IEEE Transactions on Control Systems Technology, Volume: PP Issue:99, Pages 1 - 9, 2010



# A Conjugate Gradient-Based BPTT-like Optimal Control Algorithm with Vehicle Dynamics Control Application

Josip Kasać, Joško Deur, *Senior Member, IEEE*, Branko Novaković, *Senior Member, IEEE*, Ilya V. Kolmanovsky, *Fellow, IEEE* and Francis Assadian

***Abstract*—The paper presents a gradient-based algorithm for optimal control of nonlinear multivariable systems with control and state vectors constraints. The algorithm has a backward-in-time recurrent structure similar to the backpropagation-through-time (BPTT) algorithm, which is mostly used as a learning algorithm for dynamic neural networks. Other main features of the algorithm include the use of higher order Adams time-discretization schemes, numerical calculation of Jacobians, and advanced conjugate gradient methods for favorable convergence properties. The algorithm performance is illustrated on an example of off-line vehicle dynamics control optimization based on a realistic high-order vehicle model. The optimized control variables are active rear differential torque transfer and active rear steering road wheel angle, while the optimization tasks are trajectory tracking and roll minimization for a double lane change maneuver.**

***Index Terms*—Optimal control, conjugate gradient methods, automotive applications, road vehicle control.**

## I. Introduction

Optimal control has found its applications in many different engineering fields, including aerospace [1], process control [2], robotics [3] , and automotive control [4,5]. Any control system that includes complex dynamics with constraints is a good candidate for applying optimal control. The main aim is to find control variable trajectories that minimize an optimization criterion in the presence of inequality and equality constraints on the control and state variables. By doing this in an off-line manner [1-5], the optimal control results can be used to assess the performance of control systems with different hardware and actuator configurations, set realistic targets for achievable system performance, cascade targets to subsystems and components, and guide control system design and calibration process. For example, prior to deciding on the vehicle dynamics actuators (e.g. active front/rear steering and active central/rear differential) and developing related on-line controls, the optimization results can provide valuable information on quality of various actuator configurations and establish an "idealized" benchmark to compare different control solutions and calibration in terms of their closeness to the ideal performance.

The numerical methods of solving optimal control problems can be divided into two categories: direct and indirect methods. Indirect methods based on Pontrjagin's maximum principle lead to numerical solution of the two-point boundary value problem using multiple shooting or quasi-linearization [6], [7]. Direct methods transform the original continuous-time optimal control problem into a finite-dimensional nonlinear programming (NLP) problem, which can be solved by various NLP numerical optimization algorithms such as sequential quadratic programming (SQP) [8-11]. The continuous-time state and/or control variables are represented by a finite number of parameters by time-discretization or by using suitable basis functions such as B-splines or Lagrange polynomials.

The main disadvantage of indirect methods is their need for a good initial guess of the initial conditions for the adjoint variables in order to converge. Also, this approach requires symbolic differentiation to obtain adjoint equations. On the other hand, the direct methods are characterized by a large and sparse structure of Jacobian and Hessian matrices, so that they can be computationally expensive for large systems. Direct methods are mainly applied in industrial optimization problems requiring fast and numerically robust optimization, while allowing for less accurate solutions.

In the NLP approach, the plant equation constraints are added into the cost function in extension to the penalty functions related to the state and control constraints. The control and state variables are treated as independent variables, so that the cost function gradient calculation is relatively simple. However, the optimization problem formulated in such a way may be characterized by a slow

Manuscript received October 9, 2009. This work has been supported by Ford Motor Company, and partly by Jaguar Cars Ltd. and the Ministry of Science, Education and Sports of the Republic of Croatia.

Josip Kasać, Joško Deur, and Branko Novaković are with the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, I. Lučića 5, HR-10002 Zagreb, Croatia (e-mail: josip.kasac@fsb.hr; josko.deur@fsb.hr; branko.novakovic@fsb.hr).

Ilya Kolmanovsky was with the Ford Research and Advanced Engineering at the time of writing this paper; he is now with the Department of Aerospace Engineering, University of Michigan, 1320 Beal Avenue, Ann Arbor, MI 48109-2140 (e-mail: ilya@umich.edu).

Francis Assadian was with Jaguar Cars Ltd, UK at the time of writing this paper. Currently, he is with the Cranfield University, Department of Automotive Engineering, Bedfordshire MK43 0AL, UK, (e-mail: f.assadian@cranfield.ac.uk).

convergence due to the additional plant-equations equality constraints. Also, the numerical stability may be affected by the choice of various optimization parameters such as discretization period and weighting factors of penalty functions.

An alternative direct approach of solving the optimal control problem is proposed by the authors in [12] and [13]. In contrast to the NLP approach, the plant equations constraints are not included in the cost function. The control and state variables are treated as dependent variables (coupled via plant equations), so that the final algorithm has a backward-in-time structure similar to the backpropagation-through-time (BPTT) algorithm [14], [15], which is mostly used as a learning algorithm for recurrent neural networks. Such an exact gradient algorithm is more complex than the NLP-based algorithm, but it may provide better and numerically more stable convergence properties. The algorithm has been applied for solving optimal control problems in the fields of robotics [12] and vehicle dynamics control [5].

In order to further enhance its optimization accuracy and convergence properties, the original BPTT algorithm [12,13,5] is extended in this paper by: (i) higher-order Adams numerical integration schemes instead of the basic Euler discretization method, (ii) a straightforward derivation of gradient expressions based on introducing a terminal cost function, (iii) numerical calculation of Jacobians, and (iv) implementation of more advanced, conjugate gradient methods. Exact gradient derivation based on the BPTT approach and higher-order Adams methods represents a key feature of the proposed approach and a novel contribution. It is precisely this combination that, through our numerical case studies, has been proven to be effective, numerically robust, and capable of handling models with complexity realistic for industrial applications. In regard to the vehicle dynamics control case study, a full 10-DOF vehicle model comprising a full "magic" formula tire model is used instead of a simplified vehicle/tire model considered in [5].

