Trabalho Prático 1 - Cálculo Numérico

Maria Luiza Alvez Belarmino Pedro Henrique de Almeida

UNIFAL - Universidade Federal de Alfenas

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Trabalho....: Prático 1

Disciplina..: Cálculo Numérico

Professora..: Angela Leite Moreno

Alunos......: Maria Luiza Alves Belarmino - 2023.1.08.015

Pedro Henrique de Alemida - 2022.1.08.045

Data........: 30 de abril de 2025

 # # # # # # Questão 1 # # # # # #

Considere q(x) = 816x^3 − 3835x^2 + 6000x − 3125 e p(x) = q(x +), onde é a

média aritméticado último dígito da matrícula da dupla que faz o trabalho.

 # # (a) # #

Quais são as raízes de p(x)?

```r
# Biblioteca para polinomios
library(polynom)

# valor de omega dado que o numero final das matriculas
omega <- (5 + 5) / 2

p <- 816 * (polynomial(c(5,1)))^3 -
  3835 * (polynomial(c(5,1)))^2 +
```

```r
  6000 * (polynomial(c(5,1))) -
  3125

# raizes de p(x)
raizes <- solve(p)
raizes
```

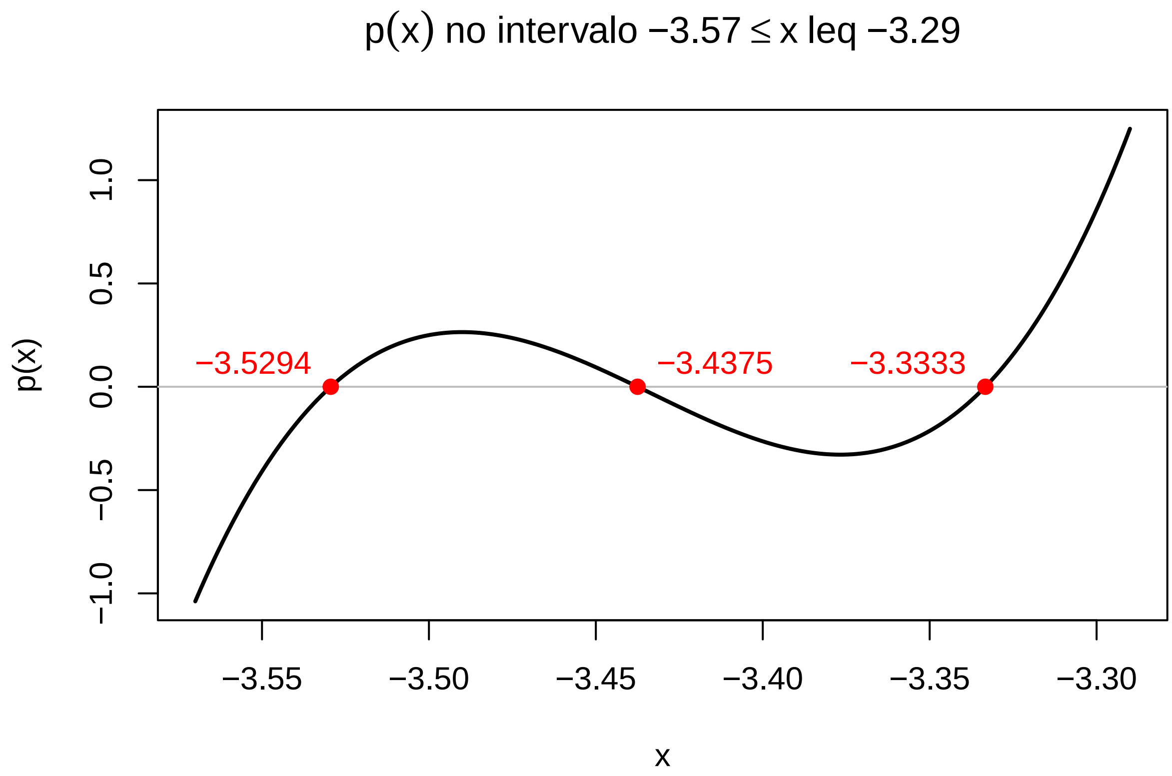
```
## [1] -3.529412 -3.437500 -3.333333
```

