Numerical solutions for nonlinear constrained optimal control problems in discrete time

van Garderen, M.J.M.

Award date:
1995

EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
Measurement and Control Section

NUMERICAL SOLUTIONS FOR NONLINEAR CONSTRAINED OPTIMAL CONTROL PROBLEMS IN DISCRETE TIME

by M.J.M. van Garderen

M. Sc. Thesis
carried out from February 1995 to October 1995
commissioned by prof. P.P.J. van den Bosch
under supervision of Ir. J. Mazák
date October 16, 1995.

Abstract

In this thesis report a method to solve general constrained discrete-time nonlinear optimal control problems is discussed.

The method is based on a hybrid of the maximum-principle and mathematical programming techniques. The idea is to convert the problem into a nonlinear programming problem that can subsequently be solved by existing standard optimization software. Here, the recent optimization package CFSQP[1] is used (C-code For Sequential Quadratic Programming).

The computational effort to solve constrained optimal control problems can greatly be reduced by the constraint transcription method and the parametrization concept. The constraint transcription method allows problems with multiple constraints to be uniformly tackled. It is possible to transform all-time-step state constraints into one single nonlinear state constraint. The parametrization concept is used to decrease the number of decision variables of the programming problem. It is recommended to start an optimization using relatively few control parameters.

The constraint transcription method and the parametrization concept have successfully been implemented in CFSQP. However, the transformation from optimal control problem to mathematical programming problem is not a trivial issue. Although the method is applicable for a wide range of problems, each individual problem still needs a specific approach. This is the reason a user-friendly environment is difficult to develop.

The method works fine for relatively simple problems as well as for more difficult ones. In every case, the calculated solution satisfies all the present hard state constraints.

The 2-link inverted pendulum problem is well known in the nonlinear control theory. A successful attempt is made to solve this complex problem. It appears even possible to find feasible solutions for the more difficult problem with three links.

When the terminal time is not fixed, it becomes a system parameter, which has to be optimized at the same time with the cost functional. An example of such a terminal time free optimal control problem is solved using a simple but effective heuristic.

Remark It must clearly be stated that the results of the optimization calculations must be considered as simple feed-forward controllers from which the corresponding state trajectories can possibly serve as reference signals in more sophisticated state-tracking controllers.

[1]Acknowledgement due to *C.Lawrence, J.L.Zhou and A.L.Tits.* Electrical Engineering Department and Institute for System Research, University of Maryland

Contents

Chapter 1

Introduction

Nonlinear constrained discrete-time optimal control problems lately arise more often in multistage control and scheduling problems. A typical example is the use of neural nets in nonlinear control theory. In contrast with linear control problems, nonlinear problems cannot be solved analytically. Therefore, efficient numerical methods are needed.

Some of the early methods were based on dynamic programming techniques and suffered from limitations due to memory and computational requirements. These methods have been extended and improved in various ways. There are still, however, limitations on their ability to handle completely general constraints. One of the early approaches involved converting the problem into a mathematical programming problem (MPP) in which all the states and controls were decision variables and the dynamical equations formed part of the constrained set. Generally, this leads to problems with a large number of variables and nonlinear constraints. For example, an optimal control problem with 4 states, 1 control variable, 100 time steps and 1 continuous state constraint would convert to a 500 variable MPP with 400 nonlinear constraints and, if required, about 100 simple bounds on the variables. This approach was greatly improved by combining it with the Maximum Principle, resulting in a MPP in which only the controls were decision variables and the state variables were obtained from the dynamical equations.

The different methods can be roughly partitioned into four main classes:

1. methods based on the maximum principle [12].
2. methods using mathematical programming techniques [13].
3. methods based on a hybrid of maximum-principle and mathematical programming techniques [11].
4. methods using the dynamic-programming technique [15].

The first class of methods is based on the maximum principle. It can only be used to obtain solutions for relatively simple problems in practise.

The classical mathematical-programming approach (2) considers all the state and control variables as decision variables. Consequently, the difference equations governing the dynamics of the system are treated as algebraic constraints. There are some disadvantages associated with this approach. When the number of time steps or the dimension of the difference equations increases, the corresponding number of decision variables and algebraic constraints for the resulting mathematical programming problem becomes undesirably large. This is further aggravated by the presence of hard constraints, such as all-time step constraints on

the state and control variables. These are nonlinear inequality constraints that are required to be satisfied for all the time steps M. Each of them gives rise to an additional M nonlinear algebraic constraints for the corresponding MPP. Thus, a small optimal control problem can end up with a very large set of decision variables and many constraints.

In the third class of methods, only the control variables are regarded as decision variables. The state variables are obtained by solving the governing difference equations. Thus, the cost and hard constraint functionals depend on the control variables only implicitly through these difference equations. Nevertheless, their gradients are computable, although in a rather roundabout way. So, the problems can also be viewed as MPP's with much fewer decision variables.

In the absence of hard constraints, differential dynamic programming (4) is an excellent method for solving optimal control problems. This method is based on a combination of the dynamic and mathematical programming technique.

The method used in this thesis project belongs to the third class of methods. It is based on an algorithm that converts the original optimal control problem into a MPP in which derivatives are computed from the Hamiltonian in the Maximum Principle [2]. The resulting MPP is solved by the recent optimization software package CFSQP [1].

The most typical feature of the algorithm is the way in which constraints of a general nature are converted into a canonical form using a simple constraint transcription method that was first introduced for continuous-time problems [5]. This article also includes the idea of control parametrization where individual controls over a number of time steps are represented by a single parameter, resulting in a great reduction of decision variables in problems with a large number of time steps.

The global construction of this thesis report is as follows:

In chapter 2, the computational procedure for solving a general class of constrained discrete-time optimal control problems will be presented. The two main features of the algorithm, the constraint transcription method and the parametrization concept will extensively be handled.

A short description of the recent optimization package CFSQP is given in chapter 3. A profound analysis is omitted.

The algorithm described in chapter 2 is tested on the basis of four general optimal control problems in chapter 4.

In chapter 5, the 2-link inverted pendulum problem is tackled. This difficult nonlinear problem is first extensively discussed and subsequently the optimization results will be presented.

The problem of terminal time-free optimization is discussed in chapter 6. A simple but effective heuristic to find the time-optimal solution will be presented.

Conclusions and recommendations are given in chapter 7.

The appendix contains a summary of the source code of the 2-link inverted pendulum problem.

Chapter 2

The discrete-time optimal control problem

This chapter presents a computational procedure for solving a general class of constrained discrete-time optimal control problems. The applied approach is to convert the problem into a nonlinear programming problem that can be solved by existing nonlinear optimization methods. The two main features of the proposed algorithm, the discrete version of the control parametrization concept and the constraint transcription method are extensively handled.

2.1 Problem Definition

For p a positive integer, define

$$I_p = \{0, 1, \ldots, p\}, \qquad I_p^+ = \{1, 2, \ldots, p\}. \tag{2.1}$$

A form of the discrete-time optimal control problem is defined by

$$\min_u G_0(u) = \Theta_0[x(M)] + \sum_{t=0}^{M-1} g_0[t, x(t), u(t)] \tag{2.2}$$

subject to the system dynamics

$$x(t+1) = f[t, x(t), u(t)], \quad t = 0, 1, \ldots, M-1 \tag{2.3}$$

where M is a fixed positive integer, t is the discrete time, and

$$\begin{aligned} x &= [x_1(t), x_2(t), \ldots, x_n(t)]^T \in I\!R^n, \quad \text{for } t \in I_M \\ u &= [u_1(t), u_2(t), \ldots, u_m(t)]^T \in I\!R^m, \quad \text{for } t \in I_{M-1} \end{aligned} \tag{2.4}$$

are respectively, the state and control vectors. The initial condition for the difference equation (2.3) is supposed to be given by

$$x(0) = x_0 \in I\!R^n. \tag{2.5}$$

The control variables are subject to simple bounds:

$$u_i^L \leq u_i \leq u_i^U, \quad i = 1, \ldots, m. \tag{2.6}$$

The above described problem is subject to a variety of additional constraints:

1. *Terminal state equality constraint.*

$$C[x(M)] = 0. \tag{2.7}$$

2. *Terminal state inequality constraint.*

$$C[x(M)] \geq 0. \tag{2.8}$$

3. *Interior point state equality constraint.*

$$C[x(\xi)] = 0,\ \xi \in I_{M-1}^{+}. \tag{2.9}$$

4. *Interior point state inequality constraint.*

$$C[x(\xi)] \geq 0,\ \xi \in I_{M-1}^{+}. \tag{2.10}$$

5. *Continuous state and control equality constraint*

$$C[t, x(t), u(t)] = 0,\ \forall_t \in I_\xi,\ \xi \in I_{M-1}^{+}. \tag{2.11}$$

6. *Continuous state and control inequality constraint*

$$C[t, x(t), u(t)] \geq 0,\ \forall_t \in I_\xi,\ \xi \in I_{M-1}^{+}. \tag{2.12}$$

The following conditions are assumed throughout:

- For each $t \in I_{M-1}$, $f[t, x(t), u(t)]$ is continuously differentiable with respect to each component of x and u.
- Θ_0 is continuously differentiable in $\mathbb{R}^n$.
- for each $t = 0, 1, \ldots, M-1$, $g_0[t, x(t), u(t)]$ is continuously differentiable in $\mathbb{R}^n \times \mathbb{R}^m$.
- for each $k = 1, \ldots, N$, and for each $t \in I_M$, C_k is continuously differentiable in $\mathbb{R}^n \times \mathbb{R}^m$.

2.2 Constraint transcription

In the classical control literature, constraints of a different type are often treated differently. In this formulation, different constraints are transformed into a standard canonical form

$$\begin{aligned} G_k(u) &= \Theta_k[x(\tau_k)] + \sum_{t=0}^{\tau_k - 1} g_k[t, x(t), u(t)] = 0 \quad k = 1, 2, \ldots, N_e \\ G_k(u) &= \Theta_k[x(\tau_k)] + \sum_{t=0}^{\tau_k - 1} g_k[t, x(t), u(t)] \geq 0 \quad k = N_e + 1, \ldots, N \end{aligned} \tag{2.13}$$

where N_e is the number of equality constraints, N is the total number of constraints and $\tau_k \in I_M^{+}$ is referred to as the *characteristic* time of the kth constraint. This constraint transcription is the discrete analogue of that introduced for continuous-time systems [5]. Note that the constraint canonical form is the same as that for the cost functional (2.2), where $\tau_k = M$. This is intentional, as the formulation allows problems with multiple constraints to be uniformly tackled. The constraints are converted to the canonical form by defining Θ, g and τ for each constraint and saying whether G is an equality or inequality constraint.

The transcriptions for each of the different types of constraints are as follows:

1. *Terminal state equality constraint.* $C[x(M)] = 0$.
   Set $\Theta = C$, $g = 0$, $\tau = M$, and put $G = 0$.

2. *Terminal state inequality constraint.* $C[x(M)] \geq 0$.
   Set $\Theta = C$, $g = 0$, $\tau = M$, and put $G \geq 0$.

3. *Interior point state equality constraint.* $C[x(\xi)] = 0, \quad \xi \in I_{M-1}^{+}$.
   Set $\Theta = C$, $g = 0$, $\tau = \xi$, and put $G = 0$.

4. *Interior point state inequality constraint.* $C[x(\xi)] \geq 0, \quad \xi \in I_{M-1}^{+}$.
   Set $\Theta = C$, $g = 0$, $\tau = \xi$, and put $G \geq 0$.

5. *Continuous state and control equality constraint* $C[x(t), u(t)] = 0$, $\forall_t \in I_\xi$, $\xi \in I_{M-1}^{+}$.
   Set $\Theta = 0$, $g = C^2$, $\tau = \xi$, and put $G = 0$.

6. *Continuous state and control inequality constraint* $C[x(t), u(t)] \geq 0$, $\forall_t \in I_\xi$, $\xi \in I_{M-1}^{+}$.
   Set $\Theta = 0$, $g = \min\{C, 0\}$, $\tau = \xi$, and put $G = 0$.

Note that constraint of type 6 was originally of inequality type but becomes of equality type in its canonical form. (2.12) is equivalent to

$$G = \sum_{t=0}^{\tau-1} \min\{C[t, x(t), u(t)], 0\} = 0, \tag{2.14}$$

which is, however, nonsmooth. Using the smoothing technique of Jennings and Teo [8], we approximate the nonsmooth function $g = \min\{C, 0\}$ by the continuously differentiable[1] $g_\epsilon(C)$, which is defined by

$$g_\epsilon(C) = \begin{cases} C & \text{if } C < -\epsilon, \\ -\frac{[C-\epsilon]^2}{4\epsilon} & \text{if } -\epsilon \leq C \leq \epsilon, \\ 0 & \text{if } C > \epsilon. \end{cases} \tag{2.15}$$

Define U to be the class of all *admissible* controls:

$$U = \{v = [v_1, \ldots, v_m]^T \in I\!R^m : u_i^L \leq v_i \leq u_i^U, \quad i = 1, \ldots, m\} \tag{2.16}$$

Let u denote a control sequence $\{u(t) : t = 0, 1, \ldots, M-1\}$ in U. If $u \in U$ satisfies the constraints (2.8) to (2.12), then it is called a *feasible* control.

Define

$$G_\epsilon(u) = \sum_{t=0}^{M-1} g_\epsilon(C) \tag{2.17}$$

and let

$$G_\epsilon(u) + \delta \geq 0, \tag{2.18}$$

where $\delta = \delta(\epsilon)$. In [2] it is shown that if δ is chosen as $\epsilon/4$ then, for any $\epsilon > 0$, any feasible solution of (2.18) is also a feasible solution of the original continuous state inequality constraint (2.12). A typical value for ϵ is 10^{-3}.

[1] of the 1st order

The constrained optimal control problem is now in the form

$$\min_{u} G_0(u) \tag{2.19}$$

subject to N_e canonical equality constraints:

$$G_k(u) = 0 \qquad k = 1, \ldots, N_e, \tag{2.20}$$

and $N - N_e$ canonical inequality constraints

$$G_k(u) \geq 0 \qquad k = N_e + 1, \ldots, N, \tag{2.21}$$

where each G_k is defined on a set I_{τ_k}, $\tau_k \in I_M^+$, and the decision variables are subject to simple bounds

$$u_i^L \leq u_i \leq u_i^U, \quad i = 1, \ldots, m. \tag{2.22}$$

2.3 Control Parametrization

In this section, the discrete version of the control parametrization concept is presented [2]. The continuous version was introduced by Goh and Teo in 1988 [5]. The idea of control parametrization is to compute a suboptimal solution to a large problem by using relatively few decision variables. The quality of the suboptimal solution is then improved by successive refinement of the parametrization. In this way, control problems with a large number of time steps, can be solved in less amount of time and computational effort.

