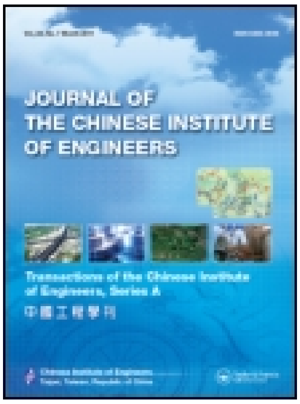



Journal of the Chinese Institute of Engineers
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcie20

# A convenient solver for solving optimal control problems

Chih-Hung Huang [a] & Ching-Huan Tseng [b]

[a] Department of Mechanical Engineering , National Chiao Tung University , Hsinchu, Taiwan 300, R.O.C.

[b] Department of Mechanical Engineering , National Chiao Tung University , Hsinchu, Taiwan 300, R.O.C. Phone: 886-3-5726111 ext. 55155 Fax: 886-3-5726111 ext. 55155 E-mail:

Published online: 04 Mar 2011.

Journal of the Chinese Institute of Engineers, Vol. 28, No. 4, pp. 727-733 (2005)


**Short Paper**

# A CONVENIENT SOLVER FOR SOLVING OPTIMAL CONTROL PROBLEMS

Chih-Hung Huang and Ching-Huan Tseng*



## ABSTRACT

This paper focuses on the development of a solver for solving optimal control problems. A developed numerical optimal control module integrated with the Sequential Quadratic Programming method is introduced. An optimal control problem solver based on the proposed method is implemented to solve optimal control problems efficiently in engineering applications. In addition, a systematic procedure for solving optimal control problems by using the optimal control problem solver is also proposed. A time-optimal benchmark problem presented in the literature is used to illustrate for the capability and facility of solving optimal control problems. The numerical results demonstrate the proposed method and the procedure suggested in this paper are helpful to engineers in solving optimal control problems in a systematic and efficient manner.

***Key Words:*** nonlinear programming (NLP), optimal control problem (OCP), sequential quadratic programming (SQP).

## I. INTRODUCTION

Two typical methods are usually used to solve optimal control problems: the indirect and direct approaches. The indirect approach is based on the solution of the first order necessary conditions for optimality. Pontryagin Minimum Principle (Pontryagin *et al*., 1962) and the dynamic programming method (Bellman 1957) are two common methods utilizing the indirect approach. The direct method (Jaddu and Shimemura 1999; Hu *et al*., 2002) is based on nonlinear programming (NLP) approaches that transcribe optimal control problems into NLP problems and apply existing NLP techniques to solve them. In most of practical applications, the control problems are described by strongly nonlinear differential equations hard to be solved by indirect methods. For those cases, direct methods can provide another choice to find the solutions.

In spite of extensive use of direct and indirect methods to solve optimal control problems, engineers still spend much effort on reformulating problems and implementing corresponding programs for different control problems. For engineers, this routine job will be tedious and time-consuming. Therefore, a systematic computational procedure for various optimal control problems has become an imperative for engineers, particularly for those who are inexperienced in optimal control theory or numerical techniques. Hence, the purpose of this paper is to apply NLP techniques to implement an OCP solver that assists engineers in solving optimal control problems with a systematic and efficient procedure. To illustrate the practicality and convenience of the proposed solver, a benchmark problem presented in the literature is chosen to illustrate the capability for solving optimal control problems. The results demonstrate the proposed solver can get the solution correctly and the procedure suggested in this paper can help engineers to deal with their problems.

The paper is organized as follows. In Section II, a general formulation of optimal control problems is given. The proposed NLP method and computational architecture for solving OCP are discussed in

*Corresponding author. (Tel: 886-3-5726111 ext. 55155; Fax: 886-3-5717243; Email: chtseng@cc.nctu.edu.tw)

The authors are with the Department of Mechanical Engineering, National Chiao Tung University, Hsinchu, Taiwan 300, R.O.C.

