ECE276B-HW3

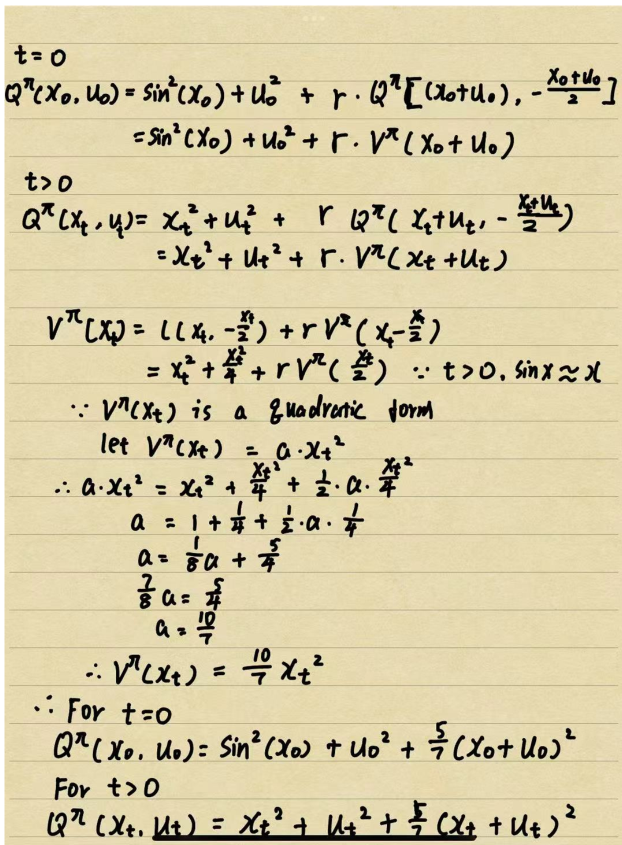
Zhenyu Wu | PID: A69030822

May 31, 2025

1 Q1

1.1 Q1.1

Below is my hand-wrriten solution:



$t=0$

$$Q^\pi(x_0, u_0) = \sin^2(x_0) + u_0^2 + r \cdot Q^\pi\left[(x_0+u_0), -\frac{x_0+u_0}{2}\right]$$
$$= \sin^2(x_0) + u_0^2 + r \cdot V^\pi(x_0+u_0)$$

$t>0$

$$Q^\pi(x_t, u_t) = x_t^2 + u_t^2 + r\, Q^\pi\left(x_t+u_t, -\frac{x_t+u_t}{2}\right)$$
$$= x_t^2 + u_t^2 + r \cdot V^\pi(x_t+u_t)$$

$$V^\pi(x_t) = l\left(x_t, -\frac{x_t}{2}\right) + r V^\pi\left(x_t - \frac{x_t}{2}\right)$$
$$= x_t^2 + \frac{x_t^2}{4} + r V^\pi\left(\frac{x_t}{2}\right) \quad \because t>0, \sin x \approx x$$

$\because V^\pi(x_t)$ is a quadratic form

let $V^\pi(x_t) = a \cdot x_t^2$

$$\therefore a \cdot x_t^2 = x_t^2 + \frac{x_t^2}{4} + \frac{1}{2} \cdot a \cdot \frac{x_t^2}{4}$$
$$a = 1 + \frac{1}{4} + \frac{1}{2} \cdot a \cdot \frac{1}{4}$$
$$a = \frac{1}{8}a + \frac{5}{4}$$
$$\frac{7}{8}a = \frac{5}{4}$$
$$a = \frac{10}{7}$$

$$\therefore V^\pi(x_t) = \frac{10}{7} x_t^2$$

$\therefore$ For $t=0$

$$Q^\pi(x_0, u_0) = \sin^2(x_0) + u_0^2 + \frac{5}{7}(x_0+u_0)^2$$

For $t>0$

$$Q^\pi(x_t, u_t) = \underline{x_t^2 + u_t^2 + \frac{5}{7}(x_t} + u_t)^2$$

Figure 1: Q1.1

1.2 Q1.2

Below is my hand-wrriten solution:

