Desarrollo de un compilador y simulación para el control de un robot manipulador de dos brazos.

Fernando Leonel Aguirre, *IEEE Student Member*
Departamento de Ingeniería Electrónica
Universidad Tecnológica Nacional – FRBA
Buenos Aires, Argentina
aguirref@ieee.org

Tadeo Carlucci Rigoni
Departamento de Ingeniería Electrónica
Universidad Tecnológica Nacional – FRBA
Buenos Aires, Argentina
tadeo.carlucci@gmail.com

***Abstract*— El desarrollo de un compilador para el control de un robot de 4 articulaciones es descripto en el presente trabajo. El mismo genera código en MATLAB para la posterior simulación del robot, utilizando los resultados obtenidos en el estudio cinemático de la estructura.**

I. INTRODUCCIÓN

Para indicarle al procesador las funciones a realizar (que motores mover, en que momento, cuanto tiempo, con que ángulo, etc) es necesario escribir un código tal que el formato sea comprendido por el procesador (Código de máquina)

Ello es logrado mediante la utilización de un compilador. El mismo recibe las instrucciones por parte del usuario, en un pseudocódigo, de alto nivel, el cual es comprensible para el programador, y genera, en base a esos datos, código comprensible para el procesador. En este caso en particular, donde la estructura es simulada en MATLAB, el código generado por el compilador es código de MATLAB.

II. COMPILACIÓN

Para cumplimentar la función descripta en la sección anterior, el compilador deberá primero, verificar que el pseudocódigo definido por el usuario sea válido en base a reglas predefinidas para luego convertir el set de instrucciones de entrada en el correspondiente set de instrucciones/símbolos de salida. Podría entonces hablarse de:

- Análisis léxico: Es el punto en que se convierte el código fuente en una serie de tokens. Se reconocen las palabras reservadas del lenguaje y se evalúan los formatos de los tipos de datos y variables utilizadas en base a las reglas léxicas.
- Análisis sintáctico: Los tokens generados previamente son procesados siguiendo a las reglas sintácticas. Las mismas definen el ordenamiento de los tokens, lo cual se traduce en las instrucciones del lenguaje.
- Generación del código objeto (target): Una vez verificada la sintaxis, se convierte el set de instrucciones de entrada al set de instrucciones del código de salida.

El IDE que se utilizó para realizar el compilador fue ANTLRWorks 1.4.2. Este combina un editor de gramática con un intérprete, lo que permite fácilmente corregir errores de gramática y contiene tanto el parser como el lexer por lo que no es necesario el uso de otras herramientas. Al compilar el programa con el ANTLR y luego correrlo, este nos pide como entrada una serie de instrucciones, las cuales traducirá a código de Matlab para indicar al robot los movimientos que debe realizar.

III. CINEMÁTICA INVERSA

Para un funcionamiento práctico del robot, el sistema tiene que ser capaz de calcular el movimiento necesario de las articulaciones para llegar a un punto determinado. Este proceso es conocido como cinemática inversa, y permite hallar la posición de las 4 articulaciones a partir del punto x,y,z indicado por el operador. Existen para ello varios métodos, entre ellos el de la matriz homogénea y el denominado geométrico, sumamente útil en sistemas de pocas articulaciones. Dada su simplicidad, en este trabajo se utiliza este último para resolver el problema cinemático inverso que plantea el sistema.

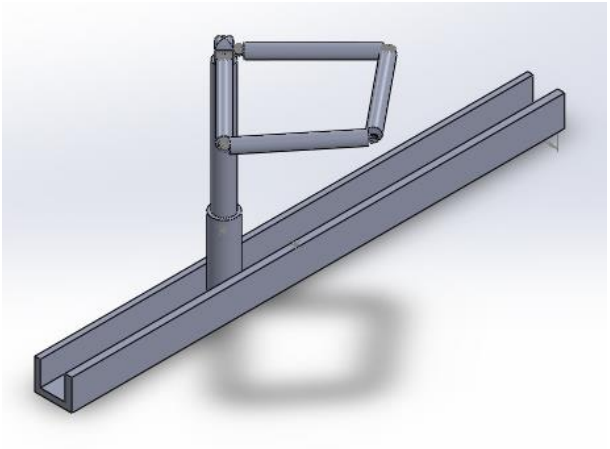

Fig. 1 Imagen orientativa para el cálculo de Q1 y Q2

De esta forma, es posible relacionar de manera directa la posición de las primeras dos articulaciones con x (posición sobre el riel longitudinal) y z (altura)

