

Application of Iterative LQR on a Mobile Robot With Simple Dynamics

A. Aaqaoui (aaqaoui.ayo@gmail.com), MS Student

Y. M. Elsheikh M. (yoelsheikh@gmail.com), MS Student

Chair of Electromobility, TU Kaiserslautern

Abstract—The aim in this paper is to apply the iLQR, iterative Linear Quadratic Regulator, to control the movement of a mobile robot following an already defined trajectory. This control strategy has proven its utility for nonlinear systems. As follows up, this work intends to concertize this statement and to evaluate the extent to which the performance is comparatively improved against the ordinary, non-iterative LQR. The method is applied to a differential robot with non-holonomic constraints. The mathematical equations, resulting description and the implementation of this method are explicitly explained, and the simulation studies are conducted in the Matlab and Simulink environment.

Index Terms—iLQR, LQR, Nonlinear Systems, Nonholonomic Constraints, Differential Robot.

I. INTRODUCTION

A. Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) control is a modern state-space technique for designing optimal dynamic regulators. It refers to a linear system and a quadratic performance index according to

$$\dot{x}(t) = Ax(t) + Bx(t). \quad (1)$$

$$x(0) = x_0 \quad (2)$$

It enables a trade-off between the regulation performance and the control effort via the a performance index when the initial state

$$x_0 \quad (3)$$

is given. LQR assumes that the system is linear and hence expressed by

$$\dot{x} = Ax + Bu. \quad (4)$$

Secondly, it assumes that the cost function, i.e. the performance index, is of the form

$$J = \int_0^\infty [(x - x^*)^T Q (x - x^*) + u^T R u] dt. \quad (5)$$

where,

$$Q = Q^T \geq 0 \quad (6)$$

and

$$R = R^T \geq 0 \quad (7)$$

x^* is the target state; Q and R , denote the state and input weighing matrices, respectively. [1] [4]

B. Why iLQR

Even in cases of non-linearity the LQR can provide optimal control, but its limitation is that the calculation of this optimal control law is conducted considering the local set of parameters without consideration of the generated changes in the future states. From this perspective the strength of the iterative linear quadratic regulator iLQR can be noticed.

The iLQR is an extension of the LQR control with idea of providing an optimal control input sequence considering the whole control sequence rather than only the control point at the current time-step. So this control strategy allows us to estimate an overall optimal sequence taking into account the changing dynamic of the system. The basics of this method can be described via the following algorithm: [2]

- 1) Initialization with: initial state x_0 and initial control input $U = [u_0, \dots, u_1]$
- 2) Forward Pass: applying the control sequence U starting from x_0 to get the trajectory from the state space x .
- 3) Backward Pass: in this step the value function and the dynamics are evaluated for each (x, u) of the state space.
- 4) Update control sequence calculation and U' and evaluating the trajectory cost from (x_0, U') then proceed a decision making of adopting U' based on $d = |cost(U) - cost(U')|$ and a threshold value—:
 - If $d < threshold$; convergence then we exit.
 - If $cost(U) < cost(U')$; we set $U = U'$, and change the update size to be more aggressive, then go to step 2.
 - If $cost(U) \geq cost(U')$; change the update size to be more modest, then go to step 3.

As for linear dynamics, in case of non-linearity is described analogously with the following equation and the non linearity is manifested via the function f :

$$x_{k+1} = f(x_k, u_k) \quad (8)$$

The equation describes the evolution of the vector state $x \in \mathbb{R}^n$ between times i and $i + 1$ given the control input $u \in \mathbb{R}^m$ at time i . And the resulting trajectory is denoted as $\{X, U\}$, where $\{X\}$ is the sequence of state $X = x_1, x_2, \dots, x_N$ and $\{U\}$ is the corresponding control inputs $U = u_1, u_2, \dots, u_{N-1}$. The underlying idea is that the generated input sequence is updated in every time-step for the to-go path. [2]

