University of Minnesota–Twin Cities

Human Robotic Control

Two Link Arm Simulation with LQR

RanTian
UMID: 4970588

Feb 21, 2015

Abstract

For the two-link arm model, we already have the nonlinear dynamic representations. Thus base on these dynamics, we can linearize the model at particular point and minimize the cost function using LQR. This file contains two parts, the first one is the simulation of linearization at the equilibrium point, the other one is trying to hit the target state with minimum cost by adding the input derivation and state derivation from previous trajectory. The second part is the extension of the first one while there are linear and constant terms in the cost function of LQR.

2-Link-Arm Model

Based on the textbook of Spongrobot[1], the *two link dynamic model* is:

$$D(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) = u \tag{1}$$

where $D(\theta)$ and $C(\theta, \dot{\theta})$ are as following:

$$\begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix}$$

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\begin{aligned}
D_{11} &= m_1 {c_1}^2 + m_2({l_1}^2 + {c_2}^2 + 2l_1 c_2 cos(\theta_2)) + I_1 + I_2 \\
D_{12} &= D_{21} = m_2({c_2}^2 + l_1 c_2 cos(\theta_2)) + I_2 \\
D_{22} &= m_2 {c_2}^2 + I_2 \\
C_{11} &= -2m_2 l_1 c_2 sin(\theta_2)\dot{\theta}_2 \\
C_{12} &= -m_2 l_1 c_2 sin(\theta_2)\dot{\theta}_2 \\
C_{21} &= m_2 l_1 c_2 sin(\theta_2)\dot{\theta}_2 \\
C_{22} &= 0
\end{aligned}$$

The state space representation yields to:

$$\begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ & 0 & -D^{-1}C & \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ D^{-1} \end{pmatrix} u \tag{2}$$

Part 1

Given the 2-link-arm linearized model,

$$x_{k+1} = A_k x_k + B_k u_k \tag{3}$$

We linearize at the equilibrium point and use this linearized model to minimize the cost function in the finite horizon-N:

$$L = \sum_{k=1}^{N-1} (u_k^T R_k u_k + x_k^T Q_k x_k) + x_N^T Q_N x_N \tag{4}$$

the optimal input of this cost function is;

$$u_k^\star = K_k x_k \tag{5}$$

where

$$K_k = -(R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k \tag{6}$$

$$P_{k-1} = Q_{k-1} + A_{k-1}^T P_k A_{k-1} - A_{k-1}^T P_k B_{k-1} (R_{k-1} + B_{k-1}^T P_k B_{k-1})^{-1} B_{k-1}^T P_k A_{k-1} \tag{7}$$

Equation (6) corresponds to the gain for the state k and equation (7) is the *Discrete Riccati Equation.*[2]
In the backward direction:

```
for i = N:-1:2
         P(:, :, i-1) = Q + AT*P(:, :, i)*F - ...
                     AT*P(:, :, i)*B/(R+BT*P(:, :, i)*B)*BT*P(:, :, i)*A;
end
```

Then in the forward direction:

```
for j = 1:(N-1)
        K(:, :, j) = - (R+BT*P(:, :, j+1)*B)\BT*P(:, :, j+1)*A;
        u(:, j) = K(:, :, j)*[x1(j); x2(j); x3(j); x4(j)];
end
```

