A Fast Sequential Linear Quadratic Algorithm for Solving Unconstrained Nonlinear Optimal Control Problems

Athanasios Sideris and James E. Bobrow
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, CA 92697
{asideris, jebobrow}@uci.edu

February 10, 2005

Abstract

We develop a numerically efficient algorithm for computing controls for nonlinear systems that minimize a quadratic performance measure. We formulate the optimal control problem in discrete-time, but many continuous-time problems can also be solved after discretization. Our approach is similar to sequential quadratic programming for finite-dimensional optimization problems in that we solve the nonlinear optimal control problem using sequence of linear quadratic subproblems. Each subproblem is solved efficiently using the Riccati difference equation. We show that each iteration produces a descent direction for the performance measure and that the sequence of controls converges to a solution that satisfies the well-known necessary conditions for the optimal control. We also show that the algorithm is a Gauss-Newton method, which means it inherits excellent convergence properties. We demonstrate the convergence properties of the algorithm with two numerical examples.

1 Introduction

For many nonlinear control applications such as robotics, it is becoming increasingly important to find controls for them that minimize a performance measure [1, 2, 3]. Ideally these controls could be computed in near real-time with good numerical convergence and stability properties. However, optimal control problems can be difficult to solve numerically for a variety of reasons. For instance, discrete-time optimal control problems over a finite time-horizon can be thought of as finite-dimensional nonlinear programming problems, where the system dynamics may create an unfavorable eigenvalue structure in the Hessian of the ensuing optimization problem. For these problems, gradient-based descent methods will be inefficient. On the other hand, even linear dynamics may lead to difficulties due to the large scale. However, in the case of linear dynamics with a quadratic performance index, the resulting optimization problem is a Quadratic Program of special structure that can be exploited to obtain an efficient algorithm for its solution. This algorithm is the well-known Linear Quadratic optimal control problem. It is reviewed in Section 2.2 for completeness.

In this paper, we focus on the case of finding optimal controls for nonlinear dynamic systems with general linear quadratic performance indexes (including tracking and regulation). We are interested primarily in systems for which the derivatives of the dynamic equations with respect to the state and the control are available, but whose second derivatives with respect to the states and the controls are too complex to compute analytically. It should be noted that there are several well-established optimal control solvers, including collocation methods [10, 13] for continuous systems, and algorithms for discrete-time systems [21, 20, 18] currently available for handling a broader class of systems and performance measures than considered in this work, but our goal is to develop an algorithm that is fast and stable enough to potentially be implemented for real-time applications and which does not require difficult to obtain information such as 2^{nd}-order system derivatives. As has been demonstrated in [4], an efficient solution to the optimal control problem can be applied to the important area of Model Predictive Control [5, 6].

In our algorithm, we linearize the system dynamics about a input/state trajectory, the current candidate for an optimal solution, and we formulate a novel *nonstandard* linear quadratic optimal control subproblem from the solution of the which we obtain a search direction for an improved control. The latter is found by a line search consisting of minimizing the performance index along this search direction and implemented by direct simulation of the system dynamics. This process is repeated, and given the structure of the proposed algorithm, we refer to it in the sequel as the *Sequential Linear Quadratic* (SLQ) algorithm. We prove that search directions produced in this manner are descent directions and that this algorithm converges to a control that locally minimizes the cost function. The solution of the linear quadratic subproblem can be reliably obtained by solving a Riccati difference equation; therefore, the proposed algorithm has very desirable numerical properties. Furthermore, we show that the search directions from our algorithm are those of the Gauss-Newton method when the original optimal control problem is reformulated as an unconstrained least-squares problem. As such our algorithm can be viewed as a highly efficient and robust implementation of the Gauss-Newton method to optimal control. Finally, we provide some analysis and demonstrate with numerical experiments the conditions required for fast convergence.

We believe that our paper makes the connection between the Gauss-Newton method and an LQR based iterative process explicit for the first time. Other papers previously have considered LQR based algorithms and used the so-called *Gauss-Newton approximation* by dropping from the Hessian of the Lagrangian the contribution of the system dynamics to construct the cost function [7, 8, 9]. But these algorithms do not result in the Gauss-Newton direction at each step, and therefore differ from our algorithm.

Algorithms similar in spirit are reported in [16, 17, 18, 19]. Such algorithms implement a Newton search, or asymptotically Newton search, but require that 2^{nd}-order system derivatives be available. Newton algorithms can achieve quadratic local convergence under favorable circumstances such as the existence of continuous 3^{rd}-order derivatives [14] or Lipschitz continuity of 2^{nd}-order derivatives, but cannot guarantee global convergence (that is convergence from any starting point to a local minimum) unless properly modified. Many systems are too complex for 2^{nd}-order derivatives to be available or even do not satisfy such strong continuity assumptions (e.g. see Section 4 for examples.) In such cases, Newton's method cannot be applied, or the quadratic convergence rate of Newton's method does not materialize. This is demonstrated in Example 1 of Section 4, where our approach is shown to be faster than Newton's method. Thus our SLQ algorithm provides an alternative to these algorithms for a reliable and fast solution of difficult optimal control problems while relying only on first derivative information.

In Section 2, we give the precise formulation of the optimal control problem considered in the paper, and a brief but complete review of a general Linear Quadratic Tracking/Regulator problem used repeatedly to obtain search directions in our algorithm. Section 3 presents the proposed algorithm, establishes its descent and convergence properties, and reveals the Gauss-Newton connection discussed above. In Section 4, we discuss two numerical examples: a vehicle moving through nonlinear viscous drag and a gas actuated hopping machine. The first example, although simple to understand, can be made arbitrarily hard by choosing the drag coefficient parameter and as such provides an excellent test bed to demonstrate and compare the efficiency and robustness of our algorithm to standard algorithms such Steepest Descent and Newton's method. The second example serves to show the potential of the proposed algorithm in solving more practical control problems. Section 5 concludes the paper.

2 Problem Formulation and Background Results

We consider the following general formulation of discrete-time optimal control problems.

$$\begin{aligned} \underset{u(n),\ x(n)}{\text{Minimize}}\ J &= \psi(x(N)) + \sum_{n=0}^{N-1} L(x(n),u(n),n) &(1)\\ \text{subject to} \quad & x(n+1) = f(x(n),u(n)); \quad x(0)=x_0, &(2)\end{aligned}$$

where for fixed time n, $x(n)$ and $u(n)$ are m-dimensional and p-dimensional vectors respectively. In the formulation above we assume a quadratic performance index, namely:

$$\begin{aligned} L(x(n),u(n),n) = \frac{1}{2}\Big([x(n)-x^o(n)]^T Q(n)[x(n)-x^o(n)] + \\ [u(n)-u^o(n)]^T R(n)[u(n)-u^o(n)]\Big) \qquad (3)\end{aligned}$$

and

$$\psi(x) = \frac{1}{2}[x-x^o(N)]^T Q(N)[x-x^o(N)] \qquad (4)$$

In (3) and (4), $u^o(n)$, $n = 0, \ldots\ N-1$, and $x^o(n)$, $n = 1, \ldots\ N$ are given control input and state target sequences. In standard optimal regulator control problem formulations, $u^o(n)$, $x^o(n)$ are usually taken to be zero with the exception perhaps of $x^o(N)$, the desired final value for the state. The formulation considered here addresses the more general optimal tracking control problem and is required for the linear quadratic step in the proposed algorithm presented in Section 3.1. Since $x(0)$ is given, we assume that $Q(0) = 0$. We also assume that the weighting matrices in (3) satisfy

$$R(n) = R(n)^T > 0, \quad n = 0,\ldots,N-1, \quad \text{and} \quad Q(n) = Q(n)^T \geq 0, \quad n = 1,\ldots,N,$$

to guarantee the strict convexity of the cost function with respect to the controls under linearized system dynamics.

2.1 First Order Optimality Conditions

We next briefly review the first order optimality conditions for the optimal control problem of (1) and (2), in a manner that brings out certain important interpretations of the adjoint dynamical equations encountered in a control theoretic approach and Lagrange Multipliers found in a pure optimization theory approach.

