# An optimal control synthesis of nonlinear systems by neural network backpropagation-through-time algorithm

***Kasac J.; Novakovic B.; Majetic D. & Brezak D.***

**Faculty of Mechanical Engineering and Naval Architecture**
**University of Zagreb, Zagreb, Croatia**
***Email: {josip.kasac, danko.brezak, dubravko.majetic, branko.novakovic}@fsb.hr***

***Abstract:*** *This work presents a new numerical algorithm for the time optimal control of nonlinear multivariable systems with control and state vectors constraints. The algorithm is based on the backpropagation-through-time algorithm (BPTT), which is used as a learning algorithm for recurrent neural networks. Also, a heuristic algorithm for the time optimal control is presented. This algorithm is based on the characteristics of penalty functions for control and state vectors constraints. The proposed algorithms are especially suitable for treating complicate state vector constraints. Also, the proposed algorithms provide better convergence properties then numerical algorithms based on conversion of optimal control problem into a nonlinear programming one. The algorithms are applied on the problem of the time optimal robot control with the state vector constraints in the form of obstacle avoidance.*

***Key words:*** *optimal control, nonlinear systems, robot control, gradient algorithm, backpropagation-through-time algorithm.*

## 1. Introduction

There are many cases where the time optimal control has been applied in industry, for example in the control of industrial robots, where increasing of the speed of motion is of primary importance, and minimum-time control is an attractive control strategy for this purpose.
As it is already known from the classical optimal control theory (Bryson & Ho, 1969), the solution of the optimal control problem requires the solution of the first-order stationary conditions (two-point boundary-value problem), which can be solved analytically only for very simple problems. A review of the different approaches to the numerical solution of optimal control problems is given in survey paper (Sargent, 2000). Sargent recognize essentially three approaches to solve numerically optimal control problems: a) numerical solution of the two-point boundary value problem given by the necessary conditions, b) complete discretization of the problem, converting it into a nonlinear programming one, and c) finite parameterization of the control trajectory, again converting the problem into nonlinear programming.
The problem of state vectors constraints considerably complicates the solution of the problem both from the theoretical and numerical aspects. The penalty functions for state vectors constraints can be very complicated and impractical, particularly in robot control (avoidance of obstacles, cooperative robots work, etc).
A standard method for reducing the optimal control problem to a nonlinear programming one is adding the penalty functions for state and control vector constraints and plant equation constraints to the cost function, and optimizing the total cost function according to the control and state vectors. In such a way formulated problem has very slow convergence due to additional equality constraints for plant equation.
In this paper a new gradient-based numerical algorithm for time optimal control of nonlinear multivariable systems with control and state vectors constraints is proposed. This algorithm avoids inclusion of plant equation constraints into cost function and so provides better convergence properties. Further, the mentioned gradient-based approach provides an obvious geometric interpretation of convergence properties of the optimal solution. The approximations of the penalty function gradient means a certain deviation from the exact direction of the overall cost function gradient. However, the approximation of the gradient does not mean the approximation of the optimal solution but only slower convergence toward the optimal solution. This fact provides much easier deal with complicate state vectors constraints what is demonstrated in (Kasac & Novakovic, 2001a) on the example of cooperative work of two robots.
Further, mentioned approach provides numerical solution of a wide class of non-standard optimal control problems like minimum time control where initial and final conditions are parameterised by a coordinate transformation (Kasac & Novakovic, 2001b).
This chapter is organized as follows. Section 2 presents a formulation of an optimal control problem, and derivation of the corresponding algorithm for solving of this problem. An algorithm for a time optimal control has been considered in section 3. A

minimum-time robot control problem with state and control vectors constraints is presented in section 4. Finally, the comments and the conclusions are emphasized by section 5.

## 2. Optimal control problem formulation

### *2.1 Discrete-time optimal control problem*

A discrete nonlinear time optimal control problem is considered. The problem is to find control vector u($i$) and sampling interval $\tau = t_f / N$, where $t_f$ is terminal time, that minimizes the cost function

$$J_0 = \min_{u(i)} \tau \sum_{i=0}^{N-1} \hat{F}(x(i), u(i)), \tag{1}$$

subject to the constraints defined by the plant equations

$$x(i+1) = f(x(i), u(i)), \tag{2}$$

then the initial and final conditions of the state vector

$$x(0) = x_0, \quad x(N) = x_f, \tag{3}$$

subject to the control and state vector inequality constraints

$$g(x(i), u(i)) \geq 0, \tag{4}$$

and equality constraints

$$h(x(i), u(i)) = 0, \tag{5}$$

for $i=0,\ 1, \ldots,\ N-1$, where $N$ is number of sampling intervals, $x(i)$ is $n$-dimensional state vector, $u(i)$ is $m$-dimensional control vector, $g(i) \equiv g(x(i), u(i))$ is $p$-dimensional vector function of inequality constraints, and $h(i) \equiv h(x(i), u(i))$ is $q$-dimensional vector function of equality constraints.

### *2.2 Penalty method approach*

The next step is the expansion of the cost function (1) by adding penalty functions for constraints

$$J = J_0 + J_1 + J_2 + J_3, \tag{6}$$

where

$$J_1 = K_B \sum_{k=1}^{n} \left(x_k(N) - x_k(t_f)\right)^2, \tag{7}$$

is the penalty function for the final boundary condition, $K_B$ is the coefficient of the penalty function and $x_k(t_f)$ is the $k$-th component of the state vector at the terminal time. Further,

$$J_2 = \hat{K}_V \sum_{i=0}^{N-1} \sum_{k=1}^{N_p} g_k^2(x(i), u(i)) H^-(g_k(x(i), u(i))), \tag{8}$$

is the penalty function for inequality constraints (4), where $H^-(g_k(x(i), u(i)))$ is Heaviside step function defined as follows

$$H^-(g_k(x(i), u(i))) = \begin{cases} 0, & \text{if} \quad g_k(x(i), u(i)) \geq 0, \\ 1, & \text{if} \quad g_k(x(i), u(i)) < 0, \end{cases} \tag{9}$$

