Repulsive Pseudo code: Rigid body:

$double\ totalForce = \begin{bmatrix}0\\0\end{bmatrix}$

double totalTheta=0;

$U_{rep_j} = \begin{bmatrix}0\\0\end{bmatrix}$

$U_{att_j} = \begin{bmatrix}0\\0\end{bmatrix}$

for(int j=0; j<numPoints;j++)

{

$FK_j = \begin{bmatrix}x_j cos\theta - y_j sin\theta + x\\ x_j sin\theta + y_j cos\theta + x\end{bmatrix}$

$U_{att_j} = \begin{bmatrix}FK_{jx} - g_x\\ FK_{jy} - g_y\end{bmatrix}$

$J_j(q) = \begin{bmatrix}1 & 0 & -x_j sin\theta - y_j cos\theta\\ 0 & 1 & x_j cos\theta - y_j sin\theta\end{bmatrix}$

$totaForce_x$+=$(FK_{jx} - g_x)$

$totalForce_y$+= $(FK_{jy} - g_y)$

totalTheta +=$((FK_{jx} - g_x)(-x_j sin\theta - y_j cos\theta) + (FK_{jy} - g_y)(x_j cos\theta - y_j sin\theta))$

for(int i=0;i<numObs;i++)

{

U_{rep_j}+= $\begin{bmatrix}obs_{ix} - FK_{jx}\\ obs_{iy} - FK_{jy}\end{bmatrix}$

}

$totaForce_x$+= U_{rep_jx}

$totalForce_y$+= U_{rep_jy}

totalTheta +=$(U_{rep\_jx}(-x_j sin\theta - y_j cos\theta) + U_{rep\_jy}(x_j cos\theta - y_j sin\theta))$

```
}
```

Repulsive Pseudo code: Manipulator

```
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.