ROBOT BLANDO

1st Martin Parada Rodriguez
Ingenieria Electronica
UNIVERSIDAD MANUELA BELTRAN
BOGOTA

***Abstract*—In the following document we will introduce the creation of a model of a soft robot, which has multiple degrees of freedom, which through the creation of a dataset we will obtain a neural network LSTM and based on it perform a trajectory planning which is the search for a succession of positions for a robot, which will take it from an initial state to a final one.**

En el siguiente documento nos introduciremos a la creación de un modelo de un robot blando , el cual tiene múltiples grados de libertad , el cual por medio de la creación de un dataset obtendremos una red neuronal LSTM y en base en ello realizar una planificación de trayectorias lo cual es la búsqueda de una sucesión de posiciones para un robot, que permitirán llevarlo desde un estado inicial a uno final.

***Index Terms*—algoritmo, pathfinding ,Desgaste**

I. Introduction

En esta guía de laboratorio nos introduciremos , en el uso de algebra lineal como una herramienta para el análisis de sistemas de rotación y traslación , luego usar las redes neuronales para que permita obtener diferentes datos por ejemplo los ángulos con respecto a las coordenadas en el plano xyz , estas redes neuronales están basadas en las redes LSTM lo cual son un tipo especial de redes recurrentes, por ultimo se asignara una secuencia de movimiento respectiva la cual debe usar se la planificación de trayectoria correspondiente.

II. Desarrollo

A. Modelamiento

Durante la practica se utiliza lo que se conoce como URDF o Formato de descripción de robótica unificada, este paquete contiene una serie de especificaciones XML para modelos de robots, sensores, escenas, etc. Cada especificación XML tiene un analizador correspondiente en uno o más idiomas. Se usa principalmente a en el mundo académico y la industria para modelar sistemas multicuerpo, como brazos manipuladores robóticos para la fabricación de líneas de montaje y robots animatrónicos para parques de atracciones. [1]

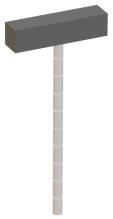

Fig. 1. Path planning

Para el laboratorio también fue necesario usar en Matlab algo conocido como Simscape Multibody lo cual proporciona un entorno de simulación multicuerpo para sistemas mecánicos 3D, como robots, suspensiones de vehículos, maquinaria de construcción y trenes de aterrizaje de aeronaves. Puede modelar sistemas multicuerpo utilizando bloques que representan cuerpos, articulaciones, restricciones, elementos de fuerza y sensores. Simscape Multibody formula y resuelve las ecuaciones de movimiento de todo el sistema mecánico. [1]

El estándar XML es una forma flexible de crear formatos de información y compartir electrónicamente datos estructurados a través de Internet pública, con lo que respecta a la practica XML se utiliza para describir todos los elementos fisicos del robot. [1]

Para dar inicio a la creación de los elementos físicos es necesario entender que es necesario colocar todo el código entre estas dos etiquetas marcando el inicio y el final, ademas con el nombre correspondiente de proyecto (”rob1”):

```
<robot name="rob1">.....</robot>
```

Del mismo modo con cada una de las etiquetas , por con siguiente se agrega un “link” lo que es un enlace físico con inercia, colisión y propiedades visuales. Un vínculo debe ser un elemento secundario de un modelo y puede existir cualquier número de vínculos en un modelo, por completar es necesario un “join” que es una articulación que es una junta conecta dos enlaces con propiedades cinemáticas y dinámicas . De forma predeterminada, la pose de una articulación se expresa en el marco del vínculo secundario. [2]