2. For $t=0$

$$Q^\pi(x_0, u_0) = \sin^2(x_0) + u_0^2 + \frac{5}{7}(x_0+u_0)^2$$

$$\frac{dQ^\pi}{du} = 2u + \frac{10}{7}(x+u)$$

$$0 = 2u + \frac{10}{7}x + \frac{10}{7}u$$

$$24u = -10x$$

$$u = -\frac{5}{12}x$$

For $t>0$

$$Q^\pi(x_t, u_t) = x_t^2 + u_t^2 + \frac{5}{7}(x_t+u_t)^2$$

$$\frac{dQ^\pi}{du} = 2u + \frac{10}{7}x + \frac{10}{7}u$$

$$u = -\frac{5}{12}x$$

$$\therefore \pi'(x) = -\frac{5}{12}x$$

Figure 2: Q1.2

2 Q2

2.1 Q2.1

Below is my hand-wrriten solution:

2. $w_t \sim N(0, \frac{1}{2})$

$l(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2 \qquad r = \frac{1}{2}$

$V^*(x) = ax^2 + bx + c$

$x_{t+1} = \sqrt{2}x_t + u_t + 2w_t$

$\because w \sim N(0, \frac{1}{2})$

$2w \sim N(0, 2)$, let $\varepsilon = 2w$

$\therefore x' = \sqrt{2}x + u + \varepsilon$, $\varepsilon \sim N(0, 2)$

$\varepsilon = x' - (\sqrt{2}x + u)$

$\therefore P(x' \mid x, u) = P(\varepsilon = x' - (\sqrt{2}x + u))$

$$= \frac{1}{\sqrt{4\pi}} \exp\left(-\frac{(x' - \sqrt{2}x - u)^2}{4}\right)$$

$\therefore x' \sim N(\sqrt{2}x + u, 2)$

$V^*(x) = \min\limits_u \left(\frac{1}{2}x^2 + \frac{1}{2}u^2 + r\,\mathbb{E}(V^*(x'))\right)$

For $\mathbb{E}(V^*(x'))$

$\because V^*(x) = ax^2 + bx + c$

$\therefore \mathbb{E}(V^*(x')) = \mathbb{E}(ax'^2 + bx' + c)$

$\because x' = \sqrt{2}x + u + 2w$, $w \sim N(0, \frac{1}{2})$

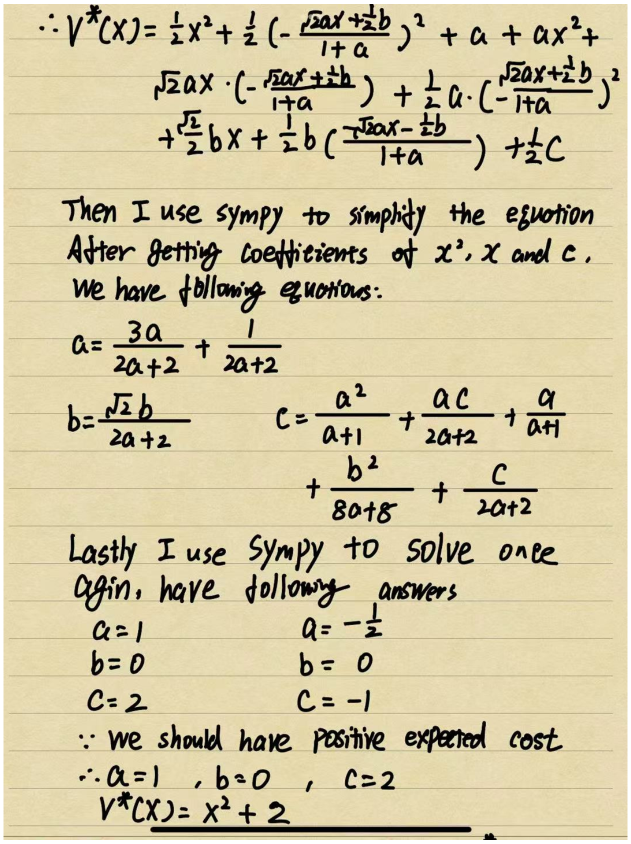
For x', mean $\mu = \sqrt{2}x + u$, variance $\delta^2 = 2$