The problem described in the previous section is converted to an optimization problem over $\mathbb{R}^{n_p}$, where n_p represents the total number of parameters constructed from the control variables. Let the set I_{M-1} be partitioned for the ith control function u_i, $i = 1, \ldots, m$, into k_i disjoint subsets of the form

$$P_j^i = \{t_{j-1}^i, t_{j-1}^i + 1, t_{j-1}^i + 2, \ldots, t_j^i - 1\} \tag{2.23}$$

where $t_j^i \in I_M$, satisfying

$$0 = t_0^i < t_1^i <, \cdots, t_{k_i}^i = M \tag{2.24}$$

are known as knots. Note that for each i

$$I_{M-1} = \bigcup_{j=1}^{k_i} P_j^i. \tag{2.25}$$

The above is illustrated by the following example for u_i:

$$\overbrace{0 \quad 1 \quad 2}^{P_1^i} \quad \overbrace{3 \quad 4 \quad 5 \quad 6}^{P_2^i} \quad \overbrace{7 \quad 8 \quad 9}^{P_3^i} \quad \ldots\ldots \quad M. \tag{2.26}$$
$$t_0^i \qquad\qquad t_1^i \qquad\qquad t_2^i \qquad\qquad\qquad t_{k_i}^i$$

Note that an equal-partition parametrization is not a strict requirement but rather a computational convenience. At times for which it is known *a priori* that the control changes rapidly during certain intervals and changes relatively little during others, it will be more effective to use unequal parametrization.

For each set P_j^i we associate a parameter σ_j^i so that

$$u_i(t) = \sigma_j^i, \qquad \forall t \in P_j^i, \tag{2.27}$$

that is, $u(t)$ is constant on each set P_j^i and $\sigma_i = [\sigma_1^i, \ldots, \sigma_{k_i}^i]^T$ is then a vector of parameters describing u_i.

If θ denotes the vector of all control parameters

$$\theta = [\sigma_1^1, \ldots, \sigma_{k_1}^1, \sigma_1^2, \ldots, \sigma_{k_2}^2, \ldots, \sigma_1^m, \ldots, \sigma_{k_m}^m]^T \tag{2.28}$$

then the combined optimal control problem becomes a MPP in the control vector $\theta \in \mathbb{R}^{n_p}$, where $n_p = \sum_{i=1}^m k_i$. After the control parametrization, the cost functional as well as the constraint functional can be regarded as implicit functions of the parameter vector θ. More precisely, the corresponding approximate problem may be written as

$$\min_{\theta} G_0(\theta) \tag{2.29}$$

subject to the constraints

$$\begin{aligned} G_k(\theta) &= 0, \quad k = 1, \ldots, N_e, \\ G_k(\theta) &\geq 0, \quad k = N_e + 1, \ldots, N \end{aligned}$$

with the control parameters σ being bounded by a set of upper and lower bounds:

$$u_i^L \leq \sigma_j^i \leq u_i^U, \quad j = 1, \ldots, k_i, \; i = 1, \ldots, m. \tag{2.30}$$

This approximate problem can be regarded as a standard nonlinear constrained mathematical optimization problem which can be solved by a sequential quadratic programming technique. This technique requires the analytical gradients of the cost functional G_0 as well as the constraint functionals G_k, $k = 1, 2, \ldots, N$. However, because of the nonexplicit dependency of G_k on u, the computation of the gradients of each canonical function $G_k, k = 0, 1, \ldots, N$ has to be done in a somewhat roundabout way. Notice that only the controls are decision variables and the state variables are obtained from the dynamical equations (2.3).

2.4 Derivation of the Gradient Formulae

In this section, the gradients for the canonical cost functional and the canonical constraints are derived.

The discrete-time optimal control problem is concerned with the determination of a control sequence for the system

$$x(t+1) = f[t, x(t), u(t)], \qquad t = 0, 1, \ldots, M-1 \tag{2.31}$$

such that the cost functional

$$G_0(u) = \Theta_0[x(M)] + \sum_{t=0}^{M-1} g_0[t, x(t), u(t)] \tag{2.32}$$

is minimized subject to the following constraints (in canonical form):

$$G_k(u) = \Theta_k[x(\tau_k)] + \sum_{t=0}^{\tau_k - 1} G_k[t, x(t), u(t)] \tag{2.33}$$

where

$$\begin{aligned} G_k(u) &= 0 \quad k = 1, 2, \ldots, N_e \\ G_k(u) &\geq 0 \quad k = N_e + 1, \ldots, N. \end{aligned}$$

The functions Θ_k, G_k, $k = 0, 1, \ldots, N$, are assumed to be continuously differentiable with respect to each of their arguments. Note that the canonical constraints are expressed in a similar form to the cost functional. This feature is to allow gradients of the cost functional as well as the gradients of the constraints functionals to be computed in a unified way.

Let the control parameter vector θ be perturbed by $\epsilon\rho$, where $\epsilon > 0$ is a small real number and ρ is an arbitrary fixed perturbation of θ. Then we have

$$\theta(\epsilon) = \theta + \epsilon\rho. \tag{2.34}$$

Consequently, the system state will be perturbed as well as the cost functional and the constraint functionals.
Define

$$x(t, \epsilon) = x[t|\theta(\epsilon)], \quad t = 1, 2, \ldots, M. \tag{2.35}$$

Then

$$x(t+1, \epsilon) = f[t+1, x(t, \epsilon), \theta(\epsilon)]. \tag{2.36}$$

The variation of the state for $t = 0, 1, \ldots, M-1$ is

$$\begin{aligned} \Delta x(t+1) &= \left.\frac{d\, x(t+1, \epsilon)}{d\, \epsilon}\right|_{\epsilon=0} \\ &= \frac{\partial f[t, x(t), \theta]}{\partial x(t)} \Delta x(t) + \frac{\partial f[t, x(t), \theta]}{\partial \theta} \rho \end{aligned} \tag{2.37}$$

with

$$\Delta x(0) = 0. \tag{2.38}$$

For the ith constraint ($i = 0$ denotes the cost functional and $\tau_0 = M$), we have

$$\begin{aligned} \nabla G_k \rho &= \left.\frac{d\, G_k(u(t, \epsilon))}{d\, \epsilon}\right|_{\epsilon=0} = \frac{\partial \Theta_k[x(\tau_k)]}{\partial x(\tau_k)} \Delta x(\tau_k) \\ &\quad + \sum_{t=0}^{\tau_k - 1} \left[\frac{\partial G_k[t, x(t), \theta]}{\partial x(t)} \Delta x(t) + \frac{\partial G_k[t, x(t), \theta]}{\partial \theta} \rho \right]. \end{aligned} \tag{2.39}$$

Define the Hamiltonian sequence

$$H_k[t, x(t), \theta, \lambda_k(t+1)] = G_k[t, x(t), \theta] + \lambda_k^T(t+1) f[t, x(t), \theta]. \tag{2.40}$$

where $\lambda_k(t) \in I\!R^n, k = 0, 1, \ldots, N$, $t = \tau_k, \tau_k - 1, \ldots, 0$, are referred to as the costate sequence for the kth canonical constraint. (2.39) becomes

$$\begin{aligned} \nabla G_k[\theta] \rho &= \frac{\partial \Theta_k[x(\tau_k)]}{\partial x(\tau_k)} \Delta x(\tau_k) + \sum_{t=0}^{\tau_k - 1} \left\{ \frac{H_i[t, x(t), \theta, \lambda_k(t+1)]}{\partial x(t)} \Delta x(t) \right. \\ &\quad - \lambda_k^T(t+1) \frac{\partial f[t, x(t), \theta]}{\partial x(t)} \Delta x(t) + \frac{\partial H_i[t, x(t), \theta, \lambda_k(t+1)]}{\partial \theta} \rho \\ &\quad \left. - \lambda_k^T(t+1) \frac{\partial f[t, x(t), \theta]}{\partial \theta} \rho \right\} \end{aligned} \tag{2.41}$$

Let the costate $\lambda_k(t)$ be determined by the following system of difference equations:

$$\begin{aligned} \lambda_k^T(t) &= \frac{\partial H_i[t, x(t), \theta, \lambda_k(t+1)]}{\partial x(t)} \\ &= \frac{\partial G_k[t, x(t), \theta]}{\partial x(t)} + \lambda_k^T(t+1) \frac{\partial f[t, x(t), \theta]}{\partial x(t)} \end{aligned} \tag{2.42}$$

and

$$\lambda_k(\tau_k) = \left(\frac{\partial \Theta_k[x(\tau_k)]}{\partial x(\tau_k)}\right)^T \tag{2.43}$$

for $k = 0, 1, \ldots, N$. Hence on noting (2.37), equation (2.41) becomes

$$\nabla G_k[\theta]\rho = \sum_{t=0}^{\tau_k - 1} \frac{\partial H_k[t, x(t), \theta, \lambda_k(t+1)]}{\partial \theta}\rho. \tag{2.44}$$

Since ρ is arbitrary, the components of the gradients with respect to the control parameters are:

$$\begin{aligned} \frac{\partial G_k}{\partial \sigma_j^i} &= \sum_{t=0}^{\tau_k - 1} \frac{\partial H_k}{\partial u_i}\frac{\partial u_i}{\partial \sigma_j^i} \\ &= \begin{cases} \sum_{t=t_{j-1}^i}^{t_j^i - 1} \frac{\partial H_k}{\partial u_i}, & \text{if } t_j^i \le \tau_k, \\ \sum_{t=t_{j-1}^i}^{\tau_k - 1} \frac{\partial H_k}{\partial u_i}, & \text{if } t_{j-1}^i \le \tau_k, \le t_j^i, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.45}$$

Note that, if no parametrization is applied, the components of the gradients with respect to the individual control variables $u(t)$, become

$$\frac{\partial G_k}{\partial u_i(t)} = \frac{\partial H_k(t, x(t), u(t), \lambda_k(t+1)}{\partial u_i(t)} \tag{2.46}$$

The gradient algorithm can be summarized as follows:

1. Given θ and G_k, solve the system of the state difference equations forward from $t = 0, 1, \ldots, \tau_k - 1$.

2. Solve the system of the costate difference equations $\lambda_k(t+1)$ backward from $t = \tau_k - 1, \tau_k - 2, \ldots, 0$ for $i = 0, 1, \ldots, N$

3. Compute the gradient of G_k, $k = 0, 1, \ldots, N$

Note that the gradient algorithm is especially suitable for control problems, whose dynamics are described by *explicit* discretization schemes.

Chapter 3

CFSQP: A C-Code for Solving Constrained Nonlinear Optimization Problems

3.1 Introduction

In the previous chapter a procedure was presented for converting a discrete time optimal control problem into a constrained nonlinear mathematical programming problem. This problem could then be solved by the recent optimization software package CFSQP. This thesis report is not the right place to give an extensive description of all the complex procedures used by CFSQP, and thus only a brief description will be given here to give the reader at least some insight. It must clearly be said that CFSQP has only been *applied* and not thoroughly analysed.

3.2 The programming problem

CFSQP is a set of C functions for the minimization of the maximum of a set of smooth objective functions (possible a single one) subject to (non)linear (in)equality constraints and simple bounds on the variables. In mathematical terms, CFSQP tackles optimization problems of the form

$$\text{minimize} \max_{i \in I^f} \{f_i(x)\} \qquad \text{s.t.} \quad x \in X$$

where $I^f = \{1, \ldots, n_f\}$ and X is the set of points $x \in I\!R^n$ satisfying

$$\begin{aligned}
&bl \leq x \leq bu \\
&g_j(x) \leq 0, \qquad j = 1, \ldots, n_i \\
&g_j(x) \equiv < c_{j-n_i}, x > -d_{j-n_i} \leq 0, \quad j = n_i + 1, \ldots, t_i \\
&h_j(x) = 0, \qquad j = 1, \ldots, n_e \\
&h_j(x) \equiv < a_{j-n_e}, x > -b_{j-n_e} = 0, \quad j = n_e + 1, \ldots, t_e
\end{aligned}$$

with $bl \in I\!R^n$; $bu \in I\!R^n$; $f_i : I\!R^n \to I\!R, i = 1, \ldots, n_f$ smooth; $g_j : I\!R^n \to I\!R, j = 1, \ldots, n_i$ nonlinear and smooth; $c_j \in I\!R^n, d_j \in I\!R^n, j = 1, \ldots, t_i - n_i; h_j : I\!R^n \to I\!R, j = 1, \ldots, n_e$ nonlinear and smooth; $a_j \in I\!R^n, b_j \in I\!R^n, j = 1, \ldots, t_e - n_e$..

The user has to provide an initial guess x_0 for the decision variables. If this initial guess does not satisfy all the present constraints, CFSQP himself generates a point satisfying these

constraints. After finding a feasible point, the minimization of the cost functional can start. The user has the option of either requiring that this cost functional decreases at each iteration or requiring a decrease within at most four iterations.

Besides the cost functional, the user has to provide functions that define the constraints and may either provide functions to compute the gradients. It is also possible to let CFSQP himself estimate the gradients by forward finite differences. However, in the case of an optimal control problem, it is not recommended to use this method, as this could lead to convergence problems [3].

Note that the designers of CFSQP speak of a feasible *point*, rather than a feasible control sequence. This is because the software package has not specifically been made for solving optimal control problems.

3.2.1 Relation between variables optimal control problem and parameters optimization problem

The relation between the parameters of the optimization problem and the variables in a general optimal control problem is made more clear on the basis of the following remarks:

- I^f is the set of objective functions and this set has in the case of a general optimal control problem just one element: namely the cost functional. So $I^f = \{G_0\}$ and $n_f = 1$.
- The consequence of this set having only one element is that the problem of minimization of the maximum of a set of objective functions is reduced to a problem of minimizing just one single cost functional.
- The parameter n stands for the number of variables in the programming problem. Thus, in the case of an optimal control problem, this n denotes the number of decision variables, which is equal to $m \times k_i$ or $m \times M$, if no parametrization is applied.
- the vectors bl and bu can be regarded as the upper and lower bounds on the control (decision) variables.
- the inequalities $g_j(x) \leq 0\ j = 1, \ldots, n_j$ and the equations $h_j(x) = 0\ j = 1, \ldots, n_e$ can respectively be considered as the nonlinear inequality constraints and nonlinear equality constraints of the optimal control problem. the inequalities $g_j(x) \leq 0\ j = n_i + 1, \ldots, t_i$ and the equations $h_j(x) = 0\ j = n_e + 1, \ldots, t_e$ can respectively be considered as the linear inequality constraints and linear equality constraints.

In the Appendix, a part of the source code for the 2-link inverted pendulum problem (see Chapter 5) can be found. For a complete description of the algorithms used by CFSQP I refer to the user's Guide [1].

Chapter 4

Examples of Optimal Control Problems

In this chapter, four general optimal control problems, including a variety of different types of constraints, will be solved using the algorithm described in Chapter 2. For problem 1, the transformation from constrained optimal control problem to mathematical programming problem will extensively be described.

