Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Journal of Computational and Applied Mathematics 185 (2006) 404–421

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Symbolic-numeric efficient solution of optimal control problems for multibody systems

Enrico Bertolazzi*, Francesco Biral, Mauro Da Lio

Dipartimento di Ingegneria Meccanica e Strutturale, Università di Trento, via Mesiano 77, I-38050 Trento, Italy

Received 26 March 2003

Abstract

This paper presents an efficient symbolic-numerical approach for generating and solving the boundary value problem-differential algebraic equation (BVP-DAE) originating from the variational form of the optimal control problem (OCP). This paper presents the method for the symbolic derivation, by means of symbolic manipulation software (Maple), of the equations of the OCP applied to a generic multibody system. The constrained problem is transformed into a nonconstrained problem, by means of the Lagrange multipliers and penalty functions. From the first variation of the nonconstrained problem a BVP-DAE is obtained, and the finite difference discretization yields a nonlinear systems. For the numerical solution of the nonlinear system a damped Newton scheme is used. The sparse and structured Jacobians is quickly inverted by exploiting the sparsity pattern in the solution strategy. The proposed method is implemented in an object oriented fashion, and coded in C++ language. Efficiency is ensured in core routines by using Lapack and Blas for linear algebra.
© 2005 Elsevier B.V. All rights reserved.

MSC: 34Bxx; 34H05; 65Fxx; 65Hxx; 65Lxx

Keywords: Optimal control; Indirect methods; Boundary value problems; Newton–Raphson

1. Introduction

In the last decade the interest on optimal control problems (OCP) grew rapidly for its ability to produce control laws, which operates a system in an optimal way according to given aims. The applications cover

* Corresponding author.
E-mail address: enrico.bertolazzi@unitn.it (E. Bertolazzi).

many engineering fields such as chemical, economics, vehicle and biomechanics dynamics, automation, and many others [5,6,10,18,19,21,23,24,29]. At the same time the improvement of available symbolic and numerical multibody software helped researchers to develop complex mathematical models of the system to be controlled. As main consequence, nowadays engineering problems yield very large systems of differential algebraic equations (DAE). Since such large number of equations being involved, the availability of an automatic procedure for deriving OCPs equations and the speed of numerical computation have become key issues. The literature review suggests that basically there are two ways to solve these kinds of problems: the direct methods and the indirect ones. The dynamic programming is ruled out being impracticable for its high dimensionality when tackling large systems.

Indirect methods. The theoretical basis of the solution of an OCP (as expressed in Section 2) comes from the calculus of variations in the form of first necessary condition for the Lagrangian function stationary condition [27]. The first necessary condition consists of the original problem equations and a new set of equations called adjoint equations whose number is equal to the problem variable ones. In addition there are the boundary condition equations. The whole set of equations (presented in Section 2.1) yields a two point boundary value problem (TPBVP), which can be numerically solved mainly using four different techniques. Invariant embedding [26] is a procedure for converting the TPBVP to an IVP. The main disadvantage is the high dimensionality of the resulting problem. Single shooting [3] technique guess missing initial conditions and the equations are integrated as IVP. By means of Newton iterations missing initial conditions are updated until the given final conditions are met. The main disadvantage of this method is its high sensitivity to guessed values and the fact that some equations (typically the co-state equations) may be very-fast diverging. Multiple shooting [4,14,25] follows the same idea of single shooting, but the domain is divided in smaller subintervals reducing sensitivity to guessed values and divergence specially when combined with direct collocation. The Newton iteration is used to enforce continuity between subintervals.

Discretization or global methods [3,10] are the most stable, and the solution is obtained simultaneously for the whole domain. Efficient factorization schemes [17,20] and structural orthogonal factorization [30] can be used to minimize the computational effort.

The solution of TPBVP arising from the necessary condition is commonly referred as indirect method. Since it solves directly the equation of first necessary condition is generally reckoned as very accurate. However, the number of equations is at least doubled, moreover such a procedure requires a massive symbolic computation to obtain the adjoint equations.

Direct methods. Instead of solving directly the necessary condition it is possible to discretize the problem in order to obtain a nonlinear program (NLP), which makes possible to use consolidated numerical techniques, such as initial value solver (IVS) and sequential quadratic programming (SQP) [2,5,16,26,23,28]. With full discretization, both control variables and state variables are discretized. This leads to a very large system, which consists of algebraic objective function and a set of algebraic constraints which can be optimized by any NLP solver The weakness are the large problem size, and the need for efficient large scale optimization algorithm. On the other hand if only controls are parameterized (partial discretization), given the initial conditions the process equations are solved with a DAE solver. This produces the value of the objective function, to be iteratively used in a optimization routine to find the optimal parameters in the control parametrization. The Jacobian is needed with a high accuracy. Moreover, both method require post calculation to check validity of first necessary condition.

Under some hypothesis, both direct and indirect methods theoretically produce the same results, but the indirect ones are more accurate [28,5,21,4]. Nevertheless, the direct methods achieved more popularity because, among other reasons, they make use of consolidated numerical techniques such as IVS and SQP and, above all, do not require the symbolic derivation of the adjoint equations. Only their estimation is needed in some cases (especially in post processing task), see the reviews by Cervantes [7] and Sargent [22]. This aspect might be an advantage when large systems are involved, because systems dimension is halved. However, accuracy and sensibility to design parameter variations, which is typical of indirect methods, might be essential in some engineering problems for example as race engineering, where a variation of the position of centre of mass of few millimeters may change substantially the global performance of the vehicle [11,8].

The intent of this paper is to present a general methodology that implements a full indirect method for OCP applied to large system of ordinary differential equations (ODE). In Section 2 the equation describing the minimum time problem to be solved are introduced. The equations of the necessary condition as well as the adjoint ones are automatically derived via symbolic derivation. In Section 3 the TPBVP arising from indirect method obtaining a large nonlinear system are discretized. In Section 4 a solution procedure for the large nonlinear system is presented. In Section 5 the method is applied to solve a minimum lap-time for a mathematical model of a racing motorcycle on a real circuit. The numerical results are compared to the data of a real vehicle acquired from onboard measurement systems.

2. The optimal control problem formulation

The engineering application we are considering is *the minimum lap-time* for a real circuit performed by a motorcycle. The mathematical model of the driver-motorcycle system is an improved version of the one presented in Cossalter et al. [10]. The variational formulation of this OCP consists in finding the control $\mathbf{u}(s) \in \mathbb{R}^m$ that minimizes the functional:

$$J[\mathbf{x}, \mathbf{u}] = \int_{s_i}^{s_f} f(\mathbf{x}(s), \mathbf{u}(s))\, \mathrm{d}s, \tag{1}$$

under differential and algebraic constraints. Here, the symbol $\mathbf{x}(s) \in \mathbb{R}^n$ denotes the state vector and s is the curvilinear abscissa of the circuit. The minimum is assumed to satisfy the following ODE (or DAE):

$$\mathbf{a}(\mathbf{x}(s), \dot{\mathbf{x}}(s), \mathbf{u}(s)) = \mathbf{0}, \quad s \in (s_i, s_f), \tag{2}$$

where $\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}) = (a_i(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}))_{i=1}^{n}$.

Algebraic constraints are divided in two groups, equality and inequality ones. These equality constraints are pointwise and are used to set initial and final boundary conditions as follows:

$$\mathbf{c}(\mathbf{x}(s_i), \mathbf{x}(s_f)) = \mathbf{0}, \tag{3}$$

where $\mathbf{c}(\mathbf{x}_i, \mathbf{x}_f) = (c_i(\mathbf{x}_i, \mathbf{x}_f))_{i=1}^{p}$. Inequality constraints are set along the trajectory and are written as follows:

$$\mathbf{d}(\mathbf{x}(s), \mathbf{u}(s)) \leqslant \mathbf{0}, \quad s \in (s_i, s_f), \tag{4}$$

where $\mathbf{d}(\mathbf{x}, \mathbf{u}) = (d_i(\mathbf{x}, \mathbf{u}))_{i=1}^{q}$. These inequalities are used to describe the domain of the solution and limitation of the control variable $\mathbf{u}(s)$. The specific expressions of Eqs. (1)–(4) can be found in appendix.