$\therefore \mathbb{E}(x') = \mu = \sqrt{2}x + u$

Figure 3: Q2.1

$$\mathbb{E}(X'^2) = \delta^2 + \mathbb{E}(X')^2 = 2 + u^2$$
$$= 2 + (\sqrt{2}x + u)^2$$
$$\therefore \mathbb{E}(V^*(x')) = a\cdot[2 + (\sqrt{2}x + u)^2] + b\cdot(\sqrt{2}x + u) + c$$
$$= a\cdot(2 + 2x^2 + 2\sqrt{2}xu + u^2) + \sqrt{2}bx + bu + c$$
$$= 2a + 2ax^2 + 2\sqrt{2}axu + au^2 + \sqrt{2}bx + bu + c$$
$$\therefore r\cdot\mathbb{E}(V^*(x')) = a + ax^2 + \sqrt{2}axu + \frac{1}{2}au^2 + \frac{\sqrt{2}}{2}bx + \frac{1}{2}bu + \frac{1}{2}c$$
$$\therefore V^*(x) = \min_u \left[\frac{1}{2}x^2 + \frac{1}{2}u^2 + a + ax^2 + \sqrt{2}axu + \frac{1}{2}au^2 + \frac{\sqrt{2}}{2}bx + \frac{1}{2}bu + \frac{1}{2}c\right]$$
$$\frac{dV^*}{du} = u + \sqrt{2}ax + au + \frac{1}{2}b = 0$$
$$(1+a)u = -\sqrt{2}ax - \frac{1}{2}b$$
$$u = \frac{-\sqrt{2}ax - \frac{1}{2}b}{1+a} = \frac{-\sqrt{2}a}{1+a}x + \frac{-\frac{1}{2}b}{1+a}$$

Figure 4: Q2.1



$$\therefore V^*(x) = \frac{1}{2}x^2 + \frac{1}{2}\left(-\frac{\sqrt{2}ax + \frac{1}{2}b}{1+a}\right)^2 + a + ax^2 + \sqrt{2}ax\cdot\left(-\frac{\sqrt{2}ax+\frac{1}{2}b}{1+a}\right) + \frac{1}{2}a\cdot\left(-\frac{\sqrt{2}ax+\frac{1}{2}b}{1+a}\right)^2 + \frac{\sqrt{2}}{2}bx + \frac{1}{2}b\left(\frac{-\sqrt{2}ax - \frac{1}{2}b}{1+a}\right) + \frac{1}{2}c$$

Then I use sympy to simplify the equation
After getting coefficients of x^2, x and c,
we have following equations:

$$a = \frac{3a}{2a+2} + \frac{1}{2a+2}$$

$$b = \frac{\sqrt{2}b}{2a+2} \qquad c = \frac{a^2}{a+1} + \frac{ac}{2a+2} + \frac{a}{a+1} + \frac{b^2}{8a+8} + \frac{c}{2a+2}$$

Lastly I use Sympy to solve once agin, have following answers

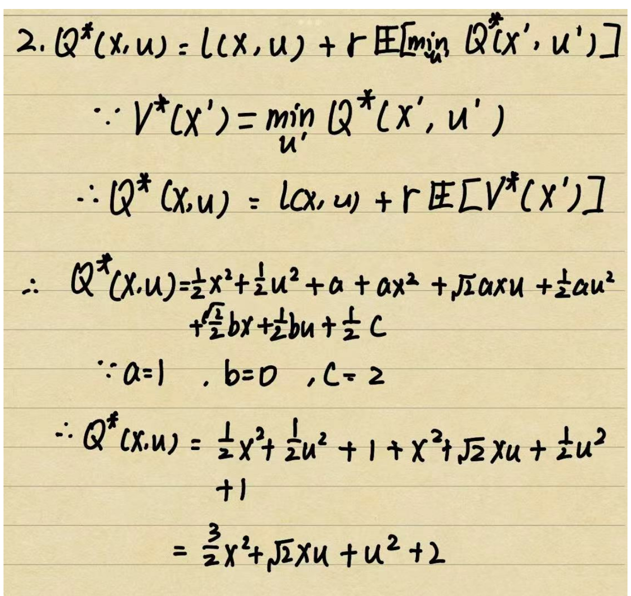
$a=1$ $\quad$ $a=-\frac{1}{2}$
$b=0$ $\quad$ $b=0$
$c=2$ $\quad$ $c=-1$