while $\hat{K}_V$ is the coefficient of the penalty function for inequality constraints. Finally, there is the penalty function for equality constraints (5)

$$J_3 = \hat{K}_E \sum_{i=0}^{N-1} \sum_{k=1}^{N_q} h_k^2 \left(x(i), u(i)\right), \tag{10}$$

where $\hat{K}_E$ is the coefficient of the penalty function of equality constraints.
If $\hat{K}_V = \tau K_V$ and $\hat{K}_E = \tau K_E$, the equation (6) can be expressed as

$$J = \tau \sum_{i=0}^{N-1} F\left(x(i), u(i)\right) + K_B \sum_{k=1}^{n} \left(x_k(N) - x_k\left(t_f\right)\right)^2, \tag{11}$$

where

$$F\left(x(i), u(i)\right) = \hat{F}\left(x(i), u(i)\right) + K_V \sum_{k=1}^{N_p} g_k^2 \left(x(i), u(i)\right) H^- \left(g_k\left(x(i), u(i)\right)\right) + K_E \sum_{k=1}^{N_q} h_k^2 \left(x(i), u(i)\right), \tag{12}$$

so that the problem (1) to (5) can be expressed in the following form

$$J = \min_{u(t)} \left( \tau \sum_{i=0}^{N-1} F\left(x(i), u(i)\right) + J_1 \right), \tag{13}$$

$$x(i+1) = f\left(x(i), u(i)\right), \quad x(0) = x_0. \tag{14}$$

The gradient descent algorithm according to the control vector is as follows:

$$u^{(l+1)}(i) = u^{(l)}(i) - \eta \frac{\partial J}{\partial u^{(l)}(i)}, \tag{15}$$

where $i = 0, 1, \ldots, N-1$, $l = 1, 2, \ldots, M$, while $\eta$ is the convergence coefficient, index $l$ presents the $l$-th iteration of the gradient algorithm, and $M$ is the number of iterations of the gradient algorithm.

## *2.3 Gradient calculation*

The gradient of the cost function (11) in $l$-th iteration of the gradient algorithm is

$$\frac{\partial J}{\partial u_k(j)} = \tau \sum_{i=0}^{N-1} \frac{\partial F(i)}{\partial u_k(j)} + \frac{\partial J_1}{\partial u_k(j)}, \tag{16}$$

where $F(i) \equiv F\left(x(i), u(i)\right)$.

Remaining from the sum on the right side of the equation (16) are only terms for which $i \geq j$,

$$\tau \sum_{i=0}^{N-1} \frac{\partial F(i)}{\partial u_k(j)} = \tau \frac{\partial F(j)}{\partial u_k(j)} + \tau \sum_{i=j+1}^{N-1} \frac{\partial F(i)}{\partial u_k(j)}. \tag{17}$$

The terms on the right side of the previous equation depend on $u_k(j)$ implicitly through $x(i)$ for $i > j$, and it follows that

$$\frac{\partial F(i)}{\partial u_k(j)} = \tau \sum_{r=1}^{n} \frac{\partial F(i)}{\partial x_r(i)} \frac{\partial x_r(i)}{\partial u_k(j)}. \tag{18}$$

On the basis of equation (12), the following partial derivatives are obtained

$$\frac{\partial F(i)}{\partial x_r(i)} = \frac{\partial \hat{F}(i)}{\partial x_r(i)} + 2K_V \sum_{k=1}^{N_p} g_k(i) \frac{\partial g_k(i)}{\partial x_r(i)} H^- \left(g_k(i)\right) + 2K_E \sum_{k=1}^{N_q} h_k(i) \frac{\partial h_k(i)}{\partial x_r(i)}, \tag{19}$$

for $r = 1, 2, \ldots, n$, and

$$\frac{\partial F(j)}{\partial u_k(j)} = \frac{\partial \hat{F}(j)}{\partial u_k(j)} + 2K_V \sum_{p=1}^{N_p} g_p(j) \frac{\partial g_p(j)}{\partial u_k(j)} H^- \left(g_p(j)\right) + 2K_E \sum_{p=1}^{N_q} h_p(j) \frac{\partial h_p(j)}{\partial u_r(j)}, \tag{20}$$

for $k = 1, 2, \ldots, m$, where $g_p(i) \equiv g_p\left(x(i), u(i)\right)$, $h_p(i) \equiv h_p\left(x(i), u(i)\right)$.

The next step is the calculation of partial derivative $\partial x_r(i)/\partial u_k(j)$ on the right side of the equation (18). On the basis of equation (14), the chain rule for ordered derivatives is obtained

$$\frac{\partial x_r(j+1)}{\partial u_k(j)} = \frac{\partial f_r(j)}{\partial u_k(j)}, \tag{21}$$

$$\frac{\partial x_r(i)}{\partial u_k(j)} = \sum_{p=1}^{n} \frac{\partial f_r(i-1)}{\partial x_p(i-1)} \frac{\partial x_p(i-1)}{\partial u_k(j)}, \tag{22}$$

for $r = 1, 2, ..., n,\ \ k = 1, 2, ..., m,\ \ j = 0, 1, ..., N-1,\ \ i = j+2, ..., N-1,$ where $f_r(j) \equiv f_r(x(j), u(j))$.

