Lista 1, aula 1 em 16/9

Henrique Ap. Laureano
laureano@ufpr.br ∧ www.leg.ufpr.br/~henrique/

September 21, 2019

Contents

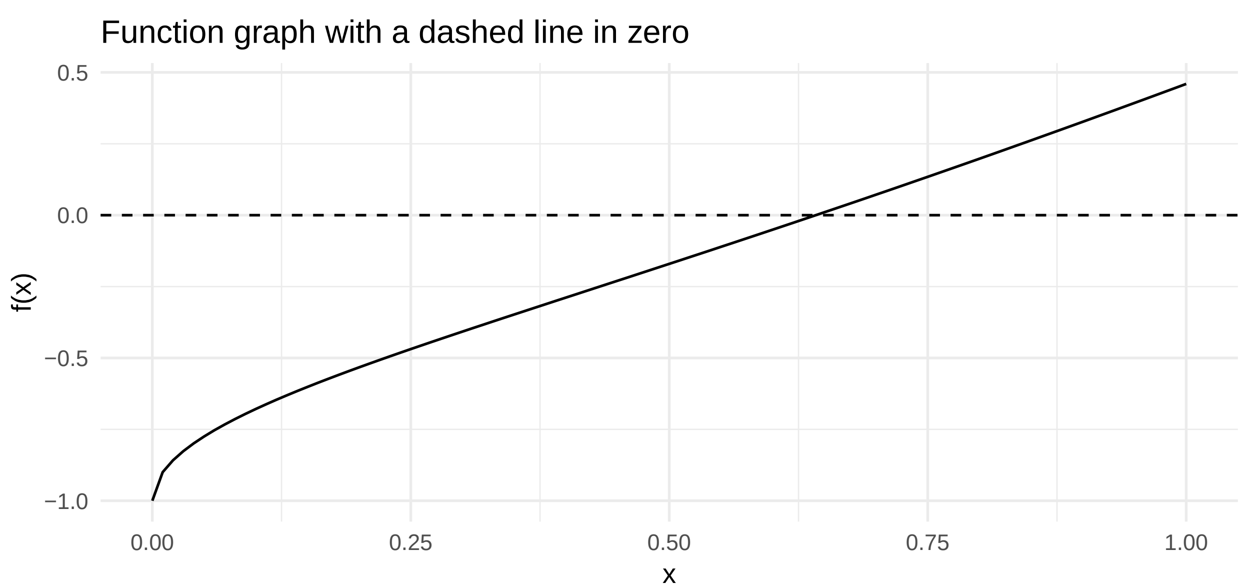
Exercise 1

Determine x_3 pelo método da Bisseção para

$$f(x) = \sqrt{x} - \cos x \quad \text{em} \quad [0, 1].$$

<r code>

```r
f1 <- function(x) { sqrt(x) - cos(x) }

library(ggplot2)

ggplot(data.frame(x = c(0, 1)), aes(x = x)) +
    theme_minimal() +
    stat_function(fun = f1) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    labs(y = "f(x)", title = "Function graph with a dashed line in zero")
```




```r
bisection <- function(f, a, b, kmax = 100, tol = 1e-3) {
    k <- 0
    data <- matrix(NA, nrow = kmax, ncol = 3)
    dimnames(data) <- list(seq(kmax), c("a", "x", "b"))
    fa <- f(a)
    x <- (a + b)/2
    while(b - a > tol & k < kmax) {
        fx <- f(x)
        if(fa * fx < 0) {
            b <- x
            fb <- fx
        } else {
```

```
            a <- x
            fa <- fx
        }
        k <- k + 1
        x <- (a + b)/2
        data[k, ] <- c(a, x, b)
    }
    return(data[seq(k), ])
}
bisection(f1, a = 0, b = 1, kmax = 3)
```

```
      a      x    b
1 0.500 0.7500 1.00
2 0.500 0.6250 0.75
3 0.625 0.6875 0.75
```

Exercise 2

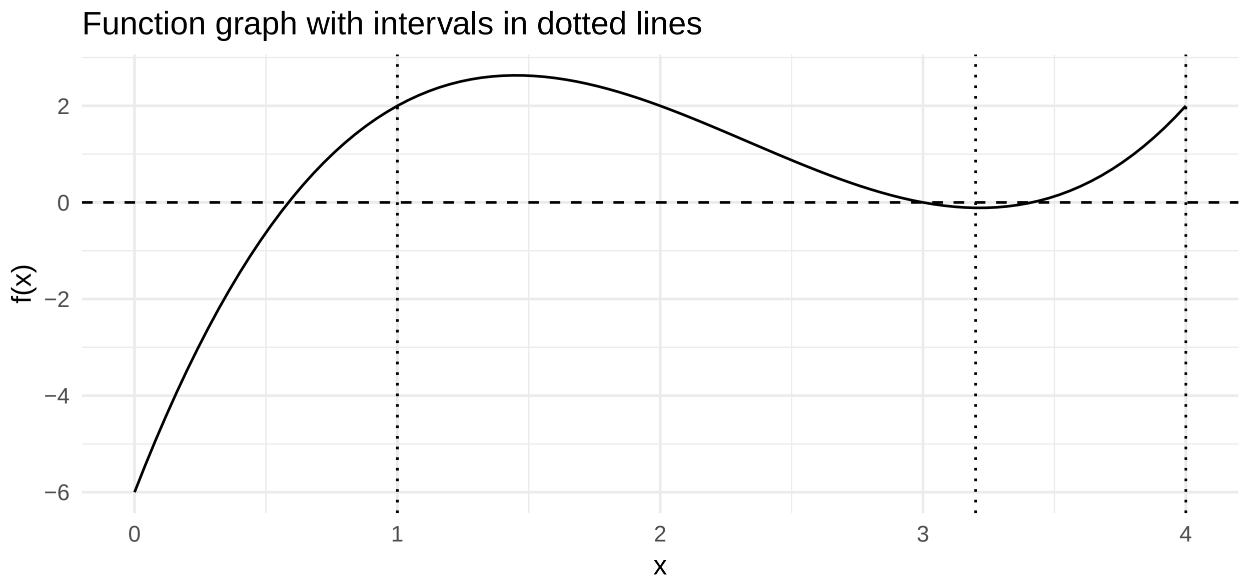
Encontrar as soluções para

$$x^3 - 7x^2 + 14x - 6 = 0,$$

com precisão de 10^{-2}, utilizando o método da Bisseção nos seguintes intervalos.