$\because$ we should have positive expected cost
$\therefore a=1$, $b=0$, $c=2$
$V^*(x) = x^2 + 2$

Figure 5: Q2.1

2.2 Q2.2

Below is my hand-wrriten solution:



$$2.\ Q^*(x,u) = l(x,u) + r\,\mathbb{E}[\min_{u'} Q^*(x',u')]$$

$$\because V^*(x') = \min_{u'} Q^*(x',u')$$

$$\therefore Q^*(x,u) = l(x,u) + r\,\mathbb{E}[V^*(x')]$$

$$\therefore Q^*(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2 + a + ax^2 + \sqrt{2}axu + \frac{1}{2}au^2 + \frac{\sqrt{2}}{2}bx + \frac{1}{2}bu + \frac{1}{2}c$$

$\because a=1$, $b=0$, $c=2$

$$\therefore Q^*(x,u) = \frac{1}{2}x^2 + \frac{1}{2}u^2 + 1 + x^2 + \sqrt{2}xu + \frac{1}{2}u^2 + 1$$

$$= \frac{3}{2}x^2 + \sqrt{2}xu + u^2 + 2$$

Figure 6: Q2.2

3 Q3

3.1 Q3.1

In the given Discounted problem, we have state space $\mathcal{X} = \{1, 2\}$, control space $\mathcal{U} = \{a, b\}$. From the given transition matrix P^a and P^b, we define the transition model as follow:

$$p_f(j|i, a) = P^a_{i,j}$$

$$p_f(j|i, b) = P^b_{i,j}$$

where i, j entries in P^a specifies the transition probability from state i to j under action a; i, j entries in P^b specifies the transition probability from state i to j under action b.

Firstly, we need to add virtual terminal state to the state space, $\mathcal{X} = \mathcal{X} \cup \{\tau\}$ and $\mathcal{T} = \{\tau\}$. Therefore, the state space is $\mathcal{X} = \{1, 2, \tau\}$ We will use the same control space. Then, for the motion model, we will use $1 - \gamma$ as the probability of terminate. Therefore, we have

$$p_f(j|i, a) = \gamma p_f(j|i, a) = \gamma P^a_{i,j}, \quad p_f(j|i, b) = \gamma p_f(j|i, b) = \gamma P^b_{i,j}, \quad \text{for j} \neq \tau$$

$$p_f(j|i, a) = p_f(j|i, b) = 1 - \gamma, \quad \text{for j} = \tau$$

$$p_f(j|\tau, a) = p_f(j|\tau, b) = 0, \quad \text{for j} \neq \tau$$

$$p_f(\tau|\tau, a) = p_f(\tau|\tau, b) = 1$$

Put together, we have following transition matrices (we consider $\tau = 3$):

$$\mathbf{P}^a := \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P}^b := \begin{bmatrix} 0.3 & 0.5 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}$$

For the stage cost, we have

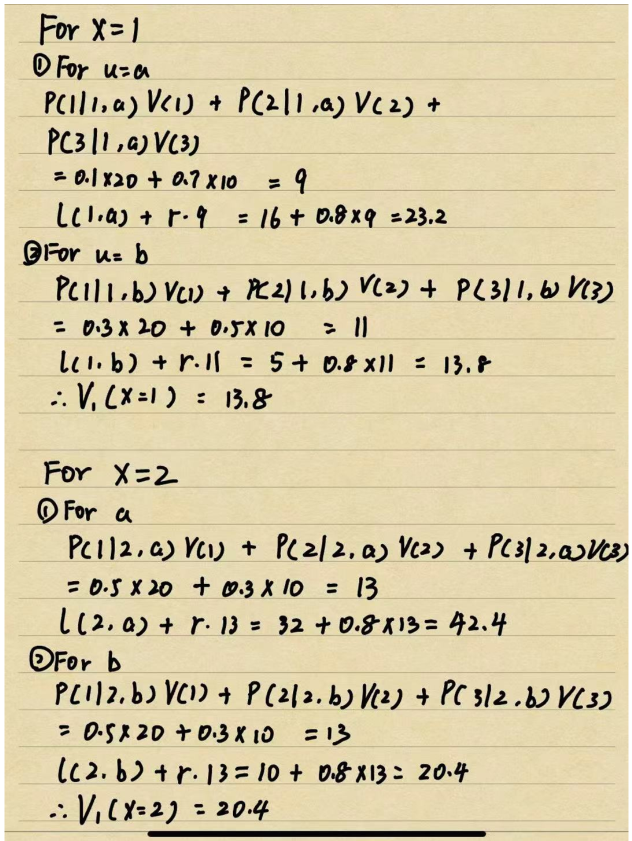
$$\ell(x, u) := \begin{cases} 16x & \text{if } u = a \text{ and } x \neq \tau \\ 5x & \text{if } u = b \text{ and } x \neq \tau. \\ 0 & \text{if } x = \tau \end{cases}$$

