Algorithm 733: TOMP — Fortran Modules for Optimal Control Calculations

DIETER KRAFT
lachhochschule München

A great number of analysis and synthesis problems of modern processes can be written as state and control constrained optimal control problems governed by ordinary differential equations with multipoint boundary values. As the software tools for following this attractive approach are still missing or can be used only by experts, the structure and usage of an easy-to-use software package is described which efficiently solves the given problem Among its features are user-orientation, applicability on personal computers and mainframes, and robustness with respect to model changes and inaccurate starting values. It has been tested on a number of complex engineering tasks, including aerospace and robotic trajectory planning.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization; G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations; G.4 [**Mathematics of Computing**]: Mathematical Software; I.2.9 [**Artificial Intelligence**]: Robotics

General Terms: Algorithms

Additional Key Words and Phrases: Boundary value problems, manipulators, optimal control, robotics, shooting method

1. INTRODUCTION

Within the last decade great progress has been achieved in solving the nonlinear programming problem

$$(\mathbf{NLP}): \quad \min_{x \in R^n} f(x) \tag{1.1}$$

subject to general equality and inequality constraints

$$g_j(x) = 0, \qquad j = 1, \ldots, m_e, \tag{1.2}$$

$$g_j(x) \geq 0, \qquad j = m_e + 1, \ldots, m, \tag{1.3}$$

and to lower and upper bounds on the variables

$$l_i \leq x_i \leq u_i, \qquad i = 1, \ldots, n, \tag{1.4}$$

for a local minimum by sequential quadratic programming (SQP) algorithms. The problem functions $f: R^n \to R^1$ and $g: R^n \to R^m$ are assumed to be continuously differentiable and to have no specific structure. A recent mono-

Author's address: Fachhochschule Munchen, D-80335 Munchen 2, Dachauerstrasse 98b, Germany

graph [Gill et al. 1991] presents an excellent overview on theoretical algorithms and their numerical implementations.

Because of the availability of efficient codes for solving (NLP) [Gill et al. 1983; Powell 1978; Schittkowski 1986] it is time to revisit the problem of calculating optimal control functions for dynamic systems described by ordinary differential equations via mathematical programming. Several authors have pursued this approach within the last 20 years [Brusch and Schapelle 1973; Kamm and Johnson 1973; Tabak and Kuo 1971] by transforming into problem (NLP) the original optimal control problem

$$\textbf{(OCP)}: \min_{\substack{u \in U \\ \pi \in R^{q_\pi}}} \int_a^b \phi_0(t, z(t), u(t), \pi)\, dt + \psi_0(a, z(a), b, z(b)), \quad (1.5)$$

subject to differential constraints

$$\dot{z}_i(t) - \phi_i(t, z(t), u(t), \pi) = 0, \qquad i = 1, \ldots, p, \quad (1.6)$$

boundary conditions, with $a < \tau < b$,

$$\psi_i(a, z(a), \tau, z(\tau), b, z(b)) = 0, \qquad i = 1, \ldots, r, \quad (1.7)$$

which in practical applications mostly appear separated as

$$\psi_{a_i}(a, z(a)) = 0, \qquad i = 1, \ldots, r_a, \quad (1.8)$$

$$\psi_{\tau_i}(\tau, z(\tau)) = 0, \qquad i = 1, \ldots, r_\tau, \quad (1.9)$$

$$\psi_{b_i}(b, z(b)) = 0, \qquad i = 1, \ldots, r_b, \quad (1.10)$$

and (algebraic) state or control constraints

$$\xi_i(t, z(t), u(t), \pi) \geq 0, \qquad i = 1, \ldots, s. \quad (1.11)$$

In problem (OCP) functions are considered depending on one independent variable $t \in [a, b] =: I$. The state $z \in R^p$ is assumed to be absolutely continuous; the control $u \in U \subset R^q$ is assumed to be bounded and measurable, which practically means piecewise continuous. The initial and final time a and b may be fixed or free. Condition (1.9) at some interior time event $\tau \in (a, b)$ characterizes a multipoint boundary value problem. The trajectory $z(t)$ is controlled by a vector of time functions $u_i(t)$, $i = 1, \ldots, q$, and a finite dimensional vector of design parameters π_i, $i = 1, \ldots, q_\pi$. Note, that within problem (OCP), the state equations (1.6) and relations (1.8) represent an initial value problem (IVP) as a subproblem.

The problem described by Equations (1.5–1.11) can be solved by an *indirect* approach by transforming problem (OCP) into a two-point boundary value problem via the necessary optimality conditions of (OCP), the maximum principle of Pontryagin et al. [1964], making use of the costate equations,

$$\dot{\lambda}(t) = -\frac{\partial}{\partial z}\phi(t, z(t), u(t), \pi)\lambda(t), \quad (1.12)$$

a set of p differential equations adjoined to the state equations (1.6). For the solution of the resulting boundary value problem mainly three approaches have been pursued [Keller 1976], e.g., finite difference techniques [Lentini

and Pereyra 1977], collocation procedures [Ascher et al. 1981], and shooting methods [Grimm et al. 1985]. Shooting methods have first been applied to the solution of optimal control problems by Bulirsch [1971]. The state equations (1.6) and costate equations (1.12) are integrated on a finite mesh in the forward direction (from t_a to t_b) with estimated initial conditions. The boundary and continuity conditions are satisfied by solving a system of nonlinear equations by a modified Newton method. The method is characterized by high accuracy and fast (quadratic) convergence near the optimum. Disadvantageous is the inherent small convergence region due to Newton's method and the necessity of formulation and solution of the costate equations. These can be partially overcome by homotopy methods or a hybrid approach [von Stryk and Bulirsch 1992], and by automatic symbolic generation of the costate equations [Schöpf and Deuflhard 1994].

This article presents a software package, Trajectory Optimization by Mathematical Programming (TOMP), by which numerical solutions of problem (OCP) may be calculated by a *direct* shooting approach.[1] This avoids the solution of the costate equations by transforming problem (OCP) into problem (NLP) as described in the sequel. This approach has the advantage of

—extremely high flexibility of the software with respect to problem reformulations which are rather the rule than the exception in industrial applications, and

—a large convergence radius together with superlinear convergence rate near the optimum due to the particular choice of the optimizer. This is an important feature in industrial applications as accurate starting values are seldom known.

Among its disadvantages are that

—it is approximative by its discrete nature.

But the approximations may be chosen arbitrarily accurate depending on the discretization of the problem and thereby on the available computer facilities. The software aims at industrial applications rather than at academic high-accuracy solutions. It can be used as a starting procedure for the latter.

