```
function [ desired_state ] = traj_generator(t_real, state, waypoints)

% TRAJ_GENERATOR: Generate the trajectory passing through all
% positions listed in the waypoints list
%
% NOTE: This function would be called with variable number of input arguments.
% During initialization, it will be called with arguments
% trajectory_generator([], [], waypoints) and later, while testing, it will be
% called with only t and state as arguments, so your code should be able to
% handle that. This can be done by checking the number of arguments to the
% function using the "nargin" variable, check the MATLAB documentation for more
% information.
%
% t,state: time and current state (same variable as "state" in controller)
% that you may use for computing desired_state
%
% waypoints: The 3xP matrix listing all the points you much visited in order
% along the generated trajectory
%
% desired_state: Contains all the information that is passed to the
% controller for generating inputs for the quadrotor
%
% It is suggested to use "persistent" variables to store the waypoints during
% the initialization call of trajectory_generator.
```

Example code:

Note that this is an example of naive trajectory generator that simply moves the quadrotor along a stright line between each pair of consecutive waypoints using a constant velocity of 0.5 m/s. Note that this is only a sample, and you should write your own trajectory generator for the submission.

```
% persistent waypoints0 traj_time d0
% if nargin > 2
%     d = waypoints(:,2:end) - waypoints(:,1:end-1);
%     d0 = 2 * sqrt(d(1,:).^2 + d(2,:).^2 + d(3,:).^2);
%     traj_time = [0, cumsum(d0)];
%     waypoints0 = waypoints;
% else
%     if(t > traj_time(end))
%         t = traj_time(end);
%     end
%     t_index = find(traj_time >= t,1);
%
%     if(t_index > 1)
%         t = t - traj_time(t_index-1);
%     end
%     if(t == 0)
%         desired_state.pos = waypoints0(:,1);
%     else
%         scale = t/d0(t_index-1);
%         desired_state.pos = (1 - scale) * waypoints0(:,t_index-1) + scale * wayp
%     end
%     desired_state.vel = zeros(3,1);
%     desired_state.acc = zeros(3,1);
%     desired_state.yaw = 0;
%     desired_state.yawdot = 0;
% end
%
```

Fill in your code here

```
persistent X1 X2 X3
persistent waypoints0 traj_time d0
if nargin > 2
    d = waypoints(:,2:end) - waypoints(:,1:end-1);
    d0 = 2 * sqrt(d(1,:).^2 + d(2,:).^2 + d(3,:).^2);
    traj_time = [0, cumsum(d0)];
    waypoints0 = waypoints;
    %    we should solve for x, y, z independently, we really need coefficients al
    %    https://www.coursera.org/learn/robotics-flight/discussions/weeks/4/thread

    syms t positive
    syms c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6 c_1_7 real
    syms c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6 c_2_7 real
    syms c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_6 c_3_7 real
    syms c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_6 c_4_7 real
    syms p1 p2 p3 p4
    S0 = traj_time(1);
    S1 = traj_time(2);
    S2 = traj_time(3);
    S3 = traj_time(4);
    S4 = traj_time(5);
    p1 = c_1_0 + c_1_1*(t-S0)/(S1-S0) + c_1_2*((t-S0)/(S1-S0))^2 + c_1_3*((t-S0)/(
    p2 = c_2_0 + c_2_1*(t-S1)/(S2-S1) + c_2_2*((t-S1)/(S2-S1))^2 + c_2_3*((t-S1)/(
    p3 = c_3_0 + c_3_1*(t-S2)/(S3-S2) + c_3_2*((t-S2)/(S3-S2))^2 + c_3_3*((t-S2)/(
    p4 = c_4_0 + c_4_1*(t-S3)/(S4-S3) + c_4_2*((t-S3)/(S4-S3))^2 + c_4_3*((t-S3)/(

    p1dot = diff(p1,t);
    p2dot = diff(p2,t);
    p3dot = diff(p3,t);
    p4dot = diff(p4,t);
    p1dot2 = diff(p1dot, t);
    p2dot2 = diff(p2dot, t);
    p3dot2 = diff(p3dot, t);
    p4dot2 = diff(p4dot, t);
    p1dot3 = diff(p1dot2, t);
    p2dot3 = diff(p2dot2, t);
    p3dot3 = diff(p3dot2, t);
    p4dot3 = diff(p4dot2, t);
    p1dot4 = diff(p1dot3, t);
