Report of ME314 Final Project

Part 1

Simulating Multi-link System

Feiyu Chen

This is the 1st part of my report. I will describe my simulation scene, coordinates of objects, calculation of Euler-Lagrange equations and impacts, etc.

More details about how I generalized my code for creating multi-link system are put in the Report_part2.pdf.

1. Proposal

My original proposal was to simulate "a two-link pendulum playing triangular ping-pong against walls" (without user input, just motion and impacts).

However, as I was doing this project, I found it too troublesome to hardcode all the variables, equations, and impacts. So I wrote an API (several encapsulated functions) for creating links (see **Report_part2.pdf**), and then I simulated a multi-link system with 8 DOF, as seen in the image below.

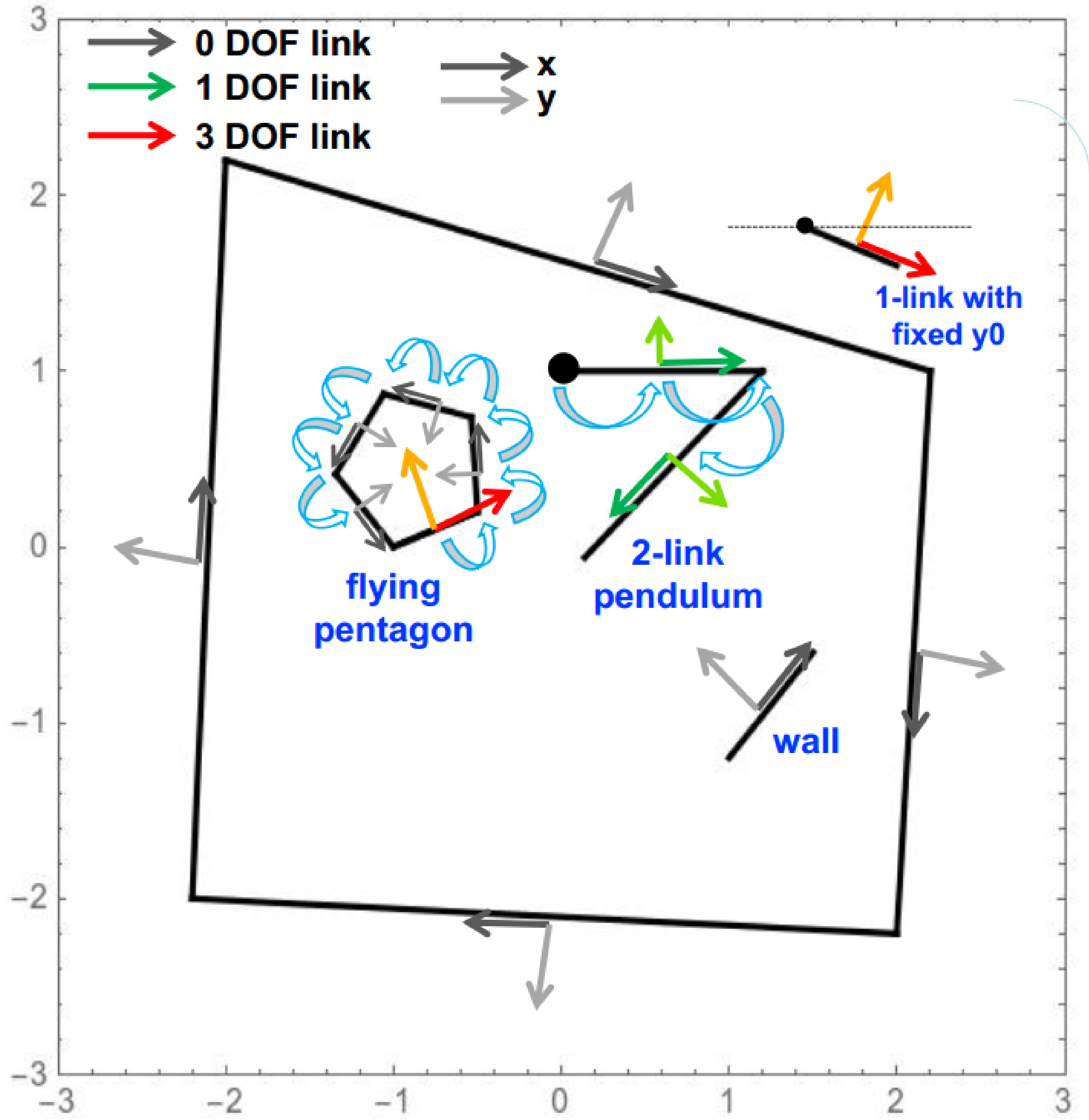


Figure 1. Simulation scene and annotations.

2. Figure of the Scene

The above figure shows my simulation scene as well as annotations of the coordinates and transformations.

There are 5 groups of links in the figure:

1. A pentagon (DOF=3)
2. A 2-link pendulum (DOF=2)
3. A 1-link pendulum (DOF=3, height of the left vertex is fixed)
4. A 1-link wall (DOF=0)
5. Four walls around the main objects (DOF=0)

The x and y axis of each frame are denoted by straight arrows. There are 3 types of frames:

- *Red frame* is for 3-DOF link that has variables $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{theta}$.
- *Green frame* is for 1-DOF link, which has variable $\mathbf{theta}$ and can rotate around a certain pivot.
- *Gray frame* is for 0-DOF link, whose two vertices are fixed in a certain frame.

The transformations between different frames are denoted by curved arrows in blue. These transformations connect up several links to form an object.