Section III. The systematic procedure by applying proposed solver to solve the OCP is described in Section IV. A benchmark problem presented in the literature is described and the numerical results obtained by applying the OCP solver are also demonstrated in Section V. Conclusions are drawn in Section VI.

## II. GENERAL FORMULATION OF OPTIMAL CONTROL PROBLEMS

The generalized Bolza problem formulation for optimal control problems can be defined as follows: Find the design variables $\boldsymbol{b}$, the control functions $\boldsymbol{u}(t)$ and terminal time $t_f$ which minimize the performance index

$$J_0 = \psi_0(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F_0(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t)dt \quad (1)$$

subject to the state (or system) equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t),\ t_0 \le t \le t_f \quad (2)$$

with initial conditions

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0(\boldsymbol{b}) \quad (3)$$

functional constraints

$$J_i = \psi_i(\boldsymbol{b}, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F_i(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t)dt \begin{cases} = 0;\ i = 1, \cdots, r' \\ \le 0;\ i = r' + 1, \cdots r \end{cases} \quad (4)$$

and dynamic point-wise constraints

$$\phi_j(\boldsymbol{b}, \boldsymbol{u}(t), \boldsymbol{x}(t), t) \le 0;\ j = 1, \cdots, q \quad (5)$$

where $\boldsymbol{b} \in \boldsymbol{R}^k$ is a vector of the design variables, $\boldsymbol{u}(t) \in \boldsymbol{R}^m$ is a vector of the control functions, and $\boldsymbol{x}(t) \in \boldsymbol{R}^n$ is a vector of the state variables. The functions $\boldsymbol{f}$, $\boldsymbol{\Psi}_0$, $F_0$, $\boldsymbol{\Psi}_i$, $F_i$ and $\phi_j$ are assumed to be at least twice differentiable.

The preceding definition extends the original Bolza problem to account for inequality constraints, as the original Bolza formulation containing only equality constraints is not general for the OCP. It also does not treat the design variables $\boldsymbol{b}$, which may serve a variety of useful purposes apart from obvious design parameters; e.g., weight and velocity of a vehicle. Also, when the terminal time $t_f$ is unconstrained (for optimization), a free time problem is obtained. Otherwise a fixed time problem is given. In addition, the initial conditions are separated from the functional constraints in Eq. (4) for practical considerations and the terminal conditions are treated as equality constraints in the first term of Eq. (4). The differential equations for the system in Eq. (2) are written in general first-order form. Eq. (5) represents the mixed state and control inequality dynamic constraints.

## III. NLP METHODS FOR SOLVING OCP

As mentioned in Section I, two common methods, the indirect and direct approaches, used to solve optimal control problems can be found in the literature. Each method has its fitness and difficulties for solving OCP. In this paper, a direct approach based on nonlinear programming (NLP) is adopted to develop an OCP solver. According to the strategies of discretization, NLP methods for solving OCP can be separated into two groups: the simultaneous and sequential strategies. In the simultaneous methods, the state and control variables are fully discretized and led to large-scale NLP problems that usually require special solution strategies (Cervantes and Biegler 2000) to obtain the solutions. In sequential NLP methods, only the control variables are discretized. Obviously, the sequential NLP method has smaller design spaces and is more efficient than simultaneous NLP methods. Therefore, this paper is focused on the sequential NLP method and applies it to develop the OCP solver.

Sequential Quadratic Programming (SQP) is one of the best NLP methods for solving large-scale nonlinear optimization and is frequently applied to solve optimal control problems (see, e.g., Gill *et al*., 2002, Betts 2000). Before applying the SQP methods, optimal control problems in which the dynamics are determined by a system of ordinary differential equations (ODEs) are usually transcribed into nonlinear programming (NLP) problems by discretization strategies.

