### Код моделирования системы (без регуляторов)

|  |
| --- |
| function system  % Время моделирования  t\_total = 30000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага.  % Количество элементов в векторе t. Поскольку вектор t содержит все временные точки симуляции, его длина соответствует количеству временных шагов 𝑁    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  x(:,1) = [20; 20; 0]; % Начальные температуры и теплота  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 10; % Постоянная времени системы  k = 592; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];    % Управляющее воздействие (постоянное для примера)  u = 100 \* ones(1, N); % Мощность нагревателя  % Функция ones в MATLAB создаёт матрицу или вектор, заполненный единицами.  % В данном случае ones(1, N) создаёт вектор длины 𝑁 (то есть количество шагов симуляции), заполненный единицами.    % Дискретное моделирование методом Эйлера  for i = 1:N-1  x(:,i+1) = (A \* x(:,i) + B \* u(i)) \* h + x(:,i);  end  % Построение графиков  figure;  subplot(1,1,1);  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры T\_{р.к.}(t) (X1)');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;    % subplot(3,1,2);  % plot(t, x(2,:), 'b', 'LineWidth', 2);  % title('Температура рубашки T\_p(t) (X2)');  % xlabel('Время (с)');  % ylabel('T\_p (°C)');  % grid on;    % subplot(3,1,3);  % plot(t, x(3,:), 'g', 'LineWidth', 2);  % title('Количество теплоты Q\_H(t) (X3)');  % xlabel('Время (с)');  % ylabel('Q\_H (Дж)');  % grid on;  end |

### Код моделирования системы с ПИ регулятором (без оптимизации). Коэффициенты вручную

|  |
| --- |
| function system\_with\_PI  % Время моделирования  t\_total = 30000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  x(:,1) = [0; 0; 0]; % Начальные температуры и теплота  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 10; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];    B = [0; 0; k/T];    % Настройка ПИ-регулятора (с минимальным перерегулированием ~4%)  % Используем метод CHR для настройки  Kp = 0.0008; % Пропорциональный коэффициент  Ti = 6; % Интегральная постоянная времени  % Управляющее воздействие (инициализация)  u = zeros(1, N); % Вектор управления  error\_integral = 0; % Интеграл ошибки  setpoint = 80; % Желаемая температура уставки (в °C)  % Основной цикл моделирования методом Эйлера  for i = 1:N-1  % Вычисляем ошибку (разница между заданной и текущей температурой)  error = setpoint - x(1,i);    % Интегрируем ошибку  error\_integral = error\_integral + error \* h;    % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral);    % Ограничение мощности нагревателя (например, мощность не может быть отрицательной)  u(i) = max(0, u(i));    % Дискретное моделирование  x(:,i+1) = (A \* x(:,i) + B \* u(i)) \* h + x(:,i);  end  % Построение графиков  %figure;  subplot(1,1,1);  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры T\_{р.к.}(t) (X1)');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

Kp = 0.0008; % Пропорциональный коэффициент

Ti = 6; % Интегральная постоянная времени

### Код моделирования с подбором коэффициентов через цикл for.

|  |
| --- |
| function optimize\_PI\_regulator  % Время моделирования  t\_total = 30000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага    % Диапазон значений для Kp и Ti  Kp\_range = 0.0001:0.0001:0.01; % Диапазон для Kp  Ti\_range = 1:0.05:10; % Диапазон для Ti  % Переменные для хранения оптимальных параметров и минимальной стоимости  optimal\_params = [0, 0];  min\_cost = inf; % Изначально ставим стоимость очень большой    % Циклы для перебора всех комбинаций Kp и Ti  for Kp = Kp\_range  for Ti = Ti\_range  % Параметры для текущей симуляции  params = [Kp, Ti];    % Вычисляем стоимость для текущих параметров  cost = simulate\_system(params, t, N, h, t\_total);    % Сравниваем с минимальной стоимостью  if cost < min\_cost  min\_cost = cost;  optimal\_params = params;  end  end  end  % Вывод результатов  fprintf('Оптимальные параметры ПИ-регулятора:\n');  fprintf('Kp = %.4f\n', optimal\_params(1));  fprintf('Ti = %.4f\n', optimal\_params(2));    % Симуляция системы с оптимальными параметрами  simulate\_system(optimal\_params, t, N, h, t\_total, true);  end  function cost = simulate\_system(params, t, N, h, t\_total, plot\_results)  % Функция для симуляции системы и оценки качества работы ПИ-регулятора  if nargin < 6  plot\_results = false; % По умолчанию графики не строим  end    % Параметры ПИ-регулятора  Kp = params(1); % Пропорциональный коэффициент  Ti = params(2); % Интегральная постоянная времени    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  x(:,1) = [0; 0; 0]; % Начальные температуры и теплота  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 10; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];    B = [0; 0; k/T];    % Управляющее воздействие (инициализация)  u = zeros(1, N); % Вектор управления  error\_integral = 0; % Интеграл ошибки  setpoint = 80; % Желаемая температура уставки (в °C)  max\_overshoot = 0; % Для отслеживания максимального перерегулирования  % Основной цикл моделирования методом Эйлера  for i = 1:N-1  % Вычисляем ошибку (разница между заданной и текущей температурой)  error = setpoint - x(1,i);    % Интегрируем ошибку  error\_integral = error\_integral + error \* h;    % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral);    % Ограничение мощности нагревателя (например, мощность не может быть отрицательной)  u(i) = max(0, u(i));    % Дискретное моделирование  x(:,i+1) = (A \* x(:,i) + B \* u(i)) \* h + x(:,i);    % Оценка максимального перерегулирование  if x(1,i+1) > setpoint  overshoot = (x(1,i+1) - setpoint) / setpoint \* 100;  max\_overshoot = max(max\_overshoot, overshoot);  end  end  % Оценка времени переходного процесса (время, за которое система стабилизируется в пределах 2%)  stable\_index = find(abs(x(1,:) - setpoint) <= 0.02\*setpoint, 1);  if isempty(stable\_index)  stable\_time = t\_total; % Если система не стабилизируется, задаем максимальное время  else  stable\_time = t(stable\_index);  end    % Ограничение на перерегулирование  if max\_overshoot > 4  cost = 1e6; % Если перерегулирование больше 4%, задаем высокую стоимость  else  cost = stable\_time; % Иначе стоимость - это время переходного процесса  end    % Построение графиков, если это финальная симуляция  if plot\_results  plot(t, x(1,:), 'blue', 'LineWidth', 2);  title('Температура рабочей камеры T\_{р.к.}(t) (X1)');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end  end |

