Identificación del modelo y control de un nuevo tipo de robot para operaciones quirúrjicas

Chacón Guillen, Flor Xiomara fchacong@unsa.edu.pe
Esquivel Yanque, Miguel Angel @unsa.edu.pe
Fernandez Flores, Yorlan Gonzalo @unsa.edu.pe
Mamani Saico, Alfredo amamanisai@unsa.edu.pe
Universidad Nacional de San Agustín
Ingeniería Electrónica
Sistemas de Control Avanzado 2021 B
Profesor Asesor: Juan C Cutipa Luque
@unsa.edu.pe

***Abstract*—Continuous robot technology is inspired by life forms in nature, it is physically very flexible and has a non-linear behavior. In addition, its dynamics are not unique, which makes its study a challenge. In this paper we are going to carry out a study on two control methods for the system of a continuous robotic manipulator, the control by sliding modes allows us to follow the trajectory of an estimated value, but it presents an error known as *Chattering*, which is solved from a *saturation* function. While the LQR control allows us to minimize the cost of a system from two intuitively chosen matrices and its control law is obtained by feedback.**

***Index Terms*—SMC, LQR, Continuous Robot, Control, Model**

***Abstract*—La tecnología de los robots continuos está inspirado en formas de vida en la naturaleza, físicamente es muy flexible y tienen un comportamineto no lineal. Además, su dinamica no es única, lo que hace que su estudio sea un desafío. En este trabajo se dispone a realizar un estudio sobre dos métodos de control para el sistema de un manipulador robotico continuo, el control por modos deslizantes nos permite seguir la trayectoria de un valor estimado, pero presenta un error conocido como *Chattering*, lo cual se soluciona a partir de una función *saturación*. Mientras que el control LQR nos permite minimizar el coste de un sistema a partir de dos matrices escogidos intuitivamente y su ley de control se obtiene por realimentación.**

***Index Terms*—SMC, LQR, Robot continuo, Control, Modelo**

I. Introducción

LAS enfermades gastrointestinales son enfermedades que afectan al esófago, estomago e intestinos son causadas generalmente por bacterias, virus, parásitos y algunos alimentos como leche y grasas, aunque a veces también son causadas por el exceso de medicamentos ingeridos. Mundialmente, las infecciones gastrointestinales son una de las causas más importantes de morbimortalidad entre los lactantes y los niños, sin embargo, las infecciones agudas del tracto gastrointestinal figuran entre las enfermedades infecciosas más frecuentes. [1]

Las diversas pruebas que se deben de realizar a este tipo de pacientes van desde Análisis ecográfico (ecografía) hasta la endoscopia y laparoscopia, las cuales se utilizan para visualización directa de la cavidad de los órganos. [2].Estas pruebas ayudan a localizar, diagnosticar y, en algunos casos, tratar el problema. Aunque las pruebas diagnósticas pueden ser muy útiles para determinar la presencia o ausencia de ciertos trastornos médicos, también pueden resultar caras y, con muy poca frecuencia, provocar hemorragia o lesión, sin embargo las molestias que causa son considerables al momento de uso.

Para reducir las molestias y el tiempo de preparación, el uso de robots continuos son una posible solución, ya que su forma y capacidad para navegar por entornos exigentes tales como canales o tuberías (esofago), los convierte en una opción viable para el diseño de asistentes robóticos orientados a la cirugía endoscópica transluminar. Basados en esto el presente estudio expone un nuevo modelo para asistente robótico que pueda ayudar a procedimientos quirurgicos como es la endoscopia. Este trabajo consiste en evaluar dos métodos de control para un manipulador robótico continuo. En la sección II, se analiza el modelado matemático de un robot continuo, seguidamente en la sección III, se diseñan dos métodos de control como SMC(Slidind Mode Control) y LQR(Linear Quadratic Regulator), en la sección IV se hacen pruebas para ambos métodos de control, en la sección V se evalúan los resultados, haciendo una comparación entre ambos métodos de control, y finalmente se desarrolan las conclusiones en la sección VI.

II. Modelo del Robot Continuo

En esta sección se estudia la ecuación diferencial que caracteriza el movimiento de un Robot Continuo(RC), se considera el analisis de diseño y modelo de cinemática [9].

