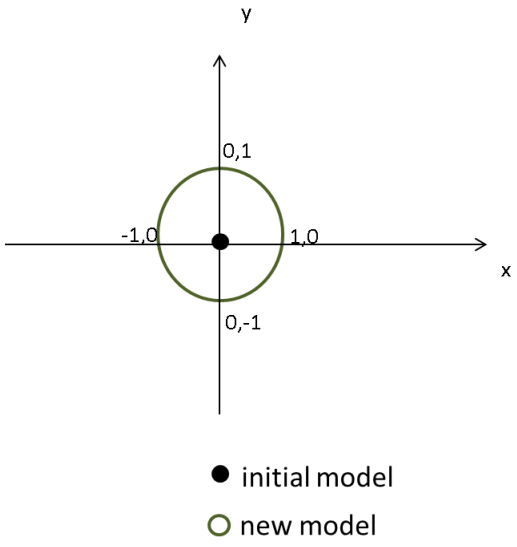
Ch3

4.

(a)

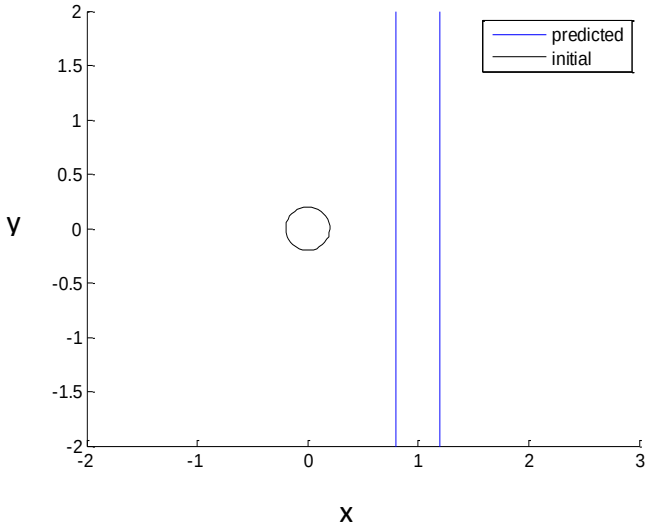


(b)

```
syms x y t d                              % parameters (x,y,theta), move d units
f = [x+d*cos(t); y+d*sin(t); t];    % transition function
v = [x, y, t];
% Derive Jacobian
G = jacobian(f, v)
G =
[              1,              0, -d*sin(t)]
[              0,              1,    d*cos(t)]
[              0,              0,              1]

Sig = [0.01, 0       , 0;
           0     , 0.01 , 0;
           0     , 0       , 10000];
x = 0; y = 0; d = 1; t = 0; R = 0;
G = [1, 0, -d*sin(t);
        0, 1,    d*cos(t);
        0, 0,              1];
% Calculate  Σ̄1
Sigb = G*Sig*G' + R
Sigb =
   1.0e+004 *
```

0.0000	0	0
0	1.0000	1.0000
0	1.0000	1.0000

(c)

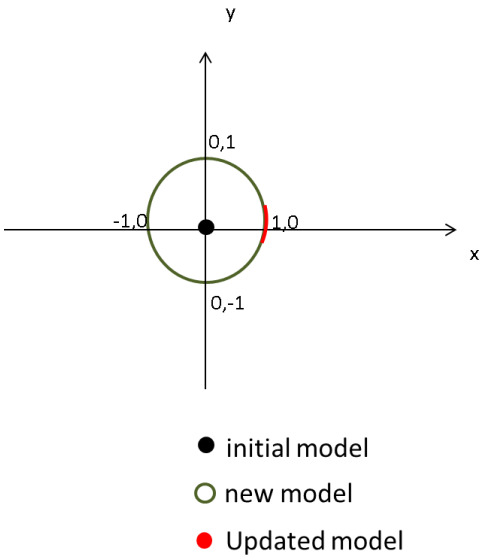


(d)

Now incorporate a measurement. Our measurement shall be a noisy projection of the x-coordinate of the robot, with covariance $Q = 0.01$. Specify the measurement model. Now apply the measurement both to your intuitive posterior, and formally to the EKF estimate using the standard EKF machinery. Give the exact result of the EKF, and compare it with the result of your intuitive analysis.

