```
%Function for evaluating the costs. These being mast shadow constraints,
%slope constraints and distance. In this function they are NOT treated as
%boundary conditions itself. Another version of this similar function
%evaluates them as boundary constraints by numerically increasing the 'Yes'
%condition.
function X = costCalc( openNode, currentNode, targetNode, coor_map, map, azimuth)
```

```
%Using haversine function for calculating
%distance between 2 points on a sphere.
```

```
%defining boundary condition variables.
slopeSafety=tand(18);
frictionSafety=30;
%0 - 1 is being scaled for 0 - 360. Just calculate the allowable arc length put the
%correspong scale between 0 - 1. Currently taken as 180 degree.
shadowSafety=0.5;
height = 400; %enter the height of the vehicle.
width  = 500; %enter the lowest width of vehicle.
length =  map(openNode(:,1), openNode(:,2)) - map(currentNode(:,1),currentNode(:,2))  ; %in m.
if length ~=0 %to prevent logical errors due to finer interpolation.
    dist_hav = haversine( coor_map(openNode(:,1),1), coor_map(currentNode(:,1),1), coor_map(openNode(:,2),2), coor_map(currentNode(:,2),2) ); %in m
    X(1) = abs(length/dist_hav);
else
    X(1) = 0;
end
%Evaluating boundary condition for slopes.
if X(1)>slopeSafety || atand(X(1))>frictionSafety
    X(1)=exp(10);
else
    X(1)=0;
end
```

```
Error using costCalc (line 20)
Not enough input arguments.
```

```
%Evaluating boundary condition for shadow function.
ratio = [(openNode(:,1)-currentNode(:,1)), (openNode(:,2)-currentNode(:,2)) ];
angle = atan2d(ratio(:,2), ratio(:,1));
if angle<0
    angle = angle+360;
end
phase_shift = angle-90;
if phase_shift<0
    phase_shift = phase_shift + 360;
end
%Restricted azi condition.
if 0<=azimuth && azimuth<180 && (180+azimuth)<phase_shift && phase_shift<360
    phase_shift = phase_shift-360;
elseif 180<=azimuth && azimuth<360 && 0 < phase_shift && phase_shift < azimuth-180
    phase_shift = phase_shift+360;
end
if (azimuth-180) <= phase_shift && phase_shift < azimuth
    X(2) = abs((phase_shift-azimuth+180)/180);
    if X(2)>shadowSafety
        X(2)=exp(10);
    else
        X(2)=0;
    end
elseif azimuth <= phase_shift && phase_shift <= azimuth+180
    X(2) = abs((phase_shift-azimuth-180)/(-180));
    if X(2)>shadowSafety
        X(2)=exp(10);
    else
        X(2)=0;
    end
end
```

```
%Evaluating distance constraints.
disxy = [ abs(targetNode(:,1)-openNode(:,1)), abs(targetNode(:,2)-openNode(:,2)) ];
X(3) = 1.414213562373095*min(disxy(:,1),disxy(:,2)) + abs(disxy(:,1)-disxy(:,2));
```

```
end
```

Published with MATLAB® R2015a