<r code>

```
f2 <- function(x) { x^3 - 7 * x^2 + 14 * x - 6 }

ggplot(data.frame(x = c(0, 4)), aes(x = x)) +
    theme_minimal() +
    stat_function(fun = f2) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    geom_vline(xintercept = c(1, 3.2, 4), linetype = "dotted") +
    labs(y = "f(x)",
         title = "Function graph with intervals in dotted lines")
```



(a) [0, 1]

<r code>

```
(ex2_a <- bisection(f2, a = 0, b = 1, tol = 1e-2))
```

```
         a         x         b
1 0.500000 0.7500000 1.0000000
2 0.500000 0.6250000 0.7500000
3 0.500000 0.5625000 0.6250000
4 0.562500 0.5937500 0.6250000
5 0.562500 0.5781250 0.5937500
6 0.578125 0.5859375 0.5937500
7 0.578125 0.5820312 0.5859375
```

```
# solucao com precisao maior do que 0.01
ex2_a[nrow(ex2_a), "b"] - ex2_a[nrow(ex2_a), "a"]
```

```
[1] 0.0078125
```

(b) [1, 3.2]

<r code>

```
(ex2_b <- bisection(f2, a = 1, b = 3.2, tol = 1e-2))
```

```
        a        x        b
1 2.10000 2.650000 3.200000
2 2.65000 2.925000 3.200000
3 2.92500 3.062500 3.200000
```

```
4 2.92500 2.993750 3.062500
5 2.99375 3.028125 3.062500
6 2.99375 3.010938 3.028125
7 2.99375 3.002344 3.010938
8 2.99375 2.998047 3.002344
```

```r
# solucao com precisao maior do que 0.01
ex2_b[nrow(ex2_b), "b"] - ex2_b[nrow(ex2_b), "a"]
```

```
[1] 0.00859375
```

(c) [3.2, 4]

<r code>

```r
(ex2_c <- bisection(f2, a = 3.2, b = 4, tol = 1e-2))
```

```
       a        x       b
1 3.2000 3.400000 3.60000
2 3.4000 3.500000 3.60000
3 3.4000 3.450000 3.50000
4 3.4000 3.425000 3.45000
5 3.4000 3.412500 3.42500
6 3.4125 3.418750 3.42500
7 3.4125 3.415625 3.41875
```

```r
# solucao com precisao maior do que 0.01
ex2_c[nrow(ex2_c), "b"] - ex2_c[nrow(ex2_c), "a"]
```

```
[1] 0.00625
```

Exercise 3

(b)

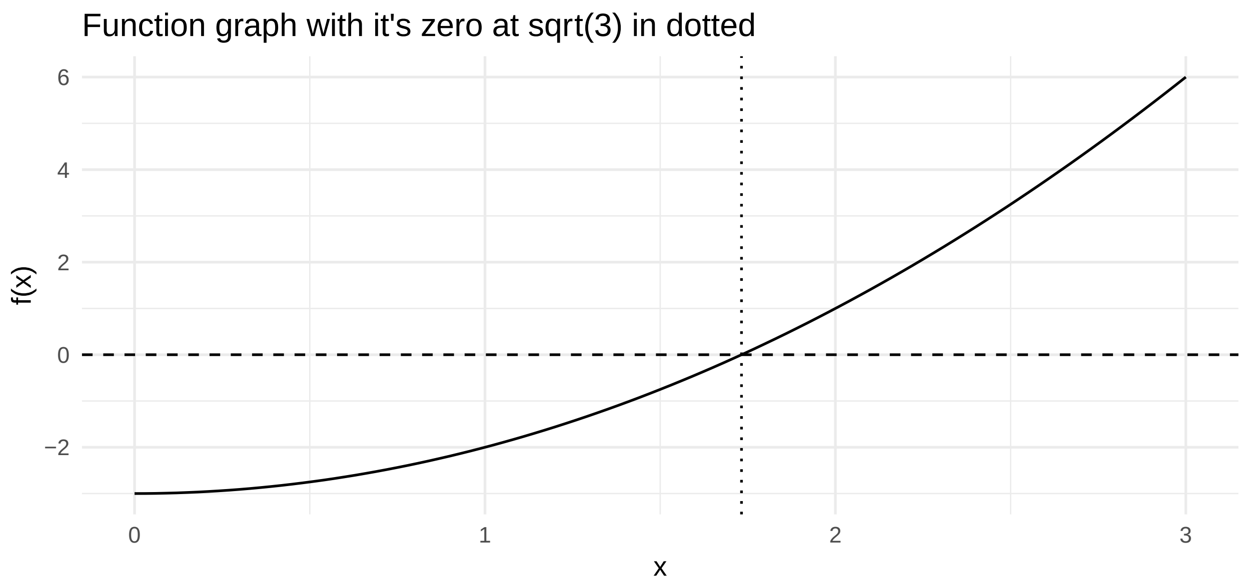
Determinar $\sqrt{3}$ com precisão 10^{-4}, utilizando o método da Bisseção. Isso implica em solucionar a seguinte função:

$$f(x) = x^2 - 3.$$

<r code>

```r
f3 <- function(x) { x^2 - 3 }

ggplot(data.frame(x = c(0, 3)), aes(x = x)) +
```

```
  theme_minimal() +
  stat_function(fun = f3) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_vline(xintercept = sqrt(3), linetype = "dotted") +
  labs(y = "f(x)",
       title = "Function graph with it's zero at sqrt(3) in dotted")
```




```
# 14 iteracoes, como calculado na letra (a)
bisection(f3, a = 1, b = 2, tol = 1e-4)
```

```
          a        x        b
1  1.500000 1.750000 2.000000
2  1.500000 1.625000 1.750000
3  1.625000 1.687500 1.750000
4  1.687500 1.718750 1.750000
5  1.718750 1.734375 1.750000
6  1.718750 1.726562 1.734375
7  1.726562 1.730469 1.734375
8  1.730469 1.732422 1.734375
9  1.730469 1.731445 1.732422
10 1.731445 1.731934 1.732422
11 1.731934 1.732178 1.732422
12 1.731934 1.732056 1.732178
13 1.731934 1.731995 1.732056
14 1.731995 1.732025 1.732056
```

```
# 15 iteracoes, como calculado na letra (a)
bisection(f3, a = 0, b = 3, tol = 1e-4)
```

```
          a        x        b
```

```
1  1.500000 2.250000 3.000000
2  1.500000 1.875000 2.250000
3  1.500000 1.687500 1.875000
4  1.687500 1.781250 1.875000
5  1.687500 1.734375 1.781250
6  1.687500 1.710938 1.734375
7  1.710938 1.722656 1.734375
8  1.722656 1.728516 1.734375
9  1.728516 1.731445 1.734375
10 1.731445 1.732910 1.734375
11 1.731445 1.732178 1.732910
12 1.731445 1.731812 1.732178
13 1.731812 1.731995 1.732178
14 1.731995 1.732086 1.732178
15 1.731995 1.732040 1.732086
```

Exercise 4

Aplique o Algoritmo do método de Newton para todos os exercícios que foram resolvidos pelo método da Bisseção. Compare os resultados.

