PSO-BASED NONLINEAR PREDICTIVE CONTROL FOR UNMANNED BICYCLE ROBOT STABILIZATION

Keywords: nonlinear predictive control, particle swarm optimization, unmanned bicycle robot

1. INTRODUCTION

Model Predictive Control (MPC) is often used in control systems, as it is intuitive and allows to take into account constraints of signals from a system. Methods for linear model predictive control are well known. This approach cannot be used directly to nonlinear problems if the optimum of nonlinear cost function, relying on exact, nonlinear system is needed. Cost functions, to be minimized at every step, are therefore nonlinear, nonquadratic and, in general, nonconvex. The approaches for exact NMPC utilize different methods for calculating the minimum. Direct methods usually rely on Sequential Quadratic Programming (SQP) [2], [3]. It is also possible, but only for a class of nonlinear models, to use Feedback Linearization (FBL), that provides exact linear model, and then use MPC algorithms [7]. However, change of variables in FBL may provide problems of nonlinear constraints.

In this paper, an exact NMPC is used and Particle Swarm Optimization is proposed to calculate minimization of a cost function. The optimization algorithm was firstly described in [5] and developed afterwards [6], [4]. It uses particles in the space of possible input variables, that changes their position by the use of optimal position met for the whole population and also optimal position met by every single particle.

This PSO-based approach to NMPC was simulated using a model of unmanned bicycle robot, the model of the real object [7]: stabilization by inertial drive is considered. Simulations for different control horizons and prediction horizons were analysed. Additionally, change in a number of iterations and number of particles were considered. Simulations allowed to choose appropriate parameters; the results show the behaviour of proposed approach to nonlinear predictive control.

2. MODEL OF UNMANNED BICYCLE ROBOT

The aim of control is to stabilize the unmanned bicycle robot, that is described in details in [7]. This robot is intended to be a riding bicycle robot, the part of the robot concerning vertical stabilization is, in fact, the dynamics of inverted pendulum with inertia wheel drive.
Tab. 1. Parameters of the unmanned bicycle robot

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_r$</td>
<td>3.962 kg</td>
<td>weight of the robot</td>
</tr>
<tr>
<td>$I_f$</td>
<td>0.0094 kg/m²</td>
<td>moment of inertia of the reaction wheel</td>
</tr>
<tr>
<td>$I_{mr}$</td>
<td>0.001 kg/m²</td>
<td>moment of inertia of the rotor</td>
</tr>
<tr>
<td>$I_{rg}$</td>
<td>0.0931 kg/m²</td>
<td>moment of inertia of the robot related to the ground</td>
</tr>
<tr>
<td>$h_r$</td>
<td>0.13 m</td>
<td>distance from the ground to the center of mass</td>
</tr>
<tr>
<td>$g$</td>
<td>9.8066 m/s²</td>
<td>gravity constant</td>
</tr>
<tr>
<td>$k_m$</td>
<td>0.421</td>
<td>motor constant</td>
</tr>
<tr>
<td>$b_r$</td>
<td>0.0001</td>
<td>friction coefficient in the robot rotation</td>
</tr>
<tr>
<td>$b_f$</td>
<td>0.0001</td>
<td>friction coefficient of the reaction wheel</td>
</tr>
<tr>
<td>$u_{max}$</td>
<td>2.1 A</td>
<td>constraint of input amperage</td>
</tr>
</tbody>
</table>

Mathematical model can be described by the following equations:

\[
\dot{x}_1 = x_2, \\
\dot{x}_2 = \frac{gh_r m_r \sin(x_1)}{I_{rg}} + \frac{b_r x_2}{I_{rg}} + \frac{b_f x_4}{I_{rg}} + \frac{k_m u}{I_{rg}}, \\
\dot{x}_3 = x_4, \\
\dot{x}_4 = \frac{k_m u}{I_f + I_{mr}} - \frac{b_f x_4}{I_f + I_{mr}},
\]

where: $x_1$ – angle of the robot from the vertical position, $x_2$ – angular velocity of the robot, $x_3$ – angle of the reaction wheel, $x_4$ – angular velocity of the reaction wheel, $u$ – current of the motor; the time variable, $(t)$, is omitted for brevity. Parameters of the model are as follows: $m_r$ – weight of the robot; moments of inertia: $I_f$ – of the reaction wheel, $I_{mr}$ – of the rotor, $I_{rg}$ – of the robot related to the ground; $h_r$ – distance from the ground to the center of mass of the robot, $g$ – gravity constant, $k_m$ – motor constant, friction coefficients: $b_r$ – in the robot rotation, $b_f$ – of the reaction wheel. The current of the motor is the input to the system, the amperage cannot exceed $u_{max}$, i.e.,

\[-u_{max} \leq u \leq u_{max}.

Constant described above are listed in Table 1.

