AML_lab3

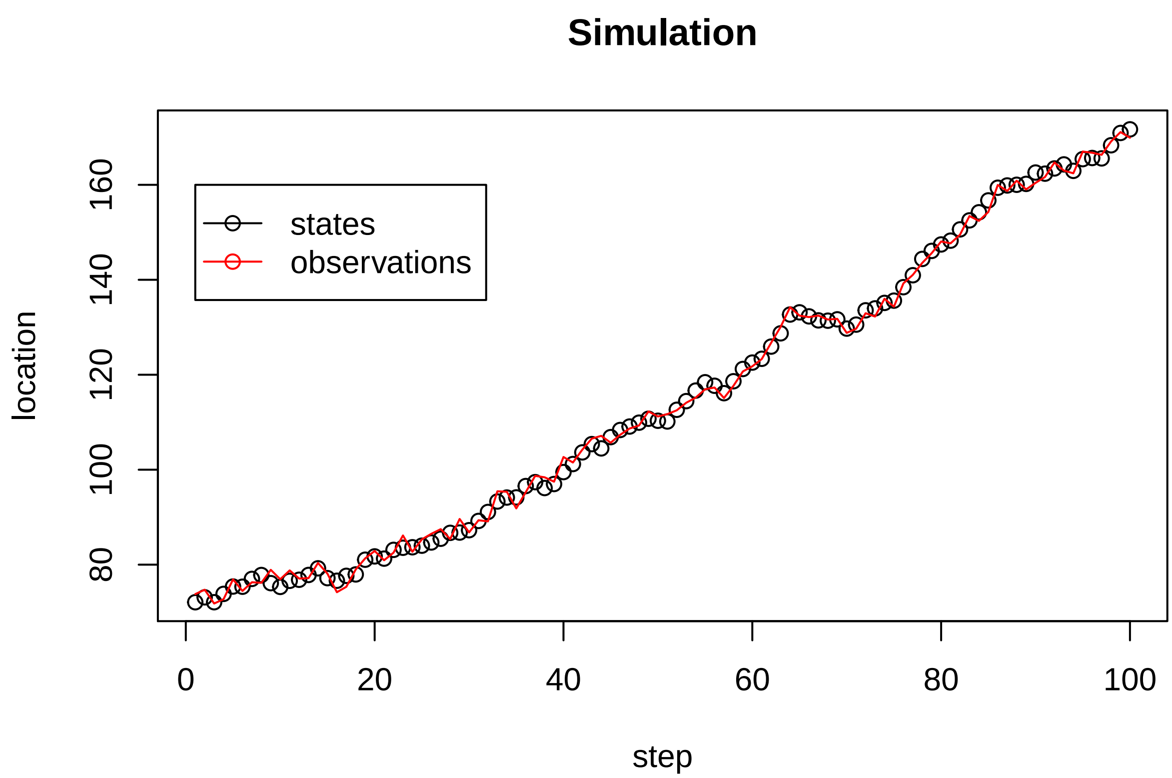
Zijie Feng

2019/10/2

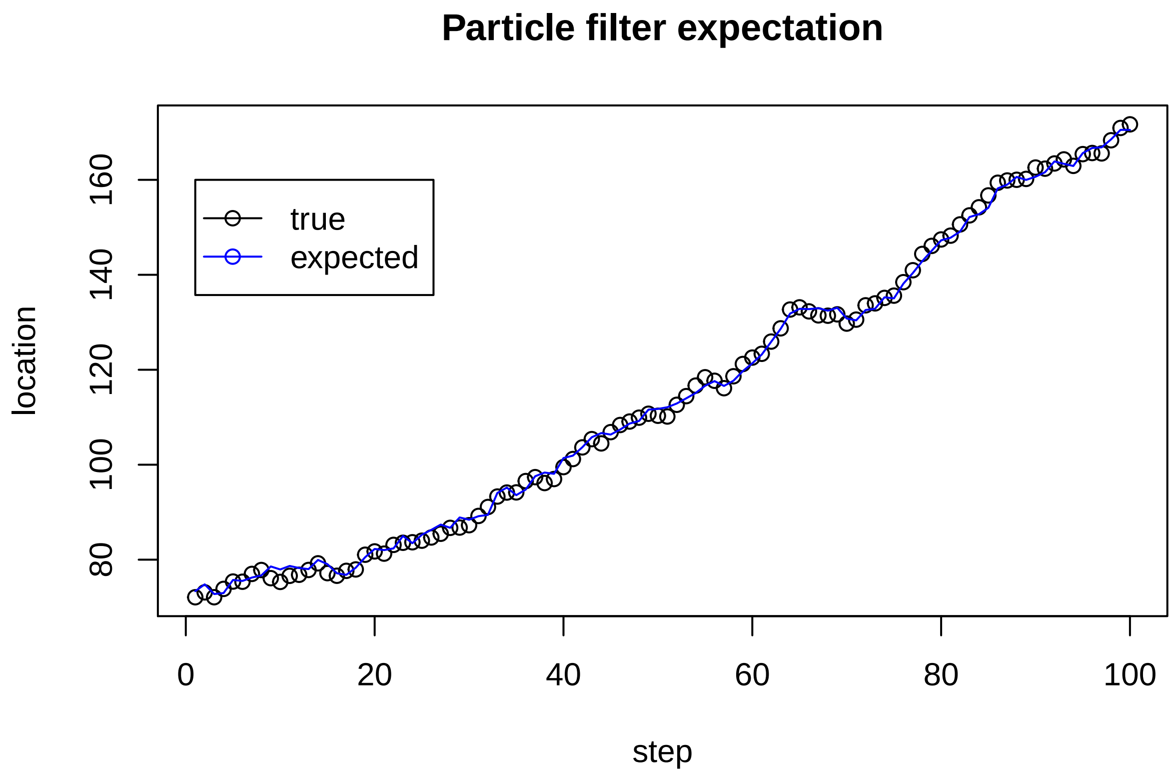
(a)

Implement the SSM above. Simulate it for $T = 100$ time steps to obtain $z_{1:100}$(i.e., states) and $x_{1:100}$(i.e., observations). Use the observations (i.e., sensor readings) to identify the state (i.e., robot location) via particle filtering. Use 100 particles. Show the