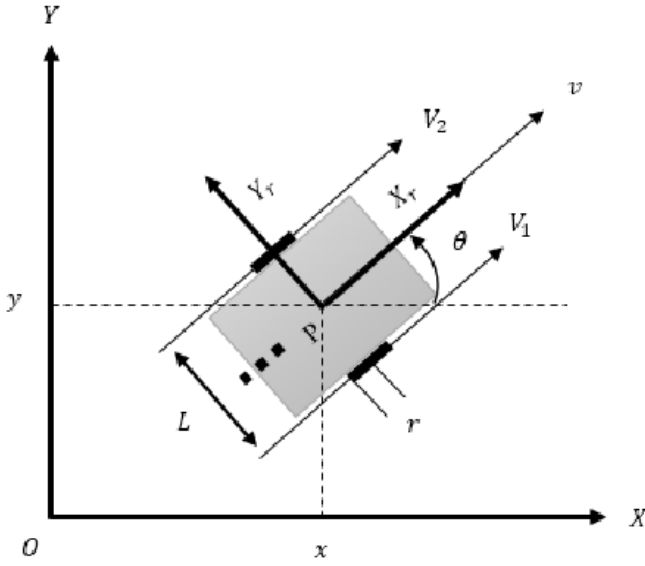


Fig. 1. Differential Mobile Robot Parameters, courtesy of Yacine Ahmine's paper [6].

C. Mobile Robot

The aim in this paper is to develop a trajectory tracking controller that follows the a Linear Quadratic Regulator in essence and simulate its performance while concurrently, developing an iLQR for the same trajectory tracking problem. The two controllers are to then be compared as per the global trajectory tracking for a wheeled mobile robot. The chosen mobile robot is a differential robot with a castor set of wheels.

The movement of the robot in the configuration space with respect to a generated global trajectory is denoted as nominal state sequence $S_{ff} = s_1, s_2, \dots, s_N$ is controlled via its coordinates (x, y) in the world frame and its orientation angle θ as illustrated in fig1.

From S_{ff} the nominal control sequence is derived by computing the needed linear velocity and angular velocity needed to reach state s_k from the preceding state s_{k-1} as shown in (x):

$$u_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}}{\frac{dt}{\omega_k - \omega_{k-1}}} \\ \frac{dt}{\omega_k - \omega_{k-1}} \end{bmatrix} \quad (9)$$

The nominal control sequence is represented by the following vector:

$$u_{ff} = u_1, u_2, \dots, u_N$$

where its components u_k represent the nominal control from state s_k to state s_{k+1} . [3]

D. Approach

The basic approach to a path tracking problem, considering a robot with non-holonomic constraints, is that the vehicle detects the environment through its sensors and then positions itself accordingly, setting the posture. After introducing the kinematics of the robot, the mathematical formulation is presented and the path tracking problem is then put forth in

detail, along with the control strategy. In accordance to the defined path, two control sequences are then produced and applied on the mobile robot.

II. MATHEMATICAL BACKGROUND

A. Cost Function and Value Function

The total cost function of a nonlinear discrete system is written in quadratic form as:

$$J_0 = \frac{1}{2}(x_N - x^*)^T Q_f (x_N - x^*) + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \quad (10)$$

It can be alternatively expressed as in equation (11), where l represents the immediate cost at each state in the trajectory and l_f the final cost:

$$J(x_0, U) = \sum_{k=0}^{N-1} l(x_k, u_k) + l_f(x_N), \quad (11)$$

the x^* denotes the target state and x^N the final state of the states sequence, Q_f and Q represent the state cost-weighting matrices which are symmetric positive semi-definite, and R the control cost-weighting matrix which is positive definite.

In iLQR as indicated by its name the algorithm for finding the optimal trajectory is iterative. The aim is to compute an improved control input sequence by iterative simulation, from an initial state and a nominal control until convergence to an optimal trajectory. The cost function will indicate how the produced trajectory is deviated from the targeted trajectory by summing the resulting deviation at each states points. From linearization process, the locale deviation at state x_k satisfy the following state equation:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k, \quad (12)$$

So the optimization formulation in iLQR can be written as follows:

$$\min_{u(t)} \frac{1}{2}(x_N - x^*)^T Q_f (x_N - x^*) + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \quad (13)$$

such that: $\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k$

In order to ease the evaluation of "the minimum", we define, from a specific state x_t (or at time t) in the trajectory, the cost-to-go as the sum of costs from time t to the last state x_N with the corresponding input sequence $U_t = u_t, u_{t+1}, \dots, u_{N-1}$:

$$J_t(x, U_t) = \sum_{k=t}^{N-1} l(x_k, u_k) + l_f(x_N), \quad (14)$$

The optimal cost-to-go from this given state is denoted as the value function V at time t , and Thus expressed as follows:

$$\begin{aligned} V_t(x) &= \min_{U_t} J_t(x, U_t); \quad t < N \\ V(x_N) &= l_f(x_N); \quad t = N \end{aligned} \quad (15)$$

as the $V(x_N)$ can be determined independently of the control input, comes the idea to evaluate the value function in a sequence starting backward from target state and proceeding recursively to the preceding state and so on. This second formulation of the value function shows this possibility, and it is written as a function of the immediate cost $l(x, u)$ and the value function in the next time-step:

$$V_t(x) = \min_u [l(x, u) + V(f(x, u))] \quad (16)$$

[5]

then the argument of the minimum in last equation is introduced in terms of perturbations in order to find, at each pair (x, u) , the supplement control input δu that would minimize the perturbations expressed as follows:

$$\begin{aligned} Q(\delta x, \delta u) &= l(x + \delta x, u + \delta u) \\ &\quad - l(x, u) + V(f(x + \delta x, u + \delta u)) \\ &\quad - V(f(x, u)), \end{aligned} \quad (17)$$

via Taylor's expansion it can be approximated under this form

$$Q(\delta x, \delta u) = \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Qu^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} \quad (18)$$

[5]

where the expansion coefficients are :

$$\begin{aligned} Q_x &= l_x + f_x^T V'_x \\ Q_u &= l_u + f_u^T V'_x \\ Q_{xx} &= l_{xx} + f_x^T V'_{xx} f_x + V'_x f_{xx} \\ Q_{uu} &= l_{uu} + f_u^T V'_{xx} f_u + V'_x f_{uu} \\ Q_{ux} &= l_{ux} + f_u^T V'_{xx} f_x + V'_x f_{ux} \end{aligned}$$

the control updating term is computed as follow :

$$\delta u^* = \min_{\delta u} Q(\delta x, \delta u) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x), \quad (19)$$

which can be divided into two terms; an open-loop term $k = -Q_{uu}^{-1}Q_u$ and a feedback gain term $K = -Q_{uu}^{-1}Q_{ux}$. Plugging the policy into the expansion of perturbations we can access to a quadratic model of the value at time i :

$$\Delta V(i) = -\frac{1}{2} Q_u Q_{uu}^{-1} Q_u \quad (20)$$

$$V_x(i) = Q_x - Q_u Q_{uu}^{-1} Q_{ux} \quad (21)$$

$$V_{xx}(i) = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux} \quad (22)$$

now we are able to start the process again from time step $i-1$. a recursive computing work is then conducted to derive at each point the local quadratic models of $V(i)$ and the control modifications $k(i)$, $K(i)$ which represents the output of the backward pass. After completion a new forward pass computes a new trajectory with new updated control sequence. [5]

$$\hat{x}(1) = x(1) \quad (23)$$

$$\hat{u}(i) = u(i) + k(i) + K(i)(\hat{x}(i) - x(i)) \quad (24)$$

$$\hat{x}(i+1) = f(\hat{x}(i), \hat{u}(i)) \quad (25)$$

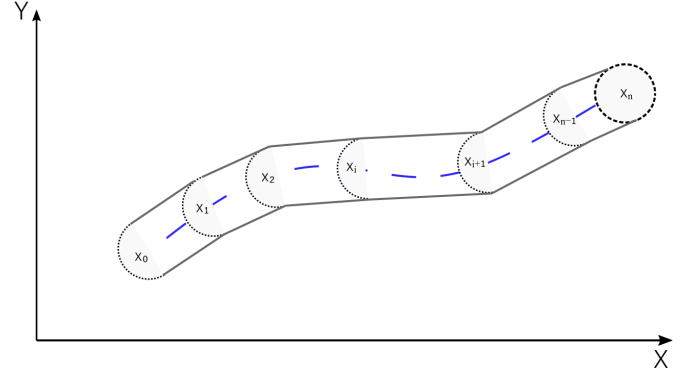


Fig. 2. Schematisation of the iLQR Working Principle.

