Two Link Biped Model

March 10, 2023

The Equations of Motion of the two link biped model are derived according to the hybrid dynamics note. The following code is supposed to run at least in MATLAB 2018b:

```
close all; clear;

% In this two link model, there are two links with one end of each inter-connected as hip
% and the other end free as point feet.
% The swing leg is labeled link 1, and the stance leg is labeled link 2.
% Refer to Figure 3.4 in the westervelt2007feedback book.
% However, notation of foot 1 and 2 are swapped.

syms g % Gravitational acceleration.
syms m % Mass of each link.
syms I % Moment of inertial of each link.
syms l % Length of each link.
syms l_c % Length from the COM of each link to the distal end.
syms q_1 % Angle from the stance leg to the swing leg, counter-clockwise.
syms q_2 % Angle from the vertical up direction to the stance leg, clockwise.
syms dq_1 % Time derivative of q_1.
syms dq_2 % Time derivative of q_2.
syms th_1 % Angle from the vertical up direction to the swing leg, clockwise.
syms th_2 % Angle from the vertical up direction to the stance leg, clockwise.
syms dth_1 % Time derivative of th_1.
syms dth_2 % Time derivative of th_2.
syms p_1x % Horizontal position of the foot of swing leg.
syms p_1y % Vertical position of the foot of swing leg.
syms dp_1x % Time derivative of p_1x.
syms dp_1y % Time derivative of p_1y.
syms p_2x % Horizontal position of the foot of stance leg.
syms p_2y % Vertical position of the foot of stance leg.
syms dp_2x % Time derivative of p_2x.
syms dp_2y % Time derivative of p_2y.

q_s = [q_1; q_2]; % Generalized coordinates.
dq_s = [dq_1; dq_2]; % Time derivative of generalized coordinates.

q_e = [q_s; p_2x; p_2y]; % Extended generalized coordinates.
dq_e = [dq_s; dp_2x; dp_2y]; % Time derivative of extended generalized coordinates.
```

```
th_1 = - q_1 + q_2;
th_2 = q_2;
dth_1 = - dq_1 + dq_2;
dth_2 = dq_2;

p_2 = [p_2x; p_2y];
p_2c = p_2 + [l_c * sin(q_2); l_c * cos(q_2)];
p_1c = p_2 + [l * sin(q_2) + (l - l_c) * sin(q_1 - q_2); l * cos(q_2) - (l - l_c) * cos(q_1 -
p_1 = p_2 + [l * sin(q_2) + l * sin(q_1 - q_2); l * cos(q_2) - l * cos(q_1 - q_2)];

dp_1 = jacobian(p_1, q_e) * dq_e;
dp_1c = jacobian(p_1c, q_e) * dq_e;
dp_2c = jacobian(p_2c, q_e) * dq_e;

% KE - Kinetic Energy, PE - Potential Energy.
% 1 - swing leg, 2 - stance leg.
KE1 = simplify(m / 2 * (dp_1c.') * dp_1c + I / 2 * dth_1^2);
KE2 = simplify(m / 2 * (dp_2c.') * dp_2c + I / 2 * dth_2^2);
KE = KE1 + KE2;
PE1 = simplify(m * g * p_1c(2));
PE2 = simplify(m * g * p_2c(2));
PE = PE1 + PE2;

% Stance phase.
D_s = simplify(jacobian(jacobian(KE, dq_s).', dq_s));
C_s = simplify(jacobian(D_s * dq_s, q_s) - jacobian(D_s * dq_s, q_s).' / 2);
G_s = simplify(jacobian(PE, q_s).');
B_s = [diag(ones(1, length(q_s)-1)); zeros(1, length(q_s)-1)];

% Impact phase.
D_e = simplify(jacobian(jacobian(KE, dq_e).', dq_e));
E = simplify(jacobian(p_1, q_e));
J_Ff = simplify(-(E / D_e * E.') \ E * jacobian(dq_e, dq_s)); % F_f / \dot{q}_s^-
J_qedot = simplify(jacobian(dq_e, dq_s) + D_e \ E.' * J_Ff); % \dot{q}_e^+ / \dot{q}_s^-
R = [-1, 0; -1, 1];
Delta_qs = R; %Delta_{q_s}
Delta_qsdot = [R zeros(2, 2)] * J_qedot; % Delta_{\dot{q}_s}
```