Universidad Zaragoza
1542

ANEXO 1
del
Trabajo Fin de Máster

Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed

CÁLCULOS DINÁMICOS

Autor

Adrián Rivero Pérez

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Curso 2011-2012

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

ÍNDICE

ÍNDICE DE FIGURAS

ANEXO 1.- Cálculos Dinámicos

1.1.- Dinámica Inversa

Los bloques simulink utilizados para el cálculo de los pares a partir de las coordenadas articulares del robot y modelo de paciente basan sus cálculos en la siguiente ecuación dinámica:

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g \qquad \textbf{Ecuación 1.1}$$

donde $q, \dot{q}\ y\ \ddot{q}$ son, respectivamente, los vectores de las coordenadas articulares generalizadas (en radianes), velocidades y aceleraciones, *M* es la matriz inercial en espacio articular, *C* es la matriz de coriolis y de acoplo centrípeto, *F* es la fuerza de fricción, *G* es la carga gravitatoria y *Q* es el vector de fuerzas generalizadas en los actuadores asociadas a las coordenadas generalizadas q.

El bloque simulink que muestra la figura 1.1 utiliza el algoritmo recursivo de Newton-Euler para calcular esta ecuación. Este algoritmo comienza en la base y hacia fuera añade la velocidad y aceleración de cada articulación para determinar la velocidad y aceleración de cada segmento. Entonces, desde el actuador final hasta la base, calcula las fuerzas y momentos actuando en cada segmento y con ello el vector de pares articulares.

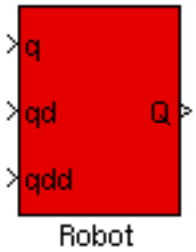

Figura 1.1: Bloque Simulink de Dinámica Inversa

Los elementos de las matrices *M,C,F* y *G* son complejas funciones de los parámetros cinemáticos e inerciales de cada segmento. Cada segmento tiene diez parámetros inerciales independientes: la masa del segmento m_j; el centro de gravedad (CdG) r_j respecto al eje de coordenadas del segmento; y seis momentos de inercia del segmento alrededor de su CdG pero respecto a los ejes alineados a los ejes del segmento.

1.1.1.- Término Gravitatorio

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + \boldsymbol{G(q)} + J(q)^T g \qquad \textbf{Ecuación 1.2}$$

Es el término normalmente dominante de la ecuación y que está presente aunque el robot esté parado ya que este necesita generar unos pares articulares capaces de mantener el brazo en una posición concreta. Este término depende únicamente de la masa de cada segmento y de la constante de gravedad (9.81 m/s^2 para todos los cálculos de esta memoria).

1.1.2.- Matriz Inercial

$$Q = \boldsymbol{M(q)}\ddot{\boldsymbol{q}} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g \qquad \textbf{Ecuación 1.3}$$

Se trata de una matriz simétrica que depende de la pose del brazo **robótico**. Los elementos diagonales Mjj describen la inercia vista por la articulación j, es decir, Qj = Mjj$\ddot{q}_j$. Los elementos no diagonales, Mij = Mji, i $\neq j$, representan el acoplo existente de las aceleraciones de la articulación j a la fuerza generalizada de la articulación i, si la articulación j acelera ejercerá un par sobre la articulación i.

1.1.3.- Matriz de Coriolis

$$Q = M(q)\ddot{q} + \boldsymbol{C(q,\dot{q})\dot{q}} + F(\dot{q}) + G(q) + J(q)^T g \qquad \textbf{Ecuación 1.4}$$

La matriz de Coriolis C es función de las coordenadas y velocidades articulares. Los pares centrípetos son proporcionales a $\dot{q}_i^2$, mientras que los pares de coriolis son proporcionales a $\dot{q}_i\dot{q}_j$. Los elementos fuera de la diagonal de la matriz C representan el acoplo de la velocidad de la articulación j en las fuerza generalizada en la articulación i.

1.1.4.- Fricción

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + G(q) + J(q)^T g \qquad \textbf{Ecuación 1.5}$$

En la mayoría de los robots eléctricos este es el mayor término después de la gravedad. Para cualquier maquinaria, motor o caja de engranajes rotacional, el par de fricción frente a la velocidad angular tiene una forma similar a la mostrada en la figura 1.2.

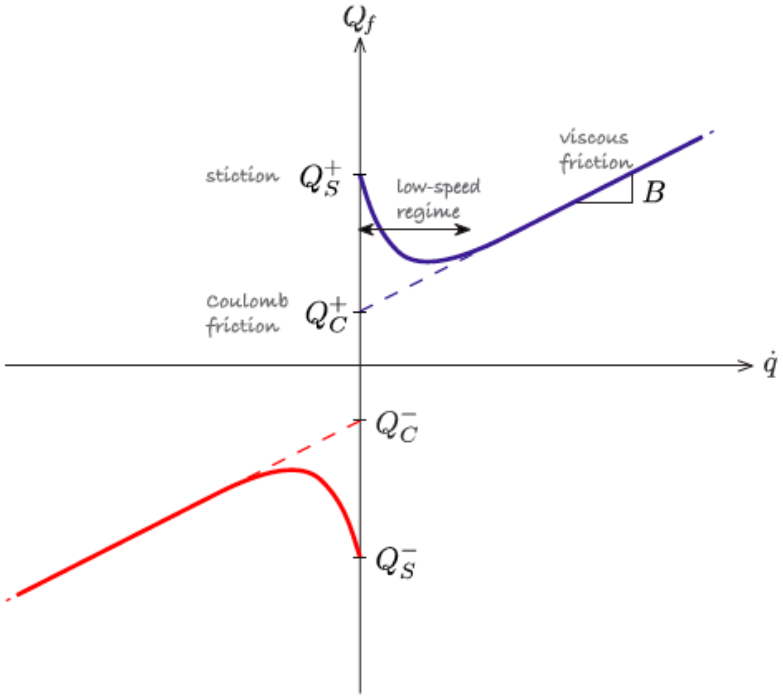

Figura 1.2: Par de fricción frente a velocidad angular [1]

A velocidad nula se observa el efecto de la fricción estática. El par aplicado debe ser superior al par debido a esta fricción antes de que se inicie cualquier movimiento. Una vez la maquinaria comience a moverse, la fricción estática disminuye y la fricción dinámica es la que domina. La fricción viscosa, mostrada por la línea de puntos de la figura 2, se modela normalmente mediante:

$$Q_f = B\dot{q} + Q_c \qquad \textbf{Ecuación 1.6}$$

donde la pendiente *B* es el coeficiente de fricción viscosa y Q_c es la fricción de Coulomb que se modela normalmente por una función no lineal.

$$Q_c = 0 \; si \;\; \dot{q} = 0 \qquad \textbf{Ecuación 1.7}$$

$$Q_c = Q_C^+ \; si \; \dot{q} > 0 \qquad \textbf{Ecuación 1.8}$$

$$Q_c = Q_C^- \; si \; \dot{q} < 0 \qquad \textbf{Ecuación 1.9}$$

En líneas generales, el valor de la fricción depende de la dirección de rotación pero esta asimetría es más pronunciada para la fricción de Couloumb que para la dinámica.

1.2.- Dinámica Directa

El bloque que calcula las posiciones, velocidades y aceleraciones articulares (figura 3), $q, \dot{q}\ y\ \ddot{q}$, según un cierto par articular Q utiliza la siguiente ecuación:

$$\ddot{q} = M^{-1}(q)(Q - C(q,\dot{q})\dot{q} - F(\dot{q}) - G(q)) \qquad \textbf{Ecuación 1.10}$$

a partir de despejar la aceleración angular de la ecuación de dinámica inversa. Tras este cálculo, se integra la ecuación para obtener primero la velocidad y luego la posición articular.

Figura 1.3: Bloque Simulink de Dinámica Directa

Por esta razón es necesario especificar la posición inicial de cada movimiento que servirán al bloque para especificar las condiciones de contorno y establecer los límites de integración.

Universidad Zaragoza
1542

ANEXO 2
del
Trabajo Fin de Máster

Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed

PROGRAMAS DE MATLAB

Autor

Adrián Rivero Pérez

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Curso 2011-2012

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

ÍNDICE

ANEXO 2.- Programas de Matlab

2.1.- Archivo de Inicialización "Inicialización.m"

El siguiente archivo de matlab es el que debe ejecutarse para iniciar el sistema. En él se pueden modificar: el coeficiente de resistencia *β*, el tiempo de movimiento *t*, el *paso*, el selector del modo en *automático* y el coeficiente de asistencia *assist*. Además se debe elegir el movimiento a simular.debajo de la línea 17, comentando las líneas del resto de movimientos. Este programa creará las gráficas de los diferentes pares.

```
%% Inicialización del Sistema
close all
clear all
%% Carga de los Par·metros Robot y Paciente
beta=0;
pacpar;
robpar;

%% Carga de los Par·metros de la simulación t debe ser al menos 100 %
veces superior al paso
t=6;
paso=0.1;
automatico=0;
%% Selección del movimiento de rehabilitación (elegir solo 1)
movimiento_alcance1;
% movimiento_flexion1;
%% Selección de Modo y Cálculo del assist

%Si automatico=1 se calcular· el assist necesario a partir de una
%enfermedad aleatoria. Si no hay que elegir el tipo de asistencia
% Si assist=0, el robot solo se encargar· de contrarrestar las fuerzas
%que él genera. El movimiento lo hará el humano mientras todo el sistema
%le sigue. Además si assist es 0 podemos elegir el nivel de "resist" que
%es el que tendrá que resistir el paciente.

if automatico==1
    [p,v,a]=jtraj(q(1,:),q(2,:),t);
    tau_sano=paciente.rne(p,v,a);
    enf=rand;
    tau_enf=enf*tau_sano;
    alfa=tau_enf./tau_sano;
    alfa=min(alfa);
    assist=1-alfa(1,1);
else
    assist=0;
    if assist == 0
        beta=1;
        pacpar;
        robpar;
    else
            beta=0;
    end
end
```

```
%% Carga de los Parámetros del Sistema
% Se ha de poner aquí el Sistema ya que sus parámetros dependen del
nivel de asistencia.
sistpar;
%% Ejecución del Sistema
%Se tiene la opción de cargar el modelo con el ploteado, lo que
%permitirá ver todo el movimiento pero irá más lento, o sin é
for i=1:length(q(:,1))-1
    qi=q(i,:);
    qf=q(i+1,:);
    options=simset('SrcWorkspace','current');
%     open('sistema_noplot.mdl');
%     set_param('sistema_noplot','AlgebraicLoopSolver','LineSearch')
%     sim('sistema_noplot.mdl');
    open('sistema_f.mdl');
    set_param('sistema_f','AlgebraicLoopSolver','LineSearch')
    sim('sistema_f');
    inic=((t/paso)+1)*(i-1)+1;
    fin=((t/paso)+1)*i;
    tau_enf(inic:fin,:)=Par_Enfermo;
    tau_assist(inic:fin,:)=Par_Assist;
    tau_robot(inic:fin,:)=Par_Min_Robot;
    tau_sistema(inic:fin,:)=Par_Sistema;
    tau_paciente(inic:fin,:)=Par_Paciente_conAssist;
    acc(inic:fin,:)=Aceleracion;
    vel(inic:fin,:)=Velocidad;
    intervalo=t*(i-1):paso:t*i;
    tiempo(inic:fin,1)=intervalo;
end
%% Ploteado de los distintos Pares del Sistema
figure(1)
subplot(2,3,1),plot(tiempo,tau_enf),title('Par      del      Paciente
Enfermo'),xlabel('tiempo(s)'),ylabel('Fuerza (N)');
subplot(2,3,2),plot(tiempo,tau_robot),title('Par      del      Robot
(solo)'),xlabel('tiempo(s)'),ylabel('Fuerza (N)');
subplot(2,3,3),plot(tiempo,tau_sistema),title('Par    del    Sistema
(robot+assist)'),xlabel('tiempo(s)'),ylabel('Fuerza (N)');
subplot(2,3,4),plot(tiempo,tau_paciente),title('Par del Paciente con
asistencia del Robot'),xlabel('tiempo(s)'),ylabel('Fuerza (N)');
if beta==0
    subplot(2,3,5),plot(tiempo,tau_assist),title(['Par  que  necesita
el                    paciente/proporciona                    robot
assist=',num2str(assist)]),xlabel('tiempo(s)'),ylabel('Fuerza (N)');
end
```

