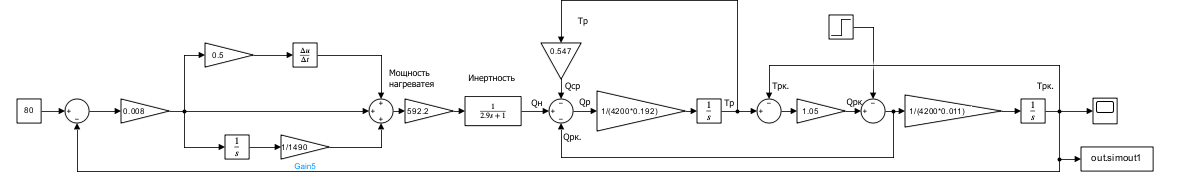
**Схема с ПИД регулятором:**

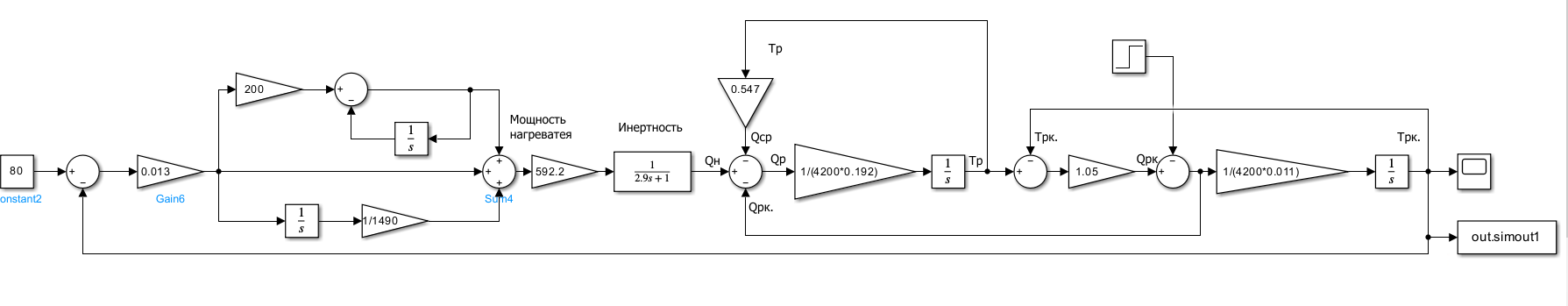


**По уставке:**

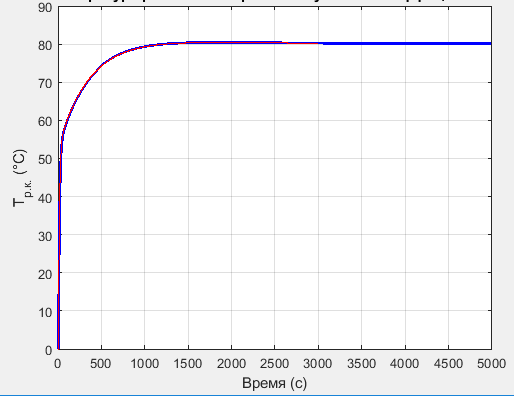
1. **Код с ПИД регулятором**

|  |
| --- |
| function PI\_regulator\_ustavka\_nonauto()  % Время моделирования  t\_total = 50000; % 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 = 592.2; % Коэффициент управляющего воздействия  setpoint = 80; % Уставка — желаемая температура  % 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  % 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];  % Параметры для стабилизации  transition\_time = NaN; % Время переходного процесса  derivative\_threshold = 0.01; % Пороговое значение производной для установившегося состояния  steady\_state\_criteria = 0.02; % Критерий ±1% от уставки  % Счетчик для проверки условия производной  derivative\_count = 0;  duration\_threshold = 500; % Продолжительность для проверки  % Параметры для поиска  Kp = 0.0006; % Значение параметар Kp  Ti = 2000; % Значение параметар Ti  Td = 500; % Дифференциальное время  error\_integral = 0; % Интеграл ошибки  previous\_error = 0; % Предыдущее значение ошибки  u = ones(1, N); % Управляющее воздействие  % 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;  % Дифференциальная ошибка  error\_derivative = (error - previous\_error) / h;  previous\_error = error;  % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral + Td \* error\_derivative);  % Дискретное моделирование  x(:,i+1) = (A\*x(:,i) + B\*u(i)) \* h + x(:,i);  % Проверка установившегося состояния  if isnan(transition\_time)  % Рассчитываем производную температуры как разность текущей и предыдущей  temperature\_derivative = abs(x(1,i+1) - x(1,i)) / h;    % Проверка на отклонение в зависимости от значения уставки  if setpoint == 0  % Если уставка 0, используем абсолютное отклонение  deviation\_met = abs(x(1,i+1)) < steady\_state\_criteria;  else  % Если уставка не 0, используем относительное отклонение  deviation\_met = abs(x(1,i+1) - setpoint) < steady\_state\_criteria \* abs(setpoint);  end  % Проверка на установившееся состояние  if deviation\_met && temperature\_derivative < derivative\_threshold  derivative\_count = derivative\_count + 1;  if derivative\_count >= duration\_threshold  transition\_time = t(i) - duration\_threshold; % Время переходного процесса с момента возмущения  end  else  derivative\_count = 0; % Сброс счётчика, если условие не выполняется  end  end  end    % Расчет перерегулирования  Tmax = max(x(1,:));  A2 = Tmax - setpoint; % Максимальное отклонение от уставки  overshoot\_ratio = (A2 / setpoint) \* 100; % Перерегулирование в процентах  if overshoot\_ratio >= 4  fprintf('Система не сбалансировалась\n');  else  % Вывод коэффициентов, времени переходного процесса и перерегулирования  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  fprintf('Kp: %.4f\n', Kp);  fprintf('Ti: %.2f\n', Ti);  fprintf('Tmax: %.4f\n', Tmax);  fprintf('A2: %.4f\n', A2);  fprintf('Перерегулирование: %.4f%%\n', overshoot\_ratio);    % % Получение информации о переходном процессе через stepinfo  % info = stepinfo(x(1,:), t); % Здесь x(1,:) - это выходная величина, а t - время  % % Вывод информации о переходном процессе  % fprintf('Время переходного процесса (stepinfo): %.2f секунд\n', info.SettlingTime);  % Построение графика  figure;  plot(t, x(1,:), 'b', 'LineWidth', 2);  title('Температура рабочей камеры с наилучшими коэффициентами');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end  end |

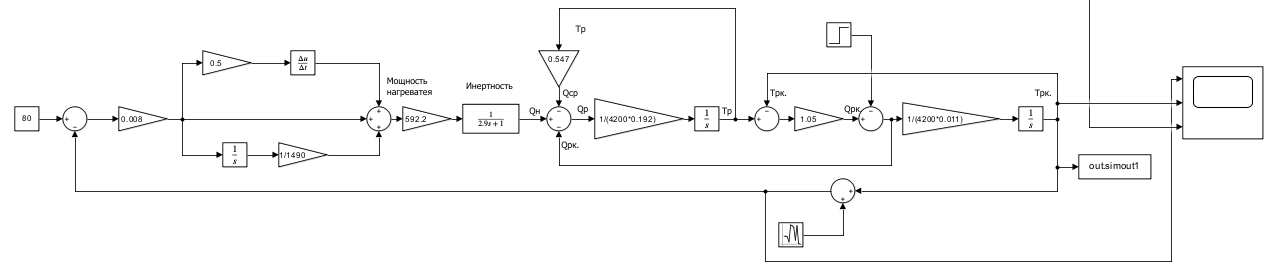
Схема с Д составляющей без шумов и фильтра



