OPTIMAL CONTROL APPLICATIONS AND METHODS
Optim. Control Appl. Meth. 2013; **34**:364–378
Published online 28 March 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/oca.2030

A back propagation through time-like min–max optimal control algorithm for nonlinear systems

Vladimir Milić*,†, Josip Kasać and Dubravko Majetić

Department of Robotics and Automation of Manufacturing Systems, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Zagreb HR-10000, Croatia

SUMMARY

This paper presents a conjugate gradient-based algorithm for feedback min–max optimal control of nonlinear systems. The algorithm has a backward-in-time recurrent structure similar to the back propagation through time (BPTT) algorithm. The control law is given as the output of the one-layer NN. Main contribution of the paper includes the integration of BPTT techniques, conjugate gradient methods, Adams method for solving ODEs and automatic differentiation, to provide an effective, numerically robust algorithm for solving optimal min–max control problems. The proposed algorithm is evaluated on a robotic system with two DOFs. Copyright © 2012 John Wiley & Sons, Ltd.

Received 4 November 2010; Revised 14 November 2011; Accepted 2 March 2012

KEY WORDS: nonlinear systems; min–max optimal control; conjugate gradient methods; back-propagation through time; automatic differentiation

1. INTRODUCTION

Over the past decades, there has been tremendous progress in the development of min–max optimal controllers (or differential game controllers) for applications in many different engineering fields, including aerospace, process control, and robotics, see for example [1–4], to name a few. The main aim is to find control variables and disturbance internal/external variables (with known limits) such that control variables minimize an optimization criterion in the presence of worst-case behavior of disturbance variables. From the standpoint of differential games [5], this means that the control input is the minimizer player, and disturbance input is the maximizer player.

It is well known that min–max control problem require solving a Hamilton–Jacobi–Isaacs (HJI) equation [5, 6]. The analytic solution of this equation is difficult or impossible to find in most cases. In [7], the HJI equation for systems with input constraints is derived. Authors have introduced a two-player policy iteration scheme that results in a frame work that allows the use of NNs to approximate optimal policies and value functions. In [8], an application of NNs to find closed-form representation of the feedback strategies and the value function that solves the associated HJI equation is presented. This approach is computationally expensive since the tuning of NN weights is based on a method of weighted residuals, which includes calculation of Lebesgue integrals over domain $\mathbb{R}^n$ where n is the dimension of the state-space. In other words, the problem is curse of dimensionality when the computational cost increase exponentially with the dimension of the state-space system.

In this paper, an alternative direct approach for solving finite-horizon optimal state feedback control problem with min–max cost is proposed. In contrast to the approaches founded on NN approximation of HJI, the tuning of NN weights is based on direct minimization of the cost

*Correspondence to: Vladimir Milić, Department of Robotics and Automation of Manufacturing Systems, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Ivana Lučića 5, Zagreb HR-10000, Croatia.
†E-mail: vladimir.milic@fsb.hr

function with simultaneous calculation of disturbance vector based on maximization of the same cost function. A conjugate gradient approach is used for minimization/maximization of the cost function, whereas the cost function gradients are calculated exactly using chain rule for ordered derivatives. Because the control, disturbance and state variables are treated as dependent variables (coupled via plant equations), the final algorithm has a backward-in-time structure similar to the back-propagation-through-time (BPTT) [9] algorithm.

The algorithm presented in this paper is an extension of the recent work in [10, 11] toward robust optimal feedback control. In [10] and [11], a conjugate gradient-based BPTT-like algorithm for optimal open-loop control of nonlinear multivariable systems with control and state vectors constraints is presented. The algorithm performance is illustrated on a realistic high-dimensional vehicle dynamics control example. The optimization results have demonstrated favorable features of the algorithm in terms of accuracy, robust numerical stability, and relatively fast execution.

In the approach presented in this paper, the feedback control law is given as the output of the one-layer NN with saturation function as activation function. In this manner, introducing additional penalty function for control vector constraints is avoided. The main reason for using NN is their ability to serve as a generic functional approximators, which makes them useful tools for synthesizing of nonlinear controller [12].

Furthermore, with the aim to enhance the accuracy of the solution, the higher-order Adams numerical integration schemes [13] are used. One of the reasons for using multistep Adams method, among other one-step methods such as Runge–Kutta, is that it can be easily transformed to the causal state-space form. The second reason, when compared with the Runge–Kutta method, is that the vector function $\mathbf{f}(\cdot,\cdot)$ must be calculated only once in the sampling time. In other words, the application of the explicit Adams method has no influence on the algorithm complexity except on extension of the overall state-space system dimension.

In contrast with [11], instead of a finite-difference formula for numerical calculation of Jacobians, an automatic differentiation (AD) technique [14] is used. The main idea behind the AD is the decomposition of differentials provided by the chain rule. It decomposes the original multivariable function into a series of elementary functions, applies the simple rules of differentiation to evaluate the partial derivatives of the elementary functions, and then accumulates them with the chain rule to obtain the derivatives of original function. There are numerous research papers for the application of AD in control algorithms, see for example, [15–17]. Usually, two distinct modes of AD are presented, forward mode and reverse mode. For the application discussed in the present work, the forward mode has been chosen.

The rest of paper is organized as follows; In Section 2, the feedback min–max optimal control problem with the NN controller is formulated. In Section 3, the BPTT-like min–max control algorithm is derived. First, the control problem is transformed into the discrete-time domain using higher-order Adams method. Then, a modified conjugate gradient ascent/descent scheme is presented. The section ends with brief explanation of how the AD is used for Jacobian matrix calculation. Section 4 illustrates the effectiveness of the proposed algorithm on a robotic system example. Finally, Section 5 concludes the paper.

2. PROBLEM FORMULATION

Consider the following affine nonlinear dynamical system of the form

$$\begin{aligned} &\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}) + \mathbf{g}_1(\mathbf{x})\mathbf{u}(t) + \mathbf{g}_2(\mathbf{x})\mathbf{d}(t),\ \mathbf{x}(0) = \mathbf{x}_0, \\ &\mathbf{z}(t) = \begin{bmatrix} \mathbf{g}_3(\mathbf{x}) \\ \mathbf{u}(t) \end{bmatrix},\ \mathbf{f}(0) = 0,\ \mathbf{g}_3(0) = 0, \end{aligned} \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_0}$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the control input, $\mathbf{d}(t) \in \mathbb{R}^{n_d}$ is the vector representing external disturbance with known limits, $\mathbf{z}(t) \in \mathbb{R}^{n_z}$ is the to-be-controlled output or penalty variable. The functions $\mathbf{f}(\cdot)$, $\mathbf{g}_1(\cdot)$, $\mathbf{g}_2(\cdot)$, and $\mathbf{g}_3(\cdot)$ are smooth functions of $\mathbf{x}$. It is assumed that $\mathbf{d}(t) \in \mathcal{L}_\infty[0,\ T]$, $T \geqslant 0$, and $\mathbf{u}(t)$ is the constrained control satisfying $-\mathbf{u}_{\max} \leqslant \mathbf{u}(t) \leqslant \mathbf{u}_{\max}$.

DOI: 10.1002/oca

The objective is to determine a state-feedback controller of the form

$$\mathbf{u}(t) = \sigma(\mathbf{x}(t)), \tag{2}$$

when all the states of the system are available, such that the closed-loop system consisting of (1) and (2) is stable, and the closed-loop system exhibits good disturbance rejection.

In control law (2), $\sigma(\cdot)$ is an n_u-dimensional appropriate generally nonlinear function to be determined. The special form of function $\sigma(\cdot)$ considered in [18] is the one-layer NN with saturation function as activation function. Using such activation function, the control vector constraints $-\mathbf{u}_{\max} \leqslant \mathbf{u}(t) \leqslant \mathbf{u}_{\max}$ are always satisfied. Therefore, an NN-based control law is

$$\mathbf{u}(t) = \sigma(\mathbf{x}(t),\ \mathbf{W}), \tag{3}$$

where $\mathbf{W}$ is the weight matrix of NN, and

$$u_j(t) = \sigma_j(\xi_j),\ \ j = 1, 2, \ldots, n_u, \tag{4}$$

$$\xi_j(t) = \sum_{k=1}^{n_0} w_{jk} x_k(t), \tag{5}$$

$$\sigma_j(\xi_j) = \begin{cases} u_{j,\max}; & \xi_j > u_{j,\max} \\ \xi_j; & -u_{j,\max} \leqslant \xi_j \leqslant u_{j,\max} \\ -u_{j,\max}; & \xi_j < -u_{j,\max}. \end{cases} \tag{6}$$

The parameters w_{jk} are synaptic weights. The performance criteria (7) implicitly depend on w_{jk}, so the matrix $\mathbf{W}$, whose elements are w_{jk}, can be determined according to the performance criteria (7).

In the present paper, we consider the problem of finding a control law (2) such that the Bolza-type cost function

$$J = \Psi_0(\mathbf{x}(T)) + \int_0^T F_0(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t))\mathrm{d}t, \tag{7}$$

for all $T \geqslant 0$, subject to the nonlinear continuous-time plant equation (1), and subject to the final conditions on the state vector

$$\mathbf{b}(\mathbf{x}(T)) = \mathbf{0}, \tag{8}$$

and subject to the control, disturbance and state vector equality constraints

$$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) = \mathbf{0}, \tag{9}$$

and inequality constraints

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \geqslant \mathbf{0}, \tag{10}$$

is minimized for $\mathbf{d}(t)$, which maximize the same cost function (the worst case scenario).

The original idea behind this approach was to formulate the disturbance attenuation problem as a differential game in which $\mathbf{u}(t)$ and $\mathbf{d}(t)$ are two opposing players [19]. It is well known [6] that this problem is equivalent to the solvability of the min–max optimization problem.

The optimization problem (7)–(10) can be reduced as follows

$$J^* = \min_{\mathbf{W}} \max_{\mathbf{d}} \left\{ \Psi(\mathbf{x}(T), T) + \int_0^T F(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t))\mathrm{d}t \right\}, \tag{11}$$

$$\text{subject to: (1)}, \tag{12}$$

DOI: 10.1002/oca

where

$$\Psi(\mathbf{x}(T),T)=\Psi_0(\mathbf{x}(T),T)+\sum_{k=1}^{q}K_{b,k}b_k^2(\mathbf{x}(T)), \tag{13}$$

$$F(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t))=F_0(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t))+\sum_{k=1}^{s}K_{h,k}h_k^2(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t))+ \tag{14}$$

$$+\sum_{k=1}^{r}K_{g,k}g_k^2(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t))H^-(g_k(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t))), \tag{15}$$

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

$$H^-(z)=\begin{cases}0 & \text{if } z\geqslant 0,\\ 1 & \text{if } z<0.\end{cases} \tag{16}$$

The second and third terms on the right-hand side of expression (15) are the penalty functions for the equality and inequality constraints (9) and (10), respectively. Similarly, the second term on the right-hand side of expression (13) is the penalty function for the final boundary condition (8). Note that although the Heaviside step function $H^-(z)$ is not continuous, the penalty terms of the form $z^2H^-(z)$ are continuously differentiable functions. The penalty function coefficients $K_{b,k}$, $K_{h,k}$ and $K_{g,k}$ should be sufficiently large to provide accurate constraints satisfaction.

In order to simplify the higher-order numerical calculation of integral expression in the cost function in [20], an additional state variable is introduced. In our case, this new variable has the following form

$$\dot{\chi}=F(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)),\quad \chi(0)=0. \tag{17}$$

To simplify the derivation of control algorithm, the system (1) with additional state (17) can be rewritten in the following form

$$\dot{\tilde{\mathbf{x}}}(t)=\phi(\tilde{\mathbf{x}}(t),\,\mathbf{u}(t),\,\mathbf{d}(t)),\;\;\tilde{\mathbf{x}}(0)=\tilde{\mathbf{x}}_0, \tag{18}$$

where $\tilde{\mathbf{x}}$ is the new $n=(n_0+1)$-dimensional state vector

$$\tilde{\mathbf{x}}(t)=\begin{bmatrix}x_1 & x_2 & \dots & x_{n_0} & \chi\end{bmatrix}^{\mathrm{T}}, \tag{19}$$

and

$$\phi=\begin{bmatrix}(\mathbf{f}+\mathbf{g}_1\mathbf{u}+\mathbf{g}_2\mathbf{d})^{\mathrm{T}} & F\end{bmatrix}^{\mathrm{T}}. \tag{20}$$

Hence, the continuous-time optimization problem is

$$\begin{aligned}&\min_{\mathbf{W}}\max_{\mathbf{d}}\;\; J(T)=\Psi(\mathbf{x}(T))+\chi(T),\\ &\text{subject to (18).}\end{aligned} \tag{21}$$

3. A CONJUGATE GRADIENT-BASED BACK PROPAGATION THROUGH TIME LEARNING ALGORITHM

This section presents the derivation of the BPTT-like conjugate gradient ascent algorithm (for disturbance vector) and descent algorithm (for control vector) for weights adjustment according to the performance criteria (7).

DOI: 10.1002/oca

3.1. Time discretization by Adams multistep method

Application of BPTT algorithm requires discretization of the system dynamics (18). Assume that the time interval $[0,\ T]$ is divided into $N-1$ sub-intervals of equal lengths. Then, the time grid consists of points $t_i = i\,\tau$ for $i = 0,\ 1,\ 2,\ \ldots,\ N-1$ where $\tau = T/N$ is the time step length.

The discrete-time form of Equation (18) is

$$\tilde{\mathbf{x}}(i+1) = \phi_d(\tilde{\mathbf{x}}(i),\ \mathbf{u}(i),\ \mathbf{d}(i)),\ \ \tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0, \tag{22}$$

where $\mathbf{u}(i)$, $\mathbf{d}(i)$ denote the control and disturbance sequence over the interval $0,\ 1,\ 2,\ \ldots,\ N-1$, respectively, whereas

$$\phi_d(\tilde{\mathbf{x}}(i),\ \mathbf{u}(i),\ \mathbf{d}(i)) = \tilde{\mathbf{x}}(i) + \tau \sum_{j=1}^{k} a_j \phi(i-j+1), \tag{23}$$

is the kth order Adams approximation of the continuous-time state equation for $i = k-1, k, k+1, \ldots$ and initial conditions $\mathbf{x}(0) = \mathbf{x}_0$, $\mathbf{x}(1) = \mathbf{x}_1,\ \ldots,\ \mathbf{x}(k-1) = \mathbf{x}_{k-1}$. The coefficients a_j satisfy

$$a_j = (-1)^{j-1} \int_0^1 \binom{-s}{j-1}\, \mathrm{d}s. \tag{24}$$

The explicit Adams method (23) is a kth order vector difference equation, which can be conveniently transformed into the following discrete-time state-space form

$$\begin{aligned} x_j(i+1) &= x_j(i) + \tau a_1 \phi_j(i) + \tau x_{n+j}(i), \\ x_{rn+j}(i+1) &= a_{r+1}\phi_j(i) + x_{(r+1)n+j}(i), \\ x_{(k-1)n+j}(i+1) &= a_k \phi_j(i), \end{aligned} \tag{25}$$