2.2.- Parámetros

Los siguientes programas inicializan, al ser llamados por el programa inicialización. m, los objetos *robot, paciente* y *sistema*. Los dos primeros son modificables y se deberán ajustar a las características de cada robot y paciente. El último se basa en los dos primeros y cambiará según el nivel de asistencia.

2.2.1.- Parámetros de Robot "robpar.m"

```
%% Parámetros del Robot Másas de los 2 enlaces del Robot 0=cuello
%hombro, 1=Brazo 2=Antebrazo
if beta==0
    m1r=25;
    m2r=25;
else
    m1r=25-m1p;
    m2r=25-m2p;
end
%Radios Cilindros Simulados Robot
Rr1=0.2;
Rr2=0.2;
%Longitudes
L1=0.5;
L2=0.3;
%Momentos de Inercia a partir del CdG(Se simulan cilindros)
if beta==0
    ILr1=m1r*(Rr1^2)/2;
    IRr1=(m1r/4)*((Rr1^2)+(L1^2)/3);
    ILr2=m2r*(Rr2^2)/2;
    IRr2=(m2r/4)*((Rr2^2)+(L2^2)/3);
else
    ILr1=m1r*(Rr1^2)/2-ILh1;
    IRr1=(m1r/4)*((Rr1^2)+(L1^2)/3)-IRh1;
    ILr2=m2r*(Rr2^2)/2-ILh2;
    IRr2=(m2r/4)*((Rr2^2)+(L2^2)/3)-IRh2;
end
%Distancia al CdG
r11=0.5*L1;
r12=0.5*L2;
clear LR;
close all;
%Matriz DH según los Enlaces(Links) del Robot
LR(1) = Link([ 0     0        0         pi/2      0]);
LR(2) = Link([ 0     0        0           pi/2   0]);
LR(3) = Link([ 0     0  -L1          0    0]);
LR(4) = Link([ 0     0      0         pi/2   0]);
LR(5) = Link([ 0     L2         0          0      0]);
%% Parámetros dinámicos del Robot
%Los que son 0 es debido a que hay 2 o m·s
% articulaciones en un mismo punto. Por tanto el enlace como tal no
% existe (m=0, I=0, r=0)
LR(1).m = 0;
LR(2).m = 0;
LR(3).m = m1r;
LR(4).m = 0;
LR(5).m = m2r;
% %Distancias a los CdG (Si el Link es 2, entonces se tomar· la
%referencia del Joint 3 (link 2 une joints 2-3)
LR(1).r = [ 0  0    0];
LR(2).r = [ 0  0  0];
LR(3).r = [r11    0    0];
LR(4).r = [ 0    0     0];
LR(5).r = [ 0    0     -r12];
% %Momentos de Inercia
LR(1).I = [0 0 0 0 0 0];
LR(2).I = [0 0 0 0 0 0];
LR(3).I = [ILr1 IRr1 IRr1 0 0 0];
LR(4).I = [0 0 0 0 0 0];
LR(5).I = [IRr2  IRr2  ILr2  0 0 0];
```

```
% %Fricción y Relaciones de Ejes
if beta==0
    LR(1).Jm =  200e-4;
    LR(2).Jm =  200e-4;
    LR(3).Jm =  200e-4;
    LR(4).Jm =  33e-4;
    LR(5).Jm =  33e-4;
else
    LR(1).Jm =  200e-4 - LH(1).Jm;
    LR(2).Jm =  200e-4- LH(2).Jm;
    LR(3).Jm =  200e-4- LH(3).Jm;
    LR(4).Jm =  33e-4- LH(4).Jm;
    LR(5).Jm =  33e-4- LH(5).Jm;
end
LR(1).G =  -62.6111;
LR(2).G =  107.815;
LR(3).G =  -53.7063;
LR(4).G =  76.0364;
LR(5).G =  71.923;
% % viscous friction (motor referenced)
if beta==0
    LR(1).B =   1.48e-3;
    LR(2).B =   .817e-3;
    LR(3).B =    1.38e-3;
    LR(4).B =   71.2e-6;
    LR(5).B =   82.6e-6;
else
    LR(1).B =   1.48e-3-LH(1).B;
    LR(2).B =   .817e-3-LH(2).B;
    LR(3).B =    1.38e-3-LH(3).B;
    LR(4).B =   71.2e-6-LH(4).B;
    LR(5).B =   82.6e-6-LH(5).B;
end
LR(1).Tc = [0   0];
LR(2).Tc = [0   0];
LR(3).Tc = [0   0];
LR(4).Tc = [0   0];
LR(5).Tc = [0   0];
%% Establecimiento del Objeto Robot.
%La base está trasladada con respecto al
%objeto paciente para permitir la simulación de ambos.
robot = SerialLink(LR, 'name', 'Robot',...
  'manufacturer', 'Unizar','base', transl(-0.5, 0.5, 0) );
```

2.2.2.- Parámetros del paciente "pacpar.m"

```
%% Parámetros del paciente
%Pesos del paciente, 1=Brazo 2=Antebrazo
m1p=0.4*(1+beta);
m2p=0.4*(1+beta);
%Radios de los "cilindros" a los que asemeja el brazo
Rh1=0.05;
Rh2=0.05;
%Longitudes
L1=0.5;
L2=0.3;
%Momentos de Inercia L longitudinal, R Radial
ILh1=m1p*(Rh1^2)/2;
IRh1=(m1p/4)*((Rh1^2)+(L1^2)/3);
ILh2=m2p*(Rh2^2)/2;
IRh2=(m2p/4)*((Rh2^2)+(L2^2)/3);
%Distancia al CdG, se suponen dos ejes alineados con el del enlace
(Link)
r11=0.5*L1;
r12=0.5*L2;
%Determinación parámetros DH en objetos Enlace (Links)
clear LH
%              th    d       a         alpha
LH(1) = Link([ 0     0        0          pi/2      0]);
LH(2) = Link([ 0     0        0            pi/2    0]);
LH(3) = Link([ 0     0   -L1          0     0]);
LH(4) = Link([ 0     0       0          pi/2    0]);
LH(5) = Link([ 0     L2          0           0       0]);
%%  Determinación  parámetros  dinámicos  (ver  robpar  para  más
%información)
%Masas
% (m=0, I=0, r=0)
LH(1).m = 0;
LH(2).m = 0;
LH(3).m = m1p;
LH(4).m = 0;
LH(5).m = m2p;
% %Distancias a los CdG
LH(1).r = [ 0   0    0];
LH(2).r = [ 0   0   0];
LH(3).r = [ r11   0    0];
LH(4).r = [ 0    0     0];
LH(5).r = [ 0    0     -r12];
% %Momentos de Inercia
LH(1).I = [0 0 0 0 0 0];
LH(2).I = [0 0 0 0 0 0];
LH(3).I = [ILh1 IRh1 IRh1 0 0 0];
LH(4).I = [0 0 0 0 0 0];
LH(5).I = [IRh2   IRh2   ILh2   0 0 0];
% %FricciÛn y Relaciones de Ejes
LH(1).Jm =  100e-6*(1+beta);
LH(2).Jm =  100e-6*(1+beta);
LH(3).Jm =  100e-6*(1+beta);
LH(4).Jm =  15e-6*(1+beta);
LH(5).Jm =  15e-6*(1+beta);
```

```
LH(1).G =  -62.6111;
LH(2).G =  107.815;
LH(3).G =  -53.7063;
LH(4).G =  76.0364;
LH(5).G =  71.923;
% % viscous friction (motor referenced)
LH(1).B =   1.48e-10*(1+beta);
LH(2).B =   .817e-10*(1+beta);
LH(3).B =    1.38e-10*(1+beta);
LH(4).B =   71.2e-10*(1+beta);
LH(5).B =   82.6e-10*(1+beta);
LH(1).Tc = [0   0];
LH(2).Tc = [0   0];
LH(3).Tc = [0   0];
LH(4).Tc = [0   0];
LH(5).Tc = [0   0];
%% CreaciÛn del objeto Paciente
paciente = SerialLink(LH, 'name', 'Paciente',...
    'manufacturer', 'Unizar');
```