A. Cinemática del sistema

El robot continuo puede ser considerado como elementos de curvatura constante de n arcos circulares [2] como muestra la figura 1. Cada arco circular representa una seccion, donde existe tres puntos de referencia(Frames) con tres ejes cada uno como se observa en la figura 2, donde los parámetros del arco están dados por:

- φ_i: ángulo respecto al plano del arco.
- θ_i: ángulo correcpondiente de la flexión de cada sección.
- l_i: longitud de arco de cada sección.
- r_i: radio de curvatura

Donde $i = 1, 2$

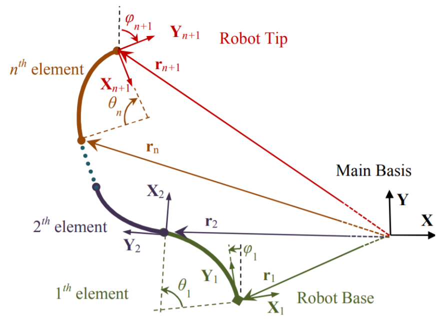

Fig. 1. Cinemática de Robot Continuo considerando n secciones [3].

Dado el espacio de configuración del robot continuo [6] se expresa como $q \in R^4$, donde $q = [\theta \; \dot{\theta} \; \varphi \; \dot{\varphi}]^T$.
Cada sección tiene una subsección con longitud s_i y angulo de flexión θ_{s_i}, que está dado como:

$$\theta_{s_i} = s_i \frac{\theta_i}{l_i} \tag{1}$$

B. Dinámica del sistema

La dinámica de un robot continuo se representa mediante la estructura de la Fig. 3, el cual está constituido de una columna primaria(CP), 3 columnas secundarios(CS) o actuadores unidos con una serie de discos espaciadores distribuidos uniformemente en cada sección. Para criterios de diseño los actuadores deben desplazarse libremente a través de los discos sin fricción para adaptarse a las fuerzas internas generadas producidas sobre los actuadores.

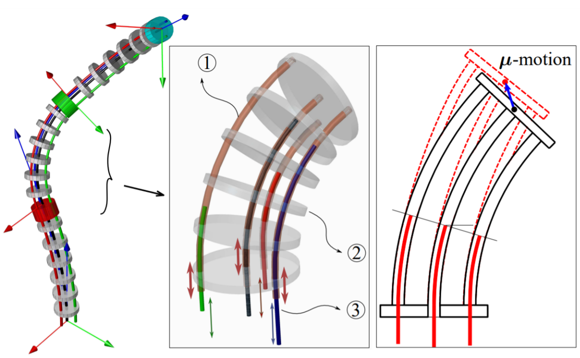

Fig. 2. Diseño de un robot continuo de 3 secciones [4] con tres actuadores secundarios.

Para controlar la dirección del robot se impuja y tira de la columna vertebral secundaria lo que provoca una deformación del robot en forma de arco.
La forma final de la ecuación de movimiento [6] se puede expresar como:

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) = \tau \tag{2}$$

Donde $M(q) \in R^{4*4}$ es la matriz de inercia,$V(q) \in R^{4*4}$ es la matriz del par centrifugo de Coriolis, $G(q) \in R^4$ es la matriz de pares giratorios y $\tau \in R^4$ los pares del robot continuo.

III. MÉTODO DE CONTROL

A. Control por modos deslizantes

La técnica de Control por Modos Deslizantes(SMC) se utiliza para en sistemas no lineales mediante el deslizamiento del sistema dinámico a lo largo de una superficie deslizante, donde la ley control tiene parametros k y λ. [6]

$$\ddot{q} = f + \tau \tag{3}$$

Donde, $f = -M^{-1}(V\dot{q} + G)$ es la dinamica del sistema sin incertidumbres.