The model (1) - (4) can be expressed in simplified form

\[
\dot{x}_1 = x_2, \\
\dot{x}_2 = \psi_2 \sin(x_1) + \psi_3 x_2 + \psi_1 \psi_5 x_4 + \psi_4 u, \\
\dot{x}_3 = x_4, \\
\dot{x}_4 = \psi_5 x_4 + \psi_4 u,
\]

where: $\psi_1 = \frac{I_f + I_{mr}}{I_{rg}}$, $\psi_2 = \frac{-b_r}{I_{rg}}$, $\psi_3 = \frac{gh_r m_r}{I_{rg}}$, $\psi_4 = \frac{k_m}{I_f + I_{mr}}$, $\psi_5 = \frac{-b_f}{I_f + I_{mr}}$.

As can be seen, nonlinearity lies in the sinusoidal function in the second equation of (5). For model predictive control discrete-time model is needed; it is computed with Euler
method, by substitution $\dot{x} = \frac{x_{k+1} - x_k}{T_s}$, where $T_s$ is a sampling interval. Hence, the discrete-time model has the form:

$$x_{k+1} = f(x_k, u_k) = x_k + T_s f_c(x_k, u_k)$$

(6)

where $f_c(x_k, u_k)$ is a right hand side of equation (5) where continuous-time variables $x, u$ are replaced by samples $x_k, u_k$. In the discrete-time model (6), $k$ indicates the number of steps, where $t = kT_s$.

3. Nonlinear Model Predictive Control

3.1. Prediction and Cost Function

The discrete-time model (6) is used for prediction in NMPC. Using initial state defined as $x_1$, with prediction model $\hat{x}_{k+1} = f(\hat{x}_k, u_k)$ one can calculate successive $\hat{x}_2, \hat{x}_3, \hat{x}_4, ...$ with control variables $u_1, u_2, ...$. The aim of predictive control is to find control variables that minimize cost function.

Cost function was chosen to control the bicycle robot

$$J = \sum_{i=1}^{H} \left( \hat{x}_{i+1}^T Q \hat{x}_{i+1} + ru_i^2 \right).$$

(7)

Therefore, the needed predicted states are from $x_2$ to $x_H$, where $H$ is a prediction horizon. Variables for which the function is to be minimized are from $u_1$ to $u_{H_u}$, where $H_u$ is a control horizon. For $i > H_u$, $u_i$ is equal to $u_{H_u}$, and the choice of horizons fulfills $H_u < H$. In the function (7) entries of the matrix $Q$ and the value $r$ are weights for corresponding signals.

Minimization of the function (7) is performed in succeeding steps. At every step, only the first calculated input is used (i.e., $u_1$) and the procedure of prediction and minimization is repeated for the new initial $x_1$ (measured from the real object or calculated as the result of simulations). Therefore, the index number is incremented by one at every step.

3.2. Particle Swarm Optimization

To minimize (7) at every step we use Particle Swarm Optimization (PSO). The optimization method performs calculations in $n_j$ iterations with the use of $n_i$ particles. At every $j$-th iteration, positions are adjusted for every particle $i$ according to the equations

$$v_{i,j} = w_j v_{i,j-1} + r_1 c_1 (p^*_i - p_{i,j-1}) + c_2 r_2 (p^* - p_{i,j-1})$$

$$p_{i,j} = p_{i,j-1} + v_{i,j}$$

(8)

(9)

where $p_{i,j}$ is the position and $v_{i,j}$ is adjustment of position (described as velocity) for particle $i$ calculated in iteration $j$.

Additional elements are used in the velocity equation:

- $r_1, r_2$ are, generated at every iteration, random variables of uniform distribution between 0 and 1. Those factors enter the element of randomness into the process of seeking optimal solution by particles.
• $c_1$, $c_2$ are constants that, after they are multiplied by $r_1$ and $r_2$, serve as weights for future positions. High $c_1$ reinforces positions that are closer to the best position visited by given (i-th) particle so far, $p^*_i$, high $c_2$ reinforces positions closer to the best position visited by the whole population, $p^*$. In this paper $c_1 = c_2 = 2$ are used.

• $w_j$ is the inertia weight that can be a constant or can change when $j$ moves to $n_j$. As every weight in (8), it should take values between 0 and 1. High inertia weight reinforces velocities in the direction of previous velocity. Small inertia weight reinforces influence of best visited positions. In the paper $w_j$ is linearly decreasing from 0.9 to 0.4. This strategy provides better accuracy than constant $w_j$ [1].

4. Simulation results

Simulations of the algorithm were performed in Matlab environment. The initial state was set as $x_0 = [0.065, 0, 0, 0]^T$. Predictive control was performed with the use of discrete-time model with sampling time $T_s = 0.01s$ and PSO of (7) with weights

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, \quad r = 1.$$  \hfill (10)

Time simulations was performed with the use of continuous-time model (5).

