^{(i)} \to \hat{y}$, as $i \to \infty$, at least R-linearly.

Proof

For $i = 0,1,2,...,$ let e_i @$\left\| y^{(i+1)} - y^{(i)} \right\|$. Then by induction, it follows from (4.12) that for all $i \ge i_0$, $e_i \le e_{i_0} c^{i-i_0}$, and hence, since $c \in (0,1)$, that $e_i \to 0$, as $i \to \infty$, R-linearly. Therefore, for any $j > k \ge i_0$, we find that

$$\left\{ \begin{array}{c} \left\| y^{(j)} - y^{(k)} \right\| = \left\| \left(y^{(j)} - y^{(j-1)} \right) + \left(y^{(j-1)} - y^{(j-2)} \right) + ... + \left(y^{(k+1)} - y^{(k)} \right) \right\| \\ \le \sum_{i=k}^{j-k} e_i \le \sum_{i=k}^{\infty} e_i \le \sum_{j=k-i_0}^{\infty} e_{i_0} c^j = c^{k-i_0} \left(\frac{e_{i_0}}{1-c} \right) \end{array} \right\} \tag{4.13}$$

That is, $\left\| y^{(j)} - y^{(k)} \right\| \to 0$ for $j > k$, as $k \to \infty$, uniformly in j. Hence the sequence $\{y^{(i)}\}_{i=0}^{\infty}$ is Cauchy, and therefore, it must converge to a point $\hat{y}$. □

Before specifically dealing with the analysis to determine the convergence rate, we have to first establish that the sequence of trial solutions produced by DISOPE has a limit point. In the following analysis $\hat{y}$ signifies the limit point.

4.5.1 Establishing the Existence and Uniqueness of $\hat{y}$ in DISOPE

To discuss the convergence rate, it is important that we first establish the existence and uniqueness of the limit point of the sequence generated by the repeated applications of the algorithm. In this subsection we supply the theorems that prove

just that. With the existence and uniqueness of the limit point, we can then go on to analyze the rate of convergence.

Theorem 4.3 - Existence

Let $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty}$ be the sequence of terms generated by the repeated applications of the algorithm DISOPE. If the successive terms contract according to

$$\left\|y^{(i+1)}(t)-y^{(i)}(t)\right\| \le \left\|y^{(i)}(t)-y^{(i-1)}(t)\right\|, \quad i=0,1,2,...,$$

then there exists a limit point $\overset{)}{y}$, such that $\lim_{i\to\infty} y^{(i)}(t) = \overset{)}{y}$.

Proof

From Theorem 3.4 we have

$$\left\|y^{(i+1)}(t)-y^{(i)}(t)\right\| \le \vartheta_1 \left\|y^{(i)}(t)-y^{(i-1)}(t)\right\|, \quad i=0,1,2,... \tag{4.14}$$

where

$$\vartheta_1 = \left(\left\|E_2 + I_{2n+m} - K_y\right\| + h_2 + \left(h_3 + \left(\left\|E_1\right\| + h_1\right)\sigma\left(t_0, t_f\right)\right)\left\|C\right\|\right) \tag{4.15}$$

with

$$\vartheta_1 \le 1 \tag{4.16}$$

From the Lipschitz continuity assumptions of Eq. (3.109), all three of the Lipschitz constants $h_1, h_2,$ and h_3 take on the positive values implying $\vartheta_1 \in (0,1]$. From Definition 3.3, this means that DISOPE is contractive. Hence, from either Theorem 3.3 or Theorem 4.2, there exists a limit point $\overset{)}{y}$, for $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty}$ such that $\lim_{i\to\infty} y^{(i)}(t) = \overset{)}{y}$. □

Our next corollary is a logical consequence of Theorem 4.3. It proves the property of uniqueness to the converged final solution of the sequence of terms generated by DISOPE.

Corollary 4.1 - Uniqueness

Let $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty}$ be the sequence in Theorem 4.3. If the limit point $\hat{y}$ exists, then it is unique.

Proof

A sequence of real numbers can converge to at most one limit (Kirkwood, 1989; Bartle and Sherbert, 1992). □

With Theorem 4.3 and Corollary 4.1, we have proven the existence and uniqueness of the limit point to the sequence of terms generated by repeated applications of the algorithm DISOPE. Our next concern is to establish the convergence rate of the algorithm. In the next subsection we direct our attention to the convergence rate of DISOPE.

4.5.2 Establishing the Convergence Rate

We have thus far proved that a unique limit point existed for the sequence generated by DISOPE. In this subsection, we proceed to establish the convergence rate for the algorithm.

Corollary 4.2

The sequence $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty}$ in DISOPE converges at least R-linearly.

Proof

Theorem 4.3 states that $\hat{y}$ exists, meaning there exist some $\vartheta_1 \in (0,1)$ and $i \in ¥$ such that $\left\|y^{(i+1)}(t) - y^{(i)}(t)\right\| \le \vartheta_1 \left\|y^{(i)}(t) - y^{(i-1)}(t)\right\|$. Therefore, from Theorem 4.2 $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty} \in ¡^{m} \times ¡^{n} \times ¡^{n}$ converges at least R-linearly. □

Corollary 4.2 proved that at the very least, the sequence of DISOPE converges R-linearly. However, in our analysis, we found that DISOPE could be proven to have at least a quadratic convergence rate. We begin with the following definition of a quotient rate, with $r > 1$.

Definition 4.4 (Polak, 1997)

We say that a sequence $\left\{ y^{(i)}(t) \right\}_{i=0}^{\infty}$ in $\mathbb{R}^n$ converges to a point $\hat{y}$ at least with quotient rate (Q-rate) $r > 1$ if there exist a $\kappa \in [0, \infty)$ and an $i_0 \in \mathbb{N}$ such that for all $i \geq i_0$

$$\frac{\left\| y^{(i+1)} - \hat{y} \right\|}{\left\| y^{(i)} - \hat{y} \right\|^r} \leq \kappa \tag{4.17}$$

With Definition 4.4 in mind, we proceed to state the following theorem, which established the quadratic convergence of DISOPE.

Theorem 4.4

Let $\left\{ y^{(i)}(t) \right\}_{i=0}^{\infty}$ of DISOPE be a sequence in $\mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$. If $\hat{y}$ is a limit point of $\left\{ y^{(i)}(t) \right\}_{i=0}^{\infty}$, then $\left\{ y^{(i)}(t) \right\}_{i=0}^{\infty}$ converges to $\hat{y}$ at least with Q-rate, $r = 2$.

Proof

Given the following error function

$$E(y^{(i)}(t)) = \tfrac{1}{2} \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]^T \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right] \tag{4.18}$$

we have proven in Section 4.3 that it is an error function for Map C such that Map C minimizes it in searching for the optimal solution of the optimal control problem when using DISOPE as the search method. Let us reproduce Map C here.

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y E'_{y^{(i)}}(y^{(i)}(t)) \tag{4.19}$$

Map C is an iteration function that updates the trial solution at each iteration.

Let $\hat{y}$ be a simple root of equation $E'_{y^{(i)}}(y^{(i)}(t)) = 0$; i.e. $E'_{y^{(i)}}(\hat{y}) = 0$, subtract $\hat{y}$ from both sides of Eq. (4.19) to produce

$$y^{(i+1)}(t) - \hat{y} = y^{(i)}(t) - \hat{y} - K_y E'_{y^{(i)}}(y^{(i)}(t)) \tag{4.20}$$

Taking the norm of both sides of Eq. (4.20) we have

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| = \left\| \left(y^{(i)}(t) - \hat{y} \right) - K_y E'_{y^{(i)}}(y^{(i)}(t)) \right\| \tag{4.21}$$

which becomes the following inequality

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| \le \left\| y^{(i)}(t) - \hat{y} \right\| + \left\| K_y E'_{y^{(i)}}(y^{(i)}(t)) \right\| \tag{4.22}$$

Rewrite (4.22) as the following and include a term of $E'_{y^{(i)}}(\hat{y}) = 0$

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| \le \left\| y^{(i)}(t) - \hat{y} \right\| \left(1 + \left\| \frac{K_y \left(E'_{y^{(i)}}(y^{(i)}(t)) - E'_{y^{(i)}}(\hat{y}) \right)}{(y^{(i)}(t) - \hat{y})} \right\| \right) \tag{4.23}$$

Using the notation of Gautschi (1997) for Newton's form of interpolation polynomial Inequality (4.23) becomes

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| \le \left\| y^{(i)}(t) - \hat{y} \right\| \left(1 + \left\| K_y[y^i, \hat{y}]E' \right\| \right) \tag{4.24}$$

where $[y^{(i)}, \hat{y}]E'_{y^{(i)}} = \dfrac{E'_{y^{(i)}}(y^{(i)}(t)) - E'_{y^{(i)}}(\hat{y})}{y^{(i)}(t) - \hat{y}}$. Multiply the right hand side of Inequality (4.24) with the term $\left\| \dfrac{y^{(i)}(t) - \hat{y}}{y^{(i)}(t) - \hat{y}} \right\|$ to get

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| \le \left\| y^{(i)}(t) - \hat{y} \right\| \left\| \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)}(t) - \hat{y})} (y^{(i)}(t) - \hat{y}) \right\|. \tag{4.25}$$

From (4.25) we get

$$\left\| y^{(i+1)}(t) - \hat{y} \right\| \le \left\| y^{(i)}(t) - \hat{y} \right\|^2 \left\| \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)}(t) - \hat{y})} \right\| \tag{4.26}$$

which simplifies into

$$\frac{\left\| y^{(i+1)}(t) - \hat{y} \right\|}{\left\| y^{(i)}(t) - \hat{y} \right\|^2} \le \left\| \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)}(t) - \hat{y})} \right\|. \tag{4.27}$$

Taking the limit on both sides of the inequality, we have

$$\lim_{i \to \infty} \left\| \frac{y^{(i+1)}(t) - \hat{y}}{\left(y^{(i)}(t) - \hat{y} \right)^2} \right\| \le \lim_{i \to \infty} \left\| \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)}(t) - \hat{y})} \right\| \tag{4.28}$$

which is equivalent to

$$\left\| \lim_{i\to\infty} \frac{y^{(i+1)}(t) - \hat{y}}{\left(y^{(i)}(t) - \hat{y}\right)^2} \right\| \le \left\| \lim_{i\to\infty} \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)}(t) - \hat{y})} \right\| \tag{4.29}$$

Since $\hat{y}$ is a root of $E(y^{(i)}(t))$, $y^{(i)}(t) \to \hat{y}$ as $i \to \infty$, the limit of the right hand side of (4.29) becomes

$$\lim_{i\to\infty} \frac{1 + K_y[y^{(i)}, \hat{y}]E'_{y^{(i)}}}{(y^{(i)} - \hat{y})} = \infty \tag{4.30}$$

since $[\hat{y}, \hat{y}]E'_{y^{(i)}} = \frac{1}{1!}\left(E'_{y^{(i)}}(\hat{y})\right)' = E''_{y^{(i)}}(\hat{y})$ is a constant. If we let

$$\lim_{i\to\infty} \frac{\left\| y^{(i+1)}(t) - \hat{y} \right\|}{\left\| y^{(i)}(t) - \hat{y} \right\|^2} = \kappa \tag{4.31}$$

then $\kappa \in [0, \infty)$. Therefore from Definition 4.4, with $i_0 = 0$, $\left\{y^{(i)}(t)\right\}_{i=0}^{\infty}$ converges to $\hat{y}$ with at least Q-rate, $r = 2$. Thus we have proven that DISOPE has a quadratic rate of convergence. □

4.6 Summary and Conclusion

This chapter presents further theoretical analyses of DISOPE. The claims that DISOPE is a gradient descent algorithm (Roberts, 1993; Becerra, 1994) and its rate of change is quadratic (Roberts, 1993) are verified here. The algorithm is decomposed into two distinct sub procedures based on the theorems of composite mapping. One of the maps is the main DISOPE algorithm and the other is the updating mechanism of the algorithm. The updating mechanism is treated as a full-fledged algorithm with an appropriate error function determined for it. The updating mechanism is established as a gradient-descent type as claimed. The quadratic rate of change is also proven in this chapter.

As a gradient descent algorithm DISOPE is shown to have the basic characteristics that are inherent to the method. These are the one-step convergence when no model-reality differences are introduced and the behavior towards initial solutions. It is shown that the closer the initial solution to the optimum, the faster the

convergence. Two numerical examples are employed to vindicate the property. In both examples, the convergence to the optimum is faster when the initial solution chosen to start the iterations is closer to the optimal solution.

In short, we managed to establish that DISOPE is a gradient descent algorithm with a quadratic rate of convergence. In the following chapter we present the first modification scheme intended to improve the performance of DISOPE.

CHAPTER 5

DISOPE-MOMENTUM ALGORITHM

5.1 Introduction

This chapter presents one modification of Map *C*, the updating mechanism of DISOPE, that exhibit improved convergence speed. In Chapter 4 we verified that DISOPE is a gradient descent algorithm. Specifically, the updating mechanism is a type of the gradient descent method. Here, a modification excerpted from the literature of the back propagation (BP) algorithm of the neural networks is tested on DISOPE.

The BP algorithm is a prolific gradient descent method. Based on this observation, modifications done to the BP algorithm for the purpose of improving its convergence speed are studied. This chapter reports the use of the momentum terms as addendums of choice to the updating mechanism.

The well-established momentum term has commendable effect in reducing oscillation of the BP algorithm. The modified DISOPE algorithm as expected show similar response with the addition. Simulations of numerical examples are used to see the effects the momentum term on the performance of DISOPE.

A new algorithm called DISOPE-MOMENTUM is developed based on the modification. A time-complexity analysis is done and the efficiency of this new

algorithm is compared to the efficiency of the original DISOPE discussed in Chapter 4.

5.2 Modification of Map *C*

As explained earlier, Map *C* or the updating mechanism of DISOPE is a type of gradient descent method and because of it, DISOPE is prone to the problem of slow convergence. The slow convergence of the algorithm is caused by either the oscillation of the search near an optimum point or the small gradient of the wide flat surface of the error function. Here we present a technique of modifying Map *C* in order to overcome the aforementioned problem. This technique is well known to the BP algorithm of the neural networks. This technique involves adding a momentum term to Map *C*. In doing so, the oscillation is reduced and the flat surface is traversed with bigger stride.

