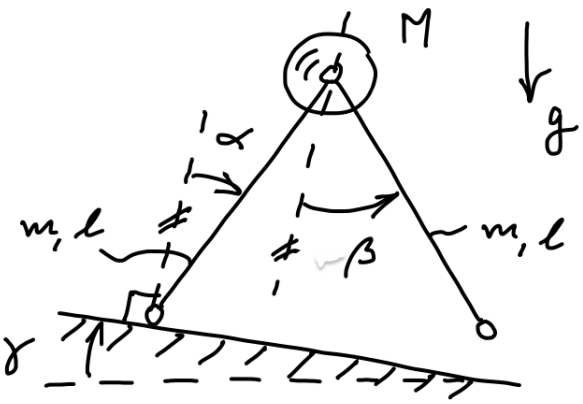
# Multibody Dynamics B - Assignment 5

July 4, 2019

**a)**




We are now working with a simplified 2D model of a bipedal walking machine, which can be seen in figure 1. The ground is tilted clockwise by an angle $\gamma$ and has a gravity field pointing in the negative y-direction. Furthermore, we take the generalized coordinates alpha, $\alpha$, and beta,$\beta$, depicted in the picture above. Alpha is the angle of the stance leg with respect to the outer normal to the ground and beta is the angle of the swing leg with respect to the outer normal to the ground, where clockwise is positive. The two legs are identcial and have length $l$ and mass $m$ distributed evenly, and point mass M at the hip. We derive the equation of motion using the Lagrange method where we have

$$Q_j = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial V}{\partial q_j} \tag{1}$$

where T is the kinetic energy

$$T = 1/2 * \dot{x_i} * M_{ij} * \dot{x_j} \tag{2}$$

V is the potential energy,

$$-\frac{\partial V}{\partial x_i} + F_i = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x_i}}\right)$$

$Q_j$ are the generalized forces and $q_j$ are the generalized coordinates where the CoM coordinates $x_i$ for the rigid bodies can be written as $x_i = x_i(q_j)$. Now we have to define our system in the CoM coordinates with respect to the generalised coordinates which gives us;

stance leg

$$x_1 = l/2 * sin(\alpha + \gamma) \quad y_1 = l/2 * cos(\alpha + \gamma) \quad \varphi_1 = \alpha + \gamma \tag{3}$$

hip

$$x_h = l * sin(\alpha + \gamma) \quad y_h = l * cos(\alpha + \gamma) \tag{4}$$

and for swing leg

$$x_2 = x_h + l/2 * sin(\beta - \gamma) \quad y_2 = x_h + l/2 * cos(\beta - \gamma) \quad \varphi_2 = \beta - \gamma \tag{5}$$

We next find the velocities of the coordinated defined above, with Matlab see code in appendix, which gives

$$\dot{x}_i = \begin{bmatrix} (\dot{\alpha} * l * cos(\alpha + \gamma))/2 \\ -(\dot{\alpha} * l * sin(\alpha + \gamma))/2 \\ \dot{\alpha} \\ \dot{\alpha} * l * cos(\alpha + \gamma) \\ -\dot{\alpha} * l * sin(\alpha + \gamma) \\ (\dot{\beta} * l * cos(\beta - \gamma))/2 + \dot{\alpha} * l * cos(\alpha + \gamma) \\ (\dot{\beta} * l * sin(\beta - \gamma))/2 - \dot{\alpha} * l * sin(\alpha + \gamma) \\ \dot{\beta} \end{bmatrix}$$

Next we find the acceleration, again with the help of Matlab, which gives

$$\ddot{x}_i = \begin{bmatrix} (\ddot{\alpha} * l * cos(\alpha + \gamma))/2 - (\dot{\alpha}^2 * l * sin(\alpha + \gamma))/2 \\ -(\dot{\alpha}^2 * l * cos(\alpha + \gamma))/2 - (\ddot{\alpha} * l * sin(\alpha + \gamma))/2 \\ \ddot{\alpha} \\ -\dot{\alpha}^2 * l * sin(\alpha + \gamma) + \ddot{\alpha} * l * cos(\alpha + \gamma) \\ -\dot{\alpha}^2 * l * cos(\alpha + \gamma) - \ddot{\alpha} * l * sin(\alpha + \gamma) \\ -(l * (2 * sin(\alpha + \gamma) * \dot{\alpha}^2 + sin(\beta - \gamma) * \dot{\beta}^2 - 2 * \ddot{\alpha} * cos(\alpha + \gamma) - \ddot{\beta} * cos(\beta - \gamma)))/2 \\ (l * (-2 * cos(\alpha + \gamma) * \dot{\alpha}^2 + cos(\beta - \gamma) * \dot{\beta}^2 - 2 * \ddot{\alpha} * sin(\alpha + \gamma) + \ddot{\beta} * sin(\beta - \gamma)))/2 \\ \ddot{\beta} \end{bmatrix}$$

