Primeira Parte

Carlos Souza

2023-11-06

(1) Responda as questoes a seguir.

a) Qual numero na base 10 e equivalente ao numero 10111001 na base 2?

```r
base.2 <- function(x){ # inserir um número binário

  numero <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo
  divisao <- x
  k <- 1

  while(divisao != 0){
    numero[k] <- divisao %% 10  # resto da divisao
    divisao <- divisao %/% 10 # resultado do quocientes
    k <- k + 1 # quantidade de iterações executadas pelo loop
  }
  cat("Numero na base 10 considerando o resto da divisao:", numero, "\n")

  j <- length(numero) # conta quantidade digitos atribuido do resto da divisao
  b <- numeric(j)
  b[j] <- numero[j] # são armazendo os valores atribuidos ao vetor b[j]
  cat("O valor de b[",j,"] =", b[j], "\n")

  for (i in (j-1):1){ # transformação da base 10 para a base 2
    b[i] <- numero[i] + 2 * b[i+1]
    cat("O valor de b[", i, "] =", b[i], "\n")
  }

  cat("O numero binario", x, "na base 10 para a base 2 e igual a", b[1], "\n")
}
base.2(10111001)
```

```
## Numero na base 10 considerando o resto da divisao: 1 0 0 1 1 1 0 1
## O valor de b[ 8 ] = 1
## O valor de b[ 7 ] = 2
## O valor de b[ 6 ] = 5
## O valor de b[ 5 ] = 11
## O valor de b[ 4 ] = 23
## O valor de b[ 3 ] = 46
## O valor de b[ 2 ] = 92
## O valor de b[ 1 ] = 185
## O numero binario 10111001 na base 10 para a base 2 e igual a 185
```

b) Qual numero na base 2 e equivalente ao numero 71 na base 10?

```r
base.10 <- function(x){

  resto <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo
  divisao <- numeric()
  k <- 1
  divisao[k] <- x

  while(divisao[k]!= 0){
    resto[k] <- divisao[k] %% 2 # resto da divisao
    cat("Resto [", k, "] =", resto[k], "\n")
    divisao[k + 1] <- divisao[k] %/% 2 # quociente da divisao
    cat("Quociente [", k + 1, "] =", divisao[k + 1], "\n")
    k <- k + 1 # incremento para passar para o proximo digito
  }

  j <- length(resto)
  numero <- 0

  for(i in 1:j){
    numero <- numero + resto[i] * 10^(i-1)
  }
  cat("O numero decimal", x, "na base 2 para a base 10 e igual a:", numero, "\n")
}

base.10(71)
```

```
## Resto [ 1 ] = 1
## Quociente [ 2 ] = 35
## Resto [ 2 ] = 1
## Quociente [ 3 ] = 17
## Resto [ 3 ] = 1
## Quociente [ 4 ] = 8
## Resto [ 4 ] = 0
## Quociente [ 5 ] = 4
## Resto [ 5 ] = 0
## Quociente [ 6 ] = 2
## Resto [ 6 ] = 0
## Quociente [ 7 ] = 1
## Resto [ 7 ] = 1
## Quociente [ 8 ] = 0
## O numero decimal 71 na base 2 para a base 10 e igual a: 1000111
```

c) Qual numero na base 10 e equivalente ao numero 0.01011 na base 2?

```r
base.2.decil <- function(x){
  resto <- decimal <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo

  resto[1] <- x # resto armazeno o valor de x atribuido
  k <- 1

  while(resto[k] != 0 & k < 100){ # se o resto for diferente de zero é igual a 1, caso contrario é zero
    if(10 * resto[k] >= 1){decimal[k] <- 1

    } else {decimal[k] <- 0}

    resto[k + 1] <- round(10 * resto[k] - decimal[k], 10)
    k <- k + 1 # quantidade de iterações executadas pelo loop
  }

  j = length(decimal) # Conta o numero de digitos encontrado
  numero <- 0

  for(i in 1:j){ # transformação da base 2 para a base 10
    numero <- numero + decimal[i] * 2^(-i)
  }

  cat("O numero binario decimal", x, "na base 10 para a base 2:", numero, "\n")
}

base.2.decil(0.01011)
```

```
## O numero binario decimal 0.01011 na base 10 para a base 2: 0.34375
```