4.1 Problem 1: A basic optimal control problem

We consider the following regular problem:

$$\begin{aligned} \min_u J(u) &= \int_0^1 x(t)^2 + u(t)^2 \, \mathrm{d}t \\ \dot{x}(t) &= u(t) \\ x(0) &= 1. \end{aligned} \tag{4.1}$$

In the absence of constraints, the analytic solution is (without proof) given by

$$\begin{aligned} u^*(t) &= -\frac{2e}{1+e^2}\sinh(1-t) \\ x^*(t) &= \frac{2e}{1+e^2}\cosh(1-t) \\ J^* &= \frac{e^4-1}{(1+e^2)^2} = 0.76159. \end{aligned} \tag{4.2}$$

The system dynamics (4.1) are discretized using the Euler scheme, with time step h = 0.01:

$$x(t+1) = x(t) + \mathrm{h}u(t) \tag{4.3}$$

for $t = 0, 1, \ldots, M-1$, where $M = 100$. The continuous cost functional J is converted to a discrete version G_0, where

$$G_0 = \sum_{t=0}^{M-1} x(t)^2 + u(t)^2 \tag{4.4}$$

An comparison with the optimal continuous cost functional J^* can be made by using the trapezoidal approximation R of J, where

$$R = \frac{\mathrm{h}}{2}[x(0)^2 + u(0)^2 + x(M)^2] \; + \; \mathrm{h}\sum_{t=1}^{M-1}[x(t)^2 + u(t)^2] \; \approx J. \tag{4.5}$$

Besides the unconstrained case, the following independent constrained problems are solved:

1. terminal state constraint: $x(M) = 1$.
2. interior point constraint: $x(\tau_2) = a$, where $\tau_2 = 50$ and $a = 0.5$.
3. linear continuous state constraint: $x(t) >= b$, where $t = 1, \ldots, M-1$ and $b = 0.75$.

If we use the constraint transcription method we have to solve 3 optimization problems, each involving 1 cost functional (G_0) and 1 canonical constraint functional. These canonical constraint functionals have the following (unparametrized) form:

1. $G_1(u) = (x[M] - 1) = 0$
2. $G_2(u) = (x[\tau_2] - a) = 0$
3. $G_3(u) = (\sum_{t=0}^{M-1} g_3) = 0 \qquad$ where $\quad g_3 = \min\{x(t) - b, 0\}$

Note that, neither one of these functionals contains a control variable.

Before we can proceed with the calculation of the different gradient formulae, constraint 3 has to be rewritten into a differentiable form. Using the smoothing technique described in the previous chapter we get:

$$G_{3,\epsilon}(u) + \epsilon/4 \geq 0 \tag{4.6}$$

where

$$G_{3,\epsilon}(u) = \sum_{t=0}^{M-1} g_{3,\epsilon} \tag{4.7}$$

and

$$g_{3,\epsilon} = \begin{cases} g_3 & \text{if } g_3 < -\epsilon, \\ -([g_3 - \epsilon]^2)/4\epsilon & \text{if } -\epsilon \leq g_3 \leq \epsilon, \\ 0 & \text{if } g_3 > \epsilon. \end{cases} \tag{4.8}$$

The (unparametrized) Hamiltonian sequences $H_i[t, x(t), u(t), \lambda_i(t+1)]$ for $i = 0, \ldots, 3$ result from (2.40):

$$\begin{array}{rcccl} H_0 & = & g_0 & + & \lambda_0(t+1)[x(t) + \mathrm{h}u(t)] \\ H_1 & = & & & \lambda_1(t+1)[x(t) + \mathrm{h}u(t)] \\ H_2 & = & & & \lambda_2(t+1)[x(t) + \mathrm{h}u(t)] \\ H_3 & = & g_{3,\epsilon} & + & \lambda_3(t+1)[x(t) + \mathrm{h}u(t)]. \end{array} \tag{4.9}$$

where $g_0 = x(t)^2 + u(t)^2$. The costates $\lambda_i[t, x(t), u(t)]$ are defined by (2.42):

$$\begin{array}{rcccl} \lambda_0(t) & = & 2\mathrm{h}x(t) & + & \lambda_0(t+1) \cdot 1 \\ \lambda_1(t) & = & & & \lambda_1(t+1) \cdot 1 \\ \lambda_2(t) & = & & & \lambda_2(t+1) \cdot 1 \\ \lambda_3(t) & = & \partial g_{3,\epsilon}/\partial x(t) & + & \lambda_3(t+1) \cdot 1 \end{array} \tag{4.10}$$

where

$$\partial g_{3,\epsilon}/\partial x(t) = \begin{cases} 1 & \text{if } g_3 < -\epsilon, \\ -(2x(t) - 2b - 2\epsilon)/4\epsilon & \text{if } -\epsilon \leq g_3 \leq \epsilon, \\ 0 & \text{if } g_3 > \epsilon. \end{cases}$$

and $\lambda_i(\tau_i)$, $i = 0, \ldots, 4$ result from (2.43):

$$\begin{array}{rcccl} \lambda_0(M) & = & \partial\Theta_0[x(M)]/\partial x(M) & = & 0 \\ \lambda_1(M) & = & \partial\Theta_1[x(M)]/\partial x(M) & = & 1 \\ \lambda_2(\tau_2) & = & \partial\Theta_2[x(\tau_2)]/\partial x(\tau_2) & = & 1 \\ \lambda_3(M) & = & \partial\Theta_3[x(M)]/\partial x(M) & = & 0 \end{array} \tag{4.11}$$

The gradient formulae are given by (2.45):

$$\begin{array}{rcrcl} \nabla_u G_0[u(t)] & = & 2\mathrm{h}u(t) & + & \lambda_0(t+1)\cdot \mathrm{h} \\ \nabla_u G_1[u(t)] & = & & & \lambda_1(t+1)\cdot \mathrm{h} \\ \nabla_u G_2[u(t)] & = & & & \lambda_2(t+1)\cdot \mathrm{h} \\ \nabla_u G_3[u(t)] & = & \partial g_{3,\epsilon}/\partial u(t) & + & \lambda_3(t+1)\cdot \mathrm{h} \end{array} \tag{4.12}$$

where

$$\partial g_{3,\epsilon}/\partial u(t) = 0.$$

With (4.12), all the necessary equations are now derived. They form a discrete constrained nonlinear optimization problem which can be solved by CFSQP. Figure (4.1) shows the cal-

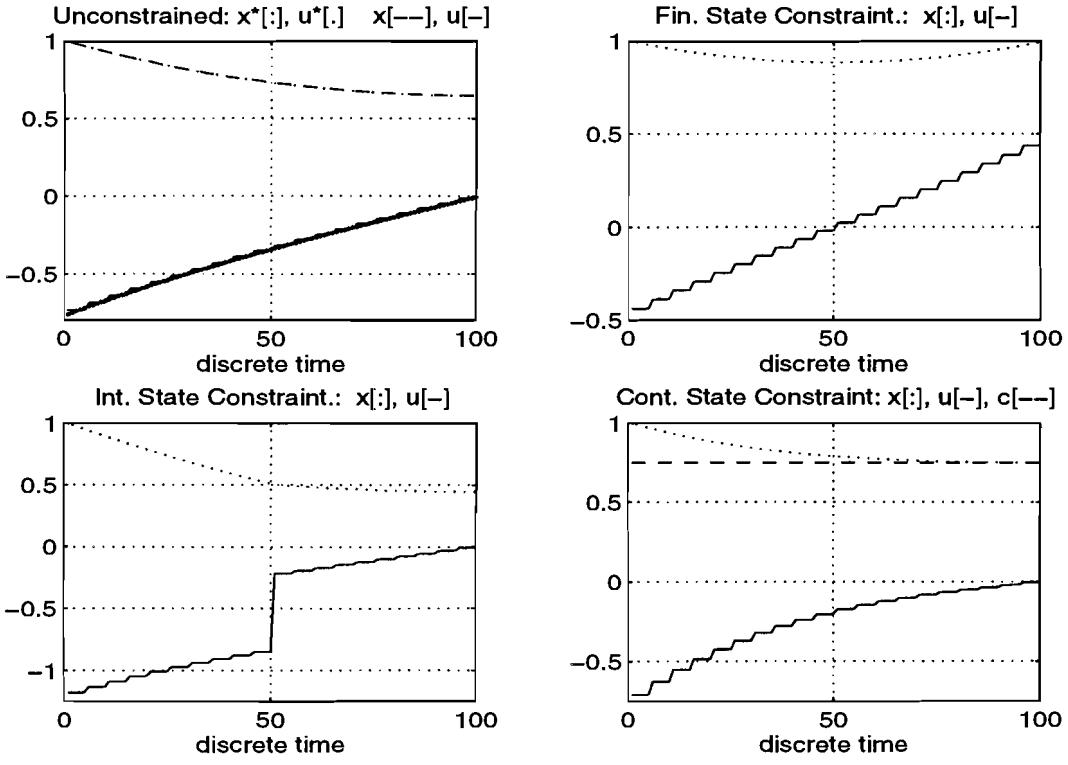

Figure 4.1: *Calculation Results Problem 1: unconstrained vs exact,* $R = 0.759028$*; final state constraint,* $R = 0.923457$*; interior point state constraint,* $R = 0.894566$*; continuous state constraint,* $R = 0.776134$.

culated solutions for the 4 different optimal control problems. For the unconstrained case, the relative difference in the optimal objective function is less than 0.0035% using $N_p = 20$. The difference between the computed state and exact state is less than 0.4% at all points. In the three other cases, the constraints are satisfied exactly.

4.2 Problem 2: A nonlinear-constrained optimal control problem

The problem is taken from [2]. The control process (discretized by the Euler scheme using time step h = 0.02) is described by the following difference equations:

$$\begin{array}{rcl} x_1(t+1) & = & x_1(t) + 0.02x_2(t) \\ x_2(t+1) & = & 0.98x_2(t) + 0.02u(t) \end{array} \tag{4.13}$$

where $t = 0, 1, \ldots, M-1$ and $M = 50$. The initial state is:

$$x_1(0) = 0, \qquad x_2(0) = -1. \tag{4.14}$$

The problem is to minimize

$$G_0 = \sum_{t=0}^{M-1} x_1(t)^2 + x_2(t)^2 + 0.005u(t)^2 \tag{4.15}$$

subject to the non-linear continuous state inequality constraint:

$$C[t, x_2(t)] = 8(0.02t - 0.5)^2 - x_2(t) - 0.5 \geq 0 \qquad \text{for } t = 0, 1, \ldots, M-1. \tag{4.16}$$

Starting with $N_p = 5$ and initial guess $u(t) = 0$, $t = 0, 1, \ldots, M-1$, improved solutions are computed by increasing N_p and by using the outcome of the previous run as the new initial guess. For the constraint transcription is chosen $\epsilon = 10^{-3}$. Table (4.1) summarizes the

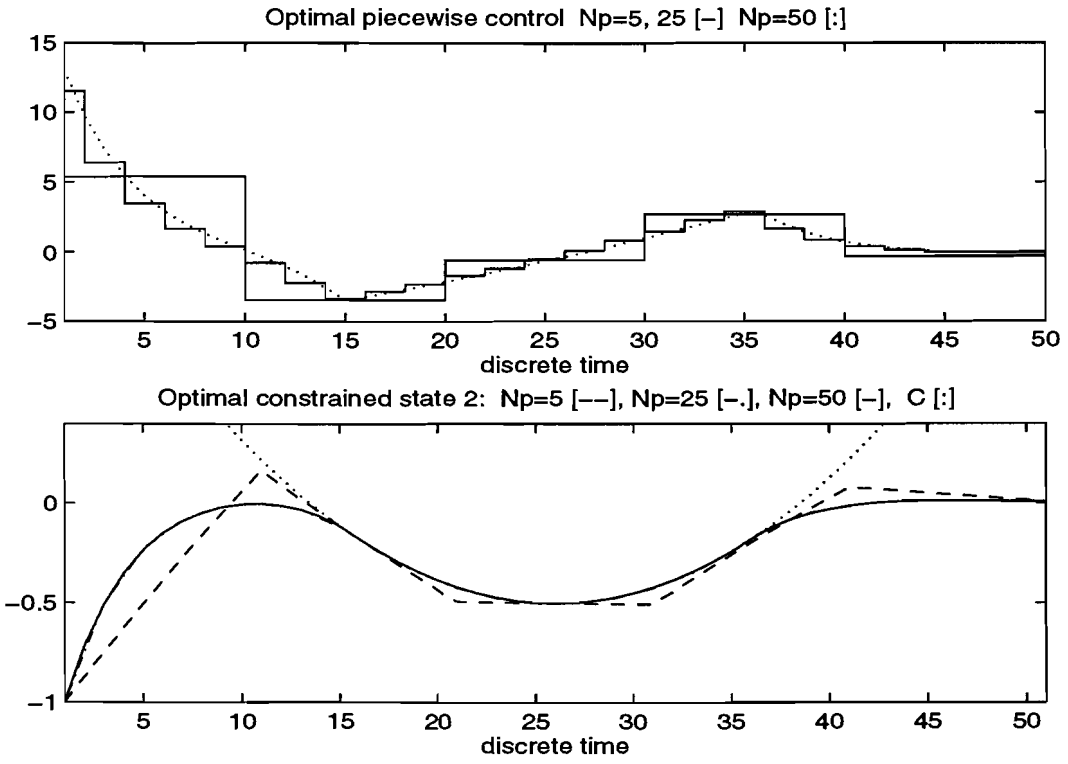

Figure 4.2: *Calculation Results Problem 2.*

Table 4.1: Optimization results problem 2

N_p	$G_0(M)$	$\epsilon/4 + G_\epsilon$	constraint violation	iterations
5	11.3413650	-7.96e−17	no	109
25	9.2251165	-2.56e−17	no	351
50	9.1415505	-2.71e−19	no	295

computed results. These results are consistent with those in [2]. This is not really surprising

because Teo and Liu implemented the optimization package NLPQL [16], which makes use of the same kind of optimization methods as CFSQP does. Figure (4.2) shows plots of the optimal control and optimal constrained state.