5.2.1 Similarities Between Map *C* and BP Algorithm

To start the analysis, we begin with comparing the structure of Map *C* with the BP algorithm. Map *C* as given in Eq. (4.3) is

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right] \tag{5.1}$$

In Section 4.3 an appropriate error function was developed for it. The function is given by Eq. (4.7) as follows

$$E_y^{(i)}(t) = \tfrac{1}{2} \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right]^T \left[y^{(i)}(t) - \hat{y}^{(i)}(t) \right] \tag{5.2}$$

In terms of the error function, Map *C* could be expressed as in Eq. (4.9), which is

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y \left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] \tag{5.3}$$

The BP algorithm of Rumelhart *et al*. (1986) is an iterative gradient descent algorithm designed to train multilayer feed forward networks of sigmoid nodes by

minimizing the mean square error between the actual output of the network and the desired output. From Eq. (2.6), the BP algorithm is defined as

$$\mathbf{w}_{ij}(n+1) = \mathbf{w}_{ij}(n) - \eta^m \frac{\partial E}{\partial \mathbf{w}_{ij}} \tag{5.4}$$

with its error function given by Eq. (2.4) as the least squares error function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{p}(t_j - x_j)^2 \tag{5.5}$$

Comparing Eqs. (5.3) and (5.4) we see that the two equations have basically the same structure. Both have the role of updating. In Eq. (5.3) Map *C* updates trial solutions produced by DISOPE, where as in Eq. (5.4), the BP algorithm updates weights used in training the neural network. In both equations the error functions are quadratic functions.

Based on this observation, one of the modifications done to BP algorithm is applied on Map *C*. The inclusion of a term registering the momentum from the previous immediate step is the chosen modification described in this chapter.

5.2.2 The Inclusion of the Momentum Term

The momentum term is a vector parallel to the previous search direction. An addition of a multiple of this vector to the current direction deflects the end point of the search vector to a new location. The new location is situated further down the line towards the optimum. Fig. 5.1 below illustrates the displacement.

In Fig. 5.1, let $y^{(i)}$ represents the position of the search at iteration *i*. The vector **a** is the direction of the previous search for $y^{(i)}$. Vector **b** is the direction given by the gradient descent algorithm for the current search. Vector **c,** a multiple of **a,** is the momentum term added to **b**. **c** deflects **b** to the new position of **d**. Thus,

instead of having $y^{(i+1)}$ as the current position for the search, we have $y^{(i+1)}$, which is further down the line towards the optimum.

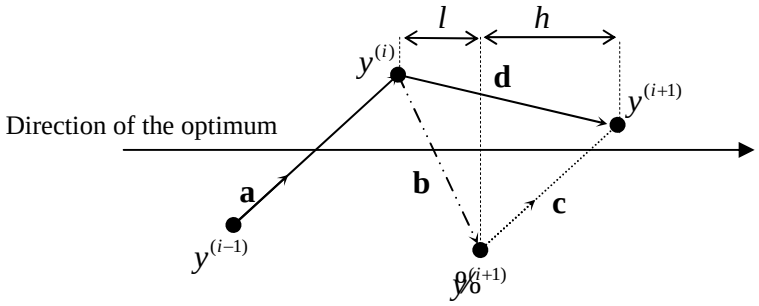


Figure 5.1: The effect of the momentum term on the search direction.

The magnitude of **c** is determined by the choice of the values given to the multipliers, called the momentum parameters. The parameters are set to be between zero and one. The momentum parameter places an upper limit on the amount by which an estimate can be changed. As such, the location of $y^{(i+1)}$ is determined by these choices. The correct choice of the parameters is crucial in determining the success of the momentum addition because the momentum can cause the estimates to be changed in the wrong direction and would instead increase the error (Jacobs, 1988).

From Fig. 5.1, we see that the displacement of the solution from the position of $y^{(i)}$ to the position of $y^{(i+1)}$ is $l + h$ units with *l* being the displacement brought about by the gradient descent algorithm and *h* by the momentum term. For the next iteration, a multiple of vector **d** would be the new momentum term.

In association with the BP algorithm, the momentum term is defined as

$$\mathbf{w}_{ij}(n) - \mathbf{w}_{ij}(n-1) \tag{5.6}$$

The inclusion of this term to the BP algorithm changes Eq. (5.4) to become

$$\mathbf{w}_{ij}(n+1) = \mathbf{w}_{ij}(n) - \eta^m \frac{\partial E}{\partial \mathbf{w}_{ij}} + \varpi(\mathbf{w}_{ij}(n) - \mathbf{w}_{ij}(n-1)) \tag{5.7}$$

with ϖ being the momentum parameter.

Following the same structure, and with the help of Fig. 5.1, the momentum term for Map *C* of DISOPE is defined as

$$y^{(i)}(t) - y^{(i-1)}(t) \tag{5.8}$$

The addition of the momentum term to Map *C* in Eq. (5.3) gives us the following equation (Rohanin *et al.* 2002)

$$y^{(i+1)}(t) = y^{(i)}(t) - K_y \left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] + W_y (y^{(i)}(t) - y^{(i-1)}(t)) \tag{5.9}$$

where

$$W_y = \begin{bmatrix} \varpi_u I_m & 0 & 0 \\ 0 & \varpi_x I_n & 0 \\ 0 & 0 & \varpi_p I_n \end{bmatrix} \tag{5.10}$$

is a matrix of momentum parameters with $\varpi_u, \varpi_x,$ and $\varpi_p \in (0,1]$.

5.3 The Effects of the Momentum Term on DISOPE

From Fig. 5.1 we see that the momentum term delivered additional *h* units of displacement to the position of the current trial solution of the search. Furthermore, the displacement is in the direction of the optimal solution. This displacement helps lengthen the stride of each iteration. With the distance between the initial and the optimal solution unchanged, this lengthening of strides makes the search arrive at the optimum faster. Thus, if the search has been oscillating near an optimum solution, the number of oscillations is undoubtedly reduced by this action. The same goes for a search moving with very small steps on a flat area. The momentum term helps widen the steps so that the search would be able to pass through the flat surface faster.

There are two sides to the concept of faster convergence. One is exhibited in the reduction in the number of iterations and the other is in the reduction of the CPU time (Rohanin and Mohd_Ismail, 2003d). If the modified algorithm exhibits both reductions then it truly has faster convergence in the true sense. The following

subsection presents simulations of numerical examples to exhibit the effects the momentum term has on the performance of DISOPE. These examples are taken from the examples simulated in Chapter 3. The results obtained here are then compared to the results recorded in that chapter.

5.3.1 Numerical Examples

Two problems from Chapter 3 are used in this section to simulate the effects of momentum terms on the performance of the algorithm.

Example 5.1

Consider Example 3.1. With every input to the problem in Example 3.1 retained, the results after the inclusion of the momentum terms are recorded in Table 5.1.

Table 5.1 compares the performance of DISOPE, to the performance of DISOPE with the momentum term. Column (a) records the number of iterations for convergence of DISOPE. Column (b) records the number of iterations needed after the modification. J_1^* and J_2^* respectively record the final values of the performance indices in both situations.

Table 5.1: Algorithm's performance of Example 5.1 with the addition of momentum terms.

Case	r_1	k_v	(a)	J_1^*	CPU Time (s)	ϖ_y	(b)	J_2^*	CPU Time (s)
i	0	1	10	0.028	1.832	[0 0.1 0]	5	0.028	0.641
ii	1	1	12	0.028	2.173	[0 0.25 0]	9	0.028	1.753
iii	0.5	1	9	0.028	1.733	[0.01 0.04 0.01]	7	0.028	0.902
iv	0	0.8	9	0.028	1.762	[0.01 0.1 0.01]	5	0.028	0.661
v	1	0.8	13	0.028	2.444	[0 0.4 0]	10	0.028	2.063

Clearly from the table, we see that the momentum terms do affect the speed of convergence for DISOPE. The addition of the term to the gradient descent search of the updating mechanism succeeded in effectively reducing the number of iterations in all the cases simulated. The longer strides discussed above do translate into reduced number of iterations.

To further synthesize the effects, we recorded the CPU time taken to run each problem. The CPU time can then be used as a gauge for the efficiency of this modified algorithm. In the simulations, the CPU time is an important measure of the worthiness of the reductions. Intuitively, the addition of the momentum term would increase the complexity of the algorithm. The results tabulated in Table 5.1 however, show that if there is an increase in the complexity of the algorithm, the increase is negligible since all the CPU times were reduced along with the reduction in the number of iterations. All the reductions in CPU times are proportional to the reductions in iterations.

For ease of reference, we will refer to the modified algorithm as DISOPE-MOMENTUM. Next we present the graphs of the performance indices of DISOPE and DISOPE-MOMENTUM.

In Fig. 5.2 we plot both performance indices of Case (i) from Example 5.1 on the same graph for ease of comparison. For this particular case, the number of iterations for DISOPE-MOMENTUM is half that of DISOPE. Analyzing the graphs, we see that in the beginning, the estimations of DISOPE-MOMENTUM are similar to DISOPE's.

However, once the momentum takes affect the graph with DISOPE-MOMENTUM shows a character of its own. This effect can be seen starting from the second iteration. The values of the estimates lie in a range smaller than that of DISOPE's. This behavior indicates that the norm between the estimates gets smaller faster. The iterations eventually stopped after the fifth iteration.

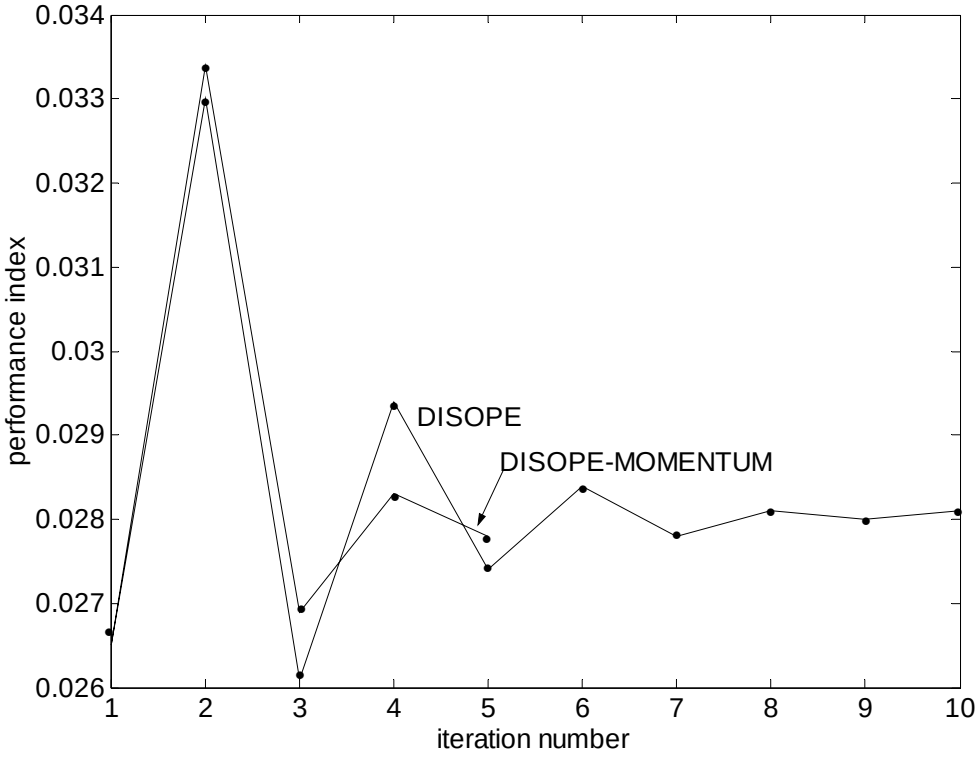


Figure 5.2: The comparison of the performance indices of DISOPE and DISOPE-MOMENTUM, for Case (i) of Example 5.1.

Fig. 5.3 shows the control variation norms, $\left\| u^{i+1}(t) - u^{i}(t) \right\|$ of both algorithms. This is the measure we use for the stopping criterion of the algorithms. We can see that the slope of the control norm of DISOPE-MOMENTUM is steeper than that for DISOPE. This is an indication that $\left\| u^{i+1}(t) - u^{i}(t) \right\| \to 0$ faster for DISOPE-MOMENTUM which translates into faster convergence.

Example 5.1 has been a simple one with the problem requiring only a small number of iterations to converge. We can conclude that for a simple problem, the addition of the momentum terms worked wonders in improving the convergence behavior of the algorithm.

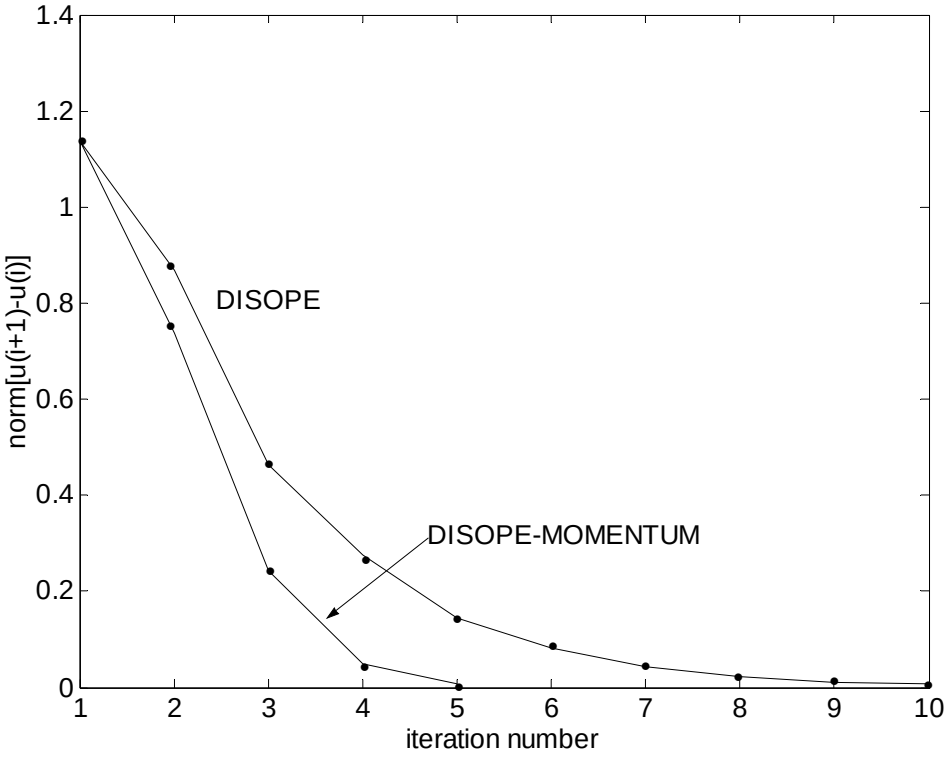


Fig. 5.3: The comparison of the control variation norms of DISOPE and DISOPE-MOMENTUM for Case (i) of Example 5.1.