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

$$\begin{aligned} x_j(k-1) &= x_{j(k-1)}, \\ x_{qn+j}(k-1) &= \sum_{l=q+1}^{k} a_l \phi_j(k-1+q-l), \end{aligned} \tag{26}$$

for $q = 1,\ 2,\ \ldots,\ k-1$. Using the vector notation, the state-space form of the kth order Adams method reads

$$\hat{\mathbf{x}}(i+1) = \hat{\phi}(\hat{\mathbf{x}}(i),\ \mathbf{u}(i),\ \mathbf{d}(i)),\ \ \hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_0, \tag{27}$$

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

$$\hat{\mathbf{x}}(i) = \begin{bmatrix} x_1(i) & x_2(i) & \ldots & x_{n\cdot k-1}(i) & x_{n\cdot k}(i) \end{bmatrix}^{\mathrm{T}}, \tag{28}$$

and

$$\hat{\phi} = \begin{bmatrix} x_1(i) + \tau a_1 \phi_1(i) + \tau x_{n+1}(i) & \ldots & a_k \phi_n(i) \end{bmatrix}^{\mathrm{T}}. \tag{29}$$

Remark 1

The Adams method of the kth order, as a multistep method, requires knowledge of k initial conditions. In this work, to determine these initial conditions, the fourth-order Runge–Kutta method is used. In the case of ODEs system described by (18), the fourth-order Runge–Kutta method is stated as follows. Start with initial point $(t_0,\ \mathbf{x}_0)$ and generate the sequence of approximations using

$$\mathbf{x}(i+1) = \mathbf{x}(i) + \frac{\tau}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \tag{30}$$

DOI: 10.1002/oca

where

$$\begin{aligned}
\mathbf{k}_1 &= \phi(\mathbf{x}(i),\, \mathbf{u}(t_i),\, \mathbf{d}(t_i)),\\
\mathbf{k}_2 &= \phi\left(\mathbf{x}(i) + \frac{\tau}{2}\mathbf{k}_1,\, \mathbf{u}\left(t_i + \frac{\tau}{2}\right),\, \mathbf{d}\left(t_i + \frac{\tau}{2}\right)\right),\\
\mathbf{k}_3 &= \phi\left(\mathbf{x}(i) + \frac{\tau}{2}\mathbf{k}_2,\, \mathbf{u}\left(t_i + \frac{\tau}{2}\right),\, \mathbf{d}\left(t_i + \frac{\tau}{2}\right)\right),\\
\mathbf{k}_4 &= \phi(\mathbf{x}(i) + \tau\mathbf{k}_3,\, \mathbf{u}(t_i + \tau),\, \mathbf{d}(t_i + \tau)).
\end{aligned} \tag{31}$$

From expression (31), it can be seen that the fourth-order Runge–Kutta method requires the calculation in mid-points $\mathbf{u}(i+1/2) \equiv \mathbf{u}(t_i + \tau/2)$ and $\mathbf{d}(i+1/2) \equiv \mathbf{d}(t_i + \tau/2)$. A rough approximation $\mathbf{u}(t_i + \tau/2) \approx \mathbf{u}(t_i)$ and $\mathbf{d}(t_i + \tau/2) \approx \mathbf{d}(t_i)$ would significantly deteriorate the algorithm accuracy. The value of control and disturbance vector in mid-points can be approximated by

$$\mathbf{u}(t_i + \tau/2) \approx \frac{\mathbf{u}(t_i) + \mathbf{u}(t_{i+1})}{2}, \quad \mathbf{d}(t_i + \tau/2) \approx \frac{\mathbf{d}(t_i) + \mathbf{d}(t_{i+1})}{2}. \tag{32}$$

This approximation is verified by simulations and provides satisfactory accuracy in the range of fourth-order approximation. More details on Adams and Runge–Kutta methods can be found in [13].

The discretized Equations (4) and (5) can be expressed as

$$\begin{aligned}
u_j(i) &= \sigma_j(\xi_j(i)),\\
\xi_j(i) &= \sum_{s=1}^{n} w_{js} x_s(i),
\end{aligned} \tag{33}$$

for $j = 1, 2, \ldots, n_u$.

The discrete-time form of the cost function (7) is

$$J = \Psi(\mathbf{x}(N)) + \chi(N). \tag{34}$$

Hence, the final discrete-time min–max control problem is

$$\begin{aligned}
&\min_{\mathbf{W}} \max_{\mathbf{d}} \ (34),\\
&\text{subject to } (27).
\end{aligned} \tag{35}$$

3.2. Gradient calculation

The optimization approach is based on conjugate gradient ascent algorithm for the disturbance vector

$$\mathbf{d}^{(l+1)}(i) = \mathbf{d}^{(l)}(i) - \eta_1^{(l)} \mathbf{s}^{(l)}(i), \tag{36}$$

and on conjugate gradient descent algorithm for the weight matrix $\mathbf{W} = [w_{\mathrm{pq}}]$ of NN

$$\mathbf{W}^{(l+1)} = \mathbf{W}^{(l)} + \eta_2^{(l)} \mathbf{S}^{(l)}, \tag{37}$$

where $i = 0, 1, \ldots, N-1$, $l = 1, 2, \ldots, M$, $p = 1, 2, \ldots, n_u$, $q = 1, 2, \ldots, n$, N is the number of time instants and M is the number of gradient algorithm iterations, and η_1 and η_2 are the learning rates. Search direction matrix and vector are

$$\mathbf{s}^{(l+1)}(i) = -\nabla J\left(\mathbf{d}^{(l+1)}\right) + \beta_1^{(l)} \mathbf{s}^{(l)}(i), \tag{38}$$

$$\mathbf{S}^{(l+1)} = -\nabla J\left(\mathbf{W}^{(l+1)}\right) + \beta_2^{(l)} \mathbf{S}^{(l)}. \tag{39}$$

Note that maximization of the cost function is provided by simple change of sign in front of search direction vector $\mathbf{s}^{(l)}(i)$ in (36).

DOI: 10.1002/oca

The standard method for computing $\eta_j^{(l)}$ for $j = 1, 2$, is the 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 [11]. To avoid these issues, in this work, we use the SuperSAB approach [21], 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 $\eta_j^{(l)}$ (as opposed to a matrix formulation), to avoid discontinuities in disturbance vector $\mathbf{d}$ and optimized control vector $\mathbf{u}$. The modified SuperSAB algorithm for disturbance vector calculation is given by

$$\eta_1^{(l)} = \begin{cases} d^+ \eta^{(l-1)} & \text{if } \frac{\partial J}{\partial \mathbf{d}^{(l)}(i)} \left[\frac{\partial J}{\partial \mathbf{d}^{(l-1)}(i)}\right]^{\mathrm{T}} \geqslant 0, \\ d_1^- \eta^{(l-1)} & \text{if } \frac{\partial J}{\partial \mathbf{d}^{(l)}(i)} \left[\frac{\partial J}{\partial \mathbf{d}^{(l-1)}(i)}\right]^{\mathrm{T}} < 0, \\ d_2^- \eta^{(l-1)} & \text{if } J(\mathbf{d}^{(l)}(i)) \geqslant J(\mathbf{d}^{(l-1)}(i)), \end{cases} \tag{40}$$

and for NN weights calculation

$$\eta_2^{(l)} = \begin{cases} d^+ \eta^{(l-1)} & \text{if } \mathrm{Tr}\left\{\left[\frac{\partial J}{\partial \mathbf{W}^{(l)}}\right]^{\mathrm{T}} \frac{\partial J}{\partial \mathbf{W}^{(l-1)}}\right\} \geqslant 0, \\ d_1^- \eta^{(l-1)} & \text{if } \mathrm{Tr}\left\{\left[\frac{\partial J}{\partial \mathbf{W}^{(l)}}\right]^{\mathrm{T}} \frac{\partial J}{\partial \mathbf{W}^{(l-1)}}\right\} < 0, \\ d_2^- \eta^{(l-1)} & \text{if } J\left(\mathbf{W}^{(l)}\right) \geqslant J\left(\mathbf{W}^{(l-1)}\right), \end{cases} \tag{41}$$

where $0 < d_2^- < d_1^- < 1 < d^+$, and $\eta^{(0)}$ is the initial learning rate.