The software is completely written in Fortran and has originally been developed on a mainframe computer with Watfiv and Fujitsu test compilers. It is also best suited for microcomputer applications. It has been tested in this environment using the Fortran compilers Watfor77 Version 3.1, MS-Fortran Version 5.0, and Salford Ftn77/486 Version 2.60, the latter being a genuine 32-bit compiler.

In this article the transformation from (OCP) to (NLP) is presented together with an overview of the software organization and usage. The necessary mathematical tools, e.g., approximation schemes, initial value problem solvers, and nonlinear programming solvers, are described in the original report [Kraft 1991]. Examples are presented to show the performance and the

[1]A direct collocation approach proceeds similarly; the difference is that the initial value problem is solved by collocation instead by a Runge-Kutta-type method [von Stryk 1993].

usage of TOMP. The Appendix gives a detailed description of the subroutine input and output parameters for a simple but comprehensive test example. The software package does not make explicit use of the optimality conditions for problem (OCP); therefore they are omitted here.

2. DIRECT SHOOTING

Direct shooting can be separated into the following steps:

(1) control parameterization,
(2) initial value problem solution, and
(3) parameter optimization.

The first step is described in the following; both the remaining steps represent basic material of numerical mathematics. A summary of the applied algorithms is given in the original report [Kraft 1991].

2.1 Control Parameterization

To use an (NLP) solver for problem (OCP), the infinite dimensional control functions $u(t)$ have to be represented by a finite set of control parameters x_u. This is accomplished as follows:

(1) a number of q time grids Δ_i is defined

$$\Delta_i := \left\{a = t_{i_1} < t_{i_2} < \cdots < t_{i_{n_i-1}} < t_{i_{n_i}} = b\right\}, \qquad i = 1, \ldots, q, \quad (2.1)$$

as partitions of the interval I, one for each control function $u_i(t)$,

(2) with each breakpoint or knot t_{i_j} a control parameter $x_{u_{i_j}}$ is associated

$$x_{u_{i_j}} = u_i\left(t_{i_j}\right), \qquad j = 1, \ldots, n_i, \quad i = 1, \ldots, q, \quad (2.2)$$

which represents the control functions $u_i(t)$ at the breakpoints, and

(3) within each subinterval

$$I_{i_j} = \left[t_{i_j}, t_{i_{j+1}}\right], \qquad j = 1, \ldots, n_{i-1}, \quad i = 1, \ldots, q, \quad (2.3)$$

of the partitions Δ_i, each control function $u_i(t)$ is approximated by an interpolating piecewise polynomial (pp) function $v_{\Delta_i}(t)$ of a certain order

$$u_i(t) \approx v_{\Delta_i}(t), \qquad i = 1, \ldots, q, \quad (2.4)$$

which is defined in Section 3.1 of Kraft [1991].

Thus the vector $u(t)$ is completely and uniquely defined by the finite vector

$$x_u := \{x_{1_1}, x_{1_2}, \ldots, x_{q_{n_q-1}}, x_{q_{n_q}}\}. \quad (2.5)$$

The entire vector of controlling parameters x is composed of the control parameters x_u and the design parameters π

$$x = (x_u^T, \pi^T)^T = \{x_{1_1}, x_{1_2}, \ldots, x_{q_{n_q-1}}, x_{q_{n_q}}\} \cup \{\pi_1, \pi_2, \ldots, \pi_{q_\pi}\}. \quad (2.6)$$

The total number of controlling parameters sums up to $n = \dim(x) = n_1 + n_2 + \cdots + n_q + q_\pi$. It should be noted that a and b are elements of the set of controlling parameters if they are free parameters.

We limit the partitions Δ_i of I to one for each control function. As we require noncoincident breakpoints, this means that except for zero-order pp-functions the approximating controls v_{Δ_i} are continuous. Piecewise continuous pp-functions require coinciding breakpoints or multiple partitions Δ_{i_j}.

2.2 Problem Approximation

2.2.1 *Initial Value Problem.* With $u(t)$ approximated by (2.4) which in turn is represented by a finite number of parameters x_u, the differential constraints (1.6)

$$\dot{\zeta}_i(t) - \phi_i(t, \zeta(t), v_\Delta(t), \pi) = 0, \qquad i = 1, \ldots, p, \tag{2.7}$$

are solved with initial values according to (1.8)

$$\psi_{a_i}(a, \zeta(a)) = 0, \qquad i = 1, \ldots, r_a, \tag{2.8}$$

where $\zeta(t)$ is an approximation to $z(t)$, and $\zeta(a) = z(a)$.

For nonstiff or mildly stiff problems one-step discretization methods of the Runge-Kutta-Fehlberg class [Prince 1981; Shampine and Watts 1976] are preferred as IVP-solvers for efficiency reasons. These methods are summarized in Section 3.2 of Kraft [1991]. The usage of the respective software is explained in Appendix C of Kraft [1991].

If the number of specified initial conditions r_a is less than the number of states p, then $p - r_a$ unspecified states are included in the set of parameters x:

$$x = \{x_{1_1}, x_{1_2}, \ldots, x_{q_{n_{q-1}}}, x_{q_{n_q}}\} \cup \{\pi_1, \pi_2, \ldots, \pi_{q_\pi}\} \cup \{\zeta_1(a), \zeta_2(a), \ldots, \zeta_{p-r_a}(a)\}, \tag{2.9}$$

where the unspecified states $\zeta_i(a)$ are reindexed here for reasons of simplified notation. In this case there are $n = \dim(x) = n_1 + n_2 + \cdots + n_q + q_\pi + p - r_a$ controlling parameters.

2.2.2 *Problem Evaluation.* For given x the trajectory ζ is uniquely defined, and it only depends on x. Thus the cost function

$$\int_a^b \phi_0(t, \zeta(t), v_\Delta(t), \pi)\, dt + \psi_0(a, \zeta(a), b, \zeta(b)) =: f(x) \tag{2.10}$$

and the boundary conditions

$$\psi_{b_i}(b, \zeta(b)) =: g_i(x) = 0, \qquad i = 1, \ldots, r_b, \tag{2.11}$$

$$\psi_{\tau_i}(\tau, \zeta(\tau)) =: g_i(x) = 0, \qquad i = r_b + 1, \ldots, r_b + r_\tau, \tag{2.12}$$

can be evaluated with $m_e = r_\tau + r_b$.