Para el control de pocision del robot se define el error de espacio [6] $\tilde{q} \in R^4$, donde:

$$\tilde{q} = q - q_d \tag{4}$$

Donde, $q_d = [\theta_{(d)}\dot{\theta}_{(d)}\varphi_{(d)}\dot{\varphi}_{(d)}]^T$ son los ángulos de espacio de configuración deseada que el robot tratara de alcanzar.
La superficie deslizante del sistema de segundo orden es:

$$s = \dot{\tilde{q}} + \lambda\tilde{q} \tag{5}$$

$$\dot{s} = \ddot{q} - \ddot{q}_d + \lambda\dot{\tilde{q}} = f + \tau - \ddot{q}_d + \lambda\dot{\tilde{q}} \tag{6}$$

La ley de control estimada $\hat{\tau}$ se obtiene llevando a cero la superficie deslizante s [6], por lo tanto:

$$\hat{\tau} = \ddot{q}_d - \lambda\dot{\tilde{q}} - f \tag{7}$$

Finalmente la ley de control puede definirse como:

$$\tau = M(\hat{u} - ksgn(s)) \tag{8}$$

Donde $sgn(s)$ está definido como:

$$sgn(s) = \begin{cases} 1 & si \quad s > 0 \\ 0 & si \quad s = 0 \\ -1 & si \quad s < 0 \end{cases}$$

La ley de control expresado en (8), obtenida por modos deslizantes, presenta un error de *Chattering*. Para eliminar el error, se suele adaptar una capa límite cerca de la superficie deslizante sustituyentendo la función $sgn(s)$, por una función de conmutación $sat(s)$.
Donde $sat(s)$ está definido de la siguiente manera: [6]

$$sat(s) = \begin{cases} sgn(s) & si \quad |s| > \delta \\ \frac{s}{\delta} & si \quad |s| < \delta \end{cases}$$

Por lo tanto la variable de control por modos deslizantes por conmutación es:

$$\tau = M(\hat{u} - ksat(s)) \tag{9}$$

B. Control LQR

La forma final de la ecuación de movimiento se puede expresar como:

$$M(q)\ddot{q} = \tau - G(q) - V(q, \dot{q})\dot{q} \tag{10}$$

Luego realizamos la transformación de las siguientes variables:

$$x_1 = q$$

$$x_2 = \dot{q}$$

Por lo tanto,el sistema se puede expresar como sigue:

$$\dot{x_1} = x_2 \tag{11}$$

$$\dot{x_2} = -M^{-1}Gx_1 - M^{-1}Vx_2 + M^{-1}\tau \tag{12}$$

Odenamos las ecuaciones (11) y (12) en forma matricial:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}G & -M^{-1}V \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} \tau \tag{13}$$

Finalmente, podemos expresar el sistema a través de las ecuaciones (14) y (15).

$$\dot{x} = Ax + B\tau \tag{14}$$

$$y = Cx + Du \tag{15}$$

Donde, x, $\dot{x}$, τ, y, A, B, C y D son el vector de estados, derivada del vector de estados, la variable de control, vector de salida, matriz de estados, matriz de control, matriz de salida y matriz de transmisión directa.

Dado la representación del sistema en la ecuación (14) y (15), el control LQR consiste en reducir la diferencia de los estados y la entradas representado mediante la siguiente función:

$$J = \int_0^\infty (x^T Q x + \tau^T R \tau) dt \tag{16}$$

La ley de control LQR se obtiene por realimentación de estados:

$$\tau = -Gx \tag{17}$$

Donde, $G = R^{-1}B^T Px$. Ahora solo P es la variable desconocida, que se obtine resolviendo la Ecuanción Algebraica de Ricatti, mediante el comando *care* en Octave.

IV. Pruebas experimentales

A. Prueba con SMC

Las simulaciones se realizaron en el entorno de Python, que es un lenguaje de alto rendimiento con librerías enfocadas en la operación con matrices, ploteo de funciones y soluciones numéricas. El codigo es adaptado de [8]. El aporte al código fue diseñar la función sat(s) y plotear las leyes de control.

La figura 3 muestra los resultados de tracking para las condiciones iniciales $q_o = [\pi/6; 1; 0; 0]$ utilizando control por modos deslizantes y la función $sgn(s)$, se observa un elevado consumo de energia en el controlador y la presencia de chattering en las leyes de control.