The scalar value $\beta_j^{(l)}$ for $j = 1, 2$, can be determined by using different methods: Fletcher–Reeves, Polak–Ribiere, Hestenes–Stiefel, Dai–Yuan [22–26]. In this paper, the Dai–Yuan method is applied

$$\beta_1^{(l)} = \min\left\{ \frac{\frac{\partial J}{\partial \mathbf{d}^{(l+1)}(i)} \left[\frac{\partial J}{\partial \mathbf{d}^{(l+1)}(i)}\right]^{\mathrm{T}}}{\mathbf{s}^{(l)}(i) \left[\frac{\partial J}{\partial \mathbf{d}^{(l+1)}(i)} - \frac{\partial J}{\partial \mathbf{d}^{(l)}(i)}\right]^{\mathrm{T}}}, \beta_{\max} \right\}, \tag{42}$$

for disturbance vector calculation, and for NN weights calculation

$$\beta_2^{(l)} = \min\left\{ \frac{\mathrm{Tr}\left\{\left[\frac{\partial J}{\partial \mathbf{W}^{(l+1)}}\right]^{\mathrm{T}} \frac{\partial J}{\partial \mathbf{W}^{(l+1)}}\right\}}{\mathrm{Tr}\left\{\left[\frac{\partial J}{\partial \mathbf{W}^{(l+1)}} - \frac{\partial J}{\partial \mathbf{W}^{(l)}}\right]^{\mathrm{T}} \mathbf{S}^{(l)}\right\}}, \beta_{\max} \right\}. \tag{43}$$

Through comparative analysis of convergence properties of different conjugate gradient methods, it is shown that The Dai–Yuan method is the fastest among the other methods for similar level of accuracy [10].

The parameter $\beta_j^{(l)}$ is limited to $\beta_{\max}$ because the algorithm (40)–(41) for learning rate tuning can induce $\left\| \frac{\partial J}{\partial \mathbf{d}^{(l)}(i)} \right\| \geqslant \left\| \frac{\partial J}{\partial \mathbf{d}^{(l-1)}(i)} \right\|$ in situations when $\frac{\partial J}{\partial \mathbf{d}^{(l)}(i)} \left[\frac{\partial J}{\partial \mathbf{d}^{(l-1)}(i)}\right]^{\mathrm{T}} < 0$, thus leading to a possible algorithm instability if $\beta_j^{(l)}$ is not saturated. If the parameter $\beta_j^{(l)}$ has a constant value, $0 < \beta_j^{(l)} < 1$, then the conjugate gradient algorithm becomes equivalent to a standard gradient algorithm with momentum.

DOI: 10.1002/oca

3.2.1. Gradient with respect to disturbance vector. The gradient of the cost function (34) according to the disturbance vector in the lth iteration of the gradient algorithm and ith sampling interval is given by

$$\frac{\partial J}{\partial d_j(i)} = \sum_{r=1}^{n\cdot k} \frac{\partial J}{\partial \hat{x}_r(N)} \frac{\partial \hat{x}_r(N)}{\partial d_j(i)}, \tag{44}$$

where $j = 1, 2, \ldots, n_d$. By introducing matrices $\mathbf{X}_d(i)$, $\Phi_x(i)$ and $\Phi_d(i)$ with the elements

$$\mathbf{X}_d(i) = \frac{\partial \hat{\mathbf{x}}(N)}{\partial \mathbf{d}(i)}, \quad \Phi_x(i) = \frac{\partial \hat{\phi}(i)}{\partial \hat{\mathbf{x}}(i)}, \quad \Phi_d(i) = \frac{\partial \hat{\phi}(i)}{\partial \mathbf{d}(i)}, \tag{45}$$

the partial derivatives $\partial \hat{x}_r(N)/\partial d_j(i)$ can be calculated by applying chain rule for ordered derivatives on Equation (27)

$$\begin{aligned}
&\mathbf{X}_d(N-1) = \Phi_d(N-1),\\
&\mathbf{X}_d(N-2) = \Phi_x(N-1)\,\Phi_d(N-2),\\
&\mathbf{X}_d(N-3) = \Phi_x(N-1)\,\Phi_x(N-2)\,\Phi_d(N-3),\\
&\mathbf{X}_d(N-4) = \Phi_x(N-1)\,\Phi_x(N-2)\,\Phi_x(N-3)\,\Phi_d(N-4),\\
&\qquad\vdots\\
&\mathbf{X}_d(i) = \Phi_x(N-1)\,\Phi_x(N-2)\cdots\Phi_x(i+1)\,\Phi_d(i).
\end{aligned} \tag{46}$$

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

$$\mathbf{J}_d(i) = \frac{\partial J}{\partial \mathbf{d}(i)}, \quad \mathbf{J}_x(N) = \frac{\partial J}{\partial \hat{\mathbf{x}}(N)}, \tag{47}$$

the final gradient of the cost function (34) according to the disturbance vector can be computed by the following backward-in-time recursive matrix relation

$$\begin{aligned}
\mathbf{D}(i) &= \mathbf{D}(i+1)\,\Phi_x(i+1),\\
\mathbf{X}_d(i) &= \mathbf{D}(i)\,\Phi_d(i),\\
\mathbf{J}_d(i) &= \mathbf{J}_x(N)\,\mathbf{X}_d(i),
\end{aligned} \tag{48}$$

for $i = N-2,\ N-3,\ \ldots,\ 0$ with the initial condition

$$\begin{aligned}
\mathbf{J}_d(N-1) &= \mathbf{J}_x(N)\,\Phi_d(N-1),\\
\mathbf{D}(N-1) &= \mathbf{I}.
\end{aligned} \tag{49}$$

3.2.2. Gradient with respect to weight matrix. The gradient of the cost function (34) according to the weights in the lth iteration of the gradient algorithm and ith sampling interval is given by

$$\frac{\partial J}{\partial w_{\mathrm{pq}}} = \sum_{r=1}^{n_0} \frac{\partial J}{\partial \hat{x}_r(N)} \frac{\partial \hat{x}_r(N)}{\partial w_{\mathrm{pq}}}. \tag{50}$$

The partial derivative $\partial \hat{x}_r(i)/\partial w_{\mathrm{pq}}$ can be calculated as follows:

$$\frac{\partial \hat{x}_r(i)}{\partial w_{\mathrm{pq}}} = \sum_{j=1}^{n_0} \frac{\partial \hat{\phi}_r(i-1)}{\partial \hat{x}_j(i-1)} \frac{\partial \hat{x}_j(i-1)}{\partial w_{\mathrm{pq}}} + \sum_{m=1}^{n_u} \frac{\partial \hat{\phi}_r(i-1)}{\partial u_m(i-1)} \frac{\partial u_m(i-1)}{\partial w_{\mathrm{pq}}}. \tag{51}$$

