Legged Robots

*Mini Project 1

Kaushik Ventakesh
Northeastern University
360 Huntington Ave,
Boston, MA 02115
email address or ORCID

Sebastian Wicke
Northeastern University
360 Huntington Ave,
Boston, MA 02115
email address or ORCID

Ali Abbas
Northeastern University
360 Huntington Ave,
Boston, MA 02115
email address or ORCID

Ibrahim Abubakar
Northeastern University
360 Huntington Ave,
Boston, MA 02115
abubakar.i@northeastern.edu

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

This report encompasses the forward kinematics of a three-link biped. The derivation of the forward kinematics allows for the eventual development of a controller that enables stable gait. The robot is assumed to have a no-slip condition and is only in contact with the ground at one point at a time. The forward kinematics were derived for the point mass of the two legs, the point mass of the hip, the point mass of the torso and the foot end location all of which are shown in Figure 1.

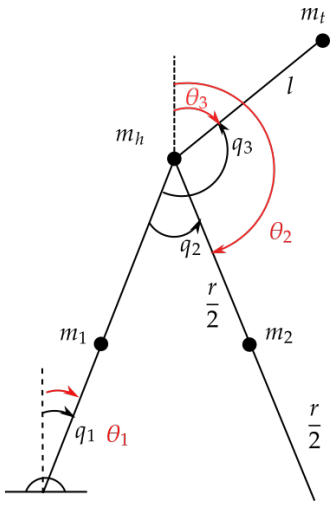

Fig. 1. Three Link Biped

II. Model Dynamics

A. Model Definition

For the three-link biped, there are 4 point masses depicted by the circles. Additionally, the angles can be defined as q_1 being the absolute angle of the stance leg, q_2 being the angle between the stance leg and the swing leg, and q_3 being the angle between the stance leg and the torso.We define these3 as the generalized co-ordinates of the system. The center of mass and the foot end position can then be defined as a function of the generalized co-ordinates. Additionally, the absolute angles for these links measured with respect to the positive vertical is taken to be as Θ

B. Forward Kinematics

The generalized co-ordinates is defined as vector $\boldsymbol{q}$,

$$\boldsymbol{q} = [q_1, q_2, q_3]$$

The absolute positions of the three links of the biped described in 1 Θ as a function of the generalized co-ordinates are shown below:

$$\Theta = \begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix} = \begin{bmatrix}q_1\\ \pi-(q_2-q_1)\\ \pi-(q_3-q_1)\end{bmatrix} \tag{1}$$

The positions of the 4 center of masses are as follows,

$$\boldsymbol{P}_{m_1} = \begin{bmatrix}\frac{r}{2} * sin(q_1)\\ \frac{r}{2} * cos(q_1)\end{bmatrix} \tag{2}$$

$$\boldsymbol{P}_{m_h} = \begin{bmatrix}r * sin(q_1)\\ r * cos(q_1)\end{bmatrix} \tag{3}$$

$$\boldsymbol{P}_{m_2} = \boldsymbol{P}_{m_h} + \begin{bmatrix}\frac{r}{2} * sin(\theta_2)\\ \frac{r}{2} * cos(\theta_2)\end{bmatrix} \tag{4}$$

$$\boldsymbol{P}_{m_t} = \boldsymbol{P}_{m_h} + \begin{bmatrix}l * sin(\theta_3)\\ l * cos(\theta_3)\end{bmatrix} \tag{5}$$

$$\boldsymbol{P}_{f_2} = \boldsymbol{P}_{m_h} + \begin{bmatrix}r * sin(\theta_2)\\ r * cos(\theta_2)\end{bmatrix} \tag{6}$$

$$\boldsymbol{P}_{cm} = \frac{\Sigma m_i \boldsymbol{P}_i}{\Sigma m_i} \tag{7}$$

where i is index of each of the point masses.