Now we define our mass matrix as $M_{ij} = diag([m, m, I, M.M.m.m, I])$ where $I$ is the inertia and is defined as $I = \frac{1}{12} * m_1 * l_1^2$ since we assume a slender beam. Now we find our potential energy, which we define as

$$V = -(m * g * y_1 + m * g * y_2 + M * g * y_h)$$

and our kinetic energy which we find by using Matlab and equation (2), which gives us

$$T = (I * \dot{\alpha}^2)/2 + (I * \dot{\beta}^2)/2 + (M * \dot{\alpha}^2 * l^2)/2 + (5 * \dot{\alpha}^2 * l^2 * m)/8 + (\dot{\beta}^2 * l^2 * m)/8 + (\dot{\alpha} * \dot{\beta} * l^2 * m * cos(\alpha + \beta))/2$$

Furthermore we define the generalized forces as $Q_j = [Q_a, Q_b]^T$. Now we can compute $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right)$, $\frac{\partial T}{\partial q_j}$ and $\frac{\partial V}{\partial q_j}$ by using Matlab, this gives us;

$$\frac{\partial V}{\partial q_j} = \begin{bmatrix} (g * l * sin(\alpha + \gamma) * (2 * M + 3 * m))/2 \\ -(g * l * m * sin(\beta - \gamma))/2 \end{bmatrix}$$

$$\frac{\partial T}{\partial q_j} = \begin{bmatrix} -(\dot{\alpha} * \dot{\beta} * l^2 * m * sin(\alpha + \beta))/2 \\ -(\dot{\alpha} * \dot{\beta} * l^2 * m * sin(\alpha + \beta))/2 \end{bmatrix}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) = \begin{bmatrix} I + M * l^2 + (5 * l^2 * m)/4 & (l^2 * m * cos(\alpha + \beta))/2 \\ (l^2 * m * cos(\alpha + \beta))/2 & (m * l^2)/4 + I \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}$$

$$+ \begin{bmatrix} -(\dot{\beta}^2 * l^2 * m * sin(\alpha + \beta))/2 - (\dot{\alpha} * \dot{\beta} * l^2 * m * sin(\alpha + \beta))/2 \\ -(\dot{\alpha}^2 * l^2 * m * sin(\alpha + \beta))/2 - (\dot{\alpha} * \dot{\beta} * l^2 * m * sin(\alpha + \beta))/2 \end{bmatrix}$$

Now we can write this as a matrix as

$$\mathbf{M} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} = \mathbf{F}$$

where

$$\mathbf{M} = \begin{bmatrix} I + M * l^2 + (5 * l^2 * m)/4 & (l^2 * m * cos(\alpha + \beta))/2 \\ (l^2 * m * cos(\alpha + \beta))/2 & (m * l^2)/4 + I \end{bmatrix}$$

and

$$\mathbf{F} = \begin{bmatrix} (m * sin(\alpha + \beta) * \dot{\beta}^2 * l^2)/2 - (g * sin(\alpha + \gamma) * (2 * M + 3 * m) * l)/2 + Q_a \\ (m * sin(\alpha + \beta) * \dot{\alpha}^2 * l^2)/2 + (g * m * sin(\beta - \gamma) * l)/2 + Q_b \end{bmatrix}$$

## b)

Now we use the TMT method to derive the equations of motions. There we use the virtual power expression and d'Alambert forces. The virtual power expression we know and is defined as

$$\delta P = \delta \dot{x}_i (F_i - Mij\ddot{x}_j)$$

Now we will use the fact that we can write $x_i = T_i(q_i)$, where $\dot{x}_i = T_{i,j}(\dot{q}_j)$ and $\ddot{x}_k = T_{k,l}(\ddot{q}_l + T_{k,lm}\dot{q}_l\dot{q}_m$. Since we introduced the generalized coordinates we have introduced generalized forces $Q_j$ which we add to our power expression which gives

