Hopping leg simulator instructions:

Yanran Ding and João Ramos

1- Model generation:

First you need to generate the kinematics and dynamic model using the function "*gen_dyn_boom_leg_v1.m*"

Define system states (four joint angles), velocities, and physical parameters of the leg plus boom (masses, lengths, inertias, etc):

```
m_list_q
m_list_dq
m_list_params
```

Instead of using the symbolic expressions for the HTM, Jacobinas, EOM matrix, etc, this function writes a .m file that returns the instantaneous value of that matrix/vector. For instance, the function:

```
write_fcn_m('fcn_p_toe.m',{'q','p'},[m_list_q;m_list_params],{p_toe,'p
_toe'});
```

Writes a file `fcn_p_toe.m` that describes a function that takes the joint angles `q` and the system parameters `p` to compute the x, y, and z spatial position of the foot. The file

```
write_fcn_m('fcn_De.m',{'q', 'p'},[m_list_q;m_list_params],{De,'De'});
write_fcn_m('fcn_Ce.m',{'q', 'dq', 'p'},[m_list_q; m_list_dq;
m_list_params],{Ce,'Ce'});
```

contains the function that return the inertia matrix `De` and Coriolis vector `Ce` of the manipulator with instantaneous configuration `q` and joint velocities `dq`. The actuation selection matrix `Be` maps the 2x1 control input vector you have access to (hip and knee torques) to the 4x1 input torque in the EOM. Remember that the first two joints are passive.
This function generates the .m files for all the required expressions for kinematics and dynamics of the manipulator.

2- System parameters:

The function "`get_params.m`" defines the parameters of the system, they are:
Stance time for force parametrization during stance: `p.Tst`
Animation time stepping (make this number larger for slow motion): `p.N_animate`
Gravity: `g`
Link lengths: `HB,LB,DB,LH,LK,DK`
Link masses: `M1,M2,M3,M4`
CoM position in link i frame: `rxi,ryi,rzi`
Components of inertia tensor of link i: `Jxi,Jyi,Jzi`
Notice that the inertia tensor here is defined as diagonal, you will need to change that for your project.

3- Stance dynamics, air dynamics, and transition events:

The hopping leg is a system with hybrid dynamics. This means that the dynamic are interrupted by discrete events (impacts with the grounds) that may cause discontinuous transition of the states (joint velocities). The system has different dynamic equations of motion during flight (foot not touching the ground) and during stance (foot in contact with the ground). These dynamics are separated by events: touch down (flight to stance) and lift off (stance to flight).

The equations of motion for the flight phase is:

$$D_e(q)\ddot{q} + C_e(q,\dot{q})\dot{q} + G_e(q) = B_e u$$

The actuation selection matrix B_e is defined as$B_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

To make sure that joints 1 are 2 are passive (you can't apply torque to them). The implemented joint level PD controller simply attempts to maintain the leg at a certain configuration `qd`.

$$u = \begin{bmatrix} K_{p1} & 0 \\ 0 & K_{p2} \end{bmatrix} (q_d - q) - \begin{bmatrix} K_{d1} & 0 \\ 0 & K_{d2} \end{bmatrix} \dot{q}$$

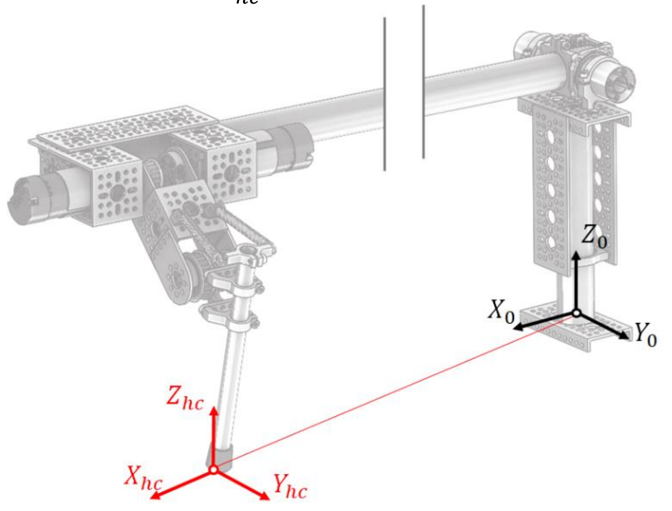
The stance dynamics are given by the equations:

$$D_e(q)\ddot{q} + C_e(q,\dot{q})\dot{q} + G_e(q) = J_{hc}^T F_{GRF} + B_e u$$
$$J_{hc}\ddot{q} + \dot{J}_{hc}\dot{q} = 0$$

The second equation defines the holonomic constraint that the foot is in contact with the ground and cannot move (velocity of the foot is zero):

$$J_{hc}\dot{q} = \begin{bmatrix} J_{Yhc} \\ J_{Zhc} \end{bmatrix} \dot{q} = 0$$

The Jacobian J_{hc} maps the joint velocities to the foot velocity in the directions $Y_{hc}Z_{hc}$. This frame does not necessarily coincide with the foot fixed frame. It changes at every stepping location. The direction X_{hc} is the radial direction between the foot and the boom base. Direction Z_{hc} is vertical and always parallel with Z_0. When the robot steps on the ground, the foot is not allowed to move vertically Z_{hc} or tangentially to the boom rotation Y_{hc}. However, it can move radially towards the boom in the X_{hc} direction. In readl life, the foot always slides a little bit in the X_{hc} direction.



