Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

VISUAL MISER: AN EFFICIENT USER-FRIENDLY VISUAL PROGRAM FOR SOLVING OPTIMAL CONTROL PROBLEMS

Feng Yang

School of Automation Engineering, University of Electronic Science and Technology of China
No.2006, Xiyuan Ave, West Hi-Tech Zone, Chengdu, Sichuan, 611731, China

Kok Lay Teo, Ryan Loxton, Volker Rehbock, Bin Li[1] and Changjun Yu

Department of Mathematics and Statistics, Curtin University
GPO Box U1987 Perth, Western Australia 6845, Australia

Leslie Jennings

Department of Mathematics, University of Western Australia
Nedlands, Western Australia 6009, Australia

(Communicated by the associate editor name)

Abstract. The FORTRAN MISER software package has been used with great success over the past two decades to solve many practically important real world optimal control problems. However, MISER is written in FORTRAN and hence not user-friendly, requiring FORTRAN programming knowledge. To facilitate the practical application of powerful optimal control theory and techniques, this paper describes a Visual version of the MISER software, called Visual MISER. Visual MISER provides an easy-to-use interface, while retaining the computational efficiency of the original FORTRAN MISER software. The basic concepts underlying the MISER software, which include the control parameterization technique, a time scaling transform, a constraint transcription technique, and the co-state approach for gradient calculation, are described in this paper. The software structure is explained and instructions for its use are given. Finally, an example is solved using the new Visual MISER software to demonstrate its applicability.

1. **Introduction.** With the advances of modern computers and the increasing emphasis on optimal design of large scale dynamical systems under scarce availability of resources, optimal control theory has become a useful tool for solving many engineering, industrial and management problems. For a brief selection, see [1, 2, 3, 4, 5, 7, 9, 10, 11, 14, 17, 18, 27, 28, 29, 30, 31, 32, 33, 34, 35, 37, 41, 44, 45, 50, 53, 63, 64, 65, 66]. The main theoretical tools for solving optimal control problems analytically are the famous Pontryagin's minimum principle [1, 2, 8, 52] and the Hamilton-Jacobi-Bellman equation [4, 59]. There are many excellent books devoted to the theoretical aspects of optimal control, such as [1, 2, 5, 8, 52]. However, most practical problems arising in real world applications are too complex to solve analytically. Hence, many computational algorithms have been developed to determine numerical solutions of optimal control problems. Many methods are now

2010 *Mathematics Subject Classification.* Primary: 49M37, 49M25; Secondary: 65K05.
Key words and phrases. Optimal control, Visual MISER, Intel Visual Fortran, Optimization.
[1]Corresponding author.

available in the literature. For a brief selection, see [7, 14, 15, 16, 19, 22, 23, 24, 25, 26, 28, 30, 36, 38, 39, 40, 42, 43, 45, 47, 48, 49, 57, 59, 60, 62, 67].

There are also several general purpose software packages for solving optimal control problems. They include:(i) Recursive Integration Optimal Trajectory Solver (RIOTS) [56]; (ii) NUDOCCCS (NUmerical Discretization method for Optimal Control problems with Constraints in Controls and States) [6]; and (iii) The optimal control software package MISER3.3 [20], which is designed as a FORTRAN package. For a Matlab version, see [12].

In this paper, we are focused on the software packages MISER 3.3 [20] and MATLAB MISER [12]. Compared with the FORTRAN MISER, MATLAB MISER is somewhat easier to use, but it is much slower computationally to execute due to the interpretative nature within the MATLAB environment. This paper presents a new version of MISER called Visual MISER. This new version makes use of the Intel FORTRAN Studio XE combined with an optimizing FORTRAN compiler with high-performance libraries, performance profiling, thread and memory checking, static security analysis and other advanced tools. Intel Performance Libraries are also included. For example, Intel@Math Kernel Library (Intel@MKL) (for advanced mathematical processing) and Intel@Integrated Performance Primitives (Intel@IPP) (for multimedia, signal and data processing). These libraries offer highly optimized, threaded, and specialized functions. Furthermore, they have an excellent graphical user interface. With these attractive features, the visual FORTRAN version of the MISER software is coded in FORTRAN 90 and compiled using Intel Visual FORTRAN Compiler. It makes extensive use of graphical user interfaces (GUIs) to allow the user to input the problem at hand and to modify data. The equations to be defined by the user are compiled as Dynamic Link Library (*.dll) files, which are called by the main program.

The new Visual MISER is more user-friendly and computationally more efficient when compared with the original FORTRAN MISER and MATLAB MISER, respectively.

The remainder of the paper is organized as follows. We first describe a standard combined optimal control and optimal parameter selection problem suitable for MISER and present some theoretical preliminaries in Section 2. In Section 3, we introduce the Visual MISER software by explaining the numerical algorithm, software architecture design and operating procedure. In Section 4, an illustrative example, which is the well-known optimal Euler buckling beam problem is solved so as to demonstrate the applicability of the Visual MISER. Finally, in Section 5, we conclude the paper with a discussion of future research topics.

2. Problem Formulation and Theoretical Preliminaries.

In this section, we present the formulation of a combined optimal control and optimal parameter selection problem subject to canonical constraints as considered in [59]. Some relevant theoretical preliminaries are also given in this section.

2.1. Problem Formulation.

Consider a dynamical system governed by the following set of differential equations defined on the time horizon $[0, t_f]$.

$$\frac{dx(t)}{dt} = f(t, x(t), u(t), z), \tag{1}$$

with initial condition

$$x(0) = x^0(z), \tag{2}$$

where $x = [x_1, \ldots, x_{n_s}]^\top$, $u = [u_1, \ldots, u_{n_c}]^\top$, and $z = [z_1, \ldots, z_{n_z}]^\top$ denote the state, the control and the system parameter vectors, respectively; $f = [f_1, \ldots, f_{n_s}]^\top$ is a given function continuously differentiable with respect to x, u and z, and piecewise continuous with respect to t; and $x^0 = [x_1^0, \ldots, x_{n_s}^0]^\top$ is a given continuously differentiable function of z.

A combined optimal control and optimal parameter selection problem subject to canonical constraints, which is referred to as Problem (P), is stated formally as follows:

Given the dynamical system (1)-(2), find a pair (u, z) such that the cost function

$$G_0(u, z) = \phi_0(x(t_f), z) + \int_0^{t_f} g_0(t, x(t), u(t), z)dt \tag{3}$$

is minimized subject to the canonical equality constraints

$$G_k(u, z) = \phi_k(x(\tau_k), z) + \int_0^{\tau_k} g_k(t, x(t), u(t), z)dt = 0, \quad k = 1, \ldots, n_e, \tag{4}$$

the canonical inequality constraints

$$G_k(u, z) = \phi_k(x(\tau_k), z) + \int_0^{\tau_k} g_k(t, x(t), u(t), z)dt \geq 0, \quad k = n_e + 1, \ldots, n_q, \tag{5}$$

the linear control constraints on the control

$$\sum_{i=1}^{n_c} \alpha_{k,i} u_i(t) + \beta_k \quad \begin{cases} = 0 \\ \geq 0 \end{cases}, \quad t \in [0, t_f), \quad k = 1, 2, \ldots, n_g, \tag{6}$$

and constraints on the system parameter vector

$$z_i^L \leq z_i \leq z_i^U, \quad i = 1, \ldots, n_z, \tag{7}$$

where $\alpha_{k,i}, k = 1, \ldots, n_g; i = 1, \ldots, n_c$, $\beta_k, k = 1, \ldots, n_g$, and z_i^L and z_i^U, $i = 1, \ldots, n_z$ are given constants. Furthermore, τ_k, $k = 0, 1, \ldots, n_q$, are referred to as characteristic times, with $\tau_0 = 0$, $\tau_{n_q} = t_f$ and

$$\tau_k \leq \tau_{k+1}, \quad k = 1, \ldots, n_q - 2.$$

Each of the constraints on the control given by (6) may be in the form of an equality or an inequality. With the use of the control parameterization method, we assume a piecewise constant or piecewise linear approximation of each component of the control. Constraints (6) then represent a finite number of constraints on the corresponding control parameters. In other words, each of these constraints is equivalent to a set of linear constraints on the control parameters.

2.2. **Control Parameterization.** The control parameterization technique involves approximating the control function by a linear combination of basis functions, where the coefficients in the linear combination are decision variables to be chosen optimally [15, 36, 38, 40, 42, 45, 59, 62]. The approximate problem, which takes the form of an optimal parameter selection problem, can be solved as a nonlinear optimization problem by using gradient-based optimization techniques, such as sequential quadratic programming [58]. For this purpose, the gradient formulae of the cost and constraints functions need to be derived.

More specifically, let the time horizon $[0, t_f]$ be subdivided into p subintervals $[t_{k-1}, t_k), k = 1, \ldots, p$, where $t_k, k = 0, 1, \ldots, p$, are fixed knot points such that

$$0 = t_0 < t_1 < \cdots < t_{p-1} < t_p = t_f.$$

Then, the control function is approximated by a piecewise constant function as defined below.

$$u(t) \approx \sum_{k=1}^{p} \sigma^{p,k} \chi_{[t_{k-1},t_k)}(t), \tag{8}$$

where $\sigma^{p,k} = [\sigma_1^{p,k}, \ldots, \sigma_{n_c}^{p,k}]^\top, k = 1, \ldots, p$, are decision vectors to be chosen optimally, and χ_I is the indicator function defined by

$$\chi_I(t) = \begin{cases} 1, & \text{if } t \in I, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Under control parameterization, Problem (P), a combined optimal control and optimal parameter selection Problem (P) is approximated by the following pure optimal parameter selection problem, referred to as Problem $P(p)$:

Subject to the dynamical system

$$\frac{dx(t)}{dt} = \tilde{f}(t, x(t), \sigma^p, z), \quad t \in [0, t_f], \tag{10}$$

with initial condition

$$x(0) = x^0(z), \tag{11}$$

where

$$\tilde{f}(t, x(t), \sigma^p, z) = f(t, x(t), \sum_{k=1}^{p} \sigma^p \chi_{[t_{k-1},t_k)}(t), z),$$

and $\sigma^p = [(\sigma^{p,1})^\top, \ldots, (\sigma^{p,p})^\top]^\top$ with $\sigma^{p,k} = [\sigma_1^{p,k}, \ldots, \sigma_{n_c}^{p,k}]^\top, \quad k = 1, \ldots, p$, find a (σ^p, z) such that the cost function

$$\tilde{G}_0(\sigma^p, z) = \phi_0(\tilde{x}(t_f), z) + \int_0^{t_f} \tilde{g}_0(t, \tilde{x}(t), \sigma^p, z) dt \tag{12}$$

is minimized subject to the canonical equality constraints

$$\tilde{G}_k(\sigma^p, z) = 0, \quad k = 1, \ldots, n_e, \tag{13}$$

the canonical inequality constraints

$$\tilde{G}_k(\sigma^p, z) \geq 0, \quad k = n_e + 1, \ldots, n_q, \tag{14}$$

and constraints on the control parameter vector

$$\sum_{i=1}^{n_c} \alpha_{k,i} \sigma_i^{p,j} + \beta_k \geq 0, \quad j = 1, \ldots, p, \; k = 1, \ldots, n_g, \tag{15}$$

and constraints on the system parameter vector

$$z_i^L \leq z_i \leq z_i^U, \quad i = 1, \ldots, n_z, \tag{16}$$

where, for each $k = 1, \ldots, n_q$,

$$\tilde{G}_k(\sigma^p, z) = \phi_k(\tilde{x}(\tau_k), z) + \int_0^{\tau_k} \tilde{g}_k(t, \tilde{x}(t), \sigma^p, z) dt,$$

with

$$\tilde{g}_k(t, x, \sigma^p, z) = g_k\left(t, x, \sum_{k=1}^{p} \sigma^p \chi_{[t_{k-1},t_k)}(t), z\right).$$

2.3. **Gradient Computation.** The approximate problem (10)-(16) takes the form of an optimal parameter selection problem. It can be solved as a nonlinear optimization problem by using a gradient-based optimization technique, such as sequential quadratic programming. For this, we need the gradient formulae for the cost and constraint functions.