$$\delta P = \delta \dot{x}_i (F_i - Mij\ddot{x}_j) + \delta \dot{q}_j Q_j$$

We define again the CoM coordinates with regard to the generalized coordinates and they are defined exactly the same as in equations (3)-(5). We now introduce a transformation matrix $T_i$ which is defined as $T_i = [x_1, y_1, \varphi_1, x_h, y_h, x_2, y_2, \varphi_2]^T$. Now the TMT method tells us that we have a solution of the kind of

$$\bar{M}_{jl}\ddot{q}_l = \bar{Q}_j$$

Now we find using Matlab, and the jacobian command see appendix for code, $T_{ij}$ and $T_{ijk}$, which are defined for the velocity and acceleration terms. Now

$$\bar{Q}_j = Q_j + Tij(F_i - Mijg_k)$$

where $Q_j = [Q_a, Q_b]^T$, $F_i = Mij * [0, g, 0, 0, g, 0, g, 0]^T$, $M_{ij} = diag([m, m, I, M.M.m.m, I])$ and $g_k = T_{ijk}\dot{q}_l\dot{q}_m$ which gives us

$$\bar{Q}_j = \begin{bmatrix} Qa - M * g * l * sin(\alpha + \gamma) - (3 * g * l * m * sin(\alpha + \gamma))/2 + (\dot{\beta}^2 * l^2 * m * sin(\alpha + \beta))/2 \\ (m * sin(\alpha + \beta) * \dot{\alpha}^2 * l^2)/2 + (g * m * sin(\beta - \gamma) * l)/2 + Qb \end{bmatrix}$$

Furthermore,

$$\bar{M}_{jl} = T_{ij}^{T} M i j T_{ij}$$

which gives

$$\bar{M}_{jl} = \begin{bmatrix} I + M * l^2 + (5 * l^2 * m)/4 & (l^2 * m * cos(\alpha + \beta))/2 \\ (l^2 * m * cos(\alpha + \beta))/2 & (m * l^2)/4 + I \end{bmatrix}$$

Now we have the equation of motion for the TMT method, represented in state space form.

## c)

Now we can compare the TMT and the Lagrange method and see if they match. Lets look at the **M** matrix first, for the Lagrange we have

$$\mathbf{M} = \begin{bmatrix} I + M * l^2 + (5 * l^2 * m)/4 & (l^2 * m * cos(\alpha + \beta))/2 \\ (l^2 * m * cos(\alpha + \beta))/2 & (m * l^2)/4 + I \end{bmatrix}$$

and for the TMT we have

$$\bar{M}_{jl} = \begin{bmatrix} I + M * l^2 + (5 * l^2 * m)/4 & (l^2 * m * cos(\alpha + \beta))/2 \\ (l^2 * m * cos(\alpha + \beta))/2 & (m * l^2)/4 + I \end{bmatrix}$$

Now by comparing every cell in the matrix we see that the 2 solutions completely compare. Now let's looka at $F_i$ and $\bar{Q}_j$. For the Lagrange we have

$$\mathbf{F} = \begin{bmatrix} (m * sin(\alpha + \beta) * \dot{\beta}^2 * l^2)/2 - (g * sin(\alpha + \gamma) * (2 * M + 3 * m) * l)/2 + Q_a \\ (m * sin(\alpha + \beta) * \dot{\alpha}^2 * l^2)/2 + (g * m * sin(\beta - \gamma) * l)/2 + Q_b \end{bmatrix}$$

and for the TMT we have

$$\bar{Q}_j = \begin{bmatrix} Qa - M * g * l * sin(\alpha + \gamma) - (3 * g * l * m * sin(\alpha + \gamma))/2 + (\dot{\beta}^2 * l^2 * m * sin(\alpha + \beta))/2 \\ (m * sin(\alpha + \beta) * \dot{\alpha}^2 * l^2)/2 + (g * m * sin(\beta - \gamma) * l)/2 + Qb \end{bmatrix}$$

At first glance they seem different in the first row but if we multiply out of the bracket $(2 * M + 3 * m)$ in **F** and rearrange a little bit we see that these equations are exactly the same. So we conclude that both equations are equal to one another.

## c)

Since both systems use generalized coordinates, we will experience some difficulties in finding coordinates that do not result in non-unique mapping. However, the computation is much faster than for say the Newton-Euler method. Also, we are not able to calculate the constraint forces except by introducing extra degrees of freedom in the system. For the TMT method there is less algebraic effort, since TMT just stays with forces all the way through while the Lagrange method is changing between energy and forces. When using CoM coordinates we have many equations to find, and many constraints to estimate, this is eliminated in the TMT and Lagrange method, so for a large system that probably has many constraints and each body has to be represented by 3 coordinates, we will result in an abundant amount of equations, which needs a lot of computational effort. Furthermore, lets say we have a large system but of a low degree of freedom, this would make the calculations considerably easier, and less effort is needed.

