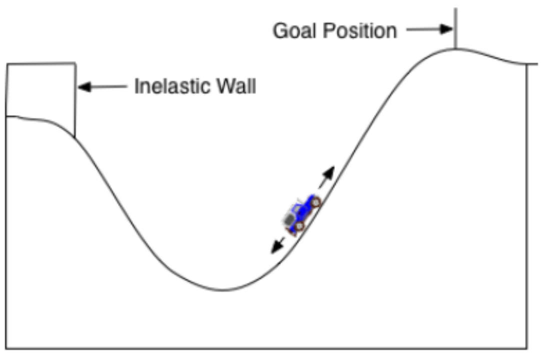
Reinforcement Learning assignment #2

According to the following figure, a car in the bottom of a valley must go up a hill which is too steep for it to drive up directly, so it should bounce back and forth to gain enough velocity to climb up. We will consider a 2D **State space** consisting of **position *x*** and **velocity *v***. The range of allowed positions is [-1.2; 0.5] and the velocity is bounded in the range [-0.07; 0.07].

Actions that the car can perform are accelerations, ***a***, in the range [-1; 1]. At the beginning of each episode, the car starts at the default initial state in the bottom of the valley, *x(0) = 0*



The dynamic equation of the car is given as:

$$v(t+1) = \text{bound}\,[v(t) + 0.001*a(t) - 0.0025 * \cos(3*x(t))]$$

$$x(t+1) = \text{bound}\,[x(t) + v(t+1)]$$

where the function *"bound"* maintains the value of ***x*** and ***v*** within the limits.

Whenever the car reaches the position limits, its velocity is set to zero so that it remains there indefinitely. When the car reaches the top of the hill, it gets a reward of +10. Otherwise, it gets a reward of -1 for every time step in which it doesn't reach the top.

Solve this problem through Q-Learning. A discretization of states and actions is needed in the case of "tabular" Q-Learning.

Let the agent learn for 100 episodes with 100'000 epochs per episode. Use $\gamma = 0.99$ and $\alpha = 0.1$. Repeat the whole process by using the following values for ε: 0.3, 0.2, 0.1, 0.

a) Is the goal achieved? And in positive case…
b) How many times the goal is reached in 100 episodes for each ε?
c) And at which episodes (and epoch) for each ε?

Hint:

To solve the exercise you can use any programming language (Java, Python, R, Matlab, …) or existing RL software libraries (e.g. BURLAP)