2.1. Constraints eliminations

The OCP based on Eqs. (1)–(4) is formulated in a constrained variational way. Inequality constraints are eliminated by means of penalty functions [22]. This is done by augmenting the original cost functional J with some penalty functions that increase sharply if the inequality constraints are violated. Differential and equality constraints can be eliminated by introducing a set of suitable Lagrangian multipliers, thus resulting in an unconstrained formulation.

This formulation is possible at the cost of n more unknowns for the elimination of the differential constraints an p more unknowns for the eliminations of the equality constraints.

The resulting functional is the following:

$$\tilde{J}[\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}] = \boldsymbol{\mu} \cdot \mathbf{c}(\mathbf{x}(s_i), \mathbf{x}(s_f)) + \int_{s_i}^{s_f} [f_p(\mathbf{x}(s), \mathbf{u}(s)) + \boldsymbol{\lambda}(s) \cdot \mathbf{a}(\mathbf{x}(s), \dot{\mathbf{x}}(s), \mathbf{u}(s))] \, \mathrm{d}s, \tag{5}$$

where

$$f_p(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u}) + \sum_{i=1}^{q} p_i(d_i(\mathbf{x}, \mathbf{u})),$$

and p_i is the penalty associated to the ith components of inequalities (4) and takes the form:

$$p_i(s) = \begin{cases} \left(1 + \dfrac{s}{h_i}\right)^{n_i} & \text{if } s > -h_i, \\ 0 & \text{otherwise,} \end{cases}$$

where h_i and n_i are the parameters used to tune the sharpness of the penalty. Typically $h_i = 0.1$ and $n_i = 10$. This kind of penalty results to be easy to use and to tune, in fact the thickness parameter h_i is used when the penalty becomes active and the sharpness parameter n_i is used to tune the penalty factor if the inequality is violated. Inequality constraints can also be eliminated by other techniques such as slack variables as barrier functions [22].

2.2. Optimality necessary conditions

Taking the first variation of (5) and setting it to zero, we obtain the following boundary value problem (BVP):

$$\frac{\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \mathbf{x}} - \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \dot{\mathbf{x}}}\right) = \mathbf{0}, \tag{6A}$$

$$\frac{\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \boldsymbol{\lambda}} = \mathbf{0}, \tag{6B}$$

$$\frac{\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \mathbf{u}} = \mathbf{0}, \tag{6C}$$

$$\frac{\partial e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu})^{\mathrm{T}}}{\partial \mathbf{x}(s_f)} + \frac{\partial d(\mathbf{x}(s_f), \dot{\mathbf{x}}(s_f), \boldsymbol{\lambda}(s_f), \mathbf{u}(s_f))}{\partial \dot{\mathbf{x}}(s_f)} = \mathbf{0}, \tag{6D}$$

$$\frac{\partial e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu})^{\mathrm{T}}}{\partial \mathbf{x}(s_i)} - \frac{\partial d(\mathbf{x}(s_i), \dot{\mathbf{x}}(s_i), \boldsymbol{\lambda}(s_i), \mathbf{u}(s_i))}{\partial \dot{\mathbf{x}}(s_i)} = \mathbf{0}, \tag{6E}$$

$$\frac{\partial e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu})^{\mathrm{T}}}{\partial \boldsymbol{\mu}} = \mathbf{0}, \tag{6F}$$

where

$$d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u}) = f_p(\mathbf{x}, \mathbf{u}) + \sum_k a_k(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})\lambda_k,$$

$$e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu}) = \sum_k c_k(\mathbf{x}(s_i), \mathbf{x}(s_f))\mu_k.$$

Eqs. (6A)–(6F) constitute a BVP with second-order differential equation due to Eqs. (6A). A further simplification is possible by specifying better the form of the differential constraint $\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ when it is derived from a multibody modeling.

Assumption 1. The differential constraint (2) is linear in $\dot{\mathbf{x}}$, i.e.:

$$\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}) = \mathbf{A}(\mathbf{x})\dot{\mathbf{x}} + \mathbf{b}(\mathbf{x}, \mathbf{u}),$$

where $\mathbf{A}(\mathbf{x}) = (A_{ij}(\mathbf{x}))_{i,j=1}^{n}$ and $\mathbf{b}(\mathbf{x}, \mathbf{u}) = (b_i(\mathbf{x}, \mathbf{u}))_{i=1}^{n}$.

Assumption 1 is true for the application presented in this paper. This is typical of multibody system with holonomic constraints [15]. Moreover if $\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ is not linear in $\dot{\mathbf{x}}$ but $\dot{\mathbf{x}}$ appears as polynomials then by adding more equations we can write another $\widetilde{\mathbf{a}}(\widetilde{\mathbf{x}}, \widetilde{\dot{\mathbf{x}}}, \mathbf{u})$ which is linear in $\widetilde{\dot{\mathbf{x}}}$. This results in a differential-algebraic system, i.e. the matrix $\mathbf{A}(\mathbf{x})$ is not necessarily of full rank.

From Assumption 1 it follows that the derivative terms $\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})/\partial\dot{\mathbf{x}}$ and $\partial d(\mathbf{x}, \dot{\mathbf{x}}, \boldsymbol{\lambda}, \mathbf{u})/\partial\mathbf{u}$ do not depend on $\dot{\mathbf{x}}$. This fact allows a simplification of Eqs. (6A)–(6F) because (6A) results a first-order ODE and (6C) becomes a system of algebraic equations. Eqs. (6A)–(6F) can be reformulated as the following boundary value problem-differential algebraic system (BVP-DAE):

$$\frac{\partial \tilde{d}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \mathbf{x}} + \mathbf{T}(\mathbf{x}, \boldsymbol{\lambda})^{\mathrm{T}}\dot{\mathbf{x}} - \mathbf{A}(\mathbf{x})^{\mathrm{T}}\dot{\boldsymbol{\lambda}} = \mathbf{0}, \tag{7A}$$

$$\mathbf{A}(\mathbf{x})\dot{\mathbf{x}} + \mathbf{b}(\mathbf{x}, \mathbf{u}) = \mathbf{0}, \tag{7B}$$

$$\frac{\partial \tilde{d}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u})^{\mathrm{T}}}{\partial \mathbf{u}} = \mathbf{0}, \tag{7C}$$

$$\frac{\partial e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu})^{\mathrm{T}}}{\partial \mathbf{x}(s_f)} + \omega(\mathbf{x}(s_f), \boldsymbol{\lambda}(s_f)) = \mathbf{0}, \tag{7D}$$

$$\frac{\partial e(\mathbf{x}(s_i), \mathbf{x}(s_f), \boldsymbol{\mu})^{\mathrm{T}}}{\partial \mathbf{x}(s_i)} - \omega(\mathbf{x}(s_i), \boldsymbol{\lambda}(s_i)) = \mathbf{0}, \tag{7E}$$

$$\mathbf{c}(\mathbf{x}(s_i), \mathbf{x}(s_f)) = \mathbf{0}, \tag{7F}$$

where

$$\tilde{d}(\mathbf{x},\boldsymbol{\lambda},\mathbf{u}) = f_p(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{b}(\mathbf{x},\mathbf{u}), \quad \omega(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{A}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\lambda},$$

$$\mathbf{T}(\mathbf{x},\boldsymbol{\lambda}) = \frac{\partial\omega(\mathbf{x},\boldsymbol{\lambda})}{\partial\mathbf{x}} - \frac{\partial\omega(\mathbf{x},\boldsymbol{\lambda})^{\mathrm{T}}}{\partial\mathbf{x}}.$$