### 1. Discretizing the Control Functions

The entire time interval $[t_0, t_f]$ is subdivided into $N$ general unequal time intervals and the grid is designated as

$$t_0, t_1, t_2, \cdots, t_{N-1}, t_N = t_f \quad (6)$$

The time intervals between the grid points are defined in a vector form as

$$\boldsymbol{T} = [T_1, T_2, \cdots, T_N]^T \quad (7)$$

where $T_i = t_i - t_{i-1}$ and $\sum_{i=1}^{N} T_i = t_f - t_0$ which generate the parameter set

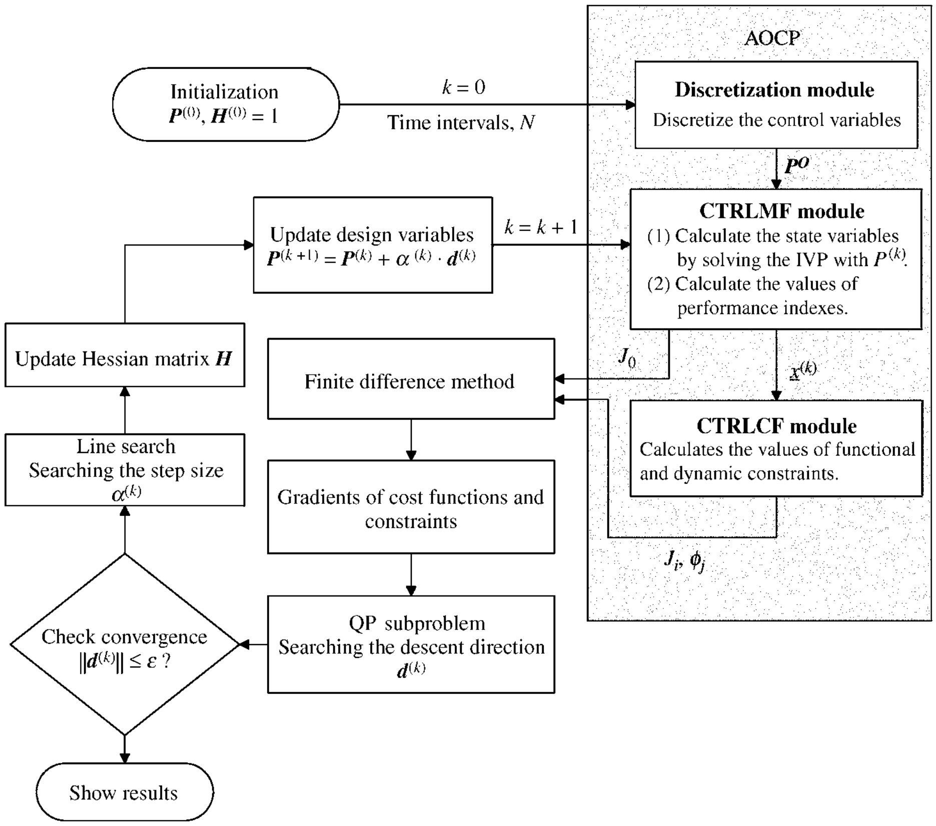



Fig. 1 Conceptual flow chart of the SQP method for solving OCP

$$\begin{aligned} U &= [u^{(1)}, u^{(2)}, \cdots, u^{(N)}]^T \\ &= [u_1(t_0), \cdots, u_m(t_0), u_1(t_1), \cdots, u_m(t_1), \cdots, \\ &\quad u_1(t_{N-1}), \cdots, u_m(t_{N-1})]^T \\ &= [U_1, \cdots, U_m, U_{m+1}, \cdots, U_{2m}, U_{2m+1}, \cdots, \\ &\quad U_{(N-1)m+1}, \cdots, U_{mN}]^T \end{aligned} \tag{8}$$

where $u^{(k)} \in R^m$ is the vector of control variables at the $k$-th time grid point. The continuity of the $u^{(k)}$ and their derivatives at the time grids are enforced by means of appropriate linear equality constraints. Any bounds on the $u^{(k)}$ at the nodes imply additional linear inequalities on the coefficients of the polynomial.