4.3 Problem 3: The vertical ascent of a rocket

This highly nonlinear problem concerns the vertical ascent of a rocket. The original formulation (continuous-time version) is taken from [11]. A discrete version of the control process is obtained by the Euler scheme to discretize the system equations, where the time step is taken as one second:

$$\begin{aligned} x_1(t+1) &= x_1(t) - u(t) \\ x_2(t+1) &= x_2(t) + x_3(t) \\ x_3(t+1) &= x_3(t) + \frac{Vu(t) - Q[x(t)]}{x_1(t)} - g, \end{aligned} \tag{4.17}$$

where $x_1(t)$ is the mass $(10^3 kg)$ of the rocket, $x_2(t)$ is the altitude (km) above the earth's surface, $x_3(t)$ is the rocket velocity (km/s), $u(t)$ is the mass flow rate, $V = 2$ is the constant gas nozzle velocity, $g = 0.01$ km/s^2 is the acceleration due to gravity (assumed constant), and $Q[x(t)]$ is the aerodynamic drag defined by the formula

$$Q[x(t)] = 0.05\,[x_3(t)]^2 \exp[0.01 x_2(t)]. \tag{4.18}$$

The initial state of the rocket is $x_1(0) = 1$, $x_2(0) = x_3(0) = 0$. At the terminal time $M = 100$, the final value of the mass is constrained to be 20% of the initial mass: $x_1(M) = 0.2$. The bounds on the control are:

$$0 \le u(t) \le 0.04 \qquad \text{for } t = 0, 1, \ldots, M-1. \tag{4.19}$$

The control objective is to maximize the rocket's peak altitude by suitable choice of the mass flow rate. In other words, we want to minimize

$$G_0 = -x_2(M) \tag{4.20}$$

subject to the terminal-state constraint

$$G_1 = [x_1(M) - 0.2] = 0, \tag{4.21}$$

Even partitions on the time interval are used for parametrizing the control. Starting with

Table 4.2: *optimization results*

N_p	$x_1(M)$	$x_2(M)$	iterations
5	0.200	34.710452	8
10	0.200	35.400736	5
20	0.200	35.888522	18
25	0.200	36.156419	2
none	0.200	35.927147	29

$N_p = 5$ and the initial guess for the optimal control $u(t) = 0.008$, $k = 0, 1, 2, \ldots, M-1$,

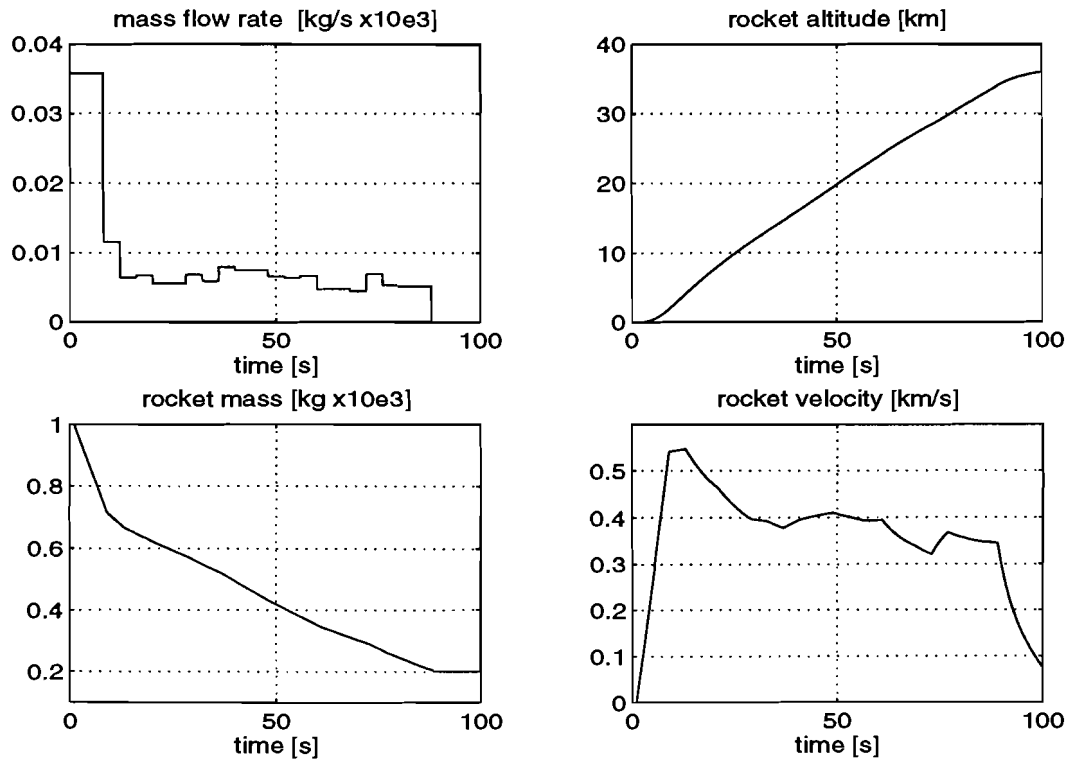

Figure 4.3: *optimization results: Mass flow rate, Mass, Altitude and velocity of the rocket*

improved solutions are obtained by increasing N_p and using the results from the previous run as the new initial guess. Table (4.2) shows the intermediate and final results. The last row indicates the optimization result when no parametrization was applied. It turns out that the result is consistent with that reported in [2]. Figure (4.3) shows the behaviour of the control and state variables.

4.4 Problem 4: A Container crane optimal control problem

A very real and complex program of transferring containers from a ship to a cargo truck at the port of Kobe was considered by Sakawa and Shondo [18]. The container crane is driven by a hoist motor ($u1$) and a trolley drive motor ($u2$). For safety reasons, the objective is to minimize the swing of the container during transfer as well as the swing at the end of transfer. The states of the system consist of vertical motion ($x1$), horizontal motion ($x2$) and diagonal motion ($x3$). The initial state is given while the terminal state is specified, thus posing a terminal state constraint. In addition the maximum torque of the hoist motor and trolley drive motor cannot be exceeded and thus the controls are bounded. Furthermore, continuous state constraints are present in that the velocity of the trolley and of the hoist are bounded. For brevity, the problem formulation in the various original physical coordinates is left out here. Only the final formulation after appropriate transformation of coordinates will be given.

The container crane problem is defined by:

$$\min_{u(\cdot)} J = \frac{1}{2}\int_0^{t_1} \{w_1 x_3(t)^2 + w_2 x_6(t)^2\} \mathrm{d}t \tag{4.22}$$

subject to the dynamical equations

$$\begin{array}{rcl} \dot{x}_1 &=& x_4 \\ \dot{x}_2 &=& x_5 \\ \dot{x}_3 &=& x_6 \\ \dot{x}_4 &=& u_1 + \delta_1 g x_3 \\ \dot{x}_5 &=& u_2 \\ \dot{x}_6 &=& -\frac{1}{x_2}[u_1 + (1+\delta_1)gx_3 + 2x_5x_6] \end{array} \tag{4.23}$$

with the initial condition for the state

$$x(0) = [0, l_2, 0, 0, -x_{5_{max}}, 0]^T, \tag{4.24}$$

terminal state constraint

$$x(t_1) = [d_1, l_3, 0, x_{4_{max}}, 0, 0]^T, \tag{4.25}$$

continuous state constraints

$$\begin{array}{rcll} |x_4(t)| &\leq& x_{4_{max}} & \\ |x_5(t)| &\leq& x_{5_{max}} & \text{for } t \in [0, t_1] \end{array} \tag{4.26}$$

and control constraints

$$|u_1(t)| \leq u_{1_{max}}, \qquad u_{2_{min}} \leq u_2 \leq u_{2_{max}}, \tag{4.27}$$

where

$$\begin{array}{rcl} u_{1_{max}} &=& v_{1_{max}} - \delta_1\beta \max\{v_{2_{max}}, -v_{2_{min}}\} \\ u_{2_{min}} &=& v_{2_{min}} + \delta_2\beta v_{1_{max}} \\ u_{2_{max}} &=& v_{2_{max}} - \delta_2\beta v_{1_{max}}. \end{array} \tag{4.28}$$

We use the following parameter values (the units are suppressed for brevity):

$$\begin{array}{c} w_1 = w_2 = 1, \qquad t_1 = 9 \\ g = 9.81, \qquad l_2 = 22, \qquad l_3 = 14, \qquad d_1 = 10 \\ \delta_1 = 1.76, \qquad \delta_2 = 0.075, \qquad \beta = 0.1 \\ v_{1_{max}} = 2.98, \qquad v_{2_{min}} = -0.831, \qquad v_{2_{max}} = 0.735 \\ x_{4_{max}} = 2.5, \qquad x_{5_{max}} = 1. \end{array} \tag{4.29}$$

To adapt the problem to the unified framework, the terminal state constraints (4.25) and continuous state constraints (4.26) are transformed into the following terminal state constraints:

$$\begin{array}{rcl} g_1 &=& [x_1(t_1) - d_1]^2 + [x_2(t_1) - l_3)]^2 + x_3(t_1)^2 + \\ && +[x_4(t_1) - x_{4_{max}}]^2 + x_5(t_1)^2 + x_6(t_1)^2 = 0 \\ g_2 &=& w \int_0^{t_1} \min\{x_{4_{max}}^2 - x_4(t)^2, 0\} \mathrm{d}t = 0 \\ g_3 &=& w \int_0^{t_1} \min\{x_{5_{max}}^2 - x_5(t)^2, 0\} \mathrm{d}t = 0. \end{array} \tag{4.30}$$

During actual computation, $w = 5$ was used. After system discretization using the Euler method, the optimal control problem was solved using $M = 90$ and $N_p = 15$. The results are shown in figure (4.4). The terminal state constraints (4.25) as well as the continuous state constraints (4.26) are satisfied. Note that only the states x_3 and x_6 contribute to the cost functional. The shape of x_3 and x_6 strongly resembles the solution found by Goh and Teo [5].

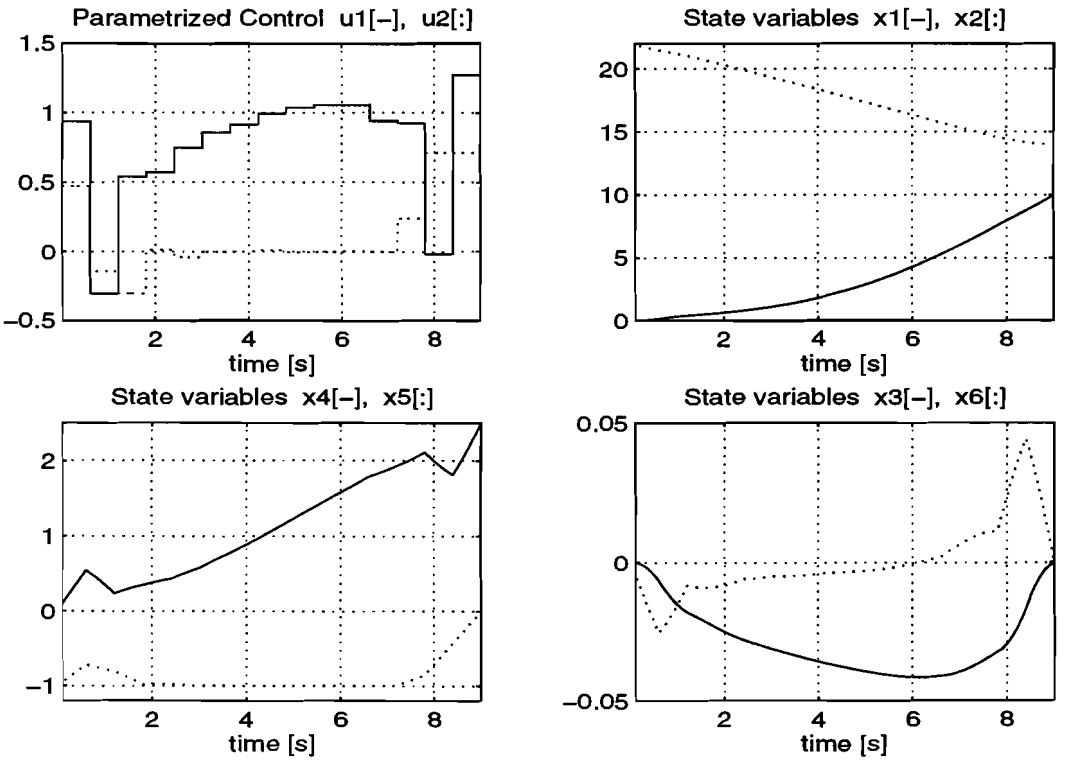

Figure 4.4: *Calculation Results Problem 4*

4.5 Intermediate Conclusions

At this moment, we can draw the conclusion that the method works correctly for relatively simple problems. The constraint transcription method and the parametrization concept have successfully been implemented in combination with CFSQP into one software package. All the calculations led to feasible solutions, satisfying final state constraints, interior point state constraints, as well as hard continuous state constraints. Besides, the computational effort to solve the problems turned out to be negligible.

Up to here, nothing has been said about the difficulties encountered during the different optimization calculations. Very often, we came upon numerical problems, which went most of the time together with an unexpected termination of the optimization process. However, in a later stage, these problems proved to deal with an inaccurate implementation of the gradient method. So, provided the user operates very carefully, the proposed algorithm works fine.

In the following chapter, a much harder problem will be tackled. The difficulty with this problem is mainly the dimension of the system dynamics and the strong nonlinearity.

Chapter 5

The 2-link inverted pendulum

5.1 Introduction

The optimal control problems discussed in the previous chapter were all taken from literature and served mostly as test-case for the algorithm described in Chapter 2.

In this chapter, a more complex optimal control problem is tackled. The problem is well known in the optimal control theory and is called the 2-link inverted pendulum problem. It will be solved using the same method as in the previous chapter. The 2-link inverted

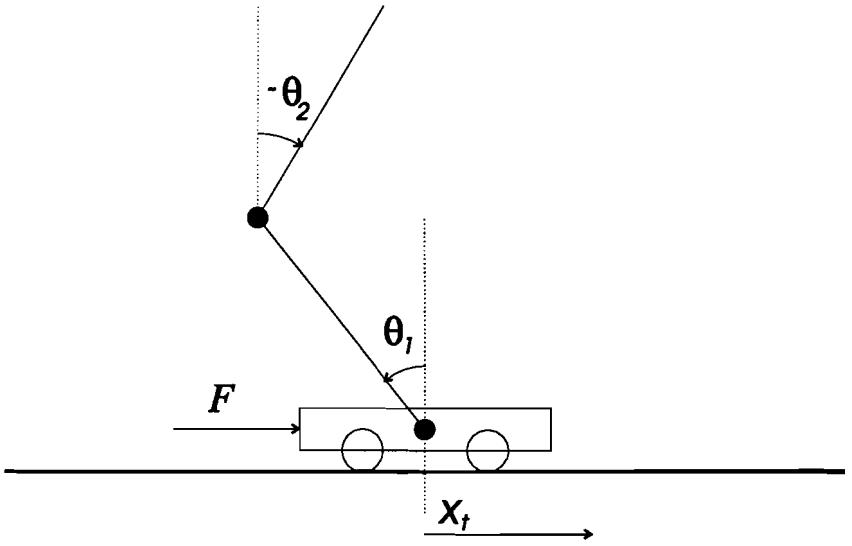

Figure 5.1: *The 2-link inverted pendulum*

pendulum system consists of two links attached to a trolley, which can move horizontally in one direction under influence of a force F. The angles between link 1 and link 2 and the vertical axis are respectively denoted as θ_1 and θ_2. The mass of the trolley is m_t and the masses and lengths of the two links are respectively denoted as m_1, m_2 and l_1, l_2. The position of the trolley along the horizontal axis is called x_t.The whole system is considered to be free of any friction.

The goal is simply to swing the 2 links from the stable initial state to the unstable final state. The initial state is defined by:

$$\theta_1(0) = \theta_2(0) = \pi. \tag{5.1}$$

In theory there is an infinite set of feasible final states defined by:

$$S_f = \{\theta_1(t_f) = n2\pi,\ \theta_2(t_f) = m2\pi \mid n, m = -\infty, \ldots, -1, 0, 1, \ldots, +\infty\}. \tag{5.2}$$

However, the standard feasible final states are:

1. $\theta_1(t_f) = \theta_2(t_f) = 0$
2. $\theta_1(t_f) = \theta_2(t_f) = 2\pi$
3. $\theta_1(t_f) = 0$ and $\theta_2(t_f) = 2\pi$
4. $\theta_1(t_f) = 2\pi$ and $\theta_2(t_f) = 0$

Let this set of 4 standard feasible solutions be denoted as S_4. (From this point on, we shall use the word "solutions" in stead of "feasible final states"). The first two possible solutions are symmetric in terms of control. The only difference is the direction of the swing: left or right. The same can be said for the last two possible solutions. So, we can define 2 different types:

- Type 1: the two links go the same direction: $\theta_1(t_f) = \theta_2(t_f) = 0$ or 2π.
- Type 2: the two links go the opposite direction $\theta_1(t_f) = 0$, $\theta_2(t_f) = 2\pi$ or $\theta_2(t_f) = 0$, $\theta_1(t_f) = 2\pi$.