$$Q1 = x_o$$
$$Q2 = z_o$$

Para el análisis de la coordenada y será necesario un análisis trigonométrico teniendo en cuenta la relación entre la coordenada de trabajo y_o y la longitud de los brazos del robot. Luego de una serie de relaciones y cálculos se llega a

$$Q3 = \cos^{-1}\left(\frac{y_o}{long.del\ brazo}\right)$$
$$Q4 = 2\sin^{-1}\left(\frac{y_o}{long.del\ brazo}\right)$$

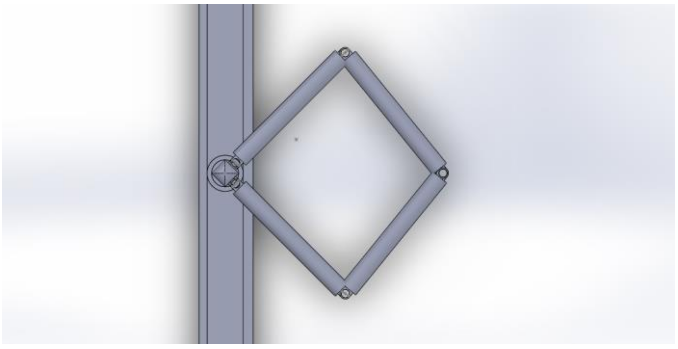

Fig. 2. Imagen orientativa para el cálculo de Q3 y Q4

IV. CÓDIGO DE PRUEBA

Para simular el funcionamiento, se implementa una rutina de ejecución que consiste en realizar una rutina al robot en la cual se desplaza entre cierta cantidad de puntos determinados por el usuario. Esta rutina es detallada por el usuario en forma de instrucciones, las cuales serán interpretadas por el compilador para generar el código que cúmplalo demandado.

En el momento de realizar el compilador, se determinó una serie de instrucciones que creímos acordes para la correcta implementación del robot en su labor. A continuación se detallarán brevemente:

- Init(): Inicializa los parámetros del robot y genera el código para el comienzo de la rutina.

- Move(x,y,z,t): Traslada el extremo del robot desde su posición actual hasta las coordenadas indicadas mediante el cálculo de la cinemática inversa. El campo t indica el tiempo que demora la operación.

- Moveq(q1,q2,q3,q4,t): Similar al move, solo que en este caso se basa en los parámetros de cada una de las articulaciones, solo es útil para casos muy específicos, como testeos, ya que siempre es conveniente obtener los parámetros de q a través de la cinemática inversa.

- Moverec(x,y,z,t):El diseño del robot fue elegido con el fin de que tenga la capacidad de realizar trabajos de remache a lo largo de una superficie plana, destinado al uso en astilleros y/o ensamblados de estructuras. Esta instrucción cumple con la necesidad de acercarse al punto de manera perpendicular a la superficie de trabajo para insertar de manera correcta el remache. Con el fin de aprovechar tiempos, primero realiza un acercamiento a la zona de trabajo, para finalmente realizar el último tramo de manera adecuada.

- Wait(t): Detiene al robot en la posición actual un tiempo determinado por el campo t.

- Home(): Lleva al robot a su posición inicial, la cual está definida en el compilador.

- End(): Simplemente indica el fin de la rutina, realiza las operaciones de cierre de archivos.

En este caso el código de prueba fue el siguiente

```
init
move( 5.0 , 3.0 , 4.0 , 10 )
wait(10)
moverec( 10.0 , 5.5 , 10.0 , 10 )
wait(10)
moveq( 8.0 , 9.0 , 1.4 , 0.3 , 10 )
wait(10)
home
end
```

Se puede ver en el Anexo I, el código resultante que implementa estas funciones.