2.2.3.- Parámetros del Sistema “sistpar.m”

```
%% Par·metros del Sistema
%Pesos del robot sistema, 1=Brazo 2=Antebrazo, h=humano, r=robot.
if beta==0

    ms1=m1r+assist*m1p;
    ms2=m2r+assist*m2p;
else
    ms1=m1r+m1p;
    ms2=m2r+m2p;
end

%Radios de los "cilindros" a los que asemeja el brazo
R1=Rh1+Rr1;
R2=Rh2+Rr2;
%Longitudes
L1=0.5;
L2=0.3;
%Momentos de Inercia L longitudinal, R Radial
if beta==0
    ILs1=(assist*ILh1)+ILr1;
    IRs1=(assist*IRh1)+IRr1;
    ILs2=(assist*ILh2)+ILr2;
    IRs2=(assist*IRh2)+IRr2;
else
    ILs1=(ILh1)+ILr1;
    IRs1=(IRh1)+IRr1;
    ILs2=(ILh2)+ILr2;
    IRs2=(IRh2)+IRr2;
end
%Distancia al CdG, se suponen dos ejes alineados con el del enlace
(Link)
r11=0.5*L1;
r12=0.5*L2;
```

```
%Determinación parámetros DH en objetos Enlace (Links)
clear LS
%              th    d       a         alpha
LS(1) = Link([ 0     0        0          pi/2      0]);
LS(2) = Link([ 0     0        0            pi/2   0]);
LS(3) = Link([ 0     0  -L1          0    0]);
LS(4) = Link([ 0     0      0          pi/2   0]);
LS(5) = Link([ 0     L2         0           0      0]);
%%  Determinación  parámetros  dinámicos  (ver  robpar  para  más
%información)
%Loa términos multiplicados por el "assist" son aquellos del robot
Humano
%Masas
% (m=0, I=0, r=0)
LS(1).m = 0;
LS(2).m = 0;
LS(3).m = ms1;
LS(4).m = 0;
LS(5).m = ms2;
% %Distancias a los CdG
LS(1).r = [ 0  0    0];
LS(2).r = [ 0  0  0];
LS(3).r = [ r11  0    0];
LS(4).r = [ 0    0     0];
LS(5).r = [ 0    0     -r12];
% %Momentos de Inercia
LS(1).I = [0 0 0 0 0 0];
LS(2).I = [0 0 0 0 0 0];
LS(3).I = [ILs1 IRs1 IRs1 0 0 0];
LS(4).I = [0 0 0 0 0 0];
LS(5).I = [IRs2  IRs2  ILs2  0 0 0];

%% FricciÛn y Relaciones de Ejes
if  beta==0
    LS(1).Jm =  (assist*100e-6)+200e-4;
    LS(2).Jm =  (assist*100e-6)+200e-4;
    LS(3).Jm =  (assist*100e-6)+200e-4;
    LS(4).Jm =  (assist*15e-6)+33e-4;
    LS(5).Jm =  (assist*15e-6)+33e-4;
else
    LS(1).Jm =(100e-6)+200e-4;
    LS(2).Jm =(100e-6)+200e-4;
    LS(3).Jm =(100e-6)+200e-4;
    LS(4).Jm =(15e-6)+33e-4;
    LS(5).Jm =(15e-6)+33e-4;
end
LS(1).G =  -62.6111;
LS(2).G =  107.815;
LS(3).G =  -53.7063;
LS(4).G =  76.0364;
LS(5).G =  71.923;
%% viscous friction (motor referenced)
if beta==0
LS(1).B =   (assist*1.48e-10)+1.48e-3;
LS(2).B =   (assist*.817e-10)+.817e-3;
LS(3).B =   (assist*1.38e-10)+1.38e-3;
LS(4).B =   (assist*71.2e-10)+71.2e-6;
LS(5).B =   (assist*82.6e-10)+82.6e-6;
else
```

```
LS(1).B =(1.48e-10)+1.48e-3;
LS(2).B =   (.817e-10)+.817e-3;
LS(3).B =   (1.38e-10)+1.38e-3;
LS(4).B =   (71.2e-10)+71.2e-6;
LS(5).B =   (82.6e-10)+82.6e-6;
end
LS(1).Tc = [0   0];
LS(2).Tc = [0   0];
LS(3).Tc = [0   0];
LS(4).Tc = [0   0];
LS(5).Tc = [0   0];
%% Creación del objeto Sistema
sistema = SerialLink(LS, 'name', 'Sistema',...
    'manufacturer', 'Unizar','base', transl(+0.5, -0.5, 0));
```

2.3.- Ejercicios

Los siguientes archivos marcan las pautas de los distintos ejercicios simulados. Están divididos en varias matrices con las posiciones articulares de cada una de las articulaciones en radianes. Además de añadir movimientos para efectuar la simulación sirven como ejemplo para otros ejercicios que puedan programarse. Las últimas líneas comentadas se utilizan para probar y pre visualizar los movimientos antes de simularlos.

2.3.1.- Movimiento_flexion1.m

```
%%Movimiento Flexión 1
%Movimiento de Flexión de Biceps con 2 repeticiones
q(1,:)=[0 pi/2 0 -pi/2 0]; %posición inicial, robot de 5 Links
q(2,:)=[0 pi/2 0 pi/4 0];
q(3,:)=[0 pi/2 0 -pi/2 0];
q(4,:)=[0 pi/2 0 pi/4 0];
q(5,:)=[0 pi/2 0 -pi/2 0];
%% Prueba del Movimiento (comentado normalmente)
% for i=1:length(q(:,1))-1
%     t=0:0.056:2;
%     qi=q(i,:);
%     qf=q(i+1,:);
%     movimiento=jtraj(qi,qf,t);
%     for j=1:length(movimiento),
%         paciente.plot(movimiento(j,:));
%     end
% end
```

2.3.2.- movimiento_alcance1.m

```
%%Movimiento Alcance 1
%Ejercicio para alcanzar un objeto frente el paciente a media altura
q(1,:)=[0 pi/2 0 -pi/2 0]; %posición inicial, robot de 5 Links
q(2,:)=[0 pi/2 0 0 0];
q(3,:)=[0 pi/2 0.3750*pi -pi/2 0];
q(4,:)=[0 pi/2 0   0 0];
q(5,:)=[0 pi/2 0 -pi/2 0];
%% Prueba del Movimiento (comentado normalmente)
% for i=1:length(q(:,1))-1
%    t=0:0.056:2;
%
%     qi=q(i,:);
%     qf=q(i+1,:);
%     movimiento=jtraj(qi,qf,t);
%     for j=1:length(movimiento),
%         paciente.plot(movimiento(j,:));
%     end
% end
```

2.3.3.- movimiento_alcance2.m

```
%%Movimiento Alcance 2
%Ejercicio para alcanzar un objeto a la izquierda del paciente a
%alta altura
q(1,:)=[0 pi/2 0 -pi/2 0]; %posición inicial, robot de 5 Links
q(2,:)=[0 pi/2 0 0 0];
q(3,:)=[0.5 pi/2 0.5750*pi -pi/2 0];
q(4,:)=[0 pi/2 0   0 0];
q(5,:)=[0 pi/2 0 -pi/2 0];
% Prueba del Movimiento (comentado normalmente)
% for i=1:length(q(:,1))-1
%    t=0:0.056:2;
%
%     qi=q(i,:);
%     qf=q(i+1,:);
%     movimiento=jtraj(qi,qf,t);
%     for j=1:length(movimiento),
%         paciente.plot(movimiento(j,:));
%     end
% end
```

2.3.4.- movimiento_alcance3.m

```
%%Movimiento Alcance 3
%Ejercicio para alcanzar un objeto a la derecha del paciente a baja
%altura
q(1,:)=[0 pi/2 0 -pi/2 0]; %posición inicial, robot de 5 Links
q(2,:)=[0 pi/2 0 0 0];
q(3,:)=[-0.5 pi/2 0.2750*pi -pi/2 0];
q(4,:)=[0 pi/2 0   0 0];
q(5,:)=[0 pi/2 0 -pi/2 0];
% Prueba del Movimiento (comentado normalmente)
% for i=1:length(q(:,1))-1
%    t=0:0.056:2;
%
%     qi=q(i,:);
%     qf=q(i+1,:);
%     movimiento=jtraj(qi,qf,t);
%     for j=1:length(movimiento),
%         paciente.plot(movimiento(j,:));
%     end
% end
```

2.3.5.- movimiento_beber1.m

```
%%Movimiento Beber 1
q(1,:)=[0 pi/2 0 -pi/2 0]; %posición inicial, robot de 5 Links
q(2,:)=[0 pi/2 0 0 0];
q(3,:)=[0 pi/2 0.3750*pi -pi/2 0];
q(4,:)=[0.2 pi/2 0.2*pi 0.9*pi/2 0];
q(5,:)=[0 pi/2 0.3750*pi -pi/2 0];
q(6,:)=[0 pi/2 0 -pi/2 0];
% Prueba del Movimiento (comentado normalmente)
% for i=1:length(q(:,1))-1
%    t=0:0.056:2;
%     qi=q(i,:);
%     qf=q(i+1,:);
%     movimiento=jtraj(qi,qf,t);
%     for j=1:length(movimiento),
%         paciente.plot(movimiento(j,:));
%     end
% end
```

2.4.- Inicialización Control inic_control.m

```
%% Inicialización del Sistema
close all
clear all
%% Carga de los Parámetros Robot y Paciente
beta=0;
pacpar;
robpar;
%% Carga de los Parámetros de la simulación
%t debe ser al menos 100 veces superior al paso
t=7;
paso=0.002;
automatico=0;
%% Parámetros de control
P=500;
I=0;
D=0;
N=1;
delay=paso;
P2=500;
I2=0;
D2=0;
N2=1;
%% Selección del movimiento de rehabilitación (elegir solo 1)
movimiento_alcance1;
% movimiento_flexion1;
%% Nivel de Asistencia
assist=0.5;
%% Carga de los Parámetros del Sistema
% Se ha de poner aquí el Sistema ya que sus parámetros dependen del
%nivel de asistencia.
sistpar;
%% Ejecución del Sistema
%Se tiene la opción de cargar el modelo con el ploteado, lo que
%permitir· ver todo el movimiento pero ir· más lento, o sin él
for i=1:length(q(:,1))-1
    qi=q(i,:);
    qf=q(i+1,:);
    options=simset('SrcWorkspace','current');
    open('control.mdl');
    set_param('control','AlgebraicLoopSolver','LineSearch')
    sim('control');
     inic=((t/paso)+1)*(i-1)+1;
    fin=((t/paso)+1)*i;
    referencia(inic:fin,:)=ref;
    posicion(inic:fin,:)=pos;
    intervalo=t*(i-1):paso:t*i;
    tiempo(inic:fin,1)=intervalo;
    par_ruido(inic:fin,:)=noise_torque;
    par_limpio(inic:fin,:)=clean_torque;
    par_control(inic:fin,:)=control_torque;

end
figure(1)
subplot(2,1,1),plot(tiempo,referencia,'--
',tiempo,posicion),title('PosiciÛn              Actual              y
Referencia'),xlabel('tiempo(s)'),ylabel('rad');
subplot(2,1,2),plot(tiempo,(referencia-posicion)),title('Error    de
PosiciÛn'),xlabel('tiempo(s)'),ylabel('rad');
```

```
figure(2)
subplot(3,1,1),plot(tiempo,par_limpio,tiempo,posicion),title('Par
sin ruido'),xlabel('tiempo(s)'),ylabel('Nm');
subplot(3,1,2),plot(tiempo,par_ruido),title('Par con
ruido'),xlabel('tiempo(s)'),ylabel('Nm');
subplot(3,1,3),plot(tiempo,par_control),title('Par de
control'),xlabel('tiempo(s)'),ylabel('Nm');
```

Universidad Zaragoza
1542

ANEXO 3
del
Trabajo Fin de Máster

Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed

ESPECIFICACIONES

Autor

Adrián Rivero Pérez

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Curso 2011-2012

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

ÍNDICE

ANEXO 3.- Especificaciones

En este anexo se especifican los parámetros numéricos utilizados en las simulaciones del capítulo 4 Resultados y además se describen la matriz DH de 5 grados de libertad utilizada para describir tanto el modelo del Robot como del Paciente.