The velocities can then be calculated with the help of the Jacobian relating the generalized velocities to the absolute velocities.

$$\boldsymbol{v}_{m_h} = \frac{\partial \boldsymbol{P}_{m_h}}{\partial \boldsymbol{q}} \left(\frac{d\boldsymbol{q}}{dt}\right) \tag{8}$$

$$\boldsymbol{v}_{m_t} = \frac{\partial \boldsymbol{P}_{m_t}}{\partial \boldsymbol{q}} \left(\frac{d\boldsymbol{q}}{dt} \right) \tag{9}$$

$$\boldsymbol{v}_{m_1} = \frac{\partial \boldsymbol{P}_{m_1}}{\partial \boldsymbol{q}} \left(\frac{d\boldsymbol{q}}{dt} \right) \tag{10}$$

$$\boldsymbol{v}_{m_2} = \frac{\partial \boldsymbol{P}_{m_2}}{\partial \boldsymbol{q}} \left(\frac{d\boldsymbol{q}}{dt} \right) \tag{11}$$

$$\boldsymbol{v}_{cm} = \frac{\Sigma m_i \boldsymbol{v}_i}{\Sigma m_i} \tag{12}$$

C. Lagrangian and dynamic equations of pinned model

To solve for the Lagrangian, the kinetic and potential energy of the system must be solved. To solve for the kinetic energy, the equation below is used:

$$K_{tot} = \sum_{i=1}^{N} K_i = \frac{1}{2} m_i (\boldsymbol{v}_i)^2 \tag{13}$$

and the potential energy can be solved using the equation:

$$V_{tot} = \sum_{i=1}^{N} V_i = m_i g(\boldsymbol{p}_i) \tag{14}$$

This can then be used to solve for the Lagrangian using the equation:

$$\mathcal{L} = K - V \tag{15}$$

The equations of motion can be then be written in the form

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - G(\boldsymbol{q}) = B\boldsymbol{u}, \tag{16}$$

where, D is the mass inertia tensor, C is the Coriolis matrix and G is the gravity vector. B is the vector that maps the external forces on to the generalized co-ordinates and $\boldsymbol{u}$ is a vector of inputs. If the full states are then defined as $\boldsymbol{x} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]$, $\dot{\boldsymbol{x}}$ can be written in the following form,

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}}_s \\ D^{-1}(C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - G(\boldsymbol{q}) + B\boldsymbol{u}) \end{bmatrix} \tag{17}$$

this can then be written as:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$$

where,

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \dot{q}_s \\ D^{-1}(C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - G(\boldsymbol{q}) \end{bmatrix} \tag{18}$$

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} 0 \\ D^{-}1(B) \end{bmatrix} \tag{19}$$

To find $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ the D matrix must be solved for using the equation:

$$D = \sum_{i=1}^{N} m_i \frac{\partial \boldsymbol{P}_{m_i}}{\partial \boldsymbol{q}}^2$$

The Christoffel symbols

$$C_{k_j} = \sum_{i=1}^{\bar{N}} \frac{1}{2} \left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \dot{q}_i \tag{20}$$

The gravity vector $G(\boldsymbol{q})$ is calculated as;

