ODEBootcamp-LucasTopham

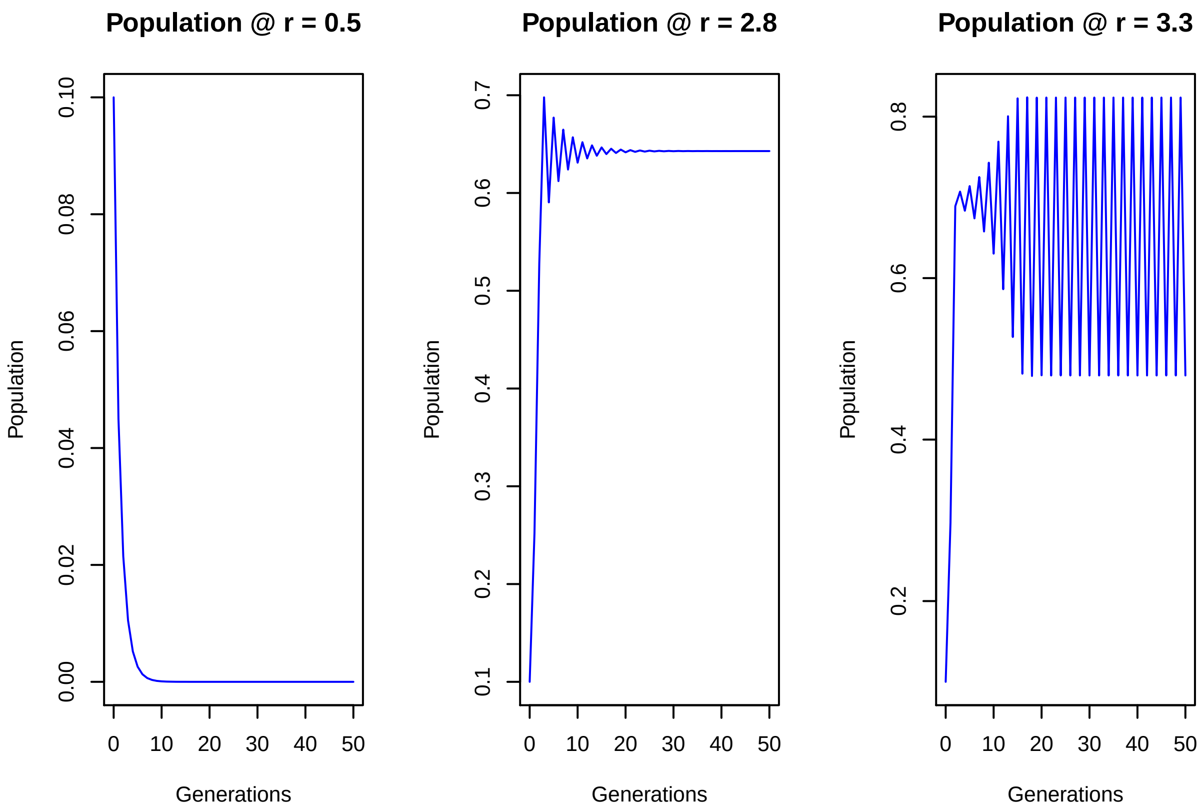
Question 1: Salmon population growth

To model the growth of the salmon population, over 50 generations (n = 50), over 3 different rates (r = 0.5, 2.8, 3.3), and with a starting population x0 = 0.1:

```
### calculate the growth of the population for different rates of growth
# initial pop
x0 <- 0.1
# growth rates
rates <- c(0.5, 2.8, 3.30)
# generations
n <- 50
t <- seq(from = 0, to = n, by = 1)

# calculate for different rates
PopGrowth <- as.data.frame(matrix(ncol = 0, nrow = length(t)))
for (r in rates) {
  name <- paste0("Pop@",r)
  N <- numeric(length(t))
  N[1] <- x0
  for (h in seq_len(length(t)-1)) {
    N[h+1] <- r*N[h]*(1-N[h])
  }
  PopGrowth[,name] <- as.vector(N)
}
```

Now plotting the population as a result of each growth rate:



For r = 0.5, the population of salmon dies out in under 10 generations. For r = 2.8, after some initial fluctuations between generations 0-25, the population of salmon settles at a level of ~0.65 or approximately (r-1)/r. For r = 3.3, after some initial smaller fluctuations, the population of salmon fluctuates every generation between ~0.48 and ~0.82. What this would indicate about the salmon population is that at r = 0.5, the reproduction rate is not enough to sustain the salmon population and the salmon quickly die out, while at r = 2.8, the reproduction rate eventually settles at a level where the number of salmon born each year is enough to sustain a greater population of salmon. At r = 3.3, the reproduction rate may be too high and cause the salmon population to be too large for their environment to sustain them, therefore in two year cycles we see a large increase in salmon being born, followed by a large die off of much of the population.

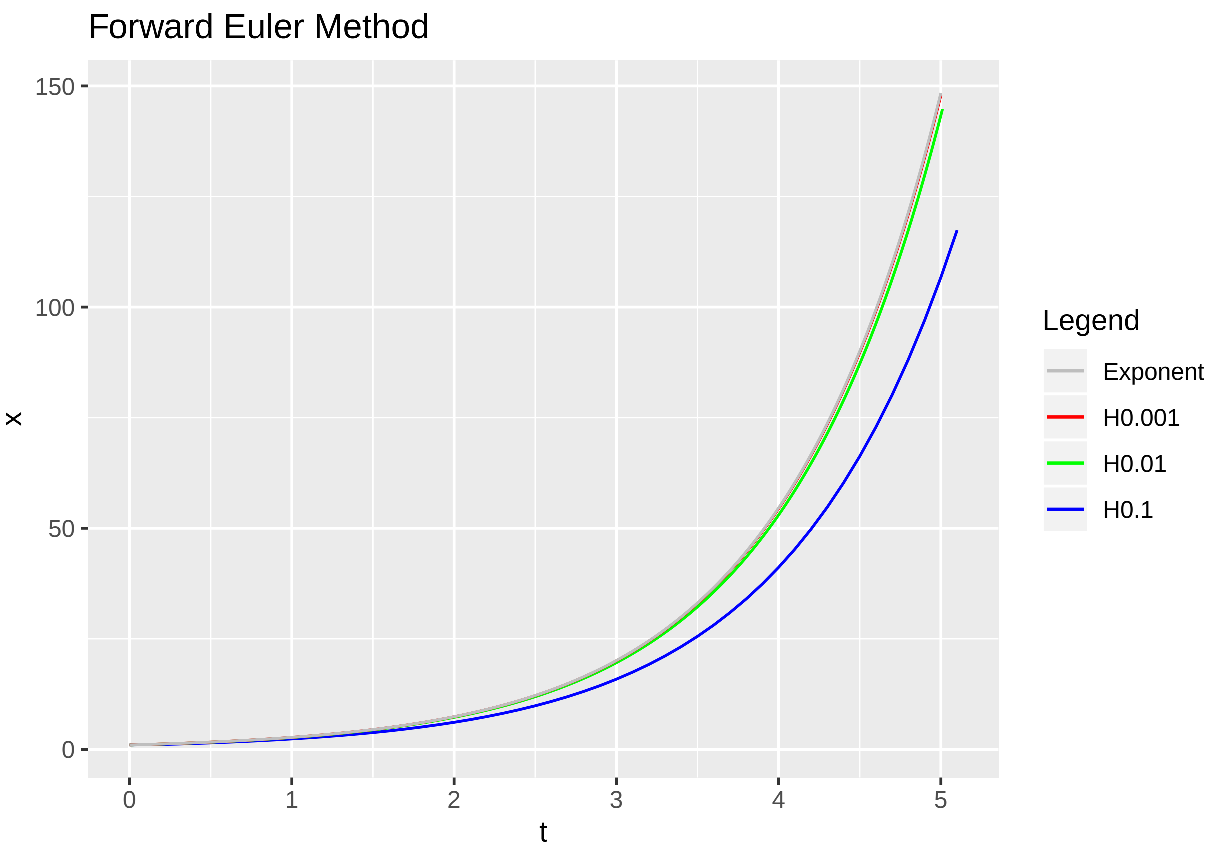
Question 2 : The Forward Euler Algorithm

```
### The Forward Euler
# f = the differential equation, dx/dt = f(x)
# x0 = the initial condition
# tmax = time to integrate until
# h = step size

forwardEuler <- function(f, x0, t0, tmax, h) {
  n <- seq(t0, tmax, by = h)
  eulerdata <- as.data.frame(matrix(nrow = length(n), ncol = 0))
  eulerdata[1,"x"] <- x0
  eulerdata[1,"t"] <- t0
  for (i in 2:length(n)) {
    eulerdata[i,"x"] <- eulerdata[i-1,"x"] + h*f(eulerdata[i-1,"x"])
    eulerdata[i,"t"] <- i*h
  }
  return(eulerdata)
```

```
}
```

Implement the above formula for the following parameters: f = dx/dt = x; x0 = 1; t0 = 0; tmax = 5; h = 0.1, 0.01, 0.001, and plot to compare it to the function



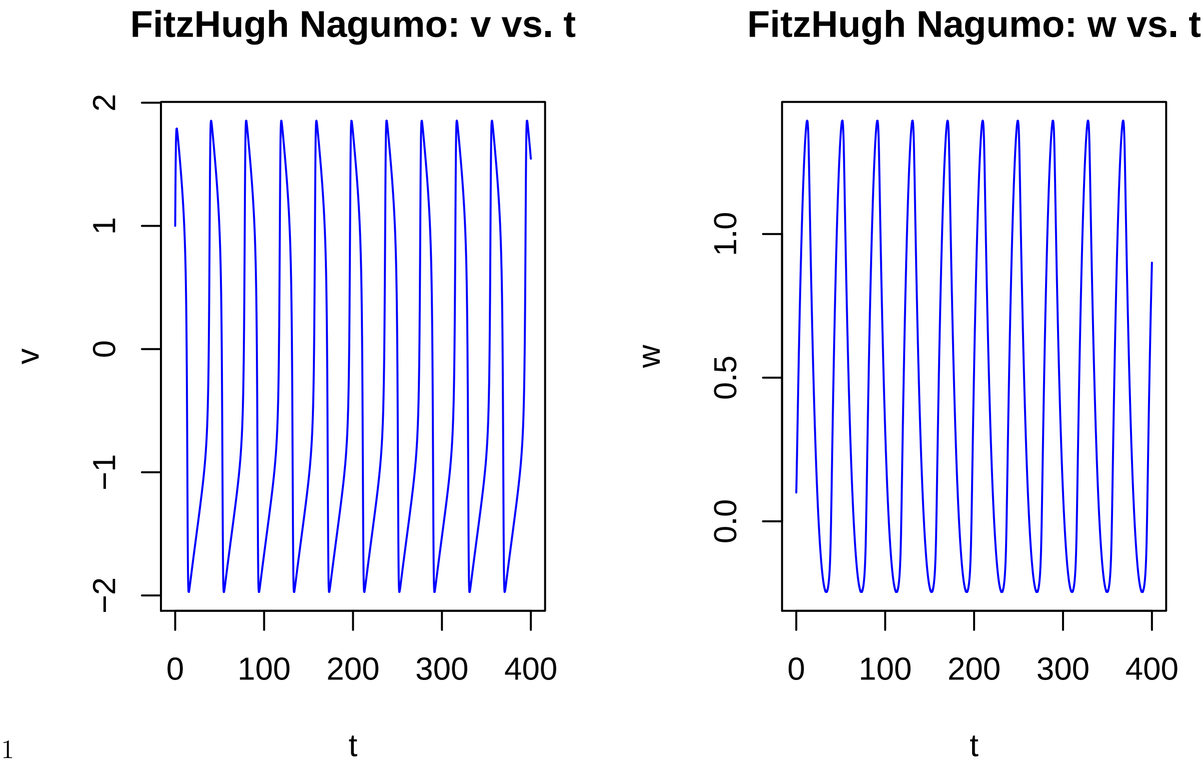
As the h-value decreases, we can see that the plotted data gets closer to the exponent line. This is expected as the h-value represents the step size between taking tangents and as we decrease the step size and more frequently take the tangent of the solution curve, we more accurately model the underlying function.

