Data-driven trajectory tracking of manipulator with event-triggered model updating

Yuanxiang Wang 1*

1School of Automation, Southeast University, Nanjing 210096, China
*Corresponding authors e-mail: wangyx_95@163.com

Abstract. In recent years, robot industry has developed rapidly, which has been widely used in industrial production and social life. The control problem of manipulator system with strong nonlinearity has become a hot spot. The difficulty of the trajectory tracking control of the manipulator is that the manipulator system is generally complex, and its dynamic model is very complex, including multiple mutually coupled inputs and outputs, sensor noise, friction in the manipulator and the flexibility of the joint link, which makes the manipulator system highly nonlinear. At the same time, due to some errors in the measurement of some inherent parameters of the manipulator, as well as different interference in the external environment, it is difficult to establish the accurate mathematical model of the manipulator system. In this paper, the Gauss process feedback linearization method based on the updating of the event triggered model is applied to a manipulator system with three degrees of freedom to realize the trajectory tracking control of the manipulator. And for the real-time requirement of trajectory tracking of manipulator, sparse Gaussian process regression is used to solve the problem of large amount of calculation of Gaussian process regression under large data samples.

1. Introduction

With the development of the times and the progress of science and technology, the application of robots in industry and public life is more and more extensive and important. The life of modern society has been inseparable from the help of robots. Since general motors first applied robots to the production line in the 1960s, robot technology has made great progress under the premise of social production development after decades of development, played a huge role in military, medical, aerospace, and the demand for robots in various fields is also increasing. In industrial production, it gradually replaces human beings to complete some repetitive and relatively heavy work, such as welding, handling, assembly, and the automation level of production has made great progress [1]. In military applications, robots help to complete the dangerous work such as detonation and reconnaissance [2]. And in recent years, robots have gradually entered into people's daily life, home service and catering service robots are also increasingly emerging [3][4].

Generally, the manipulator has multiple degrees of freedom, which enables it to complete tasks flexibly in different application scenarios, but also increases the difficulty of its control [5]. Generally, the control methods for manipulator include position control and track tracking control [6]. Position control, that is, the movement of the end of the manipulator between two designated positions, is characterized by not paying attention to the specific trajectory of the end of the manipulator, only reaching the designated target point. The trajectory tracking control is totally different. It not only needs the end of the manipulator to reach the target point accurately, but also requires the motion trajectory of its joint and the speed corresponding to the time. According to the trajectory obtained by path planning...
and the real-time angle, speed and position obtained by sensors, a suitable controller is designed to track the trajectory by controlling the joint torque of the manipulator. Obviously, trajectory tracking control is more difficult. The robot arm is engaged in all kinds of complex tasks, the premise is to be able to carry out high-speed and accurate trajectory tracking control, so the research of trajectory tracking control is very necessary, and it is also a research hotspot in the control field.

In this paper, the Gauss process feedback linearization method based on the updating of the event triggered model is applied to a manipulator system with three degrees of freedom to realize the trajectory tracking control of the manipulator.

2. Related work
Trajectory tracking control of the manipulator, that is, to design a suitable controller, the input of the controller is the desired trajectory, and the output is the driving torque of different joints, so that the actual trajectory of the manipulator is consistent with the desired trajectory [7]. The trajectory tracking control of manipulator is a necessary condition to precisely control the work and production of manipulator, so many scholars are attracted to study it. The traditional methods of trajectory tracking control of manipulator mainly include PID control, variable structure control, adaptive control, iterative learning control, data-driven control, robust control, etc. [8].

Kumar A et al. Proposed a fractional order pre compensation PID controller to solve the overshoot problem of the output response in case of external interference. [10] proposed a PID method based on neural network, which uses particle swarm optimization algorithm to update the weight of the network and reduce the steady-state error of the system. However, due to the strong nonlinearity and coupling of the manipulator system, the PID control method may lead to a considerable output torque of the controller[11]. However, in the actual application process, the control torque provided by the drive motor must be limited, and the characteristics of the mechanical arm itself are not allowed to sustain a particularly large control torque, which will cause great damage to the drive motor[12].