System (7A)–(7F) can be written in a vector form as follows

$$\mathbf{M}(\mathbf{y})\dot{\mathbf{y}} + \mathbf{n}(\mathbf{y},\mathbf{u}) = \mathbf{0}, \tag{8A}$$

$$\mathbf{h}(\mathbf{y}(s_i),\mathbf{y}(s_f),\boldsymbol{\mu}) = \mathbf{0}, \tag{8B}$$

$$\mathbf{g}(\mathbf{y},\mathbf{u}) = \mathbf{0}, \tag{8C}$$

where (8A) is the vector form of the first-order differential equations (7A)–(7B), Eqs. (8C) is the vector form of boundary conditions (7D)–(7F) and Eqs. (8B) is the vector form of algebraic equations (7C). Moreover,

$$\mathbf{y} = \begin{pmatrix}\mathbf{x}\\ \boldsymbol{\lambda}\end{pmatrix}, \quad \mathbf{M}(\mathbf{y}) = \begin{pmatrix}\mathbf{T}(\mathbf{x},\boldsymbol{\lambda})^{\mathrm{T}} & -\mathbf{A}(\mathbf{x})^{\mathrm{T}}\\ \mathbf{A}(\mathbf{x}) & \mathbf{0}\end{pmatrix}, \tag{9}$$

and

$$\mathbf{n}(\mathbf{y},\mathbf{u}) = \begin{pmatrix}\dfrac{\partial\tilde{d}(\mathbf{x},\boldsymbol{\lambda},\mathbf{u})^{\mathrm{T}}}{\partial\mathbf{x}}\\[2ex] \dfrac{\partial\tilde{d}(\mathbf{x},\boldsymbol{\lambda},\mathbf{u})^{\mathrm{T}}}{\partial\boldsymbol{\lambda}},\end{pmatrix}, \quad \mathbf{g}(\mathbf{y},\mathbf{u}) = \frac{\partial\tilde{d}(\mathbf{x},\boldsymbol{\lambda},\mathbf{u})}{\partial\mathbf{u}}, \tag{10}$$

and

$$\mathbf{h}(\mathbf{y}(s_i),\mathbf{y}(s_f),\boldsymbol{\mu}) = \begin{pmatrix}\dfrac{\partial\mathbf{c}(\mathbf{x}(s_i),\mathbf{x}(s_f))^{\mathrm{T}}}{\partial\mathbf{x}(s_i)}\boldsymbol{\mu} - \mathbf{A}(\mathbf{x}(s_i))^{\mathrm{T}}\boldsymbol{\lambda}(s_i)\\[2ex] \dfrac{\partial\mathbf{c}(\mathbf{x}(s_i),\mathbf{x}(s_i))^{\mathrm{T}}}{\partial\mathbf{x}(s_f)}\boldsymbol{\mu} + \mathbf{A}(\mathbf{x}(s_f))^{\mathrm{T}}\boldsymbol{\lambda}(s_f)\\ \mathbf{c}(\mathbf{x}(s_i),\mathbf{x}(s_f))\end{pmatrix}. \tag{11}$$

Although expressions (9)–(11) are extremely complex they can be easily generated by using symbolic manipulation software. In particular in this work a single MAPLE sheet is used to generate all the expressions and relative Jacobian matrices needed by the solution algorithm.

3. Discretization of the BVP-DAE

The algebraic-differential system (8A)–(8C) is discretized obtaining a finite dimensional algebraic problem. The interval $[s_i, s_f]$ is split into N subintervals, not necessarily of the same size, $s_i = s_0 < s_1 < \cdots < s_N = s_f$. Eqs. (8C) are evaluated on the collocation node $s_{k+(1/2)} = (s_k + s_{k+1})/2$, the midpoint of the interval $[s_k, s_{k+1}]$:

$$\mathbf{0} = \mathbf{g}(\mathbf{y}(s_{k+(1/2)}),\mathbf{u}(s_{k+(1/2)})) = \mathbf{g}(\tfrac{1}{2}(\mathbf{y}(s_{k+1}) + \mathbf{y}(s_k)),\mathbf{u}(s_{k+(1/2)})) + \mathcal{O}(h^2), \tag{12}$$

where $h = \max\{s_{k+1} - s_k;\, k = 0, \ldots, N-1\}$. Eqs. (8A) are approximated by using the midpoint quadrature rule to average on $[s_k, s_{k+1}]$ and by using finite differences in place of the derivative terms:

$$\begin{aligned}\mathbf{0} &= \frac{1}{s_{k+1} - s_k} \int_{s_k}^{s_{k+1}} \mathbf{M}(\mathbf{y}(s))\dot{\mathbf{y}}(s) + \mathbf{n}(\mathbf{y}(s), \mathbf{u}(s))\, \mathrm{d}s \\ &= \mathbf{M}\left(\frac{1}{2}(\mathbf{y}(s_{k+1}) + \mathbf{y}(s_k))\right) \frac{\mathbf{y}(s_{k+1}) - \mathbf{y}(s_k)}{s_{k+1} - s_k} \\ &\quad + \mathbf{n}\left(\frac{1}{2}(\mathbf{y}(s_{k+1}) + \mathbf{y}(s_k)), \mathbf{u}(s_{k+(1/2)})\right) + \mathcal{O}(h^2). \end{aligned} \tag{13}$$

Using (13) and (12), and neglecting the truncation term of order $\mathcal{O}(h^2)$ we obtain the following nonlinear system:

$$\mathbf{\Phi}(\mathbf{W}) = \mathbf{0}, \quad \mathbf{W} = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N, \boldsymbol{\mu}, \mathbf{u}_{1/2}, \mathbf{u}_{1+(1/2)}, \ldots, \mathbf{u}_{N-(1/2)})^{\mathrm{T}}, \tag{14}$$

where $\mathbf{y}_k \approx \mathbf{y}(s_k)$ and $\mathbf{u}_{k+(1/2)} \approx \mathbf{u}(s_{k+(1/2)})$ and the map $\mathbf{\Phi}$ is defined as follows:

$$\Phi_k(\mathbf{W}) = \begin{cases} \mathbf{M}(\frac{1}{2}(\mathbf{y}_{k+1} + \mathbf{y}_k)) \dfrac{\mathbf{y}_{k+1} - \mathbf{y}_k}{s_{k+1} - s_k} + \mathbf{n}(\frac{1}{2}(\mathbf{y}_{k+1} + \mathbf{y}_k), \mathbf{u}_{k+(1/2)}), & k = 0 \ldots N-1, \\ \mathbf{h}(\mathbf{y}_0, \mathbf{y}_N, \boldsymbol{\mu}), & k = N, \\ \mathbf{g}(\frac{1}{2}(\mathbf{y}_{k-N} + \mathbf{y}_{k-N-1}), \mathbf{u}_{k-N-(1/2)}) & k = N+1, \ldots, 2N. \end{cases}$$

The solution of this nonlinear system is a formally first-order accurate approximation of the BVP-DAE (8A)–(8C). First order is low and is relatively easy to produce higher-order solution. Unfortunately higher-order schemes produce a nonlinear system with a more complex structure than system (14). For the application presented in Section 5 the discrete solution agrees with experiments so that the scheme can be considered adequate for such application. However, if accuracy is an issue second- or higher-order schemes can be easily constructed. The price to pay for a higher-order scheme is an increased complexity in both the symbolic generation of the Jacobian matrices for the Newton scheme described in Section 4 and in the pattern of the nonzeroes of such matrices.