To calculate the state constraints

$$\xi_i(t, \zeta(t), v_\Delta(t), \pi) \geq 0, \qquad i = 1, \ldots, s, \tag{2.13}$$

a pointwise approach is applied. Equation (2.13) is evaluated on a communication grid Δ_c

$$\Delta_c := \{a = t_{c_1} < t_{c_2} < \cdots < t_{c_{t-1}} < t_{c_t} = b\} \tag{2.14}$$

pointwise at certain communication knots t_{c_i}

$$\xi_i\left(t_{c_j}, \zeta\left(t_{c_j}\right), v_\Delta\left(t_{c_j}\right), \pi\right) =: g_{m_e+k}(x) \geq 0,$$
$$i = 1, \ldots, s, \quad j = 1, \ldots, t, \quad k = 1, \ldots, s \times t, \quad (2.15)$$

and the number of inequality constraints $m_l = m - m_e$ is $m_l = s \times t$. The communication grid has to be chosen fine within subintervals in which the constraints are active; that means $g_{m_e+l}(x) = 0$, with $l \in |m_l|$, the set of inequality constraint indices, and it can be coarse elsewhere.

Another possibility to handle algebraic state constraints is to define s auxiliary differential equations which are integrated together with the state equations (2.7). These auxiliary equations determine a quadrature of the constraint violations

$$\dot{\zeta}_{p+i}(t) - \min\{\xi_i(t, \zeta(t), v_\Delta(t), \pi), 0\} = 0, \qquad i = 1, \ldots, s, \quad (2.16)$$

together with the boundary conditions

$$\zeta_{p+i}(a) = 0, \qquad \text{and} \qquad \zeta_{p+i}(b) = 0, \qquad i = 1, \ldots, s. \quad (2.17)$$

The trajectory constraints are then formulated as

$$g_{m_e+i}(x) := \zeta_{p+i}(b) \geq 0, \qquad i = 1, \ldots, s, \quad (2.18)$$

with $m_l = s$.

Now the equations (2.10, 2.11, 2.15, or 2.18) approximate problem (OCP) as problem (NLP).

2.3 Gradient Generation

SQP-based methods for solving problem (NLP) require the availability of the partial derivatives $\partial\varphi(x)/\partial x$ of the problem functions $\varphi = \{f, g_1, \ldots, g_m\}$ with respect to all parameters x. As the $\partial\varphi(x)/\partial x$ cannot be expressed analytically, they must be calculated by some numerical difference scheme. For reasons of efficiency we prefer forward differences:

$$\frac{\partial\varphi_i(x)}{\partial x_j} = \frac{\varphi_i(x_j + h) - \varphi_i(x_j)}{h}, \qquad i = 1, \ldots, m+1, \quad j = 1, \ldots, n, \quad (2.19)$$

with

$$h = \max\{\delta, |x_j|\delta\}, \quad (2.20)$$

and

$$\delta = \max\{\sqrt{\epsilon}, \vartheta/10\}, \quad (2.21)$$

where ϵ is the machine precision and ϑ is the user-required accuracy. The accuracy of this first-order approximation to the gradient is $O(h)$. The choice of the perturbation (2.21) is motivated by the analysis of Horn [1989].

The possibility to calculate the partial derivatives by the adjoint equations [Brusch and Schapelle 1973; Kraft 1985] has been eliminated because of the inherent inflexibility when the simulation model is changed, which is very often the case within design phases.

3. SOFTWARE ORGANIZATION

The software package TOMP is split into two modules, the simulator d_TOMP and the optimizer SLSQP which exchange their information by reverse communication [Gill et al. 1979]. This means that both are completely independent from each other. The disadvantage of a slightly more calling overhead is more than compensated by a large flexibility. In d_TOMP the initial value problem of Section 2.2.1 is integrated for the nominal and the perturbed control functions, e.g., Equation (2.19). In SLSQP problem (NLP) of Section 1 is solved by sequential linear least squares problems [Kraft 1988] using a slightly modified NNLS of Lawson and Hanson [1974] as core routine.

For a given parameter vector x^k, where x^0 has to be furnished by the user, the simulator calculates values of the cost function $f(x^k)$, its gradient $\nabla f(x^k)$, the constraints $g(x^k)$, and their Jacobi matrix $\nabla g(x^k)$. These values are introduced into the optimizer which either proposes a new parameter vector x^{k+1} or informs the user that certain stopping criteria are satisfied. This simple but efficient and flexible organization is shown in Figure 1.

The dashed box in Figure 1 indicates a possible user interface; here intermediate trajectories can be plotted. Thus one can easily build up an interactive version of the software.

The user has to write three subroutines:

—one in which the right-hand sides of the differential equations (2.7) are implemented,
—another in which the cost function and the constraints (2.10, 2.11, 2.15) are evaluated, and
—a third one in which the output is sampled for trajectories; this may be a subroutine with an empty subroutine body.

The handling of these subroutines will be explained in the example of Section 4.1 and in the Appendix.

4. EXAMPLES

To demonstrate the effectiveness of the solution approach two examples are given: a state-constrained brachistochrone problem and an optimal path-planning approach for an industrial robot.

4.1 State-Constrained Brachistochrone

The first example has often been treated in the literature [Bryson and Ho 1969]. Here it serves for comparison reasons and as an example of how the user subroutines have to be written. The subroutine source listings are given in Appendix A.

4.1.1 *Mathematical Model.* A bead slides on a frictionless wire between two points A and B in a constant gravitational field, g. What is the shape $\theta(t)$ of the wire that will produce a minimum-time path between the two points?

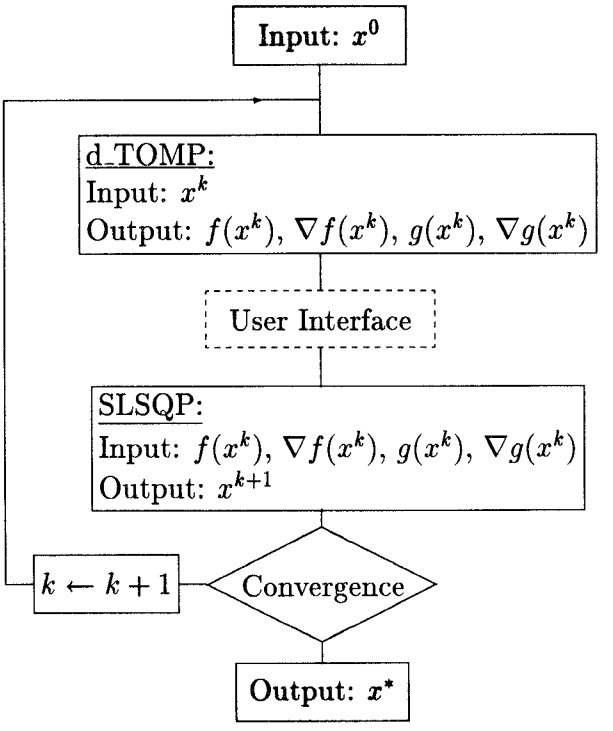

Fig. 1. Organization of TOMP with reverse communication.