Next we simulate a problem with a higher degree of nonlinearities, a problem that required a large number of iterations to converge. It is on problems like this that the contribution of the momentum terms would be most appreciated.

Example 5.2

Consider Example 3.3. The input to the problem in Example 3.3 is retained in this example. The results after the inclusion of the momentum terms are tabulated in Table 5.2. Here too the values for the momentum parameters are chosen by trial and error, with the values presented in the table being the best values found for each case.

As in Table 5.1, Columns (a) and (b) record the number of iterations for the convergence of DISOPE and DISOPE-MOMENTUM respectively. J_1^* and J_2^*

respectively records the final values of the performance indices in both algorithms. CPU times are recorded for comparison between the two algorithms.

Table 5.2: The comparison of the performances of DISOPE and DISOPE-MOMENTUM for Example 5.2.

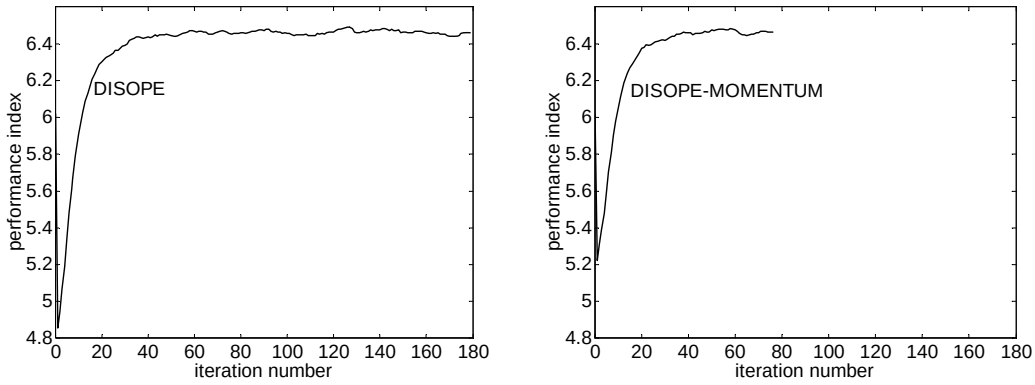
Case	$k_v = k_z$	(a)	J_1^*	CPU Time (s)	ϖ	(b)	J_2^*	CPU Time (s)
i	0.30	179	6.462	138.459	[0.1 0.03 0]	76	6.464	60.107
ii	0.35	698	6.462	558.253	[0.1 0.1 0]	83	6.457	63.191
iii	0.20	87	6.452	66.255	[0.1 0.1 0]	79	6.469	62.500
iv	0.25	91	6.460	67.577	[0.08 0.01 0]	78	6.469	57.863

The discrepancies in the values of J_1^* and J_2^* are within the tolerance specified for this problem, which is $\varepsilon = 0.01$. Again, in this example, the momentum terms succeeded in significantly reducing the number of iterations for convergence while keeping the optimal cost at relatively the same value. In cases (i) and (ii) the numbers of iterations are cut down to less than half the original values with case (ii) having the most significant improvement. Even when the algorithm converged in fewer numbers of iterations as in cases (iii) and (iv), the momentum terms still manage to further reduce the number of iterations.

We mentioned an increase in the complexity of the algorithm in analyzing Example 5.1. The results of this example clearly show that this increase is a small price to pay for the astounding end result the momentum terms could bring about. The small increase is definitely negligible in the long run. DISOPE-MOMENTUM's potential far outweighs its encumbrance. For this example, the CPU time of every case simulated has been successfully reduced. Together with the reduction in the number of iterations, DISOPE-MOMENTUM succeeded in reducing the convergence speed in the true sense of the word.

Next we present the graphs of the performance indices of the two different algorithms. For comparison purposes, we choose the graphs of Case (i) not Case (ii)

even though Case (ii) showed the best overall improvement. This is done because if we have chosen Case (ii), and use the same scale for both graphs, the graph for DISOPE-MOMENTUM would be so cramped to the y-axis that any comparison would be impossible.



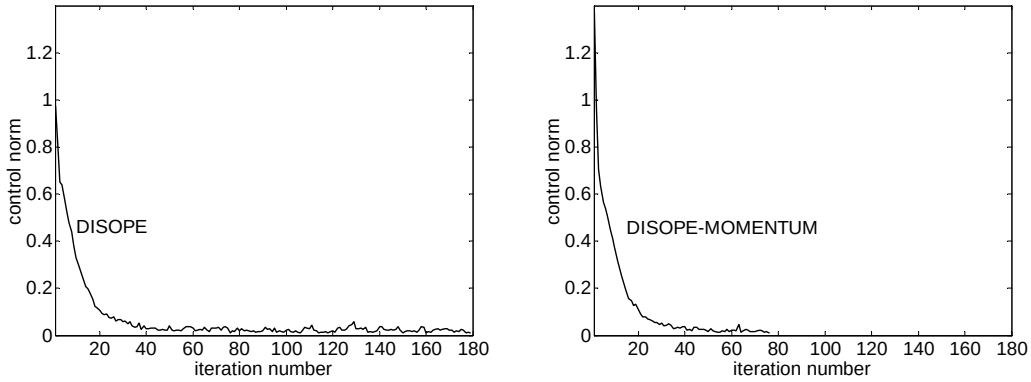
(a) (b)

Figure 5.4: The comparison of the performance indices of (a) DISOPE and (b) DISOPE-MOMENTUM, for Case (i) of Example 5.2.

Figs. 5.4 (a) and (b) show the performance indices of Case (i) for both algorithms. From the figures, we can say that the convergence pattern is basically similar. One significant different apart from the total iteration numbers is that the trial solutions for DISOPE-MOMENTUM lie in a smaller range than that of DISOPE. Also we noticed that in Fig. 5.4 (a), the trial solutions for the performance index lingered around the optimal solution beginning from about Iteration 40 up to Iteration 179. We can say that the trial solutions oscillate near the optimum for about 139 times before arriving at the optimal solution. In Fig. 5.4 (b) however, the oscillation is cut down to about only 36 times. This is the effect of the momentum. With the aid of the momentum, the algorithm managed to converge to the optimum faster. For this particular example, the momentum terms managed to make do without about a hundred redundant iterations.

The same goes for the graphs of the control variation norms in Fig. 5.5. In both Figs. 5.5 (a) and (b) the control norms start getting close to zero starting form approximately Iteration 40. However in Fig. 5.5(a) the process of reducing the norm to a value less than the given tolerance is prolonged until Iteration 179. In (b)

however, the momentum term used managed to bring the value of the norm to less than the tolerance within 79 iterations. This is a clear example of the effect that the momentum terms has on oscillations.



(a) (b)

Figure 5.5: The comparison of the control variation norms of (a) DISOPE and (b) DISOPE-MOMENTUM, for Case (i) of Example 5.2.

Example 5.2 exhibited that high degree of nonlinearity of the problem did not hinder the momentum term from accomplishing its intended purpose. In fact we see that, it managed to reduce a significantly large number of unnecessary iterations in two of the cases simulated. From the economics point of view, this achievement would mean a great saving on cost.

With everything else equal, the momentum terms in DISOPE-MOMENTUM without a doubt accomplished the feat expected of them. In both examples we see that DISOPE-MOMENTUM algorithm was able to improve on its convergence speed compared to the original algorithm DISOPE not only in terms of the reduction in the number of iterations but also in the reduction of CPU time taken to run the problem.

We observe that with the inclusion of the momentum term, the algorithm converged faster. Now the optimal solution could be reached in a lesser number of iterations and CPU time. The momentum term delivered what was expected of it. With this success, we formally define the new algorithm called the DISOPE-MOMENTUM algorithm in the next section.

5.4 DISOPE-MOMENTUM Algorithm

Eq. (5.9) is now the updating mechanism of the modified DISOPE or DISOPE-MOMENTUM. For the purpose of subsequent analysis, Eq. (5.9) will be rewritten in such a way that the momentum contribution is separated from the gradient descent step. For this purpose alone, we will use

$\tilde{y}^{(i+1)}(t) = [\tilde{u}^{(i+1)}(t)^T \quad \tilde{x}^{(i+1)}(t)^T \quad \tilde{p}^{(i+1)}(t)^T]^T$ as the notation for the intermediate step.

Thus let the gradient step be written as

$$\tilde{y}^{(i+1)}(t) = y^{(i)}(t) - K_y \left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] \tag{5.11}$$

This is Step 4 of DISOPE as found in Algorithm 3.1. The momentum contribution to the updating mechanism is then written as

$$y^{(i+1)}(t) = \tilde{y}^{(i+1)}(t) + W_y(y^{(i)}(t) - y^{(i-1)}(t)) \tag{5.12}$$

This will be Step 5 in the new DISOPE-MOMENTUM algorithm that follows.

Algorithm 5.1: DISOPE-MOMENTUM Algorithm

Data f, L, φ, x_0, t_0, t_f and means for calculating *f** and *L**.

Step 0 Compute or choose a nominal solution $\hat{u}^0(t), \hat{x}^0(t),$ and $\hat{p}^0(t)$. Set $i = 0, u^0(t) = \hat{u}^0(t)$, $x^0(t) = \hat{x}^0(t)$, $p^0(t) = \hat{p}^0(t)$, $t \in [t_0, t_f]$.

Step 1 Compute the parameters $\alpha^{(i)}(t)$, $\gamma^{(i)}(t)$ to satisfy (3.13). This is called the *parameter estimation* step.

S*tep 2* Compute the multipliers $\lambda^{(i)}(t)$ and $\beta^{(i)}(t)$ from (3.11).

Step 3 With specified $\alpha^{(i)}(t)$, $\gamma^{(i)}(t)$, $\lambda^{(i)}(t)$, and $\beta^{(i)}(t)$ solve MMOP to obtain $\hat{u}^{(i)}(t)$, $\hat{x}^{(i)}(t)$, and $\hat{p}^{(i)}(t)$. This is called the *system optimization* step.

Step 4 This step is the updating mechanism cum gradient-descent step of the algorithm.

$$\left.\begin{aligned} \tilde{u}^{(i+1)}(t) &= u^{(i)}(t) - k_u(\hat{u}(t) - u(t)) \\ \tilde{x}^{(i+1)}(t) &= x^{(i)}(t) - k_x(\hat{x}(t) - x(t)) \\ \tilde{p}^{(i+1)}(t) &= y^{(i)}(t) - k_p(\hat{p}(t) - p(t)) \end{aligned}\right\} \tag{5.13}$$

where $k_v, k_z, k_p \in (0,1]$ are scalar gains.

Step 5 This step adds momentum terms to the updates of Step 4. It also tests for the convergence of the solution to ROP.

$$\left.\begin{aligned} u^{(i+1)}(t) &= u^{0(i+1)}(t) + \varpi_u (u^{(i)}(t) - u^{(i-1)}(t)) \\ x^{(i+1)}(t) &= x^{0(i+1)}(t) + \varpi_x (x^{(i)}(t) - x^{(i-1)}(t)) \\ p^{(i+1)}(t) &= p^{0(i+1)}(t) + \varpi_p (p^{(i)}(t) - p^{(i-1)}(t)) \end{aligned}\right\} \qquad (5.14)$$

where $\varpi_u, \varpi_x,$ and $\varpi_p \in (0,1]$ are the momentum parameters. If $\left\| u^{(i+1)}(t) - u^{(i)}(t) \right\| \le \varepsilon,$ ε a given tolerance, stop, else set $i = i + 1$ and continue from step 1.

Algorithm 5.1 is the DISOPE-MOMENTUM algorithm where together Steps 4 and 5 make up the new updating mechanism. We incorporate the following flow chart to make clear Algorithm 5.1. In Fig. 5.6, the shaded box indicates the steps that are exactly the same as the ones in DISOPE.

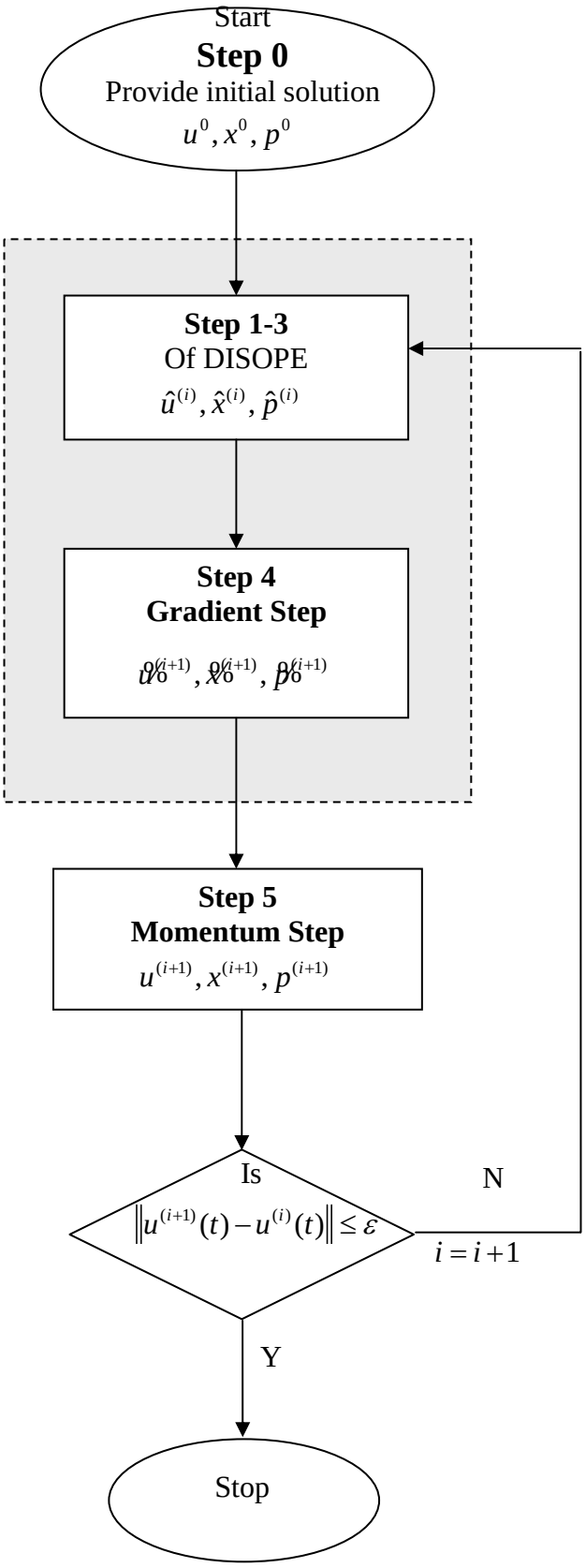


Figure 5.6: The flow chart of DISOPE-MOMENTUM algorithm.