(b) #

Faça o gráfico de p(x) para (1,43 —) x (1,71 —). Mostre onde se

localizam os zeros de p(x)

```r
# Intervalo para o gráfico
x_min <- 1.43 - omega
x_max <- 1.71 - omega

# Vetor dos valores de x e y
x_vals <- seq(x_min, x_max, length.out = 500)
y_vals <- predict(p, x_vals)

# Grafico
plot(x_vals, y_vals, type = "l", lwd = 2,
     main = bquote(p(x) ~ "no intervalo" ~ .(round(x_min, 2)) <= x ~ leq ~ .(round(x_max, 2))),
     xlab = "x", ylab = "p(x)")
abline(h = 0, col = "gray")

# Pontos nos valores 0 do grafico
raizes_no_intervalo <- raizes[raizes >= x_min & raizes <= x_max]
points(raizes_no_intervalo, rep(0, length(raizes_no_intervalo)), col = "red", pch = 19)

# Valor das raizes melhor posicionado
raizes1e3 <- raizes[c(1, 3)]
text(raizes1e3, rep(0, length(raizes1e3)) + 0.1,
     labels = round(raizes1e3, 4), pos = 2, col = "red")

raiz2 <- raizes[2]
text(raiz2, rep(0, length(raiz2)) + 0.1,
     labels = round(raiz2, 4), pos = 4, col = "red")
```



(c) #

Começando com x0 = (1,5 −), o que o Método de Newton-Raphson faz?

```
p_derivada <- deriv(p)

x0 <- 1.5 - omega
# tolerância para parada
tol <- 1e-6
max_iter <- 100

# Função Newton-Raphson
newton_raphson <- function(p, p_derivada, x0, tol, max_iter) {
  x <- x0
  for (i in 1:max_iter) {
    fx <- predict(p, x)
    dfx <- predict(p_derivada, x)

    if (abs(dfx) < 1e-10) {
      cat("Derivada próxima de zero!\n")
      return(NA)
```

```r
    }

    x_new <- x - fx / dfx

    cat(sprintf("Iteração %d: x = %.10f, p(x) = %.10f\n", i, x_new, predict(p, x_new)))

    if (abs(x_new - x) < tol) {
      cat(sprintf("\nConvergiu em %d iterações.\n", i))
      return(x_new)
    }

    x <- x_new
  }
  cat("Número máximo de iterações atingido.\n")
  return(x)
}

# Chamada da função
raiz_aproximada <- newton_raphson(p, p_derivada, x0, tol, max_iter)
```

```
## Iteração 1: x = -3.5833333333, p(x) = -1.6041666667
## Iteração 2: x = -3.5493227326, p(x) = -0.3924145203
## Iteração 3: x = -3.5336478075, p(x) = -0.0665736255
## Iteração 4: x = -3.5296710664, p(x) = -0.0038290754
## Iteração 5: x = -3.5294128322, p(x) = -0.0000156992
## Iteração 6: x = -3.5294117647, p(x) = -0.0000000003
## Iteração 7: x = -3.5294117647, p(x) = 0.0000000000
##
## Convergiu em 7 iterações.
```

```r
# Saida da resposta aproximada
cat("\nRaiz aproximada por Newton-Raphson:", raiz_aproximada, "\n")
```

```
##
## Raiz aproximada por Newton-Raphson: -3.529412
```

