A self-tuning fuzzy PD controller for a wheeled mobile robot operating in the presence of faults

Adam Srebro
Warsaw University of Technology, Institute of Control and Industrial Electronics, e-mail: srebroa@ee.pw.edu.pl

This paper presents a self-tuning fuzzy PD controller designed to improve a control strategy for a wheeled mobile robot. The dynamic model of a two-wheeled mobile robot was implemented in Matlab/Simulink environment. In this paper the trajectory tracking problem for a mobile robot in the presence of positioning system faults is considered in detail. Mamdani fuzzy reasoning is used to tune the value of proportional gain $K_p$ and derivative gain $K_d$ of a PD controller. A few simulations comparing the classical PD controller with self-tuning fuzzy PD are reported to show advantage of the designed self-tuning controller.

Keywords and phrases: fuzzy controller, mobile robot, nonholonomic system, trajectory tracking, fault-tolerant.

Introduction

In the past decade we have observed the growing number of both service and industrial mobile robots applications. They are adapted to more and more complex tasks that require the development of their environment perception system. The more complex device a robot becomes, the more likely that there may occur some faults in its modules during operation. In an industrial environment a partial destruction of a robot system can lead to uncontrolled behaviour (e.g. a change of the trajectory of motion) which may endanger the safety of people and devices. In some cases, such as an exploration of an unknown area, it is important that the robot despite the partial failure of its modules completes the main task and returns to the base. To meet this requirement a fault adaptive control system is needed. A typical fault adaptive control system includes a fault detection and isolation subsystem and a control accommodation part [1–2]. The controller can be reconfigured depending on the current degree of failure. Fault-tolerant control systems have been developed especially in aeroplanes [3], spacecraft [4] and chemical plants [5].

The fault-tolerant control is a relatively new area of research in the mobile robotics field linked to other fundamental problems such as: positioning [15], navigation [16], planning [17] and obstacle avoidance [18] that still need much attention. Faults may occur in both actuators and sensors. However, faults may also be the result of failing interaction with the environment. Actuators faults have been investigated in the publication [6].

In this paper the author focuses on partial degradation of positioning sensors system. To improve the robustness of the control system, the adaptive self-tuning fuzzy PD controller has been used. The paper consists of four sections. In section 1 the dynamic model of a mobile robot with constraints is presented. Section 2 includes a look-ahead controller based on input-output feedback linearization technique and self-tuning fuzzy PD controller applied to the linearized model. Section 3 presents a comparison of simulated trajectories for the PD and self-tuning PD controller. Section 4 summarizes the results and shows the directions of future research.

Mobile robot modelling

The mobile robot model shown in Fig. 1 is a two-wheeled differentially driven vehicle. Each wheel is independently driven by a separate dc motor. The analyzed system is non-holonomic, so the constraints equations are included in the kinematic model. For further model analysis let assume that robot is moving on a horizontal plane at a low speed. Additionally, it is assumed that the wheels are non-deformable and have the same diameter. For high load and/or high speed
conditions other models are preferable [7]. Firstly, the constraints equations are presented, then the motion equation based on the Lagrange formulation is shown. Finally, both equations are represented in the state space.

Constraints

In a kinematic model it is assumed that in each contact there exists a pure rolling motion. Assuming that the velocity of Po is in the direction of the axis of symmetry X and that the wheels do not slip, the following constrains are obtained [8]:

\begin{align}
  \dot{y} \cos \phi - \dot{x} \sin \phi - \dot{d} &= 0 \\
  \dot{x} \cos \phi + \dot{y} \sin \phi + b \dot{\theta} - r \dot{\theta} &= 0 \\
  \dot{x} \cos \phi + \dot{y} \sin \phi - b \dot{\theta} - r \dot{\theta} &= 0
\end{align}

According to the first constraint (1), the mobile robot cannot move in the lateral direction (in other words no lateral slip is allowed between the wheels and the floor). Constraints equations (2) and (3) ensure that right and left wheel roll without a longitudinal slip.