Сравнение графиков с Simulink и скрипта



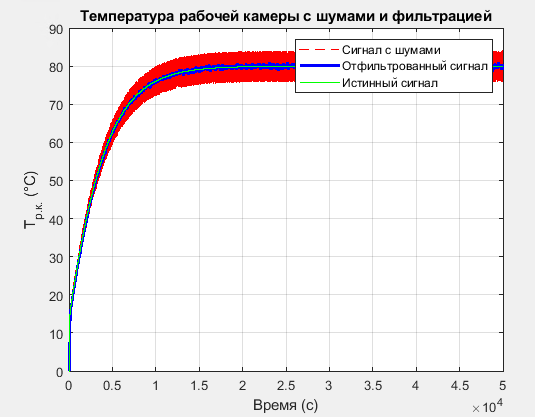
**Схема с шумами:**



1. **Код с ПИД регулятором, шумами и фильтром Д**

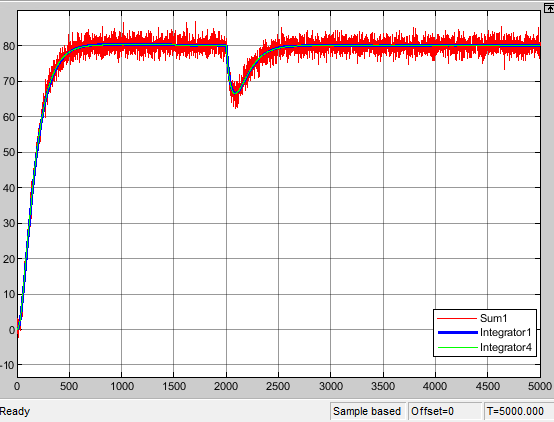
|  |
| --- |
| function PI\_regulator\_ustavka\_nonauto()  % Время моделирования  t\_total = 50000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Параметры 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 = 592.2; % Коэффициент управляющего воздействия  setpoint = 80; % Уставка — желаемая температура  % 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  % 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];  % Параметры для стабилизации  transition\_time = NaN; % Время переходного процесса  derivative\_threshold = 0.01; % Пороговое значение производной для установившегося состояния  steady\_state\_criteria = 0.02; % Критерий ±1% от уставки  % Счетчик для проверки условия производной  derivative\_count = 0;  duration\_threshold = 500; % Продолжительность для проверки  % Параметры для поиска  Kp = 0.0006; % Значение параметар Kp  Ti = 2000; % Значение параметар Ti  Td = 500; % Дифференциальное время  error\_integral = 0; % Интеграл ошибки  previous\_error = 0; % Предыдущее значение ошибки  u = ones(1, N); % Управляющее воздействие  % Параметры для шумов и фильтрации  noise\_amplitude = 0.05; % Амплитуда шума (5%)  filter\_window = 50; % Размер окна для скользящего среднего  d\_filter\_window = 50; % Размер окна фильтрации для D-составляющей  d\_filter\_buffer = zeros(1, d\_filter\_window); % Буфер для фильтрации D-составляющей  temp\_filter\_buffer = zeros(1, filter\_window); % Буфер для фильтрации температуры  % Массивы для хранения данных  noisy\_signal = zeros(1, N); % Сигнал с шумами  filtered\_signal = zeros(1, N); % Отфильтрованный сигнал  d\_filtered\_signal = zeros(1, N); % Отфильтрованная D-составляющая  % 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;  % Дифференциальная ошибка  error\_derivative = (error - previous\_error) / h;  previous\_error = error;  % Обновление буфера для фильтрации D-составляющей  d\_filter\_buffer = [d\_filter\_buffer(2:end), error\_derivative];  d\_filtered\_signal(i+1) = sum(d\_filter\_buffer) / d\_filter\_window;  % ПИ-регулятор с фильтром для D-составляющей  u(i) = Kp \* (error + (1/Ti) \* error\_integral + Td \* d\_filtered\_signal(i+1));  % Дискретное моделирование  x(:,i+1) = (A\*x(:,i) + B\*u(i)) \* h + x(:,i);  % Добавляем шум в температуру рабочей камеры  noise = noise\_amplitude \* (rand() - 0.5) \* 2; % Генерация шума