The state equations $\dot{z}_i(t) = \phi_i(t, z(t), u(t))$, are

$$\dot{z}_1(t) = z_3(t) \cos \theta(t),$$
$$\dot{z}_2(t) = z_3(t) \sin \theta(t),$$
$$\dot{z}_3(t) = g \sin \theta(t),$$

where $z_1 = x$ is the horizontal distance of the bead, $z_2 = y$ is its vertical distance, and $z_3 = v$ its velocity. All initial conditions are assumed to be zero:

$$z_i(t_A) = 0, \qquad i = 1, \ldots, 3.$$

The final horizontal distance is prescribed

$$z_1(t_B) = 1.$$

Both the other states are free.

The problem is to find a control function $u(t) = \theta(t)$ that minimizes the time to reach B from A

$$f = t_B - t_A.$$

We also solve this problem for the case where the bead has to satisfy the following state inequality constraint

$$c(z(t)) = az_1(t) - z_2(t) \geq b, \qquad \forall t \in [t_A, t_B],$$

with $a = 0.4$ and $b = 0.2$.

4.1.2 *Test Results.*

4.1.2.1 Nominal Case. This example has been solved with TOMP using a high-order Runge-Kutta method with absolute and relative accuracy 10_{-8}. The tolerance of the nonlinear programming code has been set to 10_{-6}. The initial estimates have been $\theta^0(t) = 0$ and $t_B^0 = 2.0$. The upper and lower bounds on $\theta(t)$ have been $\pm 2\pi$, respectively, and ten and zero for t_B. The number of iterations in the unconstrained case was 10, that in the constrained case 23. The results are shown in the graphs of Figures 2 and 3. (The unconstrained trajectory is dotted; the constrained trajectory is dashed.) The constrained trajectory is active within the range $0.30 \leq z_1(t) \leq 0.65$. The minimum time for $t_A = 0$ is $t_B = 1.772$ in the unconstrained case, $t_B = 1.795$ in the constrained case.

4.1.2.2 Convergence Tests. Let this example be taken as a paradigm for the large convergence region of TOMP as claimed in Section 1. The numerical experiment includes three different starting controls for three estimates of the final time. The results are summarized in Table I. The columns include different θ^0-values for a given t_B^0-value. The entries (i, j) in the table are the number of iterations for the constrained problem. All entries converge to the true solution for the data of paragraph 4.1.2.1. If the bounds on θ are expanded to ± 10, the convergence to the global minimum for the case in entry (2,1) breaks down. The solution is at a local minimum and then has a "looping" in its trajectory near the origin. This of course prolongates the final time (to $t_B = 2.013$).

4.2 Optimal Trajectories of a Manutec r3 Robot

The following example treats the calculation of robot trajectories as an optimal control problem. Optimal robot trajectories are compared with reference trajectories for the path-planning problem [Craig 1989; Pfeiffer 1987] or are computed to check the robot performance within its working space [Konzelmann et al. 1989]. The calculations are examplified on a Manutec r3 robot of Siemens, whose mathematical model has been developed by Otter and Türk [1988].

4.2.1 *Mathematical Model.* The mathematical model of a robot with chain structure is a multibody system in the configuration space of the following form

$$M(\theta)\ddot{\theta} = V(\theta, \dot{\theta}) + G(\theta) + T. \tag{4.1}$$

We restrict the model to one of three degrees of freedom. In this case the following notion is valid for Equation (4.1): M is a 3×3 symmetric, positive definite mass matrix; θ is a 3×1 matrix of generalized position coordinates; the θ_i are the relative angles between arm $i - 1$ and arm i; $\dot{\theta}$ and $\ddot{\theta}$ are 3×1 matrices of angle velocity and angle acceleration; V is a 3×1 matrix of the centrifugal and Coriolis terms; G is a 3×1 matrix of the gravitational terms, and T is a 3×1 matrix of the motor moments.

The following assumptions have been made.

—The robot consists of one fixed basis body, three arms and three rotors in which motors and gears are integrated.

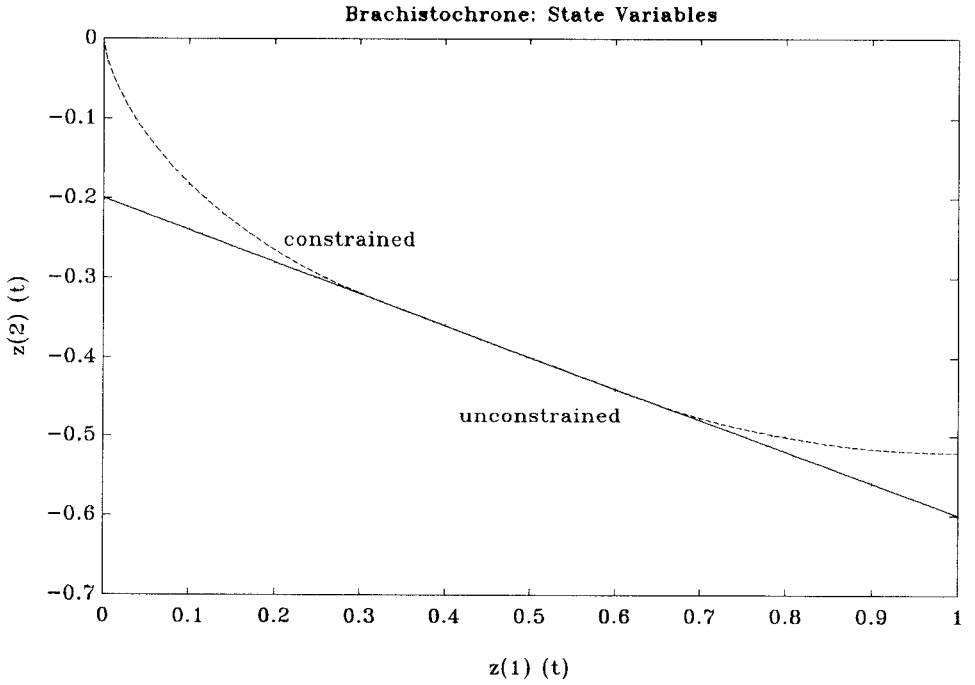

Fig. 2. State constrained brachistochrone: state variables.

