```
clear all
close all
```

Trajectory Generation

Generate **minimum time** trajectory for a ground robot. The vehicle's dynamics are governed by

$$\dot{p}_x = V\cos(\psi)$$
$$\dot{p}_y = V\sin(\psi)$$
$$\dot{\psi} = \omega$$

Initial and final conditions are

$$p_{init} = [0,0]^\top,\ \psi_{init} = \pi/3,$$

$$p_{fin} = [10,10]^\top,\ \psi_{fin} = \pi/3.$$

The vehicle must obey to speed and angular rate constraints, i.e.,

$$0 \le V \le 5 \qquad |\omega| \le 1.$$

Finally, the vehicle must avoid an obstacle positioned at

$$p_{obs} = [5,5]^\top$$

by maintaining a minimum separation of $d_{sep} = 0.5$ from p_{obs}.

Load Parameters

```
CONSTANTS.N = 4; % Order of approximation
N = CONSTANTS.N;
CONSTANTS.pinit = [0;0];
CONSTANTS.pfin = [10;10];
CONSTANTS.headin = pi/3;
CONSTANTS.headout = pi/3;
CONSTANTS.pobs = [5 5]';
CONSTANTS.sep = 0.5;
CONSTANTS.vmax = 5;
CONSTANTS.omegamax = 1;
```

Initial guess

```
x1 = linspace(CONSTANTS.pinit(1),CONSTANTS.pfin(1),N+1)';
x2 = linspace(CONSTANTS.pinit(2),CONSTANTS.pfin(2),N+1)';
psi = linspace(CONSTANTS.headin,CONSTANTS.headout,N+1)';
V = ones(1,N+1)';
omega = ones(1,N+1)';
```

```
T = 10;
x0 = [x1;x2;psi;V;omega;T];
```

Linear Constraints and UL Bounds

```
A=[]; b=[]; Aeq=[]; beq=[]; lb=[]; ub=[];
```

Optimize

```
options = optimoptions(@fmincon,'Algorithm','sqp','MaxFunctionEvaluations',3000000);
tic
[x,f] =
fmincon(@(x)costfun(x,CONSTANTS),x0,A,b,Aeq,beq,lb,ub,@(x)nonlcon(x,CONSTANTS),optio
ns);
toc
```

Plot

```
N = CONSTANTS.N;

%% Grab LGL Trajectories

x1 = x(1:N+1);
x2 = x(N+2:2*N+2);
psi = x(2*N+3:3*N+3);
V = x(3*N+4:4*N+4);
omega= x(4*N+5:5*N+5);
T = x(end);

[tnodes,w,Diff] = BeBOT(CONSTANTS.N,T);


%% Plot
t = 0:0.01:T;
figure
plot(x1,x2,'o'); hold on
plot(BernsteinPoly(x1',t),BernsteinPoly(x2',t));
grid on
pos = [CONSTANTS.pobs(1)-CONSTANTS.sep CONSTANTS.pobs(2)-CONSTANTS.sep
2*CONSTANTS.sep 2*CONSTANTS.sep];
%pos = [1 2 4 4];
rectangle('Position',pos,'Curvature',[1 1])
axis equal
```

```
figure
plot(t,BernsteinPoly(V',t)); hold on
plot(tnodes,V,'o');
axis([0 T 0 6]);
grid on
figure
plot(t,BernsteinPoly(omega',t)); hold on
plot(tnodes,omega,'o');
grid on
```

Cost Function

```
function J = costfun(x,CONSTANTS)
%COSTFUN Summary of this function goes here
N = CONSTANTS.N;
x1 = x(1:N+1);
x2 = x(N+2:2*N+2);
psi = x(2*N+3:3*N+3);
V = x(3*N+4:4*N+4);
omega= x(4*N+5:5*N+5);
T = x(end);

J = T;
end
```

Nonlinear Constraints

```
function [c,ceq] = nonlcon(x,CONSTANTS)
%NONLCON Summary of this function goes here
%   Detailed explanation goes here
N = CONSTANTS.N;
x1 = x(1:N+1);
x2 = x(N+2:2*N+2);
psi = x(2*N+3:3*N+3);
V = x(3*N+4:4*N+4);
omega= x(4*N+5:5*N+5);
T = x(end);

[~,~,Diff] = BeBOT(N,T);

dyn1 = x1'*Diff - (V.*cos(psi))';
dyn2 = x2'*Diff - (V.*sin(psi))';
dyn3 = psi'*Diff - omega';
```

```
%dist2obs = sqrt((x1-CONSTANTS.pobs(1)).^2 + (x2-CONSTANTS.pobs(2)).^2);

dist2obs_square = BernsteinProduct(x1-CONSTANTS.pobs(1),x1-CONSTANTS.pobs(1)) +
BernsteinProduct(x2-CONSTANTS.pobs(2),x2-CONSTANTS.pobs(2));
%dist2obs_square = ((x1-CONSTANTS.pobs(1)).^2 + (x2-CONSTANTS.pobs(2)).^2)';
E = DegElevMatrix(length(dist2obs_square)-1,length(dist2obs_square)+1);
dist2obs_square_elev = dist2obs_square*E;

%%
N = CONSTANTS.N;

%% Grab LGL Trajectories

x1 = x(1:N+1);
x2 = x(N+2:2*N+2);
psi = x(2*N+3:3*N+3);
V = x(3*N+4:4*N+4);
omega= x(4*N+5:5*N+5);
T = x(end);

[tnodes,w,Diff] = BeBOT(CONSTANTS.N,T);


%% Plot
t = 0:0.01:T;
figure
plot(x1,x2,'o'); hold on
plot(BernsteinPoly(x1',t),BernsteinPoly(x2',t));
grid on
pos = [CONSTANTS.pobs(1)-CONSTANTS.sep CONSTANTS.pobs(2)-CONSTANTS.sep
2*CONSTANTS.sep 2*CONSTANTS.sep];
%pos = [1 2 4 4];
rectangle('Position',pos,'Curvature',[1 1])
axis equal
%%


%c=[V-CONSTANTS.vmax;omega-CONSTANTS.omegamax;-omega-CONSTANTS.omegamax; -dist2obs
+ CONSTANTS.sep;];
c=[V-CONSTANTS.vmax;omega-CONSTANTS.omegamax;-omega-CONSTANTS.omegamax;
-dist2obs_square_elev' + CONSTANTS.sep^2];
ceq=[x1(1)-CONSTANTS.pinit(1);x2(1)-CONSTANTS.pinit(2);x1(end)-
CONSTANTS.pfin(1);x2(end)-CONSTANTS.pfin(2);psi(1)-CONSTANTS.headin;psi(end)-
CONSTANTS.headout;dyn1'; dyn2'; dyn3'];

end
```