The final nonlinear system can be very large; as an example for the Adria test (see Section 5) we have $N = 3000$, $n = 19$, $m = 3$, $p = 29$ and the total number of equations is $2999 \times 19 \times 2 + 3000 \times 3 + 2 \times 29 = 123\,020$.

3.1. Elimination of controls

To reduce the complexity of the problem (14) we use the fact that the equation $\mathbf{g}(\mathbf{y}, \mathbf{u}) = \mathbf{0}$ can be solved with respect to $\mathbf{u}$ to obtain a function $\mathbf{u}(\mathbf{y})$ such that $\mathbf{g}(\mathbf{y}, \mathbf{u}(\mathbf{y})) = \mathbf{0}$. From (10) it follows that $\mathbf{u}(\mathbf{y})$ is a stationary point of the function $\tilde{d}(\mathbf{y}, \mathbf{u})$ so that Jacobian matrix of $\mathbf{g}(\mathbf{y}, \mathbf{u})$ with respect to $\mathbf{u}$ is the Hessian matrix of $\tilde{d}(\mathbf{y}, \mathbf{u})$ respect to $\mathbf{u}$. In many cases $\mathbf{u}(\mathbf{y})$ can be derived symbolically, otherwise a simple Newton–Raphson procedure can be used to compute $\mathbf{u}(\mathbf{y})$ for any given $\mathbf{y}$.

From now on it is assumed that the vector $\mathbf{u}(\mathbf{y})$ is known so that problem (14) is substituted by the following nonlinear system:

$$\mathbf{\Psi}(\mathbf{Z}) = \mathbf{0}, \quad \mathbf{Z} = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N, \boldsymbol{\mu})^{\mathrm{T}}, \tag{15}$$

where

$$\Psi_k(\mathbf{Z}) = \begin{cases} \mathbf{M}(\frac{1}{2}(\mathbf{y}_{k+1}+\mathbf{y}_k))\dfrac{\mathbf{y}_{k+1}-\mathbf{y}_k}{s_{k+1}-s_k} + \mathbf{n}(\frac{1}{2}(\mathbf{y}_{k+1}+\mathbf{y}_k), \mathbf{u}(\frac{1}{2}(\mathbf{y}_{k+1}+\mathbf{y}_k))), & k = 0 \dots N-1, \\ \mathbf{h}(\mathbf{y}_0, \mathbf{y}_N, \boldsymbol{\mu}), & k = N. \end{cases}$$

In the Adria test case the total number of equations is $2999 \times 19 \times 2 + 2 \times 29 = 114\,020$, saving about 8% of equations. However the advantage is not in reducing the number of equations because this is compensated by a slight increase of the complexity of the map. The major simplification is in the structure of the resulting Jacobian matrix used in the Newton iteration used to solve (15). Moreover, the strict coupling of $\mathbf{u}$ with $\mathbf{y}$ by $\mathbf{u}(\mathbf{y})$ results in accelerating the convergence of the iterative scheme presented below.

4. A solver for the resulting nonlinear system

A variant of the Newton–Raphson scheme, the affine invariant Newton scheme of Ref. [13], is used to solve the nonlinear system (15). The scheme is read as

- Let $\mathbf{z}^0$ assigned.
- For $j = 0, 1, \dots$, until convergence

  $$\mathbf{z}^{j+1} = \mathbf{z}^j - \alpha_j \mathbf{J}_j^{-1}\boldsymbol{\Psi}(\mathbf{z}^j),$$

  where $\mathbf{J}_j = \partial\boldsymbol{\Psi}(\mathbf{z}^j)/\partial\mathbf{z}$ and α_j is the first number satisfying:

  $$\|\mathbf{J}_j^{-1}\boldsymbol{\Psi}(\mathbf{z}^{j+1})\| \leqslant \kappa_j \|\mathbf{J}_j^{-1}\boldsymbol{\Psi}(\mathbf{z}^j)\|, \tag{16}$$

  and chosen in the sequence $\{1, q, q^2, q^3, \dots, q^{\max}\}$ with $q \in [\frac{1}{2}, 1)$ and $\kappa_j < 1$ chosen in the sequence $\{1 - q^i/2\}$.

Convergence is reached when $\alpha_j = 1$ and $\|\mathbf{J}_j^{-1}\boldsymbol{\Psi}(\mathbf{z}^j)\|$ is less than a prescribed tolerance. Although the previous algorithm works for small problems condition (16) for accepting a step is too restrictive for large size problems. This results in iterative stagnation and algorithm spend long time in very small steps. A simple modification of the algorithm allows to speed up the algorithm; in particular, we remove the monoticity request of $\kappa_j < 1$ and substitute κ_j with a constant greater then 1 when the residual $\|\mathbf{J}_j^{-1}\boldsymbol{\Psi}(\mathbf{z}^j)\|$ is large.

The resulting algorithm requires the formal inversion of the matrix $\mathbf{J}_j$ for each steps, in particular at least two inversions per iteration are needed. For this reason it is important to store the LU decomposition of $\mathbf{J}_j$ in order to reduce the computational costs. The matrix $\mathbf{J}_j$ is sparse and has the following block structure:

$$\mathbf{J}_j = \begin{pmatrix} \mathbf{A}_0^- & \mathbf{A}_0^+ & & & & \mathbf{0} \\ & \mathbf{A}_1^- & \mathbf{A}_1^+ & & & \\ & & \ddots & \ddots & & \vdots \\ & & & \mathbf{A}_{N-1}^- & \mathbf{A}_{N-1}^+ & \mathbf{0} \\ \mathbf{H}_0 & & \cdots & & \mathbf{H}_N & \mathbf{H}_{N+1} \end{pmatrix},$$

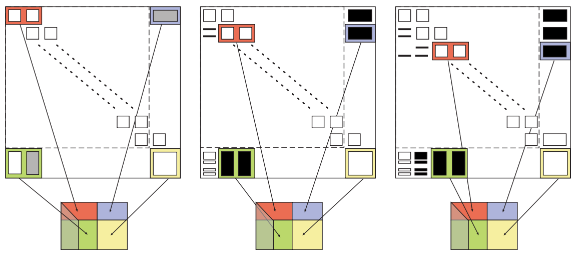

Fig. 1. Implementation of block LU decomposition for the first three steps.

where

$$\begin{aligned}\mathbf{A}_k^{\pm} &= \pm\frac{1}{s_{k+1}-s_k}\mathbf{M}(\mathbf{y}_{k+(1/2)}) + \frac{1}{2}\frac{\partial \mathbf{M}}{\partial \mathbf{y}}(\mathbf{y}_{k+(1/2)})\frac{\mathbf{y}_{k+1}-\mathbf{y}_k}{s_{k+1}-s_k} + \frac{1}{2}\frac{\partial \mathbf{n}}{\partial \mathbf{y}}(\mathbf{y}_{k+(1/2)}, \mathbf{u}_{k+(1/2)})\\ &\quad - \frac{1}{2}\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_{k+(1/2)}, \mathbf{u}_{k+(1/2)}) \times \frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\mathbf{y}_{k+(1/2)}, \mathbf{u}_{k+(1/2)})^{-1} \times \frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{y}_{k+(1/2)}, \mathbf{u}_{k+(1/2)}),\\ \mathbf{H}_0 &= \frac{\partial \mathbf{h}(\mathbf{y}_0, \mathbf{y}_N, \boldsymbol{\mu})}{\partial \mathbf{y}_0}, \quad \mathbf{H}_N = \frac{\partial \mathbf{h}(\mathbf{y}_0, \mathbf{y}_N, \boldsymbol{\mu})}{\partial \mathbf{y}_N}, \quad \mathbf{H}_{N+1} = \frac{\partial \mathbf{h}(\mathbf{y}_0, \mathbf{y}_N, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}}.\end{aligned}$$