# Appendix A

## Matlab code

```
%% Set up EoM a)
% Set up variables
syms alpha beta gamma
syms alphadot betadot % for dx/dt I use xd etc
syms alpha2dot beta2dot
syms l m M I g Qa Qb
q=[alpha; beta];
qd=[alphadot; betadot];

% Define CoM coordinates
x1=sin(alpha+gamma)*l/2;        % x coordinates of stance leg
y1=cos(alpha+gamma)*l/2;        % y coordinates of stance leg
phi1=alpha+gamma;               % angle of stance leg
xh=sin(alpha+gamma)*l;          % x coordinates of hips
yh=cos(alpha+gamma)*l;          % x coordinates of hips
x2=xh + sin(beta-gamma)*l/2;    % x coordinates of swing leg
y2=yh - cos(beta-gamma)*l/2;    % x coordinates of swing leg
phi2=beta-gamma;                % angle of swing leg

% Coordinate matrix
X=[x1;y1;phi1;xh;yh;x2;y2;phi2];
Xij=simplify(jacobian(X,q));
Xd=Xij*qd;

Tijk=zeros(8,2);
for i=1:2
    Tijk=Tijk+jacobian(Xij(:,i),q);
end
% find acceleration
xdd = simplify(Xij*[alpha2dot;beta2dot] + Tijk*(qd.*qd));

% Define mass matrix
Mij=diag([m, m, I, M, M, m, m, I]);

% Define potential energy
V=-(m*g*y1 + m*g*y2+ M*g*yh);

% Define kinetic energy
T = simplify(1/2*Xd.'*Mij*Xd);

% External forces
Q = [Qa;Qb];

% Find lagrange
Vdq = simplify(jacobian(V,q));
Tdq = simplify(jacobian(T,q));
Tdqd = simplify(jacobian(T,qd));
Tdqdd = simplify(jacobian(Tdqd,q));

% Right hand side of the matrix
F_l = simplify(Q-Vdq.' + Tdq.' - Tdqdd*qd);
```

```
52
53 % Left hand side of the matrix
54 M_l = jacobian(Tdqd,qd);
55
56 %% Set up EoM b)
57 % Set up variables
58 syms alpha beta gamma
59 syms alphadot betadot % for dx/dt I use xd etc
60 syms alpha2dot beta2dot
61 syms l m M I g Qa Qb
62
63 q=[alpha; beta];
64 qd=[alphadot; betadot];
65
66 % Define CoM coordinates
67 x1=sin(alpha+gamma)*l/2;        % x coordinates of stance leg
68 y1=cos(alpha+gamma)*l/2;        % y coordinates of stance leg
69 phi1=alpha+gamma;               % angle of stance leg
70 xh=sin(alpha+gamma)*l;          % x coordinates of hips
71 yh=cos(alpha+gamma)*l;          % x coordinates of hips
72 x2=xh + sin(beta-gamma)*l/2;    % x coordinates of swing leg
73 y2=yh - cos(beta-gamma)*l/2;    % x coordinates of swing leg
74 phi2=beta-gamma;               % angle of swing leg
75
76 % Define transformation matrix
77 Ti = [x1; y1; phi1; xh; yh; x2; y2; phi2];
78
79 % Find Tij, where xd = Tij*qd
80 Tij = jacobian(Ti,q);
81
82 % Find Tijk where xdd = Tij*qdd + Tijk*qd*qd
83 Tijk=zeros(8,2);
84
85 for i=1:2
86     Tijk=Tijk+jacobian(Tij(:,i),q);
87 end
88 % Mass matrix
89 Mij=diag([m, m, I, M, M, m, m, I]);
90
91 % applied forces
92 Fi=Mij*[0,g,0,0,g,0,g,0].';
93
94 % External forces
95 Q = [Qa;Qb];
96
97 % Find reduced mass matrix
98 Mbar = simplify(Tij.'*Mij*Tij);
99
100 % Convective acceleration terms
101 gk=Tijk*(qd.*qd);
102
103 % Combined force matrix
104 Qbar = simplify(Q+Tij.'*(Fi-Mij*gk));
```