If the second term on the right side of the expression (17) is denoted as

$$S_k(j) = \sum_{i=j+1}^{N-1} \frac{\partial F(i)}{\partial u_k(j)}, \tag{23}$$

then following recurrent algorithm for the calculation of the sum (23) can be obtained

$$\begin{aligned} &P_r(N-1) = 0\ , \\ &P_r(j) = \frac{\partial F(j+1)}{\partial x_r(j+1)} + \sum_{l=1}^{n} P_l(j+1) \frac{\partial f_l(j+1)}{\partial x_r(j+1)}\ , \\ &S_k(j) = \sum_{r=1}^{n} \frac{\partial f_r(j)}{\partial u_k(j)} P_r(j)\ , \end{aligned} \tag{24}$$

where $j = N-2, N-3, ..., 0;\ r = 1, 2, ..., n;\ k = 1, 2, ..., m.$

In the matrix representation

$$X(j) = \left[\frac{\partial f_l(j)}{\partial x_r(j)}\right]_{n\times n},\quad U(j) = \left[\frac{\partial f_r(j)}{\partial u_k(j)}\right]_{n\times m},\quad F_x(j) = \left[\frac{\partial F(j)}{\partial x_k(j)}\right]_{n\times 1}\ ,$$

equations (24) can be expressed in the form

$$\begin{aligned} &P(N-1) = 0, \\ &P(j) = F_x(j+1) + X^T(j+1)\cdot P(j+1)\ , \\ &S(j) = U^T(j)\cdot P(j)\ , \end{aligned} \tag{25}$$

where *P(j)* is an *n*-dimensional vector in *j*-th time interval, *S(j)* is an *m*-dimensional vector in *j*-th time interval and *0* is an *n*-dimensional zero vector.

Final step is the calculation of the second term on the right side of the equation (16), i.e. the calculation of the penalty function gradient for the final condition of the state vector

$$\frac{\partial J_1}{\partial u_k(j)} = \sum_{p=1}^{n} \frac{\partial J_1}{\partial x_p(N)} \frac{\partial x_p(N)}{\partial u_k(j)}\ . \tag{26}$$

In the matrix representation

$$Y(j) = \left[\frac{\partial x_p(N)}{\partial u_k(j)}\right]_{n\times m},$$

following recurrent algorithm for the calculation of the $Y(j)$ can be obtained

$$\begin{aligned} &D(N-1) = I\ , \\ &D(j) = D(j+1)\cdot X(j+)\ , \\ &Y(j) = D(j)\cdot U(j)\ , \end{aligned} \tag{27}$$

for $j = N-2, N-3, ..., 1, 0,$ where *D(j)* is the $n \times n$ matrix of *j*-th time interval, whereas

$I$ is the unit $n \times n$ matrix.

*2.4 Final algorithm*

Using the following matrix representation,

$$F_u(j) = \left[\frac{\partial F(j)}{\partial u_k(j)}\right]_{m\times 1},\ V_u(j) = \left[\frac{\partial g_r(j)}{\partial u_k(j)}\right]_{m\times Np},\ V_x(j) = \left[\frac{\partial g_r(j)}{\partial x_k(j)}\right]_{n\times Np},\ E_u(j) = \left[\frac{\partial h_r(j)}{\partial u_k(j)}\right]_{m\times Nq},$$

$$E_x(j) = \left[\frac{\partial h_r(j)}{\partial x_k(j)}\right]_{n\times Nq},\ G_x(N) = \left[\frac{\partial J_1}{\partial x_k(N)}\right]_{nx1},\ \hat{g}(j) = \left[g_k(j) H^-(g_k(j))\right]_{N_p x1}.$$

we can express the complete algorithm for the optimal control problem (1) to (5), by the following four steps:

1. *Initialization of gradient algorithm*. Starting with $l$ = 0 we put in the control vectors $\left(u_0^{(0)}, u_1^{(0)}, \ldots, u_{N-1}^{(0)}\right)$ arbitrary values, which can be outside of the allowed area defined by constraints.

2. *Calculation of state vectors*

$$x^{(l)}(i+1) f\left(x^{(l)}(i), u^{(l)}(i)\right), \quad x^{(l)}(0) = x_0, \tag{28}$$

for $i$ = 0, 1, …, N-1 in $l$-th iteration of the gradient algorithm.

3. *Calculation of gradient*

$$J_u(j) \equiv \frac{\partial J}{\partial u^{(l)}(j)}, \tag{29}$$

for $l$-th iteration of the gradient algorithm.

3.1 *Initialization for j=N-1.*

$$\begin{aligned} &J_u(j) = \tau\left[\hat{F}_u(j) + 2K_V V_u^T(j)\cdot \hat{g}(j) + 2K_E E_u^T(j)\cdot h(j)\right] + G_x(N) U(j), \\ &P(j) = 0, \\ &D(j) = I. \end{aligned} \tag{30}$$

3.2 *Iteration for j* = N-2, N-3,…, 0.

$$\begin{aligned} &F_u(j) = \hat{F}_u(j) + 2K_V V_u^T(j)\cdot \hat{g}(j) + 2K_E E_u^T(j)\cdot h(j), \\ &F_x(j+1) = \hat{F}_x(j+1) + 2K_V V_x^T(j+1)\cdot \hat{g}(j+1) + 2K_E E_x^T(j+1)\cdot h(j+1), \\ &P(j) = F_x(j+1) + X^T(j+1)\cdot P(j+1), \\ &D(j) = D(j+1)\cdot X(j+1), \\ &J_u(j) = \tau\left[F_u(j) + U^T(j)\cdot P(j)\right] + G_x(N)\cdot D(j)\cdot U(j). \end{aligned} \tag{31}$$

4. *Calculation of a new iteration of control vectors on the basis of the gradient algorithm*

$$u^{(l+1)}(j) = u^{(l)}(j) - \eta J_u(j), \tag{32}$$

for $l$ = 0, 1,…, M. We shift the index by one, $l \to l+1$ and go back to step two.

In Figure 1 we can see the basic structure of the numerical algorithm for the optimal control problem, which is a backward in time iterative algorithm, similar like backpropagation-through-time (BPTT) algorithm (Werbos, 1990). The BPTT algorithm is time generalization of the neural network backpropagation algorithm, in the case when the error, which is minimized, is given along the specified time interval. This is the main reason that the BPTT algorithm is mostly used as a learning algorithm for recurrent neural networks (Pearlmutter, 1995; Baldi, 1995**)**.