$$G(\boldsymbol{q}) = \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

D. Unpinned model with augmented states

To model the unpinned model, we augment the generalized co-ordinates q of the pinned model with an arbitrary fixed point on the three link biped, denoted by $\boldsymbol{p}_e$. The new co-ordinates are then defined as q_e. The dynamic model of this unpinned model is found by steps followed in II-C.

```
% This .m file is used to symbolically derive:
%           the dynamics of the 3 link biped
%           impact map
%           controller
%           zero dynamics
% and write them onto function files
%
% Biped model:
%   D, C, G matrices are found using a defined forward position kinematics,
%   B matrix must be defined manually
%
% Impact map:
%   De, E. are derived using extended coordinates: p_e = [p_h; p_v]
%
% Controller:
%   The L2fh and LgLfh matrices used in feedback linearization are
%   symbolically derived
%
% Zero dynamics:
%   Vectors used in zero dynamics (eta2) are also derived
%

%-------------------------------------------------------------------------%
%%%% DCG matrices

syms q1 q2 q3 p_h p_v dp_h dp_v dq1 dq2 dq3 real
syms r m Mh Mt l g real

% Define parameters in a vector
params = [r,m,Mh,Mt,l,g];

% Include the util and autogen folder to use write_symbolic_term_to_mfile.m
% and export outputs to autogen folder
set_path

%Mh -  mass of hip, Mt - mass of torso, m - mass of legs
%l - length from hip to torso, r - length of legs

% Defining generalized coordinates:
% Angular positions:
%           q1: stance leg (absolute, w.r.t. y axis of
%           q2: swing leg (relative to q1)
%           q3: torso (relative to q1)
% Angular velocities dq/dt:
%           dq1: stance leg
%           dq2: swing leg
%           dq3: torso
q = [q1; q2; q3];
dq = [dq1; dq2; dq3];
```

```
% q1 is cyclic, and negative pre-impact using convention provided in the
% figure

theta1 = q(1);
theta2 = pi - q(1) - q(2);
theta3 = pi - q(3) + q(1);

theta= [theta1; theta2; theta3];

% Forward Kinematics - position of point masses
% hip
pMh = [r*sin(theta1) ; r*cos(theta1)];
% torso
pMt = pMh + [l*sin(theta3); l*cos(theta3)];
% stance leg
pm1 = [r*sin(theta1)/2 ; r*cos(theta1)/2];
% swing leg
pm2 = pMh + [r*sin(theta2)/2; r*cos(theta2)/2];
% center of mass
pcm = (Mh*pMh + Mt*pMt + m*pm1 + m*pm2)/(Mh + Mt + m + m);
% end of swing leg
% P2 = [r*sin(q(1)) + r*sin(q(2)-q(1)); r*cos(q(1))-r*cos(q(2)-q(1))];

P2 = pMh + [r*sin(theta2)/2; r*cos(theta2)/2];

%%


% Write positions to a file
% Inputs:
%       q
%       dq
%       params
%
% Outputs: Position vectors with x and y coordinates of position
%       pMh
%       pMt
%       pm1
%       pm2
%       pcm
%       P2
%
write_symbolic_term_to_mfile(q,dq,params,pMh,pMt,pm1,pm2,pcm,P2)


%%


% Velocities - found by taking partial derivative w.r.t. q, then multiply
% by dq/dt
```

```
vMh = jacobian(pMh,q)*dq;

vMt = jacobian(pMt,q)*dq;

vm1 = jacobian(pm1,q)*dq;

vm2 = jacobian(pm2,q)*dq;

vcm = (Mh*vMh + Mt*vMt + m*vm1 + m*vm2)/(Mh + Mt + m + m);

write_symbolic_term_to_mfile(q,dq,params,vMh,vMt,vm1,vm2,vcm)

% Write velocities to a file
% Inputs:
%       q
%       dq
%       params
%
