# Numerically Solving Differential Equations

John M. Drake

November 4, 2024

This handout shows how to use R to numerically solve a differential equation or a system of differential equations. In what follows, we use the function `ode` from the package `deSolve`.

## Load Required Package

```
require(deSolve)
```

```
## Loading required package: deSolve
```

```
## Warning: package 'deSolve' was built under R version 4.3.1
```

To solve the theta logistic model, $\frac{dn}{dt} = rn\left(1 - \left(\frac{n}{k}\right)^{\theta}\right)$, we need a function that evaluates this derivative and returns the numerical result in the form of a list. Note that the arguments of this function follow the requirements of ode, i.e., a time variable, followed by the state vector, followed by the parameters.

```
theta.logistic <- function(t, x, parms){
  r <- parms[1]
  k <- parms[2]
  theta <- parms[3]
  dy <- list(r * x * (1 - (x / k)^theta))
}
```

To generate a solution of this model, we first declare vectors of states (population size) and parameters. We also declare a vector of times at which we want to observe the solution.

```
population.size <- 2
parameters <- c(1, 200, 0.2)
times <- seq(0, 40)
```

Now we can use `ode` to obtain a solution.

```
solution <- ode(y = population.size, times = times,
                func = theta.logistic,
                parms = parameters)
```

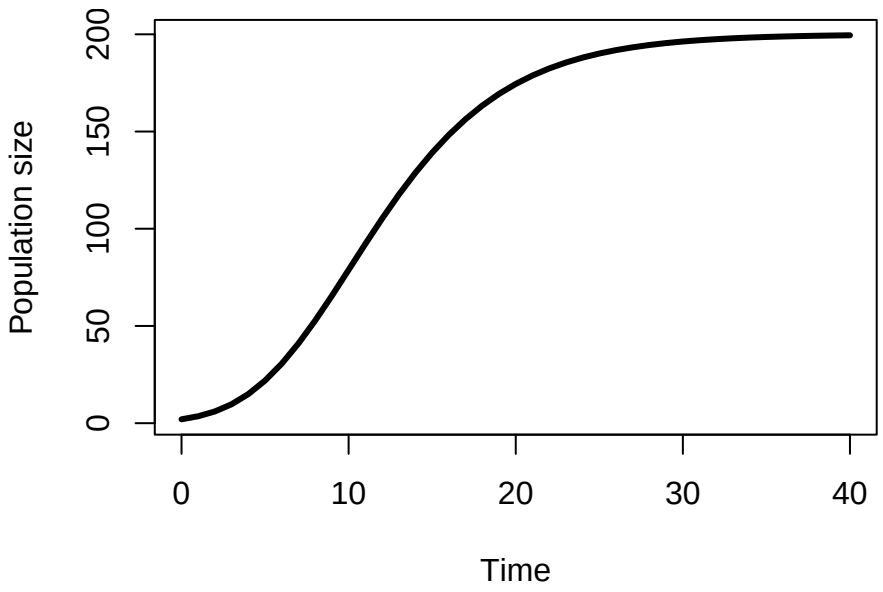
Using the function head, we inspect the first five lines of output.

```
head(solution)
```

```
##      time         1
## [1,]    0  2.000000
## [2,]    1  3.563662
## [3,]    2  6.044046
## [4,]    3  9.751733
## [5,]    4 14.974215
## [6,]    5 21.913642
```

We can plot the solution using a line plot.

```
plot(solution[,1], solution[,2], type = 'l', lty = 1, lwd = 3,
     xlab = 'Time',
     ylab = 'Population size')
```




This technique is readily extended to multiple dimensions.

```
lotka.volterra <- function(t, x, parms){
  N <- x[1]
  P <- x[2]
  r <- exp(parms[1])
  k <- exp(parms[2])
  a <- exp(parms[3])
  h <- exp(parms[4])
  e <- exp(parms[5])
  mu <- exp(parms[6])
  dN <- r * N * (1 - N / k) - P * N
  dP <- e * P * N - mu * P
  list(c(dN, dP))
}

population.size <- c(N = 10, P = 2)
parameters <- c(log(1.8), log(500), log(0.045), log(0.021), log(0.3), log(1.0))
times <- seq(0, 100, by = 0.01)
solution <- lsoda(y = population.size, times = times,
                  func = lotka.volterra,
                  parms = parameters)
head(solution)
```

```
##      time         N        P
## [1,] 0.00 10.000000 2.000000
## [2,] 0.01  9.974424 2.040325
## [3,] 0.02  9.944883 2.081293
## [4,] 0.03  9.911345 2.122879
## [5,] 0.04  9.873800 2.165069
## [6,] 0.05  9.832233 2.207835
```

```r
plot(solution[,1], solution[,2], type = 'l', lty = 1, lwd = 3,
     xlab = 'Time', ylab = 'Population size')
lines(solution[,1], solution[,3], lty = 1, lwd = 3, col = 'blue')
legend('topright', legend = c('Prey', 'Predators'), lty = 1,
       col = c('black', 'blue'), bty = 'n')
```