PRÁCTICO 3

1.1. Explique cómo resolvería, usando dos técnicas distintas de control de movimientos restringidos, la tarea de atornillar con un robot tipo SCARA.

Control de Elasticidad

Se supone que las tareas incluyen la ubicación del tornillo sobre cualquier parte de la superficie (no necesariamente sobre el agujero), posteriormente, la ubicación sobre el orificio, el cambio de herramienta a un atornillador y finalmente la introducción del objeto hasta donde sea prudente. Por ende, la tarea se divide en pasos:

Primer paso

Definir la matriz de elasticidad **K** diagonal referida al marco de acomodación y dar la instrucción de movimiento en el eje Z. Para la ubicación del tornillo sobre la superficie, K_{f_z} deberá tener un valor alto para lograr alcanzar la superficie, con el resto de valores preferentemente nulos.

Segundo paso

La correcta ubicación sobre el orificio se logrará con valores bajos de K_{f_x} y K_{f_y}, manteniendo K_{f_z} grande hasta lograr la inserción de la punta del tornillo, con valores de $K_{\tau_x}, K_{\tau_y}, K_{\tau_z}$ preferentemente nulos.

Tercer paso

La acomodación del elemento a insertar previo al atornillado puede hacerse considerando valores bajos de K_{τ_x}, K_{τ_z}. Esto facilitará la reorientación del tornillo para una correcta aplicación de fuerza y torque posterior.

Cuarto paso

Una vez centrado adecuadamente, el valor de K_{τ_y} debe ser alto para introducir el tornillo en el orificio. En este momento, considerando un centrado adecuado del objeto, los valores de K_{f_y}, K_{τ_y} deben ser altos para mantener una presión y atornillado, mientras que el resto de valores deben continuar siendo bajos para una posible acomodación.

Quinto paso

El atornillado se ejecutará hasta que se detecte un umbral de fuerza que indique que se ha cumplido con el objetivo.

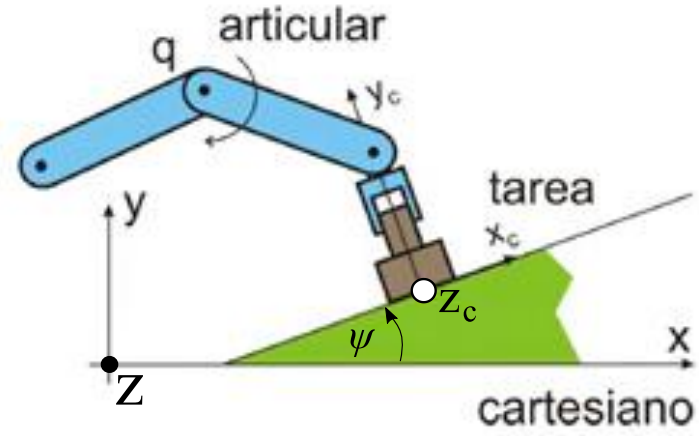
Control de impedancia

Para este control, se definen valores en la matriz de elasticidad **K** , matriz de inercia **I** y matriz de amortiguamiento **D** . Considerando los criterios:

k_j	Alta: Precisión de posicionamiento Baja: Pequeñas fuerzas de interacción
d_j	Alto: Alta disipación de energía
i_j	Alto: Comportamiento suave del extremo

Para este caso, los valores de K_x y K_y deben ser bajos para tener fuerzas de interacción pequeñas y acomodar el tornillo, mientras que K_z debe ser alto para lograr atornillar el objeto. Los valores de D_x, D_y, D_z deben ser bajos para que no exista una disipación de energía elevada. Finalmente, los valores de la diagonal de **I** deber ser altos para tener un comportamiento suave del extremo con el objetivo de acomodación adecuada del tornillo.

1.2. Para un robot rotacional de dos grados de libertad, considere la tarea de pulir sobre una superficie plana inclinada. En referencia al control híbrido, dé las restricciones naturales y artificiales, dibuje el esquema del sistema de control y dé la matriz de transformación de coordenadas.



Restricciones naturales: Dadas por el medio, generalmente restricciones dadas en algunas direcciones del espacio de trabajo.

$$v_y = 0$$
$$w_z = w_x = 0$$
$$f_z = f_x = 0$$
$$\tau_y = 0$$

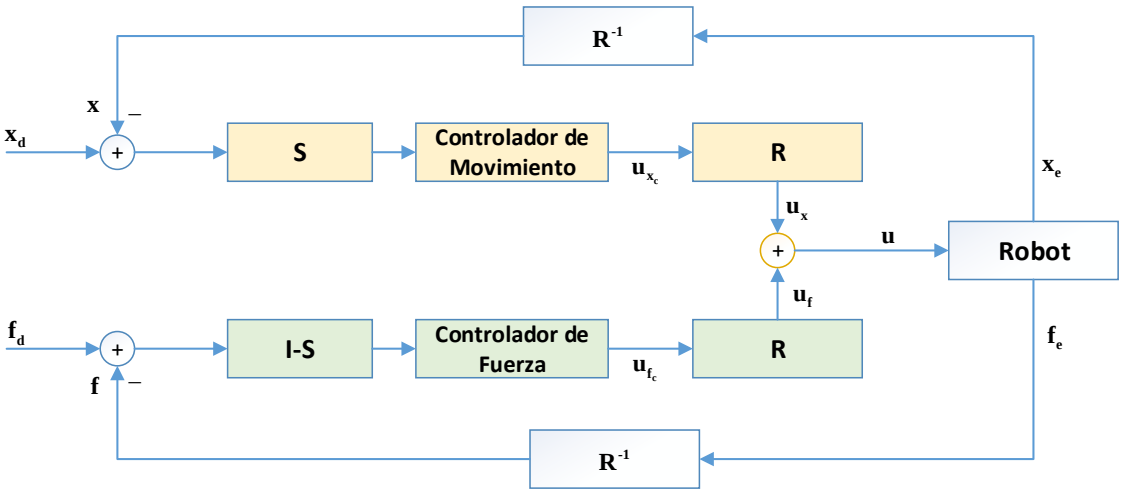
Restricciones artificiales: Dadas por la tarea que se desea cumplir, deben ser compatibles con las restricciones naturales.

$$v_x = c_1$$
$$v_z = c_2$$
$$f_y = c_3$$
$$w_y = c_4$$
$$\tau_x = \tau_z = 0$$

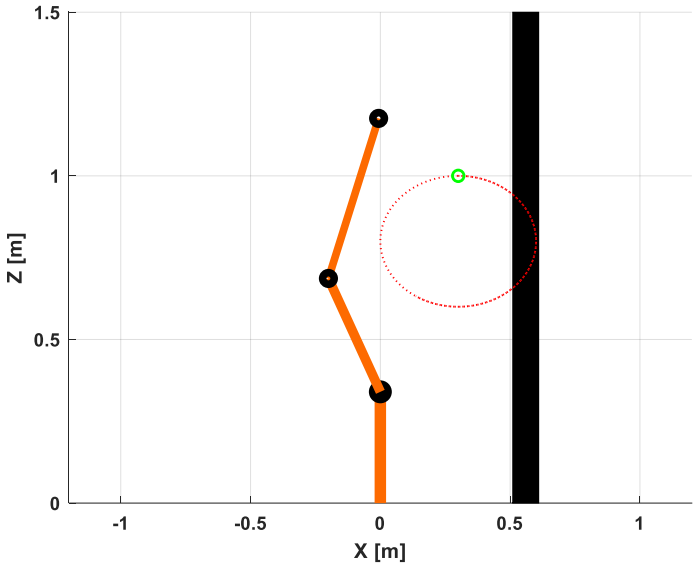
Matriz de transformación: Considerando que existe una rotación entre la referencia del robot y la referencia del espacio de trabajo, es requerido considerar la matriz de transformación que gira en sentido anti horario sobre el eje Z:

$$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Esquema del controlador híbrido:



1.3. Para el manipulador y el medio esquematizados en la figura 1, proponer un experimento simulado y diseñar un sistema de control de impedancia y otro de control híbrido. Considerar el modelo elástico del medio.



