Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Курсовая работа

«»

Выполнила:
студентка 314 группы
Спиркина Д. О.

Научный руководитель:
Фурсов А. С.

Москва
2025

Содержание

1. Постановка задачи

Рассмотрим переключаемую систему

$$\dot{x} = A_\sigma x + b_\sigma x,$$

где:

- $x(t) \in R^n$ – вектор состояния системы
- $\sigma(t) : [0, +\infty) \to \{1, \ldots, m\}$ – кусочно-постоянная функция переключения
- $A_i \in R^{n \times n}$, $b_i \in R^n$ – матрицы и векторы коэффициентов для i-го режима

В определенный момент времени происходит переключение правой части и один из режимов нашей системы становится активным.

Для обеспечения устойчивости каждого режима используем линейный регулятор

$$u = -kx$$

Подставляя уравнение регулятора, получаем замкнутую систему вида

$$\dot{x} = (A_\sigma - b_\sigma k)x$$

Поставим вопрос об устойчивости данной системы.

Попробуем ограничить время между соседними переключениями. Если переключаться не силишком быстро, то система сохранит свою усточивость. Введем время задержди (τ) - время между соседними переключениями у устойчивой системы.

Попробуем найти такое k, что все режимы и система будут устойчивы при заданном τ. Среди таких k оценим τ. Для этого воспользуемся двумя методами.

Методы оценки времени dwell-time

1. **Метод функций Ляпунова**

   - Проверим, что каждый режим управляемый
   - Проверим, что $H_i = A_i - b_i k$ устойчива для всех $i = \overline{1, m}$
   - Решим уравнение Ляпунова:

   $$H_i^\top Q_i + Q_i H_i = -I$$

   - Введем параметры:

   $$M_i = \lambda_{\max}(Q_i), \quad m_i = \lambda_{\min}(Q_i), \quad c_i = \sqrt{\frac{M_i}{m_i}}, \quad \nu_i = \frac{1}{2M_i}$$

- Определим:
$$\rho = \max_i c_i, \quad \theta = \min_i \nu_i$$
- Критерий устойчивости:
$$\tau > \frac{2}{\theta} \ln(\rho)$$
- Оптимальная оценка: минимальное значение $\frac{2}{\theta} \ln(\rho)$ среди всех допустимых k

2. **Метод диагонализации**

- Проверим, что каждый режим управляемый
- Пусть для k^j матрицы $A_i - b_i k^j$ устойчивы $\forall i = \overline{1, m}$
- Проверим, что все собтвенные значения $\lambda_1^i, ..., \lambda_n^i$ вещественные и различные
- Пусть $\gamma = \max_i |\lambda_1^i, ..., \lambda_n^i|$
- Диагонализируем матрицы: $T_i^{-1}(A_i - b_i k^j)T_i = \Lambda_i$
- Вычислим: $\mu_i = \|T_i\| \cdot \|T_i^{-1}\|$, где $\|T_i\| = \max_k \sum_l |t_{kl}|$
- Определим $\mu = \max_i \mu_i$
- Оценка:
$$\tau = \frac{2}{\gamma} \ln(\mu)$$

Подбор регуляторов для каждого режима

Для каждого режима $i = 1, \ldots, m$ подбирается свой регулятор k_i, обеспечивающий желаемый спектр матрицы $H_i = A_i - b_i k_i$. Затем для каждого k_i оценивается время τ с помощью двух методов.

В результате численного моделирования получены графики зависимости спектра от нормы k и график зависимости спктора от времмени τ для двух методов.

Входные данные:

```
Введите время задержки: 1
Введите число, которое начинает последовательность для спектра: -0.25
Введите шаг: -0.001
Введите конец последовательности: -0.3
Введите матрицу A_1: [1 3; 4 2]
Введите матрицу b_1: [2; 4]
```