4.1. Simulations with different predictive and control horizons

In the first part of simulations, the number of particles and number of simulations was chosen high enough to obtain solutions of PSO close to the global optimum ($n_i = 100$, $n_j = 100$). Figure 1 and 2 show simulation results for prediction horizon $H = 100$ and with the change of control horizon, $H_u$. Figure 1 shows results for $H_u = 1$ (solid line), $H_u = 2$ (dashed line), $H_u = 3$ (dotted line), in Figure 2 results with $H_u = 2$ and $H_u = 3$ are presented with different scale. One can see that $H_u = 1$ is an insufficient control horizon, there is no guarantee that simulation will stabilize state signals. For $H_u = 2$ and $H_u = 3$ simulations are similar, however $H_u = 3$ provides slightly better performance (it is visible for state variable $x_3$). Further increase in $H_u$ does not have much positive influence. Therefore, for further simulations control horizon $H_u = 3$ is chosen.

Further analysis concern changes in prediction horizon $H$. Figure 3 shows results with too low $H$, for which the control system is unstable, whereas Figure 4 presents results with $H$ high enough for the system to be stable. $H = 56$ is the lowest horizon that assures stable behaviour. For higher $H$ control quality is better; however, from about $H = 76$ the improvement is not significant. For further simulations predictive horizon $H = 80$ was chosen.

4.2. Simulations with different number of particles and iterations

In this subsection, Figures 5 - 8 show the influence of change of particle number, $n_i$, and iterations number $n_j$. All simulations were obtained for $H = 80$, $H_u = 3$ and for every Figure three simulations were performed. Lower numbers of particles and iterations than in...
the previous simulations and random features in PSO make the differences between the three
simulations. Obviously, when $n_i$ and $n_j$ are high, the quality of control system is also high;
Figure 5 shows result obtained for $n_i = 20, n_j = 20$ that provide stable results. Change of
values $n_i$ and/or $n_j$ to 10 are presented in Figures 6, 7, 8; at every of these Figures at least
one (of the three) simulation can be unstable; however, one can see that in this case the better
choice is to use lower number of particles (7) than lower number of iterations (8).

5. CONCLUSIONS

In the article, particle swarm optimization was used as a method for solving optimization
problems in nonlinear model predictive control. The PSO method makes a good alternative
to different methods of optimizations that are likely to stuck in local minima. The approach
of PSO-based NMPC was tested with the model of bicycle robot. Several arrangement of
parameters was used, different prediction and control horizons and numbers of iterations
and particles were analysed. Simulations show, that, with carefully chosen parameters, the
approach provides good control quality.

REFERENCES

strategies in particle swarm optimization. In 2011 Third World Congress on Nature and Biologically

© Authors and Poznańskie Towarzystwo Przyjaciół Nauk 2017, sait.cie.put.poznan.pl
Fig. 2. Time simulations with \( n_i = 100 \), \( n_j = 100 \), \( H = 100 \), and \( H_u = 2 \) (dashed line), \( H_u = 3 \) (dotted line).


© Authors and Poznańskie Towarzystwo Przyjaciół Nauk 2017.
Fig. 3. Time simulations with $n_i = 100$, $n_j = 100$, $H_u = 3$, and $H = 51$ (dotted line), $H = 53$ (dashed line), $H = 55$ (solid line)

Fig. 4. Time simulations with $n_i = 100$, $n_j = 100$, $H_u = 3$, and $H = 56$ (dash-dotted line), $H = 66$ (dotted line), $H = 76$ (dashed line), $H = 86$ (solid line)

© Authors and Poznański Towarzystwo Przyjaciół Nauk 2017, sait.cie.put.poznan.pl
Fig. 5. Time plots with $H = 80$, $H_u = 3$ and $n_i = 20$, $n_j = 20$; three simulations

Fig. 6. Time plots with $H = 80$, $H_u = 3$ and $n_i = 10$, $n_j = 10$; three simulations
Fig. 7. Time plots with $H = 80$, $H_u = 3$ and $n_i = 20, n_j = 10$; three simulations

Fig. 8. Time plots with $H = 80$, $H_u = 3$ and $n_i = 10, n_j = 20$; three simulations
ABSTRACT

The paper considers vertical stabilization of an unmanned bicycle-like robot. Nonlinear predictive control is utilized for the purpose; at every step optimization of a nonlinear cost function using particle swarm optimization is performed. This allows to find optimal global solution or solution close to this optimum, depending on the employed time, even for nonconvex functions. Simulations show that this approach to model predictive control can provide satisfactory results.

Received: 2017-10-20
Accepted: 2017-11-28

© Authors and Poznańskie Towarzystwo Przyjaciół Nauk 2017, sait.cie.put.poznan.pl