(d) #

Começando com x0 = − e x1 = (1 −), o que o Método da Secante faz?

```r
# Função secante
secante <- function(p, x0, x1, tol, max_iter) {
  for (i in 1:max_iter) {
    fx0 <- predict(p, x0)
    fx1 <- predict(p, x1)

    if (abs(fx1 - fx0) < 1e-10) {
```

```r
      cat("Diferença muito pequena!\n")
      return(NA)
    }

    x2 <- x1 - fx1 * (x1 - x0) / (fx1 - fx0)

    cat(sprintf("Iteração %d: x = %.10f, p(x) = %.10f\n", i, x2, predict(p, x2)))

    if (abs(x2 - x1) < tol) {
      cat(sprintf("\nConvergiu em %d iterações.\n", i))
      return(x2)
    }

    x0 <- x1
    x1 <- x2
  }
  cat("Número máximo de iterações atingido.\n")
  return(x2)
}

# Definições iniciais
x0 <- -omega
x1 <- 1 - omega

# Chamada da função
raiz_secante <- secante(p, x0, x1, tol, max_iter)
```

```
## Iteração 1: x = -3.9516940624, p(x) = -109.5625149273
## Iteração 2: x = -3.7980092042, p(x) = -36.7162658520
## Iteração 3: x = -3.7205483303, p(x) = -17.0941182467
## Iteração 4: x = -3.6530671675, p(x) = -6.9546369930
## Iteração 5: x = -3.6067820589, p(x) = -2.9224779506
## Iteração 6: x = -3.5732349671, p(x) = -1.1644456359
## Iteração 7: x = -3.5510148041, p(x) = -0.4355910867
## Iteração 8: x = -3.5377351931, p(x) = -0.1391545714
## Iteração 9: x = -3.5315014179, p(x) = -0.0317638025
## Iteração 10: x = -3.5296576057, p(x) = -0.0036295244
## Iteração 11: x = -3.5294197407, p(x) = -0.0001173091
## Iteração 12: x = -3.5294117959, p(x) = -0.0000004593
## Iteração 13: x = -3.5294117647, p(x) = -0.0000000001
##
## Convergiu em 13 iterações.
```

```r
# Saida da resposta aproximada
cat("\nRaiz aproximada pela secante:", raiz_secante, "\n")
```

```
##
## Raiz aproximada pela secante: -3.529412
```

(e) #

Começando no intervalo [1 — , 2 —], o que o Método da Bissecçao faz?

```r
# Função Bissecção
bisseccao <- function(p, a, b, tol, max_iter) {
  fa <- predict(p, a)
  fb <- predict(p, b)

  if (fa * fb > 0) {
    cat("Erro: p(a) e p(b) têm o mesmo sinal!\n")
    return(NA)
  }

  for (i in 1:max_iter) {
    c <- (a + b) / 2
    fc <- predict(p, c)

    cat(sprintf("Iteração %d: c = %.10f, p(c) = %.10f\n", i, c, fc))

    if (abs(fc) < tol || abs(b - a) < tol) {
      cat(sprintf("\nConvergiu em %d iterações.\n", i))
      return(c)
    }

    if (fa * fc < 0) {
      b <- c
      fb <- fc
    } else {
      a <- c
      fa <- fc
    }
  }
  cat("Número máximo de iterações atingido.\n")
  return((a + b) / 2)
}

# Definições iniciais
a <- 1 - omega
b <- 2 - omega

# Chamada da função
raiz_bisseccao <- bisseccao(p, a, b, tol, max_iter)
```

```
## Iteração 1: c = -3.5000000000, p(c) = 0.2500000000
## Iteração 2: c = -3.7500000000, p(c) = -23.4375000000
## Iteração 3: c = -3.6250000000, p(c) = -4.2656250000
## Iteração 4: c = -3.5625000000, p(c) = -0.7734375000
## Iteração 5: c = -3.5312500000, p(c) = -0.0278320312
## Iteração 6: c = -3.5156250000, p(c) = 0.1602172852
```

```
## Iteração 7: c = -3.5234375000, p(c) = 0.0796432495
## Iteração 8: c = -3.5273437500, p(c) = 0.0294141769
## Iteração 9: c = -3.5292968750, p(c) = 0.0016864538
## Iteração 10: c = -3.5302734375, p(c) = -0.0128466636
## Iteração 11: c = -3.5297851562, p(c) = -0.0055238586
## Iteração 12: c = -3.5295410156, p(c) = -0.0019046764
## Iteração 13: c = -3.5294189453, p(c) = -0.0001056093
## Iteração 14: c = -3.5293579102, p(c) = 0.0007912972
## Iteração 15: c = -3.5293884277, p(c) = 0.0003430628
## Iteração 16: c = -3.5294036865, p(c) = 0.0001187815
## Iteração 17: c = -3.5294113159, p(c) = 0.0000065998
## Iteração 18: c = -3.5294151306, p(c) = -0.0000495013
## Iteração 19: c = -3.5294132233, p(c) = -0.0000214499
## Iteração 20: c = -3.5294122696, p(c) = -0.0000074249
## Iteração 21: c = -3.5294117928, p(c) = -0.0000004125
##
## Convergiu em 21 iterações.
```

```r
# Saida da resposta aproximada
cat("\nRaiz aproximada pela bissecção:", raiz_bisseccao, "\n")
```

```
##
## Raiz aproximada pela bissecção: -3.529412
```