Ecole Centrale de Nantes

MASTER CORO-IMARO

Humanoid Robotics

Passive walking of a compass robot

Serena Roncagliolo, Camille Vindolet

28 December 2020

Contents

1 Introduction

In this report, we show the study of the motion of a passive compass robot as shown in Fig (1)

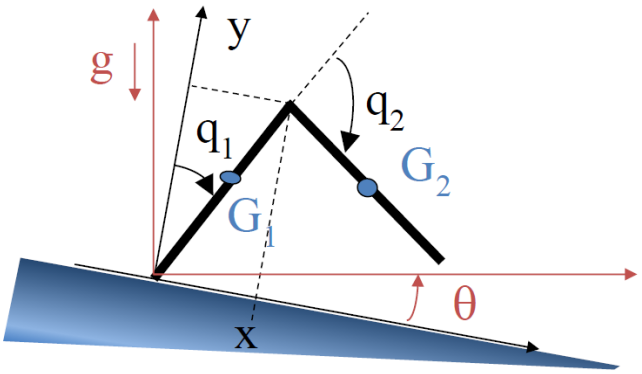

Figure 1: Geometrical representation compass robot

The black frame corresponds to the reference frame of the compass robot. This frame is attached to the tilted ground wit and it is rotated of angle $-\theta$ with respect to the absolute frame in red.
The gravity is directed along the y axis of the absolute frame and the two articulations q1 and q2 are not actuated, the motion is created by the action of gravity on the robot.

2 Simulation of the passive gait

We consider a passive gait, in which no torque is applied. The robot is placed on a slope inclined of $\theta = \frac{3\pi}{180}$ rad.
The gait is composed of:

- phase of *support* on leg 1
- phase of *impact* where leg 2 is touching the ground and the leg 1 is taking off the ground
- phase of *support* on leg 2
- phase of *impact* where leg 1 is touching the ground and the leg 2 is taking off the ground

We will use the functions we have computed in the first lab:

- *function_dyn*$(q_1, q_2, \dot{q}_1, \dot{q}_2, \theta)$
- *function_reactionforce*$(q_1, q_2, \dot{q}_1, \dot{q}_2, \ddot{q}_1, \ddot{q}_2)$
- *function_impact*(q_1, q_2)

The robot and slope characteristics used are:

- l=0.8m

- m=2 kg
- I=0.1 kg·m^2
- s=0.5m
- θ=3*pi/180 rad

where l is length of the leg, m the value of the mass of one leg, I the inertia, s is the distance between the position of center of gravity and position of the hip.
The initial state of the robot is:

- q_1 = 0.1860
- q_1 = 2.7696
- $\dot{q}_1$ = -1.4281
- $\dot{q}_2$ = 0.3377

2.1 Simulation of one single support phase

In this section we define and simulate the phase of *support* on leg 1 only. The robot performs one half step and stops before the impact of leg 2.
Our simulation is carried out in Matlab, mainly using the function *ode45*.
From the dynamic model, we can deduce the evolution of the state of the robot, denoted as z and given by (1)

$$z = \begin{pmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} \tag{1}$$

The function *SS_passif* will define the derivative of the states as function of the state giving $\dot{z} = SS\_passif(t, z)$.

2.1.1 ode45

The outputs of the function *ode45* are the time vector **t** and the corresponding value of the state **z**. We will use the function $[t, z, t_e, z_e] = ode45(odefun, tspan, z_0, options)$ where:

- *odefun* in which we pass @SS_passif
- $t_{span} = [t_0 : t_f]$ where t_{span} is the interval on time on which the equation is solved. We set it from $t_0 = 0s$ to $t_f = 10s$ with a time step of $20ms$, but the simulation will be stop before $10s$ when the output of the function *Events* will be 0
- z_0 initial conditions
- *options* defines the integration setting, which is an argument created using the *odeset* and connected to PEvents (i.e.: *options = odeset('Events', @PEvents)*)

This function returns the results of the integration which is the state vector z containing the joint positions and velocities and the time instances. It also returns the state vector corresponding to the last state obtained before the integration was stopped.

