# Space feedback control

## Linear Quadratic Regulator (LQR control)


Dr. Ing. Rodrigo Gonzalez

rodrigo.gonzalez@ingenieria.uncuyo.edu.ar

Control y Sistemas

Ingeniería Mecatrónica,
Facultad de Ingeniería,
Universidad Nacional de Cuyo


June 2020

# Linear Quadratic Regulator

An alternative method to pole placement is to place the poles so that the closed loop system optimizes a cost function:

$$J = \int_0^\infty (x^T Q_x x + u^T Q_u u)dt$$

Busca una función de costo que penalice las desviaciones de los estados del sistema y de los controles de accion sobre el sistema

# Linear Quadratic Regulator

An alternative method to pole placement is to place the poles so that the closed loop system optimizes a cost function:

$$J = \int_0^\infty (x^T Q_x x + u^T Q_u u) dt$$

Qx def pos
Qu semidef pos

where $x^T Q_x x$ is the ***state cost*** and $u^T Q_u u$ is the ***control cost***. The matrices $Q_x$ and $Q_u$ are symmetric, positive (semi-) definite matrices. This is called the ***linear quadratic regulator (LQR)*** problem.

definida positiva-autovalores>0
semidef pos -autovalores>=0

The solution to the LQR problem is given by

$$u = -Kx, \qquad K = Q_u^{-1} B^T S$$

La solución esta dada por encontrar K, es decir S

where $S$ is a positive definite, symmetric matrix given by

$$A^T S + SA - SBQ_u^{-1} B^T S + Q_x = 0$$

This equation is called the ***algebraic Riccati equation***.

# Linear Quadratic Regulator

The tuning of the LQR is to choose the weighting matrices $Q_x$ and $Q_u$. To guarantee that a solution exists, the system must be ***reachable*** and that $\boldsymbol{Q_x} \succcurlyeq \mathbf{0}$ and $\boldsymbol{Q_u} \succ \mathbf{0}$ .

semidef pos definida positiva

# Linear Quadratic Regulator

The tuning of the LQR is to choose the weighting matrices $Q_x$ and $Q_u$. To guarantee that a solution exists, the system must be ***reachable*** and that $\boldsymbol{Q_x \succcurlyeq 0}$ and $\boldsymbol{Q_u \succ 0}$ .

1. Simplest choice: $Q_x = I$ and $Q_u = \rho I$

$$J = \int_0^\infty (x^T x + \rho u^T u) dt \quad \Longrightarrow \quad \text{trade-off} \Longrightarrow \|x\|^2 \;\; vs \;\; \rho\|u\|^2$$

This reduce the tuning to select $\rho$, which then becomes a trade-off between state cost and control cost.

Para rho grande penalizamos más acciones de control(más lentas o de menor valor) y si rho<1 penalizamos más acciones sobre los estados.

# Linear Quadratic Regulator

The tuning of the LQR is to choose the weighting matrices $Q_x$ and $Q_u$. To guarantee that a solution exists, the system must be ***reachable*** and that $\boldsymbol{Q_x} \succcurlyeq \mathbf{0}$ and $\boldsymbol{Q_u} \succ \mathbf{0}$.

1. Simplest choice: $Q_x = I$ and $Q_u = \rho I$

$$J = \int_0^\infty (x^T x + \rho u^T u) dt \quad \Longrightarrow \quad \text{trade-off} \Longrightarrow \|x\|^2 \;\; vs \;\; \rho\|u\|^2$$

This reduce the tuning to select $\rho$, which then becomes a trade-off between state cost and control cost.

Otro método es penalizar los estados que se ven reflejados en la salida del sistema

2. Output weighting. Let $z = C_z x$ be the output you want to keep small.

Choose $Q_x = C_z^T C_z$, and $Q_u = \rho I$. $\quad \Longrightarrow \quad$ trade-off $\Longrightarrow \|z\|^2 \;\; vs \;\; \rho\|u\|^2$

salida vs acciones de control

# Linear Quadratic Regulator

3. Diagnonal weighting.

$$Q_x = \begin{bmatrix} q_1 & & 0 \\ & \ddots & \\ 0 & & q_n \end{bmatrix} \quad Q_u = \begin{bmatrix} \rho_1 & & 0 \\ & \ddots & \\ 0 & & \rho_p \end{bmatrix}$$

Choose the individual diagonal elements based on how much each state or input signal should contribute to the overall cost.

