```
%% Problem 2b
close all;
clear all;
clc;

%% Input
%% Problem 2a K>50
% Working
% Pb4 = [-5 0 .9 0 1.1;0 1 1 1 1;0 1 0 -10 0];

%% Problem 2a T<30
% Working
% Pb4 = [1 3 2 2 2;1 5 1 5 2; 1 2 3 4 5];

%% User Input
fprintf('Input the P0, P1, P0dot_bar, P1dot_bar as [#;#;#] for each when prompt and this↙
function will plot the Bezier Curve.\n')
fprintf('or\n')
fprintf('Input the Ph as a 3x5 matrix when prompt and this function will plot the Bezier↙
Curve.\n\n')
fprintf('Do you want to input as vectors (type "1") or matrix (type "2").\n')
type = input('Input Choice Here ("1" or "2" only):');
typeif = num2str(type);

if strcmp(typeif,'1')
    fprintf('\nType in a 3x1 matrix for each positon or tangent vector.\n')
    P0_bar = input('P0_bar (3by1 matrix) as [#;#;#]:');            % 3by1 matrix
    P1_bar = input('P1_bar (3by1 matrix) as [#;#;#]:');            % 3by1 matrix
    P2_bar = input('P2_bar (3by1 matrix) as [#;#;#]:');      % 3by1 matrix
    P3_bar = input('P3_bar (3by1 matrix) as [#;#;#]:');      % 3by1 matrix
    P4_bar = input('P4_bar (3by1 matrix) as [#;#;#]:');      % 3by1 matrix
        Pb4 = [P0_bar P1_bar P2_bar P3_bar P4_bar];
end
if strcmp(typeif,'2')
    fprintf('\nType in a 3x5 matrix.\n')
    Pb4 = input('Ph :');
        P0_bar = Pb4(:,1);       % 3by1 matrix
        P1_bar = Pb4(:,2);       % 3by1 matrix
        P2_bar = Pb4(:,3);    % 3by1 matrix
        P3_bar = Pb4(:,4);    % 3by1 matrix
        P4_bar = Pb4(:,5);    % 3by1 matrix
end

%% Output
P0_bar = Pb4(:,1);
P1_bar = Pb4(:,2);
P2_bar = Pb4(:,3);
P3_bar = Pb4(:,4);
P4_bar = Pb4(:,5);
Pb4 = [P0_bar P1_bar P2_bar P3_bar P4_bar];
% Bezier Curve Define in term of u
```

```
syms u
Mb4 = [1 -4 6 -4 1;...
      -4 12 -12 4 0;...
      6 -12 6 0 0;...
      -4 4 0 0 0;...
      1 0 0 0 0];
U = [u^4; u^3; u^2; u; 1];
Bh = Mb4*U;              % Berzier
pu = Pb4*Bh;             % P(u) function

% Constant K (Curvature) equation in term of u
pt = pu;
pt_t = diff(pt,u);
pt_tt = diff(pt_t,u);
pt_ttt = diff(pt_tt,u);
K = norm(cross(pt_t,pt_tt))/norm(pt_t)^3;
% Constant T (Torsion) equation in term of u
T = dot(pt_t,cross(pt_tt,pt_ttt))/norm(cross(pt_t,pt_tt))^2;
%% Plot
figure
t = 1*10^-15:.01:1; % linspace(0,1,100);

pusub = zeros(3,length(t));
for i = 1:length(t)
pusub(:,i) = subs(pu, u, t(i));
end
x = pusub(1,:);
y = pusub(2,:);
z = pusub(3,:);

plot3(x,y,z);
grid on
hold all
scatter3(P0_bar(1),P0_bar(2),P0_bar(3),'*r');
scatter3(P1_bar(1),P1_bar(2),P1_bar(3),'*g');
scatter3(P2_bar(1),P2_bar(2),P2_bar(3),'*c');
scatter3(P3_bar(1),P3_bar(2),P3_bar(3),'*m');
scatter3(P4_bar(1),P4_bar(2),P4_bar(3),'*y');
text(P0_bar(1)+.05,P0_bar(2)+.05,P0_bar(3)+.05,'P_0');
text(P1_bar(1)+.05,P1_bar(2)+.05,P1_bar(3)+.05,'P_1');
text(P2_bar(1)+.05,P2_bar(2)+.05,P2_bar(3)+.05,'P_2');
text(P3_bar(1)+.05,P3_bar(2)+.05,P3_bar(3)+.05,'P_3');
text(P4_bar(1)+.05,P4_bar(2)+.05,P4_bar(3)+.05,'P_4');
%% Finding max Curvature K and Torsion T
for i = 1:length(t)
    Kvalue(i) = subs(K,u,t(i));
    Tvalue(i) = subs(T,u,t(i));
% Positon for K value
end
Kvalue = abs(Kvalue);
Tvalue = abs(Tvalue);
```

```
[Kmax, Klocation] = max(double(Kvalue));
[Tmax, Tlocation] = max(double(Tvalue));

% Max K and T.
Kmax = Kmax(1);
Klocation_at_u = t(Klocation(1));
Tmax = Tmax(1);
Tlocation_at_u = t(Tlocation(1));

% K and T location
maxKlocation = subs(pu, u, Klocation_at_u);
maxKlocation = double(maxKlocation);
maxTlocation = subs(pu, u, Tlocation_at_u);
maxTlocation = double(maxTlocation);
scatter3(maxKlocation(1,1),maxKlocation(2,1),maxKlocation(3,1),'xc');
scatter3(maxTlocation(1,1),maxTlocation(2,1),maxTlocation(3,1),'xr');

% Plot Line Segment
Pb4_t = Pb4'; % Pb4 Transpose
plot3(Pb4_t(:,1),Pb4_t(:,2),Pb4_t(:,3)); % plot of segment

xlabel('x'); ylabel('y'); zlabel('z')
legend('Bezier Curve','P_0','P_1','P_2','P_3','P_4','Max K','Max T');


fprintf('\nThis program output are P and numerical values upon calling. Then plot the↙
Bezier Curve.\n\n')
fprintf('Pb =\n'); disp(Pb4)
fprintf('Max Curvature:'); disp(Kmax)
fprintf('Max Curvature location u ='); disp(Klocation_at_u)
fprintf('Max Curvature x,y,z position =\n'); disp(maxKlocation)

fprintf('Max Torsion:'); disp(Tmax)
fprintf('Max Torsion location u ='); disp(Tlocation_at_u)
fprintf('Max Torsion x,y,z position =\n'); disp(maxTlocation)

fprintf('Check graph in plot.\n');
fprintf('For Numerical Values of type in desire values base off WorkSpace:\n')
```