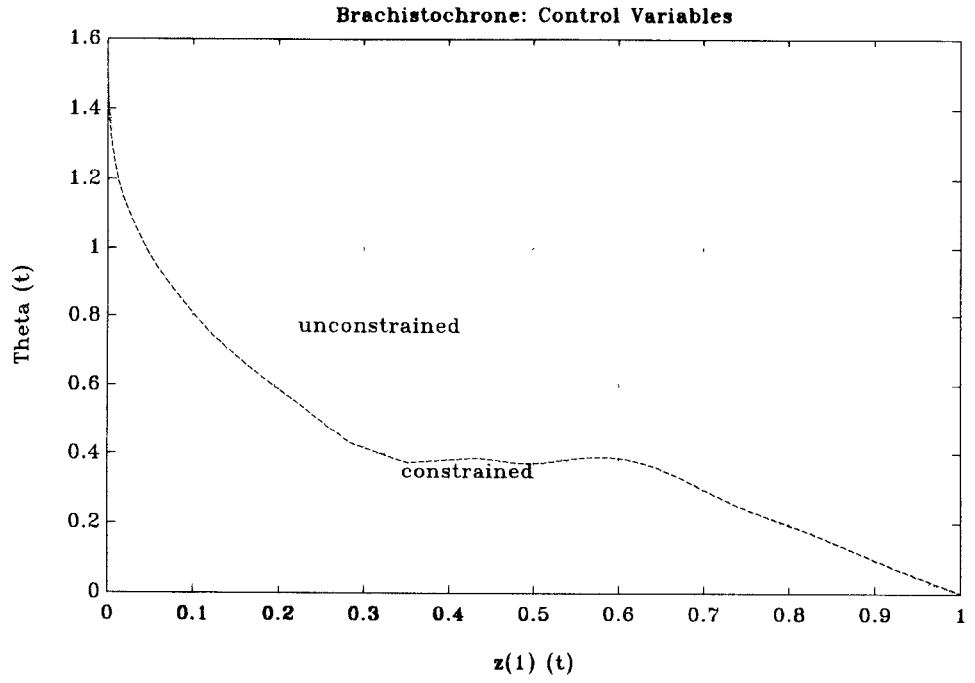

Fig. 3. State constrained brachistochrone: control variables.

Table I. Number of Iterations for Different Estimates

$\theta^0(t)$	$t_B^0 = 1.0$	$t_B^0 = 2.0$	$t_B^0 = 3.0$
$-\pi/2$	49	26	22
0.0	49	23	22
$+\pi/2$	50	22	22

—The arms are nonelastic links which are connected by joints with one rotational degree of freedom.
—The gears are assumed as ideal; their elasticity, friction, and backlash are neglected.
—The dynamics of the motors and gears are also neglected, which means that the input voltages are transformed into motor moments without delay.

Further details as well as all model parameters (robot geometry, masses, moments of inertia, etc.) can be found in Otter and Türk [1988] where also a Fortran procedure `R3M2SI` is described in which for given θ, $\dot{\theta}$, and T the angular accelerations $\ddot{\theta}$ are computed according to the algorithms of Brandl et al. [1986].

The state vector z of the model is composed from angle and angular velocity $z = (\theta, \dot{\theta})$, and the motor moments T is included in the control vector u. The state equations $\dot{z} = f(t, z, u)$ are thus given as follows:

$$\frac{d}{dt}\theta = \dot{\theta}, \tag{4.2}$$

$$\frac{d}{dt}\dot{\theta} = M^{-1}(\theta)[V(\theta, \dot{\theta}) + G(\theta) + T]. \tag{4.3}$$

The boundary values are given at t_0 and t_f:

$$\begin{aligned}
\theta_1(t_0) &= +0.2286, & \dot{\theta}_1(t_0) &= 0,\\
\theta_2(t_0) &= +0.3107, & \dot{\theta}_2(t_0) &= 0,\\
\theta_3(t_0) &= +1.7541, & \dot{\theta}_3(t_0) &= 0,\\
\theta_1(t_f) &= -0.1466, & \dot{\theta}_1(t_f) &= 0,\\
\theta_2(t_f) &= +0.7313, & \dot{\theta}_2(t_f) &= 0,\\
\theta_3(t_f) &= -1.8929, & \dot{\theta}_3(t_f) &= 0.
\end{aligned}$$

These boundary values define a wide-ranging overhead motion. They have been chosen because they can be compared with test results obtained for an industrial controller RCM and another suboptimal path-planning method due to Kempkens [1990].

All state variables as well as all control variables are constrained within the entire interval $t_0 \le t \le t_f$:

$$-2.97 \le \theta_1(t) \le +2.97, \qquad -3.0 \le \dot{\theta}_1(t) \le +3.0,$$
$$-2.01 \le \theta_2(t) \le +2.01, \qquad -1.5 \le \dot{\theta}_2(t) \le +1.5,$$
$$-2.86 \le \theta_3(t) \le +2.86, \qquad -5.2 \le \dot{\theta}_3(t) \le +5.2,$$
$$-945 \le T_1(t) \le +945, \quad -1890 \le T_2(t) \le +1890, \quad -540 \le T_3(t) \le +540,$$

where the index counts from the basis to the exterior. Angles are given in radian and moments in Nm. The robot carries a load L in kg, which takes values from the following domain:

$$0 \le L \le 15.$$

As cost function the final time $f = t_f$ is to be minimized.

Equation (4.3) is only for formal reasons written explicitly. The algorithm from Brandl et al. [1986] avoids the inversion of the mass matrix, and treats the descriptor form (4.1) directly.

The state equations are implemented in the following procedure which is to be given by the user as the actual subroutine for the formal RHS. By the same time it is the interface to the robot model R3M2SI.

```
      SUBROUTINE ROBOT (t, Z, DZ, U, IU)
      DOUBLE PRECISION t, Z(6), DZ(6), U(*)
      DOUBLE PRECISION LOAD, Torque (3)
      INTEGER          IU(*)
      COMMON /CLOAD /  LOAD
      DO 10, I = 1,3
   10 DZ (I) = Z(I + 3)
      CALL CONTRL (t, U, IU, Torque)
      CALL R3M2SI (LOAD, Z, DZ, Torque, DZ(4))
      RETURN
      END
```

4.2.2 *Test Results.* A few results of the simulation are given in Figure 4 for the robot the states of which are not constrained and in Figure 5 for the state-constrained robot. An extensive study is included in Hartl [1991].