2.1.2 PEvents

PEvents describes the event triggering the end of the simulation. This event occurs when the value defined in *Pevent* is null. We want to use it to stop the simulation before an impact to define properly this impact and start another half step after.
z is the vector containing the state variables of the system and it is given in (1). It characterises the current state of the robot, thus $z(1) = q_1$ and $z(2) = q_2$.
In our case the simulation has to stop when the swinging foot reaches the ground, so when its height is null. The height of the swinging foot can be expressed in function of q_1 and q_2:

$$y_{swingingFoot} = l(cosq_1 + cos(q_1 + q_2)) \tag{2}$$

The value of *PEvent* should be equal to the height of the swinging foot. However its position along the x axis has to be positive for the event to be triggered in order to not have the compass robot stopping if both its feet are in (0,0). Thus:

$$x_{swingingFoot} = -l(sinq_1 + sin(q_1 + q_2)) \tag{3}$$

VALUE is then given by $y_{swingingFoot}$ when $x_{swingingFoot} > 0.1$, else it's positive and assigned to 1.

2.1.3 SS_Passif

The function *SS_passif* gives the derivative of the state variable z with inputs z (see Eq. (1)) and time t. This function is called in the differential equation resolution solved by the function *ode45* in Matlab.
SSPassif is based on the resolution of the inverse dynamic model.
The function $[A, H] = function\_dyn(z(1), z(2), z(3), z(4), theta)$ gives the values of the matrices $\mathbf{A}$ and $\mathbf{H}$ given the state of the system.
We need also to add the gravity effects given by $\mathbf{H_g}$.

$$\ddot{\boldsymbol{q}}_{\boldsymbol{a}} = -\mathbf{A}^{-1}(\mathbf{H} + \mathbf{H_g}) = \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} \tag{4}$$

In order to compute the derivative of the state $\dot{z}$ we apply the following:

$$\dot{\boldsymbol{z}} = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} \tag{5}$$

The effect of gravity is computed as follows:

$$\mathbf{H_g} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} \tag{6}$$

where:

$$Q_1 = -mg((2l - s)sin(q_1 - \theta) + ssin(q_1 + q_2 - \theta)) \tag{7}$$

$$Q_2 = -mgssin(q_1 + q_2) \tag{8}$$

2.1.4 Differential Equation

The differential equation is thus resolved by the use of the function ode45, $[t, y, te, ye, ie] = ode45(odefun, tspan, q0, options)$,
In our case $t_{span} = [0 : 0.02 : 10]$, *odefun* is given by *SSPassif*, q_0 are the initial conditions, given by:

$$q_0 = \begin{pmatrix} q_{10} \\ q_{20} \\ \dot{q_{10}} \\ \dot{q_{20}} \end{pmatrix} = \begin{pmatrix} 0.1860 \\ 2.7696 \\ -1.4281 \\ 0.3377 \end{pmatrix} \tag{9}$$

Moreover *options* describes the *events* that are of value 0 at time t_e and for the joint value of q_e.

2.1.5 Animation of the passive walking

Once we obtain the state corresponding to the joint positions at every time instant, we prepare a Matlab function called *simulation* to prepare a simulation of the motion of this robot. The animation of the passive walking can be observed below:

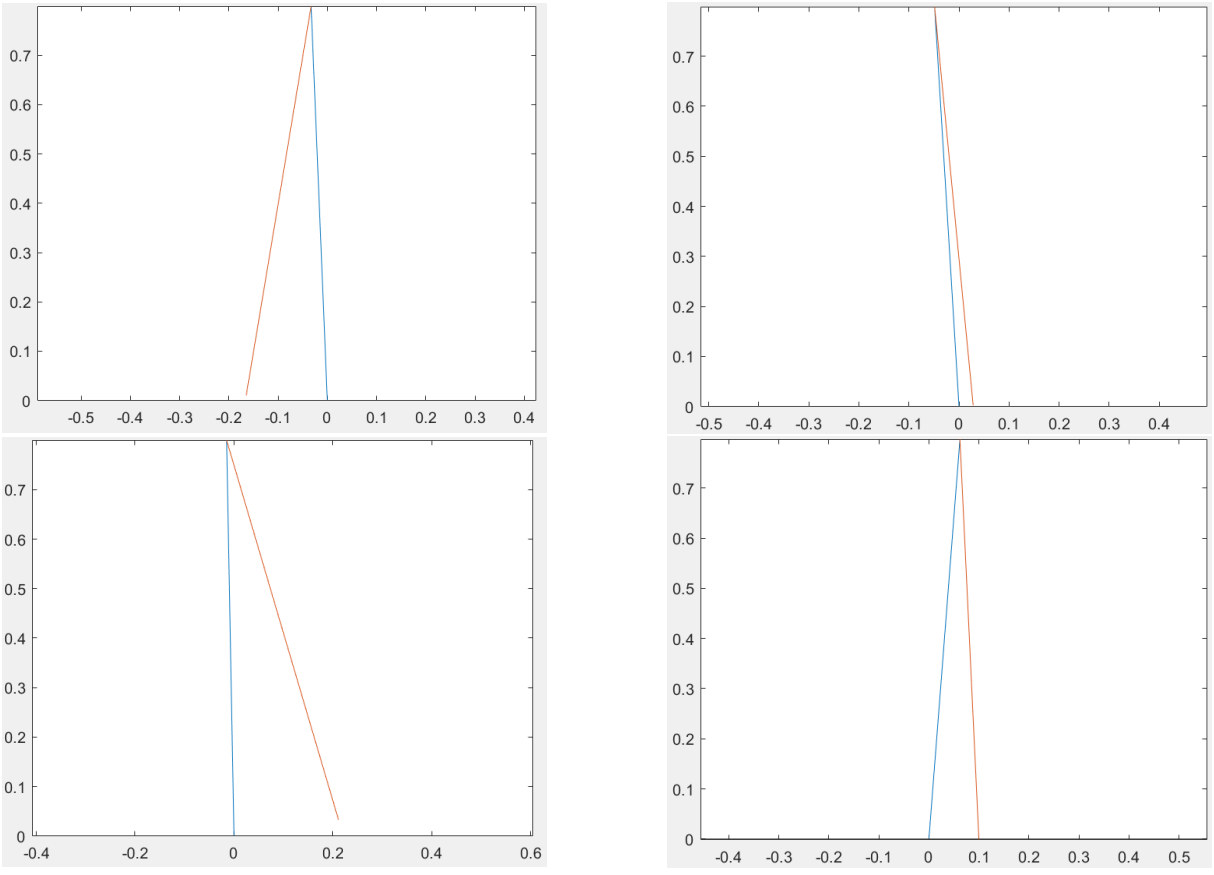

Figure 2: Animation for one step of a compass walk

2.2 Simulation of several half steps

Once one half step is solved the simulation stops before the impact of the swinging foot. The aim is thus to define the impact and relabel the joint characteristics to exchange both

roles of the legs to simulate an extra step using the same functions as previously. In order to simulate several half steps we may follow these steps, starting from a configuration of DS and initial velocity for q_1 and q_2:

1. simulate one single support until impact
2. calculate the state before impact without implicit constraint
3. calculate the joint velocity after impact
4. change the label of the joint, this way it will be ready for a new single support on leg 1
5. repeat Step 1 to simulate a next half step

2.2.1 Impact model

First of all we execute the following

```
options = odeset('Events',@PEvents);
[t,z,te,ze] = ode45(@SS_passif,[0:0.02:10],z0,options);
```

This way we are able to calculate the state before the impact without constraints.
We will the run function *[z_afterImpact] = impactModel(ze)* to compute the state after the impact.
Inside this function we are using the function computer for Lab 1, which gives us the matrices A_1 and J_{r2} as follows:

$$A_1 = \begin{bmatrix} 2m & 0 & sm(cosq_1 - cos(q_1+q_2)) & -mscos(q_1+q_2) \\ 0 & 2m & ms(sinq_1 - sin(q_1+q_2)) & -mssin(q_1+q_2) \\ sm(cosq_1 - cos(q_1+q_2)) & ms(sinq_1 - sin(q_1+q_2)) & 2ms^2+2I & I+ms^2 \\ -mscos(q_1+q_2) & -mssin(q_1+q_2) & I+ms^2 & ms^2+I \end{bmatrix} \tag{10}$$

$$J_{2r} = \begin{bmatrix} -lcos(q_1+q_2) & -lcos(q_1+q_2) & 1 & 0 \\ -lsin(q_1+q_2) & -lsin(q_1+q_2) & 0 & 1 \end{bmatrix} \tag{11}$$

To calculate the impact model, we use the z_e and t_e corresponding to the state and time instance just *before the impact* during one cycle of single support. To calculate the joint velocity after impact, we also have to use the relabeling equations that will change the role of both legs.