## II. Optimal Control Problem Formulation

### A. Continuous-Time Problem Formulation

A continuous-time nonlinear optimal control problem is considered. The problem is to find a control vector input $\mathbf{u}(t)$, $0 \le t \le t_f$, which minimizes the Bolza-type cost function

$$J_0 = \Phi_0\left(\hat{\mathbf{x}}(t_f)\right) + \int_0^{t_f} F_0\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) dt\,, \tag{1}$$

subject to the nonlinear continuous-time plant equations

$$\dot{\hat{\mathbf{x}}}(t) = \boldsymbol{\phi}\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right), \qquad \hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_0, \tag{2}$$

and subject to the final conditions on the state vector

$$\mathbf{b}\left(\hat{\mathbf{x}}(t_f)\right) = 0, \tag{3}$$

and subject to the control and state vector inequality constraints

$$\mathbf{g}\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) \ge 0\,, \tag{4}$$

and equality constraints

$$\mathbf{h}\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) = 0\,, \tag{5}$$

where $\hat{\mathbf{x}}(t)$ is an $n_0$-dimensional state vector, $\mathbf{u}(t)$ is an $m$-dimensional control vector, $\mathbf{g}(\hat{\mathbf{x}}(t), \mathbf{u}(t))$ is a $p$-dimensional vector function of inequality constraints, $\mathbf{h}(\hat{\mathbf{x}}(t), \mathbf{u}(t))$ is a $q$-dimensional vector function of equality constraints, $\mathbf{b}(\hat{\mathbf{x}}(t_f))$ is a $r$-dimensional vector function of final boundary condition constraints, and $t_f$ is the terminal time. We assume that $F_0(\cdot)$, $\Phi_0(\cdot)$, $\boldsymbol{\phi}(\cdot)$, $\mathbf{b}(\cdot)$, $\mathbf{g}(\cdot)$, and $\mathbf{h}(\cdot)$ are continuously differentiable functions.

In general, additional constraints which ensure robustness may be augmented to the optimal control problem formulation to avoid regions with large model uncertainties (which the optimization may otherwise incorrectly exploit) or to reduce parametric sensitivity of optimized control input trajectories (see e.g. [26, Chap. 10]).

### B. Transformation of Continuous-Time Optimization Problem

The optimization problem (1)-(5) can be reduced to the problem of finding the control vector $\mathbf{u}(t)$ that minimizes the cost function

$$J = \Phi\left(\hat{\mathbf{x}}(t_f)\right) + \int_0^{t_f} F\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) dt\,, \tag{6}$$

subject to the plant equations

$$\dot{\hat{\mathbf{x}}}(t) = \boldsymbol{\phi}\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right), \qquad \hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_0, \tag{7}$$

where

$$\begin{aligned} F\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) = F_0\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) + \sum_{k=1}^{q} K_{h,k} h_k^2\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) + \\ + \sum_{k=1}^{p} K_{g,k} g_k^2\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right) H^-\left(g_k\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right)\right), \end{aligned} \tag{8}$$

$$\Phi\left(\hat{\mathbf{x}}(t_f)\right) = \Phi_0\left(\hat{\mathbf{x}}(t_f)\right) + \sum_{k=1}^{r} K_{b,k} b_k^2\left(\hat{\mathbf{x}}(t_f)\right), \tag{9}$$

and $H^-(z)$ is Heaviside step function defined as

$$H^-(z) = \begin{cases} 0, & \text{if} \quad z \ge 0\,, \\ 1, & \text{if} \quad z < 0\,. \end{cases} \tag{10}$$

The second and third terms on the right-hand side of expression (8) are the penalty functions for the inequality and equality constraints (4) and (5), respectively. Similarly, the second term on the right-hand side of expression (9) is the penalty function for the final boundary condition (3). Note that although the Heaviside step function $H^-(z)$ is not continuous, the penalty terms of the form $z^2 H^-(z)$ in equation (8) are

continuously differentiable functions. The penalty function coefficients $K_{h,k}$, $K_{g,k}$ and $K_{b,k}$ should be sufficiently large to provide accurate constraints satisfaction.

In order to simplify application of higher-order numerical integration methods for the plant equations (7) and the integral term in the cost function (6), the problem (6)-(9) is reformulated, so that an additional state variable $x_n(t)$ is introduced such that

$$\dot{x}_n = F\left(\hat{\mathbf{x}}(t), \mathbf{u}(t)\right), \qquad x_n(0) = 0, \tag{11}$$

where $n = n_0 + 1$. Hence, the final continuous-time optimization problem is to find the control vector $\mathbf{u}(t)$ that minimizes the terminal cost function

$$J\left(t_f\right) = \Phi\left(\hat{\mathbf{x}}(t_f)\right) + x_n\left(t_f\right), \tag{12}$$

subject to the differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right), \qquad \mathbf{x}(0) = \mathbf{x}_0, \tag{13}$$

where $\mathbf{x}(t)$ is the new $n$-dimensional state vector

$$\mathbf{x}(t) = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 & \dots & \hat{x}_{n_0} & x_n \end{bmatrix}^T,$$

and

$$\mathbf{f} = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_{n_0} & F \end{bmatrix}^T.$$

### C. Time Discretization Based on Adams Method

The Adams method [17] belongs to the class of multistep numerical methods for an approximate solution of the system of ordinary differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right), \qquad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{14}$$

The $k$-th order Adams method has the following form:

$$\mathbf{x}(i+1) = \mathbf{x}(i) + \tau \sum_{j=1}^{k} a_j^{(k)} \mathbf{f}(i-j+1), \tag{15}$$

for $i = k-1, k, k+1, \dots$, and the initial conditions $\mathbf{x}(0) = \mathbf{x}_0$, $\mathbf{x}(1) = \mathbf{x}_1$, $\mathbf{x}(2) = \mathbf{x}_2$, …, $\mathbf{x}(k-1) = \mathbf{x}_{k-1}$; where $\tau$ is the time step, $a_j^{(k)}$ are the coefficients of the Adams method [17], and

$$\mathbf{f}(i) \equiv \mathbf{f}\left(\mathbf{x}(i), \mathbf{u}(i)\right). \tag{16}$$

The Adams method of the $k$-th order, as a multistep method, requires knowledge of $k$ initial conditions. These initial conditions are determined from the basic initial condition $\mathbf{x}(0) = \mathbf{x}_0$ by using the 4th-order (one-step) Runge-Kutta method.