B. Orientacion y Coordenadas

El encontrar la rotación de la articulación es necesaria para encontrar la orientación del robot, este se basa en una

geometría euclidiana por :

$$e^{j\theta} = cos(\theta) + jsen(\theta) \tag{1}$$

Por medio de la ecuación encontramos que la rotación , basados en el teorema de screw , que es el movimiento de rotación sobre un eje con una traslación a lo largo de ese mismo eje se llama screw, por el parecido que tiene dicho movimiento con el realizado por un tornillo. Se comprende, que un twist está relacionado con esta geometría, de manera que un twist es la versión infinitesimal de un screw. [3]

$$R = I + sen(\theta)K + (1 - cos(\theta))K \tag{2}$$

Siendo K el plano donde rota el sistema y I la matriz de identidad :

$$K = \begin{bmatrix} 0 & -kz & ky \\ kz & 0 & -kx \\ -ky & kx & 0 \end{bmatrix} I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Otra forma de escribir la ecuación , usando la ecuacion 1 podemos obtener:

$$R = e^{\theta * \xi}.Ph(ubicación - inicial) \tag{4}$$

Para obtener la matriz twist es por medio de los valores de K y del punto P , donde P es la coordenada donde se encuentra la articulacion

$$\xi = \begin{bmatrix} K & -K.P' \\ 0 & 0 \end{bmatrix} \tag{5}$$

Durante la practica se realizan 10 links o enlaces físicos , el cual entre estos se encuentra una articulación para mover alrededor del eje Z ,en ese caso la matriz K (plano de rotación del sistema) son iguales en todas las articulación , pero el punto P cambia con respecto a cada articulación dependiendo del tamaño de los enlaces físicos. [3]

$$K = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Con ayuda de Matlab y Simulink se puede obtener datos más precisos para y además mejora la eficiencia de su obtención , por lo que el código se vería de la siguiente forma :

((anexo 1))

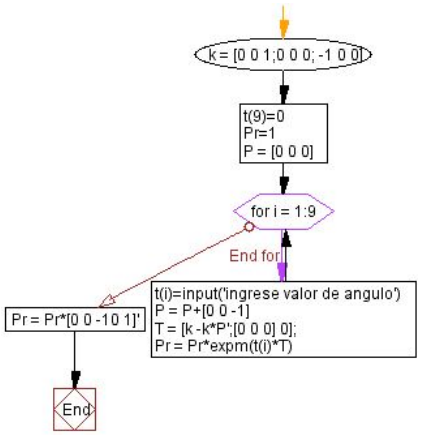

Fig. 2. Diagrama dse flujo de codigo

Debido a que son 9 articulaciones completas sin tomas por completo la base , obtenemos que la distancia máxima en el eje z es de -10.

C. Dataset

Un Dataset es un conjunto o colección de datos. Este conjunto normalmente se presenta en un patrón tabular. Cada columna describe una variable particular. Y cada fila corresponde a un miembro dado del conjunto de datos, según la pregunta dada. Esto es una parte de la gestión de datos . Los conjuntos de datos describen valores para cada variable para cantidades desconocidas como altura, peso, temperatura, volumen, etc. de un objeto o valores de números aleatorios. Los valores de este conjunto se conocen como dato . El conjunto de datos consta de datos de uno o más miembros correspondientes a cada fila. En este artículo, aprendamos la definición del conjunto de datos, los diferentes tipos de conjuntos de datos, las propiedades, etc. con muchos ejemplos resueltos. [4]

El conjunto de datos contiene información para crear nuestro modelo. Es una recopilación de datos estructurada como una tabla en filas y columnas. Existen diversos tipos de valores en un dataset como por ejemplo :

- **Las variables de entrada**: son las variables independientes en el modelo. También se denominan características o atributos. Las variables de entrada pueden ser continuas, binarias o categóricas. [4]
- **Las variables objetivo**: son las variables dependientes en el modelo. En los problemas de regresión , los objetivos son variables continuas (consumo de energía, calidad del producto...).En los problemas de clasificación , los objetivos son binarios (fallo, abandono...) o categóricos (el tipo de objeto, actividad...). En este tipo de aplicación, los objetivos también se denominan categorías o etiquetas. [4]
- **Las variables no utilizadas**: no son entradas ni objetivos. Podemos configurar una variable como No utilizada cuando no proporciona ninguna información al modelo (número de identificación, dirección, etc.). Las variables constantes son aquellas columnas en la matriz de datos, que siempre tienen el mismo valor. Deben establecerse como No utilizados ya que no proporcionan ninguna información al modelo pero aumentan su complejidad. [4]

Las muestras son las filas en la tabla de datos. También se les llama instancias o puntos.No es útil diseñar una red neuronal para memorizar un conjunto de datos simplemente. En cambio, queremos que la red neuronal funcione con precisión en los nuevos datos, es decir, que generalice. Para lograr eso, dividimos el conjunto de datos en diferentes subconjuntos:

- **Muestras de entrenamiento**:Durante el diseño de un modelo, generalmente necesitamos probar diferentes configuraciones. Por ejemplo, podemos construir varios modelos con diferentes arquitecturas y comparar su

rendimiento. Para construir todos estos modelos, usamos las muestras de entrenamiento.