IV. SIMULACIÓN

Para poder visualizar lo calculado por el código que genera el compilador se realizó un script de Matlab. Este consiste en la lectura de todos los valores de las articulaciones para evaluar cada una de las matrices homogéneas del robot, para así poder obtener la ubicación de los extremos de cada uno de las articulaciones que componen a nuestro robot. Para estos cálculos se utilizaron las matrices ya calculadas en el primer trabajo de análisis de cinemática directa del robot.

A continuación se mostrarán imágenes de algunos de los posicionamientos del robot generados por el script de Matlab.

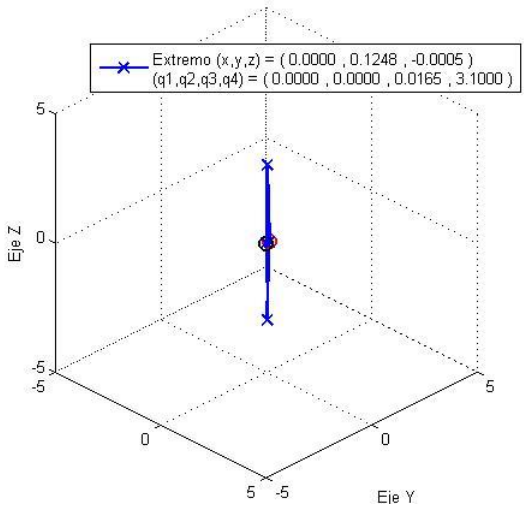

Fig. 3. Robot en el origen, con ambos brazos retraído, listo para comenzar una rutina

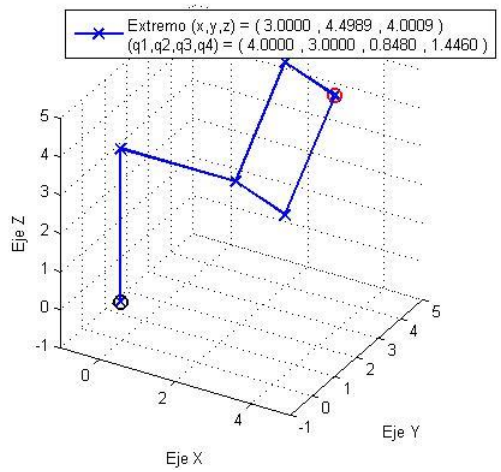

Fig. 4 Robot aproximándose a la zona de trabajo (x,y,z) = (3,5,4)

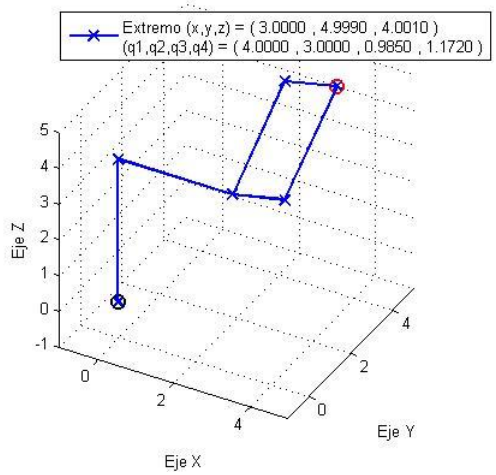

Fig. 5 Robot posicionado en el punto de trabajo(x,y,z) = (3,5,4)

Como se puede deducir, las figuras 3, 4 y 5 describen la trayectoria generada por la instrucción "movrec (3,5,4,t)", en la cual queda demostrada su principio de funcionamiento.

Tener en cuenta que para al aplicar el el algoritmo de Denavit – Hartenberg se tomó como eje Z al que conduce el riel del robot (horizontal), el eje X como la altura y el eje Y como al profunidad. Por lo tanto en los graficos estaremos visualizando el robot de manera "vertical".

En el Anexo II se puede ver el script de matlab que genera las trayectorias a partir de los valores que entrega el código generado por el compilador.

V. CONCLUCIONES

A lo largo de este trabajo adquirimos la experiencia en el desarrollo de compiladores. Cabe destacar que encontramos esta como una herramienta muy potente, ya que permite poner una capa de abstracción muy importante y permitiendo hacer las cosas más sencillas para el usuario. Por lo tanto lo tomamos como algo más para hacer más fácil la realización de los proyectos.

