Time-Varying Linear Quadratic Regulator Design for a 2R Planer Robotic Arm

Malintha Fernando

Introduction

The use of robotic manipulators are rapidly increasingly in numerous domains including the manufacturing industry, mobile robots and even on unmanned aerial vehicles as grasping devices. Therefore, controlling of such manipulators are getting a wide popularity in the robotics community. In real-world, it is necessary for the robots to arbitrate between the decisions pertaining to the optimality of the system performances such as the required precision and power constraints in different circumstances. Optimal control theory a) provides much more authority over the convergence and the control effort in contrast to classical feedback control methods and, b) shows more robustness toward the modelling errors that are rife in practical systems.

This report presents a technical discussion of a Time-Varying Linear Quadratic Regulator (TVLQR) design for a revolute-revolute (RR) robotic arm. We represent the kinematics of an RR robotic arm as a non-linear dynamic systems and perform trajectory tracking with assistance from solving the inverse kinematics (IK). We perform the system linearization around fixed points that are provided by the IK solver prior to applying the LQR. The numerical simulation results show that the control system is capable of rapidly stabilizing the end-effector of the robotic arm along a predefined trajectory even at the presence of model uncertainty.

System Design

We first consider the forward kinematics of a planer RR robotic arm showed in Fig.1 and represent in the state space form. Let $x \in \mathbb{R}^{2\times1}$, $u \in \mathbb{R}^{2\times1}$, be the system state space and the control inputs. For $A \in \mathbb{R}^{2\times2}$, $B \in \mathbb{R}^{2\times1}$, we can write the system in the standard form,

$$\dot{x} = Ax + Bu. \tag{1}$$

We define the state space of the system as the $[x_1 x_2]^T$ where x_1,x_2 are the Cartesian coordinates of the end-effector position and the control inputs as $[u_1 u_2]^T$ where $u_1 = \dot{\theta}_1$ and $u_2 = \dot{\theta}_2$ are the angular velocities of the first and the second joint respectively. Therefore, the state space form of the system can be written as

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}-L_1\sin(\theta_1) - L_2\sin(\theta_1+\theta_2) & -L_2\sin(\theta_1+\theta_2)\\ L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) & L_2\cos(\theta_1+\theta_2)\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}.$$

As θ_1, θ_2 changes with time as the arm moves, we let $B_t = B(\theta_1, \theta_2)$. Further, we identify the kinematic singularities of the B matrix when the arm is fully extended,

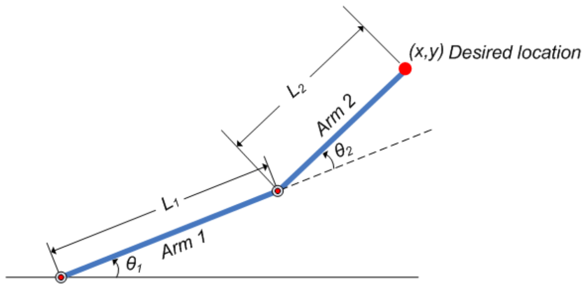


Figure 1: A 2R planer robotic arm.

making it impossible to enforce a linear velocity along the axis of the arm itself. Thus, we initialize the system with a slightly bent home configuration ($\theta_1 = \theta_2 = \delta\theta$) and limit the workspace to a subspace constrained by $\delta\theta$ of the original space.

Linear Quadratic Regulator Design

One prevailing condition in the LQR is that the linearity of the system defined by the state space form. As the analytical B matrix of our system is trigonometric, we attempt to obtain a linearized version of it at every discretized time-step. The standard procedure of linearizing a non-linear dynamic system is expanding the system as a Taylor series around a given fixed position $(\bar{x}, \bar{u})$, where the system is assumed to be stabilized. However, as the dynamic system is already linear in terms of the state and control inputs (A and B matrices do not contain state and control values in them), their partial derivatives with respect to state and control would remain the same. Thus, the first order derivative matrices resulting from the Taylor expansion would be identical to the current A, B matrices. Therefore, we simply obtain the linearized system by evaluating B matrix at $(\bar{x}, \bar{u})$ coordinates of the fixed points. we consider the fixed points for the system as a nearby state provided by an external system such as a trajectory generator. The LQR provides the optimal control input that requires to stabilize the system at the fixed points given the current state, as long as they are close enough, thus the linearization is valid.