2.2.2 Relabeling equations

In order to avoid deriving a new model, we will introduce a relabeling of the number of the leg. q_1 defines the orientation of the leg in support and q_2 of the leg swinging. This should be the case for any leg in support. Before the *impact* leg 1 is in support, while after impact leg 2 is in support.
We have to define a new set of coordinates which we call q_{1n} and q_{2n}, which will be the use for the support on leg 2. As shown in Fig. (3), we define q_{1n} and q_{2n} as function of q_1 and q_2 and obtain the relabelling equations. These equations will be derive to obtain the relabelling relation of the velocity

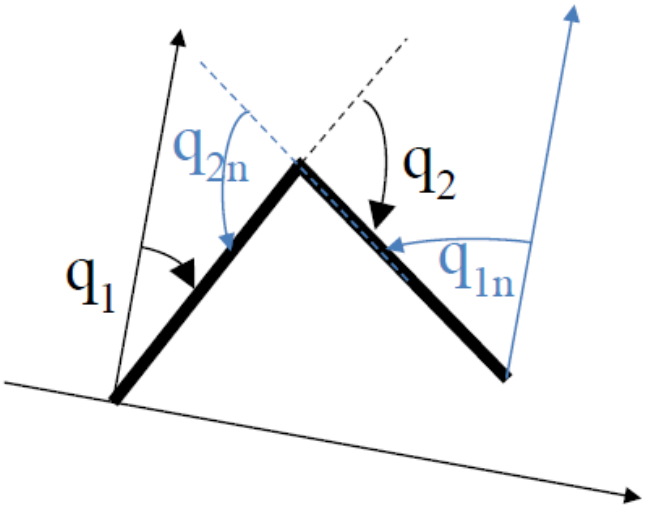

Figure 3: Relabel equation scheme

At *impact*, the joint values should change as shown below:

$$q_{1n} = \pi + q_1 + q_2; \tag{12}$$

$$q_{2n} = -q_2; \tag{13}$$

We then use the state variables obtained after the relabelling to run again *ode45 function* to find the solution of the next half step differential equation.

3 Periodic motion

In this section, we want to obtain a cyclic passive motion on the slope. We want to have a cyclic passive motion which is stable. A periodic motion is a fixed point of the Poincare return map. We simulate the periodic motion using the following parameters:

- *Case 1*:
  - l=0.8m
  - m=2 kg
  - I=0.1 kg·m²
  - s=0.5m
  - θ=3*pi/180 rad
  - q_1 = -0.1860
  - $\dot{q}_1$ = -2.0504
  - $\dot{q}_2$ = -0.0428

- *Case 2*:
  - l=0.8m
  - m=2kg
  - I=0.08 kg·m^2
  - s=0.45m
  - θ=3*pi/180 rad
  - q_1 = -0.1933
  - $\dot{q}_1$ = -2.0262
  - $\dot{q}_2$ = -0.1253

3.1 The Poincaré return map

A *Poincaré return map* is helpful in analysing a system that appears to have periodic behaviour. A Poincaré section is the black line represented in Fig (4) which intersects the solution of the differential equations supposed continuous in points P_1, P_2, P_3 and is useful to reduce the field of study to a one dimensional study. One point of the poincaré section returns one point by the function P.m. We obtain a sequence of points and the stability is analysed by observing this sequence.

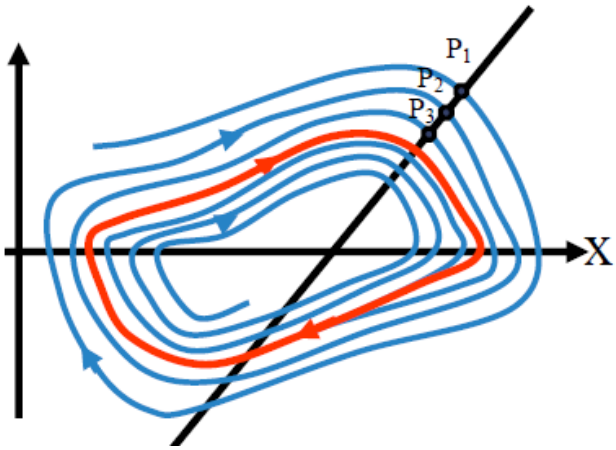

Figure 4: Geometrical representation compass robot

The Poincaré return map is a function which starts from one generic point P_i in the Poincaré section and then evolves following the evolution of the system. For example if we start from P_1 we will arrive in P_2, in this case what we call *Poincaré return map* is a function that associate to the point P_1 the point P_2, and that will associate the point P_2 to the point P_3 and so on.
If we start on the periodic motion we will continue on the periodic motion and we obtain the same point we start from by applying the poincaré function P.m. It is a fixed point for this function. The red line in Fig (4) represents the cycle of a periodic motion and we want to know if our motion will be stable.