MISER uses the co-state method for gradient computation [59]. First, define the Hamiltonian function for each of the canonical functions $\tilde{G}_k, k = 0, 1, \ldots, n_q$, as follows:

$$H_k(t, \tilde{x}, \sigma^p, z, \lambda) = \tilde{g}_k(t, \tilde{x}, \sigma^p, z) + \left(\lambda^k\right)^\top \tilde{f}(t, \tilde{x}, \sigma^p, z) \tag{17}$$

where $k = 0$ corresponds to the cost function, $k > 0$ corresponds to the constraint functions, and λ^k is called the co-state or adjoint vector.

For each $k = 0, 1, \ldots, n_q$, the co-state system corresponding to the canonical function $\tilde{G}_k$ is given by

$$\frac{d\lambda^k(t)}{dt} = -\left[\frac{\partial H_k(t, \tilde{x}(t), \sigma^p, z, \lambda^p(t))}{\partial \tilde{x}}\right]^\top, \quad t \in [0, \tau_k), \tag{18}$$

with terminal condition

$$\lambda^k(\tau_k) = \left[\frac{\partial \phi_k(\tilde{x}(\tau_k), z)}{\partial \tilde{x}}\right]^\top. \tag{19}$$

Let $\lambda^k(\cdot|\sigma^p, z)$ be the solution of the co-state system (18)-(19) corresponding to (σ^p, z).

The gradient formula for each canonical function $\tilde{G}_k$, $k = 0, 1, \ldots, n_q$, with respect to σ^p is then given (see [59]) by

$$\frac{\partial \tilde{G}_k(\sigma^p, z)}{\partial \sigma^p} = \int_0^{\tau_k} \frac{\partial H_k(t, \tilde{x}(t), \sigma^p, z, \lambda^p(t))}{\partial \sigma^p} dt. \tag{20}$$

Furthermore, the gradient of each canonical function $\tilde{G}_k$, $k = 0, 1, \ldots, n_q$, with respect to the system parameter vector z is given (see [59]) by

$$\frac{\partial \tilde{G}_k(\sigma^p, z)}{\partial z} = \left[\lambda^k(0)\right]^\top \frac{\partial x^0(z)}{\partial z} + \frac{\partial \phi_k(\tilde{x}(t_f), z)}{\partial z} + \int_0^{\tau_k} \frac{\partial H_k(t, \tilde{x}(t), \sigma^p, z, \lambda^p(t))}{\partial z} dt. \tag{21}$$

2.4. **Variable Time Points.** Control parameterization requires the planning horizon $[0, t_f]$ to be partitioned into p subintervals $[t_{k-1}, t_k), k = 1, \ldots, p$, where

$$0 = t_0 < t_1 < \cdots < t_{p-1} < t_p = t_f.$$

As before, let us assume that the control function is approximated by a piecewise constant function, i.e.

$$u(t) \approx u^p(t) = \sum_{k-1}^{p} \sigma^{p,k} \chi_{[t_{k-1}, t_k)}(t), \tag{22}$$

where the control parameter vectors $\sigma^{p,k} = [\sigma_1^{p,k}, \ldots, \sigma_{n_c}^{p,k}]^\top$, $k = 1, \ldots, p$, are decision vectors to be chosen optimally.

The MISER packages assumes that the switching times t_k, $k = 1, \ldots, p-1$, of the approximate piecewise constant controls are fixed. However, in practice, it is desirable that these are also considered as decision variables to be chosen optimally. Thus, define

$$\nu_p = [t_1, \ldots, t_{p-1}]^\top. \tag{23}$$

Restricting controls to be of the form (22), the system (1)-(2) becomes:

$$\frac{dx(t)}{dt} = \hat{f}(t, x(t), \sigma^p, \nu^p, z), \quad t \in [0, t_f], \tag{24}$$

with the initial condition

$$x(0) = x^0(z), \tag{25}$$

where

$$\hat{f}(t, x(t), \sigma^p, \nu^p, z) = f\left(t, x(t), \sum_{k=1}^{p} \sigma^{p,k} \chi_{[t_{k-1}, t_k)}(t), z\right). \tag{26}$$

Similarly, by restricting controls to be of the form (22), the cost function (3), and the constraints (4) and (5) become:

$$\hat{G}_0(\sigma^p, \nu^p, z) = \phi_0(x(t_f), z) + \int_0^{t_f} \hat{g}_0(t, x(t), \sigma^p, \nu^p, z)dt, \tag{27}$$

$$\hat{G}_k(\sigma^p, \nu^p, z) = 0, \quad k = 1, \dots, n_e, \tag{28}$$

and

$$\hat{G}_k(\sigma^p, \nu^p, z) \geq 0, \quad k = n_e + 1, \dots, n_q, \tag{29}$$

respectively, where, for each $k = 1, \dots, n_q$,

$$\hat{G}_k(\sigma^p, \nu^p, z) = \phi_k(x(\tau_k), z) + \int_0^{\tau_k} \hat{g}_k(t, x(t), \sigma^p, \nu^p, z)dt, \tag{30}$$

with

$$\hat{g}_k(t, x, \sigma^p, \nu^p, z) = g_k\left(t, x, \sum_{k=1}^{p} \sigma^{p,k} \chi_{[t_{k-1}, t_k)}(t), z\right), \tag{31}$$

while the constraints (15) and (16) remain unchanged:

$$\sum_{i=1}^{n_c} \alpha_{k,i} \sigma_i^{p,j} + \beta_k \begin{cases} = 0 \\ \geq 0 \end{cases}, \quad j = 1, \dots, p; k = 1, \dots, n_g, \tag{32}$$

and

$$z_i^L \leq z_i \leq z_i^U, \quad i = 1, \dots, n_z. \tag{33}$$

It may be desirable to also allow for variable characteristic times in the canonical constraints (27). Thus, we need to ensure that there exist $m_i \in \{1, \dots, p-1\}$, $i = 1, \dots, n_p$, such that

$$t_{m_i} = \tau_i, \; i = 1, \dots, n_q. \tag{34}$$

For reference, we recall that

$$0 = t_0 < t_1 < \cdots < t_{p-1} < t_p = t_f. \tag{35}$$

We may now specify the approximate problem with variable time points as follows. It is referred to as Problem $(\hat{P}(p))$.

Given the dynamical system (24)-(25), find a (σ^p, ν^p, z) such that the cost function (27) is minimized subject to the constraints (28)-(29) and (32)-(35).

The gradient formulae of the cost function and the constraint functions with respect to the switching time vector ν^p can be readily obtained (see Theorem 5.4.1 of [59]). However, in view of the difficulties mentioned in [36, 42], the following time scaling transformation [60] is introduced.

Consider the new time variable $s \in [0, p]$, where $[0, p]$ is a new time horizon. The transformation needs to map from $[0, t_f]$ to $[0, p]$ such that the variable knots

$$t_1, \dots, t_{p-1}, t_p = t_f \tag{36}$$

in $[0, t_f]$ are mapped to the fixed knots

$$0, 1, \dots, p-1, p \tag{37}$$

in $[0, p]$. Note that since $\tau_i = t_{m_i}$, $i = 1 \dots, n_q$, each $\tau_i \in [0, t_f]$ is mapped to $m_i \in [0, p]$, $i = 1, \dots, n_q$.

The required transformation from $t \in [0, t_f]$ into $s \in [0, p]$ can be defined by the following differential equation:

$$\frac{dt(s)}{ds} = \sum_{k=1}^{p} \theta_k^p \chi_{[k-1,k)}(s) \tag{38}$$

with the initial condition

$$t(0) = 0, \tag{39}$$

where

$$\theta_k^p \geq 0, \quad k = 1, \dots, p. \tag{40}$$

Here, $\theta_k^p, k = 1, \dots, p$ are decision variables, and χ_I is the indicator function as defined in (9). Define $\theta^p = [\theta_1^p, \dots, \theta_p^p]^\top$ and note that this replaces the role of ν^p defined earlier in this section. Clearly, the piecewise constant control defined in (22) can be written as

$$u^p(t(s)) = \sum_{k=1}^{p} \sigma^{p,k} \chi_{[k-1,k)}(s), \quad s \in [0, p]. \tag{41}$$

Let $y(s) = x(t(s))$ and $\sigma^p = \left[(\sigma^{p,1})^\top, \dots, (\sigma^{p,p})^\top\right]^\top$. Then the time scaling transformation results in the following dynamical system:

$$\frac{d\hat{y}(s)}{ds} = F(\hat{y}(s), \sigma^p, \nu^p, z), \quad s \in [0, p], \tag{42}$$

with initial condition

$$\hat{y}(0) = \begin{bmatrix} x^0(z) \\ 0 \end{bmatrix}, \tag{43}$$

where

$$\hat{y}(s) = \begin{bmatrix} y(s) \\ t(s) \end{bmatrix}, \tag{44}$$

and

$$F(\hat{y}(s), \sigma^p, \theta^p, z) = \begin{bmatrix} \sum_{k=1}^{p} \theta_k^p f(\hat{y}(s), \sum_{k=1}^{p} \sigma^{p,k} \chi_{[k-1,k)}(s), z) \\ \sum_{k=1}^{p} \theta_k^p \chi_{[k-1,k)}(s) \end{bmatrix}. \tag{45}$$

Finally, the transformed optimal parameter selection problem may be stated as follows. It is referred to as Problem $(Q(p))$.

Subject to dynamical system (42)-(43), find a (σ^p, θ^p, z) such that the cost function

$$\gamma_0(\sigma^p, \theta^p, z) = \phi_0(y(p), z) + \int_0^p L_0(\hat{y}(s), \sigma^p, \theta^p, z) ds, \tag{46}$$

is minimized subject to the constraints

$$\gamma_k(\sigma^p, \theta^p, z) = 0, \quad k = 1, \dots, n_e, \tag{47}$$

$$\gamma_k(\sigma^p, \theta^p, z) \geq 0, \quad k = n_e + 1, \dots, n_q, \tag{48}$$

where, for each $k = 1, \dots, n_q$

$$\gamma_k(\sigma^p, \theta^p, z) = \phi_k(y(m_k), z) + \int_0^{m_k} L_k(\hat{y}(s), \sigma^p, \theta^p, z) ds \tag{49}$$

with

$$L_k(\hat{y}(s), \sigma^p, \theta^p, z) = \sum_{k=1}^{p} \theta_k^p g_k \left(\hat{y}(s), \sum_{k=1}^{p} \sigma^{p,k} \chi_{[k-1,k)}(s), z \right), \tag{50}$$

as well as subject to the constraints

$$\sum_{i=1}^{n_c} \alpha_{k,i}\sigma_i^{p,j} + \beta_k \begin{cases} = 0 \\ \geq 0 \end{cases}, \; j = 1, \dots, p, \; k = 1, \dots, n_g, \tag{51}$$

$$z_i^L \leq z_i \leq z_i^U, \quad i = 1, \dots, n_z, \tag{52}$$

and

$$\theta_k^p \geq 0, \quad k = 1, \dots, p. \tag{53}$$

Remark 1. Note that in the transformed problem $(Q(p))$, only the knots of the piecewise constant control contribute to the discontinuities of the right-hand side of the state differential equation. Thus, all locations of the discontinuities of the state differential equation are known and fixed during the optimization process. These locations will not change from one iteration to the next. Even when two or more of the original switching times coalesce, the number of these locations remains unchanged in the transformed problem. Furthermore, the gradient formulae of the cost function and constraint functions in the transformed problem with respect to θ^p can be obtained directly from the gradient formulae for the optimal parameter selection problem $(P(p))$ described in Section 2.3. The gradient formulae with respect to the control parameter vector and the system parameter vector can also be obtained from those given for Problem $(P(p))$ in Section 2.3.