The dimension of the blocks $\mathbf{A}^{\pm}$ is $2n \times 2n$, while the dimension of $\mathbf{H}_0$ and $\mathbf{H}_N$ is $(2n+p) \times 2n$ and the dimension of $\mathbf{H}_{N+1}$ is an $(2n+p) \times (2n+p)$. The blocks $\mathbf{A}_i^{\pm}$ and $\mathbf{H}_k$ are again sparse but the number of nonzeroes for each block is about 25% of the number of elements for each block. This sparsity pattern is not enough to make competitive a sparse algorithm for the single block so that this block matrices are assumed as full. Assuming the blocks full of nonzero it is easy to apply block Gauss algorithm to obtain LU decomposition of $\mathbf{J}_k$ obtaining virtually no fill-in. Unfortunately, such algorithm is not stable and at least partial pivoting is needed. However, also with the partial pivoting it is possible to take advantage of the block form of $\mathbf{J}_k$, in fact is possible to perform n steps of LU decomposition to extract sub-block as shown in Fig. 1 and by calling LAPACK routines [1] obtain good performance. The algorithm can be sketched as follows:

- For $k = 0$ to $N-1$,
  - extract the blocks $\mathbf{A}_k^{\pm}$, $\mathbf{H}_k$, $\mathbf{H}_{k+1}$ and the filled block as shown in Fig. 1 in the single rectangular block of dimension $(6n+p) \times (4n+p)$;
  - perform $2n$ step of Gauss elimination with partial pivoting; this operation modifies the block as shown in Fig. 2 on the left where the white part of the block corresponds to the eliminated columns;
  - restore the modified blocks in the original matrix with the corresponding permutation of rows.
- As shown in Fig. 1 this operation may produce a fill-in the block $(k, k+1)$. This is due to the pivoting search and correspondingly performed row permutation. However, analyzing Gauss elimination as a row is filled-in in the block $(k, k+1)$ a correspond row in the block (k, N) is freed.
- Extract the blocks $\mathbf{H}_{N-1}$, $\mathbf{H}_N$, $\mathbf{H}_{N+1}$, $\mathbf{A}_{N-1}^{+}$ and the filled block as shown in Fig. 3.
- Perform the final LU decomposition as shown in Fig. 2 on the right.

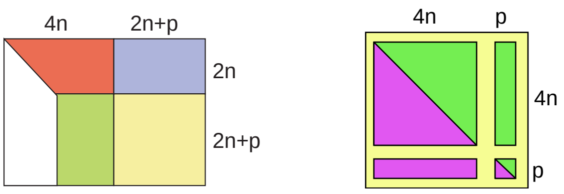

Fig. 2. Extracted subblock for the LU decomposition.

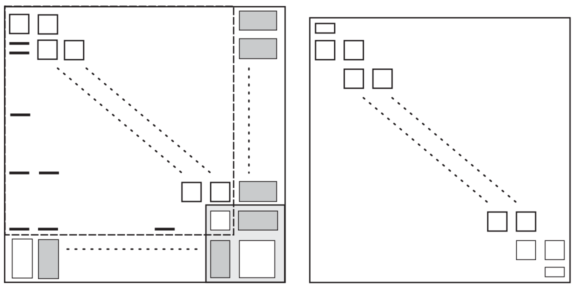

Fig. 3. On the left, block implementation of LU decomposition for the last step. On the right Jacobian matrix after the reordering in the case of well separated boundary conditions and after elimination of $\boldsymbol{\mu}$.

The computational costs in term of multiplication and division operations can be evaluated as follows:

- $(N-1)$ block Gauss steps of costs

$$\approx 16\,pn^2 + 2\,p^2 n + \frac{92}{3}n^3 \quad \text{operations * and /.}$$

- 1 final LU decomposition

$$\approx 16\,pn^2 + 4\,p^2 n + \frac{64}{3}n^3 + 1/3\,p^3 \quad \text{operations * and /.}$$

Assuming as in our test $n \approx p$ the leading term of the total cost is about $50Nn^3$. When it is possible to compute explicitly the Lagrange multiplier $\boldsymbol{\mu}$ and the boundary conditions can be distinguished, i.e.

$$\mathbf{c}(\mathbf{x}_i, \mathbf{x}_f) = \begin{cases} c_i(\mathbf{x}_i), & i = 1, \ldots, r, \\ c_i(\mathbf{x}_f), & i = r+1, \ldots, p, \end{cases}$$

the Jacobian matrix can be reordered as depicted in Fig. 3 on the right. In this case when $n \approx p$ the cost to perform LU decomposition is about $19Nn^3$ and there is no fill-in in the case of full blocks. It is clear that such approach is better from the computational point of view. However, there are situations where boundary conditions are not well separated, this is the case when cyclic conditions are imposed, moreover, this approach needs to symbolically eliminate the Lagrange multiplier $\boldsymbol{\mu}$.

A final remark should be done for the choice of $\mathbf{z}^0$. As very simple choice is as follows:

- All the $\mathbf{z}_k^0$ related to the Lagrange multiplier and controls $\mathbf{u}$ are set to 0.
- All the $\mathbf{z}_k^0$ related to the status variables $\mathbf{x}$ are evaluated as the motorcycle run slowly in the middle of the road.

5. Numerical test: solution of minimum lap-time problem

The methodology proposed above has been applied to solve a minimum lap-time problem for a sports motorcycle on a real race track (the circuit of Adria in Italy). The numerical results have been compared to data acquired from real vehicle's onboard inertial measurement unit. The motorcycle was driven by an expert test driver. Solutions of optimal control problem in racing environment are mainly useful for two purposes:

- to assess vehicle performance for first stages of vehicle design and/or for vehicle setup on a specific circuit;
- to improve race driver's skills.

Of course the agreement of numerical simulations with experimental data depends on the complexity of the mathematical model describing the dynamic behaviour of the motorcycle (system equations (1)).

Thee mathematical model described in this paper is capable of reproducing important issue of the vehicle gross motion, despite its simplicity when compared to other literature models, see Ref. [9]. The model is an improved version of the one presented in [10]. It has four degrees of freedom (x, y, roll, yaw, and steering angle), tire equations and curvilinear path equations for a total of 19 ODE equations. The parameters used were obtained by accurate laboratory measurements of the motorcycle [12] and tyres [9]. Overall, inertia properties were calculated by combining the inertia properties of the motorcycle with those of a rider of average size. The Adria circuit is described in curvilinear abscissa with constant width and discretized with 3149 mesh points. Simulation results are compared with pure gyroscopes signals and longitudinal accelerometer signals, as measured by the onboard equipment. Thus, the measured signals are

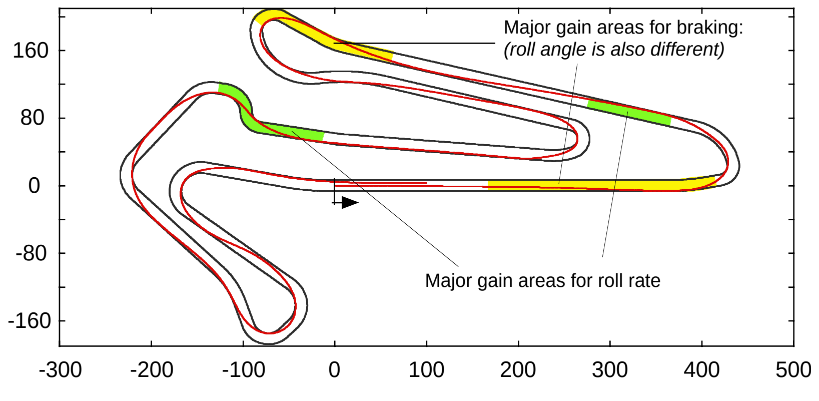

Fig. 4. Simulated minimum-time trajectory.