2.1.- Parámetros Robot

Masa del Brazo m1r (kg)	25
Masa del Antebrazo m2r (kg)	25
Radio del Brazo R1r (m)	0.2
Radio del Antebrazo R2r (m)	0.2
Longitud del Brazo L1	0.3
Longitud del Antebrazo L2	0.5

2.2.- Parámetros Paciente

Masa del Brazo m1r (kg)	0.4
Masa del Antebrazo m2r (kg)	0.4
Radio del Brazo R1r (m)	0.05
Radio del Antebrazo R2r (m)	0.05
Longitud del Brazo L1	0.3
Longitud del Antebrazo L2	0.5

2.3.- Matriz Dh

θ	**d**	**a**	**α**
0	0	0	**π**/2
0	0	0	**π**/2
0	0	-L1	0
0	0	0	**π**/2
0	L2	0	0

Universidad Zaragoza
1542

ANEXO 4
del
Trabajo Fin de Máster

Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed

ALTERNATIVAS ESTUDIADAS

Autor

Adrián Rivero Pérez

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Curso 2011-2012

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

ÍNDICE

ÍNDICE DE FIGURAS

ANEXO 4.- Anexo IV Alternativas Estudiadas

Este anexo muestra otra alternativa al diseño actual del sistema. Esta alternativa se diseñó después de descartar un modelo programado enteramente en matlab, sin usar simulink. Sin embargo la existencia de algunos problemas en funciones de la toolbox de robótica obligaron a utilizar el simulink. El diseño que se describe a continuación fue el más robusto de todos sus predecesores. Se ha decidido describirlo ya que alberga ideas y conceptos muy útiles que podrían fomentar otros diseños o ser utilizadas en otros proyectos.

Este diseño partía de la idea de diseñar un bloque simulink llamado “Assist” que actuara como interfaz entre los modelos del robot y el paciente. La figura 1 muestra el sistema.

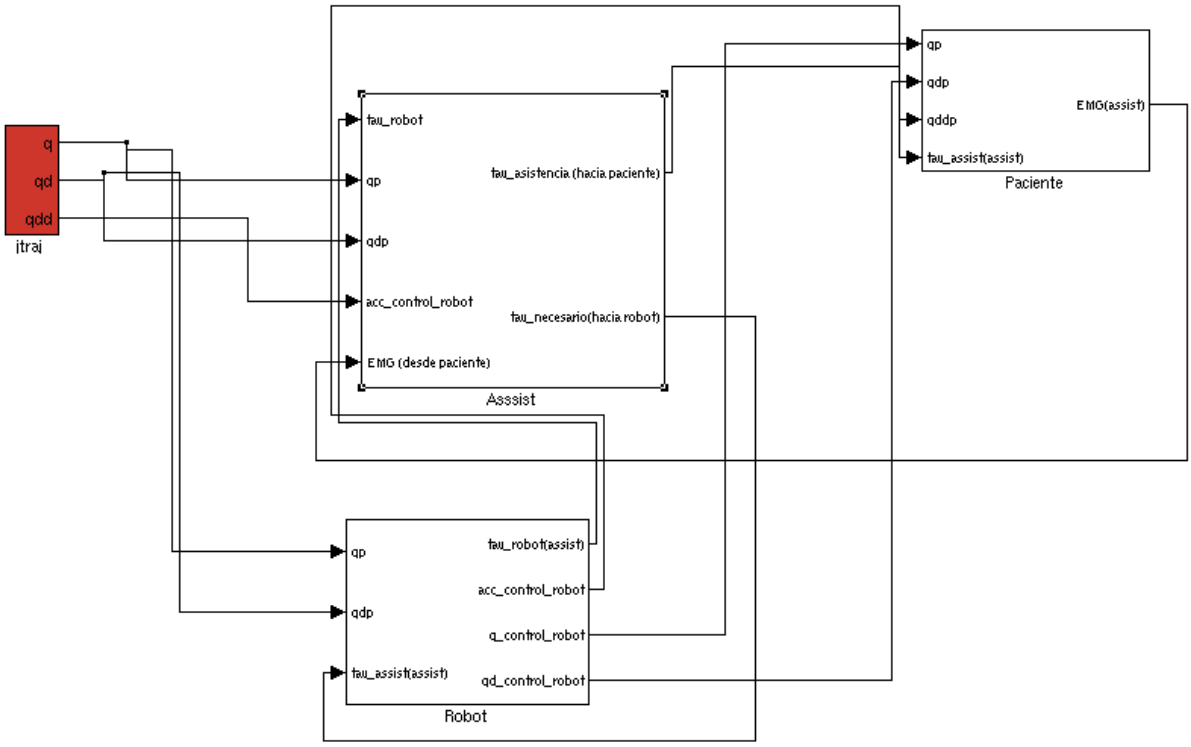

Figura 4.1 Sistema Alternativo

A simple vista el sistema se muestra más complicado que el sistema final. En este caso las consignas de posición, velocidad y aceleración angular entran en dos bloques: Robot y Assist.

El bloque Assist, detallado en la figura 4.2, utiliza estas consignas de movimiento articular para calcular el par que harían dos modelos: el paciente sano y el robot si solo tuviese que moverse a si mismo, es decir, como si no estuviera anclado al brazo del paciente.

Al primero se le resta el par que hace el paciente real, con la discapacidad y luego al resultante se le aplica el nivel de asistencia; igual que en el modelo final. Este par se envía al modelo del robot, y es el par que debe suministrar este para ayudar al paciente (salida 2). En cuanto al par del modelo del robot, este se resta del par que está haciendo el robot realmente mientras ayuda al paciente (entrada 1). Al hacer esto lo que queda es el par de asistencia que le da el paciente al robot y que irá al bloque del paciente (salida 1).

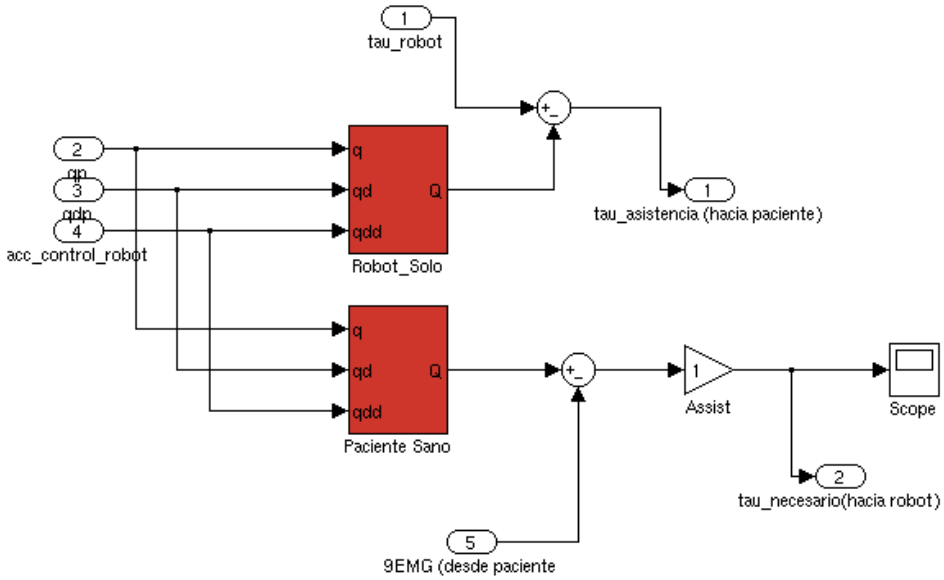

Figura 4.2 Detalle Bloque Assist del Sistema Alternativo

La figura 4.3 detalla el bloque del paciente. Para empezar la aceleración angular (entrada 3) no se utiliza en este bloque. En vez de eso, la aceleración que se utiliza como base para calcular el par del paciente sano, viene de un circuito de control. Esto se hizo con la intención de controlar el movimiento del modelo paciente para que siguiera siempre las consignas marcadas por el movimiento del robot: el control compara el movimiento que debería seguir el modelo, dado por las entradas 1 y 2 con las que sigue realmente, y acelera o desacelera el modelo produciendo más o menos par. A este par se le aplica un coeficiente de discapacidad, complementario al de asistencia, para simular la discapacidad del paciente y que se envía al bloque assist (salida 1) y luego se le suma el par de asistencia proveniente y calculado en el bloque assist (entrada 4).

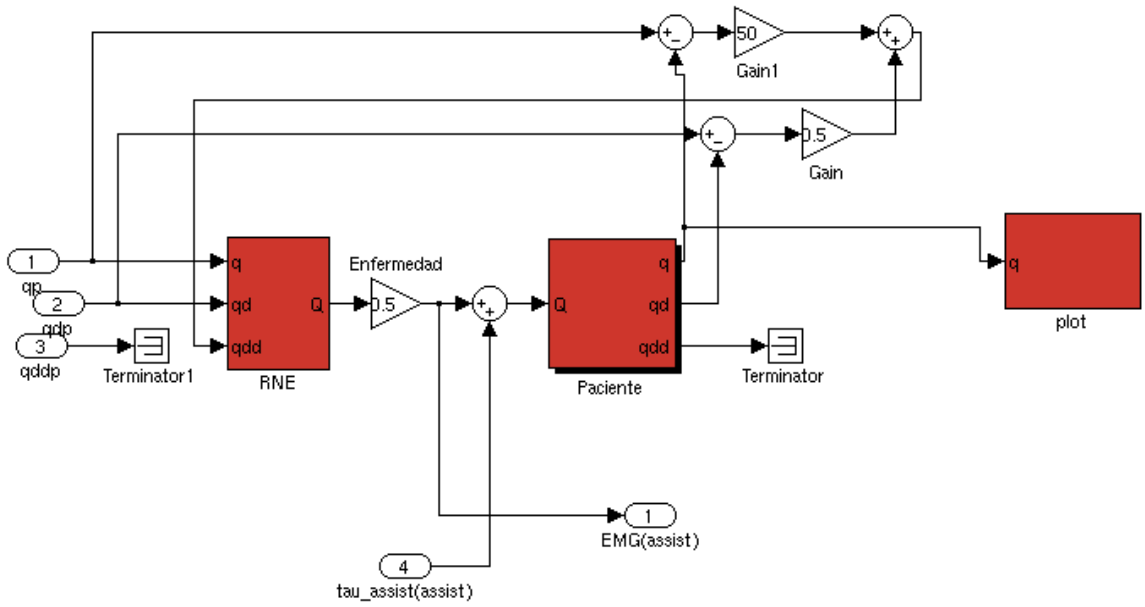

Figura 4.3 Detalle Bloque Paciente del Sistema Alternativo

Por último, la figura 4 muestra el detalle del bloque robot. En este sistema alternativo este bloque el que más ha cambiado respecto al del sistema final. En un principio funciona al igual que el bloque del paciente, con dos consignas de movimiento (entradas 1 y 2) para posición y velocidad angular y luego la aceleración proveniente del control. Descontando esto, la mayor diferencia está en el bloque de cálculo del par del robot. Este ha sido modificado a partir del de "robotics toolbox" para incluir una entrada extra: una fuerza externa aplicada en el efector final del robot, esto es, en la muñeca. La idea era asemejar el par que debe hacer el robot para ayudar al paciente como una fuerza puntual aplicada en el extremo. Así, se esperaba que el modelo de cálculo de par suministrase un par adicional que compensase esta fuerza y ayudase al paciente.