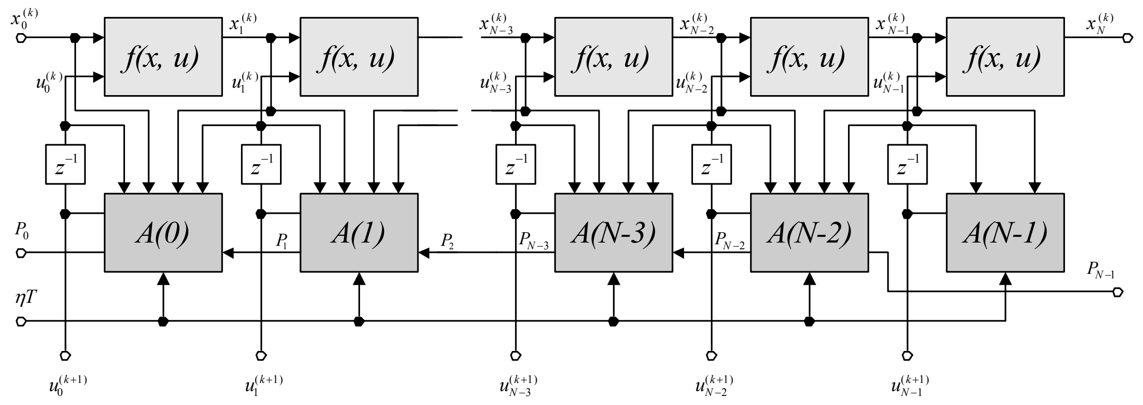


Fig. 1. The structure of the BPTT algorithm for optimal control problem.

## 3. Time optimal control

This section presents the problem of time optimal control (TOC), i.e. control in a situation which requires the minimization of the time interval in accordance with the given constraints and cost function. There are two approaches to the numerical solution of that problem. In the case of time discretization, the control time interval equals the product of the number of sampling intervals $N$ and sampling interval $\tau$, so that the variability of the time interval can be expressed with one of those variables taking the other one as the constant. In the application of the neural network BPTT algorithm in time-optimal control (Plumer, 1996), this problem has been solved by taking sampling interval $\tau$ as the constant and varying the number of sampling intervals $N$. There is a problem with this approach because we have the integer value of variable $N$ in the application of the gradient method. This problem can be avoided by taking discretization sampling as a continuous variable. However, here the problem of non-stability arises, and it can be solved only by taking a very small value of the convergence coefficient in the gradient algorithm for variable $\tau$, which drastically slows down algorithm convergence.

This section describes a heuristic approach to solving the problem of time optimal control, which is relatively effective in avoiding the above-mentioned problems. The method is based on choosing sampling interval $\tau$ to minimize the cost function. The method also uses the characteristics of penalty functions for boundary conditions and constraints. Minimum time, which is the solution of the TOC problem, can be marked by $t_{min} = N\tau_{min}$. The basic idea is to keep the previously obtained algorithm for calculating control vectors for the given constant sampling interval, so that along with calculating control vectors in every iteration of the gradient algorithm, the new value of sampling interval is being calculated. To emphasize the variability of sampling interval $\tau$ we will hereinafter use the symbol $\tau \rightarrow \tau^l$, which represents sampling interval in $l$-th iteration of the gradient algorithm.

We define the new cost function as the sum of penalty functions depending on the variable sampling interval in $l$-th iteration of the gradient algorithm

$$J_T\left(\tau^l\right) = J_1 + J_2 + J_3 , \qquad (33)$$

with the difference that the coefficients of penalty functions $\hat{K}_V$ and $\hat{K}_E$ are defined as constant and independent of sampling interval $\tau$, so that the explicit dependence of function $J_T(\tau^l)$ on $\tau$ has been avoided. Determining minimum time for the transition from the initial to the final state is meaningful only if constraints on control vectors are given. For a given terminal time less than minimum, $t_f < t_{min}$, the inevitable result would be violation of control vector constraints, which would lead to an increase in the value of penalty functions. On the other hand, for a time greater than minimum, $t_f \geq t_{min}$, the control vector satisfies constraints and penalty functions converge to zero.

In the numerical calculation this means that for $t_f \geq t_{min}$ the value of the sum of penalty functions will converge to zero, whereas for $t_f < t_{min}$, that value will converge to a positive number, as it is shown in Figure 2.

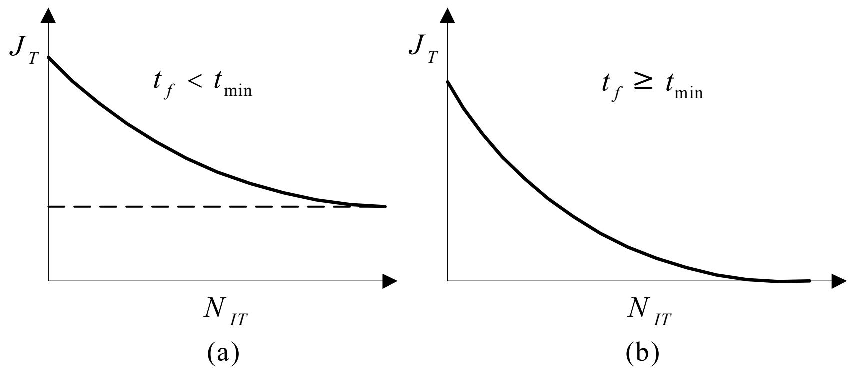


Fig. 2. Dependence of $J_T(\tau)$ on the number of iterations for (a) $t_f < t_{min}$, and (b) $t_f \geq t_{min}$.

On the basis of previous conclusions, we can define the minimum value of function $J_T(\tau)$ by defining it as $J_\varepsilon$, which is the measure of accuracy of the solution of the TOC problem for the given $\tau$. The decreasing of the value of the sum of penalty functions, gives the solution closer to the optimum, as it is shown in Figure 3. However, the determination of that value requires a greater number of iterations.