B. Algorithms

First with LQR Approach:

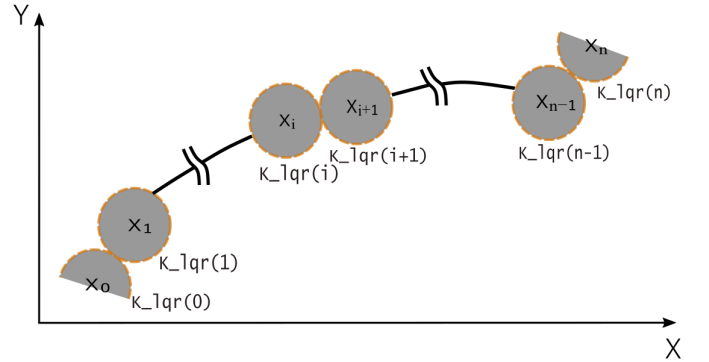


Fig. 3. Schematizing of the LQR approach.

- 1: set initial state and target state, x_0 and x_N .
- 2: set remaining states and compute corresponding control sequence: $(X^*, U^*) = ([x_1, \dots, x_N], [u_1^*, \dots, u_{N-1}^*])$.
- *Linearization & Updating control sequence:
- 3: **for** $i = 1; i < N, i++$ **do**
- 4: Compute Jacobians J_x, J_u at operating points $S_i = (x_i, u_i)$.
- 5: LQR formulation around $S_i: K_i = lqr(A_i, B_i, C_i, D_i, Q, R)$, with $A_i = J_x; B_i = J_u; C_i = I; D_i = 0$
- 6: compute control law : $\delta u = K_i \delta x$
- 7: updating control input at S_i : $u(i) = u_i^* + \delta u_i$
- 8: **end for**

Then with iLQR Approach :

- 1: evaluate initial input V_N and then $V_{N,x}, V_{N,xx}$
- 2: recall the computed values in forward pass of derivatives of local cost dynamic function at time step i : $l_x(i), l_u(i), l_{xu}(i), \dots$ and $f_x(i), f_u(i), f_{xu}(i), \dots$

- 3: deduction of different derivatives of value perturbation at time step i $Q_x(i), Q_u(i), Q_{xu}(i), \dots$
- 4: compute control input update; feed-forward term k and feedback term $K : ki, ki$
- 5: evaluate $V_{i,x}, V_{i,xx}$
- 6: set $i = i + 1$

III. EXAMPLE OF APPLICATION

A. Kinematic Model

the movement of the differential mobile robot can be described with simple kinematic model emanating from its principle of manoeuvring. Its movement in the configuration space is ensured only by the rotation of the wheels around their axis (no steering). To produce a linear movement the wheels have to rotate with same velocity and when they have different angular velocities the robot's body will rotate correspondingly to this difference. to quantify it's movement in any unknown environment (planar), we need the values of three parameters: (x, y) indicating the coordinate of the robot center of mass, and θ the orientation which the angle between the line perpendicular to the wheel axis and the X-axis as shown in the fig (1) these parameters are satisfying the following equations and thus providing the kinematic model of the differential robot.

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \omega\end{aligned}\quad (26)$$

For tracking, as shown in the equation, the robot is controlled by the input u , which is composed by the linear velocity v and angular velocity ω .

$$u = [v, \omega]^T$$

since we are dealing with a nonlinear system, the sensors information about the position will captured as they are required in the linearization process. As a result, the state vector x is including extra necessary states: $u = [x, y, \theta, v, \theta_{k+1}, \omega, \delta\omega]^T$. [3]

$$x_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ \omega_{k+1} \\ \delta v_{k+1} \\ \delta \omega_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k \cos \theta_k dt \\ y_k + v_k \sin \theta_k dt \\ \theta_k + \omega_k dt \\ u_k(0) \\ u_k(1) \\ u_k(0) - v_k \\ u_k(0) - \omega_k \end{bmatrix} = f(x_k, u_k) \quad (27)$$

IV. SIMULATION AND DISCUSSION

In matlab-simulink environment a path in form of inclined bell shape is constructed to conduct the simulation. it is a general very encountered form in real application, and a form with which steering performance can be checked. the two case scenarios are simulated: in the first scenario the ordinary LQR strategy is deployed for following the defined path, while in the second scenario the iLQR approach is considered. the following figures show the resulting path from both cases.

Both scenarios provide good tracking behavior with optimal error, with slight improvement with the case of iLQR. The resulting path using iLQR method is not fully reflecting its advantage because the optimization calculation are based on the discrete points represented by the operating points, whereas the calculation conducted in the LQR case is not quite similar since the controller is updating the control input within time between two operating points. A proper comparison would be by considering same rate of updating control, and this it would be taken into account in future work. Furthermore, in simulation we notice the impact of the initial control input quality and also the impact of the number of points constituting the path; the higher their number the more the tracking is stable and approaching the targeted path.

