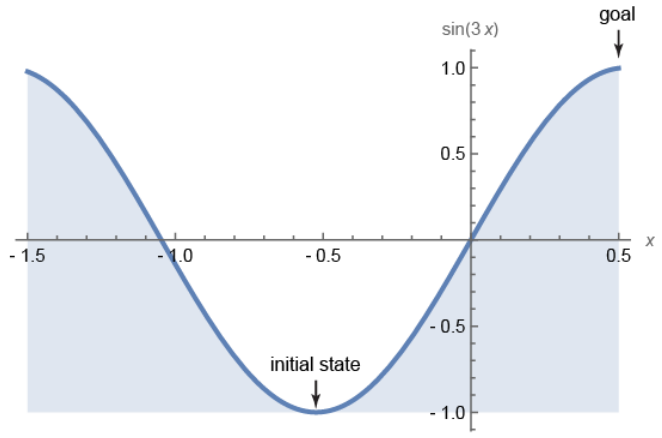
The mountain car problem via the Bellman equations

We have a car that needs to climb a hill. When the car is sitting at the bottom of the hill, the engine is not strong enough to climb this hill. However, it could climb the hill if it went backwards, and then went forwards, thus using gravity to gain speed when it reached the bottom of the hill. Thus, the interesting idea is that the controller needs to figure out that it must move away from the goal in order to eventually reach the goal. Our objective is to see if the Bellman Equations can find this solution.

Below is a plot of the terrain. The state of the car, i.e., its position and velocity, is defined as $[x, \dot{x}]$, and the height of the hill is $\sin(3x)$. The car is starting near the bottom of the hill at state $[-0.5, 0]$, facing forward (toward increasing value of x).



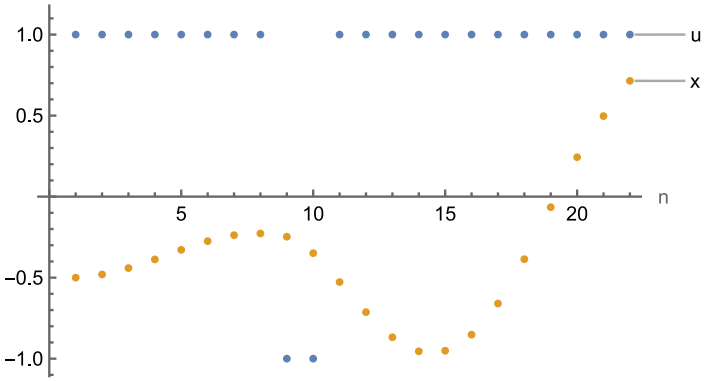
The controller can produce two actions: $u = +1$, indicating move forward, and $u = -1$, indicating move backward. The car's engine will produce a force $F = 0.2u$. The car will respond by moving forward or backward based on the following state update equations:

$$x^{(n+1)} = x^{(n)} + \Delta t \dot{x}^{(n)}$$

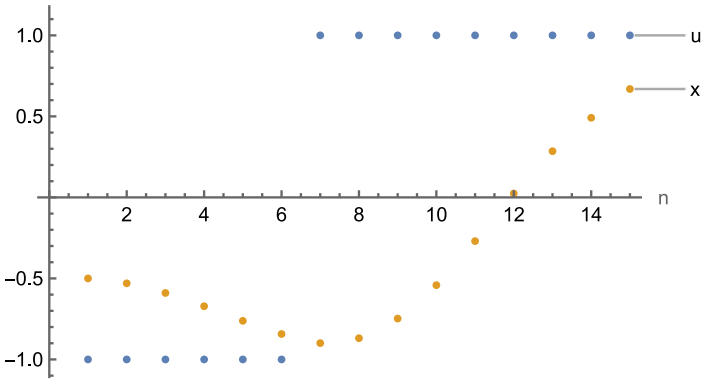
$$\dot{x}^{(n+1)} = \dot{x}^{(n)} + \Delta t(-3g\cos(3x^{(n)}) + \frac{u^{(n)}f}{m} - \frac{k_f}{m}\dot{x}^{(n)})$$

We have the following parameters: $\Delta t = 0.05,\ f = 0.2, g = 9.8, m = 0.02, k_f = 0.5m$. Thus, g is gravity and $\frac{k_f}{m}$ is the coefficient of friction.