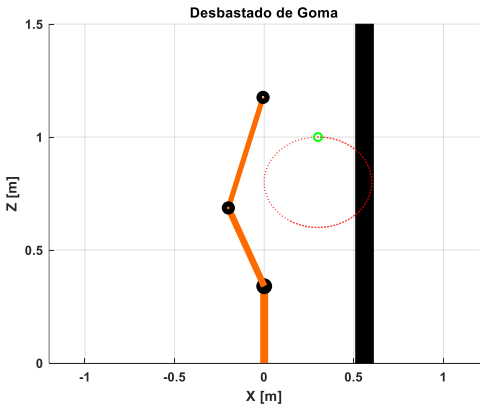
Sistema de control de impedancia

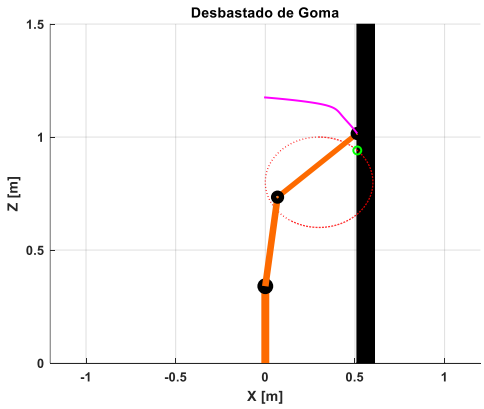
El ejercicio propone un control de impedancia para una tarea específica, considerando el modelo dinámico de un robot de dos grados de libertad. Se considera una pared ubicada a cierta distancia del robot con una trayectoria que producirá un desbastado sobre el objeto.

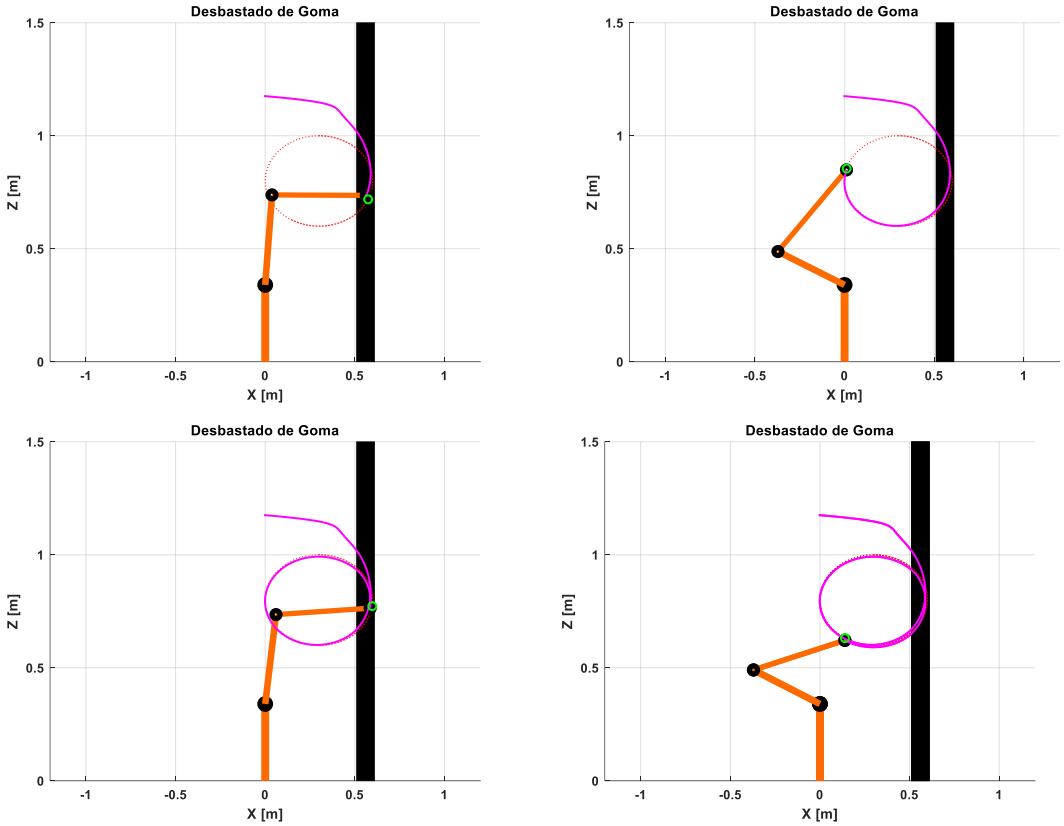
Dado que los parámetros de fuerza aplicada no se pueden medir directamente, se considera el modelo elástico del medio [1]. En este aspecto, se consideran tres tipos de materiales a través de su módulo de elasticidad longitudinal siendo: 7 Mpa para la goma, 600 Mpa para el tendón humano y 7000 Mpa para la madera. Seguidamente, se grafican los resultados de la ejecución donde se muestra los errores de posición, las aceleraciones dadas por el lazo de control exterior y las fuerzas dadas por el lazo de control interior para dada uno de los casos.

Desbastado para Goma

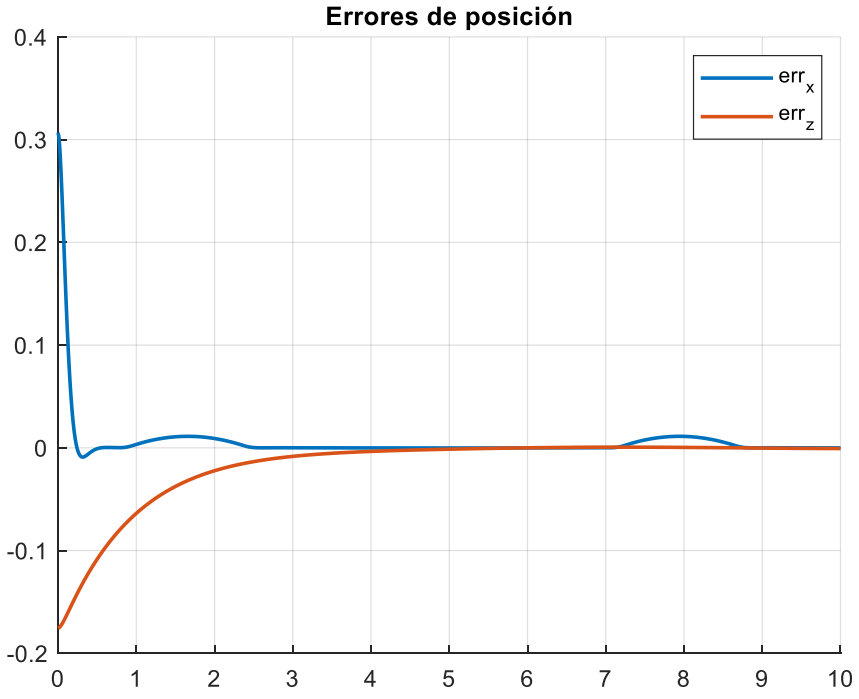
Simulación:

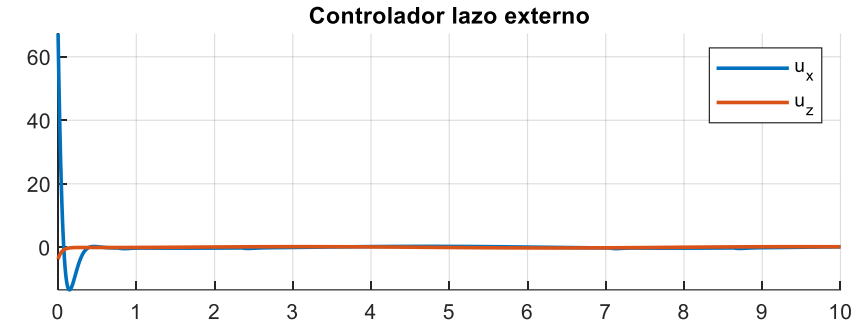


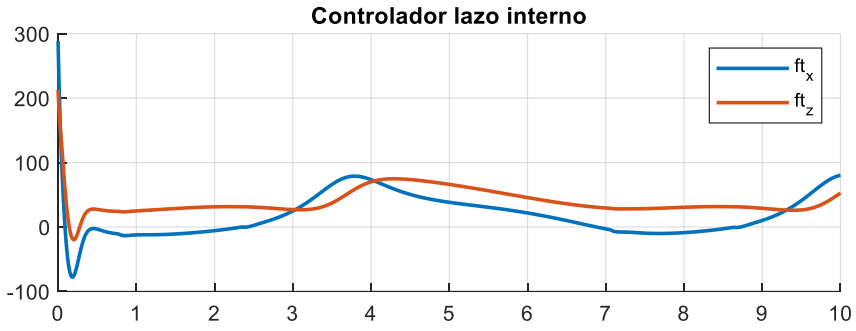




Graficación de resultados:

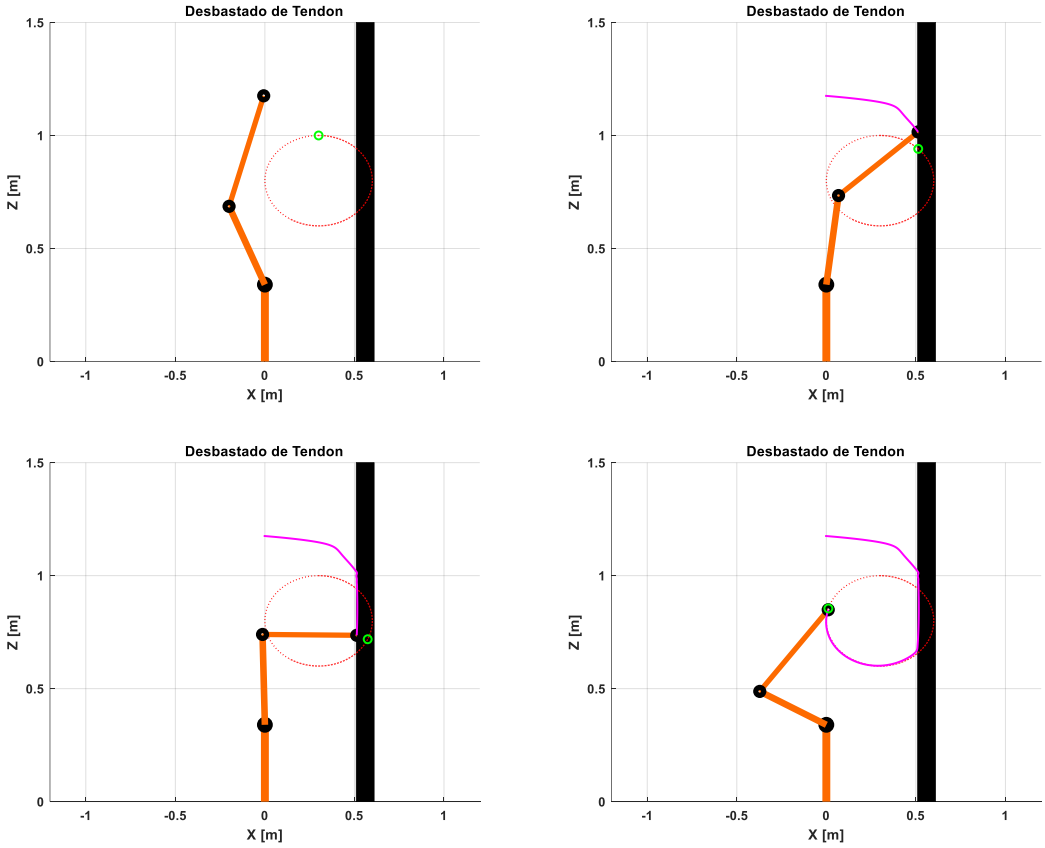


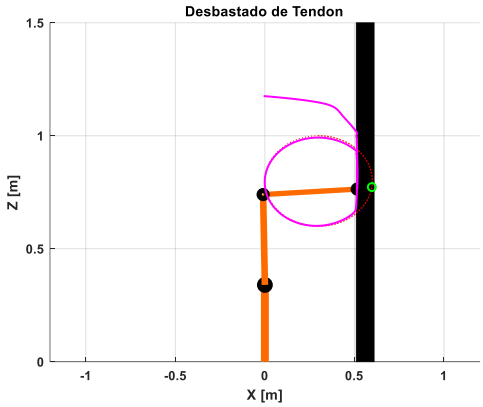


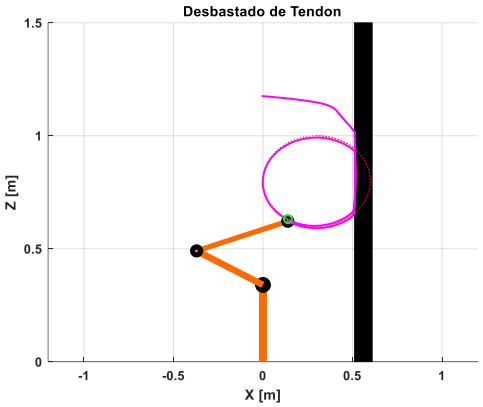


Desbastado para Tendón

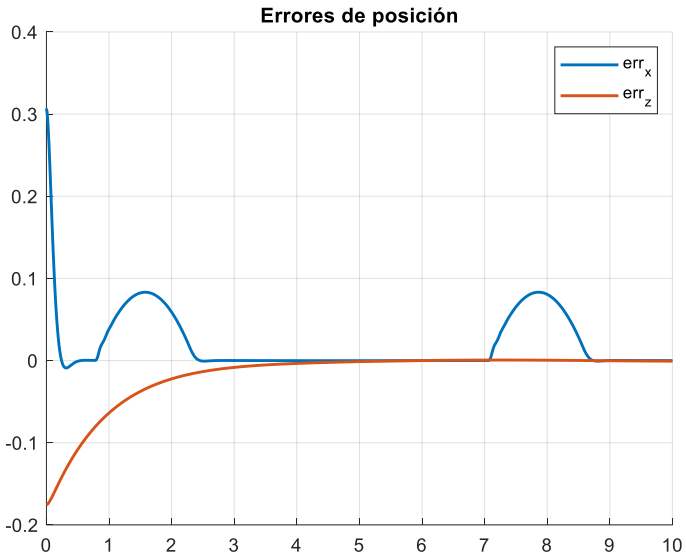
Simulación:

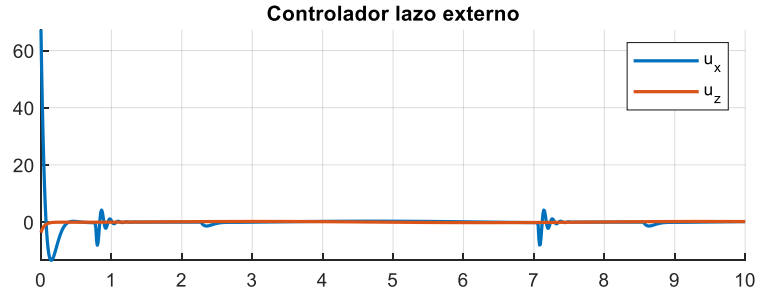


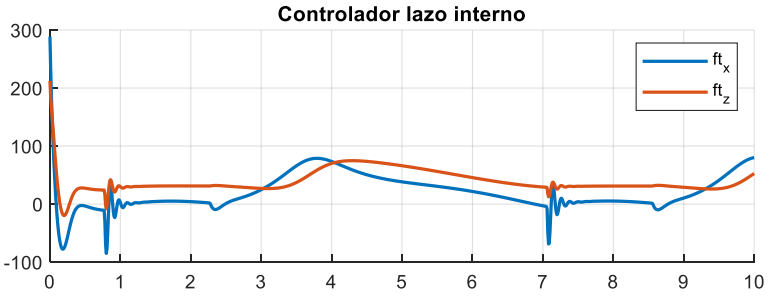




Graficación de resultados:

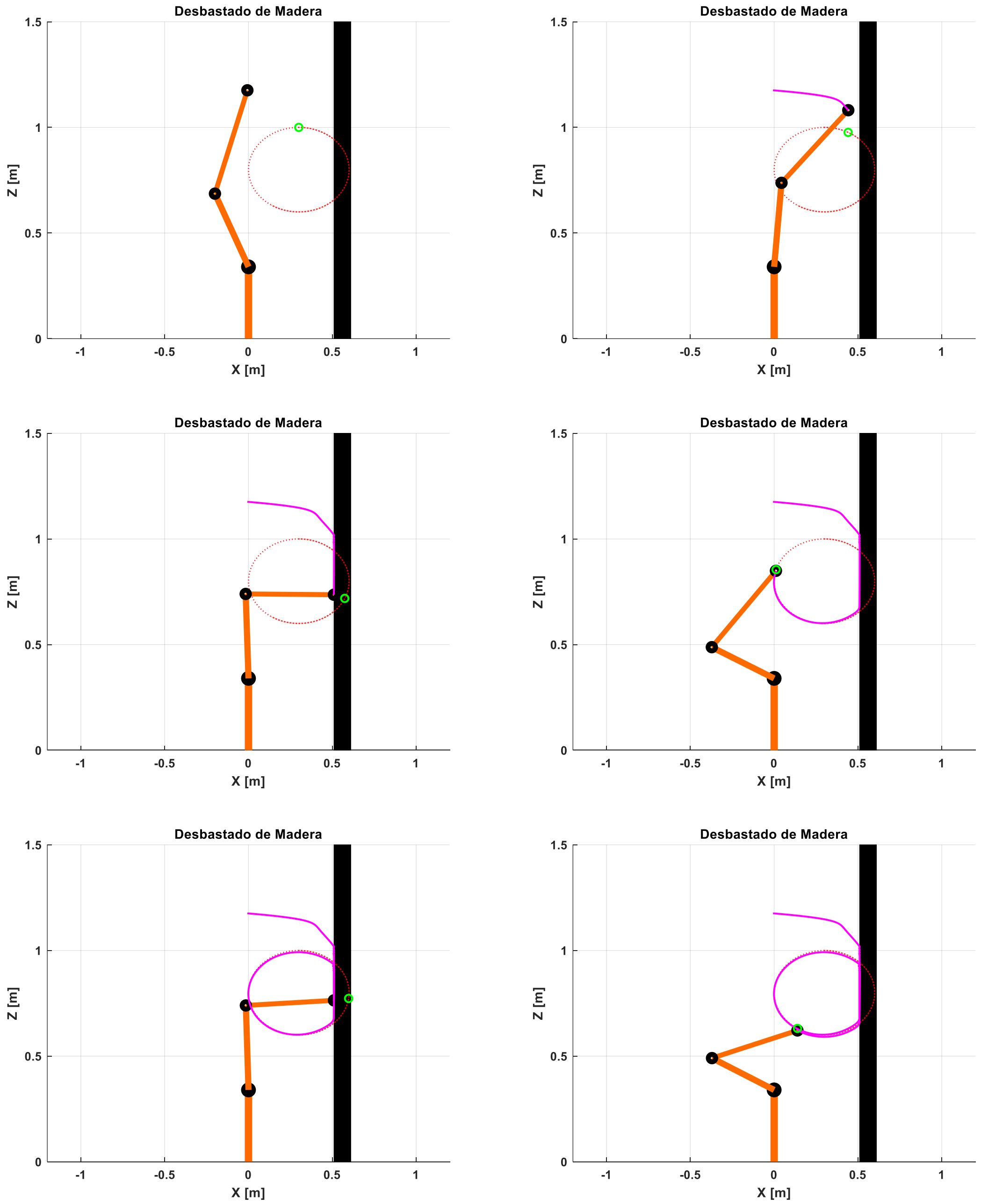




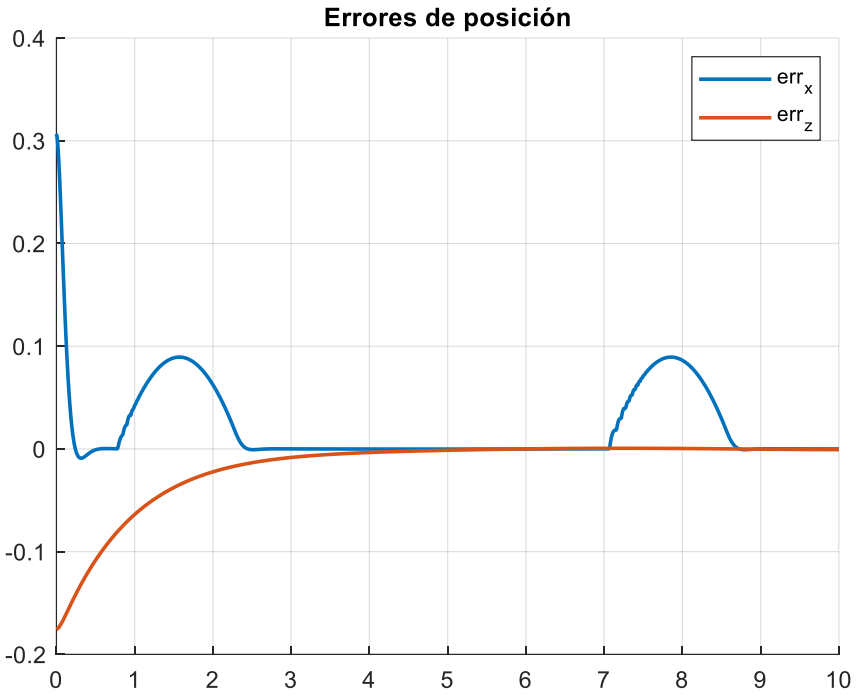


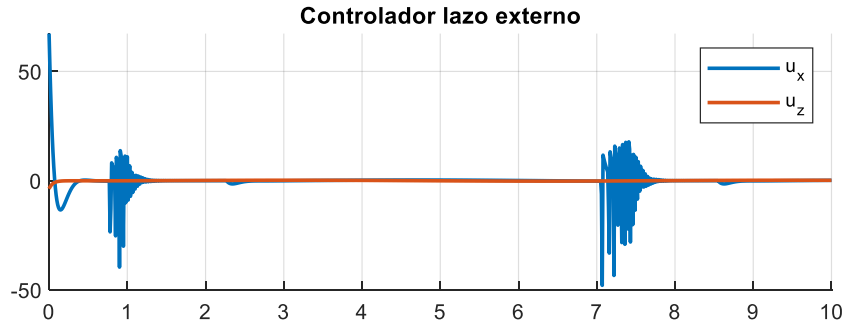
Desbastado para madera

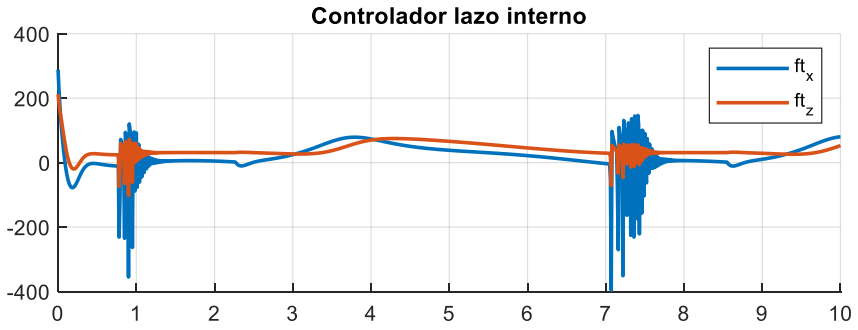
Simulación:



Graficación de resultados:







Programación

```
clc, clear, close all;
%% Parametros del Robot y valores fijos
m1= 5.067; m2= 3.152;
l1= 0.4; l2= 0.526; lb= 0.34;
g=9.807;
ts= 0.01; tf= 10;
t= 0:ts:tf;
```

```
% Definicion de Trayectorias
radio= 0.3;
xd= radio*sin(t)+0.3;       zd= 0.2*cos(t)+ 0.8;
xd_p= radio*cos(t);     zd_p= -0.2*sin(t);
xd_pp= -radio*sin(t);   zd_pp= -0.2*cos(t);

% Condiciones iniciales
q=[120*pi/180;-0.9];
q_p= [0;0];

% Asignacion de constantes para control de fuerza
Ke= diag([600;0]);        % Modulo de elasticidad longitudinal
(Mpa,megaPascales:kg/(m*s^2)), Goma: 7, cartilago:24, Madera:7000
I= 0.25*diag([1,1]);     % Provee comportamiento suave del extremo ante fuerzas de
contacto
D= 5.0*diag([1,1]);     % Valores alto: alta disipacion de energia
K= 5*diag([10,1]);    % Valores altos en direcciones que requieren alta precision de
posicion. Bajos en direcciones que req. pequeñas fuerzas de interaccion

k=1;
xEslabon1= l1*cos(q(1,k));
xEslabon2= xEslabon1 + l2*cos(q(1,k)+q(2,k));
zEslabon1= lb + l1*sin(q(1,k));
zEslabon2= zEslabon1 + l2*sin(q(1,k)+q(2,k));

posReal(:,k)= [xEslabon2;zEslabon2];
velReal(:,k)=[0;0];

% Posicion Obstaculo
posObstaculo= [0.51;0];

%% Programa general
for k=1:length(t)

    f(:,k)= [0; 0];
    if posReal(1,k)>posObstaculo(1)            % El obstáculo esta en la posicion 0.5,
tiene forma de barra
        f(:,k)= Ke*(posReal(:,k) - posObstaculo);
    else
        f(:,k)= [0; 0];
    end

    % Parametros del modelo dinámico en coordenadas articulares
    Mq= [(m1+m2)*l1^2 + m2*l2^2 + 2*m2*l1*l2*cos(q(2,k)),
m2*l2^2+m2*l1*l2*cos(q(2,k));
        m2*l2^2+m2*l1*l2*cos(q(2,k))                    , m2*l2^2];
    Cq= [-m2*l1*l2*sin(q(2,k))*q_p(2,k),  -m2*l1*l2*sin(q(2,k))*(q_p(1,k)+q_p(2,k));
        m2*l1*l2*sin(q(2,k))*q_p(1,k) , 0];
    Gq= [(m1+m2)*g*l1*cos(q(1,k)) + m2*g*l2*cos(q(1,k)+q(2,k));
        m2*g*l2*cos(q(1,k)+q(2,k)) ];

    % Jacobianita
    Ja=[-l1*sin(q(1,k))-l2*sin(q(1,k)+q(2,k))  -l2*sin(q(1,k)+q(2,k));
    l1*cos(q(1,k))+l2*cos(q(1,k)+q(2,k))    l2*cos(q(1,k)+q(2,k))];

    % Derivada de la Jacobianita (temporal)
    Ja_p= [-l1*cos(q(1,k))*q_p(1,k)-l2*cos(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k)), -
l2*cos(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k));...
    -l1*sin(q(1,k))*q_p(1,k)-l2*sin(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k)),        -
l2*sin(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k))];

    % Conversion del modelo dinámico a coordenadas cartesianas
    Mcart= inv(Ja)'*Mq*inv(Ja);
    Ccart= inv(Ja)'*Cq*inv(Ja) - inv(Ja)'*Mq*inv(Ja)*Ja_p*inv(Ja);
    Gcart= inv(Ja)'*Gq;

    % Calculo de errores
    err_p(:,k)= [xd_p(k); zd_p(k)] - velReal(:,k);
    err(:,k)= [xd(k); zd(k)] - posReal(:,k);

    % Calculo del controlador de fuerza
    u(:,k)= [xd_pp(:,k); zd_pp(:,k)] + inv(I)*(D*err_p(:,k) + K*err(:,k) - f(:,k));
% Calculo de u (Ley de control, lazo externo)
    ft(:,k)= Mcart*u(:,k) + Ccart*velReal(:,k) + Gcart + f(:,k);       % Calculo de
ftA (Ley de control, lazo interno)
```

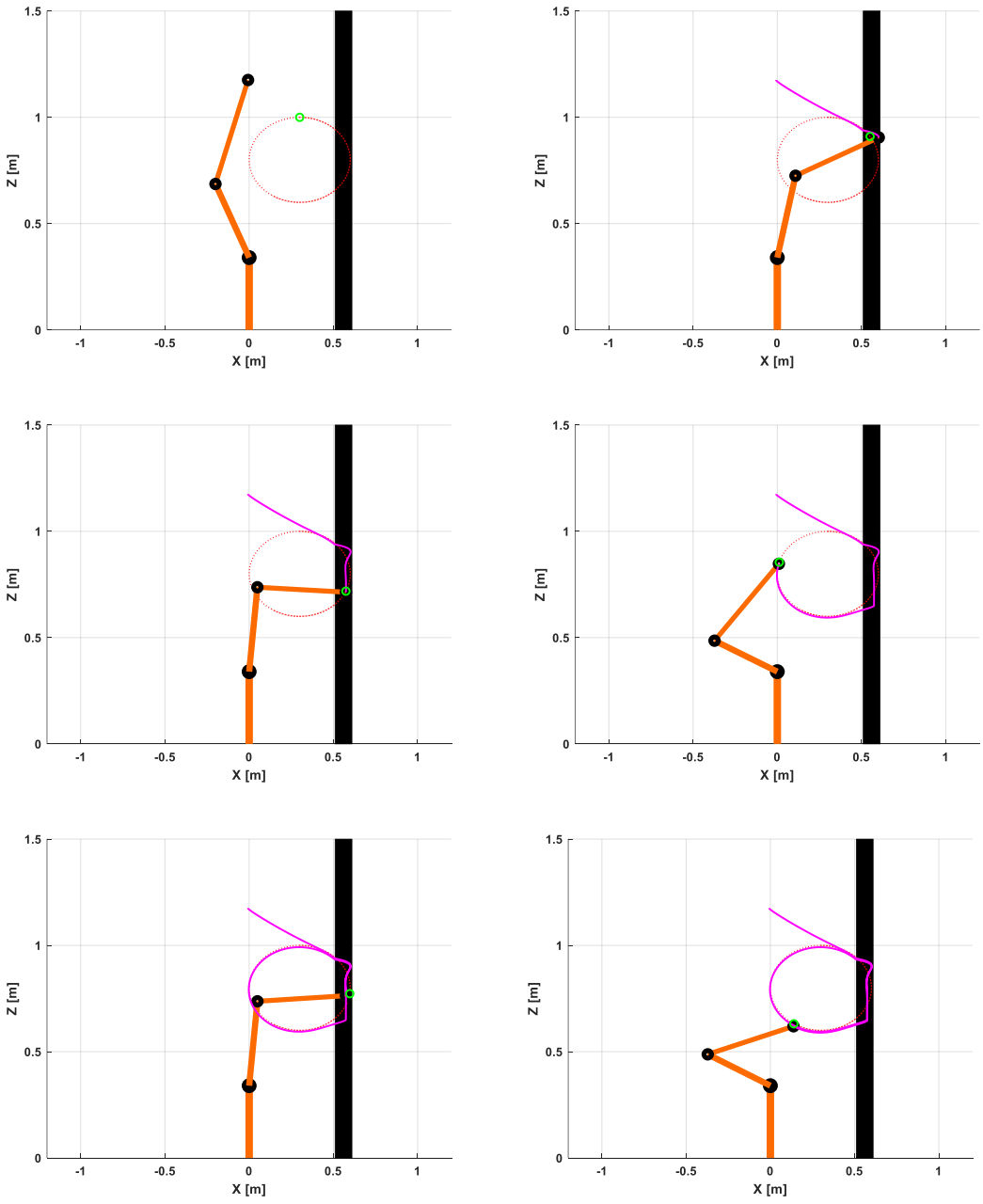
```
    % Aplicacion al robot
    accReal(:,k+1)= inv(Mcart)*(ft(:,k) - Ccart*velReal(:,k) - Gcart - f(:,k));
    velReal(:,k+1)= velReal(:,k) + accReal(:,k+1)*ts;
    posReal(:,k+1)= posReal(:,k) + velReal(:,k+1)*ts;

    % Obtención de q_p y q para la siguiente iteracion
    q_p(:,k+1)= inv(Ja)*velReal(:,k+1);
    q(:,k+1)= q(:,k) + q_p(:,k+1)*ts;
end

%% Animacion
close all
qr= q;
% Calculo de las posiciones de los eslabores
xEslabon1= l1*cos(qr(1,:));
xEslabon2= xEslabon1 + l2*cos(qr(1,:)+qr(2,:));
zEslabon1= lb + l1*sin(qr(1,:));
zEslabon2= zEslabon1 + l2*sin(qr(1,:)+qr(2,:));
slab1=[]; slab2=[]; slab=[]; slab3=[]; slab4=[];

figure(11)
hold on, grid on
patch([posObstaculo(1) posObstaculo(1) posObstaculo(1)+0.1 posObstaculo(1)+0.1],[0 1.5
1.5 0],'black')
axis([-1.2 1.2 0 1.5])
plot(xd,zd,'r:','linewidth',1)
% Ploteo de la base
plot([0 0],[0 lb],'color',[253 106 0]/255,'linewidth',6)
plot(0,lb,'ok','linewidth',6)
% Animacion de lo que se ejecuta
for k= 1:length(xEslabon1)-1
    delete(slab1);delete(slab2);delete(slab);delete(slab3); delete(slab4)
    slab1= plot([0 xEslabon1(k)],[lb zEslabon1(k)],'color',[253 106
0]/255,'linewidth',5);
    slab2= plot([xEslabon1(k) xEslabon2(k)],[zEslabon1(k) zEslabon2(k)],'color',[253
106 0]/255,'linewidth',4);
    slab= plot([xEslabon1(k) xEslabon2(k)],[zEslabon1(k)
zEslabon2(k)],'ok','linewidth',4);
    slab3= plot([xEslabon2(1:k)],[zEslabon2(1:k)],'m','linewidth',1.5);
    slab4= plot(xd(k),zd(k),'og','linewidth',1.5);
    pause(0.02)
end
set(gca,'FontWeight','Bold')
xlabel('X [m]')
ylabel('Z [m]')
title('Desbastado de Tendon')

%% Graficacion de Resultados
figure(9),
hold on,grid on
plot(t,err','linewidth',1.5)
legend('err_{x}','err_{z}');
title('Errores de posición')
figure(10)
subplot(2,1,1)
hold on, grid on
plot(t,u','linewidth',1.5)
legend('u_{x}','u_{z}');
title('Controlador lazo externo')
subplot(2,1,2)
hold on, grid on
plot(t,ft','linewidth',1.5)
legend('ft_{x}','ft_{z}');
title('Controlador lazo interno')
```