5.5 Summary and Conclusion

The updating mechanism of DISOPE has been established as a type of a gradient descent method. As such, it inherits the problems associated with the method. In an earlier chapter, we have narrowed down the problems faced by DISOPE to only the problem of slow convergence either caused by the oscillations of the search when nearing an optimum or the slow advancement of the search when it faces a relatively flat surface.

In the effort of overcoming the problem, we observed that the equations used in the updating mechanism of DISOPE were similar in structure to the BP algorithm of the artificial neural networks. Taking advantage of the similarities, the improvement done to the BP algorithm namely the addition of momentum was studied and applied to the updating mechanism of DISOPE. The inclusion of momentum does reduce the number of iterations needed to arrive at the optimal solution. Furthermore, this move also succeeded in reducing the CPU time needed to execute the algorithm. A new algorithm named DISOPE-MOMENTUM, based on the modification was then formally developed.

In short we conclude that the addition of the momentum terms successfully overcome the problem of slow convergence in DISOPE. In the next chapter we present an alternative modification to DISOPE using the parallel tangent algorithm that can overcome the same problem.

CHAPTER 6

DISOPE-PARTAN ALGORITHM

6.1 Introduction

This chapter presents another modification to Map C, the updating mechanism of DISOPE, which successfully speeds up its convergence. Here we incorporate a mechanism called the parallel tangent (PARTAN) algorithm to the updating mechanism. This mechanism proved to be a worthy candidate for improving the performance of DISOPE algorithm.

The chapter begins with a description of the algorithm and its effects on the improvement of DISOPE. Simulations of numerical examples are presented to highlight these improvements. Next we state formally the new algorithm appropriately called the DISOPE-PARTAN algorithm. A time complexity analysis is also included to compare the efficiency of DISOPE-PARTAN to DISOPE.

6.2 Modification of Map *C*

Again in this chapter, we analyze Map *C* for the necessary alteration in order to speed up the convergence of the algorithm. As explained earlier, the modification is necessitate by the nature of Map *C*, a type of a gradient descent method. The zigzagging phenomenon is the behavior of the method that needs rectifying.

The modification highlighted in this chapter is also excerpted from the literature of BP algorithm. It is a form of the *gradient parallel tangent* method or gradient-PARTAN. Except for the first iteration, this method involves replacing every odd-numbered gradient search with a search called an acceleration step. The acceleration step helps widen the strides of the search resulting in reduced oscillation in ravines and faster crossings of flat surfaces or basins.

This method is not as popular as the method of momentum discussed earlier in Chapter 5; however, its simplicity and effectiveness rival that of the momentum. In some cases, the effectiveness of this method well surpassed the method in Chapter 5. In fact, when the error function is a 2-variables quadratic function, the convergence is guaranteed to take place in 3 iterations, making the procedure a quadratic convergent procedure (Pierre, 1969). For the *n*-dimensional case, if the error function is quadratic with a well-defined optimum, the exact optimal point is located after 2*n*-1 searches, except for round off error (Pierre, 1969).

6.2.1 The Gradient-PARTAN Method

The gradient-PARTAN algorithm uses the deflecting gradient technique that it may be considered as a special case of the conjugate gradient method (Ghorbani and Bhavsar, 1993). Conjugate gradient techniques are known to be the most effective minimization methods that use only the first derivative information (Porter, 1997). They compute the new search direction by using the gradient direction and the previous search direction. Their advantage over the optimal gradient algorithm is a faster convergence near an optimum point.

To observe the workings of gradient-PARTAN, reconsider the elliptic quadratic function depicted in Fig. 2.3 duplicated in Fig. 6.1 below (Rohanin and Mohd_Ismail, 2004). If the initial point for a gradient search is not precisely on one of the axes of the systems of ellipses, the search will follow a zigzag course from p_0 to p_1 to p_2 to p_3, etc., as depicted in Fig. 6.1 below.

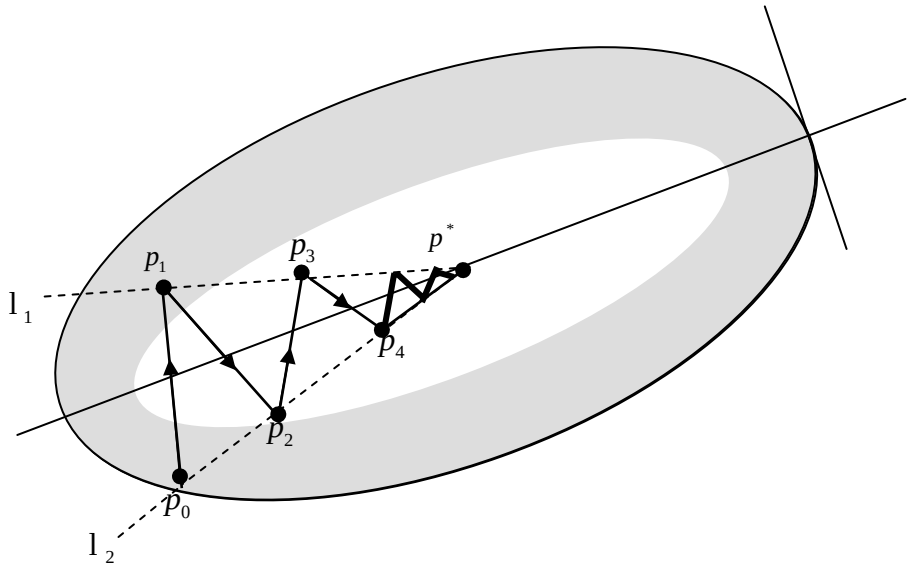


Figure 6.1: The zigzagging phenomenon.

Two straight lines l_1 and l_2 determined by the ridges of the path however, bound the path. These two lines intersect at the optimum point. This suggests that the search from p_2 could be conducted not in the gradient direction towards p_3 but along the line determined by p_0 and p_2. This would make the peak $p*$ reachable within three steps: first from p_0 to p_1 along the gradient at p_0, then from p_1 to p_2 along the gradient at p_1 and finally, along the line through p_0 and p_2 from p_2 to the peak $p*$. Fig. 6.2 illustrates the move. The last step is the parallel tangent or PARTAN step.

The search method explained above is a combination between the gradient descent method and the PARTAN method. Hence the name gradient-PARTAN. It consists of two phases, namely climbing through the gradient and accelerating through parallel tangent as shown in Fig. 2.10.

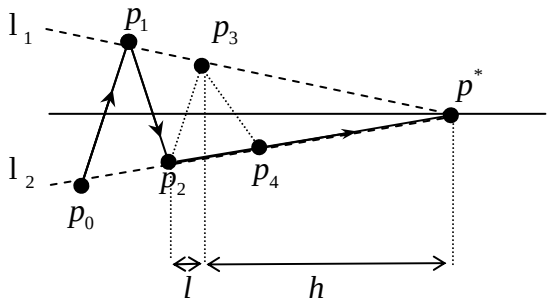


Figure 6.2: The optimum is reachable along the line through p_0 and p_2.

Gradient-PARTAN overcomes the problem of zigzagging by deflecting the gradient step along the ridge (Ghorbani and Bayat, 2000) and it overcomes the problem of slow crossing of flat surfaces by widening the strides of the search. Its simplicity and ridge following property is very desirable. Ghorbani and Bayat (2000) claimed that regardless of the degree of the complexity of the problem used, the gradient-PARTAN algorithm shows at least two times faster convergence to the solution than the gradient method alone.

6.2.2 The Incorporation of PARTAN Step

To modify Map *C* to become a gradient-PARTAN search, the PARTAN step or the acceleration step is to be incorporated in it. We look into this by re-examining the workings of the gradient-PARTAN method. We refer to Fig. 6.3 for the subsequent analysis.

The method alternately performs a gradient descent step followed by an acceleration step. To begin the gradient-PARTAN search given an initial solution p_0, we search for p_1 and p_2 using the gradient descent search. Thus in Fig. 6.3, the vectors **a** and **b** are vectors from the gradient method. To get p_3 we perform the acceleration step. This step locates p_3 along the line through p_0 and p_2 in the

direction of the optimal solution. The vector $\mathbf{c} = \overrightarrow{p_0 p_2}$ gives the direction of the search. In Fig. 6.3, the search for p_3 was done by a multiple of **c,** the vector α**c**, with $\alpha \in (0,1]$.

Let $p_0 = y^{(i-1)}$, $p_1 = y^{(i)}$, $p_2 = y^{(i+1)}$, and $p_3 = y^{(i+2)}$. Following the above analysis and with the help of Fig. 6.3, we modeled the searches with the following equations. From Eq. (4.9), the gradient search for the term $y^{(i+1)}$, is given by the equation

$$y^{(i+1)}(t) = y^{(i)} - K_y \left[\frac{\partial E_y^{(i)}(t)}{\partial y^{(i)}(t)} \right] \tag{6.1}$$

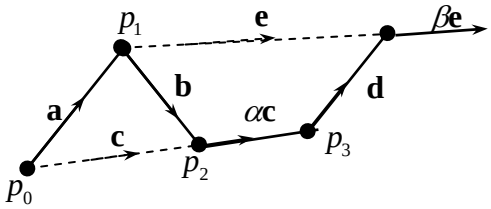


Figure 6.3: The vectors involved in the general gradient-PARTAN search.

The next term given by the acceleration step would then be

$$y^{(i+2)} = p_3 = p_2 + P_y \left[p_2 - p_0 \right] \tag{6.2}$$

or

$$y^{(i+2)}(t) = y^{(i+1)} + P_y \left[y^{(i+1)} - y^{(i-1)} \right] \tag{6.3}$$

where

$$P_y = \begin{bmatrix} \wp_u I_m & 0 & 0 \\ 0 & \wp_x I_n & 0 \\ 0 & 0 & \wp_p I_n \end{bmatrix} \tag{6.4}$$

with $\wp_u, \wp_x,$ and $\wp_p \in (0,1]$ as the PARTAN parameters.

One complete cycle of the gradient-PARTAN consists of one gradient search

followed by an acceleration search. Thus in Fig. 6.3, vector **d** is a gradient direction for the search of the next estimate. After that, β**e** is the direction of the acceleration step that follows. The alternating searches would continue until the optimal solution is found.

To summarize the gradient-PARTAN technique, we state the following Procedure 6.1 (Rohanin and Mohd_Ismail, 2004), which out lines the algorithm. In the procedure, *i* represents the iteration number.

Procedure 6.1

Begin

if $i = 1$

do one gradient step

else if i is even

do one gradient step

else

do one acceleration step

end

In short, with the gradient-PARTAN technique, Eq. (6.1) and (6.3) alternately act as the updating mechanism of the modified DISOPE. This modified algorithm is called DISOPE-PARTAN.

6.3 The Effects of Gradient-PARTAN

From the example in Fig. 6.1, we see that when using the gradient search, the optimum is reachable from p_2 in three gradient searches. The first search is from p_2 to p_3, the second search is from p_3 to p_4, and the third search is from p_4 to p^*. Comparing that to Fig. 6.2, the most prominent effect of the PARTAN step is the by passing of both p_3 and p_4 to reach p^*. We needed only one acceleration step from

p_2 to p^*. Thus for this particular case with the PARTAN technique we are saving two unnecessary searches to arrive at the optimum. This by passing of p_3 and p_4 also means a reduction in the oscillation of the search.

The next important effect of the PARTAN step is the widening of the strides of the search. Analyzing Fig. 6.2 further, we see that with the gradient search, the length of the stride from p_2 to p_3 is l units. On the other hand, the PARTAN step has a stride length of $l + h$ units. This lengthening of strides also means that the search can now move faster on flat surfaces. This would inevitably translate into faster accessibility of p^*.

In general, with the gradient-PARTAN method, where the PARTAN steps are alternated with gradient searches again and again (Rohanin and Mohd_Ismail, 2004), the cumulative effects of the cut in the number of searches and the lengthening of strides would make the final number of searches remarkably less and the CPU time reduced. In short the altered Map *C* succeeded in overcoming the problems of oscillation and the slow advancement on flat surfaces of DISOPE.

The following section presents the simulations of numerical examples to demonstrate the effects of the gradient-PARTAN approach on the convergence speed of DISOPE-PARTAN. These examples are taken from Chapter 3. The results obtained from these simulations are then compared to the results of Chapter 3.

6.3.1 Numerical Examples

Numerical examples are simulated to see the effects of the gradient-PARTAN adaptation of the updating mechanism of DISOPE. The problems used in the simulations are problems from Examples 3.1 and 3.2 of Chapter 3.

Example 6.1

This first example is taken from Example 3.1. A few of the cases simulated there are used here. All the values of the input parameters for these cases are kept the same. With the incorporation of the PARTAN step, the additional inputs for these simulations are the PARTAN parameters. The results of the simulations are tabulated in Table 6.1.

Table 6.1: The algorithm's performance of Example 6.1 with the incorporation of the PARTAN step.