Let us consider the *cost-to-go*:

$$J(n) \equiv \sum_{k=n}^{N-1} L(x(k),u(k),k) + \psi(x(N)) \qquad (5)$$

with L and ψ as defined in (3) and (4) respectively. We remark that $J(n)$ is a function of $x(n)$, and $u(k)$, $k = n, \ldots, N-1$ and introduce the sensitivity of the cost to go with respect to the current state:

$$\lambda^T(n) = \frac{\partial J(n)}{\partial x(n)} \tag{6}$$

From

$$J(n) = L(x(n), u(n), n) + J(n+1), \tag{7}$$

and since $\frac{\partial J(n+1)}{\partial x(n)} = \frac{\partial J(n+1)}{\partial x(n+1)} \cdot \frac{\partial x(n+1)}{\partial x(n)} = \lambda^T(n+1) f_x(x(n), u(n))$, (3) results in the recursion:

$$\begin{aligned} \lambda^T(n) &= L_x(x(n), u(n), n) + \lambda^T(n+1) f_x(x(n), u(n)) \\ &= [x(n) - x^o(n)]^T Q(n) + \lambda^T(n+1) f_x(x(n), u(n)), \end{aligned} \tag{8}$$

where L_x and f_x denote the partials of L and f respectively with respect to the state variables. The previous recursion can be solved backward in time ($n = N-1, \ldots, 0$) given the control and state trajectories and it can be started with the final value:

$$\lambda^T(N) = \frac{\partial L(N)}{\partial x(N)} = [x(N) - x^o(N)]^T Q(N) \tag{9}$$

derived from (4). In a similar manner, we compute the sensitivity of $J(n)$ with respect to the current control $u(n)$:

$$\begin{aligned} \frac{\partial J(n)}{\partial u(n)} &= L_u(x(n), u(n), n) + \lambda^T(n+1) f_u(x(n), u(n)) \\ &= [u(n) - u^o(n)]^T R(n) + \lambda^T(n+1) f_u(x(n), u(n)), \end{aligned} \tag{10}$$

where in (10), L_u and f_u denote the partials of L and f respectively with respect to the control variables.

Next note that $\frac{\partial J}{\partial u(n)} = \frac{\partial J(n)}{\partial u(n)}$ since the first n terms in J do not depend on $u(n)$. We have then obtained the gradient of the cost with respect to the control variables, namely:

$$\nabla_U J = \left[\frac{\partial J(0)}{\partial u(0)} \; \frac{\partial J(1)}{\partial u(1)} \; \cdots \; \frac{\partial J(N-1)}{\partial u(N-1)} \right]. \tag{11}$$

Assuming u is unconstrained, the first order optimality conditions require that

$$\nabla_U J = 0. \tag{12}$$

We remark that by considering the Hamiltonian

$$H(x, u, \lambda, n) \equiv L(x, u, n) + \lambda^T f(x, u), \tag{13}$$

we have that $H_u(x(n), u(n), \lambda(n+1), n) \equiv \frac{\partial J}{\partial u(n)}$, i.e. we uncover the generally known but frequently overlooked fact that the partial of the Hamiltonian with respect to the control variables u is the gradient of the cost function with respect to u. We emphasize here that in our approach for solving the optimal control problem, we take the viewpoint of the control variables $u(n)$ being the independent variables of the problem since the dynamical equations express (recursively) the state variables in terms of the controls and thus can be eliminated from the cost function. Thus in taking the partials of J with respect to u, J is considered as a function $u(n)$, $n = 0, \ldots, N-1$ alone, assuming that $x(0)$ is given. With this perspective, the problem becomes one of unconstrained minimization, and having computed $\nabla_U J$, Steepest Descent, Quasi-Newton, and other first derivative methods can be brought to bear to solve it. However, due to the large-scale character of the problem, only methods that take advantage of the special structure of the problem become viable. The Linear Quadratic optimal control algorithm is such an approach in case of linear dynamics. We briefly review it next.

2.2 Linear Quadratic Tracking Problem

We next consider the case of linear dynamics in the optimal control problem of (1) and (2). In the following, we distinguish all variables corresponding to the linear optimal control problem from those of the nonlinear optimal control problem by using an over-bar. When the cost is quadratic as in (3) we have the well-known Linear Quadratic Tracking problem. The control theoretic approach to this problem is based on solving the first order necessary optimality conditions (also sufficient in this case) in an efficient manner by introducing the Riccati equation. We briefly elaborate on this derivation next, for completeness and also since most references assume that the target sequences $x^o(n)$ and $u^o(n)$ are zero. First, we summarize the first order necessary optimality conditions for this problem.

$$\begin{aligned} \bar{x}(n+1) &= A(n)\bar{x}(n) + B(n)\bar{u}(n) &\quad (14)\\ \bar{\lambda}^T(n) &= [\bar{x}(n) - \bar{x}^o(n)]^T Q(n) + \bar{\lambda}^T(n+1)A(n) &\quad (15)\\ \partial \bar{J}(n)/\partial \bar{u}(n) &= [\bar{u}(n) - \bar{u}^o(n)]^T R(n) + \bar{\lambda}^T(n+1)B(n) \ = \ 0 &\quad (16) \end{aligned}$$

Note that the system dynamical equations (14) run forward in time $n = 0, \ldots, N-1$ with initial conditions $\bar{x}(0) = \bar{x}_0$ (given), while the adjoint dynamical equations (15) run backward in time, $n = N-1, \ldots, 0$ with final conditions $\bar{\lambda}^T(N) = [\bar{x}(N) - \bar{x}^o(N)]^T Q(N)$. From (16), we obtain

$$\bar{u}(n) = \bar{u}^o(n) - R^{-1}(n)B^T(n)\bar{\lambda}(n+1) \tag{17}$$

and by substituting in (14) and (15), we obtain the classical two-point boundary system but with additional forcing terms due to the $\bar{x}^o(n)$ and $\bar{u}^o(n)$ sequences:

$$\begin{aligned} \bar{x}(n+1) &= A(n)\bar{x}(n) - B(n)R^{-1}(n)B^T(n)\bar{\lambda}(n+1) + B(n)\bar{u}^o(n) &\quad (18)\\ \bar{\lambda}^T(n) &= Q(n)\bar{x}(n) + A^T(n)\bar{\lambda}(n+1) - Q(n)\bar{x}^o(n) &\quad (19) \end{aligned}$$

The system of (18) and (19) can be solved by the *sweep method* [11], based on the postulated relation

$$\bar{\lambda}(n) = P(n)\bar{x}(n) + s(n) \tag{20}$$

where $P(n)$ and $s(n)$ are appropriate matrices that can be found as follows. For $n = N$, (20) holds with

$$P(N) = Q(N), \qquad s(N) = -Q(N)\bar{x}^o(N). \tag{21}$$

We now substitute (20) in (18) and after some algebra we obtain

$$\bar{x}(n+1) = M(n)A(n)\bar{x}(n) + v(n) \tag{22}$$

where we defined

$$\begin{aligned} M(n) &= \left[I + B(n)R^{-1}(n)B^T(n)P(n+1)\right]^{-1} &\quad (23)\\ v(n) &= M(n)B(n)[\bar{u}^o(n) - R^{-1}(n)B^T(n)s(n+1)] &\quad (24) \end{aligned}$$

By replacing $\bar{\lambda}(n)$ and $\bar{\lambda}(n+1)$ in (19) in terms of $\bar{x}(n)$ and $\bar{x}(n+1)$ from (20), we obtain

$$P(n)\bar{x}(n) + s(n) \;=\; Q(n)\bar{x}(n) + A^T(n)\left[P(n+1)\bar{x}(n+1) + s(n+1)\right] - Q(n)\bar{x}^o(n),$$

and by expressing $\bar{x}(n+1)$ from (22) and (24) above, we get

$$\begin{aligned} P(n)\bar{x}(n) + s(n) \;=\;& Q(n)\bar{x}(n) + A^T(n)P(n+1)M(n)A(n)\bar{x}(n) \\ & -A^T(n)P(n+1)M(n)B(n)R^{-1}(n)B^T(n)s(n+1) \\ & +A^T(n)P(n+1)M(n)B(n)\bar{u}^o(n) + A^T(n)s(n+1) - Q(n)\bar{x}^o(n) \end{aligned}$$