The minimal final time of $t_f = 0.599$ seconds for the robot the states of which are unconstrained is determined by the maximal motor moments T_2 and T_3, the latter controlling the robot in a bang-bang-type structure. This behavior can be expected, as the controls T enter the differential equations (4.1) linearly [Bryson and Ho 1969, pp. 110–117]. Note that the maximal angular velocities of the corresponding joints $\dot{\theta}_2$ and $\dot{\theta}_3$ are dissatisfied along a long arc of their trajectories.

The minimal final time of $t_f = 0.848$ seconds for the state-constrained robot is determined by the maximal angular velocity of the third joint $\dot{\theta}_3$. This constraint is active within the time interval $t \in [0.2, 0.7]$ while outside this interval the corresponding motor moment T_3 is at its boundary. Here the angular acceleration is maximal.

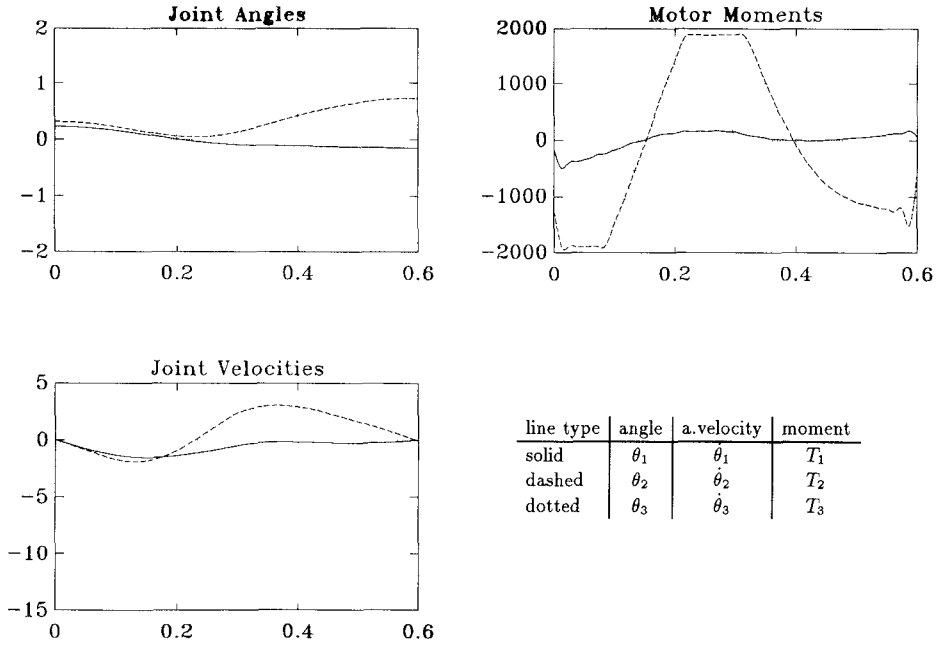

Fig. 4. States and controls for the unconstrained robot $r3$, $L = 0$ kg.

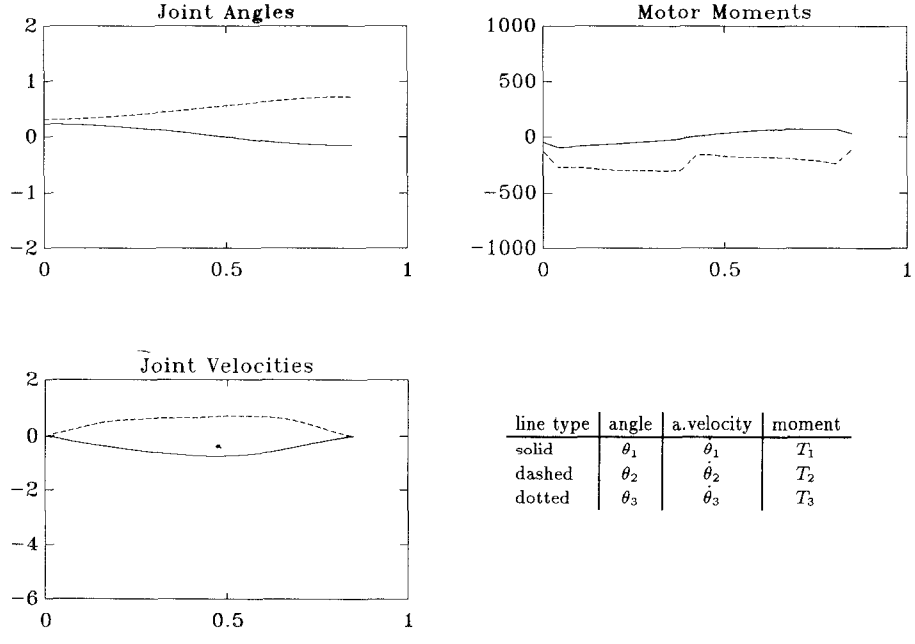

Fig 5. States and controls for the constrained robot $r3$, $L = 0$ kg.

Table II. Minimum Final Times for Different Robot Control

	RCM	Kemp	TOMP
unconstrained	—	1.125	0.599
constrained	1.664	1.280	0.848

Comparing these results with those of Kempkens [1990] in Table II demonstrates drastic time gains of even 100% by optimization which could considerably raise the productivity of modern manipulators.

The given examples demonstrate that the problem of robot path planning in a highly constrained environment can be very efficiently solved by optimal control calculations. Included also are problems of collision avoidance in multirobot configurations where optimal control problems with multipoint boundary values have to be solved [Hartl 1991].

APPENDIX A. USER-SUPPLIED SUBROUTINES

The user has to write three subroutines: RHS, VALUES, OUTPUT, which in the brachistochrone example of Section 4.1 are named BRACH, VBRACH, OBRACH. The last subroutine may contain no executable statements, if the user does not want to plot other trajectories than the states and the controls, which are automatically stored in d_TOMP.