    p2dot4 = diff(p2dot3, t);
    p3dot4 = diff(p3dot3, t);
    p4dot4 = diff(p4dot3, t);
    p1dot5 = diff(p1dot4, t);
    p2dot5 = diff(p2dot4, t);
```

```
p3dot5 = diff(p3dot4, t);
p4dot5 = diff(p4dot4, t);
p1dot6 = diff(p1dot5, t);
p2dot6 = diff(p2dot5, t);
p3dot6 = diff(p3dot5, t);
p4dot6 = diff(p4dot5, t);

A = zeros(32, 32);
b = zeros(32, 1);


A(1, 1:8) = subs(jacobian(p1, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6 c_1_7
A(2, 1:8) = subs(jacobian(p1, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6 c_1_7
A(3, 9:16) = subs(jacobian(p2, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6 c_2_
A(4, 9:16) = subs(jacobian(p2, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6 c_2_
A(5, 17:24) = subs(jacobian(p3, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_6 c_3
A(6, 17:24) = subs(jacobian(p3, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_6 c_3
A(7, 25:32) = subs(jacobian(p4, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_6 c_4
A(8, 25:32) = subs(jacobian(p4, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_6 c_4
A(9, 1:8) = subs(jacobian(p1dot, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6 c_
A(10, 25:32) = subs(jacobian(p4dot, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_6
A(11, 1:8) = subs(jacobian(p1dot2, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(12, 25:32) = subs(jacobian(p4dot2, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_
A(13, 1:8) = subs(jacobian(p1dot3, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(14, 25:32) = subs(jacobian(p4dot3, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_

A(15, 1:8) = subs(jacobian(p1dot, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6 c
A(15, 9:16) = -subs(jacobian(p2dot, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(16, 1:8) = subs(jacobian(p1dot2, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(16, 9:16) = -subs(jacobian(p2dot2, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_
A(17, 1:8) = subs(jacobian(p1dot3, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(17, 9:16) = -subs(jacobian(p2dot3, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_
A(18, 1:8) = subs(jacobian(p1dot4, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(18, 9:16) = -subs(jacobian(p2dot4, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_
A(19, 1:8) = subs(jacobian(p1dot5, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(19, 9:16) = -subs(jacobian(p2dot5, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_
A(20, 1:8) = subs(jacobian(p1dot6, [c_1_0 c_1_1 c_1_2 c_1_3 c_1_4 c_1_5 c_1_6
A(20, 9:16) = -subs(jacobian(p2dot6, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_

A(21, 9:16) = subs(jacobian(p2dot, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(21, 17:24) = -subs(jacobian(p3dot, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
A(22, 9:16) = subs(jacobian(p2dot2, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(22, 17:24) = -subs(jacobian(p3dot2, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3
A(23, 9:16) = subs(jacobian(p2dot3, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(23, 17:24) = -subs(jacobian(p3dot3, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3
A(24, 9:16) = subs(jacobian(p2dot4, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(24, 17:24) = -subs(jacobian(p3dot4, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3
A(25, 9:16) = subs(jacobian(p2dot5, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(25, 17:24) = -subs(jacobian(p3dot5, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3
A(26, 9:16) = subs(jacobian(p2dot6, [c_2_0 c_2_1 c_2_2 c_2_3 c_2_4 c_2_5 c_2_6
A(26, 17:24) = -subs(jacobian(p3dot6, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3

A(27, 17:24) = subs(jacobian(p3dot, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_6