The explicit Adams method (15) is a $k$-th order vector difference equation, which can be conveniently transformed into the following discrete-time state-space form

$$\begin{aligned} x_j(i+1) &= x_j(i) + \tau a_1^{(k)} f_j(i) + \tau x_{n+j}(i), \\ x_{rn+j}(i+1) &= a_{r+1}^{(k)} f_j(i) + x_{(r+1)n+j}(i), \\ x_{(k-1)n+j}(i+1) &= a_k^{(k)} f_j(i), \end{aligned} \tag{17}$$

for $r = 1,2,\dots,k-2$, $j = 1,2,\dots,n$, $i = k-1, k, k+1, \dots$, and the initial conditions

$$\begin{aligned} x_j(k-1) &= x_{j(k-1)}, \\ x_{qn+j}(k-1) &= \sum_{l=q+1}^{k} a_l^{(k)} f_j(k-1+q-l), \end{aligned}$$

for $q = 1,2,\dots,k-1$. Finally, using the vector notation (cf. (13)), the state-space form of the $k$-th order Adams method reads

$$\tilde{\mathbf{x}}(i+1) = \tilde{\mathbf{f}}\left(\tilde{\mathbf{x}}(i), \mathbf{u}(i)\right), \qquad \tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0, \tag{18}$$

where $\tilde{\mathbf{x}}(t)$ is the extended $n \cdot k$-dimensional state vector

$$\tilde{\mathbf{x}}(t) = \begin{bmatrix} x_1 & x_2 & \dots & x_{n \cdot k-1} & x_{n \cdot k} \end{bmatrix}^T,$$

and

$$\tilde{\mathbf{f}} = \begin{bmatrix} x_1(i) + \tau a_1^{(k)} f_1(i) + \tau x_{n+1}(i) & \dots & a_k^{(k)} f_n(i) \end{bmatrix}^T.$$

## III. BPTT Optimal Control Algorithm

### A. Discrete–Time Optimization Problem

The discrete-time optimization problem is to find the control sequence $\mathbf{u}(i)$, $i = 0,1,\dots,N-1$, which minimizes the discrete-time form of the cost function (12):

$$J = \Phi\left(\tilde{\mathbf{x}}(N)\right) + x_n\left(N\right), \tag{19}$$

subject to the $k$-th order Adams approximation of the continuous-time state equations (13):

$$\tilde{\mathbf{x}}(i+1) = \tilde{\mathbf{f}}\left(\tilde{\mathbf{x}}(i), \mathbf{u}(i)\right), \qquad \tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0, \tag{20}$$

for $i = k-1, k, k+1, \dots, N-1$.

The cost function $J$ depends explicitly only on the state vector at the terminal time, $\mathbf{x}(N)$, but an implicit dependence on $\mathbf{x}(i)$ follows from the discrete-time state equations (20). This fact will be used in derivation of the gradient decent algorithm below for exact calculation of the gradient of cost function $J$ with respect to control vector $\mathbf{u}(i)$, $i = 0,1,\dots,N-1$.

### B. Gradient Calculation

The gradient descent algorithm with respect to control vector is as follows:

$$\mathbf{u}^{(l+1)}(i) = \mathbf{u}^{(l)}(i) - \eta \frac{\partial J}{\partial \mathbf{u}^{(l)}(i)}, \tag{21}$$

where $i$ = 0, 1, …, $N$-1, $l$= 1, 2, …, $M$, $\eta$ is the learning-rate, $N$ is the number of time instants, and $M$ is the number of gradient algorithm iterations.

The gradient of the cost function (19) in the $l$-th iteration of the gradient algorithm and $i$-th sampling interval is given by

$$\frac{\partial J}{\partial u_j(i)} = \sum_{r=1}^{nk} \frac{\partial J}{\partial \tilde{x}_r(N)} \frac{\partial \tilde{x}_r(N)}{\partial u_j(i)}, \tag{22}$$

where $j = 1, 2, ..., m$. The partial derivatives $\partial \tilde{x}_r(N)/\partial u_j(i)$ can be calculated backward in time, starting from $i = N-1$:

$$\frac{\partial \tilde{x}_r(N)}{\partial u_j(N-1)} = \frac{\partial \tilde{f}_r(N-1)}{\partial u_j(N-1)},$$

where $\tilde{f}_r(i) \equiv \tilde{f}_r(\tilde{\mathbf{x}}(i), \mathbf{u}(i))$, and $r = 1, 2, ..., nk$. Further, for $i = N-2$:

$$\frac{\partial \tilde{x}_r(N)}{\partial u_j(N-2)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \frac{\partial \tilde{x}_q(N-1)}{\partial u_j(N-2)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \frac{\partial \tilde{f}_q(N-2)}{\partial u_j(N-2)}$$

and for $i = N-3$:

$$\frac{\partial \tilde{x}_r(N)}{\partial u_j(N-3)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \frac{\partial \tilde{x}_q(N-1)}{\partial u_j(N-3)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \frac{\partial \tilde{f}_q(N-2)}{\partial u_j(N-3)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \sum_{p=1}^{nk} \frac{\partial \tilde{f}_q(N-2)}{\partial \tilde{x}_p(N-2)} \frac{\partial \tilde{x}_p(N-2)}{\partial u_j(N-3)} = \sum_{q=1}^{nk} \frac{\partial \tilde{f}_r(N-1)}{\partial \tilde{x}_q(N-1)} \sum_{p=1}^{nk} \frac{\partial \tilde{f}_q(N-2)}{\partial \tilde{x}_p(N-2)} \frac{\partial \tilde{f}_p(N-3)}{\partial u_j(N-3)}.$$

By introducing matrices $\tilde{\mathbf{U}}(i)$, $\tilde{\mathbf{X}}(i)$, $\mathbf{Y}(i)$ with the elements

$$\tilde{\mathbf{U}}(i) = \frac{\partial \tilde{\mathbf{f}}(i)}{\partial \mathbf{u}(i)}, \quad \tilde{\mathbf{X}}(i) = \frac{\partial \tilde{\mathbf{f}}(i)}{\partial \tilde{\mathbf{x}}(i)}, \quad \mathbf{Y}(i) = \frac{\partial \tilde{\mathbf{x}}(N)}{\partial \mathbf{u}(i)},$$

the above derivatives $\partial \tilde{x}_r(N)/\partial u_j(i)$, $i = N-1, N-2, N-3$, can be expressed in a more compact matrix form as

$$\mathbf{Y}(N-1) = \tilde{\mathbf{U}}(N-1), \tag{23}$$

$$\mathbf{Y}(N-2) = \tilde{\mathbf{X}}(N-1)\tilde{\mathbf{U}}(N-2), \tag{24}$$

$$\mathbf{Y}(N-3) = \tilde{\mathbf{X}}(N-1)\tilde{\mathbf{X}}(N-2)\tilde{\mathbf{U}}(N-3). \tag{25}$$

This procedure can be further continued as follows

$$\mathbf{Y}(N-4) = \tilde{\mathbf{X}}(N-1)\tilde{\mathbf{X}}(N-2)\tilde{\mathbf{X}}(N-3)\tilde{\mathbf{U}}(N-4), \tag{26}$$

$$\vdots$$

$$\mathbf{Y}(i) = \tilde{\mathbf{X}}(N-1)\tilde{\mathbf{X}}(N-2)\ldots\tilde{\mathbf{X}}(i+1)\tilde{\mathbf{U}}(i). \tag{27}$$