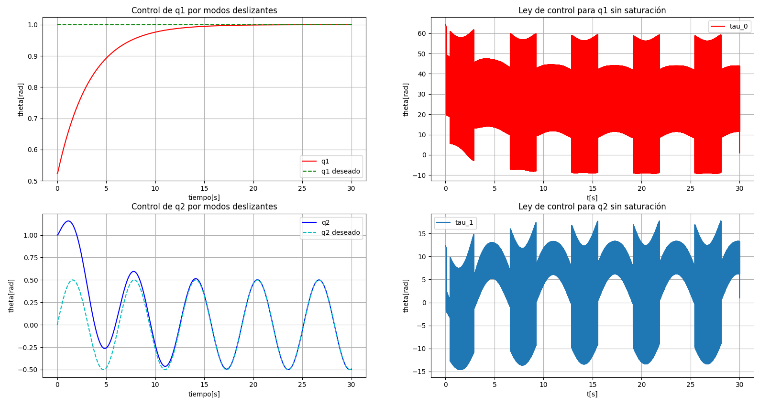

Fig. 3. Control por Modos Deslizantes

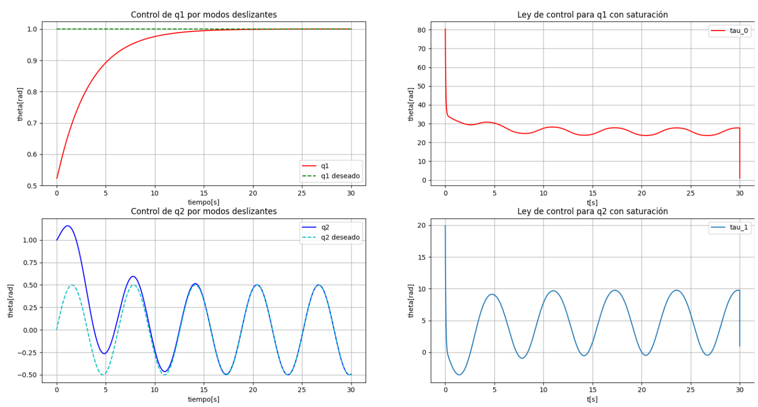

Fig. 4. Control por Modos Deslizantes por conmutación

La figura 4 muestra los resultados del control de modos deslizantes por conmutación o switching. La aproximación nos permitió eliminar el efecto de castañeteo suavizando la discontinuidad a una región limitada alrededor de la superficie deslizante y provocando menor consumo de energia del controlador.

B. Prueba con LQR

Las pruebas para el control LQR se desarrollo en Octave. Para la primera prueba (Fig. 7) hemos considerado $Q = 30(I)$,lo cual acelera el rendimiento del sistema reduciendo la estabilización de θ a 7 segundos aproximadamente. Si $Q = I$, θ demora hasta 30 segundos.

La segunda prueba consiste en modificar $R = 20(I)$, lo cual muestra que la velocidad angular tarda en estabilizarse 20 segundos, mientras que para $R = 0.01(I)$, $\dot{\theta}$ tarda 10 segundos.