Yagiz et al. [13] applied the sliding mode variable structure control to the more complex manipulator system to achieve high-precision trajectory tracking at the end of the manipulator. The advantage of sliding mode control law is that it is not very sensitive to external disturbances and sudden changes of parameters [14]. This advantage makes it very suitable for the manipulator system with strong nonlinear characteristics. However, the disadvantage of sliding mode variable structure control is that when it moves from the state of Chu state to the sliding mode surface, the sliding mode variable structure control may cause chattering of the system, which will cause great damage to the driving motor of the manipulator. Kim P and lechewan n. [15] proposed a robust control method based on disturbance observer to realize the compensation of external disturbance. In reference [16], an adaptive robust control strategy based on fuzzy algorithm is designed to solve the problem of system error and interference, and the effectiveness of tracking for manipulator is verified by experiments.

Many of the above control methods need to obtain the mathematical model of the manipulator system. When the system model can be accurately obtained, these control methods can accurately control the manipulator system[17][18]. However, in the practical application, the manipulator system is often disturbed by the external environment, and there are many uncertainties in the system itself, so the accurate dynamic model is generally difficult to obtain. The data-driven control method can solve these problems well. At present, the combination of data-driven model and model predictive control [19][20][21] shows great potential in solving this challenge, which has been widely used in recent years.

3. Problem definition
The dynamic equation of the manipulator can generally be expressed as

\[ u = M(\theta)\dot{\theta} + \tau_c(\theta, \dot{\theta}) + \tau_r(\theta) + \tau_d(\theta, \dot{\theta}) \]  

(1)

Among them, \( \theta, \dot{\theta}, \ddot{\theta} \in \mathbb{R}^n \) respectively represent the angle, angular velocity and acceleration vector of each joint of the robot arm, \( M(\theta) \in \mathbb{R}^{n \times n} \) is the positive definite inertia matrix of the robot arm.
during operation, and \( \tau_\varphi(\theta, \dot{\theta}) \in \mathbb{R}^{n \times n} \) is the matrix force and centrifugal force matrix during operation. \( \tau_g(\theta) \in \mathbb{R}^n \) is the gravity vector of the robot arm. \( \tau_f(\theta, \dot{\theta}) \) is the friction force vector.

Consider the above manipulator dynamic equations, rewrite it as state space

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -M(x_1)^{-1}\left(\tau_\varphi(x_1, x_2)x_2 + \tau_g(x_1) + \tau_f(x_1, x_2)x_2\right) + M(x_1)^{-1}u
\end{align*}
\]

where \( x_1 = \theta, x_2 = \dot{\theta} \).

Define

\[
\begin{align*}
 f(x) &= -M(x_1)^{-1}\left(\tau_\varphi(x_1, x_2)x_2 + \tau_g(x_1) + \tau_f(x_1, x_2)x_2\right) \\
g(x) &= M(x_1)^{-1}
\end{align*}
\]

where \( x = [x_1, x_2]^T \). Then formula (2) can be written as

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= f(x) + g(x)u
\end{align*}
\]

4. Sparse Gaussian process

Gaussian process regression is a data-driven machine learning method, which is a nonparametric model. Its prediction output has probability characteristics and can predict the uncertainty of the output. It is widely used in nonlinear problems. The amount of computation consumed by training and prediction in Gaussian process is related to the number of training samples in power and square. When the number of samples in the training set is increasing, the time required for Gaussian process regression is multiplied. Therefore, in practical application, it is very important to reduce the amount of calculation and accelerate the Gaussian process regression training and prediction. The way to solve this problem is to sparse the Gaussian process regression.