в пределах ±5%  noisy\_signal(i+1) = x(1,i+1) \* (1 + noise);    % Обновление буфера для фильтрации температуры  temp\_filter\_buffer = [temp\_filter\_buffer(2:end), noisy\_signal(i+1)];  filtered\_signal(i+1) = sum(temp\_filter\_buffer) / filter\_window;  % Проверка установившегося состояния  if isnan(transition\_time)  % Рассчитываем производную температуры как разность текущей и предыдущей  temperature\_derivative = abs(filtered\_signal(i+1) - filtered\_signal(i)) / h;    % Проверка на отклонение в зависимости от значения уставки  if setpoint == 0  % Если уставка 0, используем абсолютное отклонение  deviation\_met = abs(filtered\_signal(i+1)) < steady\_state\_criteria;  else  % Если уставка не 0, используем относительное отклонение  deviation\_met = abs(filtered\_signal(i+1) - setpoint) < steady\_state\_criteria \* abs(setpoint);  end  % Проверка на установившееся состояние  if deviation\_met && temperature\_derivative < derivative\_threshold  derivative\_count = derivative\_count + 1;  if derivative\_count >= duration\_threshold  transition\_time = t(i) - duration\_threshold; % Время переходного процесса с момента возмущения  end  else  derivative\_count = 0; % Сброс счётчика, если условие не выполняется  end  end  end    % Расчет перерегулирования  Tmax = max(filtered\_signal);  A2 = Tmax - setpoint; % Максимальное отклонение от уставки  overshoot\_ratio = (A2 / setpoint) \* 100; % Перерегулирование в процентах  if overshoot\_ratio >= 4  fprintf('Система не сбалансировалась\n');  else  % Вывод коэффициентов, времени переходного процесса и перерегулирования  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  fprintf('Kp: %.4f\n', Kp);  fprintf('Ti: %.2f\n', Ti);  fprintf('Tmax: %.4f\n', Tmax);  fprintf('A2: %.4f\n', A2);  fprintf('Перерегулирование: %.4f%%\n', overshoot\_ratio);    % % Получение информации о переходном процессе через stepinfo  % info = stepinfo(x(1,:), t); % Здесь x(1,:) - это выходная величина, а t - время  % % Вывод информации о переходном процессе  % fprintf('Время переходного процесса (stepinfo): %.2f секунд\n', info.SettlingTime);  % Построение графиков  figure;  plot(t, noisy\_signal, 'r--'); % Сигнал с шумами  hold on;  plot(t, filtered\_signal, 'b', 'LineWidth', 2); % Отфильтрованный сигнал  plot(t, x(1,:), 'g', 'LineWidth', 1); % Истинный сигнал  title('Температура рабочей камеры с шумами и фильтрацией');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  legend('Сигнал с шумами', 'Отфильтрованный сигнал', 'Истинный сигнал');  grid on;  end  end |