<r code>

```
newton <- function(f, fline, init, kmax = 100, tol = 1e-3) {
    xs <- numeric(kmax)
    xs[1] <- init - f(init)/fline(init)
    xs[2] <- xs[1] - f(xs[1])/fline(xs[1])
    k <- 2
    while(abs(diff(xs[(k - 1):k]))/abs(xs[k]) > tol & k < kmax) {
        k <- k + 1
        xs[k] <- xs[k - 1] - f(xs[k - 1])/fline(xs[k - 1])
    }
    return(xs[seq(k)])
}
```

Exercise 1

<r code>

```
f1line <- function(x) { 1/(2 * sqrt(x)) + sin(x) }

# com uma tolerancia de 0.001, com Newton ja' obtemos convergencia em
# 3 iteracoes
newton(f1, f1line, init = 1)
```

```
[1] 0.6573182 0.6417461 0.6417144
```

```r
# com o metodo da Bissecao precisamos de 10 iteracoes
bisection(f1, a = 0, b = 1)
```

```
           a         x         b
1  0.5000000 0.7500000 1.0000000
2  0.5000000 0.6250000 0.7500000
3  0.6250000 0.6875000 0.7500000
4  0.6250000 0.6562500 0.6875000
5  0.6250000 0.6406250 0.6562500
6  0.6406250 0.6484375 0.6562500
7  0.6406250 0.6445312 0.6484375
8  0.6406250 0.6425781 0.6445312
9  0.6406250 0.6416016 0.6425781
10 0.6416016 0.6420898 0.6425781
```

Exercise 2. (a) [0, 1]

<r code>

```r
f2line <- function(x) { 3 * x^2 - 14 * x +  14 }

# com uma tolerancia de 0.01, com Newton obtemos convergencia em:
# 2 iteracoes, usando um chute inicial em 0.5
newton(f2, f2line, tol = 1e-2, init = .5)
```

```
[1] 0.5806452 0.5857663
```

```r
# 4 iteracoes, usando um chute inicial em 1
newton(f2, f2line, tol = 1e-2, init = 1)
```

```
[1] 0.3333333 0.5478927 0.5847302 0.5857856
```

```r
# com o metodo da Bissecao precisamos de 7 iteracoes
bisection(f2, a = 0, b = 1, tol = 1e-2)
```

```
         a         x         b
1 0.500000 0.7500000 1.0000000
2 0.500000 0.6250000 0.7500000
3 0.500000 0.5625000 0.6250000
4 0.562500 0.5937500 0.6250000
5 0.562500 0.5781250 0.5937500
6 0.578125 0.5859375 0.5937500
7 0.578125 0.5820312 0.5859375
```

Exercise 2. (b) [1, 3.2]

```r
# com uma tolerancia de 0.01, com Newton obtemos convergencia em:
# 10 iteracoes, usando um chute inicial em 1.5
# contudo, com tal chute obtemos um diferente zero da funcao
newton(f2, f2line, tol = 1e-2, init = 1.5)
```

<r code>

```
 [1] 12.000000  8.827338  6.733875  5.367206  4.492787  3.953119  3.640792
 [8]  3.482097  3.423790  3.414456
```

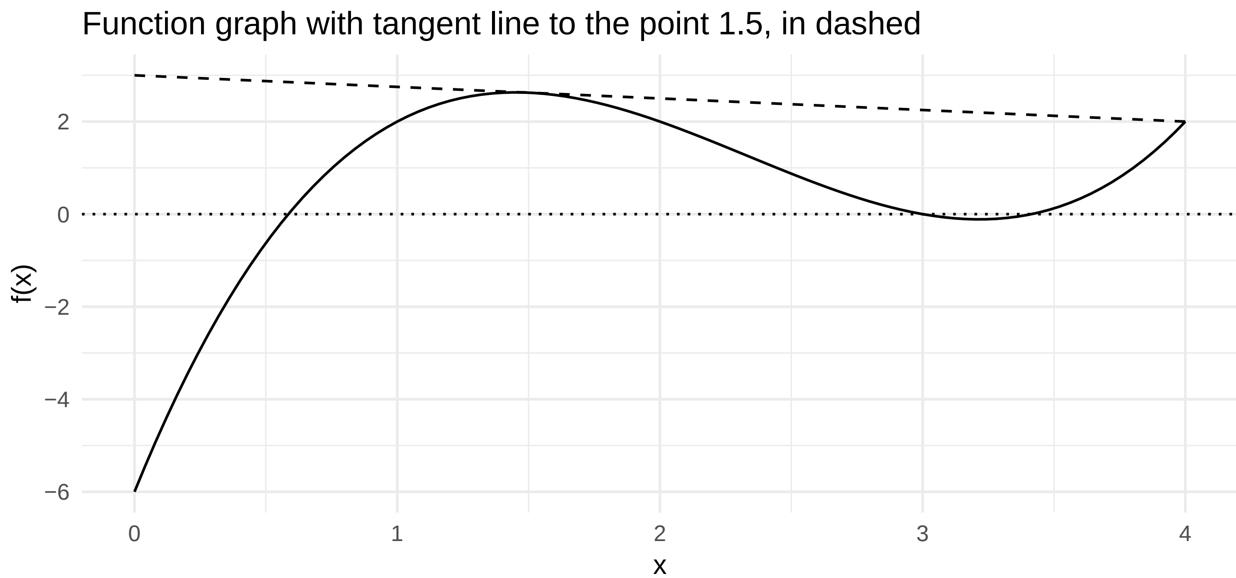
```r
# 2 iteracoes, usando um chute inicial em 2
newton(f2, f2line, tol = 1e-2, init = 2)
```

```
[1] 3 3
```

```r
# com o metodo da Bissecao precisamos de 8 iteracoes
bisection(f2, a = 1, b = 3.2, tol = 1e-2)
```

```
        a        x        b
1 2.10000 2.650000 3.200000
2 2.65000 2.925000 3.200000
3 2.92500 3.062500 3.200000
4 2.92500 2.993750 3.062500
5 2.99375 3.028125 3.062500
6 2.99375 3.010938 3.028125
7 2.99375 3.002344 3.010938
8 2.99375 2.998047 3.002344
```

```r
ggplot(data.frame(x = c(0, 4)), aes(x = x)) +
    theme_minimal() +
    stat_function(fun = f2) +
    stat_function(fun = function(t) { 3 - t/4 }, linetype = "dashed") +
    geom_hline(yintercept = 0, linetype = "dotted") +
    labs(y = "f(x)",
         title = "Function graph with tangent line to the point 1.5, in dashed")
```



No ponto 1.5, um máximo local, a curvatura é alta, fazendo com que a reta tangente à esse ponto corte o eixo x num ponto muito a frente, $x = 12$. Ao final da primeira iteração estamos em 12, e assim o algorimo começa a retornar, atingindo o ponto de zero mais próximo, 3.414456.