В данном коде реализован подбор параметров, который осуществляется через цикл for, выполняющий полный перебор значений параметров 𝐾𝑝 и 𝑇𝑖.

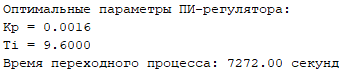
Kp\_range = 0.0001:0.0001:0.01; % Диапазон для Kp

От 0.0001 до 0.01 с шагом 0.0001

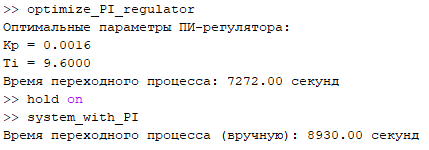
Ti\_range = 1:0.05:10; % Диапазон для Ti

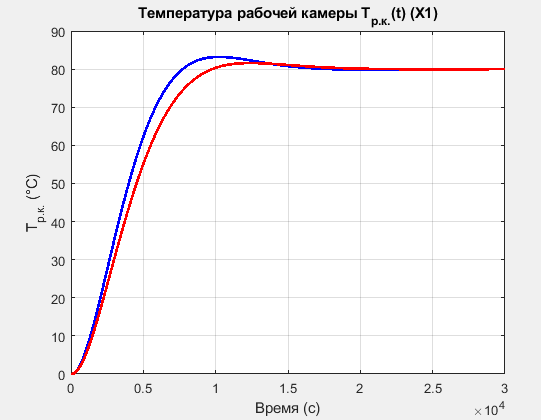
От 1 до 10 с шагом 0.05

Найденный параметры автоматическим способом:



Сравнение параметров, подобранных автоматическим способом и вручную:

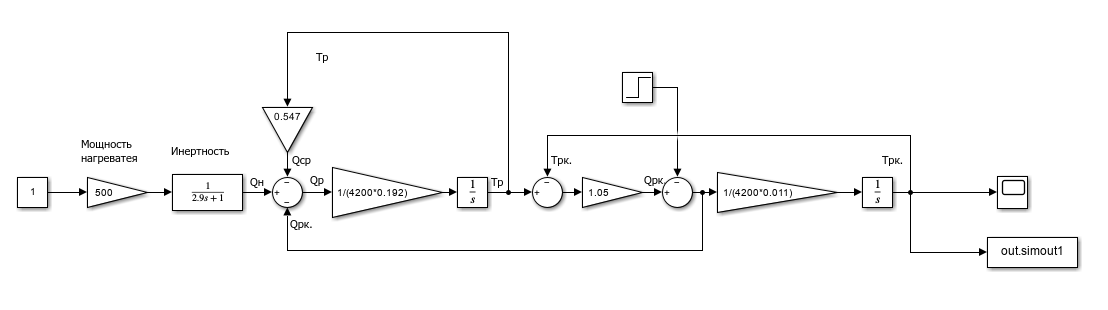




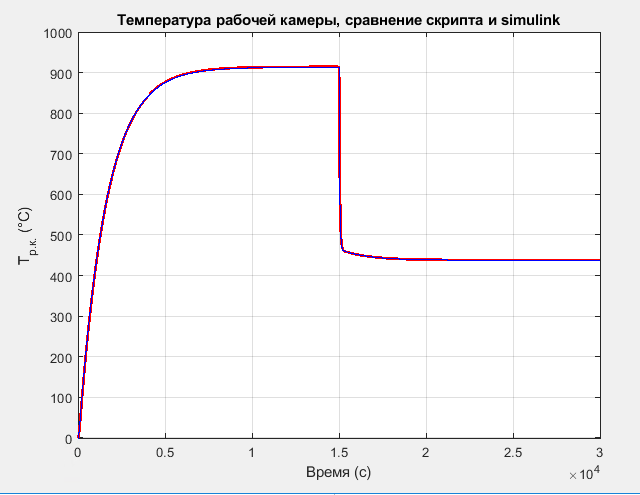
### Система с возмущением без регулятора