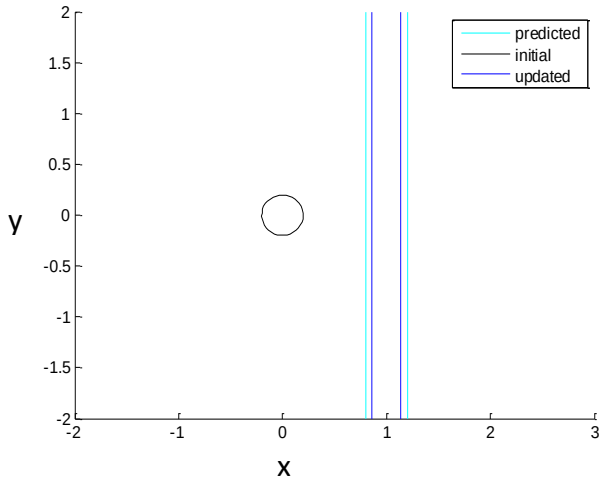
The measurement model $h(\bar{\mu}) = \bar{\mu}_x$

By intuition:



Using EKF:

```
H=[1, 0, 0];
K = Sigb*H' * inv(H*Sigb*H'+Q);
z = 1;
% Calculate μ
u = ub + K*(z-ub(1));
u =
     1
     0
     0
% Calculate Σ
Sig = (eye(3,3)-K*H)*Sigb;
Sig =
  1.0e+004 *
    0.0000         0         0
         0    1.0000    1.0000
         0    1.0000    1.0000
```



(e)

Discuss the difference between your estimate of the posterior, and the Gaussian produced by the EKF. How significant are those differences? What can be changed to make the approximation more accurate? What would have happened if the initial orientation had been known, but not the robot's y-coordinate?

My posterior seems to converge after the measurement is used for update, while the EKF still have a great uncertainty along y axis. At the prediction step of EKF, the Gaussian estimate didn't produce a correct presentation for the uncertainty since the possible robot pose should be a circle centered at the initial point. Due to linearization on *g* the EKF prediction have a great linearization error on the nonlinear function on θ.

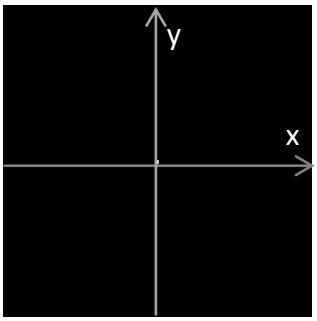
If the orientation had been known, but not the robot's y-coordinate, from my point of view, the EKF prediction will be more accurate, since y is a linear variable for *g*. Therefore the linearization error is smaller. However, the update won't have much effect because the measurement is the projection of x coordinate of robot pose. This won't help to lower the uncertainty on y axis.

Ch4

2.

(a)

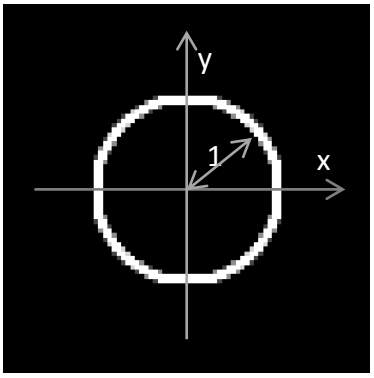
Propose a suitable initial estimate for a histogram filter, which reflects the state of knowledge in the Gaussian prior.