Further, on the base of (33), the partial derivative $\partial u_m(i)/\partial w_{\mathrm{pq}}$ can be calculated as follows:

$$\frac{\partial u_m(i)}{\partial w_{\mathrm{pq}}} = \mathcal{D}_m(\xi_m(i)) \sum_{j=1}^{n_0} \left[\frac{\partial w_{\mathrm{mj}}}{\partial w_{\mathrm{pq}}} \hat{x}_r(i) + w_{\mathrm{mj}} \frac{\partial \hat{x}_r(i)}{\partial w_{\mathrm{pq}}} \right], \tag{52}$$

DOI: 10.1002/oca

where $\mathcal{D}_m(\xi_m(i))$ is the derivation of function $\sigma_m(\xi_m(i))$ from equation (6)

$$\mathcal{D}_m(\xi_m(i)) = \begin{cases} 0; & \xi_m(i) > u_{j,\max}, \\ 1; & -u_{j,\max} \leqslant \xi_m(i) \leqslant u_{j,\max}, \\ 0; & \xi_m(i) < -u_{j,\max}. \end{cases} \tag{53}$$

The next step is the determination of the initial condition for the previously mentioned recurrent relations. The initial condition of the state vector is independent of the weights w_{jk}, so that

$$\frac{\partial \hat{x}_m(0)}{\partial w_{\mathrm{pq}}} = 0. \tag{54}$$

The second set of the initial condition is

$$\frac{\partial u_m(0)}{\partial w_{\mathrm{pq}}} = \mathcal{D}_m(\xi_m(0)) \sum_{j=1}^{n_0} \frac{\partial w_{\mathrm{mj}}}{\partial w_{\mathrm{pq}}} \hat{x}_r(0). \tag{55}$$

The final gradient of the cost function (34) according to the weights matrix can be computed by the following recursive algorithm:

$$\begin{aligned} \mathbf{X}_W(i) &= \Phi_x(i-1)\,\mathbf{X}_W(i-1) + \Phi_u(i-1)\,\mathbf{U}_W(i-1), \\ \mathbf{U}_W(i) &= \mathcal{D}(i)\,[\mathbf{W}_W\,\hat{\mathbf{x}}(i) + \mathbf{W}\,\mathbf{X}_W(i)], \\ \mathbf{J}_W &= \mathbf{J}_x(N)\,\mathbf{X}_W(N). \end{aligned} \tag{56}$$

where

$$\begin{aligned} &\mathbf{X}_W(i) = \frac{\partial \hat{\mathbf{x}}(i)}{\partial \mathbf{W}},\ \mathbf{U}_W(i) = \frac{\partial \mathbf{u}(i)}{\partial \mathbf{W}},\ \Phi_x(i) = \frac{\partial \hat{\phi}(i)}{\partial \hat{\mathbf{x}}(i)},\ \Phi_u(i) = \frac{\partial \hat{\phi}(i)}{\partial \mathbf{u}(i)}, \\ &\mathbf{J}_W = \frac{\partial J}{\partial \mathbf{W}},\ \mathbf{J}_x(N) = \frac{\partial J}{\partial \hat{\mathbf{x}}(N)},\ \mathbf{W}_W = \frac{\partial \mathbf{W}}{\partial \mathbf{W}}. \end{aligned} \tag{57}$$

3.3. Jacobians calculation

The extended Jacobians $\Phi_x(i)$, $\Phi_d(i)$, $\Phi_u(i)$ and extended gradient $\mathbf{J}_x(N)$ for Adams method can be calculated based on (25) as functions of the basic Jacobians and basic gradient as follows:

$$\begin{aligned} \Phi_x(i) &= \begin{bmatrix} \mathbf{I} + \tau a_1 \frac{\partial \phi(i)}{\partial \hat{\mathbf{x}}(i)} & \tau\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ a_2 \frac{\partial \phi(i)}{\partial \hat{\mathbf{x}}(i)} & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k-1} \frac{\partial \phi(i)}{\partial \hat{\mathbf{x}}(i)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \\ a_k \frac{\partial \phi(i)}{\partial \hat{\mathbf{x}}(i)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\ \Phi_d(i) &= \begin{bmatrix} \tau a_1 \left(\frac{\partial \phi(i)}{\partial \mathbf{d}(i)}\right)^{\mathrm{T}} & a_2 \left(\frac{\partial \phi(i)}{\partial \mathbf{d}(i)}\right)^{\mathrm{T}} & \cdots & a_k \left(\frac{\partial \phi(i)}{\partial \mathbf{d}(i)}\right)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \\ \Phi_u(i) &= \begin{bmatrix} \tau a_1 \left(\frac{\partial \phi(i)}{\partial \mathbf{u}(i)}\right)^{\mathrm{T}} & a_2 \left(\frac{\partial \phi(i)}{\partial \mathbf{u}(i)}\right)^{\mathrm{T}} & \cdots & a_k \left(\frac{\partial \phi(i)}{\partial \mathbf{u}(i)}\right)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \\ \mathbf{J}_x(N) &= \begin{bmatrix} \frac{\partial \Psi}{\partial \mathbf{x}(N)} & 1 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned} \tag{58}$$

These basic Jacobians $\frac{\partial \phi(i)}{\partial \hat{\mathbf{x}}(i)}$, $\frac{\partial \phi(i)}{\partial \mathbf{d}(i)}$, $\frac{\partial \phi(i)}{\partial \mathbf{u}(i)}$ and basic gradient $\frac{\partial \Psi}{\partial \mathbf{x}(N)}$ can be calculated using AD.

DOI: 10.1002/oca

Automatic (or algorithmic) differentiation is now a widely used tool within scientific computing. The standard reference is the book by Griewank *et al*. [14]. AD technique takes the view that the computer code for evaluating the function, no matter how complicated, can be broken down into composition of elementary arithmetic operations involving just one or two arguments at a time. Two-argument operations include addition, multiplication, division and the power operation. Examples of single-argument operations include the trigonometric, exponential, and logarithmic functions [22]. Another common ingredient of the various AD tools is their use of the *chain rule*. This is the well-known rule from elementary calculus.

Over the past decades, extensive research activities led to a thorough understanding and analysis of two basic modes of AD: the *forward* and *reverse* modes. For the application discussed in the present work, the forward mode has been chosen for the efficient computation of Jacobians and gradient (58).

A variety of tools exists for AD of the standard programming languages including: ADIFOR, TAF/TAMC, TAPENADE, ADO1, ADIC, ADOL-C. In this paper, TOMLAB/MAD (Tomlab Optimization Inc., San Diego, CA, USA) [27, 28] mathematical software is used. It is important to emphasize that software tools for AD do not require the user to break down the code for evaluating the function into its elements. Identification of intermediate quantities and construction of the computational graph is carried out, explicitly or implicitly, by the software tool itself.

Application of AD comparing with numerical differentiation [11] provides significant reduction of the algorithm computational time.