particles, the expected location and the true location for the first and last time steps, as well as for two intermediate time steps of your choice.

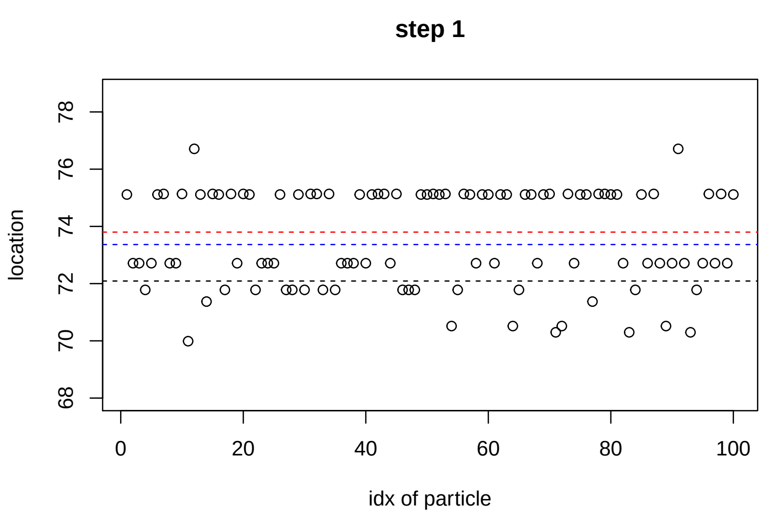
Firstly, we simulate 100 iterations for robot movement. The circle represents the actual location/state of robot in step t, and the red solid line represents the our observations for the robot location.