d) Qual numero na base 2 e equivalente ao numero 0.328125 na base 10?

```r
base.10.decil <- function(x){

  resto <- decimal <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo

  resto[1] = x
  cat("O valor do resto na posicao [", 1,"] =", resto[1], "\n")
  k <- 1

  while(resto[k] != 0){ # método da transformação da base 2 fracionaria para a base 10
    if(2 * resto[k] >= 1){
      decimal[k] = 1        # verifica se o valor obtido da equação em que o resto é igual ou maior que 1, caso c
ontrario é 0
    } else {decimal[k] = 0}

    resto[k + 1] = 2 * resto[k] - decimal[k]
    cat("O valor do resto na pasicao [", k + 1, "] =", resto[k + 1], "\n")
    k <- k + 1 # incremento para passar para o proximo digito
  }
  cat("O numero decimal",x, "na base 2 para a base 10: 0.", decimal, "\n")
}
base.10.decil(0.328125)
```

```
## O valor do resto na posicao [ 1 ] = 0.328125
## O valor do resto na pasicao [ 2 ] = 0.65625
## O valor do resto na pasicao [ 3 ] = 0.3125
## O valor do resto na pasicao [ 4 ] = 0.625
## O valor do resto na pasicao [ 5 ] = 0.25
## O valor do resto na pasicao [ 6 ] = 0.5
## O valor do resto na pasicao [ 7 ] = 0
## O numero decimal 0.328125 na base 2 para a base 10: 0. 0 1 0 1 0 1
```

2) Adote a funcao f(x) = 5x^3 - 8x - 1/2

i) Encontre os intervalos de comprimento 1/2 que contenha uma unica raiz em cada intervalo.

```r
# encontrando as raizes aproximadas
funcao.2 <- function(x){ # Função para encontrar as raizes aproximadas
  f <- 5*x^3 - 8*x - 1/2
  return(f)
}

cat("Mundança de sinal para f(-1.5) =", funcao.2(-1.5), "\n")
```

```
## Mundança de sinal para f(-1.5) = -5.375
```

```r
cat("Mundança de sinal para f(-1.0) =", funcao.2(-1.0), "\n")
```

```
## Mundança de sinal para f(-1.0) = 2.5
```

```r
cat("Mundança de sinal para f(-0.5) =", funcao.2(-0.5), "\n")
```

```
## Mundança de sinal para f(-0.5) = 2.875
```

```r
cat("Mundança de sinal para f(0) =", funcao.2(0), "\n")
```

```
## Mundança de sinal para f(0) = -0.5
```

```r
cat("Mundança de sinal para f(0.5) =", funcao.2(0.5), "\n")
```

```
## Mundança de sinal para f(0.5) = -3.875
```

```r
cat("Mundança de sinal para f(1.0) =", funcao.2(1.0), "\n")
```

```
## Mundança de sinal para f(1.0) = -3.5
```

```r
cat("Mundança de sinal para f(1.5) =", funcao.2(1.5), "\n")
```

```
## Mundança de sinal para f(1.5) = 4.375
```

No entanto foi encontrada no intervalo de -1,5 e -1,0 pelo menos uma raiz. Além disso, quando a = -.5 e b = 0, existe pelo menos uma raiz e quando f(x = 1,0 e 1.5), também é encontrada pelo menos uma raiz aproximada.

