Московский государственный технический университет
Факультет ИУ «Информатика и системы управления»
Кафедра ИУ-1 «Системы автоматического управления»

ОТЧЕТ

по лабораторной работе №1

«Динамическое программирование Беллмана»

по дисциплине

«Оптимальное управление»

Выполнили: Кочеткова А.А.
Жидкова М.А.
Группа: ИУ1-73

Проверил: Щербак О. Ю.

Работа выполнена: 20/11/2024
Отчет сдан: 20/11/2024
Оценка

Динамическое программирование Беллмана

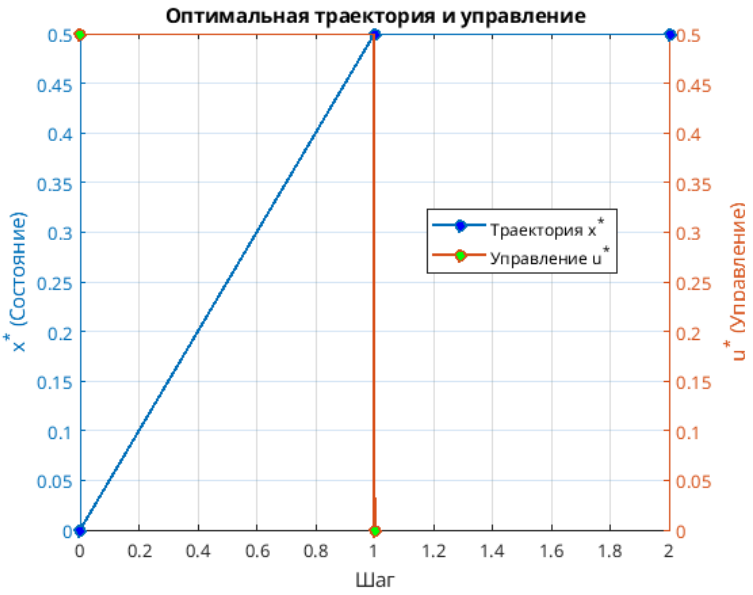
Цель работы: реализовать решение задачи выбора оптимальной траектории методом динамического программирования Беллмана в среде научных вычислений MathWorks MATLAB.

Вариант 1

Условия задачи
Разностное уравнение системы: $a=0;\quad b=1;\quad \lambda=2;\quad \Delta t=1$ $x(k+1)=[1+a\,\Delta t]\,x(k)+b\,\Delta t\,u(k)$
Ограничения: $0.0\le x(k)\le 1.5$ $-1.0\le u(k)\le 1.0$
Функционал качества: $J=x^2(k+1)+\lambda\,\Delta t\,u^2(k)$
При этом x и u квантованы по уровню: $x(k)=[0.0\quad 0.5\quad 1.0\quad 1.5]^{\mathrm{T}}$ $u(k)=[-1.0\quad -0.5\quad 0.0\quad 0.5\quad 1.0]^{\mathrm{T}}$

Количество шагов N	Начальная точка x_0	Конечная точка x_f
2	0.0	0.5

Полученный график:



Проблемы, возникшие в ходе решения задачи:

Результаты для оптимальной траектории: Восстановленная траектория (шаги x) показывает, что система остается в состоянии x=0.0 при управлении u=0.0, вместо того, чтобы двигаться к x=0.5, как того требует задача.

Квантование состояний: Диапазон x слишком узкий, и кажется, что алгоритм "зациклился" на минимальной стоимости в состоянии x=0.0. Стоимость перехода к x=0.5 выше из-за значений функции стоимости J(x,u).

Неподходящий функционал качества:

- Функционал качества J(x,u)=x^2+λ · u^2 сильно штрафует за отклонения от x=0.0, особенно для состояний с ненулевым управлением u
- Для достижения цели x=0.5 требуется изменить функционал так, чтобы он поощрял нахождение вблизи этой цели.

Квантование диапазона состояний:

- При таком разреженном диапазоне оптимальное x=0.5 может быть проигнорировано из-за недостатка интерполяции или из-за высокого значения функционала.