Remark 2. The transformation described in this section has to be performed by the user before the problem is entered int any version of o the MISER software.

2.5. **Constraint Transcription.** Consider Problem (P) defined in Section 2.1 with the additional continuous inequality constraints

$$h_k(t, x(t), z) \geq 0, \quad \forall t \in [0, t_f], \quad k = 1, \dots, n_s, \tag{54}$$

where h_k, $k = 1, \dots, n_s$, are continuously differentiable functions of t, x and z.

After control parameterization and the time scaling transformation, the combined optimal control and optimal parameter selection problem is approximated by Problem $(Q(p))$ with the additional constraints:

$$\hat{h}_k(y(s), z) \geq 0, \quad \forall s \in [0, p], \quad k = 1, \dots, n_s, \tag{55}$$

where, for each $k = 1, \dots, n_s$,

$$\hat{h}_k(y(s), z) = h_k(t(s), y(s), z). \tag{56}$$

Let this problem be referred to as Problem $(P_s(p))$.

For each $k = 1, \dots, n_s$ the corresponding continuous state inequality constraint in (55) is equivalent to

$$\gamma_k(\sigma_p, \theta^p, z) = \int_0^p \min\{\hat{h}_k(\hat{y}(s), z), 0\} ds = 0. \tag{57}$$

However, the equality constraint (57) is non-smooth at those (σ_p, θ^p, z) which result in $\min\{\hat{h}_i(\hat{y}(s), z)), 0\} = 0$.

As the optimization routines built into Miser have difficulties with these non-smooth equality constraints, we approximate $\min\{\hat{h}_i(\hat{y}(s), z),), 0\}$ by

$$L_{k,\varepsilon}(\hat{y}(s), z) = \begin{cases} \tilde{h}_k(y(s), z), & \text{if} \quad \tilde{h}_k(y(s), z) > \varepsilon, \\ (\tilde{h}_k(y(s), z) + \varepsilon)^2 / 4\varepsilon, & \text{if} \quad -\varepsilon \leq \tilde{h}_k(y(s), z) \leq \varepsilon \\ 0, \quad \text{if} \quad \tilde{h}_k(y(s), z) \leq -\varepsilon, \end{cases} \tag{58}$$

where $\varepsilon > 0$ is an adjustable small positive parameter. This function is obtained by simply smoothing out the sharp corner of the function $\min\{\hat{h}_i(\hat{y}(s), z),), 0\} = 0$. See [36, 38, 59] for further details.

For each $k = 1, \ldots, n_s$ define

$$\gamma_{k,\varepsilon}(\sigma^p, \theta^p, z) = \int_0^p L_{k,\varepsilon}(\hat{y}(s), z)ds \tag{59}$$

We now define an approximate version of Problem $(P_s(p))$ by simply replacing the constraints (57) by

$$-\gamma + \Upsilon_{k,\varepsilon}(\sigma^p, \theta^p, z) \geq 0, \quad k = 1, \ldots, n_s. \tag{60}$$

Let the resulting approximate problem be referred to as Problem $(P_{S,\varepsilon,\gamma}(p))$. Note that the constraints (59) are already in canonical form, i.e., in the form of (47), where the functions ϕ_k are equal $-\gamma$.

Since the additional constraints are in canonical form, their gradient formulae with respect to θ^p, σ^p and z can be obtained from those in Sections 2.3 and 2.4.

Remark 3. Note that the constraint transcription described in this section has already been incorporated into the MISER software, so that the user merely has to specify these constraints in their original form (54).

3. **Software Structure.**

This section presents the key aspects of the Visual MISER software program.

3.1. **Numerical Computation.**

After applying control parameterization, Problem (P) is approximated by an optimal parameter selection problem in the form of Problem $(P(p))$. If the time scaling transform is used to supplement the control parameterization technique, the approximate problem obtained is in the form of Problem $(\hat{P}(p))$, which has the same structure as Problem $(P(p))$. For problems involving continuous state inequality constraints, the constraint transcription technique is used to approximate these continuous state inequality constraints by inequality constraints in canonical form. Thus, the resulting approximate problems for all of the situations mentioned above are optimal parameter selection problems in the form of Problem $(P(p))$. Problem $(P(p))$ can be solved as a nonlinear optimization problem by using a gradient-based optimization technique, such as sequential quadratic programming. The gradients of the cost function and the constraint functions are derived by using the co-state method.

In summary, to solve a problem the user needs to construct the mathematical formulation of the combined optimal control and optimal parameter selection problem. Then, the user also needs to apply the control parameterization method to approximate the combined optimal control and optimal parameter selection problem as a pure optimal parameter selection problem in the form of Problem $(P(p))$. If the switching times are also considered as decision variables, then the user needs to apply the time scaling transform to map the variable switching times into pre-fixed switching times. This yields an optimal parameter selection problem which is again in the form of Problem $(P(p))$. If the problem had continuous state inequality constrains, the user approximate Problem $(P_s(p))$ by using the constraint transcription method.

The resulting optimal parameter selection problem $(P(p))$ can be solved as a nonlinear optimization problem by using the SQP method (See [58]). To apply SQP, the values of the cost function, the constraint functions and the gradients are needed. For details, please refer to the following algorithm.

Algorithm 1

- Step 1. Choose an initial starting vector $(\sigma^{p,(0)}, z^{(0)})$.
- Step 2. Solve the state differential equations (10) with initial condition (11).
- Step 3. Compute the values of the cost function (12) and constraint functions (13) and (14) evaluated at $(\sigma^{p,(m)}, z^{(m)})$.
- Step 4. Solve the co-state differential equations (18) with terminal condition (19).
- Step 5. Compute the gradients of the cost function (12) and the constraint functions (13) and (14) with respect to $\sigma^{p,(m)}$ and $z^{(m)}$ by using the gradient formulas (20) and (21), respectively. For the constraint functions (15), they depend only on the control variables. Thus, they can be computed readily without involving co-state.
- Step 6. Call the SQP package.

3.2. **Optimization.** The sequential quadratic programming approach is used to solve the optimal parameter selection problem obtained via control parameterization. It is at the optimization stage that any error in function and gradient values is important. If function values are only accurate to 10^{-6} for example, then the convergence of the optimization to an accuracy of less than 10^{-6} is unlikely. Gradient values determine search directions and these values can be less accurate, except when testing the gradient of the Lagrangian or the projected gradient against zero for convergence. MISER's default accuracy for solving the state and co-state differential equations is 10^{-9}. For each of the constraints $\tilde{G}_k, k = 0, 1, \ldots, n_q$, the quadratures are computed to an accuracy to at least 10^{-8}. For the values of each of their gradients, the accuracy is at least 10^{-7}. The optimization convergence criteria defaults are 10^{-7} for constraints and 10^{-5} for a zero of the gradient.

3.3. **Ill- Conditioning.** A problem is said to be ill-conditioned if approximate solutions with large differences in the control or system parameters have similar cost values. This implies that even though the optimization has converged to an accuracy of 10^{-5} say, the control values could still be far from the exact optimal control values, even on the subspace of piecewise constant functions over a chosen set of knots. Yet, the computed control values are such that the same optimality conditions are satisfied as if they were the exact optimal control values to within 10^{-5}. One possible effect of ill-conditioning is that the computed values of the Lagrange multipliers do not appear to converge to any fixed values. Another effect is that the convergence of the optimization process slows down, and the decrease of the cost function values between successive iterations is negligible. Yet, the change of the decision parameters is markedly. This situation is similar to the case of zig-zagging across a long slowly downhill valley floor with steep sides.

3.4. **Gradient Checks.** There are two forms of gradient checks in MISER. The first is on the user supplied gradients $\nabla_x \tilde{f}, \nabla_u \tilde{f}, \nabla_z \tilde{f}, \nabla_x \tilde{g}_k, \nabla_u \tilde{g}_k, \nabla_z \tilde{g}_k, \nabla_x \phi_k, k = 0, 1, \ldots, n_q$, and $\nabla_z x^0$. They are compared with the second order finite difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

with the optimal increment being $\varepsilon^{1/3}$, where ε is ten times the machine precision. This assumes that $\tilde{f}, \tilde{g}_k, \phi_k$ and x^0 can be computed to an accuracy of ten times the machine precision. This checking procedure can be invoked a various stages in the optimization process.

The second gradient check is to compute an approximation (to second order again) to $\nabla_x \tilde{g}_k, \nabla_u \tilde{g}_k, \nabla_z \tilde{g}_k, k = 0, 1, \ldots, n_q,$. If the user supplied gradients are correct, these should be correct. Discrepancy at this level indicates that the package is not doing what it should be doing or that the user has set the precision levels on solving the differential equations too large, or that the quadrature is not accurate enough (not enough function values are used) or that the cost function or constraint functions change quickly for small changes in the control parameters. The optimal value of h in these approximations is $\delta^{1/3}$, where δ is the bound on the error in the values of the functions $\tilde{g}_k, k = 0, 1, \ldots, n_q$. Subsequently, it is sometimes difficult to distinguish whether or not the finite difference approximation is actually close to the gradient computed or not.

3.5. **Other Features.**

3.5.1. *Smoothing the absolute valued function.* Suppose that the integral term of one or more of the canonical functions involved an absolute valued function of the form

$$\int_0^{t_f} |L(t, x(t), u(t), z)| dt.$$

Clearly, the absolute valued function L is not differentiable when $L = 0$. Thus, the smoothing technique proposed in [59] is applied to construct a smooth function to approximate these non-differentiable functions. Subsequently, it yields a sequence of approximate optimal parameter selection problems, where the smoothing parameter ε is varied form 10^{-2} to 10^{-3} to 10^{-4}. Facilities exist in MISER can automatically compute the approximation of $|L|$ and the derivative of the approximation of L.

3.5.2. *Cost of Changing Control.* MISER has the facility to automatically add a penalty term to the objective which measures the change of the control. A large value of the penalty parameter will penalize a large change in the control, while a small value of the penalty parameter is used to regularize an ill-conditioned computation. Details can be found in [20, 41, 59]. The inclusion of these terms automatically invokes the absolute value function smoothing so a sequential optimization process is executed.

3.5.3. *Piecewise Linear Control Approximation.* In the control parameterization, the control function can be approximated by piecewise constant or piecewise linear functions. In fact, it can be approximated by functions of higher order of smoothness through introducing additional differential equations. For details, see Section 6.8.2 of [59]. Piecewise linear continuous controls can be specified directly in MISER without the suer having to transform their problem formulation.

3.5.4. *Special Classes of Constraints.* Consider a combined optimal control and optimal parameter selection problem. If a constraint involves only the system parameters, or only the control functions, or only the system parameters and control functions, they can be regarded as canonical constraints, and hence are readily handled by MISER. However, it is more efficient to regard them as standard constraints of underlying optimization problem and MISER allows the user to do so.

3.6. **Software Architecture.** The proposed software architecture is conceptualized to provide a consistent organization of the generic formulation (see Figure 1). There are six types of modules.

