Write MATLAB code to implement model-predictive control to optimize the trajectory of a aircraft.

To implement Model Predictive Control (MPC) for an aircraft trajectory optimization in MATLAB, you will need to define the system model, constraints, cost function, and control horizon. Here's a simple example of an MPC implementation for an aircraft trajectory optimization:

1. Define the system model (e.g., a discrete-time linear state-space model):

MATLAB — Copy code

```
A = [1.0 0.1; 0 0.95];    % State matrix
B = [0; 0.1];             % Input matrix
C = [1 0];                % Output matrix
D = 0;                    % Feed-through matrix
Ts = 0.1;                 % Sampling time in seconds

sys = ss(A, B, C, D, Ts); % Discrete-time linear state-space model
```

2. Define the MPC controller:

MATLAB — Copy code

```
% Set MPC parameters
control_horizon = 10;
prediction_horizon = 30;
weights = struct('MV', 0.1, 'MVRate', 0.01, 'OV', 1);

% Create the MPC controller
mpc_controller = mpc(sys, Ts, control_horizon, prediction_horizon, weights);
```

3. Set constraints on the input and states:

MATLAB — Copy code

```
% Input constraints (e.g., control input limits)
mpc_controller.MV(1).Min = -0.5;
mpc_controller.MV(1).Max = 0.5;

% State constraints (e.g., altitude and velocity limits)
mpc_controller.States(1).Min = -Inf;
mpc_controller.States(1).Max = Inf;
mpc_controller.States(2).Min = -Inf;
mpc_controller.States(2).Max = Inf;
```

4. Define the simulation parameters and initial conditions:

MATLAB — Copy code

```
% Simulation parameters
t_sim = 20;                    % Simulation time in seconds
n_steps = floor(t_sim / Ts);   % Number of simulation steps

% Initial conditions
x0 = [0; 0];                   % Initial state vector (altitude and velocity)
u = zeros(1, n_steps);         % Control input vector
y = zeros(1, n_steps);         % Output vector (altitude)
y_ref = ones(1, n_steps) * 100;  % Reference altitude (desired altitude)
```

5. Run the simulation and apply the MPC controller:

MATLAB — Copy code

```
x = x0;
for k = 1:n_steps
    % Get the optimal control input
    u(k) = mpcmove(mpc_controller, x, y(k), y_ref(k));

    % Update the system state
    x = A * x + B * u(k);

    % Update the output
    y(k+1) = C * x + D * u(k);
end
```

6. Plot the results:

MATLAB — Copy code

```
time = 0:Ts:(n_steps * Ts);

figure;
subplot(2, 1, 1);
plot(time, [x0(1), y(1:end-1)], 'LineWidth', 2);
hold on;
plot(time, y_ref, '--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Altitude (m)');
legend('Aircraft Altitude', 'Reference Altitude');
title('Aircraft Trajectory Optimization using MPC');
grid on;

subplot(2, 1, 2);
stairs(time(1:end-1), u, 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input');
title('Control Input vs. Time');
grid on;
```

This is

continue

a simple example, and you may need to customize the code based on your specific aircraft model, constraints, and performance objectives.

Keep in mind that more complex aircraft models might need to consider additional factors, such as aerodynamic forces, aircraft dynamics, and environmental conditions. In these cases, you might need to use a nonlinear model and implement a nonlinear MPC algorithm.

You can also utilize the built-in functions in MATLAB and Simulink for implementing MPC. For example, you can use the "mpc" function to create an MPC controller, and "mpcmove" to compute the optimal control input. Furthermore, you can integrate your MPC controller into a Simulink model to simulate and analyze the performance of the aircraft.

Here is the complete MATLAB code for the simple MPC implementation described above:

MATLAB — Copy code

```
% Define the system model
A = [1.0 0.1; 0 0.95];
B = [0; 0.1];
C = [1 0];
D = 0;
Ts = 0.1;

sys = ss(A, B, C, D, Ts);

% Define the MPC controller
control_horizon = 10;
prediction_horizon = 30;
weights = struct('MV', 0.1, 'MVRate', 0.01, 'OV', 1);

mpc_controller = mpc(sys, Ts, control_horizon, prediction_horizon, weights);

% Set constraints
mpc_controller.MV(1).Min = -0.5;
mpc_controller.MV(1).Max = 0.5;
mpc_controller.States(1).Min = -Inf;
mpc_controller.States(1).Max = Inf;
mpc_controller.States(2).Min = -Inf;
mpc_controller.States(2).Max = Inf;

% Simulation parameters
t_sim = 20;
n_steps = floor(t_sim / Ts);

% Initial conditions
x0 = [0; 0];
u = zeros(1, n_steps);
y = zeros(1, n_steps);
y_ref = ones(1, n_steps) * 100;

% Run the simulation
x = x0;
for k = 1:n_steps
    u(k) = mpcmove(mpc_controller, x, y(k), y_ref(k));
    x = A * x + B * u(k);
    y(k+1) = C * x + D * u(k);
end

% Plot the results
time = 0:Ts:(n_steps * Ts);

figure;
subplot(2, 1, 1);
plot(time, [x0(1), y(1:end-1)], 'LineWidth', 2);
hold on;
plot(time, y_ref, '--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Altitude (m)');
legend('Aircraft Altitude', 'Reference Altitude');
title('Aircraft Trajectory Optimization using MPC');
grid on;

subplot(2, 1, 2);
stairs(time(1:end-1), u, 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input');
title('Control Input vs. Time');
grid on;
```

This code provides a simple demonstration of using MPC to optimize the trajectory of an aircraft. You can adapt and expand this code based on the specific requirements of your project.