By introducing matrices $\mathbf{J}_u(i)$ and $\mathbf{J}_x(N)$ such that

$$\mathbf{J}_u(i) = \frac{\partial J}{\partial \mathbf{u}(i)}, \qquad \mathbf{J}_x(N) = \frac{\partial J}{\partial \tilde{\mathbf{x}}(N)},$$

the final gradient $\mathbf{J}_u(i)$ in (21) can be computed by the following backward-in-time recursive matrix relations:

$$\mathbf{J}_u(i) = \mathbf{Y}(i)^T \mathbf{J}_x(N), \tag{28}$$

$$\mathbf{Y}(i) = \mathbf{D}(i)\tilde{\mathbf{U}}(i), \tag{29}$$

$$\mathbf{D}(i) = \mathbf{D}(i+1)\tilde{\mathbf{X}}(i+1), \tag{30}$$

for $i = N-2, N-3, ..., 0$, with the initial conditions

$$\mathbf{J}_u(N-1) = \tilde{\mathbf{U}}(N-1)^T \mathbf{J}_x(N), \tag{31}$$

$$\mathbf{D}(N-1) = \mathbf{I}. \tag{32}$$

### C. Calculation of Extended Jacobians

The extended Jacobians $\tilde{\mathbf{X}}(i)$ and $\tilde{\mathbf{U}}(i)$ can be expressed as functions of the basic Jacobians

$$\mathbf{X}(i) = \frac{\partial \mathbf{f}(i)}{\partial \mathbf{x}(i)}, \quad \mathbf{U}(i) = \frac{\partial \mathbf{f}(i)}{\partial \mathbf{u}(i)},$$

as given by the following equations based on (18):

$$\tilde{\mathbf{X}}(i) = \begin{bmatrix} \mathbf{I} + \tau a_1^{(k)} \mathbf{X}(i) & \tau \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ a_2^{(k)} \mathbf{X}(i) & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ a_3^{(k)} \mathbf{X}(i) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k-1}^{(k)} \mathbf{X}(i) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \\ a_k^{(k)} \mathbf{X}(i) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{33}$$

$$\tilde{\mathbf{U}}(i) = \begin{bmatrix} \tau a_1^{(k)} \mathbf{U}(i) \\ a_2^{(k)} \mathbf{U}(i) \\ a_3^{(k)} \mathbf{U}(i) \\ \vdots \\ a_{k-1}^{(k)} \mathbf{U}(i) \\ a_k^{(k)} \mathbf{U}(i) \end{bmatrix}, \tag{34}$$

Similarly, based on (19), the extended gradient $\mathbf{J}_x(N) = \partial J / \partial \tilde{\mathbf{x}}(N) \in \mathbf{R}^{nk}$ is related to the basic gradient $\partial \Phi(\tilde{\mathbf{x}}(N)) / \partial \tilde{\mathbf{x}}(N) \in \mathbf{R}^{n_0}$ as follows

$$\mathbf{J}_x(N) = \frac{\partial J}{\partial \tilde{\mathbf{x}}(N)} = \left[ \left( \frac{\partial \Phi}{\partial \tilde{\mathbf{x}}(N)} \right)^T \quad 1 \quad 0 \quad \cdots \quad 0 \right]^T. \tag{35}$$

The basic Jacobians $\mathbf{X}(i)$ and $\mathbf{U}(i)$ can be calculated analytically [5,13] or numerically based on an appropriate finite-difference formula for the first derivatives [18].

## IV. Algorithm Convergence Speed-Up

The main weaknesses of the standard gradient algorithm (21), with a constant learning rate $\eta$, include a rather slow convergence and difficulties in tuning the learning rate appropriately. A small learning rate will result in a slow algorithm convergence, while a large learning rate can lead to numerical instabilities.

In this work we apply the conjugate gradient (CG) algorithm [19]-[21] which has the following form

$$\mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} + \eta_l \mathbf{d}^{(l)}, \tag{36}$$

$$\mathbf{d}^{(l+1)} = -\mathbf{g}^{(l+1)} + \beta_l \mathbf{d}^{(l)}, \tag{37}$$

where $\eta_l$ and $\beta_l$ are positive scalars, and

$$\mathbf{w} \equiv \left[ u_1(0) \quad u_1(1) \quad \ldots \quad u_m(N-2) \quad u_m(N-1) \right]^T \in \mathbf{R}^{mN},$$

$$\mathbf{g} \equiv \left[ \frac{\partial J}{\partial u_1(0)} \quad \frac{\partial J}{\partial u_1(1)} \quad \cdots \quad \frac{\partial J}{\partial u_m(N-2)} \quad \frac{\partial J}{\partial u_m(N-1)} \right]^T.$$

The standard method for computing $\eta_l$ is steepest descent or line search algorithm which requires one-dimensional minimization of the cost function. This is a computationally expensive method which may require many evaluations of the cost function during one iteration of the gradient algorithm. Also, if the cost function is not appropriately scaled, the steepest-descent algorithm may exhibit poor convergence properties. In order to avoid these issues, we use the SuperSAB approach [21], [22] which requires only the information on gradient directions in two consecutive iterations of the gradient algorithm. The algorithm is modified in terms of using a scalar learning rate $\eta_l$ as oppose to a matrix formulation $\eta_l(i,j)$, $i = 0,...,N-1$, $j = 1,...,m$, in order to avoid discontinuities in the optimized control vector $\mathbf{u}$. The modified SuperSAB algorithm is given by

$$\eta_l = \begin{cases} d^+ \eta_{l-1} & \text{if} \quad \left(\mathbf{g}^{(l)T}\mathbf{g}^{(l-1)} \geq 0\right) \& \left(J\left(\mathbf{w}^{(l)}\right) < J\left(\mathbf{w}^{(l-1)}\right)\right) \\ d_1^- \eta_{l-1} & \text{if} \quad \left(\mathbf{g}^{(l)T}\mathbf{g}^{(l-1)} < 0\right) \& \left(J\left(\mathbf{w}^{(l)}\right) < J\left(\mathbf{w}^{(l-1)}\right)\right) \\ d_2^- \eta_{l-1} & \text{if} \quad J\left(\mathbf{w}^{(l)}\right) > J\left(\mathbf{w}^{(l-1)}\right) \end{cases} \tag{38}$$

where $0 < d_2^- < d_1^- < 1 < d^+$, and $\eta_0$ is the initial learning rate. If the angle between two consecutive gradients is smaller than 90°, $\mathbf{g}^{(l)T}\mathbf{g}^{(l-1)} \geq 0$, the learning rate $\eta_l$ is increased starting from some initial value. On the other hand, negative sign of the scalar product of the gradients, $\mathbf{g}^{(l)T}\mathbf{g}^{(l-1)} < 0$, indicates that the iterative procedure has overshot the minimum of the cost function, and the learning rate $\eta_l$ is decreased by multiplying it with the decreasing factor $d_1^-$ smaller than unity. Also, the algorithm (38) decreases the learning rate by a decreasing factor $d_2^- < d_1^-$ if the condition $J(\mathbf{w}^{(l)}) > J(\mathbf{w}^{(l-1)})$ is satisfied, in order to avoid occasional cost function spikes that may affect the optimization accuracy [25].

The scalar value $\beta_l$ in (37) can be determined by using different methods [21]. A comparison of these methods on the vehicle dynamics control example [25] have pointed out that the following Dai-Yuan approach gives the fastest convergence:

$$\beta_l = \min\left\{ \frac{\mathbf{g}^{(l+1)T}\mathbf{g}^{(l+1)}}{\mathbf{d}^{(l)T}\left(\mathbf{g}^{(l+1)} - \mathbf{g}^{(l)}\right)},\ \beta_{\max} \right\}. \tag{39}$$

The parameter $\beta_l$ is limited to $\beta_{\max}$, because the algorithm (38) for learning-rate tuning can induce $\left\|\mathbf{g}^{(l)}\right\| \geq \left\|\mathbf{g}^{(l-1)}\right\|$ in situations when $\mathbf{g}^{(l)T}\mathbf{g}^{(l-1)} < 0$, thus leading to a possible algorithm instability if $\beta_l$ is not saturated. The limit value of $\beta_l$ which guaranties numerical stability is $\beta_{\max} = 1$. But, depending on particular optimization problem this limit can be increased (e.g. $\beta_{\max} = 1.2$), in order to provide a faster convergence of the algorithm. If the parameter $\beta_l$ has a constant value, $0 < \beta < 1$, then the CG algorithm becomes equivalent to a standard gradient algorithm with momentum [23,24].

## V. Vehicle Dynamics Control Application

The proposed optimal control algorithm has been verified in [18] on a chemical reaction model example with a known analytical solution. In this section, the algorithm is used for more realistic and complex off-line optimization of vehicle dynamics control variables.

### A. Vehicle Dynamics Model

A 10 degree of freedom (DOF) vehicle dynamics model [16], provided by the Jaguar Cars, has been used in the bellow optimization study. The main 10 state variables are longitudinal and lateral velocities $U$ and $V$; roll, pitch, yaw, and heave rates $p$, $q$, $r$, and $W$; and four wheel angular speeds. The auxiliary state variables are roll, pitch and yaw angles, as well as the heave, which are used to determine the vehicle position ($X$, $Y$) in the inertial coordinate frame and determine the suspension deflections. The tire forces are obtained from the 1994 "Magic formula" tire model (see [16] and references therein).

The rear steering and rear differential actuator dynamics are approximately described by first order lag terms with the time constant of 25 ms. Two types of active differentials are considered [16]: Active Limited Slip Differential (ALSD) and Torque Vectoring Differential (TVD). The ALSD always transfer the torque to the slower wheel. The TVD can also transfer the torque to the faster wheel, provided that the faster/slower wheel speed ratio is not larger than a design factor $k_{\text{AWSD}}$ that has a typical value of 1.25. The torque transfer constraints have been implemented in the vehicle model, as explained in [5]. The overall nonlinear dynamics model has 18 state variables and two control variables.

### B. Optimization Problem Formulation

Several optimization problems and control/state vector constraints are considered.

1) *Cost Functions Definition*

*Trajectory tracking*. The optimal control objective for the trajectory tracking task is to find the active rear differential and active rear steering control variables $\Delta T_r$ and $\delta_r$, respectively, which ensure that the vehicle follows the reference trajectory in the $X$-$Y$ inertial coordinate system with a minimum tracking error. In other words, the problem is to find the inputs $\Delta T_r$ and $\delta_r$ that minimize the cost function:

$$J = \int_0^{t_f} \left[ K_{11}(X - X_R)^2 + K_{12}(Y - Y_R)^2 \right] dt \tag{40}$$

where $X_R$ and $Y_R$ are coordinates of the reference trajectory, and $K_{11}$ and $K_{12}$ are the cost function weighting factors. The double lane change reference trajectory of Gaussian type is considered

$$X_R(t) = U_0\, t \tag{41}$$

$$Y_R(X_R) = Y_{\max} e^{-(X_R - X_0)^2/\sigma^2} \quad . \tag{42}$$

where $U_0$ is the initial longitudinal velocity, $Y_{max}$, $X_0$, and $\sigma$ are maximum, center, and bandwidth of the Gaussian curve.

*Roll minimization*. The optimal control objective for the roll minimization task is to find the control variables that ensure minimum chassis roll during the double lane change maneuver:

$$J = \int_0^{t_f} \phi^2 dt \,, \tag{43}$$

where $\phi$ is the roll angle.

2) *Control and State Vector Constraints Formulation*

The following inequality and equality constraints of the state variables are considered and realized through the cost function penalty terms.

*Trajectory constraints* (applies to roll minimization only). The road trajectory is constrained as given by (cf. Fig. 6):

$$-Y_{offset} + Y_R(X_R(t)) \le Y(t) \le Y_{offset} + Y_R(X_R(t)) \,, \tag{44}$$

where the reference trajectory function $Y_R(.)$ is given by (42).

*Boundary conditions*. The following boundary constraints on the exit vehicle trajectory are considered (cf. Fig. 1):

$$Y(t_f) = Y(0)$$
$$\frac{dY(t_f)}{dt} = 0 \tag{45}$$

### C. Optimization Results

The C code optimization program for the vehicle control system has been executed on a personal computer with Intel Core Duo CPU (2.00GHz). The terminal time is $t_f$ = 6s and the sampling interval is $\tau$ = 0.003s, so that the number of optimization time intervals is $N$=2000. The number of iterations of the Dai-Yuan conjugate gradient algorithm is $M$=400, while the standard gradient algorithm (with $\eta$ = const.) required $M$ = 4000 iterations for the same level of accuracy [25]. The Jacobians are calculated numerically. The simple first-order Adams method is used, because the higher-order Adams methods have not given notable improvements for the particular example (see [18] for another benchmark example that demonstrates performance gains reached by using the higher-order Adams methods). The algorithm execution time is about 13 min, and it can be reduced by four times if the simplified 7 DOF vehicle dynamics model [5] is used.

1) *Double Lane Change Example*

Figure 1 shows the results of optimization of the front road wheel angle input $\delta_f$ for the trajectory tracking problem (40) and a dry asphalt road ($\mu$ = 1). That is, the optimization task is to find an "ideal" driver steering input referred to the road wheel angle. The results in Fig. 1 indicate that the "ideal" driver can provide accurate tracking of the sharp reference trajectory, where the absolute value of lateral acceleration saturates to its maximum value $a_{ym} \approx 1g = 9.81$ m/s$^2$.