Solutions which do not belong to S_4 are considered to be alternative. (See section 5.4)

Besides the main goal (getting the two links upwards), we want to minimize the speed ($\dot{x}_t$) of the trolley and the movement ($\dot{\theta}_1, \dot{\theta}_2$) of the two links at the final time t_f. On top of that, the final position of the trolley (x_t) must be the same as the initial one.

5.2 State-Space Description of 2-link inverted pendulum

The dynamics of the 2-link inverted pendulum can be described by the following set of differential equations [17]:

$$M(\theta) \begin{bmatrix} \ddot{x}_t \\ \ddot{\theta} \end{bmatrix} = N(\theta, \dot{\theta}, F) \tag{5.3}$$

where

$$M = \begin{bmatrix} m_t + m_1 + m_2 & (\frac{1}{2}m_1 + m_2)l_1\cos(\theta_1) & \frac{1}{2}m_2 l_2 \cos(\theta_2) \\ (\frac{1}{2}m_1 + m_2)l_1\cos(\theta_1) & (\frac{1}{3}m_1 + m_2)l_1^2 & \frac{1}{2}l_1 l_2 \cos(\theta_2 - \theta_1) \\ \frac{1}{2}m_2 l_2 \cos(\theta_2) & \frac{1}{2}l_1 l_2 \cos(\theta_2 - \theta_1) & \frac{1}{3}m_2 l_2^2 \end{bmatrix}$$

and

$$N = \begin{bmatrix} (\frac{1}{2}m_1 + m_2)l_1\sin(\theta_1) & \frac{1}{2}m_2 l_2 \sin(\theta_2) \\ 0 & \frac{1}{2}m_2 l_1 l_2 \sin(\theta_2 - \theta_1) \\ -\frac{1}{2}m_2 l_1 l_2 \sin(\theta_2 - \theta_1) & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + \begin{bmatrix} F \\ (\frac{1}{2}m_1 + m_2)l_1 g \sin(\theta_1) \\ \frac{1}{2}m_2 l_2 g \sin(\theta_2) \end{bmatrix}$$

where $g = 10$ is the gravitation constant.

In order to calculate a feasible control sequence we have to describe the system dynamics in the state-space.

Define the control variable $u = F$ and the state vector:

$$x = [x_1, \ldots, x_6]^T = [x_t, \theta_1, \theta_2, \dot{x}_t, \dot{\theta}_1, \dot{\theta}_2]^T \tag{5.4}$$

We can now write:

$$f(x,u) = \dot{x} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ M^{-1}N \end{bmatrix} \tag{5.5}$$

Now the dynamics of the inverted pendulum have been described by differential equations in a state space notation. Note that equation (5.5) contains the inverted version of the 3×3 matrix M. The initial state of the process is now given by

$$x_{t_0} = [0, \pi, \pi, 0, 0, 0]^T \tag{5.6}$$

and the desired final state is equal to

$$x_{t_f} = [0, \{0, 2\pi\}, \{0, 2\pi\}, 0, 0, 0]^T. \tag{5.7}$$

The control $u(t)$ is bounded by $-F_{max} \leq u(t) \leq F_{max}$.

5.2.1 Derivation of the gradients

The derivation of the gradient formulae in the case of the inverted pendulum is not trivial and deserves some extra consideration.

The Jacobian is the matrix describing the partial derivatives of the state space equations with respect to the state variables:

$$J = \nabla_x f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & J_{42} & J_{43} & 0 & J_{45} & J_{46} \\ 0 & J_{52} & J_{53} & 0 & J_{55} & J_{56} \\ 0 & J_{62} & J_{63} & 0 & J_{65} & J_{66} \end{bmatrix}, \tag{5.8}$$

where J_{nm} denotes $\partial f_n / \partial x_m$. These Jacobian equations are far too big to calculate manually and thus have been derived, transformed to C-code and optimized by the software-package MapleV3. Optimizing means here: removing redundancy, which is greatly recommended in this case.

$\nabla_u f$ is the vector describing the partial derivatives of the state space equations with respect to the single control variable u:

$$\nabla_u f = \left[\frac{f_1}{u} \frac{f_2}{u} \ldots \frac{f_6}{u} \right] = [0\ 0\ 0\ J_4\ J_5\ J_6] \tag{5.9}$$

where J_n denotes $\partial f_n / \partial u$. These equations have also been derived and transformed to C-code by MapleV3. The gradients $\nabla_x f$ and $\nabla_u f$ have been checked against the second-order finite difference approximation

$$\frac{\partial f(\mathrm{x})}{\partial \mathrm{x}} \approx \frac{f(\mathrm{x} + \zeta) - f(\mathrm{x} - \zeta)}{2\zeta} + O(\zeta^2) \tag{5.10}$$

with the optimal ζ_{opt} being $\varepsilon^{1/3}$, where ε is ten times the machine precision [3]. There were no significant differences noticeable between the Maple-supplied gradient formulae and the second-order finite difference approximation.

5.2.2 The cost functional

In order to satisfy the final state constraints as best as possible, we are going to minimize the following cost functional:

$$\begin{aligned} G_0(u) &= rx_t(t_f)^2 + \dot{x}_t(t_f)^2 + \dot{\theta}_1(t_f)^2 + \dot{\theta}_2(t_f)^2 + \\ &\quad \cos[\theta_1(t_f)][\dot{x}_t(t_f)\dot{\theta}_1(t_f) - g] + \cos[\theta_2(t_f)][\dot{x}_t(t_f)\dot{\theta}_2(t_f) - g] \end{aligned} \tag{5.11}$$

subject to

$$|u(t)| \leq u_{max} \qquad \forall t \in [0, t_f) \tag{5.12}$$

The thought behind the cost functional is to minimize the kinetic energy and to maximize the potential energy of the final state. To limit the final position of the trolley, the term $rx_t(t_f)^2$ is incorporated. The weighting factor r was during optimization equal to 1. A more sensible method is probably to start with $r = 1$ and step by step decrease the weighting factor to a negligible value.

In the absence of hard state constraints, every admissible solution is also a feasible solution. However, we consider a solution being feasible only if the cost functional is exactly -20. Note that this value corresponds to a final state where there is no kinetic energy, the potential energy is maximal and the position of the trolley is $x_t = 0$.

5.3 Calculation results

During calculation, the different system parameters had the following values:

$$m_t = 1kg. \quad m_1 = m_2 = 0.1kg. \quad l_1 = l_2 = 1m. \quad 1 \leq t_f \leq 2 \tag{5.13}$$

The choice for t_f is based upon the mass of the trolley, the mass and length of the two links and the fact that the whole system is free of any friction.

By using the parametrization concept of Chapter 2, we can reduce the dimension of the problem considerably, and at the same moment, restrict the amount of computational effort. During most of the calculations, N_p was 20.

Using the Euler scheme to discretize the system dynamics, we get the following set of difference equations:

$$F(x_t) = x_{t+1} = x_t + \mathrm{h} f(x_t, u_t) \tag{5.14}$$

where $x \in \mathbb{R}^6$, $u \in \mathbb{R}$ and $t = 0, 1, \ldots, M-1$. h=0.005 denotes the discretization step and M is the number of time steps.

The initial guess $u_{ini}(t)$ for the control sequence is a pseudo-random sequence, where $|u_{ini}(t)| < u_{max,ini} \leq 40 \quad \forall t \in I_{M-1}$. There are three reasons to choose a random initial control sequence:

- the lack of any information about a possible feasible control.
- repeated runs with different initial control sequences increases the chance to find a good minimum in the N_p-dimensional space we are dealing with.
- a random initial control sequence leads to different types of solutions, including alternative ones.

Figure (5.2) shows one of the many different solutions found for the 2-link inverted pendulum problem. The final time $t_f = 1.6s$. The number of time steps is $M = 320$ and $N_p = 20$.

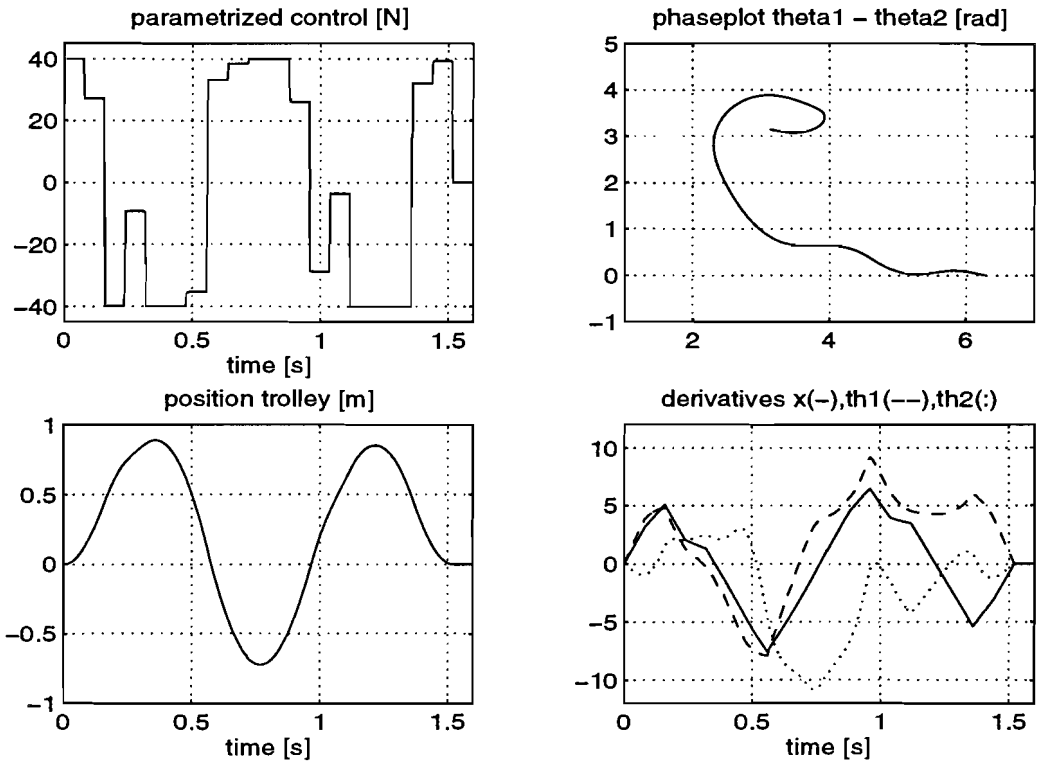

Figure 5.2: *Calculation Results using* $N_p = 20, t_f = 1.6s, \theta(t_f) = (2\pi, 0)^T$*: 1. Piecewise Bounded Control 2. Phase plot 3. Trolley path 4. Derivatives*

As we can see, the final state constraints are all satisfied. The maximum position of the trolley is about 0.8m. This is quite acceptable, keeping in mind that the length of each of the 2 links is 1.0m. The total length the trolley travels is about 5.0m. Note that the direction of the swing of the 2 links is reversed: $\theta(t_f) = (2\pi, 0)^T$. This means that the solution is of type 2.

At first sight, the relation between the parametrized control variable and the position of the trolley, looks perhaps a bit questionable. Approximately, these two variables are related by the formula $F = m_t \ddot{x}_t$. The double integrator and the influence of the two links explain the smoothly sinusoidal path of the trolley on the one side and the piecewise constant form of the control variable F on the other side.

It stands out that the upper and lower bounds on the control variables are not continually reached. This phenomenon implies that the calculated control sequence is not yet optimal and thus that the final time $t_f = 1.6$ is not severe enough: the control variables do not have to be steered out completely to satisfy the final state constraints.

However, when we reduced the time interval $[0, t_f]$ further to $1.4s$ and $1.2s$, the optimizer did not reach the desired cost value of exactly -20. The solutions found with $t_f = 1.4s$ were still quite acceptable ($G_0 = -19,99$), but we decided to consider these solutions as being not feasible.

Figure (5.3) shows a solution of type 1: $\theta(t_f) = (2\pi, 2\pi)^T$. The final time is here $t_f = 2.0s$. The number of time steps is $M = 400$ and $N_p = 20$. Besides the direction of the swing and the final time t_f, there is another significant difference noticeable between this solution and the above discussed solution. The maximum position of the trolley is about 1.5m, which is considerably larger than the earlier mentioned 0.8m. The total length the trolley travels is about 5.5m, which is slightly more than in the type 2 solution. To keep the position of the

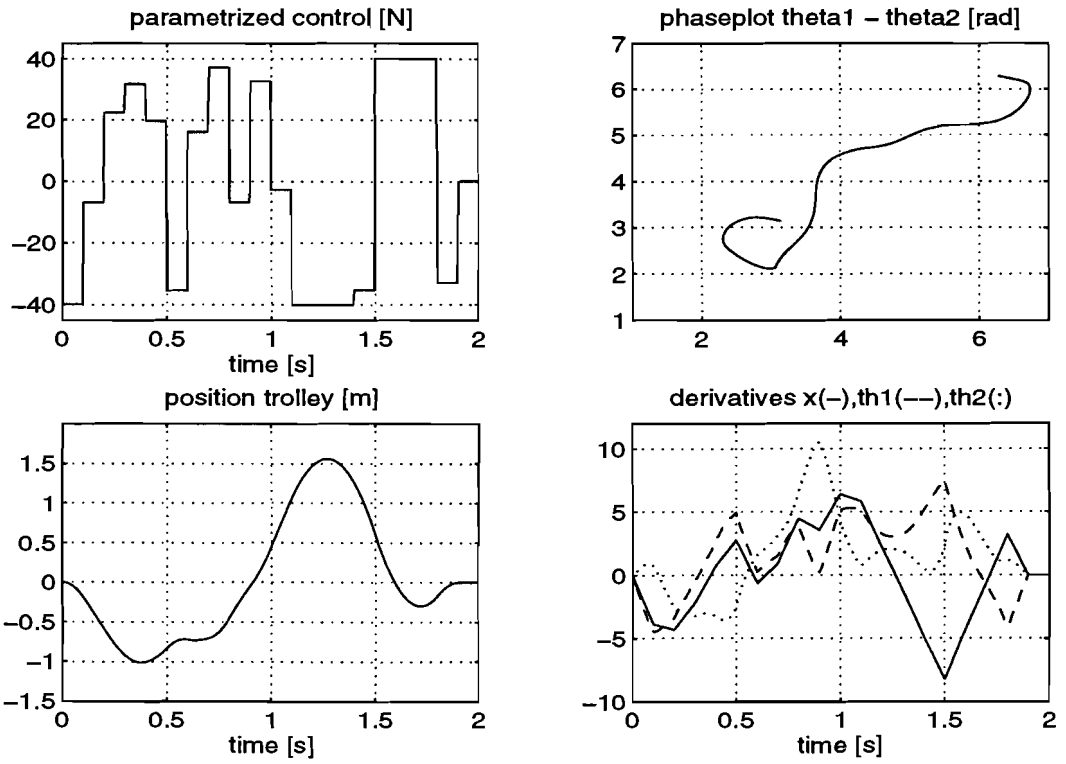

Figure 5.3: *Calculation Results using* $N_p = 20, t_f = 2.0s, \theta(t_f) = (2\pi, 2\pi)^T$*: 1. Piecewise Bounded Control 2. Phaseplot 3. Trolley path 4. Derivatives*

trolley within reasonable bounds we added the following continuous state constraint to the system:

$$|x_t(t)| \leq 1 \quad \forall t \in I_M^+. \tag{5.15}$$

However, this more difficult control problem led only to feasible solutions of type 2. Our conclusion is that there are no feasible solutions of type 1, satisfying constraint (5.15).