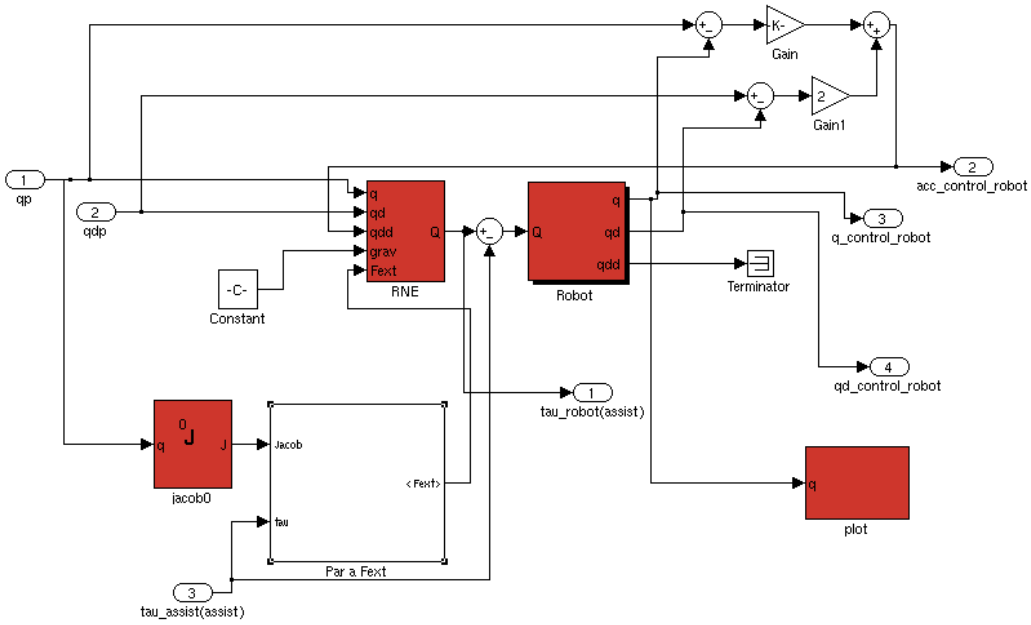

Figura 4.4 Detalle Bloque Robot del Sistema Alternativo

Para transformar el par extra en una fuerza se utiliza el bloque "Par a Fext" que como entradas utiliza el jacobiano, que se calcula en cada posición angular q y el par de asistencia (entrada 3). Este bloque, que se detalla en la figura 4.5, utiliza varias funciones de matlab para, en primer lugar calcular si el brazo robótico ha perdido algún grado de libertad. Esto suele ocurrir cuando una o más articulaciones se alinean y hace que el jacobiano muestre matrices dependientes de manera que, por tanto, su inversa es imposible. Cuando se calculan las articulaciones dependientes, es posible reducir el jacobiano borrando estas columnas. Se hace lo mismo con el par de asistencia, de forma que se puedan multiplicar las matrices, y luego se multiplican para hallar la fuerzas externas.

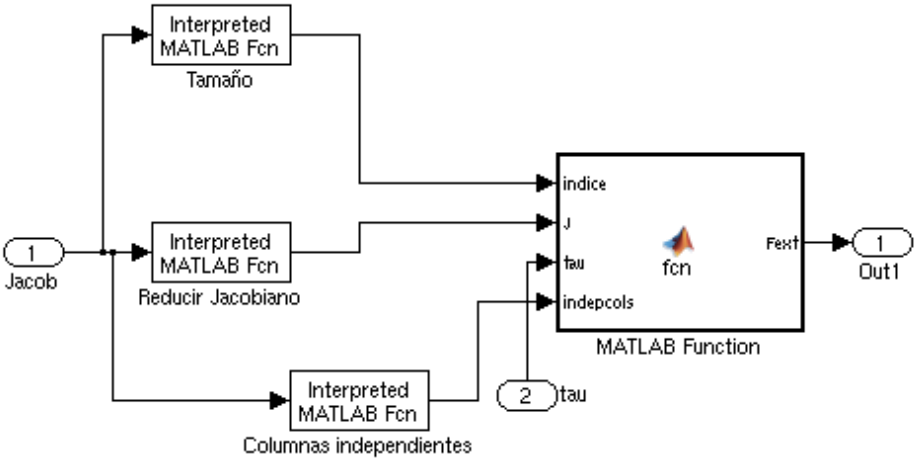

Figura 4.5 Detalle del Bloque de Par a Fext

Debido a la limitación que tiene el simulink de implementar funciones del matlab directamente. Se ha tenido que separar la función de calcular la fuerza externa en cuatro bloques. Estos bloques llaman a distintas funciones de matlab descritas a continuación.

```
function[cero]=tamanyo(J)
[R, jb] = rref(J);
    depcols = setdiff( 1:numcols(J), jb);

cero=length(depcols);
end
```

Figura 4.6 Función Tamaño del Bloque Par a Fext

La función tamaño, descrita en la figura 4.6, lo que hace en primer lugar es buscar, con el comando *rref* y *depcols* para encontrar las columnas dependientes del Jacobiano. Entonces el vector *cero* guarda el tamaño del vector *depcols* que indica las columnas dependientes. Si el jacobiano tiene dependientes las columnas 4 y 6 entonces *depcols*=[4 6] y cero=2. En cambio si las columnas dependientes son la 2, la 3 y la 5, *depcols*=[2 3 5] y cero=3.

La función redJ, descrita en la figura 4.7, reduce el jacobiano quitando las columnas de las articulaciones dependientes. Para ello lo primero que hace es generar los mismos vectores *depcols* y *cero* que en la función tamaño. A partir de *cero* se crea una matriz de 6 filas y de columnas *O* según el número que indique el vector *cero*. Es decir, como el número de columnas dependientes del jacobiano.

```
function[JR]=redJ(J)
%Primero buscamos las columnas dependientes
 [R, jb] = rref(J);
   depcols = setdiff( 1:numcols(J), jb);
%Ahora reducimos el Jacobiano
cero=length(depcols);
indice=setdiff(1:5,depcols);
JR=J(:,indice);
O=zeros(6,cero);
JR=[JR O];
end
```

Figura 4.7 Función Reducir Jacobiano del Bloque Par a Fext

A partir de *depcols* se genera el vector índice. Este vector contiene cuales son las columnas independientes del Jacobiano. A partir de índice y el jacobiano original se genera un jacobiano reducido *JR*. Con esto debería bastar, pero este tipo de funciones de matlab en simulink no permite un vector cuyas dimensiones sean variables. Por ello utilizamos la matriz *O* para sumarle al jacobiano reducido tantas columnas como sean necesarias para que tenga las mismas dimensiones que el jacobiano original. Esto hace que la función produzcan una matriz de [6 5] sin importar el número de columnas dependientes del jacobiano.

La tercera función j_dep, columnas independientes, crea un vector de [1 5] que contiene primero las columnas independientes y luego se rellena de ceros. Este procedimiento es el mismo que en los casos anteriores. Se puede ver en detalle esta función en la figura 4.8.

```
function[indepcols]=j_dep(J)
[R, jb] = rref(J);

depcols = setdiff( 1:numcols(J), jb);
indice=setdiff(1:5,depcols);

cero=length(depcols);
O=zeros(1,cero);
indepcols=[indice O];
end
```

Figura 4.8 Función Columnas Independientes del Bloque Par a Fext

Por último, el bloque final integra las salidas de las funciones anteriores y además tiene como entrada el par de todas las articulaciones. Lo primero que hace esta función es quitar los ceros sobrantes de la matriz *JR* utilizando para ello el vector *índice.* Además también se reduce el tamaño de la matriz de par, *tau*, utilizando el vector *indepcols.* Una vez hecho esto se procede a realiza los cálculos para calcular la fuerza externa a partir del par aplicado en las articulaciones:

$$Fext = (J')^{-1}\tau$$

Puesto que el jacobiano no es una matriz cuadrada, para calcular su inversa se utilizará la pseudoinversa.

Esta última función se describe en la figura 4.9:

```
function Fext = fcn(indice,J,tau,indepcols)
indep=5-indice;
JR =J(:,1:indep);
indepcols=indepcols(1,1:indep);
taur=tau(indepcols,:);
JRinv=pinv(JR);
%% Pseudoinversa y cálculo de las Fuerzas equivalentes en efector
Final

Fext=(JRinv')*(taur); %Este vector siempre será de 1x6
end
```

Figura 4.9 Función de Cálculo de Fuerza externa del bloque de Par a Fext

La salida de este bloque produce un vector de Fuerza externa, con seis elementos: 3 fuerzas y 3 momentos en cada uno de los ejes. Este vector entra en el bloque modificado que debe producir un mayor par a la salida para compensar el brazo del paciente. Para tener en cuenta el control y realizar el ploteado se resta de este par el par de asistencia.

Matemáticamente esto debería funcionar. Sin embargo el bloque modificado no responde como debería ante esta fuerza externa. Debido a que estamos simplificando no solo el movimiento del antebrazo, sino también del brazo en una fuerza aplicada en un único punto, el bloque no sabe realmente que articulaciones debe mover. Esto hace que algunas articulaciones no se muevan para un cierto movimiento o que lo hagan otras que no deberían moverse.

Esta es la razón por la que se decidió descartar este procedimiento y se planteó el sistema que desembocaría en el sistema final. No obstante este diseño puede ser útil en otras aplicaciones sobretodo la parte que permite utilizar el jacobiano aún cuando se producen singularidades en el robot.

Universidad Zaragoza
1542

ANEXO 5
del
Trabajo Fin de Máster

Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed

EXPERIMENTACIÓN

Autor

Adrián Rivero Pérez

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Curso 2011-2012

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

ÍNDICE

ÍNDICE DE FIGURAS

ANEXO 5.- Anexo V Experimentación

5.1.- Introducción

La parte experimental de este proyecto se realizaró en el Instituto de la Bioingeniería de Cataluña (IBEC) situado en el Campus Norte Universitario de Barcelona. Aquí se trabajó con el grupo de Robótica liderado por la profesora doctora Alicia Casals. Este equipo también está involucrado en el proyecto Hyper (ver capitulo 1) y utilizaban un robot Kuka para realizar diferentes pruebas, concentrando su línea de trabajo en investigación en el control del mismo. Además de este robot el laboratorio cuenta con: un equipo básico de electrónica, un monitor 3D de 50', probeta ultrasónica, 2 ordenadores con arquitectura de multiprocesadores, una plataforma BCI (Interfaz cerebro computador), un sistema de iluminación LED controlado por ordenador y una cocina robótica experimental.

Durante los días 4 y 5 de Septiembre de 2012 el alumno autor de esta tesis trabajó con los ingenieros estudiantes de doctorado D. Xavier Giralt y D. Luis Amigo, aunque previamente ya se había contactado con todo el equipo varias veces, vía video conferencia y correo electrónico, para establecer los objetivos de este trabajo en conjunto, además de coordinar los preparativos necesarios para un trabajo más fluido.

5.2.- Sistema

El **Robot Kuka REL** (figura 5.1) es un robot de estructura liviana creado por la empresa **Kuka Robotics**. Este robot dispone de una estructura exterior de aluminio con una carga útil de 7 kg y está enfocado a tareas de manipulación y montaje. El peso propio del robot es de 16 kg lo que le permite obtener una gran eficiencia energética.