- **Muestras de selección**:Las muestras de selección se utilizan para elegir la red neuronal con las mejores propiedades de generalización. De esta forma, construimos diferentes modelos con el subconjunto de entrenamiento y seleccionamos el que mejor funciona en el subconjunto seleccionado.
- **Muestras de prueba**:Las muestras de prueba se utilizan para validar el funcionamiento del modelo. Entrenamos diferentes modelos con las muestras de entrenamiento, seleccionamos el que funciona mejor en las muestras seleccionadas y probamos sus capacidades con muestras de prueba.
- **Muestras no utilizadas**:Algunas muestras pueden distorsionar el modelo en lugar de proporcionar información útil al modelo. Por ejemplo, los valores atípicos en los datos pueden hacer que la red neuronal funcione de manera ineficiente. Para solucionar estos problemas, podemos configurar esas muestras como No utilizadas.
  El estándar es usar el 60% de las muestras para entrenamiento, el 20% para selección y el 20% para prueba. La división de las muestras se puede realizar en orden secuencial o al azar. [4]

D. Redes neuronales LSTM

Los humanos no comienzan su pensamiento desde cero cada segundo. A medida que lee este ensayo, comprende cada palabra en función de su comprensión de las palabras anteriores. No tiras todo y empiezas a pensar desde cero otra vez. Tus pensamientos tienen persistencia [5].

Las redes neuronales tradicionales no pueden hacer esto y parece una gran deficiencia. Por ejemplo, imagine que desea clasificar qué tipo de evento está sucediendo en cada punto de una película. No está claro cómo una red neuronal tradicional podría usar su razonamiento sobre eventos anteriores en la película para informar los posteriores [5].

Las redes neuronales recurrentes abordan este problema. Son redes con bucles en ellas, lo que permite que la información persista. [5]

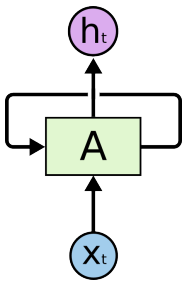

Fig. 3. Lstm1

En el diagrama anterior, un trozo de red neuronal,**UN**, mira alguna entrada **Xt** y genera un valor **ht**. Un bucle permite que la información pase de un paso de la red al siguiente.

Estos bucles hacen que las redes neuronales recurrentes parezcan algo misteriosas. Sin embargo, si piensa un poco más, resulta que no son tan diferentes de una red neuronal normal. Se puede pensar en una red neuronal recurrente como múltiples copias de la misma red, cada una de las cuales pasa un mensaje a un sucesor. Considere lo que sucede si desenrollamos el ciclo:

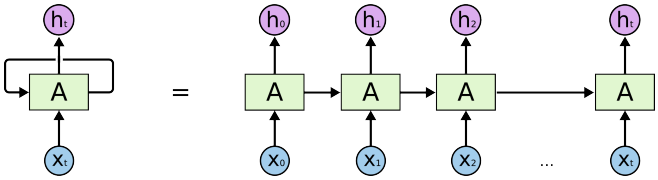

Fig. 4. Lstm1

Esta naturaleza de cadena revela que las redes neuronales recurrentes están íntimamente relacionadas con secuencias y listas. Son la arquitectura natural de la red neuronal que se utiliza para tales datos [5].

Los LSTM están diseñados explícitamente para evitar el problema de la dependencia a largo plazo. Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado, ¡no es algo que les cueste aprender!.

Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetidos de red neuronal. En RNN estándar, este módulo repetitivo tendrá una estructura muy simple, como una sola capa de tanh. [5]

La idea central detrás de los LSTM:La clave de los LSTM es el estado de la celda, la línea horizontal que atraviesa la parte superior del diagrama.
El estado de la celda es como una cinta transportadora. Se ejecuta directamente a lo largo de toda la cadena, con solo algunas interacciones lineales menores. Es muy fácil que la información fluya sin cambios. [5]

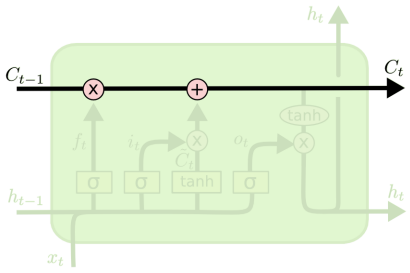

Fig. 5. Lstm3

El LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulado

por estructuras llamadas puertas.

Las puertas son una forma de dejar pasar información opcionalmente. Están compuestos por una capa de red neuronal sigmoidea y una operación de multiplicación puntual.

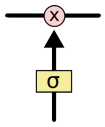

Fig. 6. Lstm1

E. Algoritmo de Path Planning

La planificación de trayectorias es un problema no determinista de tiempo polinómico ("NP") difícil con la tarea de encontrar un camino continuo que conecte un sistema desde una configuración inicial a una configuración final. La complejidad del problema aumenta con el incremento de los grados de libertad del sistema. El camino a seguir (el camino óptimo) se decidirá en función de una restricciones y condiciones, por ejemplo, considerando el camino más corto entre los puntos finales o el tiempo mínimo para recorrerlo sin colisiones. A veces, las restricciones y los objetivos se mezclados, por ejemplo, buscando minimizar el consumo de energía sin que el tiempo de viaje que el tiempo de viaje supere algún valor umbral. [6]

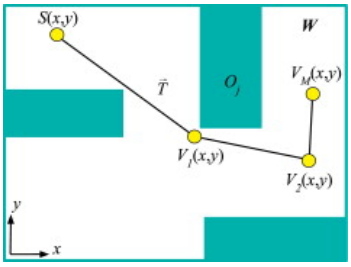

Fig. 7. Path planning

Los algoritmos de planificación de trayectorias se diferencian en función del conocimiento ambiental disponible conocimiento del entorno. A menudo, el entorno sólo es conocido parcialmente por el robot/AV. La planificación de la trayectoria puede ser local o global. La planificación global de trayectorias busca una trayectoria óptima dada la información ambiental más completa y se realiza mejor cuando el entorno es estático y perfectamente conocido por el robot. En este caso, el algoritmo de planificación de trayectorias produce una trayectoria completaruta completa desde el punto inicial hasta el punto final antes de que el robot comience a seguir la trayectoria planificada. [6]

F. Algoritmo Path Planning

El algoritmo de path planning es necesario conocer ciertas aspectos como por ejemplo una función que permita modelar el punto inicial al punto final, por lo que sale este polinomio de 3 orden:

$$F(t) = At^3 + Bt^2 + Ct + D \tag{7}$$

Es necesario encontrar los valores de A,B,C,D para obtener el polinomio completo que solo dependa del tiempo , cabe aclarar que por cada articulación es necesario realizar el calculo de estas 4 constantes de forma independiente.

$$F(0) = A(0)^3 + B(0)^2 + C(0) + D = D \rightarrow \theta_0 \tag{8}$$

El valor de la constante D es el valor inicial de la posición con lo que respecta al ángulo, para las demás constantes es necesario realizar derivadas , por ejemplo :

$$F'(t) = 3At^2 + 2Bt + C \tag{9}$$

$$F'(0) = C = 0 \tag{10}$$

Ya para los valores de A y B es necesario despejar ambas variables para obtener los valores se muestra de forma (t se refiere al tiempo que quiere completar el movimiento):

$$A = \frac{3}{t^2}(\theta_f - \theta_0) \tag{11}$$

$$B = \frac{-2}{t^3}(\theta_f - \theta_0) \tag{12}$$

Al obtener los valores de las constantes (A,B,C,D) se colocan le asignan al polinomio y obtendremos el movimiento de la articulación con respecto al tiempo.