5.4 Alternative solutions

During the search for feasible solutions to the 2-link inverted problem, we sometimes found solutions which did not belong to S_4. There appeared to be a relation between the parameter $u_{max,ini}$ and the type of solution. When the maximum bound on the initial control was made larger, more often an alternative solution was found. In these cases, one of the 2 links turns $\pm 540°$ while the other link rotates the standard $\pm 180°$. The price the system pays for this rather wild behaviour is the maximum position of the trolley, which now goes beyond 2 metres. Strikingly was the faster convergence of these alternative solutions. It remains the question whether this better convergence behaviour also means that these "alternative" solutions must all the same be considered superior in terms of optimal control. Figure (5.4) shows one of the alternative feasible solutions.

5.5 Rate of convergence

In the previous chapters no concrete remarks have been made about the rate of convergence. In most cases, it took no longer than a few seconds to find the optimal solution. From

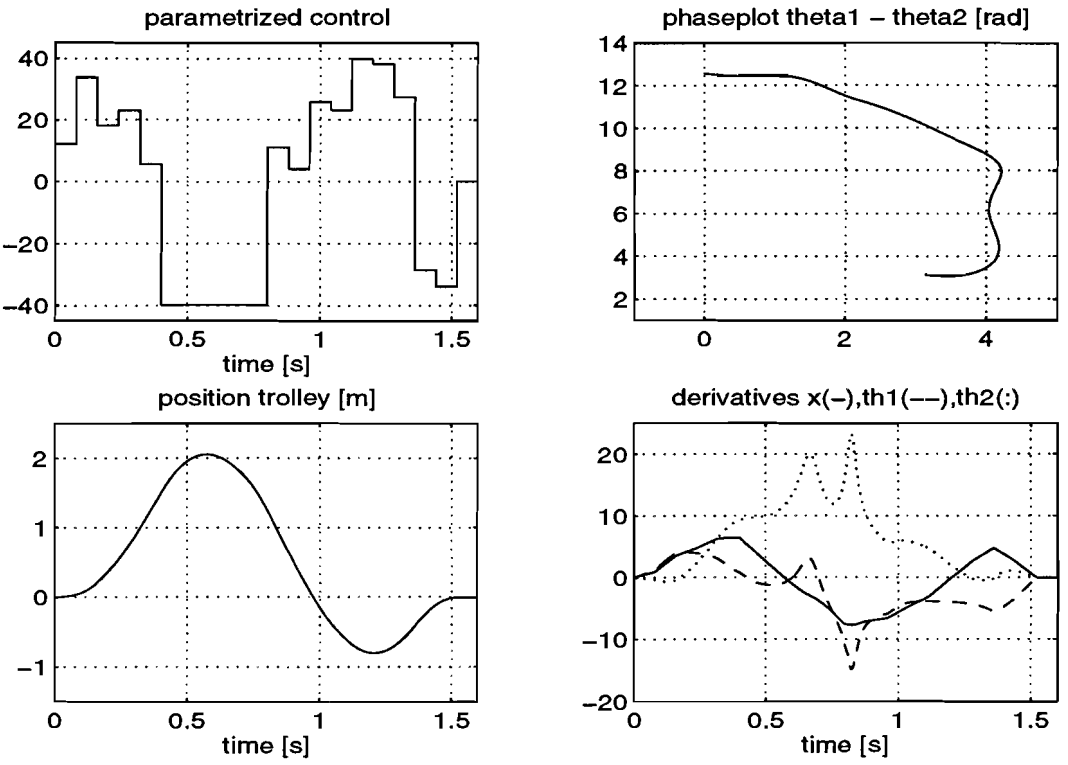

Figure 5.4: *Alternative Solution,* $\theta(t_f) = (0, 4\pi)^T$*: 1. Piecewise Bounded Control 2. Phaseplot 3. Trolley path 4. Derivatives*

computational effort point of view, the 2-link inverted pendulum problem is more demanding than the examples of Chapter 4. However, the duration to find a feasible solution is still quite acceptable, especially when the number of parametrized control variables N_p is kept low. Most of the optimization runs have been made on a Alpha axp 175Mhz DEC3000 work station. A few other on a 486DX2 66Mhz (8Mb) PC. The last machine could only be used when N_p was less than 30, because of limited memory allocation.

Table 5.1: Optimization figures inverted pendulum

N_p	Iterations	Computation Time
20	1302	25s $(7m)_{486}$
50	1351	55s
100	1873	7m48s
200	2794	1h48m46s

Table (5.1) shows some figures of optimization runs on the Alpha processor. The time interval is $t_f = 2.0s$, the number of time steps is $M = 400$ (h=0.005) and the stopping criterion is `eps=1e-10`. (This means that the optimization stops when the difference between 2 consecutive iterations is less than `eps`). The figures are averaged values because the convergence of the optimization depends on the initial control, which is a random sequence. The figures of table (5.1) represent solutions of type 1.

Every optimization used the full amount of allowed working memory of 5Mb. For this

reason, there are no specific allocation figures included.

It turns out that the computation time grows rapidly when the number of parametrized control variables increases. Note that the computation time for $N_p = 20$ is about 300 times less than for $N_p = 200$. For $N_p = 20$, a comparison between the Alpha processor and the 486 can be made. The workstation turns out be about 20 times faster than the PC for this specific problem.

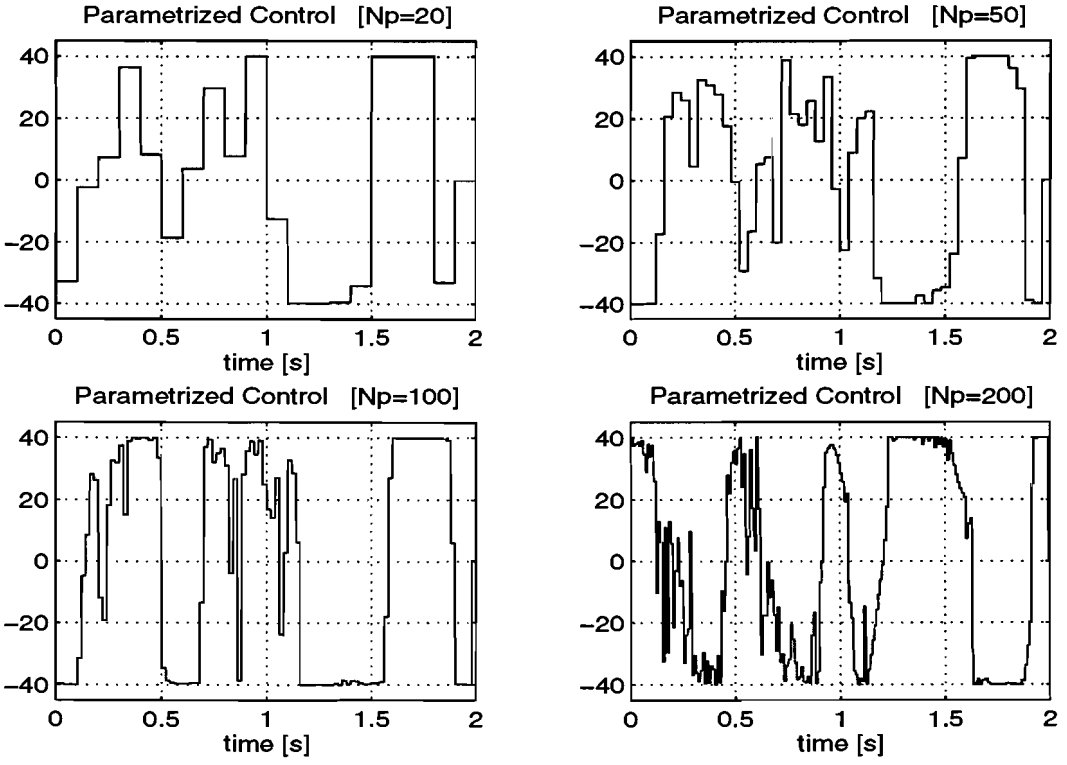

Figure 5.5: *Parametrized Control versus* N_p.

It appeared that using $N_p = 20$, led to better results than using $N_p = 100$ or $N_p = 200$. The solution found by using $N_p = 20$ was superior because of the strict piecewise constant path of the control sequence. By using $N_p = 100$ or 200, one increases the number of degrees of freedom. The unpleasant consequence of this apparent advantage is the wild "high frequency" behaviour of the calculated control sequence.

Figure (5.5) shows the relation between N_p and the calculated (feasible) control sequence. The plots correspond to the figures depicted in table (5.1). Note that the control sequence for $N_p = 200$ is the mirrored version of the other three plots. This is because the final state of this solution was $\theta(t_f) = (0, 0)^T$, while the other three control sequences led to $\theta(t_f) = (2\pi, 2\pi)^T$.

An example of a typical convergence behaviour of the cost functional (5.11) for type 2 solutions is depicted in figure (5.6). The convergence looks quite sound, never appearing to swim in a local minimum. Note that the number of iterations is less than the figures depicted in table (5.1). This is because the curves in this plot correspond to the convergence of type 2 solutions.

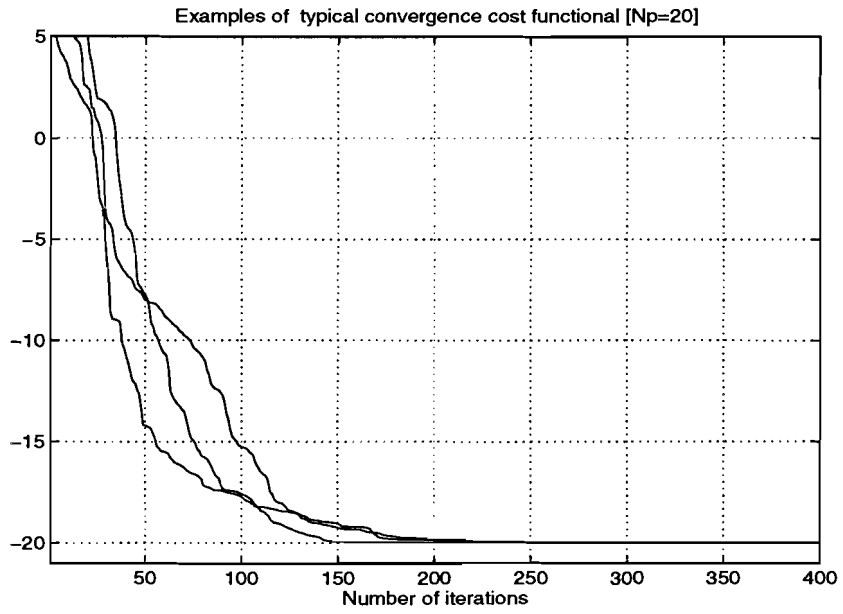

Figure 5.6: *Example of convergence cost functional (type 2 solution)*

5.6 The 3-link inverted pendulum

After having found mathematical solutions for the 2-link inverted pendulum, a logical thing to do is tackling the much harder problem with 3 links. As in many other problems, the curse of dimension applies here. However, with the aid of MapleV3, it is possible to derive the state-space dynamics and the necessary gradient formulae.

The dynamics of the 3-link inverted pendulum are taken from [17] and can be described by:

$$M(\theta)\begin{bmatrix} \ddot{x}_t \\ \ddot{\theta} \end{bmatrix} = N(\theta, \dot{\theta}, F) \tag{5.16}$$

where the symmetric matrix M is defined by:

$$M = \begin{bmatrix} m_t + \sum_{k=1}^{3} m_k & f(m_{1,3})l_1\cos(\theta_1) & f(m_{2,3})l_2\cos(\theta_2) & f(m_{3,3})l_3\cos(\theta_3) \\ & g(m_{1,3})l_1^2 & f(m_{2,3})l_1l_2\cos(\theta_2-\theta_1) & f(m_{3,3})l_1l_3\cos(\theta_3-\theta_1) \\ & & g(m_{2,3})l_2^2 & f(m_{3,3})l_2l_3\cos(\theta_3-\theta_2) \\ & & & g(m_{3,3})l_3^2 \end{bmatrix}$$

and

$$N = \begin{bmatrix} f(m_{1,3})l_1\sin(\theta_1) & f(m_{2,3})l_2\sin(\theta_2) & f(m_{3,3})l_3\sin(\theta_3) \\ 0 & f(m_{2,3})l_1l_2\sin(\theta_2-\theta_1) & f(m_{3,3})l_1l_3\sin(\theta_3-\theta_1) \\ -f(m_{2,3})l_1l_2\sin(\theta_2-\theta_1) & 0 & f(m_{3,3})l_2l_3\sin(\theta_3-\theta_2) \\ -f(m_{3,3})l_1l_3\sin(\theta_3-\theta_1) & -f(m_{3,3})l_2l_3\sin(\theta_3-\theta_2) & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \\ \dot{\theta}_3^2 \end{bmatrix} + \begin{bmatrix} F \\ f(m_{1,3})gl_1\sin(\theta_1) \\ f(m_{2,3})gl_2\sin(\theta_2) \\ f(m_{3,3})gl_3\sin(\theta_3) \end{bmatrix}$$

where $g = 10$ is the gravitation constant and $f(\cdot), g(\cdot)$, are functions of the link masses:

$$f(m_{t,n}) = \sum_{s=t}^{n}(1 - \frac{1}{2}\delta_{ts})m_s \tag{5.17}$$

$$g(m_{t,n}) = \frac{1}{2}\sum_{s=t}^{n}(1 - \frac{2}{3}\delta_{ts})m_s \tag{5.18}$$

with δ_{ij} denoting the usual Delta function.

The applied method to solve the problem is analogous to the approach used with the 2-link inverted pendulum. The only difference was the dimension of the problem: the data describing the system dynamics, the Jacobian and the gradients of the 3-link inverted pendulum, became 10 times larger. The cost functional was of the same form as (5.11).

5.6.1 Calculation results

During calculation, the different system parameters had the following values:

$$m_t = 1kg. \quad m_1 = m_2 = m_3 = 0.1kg. \quad l_1 = l_2 = l_3 = 0.6m. \quad t_f \approx 2s \tag{5.19}$$

Note that the total length of the links (1.8 m) is a little shorter than in the 2-link inverted pendulum problem. The total mass is almost the same. For the above reasons, the guess for the final time t_f was once more 2s. With a discretization step of h=.005, the number of time steps became $M = 400$. The number of parametrized control intervals was $N_p = 25$ and the initial guess was taken random.