ii) utilizando o valor −2 como chute inicial e o metodo de Newton, qual e a primeira aproximacao da raiz que satisfaz o criterio de parada |f(x_)| < 10^(-4)? Apresente a evolu¸c˜ao das aproximacoes.

```r
newton <- function(epsilon){ # Método de Newton
  funcao <- function(x){ # Entrada da função para calcular o valor de X atribuido a função
    f <- 5*x^3 - 8*x - 0.5
    return(f)}

  deriv.funcao <- function(x){ # derivada da funcão acima
    f <- 15*x^2 - 8
  return(f)}

  x <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo
  x[1] = -2 # chute inicial

  precisao <- abs(funcao(x[1]))
  k <- 1
  while(precisao > epsilon){ # o loop verifica se a precisao calculada dos novos valores da função é menor que ep
silon
    x[k + 1] <- x[k] - funcao(x[k]) / deriv.funcao(x[k]) # método de newton para obter os valores das proximas ra
izes aproximadas
    k <- k + 1 # incremento da quantidade de iterações realizadas pelo loop
    precisao <- abs(funcao(x[k])) # precisao da nova raiz encontrada pela função de x
  }

  cat("A raiz aproximada para a funcao: ", x[k], "\n")
  cat("Erro relativo:", precisao, "\n")
  cat("Quantidade de iterações realizadas:", k, "\n")
}
newton(10^(-4))
```

```
## A raiz aproximada para a funcao:  -1.23242
## Erro relativo: 2.30046e-08
## Quantidade de iterações realizadas: 6
```

3) Adote a funcao f(x) = 10e^(-10x) - 30

i) Encontre os intervalos de comprimento 1 que contenha uma unica raiz em cada intervalo.

```r
# encontrando as raizes aproximadas
funcao.3 <- function(x){ # Função para encontrar as raizes aproximadas
  f <- 10*exp(-10*x) - 30
  return(f)
}

cat("Mundança de sinal para f(-2) =", funcao.3(-2), "\n")
```

```
## Mundança de sinal para f(-2) = 4851651924
```

```r
cat("Mundança de sinal para f(-1) =", funcao.3(-1), "\n")
```

```
## Mundança de sinal para f(-1) = 220234.7
```

```r
cat("Mundança de sinal para f(0) =", funcao.3(0), "\n")
```

```
## Mundança de sinal para f(0) = -20
```

```r
cat("Mundança de sinal para f(1) =", funcao.3(1), "\n")
```

```
## Mundança de sinal para f(1) = -29.99955
```

```r
cat("Mundança de sinal para f(2) =", funcao.3(2), "\n")
```

```
## Mundança de sinal para f(2) = -30
```

No entanto foi encontrada no intervalo de -1 e 0 pelo menos uma raiz aproximada.

ii) utilizando o metodo da secante e os valores x[1] = −0.5 e x[2] = −1.0, encontre a primeira aproximacao para a unica raiz da funcao que satisfaz o criterio |xk+1 − xk| < 0.0001. Apresente a evolucao das

aproximacoes.

```r
secante <- function(epsilon) { # Método  Secante
  funcao <- function(x) { # Entrada da função para calcular o valor de X atribuido a função
    f <- 10*exp(-10*x) - 30
    return(f)
  }

  x <- numeric() # atribuindo o vetor para receber os valores do algoritmo abaixo
  x[1] <- -0.5 # valores atribuidos a [x[1] e x[2]] como valores iniciais
  x[2] <- -1.0

  k <- 2

  while (abs(x[k] - x[k - 1]) > epsilon) { # método da secante
    x[k + 1] <- (x[k - 1] * funcao(x[k]) - x[k] * funcao(x[k - 1])) / (funcao(x[k]) - funcao(x[k - 1]))
    k <- k + 1 # incremento da quantidade de iterações realizadas pelo loop
    criterio <- abs(x[k] - x[k - 1]) > epsilon # criterio de parada do algoritimo
  }

  cat("A raiz aproximada para a função: ", x[k], "\n")
  cat("Quantidade de iterações realizadas:", k - 2, "\n")
}

secante(0.0001)
```

```
## A raiz aproximada para a função:  -0.1098613
## Quantidade de iterações realizadas: 12
```