Simulation Results

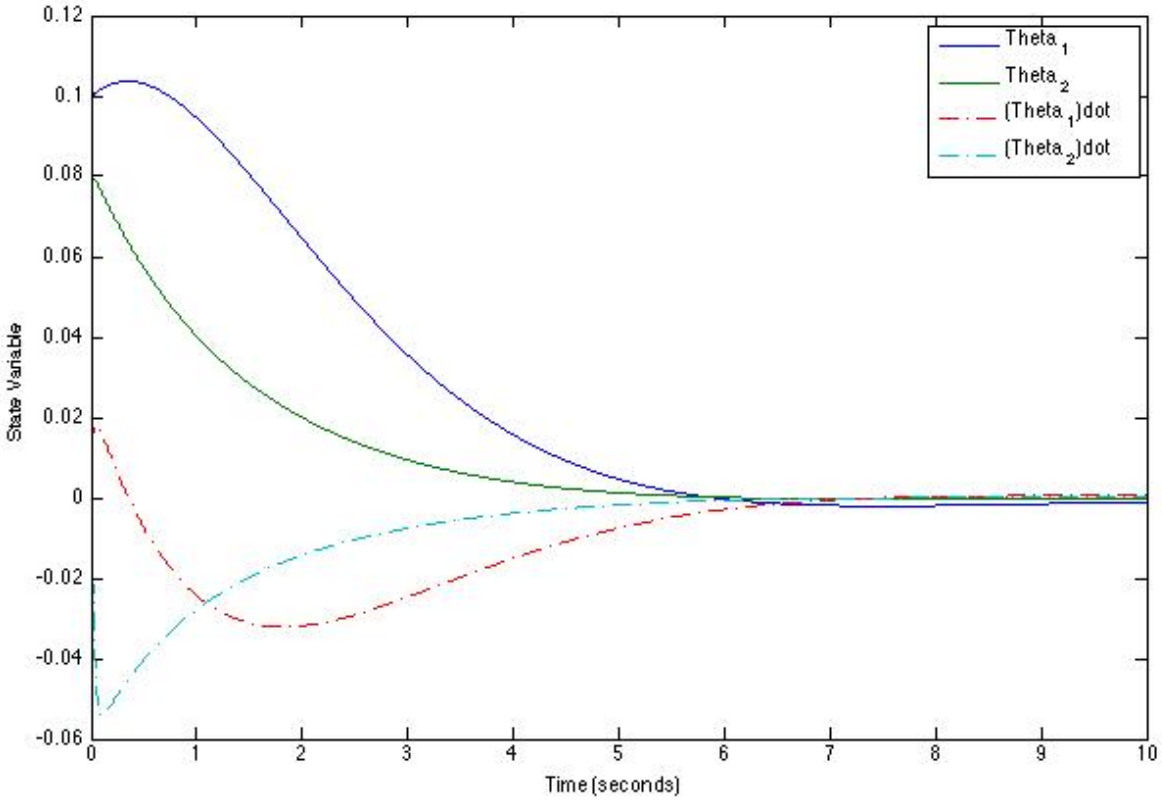


Figure 1: Initialization Around Equilibrium Point with LQR

The state will converge to 0 in around 5 seconds.[3]

Part 2

For the target state which is not far away from the initial state, it is possible for us to use linearized model to minimize the cost function with LQR. Given the finite horizon cost function:

$$L = \sum_{k=1}^{N-1} u_k^T R_k u_k + (x_N - x_{target})^T Q_N (x_N - x_{target}) \tag{8}$$

Initially, like part1, we linearize around the initial state and get the first trajectory to the target state:

$$x_{k+1}^1 = A_k^1 x_k^1 + B_k^1 u_k^1 \tag{9}$$

Obviously, this trajectory is not optimal with respect to our cost function (8), thus we add input derivation (Δu_k)and state derivation(Δx_k)from previous trajectory and finally we get the convergence after few iterations. The form of the cost function after we add the derivation each time is:

$$L^{m+1} = (x_N^m + \Delta x_N^m)^T Q_N (x_N^m + \Delta x_N^m) + \sum_{k=1}^{N-1} (u_k^m + \Delta u_k^m)^T R_k (u_k^m + \Delta u_k^m) \tag{10}$$

which then yields to:

$$L^{m+1} = L^m + \Delta x_N^{mT} Q_N \Delta x_N^m + 2({h_N}^T + {x_N}^T Q_N)\Delta x_N^m + \sum_{k=1}^{N-1} \Delta u_k^{mT} R_k \Delta u_k^m + 2(R_k u_k^m)^T \Delta u_k^m \tag{11}$$

Then we end up getting the LQR optimal problem with respect to (Δu_k) and (Δx_k), where the state space representation is:

$$\Delta x_{k+1}^m = A_k^m \Delta x_k^m + B_k^m \Delta u_k^m \tag{12}$$

$A_k^m$and $B_k^m$are derived from linearization:

$$\Delta \dot{x}^m \approx \Delta x^m \frac{\partial f}{\partial x}|(x^m, u^m) + \Delta u^m \frac{\partial f}{\partial x}|(x^m, u^m) \tag{13}$$

Therefore, for the $(m+1)_{th}$ trajectory, it follow the form as

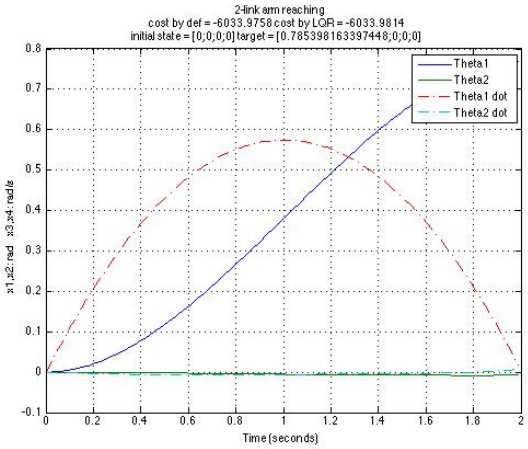
$$x_k^{m+1} = x_k^m + \Delta x_k^m \tag{14}$$

With several iterations, the cost function will converge and achieve the minimum value.[4]