Case	r_1	k_v	(a)	J_1^*	CPU time (s)	$\wp_y$	(b)	J_2^*	CPU time (s)
i	0	1	10	0.028	1.832	[0.05 0 0]	8	0.028	1.192
ii	0	0.3	9	0.028	1.683	[0.1 0 0]	7	0.028	1.022
iii	0.5	1	9	0.028	1.733	[0.05 0 0]	5	0.028	0.751
iv	1	1	12	0.028	2.173	[0.1 0 0]	7	0.028	0.951
v	1	1	12	0.028	2.173	[0.05 0 0]	5	0.028	0.711
vi	1	0.8	13	0.028	2.444	[0.1 0 0]	7	0.028	0.971
vii	1	0.8	13	0.028	2.444	[0.05 0 0]	5	0.028	0.741

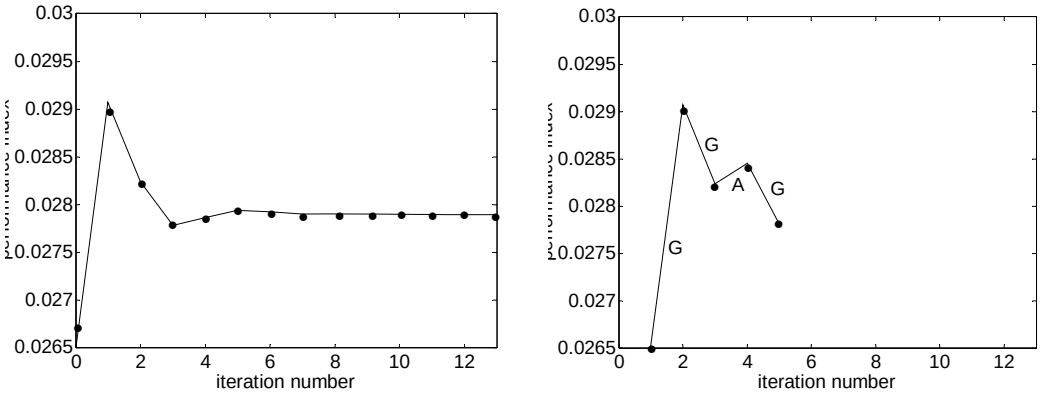
Table 6.1 compares the performance of DISOPE with the performance of DISOPE-PARTAN. Column (a) records the number of iterations for convergence of DISOPE. Column (b) records the number of iterations for convergence of DISOPE-PARTAN. Both J_1^* and J_2^* respectively record the final values of the performance indices of both algorithms. Column $\wp_y$ registers the values chosen for the PARTAN parameters. These values are user supplied and in these cases, they are chosen by trial and error. However, the values that we record are the best values for conveying our point.

As expected, in all the cases, DISOPE-PARTAN algorithm succeeded in reducing the number of iterations needed for convergence. Comparing the results recorded in the columns for CPU time, we see that the new algorithm also succeeded in reducing the time taken to arrive at the optimal solution.

Moreover, starting with one set of inputs, as in Cases (iii) and (iv), the different sets of values given to the PARTAN parameters, produce different speed of convergence to the algorithm. For a relatively simple problem as the one simulated for Example 6.1, this gradient-PARTAN algorithm managed to reduce the number of iterations up to basically less than half the original number recorded for DISOPE.

To further explore the effects of PARTAN, we graph the performance indices of DISOPE and DISOPE-PARTAN in Fig. 6.4. We are using the results of Case (iv) for both figures. Fig. 6.4(a) is the graph of the original performance index using DISOPE. The optimal solution was obtained after twelve iterations. Fig. 6.4(b) is the graph of the performance index after using DISOPE-PARTAN. The solution was reached only after five iterations. By comparing Figs. 6.4 (a) and (b) we can see that the acceleration step of PARTAN only takes effect at the third iteration. Hence after, the PARTAN and gradient steps alternate.

In Fig. 6.4 (a), the graph of the performance index diminishes into a very gentle slope after the seventh iteration. This gentle slope happens when consecutive values of the performance index do not differ significantly from each other. This must have happened whilst the gradient-descent algorithm oscillates when nearing the optimum. In Fig. 6.4 (b) however, the situation is efficiently remedied by PARTAN by eliminating the redundant iterations. The result is a faster convergence.



(a) (b)

Fig. 6.4: The comparisons between the performance indices of (a) DISOPE; (b) DISOPE-PARTAN.

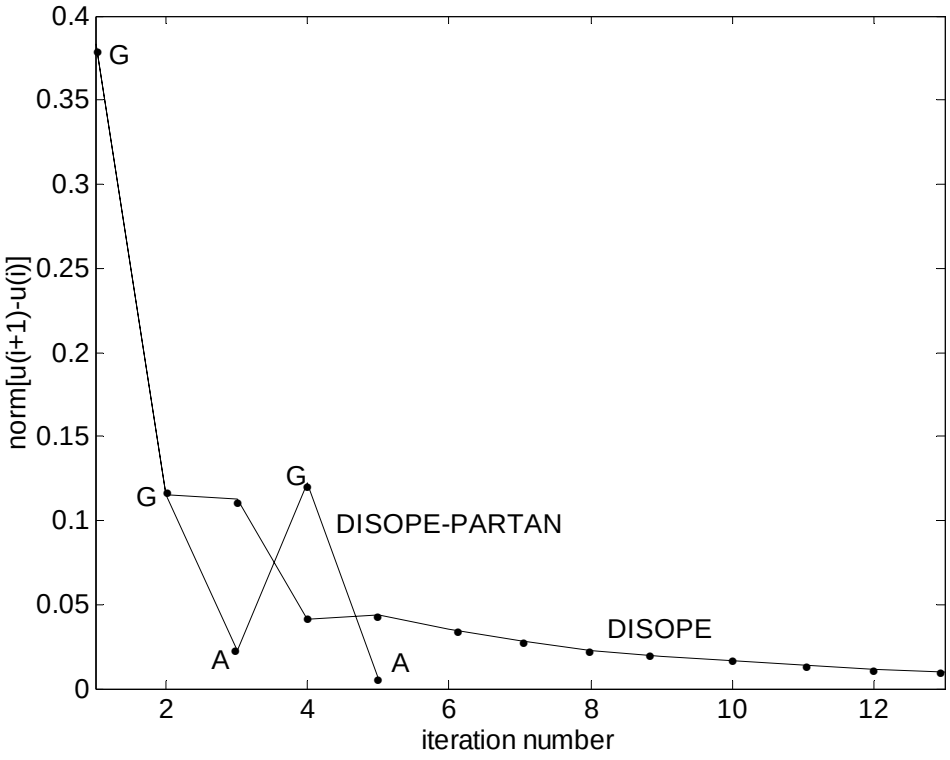


Fig. 6.5: The comparisons between the control variation norms of DISOPE and DISOPE-PARTAN of Example 6.1.

Fig. 6.5 is the graphs of the control variation norms, $\left\| u^{i+1}(t) - u^{i}(t) \right\|$, of Case (v) for both algorithms. For the purpose of stopping the iterations of both algorithms, this norm was set to be $\left\| u^{i+1}(t) - u^{i}(t) \right\| \le 0.01$. The purpose of this depiction is to observe the convergence pattern of DISOPE-PARTAN compared to DISOPE.

In Fig. 6.5, 'G' stands for the values of the control norm after application of the gradient step. 'A' stands for like values after application of the acceleration step. For this particular case, the pattern for convergence of DISOPE closely resembles a monotonic convergence. The values reduced slowly to the given tolerance.

DISOPE-PARTAN one the other hand has an oscillating convergence pattern. This is typical of the gradient-PARTAN algorithm where every pair of trials solutions is generated through two different methods. From the graph, we see that the norm was sharply reduced to the given tolerance after the fifth iteration. The

search was terminated at the fifth iteration. Without a doubt, DISOPE-PARTAN succeeded in achieving what it is designed for, accelerating the gradient descent algorithm.

Example 6.1 has been a simple problem. For a simple problem, the gradient-PARTAN method successfully reduces the oscillations of the search. In the next example we simulate a problem with a higher degree of nonlinearities.

Example 6.2

This second example is the third order nonlinear systems problem of Example 3.2. All the values of the basic input parameters for these cases are kept the same as that in Example 3.2. In these simulations the user supplied values of the PARTAN parameters are again chosen by trial and error.

The problem solved for this example is more complex than the problem in Example 6.1 in the sense that one of its state variables has a terminal constraint. The solution generated by the algorithm has to satisfy one additional condition on top of the standard conditions of optimality. The effective values of $\wp_y$ that we found, reduce the number of iterations. The results of the simulations are tabulated in Table 6.2. The values registered here are the values that best represent the improvements expected.

Table 6.2: Comparisons of the final performance of DISOPE and DISOPE-PARTAN for Example 6.2.

case	r_2	k_x	(a)	J_1^*	CPU time (s)	$\wp_y$	(b)	J_2^*	CPU time (s)
i.	1	0.3	12	0.663	2.794	[0.1 0.35 0]	9	0.663	1.271
ii.	1	0.4	10	0.660	2.344	[0.04 0.35 0]	7	0.659	1.051
iii.	1	0.5	8	0.659	1.973	[0.01 0.35 0]	5	0.657	0.831
iv.	1	0.7	8	0.656	1.912	[0.09 0 0]	7	0.657	1.011
v.	1	0.8	9	0.656	2.244	[0.2 0 0]	5	0.659	0.781

As in Table 6.1, Columns (a) and (b) record the number of iterations for the convergence of DISOPE and DISOPE-PARTAN respectively. Columns J_1^* and J_2^* register the final values of the performance indices of the problems when using DISOPE and DISOPE-PARTAN respectively. From these two columns, we see that the variations of the values are well within the tolerance accepted which is 0.01.

All the results cited here satisfy the conditions mentioned above. Thus all the solutions acquired by these simulations are optimal solutions. In all the cases we manage to reduce the number of iterations and the CPU time taken to find the optimal solution. Hence we succeeded in increasing the speed of convergence.

Case (iii) of the above simulations is chosen for detailed discussion. Fig. 6.6 below shows that along with three optimality conditions, the terminal condition of $x_1(2) = 0$ is also satisfied by the solution.

Next we compare the graphs of the performance indices of DISOPE and DISOPE-PARTAN in Figs. 6.7 (a) and (b). As mentioned in Example 6.1, the PARTAN algorithm only takes effect beginning at the third iteration. There after, the gradient step and the PARTAN step alternates.

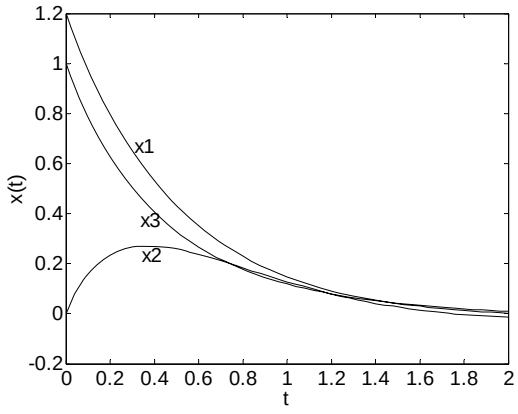
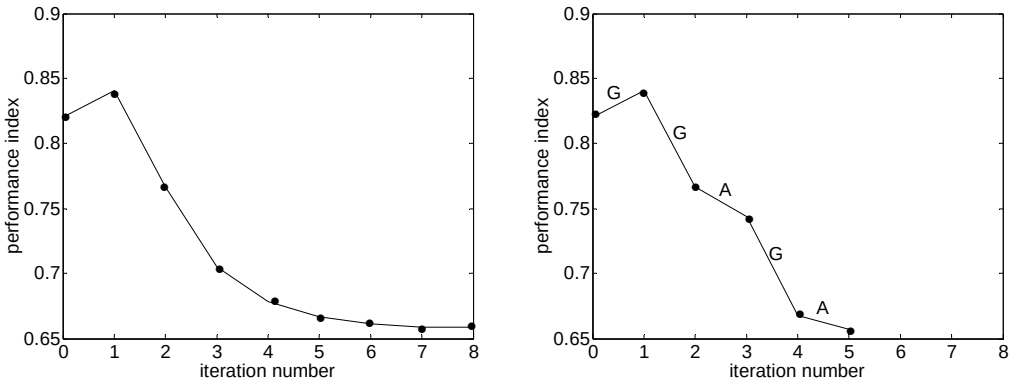


Figure 6.6: The graph showing the final states $x(t)$ of Case (iii) satisfying the end-point condition of $x_1(2) = 0$.



(a) (b)

Fig. 6.7: The comparison of the performance indices of (a) DISOPE and (b) DISOPE-PARTAN of Case (iii).

In Fig. 6.7 (a), the values of the performance index, decreases slowly after the fifth iteration suggested by the gentle slope. Once again this is caused by the oscillating gradient-descent algorithm when nearing an optimum. Fig. 6.7 (b) shows the after effect of using the gradient-PARTAN algorithm. The redundant iterations are eliminated for faster convergence.

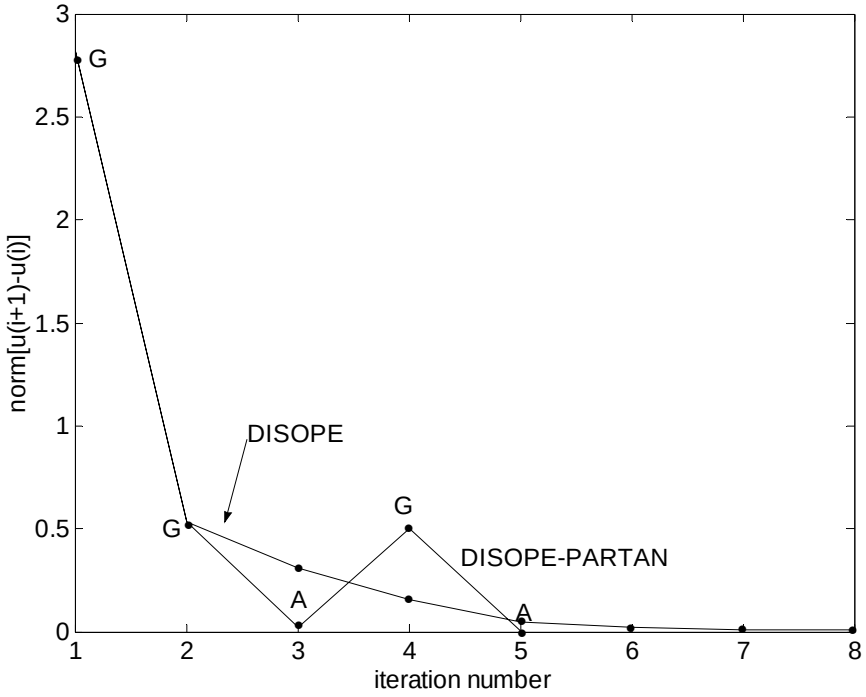


Fig. 6.8: Comparisons between the control norms of DISOPE and DISOPE-PARTAN of Case (iii).