A.1 Main Program

A possible structure of a main program with reverse communication between SLSQP and d_TOMP according to Figure 1 is shown in the following program section.

```
      PROGRAM TOMP
C     DECLARATIONS
      EXTERNAL RK54, BRACH, VBRACH, OBRACH
C          .
C          .
C     INITIAL ESTIMATE OF CONTROL PARAMETERIZATION {IX, X} ETC.
C          .
C          .
   10 CALL D_TOMP (RK54, BRACH, VBRACH, OBRACH, IX, X, TC,
     *             Z, Z0, Z1, TOL, F, C, DF, DC, LC)
C          .
C          .
C     USER INTERFACE
C          .
C          .
      CALL SLSQP (M, MEQ, LA, N, X, XL, XU, F, C, G, A,
     *            ACC, ITER, MODE, W, L_W, JW, L_JW)
      IF (ABS(MODE).EQ.1) GO TO 10
C          .
C          .
C     FINAL OUTPUT, etc.
C          .
C          .
      END
```

A.2 Right-Hand Sides of Differential Constraints

```
      SUBROUTINE BRACH (T,Z,DZ,U,IU)
C     THIS SUBROUTINE COMPUTES THE VALUES OF THE RIGHT HAND SIDES
C     FOR THE BRACHISTOCHRONE
C     IT IS CALLED FROM SUBROUTINE SLV WHICH IS CALLED FROM D_TOMP


C     INPUT:
C     ------
C     T TIME, DOUBLE PRECISION SCALAR
C     Z STATE, DOUBLE PRECISION VECTOR
C     U CONTROL, DOUBLE PRECISION VECTOR
C     IU CONTROL, INTEGER VECTOR
C     THE PAIR {U,IU} REPRESENTS THE CONTROL PARAMETERIZATION
C     FOR FURTHER INFORMATION LOOK AT THE PARAMETERS OF D_TOMP


C     OUTPUT:
C     -------
C     DZ STATE DERIVATIVE, DOUBLE PRECISION VECTOR

      INTEGER          IU(20)
      DOUBLE PRECISION T, Z(*), DZ(*), U(*)
      DOUBLE PRECISION G, THETA(1), TF
      DATA             G/1.0D0/

      NC=IU(01)
      NP=IU(13)
      TF=U(NC+NP)
      CALL CONTRL(T,U,IU,THETA)
      DZ(1)=TF*Z(3)*COS(THETA(1))
      DZ(2)=TF*Z(3)*SIN(THETA(1))
      DZ(3)=TF*G*SIN(THETA(1))
      END
```

A.3 Cost Function and Constraints

```
      SUBROUTINE VBRACH (RHS,IU,T,U,TC,Z,Z0,Z1,W,F,C,MODE)
C     THIS SUBROUTINE COMPUTES THE VALUES OF THE COST FUNCTION AND
C     CONSTRAINTS FOR THE BRACHISTOCHRONE
C     VBRACH IS CALLED AS SUBROUTINE VALUES IN D_TOMP
C     MAY ALSO BE CALLED IN THE USER INTERFACE
```

A.3 Cost Function and Constraints—*Continued*

```
C     INPUT:
C     ------
C     RHS SUBROUTINE IN WHICH THE RIGHT HAND SIDES OF THE DIFFERENTIAL
C         CONSTRAINTS ARE CALCULATED (NOT USED IN OUR CASE)
C     T TIME, DOUBLE PRECISION SCALAR
C     TC TIME GRID FOR COMMUNICATION, DOUBLE PRECISION VECTOR
C     Z STATE, DOUBLE PRECISION VECTOR
C     U CONTROL, DOUBLE PRECISION VECTOR
C     IU CONTROL, INTEGER VECTOR
C        THE PAIR {U,IU} REPRESENTS THE CONTROL PARAMETERIZATION
C        FOR FURTHER INFORMATION LOOK AT THE PARAMETERS OF D_TOMP
C     MODE INDICATES STATUS OF INTEGRATION, INTEGER
C     MODE=0: SET INITIAL CONDITIONS
C     MODE=NT: EVALUATE BOUNDARY VALUES
C     0<MODE<NT: EVALUATE STATE AND CONTROL CONSTRAINTS


C     OUTPUT:
C     -------
C     Z0 INITIAL STATE, DOUBLE PRECISION VECTOR
C     Z1 FINAL STATE, DOUBLE PRECISION VECTOR
C     F COST FUNCTION, DOUBLE PRECISION SCALAR
C     C CONSTRAINTS, DOUBLE PRECISION VECTOR
C       EQUALITY CONSTRAINTS (BOUNDARY VALUES) MUST BE SET FIRST


C     INNPUT/OUTPUT:
C     --------------
C     W WORKING ARRAY, DOUBLE PRECISION VECTOR


      INTEGER          IU(20),MODE
      DOUBLE PRECISION U(*),T,TC(*),Z0(*),Z1(*),Z(*),W(*),F,C(*)
      EXTERNAL         RHS
      SAVE             J

      IF (MODE .EQ. 0) THEN
```

A.3 Cost Function and Constraints—*Continued*

```
C SET SOME PARAMETERS

          NC=IU(01)
          NP=IU(13)
          NT=IU(15)
          NZ=IU(16)
          M =IU(17)
          ME=1
          J=ME


C BOUNDARY VALUES

          Z0(1)=0.0D0
          Z0(2)=0.0D0
          Z0(3)=0.0D0

          Z1(1)=1.0D0
          Z1(2)=0.0D0
          Z1(3)=0.0D0

      END IF


C STATE CONSTRAINTS

      IF (M.NE.ME) THEN
          IF (MODE.GT.1 .AND. MODE.LT.NT) THEN
             J=J+1
             C(J)=0.2D0+0.4D0*Z(1)-Z(2)
          END IF
      END IF


C COST AND FINAL STATES

      IF (MODE.EQ.NT) THEN

          F=U(NC+NP)
          C(1)=Z(1)-Z1(1)

      END IF
      END
```

A.4 Output Routine

```
      SUBROUTINE OBRACH(T,INDEX,G,LONG,NO,Z,U,IU)
C     OUTPUT ROUTINE IS CALLED AS SUBROUTINE OUTPUT FROM D_TOMP

C     INPUT:
C     ------
C     T TIME, DOUBLE PRECISION SCALAR
C     K INTEGRATION INDEX SET BY D_TOMP BEGINNING FROM 1,
C       K IS SET TO K+1, EACH TIME OBRACH IS CALLED, INTEGER
C     G REAL ARRAY OF DIMENSION (LONG, *)
C       STORES COLUMNWISE STATES AND CONTROLS
C     Z STATE, DOUBLE PRECISION VECTOR
C     U CONTROL, DOUBLE PRECISION VECTOR
C     IU CONTROL, INTEGER VECTOR
C        THE PAIR {U,IU} REPRESENTS THE CONTROL PARAMETERIZATION
C        FOR FURTHER INFORMATION LOOK AT THE PARAMETERS OF D_TOMP

C     OUTPUT:
C     -------
C     G THE USER MAY ADD NO-(NU+NZ) COLUMNS TO G
C       WHERE NU=IU(1) AND NZ=IU(16)

      INTEGER          K,LONG,NO,IU(20)
      DOUBLE PRECISION T,Z(*),U(*),THETA(1)
      REAL             G(LONG,*)
C           .
C           .
C USER'S OUTPUT, E.G.
*     CALL CONTRL(T,U,IU,THETA)
*     WRITE(*,'(8(1PD10.2))') T, (Z(I),I=1,3), THETA(1)
C           .
C           .
      END
```

REFERENCES

ASCHER, U., CHRISTIANSEN, J., AND RUSSEL, R. D. 1981. Collocation software for boundary-value odes. *ACM Trans. Math. Softw. 7,* 2 (June), 209–222.

