Anexos

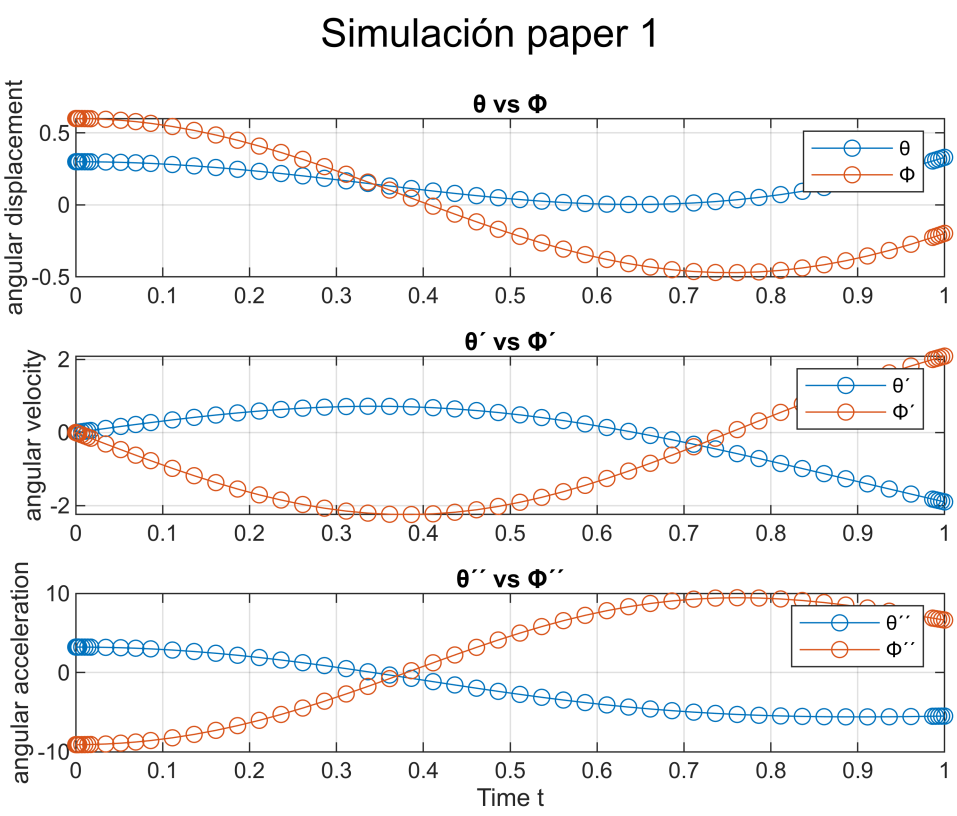
- **Modelo básico del primer paper**

Inicialmente, se realizó una simulación basada en el modelo planteado en el primer paper, el cual no tomaba en cuneta la pendiente. Este presentó curvas aproximadas a las presentadas en la sección de resultados, con la diferencia de que se cambiaron las condiciones iniciales de las velocidades angulares a 0.

Así mismo, no toma en cuenta los estados de transición presentados en los métodos del paper, pues se realizó este proceso en el modelo completo que se presenta más adelante.

```
clear all
M=80;
l=0.9;
s=0.00015;
g=9.81;


[t,y] = ode45(@vdp1,[0 1],[0.3; 0; 0.6; 0]);



dT2=(g/l).*sin(y(:,1)-y(:,3));
dp2=dT2+(y(:,2)).^2.*sin(y(:,3))-(g/l).*cos(y(:,1)-y(:,3)).*sin(y(:,3));


subplot(3,1,1)
plot(t,y(:,1),'-o',t,y(:,3),'-o')
grid on
title('θ vs Φ');

ylabel('angular displacement');
legend('θ','Φ')

subplot(3,1,2)
plot(t,-y(:,2),'-o',t,y(:,4),'-o')
grid on
title('θ´ vs Φ´');

ylabel('angular velocity');
legend('θ´','Φ´')

subplot(3,1,3)
plot(t,-dT2,'-o',t,dp2,'-o')
grid on
title('θ´´ vs Φ´´');
xlabel('Time t');
ylabel('angular acceleration');
legend('θ´´','Φ´´')
```

```
sgtitle('Simulación paper 1')
```