Fig. 6.8 is the graph of the control variations norm, $\left\|u^{i+1}(t)-u^{i}(t)\right\|$, of Case (iii) for both algorithms. Again DISOPE displays monotonic convergence pattern. However the gentle slope after the second iteration suggests that the norm between two consecutive terms is gradually reduced until it satisfies the stopping criterion of the algorithm. The gradual reduction causes the algorithm three more iterations compared to DISOPE-PARTAN. The DISOPE-PARTAN algorithm on the other hand, sharply reduces the norm after the second iteration. The algorithm succeeded in increasing the convergence speed to reach the optimum.

From this section we gather that there are two significant effects of using the gradient-PARTAN method in DISOPE-PARTAN. One is the reduction in the iterations number and the other is the reduction in CPU time for convergence. To end this section we conclude that this scheme successfully does its feat. In the next section, we formally define the new DISOPE-PARTAN algorithm.

6.4 DISOPE-PARTAN Algorithm

Algorithm 6.1

Data $f,\ L,\ \varphi,\ x_0,\ t_0,\ t_f$ and means for calculating *f** and *L**.

Step 0 Compute or choose a nominal solution $\hat{u}^0(t), \hat{x}^0(t),$ and $\hat{p}^0(t)$. Set $i=0, u^0(t)=\hat{u}^0(t)$, $x^0(t)=\hat{x}^0(t)$, $p^0(t)=\hat{p}^0(t)$, $t\in[t_0,t_f]$.

Step 1 Compute the parameters $\alpha^{(i)}(t),\ \gamma^{(i)}(t)$ to satisfy (3.13). This is called the *parameter estimation* step.

S*tep 2* Compute the multipliers $\lambda^{(i)}(t)$ and $\beta^{(i)}(t)$ from (3.11).

Step 3 With specified $\alpha^{(i)}(t),\ \gamma^{(i)}(t),\ \lambda^{(i)}(t)$, and $\beta^{(i)}(t)$ solve MMOP to obtain $\hat{u}^i(t),\ \hat{x}^i(t)$, and $\hat{p}^i(t)$. This is called the *system optimization* step.

Step 4 This step is the gradient step of the updating mechanism. It updates the estimates for the solution of ROP and tests for convergence.

$$\left.\begin{aligned} u^{(i+1)}(t) &= u^{(i)}(t) + k_u(\hat{u}^{(i)}(t) - u^{(i)}(t)) \\ x^{(i+1)}(t) &= x^{(i)}(t) + k_x(\hat{x}^{(i)}(t) - x^{(i)}(t)) \\ p^{(i+1)}(t) &= p^{(i)}(t) + k_p(\hat{p}^{(i)}(t) - p^{(i)}(t)) \end{aligned}\right\} \tag{6.5}$$

where $k_v, k_z, k_p \in (0,1]$ are scalar gains. If $\left\|u^{i+1}(t) - u^{i}(t)\right\| \le \varepsilon$, ε a given tolerance, stop, else set $i = i+1$. If $i+1 = 2$, go to step 1, else proceed to step 5.

Step 5 This is the acceleration part of the updating mechanism. It updates the estimates and also tests for convergence.

$$\left.\begin{aligned} u^{(i+1)} &= u^{(i)} + \wp_u(u^{(i)} - u^{(i-2)}) \\ x^{(i+1)} &= x^{(i)} + \wp_x(x^{(i)} - x^{(i-2)}) \\ p^{(i+1)} &= p^{(i)} + \wp_p(p^{(i)} - p^{(i-2)}) \end{aligned}\right\} \tag{6.6}$$

where $\wp_u, \wp_x,$ and $\wp_p \in (0,1]$ are the PARTAN parameters. If $\left\|u^{i+1}(t) - u^{i}(t)\right\| \le \varepsilon$, ε a given tolerance, stop, else set $i = i+1$ and continue from Step 1.

Algorithm 6.1 is the DISOPE-PARTAN algorithm where the updating mechanism has been amended to include the PARTAN algorithm. It is different from DISOPE-MOMENTUM in that the original DISOPE algorithm is only used to generate the first and the even numbered terms. The odd numbered terms on the other hand are generated purely through the shift done to the even numbered terms by adding a multiple of the vector determined by the $(i+1)$ th and the $(i-1)$ th terms. The flow chart of Fig. 6.9 clarifies the state of affairs.

In Fig. 6.9 the two shaded rectangular boxes represent the two alternating steps taken by the algorithm. The bigger box represents the steps taken by the algorithm to execute one gradient descent step. The smaller of the two represents the one step taken to execute the PARTAN step.

Thus far, we have been discussing the achievement of DISOPE-PARTAN over DISOPE. In the next section we present the efficiency analysis of the new algorithm in the form of an every-case time complexity analysis. Using this analysis,

the efficiency of DISOPE-PARTAN will be compared to that of DISOPE.

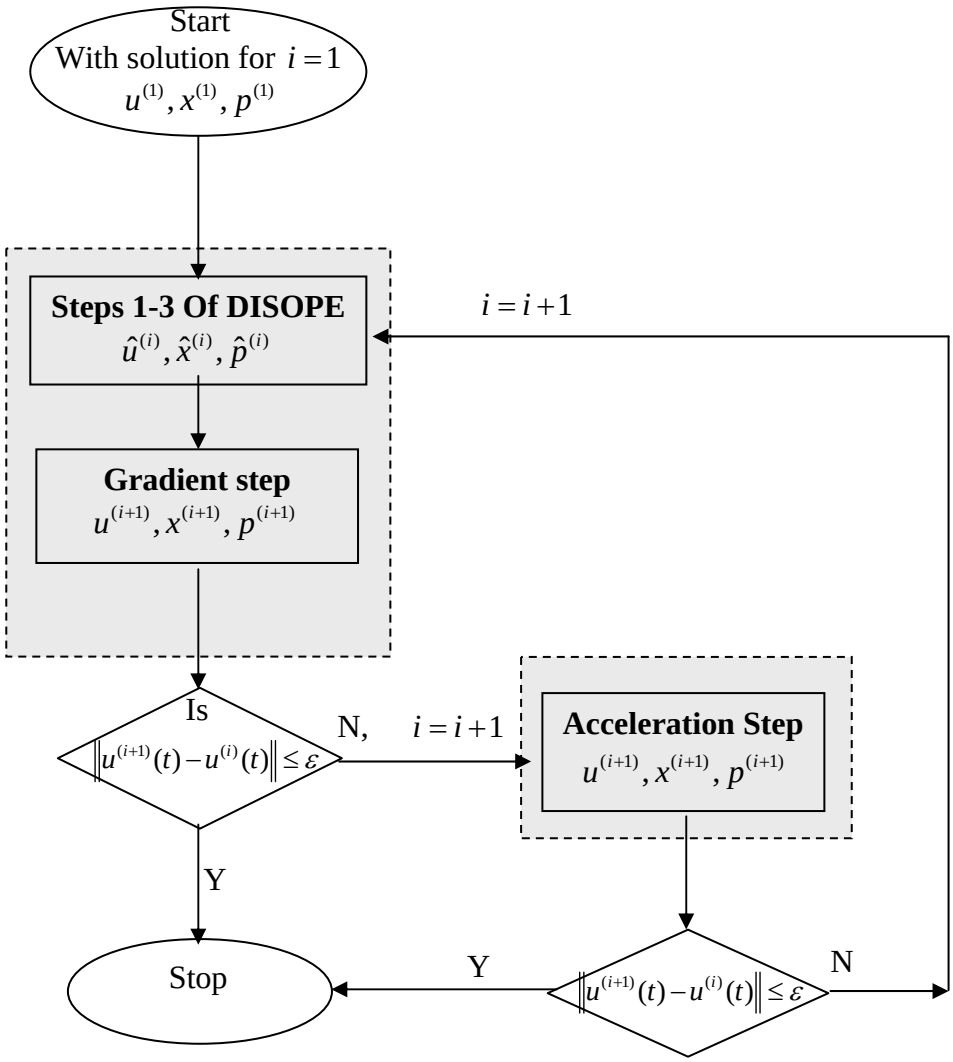


Figure 6.9: The flow chart of DISOPE-PARTAN algorithm.

6.5 Summary and Conclusion

Based on the problems faced by DISOPE in association with its updating mechanism, we proposed a method called the parallel-tangent or PARTAN algorithm to be incorporated in the updating mechanism of DISOPE with the purpose of overcoming the aforementioned problems. It is a form of conjugate gradient method known for its effectiveness near optimal points.

The incorporation of PARTAN in Map *C* sees that a gradient step is alternated with the PARTAN step. The move successfully speed up the convergence of DISOPE. The results of the simulations clearly demonstrate the effectiveness of the parallel tangent algorithm in speeding up the convergence of DISOPE. The modification of the updating mechanism successfully decreases the number of iterations needed to arrive at the optimum. Further more the CPU time needed to complete the iterations were also reduced in the process. Thus we conclude that the PARTAN algorithm succeeded in improving the convergence speed in the true sense of the word. Thus we illustrate that DISOPE-PARTAN is a more efficient algorithm than DISOPE.

CHAPTER 7

CONCLUSION

7.1 Introduction

This chapter summarizes the materials presented in the foregoing chapters. The results and findings are highlighted and conclusions are drawn from them. From these, suggestions for further research are presented.

7.2 Summary of Significant Findings

The central theme of this research is the gradient-based modifications done to the *Dynamic Integrated System Optimization and Parameter Estimation* (DISOPE) with the purpose of overcoming the slow convergence trait of the algorithm. This algorithm is designed to solve nonlinear optimal control problems.

An important part of the algorithm is the updating mechanism. This is where the trial solutions of one iteration are updated as input to the next iteration. The performance of this mechanism has been largely overlooked in the past. This mechanism is a form of the gradient descent search algorithm. Intrinsic to the method are the problems of slow convergence that might even lead to convergence to false optima. False optima can be in the form of local minima disguised as global minima or terms that are truly not optimal at all.

We observed that DISOPE inherited this problem of slow convergence. With DISOPE the problem of convergence to local optima has been overcame by the use of Linear Quadratic Regulator problem as model. Thus, we are left with the problem of slow convergence with the possibility of converging to false optima that are not local minima. Based on the gradient form of the mechanism, we proposed modifications that would improve the performance of the mechanism per se and DISOPE as a whole.

The main goal of this research is to improve the convergence speed of DISOPE via the modifications of its updating mechanism. In Chapter 1, we gave a list of objectives to be followed by the research in order to achieve this goal.

There are two possible reasons for the problem of slow convergence. The first is the oscillation of the search in areas of ravines and the second is slow advancement of the search on large flat basins on the surface of the function to be minimized. To get the appropriate methods of improvement we analyzed a well-known algorithm; the back propagation algorithm of the neural networks, that uses the gradient descent method as basis. From the literature of the back propagation algorithm, we resolved to using two of the many modifications found in an effort to improve the convergence speed. The choice has been based on the simplicity and effectiveness of the methods. Furthermore, all the information needed by the methods is readily available while executing DISOPE.

The first of the said modifications is an inclusion of momentum terms to the basic gradient descent search of the updating mechanism of DISOPE. It is one of the most popular learning paradigms in the back propagation algorithm. The momentum terms worked at reducing the oscillation of the search when it jumps over the bottom of ravines. After the search crossed a ravine, the momentum vector corrected the gradient vector by deflecting it further down the line in the direction of the optimum. These deflections make the search traverse the area faster.

The inclusion of the momentum terms, has successfully improved the convergence speed of the original DISOPE. This observation has been vindicated in Chapter 5 where simulation of problems with differing degrees of nonlinearities have been done. DISOPE with the momentum terms converged faster in the sense that the number of iterations and CPU time for each case simulated are reduced significantly. This new modified algorithm is named DISOPE-MOMENTUM.

The improvement could be explained by deflecting effect of the momentum terms. The deflection widens the stride of the search and hence the search traversed the ranives and the plateaus faster. In doing so, the momentum terms overcame the problem of slow convergence and possibly the problem of convergence to false optima too. False minima happened when two consecutive terms on plateaus are very close together that the difference between the two is so small; it satisfies the predetermined condition imposed on the stopping criteria. This makes the search stops even though the solution is not optimal. The deflection widens the stride of the search and hence reduces the likelihood that any two consecutive terms on a plateau being too close together.

The second choice of modification is the use of parallel tangent (PARTAN) algorithm. It uses deflecting gradient technique and may be considered as a special case of the conjugate gradient method. Its simplicity and ravine following properties are very attractive. This technique alternates between the use of DISOPE as a whole with PARTAN step in producing consecutive trial solutions. The part where DISOPE is used the trial solution is updated using the gradient descent algorithm. For the immediate consecutive term, the solution produced by the gradient descent method is updated using PARTAN step. PARTAN deflected the solution by using a vector determined by the ith and the (i-2) terms.

The deflecting action has basically similar effects on the convergence speed and the ability to avoid false minima as with the modification using momentum terms discussed above. The use of PARTAN step successfully overcame the problems of DISOPE. This is reflected in the results of the simulation done in Chapter 6. DISOPE with the inclusion of the PARTAN step called DISOPE-

PARTAN, excelled over DISOPE alone. In all the simulation, the new algorithm managed to reduce the number of iterations and CPU time in executing each search to the optimal solution successfully. The optimality of this algorithm is established as its optimality conditions at convergence agree with the solution of the ROP. The new algorithm also satisfies the sufficient and necessary condition for local stability. The sufficient condition for the asymptotic convergence shows that the convergence property has an extra term resulting from the coefficients of the momentum parameters compared to similar condition for DISOPE. We observed that all the other terms remained the same as DISOPE's, hence this extra term could only influenced the values for the Lipschitz constants which would have to be reduced for the condition to be satisfied. A reduction in the values of the Lipschitz would have to mean that the contraction of the terms in DISOPE-MOMENTUM is faster than the terms in DISOPE. With that we succeeded in proving a conjecture that the contraction speed of the DISOPE-MOMENTUM is faster than DISOPE hence the plausible explanation for the faster convergence speed.