En el aspecto del robot, se destaca en este trabajo la comprensión final a la que se llega realizando los análisis de cinemática directa e inversa, con las cuales se pudo calcular cada uno de los puntos. Es también muy importante poder llegar a la simulación pretendida pudiendo así ver los resultados del trabajo realizado durante estos tres trabajos que abarcan la materia de Robótica.

Al final de este desarrollo uno cuenta con bastas herramientas para el desarrollo de robots. Abarcando áreas como el análisis dinámico, implementación en DSP, análisis cinemático, desarrollo en Solidworks, desarrollos en VHDL, desarrollo de compiladores y vasta experiencia en el uso de Matlab.

VI. REFERENCIAS

[1] *MATLAB, The MathWorks, Inc., Natick, Massachusetts, United States*

[2] Barrientos, Peñin, Balaguer y Aracil, "Fundamentos de Robótica", Mc Graw Hill, 2001

[3] Hernan Gianetta, Robótica, Topico: "Unidad Temática 01 – Cinemática del robot (2012)", Universidad Tecnológica Nacional – Facultad Regional Buenos Aires, Buenos Aires, 14 de Marzo, 2014.

[4] ANTLR, Terrence Parr, University of San Francisco.

[5] Hernan Gianetta, Robótica, Topico: "Unidad Temática 07 Compiladores y lenguaje de programación", Universidad Tecnológica Nacional – Facultad Regional Buenos Aires, Buenos Aires, 14 de Marzo, 2014.