Exercise 2. (c) [3.2, 4]

<r code>

```
# com uma tolerancia de 0.01, com Newton obtemos convergencia em:
# 2 iteracoes, usando um chute inicial em 3.5
newton(f2, f2line, tol = 1e-2, init = 3.5)

[1] 3.428571 3.414747

# 4 iteracoes, usando um chute inicial em 4
newton(f2, f2line, tol = 1e-2, init = 4)

[1] 3.666667 3.493827 3.426846 3.414629

# com o metodo da Bissecao precisamos de 7 iteracoes
bisection(f2, a = 3.2, b = 4, tol = 1e-2)

       a        x       b
1 3.2000 3.400000 3.60000
2 3.4000 3.500000 3.60000
3 3.4000 3.450000 3.50000
4 3.4000 3.425000 3.45000
5 3.4000 3.412500 3.42500
6 3.4125 3.418750 3.42500
7 3.4125 3.415625 3.41875
```

Exercise 3

<r code>

```
f3line <- function(x) { 2 * x }

# com uma tolerancia de 0.0001, com Newton obtemos convergencia em:
# 4 iteracoes, usando um chute inicial em 1
newton(f3, f3line, tol = 1e-4, init = 1)
```

```
[1] 2.000000 1.750000 1.732143 1.732051
```

```
# 3 iteracoes, usando um chute inicial em 2
newton(f3, f3line, tol = 1e-4, init = 2)
```

```
[1] 1.750000 1.732143 1.732051
```

```
# com o metodo da Bissecao precisamos de 14 iteracoes,
# usando o intervalo [1, 2]
bisection(f3, a = 1, b = 2, tol = 1e-4)
```

```
          a        x        b
1  1.500000 1.750000 2.000000
2  1.500000 1.625000 1.750000
3  1.625000 1.687500 1.750000
4  1.687500 1.718750 1.750000
5  1.718750 1.734375 1.750000
6  1.718750 1.726562 1.734375
7  1.726562 1.730469 1.734375
8  1.730469 1.732422 1.734375
9  1.730469 1.731445 1.732422
10 1.731445 1.731934 1.732422
11 1.731934 1.732178 1.732422
12 1.731934 1.732056 1.732178
13 1.731934 1.731995 1.732056
14 1.731995 1.732025 1.732056
```

Exercise 5

Aplique o Algoritmo do método Quase-Newton (Secante) para todos os exercícios anteriores. Compare os resultados.

<r code>

```
secant <- function(f, inits, kmax = 100, tol = 1e-3) {
    a <- inits[1] ; fa <- f(a)
    b <- inits[2] ; fb <- f(b)
    xs <- numeric(kmax)
```

```
    xs[1] <- (fb * a - b * fa)/(fb - fa) ; f1 <- f(xs[1])
    xs[2] <- (f1 * b - xs[1] * fb)/(f1 - fb)
    k <- 2
    while(abs(diff(xs[(k - 1):k]))/abs(xs[k]) > tol & k < kmax) {
        k <- k + 1
        f1 <- f(xs[k - 1])
        f2 <- f(xs[k - 2])
        xs[k] <- (f1 * xs[k - 2] - xs[k - 1] * f2)/(f1 - f2)
    }
    return(xs[seq(k)])
}
```

Exercise 1

<r code>

```
# usando diferentes pontos iniciais pra ver como isso impacta o resultado

## diferentes intervalos, mas de mesmo tamanho
secant(f1, inits = c(0, .25))

[1] 0.4707322 0.6423949 0.6417013 0.6417144

secant(f1, inits = c(.25, .5))

[1] 0.6428073 0.6416964 0.6417144

secant(f1, inits = c(.5, .75))

[1] 0.6398203 0.6416871 0.6417144

secant(f1, inits = c(.75, 1))

[1] 0.6467789 0.6419525 0.6417145

## agora, considerando intervalos mais curtos
secant(f1, inits = c(.2, .25))

[1] 0.6166784 0.6422015 0.6417128

secant(f1, inits = c(0, .05))

[1] 0.2223640 0.5424335 0.6424509 0.6417055 0.6417144

### no melhor cenario levamos 3 iteracoes, no pior, levamos 5

# com o metodo de Newton levamos 3 iteracoes
newton(f1, f1line, init = 1)

[1] 0.6573182 0.6417461 0.6417144
```

Exercise 2. (a)[0, 1]

<r code>

```r
secant(f2, inits = c(.5, .55), tol = 1e-2)
```

```
[1] 0.5835841 0.5857272
```

```r
secant(f2, inits = c(.5, .6), tol = 1e-2)
```

```
[1] 0.5866852 0.5857765
```

```r
secant(f2, inits = c(.5, .75), tol = 1e-2)
```

```
[1] 0.5970874 0.5842026 0.5858003
```

```r
newton(f2, f2line, tol = 1e-2, init = .5)
```

```
[1] 0.5806452 0.5857663
```

Exercise 2. (b)[1, 3.2]

<r code>

```r
# para diferentes conjuntos de pontos iniciais, chegamos em diferentes
# zeros, dado os pontos que as retas secantes atingem no eixo x
secant(f2, inits = c(1.2, 1.25), tol = 1e-2)
```

```
[1] -0.6099815  1.0147183  0.8437260  0.4453280  0.6154363  0.5887888
[7]  0.5857168
```

```r
secant(f2, inits = c(1.7, 1.75), tol = 1e-2)
```

```
[1] 3.731084 4.583585 3.638019 3.578869 3.466785 3.429857 3.416163
```

```r
secant(f2, inits = c(2, 2.05), tol = 1e-2)
```

```
[1] 2.976801 2.988804
```

```r
secant(f2, inits = c(2, 2.25), tol = 1e-2)
```

```
[1] 2.914286 2.963262 2.995205 2.999681
```

```r
secant(f2, inits = c(2, 2.5), tol = 1e-2)
```

```
[1] 2.888889 2.959481 2.993523 2.999531
```

```r
newton(f2, f2line, tol = 1e-2, init = 2)
```

```
[1] 3 3
```

Exercise 2. (c)[3.2, 4]

<r code>

```
secant(f2, inits = c(3.25, 3.5), tol = 1e-2)
```

```
[1] 3.366667 3.403928 3.415792
```

```
secant(f2, inits = c(3.5, 3.75), tol = 1e-2)
```

```
[1] 3.453488 3.433182
```

```
secant(f2, inits = c(3.45, 3.475), tol = 1e-2)
```

```
[1] 3.419083 3.414919
```

```
secant(f2, inits = c(3.5, 3.525), tol = 1e-2)
```

```
[1] 3.431996 3.418357
```

```
newton(f2, f2line, tol = 1e-2, init = 3.5)
```

```
[1] 3.428571 3.414747
```