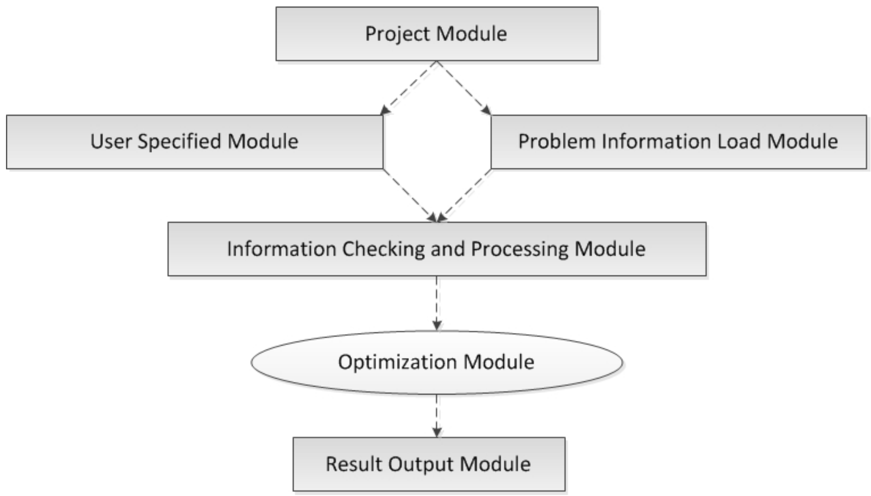

FIGURE 1. Visual MISER software modules

(i) The Project Module is used to create a new project corresponding to an optimal control problem.
(ii) The User Specified Module records the problem information from the user.
(iii) The Problem Information Load module is used to load all the problem information from an existing project, such as the number of the state, the number of the constraints, the initial state, the objective function, the constraint functions and their derivatives.
(iv) Information Checking and Processing Module checks whether or not the input parameters are within the allowable ranges. For example, if the maximum allowable number of the control variables is set as 50, then this module will check whether or not the number of control variables exceeds 50. It also checks the grammar of the functions defined by the user and then compiles these functions as a dynamic linking library file to be called by the main program when they are needed for carrying out computation. The advantage of this arrangement is that the main program will remain the same, even when the functions defined by the user are changed.
(v) The Optimization Module computes the values of the cost and constraint functions, their corresponding gradients, and then calls a nonlinear optimization solver to perform the optimization process.
(vi) The Results Output Module displays all outputs, which include state, control, co-state and the minimum value of the cost function obtained, either as graphs or text files.

3.7. **Parameters Input and Processing.** To solve an optimal control problem, nine types of parameters are required. Figure 2 gives an illustration of these parameters.

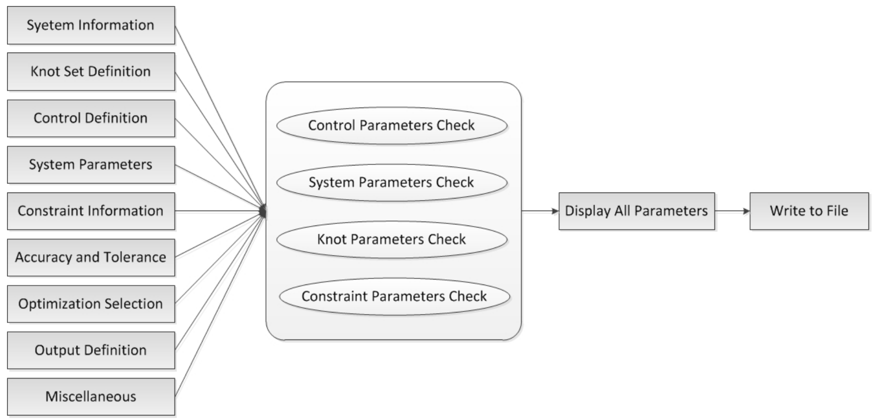

FIGURE 2. Input Parameters.

(i) The system information parameters includes the number of state variables, the number of control variables and and the number of system parameters, the initial time and the final time.

(ii) The knot set defines the partition of the time horizon for control parameterization. It includes the total number of the control knot sets, the types of the knot sets (equally spaced or specified by the user), and the total number of knots in each knot set.

(iii) The control definition can only be set after completing specification of the knot sets. The user can select the type of the continuity of the control function (piecewise constant or piecewise linear continuous) and set the control bounds and initial value of the control parameters for each knot. Moreover, it is also possible to specify various types of regularization on the control ill-conditioned problems.

(iv) The system parameter specification only requires the initial values and bound values of the system parameter vector z.

(v) The constraint information is relatively complex because ten distinct types of constraints are allowed in accordance to different processing situations, and there are different parameter settings for each type. Information required includes the number of characteristic times, their values, and wheatear to select $\varepsilon - \tau$ smoothing or absolute valued function smoothing if required. Absolute valued function smoothing can be used for constraints of the form of

$$\int_0^{\tau_i} |L_i(t, x(t), u(t), z)| dt.$$

(vi) The accuracy and tolerance module includes initial values of the $\varepsilon - \tau$ smoothing and the initial values of the absolute valued function smoothing if this is used. It also includes the tolerances of the numerical state and co-state solutions.

(vii) The optimization selection module is primarily for the selection of the optimization solver (NLPQL, FFSQP or NLPQLP) that is to be used to solve the underlying optimization problem, setting various operating modes for these solvers, accuracy of the constraints to be satisfied, and the convergence criterion.

(viii) The output definition module is used to specify the names of the error file, restart file, solution file, TTY file, plot file and save file.

(ix) The miscellaneous option module includes the desired frequency of the user supplied derivatives check, the type of the absolute value function smoothing, the choice of automatic restart (same number of knots, or double the number of knots or three times the number of knots), and a set of user specified (typically used for model constraints) parameters.

The Visual MISER software will check the validity of the information supplied before writing it into a data file. This checking is executed by Visual MISER automatically. It is tedious because if one of the parameters is changed, then it is likely that all the other parameters would also be changed accordingly. Finally, the validated information is written into a data file which will be called in subsequent stages.

3.8. **Functions Input and Processing.** Visual MISER requires user to input the cost and the constraint functions as well as their derivatives. These functions are divided into seven groups: (i) G0, (ii) G, (iii) PHI, (iv) F, (v) XZERO, (vi) GZ, (vii) H. For details, see Figure 3. Note that all REAL variables are DOUBLE PRECISION in these functions.

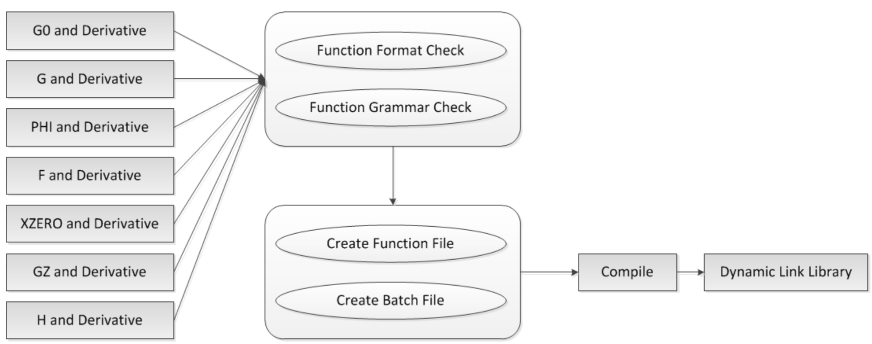

FIGURE 3. Functions input and processing

There are four functions in the G0 and Derivative: (i) OCG0 (g_0 in (3)); (ii) OCDG0DX; (iii) OCDG0DU; and (iv) OCDG0DZ. These functions describe the cost and its derivatives with respect to the state x, the control u, and the system parameter z, respectively.

There are four functions in the G and Derivative: (i) OCG (g_k in (4)); (ii) OCDGDX; (iii) OCDGDU; and (iv) OCDGDZ. These functions describe the canonical constraints and their derivatives with respect to the state x, the control u, and the system parameter z, respectively.

There are three functions in the PHI and Derivative: (i) OCPHI (ϕ_0 in (3) and ϕ_k in (4)); (ii) OCDPDX: and (iii) OCD-PDZ. These functions describe the terminal cost and the terminal canonical constraints as well as their derivatives with respect to the state x, and the system parameter z, respectively.

There are four functions in the F and Derivative: (i) OCF (f in (1)); (ii) OCDFDX; (iii) OCDFDU; and (iv) OCDFDZ. These functions describe the right hand side of the dynamic system and its derivatives with respect to the state x, the control u, and the system parameter z, respectively.

There are two functions in the XZERO and Derivative: (i) OCXZERO ($x(0)$ in (2)); and (ii) OCDX0DZ. These functions describe the initial condition for the state and its derivative with respect to the system parameter z.

There are two functions in the GZ and Derivative: (i) OCGZ ((7)); and (ii) OCDGZDZ. These functions describe the constraints on the system parameter and their derivatives with respect to the system parameter z.

There are three functions in the H and Derivative: (i) OCH; (ii) OCDHDX; and (iii) OCDHDZ. The OCH is catered for state differential equations where the state can experience jumps at various time points through given jump functions. For such situations, OCH will return the value of the jump function at each of these jump points for the state differential equations. The OCDHDX returns the gradient with respect to the state of each of the jump functions of the state differential equations. The OCDHDZ returns the gradient with respect to the system parameter of each of the jump functions of the state differential equations.

Similar to the case of parameters input, the Visual MISER will check the format and grammar of all the input functions defined by the user and saved them as a FORTRAN source file. An executable batch file will be generated so as to compile the source file automatically. In the batch file, the directory and name of the compiled file are required to be clearly stated. The main program will call the batch file once it is generated, and the batch file will call the FORTRAN compiler to compile the FORTRAN source file as a dynamic link library file.

3.9. **Optimization Calculation.** After the setting up of the parameters and each of the functions defined by the user being loaded as a dynamic link library file, the optimization calculation can be executed to solve the optimal parameter selection problem.

At first, the software calls windows system command “LoadLibrary” to load the functions as a dynamic link library file and to initialize the user’s defined functions. Then, it prepares internal parameters for optimization running environment and checks the gradients of the cost and constraint functions. Finally, it reads parameter data file to load all the optimization parameters which have been chosen by the user.

Now, the optimization iterations can be executed. The constraint transcription technique ($\varepsilon - \tau$ technique) will be applied if there are continuous state inequality constraints (i.e., the inequality constraints are to be satisfied for all time $t \in [0, t_f]$).

In every iteration, it will also calculate the values of the cost and constraint functions as well as their gradients defined by the user. If revived, the software can also calculate the numerical gradients of the cost and constraint functions. Finally, the values of the cost and constraint functions and their gradients are provided to the optimization solver NLQPL or NLQPLP or FFSQP to execute the optimization calculation.

At the completion of an iteration, the software will check the stopping condition. If the stopping condition is not satisfied, it will restart a new iteration; otherwise, it will write the solutions to different files listed below:

- .err–error messages are written to this file.
- .res–restart file, which is in the same format as the data input file, but contains the optimal values of the control and system parameters just found.
- .sol–solution file, which contains the final solution.
- .tty–terminal output.
- .plt–offline plot file.

- .sav–save file for intermediate results (It is possible to save the current iterate at any step of the optimization).

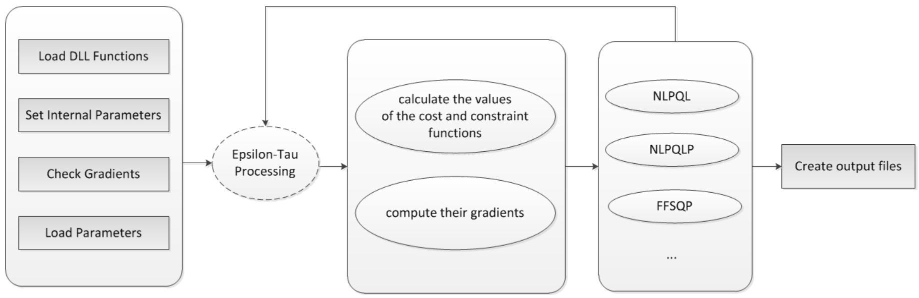

FIGURE 4. Optimization calculation process

3.10. **Execution.** For the Execution, all the parameters needed by the optimization calculation (such as system information, knot sets, control definition, system parameter, constraint information) are set and stored in a data file. Then, the user defines the cost function, the constraint functions and their derivatives, which will be compiled as a dynamic link library file. If the creation of the dynamic link library file fails, the user has to correct the format or grammar of the functions defined by the user according to the error messages in the compiled report. Once this is done, the user selects the "calculate" command to compute the desired solution and the results information will be displayed on the screen. If the optimization computation is interrupted abnormally, the user may choose to adjust the initial values of the control or system parameters and run the program again. The minimum value of the cost function and the running time will be displayed on the screen if the optimization computation is successful. Then, the user can check all the result files or plot the state and control curves (see Figure 5).