The above equation is satisfied by taking

$$\begin{aligned} P(n) &= Q(n) + A^T(n)P(n+1)M(n)A(n) && (25) \\ s(n) &= A^T(n)[I - P(n+1)M(n)B(n)R(n)^{-1}B(n)^T]s(n+1) \\ &\quad +A^T(n)P(n+1)M(n)B(n)\bar{u}^o(n) - Q(n)\bar{x}^o(n) && (26) \end{aligned}$$

Equation (25) is the well-known Riccati difference equation and together with the auxiliary equation (26), which is unnecessary if $\bar{x}^o(n)$ and $\bar{u}^o(n)$ are zero, are solved backward in time ($n = N-1,\ldots,1$), with final values given by (21) and together with (23) and (24). The resulting values $P(n)$ and $s(n)$ are stored and used to solve forward in time ($n = 0,\ldots,N-1$), (22) and (17) for the optimal control and state trajectories. These equations are summarized in Algorithm 1.

Algorithm 1 Discrete-Time Finite-Horizon Linear Quadratic Optimal Control Problem

Step 1: **Solve backward ($n = N-1,\ldots,0$) with $P_N \equiv Q_N$ and $s_N \equiv -Q_N\bar{x}_N^o$:**

$$\begin{aligned} M(n) &= \left[I + B(n)R^{-1}(n)B^T(n)P(n+1)\right]^{-1} \\ P(n) &= Q(n) + A^T(n)P(n+1)M(n)A(n) \\ s(n) &= A^T(n)\left[I - P(n+1)M(n)B(n)R^{-1}(n)B^T(n)\right]s(n+1) \\ &\quad +A^T(n)P(n+1)M(n)B(n)\bar{u}^o(n) - Q(n)\bar{x}^o(n) \end{aligned}$$

Step 2: **Solve forward ($n = 0,\ldots,N-1$) with $\bar{x}(0) = \bar{x}_0$:**

$$\begin{aligned} v(n) &= M(n)B(n)[\bar{u}^o(n) - R^{-1}(n)B^T(n)s(n+1)] \\ \bar{x}(n+1) &= M(n)A(n)\bar{x}(n) + v(n) \\ \bar{\lambda}(n+1) &= P(n+1)\bar{x}(n+1) + s(n+1) \\ \bar{u}(n) &= \bar{u}^o(n) - R^{-1}(n)B^T(n)\bar{\lambda}(n+1) \end{aligned}$$

3 Main Results

3.1 Formulation of the SLQ Algorithm

In the proposed SLQ algorithm, the control at stage $k+1$ is found by performing a one-dimensional search from the control at stage k and along a search direction that is found by solving a Linear Quadratic (LQ) optimal control problem. Specifically, let $U_k = [u^T(0)\; u^T(1)\ldots u^T(N-1)]^T$ be the optimal solution candidate at step k, and $X_k = [x^T(1)\ x^T(2) \ldots x^T(N)]^T$ the corresponding state trajectory obtained by solving the dynamical equations (2) using U_k and with the initial conditions $x(0)$. We next linearize the state equations (2) about the nominal trajectory of U_k and X_k. The linearized equations are

$$\bar{x}(n+1) = f_x(x(n), u(n))\ \bar{x}(n) + f_u(x(n), u(n))\ \bar{u}(n) \tag{27}$$

with initial conditions $\bar{x}(0) = 0$. We then minimize the cost index (1) with respect to $\bar{u}(n)$. The solution of this LQ problem gives $\bar{U}_k = [\bar{u}^T(0)\ \bar{u}^T(1) \ldots \bar{u}^T(N-1)]^T$, the proposed search direction. Thus, the control variables at stage $k+1$ of the algorithm are obtained from

$$U_{k+1} = U_k + \alpha_k \cdot \bar{U}_k \tag{28}$$

where $\alpha_k \in \mathbb{R}^+$ is appropriate stepsize, the selection of which is discussed later in the paper. Note again our perspective of considering the optimal control problem as an unconstrained finite-dimensional optimization problem in U. We emphasize that $\bar{U}_k$ as computed above is not the steepest descent direction. It is the solution to a linear quadratic tracking problem for a nonlinear system that has been linearized about U_k. Note that the objective function is not linearized for this solution.

3.2 Properties of the SLQ Algorithm

In this section, we prove two important properties of the proposed algorithm. First, we show that search direction $\bar{U}_k$ in (28) is a descent direction.

Theorem 1 *Consider the discrete-time nonlinear optimal control problem of (1) and (2), and assume a quadratic cost function as in (3) and (4) with $R(n) = R^T(n) > 0$, $Q(n) = Q^T(n) \geq 0$, $n = 0, 1, \ldots, N-1$, and $Q(0) = 0$, $Q(N) = Q^T(N) \geq 0$. Also consider a control sequence $U \equiv [u^T(0) \ldots u^T(N-1)]^T$ and the corresponding state trajectory $X \equiv [x^T(1) \ldots x^T(N)]^T$. Next, linearize system (2) about U and X and solve the following linear quadratic problem:*

$$\begin{aligned} \underset{\bar{u}(n), \bar{x}(n)}{\text{Minimize}}\ \bar{J} &= \frac{1}{2}[\bar{x}(N) - \bar{x}^o(N)]^T Q(N) [\bar{x}(N) - \bar{x}^o(N)] \\ &+ \frac{1}{2} \sum_{n=0}^{N-1} \left\{ [\bar{x}(n) - \bar{x}^o(n)]^T Q(n) [\bar{x}(n) - \bar{x}^o(n)] \right. \\ &\qquad \left. + [\bar{u}(n) - \bar{u}^o(n)]^T R(n) [\bar{u}(n) - \bar{u}^o(n)] \right\} \end{aligned} \tag{29}$$

$$\text{subj. to} \qquad \bar{x}(n+1) = f_x(x(n), u(n))\bar{x}(n) + f_u(x(n), u(n))\bar{u}(n); \qquad \bar{x}(0) = 0, \tag{30}$$

where $\bar{x}^o(n) \equiv x^o(n) - x(n)$, $\bar{u}^o(n) \equiv u^o(n) - u(n)$. Then if $\bar{U} \equiv [\bar{u}^T(0) \ldots \bar{u}^T(N-1)]^T$ is not zero, it is a descent direction for the cost function (1), i.e. $J(U + \alpha \cdot \bar{U}) < J(U)$ for some $\alpha > 0$.

Proof: We establish that $\bar{U}$ is a descent direction by showing that:

$$\nabla_U J \cdot \bar{U} = \sum_{n=0}^{N-1} \frac{\partial J(n)}{\partial u(n)} \bar{u}(n) < 0, \tag{31}$$

since $\nabla_U J$ in (11) is the gradient of the cost function with respect to the control variables. Now, the components of $\nabla_U J$ are expressed in (10) in terms of the adjoint variables $\lambda(n)$ that satisfy recursion (8) with final values given by (9). On the other hand, $\bar{x}(n)$ and $\bar{u}(n)$ together with adjoint variables $\bar{\lambda}(n)$ satisfy the first order optimality conditions for the linear quadratic problem given in (14), (15) and (16), where $A(n) = f_x(x(n), u(n))$ and $B(n) = f_u(x(n), u(n))$. Let us define

$$\tilde{\lambda}(n) = \bar{\lambda}(n) - \lambda(n) \tag{32}$$

and note from (8) and (15) that

$$\tilde{\lambda}^T(n) = \bar{x}^T(n)Q(n) + \tilde{\lambda}^T(n+1)A(n); \quad \tilde{\lambda}(N) = Q(N)\bar{x}(N). \tag{33}$$