В данный код добавлены 5% шумов и их фильтрация



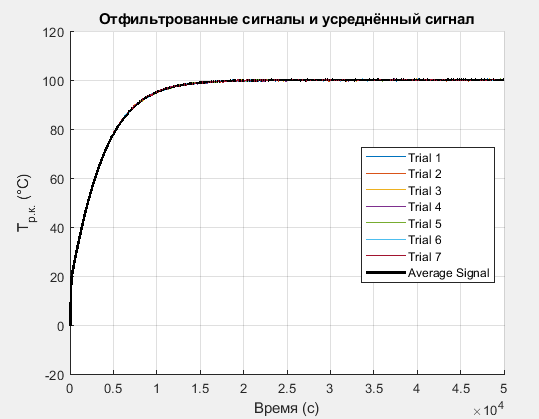
1. **Код с ПИД регулятором и возмущением**

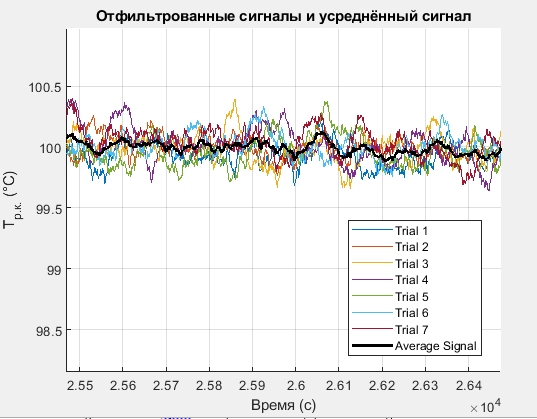
|  |
| --- |
| function PI\_regulator\_ustavka\_nonauto()  % Время моделирования  t\_total = 5000; % Общее время моделирования (в секундах)  h = 1; % Шаг по времени (в секундах)  t = 0:h:t\_total; % Вектор времени  N = length(t); % Количество шагов на основе времени моделирования и шага  % Параметры 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 = 592.2; % Коэффициент управляющего воздействия  setpoint = 80; % Уставка — желаемая температура  % Возмущение: изменение температуры окружающей среды  f = zeros(1, N);  start\_disturbance = 2000;  f(start\_disturbance:end) = 20;    % 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  % 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];  % Параметры для стабилизации  transition\_time = NaN; % Время переходного процесса  derivative\_threshold = 0.01; % Пороговое значение производной для установившегося состояния  steady\_state\_criteria = 0.02; % Критерий ±1% от уставки  % Счетчик для проверки условия производной  derivative\_count = 0;  duration\_threshold = 500; % Продолжительность для проверки  % Параметры для поиска  Kp = 0.008; % Значение параметар Kp  Ti = 1490; % Значение параметар Ti  Td = 0.5; % Дифференциальное время  error\_integral = 0; % Интеграл ошибки  previous\_error = 0; % Предыдущее значение ошибки  u = ones(1, N); % Управляющее воздействие  % 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;  % Дифференциальная ошибка  error\_derivative = (error - previous\_error) / h;  previous\_error = error;  % ПИ-регулятор  u(i) = Kp \* (error + (1/Ti) \* error\_integral + Td \* error\_derivative);  % Дискретное моделирование с учётом возмущения  x(:,i+1) = (A\*x(:,i) + B\*u(i) + F\*f(i)) \* h + x(:,i);  % Проверка установившегося состояния после возмущения  if i >= start\_disturbance && isnan(transition\_time)  % Рассчитываем производную температуры как разность текущей и предыдущей  temperature\_derivative = abs(x(1,i+1) - x(1,i)) / h;    % Проверка на отклонение в зависимости от значения уставки  if setpoint == 0  % Если уставка 0, используем абсолютное отклонение  deviation\_met = abs(x(1,i+1)) < steady\_state\_criteria;  else  % Если уставка не 0, используем относительное отклонение  deviation\_met = abs(x(1,i+1) - setpoint) < steady\_state\_criteria \* abs(setpoint);  end    % Проверка на установившееся состояние  if deviation\_met && temperature\_derivative < derivative\_threshold  derivative\_count = derivative\_count + 1;  if derivative\_count >= duration\_threshold  transition\_time = t(i) - t(start\_disturbance) - duration\_threshold; % Время переходного процесса с момента возмущения  end  else  derivative\_count = 0; % Сброс счётчика, если условие не выполняется  end  end  end    % Расчет перерегулирования  Tmax = max(x(1, start\_disturbance:end));  Tmin = min(x(1, start\_disturbance:end));  A2 = Tmax - setpoint;  A1 = Tmin - setpoint;  overshoot\_ratio = abs(A2 / A1)\*100;  if overshoot\_ratio >= 4  fprintf('Система не сбалансировалась\n');  else  % Вывод коэффициентов, времени переходного процесса и перерегулирования  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  fprintf('Kp: %.4f\n', Kp);  fprintf('Ti: %.2f\n', Ti);  fprintf('Tmax: %.4f\n', Tmax);  fprintf('A2: %.4f\n', A2);  fprintf('Перерегулирование: %.4f%%\n', overshoot\_ratio);    % % Получение информации о переходном процессе через stepinfo  % info = stepinfo(x(1,:), t); % Здесь x(1,:) - это выходная величина, а t - время  % % Вывод информации о переходном процессе  % fprintf('Время переходного процесса (stepinfo): %.2f секунд\n', info.SettlingTime);  % Построение графика  figure;  plot(t, x(1,:), 'b', 'LineWidth', 2);  title('Температура рабочей камеры с наилучшими коэффициентами');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  grid on;  end  end |



1. **ПИД регулятор с фильтром и шумами. Моделирование сигнала 7 раз и усреднение**

