The Circular Drift-Diffussion Model: A tutorial.

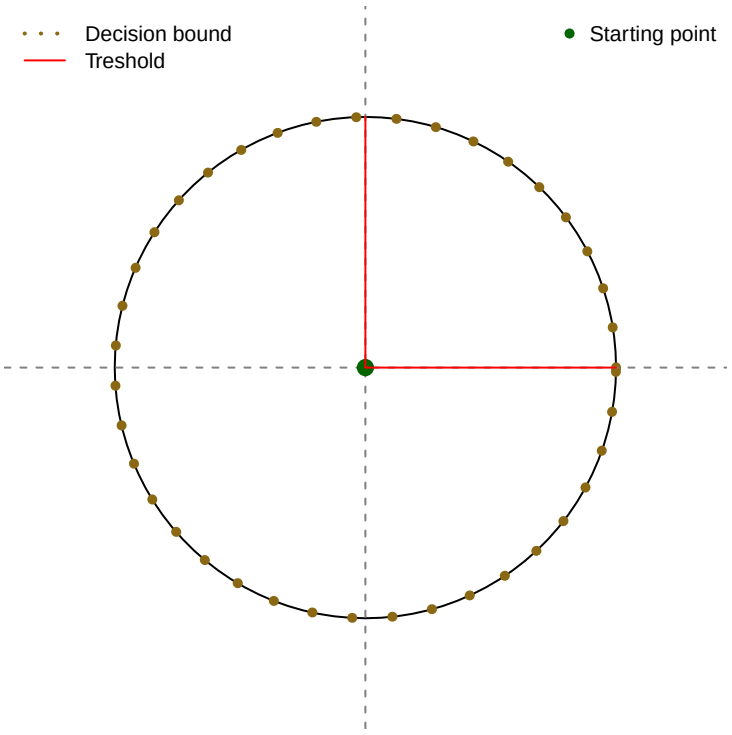
Adriana F. Chávez De la Peña

Summer, 2021

1. General description

The Circular Drift-Diffusion Model (CDDM) works as a two-dimensional extension of the classical Drift-Diffusion Model. The CDDM defines the decision space as a bounded continuum, graphically represented with a circle, which contrasts to the usual binary-choice setting with just two opposinbg boundaries. To capture the cumulative-sampling decision process, the CDDM model uses the radius of the circle to represent the threshold defining the amount of information required to settle on a specific response, depending on where this random-walk falls on the circumference.

Just as the classical DDM, the CDDM assumes that the decision-making system gathers some average amount of information at each point in time, moving away from a starting point. For an unbiased system, we would expect said initial sampling point to fall on the center of the circle.



Footnote about the graph: The `plotrix` R package used to draw the circles rescales the x and y axis (thus forming a circle-looking ellipse). To ensure that the radii are being correctly represented, we include a red mark on the X and Y axis to corroborate proportionality.

2. Single trial emulation of the decision process

To get a better understanding on how the CDDM works, let us emulate a single trial. We'll start with the set up:

```
iter <- 5000        # Maximun number of steps/iterations for gathering information
thresh <- 7.5             # 1) Threshold for making a response (Radius)
s.init <- c(0,0)          # 2) Initial state; c(0,0) indicates no bias
mu1.drift <- 0.3          # 3) Average step size on X
mu2.drift <- 0.4          # 4) Average step size on Y
sigma.driftrate <- 0.6    # 5) Variance in the length of steps given in either X or Y

state <- matrix(nrow=iter,ncol=2) #Empty matrix to be filled with coordinates (states)
state[1,] <- s.init  #Declare initial state
```

The core idea is that in each moment in time, the system gathers information that moves the decision process away from the initial state and towards the circumference of the circle in a fashion similar to that of the standard DDM. Similarly, there's a threshold that works as a stopping rule that determines the response to produce.

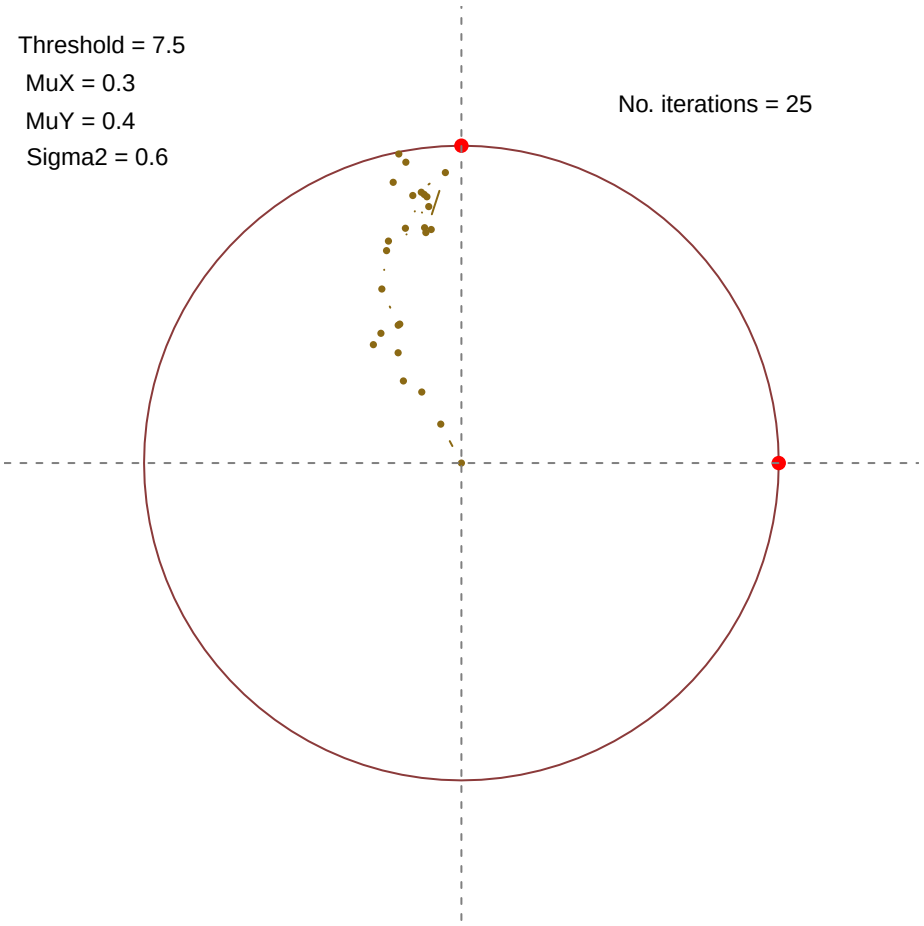
```
#Part 1: Acquiring/Accumulating/Updating information
for(t in 2:iter){    #In each iteration/step...
  d1 <- rnorm(1,mu1.drift,sigma.driftrate)  #A random step-size is sampled from a Normal distribution
  d2 <- rnorm(1,mu2.drift,sigma.driftrate)      #...for each, the X (d1) and Y (d2) axes
  state[t,] <- state[t-1,]+c(d1,d2)   #We update the information state

#Part 2: Evaluate whether we have passed the threshold
  pass <- sqrt(sum(state[t,]^2))  # The distance between current state and origin (0,0) is measured
  if(pass >= thresh){ #If this distance is greater than the threshold (a.k.a. ratio), then..
    print(paste("Finished in", t, "iterations"))  # We identify the number of iterations passed
    break}} #The accumulation process ends
```

```
## [1] "Finished in 25 iterations"
```

```
#Part 3: Make sure the decision process and the line drawn end exactly at threshold
if(pass> thresh){   #If the last state is further away than the threshold (i.e. out of the circle)
  A <- state[t-1,]    #Then we take the second to last state
  B <- state[t,]      #and this final state as reference points
  last.step.x <- seq(A[1],B[1],length.out = 100)  #We define 100 different intermediate points
  last.step.y <- seq(A[2],B[2],length.out = 100)      #.. on each the X and Y axes
  for(i in 1:length(last.step.x)){    # Now, each of these middle points gets examined
    pass2 <- sqrt(last.step.x[i]^2+last.step.y[i]^2)  # in terms of their distance from origin
    if(round(pass2,1) == thresh){  # so that we can stop right at the threshold
      circunf <- c(last.step.x[i],last.step.y[i]) #and locate the exact point on the circumference
      break}} # where the decision process ends
  state[t,] <- circunf} #and we redefine our final state
```

Plotting the single-trial random-walk process

```
library(plotrix) #Library to draw circles
plot(-10:10,-10:10,type="n", ann = FALSE, axes = FALSE) #Create empty plot
draw.circle(0,0,radius = thresh, border = "indianred4") #Draw circle
points(state, type = "b", col = "goldenrod4", pch=16, cex=0.5) #Accumulation process
points(0,thresh, col ="red", pch=16) #Make sure ratio is correct on Y
points(thresh,0, col= "red", pch=16) #..and X axis
abline(h = 0, lty=2, col="gray50") # Draw reference lines at y=0
abline(v = 0, lty=2, col="gray50") #... and x=0
text(-8.5,9.9,paste("Threshold =", thresh), cex = 0.75) #Print values used on our emulation
text(-9,9,paste("MuX =", mu1.drift), cex = 0.75)
text(-9,8.1,paste("MuY =", mu2.drift), cex = 0.75)
text(-8.6,7.2,paste("Sigma2 =", sigma.driftrate), cex = 0.75)
text(6,8.5,paste("No. iterations =", t), cex = 0.75)
```

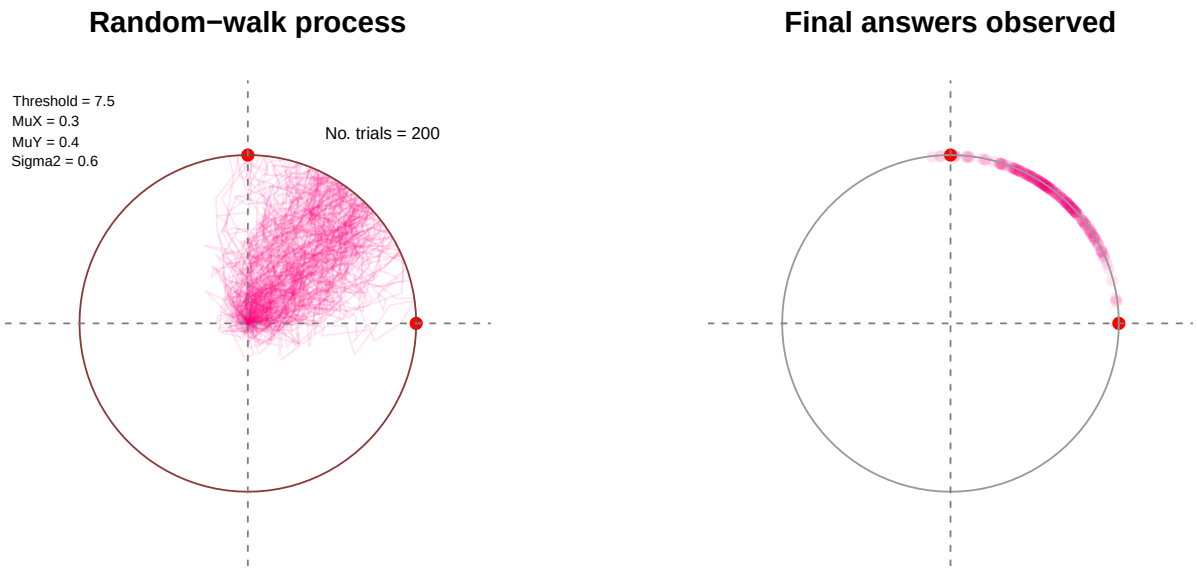


3. Multiple trials

Now, let us observe what would happen if we were to repeat the very same accumulation process 200 times:

```
trials <- 200  #Number of repetitions of the accumulation process
state <- array(NA, dim = c(iter, 2, trials))   #States are saved in a 3dimensional array
finalT <- NA #An additional empty vector to store RT (a.k.a. total number of iterations)

#This part of the code is exactly the same as before, except that:
for(a in 1:trials){  #1) Now we have an additional "for" loop for each trial
state[1,,a] <- s.init   #2) and States are stored in our new 3D array
for(t in 2:iter){
  d1 <- rnorm(1,mu1.drift,sigma.driftrate)
  d2 <- rnorm(1,mu2.drift,sigma.driftrate)
  state[t,,a] <- state[t-1,,a]+c(d1,d2)
  pass <- sqrt(sum(state[t,,a]^2))
  if(pass >= thresh){
    finalT[a] <- t   #We store the number of total iterations required on each trial
    break}}

if(pass> thresh){
  A <- state[t-1,,a];   B <- state[t,,a]
  last.step.x <- seq(A[1],B[1],length.out = 100)
  last.step.y <- seq(A[2],B[2],length.out = 100)
  for(i in 1:length(last.step.x)){
    pass2 <- sqrt(last.step.x[i]^2+last.step.y[i]^2)
    if(round(pass2,1) == thresh){
      circunf <- c(last.step.x[i],last.step.y[i])
      break}}
  state[t,,a] <- circunf}}
```

Plotting the random-walk process over 200 trials

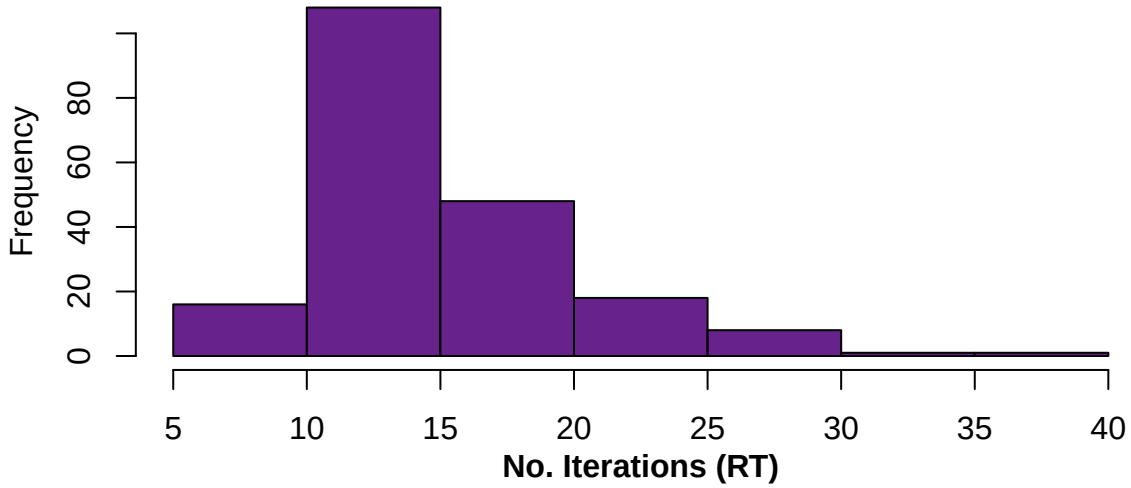


As we can see, over these 200 trials, only a fraction of the circumference was ever visited. Responses associated with the bottom-left region of the circle were not visited once.

Distribution of RT (total no. of iterations) required

On every trial, we take the number of iterations required to reach the threshold as a measure of Response Times (RT). Let us now present a histogram of the RT observed along these 200 trials. This distribution, as expected for any RT distribution, seems to be skewed towards positive values.

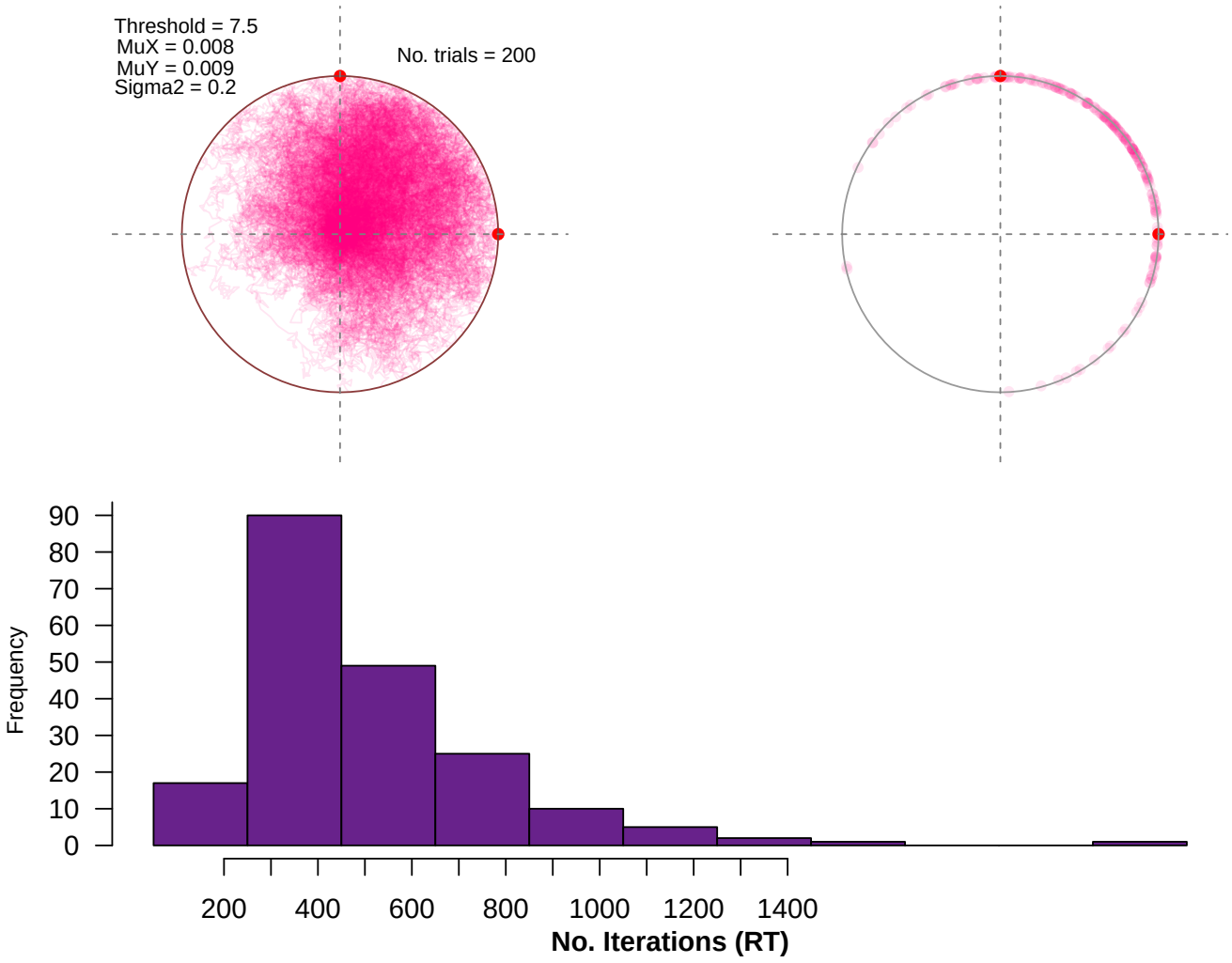
```
hist(finalT, col = "darkorchid4", breaks = 10, ann=FALSE)
mtext("No. Iterations (RT)", 1, line=2, f=2)
mtext("Frequency", 2, line = 2.5)
```



4. Parameter values

To ensure that the number of iterations resemble RT data, we will force an increment in the the number of iteration it takes for the decision process to reach the circumference. We will do this by either increasing the threshold value, or decreasing the values chosen for `mu.1`, `mu.2` and `sigma.driftrate`.

```
mu1.drift <- 0.008          # Much smaller
mu2.drift <- 0.009          # Much smaller
sigma.driftrate <- 0.2      # Smaller
```



5. Parameters in the CDDM

Starting point

The evidence accumulation process is captured at each point in time as a pair of (x,y) coordinates that are assumed to move from an initial point towards a specific location on the circumference.

For an unbiased system, the starting point is assumed to be located at the origin (0,0), since it would imply that the distance between the starting point and any given point on the circumference is the same.

Threshold (a)

Under the CDDM, the ratio of the circle representing the set of choices available to the decision-making agent embodies the threshold (a.k.a. "criteria" or "boundary") that determines when to respond.

The threshold is assumed to be under the decision-maker's control, as it is typically associated with the speed-accuracy trade-off settings of the task. Overall, higher criteria lead to slower but more accurate responses.

Drift rates (μ_1 and μ_2)

The CDDM incorporates two drift rate parameters μ_1 and μ_2 that capture the average step-size observed on the x and y axes per unit of time (i.e. the mean deviation on the x and y axis observed on each iteration of the information accumulation process).

Just like in the DDM, these drift rate parameters are assumed to depend on the stimuli observed (i.e. the information contained in the stimulus). When a stimulus is uninformative, we expect a "zero-drift process" (i.e. $\mu_1 = 0$ and $\mu_2 = 0$), where very response is equally likely to be observed.

Diffussion coefficient (σ^2)

The diffusion coefficient represents a stimulus-independent source of variation (i.e. spread of evidence). It is assumed to be independent and identical across the horizontal and the vertical dimensions of the evidence accumulation process.

6. Angle response

The coordinates on the circumference reached as observed responses are not easy to interpret, as their units depend on the threshold. Instead, these coordinates get transformed into **angles**.

Below, we have constructed two R functions. The first one, takes the x,y coordinates and turns them into a degree angle. The second one, takes these degree angles and transforms them into radians.

```
#X,Y coordinates are transformed into an angle (in degrees)
theta.d <- function(x,y){   #This function takes the X,Y coordinates as inputs
  theta <- atan2(y,x) * 180 / pi  #Angle with respect of y=0
  if(theta<0){theta <- theta+360}   #Correction for whole circle (360)
  return(theta)} #return Theta value

#The angle (in degrees) is now express in radians.
theta.r <- function(theta.d){  #This functiont takes angle in degrees as input
  theta <-  theta.d * pi /180  #Transform to radians
  return(theta)}
```

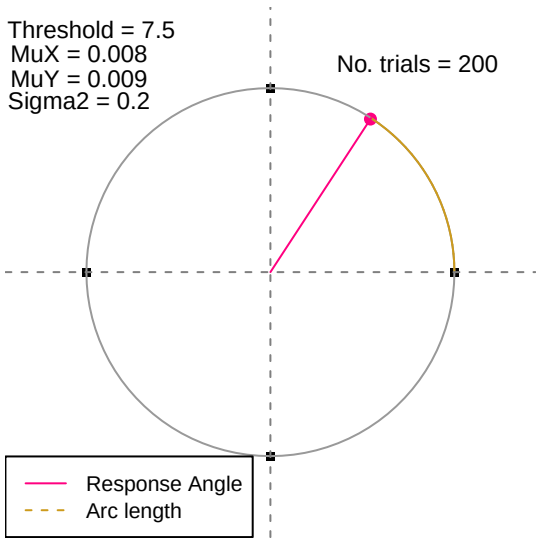
For example, let's get the angles corresponding to the circumference coordinates of the last trial.

```
x <- state[t,1,trials]   #Final state x coordinate recorded
y <- state[t,2,trials]   #Final state y coordinate recorded
angle <- theta.d(x,y); angle
```

```
## [1] 56.82315
```

```
radians <- theta.r(angle); radians
```

```
## [1] 0.9917511
```



7. Responses and Response Times (Probability functions)

From our initial simulation presented in Section 2, we know that there are 5 different parameters that we need to specify in order to be able to emulate the information accumulation process that takes place on any given trial:

1. Initial state (s.init)
2. Threshold (thresh)
3. Average step size on X (Mu.1)
4. Average step size on Y (Mu.2)
5. The variance on the steps size

As illustrated in Sections 3 and 4, the specific values used across **these parameters have an impact in both** defining which are the most likely subsets of **responses** (i.e. the overall direction in which information is accumulated, θ) **and response times** (i.e. how quickly the system can reach the circumference, t) to be observed. Hence, performance on any trial is captured by the random pair (θ,t).

The model establishes that the probability of observing a certain response (as defined by its angle θ, measured in radians) with Response Time t ($\Pr(\text{Angle} = \theta \cap RT = t)$) is defined as:

$$\Pr(\text{Angle} = \theta \cap RT = t) = \left[\exp\left(\frac{a(\mu_1\cos(\theta) + \mu_2\sin(\theta))}{\sigma^2}\right) - \frac{|\mu|^2(t - t_{ND})}{2\sigma^2}\right] \times \left[\frac{\sigma^2}{a^2}\sum_{k=1}^{n}\frac{j_{0,k}}{J_1(j_{0,k})}\exp\left(-\frac{j_{0,k}^2\sigma^2(t - t_{ND})}{2a^2}\right)\right]$$

Important note: One of the main references used to create this tutorial was Kvam and Turner, (2021), which was found to be really helpful in understanding the overall dynamics of the model. However, it may be noted that there's an error in the main equation they present for this model, where they had written $\frac{a^2}{\sigma^2}$ instead of $\frac{\sigma^2}{a^2}$, as originally proposed by Smith (2016).

Under the CDDM framework, the angle corresponding to the response observed is assumed to be independent from the RT registered, and so the equation presented consists on the product of what can be think of as two different functions: a function describing the probability of observing a certain response angle (first square bracket) and a function related to the probability of observing a certain response time (second square bracket).

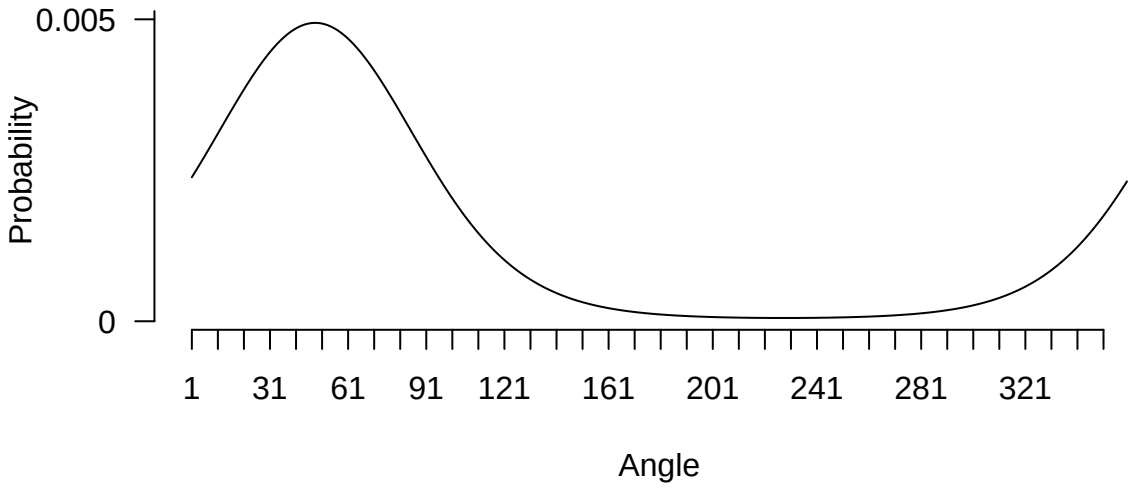
In the probability function presented above a represents the threshold, t_{ND} captures the No-Decision time (i.e. elapsed time that cannot be attributed to the decision process), μ_1 is the average step size on X, μ_2 is the average step size on Y and σ^2 is the variance on step sizes. The function $J(\cdot)$ represents the **first-order Bessel function of the first kind**, with the elements $j_{0,k}$ representing the zeros of the zero-order Bessel function of the first kind. The sum $\sum_{k=1}^{n}$ allows us to iterate and evaluate the function for the first n zeros. Hypothetically, this process should extend to infinite ($n = \infty$), but it has found to be very well approximated with $n \leq 150$. It is also important to note that **the angle θ, is expressed in radians.**

```
## Write our probability function in R for the CDDM model:
probFunction <- function(theta,mu1,mu2,sigma.driftrate,thresh,tnd){
  #### Step 1: Let's begin with some definitions
  mu.vector <- c(mu1,mu2)            # Define Mu vector
  mu.mag <- sqrt(sum(mu.vector^2))   # Magnitude of Mu vector

  ### Step 2: Let's work terms related to the Probability of Response
  pResp.frac1 <- (thresh*((mu.vector[1]*cos(theta))+(mu.vector[2]*sin(theta))))/(sigma.driftrate^2)
  pResp.frac2 <- ((mu.mag^2)*(t-tnd))/(2*(sigma.driftrate^2))
  pResp <- exp(pResp.frac1 - pResp.frac2)

  #### Step 3: Let's build terms related to the Probability of Response Time
  library(CircularDDM)  # Loading besselzero function
  bess <- NULL  #Empty vector to store outcome of bessel function
  for(k in 1:150){  #Iterate the sum up to n = 150
    j <- besselzero(0,k,1)
    pRT.frac1 <- j/besselJ(j,1)
```

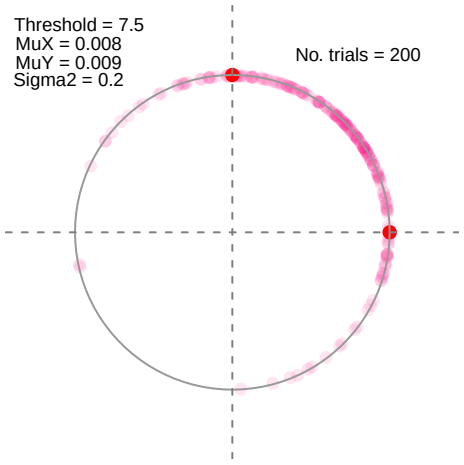
```
    pRT.frac2 <- exp(-(((j^2)*(sigma.driftrate^2)*(t-tnd))/(2*(thresh^2))))
    b <- pRT.frac1 * pRT.frac2
    bess[k] <- sum(b)
      if(k>1){if(round(bess[k],3) == round(bess[k-1],3)){break}}} #End when convergence is achieved
  pRT.term1 <- (sigma.driftrate^2)/(thresh^2)
  pRT.term2 <- sum(bess)
  pRT <- pRT.term1 * pRT.term2
  probFunction <- pRT*pResp #Overall joint probability function
return(probFunction)} # Retrieve probability
```

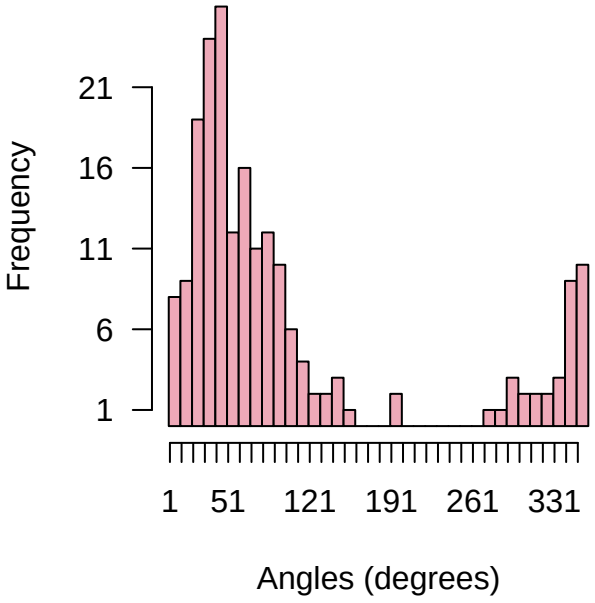
Since Response Times and Response Angles are assumed to be independent from each other for unbiased systems with no inter-trial variability in the starting point nor the values of μ_1 and μ_2, we can explore how this probability function works for every possible angle to be observed, keeping the response time and every other parameter constant, using the same values as in Section 4:

```
tnd <- 0  # Define some No-decision time
angles <- seq(1,360,1)  # All possible angles
angles <- theta.r(angles) # Transform into radians
Probs <- NULL  #Empty vector to store probabilities
for(a in 1:length(angles)){ # For each angle defined compute the probability
    Probs[a] <- probFunction(angles[a],mu1.drift,mu2.drift,sigma.driftrate,thresh,tnd)}
```



Now, for comparison, let's take a look at responses and response angles observed in our simulation..





Expected Angle response (θ_μ)

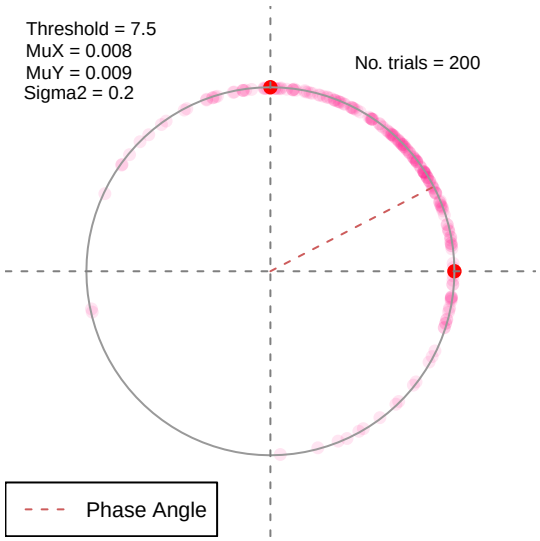
If we think of the drift parameters μ_1 and μ_2 as a vector $\mu = (\mu_1, \mu_2)$, its direction can be used to capture the angle (i.e. the **phase angle**, θ_μ) corresponding to position on the circumference that is more likely to be reached based on the mean rate of evidence growth expected on the x (μ_1) and y (μ_2) axes.

```
PhaseAngle <- 1/tan((mu2.drift)/(mu1.drift)); PhaseAngle
```

```
## [1] 0.477881
```

When the accumulating evidence hits the circumference at the point that corresponds to the phase angle of the drift (and therefore, $\theta - \theta_\mu = 0$), the system is said to behave as a maximum likelihood observer, as this point is assumed to correspond to the most probable response angle.

The phase angle is assumed to depend on the identity of the stimuli.



Expected RT

From the density expressed in the joint probability function presented before, we can obtain the expected response time by computing the expectation after conditioning with respect to the hitting probability.

$$\tilde{E}[T] = \frac{aI_1(\frac{a||\mu||}{\sigma^2})}{||\mu||I_0(\frac{a||\mu||}{\sigma^2})}$$

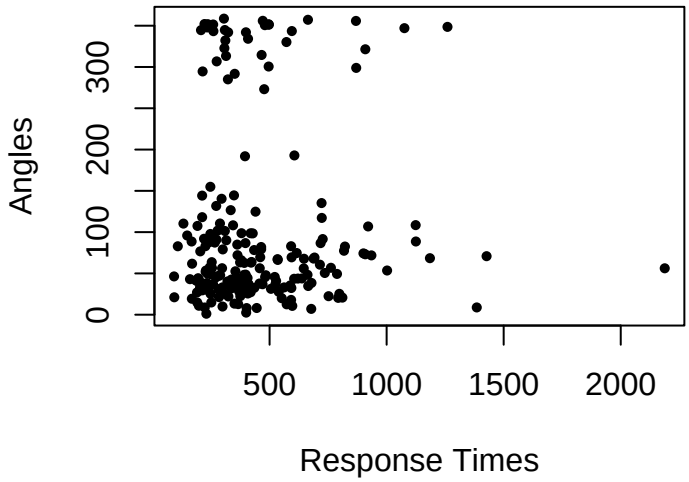
```
ExpRT <- function(thresh, mu1.drift, mu2.drift, sigma.driftrate){
  mus <- c(mu1.drift,mu2.drift)
  mu.mag <- sqrt(sum(mus^2))
  library(CircularDDM)
  Num <- thresh*besselI((thresh*mu.mag)/sigma.driftrate^2,1)
  Denom <- mu.mag*besselI((thresh*mu.mag)/sigma.driftrate^2,0)
  ExpRT <- Num/Denom
  return(ExpRT)}

ExpRT(thresh,mu1.drift,mu2.drift,sigma.driftrate)
```

```
## [1] 458.3348
```

Response Time and Response Angle independence

As established by Smith (2016), for an unbiased system that doesn't display any across-trial variability in the starting point nor in the drift and diffussion parameters, we assume that Response Times and Response Angles are independent (hence, the joint probability function presented above). These conditions are met by the simulation process presented in Sections 2, 3 and 4, where we didn't induce any kind of inter-trial variability in terms of either μ_1, μ_2, σ^2 nor the starting point.

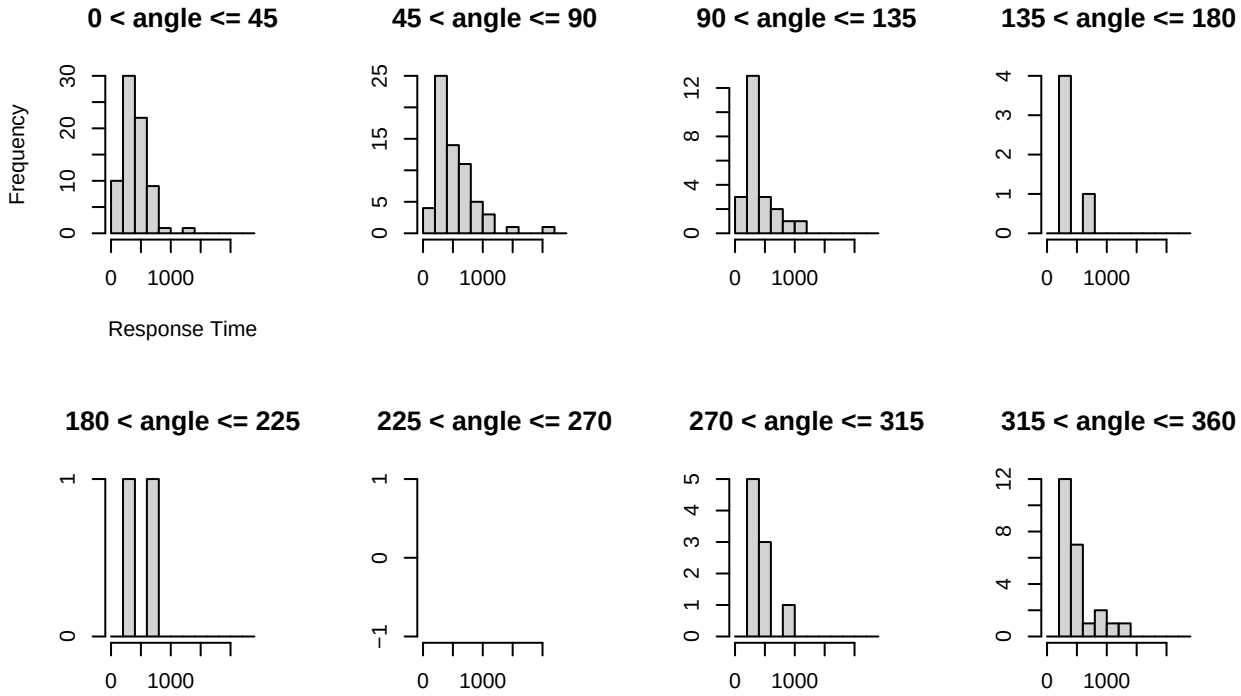
For proof, let us explore the correlation between the response times and response angles observed in our simulated data:




```
cor(finalT,thetas) #Correlation between RT and Response Angles
```

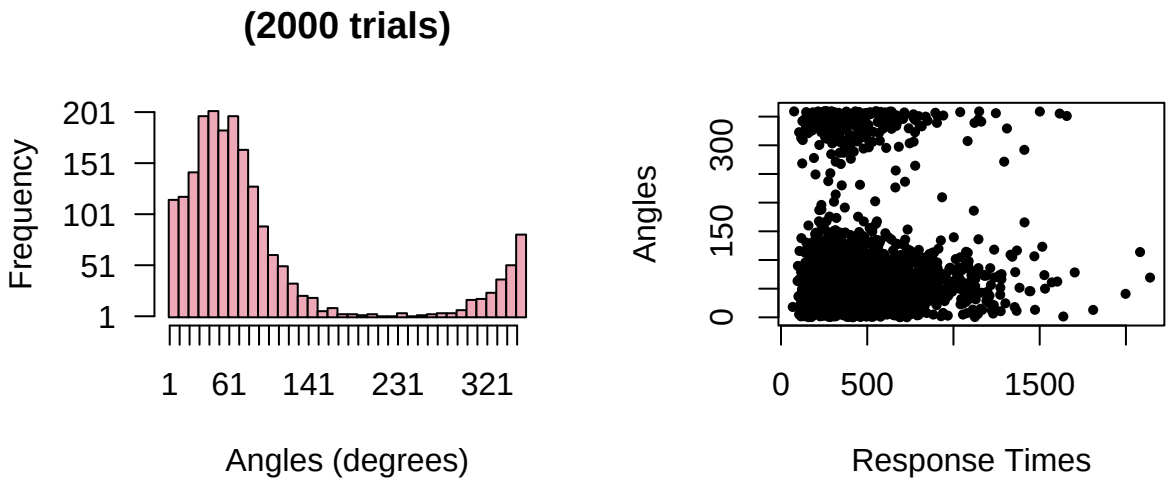
```
## [1] 0.02704574
```

Let us define multiple subsets, so that we can observe the corresponding Response time histogram for each one of them:

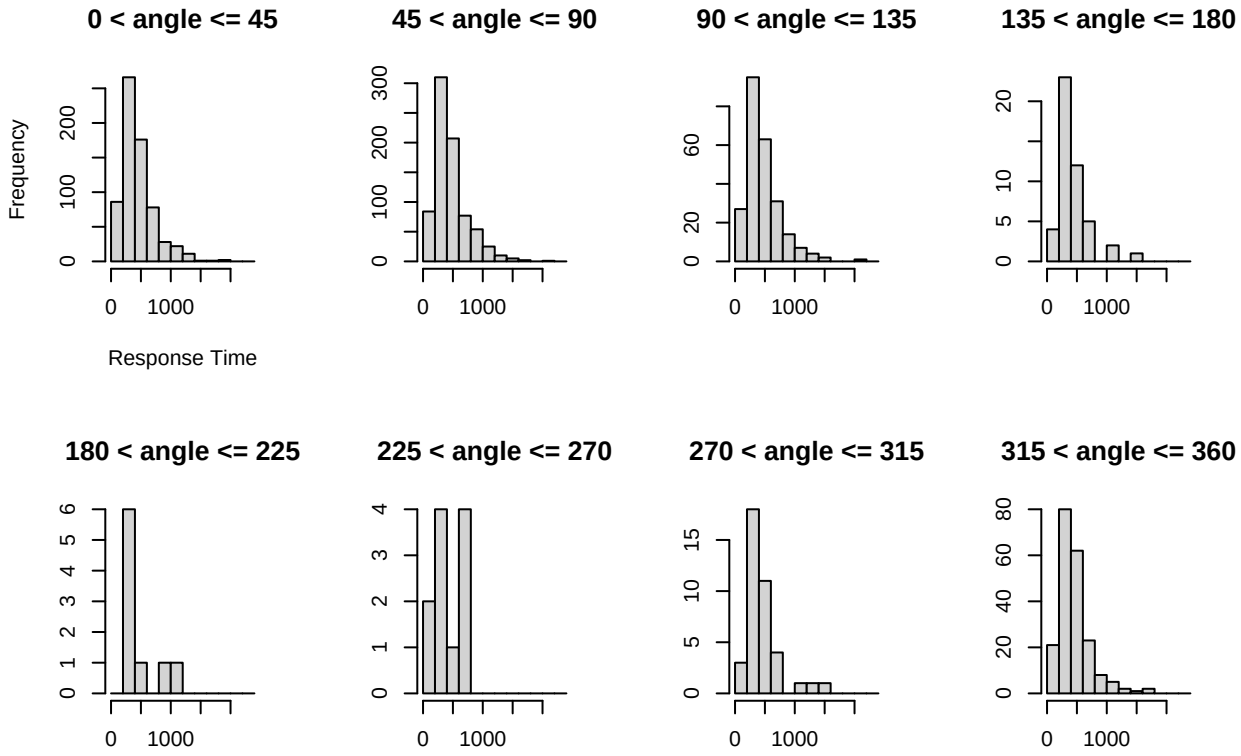


All of the histograms presented peak at very similar Response Times, even though the overall shape of these histograms isn't exactly the same across subsets. This is most likely due to the fact that we only observed 200 trials, so that a few of these histograms correspond to very few occasions, or none at all.

With this in mind, we may want to repeat our simulation, this time using $10x$ trials:



As we can see, the histogram of response angles observed on the 2,000 trials resembles even more the pattern portrayed by the pdf function previously presented. Furthermore, we confirm that there's not a correlation between response angles and RTs (Corr = 0.0010383), which can also be appreciated if we plot again the different histograms for different angle subsets.



8. Parameter recovery test