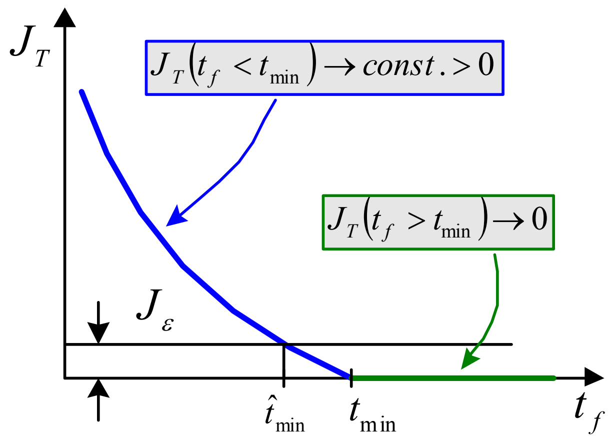


Fig. 3. Dependence of $J_T(\tau)$ on the terminal time.

Thus, the general heuristic algorithm for the solution of the TOC problem is given as follows.
The first step is the initialisation of $\tau^0$, on condition that $\tau^0 > \tau_{min}$. The next steps are: If $J_T(\tau^l) < J_\varepsilon$ Then $\tau^{l+1} < \tau^l$, or If $J_T(\tau^l) \geq J_\varepsilon$ Then $\tau^{l+1} = \tau^l$, for $l$ = 0, 1, …, M. Since we do not know the value of $\tau_{min}$, the initial sampling interval $\tau^0$ can be some sufficiently large value. If one takes $\tau^0 < \tau_{min}$, then $J_T$ will not achieve the value of $J_\varepsilon$ during iteration, which means that an inadequate initial value has been selected.
In other words, the next iteration of sampling interval $\tau^l$ will be ensued only when the value of function $\tau^l$ achieves the given value $J_\varepsilon$, and this means that $\tau^l > \tau_{min}$. As $\tau^l$ decreases, $J_T$ converges to an increasingly slow pace toward $J_\varepsilon$ until it reaches a certain value of $\tau^l$ for which $J_T$ will not be able to reach $J_\varepsilon$ or will converge toward it very slowly, which means that $\tau^l < \tau_{min}$. In that case, one can take $\tau_{min} \approx \tau^{l-1}$ as an approximation for $\tau_{min}$.
This algorithm structure guarantees stability and convergence toward $\tau_{min}$, because it does not change the value of $\tau^l$ until the value of function $J_T$ falls below the given, sufficiently low value of $J_\varepsilon$. The second step in this algorithm can be expressed with the following equation:

$$\tau^{l+1} = \tau^l - \Delta\tau H^-\left(J_T(\tau^l) - J_\varepsilon\right), \tag{34}$$

i.e. each time the condition $J_T(\tau^l) < J_\varepsilon$ is met, $\tau^l$ decreases by the constant value $\Delta\tau$.
This form of algorithm for the TOC problem enables a simple generalization of the algorithm for the optimal control problem (with the fixed control time interval) through the expansion of the fourth step of the algorithm by the equation (34).

## 4. Minimum-time robot control with obstacle avoidance constraints

Optimal control of a non-linear robot model with a defined cast function is still a relatively difficult task. The problem becomes more complex when two or more robots work in cooperation on a common task sharing workspace, time, constraints, and the cost function. In this section, the previously derived time optimal control algorithm will be applied to minimum-time control of a robot with two degrees of freedom.

### *4.1 Dynamics of the robot with two degrees of freedom*

The non-linear dynamic model of the robot with two degrees of freedom is presented through cylindrical coordinates in the form (Heiman, 1981)

$$\begin{bmatrix} M_{11}(q) & 0 \\ 0 & M_{22}(q) \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} N_1(q,\dot{q}) \\ N_2(q,\dot{q}) \end{bmatrix} = \begin{bmatrix} P_1(t) \\ P_2(t) \end{bmatrix}, \tag{35}$$

and

$$\begin{aligned} M_{11} &= I_1 + I_2 + (m+M)q_2^2 + 2Maq_2 + Ma^2, \qquad M_{22} = m + M, \\ N_1 &= 2[(m+M)q_2 + Ma]\dot{q}_1\dot{q}_2, \qquad N_2 = -[mq_2 + M(a+q_2)]\dot{q}_1^2, \end{aligned} \tag{36}$$

where $q = [q_1\ q_2]^T$ are cylindrical coordinates of the centre of the mass of link 3, $M$ is

a total mass (manipulator hand and load), $m$ is link mass, $I_1$ is the total moment of inertial of link 1 and 2 in relation to axis $Z$, $I_2$ is the moment of inertial of link 3 in relation to the axis which is parallel to axis $Y$ and goes through point S, and $a$ is the distance between the centre of mass $M$ and point S (Figure 4.). The variable $P_1(t)$ stands for the control torque of the rotation $q_1$, while $P_2(t)$ is the control force of the translation $q_2$.
Numerical values of the above-motioned parameters are:

$$M = 50kg,\quad m = 97kg,\quad I_1 + I_2 = 193kgm^2,\quad a = 1.1m,\quad P_{1_{max}} = 600Nm,\quad P_{2_{max}} = 500N,$$

where $P_{1_{max}}$ is the maximum allowed moment and $P_{2_{max}}$ is the maximum allowed force. The above-mentioned system of the second-order differential equations can be transformed into a system of the first-order differential equations, by introducing the following coordinate transformation:

$$x_1 = q_1,\quad x_2 = \dot{q}_1,\quad x_3 = q_2,\quad x_4 = \dot{q}_2,\quad u_1 = P_1,\quad u_2 = P_2\ .$$

Also, for the sake of a more elegant expression, the following constants are employed:

$$A_1 = I_1 + I_2 + Ma^2,\quad A_2 = 2Ma,\quad A_3 = m + M\ .$$

As the result we obtain a robot model in the form:

$$\begin{aligned} \dot{x}_1 &= x_2\ , \\ \dot{x}_2 &= \frac{-2A_3x_2x_3x_4 - A_2x_2x_4}{A_1 + A_2x_3 + A_3x_3^2} + \frac{u_1}{A_1 + A_2x_3 + A_3x_3^2}\ , \\ \dot{x}_3 &= x_4\ , \\ \dot{x}_4 &= x_3x_2^2 - \frac{A_2}{2A_3}x_2^2 + \frac{u_2}{A_3}\ . \end{aligned} \tag{37}$$

Thus, the dynamics of the robot with two degrees of freedom has been presented by the system of four non-linear first-order differential equations.