Поэтому добавили в функционал качества целевое значение:

Реализация алгоритма:

```
clear all
clc
% возможные квантованные состояния и управления
range_x = [0.0, 0.5, 1.0, 1.5];
range_u = [-1.0, -0.5, 0.0, 0.5, 1.0];
n = 2;
function x_k1 = razn(x_k, u_k)
  a = 0;
  b = 1;
  delta_t = 1;
  x_k1 = (1 + a * delta_t) * x_k + b * delta_t * u_k;
end
function J = fun_J(x_k1, u_k)
  lambda = 2;        % вес управления
  x_target = 0.5;    % цель
  delta_t = 1;
  J = (x_k1 - x_target)^2 + lambda * delta_t * u_k^2;
end
% функция линейной интерполяции для нахождения стоимости
function cost_interp = interpolate_cost(cost_array, range_x, x_next)
% ищем диапазон, в каких индексах находится x
  idx1 = find(range_x <= x_next, 1, 'last');
  idx2 = find(range_x >= x_next, 1, 'first');
```

```
    if isempty(idx1) || isempty(idx2)
        cost_interp = inf; % штрафуем за невозможный переход
    elseif range_x(idx1) == x_next
        cost_interp = cost_array(idx1);
    elseif range_x(idx2) == x_next
        cost_interp = cost_array(idx2);
    else
        x1 = range_x(idx1);
        x2 = range_x(idx2);
        c1 = cost_array(idx1);
        c2 = cost_array(idx2);
        cost_interp = c1 + (c2 - c1) * (x_next - x1) / (x2 - x1);
    end
end
function [u_opt, x_traj, u_traj] = dynamic_prog(range_x, range_u, fun_razn, fun_J, n)
    x_0 = 0.0;
    % инициализируем функцию стоимости
    C = inf(length(range_x), n+1);
    u_opt = zeros(length(range_x), n);
    % Стоимость при нулевом управлении для x(2)
    for i = 1:length(range_x)
        C(i, n+1) = fun_J(range_x(i), 0);
    end
    % Итерации по шагам
    for k = n:-1:1
        fprintf('Таблица %d\n', k);
        fprintf('%8s%d%s %8s%d%s %8s%d%s %8s%d%d %8s%d%d %10s%d%s%d%s\n', ...
            'x(', k-1,')', ' u(', k-1,')', 'x(', k,')', 'C', k-1, k, 'J*', k-1, k, 'u*(x(',k-1,'),',k-1,')');
        fprintf('-------------------------------------------------------------------------\n');
        for i = 1:length(range_x)
            min_cost = inf;
            best_u = 0;
            for j = 1:length(range_u)
                u_k = range_u(j);
                x_next = fun_razn(range_x(i), u_k);
                if x_next < min(range_x) || x_next > max(range_x)
                    cost = inf;
                else
                    cost = interpolate_cost(C(:, k+1), range_x, x_next) + fun_J(x_next, u_k);
                end
                if cost < min_cost
                    min_cost = cost;
                    best_u = u_k;
                end
                    fprintf(' %9.1f %9.1f %9.1f %12.2f %12.2f %12.1f\n', range_x(i), u_k, x_next, cost, min_cost,
best_u);
            end
            C(i, k) = min_cost;
            u_opt(i, k) = best_u;
        end
    end
    % Восстановление оптимальной траектории
    x_traj = zeros(1, n+1);
    u_traj = zeros(1, n);
    x_traj(1) = x_0;
    %fprintf('Восстановление траектории:\n');
    for k = 1:n
        idx = find(abs(range_x - x_traj(k)) < 1e-6, 1);
        if ~isempty(idx)
            u_traj(k) = u_opt(idx, k);
```

```
        x_traj(k+1) = fun_razn(x_traj(k), u_traj(k));
        fprintf('Шаг %d: x = %.2f, u = %.2f, x_next = %.2f\n', k-1, x_traj(k), u_traj(k), x_traj(k+1));
      else
        x_traj(k+1) = nan;
        u_traj(k) = nan;
      end
    end
end
[u_opt, x_traj, u_traj] = dynamic_prog(range_x, range_u, @razn, @fun_J, n);
% Построение графиков
figure;
yyaxis left;
plot(0:n, x_traj, '-o', 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
ylabel('x^* (Состояние)');
hold on;
yyaxis right;
stairs(0:n-1, u_traj, '-o', 'MarkerFaceColor', 'g', 'LineWidth', 1.5);
ylabel('u^* (Управление)');
xlabel('Шаг');
title('Оптимальная траектория и управление');
legend('Траектория x^*', 'Управление u^*', 'Location', 'Best');
grid on;
grid on;
```