4. An Illustrative Example.

4.1. **Optimal Euler Buckling Beam.** Let us consider an interesting yet simple example, where the minimal cross-sectional area is to be obtained for a beam subject to a force being applied to its ends [65].

$$\min_{u,z_1} g_0(u, z_1) = -z_1$$

subject to

$$\begin{aligned} \dot{x}_1(t) &= x_2(t), \quad x_1(0) = 0, \\ \dot{x}_2(t) &= \frac{-z_1 x_1(t)}{x_3^2(t)}, \quad x_2(0) = 1, \\ \dot{x}_3(t) &= u(t), \end{aligned}$$

the constraints are:

$$\begin{aligned} x_1(1) &= 0, \\ \int_0^1 x_3(t)dt - 1 &= 0, \\ x_3(t) - 0.5 &\geq 0, \quad \forall t \in [0, 1]. \end{aligned}$$

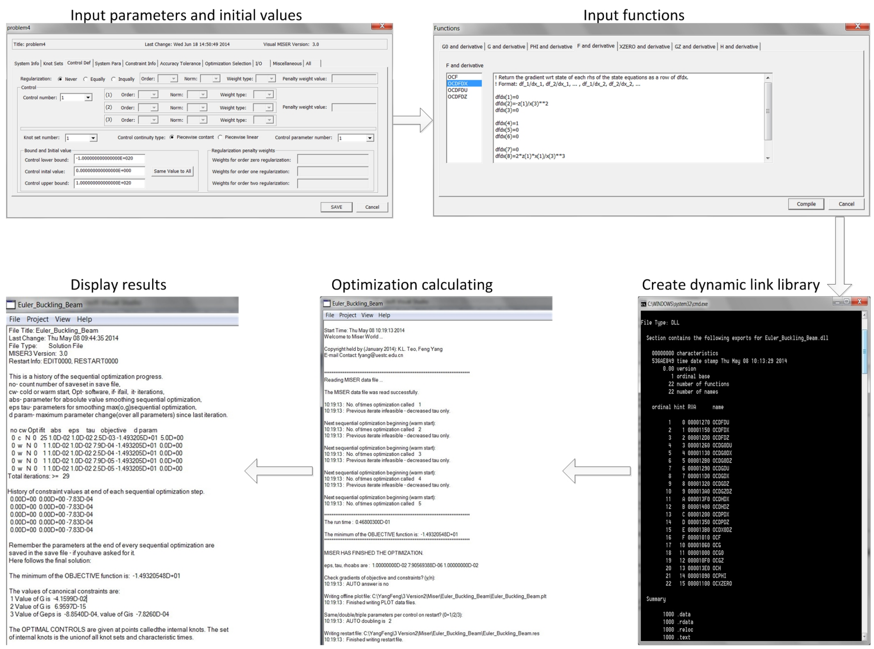

FIGURE 5. The operation procedures

Since there is no initial condition on x_3, we introduce a new system parameter z_2, with $z_2 \geq 0.5$, so that $x_3(0) = z_2$. Now, the optimization problem has one control and two system parameters. The third constraint is transcribed automatically by the software using the $\epsilon - \tau$ algorithm.

4.2. **Running Environment.** We need to identify the information on the running environment for the Visual MISER test problem. This crucial information includes operating system, central processing unit and memory size. Table 1 shows the details of the environment information that is conveniently available.

Item	**Value**
System Manufacturer	Dell Inc.
System Model	OptiPlex 790
Operating system	Windows 7 Enterprise Service Pack1
System type	64-bit Operating System
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz, 4Core(s)
Installed memory(RAM)	8.00GB

TABLE 1. Running environment information

4.3. **Problem Parameters.** For the optimal Euler buckling beam problem, the state variables are x_1, x_2, x_3, the control variable is u_1, and the system parameters are z_1, z_2. To solve this problem, we define a knot set that contains 21 knots. The knot type is defined to be equally spaced and the control continuity type is defined as piecewise constant. The control initial value is set to 0 for all time $t \in [0, t_f]$, and the initial values of the two system parameters are set as 10 and 1, respectively. The optimization solver choice is between NLPQLP, NLPQL, and FFSQP. Table 2 shows the details of the problem parameters setting.

Item	**Value**
number of states	3
number of controls	1
number of system parameters	2
number of knot sets	1
knot type	Equally spaced
number of knots	21
Control regularization	Never
Control continuity type	Piecewise constant
Control initial	0
System Parameter initial 1	10
System Parameter initial 2	1
Optimization method	NLPQLP/NLPQL/FFSQP

TABLE 2. Problem parameter setting

4.4. **Running results.**

4.4.1. *MATLAB MISER.* The Optimal Control Toolbox, the Matlab version of MISER3.2, requires Matlab version 5.3 or later. Here, we use Matlab 2012b version for testing. Upon satisfactory completion, the program provides separate plots of the control variables and the state variables against time (see Figure 6).

4.4.2. *Visual MISER.* The standard Visual MISER will calculate optimal control problems with up to 20 state variables, 10 control variables, 10 system parameters, 10 canonical constraints, and up to 481 quadrature subintervals. For the illustrative example, the first step creates a new project and we enter the parameters as shown in Table 2. Then we enter the user defined functions and their gradients. In particular, note the ordering of Jacobian matrices in the one dimensional arrays. After the user defined functions and their gradients are successfully compiled as a the dynamic link library file, we call the"calculate" menu command to calculate and generate the result files of this problem.

There is a feedback to the user if the optimization fails to converge. The user is then able to change options for optimization parameters or edit the user defined functions and their gradients, and restart the optimization calculation. In this illustrative example, we choose different nonlinear optimization solvers (NLPQLP/NLPQL

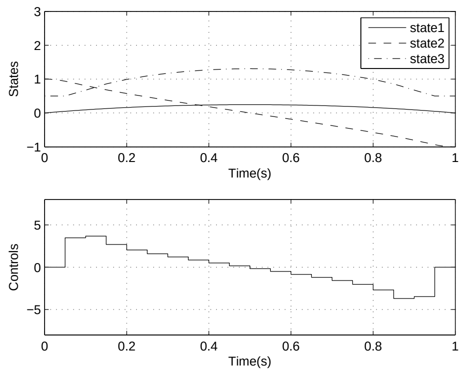

FIGURE 6. MATLAB results

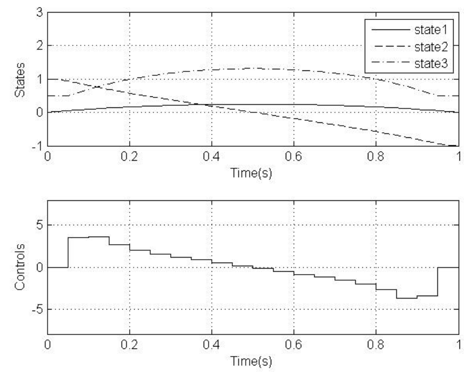

FIGURE 7. NLPQLP results

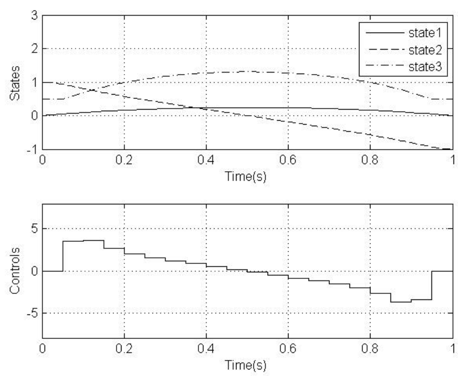

FIGURE 8. NLPQL results

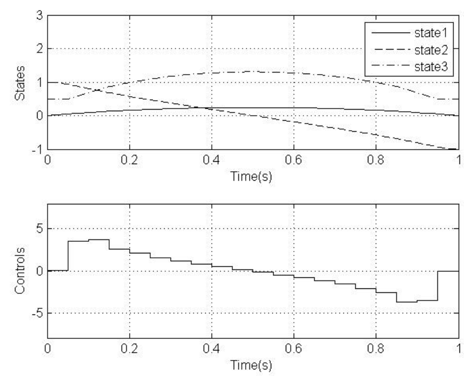

FIGURE 9. FFSQP results

or FFSQP) to solve the same optimal control problem and output the results as shown in Figure 7, Figure 8 and Figure 9. It shows that all results obtained by Visual MISER are similar to those obtained by MATLAB MISER3.2.

4.5. **Efficiency Analysis.** Now let us focus on the efficiency analysis of different software packages and optimization solvers. We will compare the optimization results obtained without compiler optimization. Because all tests use double precision computation, the optimized cost function value is double precision also. The calculated results show that the minimum cost function value is -12.7668247 that where using MATLAB MISER3.2. It is -12.7668070 when using the Visual MISER with the NLPQL/NLPQLP solver, and it is -12.7697869 when using the Visual MISER with the FFSQP solver. All the optimal cost values obtained are reasonably similar.

The computational times of different software packages and optimization solvers are compared also. The MATLAB MISER3.2 uses “tic” and “toc” function and the Visual MISER call the “cpu_time” function to evaluate optimization calculation time. The elapsed time are measured in seconds and displayed in Figure 10.

We have tested twelve times for all situations and the optimization calculation time is marked on the curves (see Figure 10). The MATLAB MISER3.2, takes

51-52 seconds, the Visual MISER with FFSQP takes 2.4-2.5 seconds, while the Visual MISER with NLPQL/NLPQLP takes 0.10-0.14 seconds to complete the optimization calculation. These results show that Visual MISER is much faster than MATLAB MISER3.2 in a range from one to two orders of magnitude. Meanwhile, it is noted that the Visual MISER with the FFSQP solver needs slightly more computational time than Visual MISER with the NLPQL/NLPQLP solver.

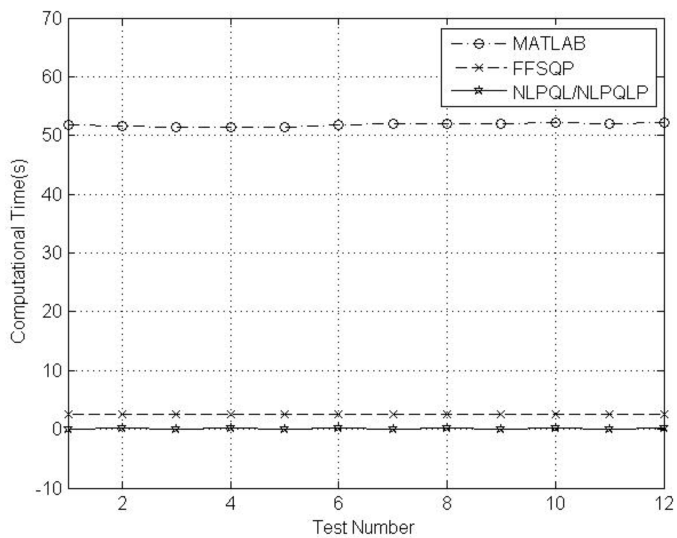

FIGURE 10. Efficiency comparison

4.6. **Compiler optimization.**

4.6.1. *Compiler optimization parameters.* The optimization option of Intel FORTRAN compiler is another factor that affects the computational efficiency, in addition to the Visual MISER software itself. The compiler optimization includes the following options:

- Optimization. We choose the option of "Maximize Speed plus Higher Level Optimization (/O3)". The compiler with such an option has a much faster speed in execution. This is especially so for many routines in the shared libraries for Intel microprocessors.
- Inline Function Expansion. We choose the option of "Any Suitable". The reason for such a choice is that the execution speed of functions compiled in the form of Inline Function Expansion is much faster than in other form. Thus, the functions to be used should be compiled as in the form of Inline Function Expansion when possible.
- Favor Size or Speed. We choose the option of "Favor Fast Code". This option is to ensure that the fastest execution speed is achieved.
- Parallelization. We choose the option of " Yes (/Q-parallel)". The choice of this option is to allow the codes within the loop processing be automatically converted to parallel multi-threated codes. In this way, the execution speed of the code will be increased significantly on a multi-processor machine.