|  |
| --- |
| function sravni % Время моделирования  t\_total = 30000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];    F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Сброс начальных условий  x(:,1) = [0; 0; 0];  u = 500\*ones(1, N);    % Возмущение: изменение температуры окружающей среды  f = zeros(1, N);  f(15000:end) = 500;    for i = 1:N-1  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  end    % Построение графика с коэффициентами  %figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры с фиксированными параметрами ПИ-регулятора');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

Проверим с системой в Simulink:







Графики совпадают, значит систему составили правильно

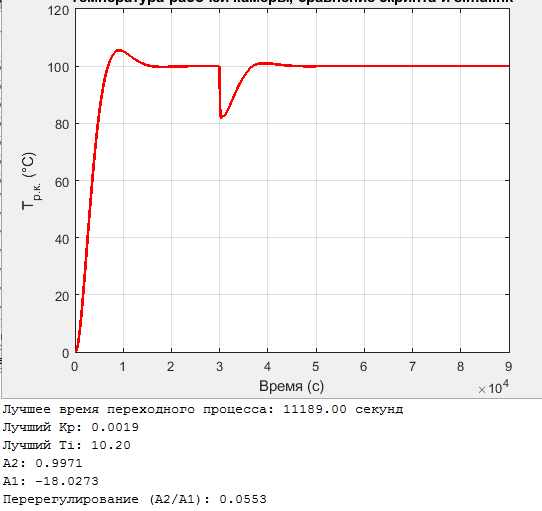
### Код моделирования системы с возмущением, вручную и подсчётом перерегулирования и времени переходного процесса

|  |
| --- |
| function PI\_regulator\_disturbance % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  % Параметры ПИ-регулятора  Kp = 0.0015; % Пропорциональный коэффициент  Ti = 11.70; % Интегральная постоянная времени  setpoint = 0; % Уставка — желаемая температура  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];    F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Сброс начальных условий  x(:,1) = [0; 0; 0];  error\_integral = 0; % Интеграл ошибки  u = 500\*ones(1, N);  % Возмущение: изменение температуры окружающей среды  f = zeros(1, N);  start\_disturbance = 30000;  f(start\_disturbance:end) = 1;  % Ограничение на мощность нагревателя  u\_max = 500; % Максимальная мощность нагревателя  %параметры для подсчёта времени переходного процесса  transition\_time = NaN; % Время переходного процесса  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 5000; % Количество шагов для установившегося состояния  steady\_steps\_count = 0; % Счётчик шагов в установившемся состоянии  for i = 1:N-1  % Вычисляем ошибку (разница между уставкой и текущей температурой)  error = setpoint - x(1,i);    % Интегрируем ошибку  error\_integral = error\_integral + error \* h;    % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка на установившееся состояние после возмущения  if i >= start\_disturbance && isnan(transition\_time)  deviation = abs(x(1,i) - setpoint);  if setpoint == 0  % Если уставка 0, используем фиксированный критерий для отклонения  if deviation < steady\_state\_criteria  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  else  % Если уставка не 0, используем относительное отклонение  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  end  if steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i) - t(start\_disturbance); % Время переходного процесса  end  end  end  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  % Расчет перерегулирования (A2/A1)  % Найдем минимальное значение температуры до возмущения (Tmin до возмущения)  Tmin = min(x(1, start\_disturbance:end));  % Найдем максимальное значение температуры после возмущения (Tmax после возмущения)  Tmax = max(x(1, start\_disturbance:end));  % Расчет перерегулирования  A2 = Tmax - setpoint;  A1 = Tmin - setpoint;  overshoot\_ratio = abs(A2/A1);  % Вывод перерегулирования  disp(['A1: ', num2str(A1)]);  disp(['A2: ', num2str(A2)]);  disp(['Перерегулирование (A2/A1): ', num2str(abs(overshoot\_ratio))]);  % Построение графика с коэффициентами  figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

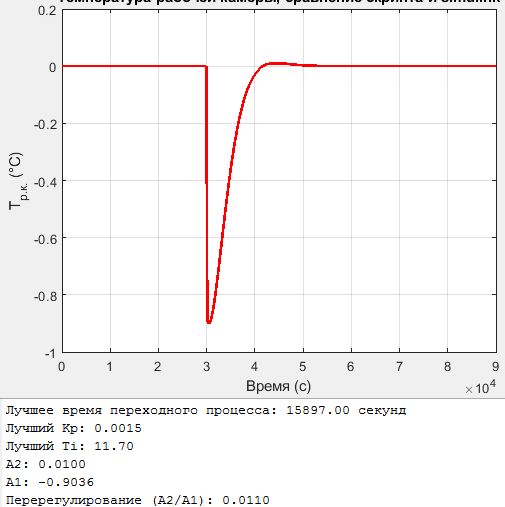
### Код моделирования системы с возмущением, подсчётом времени переходного процесса и автоматическим перебором коэффициентов