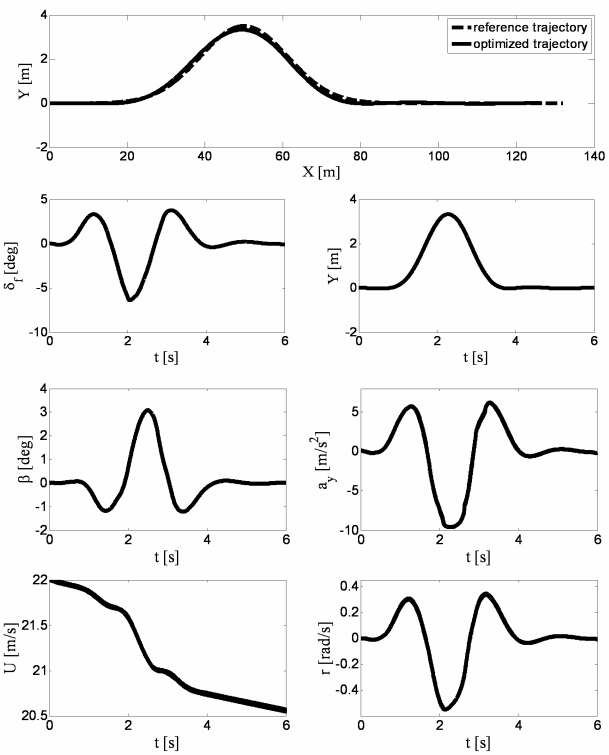


Fig. 1. Front wheel steering optimization results for asphalt road ($\mu$ = 1).

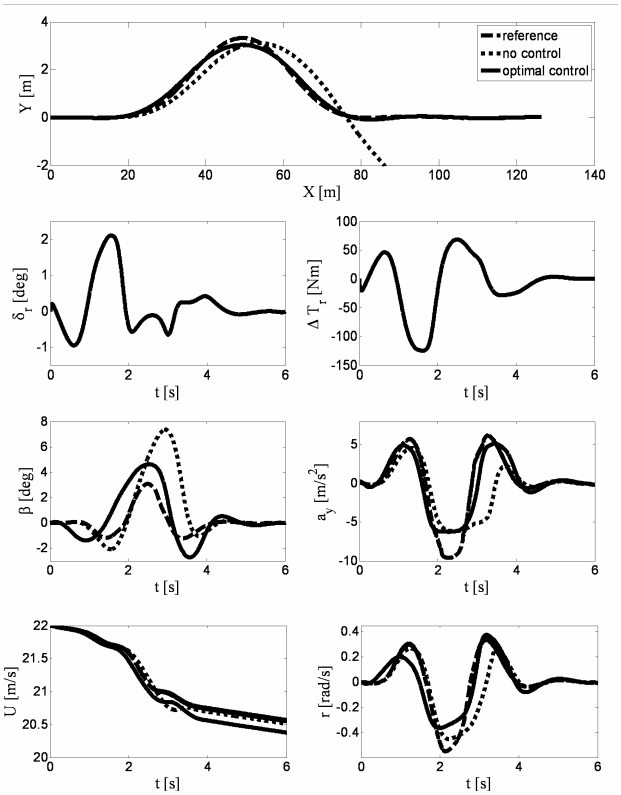


Fig. 2. Optimization results for ARS+TVD control ($\mu$ = 0.6).

Figures 2-5 show the optimization results for reduced tire-road friction coefficient $\mu$ = 0.6 (e.g. wet asphalt surface) and different vehicle dynamics actuators: active rear steering

(ARS), active torque vectoring differential (TVD, $k_{AWSD}$=1.25), combined ARS and TVD, and active limited slip differential (ALSD). The top plot of each figure includes three trajectories: (i) reference trajectory that corresponds to the optimized (reached) trajectory for $\mu$ = 1 (solid line in Fig. 1), (ii) trajectory when no control action is used ($\delta_r$ = 0, $\Delta T_r$ = 0), and (iii) trajectory reached by using the optimal control action. The front steering input $\delta_f$ is taken from Fig. 1 (no driver model is used).

The ARS intervention provides quite accurate trajectory tracking despite the worsen road condition (Fig. 3), thus preserving the driver's feeling of (safe) driving on high-$\mu$ road. The tracking accuracy is somewhat worse for TVD control (Fig. 4), and the peak of side slip angle $\beta$ = atan($V$/$U$) is increased by about 20%. The combined ARS and TVD control (Fig. 2) gives comparable tracking accuracy as in the case of individual ARS control. However, the control effort is reduced when compared to the individual ARD or TVD control. The ALSD cannot compensate for the understeer behavior in the first part of maneuver, and the corresponding tracking error is similar as in the case of passive vehicle (Fig. 5). Namely, since the ALSD can transfer the torque to slower (inner) wheel only, it can only generate understeer (i.e. compensate for oversteer) [16]. Owing to the oversteer compensation, the ALSD effectively stabilizes the vehicle in the second part of maneuver (Fig. 5).

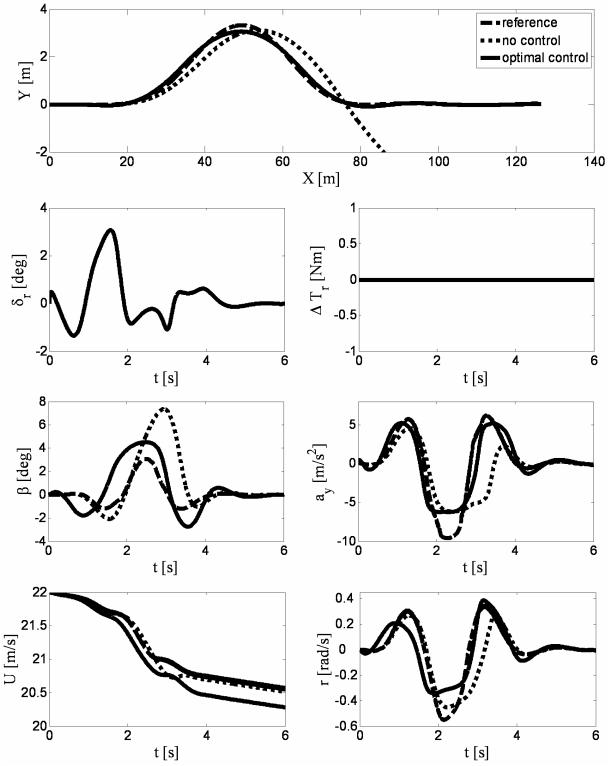


Fig. 3. Optimization results for ARS control ($\mu$ = 0.6).