## 2. Admissible Optimal Control Problem Formulation

In this paper, the Admissible Optimal Control Problem (AOCP) formulation, which is based on sequential NLP methods, is developed and implemented. With AOCP, the system equation in Eq. (2) with initial condition in Eq. (3) is formed as an initial value problem (IVP) and the corresponding values of state variables can be calculated by solving the problem with the initial conditions $x_0$ and the values of design variables in each iteration. As mentioned before, the values of control can be approximated by a piecewise polynomial function, in which the coefficients are treated as design variables and determined in each iteration of SQP. Hence, Eqs. (2) and (3) form an IVP of state variables. Some good first order differential equation methods having variable step size and error control are available to solve the IVP, e.g. Adam's method and the Runge-Kutta-Fehlberg method. These solvers can give accurate results with user-defined error control. The state trajectories are internally approximated using interpolation functions in the differential equation solvers. Values of the state and control variables between the grid points can be also obtained with different kinds of interpolation schemes.

## 3. Computational Algorithm of AOCP

The architectural framework of the OCP solver, illustrated in Fig. 1, is composed of SQP and AOCP algorithms. The AOCP algorithm contains three major modules: discretization, CTRLMF and CTRLCF. The discretization module, which is mentioned in Section III.1, discretizes the control inputs according to specified time intervals. The computational algorithm of the OCP solver which integrates AOCP with SQP can be described as the following steps:

*Given*: Initial values of the design variables vector $\boldsymbol{P}^{(0)} = [\boldsymbol{b}^{(0)}, \boldsymbol{U}^{(0)}, \boldsymbol{T}^{(0)}]$ and Number of time intervals, $\boldsymbol{N}$. Initialize iteration counter $k$: = 0 and Hessian Matrix $\boldsymbol{H}^{(0)}$: = Identity $\boldsymbol{I}$.

1. Current design variable vector, $\boldsymbol{P}^{(k)}$, is passed to **CTRLMF** module of AOCP.
2. Evaluate the values of state variable, $\boldsymbol{x}^{(k)}$, by solving the IVP by substituting $\boldsymbol{P}^{(k)}$ into the system equation.

$$\dot{\boldsymbol{x}}^{(k)} = \boldsymbol{f}(\boldsymbol{b}^{(k)}, \boldsymbol{u}^{(k)}, \boldsymbol{x}^{(k)}, t),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0(\boldsymbol{b}^{(k)}) \tag{9}$$

3. Compute the values of performance indexes, $\boldsymbol{J}_0^{(k)}$.

$$J_0^{(k)} = \psi_0(\boldsymbol{b}^{(k)}, x(\boldsymbol{b}^{(k)}, \boldsymbol{U}^{(k)}, \boldsymbol{T}^{(k)}, t_f), t_f) + \int_{t_0}^{t_f} F_0(\boldsymbol{b}^{(k)}, \boldsymbol{U}^{(k)}, \boldsymbol{x}^{(k)}, t)dt \tag{10}$$

4. Substitute $\boldsymbol{x}^{(k)}$ into Eqs. (4) and (5) to evaluate the values of functional and dynamic constraints.
5. Evaluate $\nabla \boldsymbol{J}_0^{(k)}$, $\nabla \boldsymbol{J}_j^{(k)}$, and $\nabla \phi_j^{(k)}$ by using the finite difference method.
6. Find the descent direction, $\boldsymbol{d}^{(k)}$, by solving the QP subproblem.
7. Check convergence criteria, $\boldsymbol{d}^{(k)} \le \varepsilon$. If satisfied, stop and show the results.
8. Compute the step size, $\alpha^{(k)}$.
9. Update Hessian Matrix $\boldsymbol{H}^{(k)}$ by applying BFGS method.
10. Update design variables

$$\boldsymbol{P}^{(k+1)} = \boldsymbol{P}^{(k)} + \alpha \cdot \boldsymbol{d}^{(k)} \tag{11}$$

