CS-7863: Scientific and Statistical Computing
Homework #2
Noah L. Schrick – 1492657

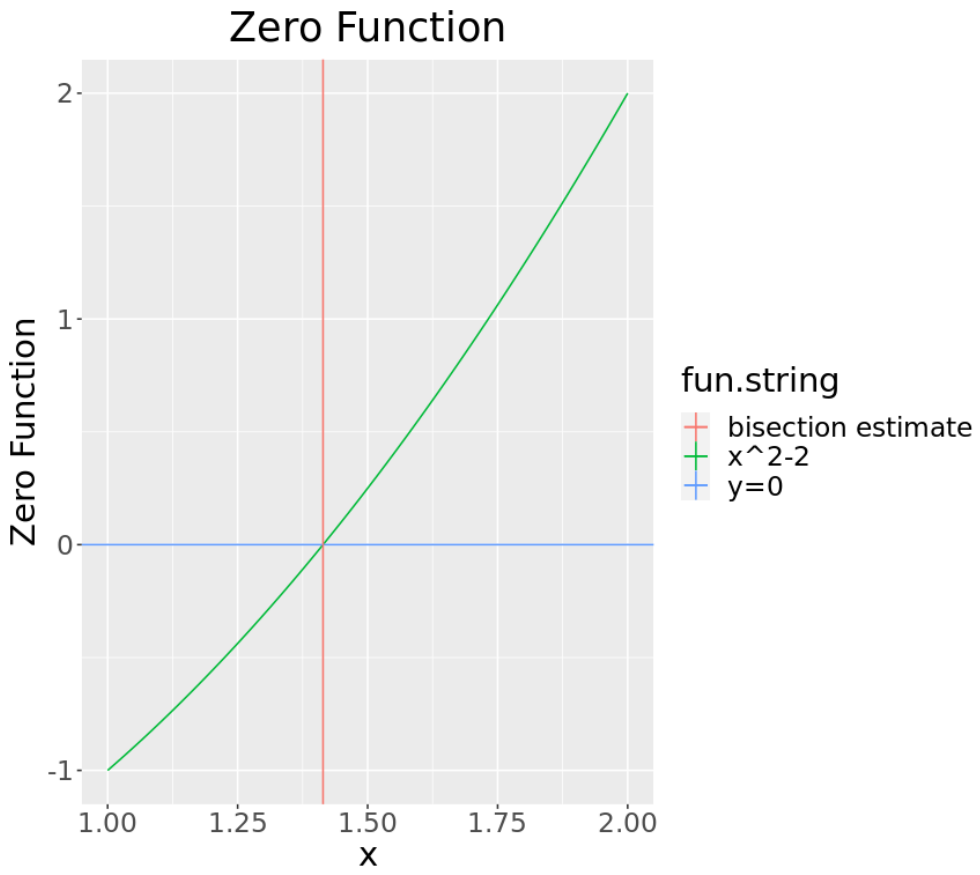
1.)

```
## Part 1: Bisection
findZeroBisect <- function(func, xl, xr, tol, maxsteps=1e6){
  steps <- 0
  if (func(xl) * func(xr) >= 0){
    stop("Bad interval: try again with different endpoints")
  }

  repeat{
    xm <- (xl + xr)/2
    if(abs(func(xm)) < tol || steps >= maxsteps){
      break
    }

    ifelse(func(xl) * func(xm) > 0, xl<-xm, xr<-xm)
    steps <- steps + 1
  }
  return(c(xm, func(xm), steps))
}
```

Note: The plotting code shown in part 1a is used for all subsequent parts and sub-parts. The code is not pasted in this document again for sake of brevity, but it is present in the attached R script. For questions that involve multiple roots, extra lines for `geom_vline` were added.

1.a)

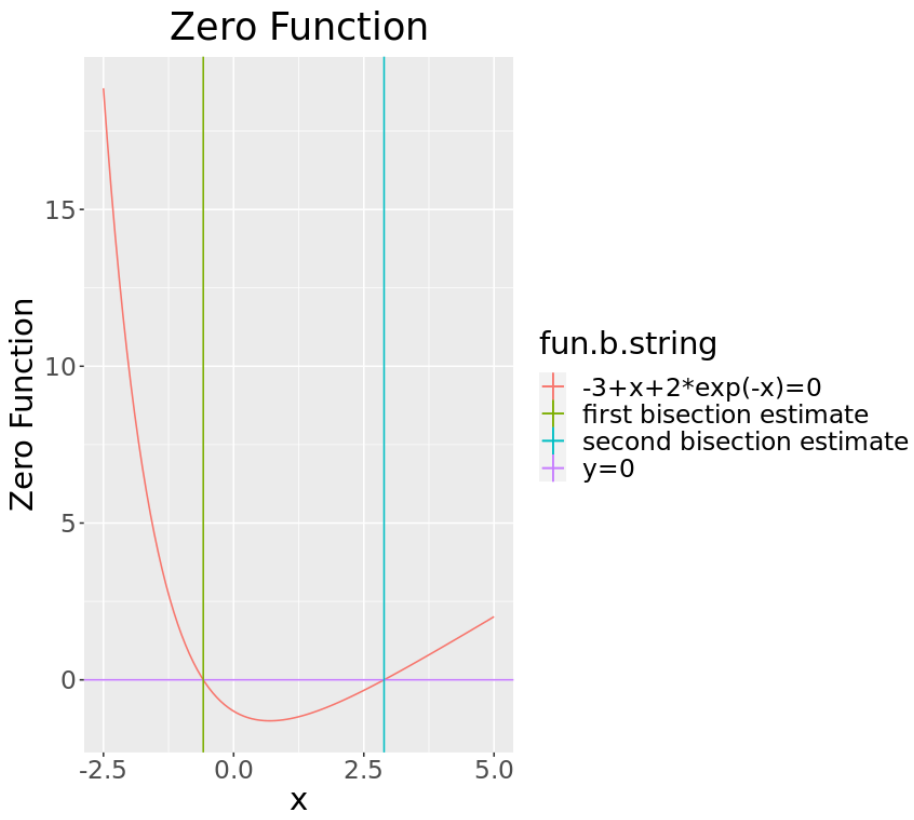
```
# Solve 1a
fun.a <- function(x) {x^2-2}
fun.a.string <- "x^2-2=0"
sqrt2.estimate <- findZeroBisect(fun.a,1,2,1e-6)

# Plot 1a
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)
ggplot(data.frame(x=seq(1,2,.1)), aes(x)) +
  stat_function(fun=fun.a, aes(col=fun.string)) +
  geom_hline(aes(yintercept=0, col = "y=0"), show.legend=TRUE)+
  geom_vline(aes(xintercept=sqrt2.estimate[1],
                 col = "bisection estimate"), show.legend=TRUE)+
  ggtitle("Zero Function") +
  xlab("x") + ylab("Zero Function") +
  theme(text = element_text(size=20), plot.title = element_text(hjust = 0.5))
```




```
> sqrt2.estimate
[1] 1.414214e+00      2.687177e-07          2.000000e+01
```