Lastly, for the terminal cost, we have $q(x_\tau) = 0$. The problem terminates at $T := inf\{t \geq 0 | x_t = 3\}$. Our objective is

$$\begin{aligned} V^*(\mathbf{x}) = \min_\pi \; & V^\pi(\mathbf{x}) := \mathbb{E}\left[q(x_\tau) + \sum_{t=0}^{T-1} \ell(x_t, \pi(x_t)) \,\middle|\, x_0 = \mathbf{x} \right] \\ \text{s.t.} \quad & x_{t+1} \sim p_f(\cdot \mid x_t, \pi(x_t)), \\ & x_t \in \mathcal{X}, \quad \pi(x_t) \in \mathcal{U} \end{aligned}$$

3.2 Q3.2

Below is my hand-written solution:



For $X=1$

① For $u=a$

$P(1|1,a)V(1) + P(2|1,a)V(2) + P(3|1,a)V(3)$

$= 0.1\times 20 + 0.7\times 10 = 9$

$L(1,a) + r\cdot 9 = 16 + 0.8\times 9 = 23.2$

② For $u=b$

$P(1|1,b)V(1) + P(2|1,b)V(2) + P(3|1,b)V(3)$

$= 0.3\times 20 + 0.5\times 10 = 11$

$L(1,b) + r\cdot 11 = 5 + 0.8\times 11 = 13.8$

$\therefore V_1(X=1) = 13.8$

For $X=2$

① For a

$P(1|2,a)V(1) + P(2|2,a)V(2) + P(3|2,a)V(3)$

$= 0.5\times 20 + 0.3\times 10 = 13$

$L(2,a) + r\cdot 13 = 32 + 0.8\times 13 = 42.4$

② For b

$P(1|2,b)V(1) + P(2|2,b)V(2) + P(3|2,b)V(3)$

$= 0.5\times 20 + 0.3\times 10 = 13$

$L(2,b) + r\cdot 13 = 10 + 0.8\times 13 = 20.4$

$\therefore V_1(X=2) = 20.4$

Figure 7: Q3.2

3.3 Q3.3

Let ℓ^a denotes the stage cost if $u = a$ and ℓ^b denotes the stage cost if $u = b$, and we have

$$\ell^a = [16 \times 1, 16 \times 2]^T = [16, 32]^T$$

$$\ell^b = [5 \times 1, 5 \times 2]^T = [5, 10]^T$$

since $x \in \{1, 2\}$. Then, based on the lecture slides, we can formulate the given discounted problem as LP problem as follow:

$$\max_V w^T V$$
$$\text{s.t.} \quad (I - \gamma P^u)V \leq \ell^u, \forall u \in \mathcal{U}$$

where P^u is the 2×2 transition matrix, $V \in R^2$ is the value vector, $w \in R^2$ is the weight vector and $w_i > 0 \quad \forall i = \{1, 2\}$.