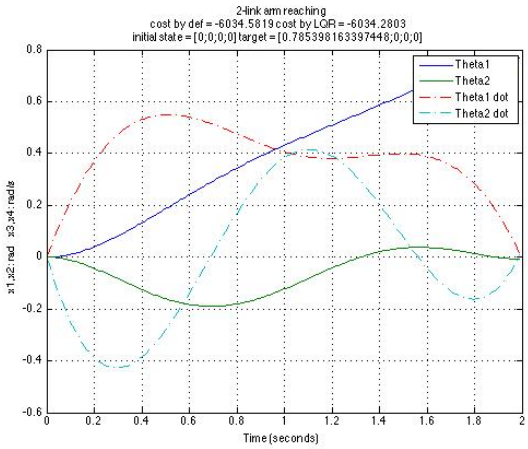
Simulation Results

The parameters for 2-link follow as

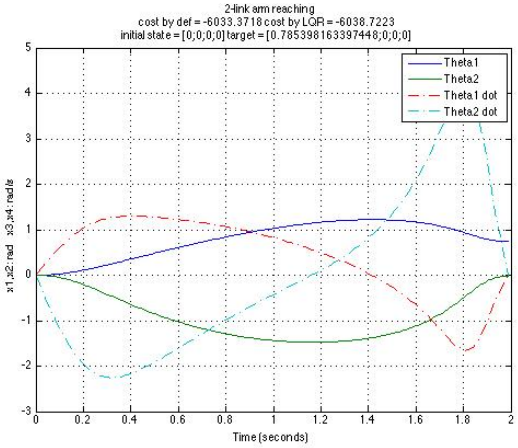
$$\begin{aligned} m_1 = 1, m_2 = 1 \\ l_1 = 1, l_2 = 1.5 \\ c_1 = 0.5, c_2 = 0.75 \end{aligned}$$



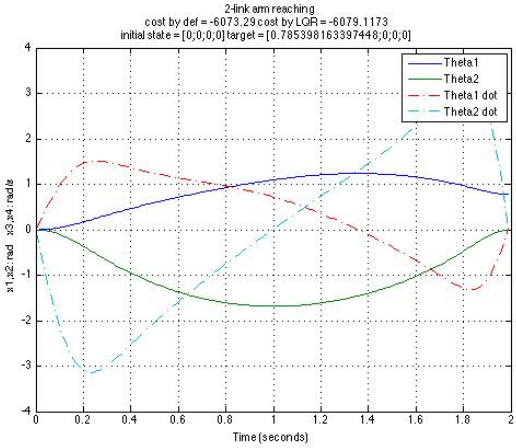
(a) First Iteration



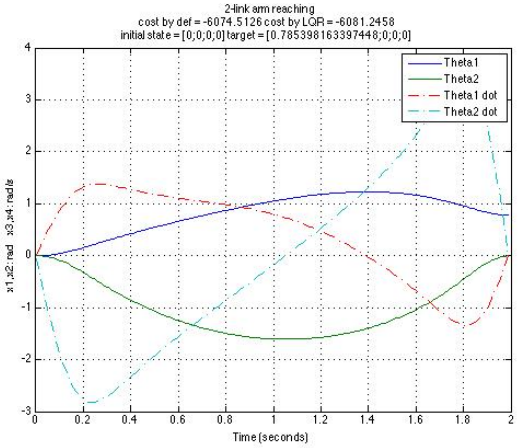
(b) Second Iteration



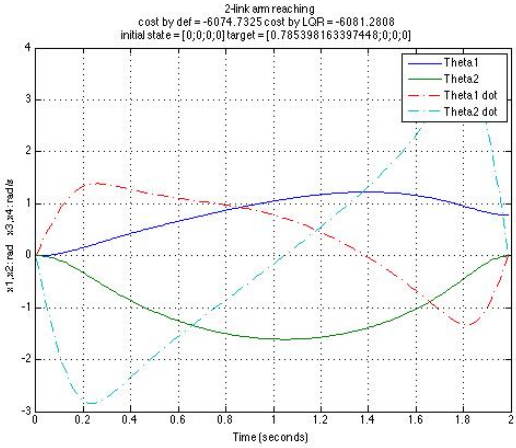
(c) Third Iteration



(d) Fouth Iteration



(e) Fifth Iteration



(f) Sixth Iteration

Figure 2: Simulation for 6 iterations

The results shows good convergence after 5 iterations. Actually as long as the target state is not far away from the initial state, the convergence will happen within 6 iterations. However, if the target state is far away, this linearized model does not hold any more.

References

[1] Textbook Spongrobot 2005

[2] General LQR Problem in Google Drive

[3] Matlab code $LQR - 2links - arm$ in Google Drive

[4] Matlab code $LQR - 2 - link - robot - arm - reaching$ in Google Drive