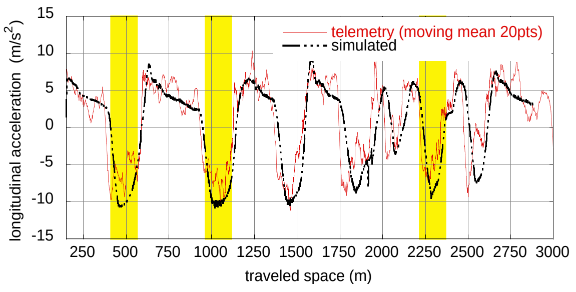

Fig. 5.

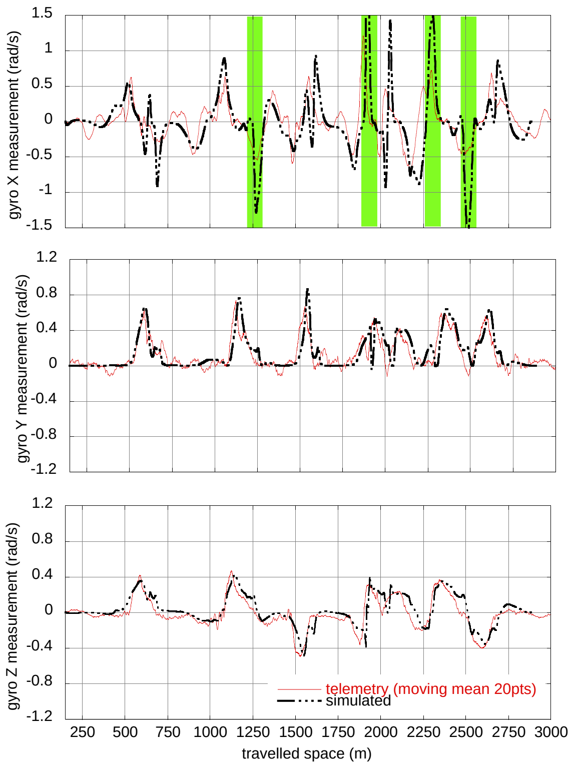

Fig. 6.

filtered by moving average to reduce noise. Then, the simulated acceleration and angular rates are projected into a moving reference frame consistent with the one of the onboard equipment for comparisons. The trajectory in Fig. 4 is the simulated one. In terms of lap-time the test driver was 6 s slower than simulated optimal performance. Fig. 4 highlights the areas where major differences between simulation estimated performance and real motorcycle performance occur. The comparison of longitudinal acceleration (see Fig. 5) shows that maximum acceleration and minimum deceleration values are almost the same except for two corners (highlighted in Fig. 4), where the test driver does not use all the longitudinal available force. This is the main cause why the test driver falls behind the simulated manoeuvre.

Fig. 6 shows the comparison of telemetry angular rates with those of numerical solution for the entire lap. The comparison shows quite good agreement for all motorcycle angular rates except for some peaks of the simulated roll rate, compared to *x* gyroscopes signal, as highlighted in the corresponding figure. The higher roll rate reached by the simulation is possible because the model does not include suspensions, which limit roll rate during fast lateral change manoeuvres. This basically occur in *S* shaped curves indicated in Fig. 4.

6. Conclusion

This paper presents a full direct approach technique for solving OCP applied to multibody mathematical models. Lagrange multipliers are used to eliminate equality constraints, while penalty formulation for inequality ones. Symbolic procedure have been used to obtain the optimality necessary condition equations, including adjoint ones. In these procedures the particular structure of multibody system with holonomic constraints is exploited, to handle the problem better with large number of equations. The resulting TPBVP is discretized by a finite difference scheme. The resulting nonlinear system of equations are automatically translated into high-level programming language along with their Jacobian matrices. The solution algorithm is based on a modified version of the affine invariant Newton scheme. In the last part of article the method is employed to solve a minimum time-lap problem for a sports motorcycle on the circuit of Adria. A ODEs mathematical model consisting of 19 equations is used. The circuit length is discretized with a mesh of about 3000 points. Comparisons with data acquired by a inertial measurements unit mounted on a similar sports motorcycle show good agreement with numerical simulations. The method is proved to be effective and able to handle large problem.

Acknowledgements

The authors would like to thank the Motorcycle Dynamics Research Group (MDRG, `www.dinamoto.it`) and in particular professor V. Cossalter and ing. R. Berritta, ing. S. Garbin, ing. F. Maggio, ing. M. Mitolo, ing. M. Salvador, for having supplied the acquired data used for comparisons, and for their support during data analysis.

Appendix A. The Model

For completeness the differential constraint (2) and the functional (1) are here stated. This model is an improvement of the model presented in [10].

The functional (1):

$$J[\mathbf{x},\mathbf{u}] = \int_{s_i}^{s_f} \frac{\mathrm{d}s}{x_{19}}.$$

The differential constraints (2) are the following expressions where the shortcuts $\mathscr{C}(x) \equiv \cos(x)$ and $\mathscr{S}(x) \equiv \sin(x)$ are used:

$$x_{19}x_3' - x_4 = 0, \quad x_{19}x_{12}' - \kappa(x_{13})x_{19}x_{13}' + x_5 = 0,$$

$$(1 - x_{14}\kappa(x_{13}))x_{19}x_{13}' - x_2\mathscr{S}(x_{12}) - x_1\mathscr{C}(x_{12}) = 0, \quad x_{19}x_{14}' - x_2\mathscr{C}(x_{12}) + x_1\mathscr{S}(x_{12}) = 0,$$

$$\frac{K_r x_1^2}{m} - \frac{x_{10}+x_{11}}{m} - 2hx_5x_4\mathscr{C}(x_3) - bx_5^2 - h\mathscr{S}(x_3)x_{19}x_5' + x_{19}x_1' - x_5x_2 = 0,$$

$$-\frac{x_9}{m} - \frac{x_8}{m} - hx_5^2\mathscr{S}(x_3) - hx_4^2\mathscr{S}(x_3) + bx_{19}x_5' + h\mathscr{C}(x_3)x_{19}x_4' + x_5x_1 + x_{19}x_2' = 0,$$

$$\frac{x_7+\tau_2x_{19}x_7'}{m} + \frac{x_6+\tau_2x_{19}x_6'}{m} + (h-r_t)x_4^2\mathscr{C}(x_3) + (h-r_t)\mathscr{S}(x_3)x_{19}x_4' - g = 0,$$

$$\begin{aligned}&\frac{I_w^f}{r^f}\mathscr{C}(\varepsilon)x_1x_{16} + (h-r_t^f)x_9\mathscr{C}(x_3) + (h-r_t^r)x_8\mathscr{C}(x_3) + I_ex_5x_1\mathscr{C}(x_3)\\&\quad + (-I_y+I_z)x_5^2\mathscr{C}(x_3)\mathscr{S}(x_3) + (r_t^f-h)(x_7+\tau_2x_{19}x_7')\mathscr{S}(x_3)\\&\quad + (r_t^r-h)(x_6+\tau_2x_{19}x_6')\mathscr{S}(x_3) - I_{xz}\mathscr{C}(x_3)x_{19}x_5' + I_xx_{19}x_4' + r_t^fx_9 + r_t^rx_8 = 0,\end{aligned}$$

$$\begin{aligned}&-\frac{I_w^f}{r^f}\mathscr{S}(\varepsilon)x_1x_{16} - I_{xz}x_5^2\mathscr{C}(x_3)^2 + (I_x+I_y-I_z)x_5x_4\mathscr{C}(x_3) + (p-b)(x_7+\tau_2x_{19}x_7')\mathscr{C}(x_3)\\&\quad + b(x_6+\tau_2x_{19}x_6')\mathscr{C}(x_3) + (p-b)x_9\mathscr{S}(x_3) + bx_8\mathscr{S}(x_3) + I_y\mathscr{S}(x_3)x_{19}x_5' + I_{xz}x_4^2\\&\quad - (r_t^r\mathscr{C}(x_3) + h - r_t^r)(x_{10}+x_{11}) - I_ex_{19}x_1' = 0,\end{aligned}$$