1.b)

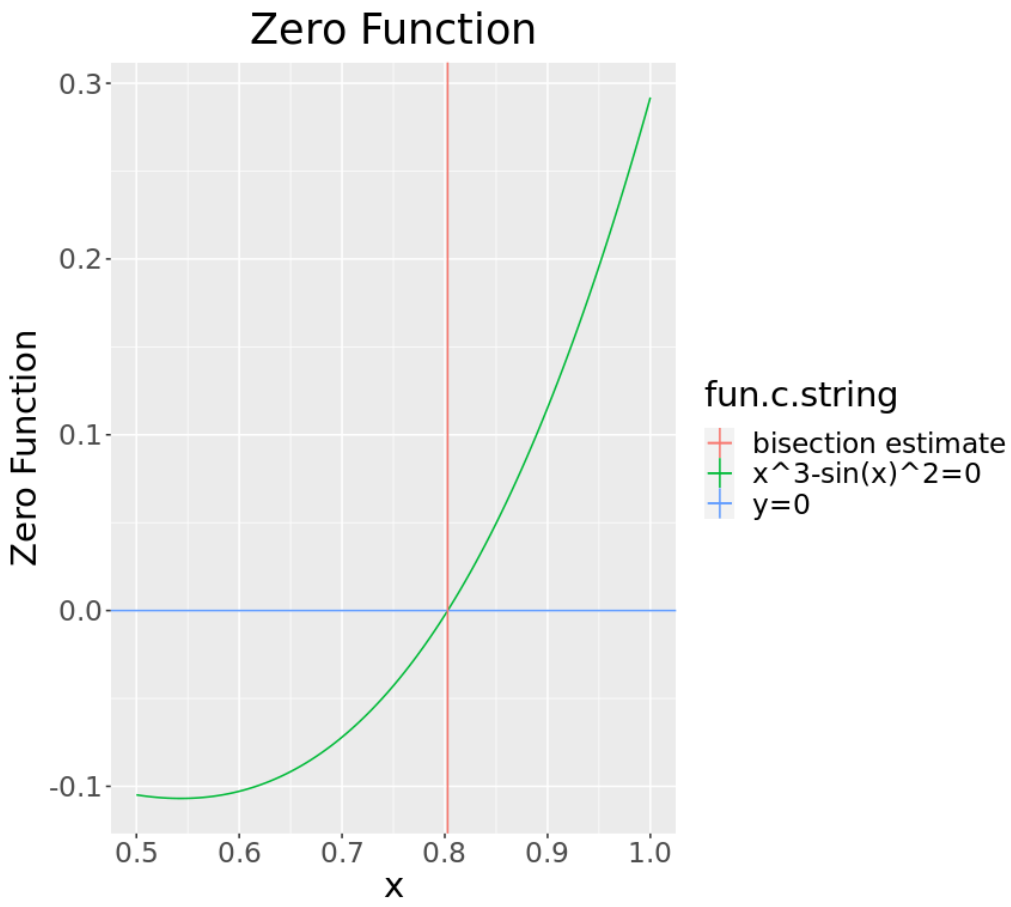
```
fun.b <- function(x) {-3+x+2*exp(-x)}
fun.b.string <- "-3+x+2*exp(-x)=0"
fun.b.estimate.left <- findZeroBisect(fun.b,-2.5,0,1e-6)
fun.b.estimate.right <- findZeroBisect(fun.b,1,4,1e-6)
```




```
> fun.b.estimate.left
[1] -5.830741e-01  5.600836e-07     1.900000e+01
> fun.b.estimate.right
[1] 2.888704e+00  8.386990e-07     1.800000e+01
```

1.c)

```
fun.c <- function(x) {x^3-sin(x)^2}
fun.c.string <- "x^3-sin(x)^2=0"
fun.c.estimate <- findZeroBisect(fun.c,0.5,1,1e-6)
```