11. Increase iteration counter, $k \leftarrow k + 1$, go back to step 1.

## IV. SYSTEMATIC PROCEDURE FOR OCP

In this paper, the OCP is converted into an NLP problem by a discretization process and an admissible optimal control formulation mentioned in Section III. Then the optimizer based on the SQP method is used to solve the NLP problem numerically. In this paper the discretization process and the numerical schemes discussed in the previous section are implemented in the OCP solver. All of the complicated details of the transformation and numerical algorithms have been implemented in the OCP solver. The optimal control and state trajectories will be obtained and recorded in the output files. With the proposed OCP solver, engineers can focus their efforts on formulating their problems and then follow an efficient and systematic procedure to solve their optimal control problems. The following steps describe a systematic procedure for solving the OCP with the proposed OCP solver:

1. Program formulation: The original optimal control problem must be formulated according to the extended Bolza formulation.
2. Preparing two parameter files: One of the parameter files describes the numerical schemes used to solve the OCP and also the relationships between performance index, constraint functions, dynamic functions, state variables and control variables. The other parameter file includes the information on SQP parameters, such as convergence parameter, upper/lower bound and initial guess of design variables, etc.
3. Implementing user-defined subroutines.
4. Execute the optimization: The user-defined subroutines are compiled and then linked with the SQP solver, MOST (Tseng *et al*., 1996). Then, execute the optimization.

Obviously, the proposed OCP solver simplifies the computational procedure for solving OCP and aids engineers and students in solving optimal control problems.

## V. NUMERICAL EXAMPLES

### Time-Optimal Rest-to-Rest Maneuvering Problem

A single-axis, rest-to-rest maneuvering problem of flexible spacecraft used as a benchmark problem in many studies (Driessen 2000, Pao 1996, Liu and Wie 1992, Wie *et al*., 1993) is chosen as an example of the time-optimal control problem in this section. The system model, shown in Fig. 2(a), only with a scalar control input $u_1$(t) is considered here. Following the NLP formulation described in Section III, the optimal control can be defined as follows.