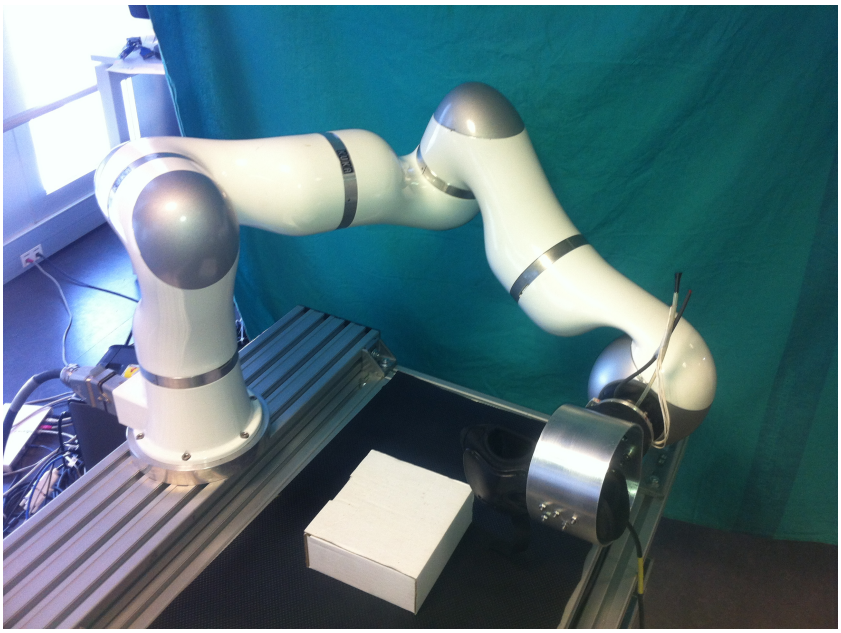

Figura 5.2: Robot Kuka REL del departamento de robótica del IBEC

El KUKA REL se suele utilizar en escuelas politécnicas, institutos de investigación científica, departamentos de empresas industriales o del sector médico dedicadas a la investigación o el pre desarrollo de productos.

Al robot se le colocó en el extremo un guante que actúa como agarre para el paciente (figura 5.2) y, además, dispone de un sensor para medir la fuerza que efectúa el paciente. El robot está anclado a una mesa mediante tornillos. La posición se puede fijar pero se puede cambiar a distintos lugares de la misma mesa para ofrecer más posibilidades a los distintos movimientos.

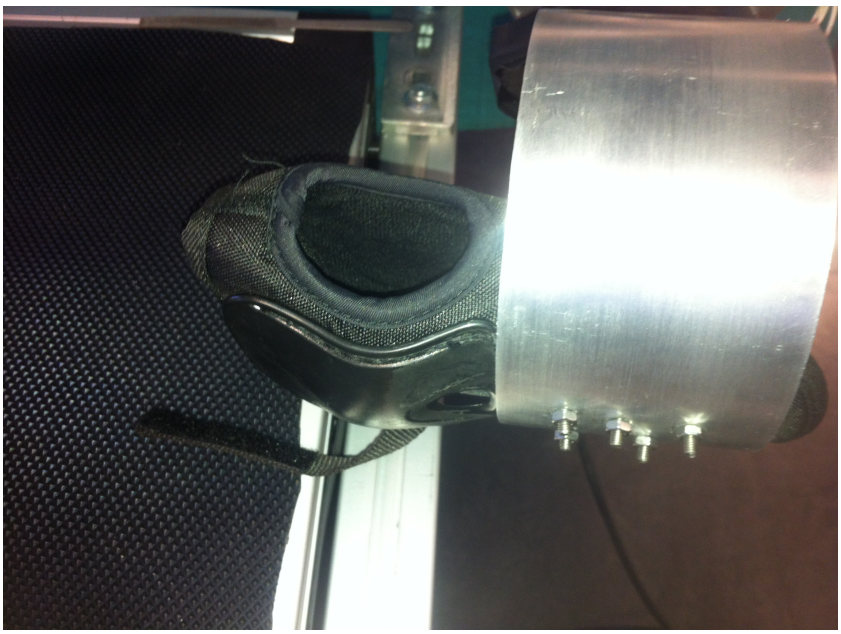

Figura 5.2: Guante colocado en el efector final del robot Kuka

En el laboratorio se utiliza el robot KUKA para realizar distintas pruebas necesarias dentro del ámbito del proyecto Hyper. Así pues, el equipo crea programas en C# que se conectan con el robot y permiten la realización de los movimientos que se ajustan a las características físicas de cada persona. En el caso del movimiento de flexión de brazo, por ejemplo, se le "indica" al robot la longitud del antebrazo del paciente. A partir de este dato y estableciendo previamente los ángulos de la flexión, el programa calcula el arco que debe hacer.

De los dos experimentos efectuados: flexión y alcance, solo el de flexión estaba programado antes de realizar las pruebas ya que también formaba parte de otro proyecto. Para el movimiento de alcance se tuvo que realizar la programación en ese instante, limitando, como se ve en los resultados, las pruebas.

El protocolo de programación del robot KUKA permite obtener los siguientes datos después de cada prueba.

- **Posiciones en grados de cada una de las articulaciones**
- **Posición en coordenadas cartesianas del efector final**
- **Par Medido en cada articulación**
- **Estimación del Par en cada articulación**
- **Fuerza Medida en el sensor**
- **Estimación de la Fuerza en el efector final**

Aparte de esto, en el laboratorio disponen de un sistema de EMG que se puede conectar al usuario que realiza las pruebas, a modo de medir la actividad muscular del brazo.

Para efectuar el paradigma "**assist as needed**" detrás del proyecto, los ingenieros diseñadores, ajustaron el ratio de desplazamiento del efector final según la fuerza medida en el sensor[1]. Es decir, a menor nivel de asistencia el robot se mueve menos ante una fuerza medida. Para el nivel de resistencia, el programa está programado para establecer el porcentaje de cierto valor máximo de fuerza resistiva en el efector final. De esta manera si se elige una fuerza máxima resistiva de 30N y un nivel de resistencia de 25, el paciente o usuario habría de vencer 7.5N.

[1] *Robotic Platform, X. Giralt, L. Amigo, A. Casals, pendiente de publicación*

5.3.- Objetivos

El objetivo de estas sesiones de experimentación es el de validar experimentalmente el esquema de control diseñado en esta memoria. Se trata percibir diferencias notables en las fuerzas según los distintos niveles de asistencia/resistencia a fin de obtener una relación entre la fuerza que hace el paciente y estos dos parámetros y poder comparar con los datos simulados.

Inicialmente se pretendía conectar el sistema diseñado en este proyecto directamente al robot KUKA con los cambios necesarios para adaptarlo al tiempo real y la arquitectura del robot. No obstante esto resulta, a día de hoy, imposible. Al usar Simulink, la conexión entre el robot y el ordenador llegaba a producirse en un primer instante para luego perder la señal en cuanto el programa producía las ordenes. No se ha descubierto la razón detrás de este problema aunque se espera que se pueda resolver en un futuro.

A partir de entonces se consideraron varias alternativas y se llegó a la conclusión de que lo mejor sería probar distintos movimientos en el robot KUKA midiendo pares y posiciones con el sistema que ya programó el equipo del IBEC. Así se pretende obtener un cierto parámetro que relacione directamente los niveles de asistencia y resistencia con el esfuerzo efectuado por el paciente, lo que permitirá una mejor comparación con otros diseños de asistencia. Del mismo modo esta experiencia práctica enriquece el proyecto Hyper ya que permite un acercamiento entre los distintos grupos involucrados de forma que se obtenga el mejor diseño en la órtesis robótica.

5.4.- Procedimiento Experimental

Durante los dos días se llevaron a cabo tres experimentos siendo uno de ellos parte de la tesis doctoral del ingeniero Luis Amigo y los otros dos parte de este documento. Los resultados presentes en este anexo solo son del experimento 1 debido a que el experimento 2 pertenece a otro proyecto y durante en el experimento 3 se sufrieron algunos problemas.

A continuación se explican los tres experimentos.

5.4.1.- Experimento 1: Ejercicio de Flexión

Este experimento se realizó sobre 3 sujetos y se repitió dos veces: una a las 11:00 horas y otra a las 17:00 horas. La prueba consistía en colocar el brazo derecho sobre un soporte debidamente ajustado al sujeto y luego, con el brazo introducido en el guante del robot Kuka, realizar una flexión de bíceps (figura 5.3) de 0º con el brazo estirado y a 120º con el brazo flexionado. Este ejercicio se repetía para distintos niveles de asistencia (0, 25, 50, 75 y 100) y resistencia (25, 50, 75 y 100) donde el segundo llegaba a un máximo de 30N. Cada una de estas series estaban formadas por diez repeticiones.

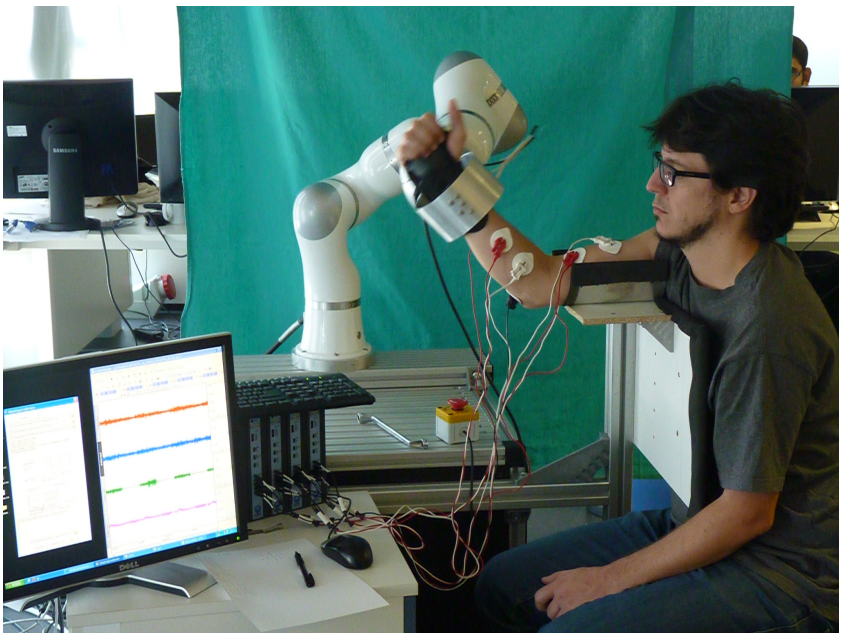

Figura 5.4: Uno de los sujetos realizando un ejercicio de flexión

5.4.2.- Experimento 2: Modificación del Arco de flexión

Este experimento es el diseñado por el ingeniero D. Luis Amigo. Debido a que se formó parte del dicho experimento y se ayudo en su realización, se añade la descripción del mismo.