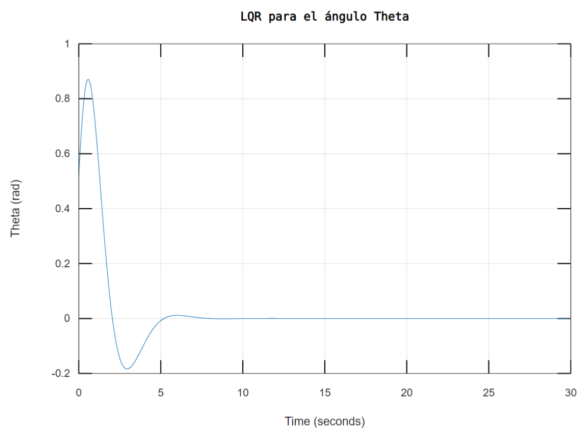

Fig. 5. LQR para θ, con $Q = 30(I)$

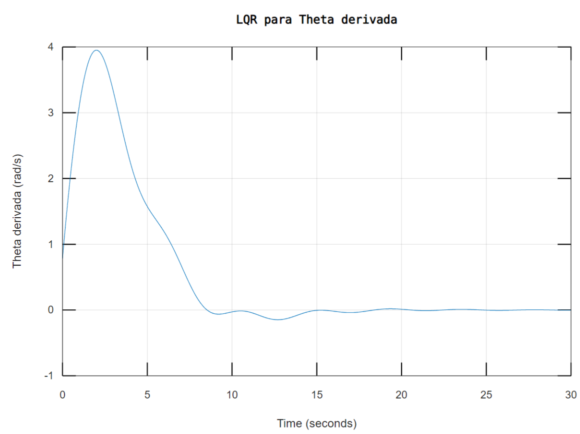

Fig. 6. LQR para $\dot{\theta}$, con $R = 20(I)$

V. Evaluación de resultados

A. Resultados para SMC

Los resultados de control por modos deslizantes se observa en la Figura 3 y 4. En la Fig. 3 se muestran los resultados para la ley de control (8), donde q_1, que representa a posición del ángulo θ en (rad) y q_2, que representa la velocidad angular de θ en (rad/s), consiguen el tracking a partir de los 17 segundos. Pero, el error de Chattering del sistema es demasiado alto. En la Fig. 4 se muentran los resultados para la ley de control (9), en donde se aprecia considerablemente la reducción del error de Chatering. Además, para ambas figuras podemos ver que la trayectoria deseada puede ser cualquier valor o función.

B. Resultados para LQR

Los resulatdos para el control LQR, se muestran en la figurra 5 y 6. Como se observa en la Fig. 5, la posición angular θ, alcanza el equilibrio en menos de 11 segundos, incluso tarda menos que el sistema controlado por modos deslizantes. Esto se consigue aumentando los valores de la matriz diagonal Q, es decir que podemos optimizar el rendimiento del sistema a partir de la matriz Q. En la Fig. 6 se puede apreciar que la velocidad angular de θ demora en estabilizarse casi 20 segundos, pero este tiempo puede ser reducido disminuyendo la matriz R. Significa que el sistema consume mayor energía a medida que R disminuye.

VI. Conclusiones

En este trabajo hemos desarrollado dos métodos de control para un manipulador robótico continuo dado su modelado descrito en la sección II. A partir de las pruebas experimentales y la evaluación de los resulados podemos comparar ambos métodos de control, las ventaja de una con respecto a la otra.

Con el controlador SMC, tenemos la ventaja de darle una trayectoria deseada a las variables de estado del sistema, es decir podemos dar un valor o una función diferente a cero, lo que no es posible con el método de control LQR, ya que este último minimiza la funcion de coste a cero. También el SMC garantiza un tracking perfecto. Sin embargo, hay discontinuidad a lo largo de la superficie s, lo que se conoce como efecto chattering, dependiendo de que tan lejos esté la trayectoria deseada menor será el rendimiento del control.

Con el controlador LQR, el robot continuo recorre desde la distancia mas corta desde su posición inicial hasta llegar a la referencia, lo cual implica más rapidez en la respuesta del sistema debido a que no hace trayectorias innecesarias, cuando se requiere que alcance a una referencia determinada. Por lo tanto su ventaja principal respecto al controlador SMC, es la versatilidad de elegir la matriz Q y R, estos nos premitirán priorizar el rendimiento o el consumo de energía del sistema, logrando incluso mayor rapidez que SMC.

References

[1] O. Organización Panamericana de la Salud, “La oms revela las principales causas de muerte y discapacidad en el mundo: 2000-2019,” 2020. [Online]. Available: https://www.paho.org/es/noticias/9-12-2020-oms-revela-principales-causas-muerte-discapacidad-mundo-2000-2019

[2] V. Mosquera, O. Vivas, and C. Rengifo, “Modelado y simulación de un robot para cirugía,” vol. 5, no. 10, pp. 45–50, 2012, revista de Ingeniería de Biomédicas. [Online]. Available: http://www.scielo.org.co/pdf/rinbi/v5n10/v5n10a06.pdf

[3] M. Dehghani and S. Mossavian, “Dynamics modeling of planar continuum robots by finite circular elements for motion control,” *Soft Robotic and Object Manipulation*, 2015.

[4] L. Wang, G. Del Giudice, and N. Simaan, “Simplified kinematics of continuum robot equilibrium modulation via moment coupling effects and model calibration,” 2019, department of Mechanical Engineering Vanderbilt University.