Therefore, when it is required to stabilize the end-effector at a configuration that is far away from the current configuration, a common practice is to couple the system with an external trajectory generation mechanism. The trajectory generation may provide a series of “near enough” configurations for the LQR system to stabilize around and thus the linearization assumption would prevail throughout the trajectory. For this phase of the system we can select from a series of example trajectory generation methods for manipulator robots such as Cartesian trajectories, constant screw paths and joint configuration trajectories. In this work, we obtained the continuous path using a Cartesian trajectory limited to the modified sub-workspace with a series of waypoints. We then used inverse kinematics of the open chains to obtain the fixed configurations for the joint angles at every time-step and calculated the time varying and linear B_t matrix evaluated the resulting θ_1, θ_2.

In this work, we discretize the system at every dt timestep to obtain the linear system around the joint angles corresponding to the desired state of the system. The LQR solves the so-called “Ricatti equation” for the continuous time dynamic system specified in Eq. (1) to produce the gain matrix k. Briefly, the LQR finds the gain matrix by minimizing the quadratic cost function defined as

$$J = \int_{t=0}^{\infty} \{x_t^T Q x_t + u_t^T R u_t\} dt,$$

where Q, R are square matrices that penalizes the state and control outputs of the system at any given time. In the discrete time systems, the solution for the above cost function can be found by performing the value iteration using the dynamic programming principal. The gain matrix can also be seen as an optimal policy to drive any near-enough system configurations to the equilibrium. The optimal control u for the system can be obtained by simply solving

$$u = -k\hat{x},$$

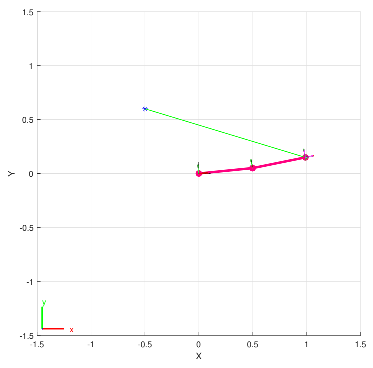
where $\hat{x} = x - \bar{x}$, the deviation of the current state from the fixed point. We iteratively perform this procedure for fixed-points generated by the Cartesian trajectory to navigate the system to the desired final configuration along the trajectory.

Implementation and Results

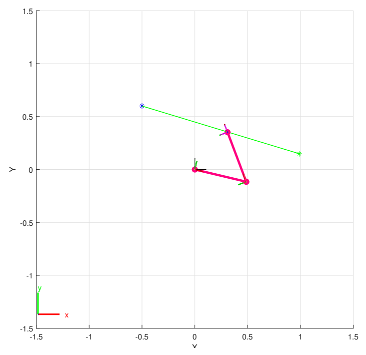
We implemented the system in MATLAB with its Robotics and Control toolboxes. The robot arm is modeled as a rigid body tree. The singularities of the system can be omitted by using constrained, generalized inverse kinematics function. However, we observed that the generalized kinematics solver to be too slow compared to the open chain inverse kinematics and as an alternative, we manually limited the desired final configuration to constrained workspace. The input Cartesian trajectory is calculated using the ModernRobotics package and used to provide the fixed points to the system for the linearization and the stabilization. We calculated the LQR gain matrix k using the inbuilt LQR function of the MATLAB Control toolbox. In order to simulate the actual behaviour of the manipulator with the resulting optimal control around the fixed points, we designed an integrator system using the ODEs and integrated using the ODE45 function of MATLAB with a much finer time resolution.