En esta sesión se efectuó de nuevo el ejercicio de flexión pero siempre con 0 en el nivel de asistencia. Sin embargo, a partir de una posición cómoda del sujeto, se cambiaba el punto del efector final (la muñeca del sujeto). Haciendo esto el arco que describía el sujeto con la muñeca al hacer la flexión cambiaba. Aquí se intenta estudiar como la posición cambia con respecto a la fuerza que ejerce el sujeto. Se probaron en total 16 posiciones distintas, cada una con flexiones de 15 repeticiones sobre 3 sujetos.

5.4.3.- Experimento 3: Ejercicio de Alcance

Este experimento fue programado in situ y consistía en que el sujeto estirase el brazo para coger un objeto en concreto (figura 5.4).

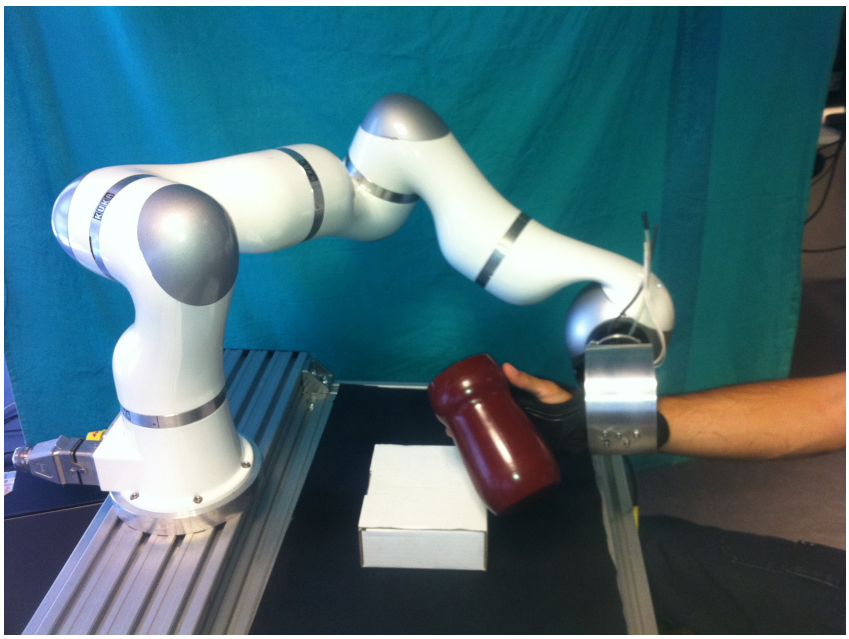

Figura 5.4: Uno de los sujetos realizando el ejercicio de alcance

En los otros experimentos se utilizó el sensor de medición en la muñeca para medir la fuerza del sujeto y activar la orden de movimiento. Debido a que el sensor estaba girado 45º respecto a esta fuerza, el sensor medía la fuerza en dos componentes, X e Y. No obstante, para el movimiento de alcance, esto no es tan simple. El movimiento de flexión solo podía realizarse con un grado de libertad no había rotación del codo debido al apoyo (figura 5.3), por lo tanto era muy fácil calcular la fuerza necesaria para activar el movimiento. En un movimiento de alcance el sujeto tiene más libertad para moverse (figura 5.4) y aunque el punto final del movimiento estaba preestablecido por el objeto, era más difícil volver a la posición inicial. El programa esperaba una fuerza en el sensor en cierta dirección, si el sujeto no conseguía moverse en la dirección de esta fuerza, el Kuka se detenía haciendo que el movimiento no fuese fluido. Por esta razón no se utilizarán los resultados de este experimento para realizar cualquier conjetura. No obstante los problemas que ocurrieron durante este experimento permitirán el desarrollo de muchas conclusiones y requisitos del robot final.

5.5.- Resultados

Se han obtenido todos los datos posibles del robot Kuka (apartado 5.2 Sistema) excepto en la primera sesión del experimento 1 donde no se obtuvieron los pares articulares ni la estimación de los mismos. Algunos de los datos tuvieron que ser anteriormente procesados. Las fuerzas del sensor fueron filtradas primero utilizando un filtro Savitsky Golay a recomendación de D. Luis Amigo, con un tamaño de ventana de 7 y un orden polinómico de 1. Las figuras 5.5 y 5.6 muestran el estado de una señal antes y después del procesado.

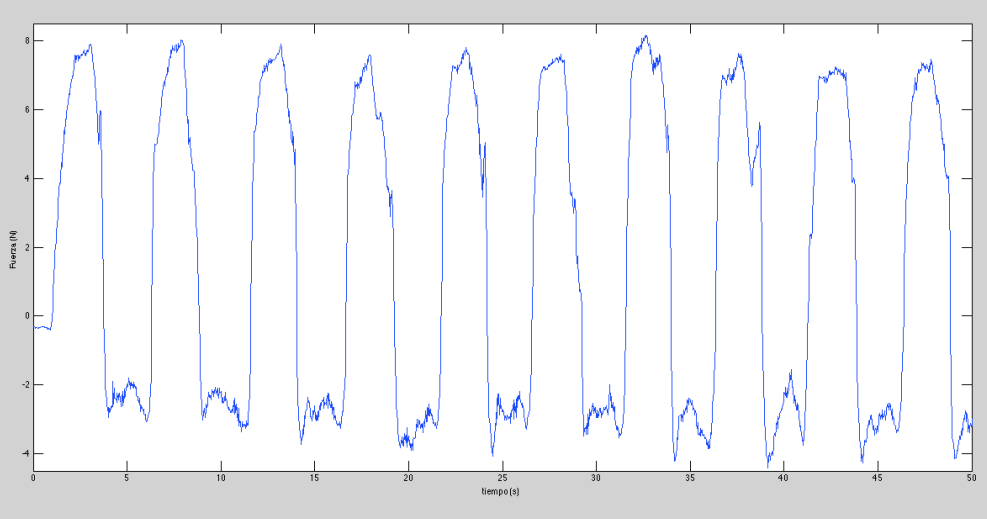

Figura 5.5: Señal de fuerza del sensor sin filtrar

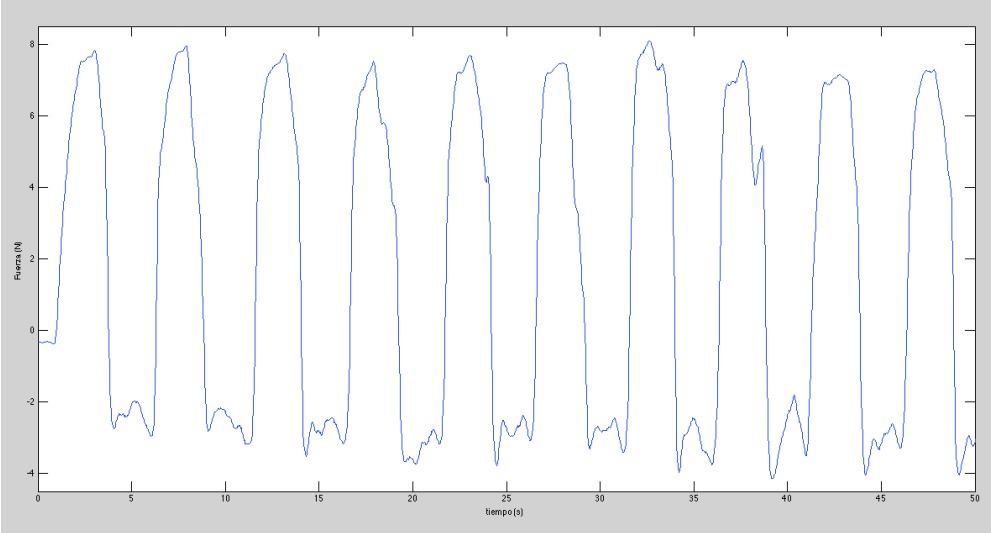

Figura 5.5: Señal de fuerza del sensor tras el filtrado

Las coordenadas cartesianas que nos da el robot son una matriz de rotación y otra de translación que se puede ver más abajo. Con estos datos y la formula inferior se puede calcular el ángulo de flexión del codo.

$$\alpha = \frac{180}{\pi} \tan^{-1} \frac{Y}{X} (^\circ)$$

Siendo Y e X las coordenadas 8 y 9 de las cartesianas que da el robot (elementos 3,1 y 3,2 de la matriz).

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{pmatrix}$$

En la matriz superior cada elemento representa el número de la señal asociada que da como salida el robot Kuka. Las señales 0-2, 4-6 y 8-10 representan una submatriz de rotación. Los elementos 3, 7 y 11 una de translación

Como se dijo anteriormente, el sensor de fuerza está girado 45º respecto a la dirección de movimiento, por lo que se gira el vector de fuerzas utilizando una matriz de rotación. Esto nos dará ahora una fuerza paralela al movimiento y dos perpendiculares. De estas dos últimas solo una está involucrada en el movimiento: la fuerza en el nuevo eje Y, que es la que produce un momento angular que permite la flexión. Multiplicando esta fuerza por la longitud del antebrazo del paciente, desde el codo a la muñeca, se obtiene un par. Finalmente, se integra este respecto al ángulo de flexión para obtener el trabajo realizado por el sujeto.

$$W = \int_0^{120} M \, d\phi \; (J)$$

Para la adquisición de datos se ha realizado un pequeño programa de Matlab que se adjunta al final de este anexo.

Si se repite el cálculo del trabajo realizado para cada sujeto y sesión, se obtiene un trabajo medio por sujeto y nivel de asistencia. En consecuencia, realizando la media, se puede obtener el trabajo medio relacionado con cada nivel de asistencia y resistencia.

La figura 5.7 resume los resultados para el nivel de asistencia.