4.1. Gaussian process basics

Suppose a prediction model with n-dimensional input \( X = [x_1, x_2, \cdots, x_n]^T \) and observation output \( y = [y_1, y_2, \cdots, y_n]^T \). Then the regression model with noise can be given by the following formula

\[
\begin{align*}
 f(X) &= X^T w, \\
y &= f(X) + \epsilon
\end{align*}
\]

where \( \epsilon \sim N(0, \sigma^2) \).

A Gaussian process can be uniquely determined by its mean function \( m(x) \) and its covariance function \( \kappa(x, x') \). Its mean value function \( m(x) \) and covariance function \( \kappa(x, x') \) are defined as

\[
\begin{align*}
m(x) &= E[f(x)] \\
\kappa(x, x') &= E[(p(x) - m(x))(p(x') - m(x'))]
\end{align*}
\]

Therefore, a Gaussian process can be defined as

\[
p(x) \sim GP(m(x), k(x, x'))
\]

Then, for dataset \( D = (X, y) \), the output result distribution expression of the prediction with noise can be obtained as follows

\[
\begin{align*}
p(f, | X, D) &\sim N(k(X, X')(K + \sigma^2 I)^{-1}y, k(X, X') - k(X, X')(K + \sigma^2 I)^{-1}k(X, X'))
\end{align*}
\]
Where, \(X\) and \(f\) are prediction input and output, \(k\) is covariance function, \(K\) is covariance matrix, and \(\sigma^2_n\) is noise distribution. From the above formula, it can be concluded that the predicted mean value of test set \(X_\ast\) is 
\[
\langle f(X_\ast) \rangle = k(X_\ast, X)(K + \sigma^2_n I)^{-1} y
\]
and the variance is 
\[
\text{Var}(f(X_\ast)) = k(X_\ast, X) - k(X_\ast, X)(K + \sigma^2_n I)^{-1} k(X, X_\ast).
\]

4.2. Sparsity of prediction process
In practical application, most of the scenarios are used for the prediction process of Gaussian process regression. The prediction process needs to be used repeatedly, and the training only needs to get the optimal parameters once. Here, we first discuss how to predict the Gaussian process regression more quickly.

The time complexity of general Gaussian process regression is \(O(n^3)\). When the number of training samples is large, it will significantly affect the response speed and real-time performance of the system, which is unacceptable.

In order to solve this problem, we need to sparse the training sample set \(X_n\) with the number of \(n_n\), that is, to select the induction point with the number of \(n_u\). Define induced input set \(X_u = \{x_u, x_{u_2}, \ldots, x_{u_{n_u}}\}\), the number of induced input set \(n_u\) is less than the number of training sample set \(n_n\). At the same time, the induction function value of the induction input set is recorded as \(f_u\). Then, for the Gaussian process regression prediction, instead of using all the training samples, the sparse induced input set is used to predict the output function value \(f_\ast\). First, we need to find the posterior distribution of \(f_u\). Suppose that \(f_m\) and \(f_u\) are conditionally independent given \(f_u\), that is
\[
p(f_m, f_u | f_u) = p(f_m | f_u) p(f_u | f_u)
\]
Then we can get the joint prior distribution of the training set function value and the induction set function value.
\[
\begin{bmatrix} f_m \\ f_u \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} m_m \\ m_u \end{bmatrix}, \begin{bmatrix} k(X_m, X_m) & k(X_m, X_u) \\ k(X_u, X_m) & k(X_u, X_u) \end{bmatrix} \right)
\]
(10)

Then, a large number of complete training sets can be discarded, and the sparse induced point function value is used to predict the output function value of the test sample. The joint prior distribution of \(f_u\) and \(f_\ast\) can be expressed as
\[
\begin{bmatrix} f_u \\ f_\ast \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} f_u \\ f_\ast \end{bmatrix}, \begin{bmatrix} m_u & m_\ast \\ m_\ast & K_m \end{bmatrix} \right)
\]
(11)
where
\[
K_m = k(X_u, X_u)
\]
\[
K_{m\ast} = k(X_u, X_\ast)
\]
(12)
The time complexity of the test phase after sparseness is \(O(n_u n_\ast^2)\). In general, the number of test samples and inducement points are far less than the number of complete training samples. Therefore,
the calculation time of conventional Gaussian process regression prediction with $O(n, n^2)$ is greatly reduced.