4. SIMULATION EXAMPLE

4.1. The dynamic model of the robot manipulator

This section presents the design process of the proposed min–max optimal control algorithm on a two-DOF robotic system to demonstrate its efficiency. The mathematical model, which describe the dynamics of the robot manipulator with two revolute joints in horizontal plane [29], is given by

$$\begin{aligned} &\theta_1\ddot{q}_1 + (\theta_3 C_{21} + \theta_4 S_{21})\ddot{q}_2 - \theta_3 S_{21}\dot{q}_2^2 + \theta_4 C_{21}\dot{q}_2^2 = T_1 + d_1,\\ &(\theta_3 C_{21} + \theta_4 S_{21})\ddot{q}_1 + \theta_2\ddot{q}_2 + \theta_3 S_{21}\dot{q}_1^2 - \theta_4 C_{21}\dot{q}_1^2 = T_2 + d_2, \end{aligned} \tag{59}$$

where $C_{21} = \cos(q_2 - q_1)$, $S_{21} = \sin(q_2 - q_1)$, $\theta_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1$, $\theta_2 = m_2 l_{c2}^2 + I_2$, $\theta_3 = m_2 l_1 l_{c2}\cos(\delta)$ and $\theta_4 = m_2 l_1 l_{c2}\sin(\delta)$. The numerical values of these parameters are: $l_1 = 1$ m, $l_{c1} = l_{c2} = 0.5$ m, $m_1 = 3$ kg, $m_2 = 4$ kg, $I_1 = 1\ \mathrm{kg\,m^2}$, $I_2 = 2\ \mathrm{kg\,m^2}$, $\delta = 0$.

Equations (59) can be written in following compact form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{u} + \mathbf{d}, \tag{60}$$

where $\mathbf{q} = [q_1\ q_2]^{\mathrm{T}}$ is the vector of joint positions, $\mathbf{u} = [T_1\ T_2]^{\mathrm{T}}$ is the control vector, $\mathbf{d} = [d_1\ d_2]^{\mathrm{T}}$ is the disturbance vector, and

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} \theta_1 & \theta_3 C_{21} + \theta_4 S_{21} \\ * & \theta_2 \end{bmatrix},\ \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 & (\theta_4 C_{21} - \theta_3 S_{21})\dot{q}_2 \\ (\theta_3 S_{21} - \theta_4 C_{21})\dot{q}_1 & 0 \end{bmatrix}.$$

Finally, introducing the state vector $\mathbf{x} = [\mathbf{q}^{\mathrm{T}}\ \dot{\mathbf{q}}^{\mathrm{T}}]^{\mathrm{T}}$, the state space formulation of (60) is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}(\mathbf{q})^{-1}(\mathbf{u} + \mathbf{d} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}) \end{bmatrix}. \tag{61}$$

4.2. Optimization problem formulation and simulation results

In the considered problem, the cost function in the lth iteration of the algorithm has the following form:

$$J^{(l)} = \chi^{(l)}(N) + \sum_{k=1}^{r} K_{g,k}(g_k^{(l)})^2(\mathbf{x}^{(l)}(i), \mathbf{u}^{(l)}(i), \mathbf{d}^{(l)}(i))H^{-}(g_k^{(l)}(\mathbf{x}^{(l)}(i), \mathbf{u}^{(l)}(i), \mathbf{d}^{(l)}(i))), \tag{62}$$

DOI: 10.1002/oca

where

$$\chi^{(l)}(i+1) = Q_1(x_1^{(l)}(i) - x_{1d})^2 + Q_2(x_2^{(l)}(i) - x_{2d})^2 + Q_3\|\mathbf{u}^{(l)}(i)\|^2 - \gamma^2\|\mathbf{d}^{(l)}(i)\|^2,$$

$$g_1^{(l)} = d_1^{(l)}(i) + d_{\max}, \quad g_2^{(l)} = d_{\max} - d_1^{(l)}(i),$$

$$g_3^{(l)} = d_2^{(l)}(i) + d_{\max}, \quad g_4^{(l)} = d_{\max} - d_2^{(l)}(i).$$

The initial robot states are $x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0$, and the desired joint positions are $x_{1\mathrm{d}} = x_{2\mathrm{d}} = \pi/4$ rad. The control and disturbance vector constraints are $\mathbf{u}_{\max} = 100$ N m, $\mathbf{d}_{\max} = 25$ N m, respectively.

In this case our controller is in the following form

$$\begin{aligned} u_j &= \sigma_j(\xi_j), \quad j = 1, 2, \\ \xi_j &= \sum_{k=1}^{n_0} w_{jk}x_k + w_{j0}, \end{aligned} \tag{63}$$

or in matrix form

$$\mathbf{u} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{w}_0), \tag{64}$$

where w_{j0} are threshold offsets. The thresholds are included as the first column of weight matrix. To accommodate this, the vectors $\mathbf{x}$ and $\sigma(\cdot)$ need to be augmented by placing 1 as their first element. In this way, any tuning of weight matrix then includes tuning of thresholds as well.

The MATLAB (MathWorks, Natick, MA, USA) program for min–max optimal control of robotic system is executed by utilizing a portable (notebook) PC computer with Intel Core Duo CPU (2.00 GHz). The terminal time is $T = 2$ s and the number of optimization time intervals is $N = 2000$ so that the sampling interval is $\tau = 0.001$ s. The conjugate gradient Dai–Yuan method is used by default. The Jacobians are calculated by AD method. The fourth-order Adams method is used.

Simulation results are shown in Figures 1–3. We can see in Figure 1 that fast convergence to the desired joint positions is achieved. We can observe from Figure 2 that the proposed controller yields control and worst case disturbance inputs $\mathbf{u} \leqslant \mathbf{u}_{max} = 100$ Nm and $\mathbf{d} \leqslant \mathbf{d}_{\max} = 25$ Nm. Further, Figure 3 illustrates the dependence of the weight parameters on number of iterations.

Furthermore, we should like to compare our control algorithm with standard proportional-derivative (PD) controller. For comparison, a standard saturated PD controller is given by

$$\mathbf{u} = \sigma\left[-\mathbf{K}_p(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_v\dot{\mathbf{q}}\right], \tag{65}$$

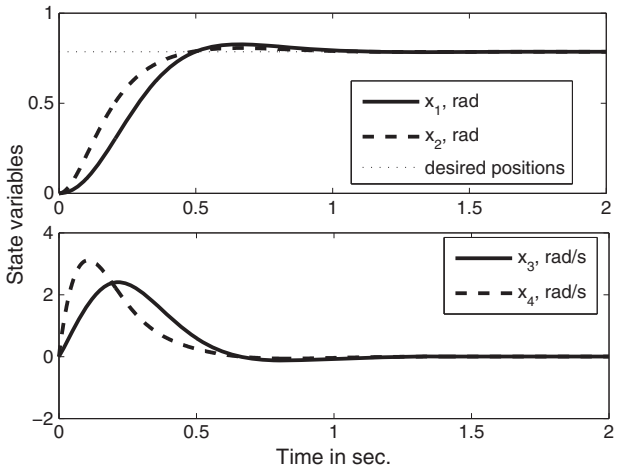


Figure 1. Time dependence of the state variables.

DOI: 10.1002/oca

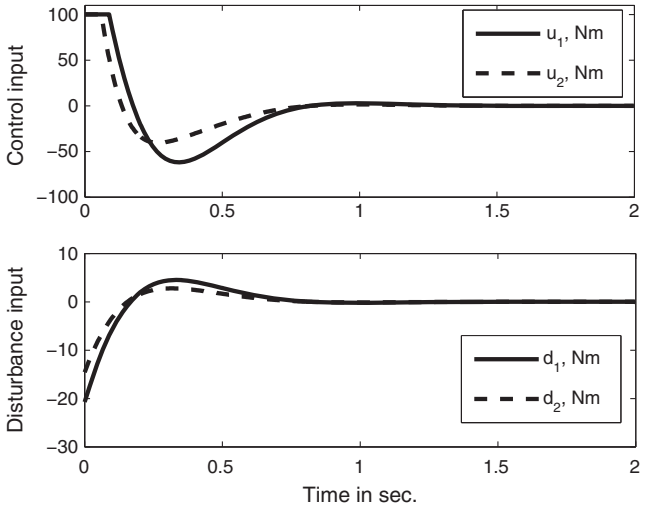


Figure 2. Time dependence of the control and disturbance variables.