In the implementation, I initialize w as random vector where each entry is uniformly sampled from $(0, 1]$. The result I have is $V^* = [35.4166667439.58333341]$, and $V^*(1) = 35.41666674$, $V^*(2) = 39.58333341$

4 Q4

Below is my hand-written process:

TD(0):

$V^\pi(x) \leftarrow V^\pi(x) + \alpha[l(x,u) + \gamma V^\pi(x') - V^\pi(x)]$

① 1st episode

$1 \xrightarrow{-3} 1 : V^\pi(1) = 0.5 + 0.2[-3 + 0.5 - 0.5]$
$= -0.1$

$1 \xrightarrow{-2} 2 : V^\pi(1) = -0.1 + 0.2[-2 + 0.5 + 0.1]$
$= -0.38$

$2 \xrightarrow{4} 1 : V^\pi(2) = 0.5 + 0.2[4 - 0.38 - 0.5]$
$= 1.124$

$1 \xrightarrow{-3} 2 : V^\pi(1) = -0.38 + 0.2[-3 + 1.124 + 0.38]$
$= -0.6792$

$2 \xrightarrow{3} 3 : V^\pi(2) = 1.124 + 0.2[3 + 0.5 - 1.124]$
$= 1.5992$

② 2nd episode

$2 \xrightarrow{4} 1 : V^\pi(2) = 1.5992 + 0.2[4 - 0.6792 - 1.5992]$
$= 1.94352$

$1 \xrightarrow{-2} 2 : V^\pi(1) = -0.6792 + 0.2[-2 + 1.94352 + 0.6792]$
$= -0.354656$

$2 \xrightarrow{3} 3 : V^\pi(2) = 1.94352 + 0.2[3 + 0.5 - 1.94352]$

Figure 8: Q4

$= 2.254816$

$\therefore V^\pi(1) = -0.554656$
$V^\pi(2) = 2.254816$
$V^\pi(3) = 0.5$

Figure 9: Q4

The result I have is $V(1) = -0554656$, $V(2) = 2.254816$, $V(3) = 0.5$

References

In [1]:

```
import sympy as sp
import numpy as np


x, u, a, b, c = sp.symbols('x u a b c')


gamma = sp.Rational(1, 2)
sqrt2 = sp.sqrt(2)

# stage cost
stage_cost = sp.Rational(1, 2) * x**2 + sp.Rational(1, 2) * u**2

# expected value of V*
E_x_next_squared = 2*x**2 + 2*sqrt2*x*u + u**2 + 2
E_x_next = sqrt2*x + u
E_V_next = a*E_x_next_squared + b*E_x_next + c

# Bellman equation RHS (before minimization)
bellman_rhs = stage_cost + gamma * E_V_next

# find optimal u by taking derivative and setting to zero
du_bellman = sp.diff(bellman_rhs, u)
u_star = sp.solve(du_bellman, u)[0]


# substitute back into bellman equation
bellman_rhs_optimal = bellman_rhs.subs(u, u_star)
bellman_rhs_optimal_simplified = sp.simplify(bellman_rhs_optimal)
bellman_expanded = sp.expand(bellman_rhs_optimal_simplified)
bellman_collected = sp.collect(bellman_expanded, x)
coeff_x2 = bellman_collected.coeff(x, 2)
coeff_x1 = bellman_collected.coeff(x, 1)
coeff_x0 = bellman_collected.coeff(x, 0)



# set up equations by matching coefficients
eq1 = sp.Eq(a, coeff_x2)
eq2 = sp.Eq(b, coeff_x1)
eq3 = sp.Eq(c, coeff_x0)



# solve the system of equations
# first solve for a from eq1
a_solutions = sp.solve(eq1, a)

# choose positive solution for a
a_val = 1  # From (2a + 1)(a - 1) = 0, we choose a = 1


# substitute a = 1 into eq2 to find b
eq2_with_a = eq2.subs(a, a_val)
b_val = sp.solve(eq2_with_a, b)[0]


# substitute a = 1 and b = 0 into eq3 to find c
```

```python
eq3_with_a_b = eq3.subs([(a, a_val), (b, b_val)])
c_val = sp.solve(eq3_with_a_b, c)[0]


print("FINAL SOLUTION:")
print(f"a = {a_val}")
print(f"b = {b_val}")
print(f"c = {c_val}")
print(f"V*(x) = {a_val}x^2 + {b_val}x + {c_val}")
```

```
FINAL SOLUTION:
a = 1
b = 0
c = 2
V*(x) = 1x^2 + 0x + 2
```