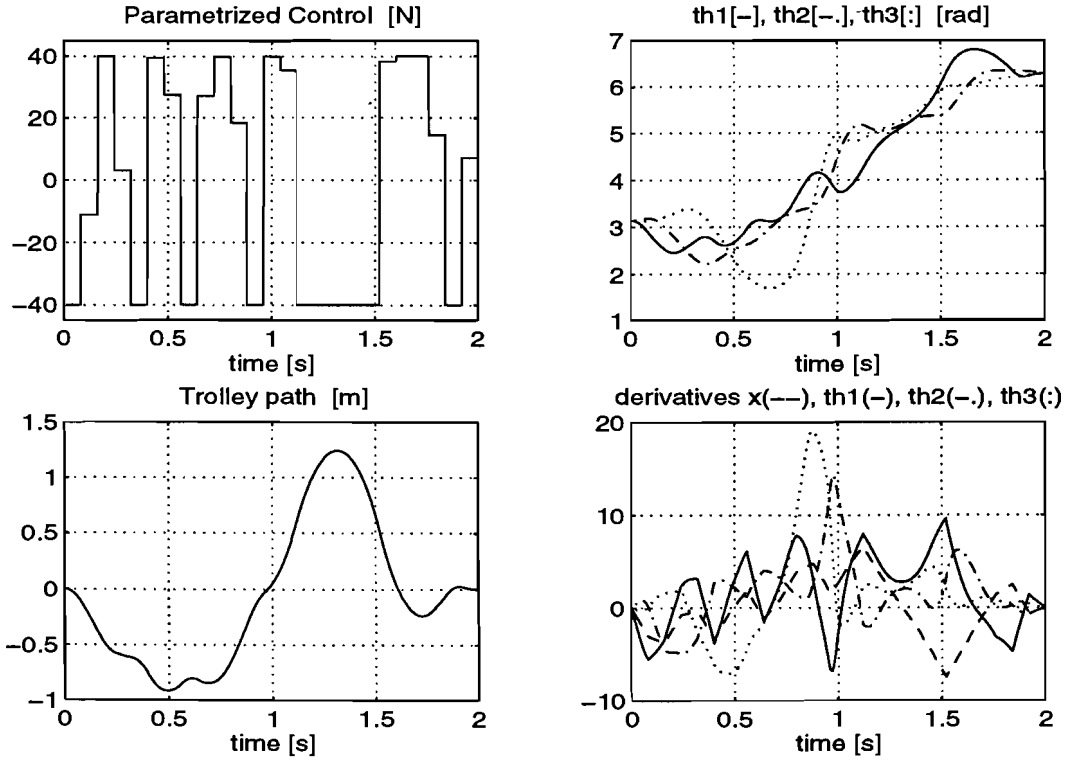

Figure 5.7: *Calculation results 3-link inverted pendulum:* $\theta(t_f) = (2\pi, 2\pi, 2\pi)^T$*:*

Figure (5.7) shows the results of one of the optimizations. It turns out that the desired final state is reached and that the 3 links all go the same direction.$\theta(t_f) = (2\pi, 2\pi, 2\pi)^T$. The maximum position of the trolley is 1.2m and the total length it travels is about 5.0m.

It stands out that the bounds on the control variables are very often reached. So, in terms of optimal control, this solution is better than the former found control sequences for the 2-link inverted pendulum, where the control bounds were less often hit.

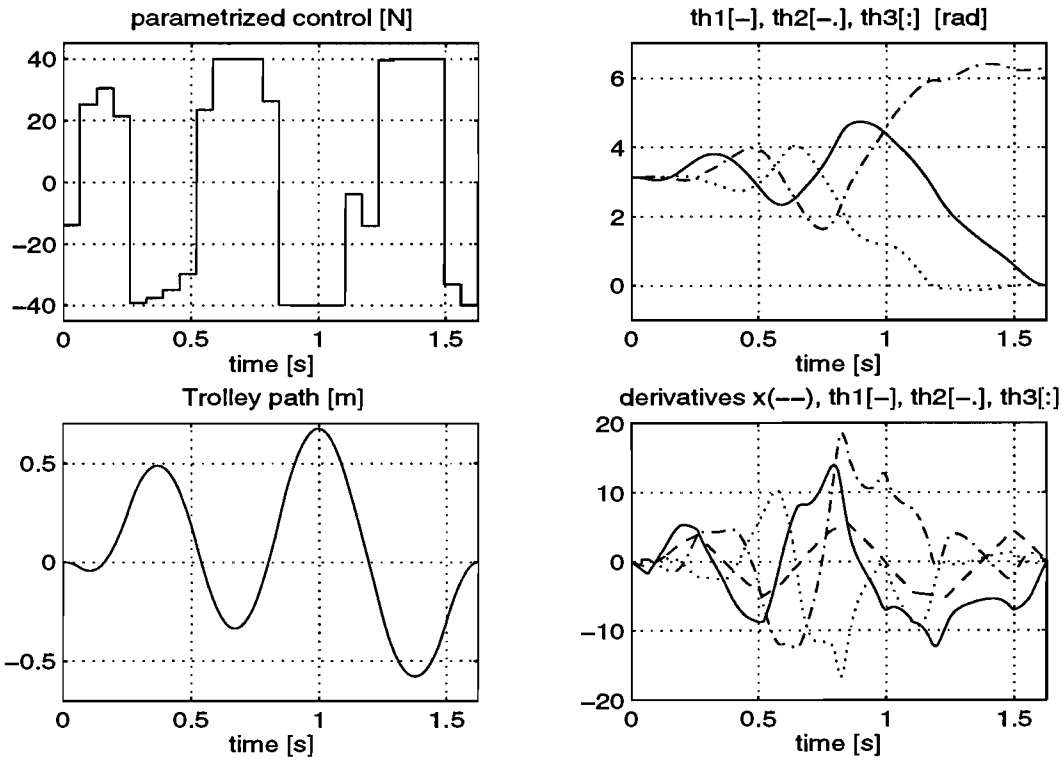

Figure 5.8: *Calculation results 3-link inverted pendulum:* $\theta(t_f) = (0, 2\pi, 0)^T$

Another feasible solution is depicted in figure (5.8). Here, link 2 rotates $+180°$ while the other two links turn $-180°$. The final time $t_f = 1.625s$. As in the case with the 2-link pendulum, this type of solution is superior because, despite the shorter time interval, the control variables do not have to be steered out completely to reach the desired final state. Note that the maximum position of the trolley is merely 0.6m. and that the total length of the trolley path is about 4.1m.

Chapter 6

Terminal time free optimization

In the previous chapters only fixed terminal time optimal control problems have been solved. The goal was simply to minimize the cost functional for a specific number of time steps M. In many cases, however, the terminal time is not fixed and becomes a system parameter, which has to be optimized at the same time with the cost functional. It will be clear that these terminal time free optimization problems are much harder to solve than the terminal time fixed versions.
When the terminal time M is left free, we can distinguish 2 different kinds of problems:

- the *time-optimal* control problem
- the *time-minimal* control problem

Before we proceed, we make the following assumptions for time-optimal control problems:

- we only consider regular cost functionals of the form:

$$G_0 = [x(M)]^T R[x(M)] + \sum_{t=0}^{M-1} [x(t)]^T Q[x(t)] + [u(t)]^T P[u(t)] \tag{6.1}$$

  where $Q > 0$ or $P > 0$.

- the cost functional G_0 is a function of the number of time steps M.
  $G_0 = G(M)$, $M = M_{min}, \ldots, M_{opt}, \ldots, \infty$ where M_{min} is the minimal number of time steps for which a feasible solution can be found and $G(M_{opt})$ is the solution for the time-optimal control problem. $M_{min} \leq M_{opt}$

- $G(M)$ is convex. Thus there is only 1 global minimum. In this minimum M is equal to M_{opt}.

This last assumption is based on 2 thoughts:

1. for $M < M_{opt}$, the system needs much energy to satisfy all the present state constraints, yielding to a large Σ-term.
2. a regular cost functional has the property to grow when the number of time steps increases.

Note that M_{min} corresponds with the time-minimal problem and a much simpler cost functional of the form:

$$G_{M_{min}} = \sum_{t=0}^{M-1} 1 \tag{6.2}$$

It is rather difficult to describe the concept time-optimal in a non-mathematical context. We make the assumption that for each element in a given range of time steps the cost functional has been minimized. So, we have a whole set of minimized cost functionals, each belonging to a certain number of time steps M. The time-optimal solution is now the number of time steps and the corresponding control and state sequence for which the minimized cost functional is overall minimal.

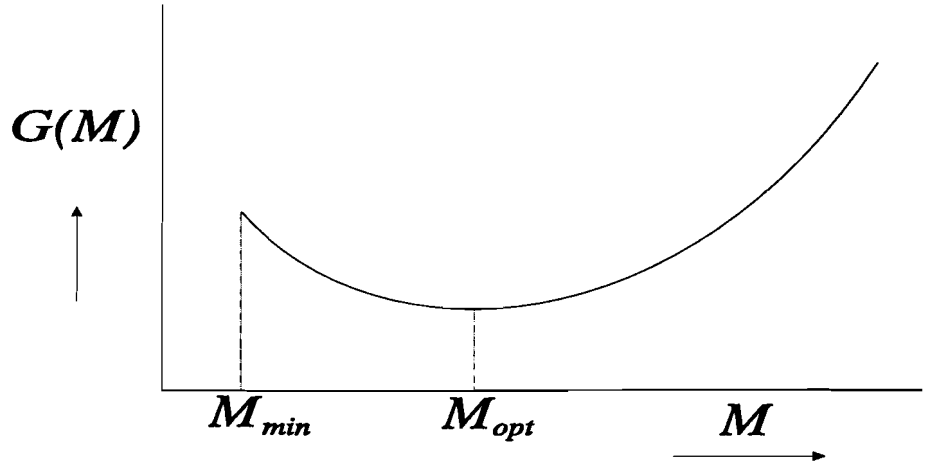

Figure 6.1: *Minimized Cost functional vs M*

The above can perhaps be made more clear by figure (6.1). It turns out that the time-optimal solution actually corresponds to the minimum of $G(M)$. In other words, we are looking for the number of time steps M, where

$$\frac{\partial G(M)}{\partial M} = 0 \tag{6.3}$$

One may ask whether it is possible to extend the method of Chapter 2 by incorporating the extra gradient (6.3) into the gradient algorithm. Indeed, this is possible but, for adding this feature, the algorithm should be converted drastically. What is left for us, is a repeatedly optimization process, each time for a different number of time steps.

In this chapter, a simple but effective heuristic to find the time optimal solution will be presented. No attention will be paid to the time-minimal problem.

6.1 Description of the time-optimal search algorithm

In this section the searching algorithm for the time-optimal solution will be explained. One of the aspects with these terminal time-free optimization problems is that the user often does not have knowledge about the minimal amount of time M_{min} a dynamical process needs to make a specific state transition, satisfying all the present constraints. That is why the algorithm has two stages. During the first stage the number of time steps is increased until a feasible control sequence is found. Next, using a pseudo binary search heuristic, the optimizer tries to find the optimal number of time steps M_{opt}.

During this first stage the number of time steps M is increased with a certain stepsize until a feasible solution is found. Feasible means here: satisfying all the constraints with a precision chosen by the user. The decision whether a control sequence generated by the optimization software package CFSQP [1] is feasible is thus made by the user. This is because CFSQP considers a control sequence only being feasible if an iteration process is ended successfully, which means that there may not have been any irregularities during the iteration process. If the final control sequence satisfies all the original constraints but the iteration process was prematurely stopped, CFSQP considers the iteration as being unsuccessfully terminated.

The search for the optimal number of time-steps is a rather difficult task because during the search there is no information available about the derivative of the cost functional with respect to the number of time-steps. The number of optimization calculations is tried to be kept as low as possible by using a pseudo "binary-search" algorithm. One of the most important requirements is to avoid double optimization, which means calculating the function $G(M)$ twice at the same number of time steps.

Assuming the cost functional (6.1) is a convex function of the number of time-steps, the heuristic can roughly be summarized as follows:

1.
```
M=MStart;
while(not feasible) {
      M=M+Step;
      Optimize G(M)
}
```

2.
```
Mopt=M;
LM=Mopt-Step;     /* lower bound M */
UM=MaxM;          /* upper bound M */
```

3.
```
M=(Mopt+LM)/2;
Optimize G(M);
if G(M) <= G(Mopt) {
      UM=Mopt;
      Mopt=M;
}
```
   proceed with 3.

```
else LM=M;
```

4.
```
M=(Mopt+UM)/2;
Optimize G(M);
if G(M) <= G(Mopt) {
      LM=Mopt;
      Mopt=M;
}
else UM=M;
```
   proceed with 3.

The above algorithm can be considered as a moving M-window, getting smaller each time a better solution has been found, eventually leading to the optimal number of time steps `Mopt`. Note that no information is displayed about stopping criteria. This is left to the user.

When a better solution is found, the algorithm first tries to find a more optimal solution at `(Mopt+LM)/2`. Only if this try turns out to be unsuccessful, a new optimization is done

at (Mopt+UM)/2. Thus this algorithm has the preference to search in the negative direction with respect to the number of time steps. This is because a standard cost functional usually increases when the number of time-steps becomes larger.

An real example of the relation between M and G_0 is calculated on the basis of the following problem:

$$\min_u G_0 = \sum_{t=0}^{M-1} x_1(t)^2 + x_2(t)^2 + 0.005u(t)^2, \tag{6.4}$$

subject to

$$\begin{aligned} x_1(t+1) &= x_1(t) + 0.02x_2(t) \\ x_2(t+1) &= 0.98x_2(t) + 0.02u(t) \end{aligned} \tag{6.5}$$

where the initial state is defined by

$$x_1(0) = -1, \qquad x_2(0) = 0. \tag{6.6}$$

The terminal state and continuous control constraints are given by:

$$x_1(M) = 0, \qquad x_2(M) = -1 \quad \text{and} \quad |u(t)| \leq 5 \quad \forall t \in I_{M-1} \tag{6.7}$$

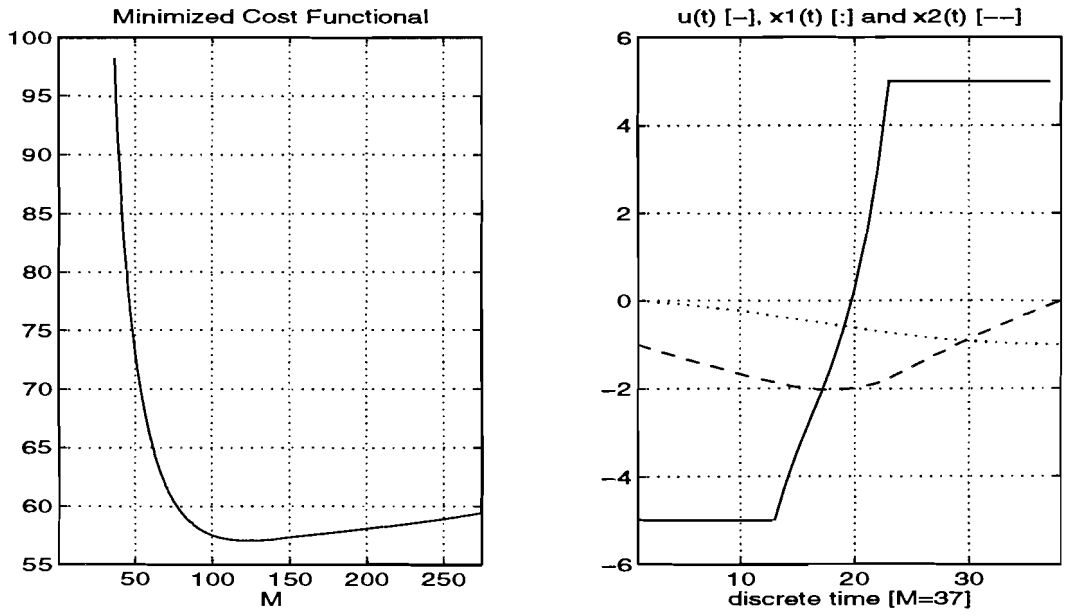

Figure 6.2: *left: Cost functional $G(M)$. right: State and Control trajectories for $M_{min} = 37$*

The left plot in figure (6.2) shows the relation between the minimized cost functional (6.4) and the number of time steps M. It turns out that, the assumption of the function $G(M)$ being convex, is correct for this specific control problem. The minimum value for (6.4) is 57.05 for $M = 125$. For $M < 37$, no feasible solutions were found.