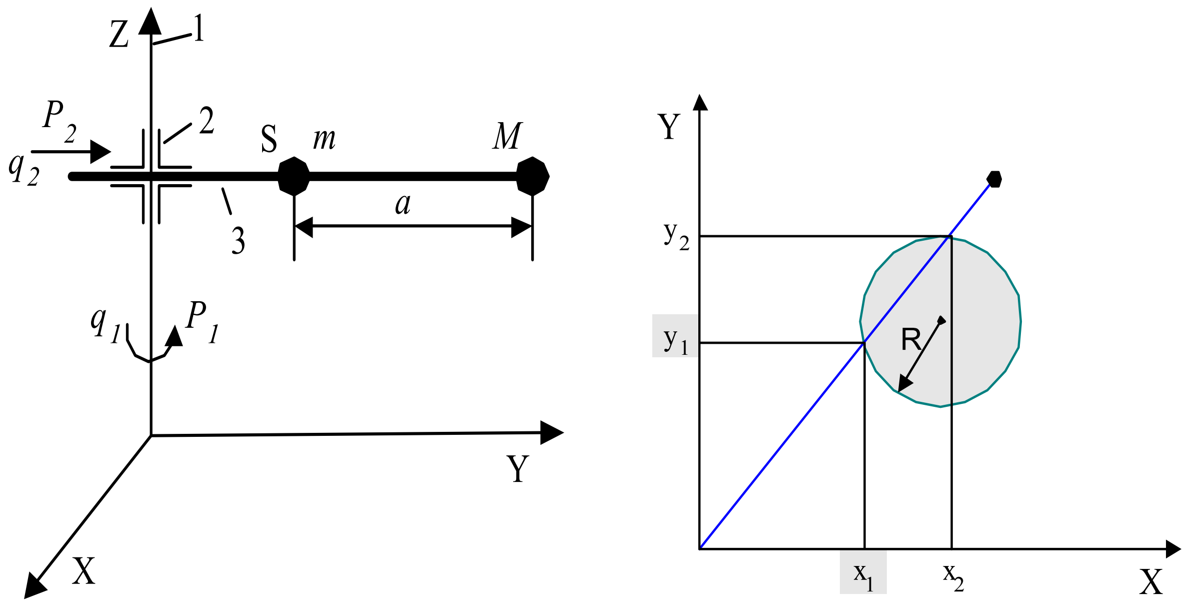


Fig. 4. Robotic manipulator with two degrees of freedom - rotation and translation (left) and intersection of the robot manipulator and the obstacle (right).

*4.2 Obstacle avoidance problem*

One of the tasks that are given to the robot is bypassing obstacles. It is assumed that the position and form of the obstacle are known. For the sake of simplicity, an obstacle shape as a circle with radius $R'$ is taken, with the centre in point $(x_0, y_0)$. The minimum distance between the circle and the robot hand is marked as $\delta R$. This is the minimum allowed distance between the robot hand and the obstacle. The equation of the circle of the radius $R = R' + \delta R$ is

$$(x - x_0)^2 + (y - y_0)^2 = R^2, \tag{38}$$

and the robot hand is shown as the length which lies on the line $y = x \tan q_1$.

From the previous two equations one can obtain conditions of the intersection of the obstacle circle and the robot line as well as the coordinates of section points (Figure 4). The coordinates of intersections of the robot line and the obstacle circle are:

$$x_{1,2} = \frac{2(x_0 + y_0 \tan q_1) \mp \sqrt{D(q_1)}}{2(1 + \tan^2 q_1)} , \qquad y_{1,2} = x_{1,2} \tan q_1 , \tag{39}$$

where $D(q_1)$ must be

$$D(q_1) = 4(x_0 + y_0 \tan q_1)^2 - 4(1 + \tan^2 q_1)(x_0^2 + y_0^2 - R^2) \geq 0. \tag{40}$$

The distances between the intersection points and the origin of the coordinate system are $r_1 = \sqrt{x_1^2 + y_1^2}$, $r_2 = \sqrt{x_2^2 + y_2^2}$. If the fewer of the two distances is marked $r_{min} = min\{r_1, r_2\}$, then it enables the formulation of condition, which guarantees that the robot hand will avoid the obstacle in *i*-th time interval

$$a + q_2(i) \leq r_{min}(q_1(i)), \tag{41}$$

for $i = 0, 1, \ldots, N$, where $a + q_2(i)$ is the length of the robot arm. The penalty function for constraints has the form

$$J_3 = K_p \sum_{i=0}^{N} (r_{min}(q_1(i)) - q_2(i) - a)^2 H^{\pm}(q_1(i), q_2(i)), \tag{42}$$

where

$$H^{\pm}(q_1(i), q_2(i)) = H^{-}(r_{min}(q_1(i)) - q_2(i) - a) H^{+}(D(q_1(i))) \tag{43}$$

and $H^{+}(x) = H^{-}(-x)$. Partial derivatives of the penalty function according to the state vector are given by the equations

$$\begin{aligned} \frac{\partial J_3}{\partial q_1(i)} &= 2K_p \frac{\partial r_{min}}{\partial q_1(i)} (r_{min}(q_1(i)) - q_2(i) - a) H^{\pm}(q_1(i), q_2(i)) , \\ \frac{\partial J_3}{\partial q_2(i)} &= -2K_p (r_{min}(q_1(i)) - q_2(i) - a) H^{\pm}(q_1(i), q_2(i)) . \end{aligned} \tag{44}$$