A(27, 25:32) = -subs(jacobian(p4dot, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4_
        A(28, 17:24) = subs(jacobian(p3dot2, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
        A(28, 25:32) = -subs(jacobian(p4dot2, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4
        A(29, 17:24) = subs(jacobian(p3dot3, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
        A(29, 25:32) = -subs(jacobian(p4dot3, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4
        A(30, 17:24) = subs(jacobian(p3dot4, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
        A(30, 25:32) = -subs(jacobian(p4dot4, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4
        A(31, 17:24) = subs(jacobian(p3dot5, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
        A(31, 25:32) = -subs(jacobian(p4dot5, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4
        A(32, 17:24) = subs(jacobian(p3dot6, [c_3_0 c_3_1 c_3_2 c_3_3 c_3_4 c_3_5 c_3_
        A(32, 25:32) = -subs(jacobian(p4dot6, [c_4_0 c_4_1 c_4_2 c_4_3 c_4_4 c_4_5 c_4
        for k = 1:3
            for i = 1:(size(waypoints, 2)-1)
                b(2*i-1) = waypoints(k, i);
                b(2*i) = waypoints(k, i+1);
            end
            if(k == 1)
                X1 = A \ b;
            elseif(k==2)
                X2 = A \ b;
            elseif(k==3)
                X3 = A \ b;
            end
        end
    else
    %     the next line, syms function, really takes a very long time, so DO NOT USE I
    %     syms t
        S0 = traj_time(1);
        S1 = traj_time(2);
        S2 = traj_time(3);
        S3 = traj_time(4);
        S4 = traj_time(5);
    %
    %     p1x = X1(1) + X1(2)*(t-S0)/(S1-S0) + X1(3)*((t-S0)/(S1-S0))^2 + X1(4)*((t-S0
    %     p2x = X1(9) + X1(10)*(t-S1)/(S2-S1) + X1(11)*((t-S1)/(S2-S1))^2 + X1(12)*((t
    %     p3x = X1(17) + X1(18)*(t-S2)/(S3-S2) + X1(19)*((t-S2)/(S3-S2))^2 + X1(20)*((
    %     p4x = X1(25) + X1(26)*(t-S3)/(S4-S3) + X1(27)*((t-S3)/(S4-S3))^2 + X1(28)*((
    %
    %     p1y = X2(1) + X2(2)*(t-S0)/(S1-S0) + X2(3)*((t-S0)/(S1-S0))^2 + X2(4)*((t-S0
    %     p2y = X2(9) + X2(10)*(t-S1)/(S2-S1) + X2(11)*((t-S1)/(S2-S1))^2 + X2(12)*((t
    %     p3y = X2(17) + X2(18)*(t-S2)/(S3-S2) + X2(19)*((t-S2)/(S3-S2))^2 + X2(20)*((
    %     p4y = X2(25) + X2(26)*(t-S3)/(S4-S3) + X2(27)*((t-S3)/(S4-S3))^2 + X2(28)*((
    %
    %     p1z = X3(1) + X3(2)*(t-S0)/(S1-S0) + X3(3)*((t-S0)/(S1-S0))^2 + X3(4)*((t-S0
    %     p2z = X3(9) + X3(10)*(t-S1)/(S2-S1) + X3(11)*((t-S1)/(S2-S1))^2 + X3(12)*((t
    %     p3z = X3(17) + X3(18)*(t-S2)/(S3-S2) + X3(19)*((t-S2)/(S3-S2))^2 + X3(20)*((
    %     p4z = X3(25) + X3(26)*(t-S3)/(S4-S3) + X3(27)*((t-S3)/(S4-S3))^2 + X3(28)*((

        if(t_real >= traj_time(end))
            t_real = traj_time(end);
        end

        if(t_real>=traj_time(1) && t_real<traj_time(2))
            desired_state.pos(1,1) = X1(1) + X1(2)*(t_real-S0)/(S1-S0) + X1(3)*((t_rea
            desired_state.pos(2,1) = X2(1) + X2(2)*(t_real-S0)/(S1-S0) + X2(3)*((t_rea
            desired_state.pos(3,1) = X3(1) + X3(2)*(t_real-S0)/(S1-S0) + X3(3)*((t_rea
            desired_state.vel(1,1) = X1(2)/(S1-S0) + X1(3)*2*(t_real-S0)/((S1-S0)^2) +
            desired_state.vel(2,1) = X2(2)/(S1-S0) + X2(3)*2*(t_real-S0)/((S1-S0)^2) +
            desired_state.vel(3,1) = X3(2)/(S1-S0) + X3(3)*2*(t_real-S0)/((S1-S0)^2) +
            desired_state.acc(1,1) = X1(3)*2/((S1-S0)^2) + X1(4)*3*2*(t_real-S0)/((S1-
            desired_state.acc(2,1) = X2(3)*2/((S1-S0)^2) + X2(4)*3*2*(t_real-S0)/((S1-
            desired_state.acc(3,1) = X3(3)*2/((S1-S0)^2) + X3(4)*3*2*(t_real-S0)/((S1-
        elseif(t_real>=traj_time(2) && t_real<traj_time(3))
      desired_state.pos(1,1) = X1(1+8) + X1(2+8)*(t_real-S1)/(S2-S1) + X1(3+8)*((t_rea