With the DISOPE-PARTAN, the modifications rendered the new algorithm more efficient than DISOPE. The time-complexity function of DISOPE-PARTAN is definitely less than DISOPE. DISOPE-PARTAN also has the same convergence rate as DISOPE which is quadratic. This algorithm also satisfies the necessary optimality conditions imposed on it. It also satisfies the local and global stability conditions imposed by their respective theorems. The global convergence analysis was divided into two parts comprising of the terms generated by the gradient descent method and the terms generated by the PARTAN step. The convergence of first part is established as being the same as DISOPE. For the second part, the sufficient condition for asymptotic convergence is unique to DISOPE-PARTAN. From this analysis, we observed that the PARTAN step of the algorithm contracts faster than the gradient descent step. Thus DISOPE-PARTAN as a whole contracts faster than DISOPE, giving us the likely explanation for its faster convergence.

To summarize, the products of these modifications are two new distinct algorithms, mentioned above as DISOPE-MOMENTUM and DISOPE-PARTAN. In short, both algorithms show significant improvement in the convergence speed over

DISOPE with DISOPE-PARTAN showing a better improvement over DISOPE-MOMENTUM. To demonstrate the robustness of the new algorithms, simulation of numerical examples were done. The examples used in the simulation are examples with differing degrees of nonlinearities. When compared to the results given by DISOPE, the outcome of each simulation shows significant reduction in the number of iterations needed to arrive at the optimal solution. Further more, the CPU time needed to find the optimal solution for each example is also reduced. All these results are backed up by the appropriate theoretical analyses for each improved algorithm.

In conclusion, the research succeeded in achieving its goal; overcoming the slow convergence of DISOPE. All the objectives outlined for the research have been satisfied. The end products of the research are two new algorithms that are more efficient than the original DISOPE and are capable of solving the same nonlinear optimal control problems in shorter time with less number of iterations. The following section lists suggestions for further research.

7.3 Further Research

In this section we present some possible avenues for the research work in the future.

a) *A link with pole placements methods.* The proper choice of the systems' weights, Q and R, is a better way to handle this problem of DISOPE instability than adjusting the convexification parameters r_1 and r_2. Thus, we propose a link up with pole placement for the determination of acceptable weights to stabilize the systems before using any one of the algorithms.

b) *Extension to hierarchical platform.* For large-scale systems, the efficient approach is to handle them in a hierarchical platform.

c) *Extension to bounded systems.* So far, we have assumed that the admissible controls and states are not constrained by any boundaries. We propose that the research is extended to include systems with control and/or state constraints.

d) *Time delay in the dynamics of the system.* In real life, situations where time delays are an integrated part of the systems are abundant.

e) Extension to stochastic dynamical system.

REFERENCES

Amin, M.H. (1985). Optimal Pole Shifting for Continuous Multivariable Linear Systems. *Int. J. Control.* 41(3): 701-707.

Anderson, B.D.O. and Moore, J.B. (1971). *Linear Optimal Control*. Englewood Cliffs, New Jersey: Prentice Hall.

Anton, H. (1992). *Calculus with Analytic Geometry*. New York: John Wiley & Sons, Inc.

Attoh-Okine, N. (1999). Analysis of Learning Rate and Momentum Term in Backpropagation Neural Network Algorithm Trained to Predict Pavement Performance. *Advances in Engineering Software.* 30: 291-302.

Baldi, P. (1995). Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective. *IEEE Transactions on Neural Networks*. 6(1): 182-195.

Bar-Ness, Y. (1978). Optimal Closed-Loop Poles Assignment. *Int. J. Control.* 27(3): 421-430.

Bartle, R.G. and Sherbert, D.R. (1992). *Introduction to Real Analysis.* New York: John Wiley & Sons, Inc.

Bazaraa, M.S., Sherali, H. D., and Shetty, C.M. (1993). *Nonlinear Programming: Theory and Algorithms*. New York: John Wiley and Sons, Inc.

Beale, E. (1988). *Introduction to Optimization*. New York: John Wiley & Sons.

Becerra, V.M. (1994). *Development and Applications of Novel Optimal Control Algorithms*. City University, UK: Ph.D. Thesis.

Becerra, V.M. and Roberts, P.D. (1996). Dynamic Integrated System Optimization and Parameter Estimation for Discrete Time Optimal Control of Nonlinear Systems. *Int. J. Control.* 63(2): 257-281.

Becerra, V.M. and Roberts, P.D. (1998). Application of a Novel Optimal Control Algorithm to a Benchmark Fed-Batch Fermentation Process. *Trans. Inst. MC*.

20**(**1): 11-18.

Becker, S. and leCun, Y. (1988). Improving the Convergence of Back-propagation learning with Secong Order Methods. *Proceedings of the 1988 Connectionist Models Summer School.* June 17-26. Carnegie Mellon University. San Mateo: Morgan Kaufmann Publishers. 29-37.

Bellman, R. (1957). *Dynamic Programming*. Princeton, N.J.: Princeton University Press.

Bellman, R. (1969). *Stability Theory of Differential Equations*. New York: Dover Publications, Inc.

Bertsekas, D.P. (2001). Neuro-Dynamic Programming: An Overview. *Encyclopedia of Optimization.* Dordrecht: Kluwer Academic Publishers.

Bertsekas, D.P., Homer, M.L., Logan, D.A., Patek, S.D., and Sandell, N.R. (2000). Missile Defense and Interceptor Allocation by Neuro-Dynamic Programming. *IEEE Transactions on Systems, Man, and Cybernetics.* A-30(1): 101-110.

Boland, F.M., Owens, D.H. (1980). Linear Multipass Processes: a Two-Dimensional Interpretation. *IEE Proceedings. Part D.* 127 (5): 189-193.

Boyce, W. E. and DiPrima, R.C. (2001). *Elementary Differential Equations*. New York: John Wiley & Sons.

Brdys, M. and Roberts, P.D. (1987). Convergence and Optimality of Modified Two-Step Algorithm for Integrated Sysyem Optimization and Parameter Estimation. *International Journal of System Science.* 18: 1305-1322.

Brdys, M., Chen, S. and Roberts, P.D. (1986). An Extension to the Modified Two-Step Algorithm for Steady-State System Optimization and Parameter Estimation. *Int. J. Systems Science.* 17: 1229-1243.

Brdyś, M., Ellis, J.E., and Roberts, P.D. (1987) "Augmented Integrated System Optimization and Parameter Estimation Technique: Derivation, Optimality and Convergence." *IEE Proceedings Part 3.* 14(3): 201-209.

Brogan, B.L. (1991). *Modern Control Theory*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Brown, S.C. and Passino K.M. (1997). Intelligent Control for an Acrobot. *Journal of Intelligent and Robotic Systems.* 18: 209-248.

Bryson, A.E. (1996). Optimal Control – 1950 to 1985. *IEEE Control Syst. Mag.* 16(3): 26-33.

Bryson, A.E. and Ho, Y. (1975). *Applied Optimal Control*. Washington D.C: Hemisphere Publishing Corporation.

Buchanan, J.L. and Turner, P.R. (1992). *Numerical Methods and Analysis*. New York: McGraw-Hill, Inc.

Bunday, B. (1984). *Basic Optimization Methods*. London: Edwards Arnold.

Cater, J.P. (1987). Successfully Using Peak Learning Rates of 10 (and greater) in Back-propagation Networks with the Heuristic Learning Algorithm. *IEEE First International Conference on Neural Networks.* San Diego, CA, II: IEEE, 645-652.

Cesari, L. (1983). *Optimization – Theory and Applications: Problems with Ordinary Differential Equations*. New York: Springer-Verlag.

Clements, D.J., Teo, K.L., and Wu, Z.S. (1982). An Implementable Algorithm for Linear Time Optimal Control. *Int. J. Systems Sci.* 13(11): 1223-1232.

Craig, J.C. (1989). *Introduction to Robotics*. Reading, USA: Addison-Wesley.

Dayhoff, Judith E., (1990). *Neural Network Architectures: An Introduction*. New York: Von Nostrand Reinhold.

Derouin, E., J. Brown, H. Beck, L. Fausett, & M. Schneider. (1991). Neural Network Training on Unequally Represented Classes. In: Dagli, C.H., Kumara, S.R.T., and Shin, Y.C. eds. *Intelligent Engineering Systems Through Artificial Neural Networks.* New York: ASME Press. 153-141.

Dorato, P. and Abdallah, V.C. (1995). *Linear-Quadratic Control: An Introduction*. Englewood Cliffs, N.J.: Prentice Hall.

Eastman, W.L. and Bossi, J.A. (1984). Design of Linear Quadratic Regulators with Assigned Eigenstructure. *Int. J. Control.* 39(4): 731-742.

Edwards, C.H. and Penney, D.E. (1994). *Calculus with Analytic Geometry*. Englewood Cliffs, New Jersey: Prentice-Hall.

Edwards, C.H. and Penney, D.E. (2000). *Differential Equations and Boundary Value Problems: Computing and Modeling*. New Jersey: Prentice Hall International, Inc.

Edwards, J.B. (1974). Stability Problems in the Control of Multipass Processes. *Proc. IEE.* 121 (11): 1425-1432.

Edwards, J.B. and Owens, D.H. (1982). *Analysis and Control of Multipass Processes*. Chichester: Wiley.

Ellis, J.E. and Roberts, P.D. (1981). Simple Models for Integrated Optimization and Parameter Estimation. *International Journal of Systems Science.* 12: 442-472.

Fahlman, S.E. (1988). Faster-Learning Variations on Back-Propagation: An Empirical Study. In: Touretsky, D., Hinton, G., and Sejnowski, T. eds. *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann. 38-51.

Fausett, L. (1994). *Fundamental of Neural Networks: Architectures, Algorithms, and applications*. New Jersey: Prentice-Hall, Inc.

Fletcher, R. (1979). "Practical Methods of Optimization." Vol. 1, Unconstrained Optimization. New York: John Wiley.

Fornasini, E. and Marchesini, G. (1978). Doubly-indexed dynamical systems: state space models and structural properties. *Math. Syst. Theory.* 12: 1502-1517.

Fujikana, T. and Omatu, S. (2001). Pole Placement Using Optimal Regulators. *T. IEE Japan Part C.* 121 (1): 240-245.

Fukuoka, Y., Matsuki, H., Minamitani, H., and Ishida, A. (1998). A Modified Back-propagation Method to Avoid False Local Minima. *Neural Networks*. 11: 1059-1072.

Gautschi, W. (1997). *Numerical Analysis*. Boston: Birkhäuser.

Ghorbani, A.A. and Bayat, L. (2000). Accelerated Backpropagation Learning: Extended Dynamic Parallel Tangent Learning Optimization Algorithm. In: Hamilton, H. ed. *Lecture Notes in Artificial Intelligence 1822.* Springer-Verlag.

Ghorbani, A.A. and Bhavsar, V.C. (1993). Accelerated Backpropagation Learning: Parallel Tangent Learning Optimization Algorithm. *Proceedings of the 1993 International Symposium on Nonlinear Theory and Its Applications.* Hawaii, USA: NOLTA, 59-62.

Golden, Richard M. (1996). *Mathematical Methods for Neural Network Analysis and Design*. Cambridge: The MIT Press.

Gopal, M. (1984). *Modern Control System Theory.* New Delhi: Wiley Eastern Limited.

Hagiwara, M. and Sato, A. (1995). Analysis of Momentum Term in Back-Propagation. *IEICE Trans. Inf. & Syst*. E78D(8): 1080-1086.

Haimes, Y.Y. and Wismer, D.A. (1972). A Computational Approach to the Combined Problem of Optimization and Parameter Estimation. *Automatica.* 8: 337-347.

Harvey, C.A. and Stein, G. (1978). Quadratic Weights for Asymptotic Regulator Properties. *IEEE Transaction on Automatic Control.* AC-23(3): 378-387.

Hassan, M. and Singh, M. (1976). The Optimization of Non-Linear Syatems Using a New Two Level Method. *Automatica.* 12: 359-363.

Haykin, S. (1994). *Neural Networks*. New York: Macmillan College Publishing Company.

Hocking, L. (1991). *Optimal Control: An Introduction to the Theory with Applications*. Oxford: Clarendon Press.

Jacobs, O.L.R. (1974). *Introduction to Control Theory*. Oxford: Clarendon Press.

Jacobs, R.A. (1988). Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*. 1(4): 295-307.

Kamarthi, S.V. and Pittner, S. (1999). Accelerating Neural Network Training Using Weight Extrapolations. *Neural Networks.* 12: 1285-1299.

Kautsky, J., Nichols, N.K., and Van Dooren, P. (1985). Robust Pole Assignment in Linear State Feedback. *Int. J. Control.* 41(5): 1129-1155.

Kawasaki, N. and Shimemura, E. (1893). Detemining Quadratic Weighting Matrices to Locate Poles in a Specified Region. *Automatica.* 19(5): 557-560.

Kirk, D.E. (1970). *Optimal Control Theory: An Introduction*. New Jersey: Prentice Hall Inc.

Kirkwood, J.R. (1989). *An Introduction to Analysis*. Boston: PWS-Kent Publishing Company.

Kogan, J. (1995). *Robust Stability and Convexity: An Introduction*. London: Springer-Verlag.

Koo, D. (1977). *Elements of Optimization: with Applications in Economics and Business*. New York: Springer-Verlag.

Kramer, A.H., & Sangiovanni-Vincentelli (1988). Efficient Parallel learning algorithms for neural networks. In Touretzky, D.S. ed. *Advances in Neural Information Processing Systems.* San Mateo: Morgan Kaufmann. 40-48.

Kundu, S. and Ubhaya, V.A. (2001). Fitting a least Squares Piecewise Linear Continuous Curve in Two Dimensions. *Computers and Mathematics with*

Applications. 41. 1033-1041.

Lam, J., Yan, W., and Hu, T. (1999). Pole Placement with Eigenvalue and Stability Robustness. *Int. J. Control.* 72(13): 1165-1174.

Lee Y., Oh S.S., & Kim M.W. (1993). An Analysis of Premature Saturation in Back Propagation Learning. *Neural Networks.* 6: 719-728

Lewis, F.L. and Syrmos, V.L. (1995). *Optimal Control.* New York: John Wiley & Sons, Inc.