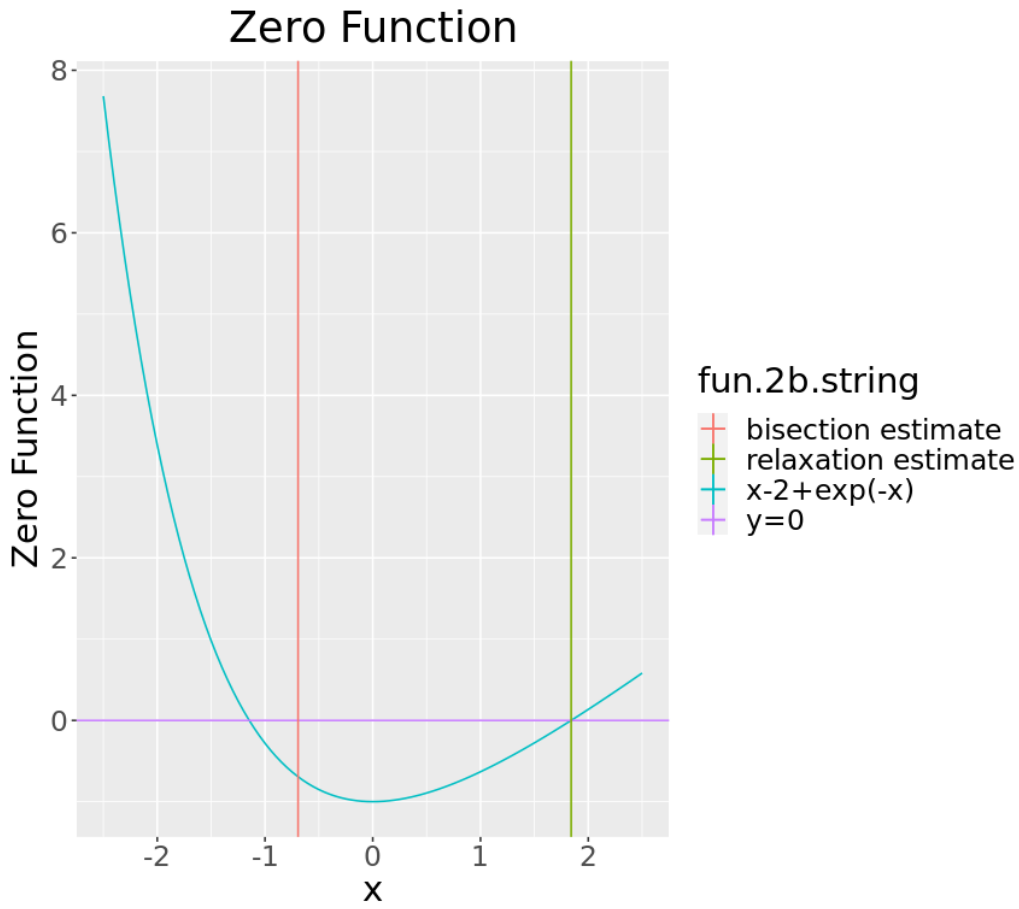

```
> fun.c.estimate
[1]  8.02803e-01 -6.46562e-07  1.50000e+01
```

2.a)

```
## Part 2: Relaxation Method
findZeroRelax <- function(g, x.guess, tol=1e-6, maxsteps=1e6){
  steps <- 0
  x.old <- x.guess
  isConverged <- FALSE
  while (!isConverged){
    x.new <- g(x.old)
    if(steps >= maxsteps || (abs(x.new-x.old)<tol)){
      isConverged <- TRUE
    }
    else{
      x.old <- x.new
    }
  steps <- steps + 1
  }
  return(c(x.new, g(x.new), steps))
}
```

2.b)

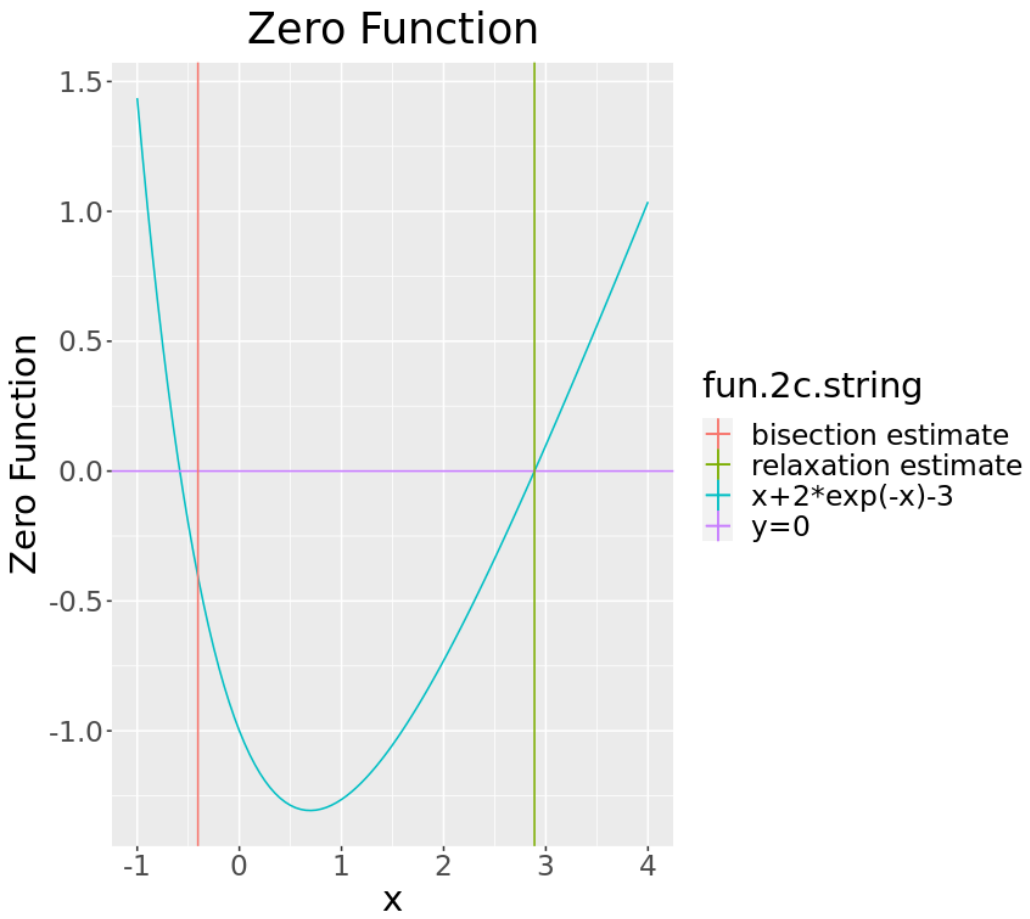
```
fun.2b <- function(x) {2-exp(-x)}
fun.2b.fx <- function(x) {x-2+exp(-x)}
fun.2b.string <- "x-2+exp(-x)"
fun.2b.estimate <- findZeroRelax(fun.2b,0)
# Find the other root with bisection
fun.2b.bisect <- findZeroBisect(fun.2b,-2,0,1e-6)
```




```
> fun.2b.estimate
[1]  1.841406  1.841406 10.000000
> fun.2b.bisect
[1] -6.931477e-01 -9.574839e-07  1.900000e+01
```