Elegimos valores individuales para los parámetros en la diagonal de estas matrices respetando características mencionadas antes

# Linear Quadratic Regulator

3. Diagnonal weighting. 3er método

$$Q_x = \begin{bmatrix} q_1 & & 0 \\ & \ddots & \\ 0 & & q_n \end{bmatrix} \qquad Q_u = \begin{bmatrix} \rho_1 & & 0 \\ & \ddots & \\ 0 & & \rho_p \end{bmatrix}$$

Choose the individual diagonal elements based on how much each state or input signal should contribute to the overall cost.

Fijamos valores máx para estados y para salidas y alfa o beta distribuye proporcionalmente penalidades sobre cada estado y accion de control

Alternative, (***Bryson's rule***) choose the diagonal weights as $q_i = \alpha_i^2 / x_{i,max}^2$ and $\rho_i = \beta_i^2 / u_{i,max}^2$, where $x_{i,max}$ *and* $u_{i,max}$ represents the largest response. $\alpha$ and $\beta$ are used for additional individual weighting of the state and control cost,

$$\sum_{i=1}^{n} \alpha_i^2 = 1 \qquad \sum_{i=1}^{p} \beta_i^2 = 1$$

Limitamos valores máximos para acciones de control y cada estado

# Linear Quadratic Regulator

Qx semidef pos, Qu def pos

3. Diagnonal weighting.

$$Q_x = \begin{bmatrix} q_1 & & 0 \\ & \ddots & \\ 0 & & q_n \end{bmatrix} \quad Q_u = \begin{bmatrix} \rho_1 & & 0 \\ & \ddots & \\ 0 & & \rho_p \end{bmatrix}$$

Choose the individual diagonal elements based on how much each state or input signal should contribute to the overall cost.

Alternative, (***Bryson's rule***) choose the diagonal weights as $q_i = \alpha_i^2 / x_{i,max}^2$ and $\rho_i = \beta_i^2 / u_{i,max}^2$, where $x_{i,max}$ *and* $u_{i,max}$ represents the largest response. $\alpha$ and $\beta$ are used for additional individual weighting of the state and control cost,

$$\sum_{i=1}^{n} \alpha_i^2 = 1 \qquad \sum_{i=1}^{p} \beta_i^2 = 1$$

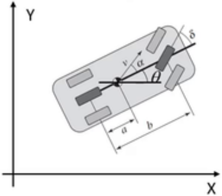
4. Trial and error

# Revisit Example - Vehicle steering (Ex 7.4)

Consider the following system:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & v_0\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}av_0/b\\ v_0/b\end{bmatrix}u$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + [0]\,u$$




Vehicle data: $v_0 = 12\ m/s$
$a = 2\ m$
$b = 4\ m$

# Revisit Example - Vehicle steering (Ex 7.4)

Consider the following system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 12 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 6 \\ 3 \end{bmatrix} u$$

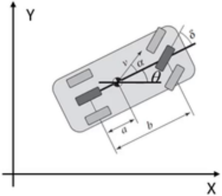
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\, u$$

Place the poles so that the closed loop system optimizes the cost function:

$$J = \int_0^\infty \left(x^T Q_x x + u^T Q_u u\right) dt$$

where

$$Q_x = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \qquad Q_u = \rho$$




Vehicle data: $v_0 = 12\, m/s$

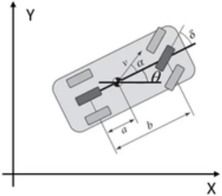
$a = 2\, m$

$b = 4\, m$

# Revisit Example - Vehicle steering (Ex 7.4)

For the case when

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$



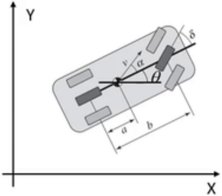
# Revisit Example - Vehicle steering (Ex 7.4)

For the case when

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$

The solution to the algebraic Ricatti equation is

$$A^T S + SA - SBQ_u^{-1}B^T S + Q_x = 0$$




In MATLAB: The LQR problem can be solved using the `lqr` command

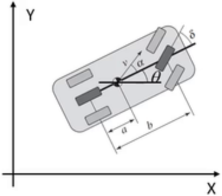
# Revisit Example - Vehicle steering (Ex 7.4)