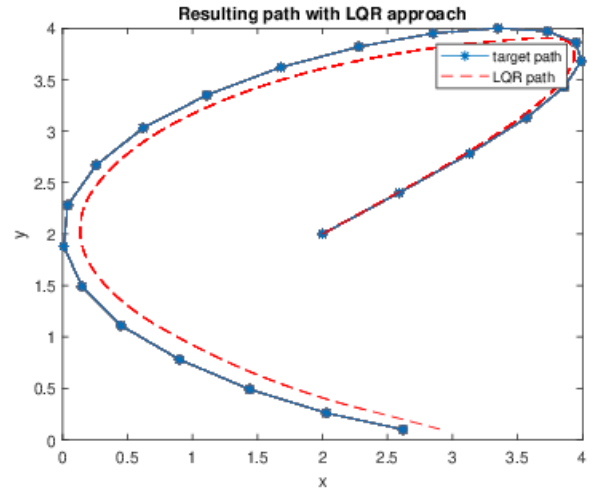


Fig. 4. Resulting Path with LQR approach.

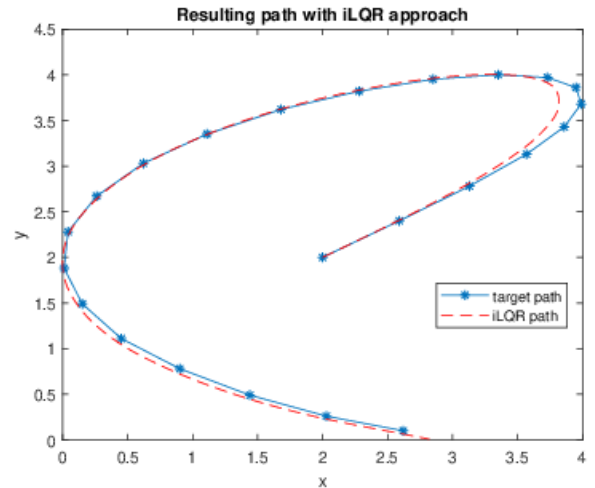


Fig. 5. Resulting Path with iLQR approach.

A. Conclusion

In this work we present the theoretical aspect behind the iLQR approach for trajectory tracking control in case of non linear dynamics and see how its concept encompasses the

concepts of LQR approach by considering the total cost. Works as [3] and [5] had shown experimentally the powerfulness of this approach and provide considerable insights for this work. In this paper a simulation was conducted by taking the dynamic of a differential mobile robot as an example of application. The results show that the approach gives stable solution with minimal errors.

REFERENCES

- [1] Ooi. Rich Chi, "Balancing a Two-Wheeled Autonomous Robot," B.Sc. thesis, Faculty of Engineering and Mathematical Sciences, University of Western Australia, Crawley, Western Australia, 2003.
- [2] Travis DeWolf, "THE ITERATIVE LINEAR QUADRATIC REGULATOR ALGORITHM,"[Online]. Available: <https://studywolf.wordpress.com/2016/02/03/the-iterative-linear-quadratic-regulator-method/> Accessed on: December, 2020.
- [3] Zhang, HJ., Gong, Jw., Jiang, Y., "An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot," in *J. Zhejiang Univ. - Sci. C 13*, 2012, pp. 593–600., doi: 10.1631/jzus.C1100379.
- [4] Travis DeWolf, "LINEAR-QUADRATIC REGULATION FOR NON-LINEAR SYSTEMS USING FINITE DIFFERENCES,"[Online]. Available: <https://studywolf.wordpress.com/2015/11/10/linear-quadratic-regulation-for-non-linear-systems-using-finite-differences/> Accessed on: December, 2020.
- [5] Y. Tassa, T. Erez and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, 2012, pp. 49064913., doi: 10.1109/IROS.2012.6386025.
- [6] A. Yacine, C. Fatima and B. Aissa, "Trajectory tracking control of a wheeled mobile robot using an ADALINE neural network," 2015 4th International Conference on Electrical Engineering (ICEE), Boumerdes, Algeria, 2015, pp. 1-5, doi: 10.1109/INTEE.2015.7416612.