|  |
| --- |
| function sravni()  % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия  setpoint = 100; % Уставка — желаемая температура  % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];  F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Ограничение на мощность нагревателя  u\_max = 500; % Максимальная мощность нагревателя  %параметры для подсчёта времени переходного процесса  transition\_time = NaN; % Время переходного процесса  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 5000; % Количество шагов для установившегося состояния  steady\_steps\_count = 0; % Счётчик шагов в установившемся состоянии  % Параметры поиска лучших коэффициентов  best\_time = Inf; % Лучшее время переходного процесса  best\_Kp = 0;  best\_Ti = 0;  best\_overshoot = Inf; % Лучшее перерегулирование  best\_tmax = Inf;  best\_tmin = Inf;  % Параметры для поиска  Kp\_values = 0.0005:0.0001:0.002; % Диапазон значений Kp  Ti\_values = 1:0.1:12; % Диапазон значений Ti  % Начало цикла перебора коэффициентов  for Kp = Kp\_values  for Ti = Ti\_values  % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  error\_integral = 0; % Интеграл ошибки  u = 500\*ones(1, N); % Управляющее воздействие  f = zeros(1, N); % Возмущение  start\_disturbance = 30000;  f(start\_disturbance:end) = 20;  transition\_time = NaN; % Время переходного процесса  steady\_steps\_count = 0; % Счётчик шагов в установившемся состоянии  % Моделирование системы  for i = 1:N-1  % Вычисляем ошибку (разница между уставкой и текущей температурой)  error = setpoint - x(1,i);  % Интегрируем ошибку  error\_integral = error\_integral + error \* h;  % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка на установившееся состояние после возмущения  if i >= start\_disturbance && isnan(transition\_time)  deviation = abs(x(1,i) - setpoint);  if setpoint == 0  % Если уставка 0, используем фиксированный критерий для отклонения  if deviation < steady\_state\_criteria  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  else  % Если уставка не 0, используем относительное отклонение  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  end  if steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i) - t(start\_disturbance); % Время переходного процесса  end  end  end  % Расчет перерегулирования (A2/A1)  Tmin = min(x(1, start\_disturbance:end));  Tmax = max(x(1, start\_disturbance:end));  A2 = Tmax - setpoint;  A1 = Tmin - setpoint;  overshoot\_ratio = abs(A2/A1);    % Проверка на перерегулирование (не более 4%) и обновление лучших параметров  if transition\_time < best\_time && ~isnan(transition\_time) && overshoot\_ratio <= 1.04  best\_time = transition\_time;  best\_Kp = Kp;  best\_Ti = Ti;  best\_tmax = A2;  best\_tmin = A1;  best\_overshoot = overshoot\_ratio;  end  end  end  if overshoot\_ratio >= 1.04  fprintf('Система не сбалилизировалась');  else  % Вывод наилучших коэффициентов, времени переходного процесса и перерегулирования  fprintf('Лучшее время переходного процесса: %.2f секунд\n', best\_time);  fprintf('Лучший Kp: %.4f\n', best\_Kp);  fprintf('Лучший Ti: %.2f\n', best\_Ti);  fprintf('A2: %.4f\n', best\_tmax);  fprintf('A1: %.4f\n', best\_tmin);  fprintf('Перерегулирование (A2/A1): %.4f\n', abs(best\_overshoot));  end  % Повторное моделирование с лучшими значениями Kp и Ti  x = zeros(3, N); % Массив для хранения результатов  error\_integral = 0; % Интеграл ошибки  u = 500\*ones(1, N); % Управляющее воздействие  f = zeros(1, N); % Возмущение  start\_disturbance = 30000;  f(start\_disturbance:end) = 20;  % Моделирование системы с лучшими Kp и Ti  for i = 1:N-1  % Вычисляем ошибку (разница между уставкой и текущей температурой)  error = setpoint - x(1,i);  % Интегрируем ошибку  error\_integral = error\_integral + error \* h;  % ПИ-регулятор  u(i) = best\_Kp \* (error + (1/best\_Ti) \* error\_integral);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  end  % Построение графика с лучшими коэффициентами  figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры, cравнение скрипта и simulink');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

При уставке 100, возмущении 20:



При уставке 0, возмущении 1:



### Каскадный ПИ-ПИ регулятор

Каскадный ПИ-ПИ регулятор состоит из двух уровней регуляторов, где первый регулятор (внутренний контур) управляет промежуточной переменной (например, температурой рубашки), а второй (внешний контур) управляет основной переменной системы (например, температурой рабочей камеры). В каскадной системе внешний регулятор рассчитывает уставку для внутреннего регулятора, а внутренний регулятор непосредственно управляет системой.

Для этой системы я добавлю каскадную структуру регуляторов:

* Внешний контур будет регулировать температуру рабочей камеры (переменная ) с помощью ПИ-регулятора.
* Внутренний контур будет регулировать температуру рубашки (переменная ) с помощью ПИ-регулятора, который будет управлять подачей тепла.

