Euler's Method

Morgan de Ferrante

Euler's Method Formula

Euler's Method is a first-order procedure for solving ordinary differential equations with the initial value problem (IVP)

$$y'(t) = f(t, y(t))$$

with initial condition

$$y_0 = y(t_0)$$

Given an initial condition, we can estimate the solution using the following iterative method:

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i))$$

where h is the step size.

The following function generates a list of t_i and $y(t_i)$ values using Eulers Method over the interval [a,b]. The initial conditions are $t_0 = a$ and y_0.

```
y <- 0
t <- 0
euler <- function(f = function(t,y){}, a, b, h, y0){
  N = (b - a)/h
  t[1] = a
  y[1] = y0
  for (i in 1:N) {
    y[i + 1] = y[i] + h*f(t, y[i])
    t[i + 1] = a + i*h
  }
  list(t = t, y = y)
}
```

Example

Exact solution:

$$y(t) = \frac{y_0 K}{y_0 + (K - y_0)e^{-rt}}$$

Initial Value Problem:

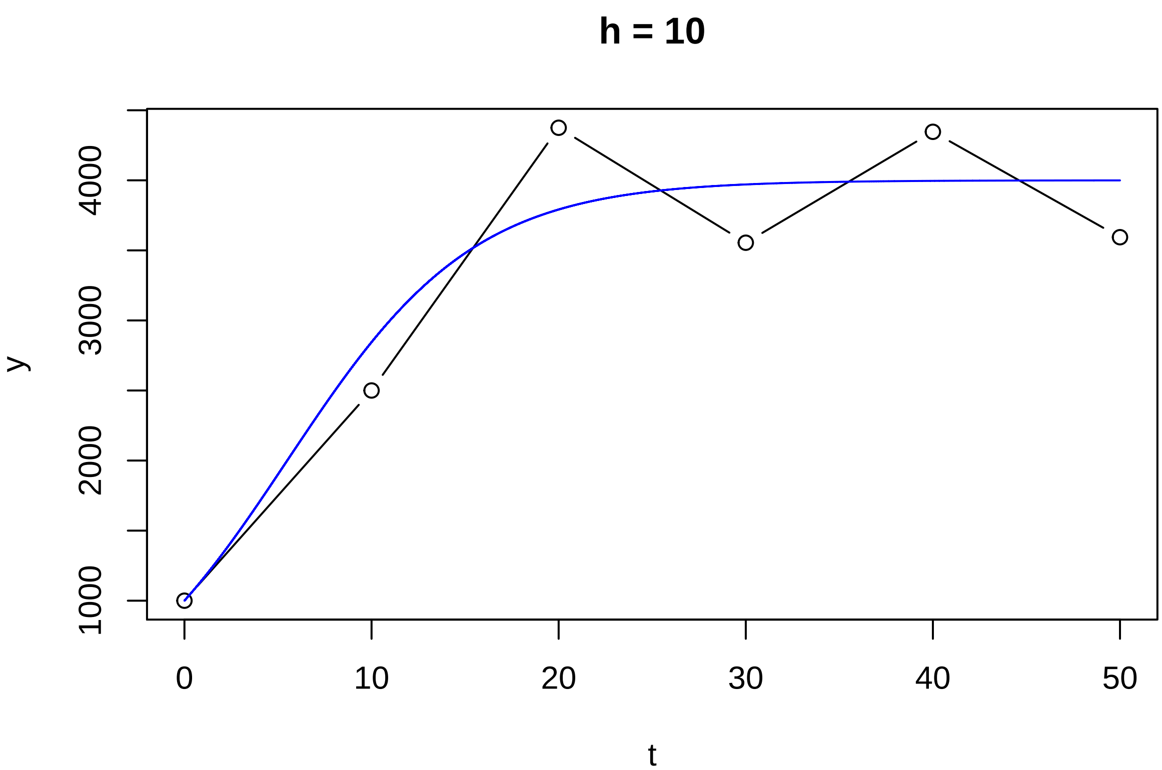
$$f(t, y) = \frac{ry}{(1 - \frac{y}{K})}$$