Итоговые таблицы:

Таблица 2

x(1)	u(1)	x(2)	C12	J*12	u*(x(1),1)
0.0	-1.0	-1.0	Inf	Inf	0.0
0.0	-0.5	-0.5	Inf	Inf	0.0
0.0	0.0	0.0	0.50	0.50	0.0
0.0	0.5	0.5	0.50	0.50	0.0
0.0	1.0	1.0	2.50	0.50	0.0
0.5	-1.0	-0.5	Inf	Inf	0.0
0.5	-0.5	0.0	1.00	1.00	-0.5
0.5	0.0	0.5	0.00	0.00	0.0
0.5	0.5	1.0	1.00	0.00	0.0
0.5	1.0	1.5	4.00	0.00	0.0
1.0	-1.0	0.0	2.50	2.50	-1.0
1.0	-0.5	0.5	0.50	0.50	-0.5
1.0	0.0	1.0	0.50	0.50	-0.5
1.0	0.5	1.5	2.50	0.50	-0.5
1.0	1.0	2.0	Inf	0.50	-0.5
1.5	-1.0	0.5	2.00	2.00	-1.0
1.5	-0.5	1.0	1.00	1.00	-0.5
1.5	0.0	1.5	2.00	1.00	-0.5
1.5	0.5	2.0	Inf	1.00	-0.5
1.5	1.0	2.5	Inf	1.00	-0.5

Таблица 1

x(0)	u(0)	x(1)	C01	J*01	u*(x(0),0)
0.0	-1.0	-1.0	Inf	Inf	0.0

0.0	-0.5	-0.5	Inf	Inf	0.0
0.0	0.0	0.0	0.75	0.75	0.0
0.0	0.5	0.5	0.50	0.50	0.5
0.0	1.0	1.0	2.75	0.50	0.5
0.5	-1.0	-0.5	Inf	Inf	0.0
0.5	-0.5	0.0	1.25	1.25	-0.5
0.5	0.0	0.5	0.00	0.00	0.0
0.5	0.5	1.0	1.25	0.00	0.0
0.5	1.0	1.5	4.00	0.00	0.0
1.0	-1.0	0.0	2.75	2.75	-1.0
1.0	-0.5	0.5	0.50	0.50	-0.5
1.0	0.0	1.0	0.75	0.50	-0.5
1.0	0.5	1.5	2.50	0.50	-0.5
1.0	1.0	2.0	Inf	0.50	-0.5
1.5	-1.0	0.5	2.00	2.00	-1.0
1.5	-0.5	1.0	1.25	1.25	-0.5
1.5	0.0	1.5	2.00	1.25	-0.5
1.5	0.5	2.0	Inf	1.25	-0.5
1.5	1.0	2.5	Inf	1.25	-0.5

Шаг 0: x = 0.00, u = 0.50, x_next = 0.50
Шаг 1: x = 0.50, u = 0.00, x_next = 0.50

Контрольные вопросы

1. Что такое принцип оптимальности Беллмана?

Уравнение Беллмана представляет собой дифференциальное уравнение в частных производных с начальными условиями, заданными для последнего момента времени (то есть справа), для функции Беллмана, которая выражает минимальное значение критерия оптимизации, которое может быть достигнуто, при условии эволюции системы из текущего состояния в некоторое конечное.

Понятие уравнения Беллмана и функции Беллмана обычно применяется для непрерывных систем. Для дискретных систем аналогом выступает рекуррентное соотношение Беллмана. Принцип оптимальности позволяет в этом случае оптимальное планирование от конца к началу.

В контексте решения задачи оптимального управления можно выделить два подхода: численный и аналитический. Численный подход основан на использовании вычислительных процедур динамического программирования, в то время как аналитический подход связан с решением уравнения Беллмана. То есть, нелинейного уравнения в частных производных, которое имеет аналитическое решение лишь в простейших случаях.

Принцип оптимальности, подходящий как для непрерывных, так и дискретных систем является основополагающим в теории управления. Две формулировки:

Если управление оптимально, то, каковы бы ни были первоначальное состояние системы и управление системой в начальный момент времени, последующее управление оптимально относительно состояния, которое система примет в результате начального управления.