We can then say that the Poincaré return map will return a sequence of points. We will then study the coordinates of the sequence of point P_i, given by X_i
In our system the Poincaré section is defined just before impact and thus uses z_e and t_e. The conditions for the compass to be in double support is that both legs touch the ground with a parametrization that implicitly constrains the height of the swing leg to be zero. This parametrization is of dimension (n-1) and is given below as:

$$X = \begin{bmatrix} q_1 \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{14}$$

The value of q_2 is given by the relation that occurs before impact:

$$q_2 = \pi - 2q_1 \tag{15}$$

The Poincaré return map is defined in the matlab funtion *P.m*
We decide to start from a known state just before impact to calculate the impact model and from the new velocity obtained we use the re-labelling equations to define the new values of the joint positions and velocities.
Then we perform one simulation in the single support mode. We have also taken care to limit the joint values to make sure that they lie between $0 < q < 2\pi$

3.2 Periodic passive motion on a slope

A cyclic motion is a fixed point that we define in the Poincaré return map.
Using optimization technique, we define a fixed point in the map such that $X^* = P(X^*)$ and we simulate a step of the robot starting from X^*. In the matlab file *periodicMotion.m*, we define the solution to linear minimization problem that is solved using the *fsolve* function to find the fixed point:

```
X = fsolve('test_periodic',X0,options);
```

We perform several step starting from this fixed point to be on a periodic motion. We then show the joint evolution in a phase plan in Fig. (5) and (6).
For the two cases we will have:

- *Case 1*: a cyclic motion exists close to $q_1 = -0.1860rd$, $\dot{q_1} = -2.0504rd/s$, $\dot{q_2} = -0.0428rd/s$
- *Case 2*: a cyclic motion exists close to $q_1 = -0.1933rd$, $\dot{q_1} = -2.0262rd/s$, $\dot{q_2} = -0.1253rd/s$