Exercise 3

<r code>

```
secant(f3, inits = c(.25, .5), tol = 1e-4)
```

```
[1] 4.166667 1.089286 1.434315 1.807885 1.725087 1.731901 1.732051
```

```
secant(f3, inits = c(1, 1.25), tol = 1e-4)
```

```
[1] 1.888889 1.707965 1.731001 1.732058 1.732051
```

```
secant(f3, inits = c(2.25, 2.5), tol = 1e-4)
```

```
[1] 1.815789 1.746951 1.732401 1.732052 1.732051
```

```
secant(f3, inits = c(2.95, 3), tol = 1e-4)
```

```
[1] 1.991597 1.797980 1.736566 1.732135 1.732051
```

```
newton(f3, f3line, tol = 1e-4, init = 2)
```

```
[1] 1.750000 1.732143 1.732051
```

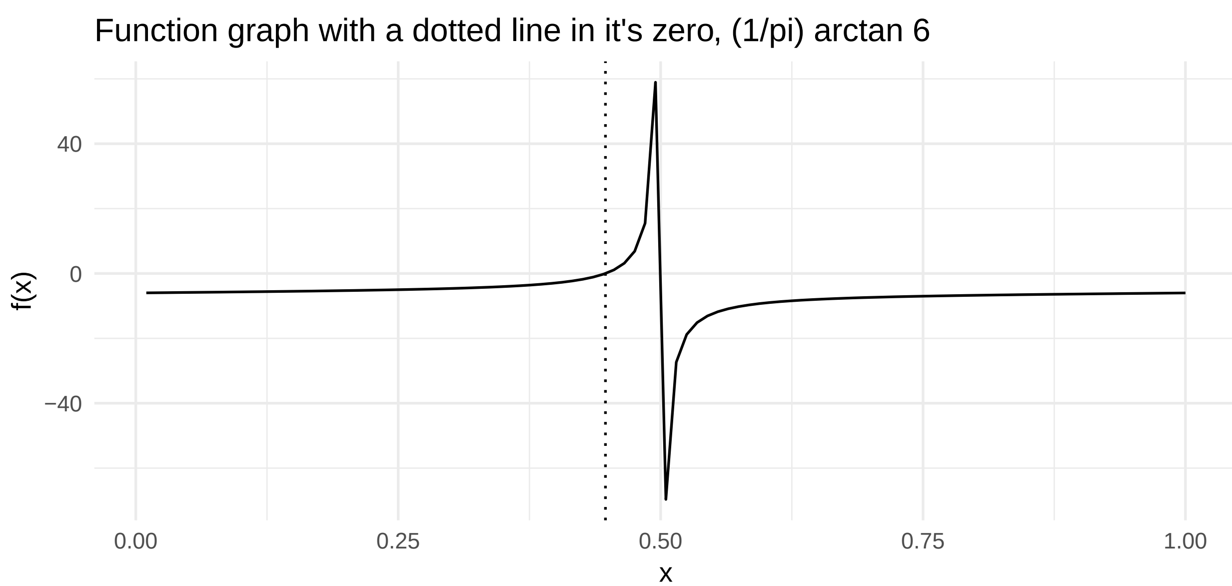
Exercise 6

A função $f(x) = \tan(\pi x) - 6$ tem um zero em $\frac{1}{\pi} \arctan 6 \approx 0.447431543$. Considere $x_0 = 0$ e $x_1 = 0.48$, e use 10 iterações para cada um dos seguintes métodos para calcular um valor aproximado desse zero. Qual dos métodos foi mais sucedido? Por quê?

(a) Método da Bisseção. (b) Método de Newton. (c) Método Secante.