            desired_state.pos(2,1) = X2(1+8) + X2(2+8)*(t_real-S1)/(S2-S1) + X2(3+8)*(
            desired_state.pos(3,1) = X3(1+8) + X3(2+8)*(t_real-S1)/(S2-S1) + X3(3+8)*(
            desired_state.vel(1,1) = X1(2+8)/(S2-S1) + X1(3+8)*2*(t_real-S1)/((S2-S1)^
            desired_state.vel(2,1) = X2(2+8)/(S2-S1) + X2(3+8)*2*(t_real-S1)/((S2-S1)^
            desired_state.vel(3,1) = X3(2+8)/(S2-S1) + X3(3+8)*2*(t_real-S1)/((S2-S1)^
            desired_state.acc(1,1) = X1(3+8)*2/((S2-S1)^2) + X1(4+8)*3*2*(t_real-S1)/(
            desired_state.acc(2,1) = X2(3+8)*2/((S2-S1)^2) + X2(4+8)*3*2*(t_real-S1)/(
            desired_state.acc(3,1) = X3(3+8)*2/((S2-S1)^2) + X3(4+8)*3*2*(t_real-S1)/(

    %    desired_state.pos(1,1) = subs(p2x, t, t_real);
    %    desired_state.pos(2,1) = subs(p2y, t, t_real);
    %    desired_state.pos(3,1) = subs(p2z, t, t_real);
    %    desired_state.vel(1,1) = subs(diff(p2x,t), t, t_real);
    %    desired_state.vel(2,1) = subs(diff(p2y,t), t, t_real);
    %    desired_state.vel(3,1) = subs(diff(p2z,t), t, t_real);
    %    desired_state.acc(1,1) = subs(diff(p2x,t,2), t, t_real);
    %    desired_state.acc(2,1) = subs(diff(p2y,t,2), t, t_real);
    %    desired_state.acc(3,1) = subs(diff(p2z,t,2), t, t_real);
        elseif(t_real>=traj_time(3) && t_real<traj_time(4))
            desired_state.pos(1,1) = X1(1+16) + X1(2+16)*(t_real-S2)/(S3-S2) + X1(3+16
            desired_state.pos(2,1) = X2(1+16) + X2(2+16)*(t_real-S2)/(S3-S2) + X2(3+16
            desired_state.pos(3,1) = X3(1+16) + X3(2+16)*(t_real-S2)/(S3-S2) + X3(3+16
            desired_state.vel(1,1) = X1(2+16)/(S3-S2) + X1(3+16)*2*(t_real-S2)/((S3-S2
            desired_state.vel(2,1) = X2(2+16)/(S3-S2) + X2(3+16)*2*(t_real-S2)/((S3-S2
            desired_state.vel(3,1) = X3(2+16)/(S3-S2) + X3(3+16)*2*(t_real-S2)/((S3-S2
            desired_state.acc(1,1) = X1(3+16)*2/((S3-S2)^2) + X1(4+16)*3*2*(t_real-S2)
            desired_state.acc(2,1) = X2(3+16)*2/((S3-S2)^2) + X2(4+16)*3*2*(t_real-S2)
            desired_state.acc(3,1) = X3(3+16)*2/((S3-S2)^2) + X3(4+16)*3*2*(t_real-S2)
        elseif(t_real>=traj_time(4) && t_real<=traj_time(5))
            desired_state.pos(1,1) = X1(1+24) + X1(2+24)*(t_real-S3)/(S4-S3) + X1(3+24
            desired_state.pos(2,1) = X2(1+24) + X2(2+24)*(t_real-S3)/(S4-S3) + X2(3+24
            desired_state.pos(3,1) = X3(1+24) + X3(2+24)*(t_real-S3)/(S4-S3) + X3(3+24
            desired_state.vel(1,1) = X1(2+24)/(S4-S3) + X1(3+24)*2*(t_real-S3)/((S4-S3
            desired_state.vel(2,1) = X2(2+24)/(S4-S3) + X2(3+24)*2*(t_real-S3)/((S4-S3
            desired_state.vel(3,1) = X3(2+24)/(S4-S3) + X3(3+24)*2*(t_real-S3)/((S4-S3
            desired_state.acc(1,1) = X1(3+24)*2/((S4-S3)^2) + X1(4+24)*3*2*(t_real-S3)
            desired_state.acc(2,1) = X2(3+24)*2/((S4-S3)^2) + X2(4+24)*3*2*(t_real-S3)
            desired_state.acc(3,1) = X3(3+24)*2/((S4-S3)^2) + X3(4+24)*3*2*(t_real-S3)
        end
        %  t_index = find(traj_time >= t,1);
        %
        %  if(t_index > 1)
        %      t = t - traj_time(t_index-1);
        %  end
        %  if(t == 0)
```

```
    %         desired_state.pos = waypoints0(:,1);
    %     else
    %         scale = t/d0(t_index-1);
    %         desired_state.pos = (1 - scale) * waypoints0(:,t_index-1) + scale *
    %     end
%     desired_state.pos = (desired_state.pos)';
%     desired_state.vel = zeros(3,1);
%     desired_state.acc = zeros(3,1);
    desired_state.yaw = 0;
    desired_state.yawdot = 0;
end
% desired_state.pos = zeros(3,1);
% desired_state.vel = zeros(3,1);
% desired_state.acc = zeros(3,1);
% desired_state.yaw = 0;
```

```
Index exceeds matrix dimensions.

Error in traj_generator (line 187)
    S0 = traj_time(1);
```

```
end
```

Published with MATLAB® 8.0