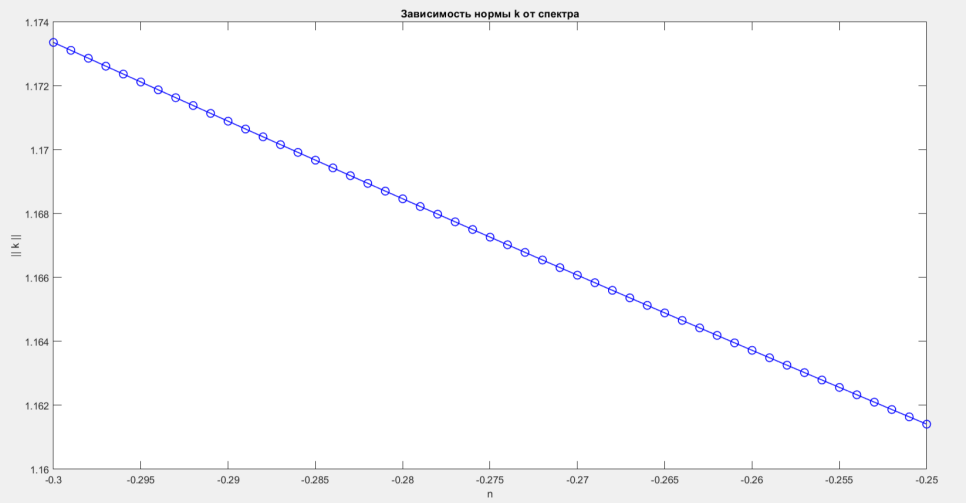

Рис. 1: Зависимость нормы регулятора $\|k\|$ от спектра режима $\sigma(A - bk) = [n, n - 1]$. По оси абсцисс отложены значения n, по оси ординат - соответствующая норма вектора коэффициентов регулятора. График демонстрирует, как увеличение нормы регулятора приводит к смещению спектра в левую полуплоскость.

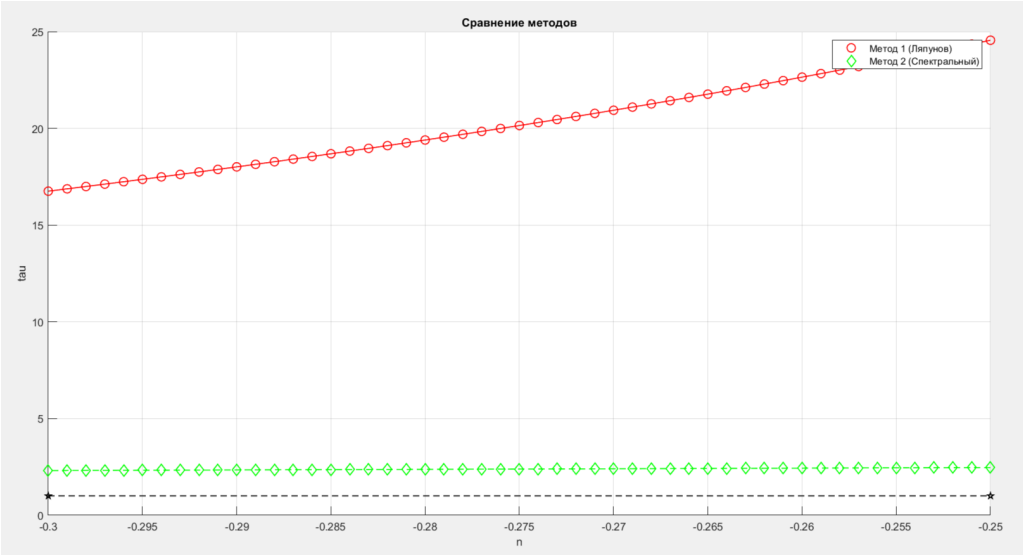

Рис. 2: Сравнение методов оценки времени dwell-time: метод функций Ляпунова и спектральный метод. По оси ординат отложено время τ, по оси абсцисс - параметры спектра режима $\sigma(A - bk) = [n, n - 1]$

