```
Tsteps = 1000
mut_delta = 0 #how to implement mutations of different sizes?
step = 0.01
int_step = step
sigma2 = 1
mut_prob = 0.01
alpha = 0.5
mix_sigma2 = 0.4

#########
int<-function(v){
    l=length(v)
    int_step/2*(sum(2*v)-v[1]-v[l])
}

mrange = seq(-2,2,by=step)
Nm = length(mrange)
mrange_orig = seq(-1,1,by=step)
frange = seq(-2,2,by=step)
Nf = length(frange)
frange_orig = seq(-1,1,by=step)

m0 = which(mrange==-1)
m1 = which(mrange==1)
f0 = which(frange==-1)
f1 = which(frange==1)

# its = 2
sigma2_vals = c(0.1,0.5,1,1.5,2,4)
Ns = length(sigma2_vals)
mix_sigma2_vals = c(seq(0.1,0.4,by=0.1),1)
Nms = length(mix_sigma2_vals)
Pm_keep = array(0,dim=c(Nm,Ns,Nms))
Pf_keep = array(0,dim=c(Nf,Ns,Nms))

for(k in 1:Ns){
    for(l in 1:Nms){
        sigma2 = sigma2_vals[k]
        mix_sigma2 = mix_sigma2_vals[l]

        Pm = matrix(0,Nm,Tsteps+1)
        # breaks = runif(n=length(mrange_orig)-1)
        # breaks = sort(breaks)
        # m_init = c(breaks[1],diff(c(breaks,1)))
        # m_init = runif(n=length(mrange_orig))
        # m_init = m_init/int(m_init)
        # Pm[m0:m1,1] = m_init
        Pm[m0] = 0.6
        Pm[m1] = 0.4
        Pm[,1] = Pm[,1]/int(Pm[,1])

        Pf = matrix(0,Nf,Tsteps+1)
        # breaks = runif( n=length(mrange_orig)-1)
        # breaks = sort(breaks)
        # f_init = c(breaks[1],diff(c(breaks,1)))
        # Pf[f0:f1,1] = f_init
        # Pf[f0,1] = .4
        # Pf[f1,1] = .6
        # Pf[,1] = Pf[,1]/int(Pf[,1])
        p = .4
        Pf[,1] = p*dnorm(frange,-1,mix_sigma2)+(1-p)*dnorm(frange,1,mix_sigma2)
        # Pf[,1] = .3*dnorm(frange,-1,mix_sigma2)+.3*dnorm(frange,0,mix_sigma2)+.4*dnorm(frange,1,mix_sigma2)

        # t = 1
        for(t in 1:Tsteps){
            Pm_adults = Pm[,t]
            Pf_adults = Pf[,t]
            pxy = matrix(0,Nm,Nf)

            for(j in 1:Nf){
                y = frange[j]
                weight = 1/sqrt(2*pi*sigma2)*exp(-(mrange-y)^2/(2*sigma2))
                # weight = matrix (0,Nf,1)
                # weight[c(f0,x1)] = 1
                # weight[j] = 1+alpha
                z = int(weight*Pm_adults)
                if(z!=0){
                    pxy[,j] = Pf_adults[j]*weight*Pm_adults/z
                    }
            }
            Pm_beforemut = matrix(0,Nm)
            for(i in 1:Nm){
                Pm_beforemut[i] = int(pxy[i,])
            }
            Pm_aftermut = matrix(0,Nm)
            Pm_aftermut = (1-mut_prob)*Pm_beforemut + mut_prob/2*c(Pm_beforemut[2:Nm],0) + mut_prob/2*c(0,Pm_beforemut[1:Nm-1])
            Pm[,t+1] = Pm_aftermut
            Pf[,t+1] = Pf_adults
        }
        Pm_keep[,k,l] = Pm[,Tsteps+1]
        Pf_keep[,k,l] = Pf[,Tsteps+1]
    }
}

######
sigma2 = 0.1
mix_sigma2 = 1
Pm = matrix(0,Nm,Tsteps+1)
        # breaks = runif(n=length(mrange_orig)-1)
        # breaks = sort(breaks)
        # m_init = c(breaks[1],diff(c(breaks,1)))
        # m_init = runif(n=length(mrange_orig))
        # m_init = m_init/int(m_init)
        # Pm[m0:m1,1] = m_init
        Pm[m0] = 0.6
        Pm[m1] = 0.4
        Pm[,1] = Pm[,1]/int(Pm[,1])

        Pf = matrix(0,Nf,Tsteps+1)
        # breaks = runif( n=length(mrange_orig)-1)
        # breaks = sort(breaks)
        # f_init = c(breaks[1],diff(c(breaks,1)))
        # Pf[f0:f1,1] = f_init
        # Pf[f0,1] = .4
        # Pf[f1,1] = .6
        # Pf[,1] = Pf[,1]/int(Pf[,1])
        p = .4
        Pf[,1] = p*dnorm(frange,-1,mix_sigma2)+(1-p)*dnorm(frange,1,mix_sigma2)
        # Pf[,1] = .3*dnorm(frange,-1,mix_sigma2)+.3*dnorm(frange,0,mix_sigma2)+.4*dnorm(frange,1,mix_sigma2)

        # t = 1
        for(t in 1:Tsteps){
            Pm_adults = Pm[,t]
            Pf_adults = Pf[,t]
            pxy = matrix(0,Nm,Nf)

            for(j in 1:Nf){
                y = frange[j]
                weight = 1/sqrt(2*pi*sigma2)*exp(-(mrange-y)^2/(2*sigma2))
                # weight = matrix (0,Nf,1)
                # weight[c(f0,x1)] = 1
                # weight[j] = 1+alpha
                z = int(weight*Pm_adults)
                if(z!=0){
                    pxy[,j] = Pf_adults[j]*weight*Pm_adults/z
                    }
            }
            Pm_beforemut = matrix(0,Nm)
            for(i in 1:Nm){
                Pm_beforemut[i] = int(pxy[i,])
            }
            Pm_aftermut = matrix(0,Nm)
            Pm_aftermut = (1-mut_prob)*Pm_beforemut + mut_prob/2*c(Pm_beforemut[2:Nm],0) + mut_prob/2*c(0,Pm_beforemut[1:Nm-1])
            Pm[,t+1] = Pm_aftermut
            Pf[,t+1] = Pf_adults
        }
```