



Write MATLAB code to implement an example mathematical model for a distillation column, which describes its dynamics and the relationships between feed rates and temperature. Do not plot the results.



In this example, we'll implement a simple mathematical model of a distillation column using a continuous-time dynamic model. The model includes two main equations: one for the temperature and another for the molar holdup (amount of substance in the column).

The code below illustrates the implementation of a distillation column model in MATLAB. The example is simplified and doesn't represent a real distillation column in detail, but it provides a starting point for understanding the relationships between feed rates and temperature.

```
matlab Copy code

% Distillation column model

% Parameters
R = 8.314;           % Gas constant (J/mol*K)
T_boil = 373.15;    % Boiling temperature of the mixture (K)
T_feed = 300;        % Feed temperature (K)
F_feed = 1;          % Feed flow rate (mol/s)
H_vap = 40e3;        % Heat of vaporization (J/mol)
V_max = 10;          % Maximum column volume (L)
k = 0.5;             % Heat transfer coefficient (W/K)

% Initial conditions
V_init = 5;          % Initial column volume (L)
T_init = 350;        % Initial column temperature (K)

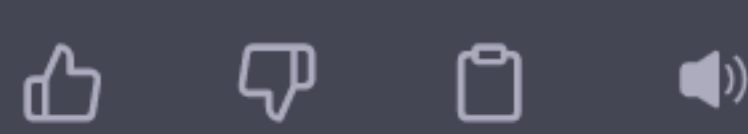
% Time span for simulation
t_span = [0, 3600]; % Simulation time (s)

% Distillation column dynamic model
dist_col_model = @(t, y) [
    F_feed - (y(1) / V_max) * H_vap * (T_boil - y(2)) / R / T_boil; % dV/dt
    k * (T_feed - y(2)) / (y(1) * R) - (y(1) / V_max) * H_vap * (T_boil - y(2)) / R / T_boil; % dT/dt
];

% Solve the ODE system
y_init = [V_init; T_init];
[t, y] = ode45(dist_col_model, t_span, y_init);

% Display the results
disp('Time (s)    Volume (L)    Temperature (K)');
disp([t, y]);
```

This code defines a distillation column model with given parameters and initial conditions. The model equations are solved using the `ode45` function, which numerically integrates the system of ordinary differential equations (ODEs) representing the column's dynamics. The results, including time, volume, and temperature, are displayed in the MATLAB console. Note that this model does not include a graphical representation of the results, as requested.



Write MATLAB code to implement a model-predictive control scheme using this mathematical model for the distillation column.



In this example, we'll implement a Model Predictive Control (MPC) scheme for the distillation column using the simplified mathematical model from the previous example. Note that this example is a very basic implementation of an MPC and can be improved in various ways for a real-world application.

```
matlab Copy code

% Distillation column model-predictive control

% Required additional parameters
T_setpoint = 365;    % Temperature setpoint (K)
F_feed_min = 0.1;    % Minimum feed flow rate (mol/s)
F_feed_max = 3;      % Maximum feed flow rate (mol/s)
dt = 1;              % Time step for control (s)
control_horizon = 100; % Control horizon (number of steps)
u_init = F_feed;      % Initial control input (feed flow rate)

% MPC cost function
mpc_cost_function = @(u) sum((y_mpc(:, 2) - T_setpoint).^2) + 1e-4 * sum(u.^2);

% Optimization options
optim_opts = optimoptions('fmincon', 'Display', 'off');

% MPC loop
N = length(t);
u_mpc = zeros(N, 1);
y_mpc = zeros(N, 2);

% Initial conditions for the MPC loop
u_mpc(1) = u_init;
y_mpc(1, :) = y_init';

for i = 2:N
    % Solve the MPC optimization problem
    u_opt = fmincon(mpc_cost_function, u_init, [], [], [], [], F_feed_min, F_feed_max, [], optim_opts);

    % Update the control input
    u_mpc(i) = u_opt;

    % Update the distillation column model with the new control input
    F_feed = u_opt;
    dist_col_model = @(t, y) [
        F_feed - (y(1) / V_max) * H_vap * (T_boil - y(2)) / R / T_boil;
        k * (T_feed - y(2)) / (y(1) * R) - (y(1) / V_max) * H_vap * (T_boil - y(2)) / R / T_boil;
    ];

    % Integrate the model for one time step
    [t_step, y_step] = ode45(dist_col_model, [t(i-1), t(i)], y_mpc(i-1, :));
    y_mpc(i, :) = y_step(end, :);
end

% Display the results
disp('Time (s)    Volume (L)    Temperature (K)    Feed flow rate (mol/s)');
disp([t, y_mpc, u_mpc]);
```

This code implements an MPC loop that controls the feed flow rate to maintain the temperature at the desired setpoint. The MPC optimization problem is solved using the `fmincon` function, which finds the optimal control input to minimize the cost function within the given constraints. The cost function incorporates both the tracking error (deviation of the temperature from the setpoint) and the control effort (magnitude of the feed flow rate).

The distillation column model is updated with the new control input and integrated for one time step. The results, including time, volume, temperature, and feed flow rate, are displayed in the MATLAB console. Note that this example is a very basic implementation of an MPC, and the performance can be improved by using more advanced optimization algorithms, adjusting the cost function weights, or incorporating additional constraints.