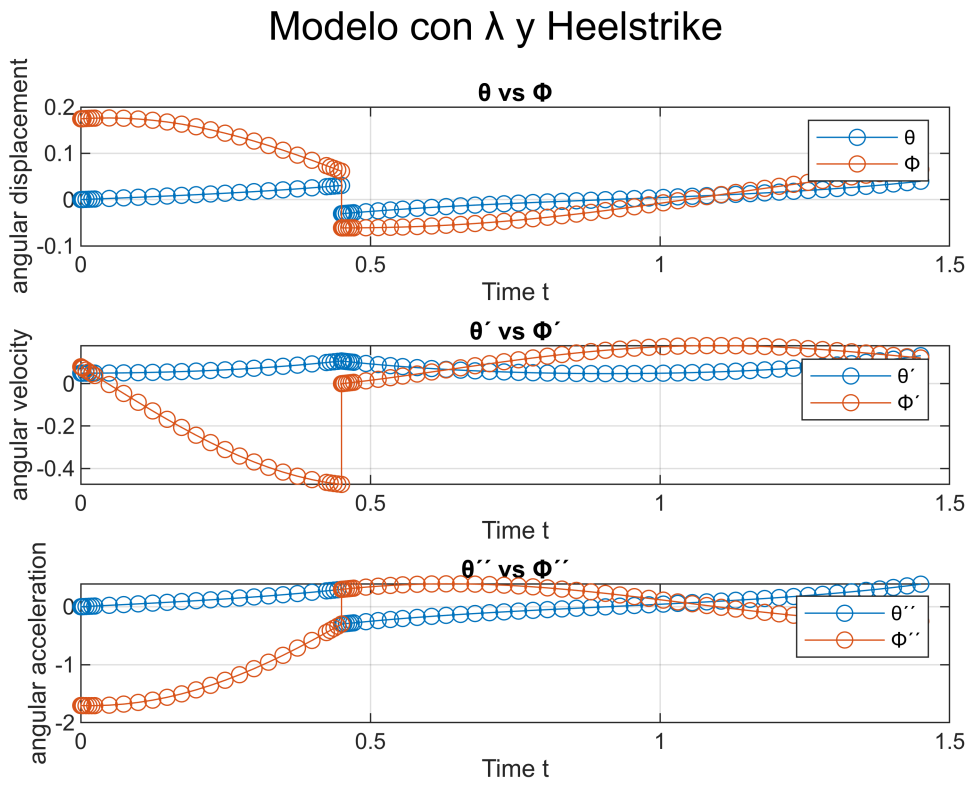
- **Modelo mejorado del segundo paper**

Ahora bien, se planteó un modelo más complejo con las especificaciones dadas en el segundo paper, así mismo se crearon funciones que simulaban el evento del heelstrike. Sin embargo, los modelos presentan saltos en la gráfica, los cuales de hecho tienen sentido con la matriz de transición mostrada en los papers, por lo que determinamos que probablemente sería necesario algún factor de corrección que no muestran en los artículos.

```
clear all
% Parámetros del modelo
g = 9.81;   % Aceleración gravitacional (m/s^2)
l = 1;      % Longitud de las piernas (m)
lambda = 0.00015;

% Condiciones iniciales
theta0 = 0;    % Ángulo inicial de la pierna de apoyo (rad)
theta_dot0 = 0.05 ;  % Velocidad angular inicial de la pierna de apoyo (rad/s)
phi0 = deg2rad(10);      % Ángulo inicial de la pierna de balanceo (rad)
phi_dot0 = 0.08;    % Velocidad angular inicial de la pierna de balanceo (rad/s)
x0 = [theta0; theta_dot0; phi0; phi_dot0];
```

```
% Límites de los ángulos de las piernas
theta_max = pi/4;   % Ángulo máximo de la pierna de apoyo y la normal (45 grados)
theta_min = -pi/4;  % Ángulo mínimo de la pierna de apoyo y la normal(-45 grados)
phi_max = pi/2;     % Ángulo máximo entre piernas (90 grados)
phi_min = -pi/2;    % Ángulo mínimo entre piernas (-90 grados)

% Función del modelo de PFW
modelo = @(t, x) pfw_modelo(t, x, g, l, lambda);

% Tiempo de simulación
tspan = [0 1];   % Tiempo inicial y final de la simulación (s)

% Evento de heelstrike
footEvent = @(t, x) heelstrike_evento(t, x, theta_max, theta_min, phi_max, phi_min);

% Opciones para la simulación
options = odeset('Events',footEvent);

% Simulación
[t, x, te, xe, ie] = ode45(@(t, x) pfw_modelo(t, x, g, l,lambda), tspan, x0, options);

% Matriz de transición
M = transicion_matriz(xe(end, 1));

% Simulación después de heelstrike
tspan = [te(end), te(end)+1];
x0 = M * xe(end, :)';
[t2, x2] = ode45(@(t, x) pfw_modelo(t, x, g, l, lambda), tspan, x0, options);