- Prefetch Insertion. We choose the option of " Aggressive (/Qopt-prefetch=3)"). This option is to ensure that the processor will access the data in Cache rather than in memory.
- Interprocedural Optimization. We choose the option of "Multi-file (/Qipo)". This option is to allow the compiler to decide whether to create one or more object files based on an estimate of the size of the application.
- Enable Matrix Multiply Library Call. We choose the option of "Yes (/Qopt-matmul)". The option is to allow the compiler to call the Matmul Library during the matrix multiplication loop nests so as to improve performance.

When the setting of these options is completed (see Figure 11), the Visual MISER is compiled again, yielding an optimized version of the compiled Visual MISER, which is ready to be run.

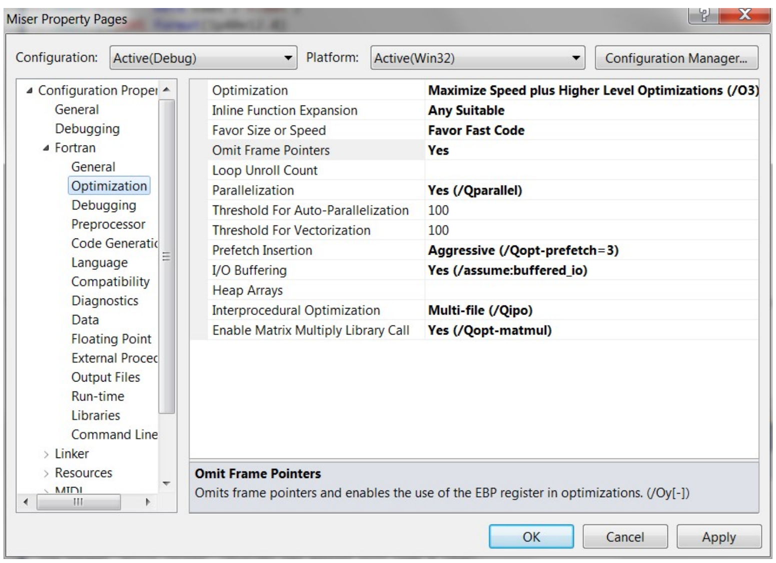

FIGURE 11. Compiler optimization parameters

4.6.2. *Compiler Optimization Efficiency.* We choose NLPQL/NLPQL and FFSQP solvers to calculate the illustrative problem again with the optimized compiled Visual MISER. The results show that the minimum cost function value is the same as above, but the computational time has been cut down significantly (see Figure 12). The FFSQP solver elapsed time is shortened from 2.4-2.5 seconds to 1.1-1.2 seconds, and NLPQL/NLPQLP solver elapsed time is shortened from 0.10-0.14 seconds to 0.06-0.07 seconds. It is obviously that the compiler optimization has improved the computational efficiency by a factor of two.

5. **More Examples.**

5.1. **Example 1.** We consider a realistic and complex problem of transferring containers from a ship to a cargo truck at the port of Kobe. It is taken from [54]. The crane is driven by a hoist motor and a trolley drive motor. For safety reason,

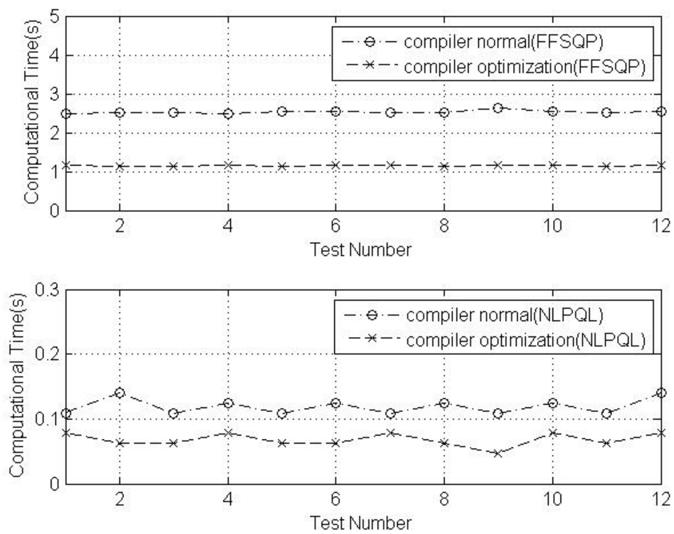

FIGURE 12. Comparison of computational time between compiler normal and compiler optimization

the objective is to minimize the swing during and at the end of the transfer. The problem is summarized after appropriate normalization as follows:

$$\text{minimize } \left\{ g_0 = 4.5 \int_0^1 \left[(x_3(t))^2 + (x_6(t))^2 \right] \mathrm{d}t \right\},$$

subject to the dynamical equations

$$\begin{aligned}
\dot{x}_1(t) &= 9x_4(t),\\
\dot{x}_2(t) &= 9x_5(t),\\
\dot{x}_3(t) &= 9x_6(t)\\
\dot{x}_4(t) &= 9(u_1(t) + 17.2656x_3(t)),\\
\dot{x}_5(t) &= 9u_2(t),\\
\dot{x}_6(t) &= -\frac{9}{x_2(t)}[u_1(t) + 27.0756x_3(t) + 2x_5(t)x_6(t)],
\end{aligned}$$

where

$$\begin{cases} x(0) = [0, 22, 0, 0, -1, 0]^\top, & (61\text{a}) \\ x(1) = [10, 14, 0, 2.5, 0, 0]^\top, & (61\text{b}) \end{cases}$$

and

$$|u_1(t)| \le 2.83374,$$
$$-0.80865 \le u_2(t) \le 0.71265, \ \forall t \in [0, 1],$$

with continuous state inequality constraints

$$|x_4(t)| \le 2.5, \ \forall t \in [0, 1],$$
$$|x_5(t)| \le 1.0, \ \forall t \in [0, 1].$$

The bounds on the states can be formulated as the continuous inequality constraints as follows:

$$g_1 = -x_4(t) + 2.5 \geq 0, \tag{62}$$

$$g_2 = x_4(t) + 2.5 \geq 0, \tag{63}$$

$$g_3 = -x_5(t) + 1.0 \geq 0, \tag{64}$$

$$g_4 = x_5(t) + 1.0 \geq 0. \tag{65}$$

The partition is set as 20, and we use Visual Miser to solve this problem. The optimal state variables and the optimal control are shown in Figure 13 and Figure 14. From Figure 13, we can see that the continuous inequality constraints (62) - (65) are stratified and the terminal state constraints (61a) - (61b). The corresponding optimal function value is $5.24207969e-03$ and it is $5.56644743e-003$ by using Matlab Miser 3.2. The CPU time for running this example with Visual Miser is 0.17160s, while it is 23.41688s with Matlab Miser 3.2.

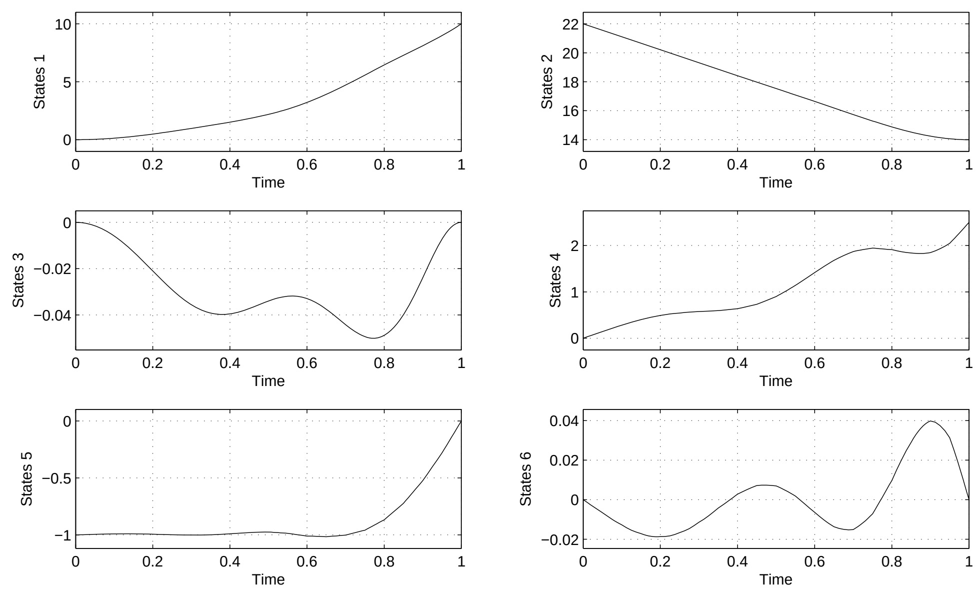

FIGURE 13. The optimal states of Example 1

5.2. **Example 2.** We consider a minimum time problem with a singular arc as given below.

$$\min_{u,t_f} J(u, t_f) = t_f$$

subject to the dynamics

$$\dot{x}_1(t) = u(t),$$

$$\dot{x}_2(t) = \cos x_1(t),$$

$$\dot{x}_3(t) = \sin x_1(t),$$

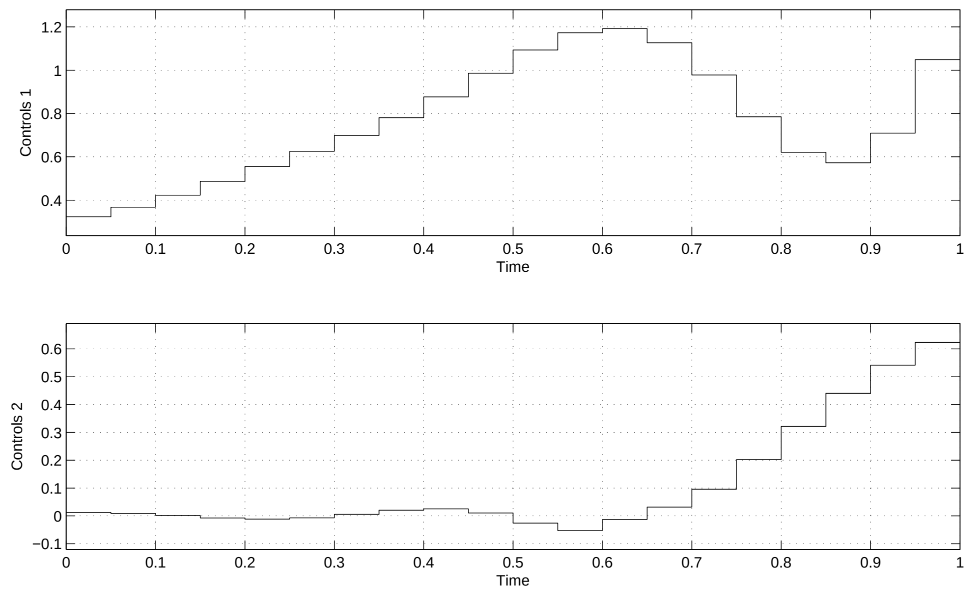

FIGURE 14. The optimal control of Example 1

with the initial conditions

$$x(0) = [\pi/2, 4, 0]^\top,$$

the terminal state constraints

$$x_2(t_f) = x_3(t_f) = 0, \tag{66}$$

and the bounds on the control

$$|u(t)| \leq 2.$$

After applying the time scaling transform, we have the canonical form

$$\min_{u,t_f} J(u, t_f) = x_4(1)$$

$$\begin{aligned}
\dot{x}_1(t) &= u_2(t)u_1(t), \\
\dot{x}_2(t) &= u_2(t)\cos x_1(t), \\
\dot{x}_3(t) &= u_2(t)\sin x_1(t), \\
\dot{x}_4(t) &= u_2(t), \\
x(0) &= [\pi/2, 4, 0]^\top, \\
G_1 = x_2(1) = 0,&\ G_2 = x_3(1) = 0.
\end{aligned}$$

The optimal states and optimal control are shown in Figure 15 and Figure 16. As it is shown in Figure 15, the terminal state constraints (66) are satisfied. The obtained minimum time is 4.32120, and it is 4.32117s by using Matlab Miser 3.2. The CPU time for running this example with Visual Miser is 0.07800s, while it is 26.85638s with Matlab Miser 3.2.