Where: \( \theta_r, \theta_l \) — angular displacement of the right and left wheels, respectively,

\( x_c, y_c \) — coordinates of the center of mass in the global reference frame,

\( \phi \) — heading angle of the platform measured from the X axis of the global reference frame,

\( b \) — half the distance between two driving wheels,

\( d \) — the distance between Po and the center of mass Pc of the robot,

\( L \) — the distance between Pc and the Pr, 

\( r \) — radius of the wheel.

The three constraints given by equations (1–3) can be written in a so-called Pfaffian constrains form:

\[ A(q)\dot{q} = 0 \] (4)

where:

\[
A = \begin{bmatrix} 
-\sin \phi & \cos \phi & -d & 0 & 0 \\
-\cos \phi & -\sin \phi & -b & r & 0 \\
-\cos \phi & -\sin \phi & b & 0 & r 
\end{bmatrix}
\]

\[ q = [x_c, y_c, \phi, \theta_r, \theta_l] \] (6)

We assume that generalized coordinates of the mobile robot are given by equation (6).

Dynamic model

Using the Euler-Lagrange formula and assuming that Lagrangian of the system equals the kinetic energy (planar motion is considered), the equation of motion for a mobile robot can be written in the matrix form [8] as following:

\[ M(q)\ddot{q} + C(q, \dot{q}) = E(q)\tau - A'(q)\lambda \] (7)

where:

\( q \in \mathbb{R}^{nq} \) — vector of generalized coordinates defined in equation (6),

\( M(q) \in \mathbb{R}^{nq\times nq} \) — symmetric positive definite inertia matrix of the system,

\( C(q, \dot{q}) \in \mathbb{R}^{nq\times n\dot{q}} \) — centripetal and coriolis forces matrix,

\( E(q) \in \mathbb{R}^{nq\times n\tau} \) — input transformation matrix,

\( \tau \in \mathbb{R}^{n\tau\times nq} \) — input vector,

\( \lambda \in \mathbb{R}^{n\lambda} \) — vector of Lagrange multipliers,

\( A'(q) \in \mathbb{R}^{n\lambda\times nq} \) — matrix associated with the kinematic constrains defined in (5).

Matrix \( A(q) \) is defined in equation (5). The other matrices from equation (7) are given by:

\[
M(q) = \begin{bmatrix}
m_r & 0 & -m_r d \sin \phi & 0 & 0 \\
0 & m_r & m_r d \cos \phi & 0 & 0 \\
-m_r d \sin \phi & m_r d \cos \phi & I_r & 0 & 0 \\
0 & 0 & 0 & I_{\omega} & 0 \\
0 & 0 & 0 & 0 & I_{\omega}
\end{bmatrix}
\]

\[
C(q, \dot{q}) = \begin{bmatrix}
-m_r d\dot{\phi} \cos \phi & 0 & 0 \\
-m_r d\dot{\phi} \sin \phi & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
E(q) = 0 \begin{bmatrix} 0 \\
0 \\
1 \end{bmatrix}
\]

Fig. 1. The mobile robot position in global reference frame.
\[
\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}
\]

where: \( m_r = m_d + 2m_w \)

\[
I_z = I_y + 2m_w (d^2 + b^2) + 2I_w
\]

- \( m_r \) — the mass of the platform without the driving wheels and the rotors of the motors;
- \( m_w \) — the mass of each driving wheel plus the rotor of its motor;
- \( I_z \) — the moment of inertia of the platform without the driving wheels and the rotors of the motors about a vertical axis through \( P_z \);
- \( I_y \) — the moment of inertia of each wheel and the motor rotor about the wheel axis;
- \( I_w \) — the moment of inertia of each wheel and the motor rotor about the wheel diameter.