% Concatenar los resultados
t = [t; t2];
x = [x; x2];

%Aceleraciones
ddT = (g/l).*sin(x(:,1)-lambda);
ddP = ddT + (x(:,2)).^2.*sin(x(:,3))-(g/l).*cos(x(:,1)-lambda).*sin(x(:,3));

% Graficar los resultados
figure
subplot(3,1,1)
plot(t,x(:,1),'-o',t,x(:,3),'-o')
grid on
title('θ vs Φ');
xlabel('Time t');
ylabel('angular displacement');
legend('θ','Φ')

subplot(3,1,2)
plot(t,x(:,2),'-o',t,x(:,4),'-o')
grid on
```

```
title('θ´ vs Φ´');
xlabel('Time t');
ylabel('angular velocity');
legend('θ´','Φ´')

subplot(3,1,3)
plot(t,ddT,'-o',t,ddP,'-o')
grid on
title('θ´´ vs Φ´´');
xlabel('Time t');
ylabel('angular acceleration');
legend('θ´´','Φ´´')
sgtitle('Modelo con λ y Heelstrike')
```




```
function dydt = vdp1(t,y)
M=80;
l=0.9;
s=0.00015;
g=9.81;
gm=pi/12;
dydt = [y(2); (g/l)*sin(y(1)-y(3)); y(4); (g/l)*sin(y(1)-y(3))+(y(2))^2*sin(y(3))-(g/l)*cos(y(1
end
function dxdt = pfw_modelo(t, x, g, l,lambda)
    theta = x(1);
    theta_p = x(2);
    phi = x(3);
```

```
    phi_p = x(4);

    dxdt = zeros(4,1);
    dxdt(1) = theta_p;
    dxdt(2) = (g/l)*sin(theta-lambda);
    dxdt(3) = phi_p;
    dxdt(4) = dxdt(2) + (theta_p^2)*sin(phi) - (g/l)*cos(theta-lambda)*sin(phi);
end

function [value, isterminal, direction] = heelstrike_evento(t, x, theta_max, theta_min, phi_max
    value = x(3) - 2*x(1); % Condición para heelstrike: phi - 2*theta = 0
    isterminal = 1; % Terminar integración cuando se alcanza el evento
    direction = -1; % Buscar evento cuando la función cruza de positivo a negativo
    % Limitar los ángulos de theta y phi
    if x(1) > theta_max
        x(1) = theta_max;
        x(2) = 0;
    elseif x(1) < theta_min
        x(1) = theta_min;
        x(2) = 0;
    end
    if x(3) > phi_max
        x(3) = phi_max;
        x(4) = 0;
    elseif x(3) < phi_min
        x(3) = phi_min;
        x(4) = 0;
    end
end


function M = transicion_matriz(theta)
    M = [-1, 0, 0, 0; 0, cos(2*theta), 0, 0; -2, 0, 0, 0; 0, cos(2*theta)*(1-cos(2*theta)), 0,
end
```