$$\begin{aligned}&(p-b)x_9\mathscr{C}(x_3) + bx_8\mathscr{C}(x_3) + I_{xz}x_5^2\mathscr{C}(x_3)\mathscr{S}(x_3) + (-I_x+I_y-I_z)x_5x_4\mathscr{S}(x_3)\\&\quad + (p-b)(x_7+\tau_2x_{19}x_7')\mathscr{S}(x_3) - b(x_6+\tau_2x_{19}x_6')\mathscr{S}(x_3)\\&\quad + r_t^r(x_{10}+x_{11})\mathscr{S}(x_3) + I_z\mathscr{C}(x_3)x_{19}x_5' - I_{xz}x_{19}x_4' - I_ex_4x_1 = 0,\end{aligned}$$

$$\frac{\sigma^f x_{19}x_8'}{x_1} + x_8 - \left(\frac{C_s^r(r_t^r(1-\mathscr{C}(x_3))x_4 - x_2)}{x_1} + C_r^\tau x_3\right)(x_6+\tau_2x_{19}x_6') = 0,$$

$$\frac{\sigma^r x_{19}x_9'}{x_1} + x_9 - \left(C_s^f\left(\frac{r_t^f(1-\mathscr{C}(x_3))x_4 - x_2 - x_5p}{x_1} + \frac{x_{15}\mathscr{C}(\varepsilon)}{\mathscr{C}(x_3)}\right) + C_r^f x_3\right)(x_7+\tau_2x_{19}x_7') = 0,$$

$$\begin{aligned}&I_z^f(\mathscr{C}(\varepsilon)x_{19}x_5'\mathscr{C}(x_3) - \mathscr{C}(\varepsilon)x_5\mathscr{S}(x_3)x_4 + \mathscr{S}(\varepsilon)x_{19}x_4' + x_{19}x_{16}')\\&\quad + (\mathscr{C}(\varepsilon)x_4\mathscr{C}(x_{15}) - \mathscr{S}(\varepsilon)x_5\mathscr{C}(x_3)\mathscr{C}(x_{15}) + x_5\mathscr{S}(x_3)\mathscr{S}(x_{15}))\end{aligned}$$

$$\begin{aligned}&\left(I_y^f(\mathscr{C}(x_3)\mathscr{S}(x_{15})x_5\mathscr{S}(\varepsilon) + \mathscr{S}(x_3)\mathscr{C}(x_{15})x_5 - x_4\mathscr{C}(\varepsilon)\mathscr{S}(x_{15})) - \frac{I_w^f x_1}{r^f}\right)\\&\quad - (\mathscr{C}(x_3)\mathscr{S}(x_{15})x_5\mathscr{S}(\varepsilon) + \mathscr{S}(x_3)\mathscr{C}(x_{15})x_5 - x_4\mathscr{C}(\varepsilon)\mathscr{S}(x_{15}))\end{aligned}$$

$$I_x^f(\mathscr{C}(\varepsilon)x_4\mathscr{C}(x_{15}) - \mathscr{S}(\varepsilon)x_5\mathscr{C}(x_3)\mathscr{C}(x_{15}) + x_5\mathscr{S}(x_3)\mathscr{S}(x_{15})) - (\ell - \mathscr{S}(\varepsilon)(r^f + \mathscr{C}(x_3)r_t^f))$$

$$\begin{aligned}&((\mathscr{C}(x_3)\mathscr{C}(x_{15}) - \mathscr{S}(x_3)\mathscr{S}(\varepsilon)\mathscr{S}(x_{15}))x_9 - (\mathscr{S}(x_3)\mathscr{C}(x_{15}) + \mathscr{C}(x_3)\mathscr{S}(\varepsilon)\mathscr{S}(x_{15}))(x_7+\tau_2x_{19}x_7'))\\&\quad + \mathscr{S}(x_3)r_t^f((\mathscr{C}(x_3)\mathscr{S}(x_{15}) + \mathscr{S}(x_3)\mathscr{S}(\varepsilon)\mathscr{C}(x_{15}))x_9 - (\mathscr{S}(x_3)\mathscr{S}(x_{15})\\&\quad - \mathscr{C}(x_3)\mathscr{S}(\varepsilon)\mathscr{C}(x_{15}))(x_7+\tau_2x_{19}x_7')) - x_{17} - K_\delta x_{15} - C_\delta x_{16} = 0,\end{aligned}$$

$$x_{19}x'_{15} - x_{16} = 0, \quad x_{19}x'_{10} - u_1 = 0, \quad x_{19}x'_{11} - u_2 = 0, \quad x_{19}x'_{17} - u_3 = 0, \quad x'_{18}x_{19} - 1 = 0,$$
$$- x'_{19} + x'_{19}x_{14}\kappa(x_{13}) + x_{19}x'_{14}\kappa(x_{13}) + x_{19}x'_{14}\kappa'(x_{13})x'_{13} + x'_2\mathscr{S}(x_{12}) + x_2\mathscr{C}(x_{12})x'_{12}$$
$$+ x'_1\mathscr{C}(x_{12}) - x_1\mathscr{S}(x_{12})x'_{12} + (x_{14}\kappa(x_{13}) - 1)x_{19} + x_2\mathscr{S}(x_{12}) + x_1\mathscr{C}(x_{12}) = 0.$$

As boundary conditions (3) the following values are assigned:

$$\mathbf{x}(s_i),\ x_2(s_f),\ x_3(s_f),\ x_4(s_f),\ x_5(s_f),\ x_8(s_f),\ x_9(s_f),\ x_{12}(s_f),\ x_{15}(s_f),\ x_{16}(s_f),\ x_{17}(s_f).$$

The inequality (4):

$$\frac{x_8^2}{f_{\lim}^2} + \frac{x_{10}^2}{s_{\lim}^2} \leqslant x_6^2, \quad \frac{x_9^2}{f_{\lim}^2} + \frac{x_{11}^2}{s_{\lim}^2} \leqslant x_7^2,$$
$$x_5 \leqslant x_{5\,\max}, \quad x_{10} \leqslant x_{10\,\max}, \quad x_{11} \leqslant x_{11\,\max}, \quad x_{14} \leqslant \text{road\_width},$$
$$x_{16} \leqslant x_{16\,\max}, \quad u_3 \leqslant u_{3\,\max}, \quad u_1 \leqslant u_{1\,\max}, \quad u_2 \leqslant u_{2\,\max}.$$

A.1. Parameter description

g	gravitational acceleration;
τ_2	time delay constant for vertical forces;
K_r	coefficient of air resistance;
m	total mass (motorcycle + driver);
b	longitudinal distance of centre of mass from rear wheel centre;
h	centre of mass height;
p	base wheel
I_x	overall x axis moment of inertia;
I_y	overall y axis moment of inertia;
I_z	overall z axis moment of inertia;
I_{xz}	overall xz moment of inertia;
I_x^f	front frame x axis moment of inertia;
I_y^f	front frame y axis moment of inertia;
I_z^f	front frame z axis moment of inertia;
I_v	flywheel moment of inertia;
I_w^r	rear wheel axial moment of inertia;
I_w^f	front wheel axial moment of inertia;
I_e	equivalent moment of inertia $=\frac{I_w^f}{r^f} + \frac{I_w^r}{r^r} + \frac{I_v}{r_t}$;
r_t^r	rear tyre toroidal radius;
r_t^f	front tyre toroidal radius;
r^r	rear tyre rolling radius;
r^f	front tyre rolling radius;
r^t	mean rolling radius;
ℓ	front frame offset;

K_δ	front frame stiffness;
C_δ	front framedamping;
σ^r	rear tyre relaxation length;
σ^f	front tyre relaxation length;
C_s^r	rear tyre sideslip stiffness;
C_s^f	front tyre sideslip stiffness;
C_r^r	rear tyre roll stiffness;
C_r^f	front tyre roll stiffness;
κ	road curvature;
f_{lim}	tyre lateral adherence limits;
s_{lim}	tyre longitudinal adherence limits.

