calculateRepulsion(double [] allLinksDeltaTheta)

{

for(i=0; i<numObstacles; i++)

{

for(j=0;j<numLinks() j++)

{

//1. Calculate the Jacobian for each link

JacoX = 0;

JacoY = 0;

for(k=0; k<=j; k++)

{

JacoX += -getLinkEndY(j)+getLinkStartY(k);

JacoY += getLinkEndX(j) –getLinkStartX(k);

}

//2. Get the values from the FK

FKx = getLinkEndX(j);

FKy = getLinkEndY(j);

//3. Find the closest point to the obstacle

Point closePointToObs = closestPointOnObs(i,FKx, FKy);

//4. Calculate the repulsion potential

URepX = closePointToObs.x – FKx;

URepY = closePointToObs.y – Fky;

//5. Multiply the repulsive potential with the Jacobian to get the value in configuration //space

allLinksDeltaTheta[j] += (URepX \* JacoX) + (URepY\*JacoY);

}

}

At the end of all of the calculations (both attractive and repulsive), I update the the allLinksDeltaTheta to

(allLinksDeltaTheta/abs(allLinksDeltaTheta)) \* thetaIncr; //to get a positive or negative incrementer.