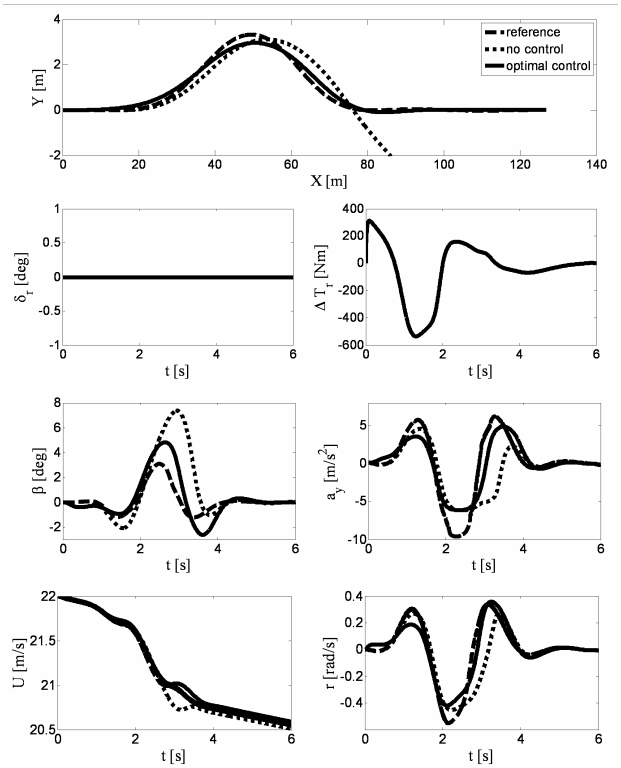


Fig. 4. Optimization results for TVD control ($\mu$ = 0.6).

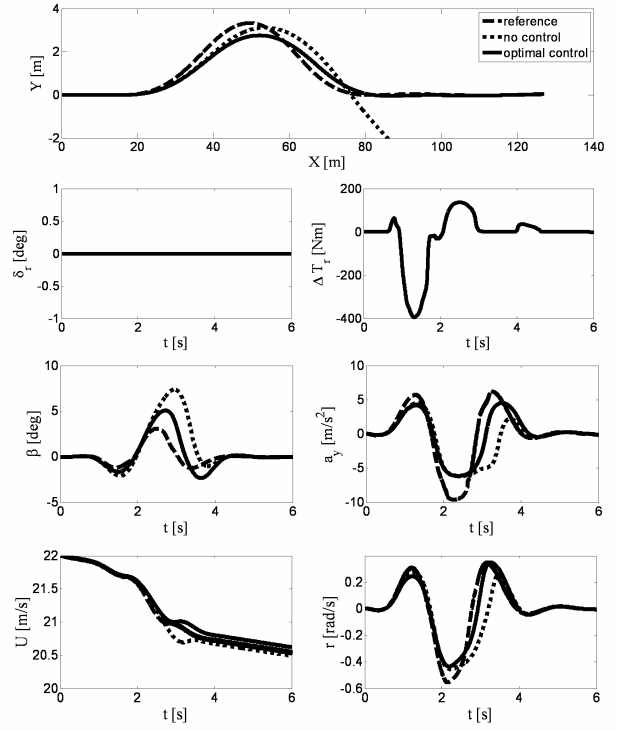


Fig. 5. Optimization results for ALSD control ($\mu$ = 0.6).

Qualitatively similar results to those in Figs. 2-5 have been obtained for lower $\mu$ values, as well (cf. [5]). Also, many other optimization tests have been carried out, e.g. those related to limited side slip angle or limited control input.

2) *Roll Minimization Example*

Figure 6 shows the "ideal" driver optimization results for the same operating conditions as in Fig. 1, but the trajectory tracking cost function (40) is replaced with the roll minimization criterion (43), and the trajectory constraint (44) is taken into account. Figure 7 shows the TVD-based optimization results for the same roll minimization problem, where the steering input $\delta_f$ is taken from Fig. 1. The comparative results from Figs. 6 and 7, including the passive vehicle response, are shown in Fig. 8.

In order to minimize the roll angle $\phi$, the TVD generates understeer around the instant of maximum lateral acceleration ($t \approx 2$ s; Fig. 7) and forces the vehicle to touch the inner trajectory bound. This results in reduction of the yaw rate $r$, the lateral acceleration $a_y$, and finally the roll angle $\phi$ when compared to the passive vehicle (Fig. 8). The roll angle reduction is paid by lower trajectory tracking accuracy.

The "ideal" driver "cuts" the road even more, so that the vehicle touches the trajectory bounds in three points (Fig. 6). This results in a substantial reduction of lateral acceleration and roll angle (Fig. 8). However, the trajectory tracking performance is further deteriorated (Fig. 6).

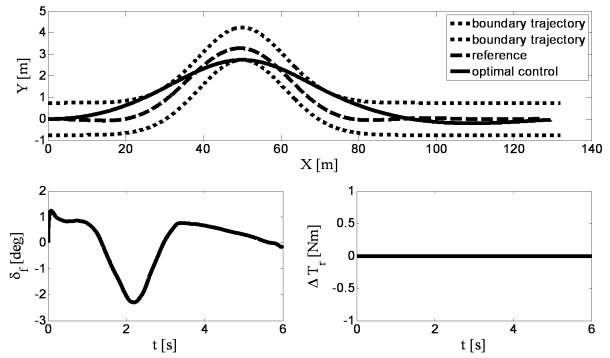


Fig. 6. Front wheel steering optimization results for roll minimization case ($\mu = 1$).

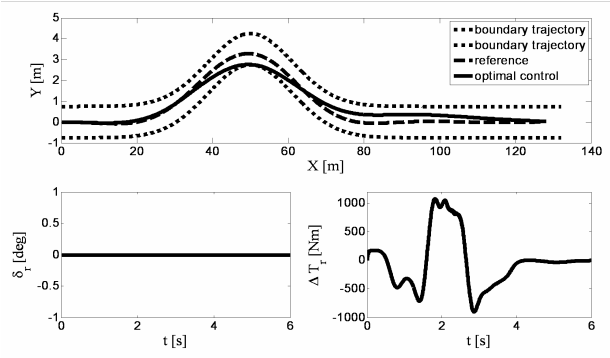


Fig. 7. TVD optimization results for roll minimization case ($\mu = 1$).