2. Код

```
function w3()
    % Ввод параметров
    m = input('Введите количество режимов: ');
    tau = input('Введите время задержки: ');
    n = input('Введите размерность k: ');

    % Инициализация
    R = eye(n);
    A_all = cell(1, m);
    b_all = cell(1, m);
    k_1 = [];
    k_2 = [];
    theta_1 = Inf;
    theta_2 = Inf;
    tol = 1e-6;

    % Ввод матриц A и b для каждого режима
    for i = 1:m
        A_all{i} = input(['Введите матрицу A_', num2str(i)]);
        b_all{i} = input(['Введите матрицу b_', num2str(i)]);
        C = ctrb(A_all{i}, b_all{i});

        % Проверка ранга матрицы управляемости
        rank_C = rank(C);

        % Если ранг равен числу состояний, система управляема
        if rank_C == size(A_all{i}, 1)
            disp('Система управляема');
        else
            error('Система не управляема');
        end
    end

    start = -500;
    finish = 500;
    step = 0.1;

    [grids{1:n}] = ndgrid(start:step:finish);
    k_combinations = reshape(cat(n+1, grids{:}), [], n);

    % считаем методом Ляпунова
    for j = 1:size(k_combinations, 1)
        k = k_combinations(j, :)';
        is_valid = true;
```

```
rho = 0;
theta = Inf;

% Проверка для каждого режима
for i = 1:m
    H = A_all{i} - b_all{i}*k';

    % Проверка устойчивости H
    if any(real(eig(H)) >= 0)
        is_valid = false;
        break;
    end

    % Решение уравнения Ляпунова
    try
        Q = lyap(H', R);
    catch
        is_valid = false;
        break;
    end

    % Проверка положительной определенности Q
    if any(eig(Q) <= 0)
        is_valid = false;
        break;
    end

    % Вычисление параметров
    lambda = eig(Q);
    M = max(lambda);
    m_val = min(lambda);
    rho = max(rho, sqrt(M/m_val));
    theta = min(theta, 1/(2*M));
end

if is_valid
    if tau > (2*log(rho)/theta)
        disp(['Подходящее k: ', mat2str(k')]);
        disp(['rho = ', num2str(rho), ', theta = ', num2str(theta)]);

        if (theta_1 == 0) || (2*log(rho)/theta) < theta_1
            theta_1 = 2*log(rho)/theta;
            k_1 = k;
        end
    end
end
```

```
end

if ~isempty(k_1)
    disp(['Оптимальное k_1: ', mat2str(k_1')]);
    disp(['Минимальное 2*log(rho)/theta: ', num2str(theta_1)]);
else
    disp('Подходящие k не найдены');
end

% 2 метод
for j_1 = 1:size(k_combinations, 1)
    k = k_combinations(j_1, :)';
    k = [1; 1];
    gamma = -Inf;
    mu = -Inf;
    is_valid = true;
    for i_1 = 1:m

        H = A_all{i_1} - b_all{i_1}*k';

        % Проверка на действительность всех собственных значений
        if ~all(abs(imag(eig(H))) < tol)
            is_valid = false;
            break;
        end
        % проверка, что все собственные значения различимы
        if length(unique(round(real(eig(H)), 10))) < length(eig(H))
            is_valid = false;
            break;
        end

         % Проверка устойчивости H
        if any(real(eig(H)) >= 0)
            is_valid = false;
            break;
        end

        if gamma < max(eig(H))
            gamma = max(eig(H));
        end

        [T, D] = eig(H);
        row_sum_T = sum(T, 2);
        row_sum_inv_T = sum(abs(pinv(T)), 2);
        mu_i = max(row_sum_inv_T) * max(row_sum_T);
```

```
            if mu_i > mu
                mu = mu_i;
            end
        end
        gamma = abs(gamma);

        if is_valid
            if (theta_2 == 0) || ((2*log(mu)/gamma) < theta_2)
                theta_2 = 2*log(mu)/gamma;
                k_2 = k;
            end
        end
    end
    if ~isempty(k_2)
        disp(['Оптимальное k_2: ', mat2str(k_2')]);
        disp(['Минимальное 2*log(gamma)/mu: ', num2str(theta_2)]);
    else
        disp('Подходящие k не найдены');
    end
end

%Введите матрицу A_1 [-2 1; 0 -3]
%Введите матрицу b_1 [0; 1]
%Введите матрицу A_2 [-1 1; 0 -2]
%Введите матрицу b_2 [0; 1]
%Введите матрицу A_3 [-3 0; 1 -1]
%Введите матрицу b_3 [1; 0]
%R = eye(2);
```