Next through the indicated algebra, we can establish the following relation:

$$\begin{aligned}
\frac{\partial J(n)}{\partial u(n)} \cdot \bar{u}(n) &= \left([u(n) - u^o(n)]^T R(n) + \lambda^T(n+1)B(n)\right)\bar{u}(n) \\
&\quad \text{(using (10))} \\
&= -\tilde{\lambda}^T(n+1)B(n)\bar{u}(n) - \bar{u}^T(n)R(n)\bar{u}(n) \\
&\quad \text{(using (16))} \\
&= -\tilde{\lambda}^T(n+1)\bar{x}(n+1) + \tilde{\lambda}^T(n+1)A(n)\bar{x}(n) - \bar{u}^T(n)R(n)\bar{u}(n) \\
&\quad \text{(using (30))} \\
&= -\tilde{\lambda}^T(n+1)\bar{x}(n+1) + \tilde{\lambda}^T(n)\bar{x}(n) - \bar{x}^T(n)Q(n)\bar{x}(n) - \bar{u}^T(n)R(n)\bar{u}(n). \\
&\quad \text{(using (33))}
\end{aligned}$$

Finally, summing up the above equation from $n = 0$ to $n = N - 1$, and noting that $\bar{x}(0) = 0$ and from (21) that $\bar{\lambda}(N) = Q(N)\bar{x}(N)$, gives:

$$\begin{aligned}
\nabla_U J \cdot \bar{U} &= \sum_{n=0}^{N-1} \frac{\partial J(n)}{\partial u(n)} \cdot \bar{u}(n) \\
&= -\sum_{n=0}^{N-1} [\bar{x}^T(n)Q(n)\bar{x}(n) + \bar{u}^T(n)R(n)\bar{u}(n)] - \bar{x}^T(N)Q(N)\bar{x}(N) \; < \; 0 \qquad (34)
\end{aligned}$$

and the proof of the theorem is complete. ■

We remark that the search direction $\bar{U}$ can be found by Algorithm 1 with $A(n) = f_x(x(n), u(n))$ and $B(n) = f_u(x(n), u(n))$.

The next result shows that the proposed SLQ algorithm does in fact converge to a control that is a stationary point of the cost function (1) subject to the system constraints (2). In the following, we denote by $J[U]$ the cost associated with the control $U = [u^T(0) \ldots u^T(N-1)]^T$ (and the given initial conditions $x(0)$.)

Theorem 2 *Starting with an arbitrary control sequence U_0, compute recursively new controls from (28) where the direction $\bar{U}_k$ is obtained as in Theorem 1 by solving the LQ problem of (29) and the linearized system (30) about the current solution candidate U_k and corresponding state trajectory X_k; also α_k is obtained by minimizing $J[U_k + \alpha\bar{U}_k]$ over $\alpha > 0$. Then every limit point of U_k gives a control that satisfies the first order optimality conditions for the cost function (1) subject to the system equations (2).*

Proof: Let us define $D_k \equiv \nabla_U J[U_k] \cdot \bar{U}_k < 0$, the derivative of $J[U_k + \alpha\bar{U}_k]$ with respect to α at $\alpha = 0$. Through rather standard arguments (see Appendix A), it can be shown that D_k has a subsequence that converges to zero as $k \to \infty$.

From (34), we obtain:

$$D_k = \nabla_U J[U_k] \cdot \bar{U}_k \leq -\rho \|\bar{U}_k\|^2 \leq 0, \tag{35}$$

where $\rho > 0$ is a uniform lower bound on the minimum eigenvalue of $R(n)$, $n = 0, \ldots, N-1$. Then (35) implies that there is a corresponding subsequence of $\bar{U}_k$ that converges to zero. For notational convenience, we identify this subsequence of $\bar{U}_k$ (and corresponding subsequences of other sequences) with the whole sequence.

But $\bar{U}_k \to 0$ implies from (14) that $\bar{X}_k \to 0$ (note that $\bar{x}(0) = 0$ and that $A(n)$ and $B(n)$ can be assumed to be uniformly bounded since $U_{k+1} - U_k = \alpha_k \bar{U}_k \to 0$.) Consequently, (33) implies that $\bar{\lambda}(n) \to \lambda(n)$ for $n = 0, \ldots, N-1$, and that the right-hand-side of (10) converges to the right-hand-side of (16) which is zero by the optimality of $\bar{U}_k$ for the LQ problem. This shows that the first order optimality conditions (10) for the nonlinear control problem are asymptotically satisfied, i.e. $\nabla_U J[U_k] \to 0$ as $k \to \infty$. We remark that the last conclusion follows for a subsequence of U_k, but since the cost $J[U_k]$ is monotonically decreasing and clearly converges to a locally minimum value, any limit point of U_k must be a stationary point and the proof is complete.■

We remark, that the exact line search in Theorem 2 can be replaced with an inaccurate line search that satisfies some standard conditions such as in the Armijo, Goldstein, or Wolfe stepsize selection rules [14]. Similar arguments as in Appendix A, can be used to show that $D_k \to 0$ still follows and the proof is completed as above.

The proposed SLQ procedure is summarized as Algorithm 2.

3.3 Connections with Gauss-Newton Method

The Gauss-Newton method [15, 12] applies to least-squares optimization problems of the form:

$$\underset{U}{\text{Minimize}} \;\; J_{GN}[U] \equiv \sum_{l=1}^{M} \phi_l^2(U), \tag{36}$$

where $\phi_l(U)$, $l = 1, \ldots\ M$ are real-valued functions of the independent variables U. Let $G_F(U)$ denote the Jacobian of $F(U) \equiv [\phi_1 \ldots\ \phi_M]^T$ with respect to U. Then, we can readily compute the gradient and Hessian of the objective function in (36) as (see, for example [12])

$$\nabla_U J_{GN}[U] = F^T(U) \cdot G_F(U) \tag{37}$$

and

$$\nabla_U^2 J_{GN}[U] = G_F^T(U) \cdot G_F(U) + \sum_{l=1}^{M} \phi_l(U) \cdot \nabla_U^2 \phi_l(U). \tag{38}$$

respectively. While Newton's method uses the update

$$U_{k+1} = U_k - \left(\nabla_U^2 J_{GN}[U_k]\right)^{-1} \nabla_U J_{GN}[U_k], \tag{39}$$

the Gauss-Newton method updates the solution estimate from

$$U_{k+1} = U_k - \alpha_k \left(G_F^T(U_k) G_F(U_k)\right)^{-1} \nabla_U J_{GN}[U_k], \tag{40}$$

where a_k is an appropriate stepsize. It can be observed from (39) and (40) that the Gauss-Newton direction is an approximation to the Newton direction obtained by omitting the second order derivative terms in the Hessian (38). Newton's method does not involve a line search and when it

Algorithm 2 Sequential Linear Quadratic Procedure

Step 0: Set $k = 0$, and let $U_k = [u^T(0) \dots u^T(N-1)]^T$ be an arbitrary control sequence.

Step 1: Compute the state trajectory $X_k = [x^T(1) \dots x^T(N)]^T$ corresponding to U_k and initial conditions $x(0) = x_0$ by recursively solving the nonlinear dynamical equations

$$x(n+1) = f(x(n), u(n)).$$

Step 2: For $n = 0, 1 \dots N-1$, let

$$\begin{aligned} A(n) &= f_x(x(n), u(n)) \qquad & B(n) &= f_u(x(n), u(n)), \\ \bar{x}^o(n) &= x^o(n) - x(n) \qquad & \bar{u}^o(n) &= u^o(n) - u(n) \end{aligned}$$

where $x^o(n)$ and $u^o(n)$ are given target control and state trajectories, and solve the following LQ problem, using Algorithm 1.

$$\begin{aligned} \underset{\bar{u}(n), \bar{x}(n)}{\text{Minimize}} \; \bar{J} &= \frac{1}{2}[\bar{x}(N) - \bar{x}^o(N)]^T Q(N) [\bar{x}(N) - \bar{x}^o(N)] \\ &+ \frac{1}{2} \sum_{n=0}^{N-1} \big\{ [\bar{x}(n) - \bar{x}^o(n)]^T Q(n) [\bar{x}(n) - \bar{x}^o(n)] \\ &\qquad + [\bar{u}(n) - \bar{u}^o(n)]^T R(n) [\bar{u}(n) - \bar{u}^o(n)] \big\} \end{aligned}$$

$$\text{subj. to} \qquad \bar{x}(n+1) = A(n)\bar{x}(n) + B(n)\bar{u}(n); \qquad \bar{x}(0) = 0,$$

Let $\bar{U}_k = [\bar{u}^T(0) \dots \bar{u}^T(N-1)]^T$. If $\|\bar{U}_k\| < tol \cdot \|U_k\|$, **EXIT** with $U^* = U_k$ being a control that satisfies the first order optimality conditions. Otherwise, continue with **Step 3**.