In [6]:

```python
import numpy as np

# Transition probability matrices
PA = np.array([[0.1, 0.7, 0.2],
               [0.5, 0.3, 0.2],
               [0.0, 0.0, 1.0]])

PB = np.array([[0.3, 0.5, 0.2],
               [0.5, 0.3, 0.2],
               [0.0, 0.0, 1.0]])

# State and control spaces
STATE = np.array([1, 2, 3])
CONTROL = ['a', 'b']
TERMINAL_STATE = 3
TERMINAL_COST = 0

def get_stage_cost(state, control):
    """Get the stage cost for a given state and control"""
    if state == TERMINAL_STATE:
        return 0
    else:
        if control == 'a':
            return 16 * state
        else:
            return 5 * state

def get_transition_prob(state, control, next_state):
    """Get transition probability P(next_state | state, control)"""
    state_index = state - 1  # Convert to 0-based index
    next_state_index = next_state - 1
    if control == 'a':
        return PA[state_index, next_state_index]
    else:
        return PB[state_index, next_state_index]

def get_updated_value(state, value_func, gamma=0.8):
    """Compute the updated value for a single state using Bellman operator"""
    if state == TERMINAL_STATE:
        return TERMINAL_COST

    min_value = np.inf
    for control in CONTROL:
        # Compute expected value for this control
```

```
        expected_value = get_stage_cost(state, control)
        for next_state in STATE:
            prob = get_transition_prob(state, control, next_state)
            expected_value += gamma * prob * value_func[next_state - 1]

        # Keep track of minimum value across all controls
        if expected_value < min_value:
            min_value = expected_value

    return min_value

def value_iteration(initial_value, gamma=0.8, max_iter=1):
    """Perform value iteration for specified number of iterations"""
    value = initial_value.copy()

    for iteration in range(max_iter):
        new_value = value.copy()

        # Update values for all non-terminal states
        for state in STATE:
            if state != TERMINAL_STATE:
                new_value[state - 1] = get_updated_value(state, value, gamma)

        # Update value function
        value = new_value

        print(f"After iteration {iteration + 1}:")
        print(f"V(1) = {value[0]:.2f}")
        print(f"V(2) = {value[1]:.2f}")
        print(f"V(3) = {value[2]:.2f}")
        print()

    return value

# Initial value function
V0 = np.array([20.0, 10.0, 0.0])

print("Initial value function:")
print(f"V0(1) = {V0[0]}")
print(f"V0(2) = {V0[1]}")
print(f"V0(3) = {V0[2]}")
print()

# Perform one iteration of value iteration
V1 = value_iteration(V0, gamma=0.8, max_iter=1)


print("Final result after one iteration:")
print(f"V1 = [{V1[0]:.1f}, {V1[1]:.1f}, {V1[2]:.1f}]")
```

```
Initial value function:
V0(1) = 20.0
V0(2) = 10.0
V0(3) = 0.0

After iteration 1:
V(1) = 13.80
V(2) = 20.40
V(3) = 0.00

Final result after one iteration:
V1 = [13.8, 20.4, 0.0]
```

In [ ]:

```python
import numpy as np
import cvxpy as cp

# params
n_control = 2
n_state = 2

# weights positive
w = np.random.rand(2)

# Problem parameters
gamma = 0.8

# Transition probability matrices
P_a = np.array([[1/8, 7/8],
                [5/8, 3/8]])

P_b = np.array([[3/8, 5/8],
                [5/8, 3/8]])
identity = np.eye(n_state)

# stage cost
cost_a = np.array([16., 32.])
cost_b = np.array([5., 10.])

# decistion variables
value = cp.Variable(n_state)

# LP objective
objective = cp.Maximize(w.T @ value)

# LP constraints
constraints = []
# control a
constraints.append((identity - gamma * P_a) @ value <= cost_a)
# control b
constraints.append((identity - gamma * P_b) @ value <= cost_b)

# solve
problem = cp.Problem(objective, constraints)
problem.solve()
print(f"Optimal value: {value.value}")
```

```
Optimal value: [35.41666674 39.58333341]
```

In [ ]: