hMC proposals paths demo

** Note: Adapted from code in Appendix C.4 of Bayesian Data Analysis by Gelman et al **

preliminaries

```
# clear workspace
rm(list=ls())

# set random seed
set.seed(5873)

#libraries
```

define simulation objects

```
# log-likelihood function
log_p_theta = function(theta) {
  theta_1 = theta[1]
  theta_2 = theta[2]
  radius = sqrt(theta_1 ^ 2 + theta_2 ^ 2)
  log_like = dnorm(radius, mu, sigma, log = TRUE)
  return(log_like)
}

# gradient of log-likelihood function
gradient_theta = function(theta) {
  theta_1 = theta[1]
  theta_2 = theta[2]
  radius = sqrt(theta_1 ^ 2 + theta_2 ^ 2)
  d_theta_1 = - (theta_1 * (radius - 1)) / (sigma^2 * radius)
  d_theta_2 = - (theta_2 * (radius - 1))/ (sigma^2 * radius)
  return( c(d_theta_1, d_theta_2) )
}

# hmc_iteration
hmc_iteration = function(theta, epsilon, L, M) {
  path = array(NA, c(L,length(theta))) #stores path within each step of hMC
  M_inv = 1/M
  phi = rnorm(2, 0, sqrt(M))
  theta_old = theta
  #log_p_old = log_p_theta(theta) - 0.5 * sum(M_inv * phi^2)
  log_p_old = log_p_theta(theta) - 0.5 * sum(M_inv * phi^2)
  phi = phi + 0.5 * epsilon * gradient_theta(theta)
  for (t in 1:L) {
    #theta = theta + epsilon * M_inv * phi
```

```
    #phi = phi + (if (t==L) 0.5 else 1) * epsilon * gradient_theta(theta)
    phi = phi + 0.5 * epsilon * gradient_theta(theta)
    theta = theta + epsilon * M_inv * phi
    phi = phi + 0.5 * epsilon * gradient_theta(theta)
    path[t,] = theta
  }
  log_p_star = log_p_theta(theta) - 0.5 * sum(M_inv * phi^2)
  r = exp(log_p_star - log_p_old)
  if (is.nan(r)) r = 0
  p_jump = min(r,1)
  theta_new = if (runif(1) < p_jump) theta else theta_old
  return(list(theta = theta_new, p_jump = p_jump, path = path))
}

# simulate potential paths
hmc_paths = function(start_theta, nPaths, epsilon, L, M) {
  d = length(start_theta)
  paths = array(NA, c(nPaths,L,d)) # stores all transitions within each hMC iteration
  p_jump = array(NA,nPaths) # stores probabily accept next
  accept = array(NA,nPaths) # stores whether proposal is accepted
  for (t in 1:nPaths) {
    temp = hmc_iteration(start_theta, epsilon, L, M)
    paths[t,,] = temp$path
    p_jump[t] = temp$p_jump
    accept[t] = if (runif(1) < temp$p_jump) 1 else 0
  }
  return(list(paths = paths, p_jump = p_jump, accept = accept))
}
```

run and visualize potential transitions

```
sigma = 0.1 # std of "marginal radial" normal distribution
mu = 1       # radius of the mode

current_theta = c(0,1)
nPaths =  10
L = 30
epsilon = 0.01
mass_vector =  c(sigma,sigma)


M1 = hmc_paths(start_theta = current_theta,
               nPaths = nPaths,
               epsilon = epsilon,
               L = L,
               M = mass_vector)

library("plotrix")
par(pty="s")
plot(1, type="n",
     xlab="theta_1", ylab="theta_2",
     xlim=c(-2,2), ylim=c(-2,2), asp=1)
```

```
draw.circle(0,0,mu, border="blue")
points(current_theta[1], current_theta[2],pch = 17,col="black", cex = 2, lty=3)
for (t in 1:nPaths) {
  points(M1$paths[t,,1],M1$paths[t,,2], type="l",
         col=rgb(red=0, green=0, blue=0, alpha=0.25))
  points(M1$paths[M1$accept==1,L,1],M1$paths[M1$accept==1,L,2], pch = 20, cex = 0.75,
         col=rgb(red=0, green=1, blue=0, alpha=0.25))
  points(M1$paths[M1$accept==0,L,1],M1$paths[M1$accept==0,L,2], pch = 20, cex = 0.75,
         col=rgb(red=1, green=0, blue=0, alpha=0.25))
}
```