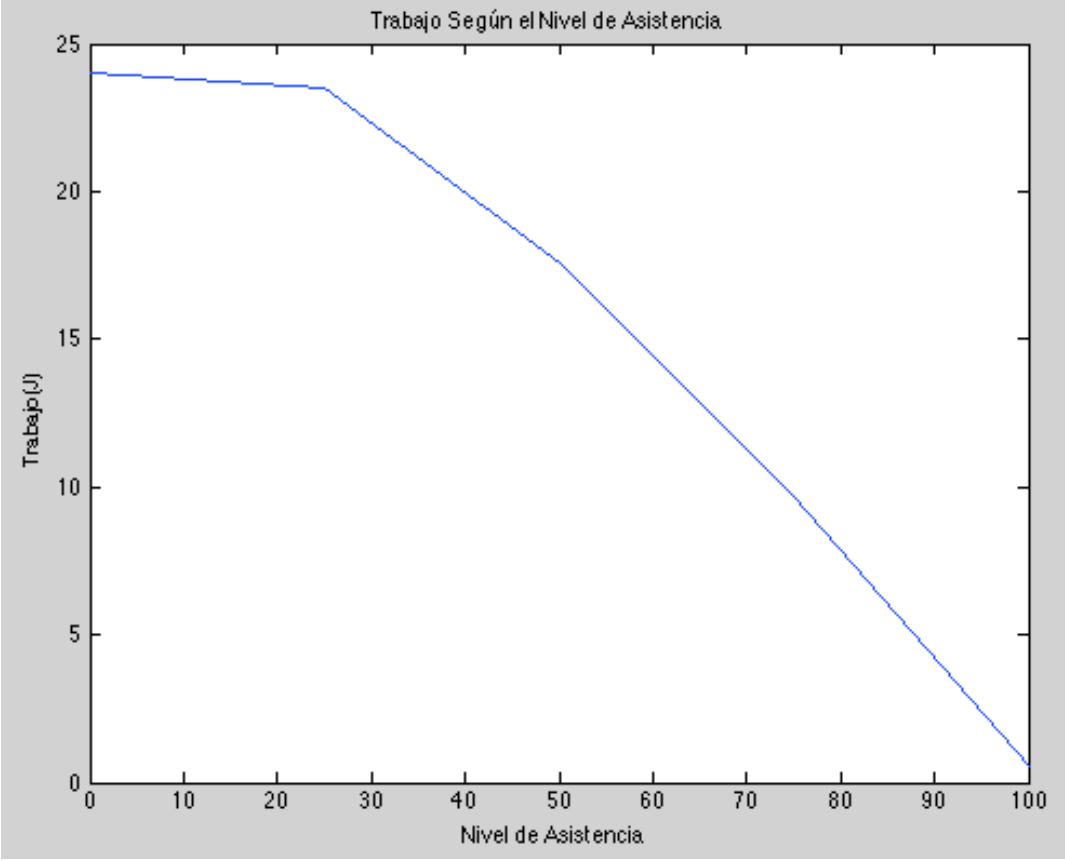

Figura 5.5: Trabajo medio realizado según el nivel de asistencia

Se puede observar en la gráfica que la relación entre el trabajo y el nivel de asistencia es prácticamente lineal entre los niveles 25 y 100% pero sin embargo se observa un gran cambio de pendiente entre los niveles 0 y 25%. Esto puede ser debido al cansancio acumulado durante los ejercicios. Hasta el 25% el robot asistía al sujeto, sin embargo en 0% el robot no hace esfuerzo alguno.

La figura 5.8 muestra el trabajo según el nivel de resistencia. Esta vez la linealidad se mantiene hasta el 50% de la resistencia pero vemos que después la pendiente baja hasta casi estar constante entre el 75% y el 100%.

De nuevo la explicación de esto puede ser debido al cansancio físico de los sujetos. A partir del 50% de resistencia ya costaba mover el robot por lo que al llegar al 100% los sujetos aplicaban la mínima fuerza necesaria para poder moverlo. No obstante, con un mayor número de sujetos, y repitiendo el experimento muchas veces más se podría hallar una recta que asociase directamente la asistencia con el trabajo

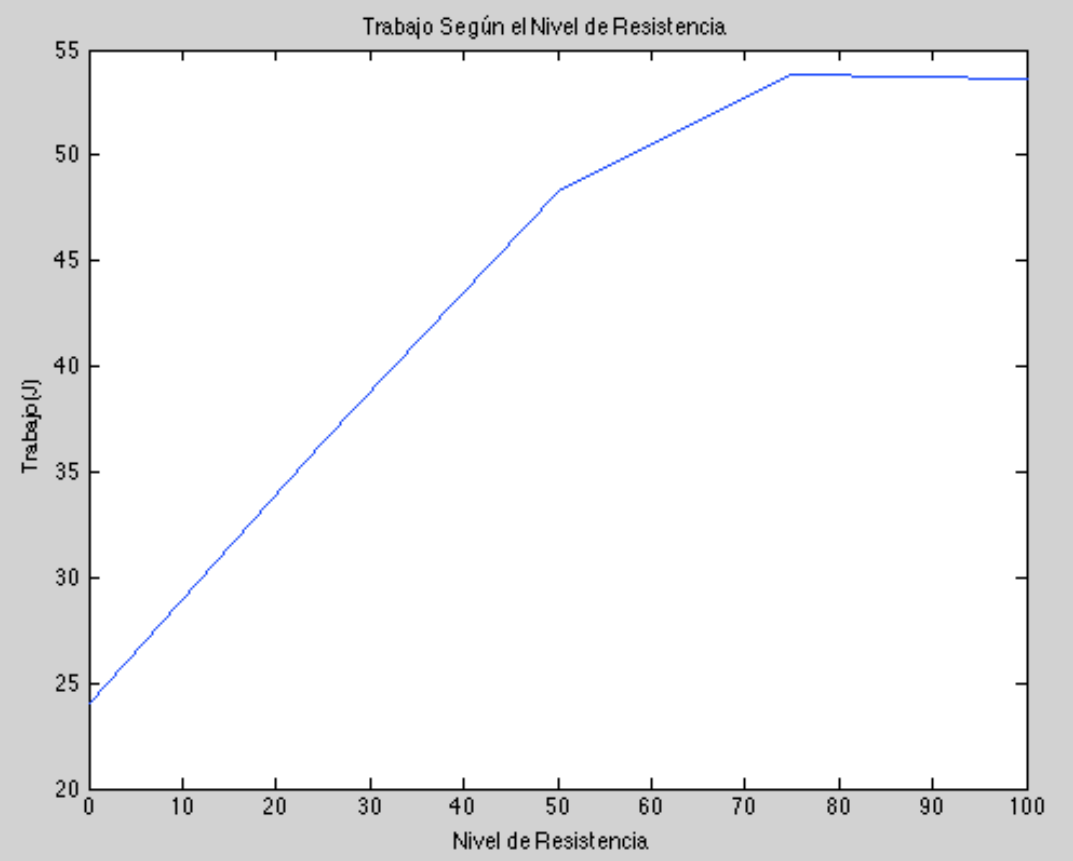

Figura 5.5: Trabajo medio realizado según el nivel de resistencia

5.6.- Conclusiones

Como se aprecia en los resultados el sistema programado funciona correctamente. Se ve claramente como el trabajo realizado por el sujeto aumenta al disminuir la asistencia o al aumentar la resistencia. Dicho esto no queda claro si los dos están proporcionalmente relacionados. En el sistema final es muy importante que el esfuerzo que realiza el paciente se incremente gradualmente a medida que se modifiquen estos valores, no se pueden permitir saltos, **la mejora ha de suceder paso a paso**.

Durante el ejercicio de flexión, en las pruebas de resistencia, hay un detalle que se debe tener en cuenta en la programación del HYPER: Un ejercicio de flexión como el que se ha probado (figura 5.3) se utiliza, normalmente, para ejercitar el bíceps. Al realizar la extensión, el bíceps se ejercita controlando la bajada del brazo, impidiendo que este caiga. No obstante cuando se realiza la extensión utilizando el robot Kuka este ofrece una resistencia que se opone a extender el brazo. Al hacer esto no se entrena el bíceps sino el tríceps. Se debería estudiar el caso, pues aunque se pretenda entrenar también el tríceps además del bíceps, el primero es mucho más pequeño que el segundo lo que hace que la carga puesta pueda ser demasiado grande para uno, en cuyo caso se podría producir una lesión, o demasiado pequeño para el otro, en cuyo caso no nos sirve el entrenamiento.

En el sistema que se ha diseñado en esta memoria, esto no ocurre. Al realizar la extensión el robot estará aplicando cierto par en la articulación del codo correspondiente, esto impedirá que el robot caiga con la aceleración de la gravedad por lo que se moverá mucho más lento. Al estar en modo resistencia, este par no será suficiente, en este sistema, el robot caerá y será un peso que el paciente deberá controlar. En este caso se entrenará el bíceps.

Otra cosa que debe estudiarse en profundidad es el establecer claramente donde empiezan y acaban los movimientos. Esta fue una de las razones por la que no funcionó la prueba de alcance, ya que era difícil para el sujeto establecer con exactitud donde comenzó poniendo el brazo. Aquí podría utilizarse un apoyo, de forma que el paciente siempre sepa donde ha de colocar el brazo o programar el robot de forma que le permita una cierta libertad al paciente sobre la trayectoria a seguir. Esto parece difícil a priori y se deberían estudiar detalladamente a sujetos haciendo movimientos de alcance (uno de los más utilizados en ejercicios de rehabilitación) para poder establecer una rutina programada. Si no lo que podrá ocurrir es que el paciente intente mover el brazo y el robot no se mueva o que lo intente mover hacia un lado y el robot siga la trayectoria prefijada. Lo segundo es bastante mejor que lo primero ya que el paciente tenderá a seguir al robot. Para que esto sea posible es recomendable que se utilice el EMG como señal de activación del robot y no un sensor que puede ser más sensible a los errores.

Algo que el sistema diseñado en esta memoria no ha tenido en cuenta es la seguridad, ya que se trata de una simulación. Es un capítulo que debe ser pensado detenidamente si este sistema se va aplicar al robot Hyper. Los compañeros del IBEC tienen un robot de paro de emergencia que detiene todo el movimiento en caso de fallo, algo muy importante para garantizar la seguridad de todos alrededor del robot. Además debería diseñarse un sistema de seguridad que detuviese los movimientos del robot en caso de que el paciente no se esforzase. Tal y como está diseñado el sistema de esta memoria, si el paciente deja de hacer esfuerzo, el robo no podría con el conjunto y podría llegar a caer. Llegado a este caso y para asegurar una terapia correcta de rehabilitación, lo mejor sería detener el movimiento hasta que el paciente haga intentos de movimiento.

Por último, de estos experimentos, queda claro que la órtesis final ha de ser ajustable. Del mismo modo que el robot Kuka se ajustaba al arco de cada persona y el apoyo se colocaba para que cada sujeto estuviese cómodo, el robot final ha de permitir un igual o mayor grado de ajuste. Los parámetros a regular serán: la altura del asiento, la altura del hombro del robot y las longitudes de brazo y antebrazo. Del mismo modo, todos los anclajes han de ser cómodos de forma que permitan una terapia continuada sin causar ningún daño al paciente.

Institut de bioenginyeria de Catalunya

Baldiri Reixac, 10-12 | 08028 Barcelona - Spain | Tel. +34(93)403 97 06 - Fax. +34(93)403 97 02

Barcelona, 6 de septiembre de 2012

Xavier Giralt Ingeniero del Departamento Robótica del IBEC

DECLARA

Que Adrián Rivero Pérez, alumno del Máster de Ingeniería Biomédica cursado en la universidad de Zaragoza, ha realizado una estancia en el departamento de robótica del Instituto de Bioingeniería de Cataluña (IBEC) durante los días 4 y 5 de septiembre de 2012. Durante dicha estancia el alumno ha realizado diferentes pruebas con el robot Kuka en relación al proyecto HYPER y su tesis final de máster *"Simulación del Control Exoesqueleto Mediante el Paradigma Assist-As-Needed"* colaborando además en pruebas requeridas por Ing. Luis Amigo.

Y para que conste, firmo el presente informe en Barcelona a 6 de Septiembre de 2012

Ing. Xavier Giralt
Robotics and Biomedical Imaging
IBEC

© Institut de Bioenginyeria de Catalunya | Baldiri Reixac, 10-12 | 08028 Barcelona - Spain | Tel. +34(93)403 97 06 - Fax. +34(93)403 97 02