State space representation

The next step is the state space representation of the motion equation (7) and the constraint equation (4). Firstly, in order to choose a state vector properly we create a matrix \( S(q) \) which satisfies the equation:

\[
A(q)S(q) = 0 \quad (8)
\]

The matrix \( A(q) \) is a matrix of constrains from the equation (5). \( S(q) \) is a matrix given by the equation (9) which two columns are in the null space of \( A(q) \) and are linearly independent.

\[
S(q) = \begin{bmatrix}
  c(b\cos\phi - d\sin\phi) & c(b\cos\phi + d\sin\phi) \\
  c(b\sin\phi + d\cos\phi) & c(b\sin\phi - d\cos\phi) \\
  c & -c \\
  1 & 0 \\
  0 & 1
\end{bmatrix} \quad (9)
\]

The constant \( c = (r/2b) \). By analogy, from the equation (4) we know that the velocity \( \dot{q} \) must be in the null space of \( A(q) \). Since both \( \dot{q} \) and columns of \( S(q) \) are in the null space of \( A(q) \), we can express the velocity \( \dot{q} \) as the linear combination of columns \( S(q) \)

\[
\dot{q} \in \text{span}\{s_1(q), s_2(q)\} \quad (10)
\]

\[
\dot{q} = S(q)\eta \quad (11)
\]

where \( \eta \) is the generalized velocity vector. Differentiating equation (11), we obtain

\[
\ddot{q} = S(q)\ddot{\eta} + \dot{S}(q)\eta \quad (12)
\]

Multiplying both sides of equation (7) by \( S'(q) \) and knowing that \( S'(q) A'(q) = 0 \) and \( S'(q) E(q) \) is the \( 2 \times 2 \) identity matrix, we obtain:

\[
S'(q)M(q)\ddot{q} + S'(q)C(q, \dot{q}) = S'(q)E(q)\tau - S'(q)A'(q)\lambda = \tau \quad (13)
\]

Substituting the equation (12) for \( \ddot{q} \) in equation (13), we get

\[
S'(q)M(q)(S(q)\eta + \dot{S}(q)\eta) + S'(q)C(q, \dot{q}) = \tau \quad (14)
\]

Using the state vector in the form as below

\[
x = \begin{bmatrix} x_o & y_o & \phi & \theta & \dot{\phi} & \dot{\theta} \end{bmatrix}^T = [q \ \eta]^T \quad (15)
\]

we can represent the equation of motion (14) in the state space form

\[
\dot{x} = f(x) + g(x)\tau \quad (16)
\]

Substituting equation (11) for \( \dot{q} \) and calculating \( \ddot{\eta} \) from equation (14) we obtain

\[
\dot{x} = \begin{bmatrix} \dot{q} \\ \eta \end{bmatrix} = \begin{bmatrix} S\eta \\ -(S'MS)^{-1}(S'M\dot{S}\eta + S'C) \end{bmatrix} + \begin{bmatrix} 0 \\ (S'MS)^{-1} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (17)
\]

To simplify the notation we introduce a new variable

\[
k = -(S'MS)^{-1}(S'M\dot{S}\eta + S'C) \quad (18)
\]

Finally, applying the following nonlinear feedback to the system given by equation (17)

\[
\tau = S'MS(u - k) \quad (19)
\]

we obtain simplified state equation as below where \( u \) is a new input vector.

\[
\begin{bmatrix} \dot{q} \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} S\eta \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ (S'MS)^{-1} \end{bmatrix} \tau = \begin{bmatrix} S\eta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = f(x) + g(x)u \quad (20)
\]

Control Algorithm

The control method used in this study is based on linearization technique via feedback [9–12]. The linearization is obtained between the control inputs and appropriate outputs and nonlinear feedback is given by the equation (24). Yamamoto and Yun [13] have proved that a nonholonomic system is not input-state linearizable due to not being involutive. However, it can be input-output linearizable. I have chosen the point \( P_r \) located on the wheel axis as a reference point for trajectory tracking due to the presence of nonholonomic constraints [14].

Output Equation

The output equation (21) is given by coordinates of \( P_r \) which is in some distance \( L \) from \( P_o \). This distance is
often called a look-ahead distance and control a point in front of the mobile robot \( (P_R) \) is called look-ahead control.

\[
y = h(x) = \begin{bmatrix} x_c + L \cos \phi \\ y_c + L \sin \phi \end{bmatrix} = \begin{bmatrix} x_c + x_c \cos \phi - y_c \sin \phi \\ y_c + x_c \sin \phi + y_c \cos \phi \end{bmatrix} = \Phi(x) \eta \tag{21}
\]