4.3. Sparsity of training process

In order to speed up the training process, each training sample can be used separately. First of all, suppose that for a given $f_x$, the $n_m$ outputs of the training sample set are completely independent, that is

$$p\left(f_{m_1}, f_{m_2}, \ldots, f_{m_n} \mid f_x\right) = p\left(f_{m_1} \mid f_x\right)p\left(f_{m_2} \mid f_x\right) \cdots p\left(f_{m_n} \mid f_x\right)$$

(13)

which is called Fully Independent Training Conditional (FITC).

So we first use measurement $(x_m, \hat{f}_m)$ to predict the distribution of $f_u$, we write this as $N(\mu_m^u, \Sigma_{mu})$. Then we use $(x_m, \hat{f}_m)$ to find $N(\mu_m^u, \Sigma_{mu})$, and so on. Finally, these distributions are combined to get the final distribution of $f_u$. The joint distribution of $f_m$ and $f_u$ can be expressed as

$$
\begin{bmatrix}
  f_m \\
  f_u
\end{bmatrix} 
\sim N
\left(
\begin{bmatrix}
  \mu_m \\
  \mu_u
\end{bmatrix},
\begin{bmatrix}
  \Sigma_{mm} & \Sigma_{mu} \\
  \Sigma_{mu} & \Sigma_{uu}
\end{bmatrix}
\right)
$$

$$
\begin{bmatrix}
  \mu_u \\
  \mu_m
\end{bmatrix} =
\begin{bmatrix}
  m_u + \Sigma_{mm}\hat{f}_m^{-1}(\hat{f}_m - m_u) \\
  m_m + \Sigma_{mm}\hat{f}_m^{-1}(\hat{f}_m - m_m)
\end{bmatrix},
$$

$$
\Sigma_{mu} = \hat{\Sigma}_{f_m}\left(\Lambda_{mm} + \hat{\Sigma}_{f_m}\right)^{-1}K_{mu}\Lambda_{uu}K_{mu}^\top,
$$

$$
\Sigma_{uw} = K_{wu}\Lambda_{wu}^{-1}K_{wu},
$$

(14)

The training sample set is divided into a certain number of subsets, which are used to obtain the distribution of induction function value $f_x$, and then combined. This algorithm is called Partially Independent Training Conditional (PITC), which is obviously more flexible.

5. Trajectory tracking control of manipulator

In this chapter, we use the Gaussian process regression to identify the manipulator system, and design the feedback linearization control law of the event trigger mechanism to realize the trajectory tracking control of the manipulator.

5.1. System identification of manipulator

The manipulator system is a multi input and multi output system, and the Gaussian process algorithms proposed in the previous chapters are all aimed at single input and single output system. Although the Gaussian process regression algorithm is naturally suitable for the case of multiple inputs, but for the case of multiple outputs, the general Gaussian process regression algorithm can not solve. Therefore, in order to identify the multi input and multi output manipulator system, it is necessary to extend the general Gaussian process regression algorithm to the multi output case.