$$\text{Minimize}\ \ J_0 = \int_0^{t_f} dt = t_f \tag{12}$$

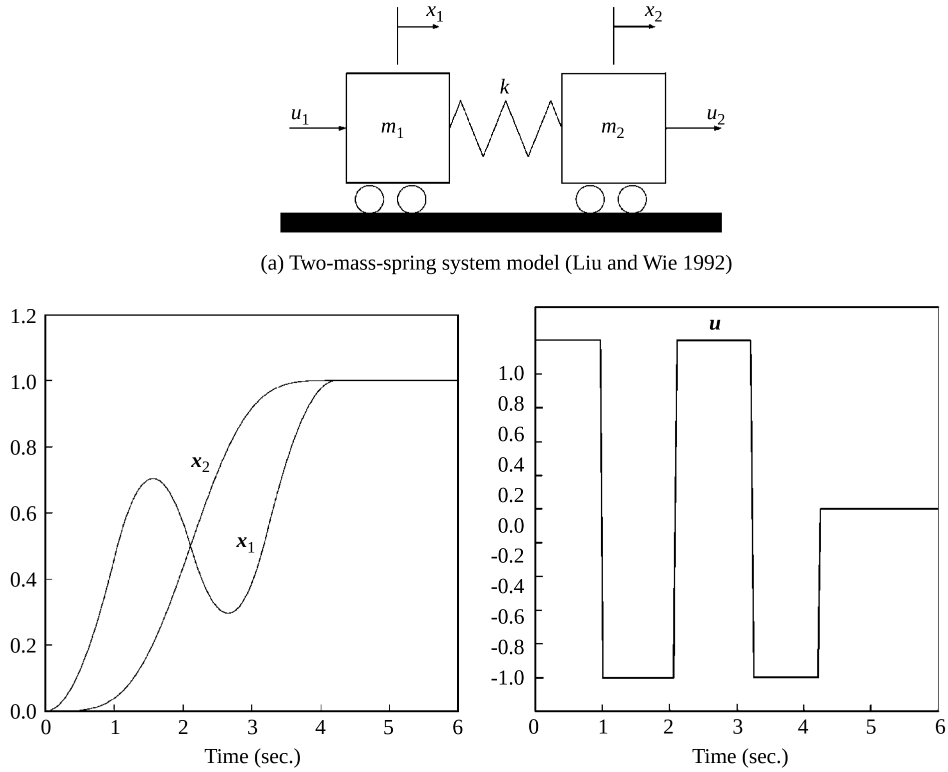
Subject to

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \frac{u_1}{m_1} - \frac{k}{m_1}(x_1 - x_2) \\ \dot{x}_4 &= \frac{k}{m_2}(x_1 - x_2) \end{aligned} \tag{13}$$

with initial states

$$\boldsymbol{x}^T(0) = [0, 0, 0, 0]^T \tag{14}$$

where $x_1$ and $x_2$ are the positions of body 1 and body




(a) Two-mass-spring system model (Liu and Wie 1992)

(i) State trajectories

(ii) Control trajectories

(b) Numerical results

Fig. 2 Time-optimal rest-to-rest maneuvering problem

2, respectively, the nominal parameters are $m_1 = m_2 = k = 1$ with appropriate units, and time is in seconds. The terminal state constraints and saturation constraints on control are described as:

$$\psi_1 = x_1(t_f) - 1 = 0 \quad (15)$$

$$\psi_2 = x_2(t_f) - 1 = 0 \quad (16)$$

$$\psi_3 = x_3(t_f) = 0 \quad (17)$$

$$\psi_4 = x_4(t_f) = 0 \quad (18)$$

$$\psi_5 = |u_1| - 1 \leq 0 \quad (19)$$

Time-optimal control problems often occur in many practical control problems. In this case, the derivation of the PMP is complex and thus the details are skipped. Following the procedure described in Section IV, users only need prepare two parameter files and user routines. Table 1 shows the user routines of this problem. By applying the proposed method and suggested procedure, a solution as $t_f = 4.2178746$ is obtained and the trajectories of the states and control input are shown in Fig. 2(b). In this problem, the proposed solver also obtains three switching times of input control as 1.00266823, 2.10892571 and 3.21518969. Those results agree with the results obtained by Liu and Wie (1992).

As the numerical results show, OCP is successfully converted into an NLP problem with the admissible control formulation and solved with the proposed method. The results show that the proposed method is applicable. According to the procedure suggested in this paper, users need not spend a vast amount of effort on programming in order to obtain solutions to problems. After formulating the problems and writing the user-defined routines, the proposed solver can solve the problems easily.

## VI. CONCLUSIONS

An optimal control problem solver, the OCP solver, based on the Sequential Quadratic Programming (SQP) method and integrated with many well-developed numerical routines is implemented in this paper. A systematic procedure for solving optimal control problems is also offered in this paper. A high-order nonlinear time-optimal control problem is used to demonstrate the capability of the OCP solver. The results show that the OCP solver can help engineers in solving optimal control problems with a systematic and efficient procedure.