|  |
| --- |
| %function PI\_regulator\_ustavka\_nonauto()  % Время моделирования  t\_total = 50000; % 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 = 592.2; % Коэффициент управляющего воздействия  setpoint = 100; % Уставка — желаемая температура  % 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  % 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];  % Параметры для стабилизации  transition\_time = NaN; % Время переходного процесса  derivative\_threshold = 0.01; % Пороговое значение производной для установившегося состояния  steady\_state\_criteria = 0.02; % Критерий ±1% от уставки  % Счетчик для проверки условия производной  derivative\_count = 0;  duration\_threshold = 500; % Продолжительность для проверки  % Переменные для усреднения  average\_filtered\_signal = zeros(1, N); % Усреднённый отфильтрованный сигнал  summed\_filtered\_signal = zeros(1, N); % Сумма отфильтрованных сигналов  % Генерация сигналов 7 раз и их обработка  num\_experiment = 7; % Количество экспериментов  figure;  hold on;  for trial = 1:num\_experiment  % Параметры для поиска  Kp = 0.0006; % Значение параметра Kp  Ti = 2000; % Значение параметра Ti  Td = 500; % Дифференциальное время    error\_integral = 0; % Интеграл ошибки  previous\_error = 0; % Предыдущее значение ошибки  u = ones(1, N); % Управляющее воздействие    % Параметры для шумов и фильтрации  filter\_window = 50; % Размер окна для скользящего среднего  d\_filter\_window = 50; % Размер окна фильтрации для D-составляющей  d\_filter\_buffer = zeros(1, d\_filter\_window); % Буфер для фильтрации D-составляющей  temp\_filter\_buffer = zeros(1, filter\_window); % Буфер для фильтрации температуры    % Массивы для хранения данных  noisy\_signal = zeros(1, N); % Сигнал с шумами  filtered\_signal = zeros(1, N); % Отфильтрованный сигнал  d\_filtered\_signal = zeros(1, N); % Отфильтрованная D-составляющая  % Обнуление начальных условий для текущего эксперимента  x(:,1) = [0; 0; 0];  temp\_filter\_buffer = zeros(1, filter\_window);  noisy\_signal = zeros(1, N);  filtered\_signal = zeros(1, N);    % 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;  % Дифференциальная ошибка  error\_derivative = (error - previous\_error) / h;  previous\_error = error;  % Обновление буфера для фильтрации D-составляющей  d\_filter\_buffer = [d\_filter\_buffer(2:end), error\_derivative];  d\_filtered\_signal(i+1) = sum(d\_filter\_buffer) / d\_filter\_window;  % ПИ-регулятор с фильтром для D-составляющей  u(i) = Kp \* (error + (1/Ti) \* error\_integral + Td \* d\_filtered\_signal(i+1));  % Дискретное моделирование  x(:,i+1) = (A\*x(:,i) + B\*u(i)) \* h + x(:,i);  % Добавляем шум в температуру рабочей камеры  noise = (rand() - 0.5) \* (0.03\*setpoint); % Генерация шума в пределах ±3%  noisy\_signal(i+1) = x(1,i+1) + noise;  % Обновление буфера для фильтрации температуры  temp\_filter\_buffer = [temp\_filter\_buffer(2:end), noisy\_signal(i+1)];  filtered\_signal(i+1) = sum(temp\_filter\_buffer) / filter\_window;  end    % Добавление текущего отфильтрованного сигнала в общую сумму  summed\_filtered\_signal = summed\_filtered\_signal + filtered\_signal;    % Построение графика для текущего эксперимента  plot(t, filtered\_signal, 'LineWidth', 1, 'DisplayName', ['Trial ' num2str(trial)]);  end  % Усреднение отфильтрованного сигнала  average\_filtered\_signal = summed\_filtered\_signal / num\_trials;  % Построение графика усреднённого сигнала  plot(t, average\_filtered\_signal, 'k', 'LineWidth', 2, 'DisplayName', 'Average Signal');  title('Отфильтрованные сигналы и усреднённый сигнал');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  legend;  grid on; |