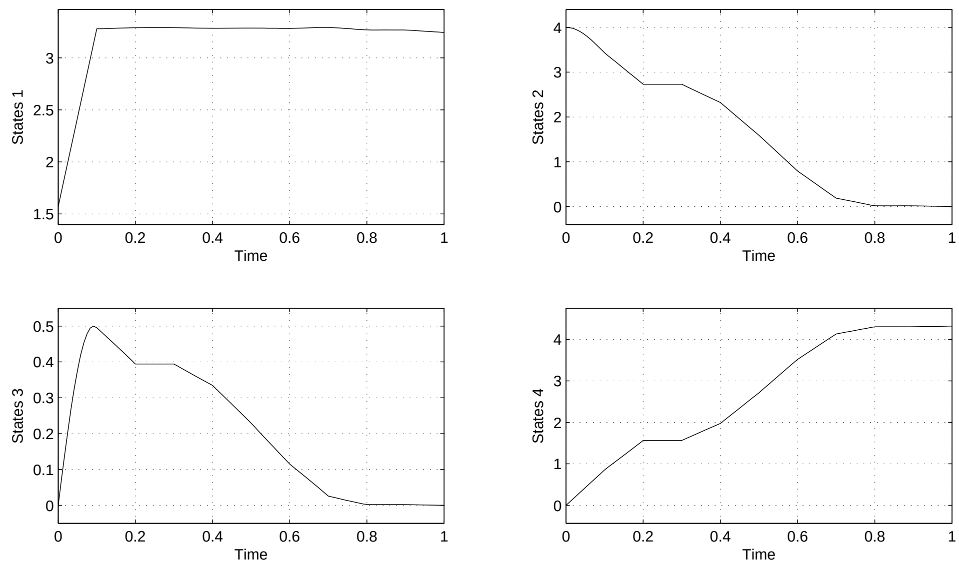

FIGURE 15. The optimal states of Example 2

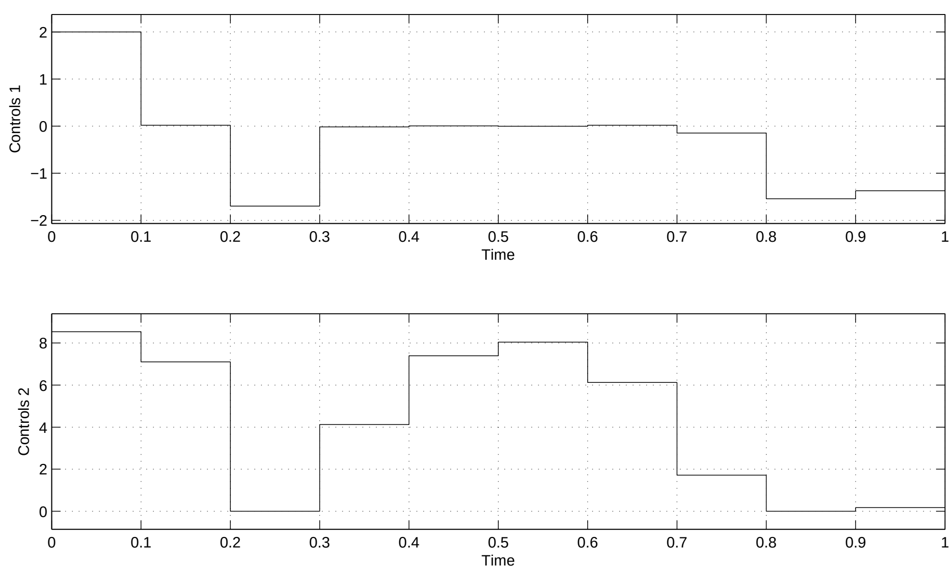

FIGURE 16. The optimal control of Example 2

5.3. **Example 3.** Consider a linear problem with bang-bang control as given below.

$$\min_u J(u) = \int_0^1 (-6x_1(t) - 12x_2(t) + 3u_1(t) + u_2(t))\,\mathrm{d}t$$

subject to the linear dynamics

$$\begin{aligned}\dot{x}_1(t) &= u_2(t),\\ \dot{x}_2(t) &= -x_1(t) + u_1(t),\end{aligned}$$

with the initial conditions

$$x(0) = [1, 0]^\top,$$

and with the bounds on the control

$$|u_1(t)| \le 10, \;\; |u_2(t)| \le 10.$$

The optimal states and optimal control are shown in Figure 17 and Figure 18. As it is shown in Figure 18, the optimal control is in the form of Bang-Bang type. The corresponding optimal objective function value is -41.34000 and it is the same by using Matlab Miser 3.2. The CPU time for running this example with Visual Miser is 0.09360s, while it is 55.83362s with Matlab Miser 3.2.

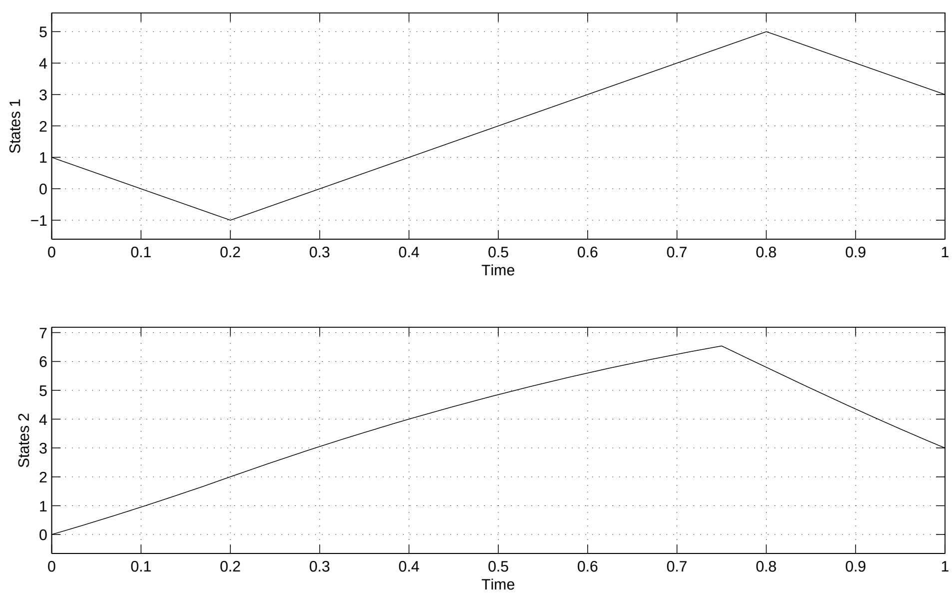

FIGURE 17. The optimal states of Example 3

6. **Concluding Remarks and Future Research.** In this paper, we decribe the result of applying the Visual Fortran Package to the Miser optimal control software. This provides a user-friendly way to carry out the cumbersome computations involved for solving a general combined optimal control and optimal parameter selection problem, where various types of constraints are formulated in a unified

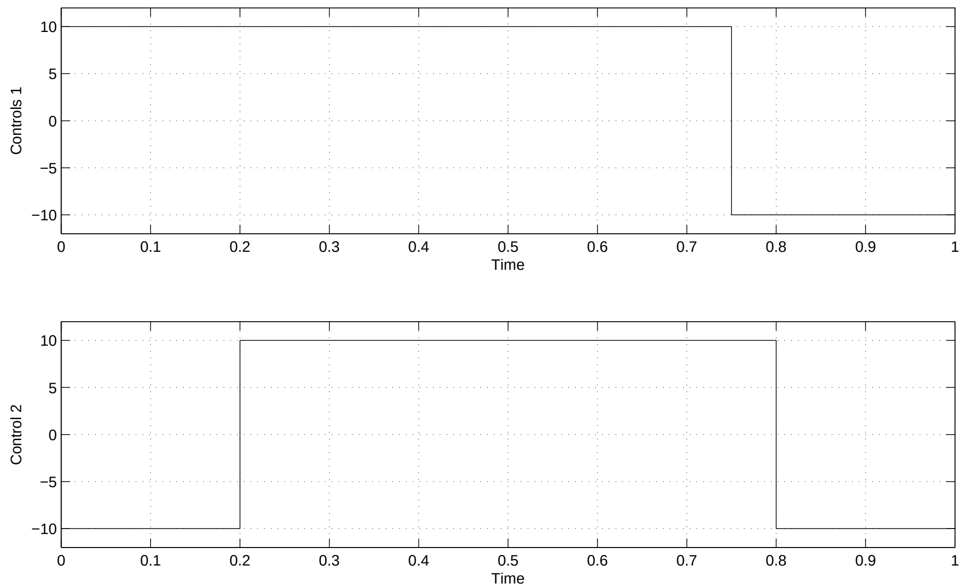

FIGURE 18. The optimal control of Example 3

canonical formulation. Experience with the software is encouraging. In view of the user-friendly nature of the software, it is expected to attract researchers and practitioners to make use of the software to solve the optimal control problems that they encounter in their research or work place.

Future work includes the extension of the software to discrete time optimal control problems, optimal control problems with time delays, and optimal control problems governed by distributed parameter systems. A real challenge is to improve the speed of the software can run fast enough to solve practical optimal control problems online. In this way, it would be possible to obtain optimal feedback control for the optimal control problem via solving a sequence of receding horizon optimal control problem online.

Acknowledgments. This work was partially supported by the Australian Research Council (under grant DP110100083) and the National Natural Science Foundation of China (under grant 61450010). We also wish to thank our colleagues Dr. Lei Wang and Dr. Zhaohua Gong who have helped test run the software on various optimal control problems.

REFERENCES

[1] N.U. Ahmed, Dynamic Systems and Control with Applications, World Scientific, Singapore, 2006.
[2] M. Athans, and P.L. Falb, Optimal Control, McGraw-Hill, 1966.
[3] V. Azhmyakov, Optimal Control of mechanical Systems, Differential Equations and Nonlinear Mechanics, Volume 2007. doi:10.1155/2007/18735
[4] R. Bellman, and R.E. Dreyfus, Dynamic Programming and Modern Control Theory, Orlands, Florida, Academic Press, 1977.

[5] A.E. Jr. Bryson, and Y.C. Ho, Applied Optimal Control, Hemisphere Publishing, DC, 1975.

[6] C. Bskens, NUDOCCCS, FORTRAN-Subroutine NUDOCCCS (Numerical Discretisation method for Optimal Control problems with Constraints in Controls and States), 2010. http://www.swmath.org/software/8606

[7] C. Buskens, and H. Maurer, Nonlinear Programming Methods for Real-Time Control of an Industrial Robot, Journal of Optimization Theory and Applications, Vol. 107, pp. 505-527, 2000.

[8] L. Cesari, Optimization: Theory and Applications, Springer-Verlag, New York, 1983.

[9] Q.Q. Chai, C.H. Yang, K. L. Teo and W.H. Gui, Optimal Control of an Industrial-scale Evaporation Process: Sodium Aluminate Solution, Control Engineering Practice, Vol. 20, pp. 618-628, 2012.

[10] B.D. Craven and S.M.N. Islam, Optimization in Economics and Finance Springer, The Netherlands, 2005.

[11] M. Fikar, M.A. Latifi and Y. Creff, Optimal Changeover Profiles for an Industrial Depropanizer, Chemical Engineering Science, Vol. 54, pp. 2715-2720, 1999.