This means that the Jacobian J_{hc} imposes a constraint to the foot motion. Taking the time derivative, we have:

$$J_{hc}\ddot{q} + \dot{J}_{hc}\dot{q} = 0$$

Now we not only have to solve for the joint acceleration $\ddot{q}$ at all the ode45 iterations, but we also need to compute the foot contact forces that make sure that the foot does not move. This model is called "Hard contact" model:

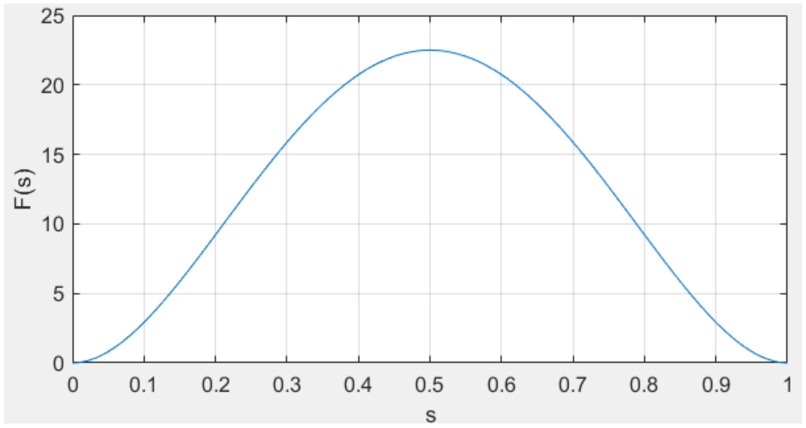
$$\begin{bmatrix} D_e & -J_{hc}^T \\ J_{hc} & 0_{2x2} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ F_{GRF} \end{bmatrix} = \begin{bmatrix} B_e u - C_e\dot{q} - G_e \\ -\dot{J}_{hc}\dot{q} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{q} \\ F_{GRF} \end{bmatrix} = \begin{bmatrix} D_e & -J_{hc}^T \\ J_{hc} & 0_{2x2} \end{bmatrix}^{-1} \begin{bmatrix} B_e u - C_e\dot{q} - G_e \\ -\dot{J}_{hc}\dot{q} \end{bmatrix}$$

The stance controller regulates the contact force between the foot and the ground. You can implement any force profile you want, but currently the for profile follows a Bezier polynomial parametrized in `s = [0,1]` . The function `b = polyval_bz(alpha, s)` evaluates the force at instant `s` using Bezier coefficients `alpha`. Value of `s` is 0 at the beginning of stance and 1 at the end. For instance, for

```
s = 0:0.01:1;
plot(s,polyval_bz([0 0 60 0 0],s)
```

gives:



The stance controller now simply applies a force upwards to maintain hopping by using the foot Jacobian in respect to the hip J_{HIP}. This Jacobian maps the relative velocity of the foot in respect to the hip. It's the same Jacobian you developed for item 1.f) of milestone #1.

$$u = J_{HIP} \begin{bmatrix} F_{Yd} \\ F_{Zd} \end{bmatrix} = J_{HIP} \begin{bmatrix} 0 \\ F_{Bezier} \end{bmatrix}$$

You can change it to regulate hopping height, speed or the boom angle θ_1.

The transition between flight and stance occurs when the foot height reaches 0. This is defined in the function "`event_touchDown(t,X,p)`".

Because the foot impacts the ground and doesn't bounce, we need to calculate what is the impact force applied to the robot that changes the velocity of the foot instantaneously. From, the joint velocities before impact $\dot{q}^-$ we need to compute the joint velocities after impact $\dot{q}^+$ and the impact F_{imp}. For that we us the function `X_post = fcn_impactMap(X_prev,p)` to solve the system:

$$\begin{aligned} &M(\dot{q}^+ - \dot{q}^-) = J_{toe}^T F_{imp} \\ &J_{toe}\dot{q}^+ = 0 \end{aligned}$$

The transition between stance and air phase occurs when the ground reaction force in the vertical direction becomes zero. For that we must monitor the value of $F_{GRF}(3)$ obtained from solving the stance dynamics.

4- Simulation:

The simulation runs for a number of `Nstep` hops. It starts at flight (foot not touching the ground) and transitions to stance when touch down happens. The pseudocode of what happens is:

```
define q0, q0_dot

for i_steps = 1:N_steps
do:
        air_phase_dynamics (ode45)
                wait for touch_down event
        stance_phase_dynamics (contact dynamics in ode45)
                wait for lift_off event
end

animateRobot
```