<r code>

```
f6 <- function(x) { tan(pi * x) - 6 }

f6line <- function(x) { pi/cos(pi * x)^2 }

ggplot(data.frame(x = c(.01, 1)), aes(x = x)) +
    theme_minimal() +
    stat_function(fun = f6) +
    geom_vline(xintercept = (1/pi) * atan(6), linetype = "dotted") +
    labs(y = "f(x)",
         title =
    "Function graph with a dotted line in it's zero, (1/pi) arctan 6")
```




```
# (a) metodo da Bissecao ===============================================
bisection(f6, a = 0, b = .48)
```

```
          a         x        b
1 0.2400000 0.3600000 0.480000
2 0.3600000 0.4200000 0.480000
3 0.4200000 0.4500000 0.480000
4 0.4200000 0.4350000 0.450000
```

```
5 0.4350000 0.4425000 0.450000
6 0.4425000 0.4462500 0.450000
7 0.4462500 0.4481250 0.450000
8 0.4462500 0.4471875 0.448125
9 0.4471875 0.4476562 0.448125

bisection(f6, a = .3, b = .45)

          a         x         b
1 0.3750000 0.4125000 0.4500000
2 0.4125000 0.4312500 0.4500000
3 0.4312500 0.4406250 0.4500000
4 0.4406250 0.4453125 0.4500000
5 0.4453125 0.4476562 0.4500000
6 0.4453125 0.4464844 0.4476562
7 0.4464844 0.4470703 0.4476562
8 0.4470703 0.4473633 0.4476562

# (b) metodo de Newton =================================================
newton(f6, f6line, init = .48)

[1] 0.4675825 0.4551292 0.4485512 0.4474552 0.4474316

newton(f6, f6line, init = .4)

[1] 0.4888264 0.4800144 0.4676003 0.4551429 0.4485552 0.4474554 0.4474316

newton(f6, f6line, init = .35)

 [1]  0.6148770  0.9581745  2.8765939  4.6248962  5.0166148  6.9046908
 [7]  8.7380937  9.7803623 11.0726305 12.8146330 14.2929995 14.8396780
[13] 16.4392033 16.4487038

# (c) metodo Secante ===================================================
secant(f6, inits = c(0, .48))

  [1]  1.811942e-01  2.861872e-01  1.091986e+00 -3.692297e+00 -2.260065e+01
  [6] -5.722283e+01  3.538758e+00 -1.139444e+02 -1.958939e+02 -2.923540e+03
 [11] -2.214725e+03 -2.716996e+03 -1.278719e+02  3.467719e+04  7.343020e+05
 [16] -5.260916e+06  3.312104e+06  6.974486e+06  1.586291e+07 -1.155064e+08
 [21]  2.245408e+08  8.550972e+08 -1.977882e+09  3.277897e+10 -2.115656e+11
 [26] -3.186281e+11 -1.312377e+11  6.374025e+11 -2.380563e+12 -4.137395e+13
 [31]  6.308449e+13  1.317103e+14 -1.213713e+14  7.601656e+14  6.227984e+15
 [36]  6.154177e+14  7.951176e+14  7.177148e+14  1.808552e+15 -1.080135e+16
 [41] -9.485429e+16  2.368874e+18 -8.291690e+18 -4.482692e+20 -2.086653e+21
 [46] -1.646282e+21 -1.964709e+21 -8.517763e+21 -2.379764e+22  2.469817e+22
 [51]  4.979333e+23  4.226362e+26 -3.569902e+27 -2.787053e+28 -7.614246e+28
 [56]  2.357168e+28 -7.442220e+28 -7.266297e+28 -2.004535e+29  1.223068e+30
 [61] -2.419707e+30 -6.422299e+30 -1.116669e+31  6.548127e+29 -1.600158e+32
```

```
 [66] -2.944007e+32 -7.207910e+31 -6.882463e+32 -6.414169e+32 -6.861985e+32
 [71] -7.107462e+32 -8.064133e+32 -1.312680e+33  1.376409e+33  3.208962e+33
 [76]  2.623495e+32 -2.287125e+34  9.972344e+34  7.111966e+36 -9.675269e+37
 [81]  6.987968e+37  1.303106e+38 -5.326046e+38  4.195466e+39  3.416342e+40
 [86]  7.126923e+40  1.394978e+40 -1.178986e+41  6.420520e+42  2.943265e+43
 [91]  2.550461e+43  2.889228e+43  5.001322e+43  9.937317e+43 -3.144715e+43
 [96]  1.410598e+45  2.946084e+46 -2.072256e+47 -1.107872e+48 -1.007816e+49
```

```
secant(f6, inits = c(.4, .48))
```

```
[1] 0.4182404 0.4294442 0.4572304 0.4441121 0.4468177 0.4474699 0.4474311
```

```
secant(f6, inits = c(.2, .45))
```

```
[1] 0.4359612 0.4468770 0.4475512 0.4474303
```

```
secant(f6, inits = c(.1, .3))
```

```
  [1]  1.179464e+00 -5.164583e+00  2.953542e+01 -3.236059e+01 -1.062551e+02
  [6] -5.774681e+02  2.660748e+02  7.437958e+02 -2.523894e+03 -1.945938e+04
 [11]  3.135565e+04  8.207334e+05  1.101432e+06  7.415477e+05 -5.710653e+05
 [16] -3.719257e+07 -2.026927e+08  1.381799e+09  4.350706e+09 -6.754052e+09
 [21]  1.650663e+09  3.348514e+08  3.706192e+09 -2.574047e+10 -3.027504e+11
 [26] -4.059512e+11 -2.560586e+11 -1.061744e+12  4.229217e+12 -3.440684e+12
 [31] -7.820942e+12 -1.948523e+13  1.988652e+13  1.580199e+14  6.480886e+15
 [36]  2.422249e+16 -3.628661e+15 -2.968985e+16 -3.017213e+17  9.252426e+17
 [41]  8.849732e+18 -1.654225e+19  7.729111e+19  5.182338e+19  6.915531e+19
 [46]  1.075834e+20  2.189804e+20 -7.141249e+20  7.410494e+21  4.586360e+22
 [51]  5.440140e+23 -1.305491e+24 -6.087997e+24  1.333473e+25  8.749912e+25
 [56] -3.629936e+26  2.137064e+26  3.144346e+26  3.349013e+25 -2.849983e+27
 [61] -1.765960e+27 -2.396448e+27  1.810203e+28  7.625017e+28 -7.550418e+28
 [66] -1.373904e+30  2.061620e+31  2.599481e+32 -1.168391e+33 -6.278245e+33
 [71]  8.480371e+34  2.980663e+34  5.544103e+34  4.928726e+34  5.404930e+34
 [76]  6.709925e+34  1.159576e+35 -1.777612e+35 -4.985813e+36 -1.085412e+38
 [81]  6.623672e+38 -3.765887e+39 -1.633604e+40 -2.034519e+41 -1.985436e+41
 [86] -2.033738e+41 -2.167102e+41 -3.726927e+41  7.398304e+42 -9.641041e+42
 [91] -2.546430e+43  1.014599e+44  1.257574e+45 -4.866859e+45 -1.104469e+48
 [96]  1.147852e+49 -5.740759e+48 -1.099387e+49 -4.636645e+49  1.715379e+50
```

Podemos dizer que o método da Bisseção é o mais sucedido, porque dado que o intervalo contém o zero da função, o método irá encontrá-lo, com o intervalo podendo ser maior ou menor. No caso da função tangante tal intervalo não pode ser muito grande, dada a infinidade de zeros que a função apresenta.

Com os métodos de Newton e Secante (Quase-Newton) temos uma dependência muito forte com o chute inicial, o que requer uma examinação cuidadosa da função. Se o(s) ponto(s) for(em) mal escolhido(s), um ponto com derivada próxima de zero neste caso, sua respectiva reta tangente/secante te jogará para muito longe do zero desejado. Como aconteceu com alguns dos chutes iniciais utilizados.

Exercise 7

Equação de anuidade devidas:

$$A = \frac{P}{i}[(1+i)^n - 1].$$

Em que

- $A = 750.000$;
- $P = 1.500$;
- $n = 20 \times 12 = 240$;
- $i = ?$.

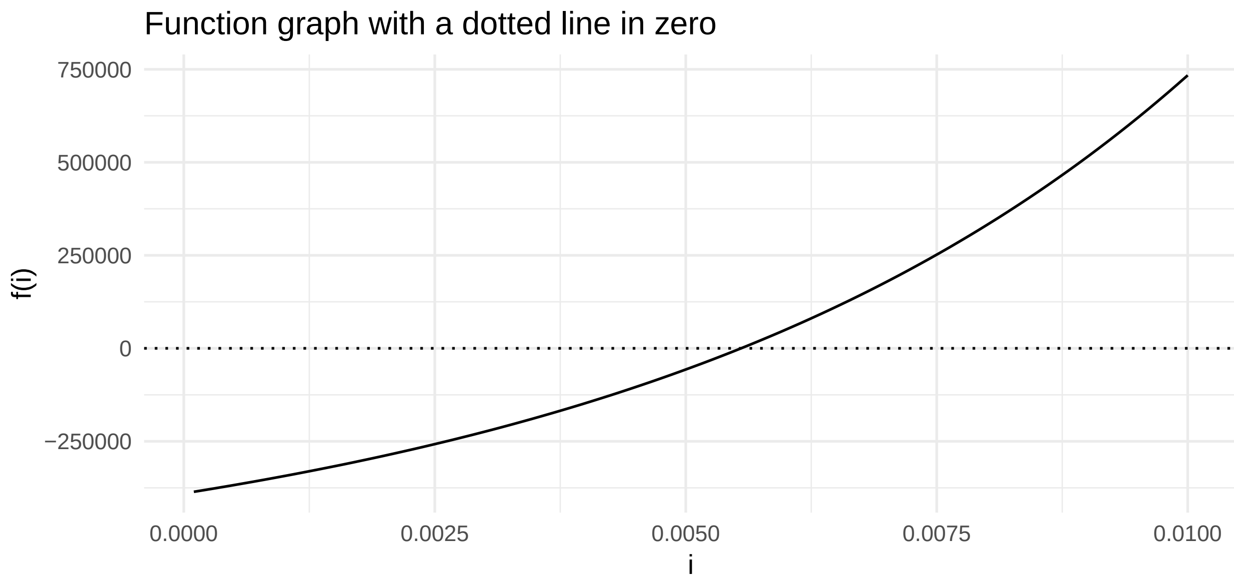
Assim, queremos encontrar a solução de:

$$\frac{1500}{i}[(1+i)^{240} - 1] - 750000 = 0.$$

Uso aqui as funções no *default*, i.e., com tolerância de 1e-3.

<r code>

```
f7 <- function(i) { (1500/i) * ((1 + i)^240 - 1) - 750000 }

# jogando alguns valores para descobrir um bom intervalo
f7(1e-2)

[1] 733883

f7(1e-3)

[1] -343354.9

f7(5e-3)

[1] -56938.66

f7(6e-3)

[1] 50643.51

ggplot(data.frame(i = c(1e-4, 1e-2)), aes(x = i)) +
    theme_minimal() +
    stat_function(fun = f7) +
    geom_hline(yintercept = 0, linetype = "dotted") +
    labs(y = "f(i)", title = "Function graph with a dotted line in zero")
```




```
# bissecao ============================================================
bisection(f7, a = 1e-4, b = 1e-2)

        a           x          b
1 0.00505 0.007525000 0.01000000
2 0.00505 0.006287500 0.00752500
3 0.00505 0.005668750 0.00628750
4 0.00505 0.005359375 0.00566875

bisection(f7, a = 1e-5, b = 1e-1)

            a           x           b
1 0.000010000 0.025007500 0.050005000
2 0.000010000 0.012508750 0.025007500
3 0.000010000 0.006259375 0.012508750
4 0.000010000 0.003134687 0.006259375
5 0.003134687 0.004697031 0.006259375
6 0.004697031 0.005478203 0.006259375
7 0.005478203 0.005868789 0.006259375

bisection(f7, a = 4e-3, b = 6e-3)

     a      x      b
0.0050 0.0055 0.0060

# secante =============================================================
secant(f7, inits = c(.0075, .01))

[1] 0.006193969 0.005768641 0.005562695 0.005551006 0.005550782

secant(f7, inits = c(.001, .0025))
```

```
[1] 0.007002154 0.005155696 0.005502354 0.005552461 0.005550775

# newton ============================================================
f7line <- function(i) {
    (360000 * (1 + i)^239 * i - 1500 * ((1 + i)^240 - 1))/i^2
}
newton(f7, f7line, init = .009)

[1] 0.006425995 0.005614522 0.005551134 0.005550782

newton(f7, f7line, init = .001)

[1] 0.007800102 0.005944819 0.005564007 0.005550797 0.005550782

newton(f7, f7line, init = .0025)

[1] 0.006489656 0.005623906 0.005551245 0.005550782
```

Querendo ter em conta um total de 750.000,00 dinheiros para efetuar retiradas após 20 anos, e podendo dispor de 1.500,00 dinheiros por mês para atingir essa meta, a taxa de juros mínima a que esse valor deve ser investido é de **0.00555**, assumindo que o período de capitalização é mensal.

Exercise 8

Duas locomotivas viajam no mesmo sentido, e trilho, com equações de movimento dadas por

$$x_1(t) = 110 - 80\exp(-t/2) \quad \text{e} \quad x_2(t) = 50t,$$

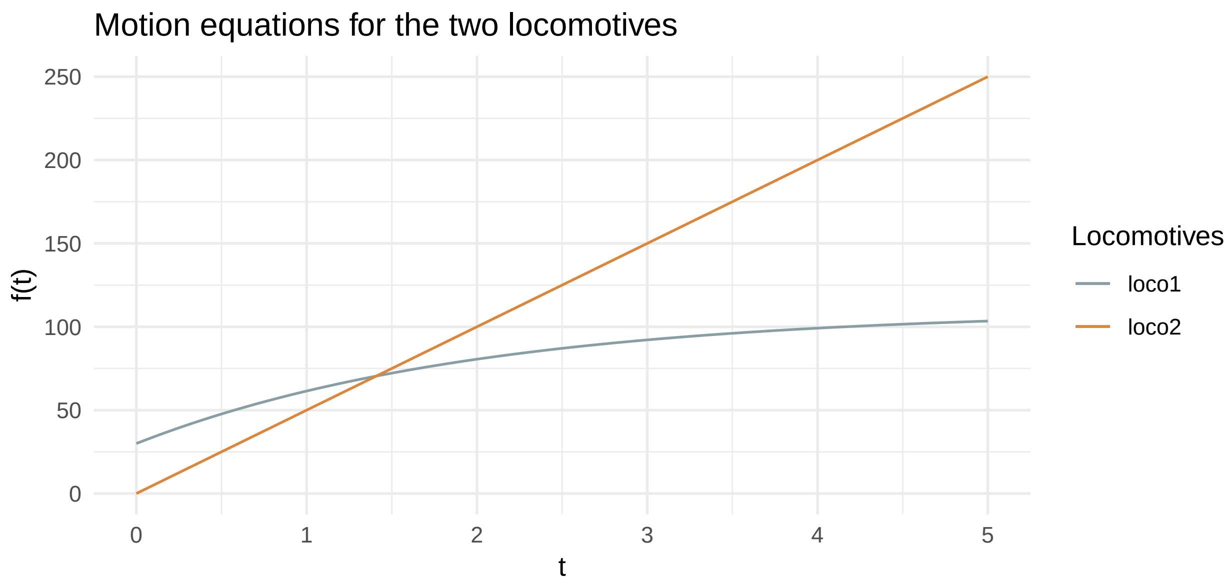
respectivamente.

Utilizando argumentos gráficos, verifique se estas locomotivas se chocam, e se isso acontecer, em quanto tempo (approx.) o acidente ocorreria?

<r code>

```
loco1 <- function(t) { 110 - 80 * exp(-t/2) }

loco2 <- function(t) { 50 * t }

library(wesanderson)
paleta <- wes_palette("Royal1", 2, "continuous")

(p1 <- ggplot(data.frame(t = c(0, 5)), aes(x = t)) +
     theme_minimal() +
     stat_function(fun = loco1, aes(colour = "loco1")) +
     stat_function(fun = loco2, aes(colour = "loco2")) +
```

```r
labs(y = "f(t)", title = "Motion equations for the two locomotives") +
scale_colour_manual("Locomotives", values = c(paleta[1], paleta[2]))
```



Sim, elas se chocam.