In the right plot of figure (6.2), the control sequence $u(t)$ and the feasible state trajectories $x_1(t), x_2(t)$, are depicted for $M = M_{min} = 37$. Note that the bounds on the control variables are still not continually reached. This phenomenon implies that this control sequence is not the solution for the time-minimal problem. This can probably explained by the presence of two contradictorily purposes: the minimization of the cost functional (6.4) and the diminishing of the number of time instants, resulting in an increase of (6.4). Besides, one must realize that each optimization for a certain number of time steps, remains a non-convex problem.

Chapter 7

Conclusions & Recommendations

The following conclusions can be drawn:

- A general constrained optimal control problem can be transformed into a mathematical nonlinear optimization problem. The gradients for this nonlinear optimization problem are derived using the Hamiltonian for each canonical functional.
- The optimization package CFSQP appears to be suitable for solving the nonlinear optimization problems.
- Representing individual controls over a number of time steps by a single parameter greatly reduces the number of decision variables in problems with a large number of time steps. In this way, the computational effort to solve optimal control problems is strongly restricted.
- It is sensible to start an optimization with a small number of parametrized control variables and gradually refining the grid (if needed).
- The constraint transcription method makes it possible to handle continuous hard state constraints quite easily.
- The implementation of the gradient formulae into the optimization package CFSQP remains a non-trivial issue. Therefore, a user-friendly environment is difficult to develop: each individual problem still needs a specific approach.
- A simple but effective terminal time free optimization heuristic has been developed.
- The software package MapleV3 proves to be useful for the derivation of the gradient formulae in complex problems.
- For the N-link inverted pendulum problem, mathematical suboptimal solutions have been found for $N = 2$ and $N = 3$.

The recommendations are:

- Development of an user-friendly environment for control optimization package.
- Incorporation of more accurate explicit discretization schemes like Adams-Bashforth.
- Improving the time-optimal search algorithm by using the Fibonacci search method.
- Development of a more sophisticated terminal time free optimization algorithm. Therefore, the extra necessary condition (6.3) for a time optimal solution must be incorporated.
- Development of a continuous-time version.
- Design of a state-tracking controller for the 2-link inverted pendulum problem. The mathematical solutions presented in this report could possibly serve as reference signals.

Appendix A

CFSQP-implementation of the inverted pendulum problem

This appendix contains the source code of the CFSQP-implementation of the 2-link inverted pendulum problem discussed in Chapter 5. For brevity, only the most important declarations and functions are given.
For further details of CFSQP I refer to the User's Guide [1]

```
/*--------The 2-link inverted pendulum problem-----------*/

#include "cfsqpusr.h"

/*---------------User Supplied Constants-----------------*/

#define M       400   /* number of time steps */
#define h       .005  /* discretization step */
#define Np      20    /* number of control parameters =
                         number of programming variables */

#define MaxIt   2000  /* maximum number of iterations */
#define Tries   10    /* number of optimizations =
                         number of random initial guesses */

#define e       1e-3  /* constraint transcription parameter */
#define Pi      3.14159265358979e0;

#define MaxF    40    /* maximum allowed force on trolley */
#define Maxx    1     /* maximum position^2 trolley */

/*--------Constants describing Cost functional---------*/

#define CON1      pow(x1[M],2)
#define CON2      cos(x2[M])*(x4[M]*x5[M]-10)
#define CON3      cos(x3[M])*(x4[M]*x6[M]-10)
#define CON4      pow(x4[M],2)
#define CON5      pow(x5[M],2)
#define CON6      pow(x6[M],2)
```

```
#define COSTF     CON1+CON2+CON3+CON4+CON5+CON6

/*--------Partial derivatives dCost[M]/dx[M]------*/

#define CON1D     2*x1[M]
#define CON2D     -sin(x2[M])*(x4[M]*x5[M]-10)
#define CON3D     -sin(x3[M])*(x4[M]*x6[M]-10)
#define CON4D     2*x4[M]+cos(x2[M])*x5[M]+cos(x3[M])*x6[M]
#define CON5D     2*x5[M]+cos(x2[M])*x4[M]
#define CON6D     2*x6[M]+cos(x3[M])*x4[M]

/*--------------CFSQP procedures--------------------------*/

void cost();
void constr();
void gr_cost();
void gr_constr();

/*-------------Procedures concerning proces dynamics------*/

void Initial_Control();
void Initial_State();
void Set_Grid();          /* sets grid parameters */
void Dynamics();          /* discrete dynamics */
void Fx();                /* continuous dynamics */

/*------------Procedures concerning Gradients----------*/

void EvalGradients();
void Costates();
void Jacobian();


/*---------------State and Costate variables---------*/

double *x1,*x2,*x3,*x4,*x5,*x6;
double *c1,*c2,*c3,*c4,*c5,*c6;


/*--------Grid vector-------------*/

int Mp[Np];

/*-------Maple generated procedure variables-----------*/

double t1,t2,t3,.............,t194;
```

```
/*-------Declaration and Initialization CFSQP variables---------*/

int
main() {
 int i,nparam,nf,nineq,neq,mode,iprint,miter,neqn,nineqn,
 ncsrl,ncsrn,nfsr,mesh_pts[1],inform;
 double bigbnd,eps,epsneq,udelta;
 double *x,*bl,*bu,*f,*g,*lambda;
 void *cd;

 mode=100;
 iprint=52;
 miter=MaxIt;
 bigbnd=1.e10;
 eps=1.e-10;
 epsneq=0.e0;
 udelta=0.e0;
 nparam=Np;
 nf=1;
 neqn=0;
 nineqn=1;
 nineq=1;
 neq=0;
 ncsrl=ncsrn=nfsr=mesh_pts[0]=0;
 bl=(double *)calloc(nparam,sizeof(double));
 bu=(double *)calloc(nparam,sizeof(double));
 x=(double *)calloc(nparam,sizeof(double));
 f=(double *)calloc(nf+1,sizeof(double));
 g=(double *)calloc(nineq+neq+1,sizeof(double));
 lambda=(double *)calloc(nineq+neq+nf+nparam,sizeof(double));

/*---------allocation user supplied variables--------*/

 x1=(double *)calloc(M+1,sizeof(double));
 ..........
 c6=(double *)calloc(M+1,sizeof(double));

/*----------------Bounds on control parameters-------*/

 for(i=0;i<Np;i++) bl[i]=-MaxF; bl[Np-1]=-1e-6;
 for(i=0;i<Np;i++) bu[i]= MaxF; bu[Np-1]=1e-6;
```

```
/*-----------------Begin Calculation-----------------*/

 Set_Grid();
 Initial_State();

 for(i=0;i<Tries;i++){

       Initial_Control(x);
       Dynamics(x);
       cfsqp(nparam,nf,nfsr,nineqn,nineq,neqn,neq,ncsrl,
             ncsrn,mesh_pts,mode,iprint,miter,&inform,
             bigbnd,eps,epsneq,udelta,bl,bu,x,f,g,
             lambda,cost,constr,gr_cost,gr_constr,cd);
       PrintInform(inform,i);
       PrintDyn(x);
 }


/*-----------------End Calculation-------------------*/

 free(bl);.....free(c6);
 return 0;
}


/*------------CFSQP functions------------------*/

void cost(nparam,j,x,fj,cd)
int nparam,j;
double *x,*fj;
void *cd;
{
int i;
 if(x_is_new) {               /* x_is_new is set by CFSQP whenever  */
      Dynamics(x);            /* the decision variables change */
      x_is_new=FALSE;
 }
 *fj=COSTF;
 return;
}

void
gr_cost(nparam,j,x,gradfj,dummy,cd)
int nparam,j;
double *x,*gradfj;
void (* dummy)();
void *cd;
{
int i,l;
```

```
double GCost[3];

 if(x_is_new) {
     Dynamics(x);
     x_is_new=FALSE;
 }
 Costates(0,x);              /* Calculate Costates Cost functional */

 for(l=0;l<Np;l++) {
 /*
 Calculate Gradient Cost Functional for
 each parametrized Control interval (= sum of individual gradients)
 */
 }

 return;
}

void constr(nparam,j,x,gj,cd)
int nparam,j;
double *x,*gj;
void *cd;
{
int i;
double C,Csum;

 if(x_is_new) {
      Dynamics(x);
      x_is_new=FALSE;
 }

 /*
 Define and Calculate continuous state inequality constraint.
 (Transcripted and smoothed )
 */

 return;
}


void
gr_constr(nparam,j,x,gradgj,dummy,cd)
int nparam,j;
double *x,*gradgj;
void (* dummy)();
void *cd;
{
int i,l;
double GCon[3];
```

```
 if(x_is_new) {
      Dynamics(x);
      x_is_new=FALSE;
 }

 Costates(1,x);        /* Calculate Costates Constraint */

 for(l=0;l<Np;l++) {
 /*
 Calculate Gradient canonical Constraint Functional for
 each parametrized Control interval (= sum of individual gradients)
 */
 }

 return;
}

/*---------------User supplied functions-----------------*/

void Initial_State()
{
 x1[0]=x4[0]=x5[0]=x6[0]=0.000;
 x2[0]=x3[0]=Pi;
 return;
}

void Initial_Control(double *x)
{
int i;

 randomize();
 for(i=0;i<Np;i++) {
      x[i]=random(10000)%MaxF;
      if((random(2)%2)==1) x[i]=-x[i];
 }
 x[Np-1]=0;
 return;
}

void Set_Grid()
{
int i;

 for(i=0;i<Np;i++)   Mp[i]=(int)floor((M/Np)*(i+1));
 return;
}

void Dynamics(double *x)
```

```
{
int i;
double k1[6];

 for(i=0;i<M;i++) {

      /* Calculate Continuous Dynamics */

      Fx(k1,x2[i],x3[i],x4[i],x5[i],x6[i],x[(int)floor((i*Np)/M)]);

      /* Calculate Euler Discretized Dynamics */

      x1[i+1]=x1[i]+h*k1[0];
      x2[i+1]=x2[i]+h*k1[1];
      x3[i+1]=x3[i]+h*k1[2];
      x4[i+1]=x4[i]+h*k1[3];
      x5[i+1]=x5[i]+h*k1[4];
      x6[i+1]=x6[i]+h*k1[5];
 }
 return;
}

void Fx(F,s2,s3,s4,s5,s6,u)
double
F[6],s2,s3,s4,s5,s6,u;
{

 /* Maple supplied differential equations */

}

void EvalGradients(Gradient,s2,s3)
double
Gradient[3],s2,s3;
{

/* Maple Supplied Gradients */

 return;

}


void Costates(int cosn, double *x)
{
int i;
double C,smgr,J4[4],J5[4],J6[4];

 /*
```

```
 Define terminal time Costates c1[M]..c6[M] for
    Cost Functional       (cosn=0)
    Constraint Functional (cosn=1)
 */

 for(i=M-1;i>=0;i--) {

   /* Calculate Jacobian */

   Jacobian(J4,J5,J6,x2[i],x3[i],x5[i],x6[i],x[(int)floor((i*Np)/M)]);

   /*
   Calculate Costates for
         Cost Functional       (cosn=0)
         Constraint functional (cosn=1)
   */
 }

 return;
}


void Jacobian(J4,J5,J6,s2,s3,s5,s6,u)
double
J4[4],J5[4],J6[4],
s2,s3,s5,s6,u;
{

    /* Maple supplied expressions concerning Jacobian. */

 return;
}
```

Bibliography

[1] Lawrence, C. J.L.Zhou & A.L.Tits, 'User's Guide for CFSQP Version 2.1: A C code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems', *Institute for Systems Research, University of Maryland, College Park, Maryland 20742, 1994*

[2] Teo, K.L. Y.Liu & C.J.Goh, 'Nonlinearly Constrained Discrete-Time Optimal-Control Problems', *Applied Mathematics and Computation 38 (1990) p227-248*

[3] Fisher, M.E. & L.S.Jennings, 'Discrete-time Optimal Control Problems with General Constraints', *ACM Transactions on Mathematical Software, Vol.18, No.4 (1992) p401-413*

[4] Rehbock, V. K.L.Teo & L.S.Jennings, 'A Penalty Function Approach to All-Time-Step Constrained Discrete-Time Optimal Control Problems', *Applied Mathematics and Computation, Vol.49 (1992) p215-230*

[5] Goh, C.J. & K.L.Teo, 'Control parametrization: a unified approach to optimal control problems with general constraints', *Automatica 24 (1988) p3-18.*

[6] Teo, K.L. & K.H.Wong, 'Nonlinearly Constrained Optimal Control Problems', *J.Australian Math. Soc. Ser.B Appl.Math. Vol.33, 1992, p507-530*

[7] Teo, K.L. V.Rehbock & L.S.Jennings, 'A New Computational Algorithm for Functional Inequality Constrained Optimization Problems', *Automatica, Vol.29, No.3, 1993, p789-792*

[8] Jennings, L.S. & K.L.Teo, "A Computational Algorithm for Functional Inequality Constrained Optimization Problems', *Automatica, Vol.26, 1990, p371-375*

[9] Jennings, L.S. M.E.Fisher, K.L.Teo & C.J.Goh, 'MISER3: Solving Optimal Control Problems-an update', *Adv.Eng.Software,1991,vol.13,No.4*

[10] Sage, A.P. & C.White.III, *Optimum Systems Control*, Prentice-Hall, Englewood Cliffs, (1977)

[11] Evtushenko, Y.G. *Numerical Optimization Techniques*, Optimization Software, New York. (1985)

[12] Boltyanskii, V.G. *Optimal Control of Discrete Systems*, Wiley, New York. (1978)

[13] Canon, M.D. C.D.Cullum & E.Polak, *Theory of Optimal Control and Mathematical Programming*, McGraw-Hill, New York, (1970)

[14] Bryson, A. *Applied Optimal Control* London, Blaisdell (1969)

[15] Bellman, R. & S.Dreyfus, *Applied Dynamic Programming*, Princeton U.P., Princeton, N.J. (1962)

[16] Schittkowski, K. 'QLD: A FORTRAN Code for Quadratic Programming, User's Guide', *MathematiSches Institut, Universitat Bayreuth, Germany, 1985.*

[17] Larcombe, P.J. 'The Dynamic Equations of Motion for a Horizontally Translating Two Dimensional N-Link Inverted Pendulum", *University of Glasgow, Department of Mechanical Engineering, Control Group. Research Report, May 1990*

[18] Sakawa,Y. & Y.Shindo, 'Optimal Control of Container Cranes', *Automatica, Vol.18, p257-266*