If it is assumed that the circle is in the first quadrant, then pair $(x_0, y_0)$ and $\tan q_1$ are positive and it is obvious that $r_{min} = r_1$. If parameter $r_1$ is expressed as a function of $q_1$, a very complicated relation and an even more complex partial derivative has been obtained

$$\frac{\partial r_{min}}{\partial q_1(i)} = \frac{\partial r_2}{\partial q_1(i)} . \tag{45}$$

In order to avoid complications with the calculation of this partial derivative, only the change of the radius, as a consequence of violation of the constraint, has been taken into account, i.e. the following equations have been employed

$$\frac{\partial J_3}{\partial q_1(i)} = 0 \, , \qquad \frac{\partial J_3}{\partial q_2(i)} = K_p H^{\pm}(q_1(i), q_2(i)) \, . \tag{46}$$

As it will be shown in the following text, the above-mentioned approximations do not affect the quality of the solution, except for possibly decreasing the speed of convergence toward the optimal solution. The previous expressions mean that in case of the intersection of the manipulator hand and the obstacle, the angle $q_1$ is not changed, whereas the length of the manipulator, i.e. $q_2$, is decreased with the constant rate represented by the positive coefficient $K_p$. This significantly simplifies the problem and reduces it to the geometrical determination of the line section where the hand of the manipulator and the edge of the obstacle are positioned.

However, another problem crops up following the avoidance of complications with the exact calculation of partial derivative. The method of the conjugated gradient does not give good results in this case because the above-mentioned partial derivatives do not belong to the penalty function (42). Therefore, the gradient algorithm with a constant learning coefficient is used.

The problem that is considered in the following text is the transformation of the initial robot state

$$x_1(0) = \pi/12[rad], \quad x_2(0) = 0[rad/s] \, , \quad x_3(0) = 1[m], \quad x_4(0) = 0[m/s] \, ,$$

into the final state

$$x_1(t_f) = 5\pi/12[rad], \quad x_2(t_f) = 0[rad/s], \quad x_1(t_f) = 1[m], \quad x_4(t_f) = 0[m/s],$$

with the condition of avoiding the obstacle in the form of a circle with radius $R' = 0.2[m]$. The coordinates of the circle centre are given by $x_0 = 1[m], y_0 = 1[m]$, and the minimum distance is $\delta R = 0.01[m]$, for minimum time $t_{min} \equiv t_f$, and with the control constraints

$$|u_1(t)| \le u_{1\max} \, , \qquad |u_2(t)| \le u_{2\max} \, , \tag{47}$$

where

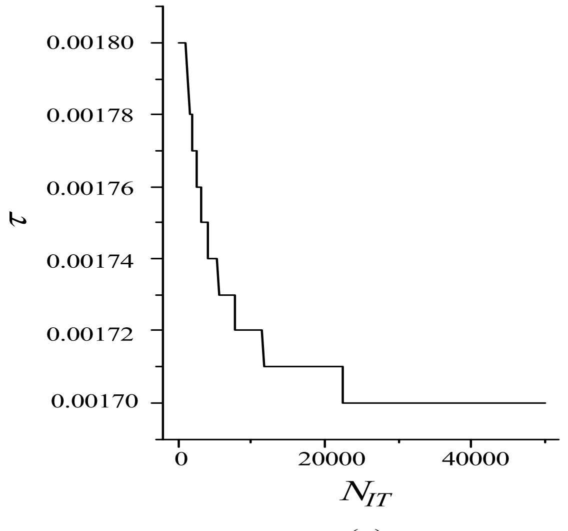
$$u_{1max} = 600[Nm], \; u_{2max} = 500[N] \, .$$

The cost function equals $J = J_1 + J_2 + J_3$, where function $J_T$ has the same form as function $J$, with the difference that instead of the variable value $\tau$ the constant $\tau_0$ is taken as the initial value of the iteration process for calculating $\tau_{min}$, in order to avoid the explicit dependence of $J_T$ on $\tau$. The gradient algorithm with the constant coefficient of convergence $\eta$ is used. The values of the constants are given as follows:

$$N = 1000, \quad M = 30000, \quad K_B = 600, \quad K_V = 0.01, \quad K_p = 1000, \quad \eta = 1000 \, .$$

As it can be seen in Figure 5., with $\Delta\tau = 0.00001[s]$ and the accuracy $J_\varepsilon = 0.01$, it is obtained the minimum time $t_{min} = 1.71[s]$. Using the neural network BPTT algorithm with the parameter $M = 30000$, the simulation results of the minimum time dynamics of the robot are shown in Figures 6 with $t_{min} = 1.71[s]$.

Figure 7. shows the trajectory of the robot hand $(q_2 + a)$ in the plane x-y.



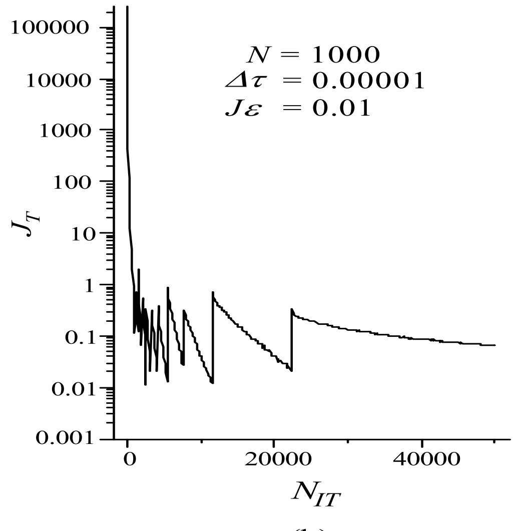
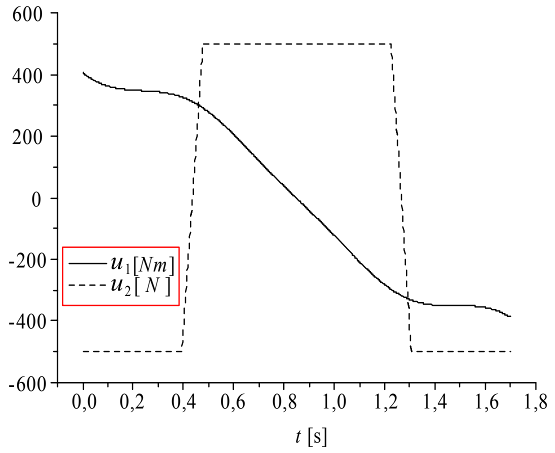


Fig. 5. Time dependence of control variables (left) and state variables (right).