III. DESCRIPCIÓN DE LA PRACTICA

Ya comprendiendo los diferentes temas a usar durante el desarrollo del proyecto y entendiendo la importancia del mismo , para comenzar con el desarrollo es necesario de la creación de un dataset que almacene por lo mínimo 300 datos sobre el movimiento del robot todo se obtiene por el código : (anexo 2)

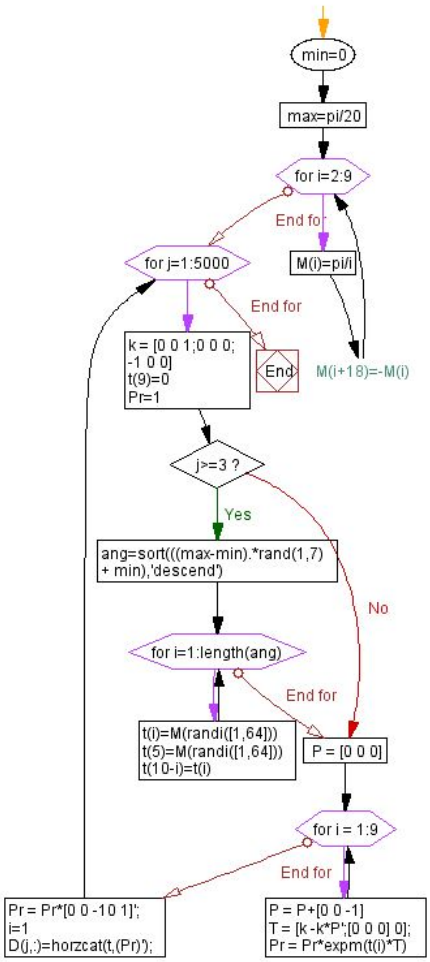

Fig. 8. diagrama de flujo de codigo para dataset

el cual debido a que se sigue una secuencia especifica el único cambio que se hacía en el código era el condicional con la intención de la formación de una U con las diferentes articulaciones, si el objeto se encontraba muy cercano al eje z , la formación del U era la mejor opción por lo que la primera y última articulación obtendrían el mismo valor y además serian el mayor ángulo con respecto a las demás articulaciones, con respecto a esto los siguientes ángulos serian de menor magnitud a excepción de la articulación 5 ya que este define que tan estrecho o abierto es la U.

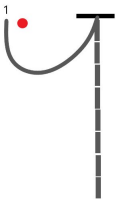

Fig. 9. tabla

En el caso de que el objeto estuviera lejos del eje z era necesario hacer la forma de una J o como si fuera anzuelo , lo único a cambiar seria el condicional que define la asignación de los ángulos ya no se colocaría el igual entre los ángulos , solo irían disminuyendo por cada articulación , pero las ultimas 3 o 2 articulaciones aumentaría drásticamente para dar ese efecto de J , en el código seria algo como : (anexo 2)

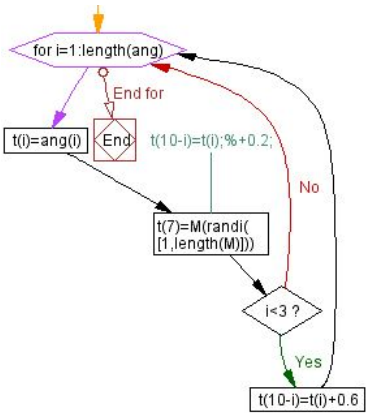

Fig. 10. condicional para tomar la forma de J

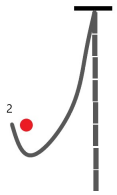

Fig. 11. tabla

todos estos valores se anexan en una tabla donde se colocan todos los 9 ángulos con su respectiva articulación y además de la posición que reflejan estos ángulos, de manera que se hacen 300 datos.