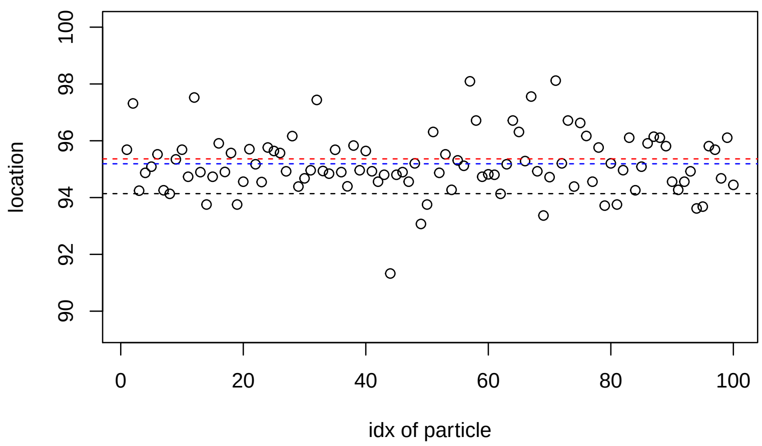
Then, we use such observations to estimate actual states via 100-particle filter. The following blue solid line represents the expected locations for robot. It seems that the blue line is quite similar to the red line in the previous plot. The only difference is that start position of blue line is very low (≈ 50), since the initial particles are sampled from `runif(0,100)`.



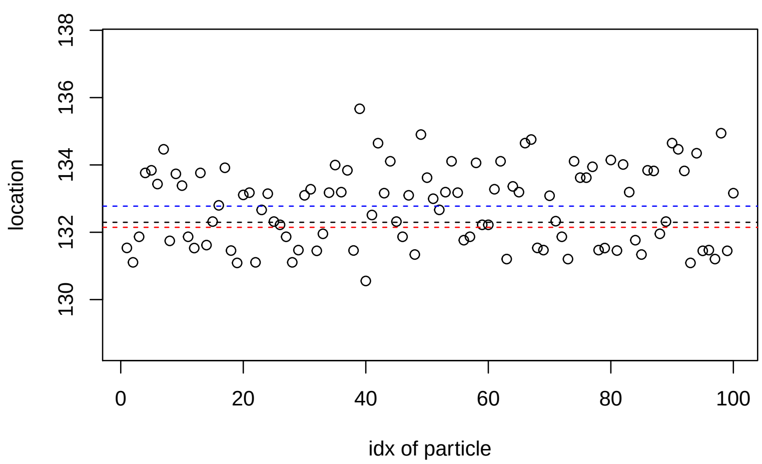
To be more specific, we put the locations of the mean-particles, observations and actual states in the same plot in a specific time. The black line represents actual state. The red and blue lines represent the observed and mean-particle locations, respectively. It is obvious that our sampled particles in each step distribute around the red line, which confirms that observations affect the expectation of particle filter.