Suppose that data sets $X = \{x_j \mid j = 1, \ldots, m, i = 1, \ldots, n^i\}$ and $Y = \{y_j \mid j = 1, \ldots, m, i = 1, \ldots, n^i\}$ are input training data sets and output data sets of different tasks. At the same time, task $j$ has a total of $n^j$ training sample points. One of the most intuitive and simple solutions for the multi output situation is to assume that these $m$ tasks are independent of each other, and use $m$ independent single output
Gaussian process regression to predict, respectively, to get the prediction values of different outputs, as shown in Figure 1 (a). However, for a manipulator system, the joints are connected, and the angles between the joints cannot be completely independent, and there must be correlation, so it is not feasible to use multiple independent single output Gaussian process regression. Therefore, another way to consider is multi output Gaussian process regression, which introduces the correlation between tasks, as shown in Figure 1 (b).

\[ \begin{align*}
\text{(a)} & \quad \text{Single output GP} \\
\text{Training} & \quad \text{Predication} \\
\text{Multiple independent single output Gaussian processes; (b) Multi output Gaussian process} \\
\text{(b)} & \quad \text{Multi-Output GP} \\
\text{Training} & \quad \text{Predication}
\end{align*} \]

Then, for the multi output Gaussian process regression method considering the correlation between tasks, its covariance function needs to be adjusted, that is, to add a covariance representing the output correlation to the covariance function of the original input sample, which is defined as

\[ k_{ul}(x, x', l, l') = k_i(l, l') \times k_j(x, x') \]  

(15)

where \( k_i \) represents the correlation between multiple different outputs, and \( k_j \) measures the correlation of the input corresponding to the output \( l \). Assume that the number of training samples output is the same, the covariance matrix of multiple output Gaussian process regression can be expressed as

\[ K_{ul}[(x, l), (x', l')] = K_i(l, l') \otimes K_j(x, x') \]  

(16)

Here, \( \otimes \) is the Kronecker product. \( K_i \) is the covariance matrix of order \( mn \times mn \).

The next problem to be solved is how to obtain the covariance matrix \( K_i \) to measure the correlation between outputs. Therefore, the spherical surface of \( K_i \) is parameterized according to the method in [22], so that

\[ K_i = S^T S \text{ diag}(\gamma) \]  

(17)

Where \( \gamma = [\gamma_1, \ldots, \gamma_m] \) is a parameter vector that adjusts the scale of different outputs,

\[ S = \begin{bmatrix}
1 & \cos(\phi_1) & \cos(\phi_2) \\
0 & \sin(\phi_1) & \sin(\phi_2) \cos(\phi_3) \\
0 & 0 & \sin(\phi_2) \sin(\phi_3)
\end{bmatrix} \]  

(18)

Where \( \phi = [\phi_1, \phi_2, \phi_3] \) is also the parameter vector. After parameterizing \( K_i \), the optimal parameters can be obtained by minimizing the log likelihood like other Gaussian process parameters. So we can get the estimates of \( f(x) \) and \( g(x) \) according to the Gaussian process regression.

5.2. Design of control law

First, we consider that the expected trajectory of state \( x_i \) is \( x_j(t) \), And \( x_j(t) \) is bounded and differentiable at least in order \( n-1 \). Definition

\[ x_j(t) = \begin{bmatrix} x_j & \dot{x}_j & \cdots & \frac{d^{n-1}x_j}{dt^{n-1}} \end{bmatrix}^T \]  

(19)
Define tracking error as
\[ e = x - x_d \] (20)

Consider the following filtered scalar States
\[ r = [\lambda^T \ 1]e \] (21)

Where \( \lambda \) is a coefficient vector, and
\[ \dot{r} = f(x) + g(x)u(x) + \rho \]
\[ \rho = \lambda^T e_{2n} - \frac{d^n x_d}{dt^n} \]

\[ e_{2n} = [e_2 \cdots e_n]^T \in \mathbb{R}^{n-1} \]

The control law \( u(x) \) defined as
\[ u_x(x) = \frac{1}{\hat{g}_x(x)} \left( -\hat{f}_x(x) + v \right) \]
\[ v = -k_r r - \rho, \quad k_r \in \mathbb{R}_+ \] (23)

5.3. Event-triggered model updating

How to determine when to acquire new training samples and update the model is still a problem. When the performance of the system is good enough, it is obviously unnecessary to update the additional calculation amount and time. Therefore, a data-dependent and efficient online learning strategy should only update the data set and model when necessary, rather than simply updating at regular intervals. This is called event-triggered control. First, analyze the convergence of the control law.

