

Mathematic Modeling of Path Tracking with Adaptive Backstepping

1 Introduction

The equation of motion for the constrained robotic manipulator with n degrees of freedom is given in the joint space as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \quad (1)$$

where:

- $q \in \mathbb{R}^{n \times 1}$ denotes the joint angles or link displacements of the manipulator
- $M(q) \in \mathbb{R}^{n \times n}$ is the robot inertia matrix which is symmetric and positive definite
- $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ contains the centripetal and Coriolis terms
- $G(q) \in \mathbb{R}^{n \times 1}$ are the gravity terms
- $u \in \mathbb{R}^{n \times 1}$ denotes the torque or the force

We want the manipulator to follow the track described by the equation $C(\epsilon) = 0$, with $\epsilon = (x, y, z)^T$, position of the end effector (e.e.). For example a circular track in the (x, y) plane can be described as:

$$C(\epsilon) = (x - x_0)^2 + (y - y_0)^2 - R^2 \quad (2)$$

2 Backstapping Control

We are looking for a control strategy such that the e.e. follows the track with a desired velocity α . Let's define a Lyapunov candidate function:

$$V = \frac{1}{2} C(\epsilon)^T C(\epsilon)$$

Its derivative is:

$$\dot{V} = C(\epsilon)^T C_\epsilon J \dot{q} = C(\epsilon)^T C_\epsilon J u'$$

with J , the Jacobian, $C_\epsilon = \partial C(\epsilon) / \partial \epsilon$ and the hypothesis that the joint velocity can be imposed (neglecting the dynamics). The expression of the control which makes the derivative of the Lyapunov candidate negative defined is:

$$u' = \Gamma_{\dot{q}}(\epsilon) = -J^R C_\epsilon C(\epsilon)^T \nu + J^R S(\epsilon) \alpha$$

with ν and α scalar parameters and (ϵ) vector tangent to $C(\epsilon) = 0$, such that $C_\epsilon^T S(\epsilon) = 0$ or rather that does not deviate the e.e. from the track. The derivative of the Lyapunov candidate will be:

$$\dot{V} = -C(\epsilon)^T C_\epsilon C_\epsilon^T C(\epsilon) \nu$$

which is negative definite. This was the first part of the backstepping. Now we introduce the dynamics of the system. Let's consider a new Lyapunov candidate function for the state (ϵ, \dot{q}) :

$$W(\epsilon, \dot{q}) = \frac{1}{2}C(\epsilon)^T C(\epsilon) + \frac{1}{2}(\dot{q} - \Gamma_{\dot{q}})^T (\dot{q} - \Gamma_{\dot{q}})$$

Now we evaluate the derivative:

$$\dot{W}(\epsilon, \dot{q}) = C(\epsilon)^T C_{\epsilon} J \dot{q} + (\ddot{q} - \dot{\Gamma}_{\dot{q}})^T (\dot{q} - \Gamma_{\dot{q}})$$

Let's note that we can not substitute \dot{q} with u' , since we are considering the model dynamics. Now we try to obtain a derivative similar to \dot{V} . We sum and subtract the term $-C(\epsilon)^T C_{\epsilon} C_{\epsilon}^T C(\epsilon) \nu$ and we obtain:

$$\dot{W}(\epsilon, \dot{q}) = -C(\epsilon)^T C_{\epsilon} C_{\epsilon}^T C(\epsilon) \nu + C(\epsilon)^T C_{\epsilon} (J \dot{q} + C_{\epsilon}^T C(\epsilon) \nu) + (\ddot{q} - \dot{\Gamma}_{\dot{q}})^T (\dot{q} - \Gamma_{\dot{q}})$$

We substitute the model equation of motion

$$\ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - G(q))$$

And we obtain:

$$\dot{W}(\epsilon, \dot{q}) = -C(\epsilon)^T C_{\epsilon} C_{\epsilon}^T C(\epsilon) \nu + C(\epsilon)^T C_{\epsilon} (J \dot{q} + C_{\epsilon}^T C(\epsilon) \nu) + (M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - G(q)) - \dot{\Gamma}_{\dot{q}})^T (\dot{q} - \Gamma_{\dot{q}})$$

So we can choose the following τ expression to make \dot{W} negative definite:

$$\tau = C(q, \dot{q})\dot{q} + G(q) + M(q)(\dot{\Gamma}_{\dot{q}} - (\dot{q} - \Gamma_{\dot{q}})^{-1}(C(\epsilon)^T C_{\epsilon} (J \dot{q} + C_{\epsilon}^T C(\epsilon) \nu)) - K_b(\dot{q} - \Gamma_{\dot{q}})$$