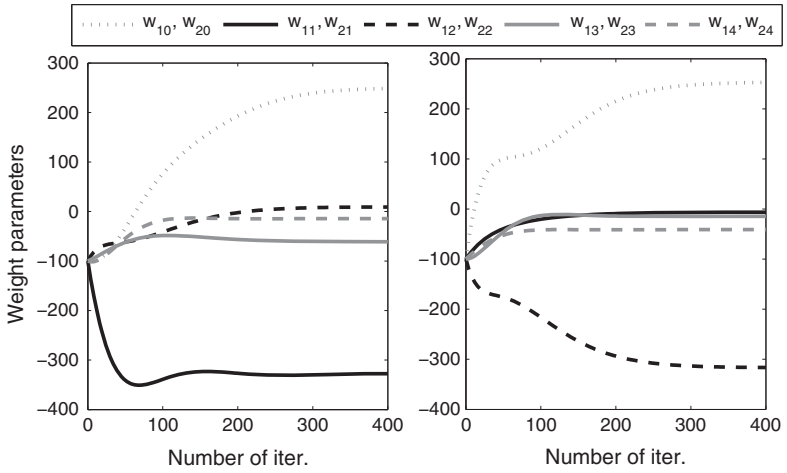


Figure 3. Dependence of the weight parameters on number of iterations.

where the vector $\mathbf{q}_d$ corresponds to the desired joint position. $\mathbf{K}_p$ and $\mathbf{K}_v$ are symmetric positive definite matrices and are commonly referred to as position gain and velocity (or derivative) gain, respectively.

Note that the controller (64) can be rewritten as

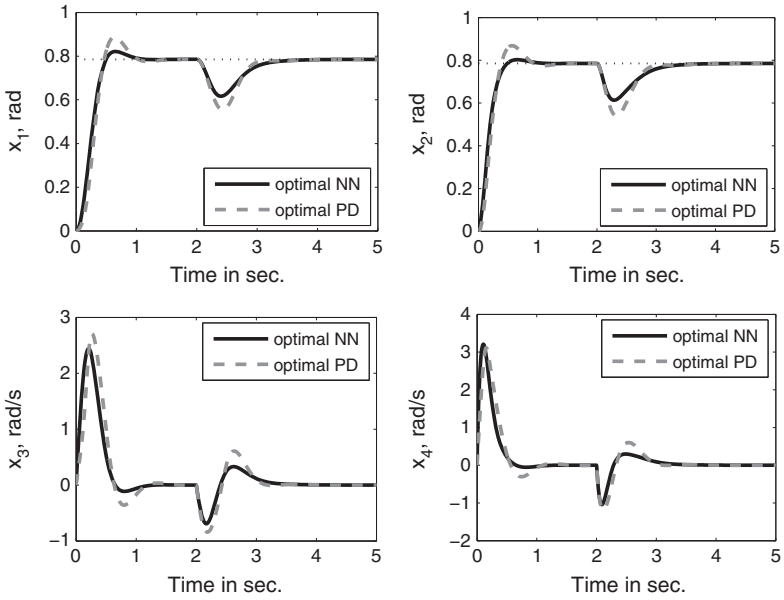
$$\mathbf{u} = \sigma\left[-\mathbf{W}_1(\mathbf{q} - \mathbf{q}_d) - \mathbf{W}_2\dot{\mathbf{q}}\right], \tag{66}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are submatrices of weight matrix $\mathbf{W} = [\mathbf{W}_1 \;\; \mathbf{W}]$, and $\mathbf{w}_0 = \mathbf{W}_1\mathbf{q}_d$. The main difference of NN controller in comparison with saturated PD controller is that $\mathbf{W}_1$ and $\mathbf{W}_2$ are arbitrary matrices, whereas $\mathbf{K}_p$ and $\mathbf{K}_v$ are usually positive definite diagonal matrices. With the aim to provide consistent comparison, the diagonal entries of matrices $\mathbf{K}_p$ and $\mathbf{K}_v$ are chosen same as diagonal elements of matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively, that is, $\mathbf{K}_p = \text{diag}\{-w_{11}, \; -w_{22}\}$, and $\mathbf{K}_v = \text{diag}\{-w_{13}, \; -w_{24}\}$.

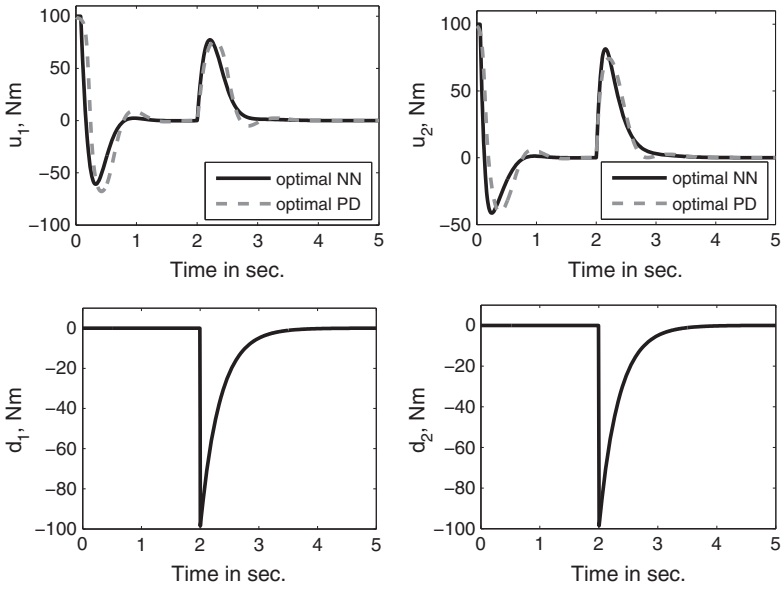
The system is experiencing a disturbances $d_1(t) = d_2(t) = -50(1 + \text{sign}(t-2))e^{-3(t-2)}$.

Figure 4(a) and (b) shows the time dependence of states trajectories and control signals in this test. From the figure, it can be seen that optimal NN controller provides better transient and disturbance attenuation properties than saturated PD controller with optimal values of diagonal gains.

DOI: 10.1002/oca



(a) Time dependence of state variables.



(b) Time dependence of control and disturbance variables.

Figure 4. Comparative diagrams between our NN optimal controller and PD controller for a robotic system exposed to external disturbance.

4.3. Discussion on numerical robustness and computing efficiency

The algorithm contains several free parameters such as the time step τ, and the parameters η_0, d_1^-, d_2^-, d^+, and $\beta_{\max}$ of the modified conjugate gradient methods. The numerical stability is not affected by decreasing the time step τ. But, naturally, there is a maximal value of the time step τ, which guarantees numerical stability of the Adams integration methods.

With respect to parameters d_1^-, d_2^-, d^+, and $\beta_{\max}$, the tuning region is known in advance, whereas the initial learning rate η_0 is dependent on specific optimization problem. The algorithm convergence is more sensitive to the choice of parameter $\beta_{\max}$ than to the choice of other parameters.

DOI: 10.1002/oca

The larger the parameter $\beta_{\max}$, the faster the convergence. However, the parameter $\beta_{\max}$ should be limited to avoid numerical instabilities.