Consider a commonly used Lyapunov function
\[ V_x(x) = \frac{r^2}{2}, \quad \forall \kappa \in \mathbb{N}_0 \] (24)

and
\[ \dot{V}_x(x) = r \dot{r} = r \left( f + gu_x + \rho \right) \]
\[ = r \left( f + \frac{g}{\hat{g}_x} \left( -\hat{f}_x + k_r r - \rho \right) + \rho \right) \]
\[ = r \left( f - \frac{g}{\hat{g}_x} \hat{f}_x - k_r \hat{g}_x r^2 + (1 - \hat{g}_x) r \rho \right) \] (25)

Suppose the function \( f(x) \) has a regenerated kernel Hilbert space norm with known hyperparameters about the square exponential kernel \( k(\cdot, \cdot) \), which is expressed as \( \|f(x)\|_2 \leq B_f \). Then there is
\[ \Pr \left\{ \|\mu_x(x) - f(x)\| \leq \beta_s \sigma_k(x), \forall x \in \tilde{X}, N_x \in \mathbb{N}_0 \right\} \geq 1 - \delta \] (26)

established on compact set \( \tilde{X} \subset \mathbb{R}^n \).

Where
\[ \delta \in (0,1) \]
\[ \beta_s = \sqrt{2B_f + 300\gamma_s \log((\kappa + 1) / \delta)} \] (27)

For arbitrary \( k_r > 0 \), \( \|e\| \) is globally consistent and ultimately bounded, and its final limit is
\[ B_x = \left\{ \forall x \in X \mid \|e\| \leq \frac{\beta_s \sigma_k}{k_r} \right\}, \quad \forall x_0 \in X \] (28)
Where \( \sigma(x) := \max_{x \in X} \sigma_x(x) \).

Then, (25) can be written as

\[
\dot{V}_s(x) \leq r \Delta f(x) - k_r x^2
\]

(29)

Where \( \Delta f(x) \) is the estimated error. Therefore, the following event trigger conditions are proposed

\[
t_{x+1} := \{ t > t_x \mid \Delta f(x) \geq k_r x^2 \}
\]

(30)

### 6. Simulation results

Consider the manipulator model as shown in Figure 2. Its physical parameters are shown in Table 1.

**Figure 2 Schematic diagram of the manipulator model**

| Symbol | Numerical value | Symbol | Numerical value |
|--------|-----------------|--------|-----------------|
| \( m_2 \) | 0.5 kg | \( r_2 \) | 0.5 m |
| \( m_3 \) | 1.5 kg | \( r_3 \) | 0.85 m |
| \( l_2 \) | 1 m | \( J_1 \) | 5 kg·m² |
| \( l_3 \) | 1 m | \( J_2 \) | 5 kg·m² |
| \( g \) | 9.81 m/s² | \( J_3 \) | 5 kg·m² |

In the experiment, the robotic arm tracks the trajectory obtained in advance. The initial angle of each joint of the robot arm is \( \theta(0) = [1.6, 0.6, 0]^T \text{ rad} \). The initial angular velocity is \( \dot{\theta}(0) = [0, 0, 0]^T \text{ rad/s} \), and the coefficient of friction is \( f_1 = f_2 = f_3 \) = 1.5. The reference trajectory is used. Figure 3 and Figure 4 show the angle tracking performance of each joint and the angle tracking error. It can be seen that the error is very large at the beginning. The reasons include that the Gaussian process regression has fewer training samples at the beginning, which leads to inaccurate model...
identification with the passage of time, the number of sample points increases, and its error decreases rapidly. Figure 5 shows the transformation of the control torque of each joint. Figure 6 shows the model update after the event is triggered.