Lewis, F.L., Abdallah, C.T., and Dawson, D.M. (1993). *Control of Robot Manipulators*. New York: Macmillan Publishing Company.

Mahmoud, M.S., Hassan, M.F., Darwish, M.G. (1985). *Large-Scale Control Systems: Theories and techniques*. New York: Marcel Dekker, Inc.

Medanic, J., Tharp, H.S., and Perkins, W.R. (1988). Pole Placement by Performance Criterion Modification. *IEEE Transaction on Automatic Control.* 33(5): 469-472.

Mohd_Ismail, B.A.A. (1999). *Development of Hierarchical Optimal Control Algorithms for the Interconnected Dynamical Systems.* City University, UK: Ph.D. Thesis.

Mohd, I.B. (1996). Kaedah Penurunan Tercuram Menggunakan Aritmetik Selang. *MATEMATIKA*. 12(1): 1-11.

Mohd_Ismail, B.A.A. and Rohanin, A. (2003). On Recent Improvements of Model Reality Based Nonlinear Optimal Control Algorithm. *Proceedings of the Annual Fundamental Science Seminar.* May 20-21. Johor Bahru: Institute Ibnu Sina, 12-20.

Mohd_Ismail, B.A.A. and Rohanin, A. (2004a). A 2-D Stability and Convergence Analysis of Optimal Control Algorithm for Systems with Model Reality Differences. Presented in: *The Annual Fundamental Science Seminar.* June 14-15. Paper no. SP52.

Mohd_Ismail, B.A.A. and Rohanin, A. (2004b). Development of an Improved Algorithm for Optimal Control of Nonlinear Dynamical System Based on Model-Reality Differences. Presented in: *International Conference on Statistics and Mathematics and Its Application in the Development of Science and Technology.* Oct. 4-6. Universitas Islam Bandung, Indonesia.

Moreira, M. and E. Fiesler. (1995). Neural Networks with Adaptive Learning Rate

and Momentum Terms. *IDIAP Technical Report*. Valais, Switzeland. 95-04.

Neapolitan, R., and Naimipour, K., (1996). *Foundations of algorithms*. Lexington: D.C. Heath and Company.

Noton, M. (1972). *Modern Control Engineering*. New York: Pergamon.

Ochiai, K., Toda, N., and Usui, S. (1994). Kick-out Learning Algorithm to Reduce the Oscillation of Weights. *Neural Networks*. 7(5): 797-807.

Omidvar, O and Elliott, D.L. (1997). *Neural Systems for Control*. San Diego: Academic Press.

Owens, D.H. (1977). Stability of Linear Multipass Processes. *Proceedings of the Institutions of Electrical Engineers: Control & Science.* 124(11): 1079-1082.

Pagilla, P.R. and Tomizuka, M. (2001). An Adaptive Output Feedback Controller for Robot Arms: Stability and Experiments. *Automatica.* 37: 983-995.

Parasini, T. and Zoppoli, R. (1994). Neural Networks for Feedback Feedforward Nonlinear Control Systems. *IEEE Trans. On Neural Networks.* 5**(**3): 436-449.

Parks, P.C. and Hahn, V. (1993). *Stability Theory*. Hertfordshire: Prentice Hall.

Perantonis, S.J. & Karras D.A. (1995). An Efficient Constrained Learning Algorithm With Momentum Acceleration. *Neural Networks.* 8(2): 237-249.

Pierre, A.D. (1969). *Optimization Theory with Applications*. New York: John Wiley & Sons, Inc.

Plumer, Edward S. (1996). Optimal Control of Terminal Processes Using Neural Networks. *IEEE Trans. On Neural Networks.* 7(2): 408-418.

Polak, E., 1997. *Optimization: Algorithms and Consistent Approximations*. New York: Springer-Verlag New York, Inc.

Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., and Mishchenko, E.F. (1962). *The mathematical Theory of Optimal Processes*. New York: Wiley Interscience.

Porter, B. (1969). *Synthesis of Dynamical Systems*. London: Thomas Nelson and Sons Ltd.

Qian, N. (1999). On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks*. 12: 145-151.

Qiu, G., Varley, M.R., and Terrel, T.J. (1992). Accelerated Training of Backpropagation Networks by Using Adaptive Momentum Step. *Electronics Letters*. 28(4): 377-379.

Rao, S. (1984). *Optimization: Theory and Application*. New Delhi: Wiley Eastern Limited.

Rice, B.J. (1972). *Applied Analysis for Physicists and Engineers*. Boston: Prindle,Weber & Schmidt, Inc.

Roberts, C.E. (1979). *Ordinary Differential Equations: A Computational Approach*. New Jersey: Prentice-Hall, Inc.

Roberts, P.D. (1979). An Algorithm for Steady State System Optimization and Parameter Estimation. *International Journal System Science.* 10: 719-734.

Roberts, P.D. (1993). An Algorithm for Optimal Control of Nonlinear Systems with Model-Reality Differences. *12th IFAC World Congress on Automatic Control.* 8: 407-412.

Roberts, P.D. (1994a). Unit Memory Repetitive Process Aspects of Iterative Optimal Control Algorithms. *Proceedings of the 33rd Conference on Decision and Control.* December 14-16. Lake Buena Vista, Florida: IEEE, 2: 1394-1399.

Roberts, P.D. (1994b). Unit Memory Repetitive Processes and Iterative Optimal Control Algorithms. *Proceeding of IEEE Int. Conf. Control '94.* 454-459.

Roberts, P.D. (1995). Coping with Model-Reality Differences in Industrial Process Optimization – A Review of Integrated System Optimization and Parameter Estimation (ISOPE). *Computers in Industry.* 26: 281-290.

Roberts, P.D. (1999). Stability Properties of an Iterative Optimal Control Algorithm. *Preprints of 14th World Congress of IFAC International Federation of Automatic Control.* F. Nonlinear Systems I: 269-274.

Roberts, P.D. (2000a). Broyden Derivative Approximation in ISOPE Optimizing and Optimal Control Algorithms. *Preprints of the 11th IFAC International Workshop on Control Applications of Optimization.* 1: 283-288.

Roberts, P.D. (2000b). Numerical Investigation of a Stability Theorem Arising from the 2-Dimensional Analysis of an Iterative Optimal Control Algorithm. *Multidimensional Systems and Signal Processing.* 11: 109-124.

Roberts, P.D. (2000c). Stability Analysis of Iterative Optimal Control Algorithms Modeled as Linear Unit Memory Repetitive Processes. *IEE Proceedings-Control Theory Applications.* 147(3): 229-238.

Roberts, P.D. (2002). Two-Dimensional Analysis of an Iterative Nonlinear Optimal Control Algorithm. *IEEE Transactions on Circuits and Systems-I:*

Fundamental Theory and Applications. 49(6): 872-878.

Roberts, P.D. and Becerra, V.M. (2000). Optimal Control of Nonlinear Systems Represented by Differential Algebraic Equations. *Proceeding of the American Control Conference.* June 2000. Chicago, Illinois: 762-763.

Roberts, P.D. and Williams, T.W.C. (1981). On an Algorithm for Combined System Optimization and Parameter Estimation. *Automatica.* 17: 199-209.

Roesser, E.P. (1975). A discrete state space model for linear image processing. *IEEE Trans. Auto. Control.* AC-20: 1-10.

Rogers, E. and Owens, D.H. (1992). "Stability Analysis for Linear Repetitive Processes." *Lecture Notes in Control and Information Sciences.* **175.** Berlin: Springer-Verlag.

Rohanin, A. and Mohd Ismail, B.A.A. (2002). The Convergence Analysis of the Updating Mechanism of the Dynamic Integrated Systems Optimization Algorithm. *Prosiding Simposium Kebangsaan Sains Matematik Ke-10*. December 23-24. Johor Bahru: PERSAMA, 326-332.

Rohanin, A. and Mohd Ismail, B.A.A. (2003a). Newton-like Properties of the Updating Mechanism of a Model-Reality Differences Algorithm. *MATEMATIKA.* 19(1): 1-13.

Rohanin, A. and Mohd Ismail, B.A.A. (2003b). Using Weights as Tools in Handling Systems Instability in Model-Reality Optimal Control Algorithm. *Proceeding of the 7th WSEAS International Conference on Systems.* July 7-10. Corfu, Greece: WSEAS, Paper No.: 457-204.

Rohanin, A. and Mohd Ismail, B.A.A. (2003c). Using Weights as Tools in Handling Systems Instability in Model-Reality Optimal Control Algorithm. In: Mastorakis,N.E., Stathopulos, I.A., Manikopoulos, C., Antoniou, G.E., Mladenov, V.M., and Gonos, I.F. *Computational Methods in Circuits and Systems Applications.* Greece: WSEAS Press. 134-139.

Rohanin, A. and Mohd Ismail, B.A.A. (2003d). An Iterative Algorithm for Optimal Control of a Robot Manipulator. *Proceeding of Malaysia-Japan Seminar on Artificial Intelligence Applications in Industry.* July 24-25. Kuala Lumpur, Malaysia.

Rohanin, A. and Mohd Ismail, B.A.A. (2004). Accelerating the Convergence of the Dynamic Integrated Systems Optimization and Parameter Estimation

Algorithm by Way of Parallel Tangent. *Jurnal Teknologi*. 40: 21-30.

Rohanin, A., Mohd_Ismail, B.A.A., and Becerra, V.M. (2002). Modification of the Updating Mechanism of the Dynamic Integrated Systems Optimization and Parameter Estimation Using the Momentum Term. Presented in: *The National Conference on Management Science/ Operations Research.* May 26-29.

Rosenthal, J. and Willems, J.C. (1998). Open Problems in the Area of Pole Placement. In: Blondel, V.D., Sontag, E.D., Vidyasager, M., and Willems, J.C. eds. *Open Problems in Mathematical Systems and Control Theory.* Berlin: Springer-Verlag. 181-191.

Rumelhart, D., Hinton, G.E., & Williams, R.J. (1986a). Learning representations by back-propagating error. *Nature*. 323: 533-536.

Rumelhart, D.E. and McClelland, J.L. (1986b). *Parallel Distributed Processing*. Cambridge: MIT Press.

Sage, A.P. and White, C.C. (1977). *Optimum Systems Control.* Second Edition. New Jersey: Prentice Hall.

Saif, M. (1989). Optimal Linear Regulator Pole-Placement by Weight Selection. *Int. J. Control.* 50**(**1): 399-414.

Sato, A. (1991). An analytical study of the momentum term in a back-propagation algorithm. In: Kohonen, T., Makisara, K., Simula,O., and Kangas, J. eds. *Artificial Neural Networks*. New York: Elsevier. 617-622.

Schwartz, A.L. (1996). *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. Univ. of California at Berkeley: Ph.D. Thesis.

Shang, Y. (1997). *Global Search Method for Solving Nonlinear Optimization Problems.* University of Illinois at Urbana-Champaign: Ph.D. Thesis.

Silva, F.M., & L.B. Almeida. (1990). Acceleration Techniques for the Back propagation Algorithm. *Lecture Notes in Computer Science*. 412: 110-119.

Singh, M.G. and Titli, A.B. (1978). *Systems: Decomposition, Optimization and Control*. Oxford: Pergamon Press.

Singhal, S. and Wu, L. (1989). Training Feedforward Networks with the extended Kalman Algorithm. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.* Scotland: 1187-1190.

Siouris, G. M. (1996). *An Engineering Approach to Optimal Control and*

Estimation Theory. New York: John Wiley & Sons, Inc.

Solheim, O.A. (1972). Design of Optimal Control Ststems with Prescribed Eigenvalues. *Int. J. Control.* 15(1): 143-160.

Solomon, R. and Leo van Hemmen, J. (1996). Accelerating Backpropagation Through Dynamic Self-Adaptation. *Neural Networks*. 9(4): 589-601.

Sontag, E.D. (1999). Stability and Stabilization: Discontinuities and the Effect of Disturbances. In: Clarke, H. and R Stern, J. eds. *Proceeding of NATO Advanced Study.* Montreal: Kluwer. 551-598.

Spong, M. (1995). "The Swing Up Control Problem For the Acrobot." *IEEE Control Systems.* Vol.15: 49-55.

Stevenson, I.A., Brdyś, M., and Roberts, P.D. (1985). Integrated System Optimization and Parameter estimation for Traveling load Furnace Control. In: Barker, H.A. and Young, P.C. eds. *Identification and System Parameter Estimation.* Oxford: Pergamon Press.

Stoer, J. and Witzgall, C. (1970). *Convexity and Optimization in Finite Dimensions I.* Berlin: Springer-Verlag.

Tatjewski, P., Abdullah, N., and Roberts, P.D. (1986). Comparative study and Development of Integrated Optimization and Parameter Estimation Algorithms for Hierarchical Steady-State Control. *Int. J. Control.* 51(2): 421-443.

Taylor, A.E. and Mann, W.R. (1972). *Advanced Calculus*. New York: John Wiley & Sons, Inc.

Teo, K.L., Wu, Z.S., and Clements, D.J. (1981). Computational Method for Convex Optimal Control Problems Involving Linear Hereditary Systems. *Int. J. Systems Sci.* 12(9): 1045-1060.

Tesauro, G., & Janssens, B. (1988). Scaling Relationships in back propagation learning. *Complex Systems*. 39-44.

Tollenaere, T. (1990). SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks.* 3: 561-573.

Van Ooyen, A. and Nienhuis, B. (1992). Improving the Convergence of the Back-propagation Algorithm. *Neural Networks.* 5: 465-471.

Watrous, R.L. (1987). Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization. In: Caudill, M. and Butler, C. eds. *Proceedings of the IEEE First International Conference on Neural*

Networks (IC|NN). San Diego: IEEE. 2: 619-627.

Weir, M. (1991). A Method for Self-Determination of Adaptive Learning Rates in Back Propagation. *Neural Networks*. 4: 371-379.

Wilf, H.S. (1986). *Algorithms and Complexity*. New Jersey: Prentice-Hall International, Inc.

Yu, Xiao-Hu and Chen, Guo-An. (1997). Efficient Backpropagation Learning Using Optimal Learning Rate and Momentum. *Neural Networks.* 10**(**3): 517-527.

Yuan, L., Achenie, L.E.K., and Jiang, W. (1996). Linear Quadratic Optimal Output Feedback Control for Systems with Poles In a Specified Region. *International Journal of Control.* 64**(**6): 1151-1164.