e1	e2	e3	e4	e5	e6	e7	e8	e9	x	Y	Z
0,26048577	0,16135891	0,14284894	0,12183141	-0,62831853	0,12183141	0,14284894	0,16135891	0,26048577	-3,53386588	0	-9,04756904
0,26070175	0,25086144	0,23217839	0,16759488	-1,04719755	0,16759488	0,23217839	0,25086144	0,26070175	-3,56420674	0	-8,72697117
0,20662927	0,14727384	0,14135346	0,05562739	-0,34906585	0,05562739	0,14135346	0,14727384	0,20662927	-3,59674399	0	-9,10135452
0,1911681	0,16010242	0,11658758	0,05689808	-0,31415927	0,05689808	0,11658758	0,16010242	0,1911681	-3,52271335	0	-9,14532998
0,10789607	0,04020706	0,02957949	0,02523748	0,34906585	0,02523748	0,02957949	0,04020706	0,10789607	-3,56187497	0	-8,9841427
0,20407803	0,11770973	0,11184132	0,04537877	-0,20943951	0,04537877	0,11184132	0,11770973	0,20407803	-3,58860945	0	-9,13586385
0,12890007	0,12638255	0,02769401	0,0017302	0,18479957	0,0017302	0,02769401	0,12638255	0,12890007	-3,59614153	0	-9,07975287
0,23491003	0,12791728	0,1173484	0,06749657	-0,34906585	0,06749657	0,1173484	0,12791728	0,23491003	-3,5729368	0	-9,12690537
0,20808116	0,14852883	0,09112935	0,07038958	-0,28559933	0,07038958	0,09112935	0,14852883	0,20808116	-3,5950093	0	-9,12422944
0,21698199	0,19938057	0,15339734	0,12153116	-0,62831853	0,12153116	0,15339734	0,19938057	0,21698199	-3,57113396	0	-9,01594879

Fig. 12. tabla

Ya teniendo la tabla completa es necesario entender las diferentes funciones de Matlab sobre la creación de red neuronal LSTM , para luego anexar los valores de la tabla anteriormente ya creada.

Para la creación de la red neuronal se hace con el siguiente código , cabe aclarar que toma anexar o tomar los valores del data set creado :

```
    % [X,Z] variables de entrada , posición
numFeatures = 2 ;
```

```
% numero de neuronas
numHiddenUnits = 3000;
numResponses = 9; %angulos

layers = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits,'OutputMode',
                    'last')
    fullyConnectedLayer(numResponses)
    regressionLayer];

maxEpochs = 800;
miniBatchSize = 50;

options = trainingOptions('adam', ...
    'ExecutionEnvironment','cpu', ...
    'MaxEpochs',maxEpochs, ...
    'MiniBatchSize',miniBatchSize, ...
    'GradientThreshold',1, ...
    'Verbose',false, ...
    'Plots','training-progress');

for i=1:length(Columna1)
    %posicion del obstaculo
    XTrain{i} = [X1(i);z1(i)];

end
%Ytrain toma los valores de los angulso
YTrain = [Columna1 Columna2 Columna3
    Columna4 Columna5 Columna6 Columna7
        Columna8 Columna9];

XTrain=XTrain';
%entrenamiento de la red
net = trainNetwork(XTrain,YTrain,
        layers,options);
%prueba de la red
XTest= [4.8468 ;-0.2244];

YPred = predict(net,XTest,'MiniBatchSize'
        ,miniBatchSize)
```

Toca Definir la cantidad de neuronas y maxEpochs lo cual entre mayor sean mejor serán los resultado obtenidos pero mayor es el peso computacional llegando a durar unos 20 minutos en terminar el proceso de entrenamiento y creación. Al iniciar el proceso se mostrara unos graficos el cual son RMSE y Loss , el cual entre menor sea el resultado en estos los valores que nos dará la red será mas cercanos al data set .

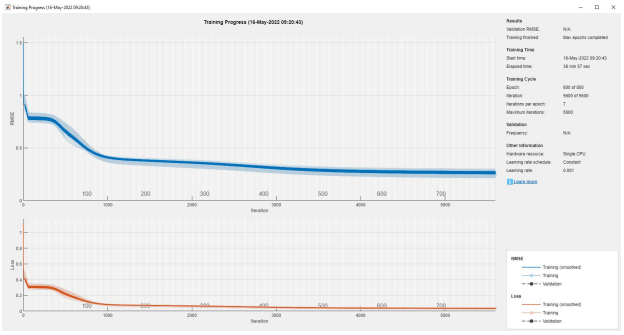

Fig. 13. Graficas

Ya teniendo la net entrenada y guardada , el siguiente paso y ultimo es realizar el pathplaninng el cual ya se explico con brevedad anteriormente .Para cargar la red guardad y comprobar el buen funcionamiento de esta se coloca la secuncia estipulada , de esta manera en un for de 61 posiciones que serian los puntos de una trayectoria para completar el recorrido de la secuencia , además de anexar el valor de las variables iniciales a usar (anexo 3)