1. **Сигнал с подсчётом времени переходного процесса и перерегулирования**

|  |
| --- |
| %function PI\_regulator\_ustavka\_nonauto()  % Время моделирования  t\_total = 50000; % 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 = 592.2; % Коэффициент управляющего воздействия  setpoint = 100; % Уставка — желаемая температура  % 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  % 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];  % Параметры для стабилизации  transition\_time = NaN; % Время переходного процесса  derivative\_threshold = 0.01; % Пороговое значение производной для установившегося состояния  steady\_state\_criteria = 0.02; % Критерий ±2% от уставки  % Счетчик для проверки условия производной  derivative\_count = 0;  duration\_threshold = 5000; % Продолжительность для проверки  % Переменные для усреднения  average\_filtered\_signal = zeros(1, N); % Усреднённый отфильтрованный сигнал  summed\_filtered\_signal = zeros(1, N); % Сумма отфильтрованных сигналов  % Генерация сигналов 7 раз и их обработка  num\_experiment = 7; % Количество экспериментов  figure;  hold on;  for trial = 1:num\_experiment  % Параметры для поиска  Kp = 0.0006; % Значение параметра Kp  Ti = 2000; % Значение параметра Ti  Td = 500; % Дифференциальное время    error\_integral = 0; % Интеграл ошибки  previous\_error = 0; % Предыдущее значение ошибки  u = ones(1, N); % Управляющее воздействие    % Параметры для шумов и фильтрации  filter\_window = 50; % Размер окна для скользящего среднего  d\_filter\_window = 50; % Размер окна фильтрации для D-составляющей  d\_filter\_buffer = zeros(1, d\_filter\_window); % Буфер для фильтрации D-составляющей  temp\_filter\_buffer = zeros(1, filter\_window); % Буфер для фильтрации температуры    % Массивы для хранения данных  noisy\_signal = zeros(1, N); % Сигнал с шумами  filtered\_signal = zeros(1, N); % Отфильтрованный сигнал  d\_filtered\_signal = zeros(1, N); % Отфильтрованная D-составляющая  % Обнуление начальных условий для текущего эксперимента  x(:,1) = [0; 0; 0];  temp\_filter\_buffer = zeros(1, filter\_window);  noisy\_signal = zeros(1, N);  filtered\_signal = zeros(1, N);    % 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;  % Дифференциальная ошибка  error\_derivative = (error - previous\_error) / h;  previous\_error = error;  % Обновление буфера для фильтрации D-составляющей  d\_filter\_buffer = [d\_filter\_buffer(2:end), error\_derivative];  d\_filtered\_signal(i+1) = sum(d\_filter\_buffer) / d\_filter\_window;  % ПИ-регулятор с фильтром для D-составляющей  u(i) = Kp \* (error + (1/Ti) \* error\_integral + Td \* d\_filtered\_signal(i+1));  % Дискретное моделирование  x(:,i+1) = (A\*x(:,i) + B\*u(i)) \* h + x(:,i);  % Добавляем шум в температуру рабочей камеры  noise = (rand() - 0.5) \* (0.03\*setpoint); % Генерация шума в пределах ±3%  noisy\_signal(i+1) = x(1,i+1) + noise;  % Обновление буфера для фильтрации температуры  temp\_filter\_buffer = [temp\_filter\_buffer(2:end), noisy\_signal(i+1)];  filtered\_signal(i+1) = sum(temp\_filter\_buffer) / filter\_window;  % Проверка установившегося состояния  if isnan(transition\_time)  % Рассчитываем отклонение от уставки  deviation = abs(filtered\_signal(i+1) - setpoint);  relative\_deviation = deviation / abs(setpoint);    % Проверка на отклонение в пределах 2% от уставки  deviation\_met = relative\_deviation <= steady\_state\_criteria;    % Если отклонение соответствует критерию, увеличиваем счётчик  if deviation\_met  derivative\_count = derivative\_count + 1;  % Если сигнал удерживается в пределах отклонения 5000 секунд  if derivative\_count >= duration\_threshold  transition\_time = t(i) - duration\_threshold; % Фиксируем время переходного процесса  end  else  derivative\_count = 0; % Сброс счётчика при выходе за пределы отклонения  end  end    end    % Добавление текущего отфильтрованного сигнала в общую сумму  summed\_filtered\_signal = summed\_filtered\_signal + filtered\_signal;    % % Построение графика для текущего эксперимента  % plot(t, filtered\_signal, 'LineWidth', 1, 'DisplayName', ['Trial ' num2str(trial)]);  end  % Усреднение отфильтрованного сигнала  average\_filtered\_signal = summed\_filtered\_signal / num\_trials;  % Расчет перерегулирования  Tmax = max(average\_filtered\_signal);  A2 = Tmax - setpoint; % Максимальное отклонение от уставки  overshoot\_ratio = (A2 / setpoint) \* 100; % Перерегулирование в процентах  if overshoot\_ratio >= 4  fprintf('Система не сбалансировалась\n');  else  % Вывод коэффициентов, времени переходного процесса и перерегулирования  fprintf('Лучшее время переходного процесса: %.2f секунд\n', transition\_time);  fprintf('Kp: %.4f\n', Kp);  fprintf('Ti: %.2f\n', Ti);  fprintf('Tmax: %.4f\n', Tmax);  fprintf('A2: %.4f\n', A2);  fprintf('Перерегулирование: %.4f%%\n', overshoot\_ratio);  % Построение графика усреднённого сигнала  plot(t, average\_filtered\_signal, 'k', 'LineWidth', 2, 'DisplayName', 'Average Signal');  title('Отфильтрованные сигналы и усреднённый сигнал');  xlabel('Время (с)');  ylabel('T\_{р.к.} (°C)');  legend;  grid on;  end |