Anexo I

```
%--------------------------------------------------------------------------
% INIT
%--------------------------------------------------------------------------
clear all;
clc;
syms q1 q2 q3 q4 q5 real
arm_lenght=3;
display('INIT()');
Eq = [ q1;
     cos(q3)*arm_lenght+sin(q4/2)*arm_lenght;
     q2];
q0=[0 0 1.55 0.033 0]; % Valores de q de la posición inicial
qa=q0;          % qa son los valores de q al finalizar cada instrucción
qmax = [20 15 degtorad(180) degtorad(180)]; % Restricciones
qmin = [0  0 degtorad(-125) degtorad(-70)];
q1=qa(1,1);q2=qa(1,2);q3=qa(1,3);
Pa= eval(Eq)';   % Punto en x,y,z

fd = fopen('outq.txt','w');

fprintf(fd,'q1      q2      q3      q4      q5\n');
fprintf(fd,'%2.4f   ',q0);
fprintf(fd,' ;\n ');


%--------------------------------------------------------------------------
% MOVE
%--------------------------------------------------------------------------
display('MOVE( 5.0 , 3.0 , 4.0 , 10 )');
Pxyz=[-0.30,0.6,0]'; % q=[0 2.0956 3.0707 0 0 ]
Pxyz=[5.0 , 3.0 , 4.0]';
px=Pxyz(1,1);
py=Pxyz(2,1);
pz=Pxyz(3,1);
%-----------------------------------------------
q1s = pz;
q2s = px;
if (py > 0.53) & (py <= 5.97)
    q3s =acos((py/2)/arm_lenght);
    q4s =2*asin((py/2)/arm_lenght);
end
if ( imag(q3s) ~= 0 ) | (imag(q4s) ~= 0 )
    display('Punto inalcanzable. Ubicado fuera del espacio de trabajo')
else
    qi=[q1s q2s q3s q4s 0];
    if ( qi(1,1:4) <= qmax ) & ( qi(1,1:4) >= qmin )
        N=10;
        inc_q=(qi-qa)/N;
        for( j=1:1:N )
            q(j,:)= qa + j*inc_q ;    % Interpolación lineal
        end
        for j=1:1:size(q,1)
```

```
            for h=1:1:size(q,2)
                fprintf(fd,'%2.3f    ',q(j,h));
            end
                fprintf(fd,' ;\n');
            qa=q(j,:);
        end
    else
        display('Los valores de q están fuera de rango');
    end
end
q=zeros(1,5);
qa;

%-------------------------------------------------------------------------
% WAIT
%-------------------------------------------------------------------------
display('WAIT()');
t=10;
for j=1:1:t
    q(j,:)= qa ;
end
for j=1:1:size(q,1)
    for h=1:1:size(q,2)
        fprintf(fd,'%2.3f    ',q(j,h));
    end
        fprintf(fd,' ;\n');
    qa=q(j,:);
end
qa;

%-------------------------------------------------------------------------
% MOVEREC
%-------------------------------------------------------------------------
display('MOVEREC( 10.0 , 5.5 , 10.0 , 10 )');
Pxyz=[10.0 , 5.5 , 10.0]';
%-------------------------------------------------------------------------
%       APROXIMACION
%-------------------------------------------------------------------------
px=Pxyz(1,1);
py=Pxyz(2,1)-.5;
pz=Pxyz(3,1);
%-----------------------------------------------
q1s = pz;
q2s = px;
if (py > 0.53) & (py <= 5.97)
    q3s =acos((py/2)/arm_lenght);
    q4s =2*asin((py/2)/arm_lenght);
end
if ( imag(q3s) ~= 0 ) | (imag(q4s) ~= 0 )
    display('Punto inalcanzable. Ubicado fuera del espacio de trabajo')
else
    qi=[q1s q2s q3s q4s 0];
    if ( qi(1,1:4) <= qmax ) & ( qi(1,1:4) >= qmin )
        N=10;
        inc_q=(qi-qa)/N;
        for( j=1:1:N )
            q(j,:)= qa + j*inc_q ;    % Interpolación lineal
        end
```

```
        for j=1:1:size(q,1)
            for h=1:1:size(q,2)