For the case when

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$

The solution to the algebraic Ricatti equation is

$$S = \begin{bmatrix} 0.292 & 0.470 \\ 0.470 & 2.754 \end{bmatrix}$$




In MATLAB: The LQR problem can be solved using the `lqr` command

# Revisit Example - Vehicle steering (Ex 7.4)

For the case when

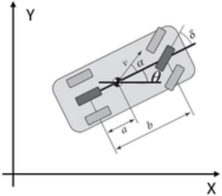
$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$

The solution to the algebraic Ricatti equation is

$$S = \begin{bmatrix} 0.292 & 0.470 \\ 0.470 & 2.754 \end{bmatrix}$$

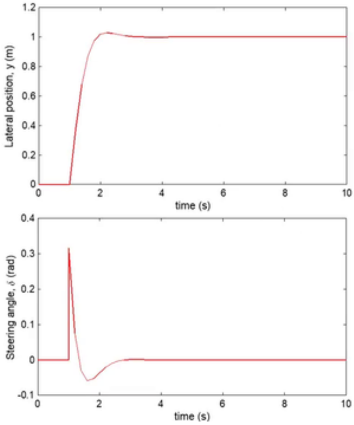
and the corresponding control law becomes

$$u = -Kx, \qquad K = Q_u^{-1} B^T S = [0.316 \quad 1.108]$$

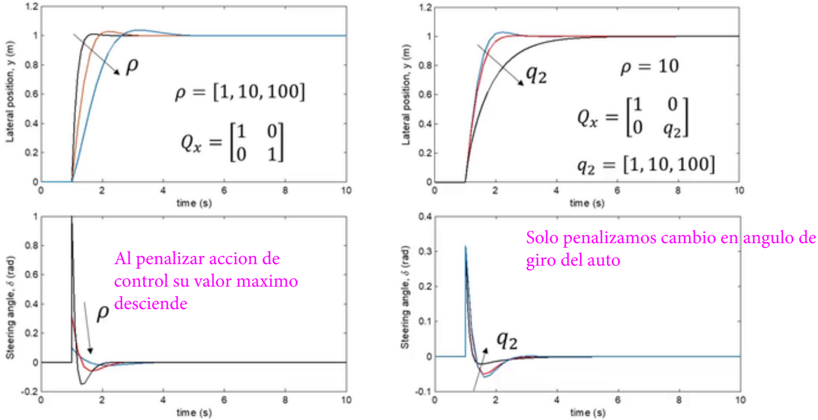



In MATLAB: The LQR problem can be solved using the `lqr` command

# Revisit Example - Vehicle steering (Ex 7.4)



# Revisit Example - Vehicle steering (Ex 7.4)



# Revisit Example - Vehicle steering (Ex 7.4)

For the case when

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$

The solution to the algebraic Ricatti equation is

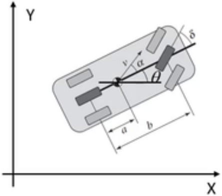
$$S = \begin{bmatrix} 0.292 & 0.470 \\ 0.470 & 2.754 \end{bmatrix}$$

and the corresponding control law becomes

$$u = -Kx, \qquad K = Q_u^{-1} B^T S = \begin{bmatrix} 0.316 & 1.108 \end{bmatrix}$$

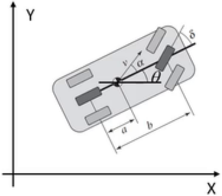
The closed loop system poles are

$$E = \begin{bmatrix} -2.6110 + 2.1371i \\ -2.6110 - 2.1371i \end{bmatrix}$$




In MATLAB: The LQR problem can be solved using the `lqr` command

# Revisit Example - Vehicle steering (Ex 7.4)




For the case when

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q_u = 10$$

The solution to the algebraic Ricatti equation is

$$S = \begin{bmatrix} 0.292 & 0.470 \\ 0.470 & 2.754 \end{bmatrix}$$

and the corresponding control law becomes

$$u = -Kx, \qquad K = Q_u^{-1} B^T S = \begin{bmatrix} 0.316 & 1.108 \end{bmatrix}$$

The closed loop system poles are

$$E = \begin{bmatrix} -2.6110 + 2.1371i \\ -2.6110 - 2.1371i \end{bmatrix}$$

Compared to the pole placement design, this corresponds to $\zeta = 0.77$ and $\omega_n = 3.44$.

In MATLAB: The LQR problem can be solved using the `lqr` command

# Bibliography


- Karl J. Astrom and Richard M. Murray *Feedback Systems*. Version v3.0i. Princeton University Press. September 2018. Chapter 7.