$$t_0 = 0$$

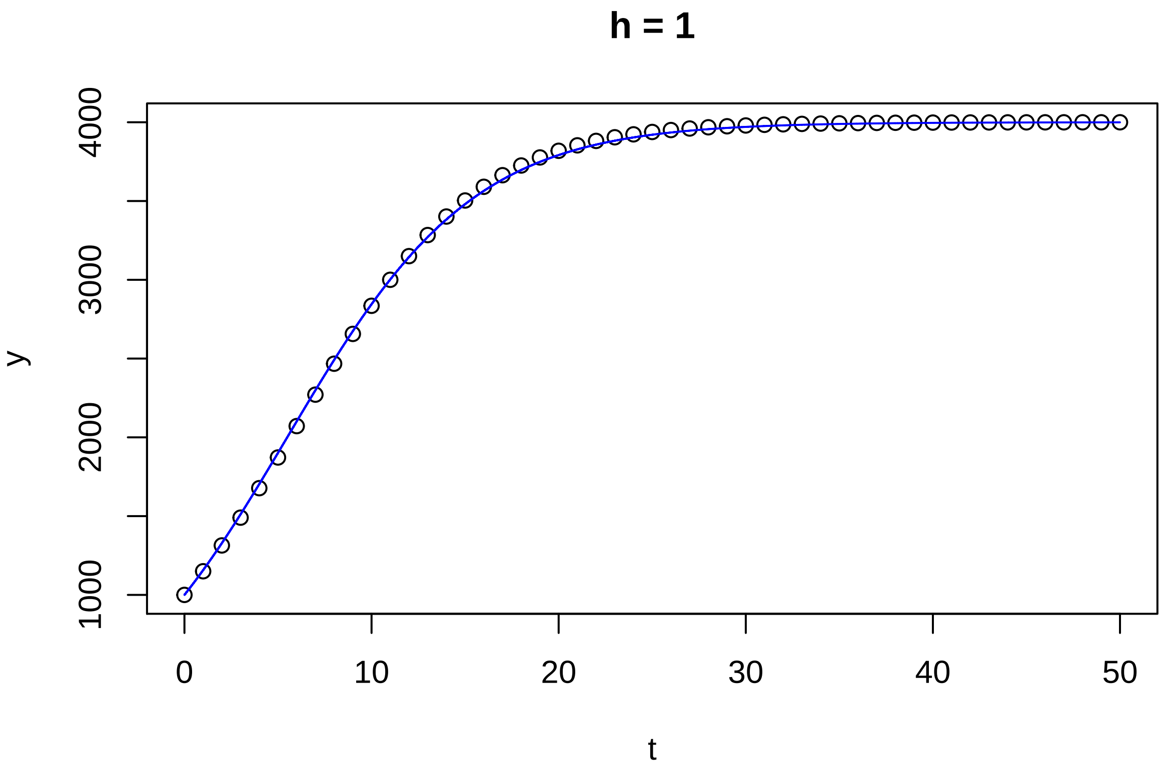
$$y_0 = 1000$$

The following solves the IVP for the given conditions and generates plots for each step value. The exact solution is shown in blue in each plot.