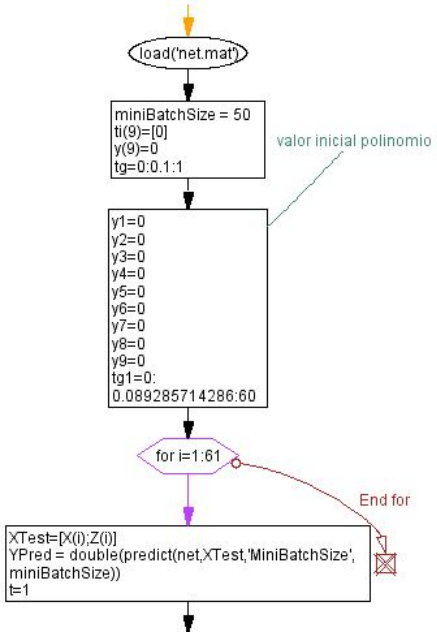

Fig. 14. pathplanning

Para dar inicio a la practica es necesario el crear una función para calcular las diferentes constantes de cada articulación:

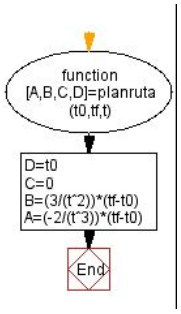

Fig. 15. funcion para obtencion de contstantes

Para luego aplicarlo en cada una de las articulaciones en este caso son 9 articulaciones , es necesario repetir la función 9 veces para obtener valores diferentes por cada una estas .

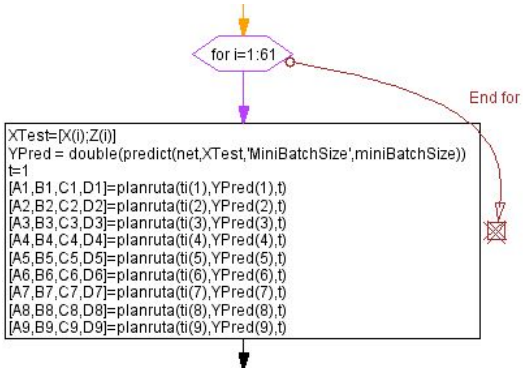

Fig. 16. pathplanning(anexo 3)

Al tener todas las contantes colocamos los valores de cada una de estas en el polinomio , dando como resultado 9 polinomios, el cual tiene un tg que es un tiempo de muestreo que toma cada 0.2 seg.

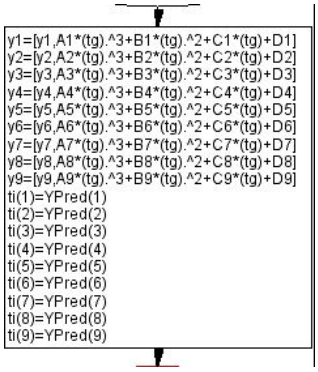

Fig. 17. pathplanning(anexo 3)

Después de tener los valores en forma de vector asignados a la variable Y (y1,y2...) es necesario agregar una matriz que contenga los valores del polinomio y del tiempo para luego ser subidos a simulink. Los valores finales de θ son los nuevos valores iniciales para el siguiente punto final .