Question 3 : The FitzHugh-Nagumo Model

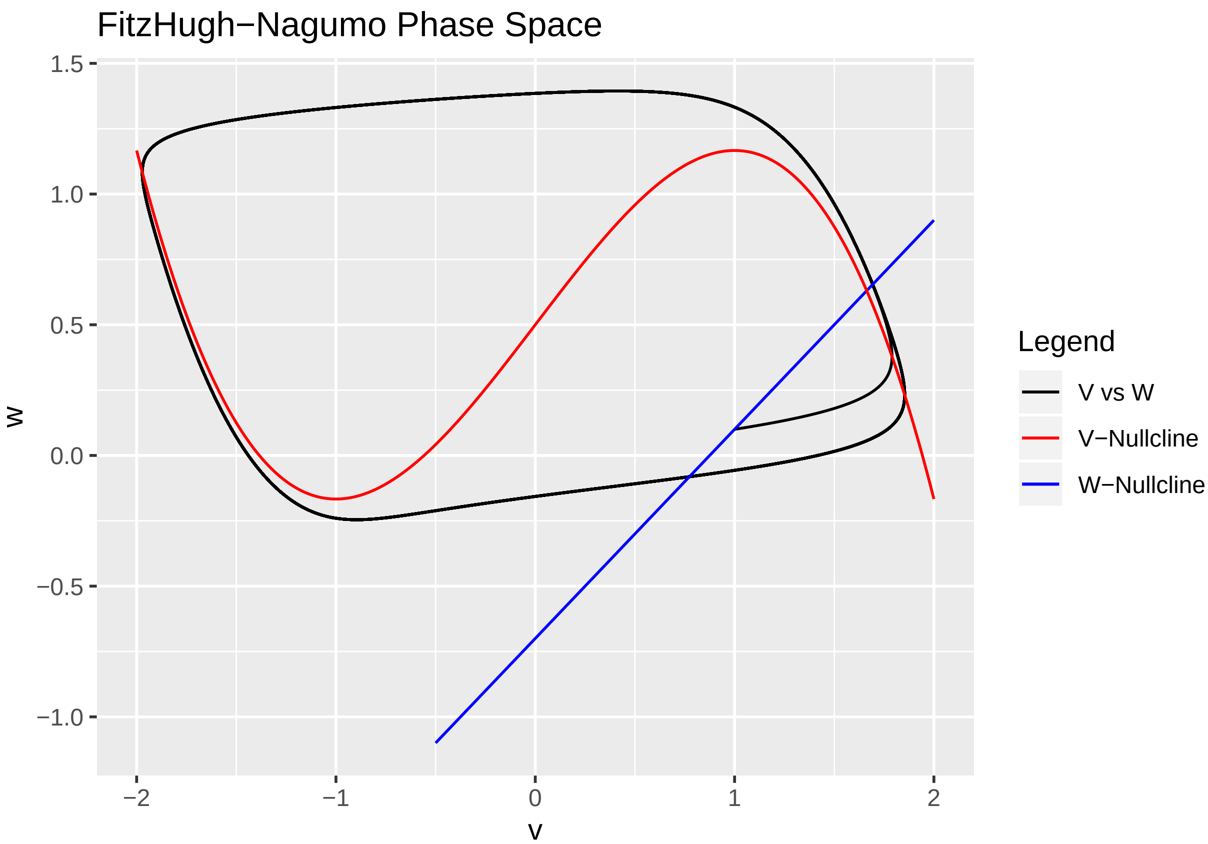
```
FHN <- function(I, v0, w0, t, fv, fw, h, a, b, e) {
  n <- seq(0, t, by = h)
  FHNdata <- as.data.frame(matrix(nrow = length(n), ncol = 0))
  FHNdata[1,"v"] <- v0
  FHNdata[1,"w"] <- w0
  for (i in 2:length(n)) {
    # the v'
    FHNdata[i,"v"] <- FHNdata[i-1,"v"] + h*fv(FHNdata[i-1,"v"], FHNdata[i-1,"w"])
    # w'
    FHNdata[i,"w"] <- FHNdata[i-1,"w"] + h*fw(FHNdata[i-1,"w"], FHNdata[i-1,"v"])
  }
  return(FHNdata)
}

# parameter values
t = 400
```

```
v0 = 1
w0 = 0.1
I = 0.5
h = 0.01
a = 0.7
b = 0.8
e = 0.08
fv = function(v, w) v - (v^3*1/3) - w + I
fw = function(w, v) e*(v + a - b*w)
```



1. Section 1
2. Section 2



3. Section 3

```
# find the roots for both v' and w'
vcoefs <- c(I, 1, 0, -1/3)
wcoefs <- c(-a, b)
vroots <- polyroot(vcoefs)
wroots <- polyroot(wcoefs)
```