2.c)

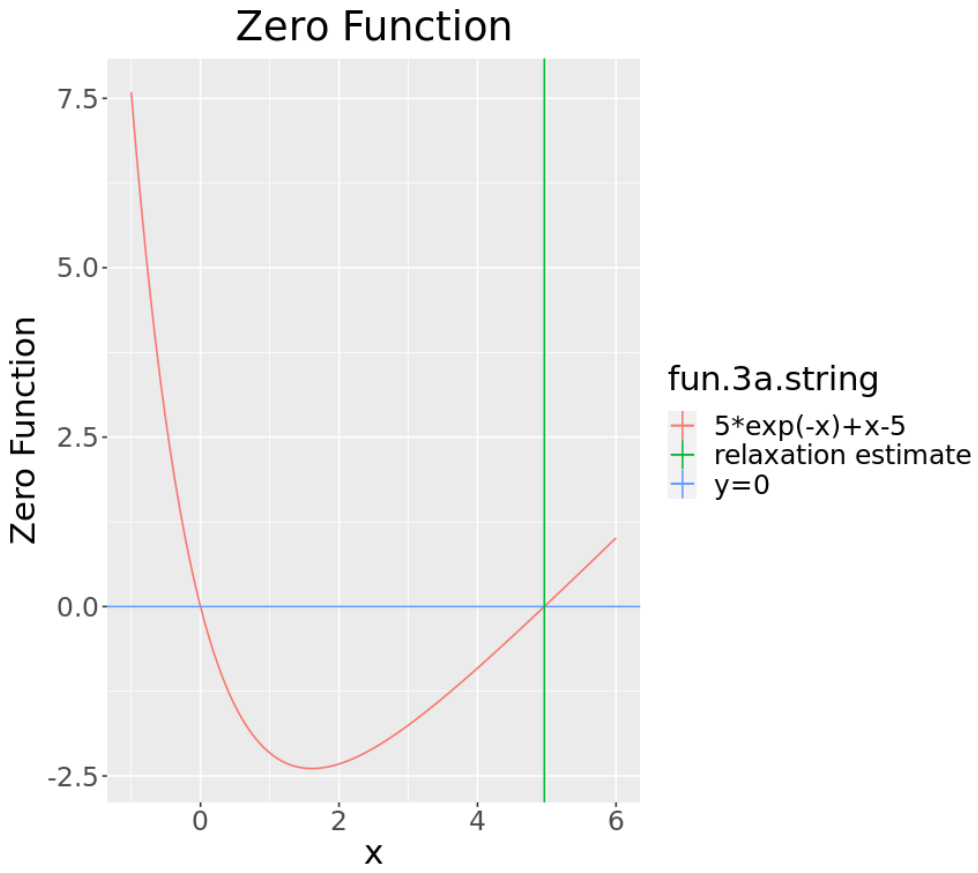
```
# Solve 2c
fun.2c <- function(x) {3-2*exp(-x)}
fun.2c.fx <- function(x) {x+2*exp(-x)-3}
fun.2c.string <- "x+2*exp(-x)-3"
fun.2c.estimate <- findZeroRelax(fun.2c,-.5)
# Find the other root with bisection
fun.2c.bisect <- findZeroBisect(fun.2c,-1,0,1e-6)
```




```
> fun.2c.estimate
[1]  2.888703  2.888703 11.000000
> fun.2c.bisect
[1] -4.054651e-01 -5.378830e-08  1.900000e+01
```

3.a)

```
fun.3a.fx <- function(x) {5*exp(-x)+x-5}
fun.3a <- function(x) {5-5*exp(-x)}
fun.3a.string <- "5*exp(-x)+x-5"
fun.3a.estimate <- findZeroRelax(fun.3a,4)
```