Step 3: Starting with $\alpha_k = 1$ find $\alpha_k^* \in (0\ 1]$ such that the control $U = U_k + \alpha_k^* \bar{U}_k$ and corresponding state trajectory X obtained by simulating the system dynamical equations as in Step 1, minimizes the cost function

$$\begin{aligned} J &= \frac{1}{2}[x(N) - x^o(N)]^T Q(N) [x(N) - x^o(N)] + \\ &\quad \frac{1}{2} \sum_{n=0}^{N-1} \big\{ [x(n) - x^o(n)]^T Q(n) [x(n) - x^o(n)] + \\ &\qquad [u(n) - u^o(n)]^T R(n) [u(n) - u^o(n)] \big\} . \end{aligned}$$

Alternatively the Armijo or some other stepsize selection rule may be used. Set $k \to k+1$ and continue with **Step 1**.

converges, under certain favorable conditions such as continuity of 3^{rd}-order derivatives or Lipschitz continuity of 2^{nd}-order derivatives, it does so quadratically. On the other hand, the Gauss-Newton direction is a descent direction, because the approximation of the Hessian used is positive semidefinite by construction; therefore, it is recommended that the Gauss–Newton method should be used along with a line search (or a trust region approach not discussed here) to guarantee global convergence. The Gauss-Newton method can ultimately achieve a quadratic rate of convergence, in spite of the fact that second order derivative information is not used. This is the case in problems with a low degree of nonlinearity or in small residual problems, i.e. with $||\nabla_U^2 \phi_l(U^*)\|$ or $|\phi_l(U^*)|$, $l = 1, \dots M$ being small at the optimal solution U^* respectively, in which case the omitted terms from the Hessian (38) will be also small. However, empirical evidence shows that the method usually performs well in general and it can be shown that the rate of convergence is no worse than linear [15].

We next cast the optimal control problem of (1) and (2) with quadratic cost given by (3) and (4) in the least squares minimization form of (36). Consider the eigenvalue decompositions

$$Q(n) = \sum_{i=1}^{m} q_i(n) v_i(n) v_i^T(n) \tag{41}$$

and

$$R(n) = \sum_{j=1}^{p} r_j(n) w_j(n) w_j^T(n) \tag{42}$$

where $q_i(n) \geq 0,\ i = 1, \dots, m$ and $r_j(n) > 0,\ j = 1, \dots, p$ are the eigenvalues of $Q(n)$ and $R(n)$ respectively and $v_i(n) \in \mathbf{R^m}$, $w_j(n) \in \mathbf{R^p}$ are corresponding sets of orthonormal eigenvectors. We remind the reader that m and p denote the dimensions of the state and control vectors respectively. We also bring in functions $g_n(U),\ n = 1, \dots, N$ that express the states $x(n)$ in terms of the input sequence $U = [u^T(0)\ u^T(1) \dots\ u^T(N-1)]^T$, that is

$$x(n) \equiv g_n(U). \tag{43}$$

Clearly, $g_1(U) \equiv f(x_0, u(0))$, $g_2(U) \equiv f(g_1(U), u(1))$, etc. can be evaluated recursively by simulating the dynamical equations (2). Now, it can be readily seen that the cost function of the optimal control problem takes the form of (36) by defining the functions

$$\phi_{ni}^Q(U) \equiv q_i^{1/2}(n+1) v_i^T(n+1)[g_{n+1}(U) - x^o(n+1)],\ n = 0, \dots, N-1,\ i = 1, \dots, m \tag{44}$$

and

$$\phi_{nj}^R(U) \equiv r_j^{1/2}(n) w_j^T(n)[u(n) - u^o(n)],\ n = 0, \dots, N-1,\ j = 1, \dots, p. \tag{45}$$

Specifically, we have that the cost in (1) and (3) can be expressed as

$$J \equiv \sum_{n=0}^{N-1} \left(\sum_{i=1}^{m} [\phi_{ni}^Q(U)]^2 + \sum_{j=1}^{p} [\phi_{nj}^R(U)]^2 \right). \tag{46}$$

Next, we apply (37) and (38) to compute $\nabla_U J$ and $\nabla_U^2 J$. Let us first define

$$Q^{1/2}(n) \equiv \begin{bmatrix} q_1^{1/2}(n) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & q_m^{1/2}(n) \end{bmatrix} \begin{bmatrix} v_1^T(n) \\ \vdots \\ v_m^T(n) \end{bmatrix}, \tag{47}$$

and

$$R^{1/2}(n) \equiv \begin{bmatrix} r_1^{1/2}(n) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & r_p^{1/2}(n) \end{bmatrix} \begin{bmatrix} w_1^T(n) \\ \vdots \\ w_p^T(n) \end{bmatrix}. \tag{48}$$

Also let $E(n) \equiv [0_{p\times(n-1)p}\ I_p\ 0_{p\times(N-n)p}]^T$ where $0_{i\times j}$ is the zero $i \times j$ matrix and I_p is the identity matrix of size p. Then straightforward algebra gives

$$F(U) = \left[\begin{array}{c} Q^{1/2}(1)[g_1(U) - x^o(1)] \\ \vdots \\ Q^{1/2}(N)[g_N(U) - x^o(N)] \\ \hline R^{1/2}(0)[u(0) - u^o(0)] \\ \vdots \\ R^{1/2}(N-1)[u(N-1) - u^o(N-1)] \end{array} \right] \tag{49}$$

and

$$G_F(U) = \left[\begin{array}{c} Q^{1/2}(1)\nabla_U g_1 \\ \vdots \\ Q^{1/2}(N)\nabla_U g_N \\ \hline R^{1/2}(0)E^T(1) \\ \vdots \\ R^{1/2}(N-1)E^T(N) \end{array} \right]. \tag{50}$$

Furthermore, substituting (50) and (49) in (37) results in the following formula:

$$\nabla_U J = \sum_{n=0}^{N-1} \left\{ [g_{n+1}(U) - x^o(n+1)]^T Q(n+1)[\nabla_U g_{n+1}]^T \ + [u(n) - u^o(n)]^T R(n) E^T(n+1) \right\}. \tag{51}$$

Similarly, from (50) and (38), we obtain

$$\nabla_U^2 J = \sum_{n=1}^{N} [\nabla_U g_n]^T Q(n) [\nabla_U g_n] \ + \ R_D \ + \ \sum_{n=1}^{N}\sum_{i=1}^{m} \phi_{ni}^Q \cdot \nabla_U^2 \phi_{ni}^Q, \tag{52}$$

where we defined $R_D \equiv diag\{R(0), \dots, R(N-1)\}$. In (38), note that $\nabla_U^2 \phi_{nj}^R = 0$ since ϕ_{nj}^R is linear in U.

Now, we are in position to prove the following result

Theorem 3 *The proposed SLQ algorithm for solving the nonlinear optimal control problem of (1) and (2) with quadratic performance as in (3) and (4) is equivalent to the Gauss-Newton method (40) applied to the objective function in (46).*

Proof: We notice that the LQ problem formulated in Theorem 1 and the solution of which provides the SLQ search direction $\bar{U}$ can be also expressed as in (46). We can then obtain the gradient and Hessian for this problem from (51) and (52) respectively when the appropriate variables are used. Specifically, using the bar notation to distinguish the variables that are different in the LQ and original nonlinear optimal control problem, we have

$$\nabla_{\bar{U}}\bar{J}[\bar{U}] = \sum_{n=0}^{N-1}\left\{[\bar{g}_{n+1}(\bar{U}) - \bar{x}^o(n+1)]^T Q(n+1)[\nabla_{\bar{U}}\bar{g}_{n+1}]^T \ + [\bar{u}(n) - \bar{u}^o(n)]^T R(n) E^T(n+1)\right\} \tag{53}$$

and

$$\nabla^2_{\bar{U}}\bar{J}[\bar{U}] = \sum_{n=1}^{N}[\nabla_{\bar{U}}\bar{g}_n]^T Q(n)[\nabla_{\bar{U}}\bar{g}_n] \ + \ R_D \ + \ \sum_{n=1}^{N}\sum_{i=1}^{m}\bar{\phi}^Q_{ni}\cdot\nabla^2_{\bar{U}}\bar{\phi}^Q_{ni}. \tag{54}$$

Next, notice that $\bar{x}(n) = \bar{g}_n(\bar{U})$ and, therefore (see (44)), $\bar{\phi}^Q_{ni}(\bar{U})$ are linear in $\bar{U}$. This implies that the last term in (54) vanishes; we also have that the LQ problem in the form of (46) becomes a QP (quadratic program). Thus the LQ solution is expressed as

$$\bar{U} = -\left(\nabla^2_{\bar{U}}\bar{J}[0]\right)^{-1}\nabla_{\bar{U}}\bar{J}[0], \tag{55}$$

which represents a Newton update step from the zero starting point. Now observe that $\bar{x}(n) = \bar{g}_n(0) \equiv 0$, the result of simulating the linearized dynamical equations (27) with zero input and initial conditions. Also, we have previously defined (see statement of Theorem 1)

$$\bar{x}^o(n) \equiv x^o(n) - x(n) = x^o(n) - g_n(U) \ \text{ and } \ \bar{u}^o(n) \equiv u^o(n) - u(n). \tag{56}$$

Finally, we claim that

$$\nabla_{\bar{U}}\bar{g}_n(0) = \nabla_U g_n(U), \tag{57}$$

where U is the solution estimate for the nonlinear control problem at the k^{th} stage. Indeed, from (2), it follows that

$$\nabla_U g_{n+1}(U) = f_x(x(n), u(n))\nabla_U g_n(U) + f_u(x(n), u(n))E^T(n+1), \quad \nabla_U g_0 \equiv 0. \tag{58}$$

Also, from (27), we have

$$\nabla_{\bar{U}}\bar{g}_{n+1}(0) = f_x(x(n), u(n))\nabla_{\bar{U}}\bar{g}_n(0) + f_u(x(n), u(n))E^T(n+1), \quad \nabla_{\bar{U}}\bar{g}_0 \equiv 0. \tag{59}$$

Clearly from (58) and (59), we have the desired result.

We now apply the above observations and in particular (56 and (57) in the expressions (53) and (54) for the gradient and Hessian of the LQ problem, and further by substituting in (55), we obtain for the SLQ direction at the k^{th} stage:

$$\begin{aligned}\bar{U} \ &= \ -\left[\sum_{n=1}^{N}[\nabla_U g_n(U)]^T Q(n)[\nabla_U g_n(U)] + R_D\right]^{-1} \\ &\qquad \left(\sum_{n=1}^{N}[g_n(U) - x^o(n)]^T Q(n)[\nabla_U g_n(U)]^T + \sum_{n=0}^{N-1}[u(n) - u^o(n)]^T R(n) E^T(n+1)\right)\end{aligned} \tag{60}$$

Then comparing (60) with (40) where (51) and (52) (with the 2nd derivatives in (52) dropped) are used, shows that the SLQ and the Gauss-Newton search direction are the same. ■

Note that (60) is not recommended for computation of the search direction. The previous analysis serves the purpose of shedding light to the properties of the SLQ algorithm, and in particular for explaining its excellent convergence properties. It also shows that the SLQ algorithm is in fact a Sequential Quadratic Programming (SQP) approach for the nonlinear optimal control problem with quadratic cost. Indeed, at each iteration an LQ problem is solved that has been shown to be equivalent with a QP. Some of the most general approaches for numerically solving optimal control problems start with discretizing the system equations by the method of collocation and reducing it to a standard (albeit large scale) finite dimensional optimization program. The method of choice for solving the latter program is SQP and much of the current research has to do with how to take advantage of the sparsity in the constraint equations induced by the state equations [10]. Our algorithm exactly accomplishes this by solving the ensuing QP as an LQ problem using the results summarized in Algorithm 1 and which take full advantage of the problem structure.

The results of this section also suggest possible optimal control problem structures for which the SLQ algorithm should enjoy fast (Newton-like) local convergence. Namely, problems with a small degree of nonlinearity or small residual problems. The latter condition is obtained, for example in set-point regulator problems, when $Q(n)$ in (3) are zero or relatively small with respect to $R(n)$ and $Q(N)$ in (4) and the $x(N) \approx x^o(N)$. However, in practice we observed that larger values of $Q(n)$ give overall better performance, most likely because the states remain smaller and larger steps can be taken initially. Another favorable situation is when the control penalty $R(n)$ is high and overpowers the dropped 2^{nd}-order system derivatives in (52). However, we emphasize that the previous conditions pertain only to the local convergence of the algorithm and that we have observed excellent overall behavior for a variety of problems in which such conditions may be satisfied or not.

4 Numerical Examples

4.1 A Vehicle Moving Through Nonlinear Viscous Drag

We consider the following continuous-time minimum fuel problem for a particle moving under nonlinear viscous friction.

$$\begin{aligned} \underset{u(t),\ x(t)}{\text{Minimize}}\ J &= q_{fin}(x(T)-10)^2 + \int_0^T \left[q\ (x(t)^2 + \dot{x}(t)^2) + r\ u^2(t)\right] dt && (61) \\ \text{subject to} \quad & m\ddot{x} = u(t) - \beta \text{sign}(\dot{x})\dot{x}^2;\ x(0) = 0 && (62) \end{aligned}$$

with u, x scalar variables. More specifically, we would like to drive the state $x(t)$ (position) from 0 to 10 within $T = 1sec$. We apply our algorithm to this problem, by first discretizing the system. We select a sampling period of $T_s = 1/N$ with $N = 500$ and we take $m = 25.9gr$. Using a simple Euler integration approach we obtain the discrete-time system:

$$\begin{aligned} x(n+1) &= v(n)T_s + x(n) && (63) \\ v(n+1) &= -\frac{1}{m}\left[\beta T_s v(n)^2 sign(v(n)) - T_s u(n)\right] + v(n) && (64) \end{aligned}$$

where $x(n)$, $v(n)$ approximate the particle position and velocity respectively at time nT_s. The cost function (61) is also discretized as follows:

$$J = q_{fin}(x(N)-10)^2 + T_s \sum_{n=0}^{N-1} \left[q\ (x(n)^2 + v(n)^2) + r\ u^2(n)\right]. \tag{65}$$

This problem has a solution that it is easy to understand but it can provide a serious challenge to any optimization algorithm by taking the drag coefficient parameter β large enough, inducing

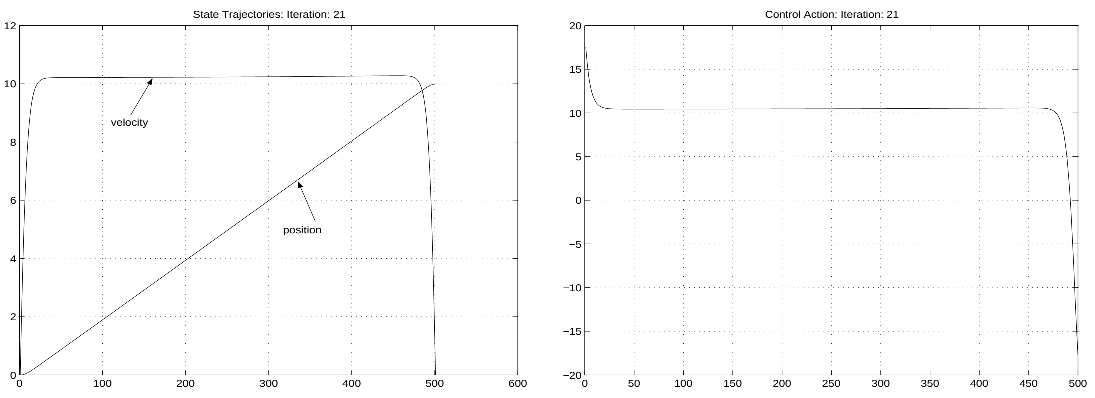

Figure 1: Optimal state, control for a particle moving under viscous drag.

sharper changes in the control at the beginning and end of the motion. While physical reasoning and boundary layers can be used to circumvent the numerical difficulties, it is desirable for an algorithm to be able to handle such issues automatically. A representative optimal solution is obtained for values of the weights $r = 10$, $q = 1$, $q_{fin} = 10^6$, and for $\beta = 0.1$ is depicted in Fig. 1. The proposed algorithm took 21 iterations to achieve an accuracy of 10^{-6} (norm of gradient at optimal solution.) In Table 1, we compare the number of iterations required by our SLQ algorithm and a modified Newton's Method (using a standard LU factorization approach when necessary to obtain a descent direction [12]) to solve this problem with the previous weight values and the indicated values of β. It is perhaps surprising that Newton's method requires significantly more iterations that are also more expensive to implement. We observed that the Newton direction is overall inferior to the SLQ (Gauss-Newton) direction, in the sense that the line search procedure (implemented by a quadratic fit algorithm) results in smaller steps with the Newton direction than the SLQ direction in all but the very few last steps of each algorithm. The relative cost reduction, i.e. cost at iteration k minus optimal cost normalized by the optimal cost is depicted for both methods and for $\beta = 0.1$ in Fig. 2, where the cost for the Steepest Descent method with exact line search is also shown.

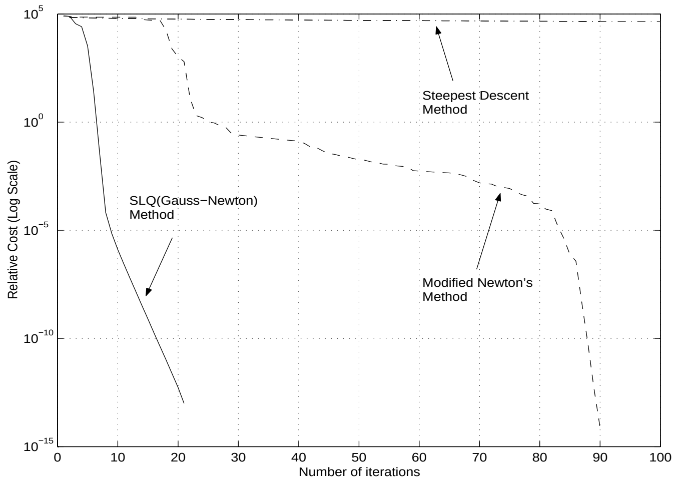

Figure 2: Relative Cost for SLQ, Modified Newton and Steepest Descent Methods for Example 1.

β	SQL Algorithm	Modified Newton's Method
0.01	9	72
0.05	20	123
0.1	21	90
0.5	35	did not converge within 300 iterations
1	did not converge within 300 iterations	did not converge within 300 iterations

Table 1: Iterations required by SLQ Algorithm and a Modified Newton's Method on Example 1.

4.2 A Gas Actuated Hopping Machine

An interesting optimal control problem is that of creating motions for an autonomous hopping machine. A simple model for a gas actuated one-dimensional hopping system is shown on the left-hand side of Figure 3. This system is driven by a pneumatic actuator, with the location of the piston relative to the mass under no external loading defined as y_p. After contact occurs with the ground with $y \leq y_p$, the upward force on the mass from actuator can be approximated by a linear spring with $F = k(y_p - y)$, where k is the spring constant. The position y_p can be viewed as the unstretched spring length and it can be easily changed by pumping air into or out of either side of the cylinder. The equations of motion for the mass are $m\ddot{y} = F(y, y_p) - mg$, where mg is the force due to gravity, and $F(y, y_p) = \left\{ \begin{array}{ll} 0 & y > y_p \\ k(y_p - y) & \text{otherwise.} \end{array} \right.$ Note that in this case $F(y, y_p)$ is not differentiable at $y = y_p$, and the proposed algorithm can not be used on this problem. However, the discontinuity in the derivative can easily be smoothed. For instance, let the spring compression be $e = y_p - y$ and choose an $\alpha > 0$, then

$$F(e) = \left\{ \begin{array}{ll} 0 & 0 > e \\ \frac{k}{2\alpha}e^2 & 0 \leq e < \alpha \\ ke - \frac{k\alpha}{2} & \text{otherwise} \end{array} \right.$$

is C^1. The final equation of motion for this system relates the air flow into the cylinder, which is the control $u(t)$, to the equilibrium position y_p of the piston. Assume for the following that the equation $\dot{y}_p = u$ approximates this relationship.

When the hopping machine begins its operation, we are interested in starting from rest, and reaching a desired hop height y_N^o at time t_f. If we minimize

$$J(u) = \frac{1}{2}q_{fin}(y(N) - y_N^o)^2 + \dot{y}(N)^2 + \frac{t_f}{2N}\sum_{n=0}^{N-1}\left[q\ y_p(n)^2 + r\ u(n)^2\right], \tag{66}$$

the terms outside the summation reflect the desire to reach the height at time t_f with zero velocity, and the terms inside the summation reflect the desire to minimize the gas used to achieve this.

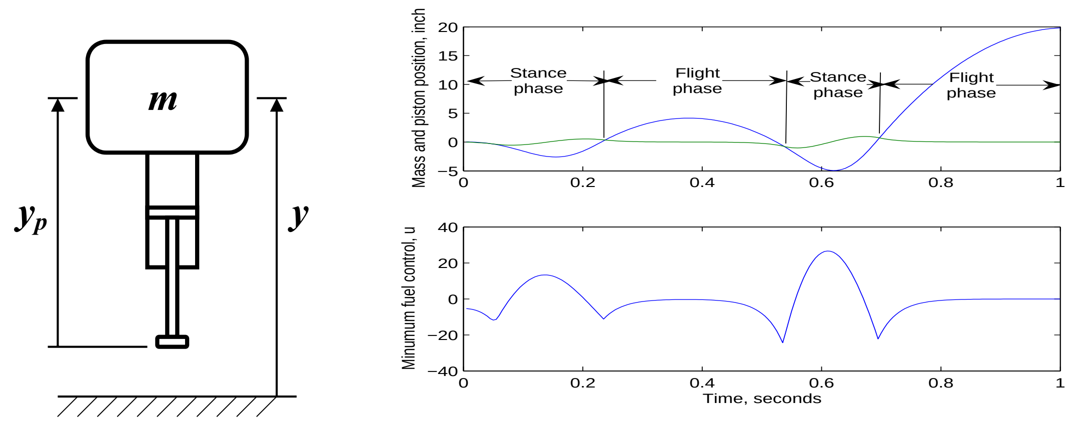

Figure 3: Maximum height hopping motion and minimum fuel control.

The weighting on y_p is used to keep the piston motion within its bounds. We conducted numerical experiments on this system with the following parameters: $k/m = 500$, $g = 386.4$, $\alpha = 0.1$. We assumed that all states were initially zero, and that the initial control sequence was zero. The cost function parameters were selected as: $y_N^o = 20$, $t_f = 1$, $q_{fin} = q = 1000$, and $r = 1.0$. As in the last example, a simple Euler approximation was used to discretize the equations, with $N = 100$.