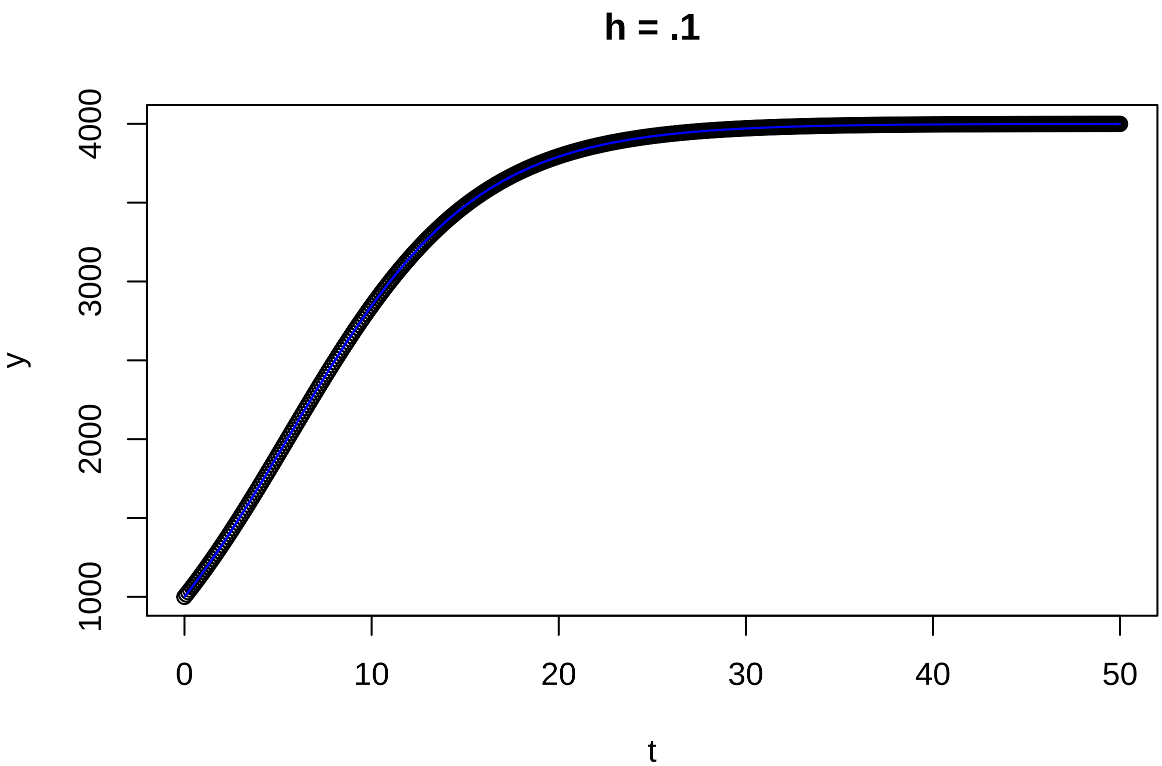
```r
r <- .2 # growth rate
K <- 4000 # 0 < y0 < K

f <- function(t,y){r*(1 - y/K)*y} # f(t,y) = y'(t)

step1 <- euler(f, 0, 50, 10, 1000) # h = 10
step2 <- euler(f, 0, 50, 1, 1000) # h = 1
step3 <- euler(f, 0, 50, .1, 1000) # h = .1

step1_t <- unlist(step1[1]) # t values for h = 10
step1_y <- unlist(step1[2]) # y values for h = 10

step2_t <- unlist(step2[1]) # t values for h = 1
step2_y <- unlist(step2[2]) # y values for h = 1

step3_t <- unlist(step3[1]) # t values for h = .1
step3_y <-  unlist(step3[2]) # y values for h = .1

# this creates the exact function, with initial value y0 = 1000
exact <- function(t, y0 = 1000) {
  yt <- y0*K/(y0 + (K - y0)*exp(-r*t))
}

plot(step1_t, step1_y, type = "b", xlab = "t", ylab = "y", main = "h = 10")
curve(exact, 0, 50, 100000, add = TRUE, col = "blue")
```




```r
plot(step2_t, step2_y, type = "b", xlab = "t", ylab = "y", main = "h = 1")
curve(exact, 0, 50, 100000, add = TRUE, col = "blue")
```




```
plot(step3_t, step3_y, type = "b", xlab = "t", ylab = "y", main = "h = .1")
curve(exact, 0, 50, 100000, add = TRUE, col = "blue")
```



The following generates the actual maximum error for each step size:

```r
library(nnet)
cat("Step size of h = 10 has a maximum error of ", max(abs(exact(step1_t) -
    step1_y)), " at  i =", which.is.max(abs(exact(step1_t) - step1_y)), "\n")
```

```
## Step size of h = 10 has a maximum error of  583.34  at  i = 3
```

```r
cat("Step size of h = 1 has a maximum error of ", max(abs(exact(step2_t) - step2_y)),
    " at  i =", which.is.max(abs(exact(step2_t) - step2_y)), "\n")
```

```
## Step size of h = 1 has a maximum error of  29.97472  at  i = 7
```

```r
cat("Step size of h = .1 has a maximum error of ", max(abs(exact(step3_t) -
    step3_y)), " at  i =", which.is.max(abs(exact(step3_t) - step3_y)))
```

```
## Step size of h = .1 has a maximum error of  2.890099  at  i = 56
```

As would be expected, the smaller the step size, the closer the estimates generated from Euler's Method are to the exact solution.