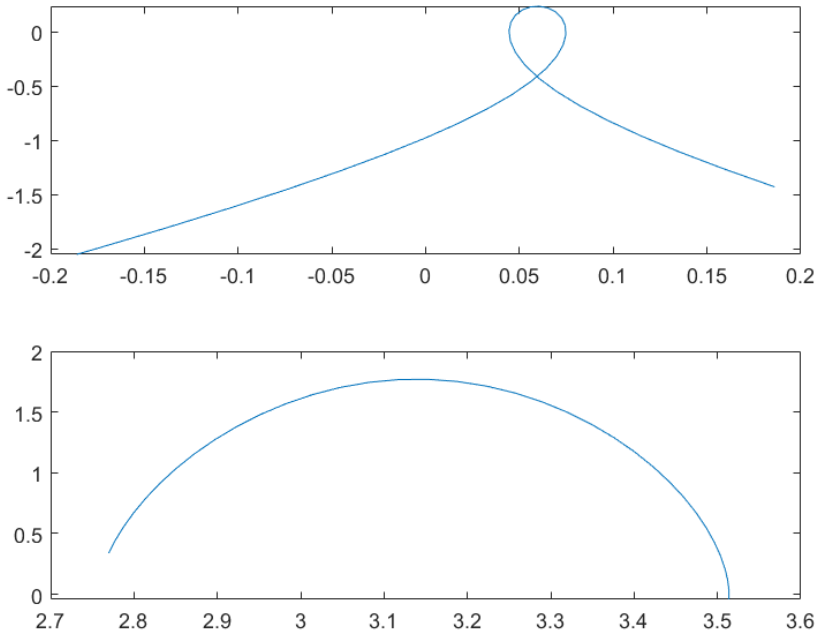

Figure 5: Phase plan case 1 for one step

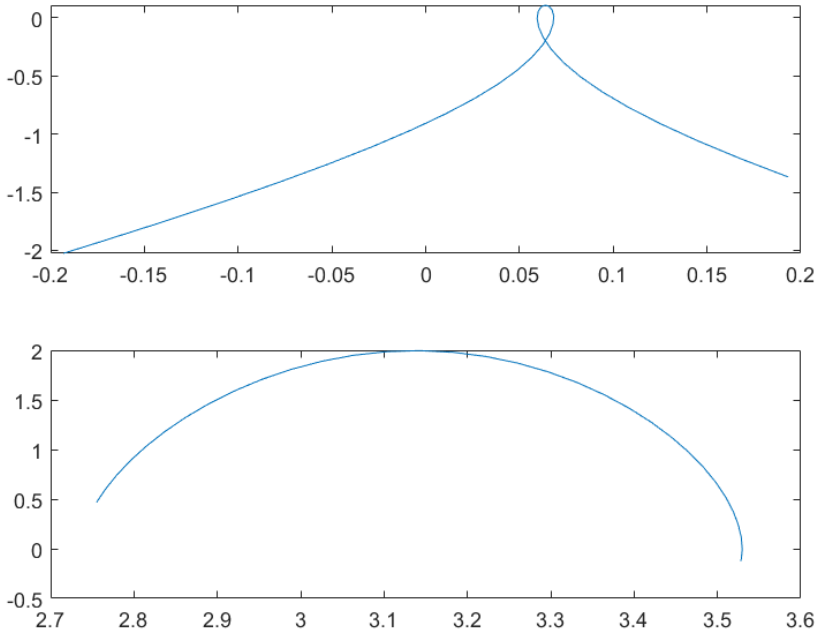

Figure 6: Phase plan case 2 for one step

4 Stability analysis

When simulating several steps, depending on the initial state values, the robot may fall during the motion. We will use stability analysis to know in advance if the motion will be stable or not.
It is possible to use the eigenvalues of the *Jacobian of the Poincaré map* to check the stability of a fixed point. All eighenvalues absolute values should be contained under 1.
We have written the function *Jacobian.m* to compute the Jacobian of the Poincaré map using the relation

$$J_i = \frac{P(X^* + \delta x_i e_i) - P(X^* - \delta x_i e_i)}{2\delta x_i} \tag{16}$$

where J_i represents the ith-column of the Jacobian
In the file *Jacobian.m* we add an error on the position and velocity variables of the state vector of the robot in periodic motion using:

- *on position*: $\pm 0.5e^{-4}$
- *on velocity*: $\pm 0.5e^{-3}$

For the 2 cases, we analyse the stability of the periodic motion, starting for the fixed point, we simulate a walking of 25/30 steps, recording the state variables and drawing the joint evolution in the phase plane and conclude.

Case 1 In this case the periodic motion exists close to $q_1 = -0.1860rd$, $\dot{q_1} = -2.0504rd/s$, $\dot{q_2} = -0.0428rd/s$.
In Eq. (17) and (18) we see the values of the Jacobian matrix and its eigenvalues in the first case. We note that one of the eigenvalues has a value greater than 1 thus the robot will fall at some point during the motion.

$$J = \begin{bmatrix} -4.7934 & 0.3784 & -0.1431 \\ 14.2204 & -0.4987 & 0.1911 \\ 106.8861 & -7.1376 & 2.6823 \end{bmatrix} \tag{17}$$

$$Eigenvalues = \begin{bmatrix} -2.7897 \\ 0.2683 \\ -0.0884 \end{bmatrix} \tag{18}$$

We can show the divergence using the Fig. (7) and simulating the motion for 30 steps.