[5] O. Moussa, M. Mira, A. Fahmy, and E. Morgan, “Behavioral assessment of various control laws formulations for position tracking of multi-sectioning modeled continuum robots,” *Mechatronics and Automation*, 2019, 2019 IEEE 7th International Conference on Control,.

[6] W. Zhipeng, L. Qiang, X. Hong, and S. Runjie, “An analytic method for the kinematics and dynamics of a multiple-backbone continuum robot,” 2012, department of Control Science Engineering, Tongji University, Shanghai, China.

[7] Y. Yan, Q. Peng, A. Kaspar, and L. H.K., “Lagrangian dynamics and nonlinear control of a continuum manipulator,” 2015, iEEE Conference on Robotics and Biomimetics.

[8] S. H. Mark W. Spong and M. Vidyasagar, “Robot dynamics and control, nonlinear control simulator,” 2010. [Online]. Available: https://github.com/dohyeoklee/Non-linear-control-simulator/tree/main/src

[9] R. J. Webster III and B. A. Jones., “Design and kinematic modeling of constant curvature continuum robots: A review.” 2010, the International Journal of Robotics Research 29.13.

[10] H. Abdul Rashid, N. A. Mohamad, and M. Y. Abdul Halim, “Chattering-free sliding mode control for an active magnetic bearing system,” 2008, world Academy of Science, Engineering and Technology.

