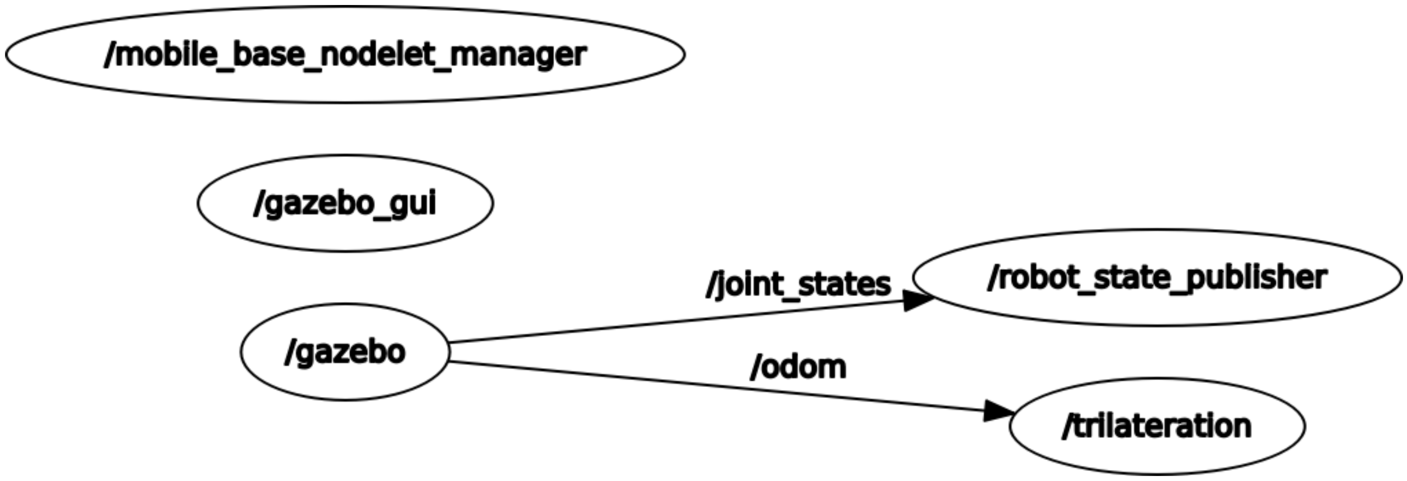
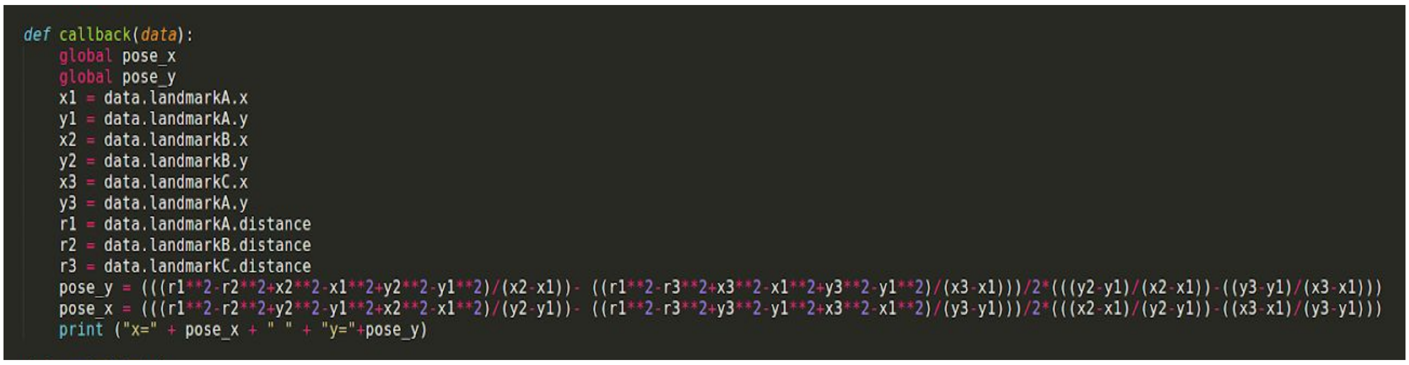
This is the basic rqt graph which we obtain while running from week3



We calculated on rough book and later on implemented same on code. You can see our code which is there in week3_190100051_190110043.



```
def callback(data):
    global pose_x
    global pose_y
    x1 = data.landmarkA.x
    y1 = data.landmarkA.y
    x2 = data.landmarkB.x
    y2 = data.landmarkB.y
    x3 = data.landmarkC.x
    y3 = data.landmarkA.y
    r1 = data.landmarkA.distance
    r2 = data.landmarkB.distance
    r3 = data.landmarkC.distance
    pose_y = (((r1**2-r2**2+x2**2-x1**2+y2**2-y1**2)/(x2-x1))- ((r1**2-r3**2+x3**2-x1**2+y3**2-y1**2)/(x3-x1)))/2*(((y2-y1)/(x2-x1))-((y3-y1)/(x3-x1)))
    pose_x = (((r1**2-r2**2+y2**2-y1**2+x2**2-x1**2)/(y2-y1))- ((r1**2-r3**2+y3**2-y1**2+x3**2-x1**2)/(y3-y1)))/2*(((x2-x1)/(y2-y1))-((x3-x1)/(y3-y1)))
    print ("x=" + pose_x + " " + "y="+pose_y)
```

We almost used the similar way of plotting waypoint here too.
We almost followed the similar control law which we did in lab2 just our pose was being calculated by yourself rather than the odom fn
We were facing an issue due to which our node was not getting initialized.