We used a slightly bent arm with $\delta\theta = 0.1$, $x_0 = [0.987\ 0.149]^T$ as the initial condition for our system, $x_F = [-0.5\ 0.6]^T$ as the final end effector configuration and $L_1 = L_2 = 0.5$m. The Cartesian trajectory is calculated using the fifth order time scaling and 0.1s time intervals with a duration of 5s. The Fig. 2 shows the initial and the desired end-effector configurations along with the trajectory. We experimented using different penalization values for P and Q matrices and observed that the system performances to be dramatically changed. Specifically, we observed that much smaller penalization values in R matrix resulting in highly aggressive control actions almost making the arm to jump between the fixed-points, while larger values makes the system to converge much slower toward the fixed points.

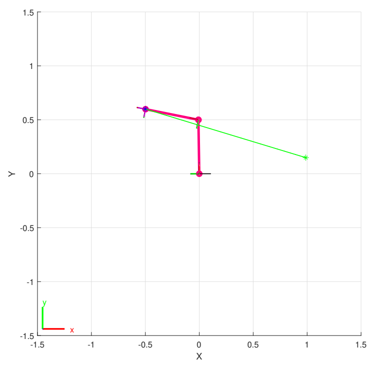
We present the results for two different control penalization values ($R_1 = I$, $R_2 = 0.1I$) where I is the two identity matrix. Fig. 3 (a) and Fig. 3 (c) shows the displacement error of the system for the two values. It can be clearly seen that the control in the latter system seeks to bring the errors to zero with much aggressive controls. As a result, the end-effector position and the joint angles changes dramatically with much larger steps between the fixed-points (Fig. 3(e) and (f)).



(a)

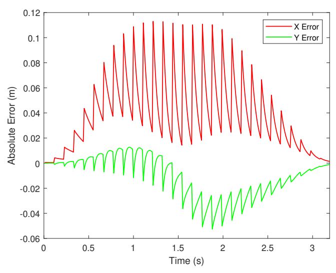


(b)

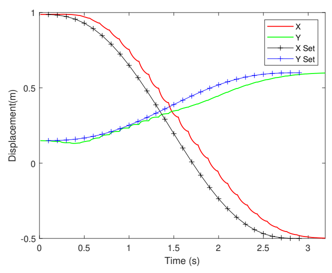


(c)

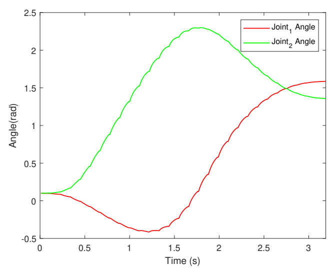
Figure 2: Initial (a), interim (b) and desired (c) configurations of the end-effector. The green path shows the trajectory in the Cartesian plane.



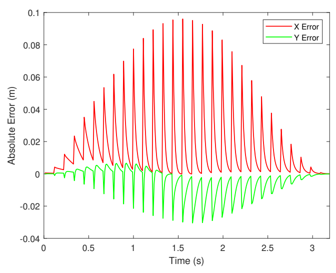
(a)



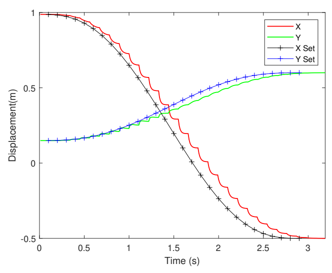
(b)



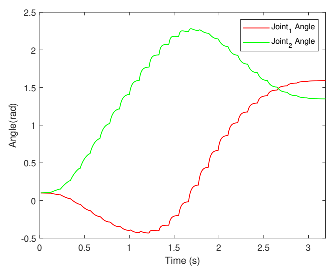
(c)



(d)



(e)



(f)

Figure 3: The plots on the top and bottom rows correspond to higher and lower penalization of the control. The displacement error of the end-effector (a)(c), end-effector position(b)(e), and joint angles (c)(f) against the time throughout the trajectories.