A.2. State vector and controls description

x_1	forward velocity;
x_2	lateral velocity;
x_3	roll angle;
x_4	roll rate;
x_5	yaw rate;
x_6	rear vertical load;
x_7	front vertical load;
x_8	rear lateral force;
x_9	front lateral force;
x_{10}	rear longitudinal force;
x_{11}	front longitudinal force;
x_{12}	yaw relative to middle road line;
x_{13}	curvilinear abscissa;
x_{14}	lateral displacement;
x_{15}	steering angle;
x_{16}	steering rate;
x_{17}	steering torque;
x_{18}	time;
x_{19}	mapping variable;
u_1	rear longitudinal force rate;
u_2	front longitudinal force rate;
u_3	steering torque rate.

References

[1] E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J.J. Dongarra, J.D. Croz, S. Hammarling, A. Greenbaum, A. McKenney, D. Sorensen, LAPACK Users' Guide, third ed., Society for Industrial and Applied Mathematics, 1999.

[2] A. Barclay, P.E. Gill, J.B. Rosen, SQP methods and their application to numerical optimal control, in: Variational Calculus, Optimal Control and Applications, Trassenheide, 1996, vol. 124, International Series in Numerical Mathematics, Birkhäuser, Basel, 1998, pp. 207–222.
[3] A.E. Bryson Jr., Y.C. Ho, Applied Optimal Control, Hemisphere Publishing Corp., Washington, DC, 1975; optimization, estimation, and control, revised printing.
[4] R. Bulirsch, E. Nerz, H.J. Pesch, O. von Stryk, Combining direct and indirect methods in optimal control: range maximization of a hang glider, in: Optimal Control, Freiburg, 1991, International Series in Numerical Mathematics, vol. 111, Birkhäuser, Basel, 1993, pp. 273–288.
[5] C. Büskens, H. Maurer, SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control, J. Comput. Appl. Math. 120 (1–2) (2000) 85–108 sQP-based direct discretization methods for practical optimal control problems.
[6] D. Casanova, R.S. Sharp, P. Symonds, Minimum time manoeuvring: the significance of yaw inertia, Vehicle System Dynamics 34 (2000) 77–115.
[7] L. Cervantes, L.T. Biegler, Optimization strategies for dynamic systems, in: C. Floudas, P. Pardalos (Eds.), Encyclopedia of Optimization, vol. 4, Kluwer, Dordrecht, 2001, pp. 216–227, URL citeseer.nj.nec.com/cervantes99optimization.html.
[8] V. Cossalter, M. Da Lio, F. Biral, L. Fabbri, Evaluation of motorcycle manoeuvrability with the optimal manoeuvre method, SAE Trans.—J. Passenger Cars, SAE Paper 983022.
[9] V. Cossalter, R. Lot, A motorcycle multi-body model for real time simulations based on the natural coordinates approach, Vehicle System Dynamics 37 (6) (2002) 423–447.
[10] M. Da Lio, V. Cossalter, R. Lot, L. Fabbri, A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles, Vehicle System Dynamics 31 (2) (1999) 113–135.
[11] M. Da Lio, V. Cossalter, R. Lot, L. Fabbri, The influence of tyre characteristics on motorcycle manoeuvrability, in: European Automotive Congress, Conference II: Vehicle Dynamics and Active Safety, 1999, Barcelona, Spain, June 30–July 2.
[12] M. Da Lio, A. Doria, R. Lot, A spatial mechanism for the measurement of the inertia tensor: theory and experimental results, ASME J. Dynamic Systems Meas. Control 121 (1999) 111–116.
[13] P. Deuflhard, G. Heindl, Affine invariant convergence theorems for Newton's method and extensions to related methods, SIAM J. Numer. Anal. 16 (1) (1979) 1–10.
[14] B.C. Fabien, Numerical solution of constrained optimal control problems with parameters, Appl. Math. Comput. 80 (1) (1996) 43–62.
[15] J. García de Jalón, E. Bayo, Kinematic and dynamic simulation of multibody systems, Mechanical Engineering Series, Springer, New York, 1994 the real-time challenge.
[16] P.E. Gill, W. Murray, M.A. Saunders, SNOPT: an SQP algorithm for large-scale constrained optimization, SIAM J. Optim. 12 (4) (2002) 979–1006 (electronic).
[17] H.B. Keller, Accurate difference methods for nonlinear two-point boundary value problems, SIAM J. Numer. Anal. 11 (1974) 305–320.
[18] B. Kugelmann, H.J. Pesch, New general guidance method in constrained optimal control, II , Application to space shuttle guidance, J. Optim. Theory Appl. 67 (3) (1990) 437–446.
[19] B. Kugelmann, H.J. Pesch, New general guidance method in constrained optimal control, I, Numerical method, J. Optim. Theory Appl. 67 (3) (1990) 421–435.
[20] M. Lentini, V. Pereyra, An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers, SIAM J. Numer. Anal. 14 (1) (1977) 94–111 papers on the numerical solution of two-point boundary-value problems (NSF-CBMS Regional Research Conference, Texas Tech University, Lubbock, Tex., 1975).
[21] H.J. Pesch, A practical guide to the solution of real-life optimal control problems, Control Cybernet. 23 (1–2) (1994) 7–60 parametric optimization.
[22] R.W.H. Sargent, Optimal control, J. Comput. Appl. Math. 124 (1–2) (2000) 361–371 numerical analysis 2000, vol. IV, Optimization and nonlinear equations.
[23] R. Serban, L.R. Petzold, COOPT—a software package for optimal control of large-scale differential-algebraic equation systems, Math. Comput. Simulation 56 (2) (2001) 187–203 method of lines (Athens, GA, 1999).
[24] T. Spägele, A. Kistner, A. Gollhofer, A multi-phase optimal control technique for the simulation of a human vertical jump, J. Biomech. 32 (1999) 87–91.
[25] J. Stoer, R. Bulirsch, Introduction to numerical analysis, Texts in Applied Mathematics, vol. 12, third ed., Springer, New York, 2002, translated from the German by R. Bartels, W. Gautschi and C. Witzgall.

[26] S. Storen, T. Hertzberg, The sequential linear-quadratic programming algorithm for solving dynamic optimization problems—a review, Comput. Chem. Eng. 19 (1995) 495–500.
[27] J.L. Troutman, Variational calculus and optimal control, Undergraduate Texts in Mathematics, second ed., Springer, New York, 1996 with the assistance of William Hrusa, Optimization with elementary convexity.
[28] O. von Stryk, Numerical solution of optimal control problems by direct collocation, in: Optimal Control, Freiburg, 1991, International Series in Numerical Mathematics, vol. 111, Birkhäuser, Basel, 1993, pp. 129–143.
[29] O. von Stryk, M. Schlemmer, Optimal control of the industrial robot Manutec r3, in: Computational Optimal Control, Munich, 1992, International Series in Numerical Mathematics, vol. 115, Birkhäuser, Basel, 1994, pp. 367–382.
[30] S.J. Wright, Stable parallel algorithms for two-point boundary value problems, SIAM J. Sci. Statist. Comput. 13 (3) (1992) 742–764.