3. Calculation of EL-eqs and Impacts

3.1 KE-V

Suppose a link has length $\mathbf{l}$. Then I assume its mass to be $\mathbf{l}$ and inertia to be $\mathbf{l^2}$. The generalized 6x6 body mass M is then obtained.

For each link, I compute the 4x4 matrix representation $\mathbf{g}$ of its center frame. Then calculate the body screw velocity $\mathbf{V}$ using $\mathbf{g}$ and $\mathbf{dg/dt}$. Then the kinetic energy is $\frac{1}{2}\mathbf{V^T M V}$.

3.2 Constraint and External Force

There is no external force in my simulation. But it's easy to add -- simply put something on the right side of EL-eqs.

I added one constraint to the 1-link pendulum on the right-up side of the figure. I first computed the {x,y} coords of its left vertex, and then set $\mathbf{y}$ as 0. Then I passed it to the EL-eqs.

3.3 Detecting Impacts

In my simulation, the impact only happens between a vertex and an edge. An impact happens when:

1. The vertex has a very small distance to the edge.
2. The projection from vertex to edge is on the edge.

I detect the impact by checking these two criteria.

3.4 Impact Update

After detecting one or more impacts, my NDSolve breaks up. Then I do the impact updates for all the impacts one by one, get the new $\mathbf{dq}$, and call the NDSolve again.

The logic of my code for impact detection looks like this:

```
Loop{
   While(no impact && t!=t_end{
       NDSolve
   }
   if(impacts){
       impact update for each impact
```

```
        }else{
            break
        }
    }
```

4. Result

4.1 How to run

To get the simulation result, please open "run_this.nb" and run its cells one by one.

The code will open another three "funcs_xxx.nb", load the functions, and computes the links' motions. It takes about 3 minutes to compute for 15 seconds simulation.

