So here's our solution

The simulator implementation can be much easier in 2D environment, so we will try to design and adjust the controller in 2D environment. See what we can get from this.

In order to add the randomness , we will set a muti-level ground to do the implementation. We will initialize several fixed heights, and choose them randomly along x-axis

In the simulation part of 2D hopper, that is pretty much like what we did in the class. We will use the solve\_ivp to solve the kinematics and dynamics both in flight phase and stance phase.

Here, for the controller part, we concentrate on the forward velocity control. And in the beginning, we will just implement the railbert forward control method, which is quite classical. We will just implement it and see what we get.

The system has obvious steady state error. How to compensate it? Our tactics is add integral item!

17

In the classical PID controller, the **integral** term increases action in relation not only to the error but also the time for which it has persisted. Integral is designed to deal with those errors cannot be solved with time, so here we will introduce the **integral** term to the controller. This is the controller in contio

In the discrete time situation, which will implement in the code, our implementation contains 3 steps. V\_serie is the buffer for velocity.