Let us begin by trying different sequences of commands. Starting from our initial conditions $x = -0.5$ and $\dot{x} = 0$, we begin by pushing forward. We notice that the car climbs but after 8 steps, it stalls. So, we then we push backwards (2 steps), taking advantage of the momentum to climb the opposite hill, then we push forward (for 12 steps) and this time, the car reaches (and passes) the goal. This policy takes around 20 steps. In the plot below x is shown in orange and u is in blue.



Next, let's try going backward first for 6 steps until the goes backwards up the hill, then go forward for 9 steps. In this case, the car reaches the goal at 14 steps.



Finally, let's find the optimal control policy.

- Implement a state space in which x is bounded by -1.5 and +0.5, with 150 bins, and $\dot{x}$ is bounded by -5 and +5, with 100 bins. Thus, we have a 2D state space of size 150x100.
- Write a function that will take the car from the real space to this grid world, a function that will take the car from the grid world to the real space, and a function that will compute the state transition (given an initial position and action u, compute the next position in the grid world). Note that the grid world is bounded so you want to take care so that if an action takes the car outside the boundary, you will reset its state to be at the edge of the boundary and within it.

In this problem we incur no costs for performing actions, but we get rewarded based on the state of the car. The reward function depends only on the position of the car, not its velocity, and is described as follows:

$$r(x) = x$$

Thus, the closer we get to the top of the hill, the more reward we acquire. Our objective is to maximize the sum of reward until the final time point $T = 20$.

$$J = \sum_{n=1}^{T} r(x^{(n)})$$

Because we incur no cost for actions, our cost per step is simply the state-dependent reward:

$$\alpha^{(n)} = r(x^{(n)})$$

We will be using the Bellman equation so that at each time point n, for every state there will be an optimal action u^*, and an optimum value V.

- Starting at the last time point $n = T$, for each state find the u^* that maximizes the cost per step $\alpha^{(T)}$. Since we don't incur any action cost, set $u^{*(T)} = +1$, and the value of each state as: $V^{(T)}(x, \dot{x}) = \alpha^{(T)}$.
- For time point $n = T - 1$, for each state find the action $u^{*(T-1)}$ that maximizes the sum of cost per step $\alpha^{(T-1)}$ plus the value of the state that the action will take you to. Assign value to each state based on the sum of cost per step and the value of the state that the optimal action will take the car to.

$$V^{(T-1)}(x, \dot{x}) = \alpha^{(T-1)} + V^{(T)}(x^{(T)}, \dot{x}^{(T)} | x^{(T-1)}, \dot{x}^{(T-1)}, u^{*(T-1)})$$

- Repeat for all time point until $n = 1$.
- Plot u^*, x, and $\dot{x}$ as a function of n.

When you solve the Bellman equations, you will get a value function at each time point $n = 1 \dots T$, and thus the optimal action $u^{*(n)}$ at state $\boldsymbol{x}^{(n)}$ is the one that maximizes the value function $V^{(n)}(\boldsymbol{x})$, i.e.,

$$u^{*(n)} = \arg\max_u V^{(n)}(\boldsymbol{x}^{(n)}, u)$$

Now try the following: use only the value function at time $t = 1$. That is:

$$u^{*(n)} = \arg\max_u V^{(1)}(\boldsymbol{x}^{(n)}, u)$$

You should find the surprising result that the solutions (for starting at the initial state at the bottom of the hill) are identical. That is, for this initial state, the value function at time 1 is sufficient to solve the problem for all time.