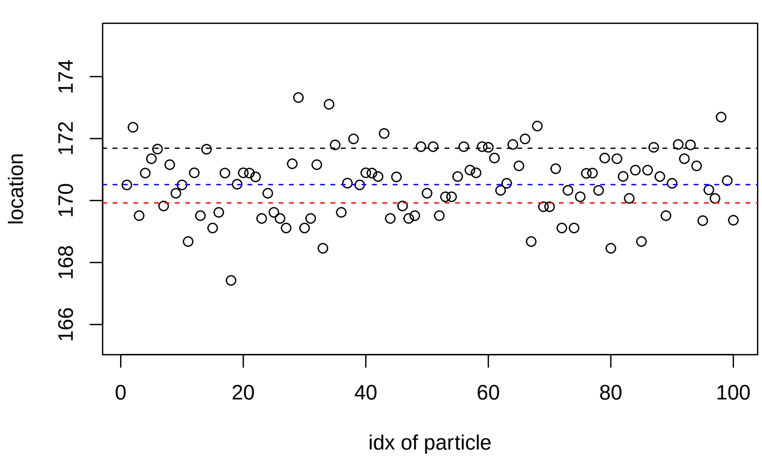
step 34



step 66



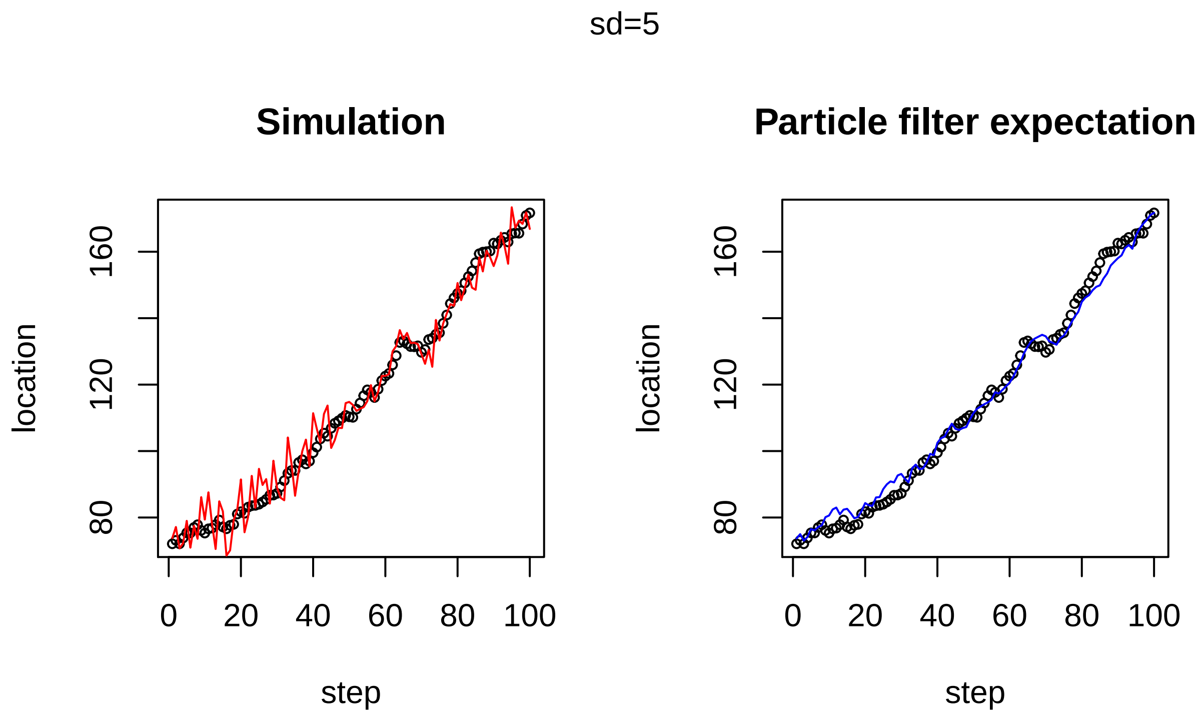
step 100



(B)