Note that the algorithm produced an alternating sequence of stance phases and flight phases for the hopping system and it naturally identified the times to switch between these phases. If one were to use collocation methods to solve this problem with explicit consideration of the different dynamics in the different phases, one would have to guess the number of switches between phases and would need to treat the times at which the switch occurs as variables in the optimization. Consistent with the discussion of the Gauss-Newton convergence rate, our algorithm converges much faster when the weighting on the control r is increased; also the number of iterations required for convergence in this problem increases for larger y_N^o, ranging from 3 for $y_N^o = 1$, to 166 for $y_N^o = 50$. In addition, the algorithm failed to converge for $\alpha < 1 \times 10^{-5}$, which demonstrates the need for the dynamics to be continuously differentiable.

5 Conclusion

We developed an algorithm for solving nonlinear optimal control problems with quadratic performance measures and unconstrained controls. Each subproblem in the course of the algorithm is a linear quadratic optimal control problem that can be efficiently solved by Riccati difference equations. We show that each search direction generated in the linear quadratic subproblem is a descent direction, and that the algorithm is convergent. Computational experience has demonstrated that the algorithm converges quickly to the optimal solution. Finally, we show that the proposed SLQ algorithm can be interpreted as the Gauss-Newton method applied to a least-squares reformulation of the optimal control problem and thus explain its excellent rate of convergence observed in simulations. The SLQ algorithm is proposed as a powerful alternative to Newton methods in situations that second order derivative information on the system dynamics is not available or expensive to obtain and a near real-time solver is required.

Appendix A

Proof of Theorem 2 We show that under the exact line search assumption $D_k = \nabla J[U_k] \cdot \bar{U}_k$ has a subsequence that converges to zero as $k \to \infty$. Notice that for any fixed ϵ such that $0 < \epsilon < 1$ and α sufficiently small, it holds

$$J[U_k + \alpha \bar{U}_k] \leq J[U_k] + \epsilon \alpha D_k. \tag{67}$$

Let $\bar{\alpha}_k$ be such that (67) is satisfied with equality (such $\bar{\alpha}_k < \infty$ exists since otherwise $J[U_k + \alpha \bar{U}_k] \to -\infty$ as $\alpha \to \infty$, contradicting the fact that J is bounded below by zero.) From the mean value theorem, there exists β_k with $0 \leq \beta_k \leq \bar{\alpha}_k$ such that

$$J[U_k + \beta_k \bar{U}_k] \cdot \bar{U}_k = \epsilon D_k. \tag{68}$$

Then, it holds

$$J[U_k] - J[U_{k+1}] \geq J[U_k] - J[U_k + \bar{\alpha}_k \bar{U}_k] = \epsilon \bar{\alpha}_k D_k. \tag{69}$$

Since $J[U_k]$ is monotonically decreasing and bounded below (by zero), it converges to some value and (69) implies that $\bar{\alpha}_k D_k \to 0$. Let us assume that $\|D_k\| \geq \eta > 0$ for all k. Then, it must be that $\bar{\alpha}_k \to 0$ and thus $\beta_k \to 0$. Now divide both sides of (68) by $\|D_k\|$ and note that since $\bar{D}_k \equiv D_k / \|D_k\|$ belongs in the closed unit ball of $\mathbb{R}$, a compact set, it has at least one limit point $\bar{D}^*$ that satisfies from (68), $\bar{D}^* = \epsilon \bar{D}^*$, or $\bar{D}^* = 0$, which a contradiction. Therefore, we conclude that there is a subsequence of D_k that converges to zero.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers and the associate editor for helpful comments and suggestions.

References

[1] Buss M, Glocker M, Hardt M, von Stryk O, Bulirsch R, and Schmidt G, " Nonlinear hybrid dynamical systems: Modeling, optimal control, and applications," *Modelling, Analysis, And Design Of Hybrid Systems Lecture Notes In Control And Information Sciences*, Vol. 279, pp. 311-335, 2002.

[2] Wang CYE, Timoszyk WK, and Bobrow JE, "Payload maximization for open chained manipulators: Finding weightlifting motions for a Puma 762 robot," *IEEE Transactions On Robotics And Automation,* Vol. 17 (2), pp. 218-224, 2001.

[3] Cortes J, Martinez S, Ostrowski JP, McIsaac KA, "Optimal gaits for dynamic robotic locomotion," *Int. J Robot Res,* Vol. 20 (9), pp. 707-728, 2001.

[4] Pannocchia, G.; Wright, S.J.; Rawlings, J.B. "Existence and computation of infinite horizon model predictive control with active steady-state input constraints," *IEEE Transactions on Automatic Control,* Vol. 48 (6), pp. 1002-1006, 2003.

[5] Morari M, Lee, J.H., "Model predictive control: past, present and future," *Computers and Chemical Engineering,* Vol. 23 (4-5), pp 667-682, 1999.

[6] Rawlings, J.B.,"Tutorial overview of model predictive control," *IEEE Control Systems Magazine,* Vol. 20 (3), pp. 38-52, 2000.

[7] M. J. Tenny, S. J. Wright, and J. B. Rawlings "Nonlinear Model Predictive Control Via Feasibility-Perturbed Sequential Quadratic Programming," *Computational Optimization and Applications,* V. 28 87-121, 2004.

[8] M. Diehl, H.G. Bock, J. P. Schlöer, R. Findeisen, Z. Nagy, and F. Allgöer, "Real-time Optimization and Nonlinear Model Predictive Control of Processes Governed by Differential-algebraic Equations," *Journal of Process Control,* V. 12, pp. 577-585, 2002.

[9] W. C. Li and L. T. Biegler, "Multistep, Newton-Type Control Strategies for Constrained, Nonlinear Processes," *Chem. Eng. Res. Des.* Vol 67, pp. 562-577, 1989.

[10] J.T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control and Dynamics,* V. 21: (2) 193-207, 1999.

[11] A.E. Bryson and Y.C. Ho, *Applied Optimal Control,* Wiley, New York, 1995.

[12] Gill, P.E., Murray, W., and Wright, M.H., *Practical Optimization*, Harcourt Brace, 1981.

[13] von Stryk O. Numerical solution of optimal control problems by direct collocation. *Optimal Control*, Bulirsch R, Miele A, Stoer J, Well KH (eds), *International Series of Numerical Mathematics,* vol. 111. Birkshauser Verlag; Basel, 1993; 129143.

[14] D.G. Luenberger, *Linear and Nonlinear Programming,* Addison Wesley, 1989.

[15] R. Fletcher, *Practical Methods of Optimization*, Wiley, 2nd edition, 1987.

[16] Pantoja, J. F., "Differential Dynamic Programming and Newton's Method," *International Journal on Control,* Vol. 47, No. 5, pp. 1539-1553, 1988.

[17] Dunn, J. C., and Bertsekas, D, P., "Efficient Dynamic Programming Implementations of Newton's Method for Unconstrained Optimal Control Problems," *Journal of Optimization Theory and Applications,* Vol. 63, No. 1, pp. 23-38, 1989.

[18] Pantoja, J. F., and Mayne, D. Q., "Sequential Quadratic Programming Algorithm for Discrete Optimal Control Problems with Control Inequality Constraints," *International Journal on Control,* Vol. 53, No. 4, pp. 823-836, 1991.

[19] Liao, L. Z., And Shoemaker, C. A., "Convergence in Unconstrained Discrete- Time Differential Dynamic Programming," *IEEE Transactions on Automatic Control,* Vol. 36, No. 6, pp. 692-706, 1991.

[20] Wright, S. J., "Interior-Point Methods for Optimal Control of Discrete-Time Systems," *Journal of Optimization Theory and Applications,* Vol. 77, No. 1, pp. 161-187, 1993.

[21] Dohrmann, C.R. and Robinett, R.D., "Dynamic Programming Method for Constrained Discrete-Time Optimal Control," *Journal Of Optimization Theory And Applications,* Vol. 101, No. 2. pp. 259-283, 1999.