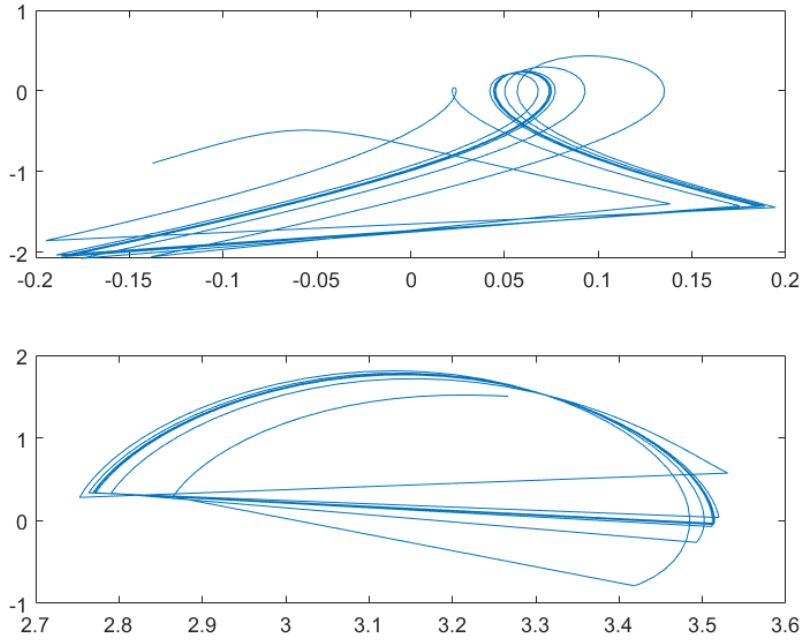

Figure 7: Phase Plan for Case 1

Case 2 In this case the periodic motion exists close to $q_1 = -0.1933rd$, $\dot{q_1} = -2.0262rd/s$, $\dot{q_2} = -0.1253rd/s$.
In Eq. (17) and (19) we see the values of the Jacobian matrix and its eigenvalues computed for the second case.

$$J = \begin{bmatrix} -3.2496 & 0.2946 & -0.1170 \\ 11.7078 & -0.3430 & 0.1618 \\ 90.3918 & -6.3656 & 2.5542 \end{bmatrix} \tag{19}$$

$$Eigenvalues = \begin{bmatrix} -0.7518 \\ -0.5189 \\ 0.2323 \end{bmatrix} \tag{20}$$

All the eighenvalues are of values contained between -1 and 1 so the motion will be stable. We simulate the motion for 30 steps in figure 8. We observe that each plot superimposes the others for all the steps and thus we do not see any divergence.

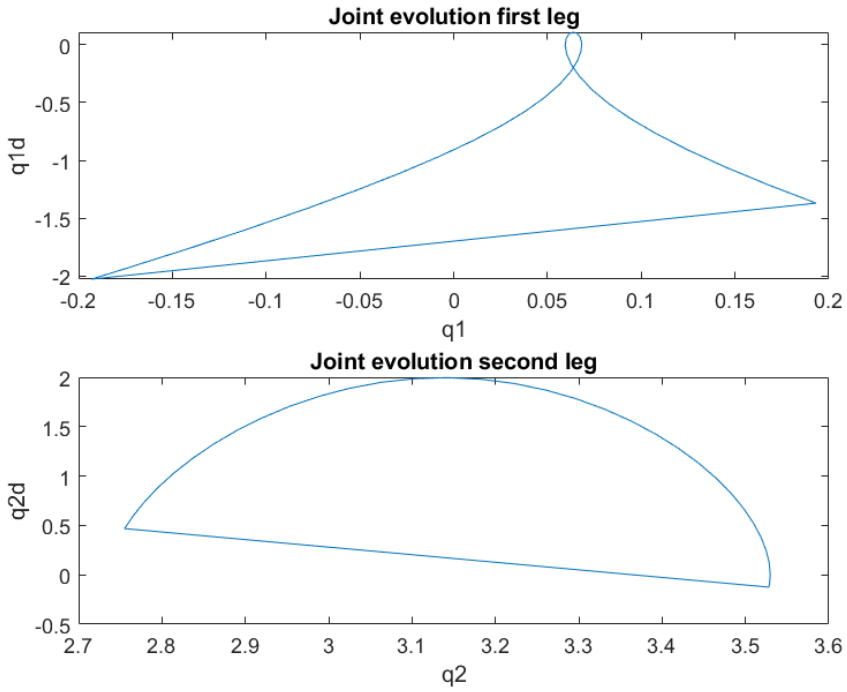

Figure 8: Phase Plan for Case 2

5 Conclusion

In conclusion we started by simulating just one half step by solving the differential equation and using the function of the previous lab. Once arriving at an impact we defined the impact model and relabelled the legs to be able to perform another step using the same method as previously. We then used the poincaré return map to start the motion from a fixed point of the poincaré return map to be on a periodic motion. We analysed the sequence of points given by the poincaré return map to conclude if our motion was stable or not. We observed that for the case 2 the motion was stable. For case 1 it was unstable and the robot would fall after a few steps. Those two cases are very similar. Their inertia and the distance between the position of the center of gravity and position of the hip are different. This last value is 5 centimeters bigger for the case 1. It may be the reason why this case is unstable compared to the case 2.