In order to find the new linearization law between the control input and output, we differentiate the output (21) with respect to time. Then we check if a new input \( u \) appears. If not, we differentiate it again. The result of the first derivative of \( y \) is given by

\[
\dot{y} = \frac{\partial y}{\partial x} \dot{x} = \frac{\partial}{\partial x} (f(x) + g(x)u) = \frac{\partial}{\partial x} \left[ \begin{bmatrix} S \eta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u \right] = \Phi(x) \eta
\]

(22), where \( b \) and \( r \) are always positive. Concluding, the distance \( L \) must be different from zero to satisfy non-singularity of matrix \( \Phi(x) \). Applying the nonlinear feedback (24) to the system given by equation (20), we obtain linearized and decoupled subsystem.

\[
\dot{\mu} = \dot{y} + K_s(y_s - \dot{y}) + K_r(y_r - y) \tag{26}
\]

A self-tuning fuzzy PD controller

In order to improve the quality of control in case of partially degradation of a robot positioning system, a self-tuning fuzzy PD controller has been designed. The controller output is given by

\[
\mu = \dot{y} + K_s(y_s - \dot{y}) + K_r(y_r - y) \tag{26}
\]

The Mamdani model has been applied as a structure of a fuzzy interface. The designed fuzzy subsystem has one input and two outputs as shown in Fig. 3. The input parameter is error and the outputs are \( K_p \) and \( K_d \). The
A self-tuning fuzzy PD controller for a wheeled mobile robot operating in the presence of faults

system contains three rules based on the mamdani reasoning. The rule table is shown in Table 1. Centroid and max-min methods are used in defuzzification and aggregation respectively. The triangular membership functions of appropriate inputs and outputs are shown in Fig. 4.

![Fig. 3. Simulink block diagram of self-tuning fuzzy PD subsystem.](image)

### Table 1. Rule table of the fuzzy interface

<table>
<thead>
<tr>
<th>e</th>
<th>Kp, Kd</th>
</tr>
</thead>
<tbody>
<tr>
<td>eL</td>
<td>KpH, KdH</td>
</tr>
<tr>
<td>eM</td>
<td>KpM, KdM</td>
</tr>
<tr>
<td>eH</td>
<td>KpL, KdL</td>
</tr>
</tbody>
</table>

The values of proportional gain $K_p$ and derivative gain $K_d$ are tuned according to the rule table based on the changes in the error. The error is defined by the equation (28).

$$e = \sqrt{(x - x_o)^2 + (y - y_o)^2}$$

### Simulations

The simulations using Matlab and Simulink environment have been carried out to verify the performance of the designed adaptive fuzzy PD controller. Additionally, for the same parameters simulations have been performed for PD controller which is widely used in mobile robotics. The parameters of the model used in simulation correspond to parameters of a real mobile robot used in experiments. Both default initial simulations and model parameters are shown in Table 2.

Figure 5 shows the block diagram of a simulated mobile robot system. The system contains several subsystems, with the most important being: S-Function block containing linearized model of robot dynamics, self-tuning fuzzy PD controller block and signal conditioning blocks. Among the conditioning blocks we can distinguish: a block for the parameters $K_p$, $K_d$ and a block for error signal $e$. The reference trajectory is generated from a separate generator. Additionally, fault signals are generated for the first three components of the vector $q$. To solve the first order ordinary differential equation (derivative of the vector $x$) the ode45 solver has been used.

As it has been mentioned in the introduction, the fault adaptive control system includes a fault detection
and isolation subsystem. In presented simulations it has been assumed that positioning system is based on absolute position measurements (e.g. Landmark Navigation, Active Beacons) [15]. Such a system may faultily identify the location of the robot as a result of temporary disturbances or partial failures of the system components. It has also been assumed that the system is able to reconfigure after the delay equal 0.8 s and after reconfiguration allows measurement of the position with an additional constant error equal 0.1 m. The amplitudes of position errors and the time needed for reconfiguration have been chosen only as an example and may change depending on the degree of failure or disturbance.

The essence of these simulations is to show that even reconfigurable systems need some time to detect failure or disturbance and reconfiguration. During this time the positioning system is operated on the erroneous data on the location of the robot. Since the fault is detected and estimated it is desirable that the robot adapted its parameters to the size or type of the fault and current robot state. In the simulations presented below adaptation is based on a robot speed reduction below a reference value when a robot is in close proximity to the desired trajectory. This step can improve the positioning accuracy of a robot.