Sistema de control híbrido

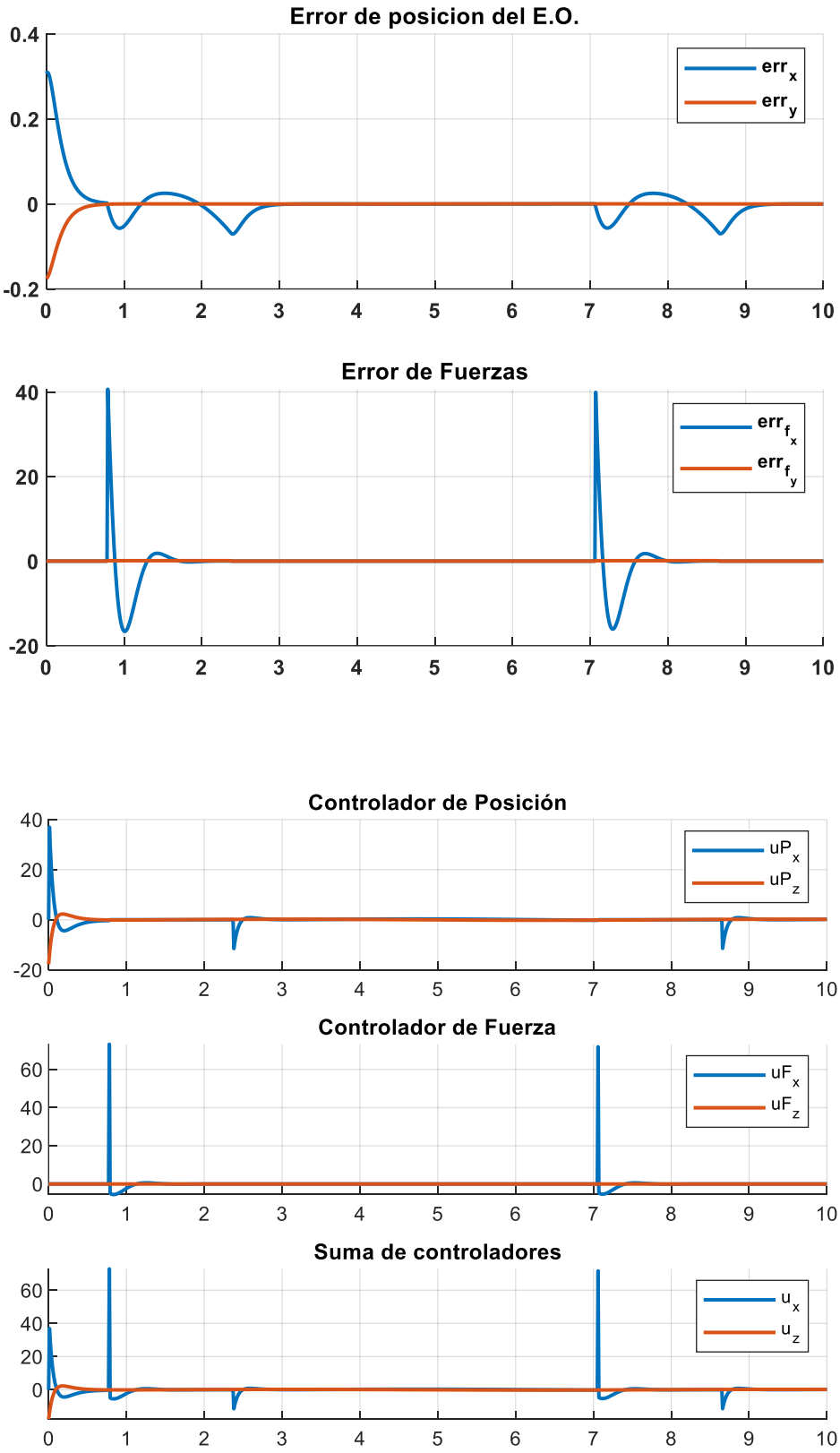
De manera similar al previo ejercicio, este problema se resuelve considerando la fusión entre el control de posición y el control de fuerza. Cuando el extremo operativo (que en el caso aplicativo incluirá la herramienta de desbastado) entre en la zona donde se encuentra el material, el control hibrido se aplicará a través de la inclusión de la matriz S. De forma similar, se asumen dos tipos de materiales a ser desbastados, con coeficientes de elasticidad de 600 Mpa para el tendón y 7000 para la madera. La trayectoria propuesta es considerada dado que la herramienta requerirá de enfriamiento o acomodación de cuchillas para evitar daños, aplicando el desbastado en una frecuencia adecuada.

Desbastado para tendón

Simulación:

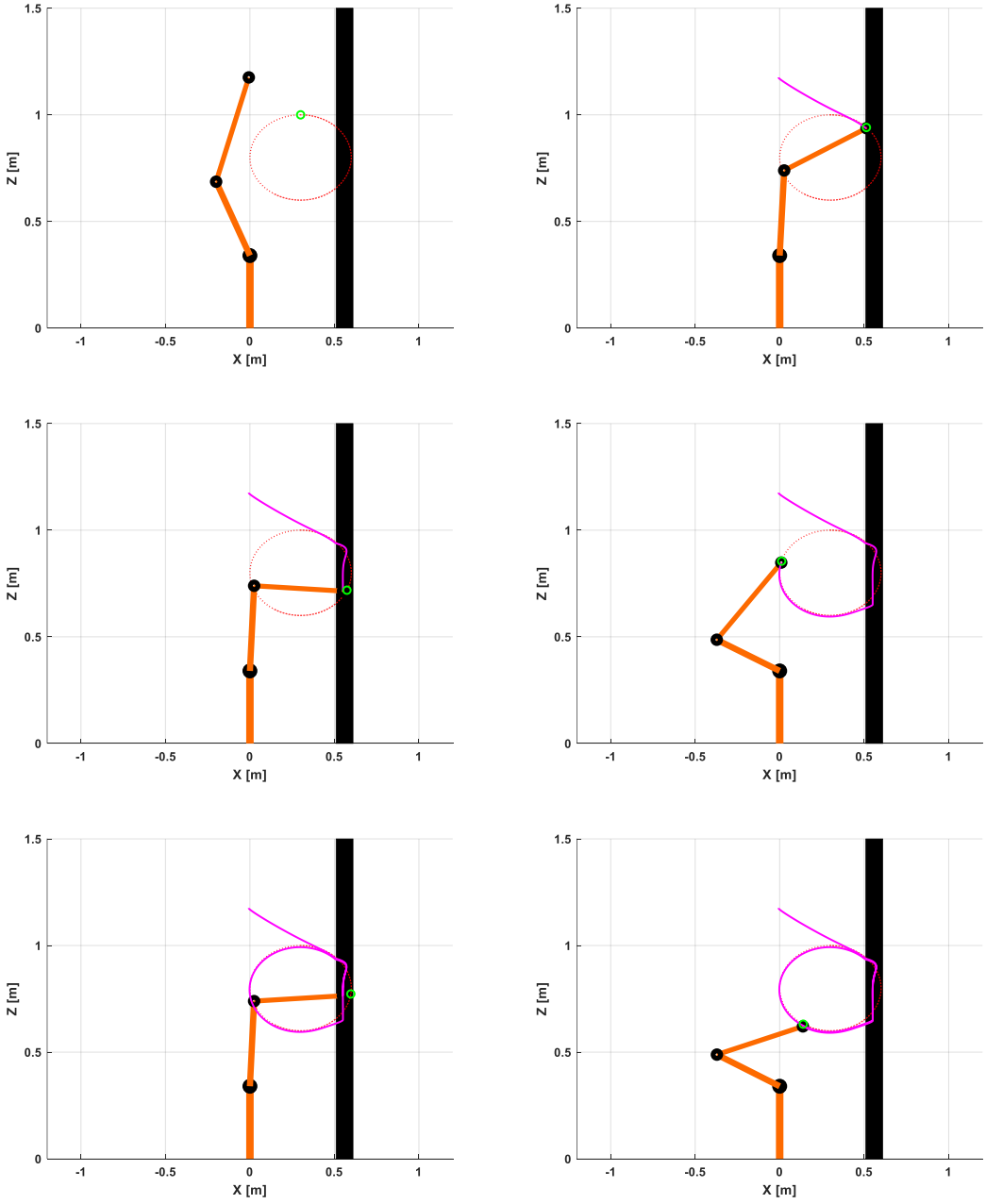


Graficación de resultados:

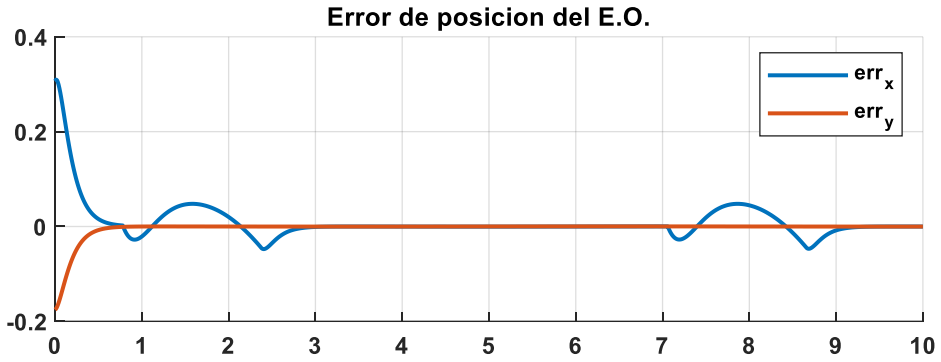


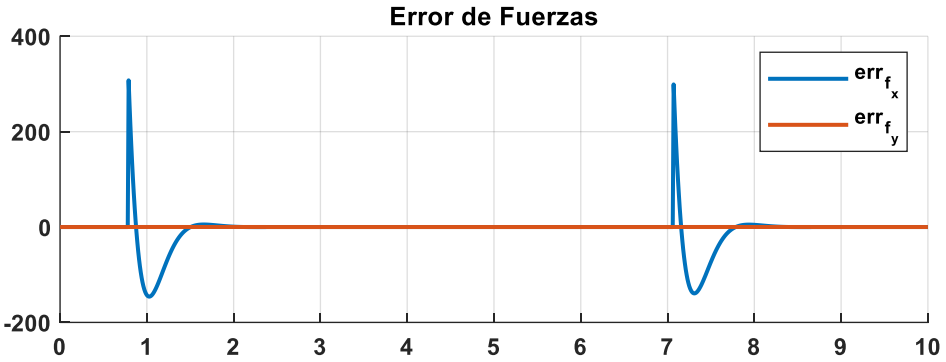
Desbastado para madera

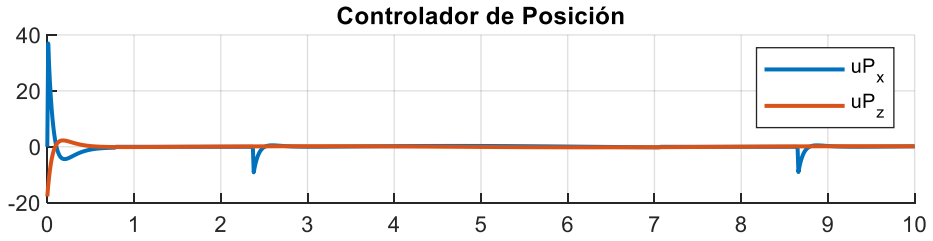
Simulación:

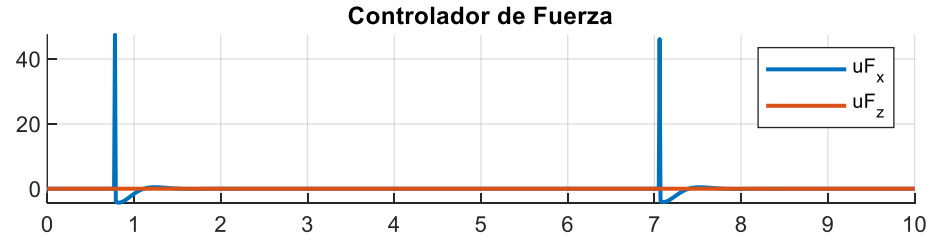


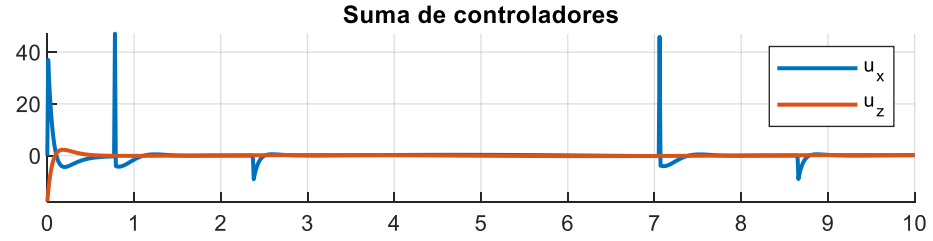
Graficación de resultados:











Programación

```
clc, clear, close all;
%% Parametros del Robot y valores fijos
m1= 5.067; m2= 3.152;
l1= 0.4; l2= 0.526; lb= 0.34;
g=9.807;
ts= 0.01; tf= 10;
t= 0:ts:tf;

% Definicion de Trayectorias
radio= 0.3;
xd= radio*sin(t)+0.3;       zd= 0.2*cos(t)+ 0.8;
xd_p= radio*cos(t);     zd_p= -0.2*sin(t);
xd_pp= -radio*sin(t);   zd_pp= -0.2*cos(t);

% Condiciones iniciales
q=[120*pi/180;-0.9];
q_p= [0;0];
k=1;
xEslabon1= l1*cos(q(1,k));
xEslabon2= xEslabon1 + l2*cos(q(1,k)+q(2,k));
zEslabon1= lb + l1*sin(q(1,k));
zEslabon2= zEslabon1 + l2*sin(q(1,k)+q(2,k));
posReal(:,k)= [xEslabon2;zEslabon2];
velReal(:,k)=[0;0];

% Asignacion de constantes para control de fuerza
Ke= diag([7000,0.01]);        % Modulo de elasticidad longitudinal
(Mpa,megaPascales:kg/(m*s^2)), Goma: 7, cartilago:24, Tendon:600, Madera:7000

% Posicion Obstaculo
posObstaculo= [0.51;0];
% posObstaculo= [1;0];

% Definicion de S: Permite seleccionar ejes de control de movimiento y ejes de control
de fuerza
S= [0 0;0 1];
Scoma= eye(2)-S;
Kv_x= 20*diag([1,1]); Kp_x= 100*diag([1,1]);
Kv_f= 10*diag([1,1]); Kp_f= 80*diag([1,1]);
fd= [300;0.1];
fd_pp= [0;0];
%% Programa general
for k=1:length(t)
    f(:,k)= [0; 0];
%     if posReal(1,k)>posObstaculo(1)            % El obstáculo esta en la posicion
0.5, tiene forma de barra
    if xd(k)>posObstaculo(1)
        f(:,k)= Ke*(posReal(:,k) - posObstaculo);
    else
        f(:,k)= [0; 0];
    end

    % Parametros del modelo dinámico en coordenadas articulares
    Mq= [(m1+m2)*l1^2 + m2*l2^2 + 2*m2*l1*l2*cos(q(2,k)),
m2*l2^2+m2*l1*l2*cos(q(2,k));
        m2*l2^2+m2*l1*l2*cos(q(2,k))                    , m2*l2^2];
    Cq= [-m2*l1*l2*sin(q(2,k))*q_p(2,k),  -m2*l1*l2*sin(q(2,k))*(q_p(1,k)+q_p(2,k));
        m2*l1*l2*sin(q(2,k))*q_p(1,k) , 0];
    Gq= [(m1+m2)*g*l1*cos(q(1,k)) + m2*g*l2*cos(q(1,k)+q(2,k));
        m2*g*l2*cos(q(1,k)+q(2,k)) ];

    % Jacobianita
    Ja=[-l1*sin(q(1,k))-l2*sin(q(1,k)+q(2,k))  -l2*sin(q(1,k)+q(2,k));
    l1*cos(q(1,k))+l2*cos(q(1,k)+q(2,k))    l2*cos(q(1,k)+q(2,k))];

    % Derivada de la Jacobianita (temporal)
    Ja_p= [-l1*cos(q(1,k))*q_p(1,k)-l2*cos(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k)), -
l2*cos(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k));...
    -l1*sin(q(1,k))*q_p(1,k)-l2*sin(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k)),        -
l2*sin(q(1,k)+q(2,k))*(q_p(1,k)+q_p(2,k))];

    % Conversion del modelo dinámico a coordenadas cartesianas
    Mcart= inv(Ja)'*Mq*inv(Ja);
    Ccart= inv(Ja)'*Cq*inv(Ja) - inv(Ja)'*Mq*inv(Ja)*Ja_p*inv(Ja);
```

```
    Gcart= inv(Ja)'*Gq;

    % Calculo de errores de posicion/velocidad y controlador de movimiento
    err_p(:,k)= [xd_p(k); zd_p(k)] - velReal(:,k);
    err(:,k)= [xd(k); zd(k)] - posReal(:,k);
    uXc(:,k)= S*[xd_pp(:,k); zd_pp(:,k)] + Kv_x*S*err_p(:,k) + Kp_x*S*err(:,k);    %
Calculo de u (Ley de control, lazo externo)

    % Calculo de errores de fuerza y controlador de fuerza
%     if posReal(1,k)>posObstaculo(1)
    if xd(k)>posObstaculo(1)
        errF(:,k+1)= fd - f(:,k);
        errF_p(:,k+1)= (errF(:,k+1) - errF(:,k))/ts;
        fXc(:,k)= inv(Ke)*(Scoma*fd_pp + Kv_f*Scoma*errF_p(:,k+1) +
Kp_f*Scoma*errF(:,k+1));
        S= [0 0;0 1];
    else
        fXc(:,k)=[0;0];
        errF(:,k+1)=[0;0];
        S= eye(2);
    end
%     uc(:,k)= [xd_pp(:,k); zd_pp(:,k)] + Kv_x*err_p(:,k) + Kp_x*err(:,k);    %
Calculo de u (Ley de control, lazo externo)

    % Suma de controladores
    uc(:,k)= uXc(:,k) + fXc(:,k);

    ft(:,k)= Mcart*uc(:,k) + Ccart*velReal(:,k) + Gcart + f(:,k);       % Calculo de
ftA (Ley de control, lazo interno)

    % Aplicacion al robot
    accReal(:,k+1)= inv(Mcart)*(ft(:,k) - Ccart*velReal(:,k) - Gcart - f(:,k));
    velReal(:,k+1)= velReal(:,k) + accReal(:,k+1)*ts;
    posReal(:,k+1)= posReal(:,k) + velReal(:,k+1)*ts;

    % Obtención de q_p y q para la siguiente iteracion
    q_p(:,k+1)= inv(Ja)*velReal(:,k+1);
    q(:,k+1)= q(:,k) + q_p(:,k+1)*ts;
end

%% Animacion
close all
qr= q;
% Calculo de las posiciones de los eslabores
xEslabon1= l1*cos(qr(1,:));
xEslabon2= xEslabon1 + l2*cos(qr(1,:)+qr(2,:));
zEslabon1= lb + l1*sin(qr(1,:));
zEslabon2= zEslabon1 + l2*sin(qr(1,:)+qr(2,:));
slab1=[]; slab2=[]; slab=[]; slab3=[]; slab4=[];

figure(11)
hold on, grid on
patch([posObstaculo(1) posObstaculo(1) posObstaculo(1)+0.1 posObstaculo(1)+0.1],[0 1.5
1.5 0],'black')
axis([-1.2 1.2 0 1.5])
plot(xd,zd,'r:','linewidth',1)
% Ploteo de la base
plot([0 0],[0 lb],'color',[253 106 0]/255,'linewidth',6)
plot(0,lb,'ok','linewidth',6)
% Animacion de lo que se ejecuta
for k= 800%:length(xEslabon1)-1
    delete(slab1);delete(slab2);delete(slab);delete(slab3); delete(slab4)
    slab1= plot([0 xEslabon1(k)],[lb zEslabon1(k)],'color',[253 106
0]/255,'linewidth',5);
    slab2= plot([xEslabon1(k) xEslabon2(k)],[zEslabon1(k) zEslabon2(k)],'color',[253
106 0]/255,'linewidth',4);
    slab= plot([xEslabon1(k) xEslabon2(k)],[zEslabon1(k)
zEslabon2(k)],'ok','linewidth',4);
    slab3= plot([xEslabon2(1:k)],[zEslabon2(1:k)],'m','linewidth',1.5);
    slab4= plot(xd(k),zd(k),'og','linewidth',1.5);
    pause(0.02)
end
set(gca,'FontWeight','Bold')
xlabel('X [m]')
ylabel('Z [m]')
```

```
%% Graficacion de Resultados
figure(9),
subplot(2,1,1)
hold on,grid on
plot(t,err','linewidth',1.5)
legend('err_x','err_y');
set(gca,'FontWeight','Bold')
title('Error de posicion del E.O.')

subplot(2,1,2)
hold on,grid on
plot(t,errF(:,1:end-1)','linewidth',1.5)
legend('err_{f_x}','err_{f_y}');
set(gca,'FontWeight','Bold')
title('Error de Fuerzas')

uc(:,k)= uXc(:,k) + fXc(:,k);
figure(10)
subplot(3,1,1)
hold on, grid on
plot(t,uXc','linewidth',1.5)
legend('uP_{x}','uP_{z}');
title('Controlador de Posición')
subplot(3,1,2)
hold on, grid on
plot(t,fXc','linewidth',1.5)
legend('uF_{x}','uF_{z}');
title('Controlador de Fuerza')
subplot(3,1,3)
hold on, grid on
plot(t,uc','linewidth',1.5)
legend('u_{x}','u_{z}');
title('Suma de controladores')
```