% Outputs: Velocity vectors with x and y coordinates of velocity
%       vMh
%       vMt
%       vm1
%       vm2
%       vcm
%


% Kinetic energy
%%%%%%%%%%%%%%%%%% Compute kinetic energy of each component here %%%%%%%%%
K_Mh =  0.5*Mh*(vMh'*vMh);
K_Mt =  0.5*Mt*(vMt'*vMt);
K_m1 =  0.5*m*(vm1'*vm1);
K_m2 =  0.5*m*(vm2'*vm2);
% Total KE
K = K_Mh + K_Mt + K_m1 + K_m2;

% Potential energy
%%%%%%%%%%%%%%%%%% Compute potnetial energy of each component here %%%%%%%%
V_Mh = Mh*g*pMh(2);
V_Mt = Mt*g*pMt(2);
V_m1 = m*g*pm1(2);
V_m2 = m*g*pm2(2);
% Total PE
V = V_m1 + V_Mh + V_Mt + V_m2;


% Inertia matrix
%%%%%%%%%%%%%%%%%% Compute D matrix here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% D = (Mh + Mt + m + m)*jacobian(pcm,q)'*jacobian(pcm,q);

D = Mh*jacobian(pMh,q)'*jacobian(pMh,q) + Mt*jacobian(pMt,q)'*jacobian(pMt,q) + ...
    m*jacobian(pm1,q)'*jacobian(pm1,q) + m*jacobian(pm2,q)'*jacobian(pm2,q);

% Coriolis matrix
%%%%%%%%%%%%%%%%%% Coriolis matrix here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = max(size(q));
syms C
for k = 1:N
    for j = 1:N
        C(k,j) = 0*g;
        for i = 1:N
            C(k,j) = 0.5*(jacobian(D(k,j),q(j)) + jacobian(D(k,i),q(j)) - ...
                jacobian(D(i,j),q(k)))*dq(i);
        end
    end
end
C = simplify(C);

% Gravity matrix
%%%%%%%%%%%%%%%%%% Compute Gravity term here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

G = jacobian(V,q)';

%%%%%%%%%%%%%%%%%% What is control input matrix? %%%%%%%%%%%%%%%%%%%%%%%%

B = jacobian(theta, q)';

% Write 3 link model to file
% Inputs:
%       q
%       dq
%       params
%
% Outputs:
%       D: Inertia matrix
%       C: Coriolis matrix
%       G: Gravity matrix
%       B:
%
write_symbolic_term_to_mfile(q,dq,params,D,C,G,B)

%-------------------------------------------------------------------------%
%%%% Impact map

% Using same psotion vectors as above, but taking partial with respect to qe
% instead
```

```
% Extended configuration variables
p_e = [p_h; p_v];

qe = [q; p_h; p_v];
dqe = [dq; dp_h; dp_v];

% Extended position
pMh_e = pMh + p_e;
pMt_e = pMt + p_e;
pm1_e = pm1 + p_e;
pm2_e = pm2 + p_e;
P2e = P2 + p_e;

% Extended velocities
vMh_e = jacobian(pMh_e,qe)*dqe;
vMt_e = jacobian(pMt_e,qe)*dqe;
vm1_e = jacobian(pm1_e,qe)*dqe;
vm2_e = jacobian(pm2_e,qe)*dqe;

K_Mhe = 0.5*Mh*(vMh_e'*vMh_e);
K_Mte = 0.5*Mt*(vMt_e'*vMt_e);
K_m1e = 0.5*m*(vm1_e'*vm1_e);
K_m2e = 0.5*m*(vm2_e'*vm2_e);

Ke = K_m1e + K_Mhe + K_Mte + K_m2e;

% Extended inertia matrix
De = Mh*jacobian(pMh_e,qe)'*jacobian(pMh_e,qe) +
Mt*jacobian(pMt_e,qe)'*jacobian(pMt_e,qe) + ...
    m*jacobian(pm1_e,qe)'*jacobian(pm1_e,qe) +
m*jacobian(pm2_e,qe)'*jacobian(pm2_e,qe);

E = jacobian(pm2_e, qe);

% Partial of any point on biped, hip chosen in this case

%Check

dY_dq = jacobian(pMh,q);

% Write impact map to a file
% Inputs:
%       q
%       dq
%       params
%
% Outputs: Matrices needed to compile impact map
%       De: Extended inertia matrix
%       E:
%       dY_dq:
```

```
%
write_symbolic_term_to_mfile(q,dq,params,De,E,dY_dq)


%-------------------------------------------------------------------------%
%%%% For controller

% Vector fields

%%%%CHECKKK
fx = [dq; inv(D)*(-C*dq-G)];
gx = [zeros(3,3); inv(D)*B];
```