# Solve Equations of Motion of the Three-link Biped: Simulation and Animation (Assignment 3)

**Submission deadline: November 19, 2018 at 23:59 (20% of the mini-project grade)**

In this task you first will rewrite the equations of motion in the state-space form. Then you will use `ode` solver of MATLAB to solve the equations of motion. Afterwards, you will animate the simulation. To do this, you will complete the following scripts.

```
eqns.m, event_func.m, solve_eqns.m (in the "solve_eqns" folder)
animate.m (in the "visualize" folder)
```

## 1. Complete `eqns.m`

We would like to solve the equations of motion:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Bu \tag{1}$$

with `ode45` solver of MATLAB. The signature of the `ode45` solver is as follows:

```
[T, Y, TE, YE] = ode45(@eqns, tspan, y0, options)
```

where `eqns.m` is a MATLAB function with the signature `dy = eqns(t, y)` representing the state-space form of the equations of motion above.

First off, run the script `set_path.m` to add the required directories to your path. Then start off by completing `eqns.m` in the `solve_eqns` folder. Remember, you need to write the equations of motion $M(q)\ddot q + C(q, \dot q) \dot q + G(q) = Bu$ in the form of $\dot y = eqns(t, y)$. You can do this in many different ways but for consistency use this definition of $y$:

$$y = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{2}$$

## 2. Complete `solve_eqns.m`

As mentioned above, the signature of the `ode45` solver is:

```
[T, Y, TE, YE] = ode45(@eqns, tspan, y0, options)
```

where `eqns.m` is the equations of motion in the state-space form, `tspan` is the time span for which you would like to solve the ode, `y0` is the initial condition and `options` defines the options for the ode solver. The *impact map* which is an event function is defined using the `options`. We get to this later.

## 3. Complete `animate.m`

If `Y` is the solution to the ode, at each time step `i` you can extract the angles `q` by `q = Y(i, 1:3)`. Use this `q` as the input the `visualize.m` function to animate the solution `Y` of the equations of motion. Note that the `visualize.m` function has an extra input `r0` which is the position of the stance foot in the global frame. Why do we need this?

Calculate the real time factor as defined in the `animate.m` script. What does a real time factor of 1 mean? How about a real time factor smaller than 1?

## 4. Define the event function and include the impact map

With `odeset` set the `options` in `solve_eqns.m` so that the relative tolerance is `1e-5` and the event function is `event_func`. Then complete the event function `event_func.m`. We want the event function to trigger when the swing foot hits the ground. To this end, set the `value` in `event_func.m` appropriately. Additionally, we only want to declare an event if the swing foot hits the ground with a negative `z` velocity. To account for this, set the `direction` in `event_func.m` appropriately.

Run the simulation and animate the results for different initial conditions to verify your animation intuitively.