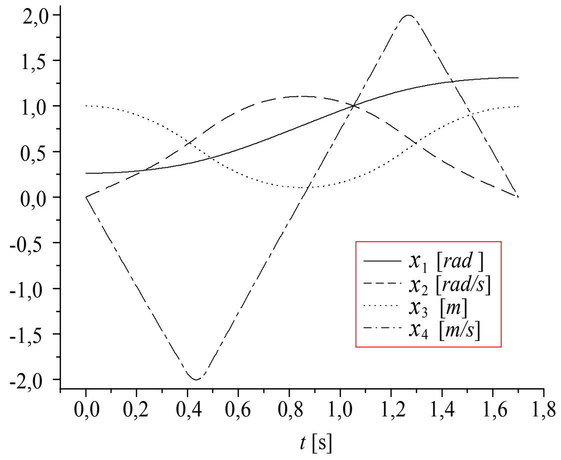


Fig. 6. Time dependence of control variables (left) and state variables (right).

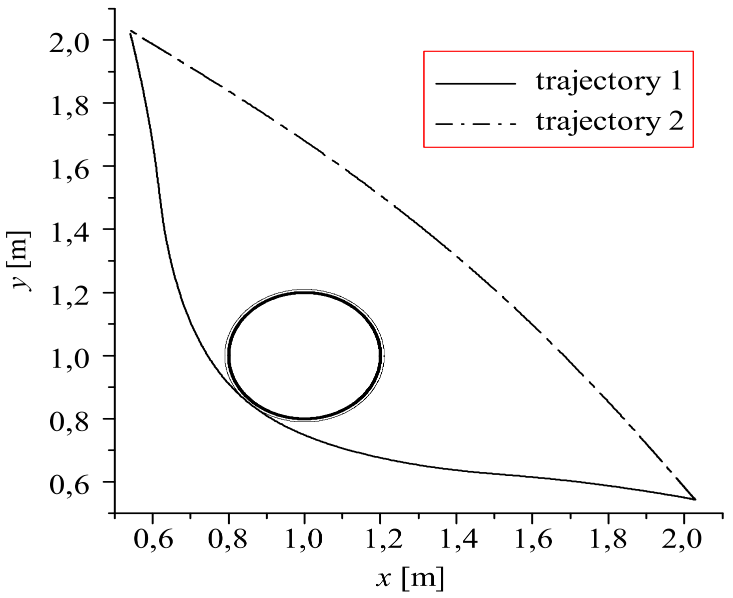


Fig. 7. Time dependence of control variables (left) and state variables (right).

The obstacle is a circle with the radius $R' = 0.2[m]$ (obstacle 1), i.e. a circle with the radius $R = 0.21[m]$ (obstacle 2) including the given minimum distance $\delta R = 0.01[m]$. It can be seen that the optimal trajectory (trajectory 1) touches the circle with the radius *R*, i.e. avoids the obstacle by reaching minimum distance in one point. The figure also shows the trajectory for the same time $t_{min} = 1.71[s]$, in the case when conditions for avoiding the obstacle do not exist (trajectory 2).

## 5. Conclusion

This work presents a new gradient-based approach to solution of the time optimal control problem, which is especially suitable for treating complicate state vector constraints. Formal similarity with neural networks learning algorithms provides high parallelism of computational tasks and solution of high dimensional optimal control problems. The future research will be focused on the problem of worst-case analysis for the problem of robot parameter uncertainty. Also, different heuristic modification of the gradient algorithm will be used for improvement of the algorithm convergence.

## 6. References

Baldi, P. (1995). Gradient descent learning algorithm overview: A general dynamical systems perspective, *IEEE Trans. On Neural Networks*, vol. 6, no. 1, pp. 182-195, ISSN 1045-9227.

Bryson, A.E. Jr. & Y. Ho, (1969). *Applied Optimal Control*, Blaisdell, ISBN 0470114819, New York.

Heiman, B., Loose, H., and Schuster, G. (1981). Contribution to optimal control of an industrial robot, *in Proc of 4th CISM-IFTOMM Symp. On Theory and Pract. Of Rob. And Manip.,* Zaborow, Poland, pp. 211-219.

Kasac, J. & Novakovic, B. (2001a). Neural Network Application to Optimal Control of Nonlinear Systems, *Proceedings of 7-th International Conference on Computer Aided Optimum Design of Structures*, pp. 359-368, ISBN 1-85312-868-6, May 28 - 30, 2001, Bologna, Italy.

Kasac, J. & Novakovic, B. (2001b). Optimal Robot Control with Unspecified Initial and Final Conditions, *Proceedings of 9-th Mediterranean Conference on Control and Automation*, ISBN 953-6037-35-1, June 27-29, 2001, Dubrovnik, Croatia.

Pearlmutter, B.A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey, *IEEE Trans. on Neural Networks*, vol. 6, no. 5, pp. 1212-1228, ISSN 1045-9227.

Plumer, E.S. (1996). Optimal control of terminal processes using neural networks, *IEEE Trans. on Neural Networks*,7, no. 2, 408, ISSN 1045-9227.

Sargent, R.W.H. (2000). Optimal Control, *Journal of Computational and Applied Mathematics*, vol. 124, December 2000, pp. 361-371, ISSN 0377-0427.

Werbos, P.J. (1990). Backpropagation through time: What it does how to do it, *Proceedings of the IEEE,* vol. 78, No. 10, October 1990, pp. 1550-1560, ISSN 0018-9219.