The algorithm convergence is largely insensitive to the choice of dilatation parameters d^+, d_1^- and whether they lie in the intervals $0.85 \leqslant d^- \leqslant 0.95$ and $1.05 \leqslant d^+ \leqslant 1.15$. Similar values of dilatation coefficients are reported in the NNs literature [21]. A recommended relationship between dilatation parameters is $d^+ \approx 1/d_1^-$. The limit value of $\beta_j^{(l)}$, 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. It is illustrated in [10] that the conjugate gradient methods are less sensitive to the choice of initial learning rate η_0 than the standard gradient algorithm. The Dai–Yuan conjugate gradient method reaches a similar level of solution accuracy for various initial learning rate values η_0, whereas for the standard gradient algorithm, the choice of $\eta = \eta_0$ largely affects the algorithm convergence and can cause numerical instabilities.

Finally, it has been found out that the same set of optimal control algorithm parameters ($\eta_0 = 0.1$, $d_1^- = 0.95$, $d_2^- = (d_1^-)^2$, $d^+ = 1.05$, $\beta_{\max} = 1.0$) can be applied with a favorable accuracy and numerical robustness to various dynamic systems (vehicle dynamics models [10], two-DOF robot, Goddard rocket, chemical reactor, etc.).

4.4. Discussion on closed-loop stability

It is hard to define the closed-loop stability during the finite terminal time for which the optimization problem is made. For the case of infinite horizon optimal feedback control, we can get approximate solution for enough large value of terminal time and then check stability using standard methods (if it is possible for some particular dynamical systems) or using simulation test during much larger time interval.

Let us note once again that the control law expression in (64) may be rewritten in (66). That is, the proposed controller is equivalent to the saturated PD controller (65). If we introduce the condition that $\mathbf{W}_1$ and $\mathbf{W}_2$ are positive definite diagonal matrices, then global asymptotic stabilization of the closed-loop system with bounded inputs is guaranteed. The proof is reported in [30].

5. CONCLUSION

The algorithm presented in this paper is an extension of the recent work in [10, 11]. The presented method is combined with NNs to obtain the optimal feedback controllers of nonlinear systems that are affine in input with actuator saturation, where the disturbance is modeled by malicious opponent. The implementation of the proposed method on a robotic system has been used to demonstrate the theoretical development of this paper.

Although the individual methods such as BPTT technique, conjugate gradient optimization algorithms, Adams method for solving ODEs, and AD are known from the literature, in our approach, they are integrated together to provide an effective, novel algorithm for numerical solution of the min–max optimal control problems.

Comparison of the algorithm with other existing methods is a subject of ongoing work and future publications. Also, in future work, the proposed static feedback controller will be extended with dynamic NN observer providing optimal output feedback control in the presence of measurement noise.

REFERENCES

1. Kreim H, Kugelmann B, Pesch HJ, Breitner MH. Minimizing the maximum heating of a re-entering space shuttle: an optimal contol problem with multiple control constraints. *Optimal Control Applications and Methods* 1996; **17**(1):45–69.
2. Chen BS, Wu CS, Uang HJ. A minimax tracking design for wheeled vehicles with trailer based on adaptive fuzzy elimination scheme. *IEEE Transactions on Control Systems Technology* 2000; **8**(3):418–434.
3. Qingguo L, Payandeh S. Planning for dynamic multiagent planar manipulation with uncertainty: a game theoretic approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 2003; **33**(5):620–626.

DOI: 10.1002/oca

4. Gruber JK, Ramirez DR, Alamo T, Bordons C, Camacho CF. Control of a pilot plant using QP based min–max predictive control. *Control Engineering Practice* 2009; **17**(11):1358–1366.
5. Isaacs R. *Differential Games. A Mathematical Theory with Application to Warfare and Pursuit, Control and Optimization*. John Wiley and Sons, Inc.: New York, 1965.
6. Basar T, Bernard P. $\mathcal{H}_\infty$ *Optimal Control and Related Minimax Design Problems*, (2nd edn). Birkhäuser: Boston, MA, 1995.
7. Abu-Khalaf M, Lewis FL, Huang J. Policy iterations on the Hamilton–Jacobi–Isaacs equation for $\mathcal{H}_\infty$ state feedback control with input saturation. *IEEE Transactions on Automatic Control* 2006; **51**(12):1989–1995.
8. Abu-Khalaf M, Lewis FL, Huang J. Neurodynamic programming and zero-sum games for constrained control systems. *IEEE Transactions on Neural Networks* 2008; **19**(7):1243–1252.
9. Werbos PJ. Backpropagation through time: what it does and how to do it. *Proceedings of IEEE* 1990; **78**(10):1550–1560.
10. Kasać J, Deur J, Novaković B, Kolmanovsky I. A conjugate gradient-based BPTT-like optimal control algorithm. *IEEE International Conference on Control*, Saint Petersburg, Russia, 2009; 861–866.
11. Kasać J, Deur J, Novaković B, Kolmanovsky I, Assadian F. A conjugate gradient-based BPTT-like optimal control algorithm with vehicle dynamics control application. *IEEE Transactions on Control Systems Technology* 2011; **19**(6):1587–1595.
12. Plumer ES. Optimal control of terminal processes using neural networks. *IEEE Transactions on Neural Networks* 1996; **7**(2):408–418.
13. Hairer E, Nørsett SP, Wanner G. *Solving Ordinary Differential Equations I—Nonstiff Problems*, (2nd Revised edn). Springer-Verlag: Berlin, 2008.
14. Griewank A, Walther A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, (2nd edn). SIAM: Philadelphia, USA, 2008.
15. Campbell SL, Moore E, Zhong Y. Utilization of automatic differentiation in control algorithms. *Computer Physics Communications* 1994; **39**(5):1047–1052.
16. Griese R, Walther A. Evaluating gradients in optimal control: continuous adjoints versus automatic differentiation. *Journal of Optimization Theory and Applications* 2004; **122**(1):63–86.
17. Walther A. Automatic differentiation of explicit Runge–Kutta methods for optimal control. *Computational Optimization and Applications* 2007; **36**(1):83–108.
18. Kasać J, Novaković B. Optimal feedback control of nonlinear systems with control vector constraints. *Strojarstvo* 2001; **43**(4-6):133–141.
19. Tsiotras P, Corless M, Rotea M. An $\mathcal{L}_2$ disturbance attenuations solution to the nonlinear benchmark problem. *International Journal of Robust and Nonlinear Control* 1998; **8**(4-5):311–330.
20. Fan LT. *The Continuous Maximum Principle*. John Wiley & Sons: New York, 1966.
21. Tollenaere T. SuperSAB: fast adaptive backpropagation with good scaling properties. *Neural Networks* 1990; **3**(5):561–573.
22. Nocedal J, Wright SJ. *Numerical Optimization*. Springer Science + Business Media, LLC: New York, 2006.
23. Snyman JA. *Practical Mathematical Optimization*. Springer Science + Business Media, Inc.: New York, 2005.
24. Dai YH, Yuan Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization* 1999; **10**(1):177–182.
25. Polak E, Ribiere G. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle, série rouge* 1969; **3**(1):35–43.
26. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**(6):409–436.
27. Forth SA. An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. *ACM Transactions on Mathematical Software* 2006; **32**(2):195–222.
28. Shampine LF, Ketzscher R, Forth SA. Using AD to solve BVPs in MATLAB. *ACM Transactions on Mathematical Software* 2005; **31**(1):79–94.
29. Kelly R, Santibáñez V, Loría A. *Control of Robot Manipulators in Joint Space*. Springer-Verlag: London, 2005.
30. Zavala-Río A, Santibáñez V. A natural saturating extension of the PD-with-desired-gravity-compensation control law for robot manipulators with bounded inputs. *IEEE Transactions on Robotics* 2007; **23**(2):386–391.

DOI: 10.1002/oca