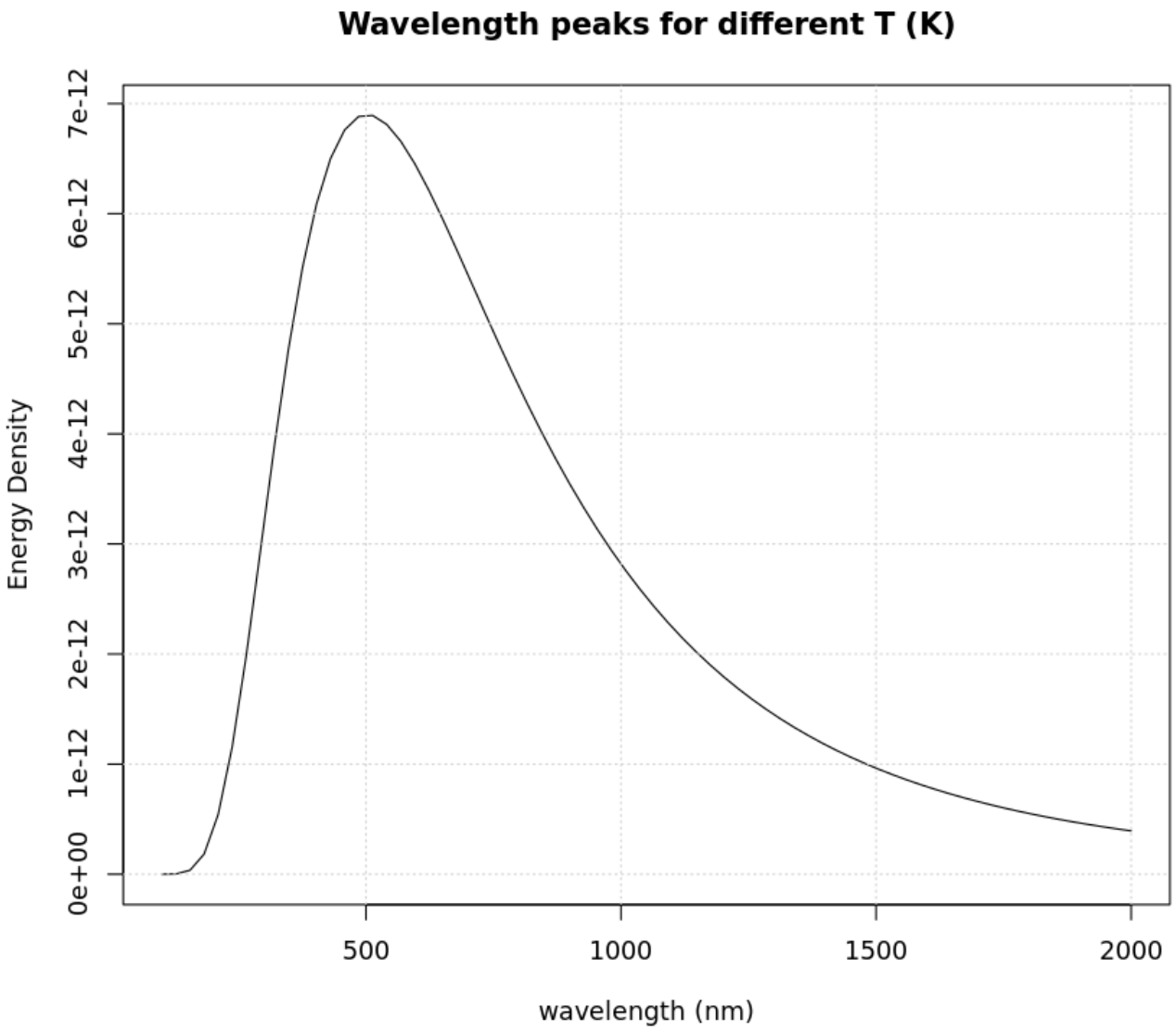

```
> fun.3a.estimate
[1] 4.965114      4.965114     6.000000
```

3.b)

```
peak.wavelength <- (1239.84)/((8.61733e-5)*(fun.3a.estimate[1])*(5778))
> peak.wavelength
[1] 501.5176
Green
```

3.c)

```
# define I_Planck function
I_Planck <- function(wavelength, temp){
  intensity =
((8*pi*1.23984e3)/(wavelength^5))/(exp(1.23984e3/(wavelength*temp*8.61733e-5))-1)
  return(intensity)
}

lam.vec <- seq(100,2000,len=70)
plot(lam.vec, I_Planck(lam.vec,5778), type = "l", lty = 1,
     xlab="wavelength (nm)", ylab="Energy Density",
     main="Wavelength peaks for different T (K)")
grid(NULL,NULL)
```