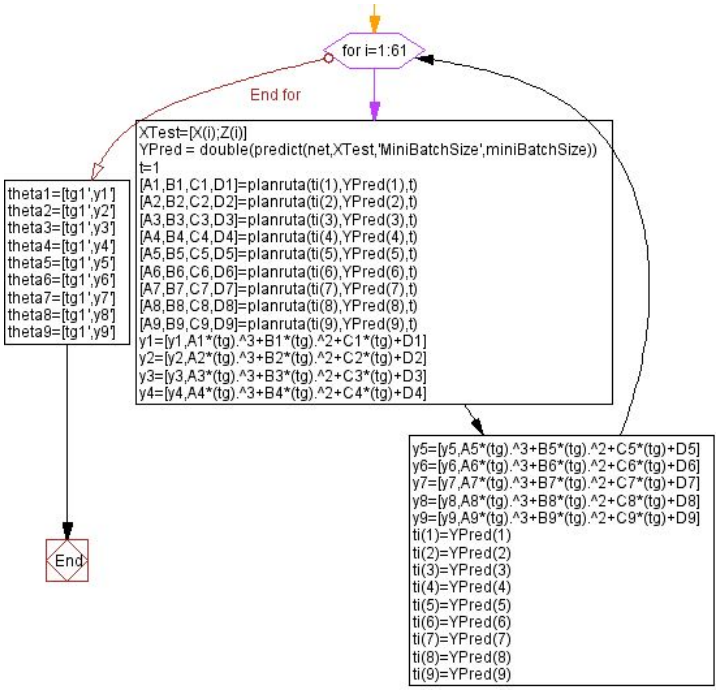

Fig. 18. pathplanning

Permitiendo que el robot realiza la secuencia de la siguiente forma :

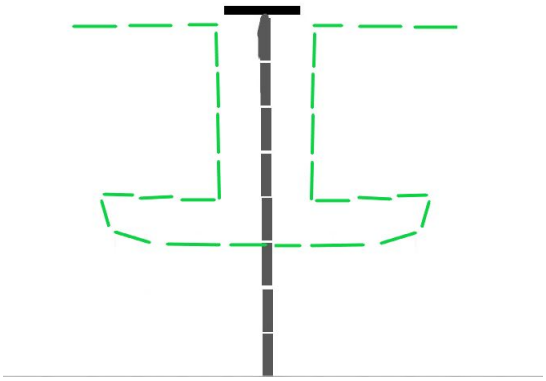

Fig. 19. secuncia

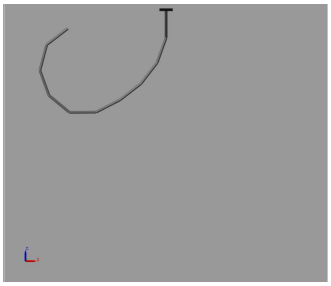

Fig. 20. simulacion

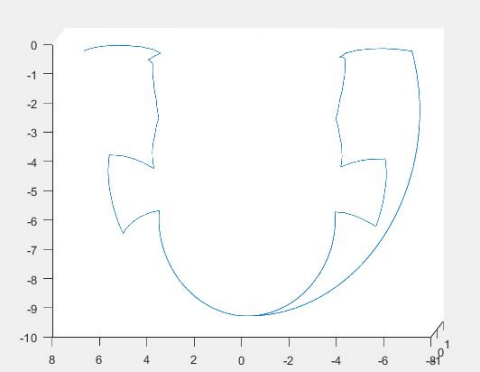

Fig. 21. grafico

IV. Conclusión

En conclusión el proyecto se cumplió , completando todos los objetivos propuestos durante el desarrollo de modo que se creo un dataset con los valores de la secuencia , además de la red neuronal que permitió el desarrollo del pathplanning obteniendo un resultado natural en el movimiento y además de realizar el movimiento de cada articulación en el transcurso del tiempo , los resultados se anexaran con el código y la simulación de simulink..

References

[1] Simscape Multibody. https://la.mathworks.com/products/simscape-multibody.html

[2] SDFormat, http://sdformat.org/spec?ver=1.9elem=link

[3] Adolfo Biosca Molina ,DESARROLLO DE UN SIMULADOR DE LA CINEMÁTICA DE LA PIERNA DE UN ROBOT HUMANOIDE , UNIVERSIDAD DE ALMERIAESCUELA SUPERIOR DE INGENIERÍAMÁSTER EN INFORMÁTICA INDUSTRIAL (2012/2013)

[4] Neural Designer , Tutorial de redes neuronales, https://www.neuraldesigner.com/company

[5] Graves, A Mohamed, G Hinton , Speech recognition with deep recurrent neural networks . 2013 IEEE international conference on acoustics

[6] Chen, B.; Quan, G. NP-Hard Problems of Learning from Examples. In Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Jinan, China, 8–20 October 2008; Volume 2, pp. 182–186. [CrossRef]