Figure 3 Angle tracking of each joint

Figure 4 Angle tracking error of each joint
References

[1] Bradley, and Brian Scassellati 2016. Autonomously constructing hierarchical task networks for...
planning and human-robot collaboration. 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE.

[2] Lemaignan, Séverin, et al 2017. Artificial cognition for social human–robot interaction: An implementation. Artificial Intelligence 247 (): 45-69.

[3] Kuehn, Johannes, and Sami Haddadin 2016. An artificial robot nervous system to teach robots how to feel pain and reflexively react to potentially damaging contacts. IEEE Robotics and Automation Letters 2.1: 72-79.

[4] Darvish, Kourosh, et al 2018. Flexible human–robot cooperation models for assisted shop-floor tasks. Mechatronics 51: 97-114.

[5] Kopeck, P. "Development trends in robotics." IFAC-PapersOnLine 49.29 (2016): 36-41.

[6] Wood, Stephen P., et al 2012. The potential regulatory challenges of increasingly autonomous motor vehicles. Santa Clara L. Rev. 52: 1423.

[7] Jaeger, Herbert 2003. Adaptive nonlinear system identification with echo state networks. Advances in neural information processing systems.

[8] Alonge, Francesco, et al 2016. Adaptive feedback linearizing control of linear induction motor considering the end-effects. Control Engineering Practice 55 (): 116-126.

[9] Kumar, Anupam, and Vijay Kumar 2017. Hybridized ABC-GA optimized fractional order fuzzy pre-compensated FOPID Control design for 2-DOF robot manipulator. AEU-International Journal of Electronics and Communications 79: 219-233.

[10] Kumar, Anupam, and Vijay Kumar 2017. Evolving an interval type-2 fuzzy PID controller for the redundant robotic manipulator. Expert Systems with Applications 73: 161-177.

[11] Jaeger, Herbert 2003. Adaptive nonlinear system identification with echo state networks. Advances in neural information processing systems.

[12] Huang, Xinge, et al 2019. Exploration in Extreme Environments with Swarm Robotic System. 2019 IEEE International Conference on Mechatronics (ICM). Vol. 1. IEEE.

[13] Chen, Yuzhen, et al 2014. Error modeling and sensitivity analysis of a parallel robot with SCARA (selective compliance assembly robot arm) motions. Chinese Journal of Mechanical Engineering 27.4: 693-702.

[14] Bouakrif, Farah, and Michel Zasadzinski 2016. Trajectory tracking control for perturbed robot manipulators using iterative learning method. The International Journal of Advanced Manufacturing Technology 87.5-8: 2013-2022.

[15] Sicard, Pierre, and Nicolas Lechevlin 1997. Variable structure observer design for flexible joint manipulators. Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041). Vol. 3. IEEE.

[16] Yu, Chong, and Xiong Chen 2015. Trajectory tracking of wheeled mobile robot by adopting iterative learning control with predictive, current, and past learning items. Robotica 33.7: 1393-1414.

[17] Beckers, Thomas, Jonas Umlauf, and Sandra Hirche 2017. Stable model-based control with Gaussian process regression for robot manipulators. IFAC-PapersOnLine 50.1: 3877-3884.

[18] Fang, Ge 2019. Vision-based online learning kinematic control for soft robots using local gaussian process regression. IEEE Robotics and Automation Letters 4.2: 1194-1201.

[19] Aswani, Anil 2013. Provably safe and robust learning-based model predictive control. Automatica 49.5: 1216-1226.

[20] Wabersich, Kim P., and Melanie N. Zeilinger 2018. "Linear model predictive safety certification for learning-based control." 2018 IEEE Conference on Decision and Control (CDC). IEEE.

[21] Koller, Torsten 2018. Learning-based model predictive control for safe exploration. 2018 IEEE Conference on Decision and Control (CDC). IEEE.

[22] Osborne, Michael A 2010. Bayesian Gaussian processes for sequential prediction, optimisation and quadrature. Diss. Oxford University, UK.