In fact, we obtain:

$$\dot{W}(\epsilon, \dot{q}) = -C(\epsilon)^T C_{\epsilon} C_{\epsilon}^T C(\epsilon) \nu - K_b(\dot{q} - \Gamma_{\dot{q}})^T (\dot{q} - \Gamma_{\dot{q}}) \quad (3)$$

3 Implementation

The control technique was implemented in Python and tested with ROS and Gazebo. We generate a class based on urdf2casadi which is able to load the motion matrices and the forward kinematics of the manipulator from an URDF file. The track $C(\epsilon)$ is passed by the user and the derivative is evaluated in Python using CasADi library. The control is then evaluated as presented before and the q value is read with a ros subscriber from the model.

It is important to notice that the ϵ value is obtained from the forward kinematics and that $\dot{\Gamma}_{\dot{q}}$ is evaluated as:

$$\dot{\Gamma}_{\dot{q}} = \frac{\partial \Gamma}{\partial q} \frac{\partial q}{\partial t} = \frac{\partial \Gamma}{\partial q} J$$

Where $\partial \Gamma / \partial q$ is again obtain by CasADi derivation.

4 Adaptive version

We tried to implement also an adaptive version of the algorithm. To reduce the complexity of the model we consider just an unknown value of gravity (representative of a condition of tilted arm). The gravity matrix $G(q)$ will be affected by this modification.

Since the gravity acceleration is a common factor in all the terms of the gravity matrix, the dynamics can be easily linearized as follows:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \bar{G}(q)g$$

So we can define the dynamics parameter $\pi = g$, the estimated gravity value $\hat{\pi} = \hat{g}$ and the difference between them $\tilde{\pi} = \hat{\pi} - \pi = \hat{g} - g$. Let's note that g is the true value of g , perceived in the direction

of the motion of the robot. We then introduce a control for the estimated gravity term, which will simply be:

$$\dot{\tilde{\pi}} = +\dot{\hat{\pi}} = +u_\pi$$

So, the imposed control will be:

$$\tau = C(q, \dot{q})\dot{q} + \hat{G}(q) + M(q)(\dot{\Gamma}_{\dot{q}} - (\dot{q} - \Gamma_{\dot{q}})^{-1}(C(\epsilon)^T C_\epsilon(J\dot{q} + C_\epsilon^T C(\epsilon)\nu)) - K_b(\dot{q} - \Gamma_{\dot{q}})$$

The new Lyapunov candidate should also consider the $\tilde{\pi}$ term:

$$W(\epsilon, \dot{q}, \tilde{\pi}) = \frac{1}{2}C(\epsilon)^T C(\epsilon) + \frac{1}{2}(\dot{q} - \Gamma_{\dot{q}})^T(\dot{q} - \Gamma_{\dot{q}}) + \frac{1}{2}R\tilde{\pi}^T \tilde{\pi}$$

And the resulting derivative will be:

$$\dot{W}(\epsilon, \dot{q}, \tilde{\pi}) = -C(\epsilon)^T C_\epsilon C_\epsilon^T C(\epsilon)\nu - K_b(\dot{q} - \Gamma_{\dot{q}})^T(\dot{q} - \Gamma_{\dot{q}}) + (M(q)^{-1}(-G(q) + \hat{G}(q)))^T(\dot{q} - \Gamma_{\dot{q}}) + R\tilde{\pi}u_\pi$$

$$\dot{W}(\epsilon, \dot{q}, \tilde{\pi}) = -C(\epsilon)^T C_\epsilon C_\epsilon^T C(\epsilon)\nu - K_b(\dot{q} - \Gamma_{\dot{q}})^T(\dot{q} - \Gamma_{\dot{q}}) + (M(q)^{-1}\bar{G})^T \tilde{\pi}(\dot{q} - \Gamma_{\dot{q}}) + R\tilde{\pi}u_\pi$$

In order to obtain a definite negative derivative, we choose u_π :

$$u_\pi = -\frac{1}{R}(M(q)^{-1}\bar{G})^T(\dot{q} - \Gamma_{\dot{q}})$$

And we obtain the final derivative:

$$\dot{W}(\epsilon, \dot{q}, \tilde{\pi}) = -C(\epsilon)^T C_\epsilon C_\epsilon^T C(\epsilon)\nu - K_b(\dot{q} - \Gamma_{\dot{q}})^T(\dot{q} - \Gamma_{\dot{q}}) \quad (4)$$

Let's note that the $\tilde{\pi}$ term is not present. The function is just semi-negative definite and the acceleration estimate will go to the real value just if the track is persintently excitatory.