Apéndice

Código Control por modos deslizantes en Phyton

```python
import numpy as np
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt


class Control():

    @staticmethod
    def modos_deslizantes(dyn,X,d_set,_):

        Lambda = 0.3
        k = 2

        q_deseado = np.array([[d_set[0]],[d_set[1]]])
        q_deseado_dot = np.array([[d_set[2]],[d_set[3]]])
        q_deseado_ddot = np.array([[d_set[4]],[d_set[5]]])

        q = X[0:2,0]
        q_dot = X[2:4,0]

        q_tilda = q-q_deseado
        q_tilda_dot = q_dot-q_deseado_dot

        S = q_tilda_dot + Lambda*q_tilda
        f = -dyn.M_inv @ (dyn.V@q_dot + dyn.G)
        u_hat = q_deseado_ddot - Lambda*q_tilda_dot - f
        #print(dyn.G)
        #
        def sat(S):
            d = np.array([[0.1],[0.1]])
            if S[0,:] > d[0,:] and S[1,:] > d[1,:] == True:
                return np.sign(S)
            else:
                return S/d
        #

        #u = dyn.M @ (u_hat - k*np.sign(S))
        u = dyn.M @ (u_hat - k*sat(S))

        return u,None

class Cinematica(object):

    m_1 = 1.0              #masa del primer frame
    m_2 = 1.0              #masa del segundo frame
    L_c1 = 1.0             #longitud del primer frame al centro de masa del primer brazo
    L_c2 = 1.0             #longitud del segundo frame al centro de masa del primer brazo
    L_1 = 2*L_c1           #longitud del primer frame
```

```
    L_2 = 2*L_c2              #longitud del segundo frame
    I_1 = (m_1*L_1**2)/3      #momento de inercia del primer frame con respecto al origen
    I_2 = (m_2*L_2**2)/3      #momento de inercia del segundo frame con respecto al primer frame
    G = 9.8                   #constante de gravedad

    def theta_nominal(self):

        theta_1 = self.m_1*self.L_c1**2 + self.m_2*(self.L_1**2+self.L_c2**2) + self.I_1 + self.I_2
        theta_2 = self.L_1*self.L_c2
        theta_3 = self.m_2*self.L_c2**2 + self.I_2
        theta_4 = self.m_1*self.L_c1 + self.m_2*self.L_1
        theta_5 = self.m_2*self.L_c2
        return np.array([[theta_1],[theta_2],[theta_3],[theta_4],[theta_5]])

class Dinamica(Cinematica):

    # M =  matriz de inercia
    # M_inv = inversa de la matriz de inercia
    # V : matriz de par de Coriolis centr fugo
    # G : matriz sobre pares gravitacionales

    def __init__(self,X,dt):
        self.X = X
        self.dt =dt
        self.M = self.matriz_M()
        self.M_inv = self.matriz_M_inv()
        self.V = self.matriz_V()
        self.G = self.matriz_G()

    def matriz_M(self):

        q_2 = self.X[1,0]
        d_11 = self.m_1*self.L_c1**2 + self.m_2*(self.L_1**2+self.L_c2**2+2*self.L_1*self.L_c2*np.cos(q_2))+self.I_1+self.I_2
        d_12 = self.m_2*(self.L_c2**2+self.L_1*self.L_c2*np.cos(q_2))+self.I_2
        d_22 = self.m_2*self.L_c2**2+self.I_2
        return np.array([[d_11,d_12],[d_12,d_22]])

    def matriz_M_inv(self):

        q_2 = self.X[1,0]
        d_11 = self.m_1*self.L_c1**2 + self.m_2*(self.L_1**2+self.L_c2**2+2*self.L_1*self.L_c2*np.cos(q_2))+self.I_1+self.I_2
        d_12 = self.m_2*(self.L_c2**2+self.L_1*self.L_c2*np.cos(q_2))+self.I_2
        d_22 = self.m_2*self.L_c2**2+self.I_2
        det = d_11*d_22-d_12**2
        inv_11 = d_22/det
        inv_12 = -d_12/det
        inv_22 = d_11/det
        return np.array([[inv_11,inv_12],[inv_12,inv_22]])
```

```
    def matriz_V(self):

        q_2 = self.X[1,0]
        q_dot_1 = self.X[2,0]
        q_dot_2 = self.X[3,0]
        h = -self.m_2*self.L_1*self.L_c2*np.sin(q_2)
        c_11 = h*q_dot_2
        c_12 = h*q_dot_2 + h*q_dot_1
        c_21 = -h*q_dot_1
        return np.array([[c_11,c_12],[c_21,0]])

    def matriz_G(self):

        q_1 = self.X[0,0]
        q_2 = self.X[1,0]
        g_1 = (self.m_1*self.L_c1+self.m_2*self.L_1)*self.G*np.cos(q_1) + self.m_2*self.L_c2*self.G
        g_2 = self.m_2*self.L_c2*self.G*np.cos(q_1+q_2)
        return np.array([[g_1],[g_2]])

    def dyn_update(self,tau):

        dt = self.dt
        q_ddot = self.M_inv @ (tau - self.V@np.array([[self.X[2,0]],[self.X[3,0]]]) - self.G)
        return self.X + np.mat([[self.X[2,0]*dt + (q_ddot[0,0]*dt**2)/2],\
            [self.X[3,0]*dt + (q_ddot[1,0]*dt**2)/2],[q_ddot[0,0]*dt],[q_ddot[1,0]*dt]])


if __name__ == '__main__':

    f = 1e3
    t_max = 30
    t_lista = np.linspace(0,t_max,int(t_max*f))
    t = t_lista
    dt = 1/f

    X = np.mat([[np.pi/6],[1],[0],[0]])

    q1_lista = [X[0,0]]
    q2_lista = [X[1,0]]

    q1_deseado = [1 for _ in t]
    q2_deseado = [0.5*np.sin(t_i) for t_i in t]
    q1_deseado_dot = [0 for _ in t]
    q2_deseado_dot = [0.5*np.cos(t_i) for t_i in t]
    q1_deseado_ddot = [0 for _ in t]
    q2_deseado_ddot = [-0.5*np.sin(t_i) for t_i in t]
    q_deseado_set = np.array([d_set for d_set in zip(q1_deseado,q2_deseado,q1_deseado_dot,q2_deseado_dot,q1_deseado_ddot,q2_deseado_ddot)])

    def normalizar_angulo(theta):
        return (((theta+np.pi) % (2*np.pi)) - np.pi)
```

```
        q1_deseado_set = [normalizar_angulo(q1_d) for q1_d in q_deseado_set[:,0]]
        q2_deseado_set = [normalizar_angulo(q2_d) for q2_d in q_deseado_set[:,1]]

        q1_e_lista = [q_deseado_set[:,0][0] -X[0,0]]
        q2_e_lista = [q_deseado_set[:,1][0] -X[1,0]]

        control = Control()
        Cinematica = Cinematica()
        theta = Cinematica.theta_nominal()
        #print(theta)

        def plotear():


            plt.figure(1)
            plt.subplot(221)
            line1,=plt.plot(t,q1_lista,'r-')
            line1_d,=plt.plot(t,q1_deseado,'g--')
            plt.legend(handles=(line1,line1_d),labels=("q1","q1 deseado"))
            plt.title("Control de q1 por modos deslizantes")
            plt.xlabel("tiempo[s]")
            plt.ylabel("theta[rad]")
            plt.grid()

            plt.subplot(223)
            line2,=plt.plot(t,q2_lista,'b-')
            line2_d,=plt.plot(t,q2_deseado,'c--')
            plt.legend(handles=(line2,line2_d),labels=("q2","q2 deseado"))
            plt.title("Control de q2 por modos deslizantes")
            plt.xlabel("tiempo[s]")
            plt.ylabel("theta[rad]")
            plt.grid()

            plt.subplot(222)
            plt.plot( t, data[:,1], 'r-', label = "tau_0")
            plt.legend()
            plt.title("Ley de control para q1 con saturaci n")
            plt.ylabel("theta[rad]")
            plt.xlabel("t[s]")
            plt.grid()

            plt.subplot(224)
            plt.plot( t, data[:,2], label = "tau_1" )
            plt.legend()
            plt.title("Ley de control para q2 con saturaci n")
            plt.ylabel("theta[rad]")
            plt.xlabel("t[s]")
            plt.grid()
            plt.show()


            tau0_matrix= np.ones(len(t_lista))
```

```
            tau1_matrix= np.ones(len(t_lista))


            for i in range(len(t_lista)-1):
            dyn = Dinamica(X, dt)
            #print(dyn)

            tau, theta = control.modos_deslizantes(dyn,X,q_deseado_set[i+1],theta
                )

            #print(tau)
            tau0_matrix[i]=tau0_matrix[i]*tau[0,:]
            tau1_matrix[i]=tau1_matrix[i]*tau[1,:]
            X = dyn.dyn_update(tau)

            q1_lista.append(normalizar_angulo(X[0,0]))
            q2_lista.append(normalizar_angulo(X[1,0]))

            q1_e_lista.append(normalizar_angulo(q_deseado_set[:,0][i+1]-X[0,0]))
            q2_e_lista.append(normalizar_angulo(q_deseado_set[:,1][i+1]-X[1,0]))

        #print(tau0_matrix)
        data = np.column_stack([t, tau0_matrix, tau1_matrix])
        np.savetxt("data.txt", data )
        plotear()
```