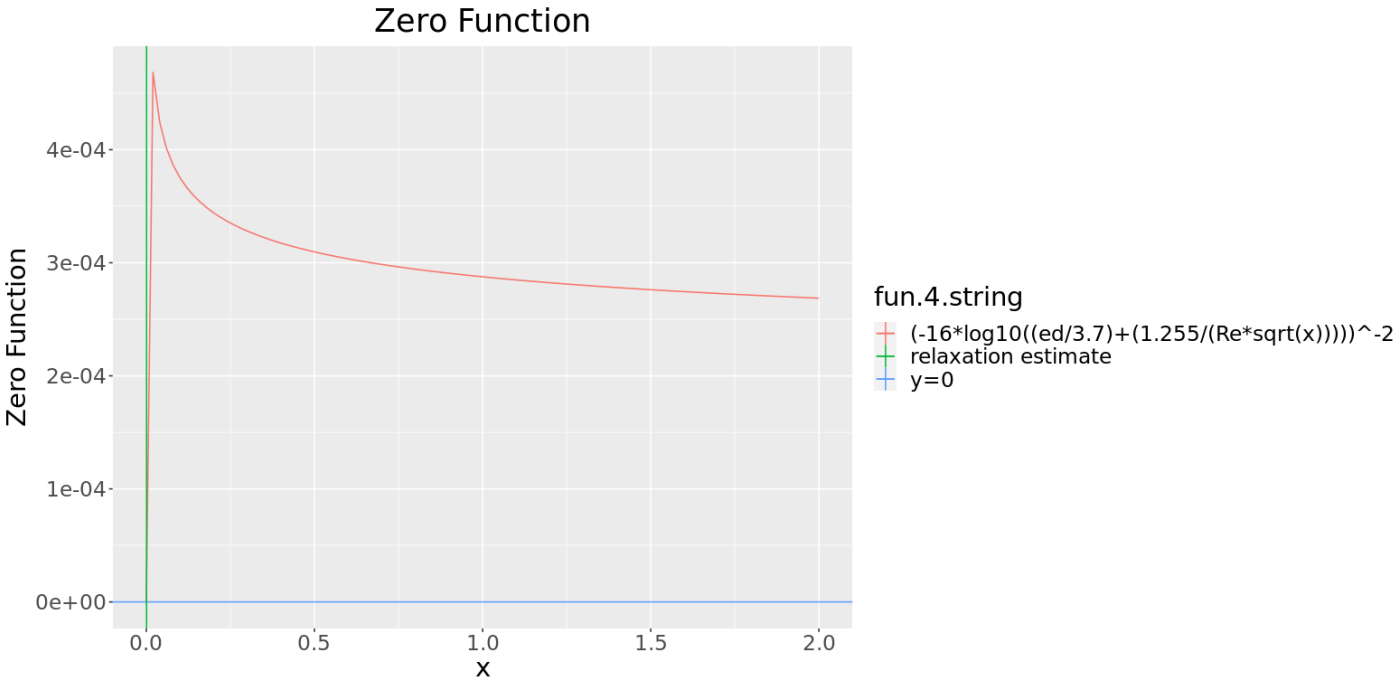


4.)

```
# Solve
implicitFF <- function(Re,ed,x.guess,tol=1e-6){
  if (Re<2000){
    f0 <- 16/Re
  } else {
    fun.4 <- function(x) {(-16*log10((ed/3.7)+(1.255/(Re*sqrt(x)))))^-2}
    fun.4.string <- "(-16*log10((ed/3.7)+(1.255/(Re*sqrt(x)))))^-2"
    fun.4.estimate <- findZeroRelax(fun.4,x.guess)
    f0 <- fun.4.estimate[1]
  }

  # Plot
  func.plot <- ggplot(data.frame(x=seq(0,2,.1)), aes(x)) +
    stat_function(fun=fun.4, aes(col=fun.4.string)) +
    geom_hline(aes(yintercept=0, col = "y=0"), show.legend=TRUE)+
    geom_vline(aes(xintercept=fun.4.estimate[1],
                col = "relaxation estimate"), show.legend=TRUE)+
    ggtitle("Zero Function") +
    xlab("x") + ylab("Zero Function") +
    theme(text = element_text(size=20), plot.title = element_text(hjust = 0.5))
  print(func.plot)
  return(f0)
}
```

4.a)

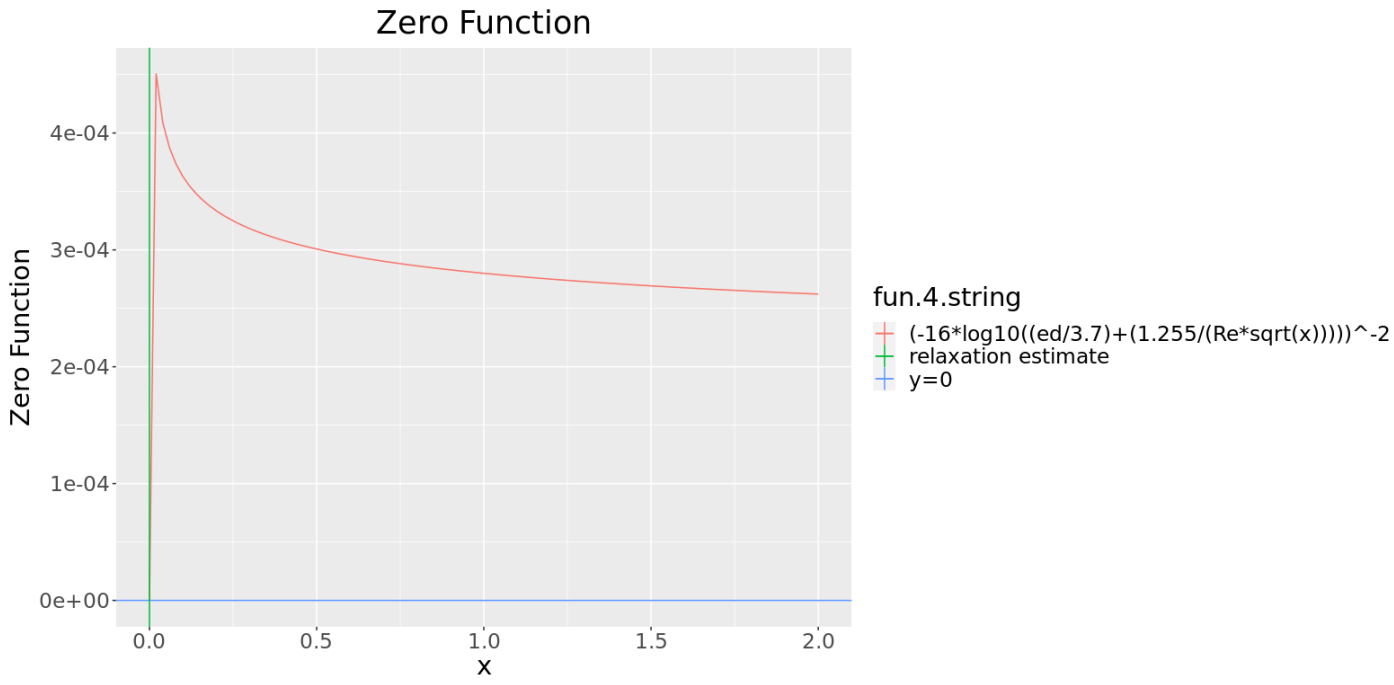
```
fa.estimate <- implicitFF(7000, 0.0001, 0.1)
```




```
> fa.estimate
[1] 0.0008082328
```