Agora, para encontrar o tempo t em que essas lomotivas colidem, precisamos encontrar a solução de:

$$x_1(t) = x_2(t) \Rightarrow x_1(t) - x_2(t) = 0.$$

i.e.,

$$110 - 80\exp(-t/2) - 50t = 0.$$

<r code>

```r
f8 <- function(t) { 110 - 80 * exp(-t/2) - 50 * t }

# bissecao ============================================================
bisection(f8, a = .5, b = 2)
```

```
          a        x        b
1  1.250000 1.625000 2.000000
2  1.250000 1.437500 1.625000
3  1.250000 1.343750 1.437500
4  1.343750 1.390625 1.437500
5  1.390625 1.414062 1.437500
6  1.390625 1.402344 1.414062
7  1.402344 1.408203 1.414062
8  1.408203 1.411133 1.414062
9  1.408203 1.409668 1.411133
10 1.408203 1.408936 1.409668
11 1.408936 1.409302 1.409668
```

```r
bisection(f8, a = 1, b = 1.5)
```

```
         a        x        b
1 1.250000 1.375000 1.500000
2 1.375000 1.437500 1.500000
3 1.375000 1.406250 1.437500
4 1.406250 1.421875 1.437500
5 1.406250 1.414062 1.421875
6 1.406250 1.410156 1.414062
7 1.406250 1.408203 1.410156
8 1.408203 1.409180 1.410156
9 1.408203 1.408691 1.409180
```

```r
# newton ============================================================
f8line <- function(t) { 40 * exp(-t/2) - 50 }

newton(f8, f8line, init = .5)
```

```
[1] 1.704158 1.420904 1.409074 1.409051
```

```r
newton(f8, f8line, init = 1.75)
```

```
[1] 1.424455 1.409090 1.409051
```

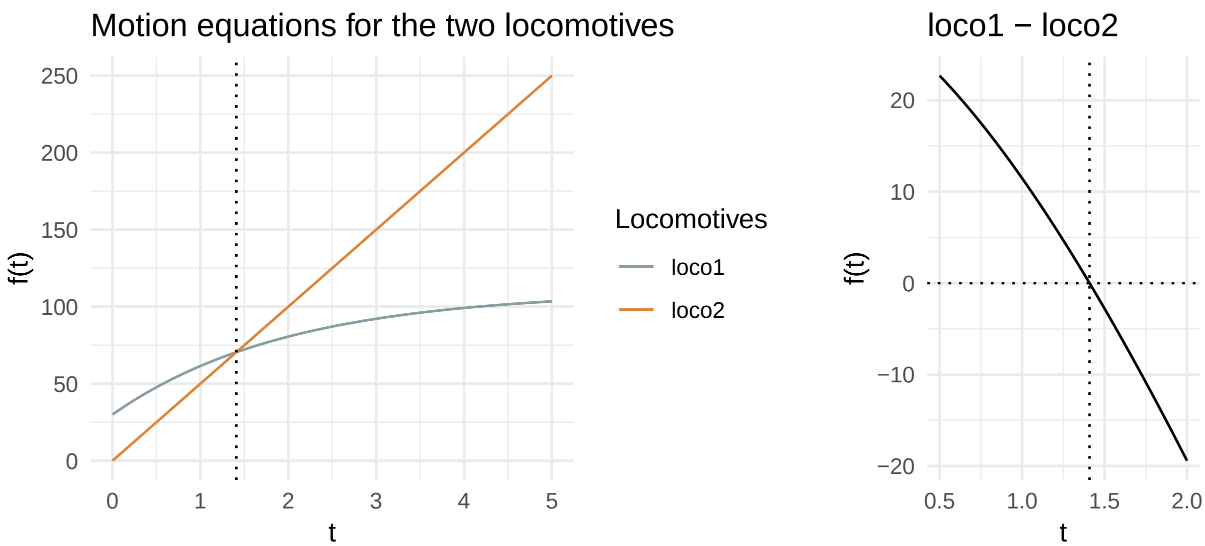
```r
# secante ===========================================================
secant(f8, inits = c(.5, .75))
```

```
[1] 1.595558 1.385115 1.408358 1.409054
```

```r
secant(f8, inits = c(1.75, 2))
```

```
[1] 1.433939 1.411045 1.409059 1.409051
```

```r
p2 <- ggplot(data.frame(t = c(.5, 2)), aes(x = t)) +
    theme_minimal() +
    stat_function(fun = f8) +
    geom_hline(yintercept = 0, linetype = "dotted") +
    geom_vline(xintercept = 1.409, linetype = "dotted") +
    labs(y = "f(t)", title = "loco1 - loco2")

library(multipanelfigure)

figure <- multi_panel_figure(rows = 1, columns = 3,
                             panel_label_type = "none")

(figure %<>% fill_panel(p1 + geom_vline(xintercept = 1.409,
                                         linetype = "dotted"),
                       column = 1:2) %<>% fill_panel(p2, column = 3))
```



Supondo que as duas locomotivas viajam no mesmo sentido e trilho, com as equações de movimento dadas, as locomotivas se chocam no tempo $t = \mathbf{1.409}$.

Exercise 9. (b)

Encontrar $\sqrt{2}$ usando a fórmula de recorrência encontrada a partir do método de Newton, usando $x_0 = 1$. Também uso aqui uma tolerância bem baixa, 1e-16.

Fórmula de recorrência para $\sqrt{2}$:

$$x_{k+1} = \frac{1}{2}(x_k + \frac{2}{x_k}).$$

<r code>

```
newton_recurrence <- function(x0, a, p, kmax = 100, tol = 1e-16) {
    xs <- numeric(kmax)
    recurrence <- function(x) { (1/p) * ((p - 1) * x + a/x^(p - 1)) }
    xs[1] <- recurrence(x0)
    xs[2] <- recurrence(xs[1])
    k <- 2
    while(abs(diff(xs[(k - 1):k]))/abs(xs[k]) > tol & k <= kmax) {
        k <- k + 1
        xs[k] <- recurrence(xs[k - 1])
    }
    return(xs[seq(k)])
}
newton_recurrence(x0 = 1, a = 2, p = 2, kmax = 3)
```

```
[1] 1.500000 1.416667 1.414216 1.414214
```

```
newton_recurrence(x0 = 1, a = 2, p = 2)
```

```
[1] 1.500000 1.416667 1.414216 1.414214 1.414214 1.414214
```

Last modification on ...

```
[1] "2019-09-21 13:12:48 -03"
```