The Fig. 6 shows a mobile robot trajectory tracking task in the presence of positioning system faults. The parameters of a PD controller have been chosen so that the robot could reach the desired trajectory in a short time. However, the overshoot effect is visible. As shown in Fig. 7 coefficients $K_p$ and $K_d$ could be chosen so that there is no overshoot. On the other hand, in these conditions the robot moves much slower and needs more time to reach the reference trajectory.

Figure 7 shows a comparison of PD and a self-tuning fuzzy PD controller. As we can see in this figure,
the trajectory of a robot over the distance of 0.45 m corresponds to the trajectory in Fig. 6. The robot slows down below this value and its trajectory is gradually becomes the trajectory of PD (a dashed dotted line).

In the trajectory tracking task it is desirable that the robot reaches the desired trajectory as soon as possible. However, the positioning system requires to reduce the speed of a robot to improve measurement accuracy, when a robot is close to the target. In case of a fault in the positioning system, it is also desirable to reduce the speed of a robot especially when it is close to a desired path.

The Fig. 8 shows the mobile robot trajectory in the presence of faults in positioning system. The amplitude of the fault in Fig. 8 is equal to 1.2 m. After the detection, isolation and partial offsetting of the impact of fault, a robot returns to the reference trajectory. The velocity amplitude is tuned by a fuzzy controller based on the equation (27).
A robot with the self-tuning fuzzy PD controller compared to the PD moves faster when it is far from the reference trajectory as shown in Fig. 9a. Another advantage of this adaptive controller as opposed to the PD is that it allows to reduce speed of a robot below a reference value (1 m/s) when a robot is close to the desired trajectory. This property is especially important in the presence of faults. A robot passes reference trajectory in a similar time for both controllers as shown in Fig. 9b. However, a self-tuning controller provides better positioning of the robot in close proximity to the desired trajectory by reducing the speed of a robot.

Figure 10 shows changes in proportional and derivative gain in time for the case shown in Fig. 7. It may be noted that in the initial phase of the simulation, when the velocity is rapidly increasing, the value of $K_p$ is greater than the value of $K_d$. In the second phase, the exceeded distance of 0.45 m from the desired trajectory is followed by a rapid increase in the value of both coefficients. This time, however, the value of $K_d$ grows much faster than the $K_p$, which leads to a rapid deceleration of a robot speed. After some time, the speed of a robot is set at the desired value, and similarly, the difference between the coefficients $K_p$ and $K_d$ is a constant value.

Permitted ranges of variations values $K_p$ and $K_d$ have been chosen based on previous simulations for a PD controller. These ranges should be selected depending on the following restrictions: maximum speed, rise time, overshoot, and a desirable speed, depending on the state and degree of failure in the measuring system.

**Conclusion**

The simulations have been carried out using the Matlab/Simulink environment. The dynamic model proposed by
Yamamoto and Yun [8] has been applied and the possibility of error injection has been added. A faulty operation of positioning system requires controller reconfiguration [19]. One of the methods of reconfiguration and adaptation to faulty conditions may be a non-linear change in control signals. To meet this need a self-tuning controller has been designed by the author.

The simulations have proved that a self-tuning fuzzy PD controller, which has the ability to tune the value $K_p$ and $K_d$ gives better results in controlling the dynamical system then the conventional PD [20]. The proposed self tuning controller has a fast rise time, however less overshoot then PD controller with the same rise time. Additionally, it improves the conditions for positioning a robot by decreasing its speed in the presence of faults.

Faults may affect the dynamics of the system. In order to compensate for the fault influence on the system, a fast and reliable detection and isolation subsystems are needed. Such subsystems have not been discussed in this paper and it is a challenging topic for further research. More research is also needed in the area of faults modelling. In some key processes, hardware redundancy is still recommended and should not be neglected. In such a case, software compensation can be used as an additional protection. It is worth bearing in mind that there exist some faults difficult to identify due to their uniqueness and their random nature.

### References