[12] M.E. Fisher and L.S. Jennings, MATLAB MISER http://www.acad.polyu.edu.hk/ majlee/AMA483-523/OCTmanual.pdf

[13] P. E. Gill, W. Murray, M. A. Saunders and M. H. Wright , User's Guide for NPSOL 5.0: Fortran package for nonlinear programming, 1986. http://web.stanford.edu/group/SOL/npsol.htm

[14] W.E. Gruver and E. Sachs, Algorithmic Methods in Optimal Control, Research Notes in Mathematics, Vol. 47, Pitman (Advance Publishing Program), London, 1981.

[15] C.J. Goh and K.L. Teo, Control Parameterization: a Unified Approach to Optimal Control Problems with General Constraints. Automatica, Vol. 24, pp. pp3-18, 1988.

[16] S. Gonzalez and A. Miele, Sequential Gradient-Restoration Algorithm for optimal Control Problems with General Boundary Conditions, Journal of Optimization Theory and Applications, Vol. 26, pp. 395-425, 1978.

[17] G.R. Duan, D.K. Gu and B. Li, Optimal Control for Final Approach of Rendezvous with Non-cooperative Target, Pacific Journal of Optimization, Vol. 6 (7), pp. 3157-3175, 2010.

[18] P. Howlett, The Optimal Control of a Train, Annals of Operations Research, Vol. 98, pp. 65-87, 2000.

[19] H. Jaddu, Direct Solution of Nonlinear Optimal Control Problems Using Quasilinearization and Chebyshev Polynomials, Journal of the Franklin Institute, Vol. 339, pp. 479-498, 2002.

[20] L. S. Jennings, M. E. Fisher, K. L. Teo and C. J. Goh, MISER3.3 Optimal Control Software Version: Theory and User Manual, the University of Western Australia, 2004.

[21] L.S. Jennings and K.L. Teo, A Computational Algorithm for Functional Inequality Constrained Optimization Problems, Automatica, Vol. 26, pp. 371-375, 1990.

[22] C. Jiang, K. L. Teo, R. Loxton and G. R. Duan, A Neighboring Extremal Solution for an Optimal Switched Impulsive Control Problem, Journal of Industrial and Management Optimization, Vol. 8, pp. 591-609, 2012.

[23] C. Jiang, K. L. Teo and G.R. Duan, A Suboptimal Feedback Control for Nonlinear Time-varying Systems with Continuous Inequality Constraints, Automatica, Vol. 48, pp. 660-665, 2012.

[24] C. Jiang, Q. Lin, C. Yu, K. L. Teo and G. R. Duan, An Exact Penalty Method for Free Terminal Time Optimal Control Problem with Continuous Inequality Constraints, Journal of Optimization Theory and Applications, Vol. 154, pp. 30-53, 2012.

[25] C.Y. Kaya and J.M. Martnez, Euler Discretization and Inexact Restoration for Optimal Control, Journal of Optimization Theory and Applications, Vol. 134, pp. 191-206, 2007

[26] C.Y. Kaya and J.L. Noakes, Leapfrog for Optimal Control, SIAM Journal on Numerical Analysis, in press, 2008.

[27] M.I. Kamien and N.L. Schwartz, Dynamic Optimization - The Calculus of Variations and Optimal Control in Economics and Management, North Holland, 1991.

[28] T.T. Lam, and Y. Bayazitoglu, Application of the Sequential Gradient Restoration Algorithm to Terminal Convective Instability Problems, Journal of Optimization Theory and Applications, Vol. 49, pp. 47-63, 1986.

[29] B. Li, C. Xu, K. L. Teo and J. Chu, Time Optimal Zermelo's Navigation Problem with Moving and Fixed Obstacles, Applied Mathematics and Computation, Vol. 224, pp. 866-875, 2013.

[30] B. Li, C. J. Yu, K. L. Teo and G.R. Duan, An Exact Penalty Function Method for Continuous Inequality Constrained Optimal Control Problem, Journal of Optimization Theory and Applications, Vol. 151, pp. 260-291, 2011.

[31] B. Li, K.L. Teo, C.C. Lim and G.R. Duan, An Optimal PID Controller Design for Nonlinear Constrained Optimal Control Problems, Discrete and Continuous Dynamical Systems Series B, Vol. 16, pp. 1101-1117, 2011.

[32] B. Li, K.L. Teo and G.R. Duan, Optimal Control Computation for Discrete Time Time-delayed Optimal Control Problem with All-time-step Inequality Constraints, International Journal of Innovative Computing, Information and Control, Vol. 6, (3), pp. 521-532, 2010.

[33] B. Li, K.L. Teo, G.H. Zhao and G.R. Duan, An Efficient Computational Approach to a Class of Minimax Optimal Control Problems with Applications, Australian and New Zealand Industrial and Applied Mathematics Journal, Vol. 51 (2), pp. 162-177, 2009.

[34] C.J. Li, K.L Teo, B. Li and G.F. Ma, A Constrained Optimal PID-like Controller Design for Spacecraft Attitude Stabilization, Acta Astronautica, Vol. 74, pp. 131-140, 2011.

[35] C.C. Lim and K.L. Teo, Optimal Insulin Infusion Control to a Mathematical Blood Glucoregulatory Model with Fuzzy Parameters, Cybernetics and Systems, Vol. 22 (1), pp. 1-16, 1991.

[36] Q. Lin, R. Loxton and K.L. Teo, The Control Parameterization for Nonlinear Optimal Control: A Survey, Journal of Industrial and Management Optimization, Vol. 10 (1), pp. 275-309, 2014.

[37] R. Loxton, K. L. Teo, V. Rehbock, and W.K. Ling, Optimal Switching Instants for a Switched-capacitor DC/DC Power Converter, Automatica, Vol. 45, pp. 973-980, 2009.

[38] R. Loxton, K. L. Teo, V. Rehbock, and K. F. C. Yiu, Optimal Control Problems with a Continuous Inequality Constraint on the State and the Control, Automatica, Vol. 45, pp. 2250-2257, 2009.

[39] R. Loxton, K. L. Teo, and V. Rehbock, Computational Method for a Class of Switched System Optimal Control Problems, IEEE Transactions on Automatic Control, Vol. 54 (10), pp. 2455-2460, 2009.

[40] R. Loxton, Q. Lin, V. Rehbock, and K. L. Teo, Control Parameterization for Optimal Control Problems with Continuous Inequality Constraints: New Convergence Results, Numerical Algebra, Control and Optimization, Vol. 2, pp. 571-599, 2012.

[41] R. Loxton, Q. Lin and K.L. Teo, Minimizing Control Variation in Nonlinear Optimal Control, Automatica, Vol. 49, pp. 2652-2664, 2013.

[42] R. Loxton, Q. Lin and K.L. Teo, Switching Time Optimization for Nonlinear Switched systems: Direct Optimization and the Time Scaling Transformation, Pacific Journal of Optimization, Vol. 10 (3), pp. 537-560, 2014.

[43] R. Luus, Iterative Dynamic Programming, Chapman & Hall/CRC, Boca Raton, 2000.

[44] R. Luus and O.N. Okongwu, Towards Practical Optimal Contorl of Batch Reactors, Chemical Engineering Journal, Vol. 75, pp. 1-9, 1999.

[45] R. Martin, and K.L. Teo, Optimal Control of Drug Administration in Cancer Chemotherapy, World Scientific, 1994, 185pp.

[46] MATLAB - The Language of Technical Computing, http://www.mathworks.com/products/matlab/, 2008.

[47] H. Maurer, C. Buskens, and G. Feichtinger, Solution Techniques for Periodic Control Problems: A Case Study in Production Planning, Optimal Control Applications and Methods, Vol. 19, pp. 185 - 203, 1998.

[48] H.H. Mehne, and A.H. Borzabadi, A Numerical Method for Solving Optimal Control Problems Using State Parametrization, Numerical Algorithms, Vol. 42, pp. 165-169, 2006.

[49] A. Miele, and T. Wang, Primal-Dual Properties of Sequential Gradient- Restoration Algorithms for Optimal Control Problems, Part 2, General Problem, Journal of Mathematical Analysis and Applications, Vol. 119, pp. 21-54, 1986.

[50] H. J. Oberle, and B. Sothmann, Numerical Computation of Optimal Feed Rates for a Fed-Batch Fermentation Model, Journal of Optimization Theory and Applications, Vol. 100, pp. 1-13, 1999.

[51] R. Petzold and A.C. Hindmarsh, LSODA, Ordinary Differential Equation Solver for Stiff or Non-Stiff System, 2005.

[52] L.S. Pontryagin, V.G. Boltayanskii, R.V. Gamkrelidze, and E.F. Mischenko. Mathematical Theory of Optimal Processes. CRC Press, 1987.

[53] V. Rehbock, and I. Livk, Optimal Control of a Batch Crystallization Process, Journal of Industrial and Management Optimization, Vol. 3 (3), pp. 331-348, 2007.

[54] Y. Sakawa, and Y. Shindo, Optimal control of container cranes, Automatica, Vol. 18, pp. 257-266, 1982.

[55] K. Schittkowski , NLPQLP: A New Fortran Implementation of a Sequential Quadratic Programming Algorithm for Parallel Computing, 2010.

[56] A. L. Schwartz, RIOTS-A Matlab Toolbox for Solving General Optimal Control Problems, 2008. http://www.accesscom/adam/RIOTS/

[57] Y. Shindo, and Y. Sakawa, Local Convergence of an Algorithm for Solving Optimal Control Problems, Journal of Optimization Theory and Applications, Vol. 46, pp. 265-293, 1985.

[58] W. Sun and Y.X. Yan, Optimization Theory and Methods, Springer, 2006.

[59] K.L. Teo, C.J. Goh, and K.H. Wong, A Unified Computational Approach to Optimal Control Problems, Longman Scientific and Technical, England, 1991.

[60] K.L. Teo, L.S. Jennings, H.W.J. Lee and V. Rehbock, The Control Parameterization Enhancing Transform for Constrained Optimal Control Problems, Journal of Australian Mathematical Society, Series B, Vol. 40, pp. 314-335, 1999.

[61] K.L. Teo, V. Rehbock and L.S. Jennings, A New Computational Algorithm for Functional Inequality Constrained optimization Problems, Automatica, Vol. 29, No. 3, pp. 789-792, 1993.

[62] K.L. Teo, and K.H. Wong, Nonlinearly Constrained Optimal Control Problems, Journal of Australian Mathematical Society, Series B, Vol. 33, pp. 517-530, 1992.

[63] K.L. Teo, C.J. Goh and C.C. Lim, A Computational Method for a Class of Dynamical Optimization Problems in which the Terminal Time is Conditionally Free, IMA - Journal of Mathematical Control and Information, Vol. 6, pp. 81-95, 1989.

[64] K.L. Teo and C.C. Lim, Time Optimal Control Computation with Application to Ship Steering, Journal of Optimization Theory and Applications, Vol. 56 (1), pp. 145-156, 1988.

[65] N.S. Trahair and J.R. Booker, Optimum Elastic Columns, International Journal of Mechanical Sciences, Vol. 12, no. 11, pp. 973-983, 1970.

[66] O. von Stryk, Optimization of Dynamic Systems in Industrial Applications, in H.J. Zimmermann (ed.): Proc. 2nd European Congress on Intelligent Techniques and Soft Computing (EUFIT), Aachen, Germany, pp. 347-351, 1994.

[67] C.Z. Wu, and K.L. Teo, Global Impulsive Optimal Control Computation, Journal of Industrial and Management Optimization, Vol. 2, pp. 435-450, 2006.

[68] J.L. Zhou and A. Tits, User's guide for FFSQP version 3.7 : A Fortran Code for Solving Optimization Programs, Possibly Minimax,with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates, (1997), Institute for Systems Research, University of Maryland, Technical Report SRC-TR-92-107r5, College Park, MD 20742.

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: fyang@uestc.edu.cn
E-mail address: k.l.teo@curtin.edu.au
E-mail address: r.loxton@curtin.edu.au
E-mail address: v.rehbock@curtin.edu.au
E-mail address: bin.li@curtin.edu.au
E-mail address: yuchangjun@126.com
E-mail address: les.jennings@uwa.edu.au