**Общая схема каскадного ПИ-ПИ-регулятора:**

1. **Внешний контур (регулятор температуры рабочей камеры)**:

* Заданная температура рабочей камеры ​ — это уставка для внешнего контура.
* Внешний ПИ-регулятор рассчитывает уставку для температуры рубашки

1. **Внутренний контур (регулятор температуры рубашки)**:

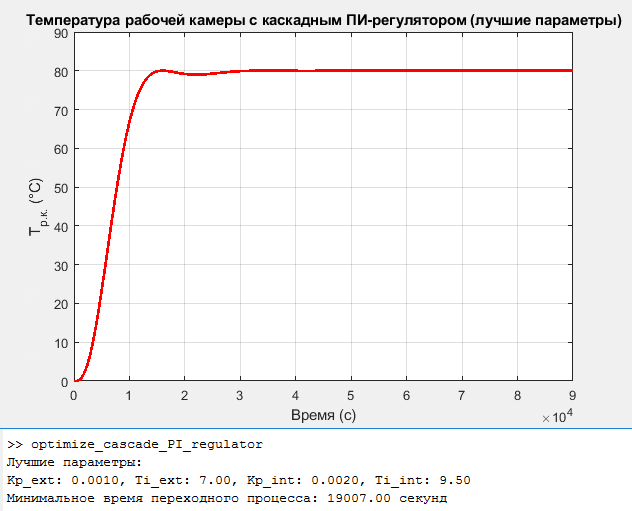
* Уставка температуры рубашки поступает от внешнего регулятора.
* Внутренний ПИ-регулятор управляет подачей тепла на рубашку, чтобы достичь заданной температуры рубашки.

### Каскадный ПИ-ПИ регулятор вручную, по уставке

|  |
| --- |
| function PI\_PI\_disturbance  % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага    % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  % Параметры внешнего ПИ-регулятора (камеры)  Kp\_outer = 0.0011; % Пропорциональный коэффициент  Ti\_outer = 6.98; % Интегральная постоянная времени  setpoint = 0; % Уставка — желаемая температура камеры  % Параметры внутреннего ПИ-регулятора (мощности нагревателя)  Kp\_inner = 0.1; % Пропорциональный коэффициент внутреннего ПИ-регулятора  Ti\_inner = 5.0; % Интегральная постоянная времени для внутреннего ПИ-регулятора  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];  F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Сброс начальных условий  x(:,1) = [0; 0; 0];  error\_integral\_outer = 0; % Интеграл ошибки внешнего регулятора  error\_integral\_inner = 0; % Интеграл ошибки внутреннего регулятора  u\_outer = zeros(1, N); % Внешний сигнал управления (целевое значение мощности нагревателя)  u = 500\*ones(1, N); % Фактическая мощность нагревателя    % Возмущение: изменение температуры окружающей среды  f = zeros(1, N);  start\_disturbance = 30000;  f(start\_disturbance:end) = 1;  % Ограничение на мощность нагревателя  u\_max = 500; % Максимальная мощность нагревателя  % Параметры для подсчёта времени переходного процесса  transition\_time = NaN; % Время переходного процесса  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 1000; % Количество шагов для установившегося состояния  steady\_steps\_count = 0; % Счётчик шагов в установившемся состоянии  for i = 1:N-1  % Внешний ПИ-регулятор  error\_outer = setpoint - x(1,i);  error\_integral\_outer = error\_integral\_outer + error\_outer \* h;  u\_outer(i) = Kp\_outer \* (error\_outer + (1/Ti\_outer) \* error\_integral\_outer);    % Внутренний ПИ-регулятор  error\_inner = u\_outer(i) - u(i);  error\_integral\_inner = error\_integral\_inner + error\_inner \* h;  u(i) = Kp\_inner \* (error\_inner + (1/Ti\_inner) \* error\_integral\_inner);    % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка на установившееся состояние после возмущения  if i >= start\_disturbance && isnan(transition\_time)  deviation = abs(x(1,i) - setpoint);  if setpoint == 0  % Если уставка 0, используем фиксированный критерий для отклонения  if deviation < steady\_state\_criteria  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  else  % Если уставка не 0, используем относительное отклонение  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  end  if steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i) - t(start\_disturbance); % Время переходного процесса  end  end  end  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  % Расчет перерегулирования (A2/A1)  Tmin = min(x(1, start\_disturbance:end));  Tmax = max(x(1, start\_disturbance:end));  A2 = Tmax - setpoint;  A1 = Tmin - setpoint;  overshoot\_ratio = abs(A2/A1);  disp(['Перерегулирование (A2/A1): ', num2str(abs(overshoot\_ratio))]);  % Построение графика с коэффициентами  figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