Указанное свойство можно сравнить с соответствующим свойством марковского процесса:

Оптимальное управление в любой момент времени не зависит от предыстории системы и определяется только состоянием системы в этот момент и целью управления.

Как следствие этого, оптимальное управление зависит только от текущего состояния системы. Последствия неоптимального управления в прошлом не могут быть исправлены в будущем.

Согласно принципу оптимальности, оптимальная стратегия гарантирует, что после первого решения последующие решения будут оптимальными относительно нового состояния, полученного в результате

первоначального решения, независимо от начального состояния и начального решения

2. Что такое динамическое программирование?

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Метод динамического программирования сверху — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. Динамическое программирование снизу включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

3. Как вычислять критерий качества, когда значения управлений не попадают в квантованную сетку?

Если значение управления не попадает в квантованную сетку, применяют интерполяцию. В данном случае используются близлежащие узлы сетки для аппроксимации значений функционала качества. Например, линейная интерполяция может быть использована для определения приблизительного значения качества на основе известных точек сетки.

4. В каком порядке происходит вычисление в алгоритме

динамического программирования?

Поиск решения задачи с помощью динамического программирования состоит из трех шагов: (i) определение класса подзадач, (ii) указание рекуррентного соотношения, основанного на решении каждой подзадачи с помощью более простых подзадач, и (iii) указание алгоритма вычисления рекуррентного соотношения .

Метод динамического программирования сверху — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем. Динамическое программирование снизу включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

5. Что делать с состоянием и управлением, которые формируют траектории, выходящие за рамки ограничений рассматриваемой задачи?

Такие состояния и управления не учитываются в вычислениях. В алгоритме динамического программирования проверяется, находится ли следующее состояние в пределах допустимого диапазона. Если оно выходит за рамки ограничений, данное управление исключается из рассмотрения.

6. В чем достоинства и недостатки применения динамического программирования Беллмана?

Достоинства алгоритма Беллмана:

1. Глобальная оптимальность (гарантирует нахождение оптимального решения)
2. Применимость к широкому кругу задач
3. Постепенное улучшение решения — строит решение поэтапно, начиная с начальных условий.
4. Гибкость — можно адаптировать для различных типов задач.

Недостатки алгоритма Беллмана:

5. Экспоненциальная сложность — требует больших вычислительных ресурсов при большом числе состояний.

6. Проблемы с памятью — требует много памяти для хранения промежуточных решений.
7. Неоптимален для непрерывных и стохастических задач — плохо работает с непрерывными состояниями или стохастическими переходами.
8. Зависимость от модели — требует точной модели задачи.
9. Чувствительность к инициализации — может работать неэффективно при неправильной начальной настройке.

Как и любой другой метод динамического программирования сильно зависит от заданных начальных условий и без них не решается. Среда должна быть определена.

7. Какие два механизма решения задачи динамического программирования существуют для непрерывных динамических систем?

Для решения задачи динамического программирования в контексте непрерывных динамических систем существуют два основных механизма:

1. **Метод Беллмана (или принцип оптимальности Беллмана):**
    - Этот метод используется для нахождения оптимального решения задачи динамического программирования. Он основывается на принципе оптимальности, который утверждает, что решение задачи состоит из оптимальных решений подзадач. В непрерывных системах этот метод используется для рекурсивного вычисления функции стоимости (или ценности) в зависимости от состояния системы. Зачастую используется в дискретном виде.
2. **Метод Гамильтона — Якоби — Беллмана (HJB):**
    - Это обобщение метода Беллмана для непрерывных систем. Метод HJB используется для решения задачи оптимального управления в контексте динамических систем с непрерывными состояниями и управлениями. Он выводит дифференциальное уравнение, которое описывает оптимальную траекторию для системы, и находит оптимальную политику управления.

8. В чем отличие решения задач с фиксированным и нефиксированным терминальным состоянием при использовании динамического программирования Беллмана?

В задачах с фиксированным терминальным состоянием конечная точка задана, и оптимальная траектория должна достигать именно её. Это влияет на расчёт конечного функционала стоимости.

В задачах с нефиксированным терминальным состоянием конечная точка не задана, и оптимальная траектория может завершиться в любой допустимой точке. В этом случае обычно добавляется штраф за отклонение от предпочтительного состояния, чтобы направить оптимизацию.