4.b)

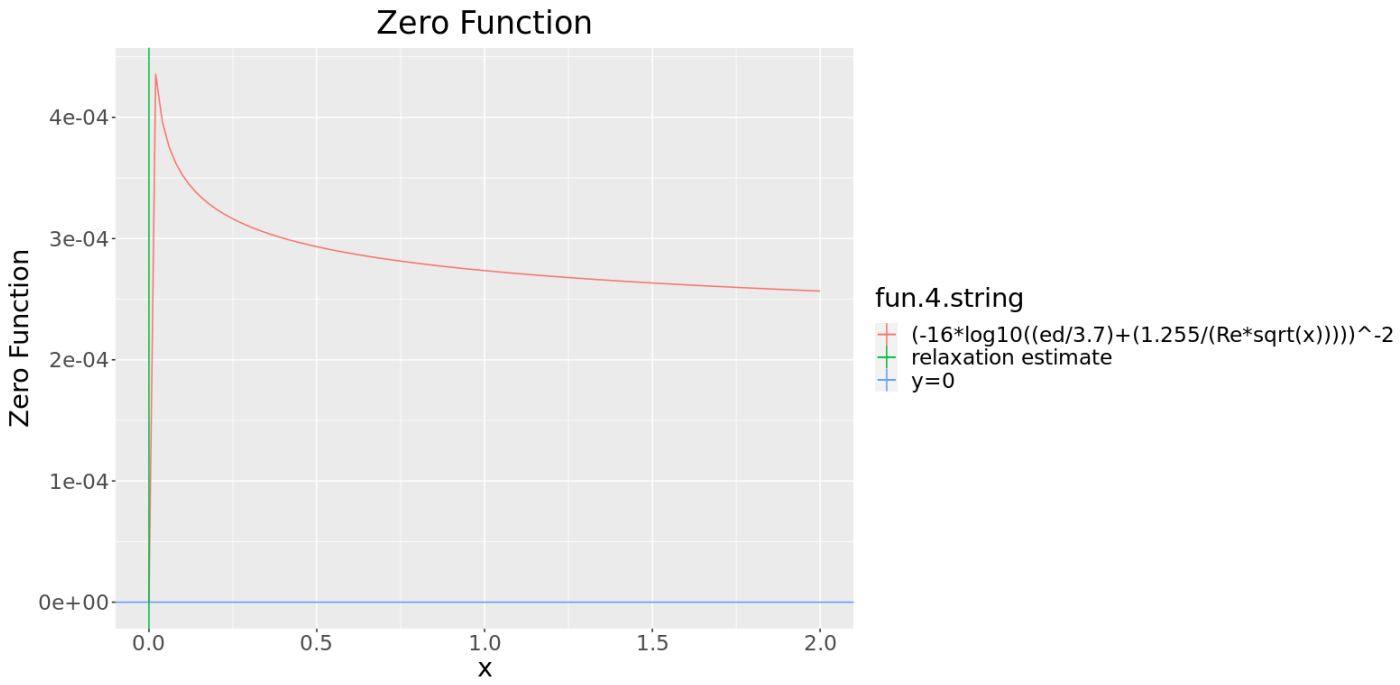
```
fb.estimate <- implicitFF(8000, 0.0001, 0)
```



> fb.estimate
[1] 0

4.c)

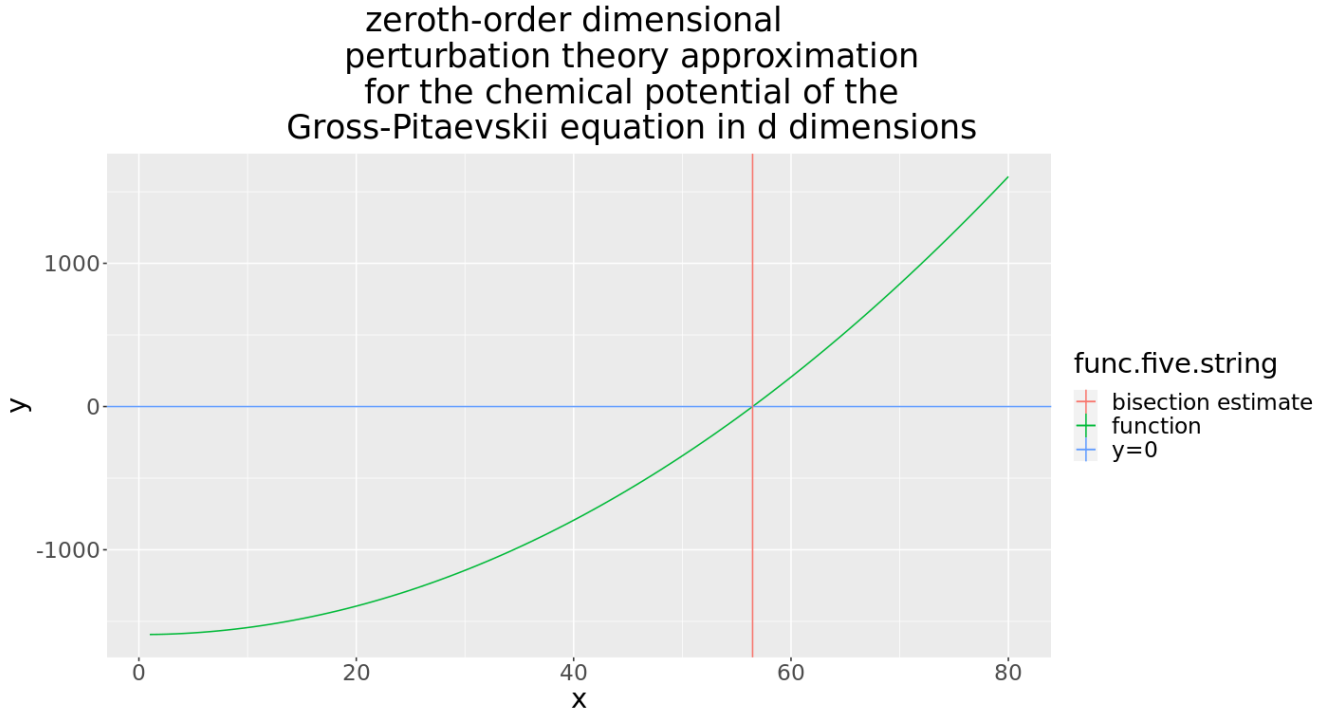
```
fc.estimate <- implicitFF(9000, 0.0001, 0)
```




```
> fc.estimate
[1] 0
```