Второй код

```
function prak()
    % Ввод параметров
    m = 1;
    n = 2;
    tau = input('Введите время задержки: ');
    start = input('Введите число, которое начинает последовательность для спектра:
    step = input('Введите шаг: ');
    finish = input('Введите конец последовательности: ');

    % Инициализация
    R = eye(n);
    A_all = cell(1, m);
    b_all = cell(1, m);
    tol = 1e-6;
```

```
% Ввод матриц A и b для каждого режима
for i = 1:m
    A_all{i} = input(['Введите матрицу A_', num2str(i), ': ']);
    b_all{i} = input(['Введите матрицу b_', num2str(i), ': ']);
    C = ctrb(A_all{i}, b_all{i});

    % Проверка ранга матрицы управляемости
    rank_C = rank(C);

    % Если ранг равен числу состояний, система управляема
    if rank_C == size(A_all{i}, 1)
        disp('Система управляема');
    else
        error('Система не управляема');
    end
end

if (start < finish) || (start >= 0) || (finish >= 0) || (step >= 0)
    error('Проверьте введённые значения start и finish и step');
end

row1 = start:step:finish;
row2 = (start-1):step:(finish-1);
M = [row1; row2];
x_values = row1;
y_values = zeros(size(x_values));
ans_values_1 = zeros(size(x_values)); % Для первого метода
ans_values_2 = zeros(size(x_values)); % Для второго метода

figure(1); % График нормы k
grid on;

figure(2); % График ans_1 и ans_2
hold on;
grid on;

for i = 1:size(M, 2)
    l = M(:, i);
    k = place(A_all{1}, b_all{1}, l);
    y = norm(k, 'fro');
    x = row1(i);
    y_values(i) = y;

    % График нормы k
    figure(1);
    plot(x, y, 'bo', 'MarkerSize', 8, 'LineWidth', 1);
```

```
hold on;

% Метод 1: Ляпунов
H = A_all{1} - b_all{1}*k;

% Проверка устойчивости H
if any(real(eig(H)) > -tol)
    error('Матрица H неустойчива');
end

try
    Q = lyap(H', R);
catch
    error('Ошибка при решении уравнения Ляпунова');
end

if any(eig(Q) <= 0)
    error('Матрица Q не положительно определена');
end

lambda = eig(Q);
M_val = max(lambda);
m_val = min(lambda);
rho = sqrt(M_val/m_val);
theta = 1/(2*M_val);
ans_values_1(i) = 2*log(rho)/theta;

% Метод 2: Спектральный
if ~all(abs(imag(eig(H))) < tol)
    error('Комплекстные собственные значения');
end
if length(unique(round(real(eig(H)), 10))) < length(eig(H))
    error('Неразличные собственные значения');
end
if any(real(eig(H)) > -tol)
    error('Матрица H неустойчива');
end

gamma = max(abs(eig(H)));
[T, D] = eig(H);
row_sum_T = sum(abs(T), 2);
row_sum_inv_T = sum(abs(inv(T)), 2);
mu = max(row_sum_inv_T) * max(row_sum_T);
ans_values_2(i) = 2*log(mu)/gamma;

% График ans_1 и ans_2
```

```
        figure(2);
        plot(x, ans_values_1(i), 'ro', 'MarkerSize', 8, 'LineWidth', 1);
        plot(x, ans_values_2(i), 'gd', 'MarkerSize', 8, 'LineWidth', 1);
    end

    % Оформление графиков
    figure(1);
    plot(x_values, y_values, 'b-', 'LineWidth', 1);
    xlabel('n');
    ylabel('|| k ||');
    title('Зависимость нормы k от спектра');
    hold off;

    figure(2);
    % Линии и точки для ans_1 (красные)
    plot(x_values, ans_values_1, 'r-', 'LineWidth', 1);
    plot(x_values, ans_values_1, 'ro', 'MarkerSize', 8, 'LineWidth', 1);
    % Линии и точки для ans_2 (зелёные)
    plot(x_values, ans_values_2, 'g--', 'LineWidth', 1);
    plot(x_values, ans_values_2, 'gd', 'MarkerSize', 8, 'LineWidth', 1);
    % tau (черный)
    x_limits = xlim; % Получаем текущие границы x
    plot([x_limits(1), x_limits(2)], [tau tau], '--p', 'LineWidth', 1, 'Color', 'k'

    xlabel('n');
    ylabel('tau');
    title('Сравнение методов');
    legend('Метод 1 (Ляпунов)', 'Метод 2 (Спектральный)');
    hold off;
end

% prak
% Введите время задержки: 0.5
% Введите число, которое начинает последовательность для спектра: -0.01
% Введите шаг: -0.01
% Введите конец последовательности: -1
% Введите матрицу A_1: [1 3; 1 2]
% Введите матрицу b_1: [2; 4]
```