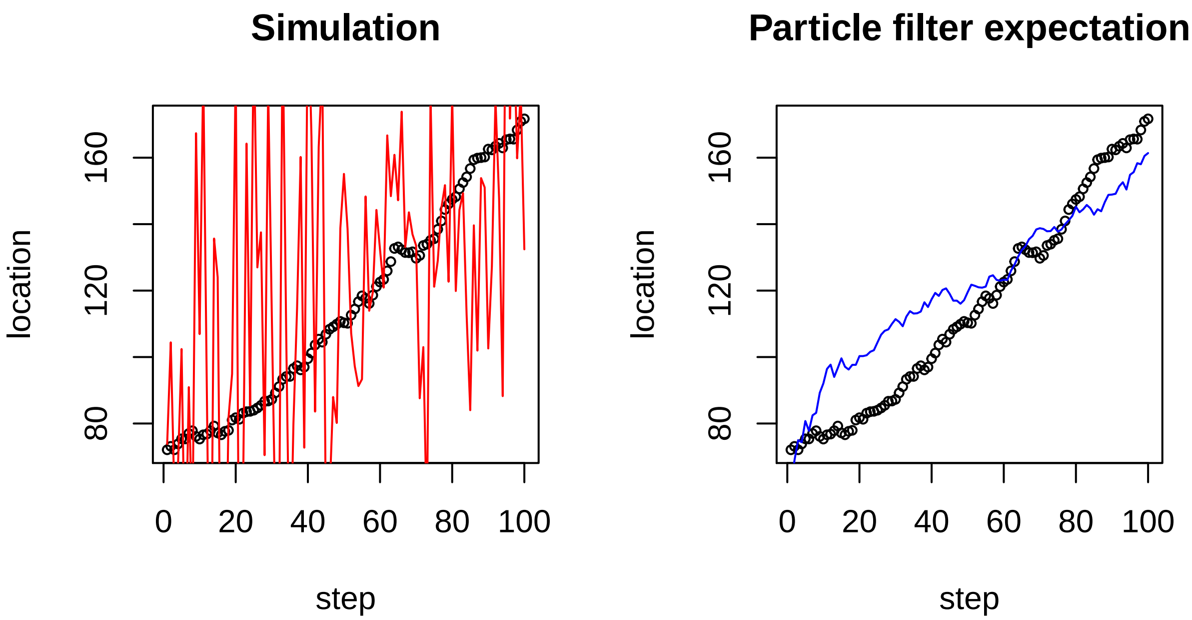
Repeat the exercise above replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results.

We re-simulate another observations for robot location with standard deviation $\Sigma = 5$, both observations and expectation by particle filter seem much rougher than in question (A). The line from particle filter is smoother and closer to original states than the one from observations. Thus, we can find the trend easily via particle filter although the variance of observations is large.



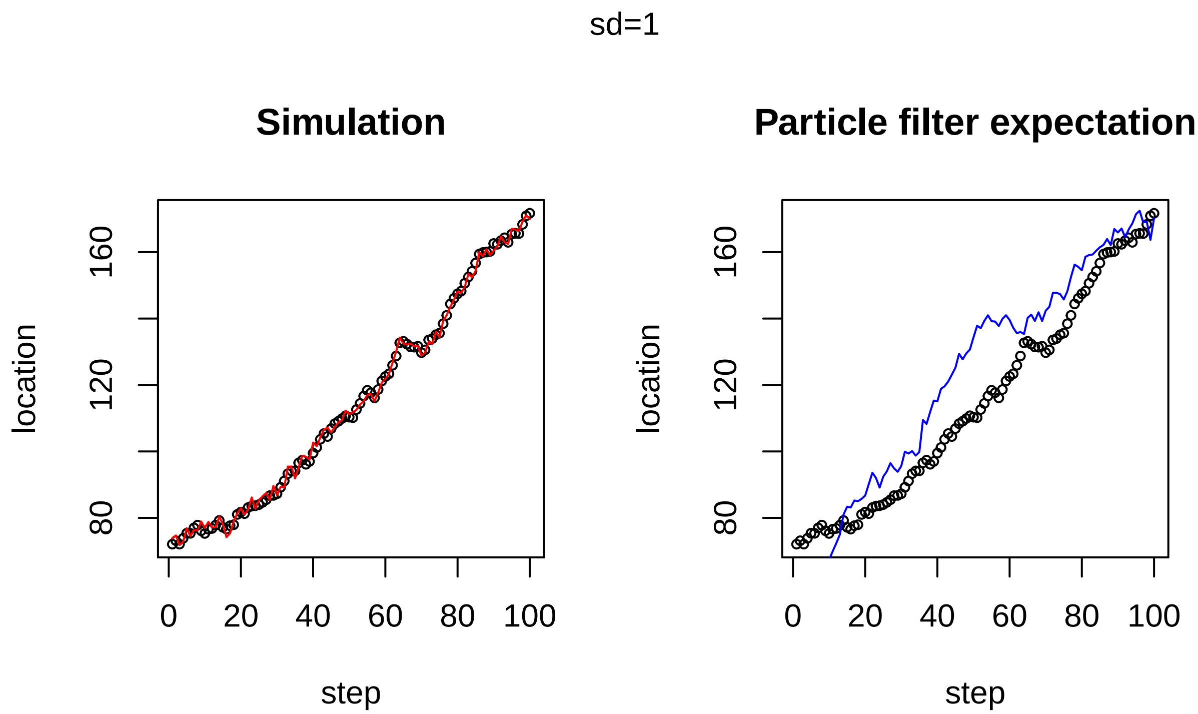
Evidently, the trend of observations is rougher when its standard derivation increasing. It is reasonable because the observations have larger noises compared with real states. In addition, we have the same results as before, and the utility of particle filter is more obvious.

sd=50


Simulation
Particle filter expectation
location
80
120
160
0
20
40
60
80
100
step

(C)

Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.



If the weights for all particles are 1, which means all particles have the same affects to the estimations of states and the estimation only depends on the defined transition and emission probabilities. Then the expected states would be guided to wrong direction, especially for intermediate time steps.

Appendix

```
########### lab3 ######################
knitr::opts_chunk$set(echo = FALSE)
library(MARSS)
########### lab3q1 #####################
rm(list=ls())

# transition A: from state to state
trans <- function(x, u){
  if(u<1/3){
    return(rnorm(1, x))
  }else if(u>2/3){
    return(rnorm(1, x+1))
  }else{
    return(rnorm(1, x+2))
  }
}

# emission C: from state to obs
emiss <- function(x, u, sd=1){
  if(u<1/3){
    return(rnorm(1, x, sd))
  }else if(u>2/3){
    return(rnorm(1, x-1, sd))
  }else{
    return(rnorm(1, x+1, sd))
  }
}

simSSM <- function(iter, sd=1){
  set.seed(12345)
  x <- rep(0,iter)
  z <- x
  # the 0th state follows uniform(0,100)
  z[1] <- runif(1, 0, 100)
  x[1] <- trans(z[1], runif(1))

  for (t in 2:iter){
    z[t] <- trans(z[t-1], runif(1))
    x[t] <- emiss(z[t], runif(1), sd)
  }
  return(list(states=z, observations=x))
}

# input the observations, the number of particles
filtering <- function(x, M, sd=1){
  iter <- length(x)
  W <- matrix(nrow=M, ncol=iter)     # weight

  Z <- matrix(nrow=M,ncol=iter+1)   # particles
  Z[,1] <- runif(M, 0, 100)         # initialize the particiles

  for(t in 1:iter){
```

```r
    zbar <- matrix(ncol=M)          # beliefs/latent particles
    for(m in 1:M){
      zbar[m] <- trans(Z[m,t], runif(1))
      W[m,t] <- (dnorm(x[t], zbar[m], sd)+dnorm(x[t],zbar[m]-1, sd)
              +dnorm(x[t],zbar[m]+1, sd))/3     # use dnorm() calculate weight
    }
    W[,t] <- W[,t]/sum(W[,t])

    # sample new particles according to beliefs
    Z[, t+1] <- sample(zbar, size=M, replace = TRUE, prob = W[,t])
  }
  return(list(Z=Z,W=W))
}

simulation <- simSSM(iter=100)
states <- simulation$states
observations <- simulation$observations

plot(1:100, states, type="p",xlab="step", ylab="location", main="Simulation")
lines(1:100, observations, col="Red")
legend(1,160,legend=c("states","observations"),
       col=c("black","red"), pch=1, lty=1)
f <- filtering(x=observations, M=100)
Z <- f$Z                    # weighted  mean
states_ex <- colSums(Z[,-1]*f$W)
plot(1:100, states, type="p",xlab="step", ylab="location", main="Particle filter expectation")
lines(1:100, y=states_ex, type="l", col="blue")
legend(1,160,legend=c("true","expected"),
       col=c("black","blue"),pch=1, lty=1)
p <- function(idx){
  plot(Z[,idx+1],ylab="location",xlab="idx of particle",
       ylim=c(min(Z[,idx+1])-2,max(Z[,idx+1])+2),
     main=paste0("step ",idx))
abline(h=states[idx], col="black",lty=2)
abline(h=observations[idx], col="red",lty=2)
abline(h=colSums(Z[,idx+1]%*%f$W[,idx]), col="blue",lty=2)
}
p(1)
p(34)
p(66)
p(100)
########### lab3q2 #####################

sd = 5
simb1 <- simSSM(100, sd)
states <- simb1$states
observations <- simb1$observations

f <- filtering(x=observations, M=100, sd)
Z <- f$Z                    # weighted  mean
states_ex <- colSums(Z[,-1]*f$W)

par(mfrow=c(1,2), oma = c(0, 0, 4, 0))
```

```r
plot(1:100, states, type="p",xlab="step", ylab="location", main="Simulation", cex=0.6)
lines(1:100, observations, col="Red")

plot(1:100, states, type="p",xlab="step", ylab="location",
     main="Particle filter expectation",cex=0.6)
lines(1:100,y=states_ex, type="l", col="blue")

mtext("sd=5", side = 3, line = 0, outer = T)
sd = 50
simb1 <- simSSM(100, sd)
states <- simb1$states
observations <- simb1$observations

f <- filtering(x=observations, M=100, sd)
Z <- f$Z                    # weighted  mean
states_ex <- colSums(Z[,-1]*f$W)

par(mfrow=c(1,2), oma = c(0, 0, 4, 0))

plot(1:100, states, type="p",xlab="step", ylab="location", main="Simulation", cex=0.6)
lines(1:100, observations, col="Red")

plot(1:100, states, type="p",xlab="step", ylab="location",
     main="Particle filter expectation",cex=0.6)
lines(1:100,y=states_ex, type="l", col="blue")

mtext("sd=50", side = 3, line = 0, outer = T)
########### lab3q3 ######################

filtering2 <- function(x, M){
  iter <- length(x)
  W <- matrix(nrow=M, ncol=iter)

  Z <- matrix(nrow=M,ncol=iter+1)   #posteriror Z
  Z[,1] <- runif(M, 0, 100)    # initialize the particiles

  for(t in 1:iter){
    zbar <- matrix(ncol=M)
    for(m in 1:M){
      zbar[m] <- trans(Z[m,t], runif(1))   # prior Z
      W[m,t] <- (dnorm(x[t], zbar[m])+dnorm(x[t],zbar[m]-1)
              +dnorm(x[t],zbar[m]+1))/3
    }
    W[,t] <- rep(1,M)
    # print(W[,t])
    Z[, t+1] <- sample(zbar, size=M, replace = TRUE, prob = W[,t])  # update Z by posterior
  }
  return(list(Z=Z,W=W))
}


sd = 1
simb1 <- simSSM(100, sd)
```

```r
states <- simb1$states
observations <- simb1$observations

f <- filtering2(x=observations, M=100)
Z <- f$Z                   # weighted  mean
states_ex <- colMeans(Z[,-1]*f$W)

par(mfrow=c(1,2), oma = c(0, 0, 4, 0))

plot(1:100, states, type="p",xlab="step", ylab="location", main="Simulation", cex=0.6)
lines(1:100, observations, col="Red")

plot(1:100, states, type="p",xlab="step", ylab="location",
     main="Particle filter expectation",cex=0.6)
lines(1:100,y=states_ex, type="l", col="blue")

mtext("sd=1", side = 3, line = 0, outer = T)
```