BRANDL, H., JOHANNI, R., AND OTTER, M. 1986. A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. In *Proceedings of IFAC / IFIP / IMACS International Symposium on Theory of Robots.*

BRUSCH, R. G., AND SCHAPELLE, R. H. 1973. Solution of highly constrained optimal control problems using nonlinear programming. *AIAA J. 11,* 135–136.

BRYSON, A. E., AND HO, Y. C. 1969. *Applied Optimal Control.* Ginn and Company, Waltham, Mass.

BULIRSCH, R. 1971. Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Report of Carl-Cranz-Gesellschaft. Oberpfaffenhofen.

CRAIG, J. J. 1989. *Robotics—Mechanics and Control.* Addison-Wesley, Reading, Mass

FEHLBERG, E. 1969. Klassische Runge-Kutta Formeln fünfter und siebter Ordnung mit Schrittweiten-Kontrolle. *Computing 4,* 93–106.

GILL, P. E., MURRAY, W., PICKEN, S. M., AND WRIGHT, M. H. 1979. The design and structure of a Fortran program library for optimization *ACM Trans. Math. Softw. 5,* 259–283.

GILL, P E., MURRAY, W., SAUNDERS, M., AND WRIGHT, M. H. 1983. User's Guide for SOL/NPSOL: A Fortran Package for Nonlinear Programming. Report SOL 83-12, Systems Optimization Laboratory, Stanford Univ., Stanford, Calif

GILL, P. E., MURRAY, W., AND WRIGHT, M. H. 1991 *Numerical Linear Algebra and Optimization* Addison-Wesley, Reading, Mass.

GRIMM, W., BERGER, E., OBERLE, H.-J. 1985. Benutzeranleitung für das Rechenprogramm BNDSCO zur Lösung beschränkter optimaler Steuerungsprobleme. DFVLR-Mitt. 85-05, Koln.

HARTL, M. 1991. Numerical and graphical evaluation of the performance of the Manutec r3 robot within its work space. Diploma Thesis, Fachhochschule München, Dept. of Mechanical Engineering, München, Germany

HORN, M. K. 1989. Solution of the optimal control problem using the software package STOMP. In the *8th IFAC Workshop on Control Applications of Nonlinear Programming and Optimization.*

KAMM, J. L., AND JOHNSON, I. L. 1973. Nonlinear programming approach for optimizing two-stage lifting vehicle ascent to orbit. *Automatica 9,* 713–720.

KELLER, H. B. 1976. *Numerical Solution of Two Point Boundary Value Problems.* SIAM, Philadelphia, Pa.

KEMPKENS, K. 1990. Optimierung der Verfahrparameter von Industrieroboter-Steuerungen *Robotersysteme 6,* 145–529

KONZELMANNN, J , BOCK, H. G., AND LONGMAN, R. W. 1989. Time optimal trajectories of elbow robots by direct methods. In *Proceedings of the AIAA Guidance, Navigation and Control Conference.*

KRAFT, D. 1991 TOMP—FORTRAN Modules for Optimal Control Calculations. VDI-Fortschrittsbericht Nr 254, VDI-Verlag, Düsseldorf, Germany.

KRAFT, D. 1988. A Software Package for Sequential Quadratic Programming. DFVLR-FB 88-28, Koln, Germany

KRAFT, D. 1987. Nonlinear system analysis by direct collocation. In *Optimal Control.* R. Bulirsch, A. Miele, J. Stoer, K. H. Well, Eds. Lecture Notes in Control and Information Sciences, 95 Springer, Berlin.

KRAFT, D. 1985. On converting optimal control problems into nonlinear programming problems. In *Computational Mathematical Programming.* K. Schittkowski, Ed. Springer, Berlin.

LAWSON, C. L., AND HANSON, R. J. 1974. *Solving Least Squares Problems* Prentice-Hall, Englewood Cliffs, N.J.

LENTINI, M., AND PEREYRA, V. 1977. An adaptive finite difference solver for nonlinear two point boundary value problems with mild boundary layers. *SIAM J. Numer. Anal 14,* 91–111.

OTTER, M., AND TÜRK, S. 1988 The DFVLR Models 1 and 2 of the Manutec r3 Robot. DFVLR-Mitt. 88-13, Köln, Germany.

PFEIFFER, F., AND REITHMEIER, E. 1987. *Roboterdynamik.* Teubner, Stuttgart.

PONTRJAGIN, L. S., BOLTJANSKIJ, V. G., GAMKRELIDZE, R. V , AND MISCENKO, E. F. 1964. *Mathematische Theorie optimaler Prozesse.* Oldenbourgh, München, Germany.

POWELL, M. J. D. 1978. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical Analysis,* G. A. Watson, Ed. Lecture Notes in Mathematics, vol. 630. Springer, Berlin.

PRINCE, P. J , AND DORMAND, J. R. 1981. High order embedded Runge-Kutta formulae. *J. Comput. Appl. Math. 7,* 67–75.

SCHITTKOWSKI, K. 1986. NLPQL: A Fortran subroutine solving constrained nonlinear programming problems. *Ann Oper. Res. 5,* 485–500.

SCHÖPF, R., AND DEUFLHARD, P. 1994. OCCAL, a mixed symbolic-numerical optimal control calculator. In *Computational Optimal Control,* R. Bulirsch and D Kraft, Eds Birkhäuser, Basel, Switzerland.

SHAMPINE, L. F., AND WATTS, H. A. 1976. Practical solution of ordinary differential equations by Runge-Kutta methods. Sandia Laboratories Report SAND 76-0585, Albuquerque, N. Mex.

VON STRYK, O. 1993. Numerical solution of optimal control problems by direct collocation. In *Optimal Control and Variational Calculus*, R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, Eds. Birkhäuser, Basel, Germany.

VON STRYK, O., AND BULIRSCH, R. 1992. Direct and indirect methods for trajectory optimization. *Ann. Oper. Res. 37*, 357–373.

TABAK, D., AND KUO, B. C. 1971. *Optimal Control by Mathematical Programming*. Prentice-Hall, Englewood Cliffs, N.J.

Received December 1991; revised June 1993; accepted June 1993