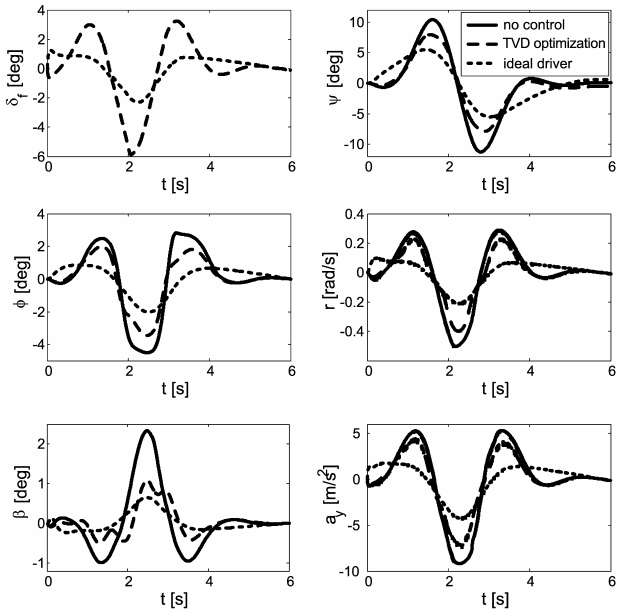


Fig. 8. Additional comparative plots for roll optimizations from Figs. 6 and 7.

## VI. Conclusion

A BPTT-like gradient-based optimal control algorithm has been proposed in the paper. The optimization accuracy can generally be improved by using higher-order Adams numerical integration schemes instead of the basic Euler method. Incorporation of numerical Jacobians enables the algorithm application for complex problems, where analytical Jacobians are difficult to derive. Finally, implementation of advanced conjugate gradient methods, such as Dai-Yuan method, results in substantial improvement of convergence properties and reduction of execution time. The algorithm can be extended for solving minimum-time optimal control problems [12].

The proposed optimal control algorithm has been tested on a relatively complex vehicle dynamics control problem with 18 state variables and two control variables. The optimization results have illustrated favorable features of the algorithm in terms of accuracy (e.g. a few thousands of time grid points can be used), consistent numerical stability, and relatively fast execution. Also, it has been demonstrated that the algorithm can be effectively used to gain insights into the ultimate vehicle dynamics control performance for various actuator configurations.

The future work will be directed towards extending the algorithm for combined parameter and control variable optimization problems, as well as feedback controller parameter optimization for either optimal control formulation or robust control formulation. Also, comparison of the algorithm with other existing methods is a subject of ongoing work and future publications.

## Acknowledgment

The authors would like to thank Dr. Davor Hrovat from Ford Research and Advanced Engineering and Dr. Matt

Hancock from Jaguar Research for their helpful suggestions and/or technical support.

## REFERENCES


[1] J. T. Betts and W. P. Huffman, "Application of sparse nonlinear programming to trajectory optimization," *J. Guidance Control Dynamics*, vol. 15, no. 1, pp. 198-206, 1992.

[2] I. Y. M. Smets, K. J. E. Versyck and J. F. M. Van Impe, "Optimal control theory - A generic tool for identification and control of (Bio-) Chemical Reactors," *Annual Reviews in Control*, vol. 26, pp. 57-73, 2002.

[3] O. von Stryk and M. Schlemmer, "Optimal control of the industrial robot Manutec," in: Bulirsch, R., Kraft D. (eds.), *Computational Optimal Control*, International Series of Numerical Mathematics 115, Basel, Birkhauser, pp. 367-382, 1994.

[4] I. V. Kolmanovsky, and A. G. Stefanopoulou, "Optimal control techniques for assessing feasibility and defining subsystem level requirements: An automotive case study," *IEEE Trans. on Control System Technology*, vol. 9, no. 3, pp. 524-534, May 2001.

[5] J. Kasać, J. Deur, B. Novaković, M. Hancock, and F. Assadian, "Optimization of global chassis control variables," *17th IFAC World Congress*, July 6-11, 2008, Seoul, Korea.

[6] A. E., Bryson, *Dynamic Optimization*, Addison-Wesley, 1999.

[7] E. Polak, *Computational Methods in Optimization*, Academic Press, New York, 1971.

[8] J. T. Betts, *Practical Methods for Optimal Control using Nonlinear Programming*, Society of Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[9] O. von Stryk, and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, pp. 357-373, 1992.

[10] O. von Stryk, "Numerical solution of optimal control problems by direct collocation," in: Bulirsch R., Miele A., Stoer J., Well K.H. (eds.), *Optimal Control-Calculus of Variations, Optimal Control Theory and Numerical Methods*, International Series of Numerical Mathematics, vol. 111, Basel, Birkhauser, pp. 129–143, 1993.

[11] A. L. Schwartz, "Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems," PhD Thesis, University Of California At Berkeley, 1996.

[12] J. Kasać, "Optimal control of nonlinear systems using neural networks," M. Sc. Thesis (in Croatian), University of Zagreb, 1998.

[13] J. Kasać and B. Novaković, "Neural network application to optimal control of nonlinear systems," *Proceedings of 7-th International Conference on Computer Aided Optimum Design of Structures*, May 28-30, 2001, Bologna, Italy, pp. 359-368.

[14] A. E. Bryson and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing Corp, 1975.

[15] P. J. Werbos, "Backpropagation through time: What it does how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560, 1990.

[16] M. Hancock, "Vehicle handling control using active differentials," Ph.D. Thesis, University of Loughborough, UK, 2006.

[17] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I - Nonstiff Problems*, 2nd ed., Springer, Berlin, 1993.

[18] J. Kasać, J. Deur, B. Novaković, and I. Kolmanovsky, "A BPTT-like optimal control algorithm with vehicle dynamics control application," *ASME International Mechanical Engineering Congress & Exposition*, October 31-November 6, 2008, Boston, Massachusetts, USA.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 2006.

[20] J. A. Snyman, *Practical Mathematical Optimization*, Springer, New York, 2005.

[21] C. A. Floudas and P. M. Pardalos, (edit.), *Encyclopedia of Optimization*, Springer, New York, 2008.

[22] T. Tollenaere, "Supersab: Fast adaptive backpropagation with good scaling properties," *Neural Networks*, no. 3, vol. 5, 1990.

[23] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons - From backpropagation to adaptive learning algorithms," *Int. J. Comput. Standards Interfaces*, vol. 16, pp. 265, 1994.

[24] C. G. H. Jondarr, "Back propagation family album," Technical Report C/TR96-05, Macquarie University, August 1996.

[25] J. Kasać, J. Deur, B. Novaković, and I. Kolmanovsky, "A Conjugate Gradient-based BPTT-like Optimal Control Algorithm," *IEEE International Conference on Control Applications*, July 8-10, 2009, Saint Petersburg, Russia.

[26] B. Wie, *Space Vehicle Dynamics and Control*, AIAA Education Series, 1998.