**Calibración de la Ecuación Diferencial.**

Para captar la idea del modelo de ecuaciones diferenciales, comencemos precisamente por saber como es el comportamiento de un motor de corriente continua, la aceleración angular evoluciona de acuerdo con:

En donde, es el voltaje de entrada, es la resistencia de devanado del rotor, es la inductancia del devanado del rotor, es la velocidad angular de rotación del rotor, es el momento de inercia del eje del rotor, es la posición angular y , son constantes de proporcionalidad. Para una mayor profundidad en estos temas ver

Para muchos motores, la inductancia se puede despreciar (). Este último hecho implica que la ecuación anterior se puede reescribir como:

Si reacomodamos términos,

…………………………….(\*)

Para resolver esta ecuación diferencial ordinaria obtenemos el polinomio asociado con la parte homogénea

Cuyas raíces son y , así, la solución de la parte homogénea es,

Para la solución particular proponemos , y , sustituyendo en la ecuación diferencial original,

Despejando, , con lo que,

Finalmente, tenemos que determinar los valores para y , obviamente necesitamos condiciones iniciales para determinarlas, sean y , entonces,

para ,

para

Para facilitar la elaboración de los algoritmos que se programarán, hagamos, el cambio de variable, y , de ésta manera obtenemos,

De donde,

De esta manera podemos obtener siempre los valores para y .

**Aproximación por mínimos cuadrados.**

Recordemos que originalmente los únicos datos que obtendremos son las velocidades registradas por el encoder del motor, es decir, las constantes y no estarán disponibles y es momento de desarrollar una técnica que nos ayude a obtenerlas, primero observemos que de manera experimental las lecturas del encoder son de la velocidad angular del motor y es la posición del rotor en radianes, por lo que para esta aplicación lo que necesitamos es , esto es, la velocidad angular, tomemos la ecuación (\*)

Si hacemos el cambio de variable y , tenemos que

Si somos observadores, es realmente la velocidad angular (por ser la derivada de ) y el encoder nos entrega precisamente , entonces si pudiéramos encontrar el valor de (la aceleración) podríamos tratar la ecuación (\*\*) como un problema de regresión lineal. El truco que usaremos será el de diferencias finitas para calcular la derivada numérica de .

Recapitulando, lo primero que debemos hacer es tomar una muestra de velocidades, se propone tomar 200 muestras en intervalos de tiempo de 0.01 segundos, es decir se debe de cubrir un total de 2 segundos de muestreo, después, calcular la derivada de esta muestra con diferencias centrales, lo que hará que se reduzca a 198 muestras pues con las diferencias centrales perdemos el primer y el último dato, pero ganamos precisión. Una vez que tenemos y podemos tratar el problema (\*\*) como un problema de mínimos cuadrados, la técnica de regresión lineal por mínimos cuadrados nos dará precisamente el valor de y que necesitábamos.

Cuando tenemos los valores de b y c, los podemos ocupar para sustituirlos en la derivada de la solución de la ecuación diferencial (\*)

Con . Recuerde que es la velocidad angular. Esta última ecuación es la que vamos a usar dentro de nuestro programa de Python.

Código en Python.

Es hora de poner todas las ideas juntas para hacer un solo script, pero antes necesitamos hacer algunas instalaciones en Python para que podamos usar bibliotecas ya establecidas, en este caso necesitamos instalar las bibliotecas para poder manipular funciones de álgebra lineal, en una terminal de la Raspberry escribimos:

>>pip3 install opencv-python

>>sudo apt-get install libcblas-dev

>>sudo apt-get install libhdf5-dev

>>sudo apt-get install libhdf5-serial-dev

>>sudo apt-get install libatlas-base-dev

>>sudo apt-get install libjasper-dev

>>sudo apt-get install libqtgui4

>>sudo apt-get install libqt4-test

También tenemos que instalar o verificar que tengamos instalado en Thonny las bibliotecas **numpy** y **sklearn**, de no tenerlas las podemos instalar directamente en el manager package.

Es importante recordar que este programa en Python se encuentra en el repositorio ----\_\_\_\_---. Y que deben estar conectados todos los dispositivos anteriores debido a que para calibrar la ecuación diferencial el motor debe estar controlado, dicho esto, en esta sección desglosamos un poco el desarrollo del script.

Ahora bien, lo primero que escribimos en el script de Python es lo que vamos a importar:

import numpy as np

from sklearn import linear\_model

Posteriormente notamos que la entrada para que se pida una calibración es cuando x=-2, así,

if x==-2:

i=0

p.ChangeDutyCycle(100) # Ponemos a velocidad máxima al motor

w=np.zeros(200) # Creamos el vector w con 200 elementos de sólo ceros

w2=np.zeros(198) # Igual que en w

# Esta sección es para tomar las 200 muestras de velocidad

while i<200:

w[i]=tach.RPM() # Aqui llenamos el vector w con las velocidades

if i>0 and i<199: # Hacemos una copia de w pero sin el primer y el último término

w2[i-1]=w[i]

i=i+1

sleep(0.01) # Tiempo de muestreo entre cada toma de velocidad

i=1 # Reiniciamos el contador i

dw=np.zeros(198) # Creamos ell vector dw (velocidad de w)

# El siguiente fragmento de código calcula la derivada central numérica para w

while i<199:

dw[i-1]=(w[i+1]-w[i-1])/0.02 # Diferencias centrales

i=i+1

p.ChangeDutyCycle(0) # Apaga el motor después de dos segundos a funcionamiento máximo

# Creo un modelo de regresión lineal

modelo = linear\_model.LinearRegression()

# Entreno el modelo con los datos (X,Y)

modelo.fit(w2.reshape(-1,1), dw)

print(modelo.coef\_[0])

print(modelo.intercept\_)

x=1

b=-modelo.coef\_[0]

A=np.array([[1,1],[0,-b]])

print(A)

caux=modelo.intercept\_

# Cálculo de los coeficientes para la ecuación diferencial

c=(caux)\*Vin/12

B=np.array([[k1],[k2-c/b]])

invA=np.linalg.inv(A) # Inversa de la matriz A

CC=np.dot(invA,B) # Producto de matrices (A^-1)B

# Recuerda que los indices comienzan en 0, así que CC[1] es CC[2]

# en matlab

dy=(c/b)-b\*CC[1]\*np.exp(-b\*t) # Solución de la ecuación diferencial con todas las constantes

t=t+0.1; # Tiempo actual

ourClient.publish("capstone/salon/virtual/RPMsim",str(dy[0])) # Publicación con MQTT de la solución a la ecuación diferencial.