5.a)

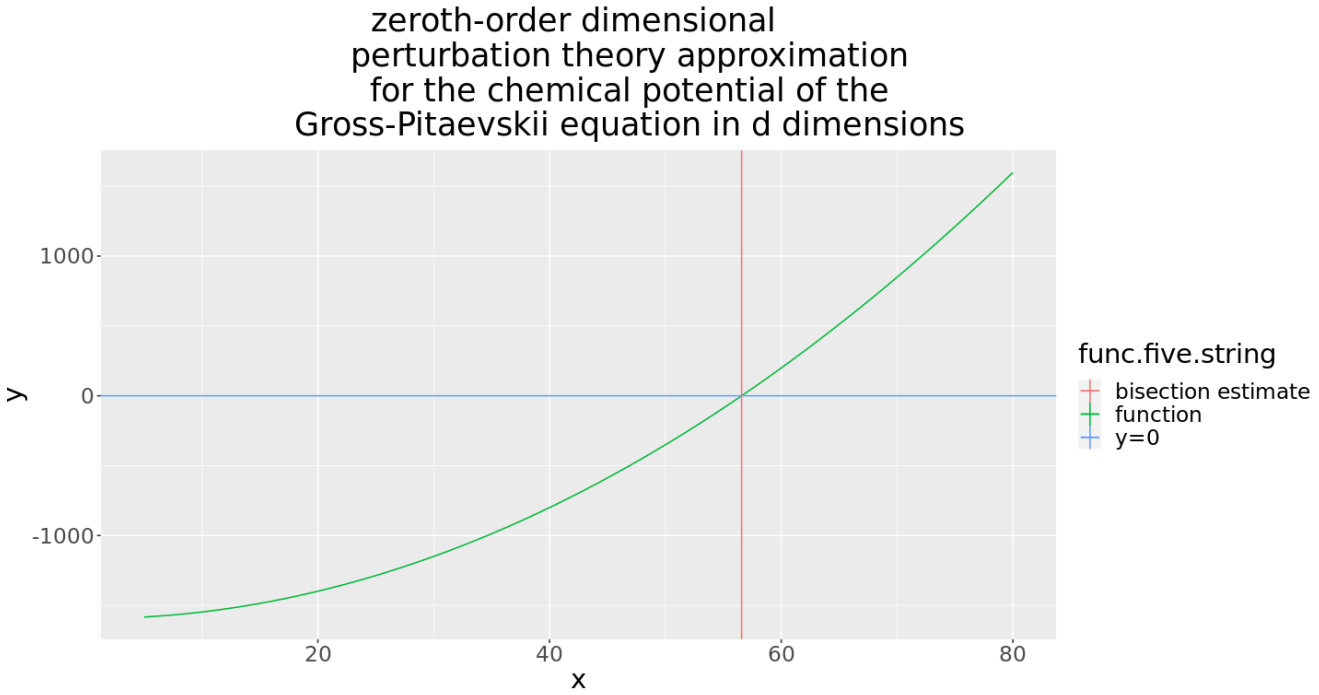
```
# a:
g <- 10000
d <- 2
l <- 0

func.five <- function(u){
  (-10000/(2*pi)) +
  (u/2) *
  sqrt((u^2)-((d+2*l)^2)/4) +
  (((d+2*l)^2)/16) *
  log((u-sqrt((u^2)-((d+2*l)^2)/4))/(u+sqrt((u^2)-((d+2*l)^2)/4)))
}

func.five.string <- "function"
func.five.estimate <- findZeroBisect(func.five,50,75,1e-6)
```




```
> func.five.estimate
[1] 5.646526e+01        2.054011e-07        2.800000e+01
```

5.b)

```
# b:
g <- 10000
d <- 2
l <- 1

func.five <- function(u){
  (-10000/(2*pi)) +
    (u/2) *
    sqrt((u^2)-((d+2*l)^2)/4) +
    (((d+2*l)^2)/16) *
    log((u-sqrt((u^2)-((d+2*l)^2)/4))/(u+sqrt((u^2)-((d+2*l)^2)/4)))
}

func.five.string <- "function"
func.five.estimate <- findZeroBisect(func.five,50,75,1e-6)
```




```
> func.five.estimate
[1]  5.657951e+01      -4.968289e-07     2.800000e+01
```