                fprintf(fd,'%2.3f    ',q(j,h));
            end
                fprintf(fd,' ;\n');
            qa=q(j,:);
        end
    else
        display('Los valores de q están fuera de rango');
    end
end
q=zeros(1,5);
qa;
%-------------------------------------------------------------------------
%       MOVIMIENTO FINAL
%-------------------------------------------------------------------------
px=Pxyz(1,1);
py=Pxyz(2,1);
pz=Pxyz(3,1);
%-----------------------------------------------
q1s = pz;
q2s = px;
if (py > 0.53) & (py <= 5.97)
    q3s =acos((py/2)/arm_lenght);
    q4s =2*asin((py/2)/arm_lenght);
end
if ( imag(q3s) ~= 0 ) | (imag(q4s) ~= 0 )
    display('Punto inalcanzable. Ubicado fuera del espacio de trabajo')
else
    qi=[q1s q2s q3s q4s 0];
    if ( qi(1,1:4) <= qmax ) & ( qi(1,1:4) >= qmin )
        N=10;
        inc_q=(qi-qa)/N;
        for( j=1:1:N )
            q(j,:)= qa + j*inc_q ;    % Interpolación lineal
        end
        for j=1:1:size(q,1)
            for h=1:1:size(q,2)
                fprintf(fd,'%2.3f    ',q(j,h));
            end
                fprintf(fd,' ;\n');
            qa=q(j,:);
        end
    else
        display('Los valores de q están fuera de rango');
    end
end
q=zeros(1,5);
qa;

%-------------------------------------------------------------------------
% WAIT
%-------------------------------------------------------------------------
display('WAIT()');
t=10;
for j=1:1:t
    q(j,:)= qa ;
end
```

```
for j=1:1:size(q,1)
    for h=1:1:size(q,2)
        fprintf(fd,'%2.3f    ',q(j,h));
    end
        fprintf(fd,' ;\n');
    qa=q(j,:);
end
qa;

%-------------------------------------------------------------------------
% MOVQ
%-------------------------------------------------------------------------
display('MOVQ( 8.0 , 9.0 , 1.4 , 0.3 , 10)');
N=10;
qi=[8.0 , 9.0 , 1.4 ,0.3 , 0 ];
if ( qi(1,1:4) <= qmax ) & ( qi(1,1:4) >= qmin )
    inc_q=(qi-qa)/N;
    q=zeros(N,5);
    for j=1:1:N
        q(j,:)= qa + j*inc_q ;
    end
    for j=1:1:size(q,1)
        for h=1:1:size(q,2)
            fprintf(fd,'%2.3f    ',q(j,h));
        end
            fprintf(fd,' ;\n');
        q1=q(j,1);q2=q(j,2);q3=q(j,3);
        Pa= eval(Eq)';
        qa=q(j,:);
    end
else
    display('Los valores de q están fuera de rango');
end
q=zeros(1,5);

%-------------------------------------------------------------------------
% WAIT
%-------------------------------------------------------------------------
display('WAIT()');
t=10;
for j=1:1:t
    q(j,:)= qa ;
end
for j=1:1:size(q,1)
    for h=1:1:size(q,2)
        fprintf(fd,'%2.3f    ',q(j,h));
    end
        fprintf(fd,' ;\n');
    qa=q(j,:);
end
qa;

%-------------------------------------------------------------------------
% HOME
%-------------------------------------------------------------------------
display('HOME()');
N=20;
qi=q0;
```

```
clear('q');
inc_q=(qi-qa)/N;
for j=1:1:N
    q(j,:)= qa + j*inc_q ;
end
for j=1:1:size(q,1)
    fprintf(fd,'%2.3f   ',q(j,:));
    fprintf(fd,' ;\n');
    qa=q(j,:);
end
q1=qa(1,1);q2=qa(1,2);q3=qa(1,3);
Pa= eval(Eq)';
qa;

%--------------------------------------------------------------------------
% END
%--------------------------------------------------------------------------
display('END()');
fclose(fd);
```