**Table 1 User routines for solving the Benchmark problem**

```
// Parameters for numerical examples
#define m1 1.0
#define m2 1.0
#define k 1.0
//Routine to calculate the integral term of the performance index or functional constraint.
void ffn(double *B, double *U, double *Z, double *T, double *F, int NV, int NU, int NEQ, int N, int NBJ)
{
    *F = 0.0;
}
// Routine to calculate the first term of the performance index or functional constraint
// or dynamic constraint.
void gfn(double *B, double *U, double *Z, double *T, double *G, int NV, int NU, int NEQ, int N, int NBJ)
{
    switch (N) {
         case 0:
            *G = B[3];  /* B[3]:terminate time tf */
              break;
         case 1:
            *G = Z[0] - 1.0;  /* terminal constraints */
              break;
         case 2:
            *G = Z[1] – 1.0;
              break;
         case 3:
            *G = Z[2] ;
              break;
         case 4:
            *G = Z[3] ;
              break;
    };
}
// Routine to calculate the state trajectory.
void hfn(double *B, double *U, double *Z, double *DZ, double *T, int NV, int NU, int NEQ)
{
         DZ[0] = Z[2];
         DZ[1] = Z[3];
         DZ[2] = (U[0]/m1)-(k/m1)*(Z[0]-Z[1]);
         DZ[3] = (k/m2)*(Z[0]-Z[1]);
}
```



## ACKNOWLEDGMENTS

The research reported in this paper, was supported by a the National Science Council Grant, Taiwan, R.O.C., NSC90-2212-E009-039, which is greatly appreciated.

## NOMENCLATURE

| | |
|---|---|
| $\boldsymbol{b}$ | design variables |
| $\boldsymbol{d}^{(k)}$ | descent direction defined in SQP algorithm |
| $t_0$ | start time |
| $t_f$ | terminal time |
| $\boldsymbol{u}$ | control variable vector |
| $\boldsymbol{x}$ | state variable vector |
| $\boldsymbol{H}$ | Hessian Matrix |
| $N$ | number of time intervals |
| $\boldsymbol{P}$ | extended design variable vector |
| $T_i$ | the $i$-th time grid point |
| $\alpha$ | step size of SQP algorithm |
| $\varepsilon$ | convergence parameter of SQP algorithm |

## REFERENCES

Bellman, R., 1957, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA.

Betts, J. T., 2000, "Very Low-Thrust Trajectory Optimization Using a Direct SQP Method," *Journal of Computational and Applied Mathematics*, Vol. 120, pp. 27-40.

Cervantes, A., and Biegler, L. T., 2000, "A Stable Elemental Decomposition for Dynamic Process Optimization," *Journal of Computational and Applied Mathematics*, Vol. 120, pp. 41-57.

Driessen, B. J., 2000, "On-Off Minimum-Time Control with Limited Fuel Usage: Near Global Optima via Linear Programming," *Proceedings of the American Control Conference*, Chicago, IL, USA, pp. 3875-3877.

Gill, P. E., Murray, W., and Saunders, M. A., 2002, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, pp. 979-1006.

Hu, G. S., ONG, C. J., and Teo, C. L., 2002, "An Enhanced Transcribing Scheme for The Numerical Solution of a Class of Optimal Control Problems," *Engineering Optimization*, Vol. 34, No. 2, pp. 155-173.

Jaddu, H., and Shimemura, E., 1999, "Computational Method Based on State Parameterization for Solving Constrained Nonlinear Optimal Control Problems," *International Journal of Systems Science*, Vol. 30, No. 3, pp. 275-282.

Liu, Q., and Wie, B., 1992, "Robust Time-Optimal Control of Uncertain Flexible Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, pp. 597-604.

Pao, L. Y., 1996, "Minimum-Time Control Characteristics of Flexible Structures," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, pp. 123-129.

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., and Mischenko, E. F., 1962, *The Mathematical Theory of Optimal Processes*, Wiley, New York, USA.

Tseng, C. H., Liao, W. C., and Yang, T. C., 1996, "MOST 1.1 User's Manual," *Technical Report No. AODL-96-01*, Department of Mechanical Engineering, National Chiao Tung University, Taiwan, R.O.C..

Wie, B., Sinha, R., and Liu, Q., 1993, "Robust Time-Optimal Control of Uncertain Structural Dynamic Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No.5, pp. 980-983.

***Manuscript Received: Jun. 07, 2004***
***Revision Received: Sep. 12, 2004***
***and Accepted: Oct. 20, 2004***