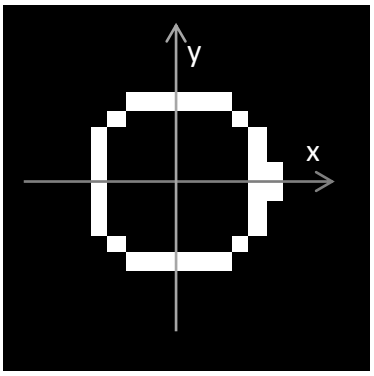

```
% a uniform distribution of probability at (x=0,y=0,theta)

for ti=t_axis

    Px(ind(0,0,ti)) = 1/t_len;

end
```

(b)

Implement a histogram filter and run its prediction step. Compare the resulting posterior with the one from the EKF and from your intuitive analysis. What can you learn about the resolution of the x-y coordinates and the orientation θ in your histogram filter?



(x, y resolution=0.05, θ =pi/180)



(x, y resolution=0.2, θ =pi/45)

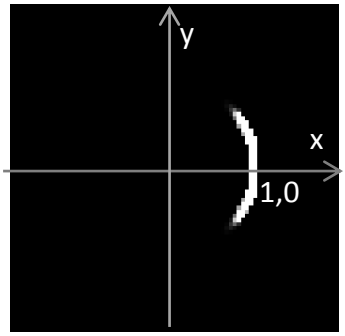
```
    % histogram filter

    eta = 0;

    for xi=-1:x_res:1

        for yi=-1:y_res:1

            for ti=t_axis

                % if succ

                %[xi yi ti]

                try

                    Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) = Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) +
Px(ind(xi,yi,ti));

                    eta = eta + Px(ind(xi,yi,ti));

                catch

                end

            end
```

```
    end
end
Pxb = Pxb / eta;
```

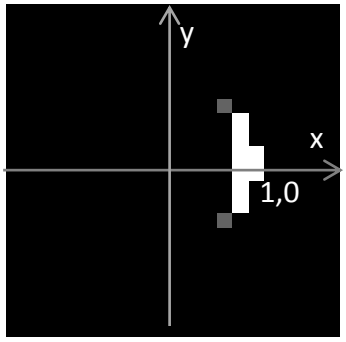
The distribution produced is more like the intuitive result than EKF. The result reflects more like real distribution. If we use raise the resolution, then the prediction result will reflect more precise distribution after the transition but the computation speed will get slower. If the resolutions of each axis double, the overall runtime will be 8 times.

(c)

Now incorporate a measurement into your estimate. As before, the measurement shall be a noisy projection of the x-coordinate of the robot, with covariance $Q = 0.01$. Implement the step, compute the result, plot it, and compare it with the result of the EKF and your intuitive drawing.



(x, y resolution=0.05, θ =pi/180)



(x, y resolution=0.2, θ =pi/45)

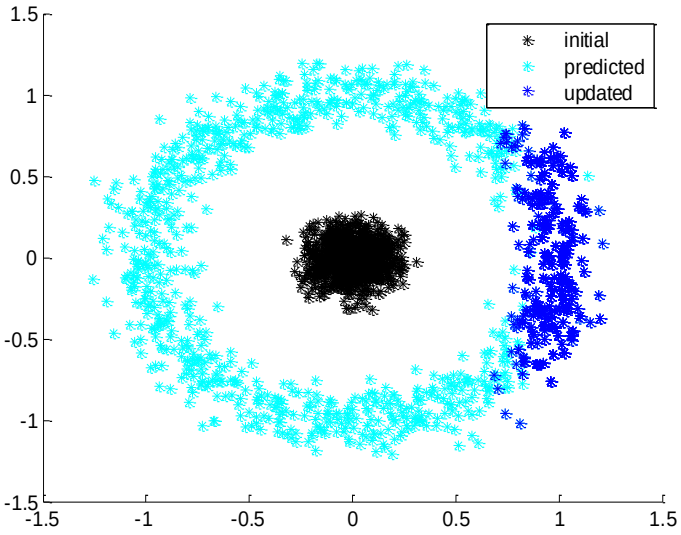
```
eta = 0;
for xi=-1:x_res:1
    for yi=-1:y_res:1
        for ti=t_axis
            % if succ
            %[xi yi ti]
            try
                pp = Px(ind(xi,yi,ti))*normpdf(xi+cos(ti),1,0.1);
                %normpdf(xi+cos(ti),3,0.1)
                Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) = Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) + pp;
                eta = eta + Px(ind(xi,yi,ti)) + pp;
            catch
```

```
            end
            % end
        end
    end
end
Pxb = Pxb / eta;
```

The updated distribution produced is more like the intuitive result than EKF.

5.

Implement Exercise 2 using particle filters instead of histograms, and plot and discuss the results. Investigate the effect of varying numbers of particles on the result.

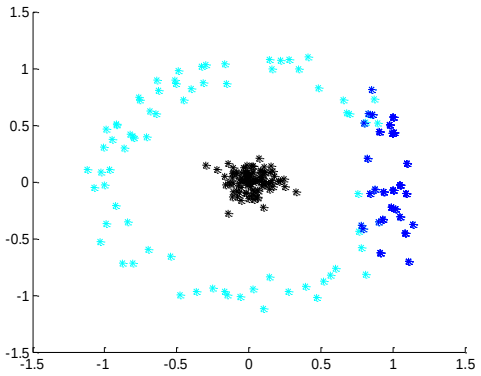


Number of particle = 1000

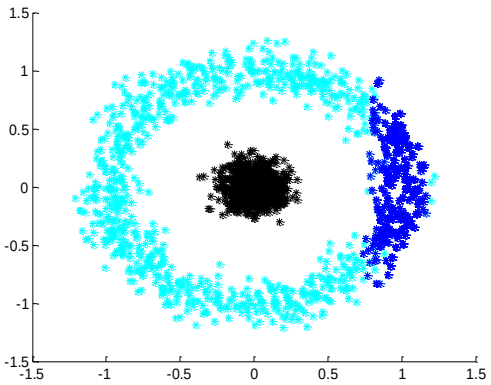
```
function ch4_2_c
M = 1000;
oldChi = [0.1 .* randn(M,1) 0.1 .* randn(M,1) 100 .* randn(M,1)];
Chib = zeros(0,4);
Chi = zeros(0,4);
wsum = 0;
for m = 1:M
   % sample x = p(x|u,x);
   x = oldChi(m,1);
   y = oldChi(m,2);
   t = oldChi(m,3);
```

```
   xt = [x+cos(t) , y+sin(t), t];
   wt = normpdf(1,x+cos(t),0.1);
   wsum = wsum + wt;
   Chib = [Chib; xt, wsum ];
end
for m=1:M
   i = bsearch(Chib, rand(1,1)*wsum);
   Chi = [Chi; Chib(i,:)];
end
hold on
plot(oldChi(:,1),oldChi(:,2),'*','color','k');
plot(Chib(:,1),Chib(:,2),'*','color','c');
plot(Chi(:,1),Chi(:,2),'*','color','b');
end
```

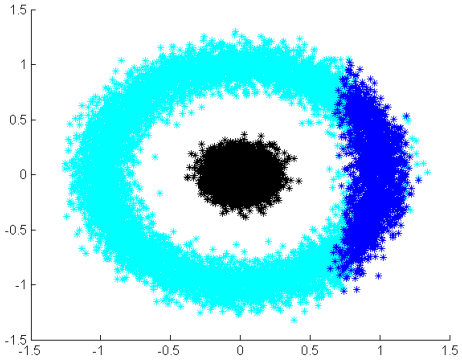
The result is much similar to the intuitive result unlike the EKF prediction and update. The runtime is much faster than using histogram because histogram is a dense calculation on grids but particle filter only predict and update on the particles.



Number of particle = 100



Number of particle = 1000



Number of particle = 10000

To compare the effect of different particle numbers, we generate the above figures. From the above figures, we can conclude that the more particles we apply the more accurate model for to describe the posterior distribution. The time complexity is $O(M \log M)$, where $\log M$ is for binary search.