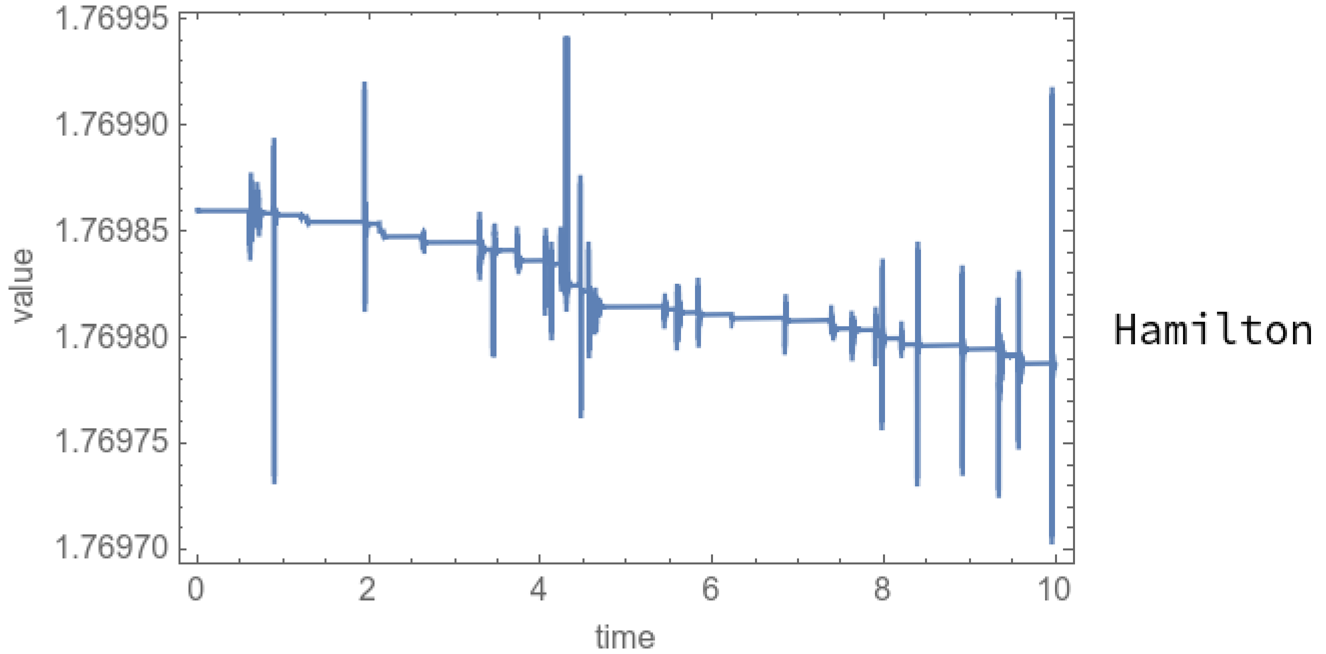
The video I recorded is in "video.mp4".

4.2 What happens in simulation

In my simulation, the pentagon and 2-link pendulum are moving and sometimes impacting with each other. The 1-link pendulum on the right-up side is just for showing that my "Constraint" is working.

4.3 Hamiltonian

Since the energy of my system is supposed to be conserved, I plot the Hamiltonian as seen below. It does conserve over time.



5 Problems

5.1 Fail to detect some impacts

Sometimes, one link will stick into the surface of the polygon.

Guess 1: Need smaller integration step
I tried to use NDSolve's EventLocator's multi-event function, but failed (I couldn't put a list variable there). So I manually code the function of detecting multiple impacts. Its problem is, if I detect an impact at time i, I need to silence it in time i+1, and then it can detect impacts later. Thus, the only case of not detecting impact that I can image is the vertex go through the edge in 2 simulation cycle. (If using EventLocator's multi-event, the special case would be 1 simulation cycle.)

Guess 2: Maybe multi impacts at the same time is also a cause for the problem. But I don't know if there is such logical error in my code.

Solution 1: Decreasing the integral step length (but at a cost of larger computation time).

Solution 2: Use the sign of distance instead of threshold
May be I can solve the problem by detecting the change of sign of the distance to know if a vertex goes through the edge. But its impact criteria would be a little bit more complex than before, and I'm still worrying about the edge cases. I'll try it next time.

5.2 Weird Error about Imaginary Number

Occationally I get such an error:

> LessEqual::nord: Invalid comparison with -0.977809-1.74032*10^-19 I attempted.

I have no idea why my calculation suddenly generates this imaginary number. I tried Re[] and Chop[] to chop out the small imaginary number, but still not totally fix this bug.

When it happens, what I did is to random the velocity and run another simulation.