Anexo II

```
%--------------------------PLOTEO TRAYECTORIA--------------------------

clear all;
clc;
syms q1 q2 q3 q4

l3=3;   %largo del 1er brazo
l4=3;   %largo del 2do brazo


    %Primer matriz: translacion

        M_0A1 =[1 0 0 0; 0 1 0 0; 0 0 1 q1; 0 0 0 1];


    %Segunda matriz: rotacion Z 90° + rotacion X 90°
        a=[cosd(-90) -sind(-90) 0 0; sind(-90) cosd(-90) 0 0; 0 0 1 0; 0 0 0 1];
        b=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        c=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        d=[1 0 0 0; 0 cosd(-90) -sind(-90) 0; 0 sind(-90) cosd(-90) 0; 0 0 0 1];

                M_1A2=a*b*c*d

    %Tercera matriz: translacion


               M_2A3=[1 0 0 0; 0 1 0 0; 0 0 1 q2; 0 0 0 1]
%--------------------------PRIMER BRAZO--------------------------

    %Cuarta matriz: rotacion en Z, angulo variable q3 + desplazamiento l3
```

```
                a=[cos(q3+(pi/2)) -sin(q3+(pi/2)) 0 0; sin(q3+(pi/2)) cos(q3+(pi/2)) 0
0; 0 0 1 0; 0 0 0 1];
                b=[1 0 0 l3; 0 1 0 0; 0 0 1 0; 0 0 0 1];


                 M_3A4=a*b

    %Quinta matriz: rotacion en Z, angulo variable q4 + desplazamiento l4


                 a=[cos(q4) -sin(q4) 0 0; sin(q4) cos(q4) 0 0; 0 0 1 0; 0 0 0 1];
                 b=[1 0 0 l4; 0 1 0 0; 0 0 1 0; 0 0 0 1];

                 M_4A5=a*b

T= M_0A1 * M_1A2 * M_2A3 * M_3A4 * M_4A5;



   %-------------------------SEGUNDO BRAZO-------------------------

     %Cuarta matriz: rotacion en Z, angulo variable q3 + desplazamiento l3

                a=[cos(-q3-(pi/2)) -sin(-q3-(pi/2)) 0 0; sin(-q3-(pi/2)) cos(-q3-
(pi/2)) 0 0; 0 0 1 0; 0 0 0 1];
                 b=[1 0 0 l3; 0 1 0 0; 0 0 1 0; 0 0 0 1];


                 M_3A4_2=a*b;

     %Quinta matriz: desplazamiento l4


                 a=[cos(-q4) -sin(-q4) 0 0; sin(-q4) cos(-q4) 0 0; 0 0 1 0; 0 0 0 1];
                 b=[1 0 0 l4; 0 1 0 0; 0 0 1 0; 0 0 0 1];

                 M_4A5_2=a*b;

               T= M_0A1 * M_1A2 * M_2A3 * M_3A4_2 * M_4A5_2;


%lectrura del archivo .txt generado por el codigo del compilador

%fd = fopen('C:\Users\Tadeo Carlucci\Dropbox\robotica\TP_final\genearados con
compilador\prueba.txt','r');

%fd = fopen('C:\Users\Tadeo
Carlucci\Dropbox\robotica\TP_final\Prueba_nuestra\outq.txt','r');
fd = fopen('C:\Users\Tadeo Carlucci\Dropbox\robotica\TP_final\genearados con
compilador\valoresq2.txt','r');

tline = fgetl(fd);  % Extrae la primera línea que no contiene números
tline = fgetl(fd);

data = 'q=[';
while ischar(tline)
    data = sprintf ('%s %s;',data,tline);
    tline = fgetl(fd);
```

```
end

data=sprintf('%s];',data);
eval(data);

fclose(fd);




% Correción de ángulos múltiplos (k + 2*pi)
for j=1:1:size(q,1)
   q(j,3)=angle(exp(i*q(j,3)));
   q(j,4)=angle(exp(i*q(j,4)));
end

display('Listado de valores de las articulaciones');
q


%Calculo de los valores del extremo de cada eslabón, para asi poder
%graficarlos todos conjuntamente.

Po=[0 0 0 1]';

for j=1:1:size(q,1)

    q1=q(j,1);
    q2=q(j,2);
    q3=q(j,4)/2;
    q4=q(j,3)*2;



    %-------------------------PRIMER BRAZO-------------------------

    M=M_0A1;
    P(:,1)=Po;

    M=eval(M);
    P(:,2)=M*Po;

    M=M*M_1A2;
    P(:,3)=M*Po;

    M=M*eval(M_2A3);
    P(:,4)=M*Po;

    M=M*eval(M_3A4);
    P(:,5)=M*Po;

    M=M*eval(M_4A5);
    P(:,6)=M*Po;

    %-------------------------SEGUNDO BRAZO-------------------------
```

```
    M=M_0A1;
    P2(:,1)=Po;

    M=eval(M);
    P2(:,2)=M*Po;

    M=M*M_1A2;
    P2(:,3)=M*Po;

    M=M*eval(M_2A3);
    P2(:,4)=M*Po;

    M=M*eval(M_3A4_2);
    P2(:,5)=M*Po;

    M=M*eval(M_4A5_2);
    P2(:,6)=M*Po;
    %-----------------------------------------------------------------

    figure(3);
    view([30 30 ]);

    xlabel('Eje X');
    ylabel('Eje Y');
    zlabel('Eje Z');
    hold on;
    grid on;

    %Grafico 1er brazo

plot3(P(1,:),P(2,:),P(3,:),'Color','b','Marker','x','LineWidth',2,'MarkerSize',10);
    %Grafico 2do brazo

plot3(P2(1,:),P2(2,:),P2(3,:),'Color','b','Marker','x','LineWidth',2,'MarkerSize',10);
    %Grafico puntos de origen y del extremo del robot

plot3(P(1,6),P(2,6),P(3,6),'Color','r','Marker','o','LineWidth',2,'MarkerSize',10);

plot3(Po(1,1),Po(2,1),Po(3,1),'Color','k','Marker','o','LineWidth',2,'MarkerSize',10);

    xyz = sprintf ('Extremo (x,y,z) = ( %1.4f , %1.4f , %1.4f )\n(q1,q2,q3,q4) = (
%1.4f , %1.4f , %1.4f , %1.4f )',P(1,6),P(2,6),P(3,6),q1,q2,q3,q4);
    legend(1,xyz);

    axis equal;
    axis ([-1 15 -5 10 -1 15]);

    pause(.01);
    hold off;

    if( j < size(q,1) )
        clf;
    end

end
```