### Каскадный ПИ-ПИ регулятор с автоматическим перебором коэффициентов, по уставке

|  |
| --- |
| function optimize\_cascade\_PI  % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 10; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия  % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];  setpoint = 80; % Уставка — желаемая температура  % Параметры для перебора коэффициентов ПИ-регуляторов  Kp\_ext\_range = 0.001:0.001:0.005; % Диапазон коэффициента Kp для внешнего регулятора  Ti\_ext\_range = 1:0.5:10; % Диапазон Ti для внешнего регулятора  Kp\_int\_range = 0.001:0.001:0.005; % Диапазон Kp для внутреннего регулятора  Ti\_int\_range = 1:0.5:10; % Диапазон Ti для внутреннего регулятора  best\_transition\_time = inf; % Для хранения лучшего времени переходного процесса  best\_params = []; % Для хранения лучших параметров регуляторов  best\_x = []; % Для хранения результатов моделирования с лучшими параметрами  % Перебор всех возможных комбинаций коэффициентов  for Kp\_ext = Kp\_ext\_range  for Ti\_ext = Ti\_ext\_range  for Kp\_int = Kp\_int\_range  for Ti\_int = Ti\_int\_range  % Сброс начальных условий для каждой комбинации  x = zeros(3, N);  x(:,1) = [0; 0; 0]; % Начальные условия  error\_integral\_ext = 0; % Интеграл ошибки для внешнего контура  error\_integral\_int = 0; % Интеграл ошибки для внутреннего контура  u\_ext = zeros(1, N); % Управление внешнего ПИ-регулятора  u\_int = zeros(1, N); % Управление внутреннего ПИ-регулятора  max\_temperature = 0; % Максимальная температура для расчета перерегулирования  transition\_time = NaN; % Время переходного процесса  steady\_steps\_count = 0; % Счётчик шагов для стабилизации  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 5000; % Количество шагов для установившегося состояния  u\_max = 500; % Ограничение на мощность нагревателя  % Основной цикл моделирования методом Эйлера  for i = 1:N-1  error\_ext = setpoint - x(1,i);  error\_integral\_ext = error\_integral\_ext + error\_ext \* h;  u\_ext(i) = Kp\_ext \* (error\_ext + (1/Ti\_ext) \* error\_integral\_ext);  u\_ext(i) = max(0, min(u\_ext(i), u\_max));  error\_int = u\_ext(i) - x(2,i);  error\_integral\_int = error\_integral\_int + error\_int \* h;  u\_int(i) = Kp\_int \* (error\_int + (1/Ti\_int) \* error\_integral\_int);  u\_int(i) = max(0, min(u\_int(i), u\_max));  x(:,i+1) = (A \* x(:,i) + B \* u\_int(i)) \* h + x(:,i);  if x(1,i+1) > max\_temperature  max\_temperature = x(1,i+1);  end  % Проверка на установившееся состояние  deviation = abs(x(1,i) - setpoint);  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  % Определение времени переходного процесса  if isnan(transition\_time) && steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i); % Время переходного процесса  end  end  % Расчёт перерегулирования  overshoot = ((max\_temperature - setpoint) / setpoint) \* 100;  % Проверяем условия: перерегулирование <= 4% и минимальное время переходного процесса  if overshoot <= 4 && ~isnan(transition\_time) && transition\_time < best\_transition\_time  best\_transition\_time = transition\_time;  best\_params = [Kp\_ext, Ti\_ext, Kp\_int, Ti\_int];  best\_x = x; % Сохраняем результаты для лучших параметров  end  end  end  end  end  % Вывод лучших параметров и времени переходного процесса  fprintf('Лучшие параметры:\n');  fprintf('Kp\_ext: %.4f, Ti\_ext: %.2f, Kp\_int: %.4f, Ti\_int: %.2f\n', best\_params(1), best\_params(2), best\_params(3), best\_params(4));  fprintf('Минимальное время переходного процесса: %.2f секунд\n', best\_transition\_time);  % Построение графика для лучшей комбинации  figure;  plot(t, best\_x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры с каскадным ПИ-регулятором (лучшие параметры)');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |



### Каскадный ПИ-ПИ регулятор с возмущением без автоматического подбора

|  |
| --- |
| function PI\_PI\_disturbance()  % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  % Параметры внешнего ПИ-регулятора (температура камеры)  Kp\_outer = 0.0015; % Пропорциональный коэффициент внешнего ПИ-регулятора  Ti\_outer = 11.70; % Интегральная постоянная времени внешнего ПИ-регулятора  setpoint = 0; % Уставка — желаемая температура  % Параметры внутреннего ПИ-регулятора (мощность нагревателя)  Kp\_inner = 0.1; % Пропорциональный коэффициент внутреннего ПИ-регулятора  Ti\_inner = 5.0; % Интегральная постоянная времени внутреннего ПИ-регулятора  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия    % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];  F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Сброс начальных условий  x(:,1) = [0; 0; 0];  error\_integral\_outer = 0; % Интеграл ошибки внешнего регулятора  error\_integral\_inner = 0; % Интеграл ошибки внутреннего регулятора  u\_outer = zeros(1, N); % Внешний сигнал управления (целевое значение мощности нагревателя)  u = 500\*ones(1, N); % Фактическая мощность нагревателя    % Возмущение: изменение температуры окружающей среды  f = zeros(1, N);  start\_disturbance = 30000;  f(start\_disturbance:end) = 1;  % Ограничение на мощность нагревателя  u\_max = 500; % Максимальная мощность нагревателя  % Параметры для подсчёта времени переходного процесса  transition\_time = NaN; % Время переходного процесса  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 5000; % Количество шагов для установившегося состояния  steady\_steps\_count = 0; % Счётчик шагов в установившемся состоянии  % Моделирование системы  for i = 1:N-1  % Внешний ПИ-регулятор (температура камеры)  error\_outer = setpoint - x(1,i);  error\_integral\_outer = error\_integral\_outer + error\_outer \* h;  u\_outer(i) = Kp\_outer \* (error\_outer + (1/Ti\_outer) \* error\_integral\_outer);  % Внутренний ПИ-регулятор (мощность нагревателя)  error\_inner = u\_outer(i) - u(i);  error\_integral\_inner = error\_integral\_inner + error\_inner \* h;  u(i) = Kp\_inner \* (error\_inner + (1/Ti\_inner) \* error\_integral\_inner);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка на установившееся состояние после возмущения  if i >= start\_disturbance && isnan(transition\_time)  deviation = abs(x(1,i) - setpoint);  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  if steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i) - t(start\_disturbance);  end  end  end  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  % Расчет перерегулирования (A2/A1)  Tmin = min(x(1, start\_disturbance:end));  Tmax = max(x(1, start\_disturbance:end));  A2 = Tmax - setpoint;  A1 = Tmin - setpoint;  overshoot\_ratio = abs(A2 / A1);  % Вывод перерегулирования  disp(['A1: ', num2str(A1)]);  disp(['A2: ', num2str(A2)]);  disp(['Перерегулирование (A2/A1): ', num2str(abs(overshoot\_ratio))]);  % Построение графика с коэффициентами  figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры с каскадным ПИ-ПИ регулятором');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

### Каскадный ПИ-ПИ регулятор с возмущением и автоматическим перебором коэффициентов

|  |
| --- |
| function optimize\_PI\_regulator\_coefficients()  % Время моделирования  t\_total = 90000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 2.9; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия  setpoint = 0; % Уставка — желаемая температура  % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];  B = [0; 0; k/T];  F = [-1/(c\*mp\_k); 1/(c\*mp); 0];  % Ограничение на мощность нагревателя  u\_max = 500; % Максимальная мощность нагревателя  % Диапазоны для перебора коэффициентов  Kp\_outer\_values = 0.0001:0.0001:0.002; % Диапазон значений Kp для внешнего ПИ  Ti\_outer\_values = 1:0.5:15; % Диапазон значений Ti для внешнего ПИ  Kp\_inner\_values = 0.01:0.01:0.2; % Диапазон значений Kp для внутреннего ПИ  Ti\_inner\_values = 1:0.5:10; % Диапазон значений Ti для внутреннего ПИ  % Инициализация переменных для хранения лучших результатов  best\_time = Inf;  best\_Kp\_outer = 0;  best\_Ti\_outer = 0;  best\_Kp\_inner = 0;  best\_Ti\_inner = 0;  best\_overshoot = Inf;  % Цикл перебора коэффициентов  for Kp\_outer = Kp\_outer\_values  for Ti\_outer = Ti\_outer\_values  for Kp\_inner = Kp\_inner\_values  for Ti\_inner = Ti\_inner\_values  % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N);  error\_integral\_outer = 0;  error\_integral\_inner = 0;  u\_outer = zeros(1, N);  u = 500 \* ones(1, N);    % Возмущение  f = zeros(1, N);  start\_disturbance = 30000;  f(start\_disturbance:end) = 1;  % Параметры переходного процесса  transition\_time = NaN;  steady\_steps\_count = 0;  steady\_state\_criteria = 0.01; % Критерий ±1% от уставки  steady\_steps\_required = 5000;  % Моделирование системы  for i = 1:N-1  % Внешний ПИ-регулятор  error\_outer = setpoint - x(1,i);  error\_integral\_outer = error\_integral\_outer + error\_outer \* h;  u\_outer(i) = Kp\_outer \* (error\_outer + (1/Ti\_outer) \* error\_integral\_outer);  % Внутренний ПИ-регулятор  error\_inner = u\_outer(i) - u(i);  error\_integral\_inner = error\_integral\_inner + error\_inner \* h;  u(i) = Kp\_inner \* (error\_inner + (1/Ti\_inner) \* error\_integral\_inner);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка на установившееся состояние после возмущения  if i >= start\_disturbance && isnan(transition\_time)  deviation = abs(x(1,i) - setpoint);  if deviation < steady\_state\_criteria \* abs(setpoint)  steady\_steps\_count = steady\_steps\_count + 1;  else  steady\_steps\_count = 0;  end  if steady\_steps\_count >= steady\_steps\_required  transition\_time = t(i) - t(start\_disturbance);  end  end  end  % Расчет перерегулирования  Tmin = min(x(1, start\_disturbance:end));  Tmax = max(x(1, start\_disturbance:end));  overshoot\_ratio = abs((Tmax - setpoint) / (Tmin - setpoint));  % Проверка и обновление лучших параметров  if ~isnan(transition\_time) && transition\_time < best\_time && overshoot\_ratio <= 1.04  best\_time = transition\_time;  best\_Kp\_outer = Kp\_outer;  best\_Ti\_outer = Ti\_outer;  best\_Kp\_inner = Kp\_inner;  best\_Ti\_inner = Ti\_inner;  best\_overshoot = overshoot\_ratio;  end  end  end  end  end  % Вывод наилучших коэффициентов  fprintf('Лучшее время переходного процесса: %.2f секунд\n', best\_time);  fprintf('Лучший Kp\_outer: %.4f\n', best\_Kp\_outer);  fprintf('Лучший Ti\_outer: %.2f\n', best\_Ti\_outer);  fprintf('Лучший Kp\_inner: %.4f\n', best\_Kp\_inner);  fprintf('Лучший Ti\_inner: %.2f\n', best\_Ti\_inner);  fprintf('Перерегулирование: %.4f\n', abs(best\_overshoot));  % Повторное моделирование с лучшими значениями коэффициентов  x = zeros(3, N);  error\_integral\_outer = 0;  error\_integral\_inner = 0;  u\_outer = zeros(1, N);  u = 500\*ones(1, N);  f = zeros(1, N);  f(start\_disturbance:end) = 1;  for i = 1:N-1  % Внешний ПИ-регулятор  error\_outer = setpoint - x(1,i);  error\_integral\_outer = error\_integral\_outer + error\_outer \* h;  u\_outer(i) = best\_Kp\_outer \* (error\_outer + (1/best\_Ti\_outer) \* error\_integral\_outer);  % Внутренний ПИ-регулятор  error\_inner = u\_outer(i) - u(i);  error\_integral\_inner = error\_integral\_inner + error\_inner \* h;  u(i) = best\_Kp\_inner \* (error\_inner + (1/best\_Ti\_inner) \* error\_integral\_inner);  % Ограничение мощности нагревателя  u(i) = max(0, min(u(i), u\_max));  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  end  % Построение графика с лучшими коэффициентами  figure;  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры с оптимизированным каскадным ПИ-ПИ регулятором');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |

### ПИД регулятор (с фильтром Д) и шумами +5%, по уставке

|  |
| --- |
| function system\_with\_PID  % Время моделирования  t\_total = 30000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Начальные условия [T\_р.к., T\_p, Q\_H]  x = zeros(3, N); % Массив для хранения результатов  x(:,1) = [0; 0; 0]; % Начальные температуры и теплота  % Параметры системы  c = 4200; % Удельная теплоёмкость воды, Дж/(кг·°C)  mp = 0.192; % Масса воды в рубашке, кг  mp\_k = 0.011; % Масса воды в рабочей камере, кг  k1 = 0.547; % Коэффициент теплопередачи в среду  k2 = 1.05; % Коэффициент теплопередачи между рубашкой и камерой  T = 10; % Постоянная времени системы  k = 1; % Коэффициент управляющего воздействия  % Матрицы системы  A = [-k2/(c\*mp\_k), k2/(c\*mp\_k), 0;  k2/(c\*mp), -(k2/(c\*mp) + k1/(c\*mp)), 1/(c\*mp);  0, 0, -1/T];    B = [0; 0; k/T];  % Настройка ПИД-регулятора (с минимальным перерегулированием ~4%)  Kp = 0.0008; % Пропорциональный коэффициент  Ti = 6; % Интегральная постоянная времени  Td = 1; % Дифференциальная постоянная времени  Nf = 10; % Коэффициент фильтрации для D-составляющей  % Управляющее воздействие (инициализация)  u = zeros(1, N); % Вектор управления  error\_integral = 0; % Интеграл ошибки  error\_prev = 0; % Предыдущее значение ошибки для D-составляющей  setpoint = 80; % Желаемая температура уставки (в °C)  % Основной цикл моделирования методом Эйлера  for i = 1:N-1  % Вычисляем ошибку с добавлением шума ±5%  noise = (rand() \* 0.1 - 0.05)\*setpoint; % ��� �� -5% �� +5%  error = setpoint - (x(1,i) + noise);    % Интегрируем ошибку  error\_integral = error\_integral + error \* h;    % Дифференциальная ошибка с фильтрацией  error\_derivative = (error - error\_prev) / h; % это х  error\_prev = error;  D\_term = Td / (Td + h/Nf) \* error\_derivative;    % ПИД-регулятор с фильтром  u(i) = Kp \* (error + (1/Ti) \* error\_integral + D\_term);    % Ограничение мощности нагревателя (например, мощность не может быть отрицательной)  u(i) = max(0, u(i));    % Дискретное моделирование  x(:,i+1) = (A \* x(:,i) + B \* u(i)) \* h + x(:,i);  end  % Построение графиков  plot(t, x(1,:), 'r', 'LineWidth', 2);  title('Температура рабочей камеры T\_{р.к.}(t) (X1)');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end |