Código Control LQR en Ocatve

```
 clear all; close all;
M=[12.21145836 4.10572918;4.10572918 2.33333333];
V=[7.30883872e-02 7.31053723e-02; -1.69850758e-05 0.00000000e+00];
G=[25.68636509 0; 8.57172686 0];
m=inv(M);
 I=[0 0 1 0; 0 0 0 1];
A=[I;-(m)*G -(m)*V]
B=[0 0; 0 0; m]
C=[1 0 0 0];
D=0;

% Initial Conditions
x0 = [pi/6;  % 3 radians
      pi/4;
      1;
      3]; % 0 rad/s
% Control Law
Q = [1 0 0 0;  % Penalize angular error
     0 1 0 0;
     0 0 1 0;
     0 0 0 1]; % Penalize angular rate
R = [0.01 0; 0 0.01];     % Penalize thruster effort
K = lqr(A,B,Q,R);

% Closed loop system
sys = ss((A - B*K), B, C, D);

% Run response to initial condition
```

```
t = 0:0.005:30;
[y,t,x] = initial(sys, x0, t);

plot(t,x(:,1,1))
grid
title('LQR para el  ngulo  Theta')
ylabel('Theta (rad)')
xlabel('Time (seconds)')
pause

plot(t,x(:,2,1))
grid
title('LQR para Theta derivada')
ylabel('Theta derivada (rad/s)')
xlabel('Time (seconds)')
pause

plot(t,x(:,3,1))
grid
title('LQR para el  ngulo  Varphi')
ylabel('Varphi (rad)')
xlabel('Time (seconds)')
pause

plot(t,x(:,4,1))
grid
title('LQR para Varphi derivada')
ylabel('Varphi derivada (rad/s)')
xlabel('Time (seconds)')
pause
```