Avaliação 3

Integração numérica e sistemas lineares

Paulo Ricardo Seganfredo Campana

10 de maio de 2023

O código fonte desta prova está disponível no Github

Funções de integração numérica

```
library(dplyr)
library(Deriv)
```

```
integral <- function(função, a, b, n, método) {
    i <- seq(a, b, length.out = n+1)
    h <- (b - a) / n
    valores <- sapply(i, função)
    pesos <- case_when(
método == "trapézio"  ~ (rep(2, n)                        - c(1, rep(0, n-1))) |> c(1),
método == "simpson13" ~ (rep(4, n) - rep(c(2, 0),    n/2) - c(1, rep(0, n-1))) |> c(1),
método == "simpson38" ~ (rep(3, n) - rep(c(1, 0, 0), n/3) - c(1, rep(0, n-1))) |> c(1)
    )
    case_when(
        método == "trapézio"  ~   h/2 * sum(valores * pesos),
        método == "simpson13" ~   h/3 * sum(valores * pesos),
        método == "simpson38" ~ 3*h/8 * sum(valores * pesos)
    )
}
```

A função `integral()` calcula uma aproximação da integral definida de $f(x)$ no intevalo $[a, b]$ com suporte para a regra dos trapézios, regra de $1/3$ de Simpson e regra de $3/8$ de Simpson.

```r
erro <- function(função, a, b, n, método) {
    f2x <- sapply(
        seq(b, a, length.out = 1000),
        Deriv(função, nderiv = 2)
    )
    f4x <- sapply(
        seq(b, a, length.out = 1000),
        Deriv(função, nderiv = 4)
    )
    case_when(
        método == "trapézio"   ~ (b-a)^3 / n^2 / 12  * max(abs(f2x)),
        método == "simpson13" ~ (b-a)^5 / n^4 / 180 * max(abs(f4x)),
        método == "simpson38" ~ (b-a)^5 / n^4 / 80  * max(abs(f4x))
    )
}
```

A função `erro()` calcula o máximo erro absoluto da aproximação da função `integral()`, com base nos seguintes limitantes do erro:

$$\begin{aligned}
\text{regra dos trapézios:} &\quad |E| \leqslant \frac{(b-a)^3}{12n^2} \max_{a \leqslant x \leqslant b} \left| f^{(2)}(x) \right| \\
\text{regra de 1/3 de Simpson:} &\quad |E| \leqslant \frac{(b-a)^5}{180n^4} \max_{a \leqslant x \leqslant b} \left| f^{(4)}(x) \right| \\
\text{regra de 3/8 de Simpson:} &\quad |E| \leqslant \frac{(b-a)^5}{80n^4} \max_{a \leqslant x \leqslant b} \left| f^{(4)}(x) \right|
\end{aligned}$$

```r
tabela <- function(função, ...) {
    tibble(
        método = c("trapézio", "simpson13", "simpson38"),
        `valor da integral` = sapply(
            método,
            function(x) integral(função, ..., x)
        ),
        `limite superior do erro` = sapply(
            método,
            function(x) erro(função, ..., x)
        )
    )
}
```

Funções de solução de sistemas lineares

```r
sistema <- function(matriz, iterações = 10) {
    nrow <- nrow(matriz)
    ncol <- ncol(matriz)
    novo <- function(tabela, matriz) {
        anterior <- tabela[nrow(tabela), ] |> as.numeric()
        tabela[nrow(tabela) + 1, ] <- lapply(
            1:(ncol-1),
            function(i) {
                prod <- anterior[-i] * -matriz[i,-c(i,ncol)]
                sum(prod, matriz[i,ncol]) / matriz[i,i]
            }
        )
        tabela
    }
    zeros <- data.frame(lapply(1:nrow, function(x) 0))
    names(zeros) <- paste0("x", 1:nrow)
    Reduce(
        f = function(a, b) novo(a, matriz),
        x = 1:iterações,
        init = zeros
    )
}
```

A função `sistema()` recebe uma matriz de tamanho arbitrário contendo os coeficientes do sistema linear e aplica o método de resolução de Gauss-Siedel para encontrar uma solução aproximada, começando por um chute inicial de $\mathbf{x} = \vec{0}$.

Questão 1.

Calcule o valor aproximado da seguinte interal usando $n = 6$ e a regra dos trapézios generalizada, a regra de 1/3 de Simpson e a regra de 3/8 de Simpson, em cada caso determine um limitante superior para o erro

$$\int_0^2 e^{-x^2} dx$$

```
função <- function(x) exp(-x^2)
tabela(função, 0, 2, n = 6)
```

método	valor da integral	limite superior do erro
trapézio	0.8814156	0.0370370
simpson13	0.8820316	0.0016461
simpson38	0.8819629	0.0037037

Questão 2.

Um radar foi usado para medir a velocidade de um corredor durante os primeiros 5 segundos de uma corrida. Use a regra 1/3 de Simpson para estimar a distância que o corredor cobriu durante aqueles 5 segundos

```
q2 <- tibble(
tempo      = c(0, 0.5,   1, 1.5,   2,  2.5,    3,  3.5,    4,  4.5,    5),
velocidade = c(0,4.67,7.34,8.86,9.73,10.22,10.51,10.67,10.76,10.81,10.81)
)
```

```
h <- 0.5
pesos <- c(1,4,2,4,2,4,2,4,2,4,1)
h / 3 * sum(q2$velocidade * pesos)
# [1] 44.735
```

A Distância estimada que o corredor percorreu nos primeiros 5 segundos da corrida é de 44.735 metros.

Questão 3.

Considere o sistema linear:

$$\begin{cases} 5x_1 + 2x_2 + x_3 = 7 \\ -x_1 + 4x_2 + 2x_3 = 3 \\ 2x_1 - 3x_2 + 10x_3 = 1 \end{cases}$$

a) Verificar a possibilidade de aplicação do método de Gauss-Seidel usando o critério de Sassenfeld.
b) Se possível, resolvê-lo pelo método do item a), obtendo um resultado com erro absoluto $< 10^{-2}$.

Primeiramente, divide-se cada linha pelo coeficiente da diagonal:

$$\begin{cases} 5x_1 + 2x_2 + x_3 = 7 \\ -x_1 + 4x_2 + 2x_3 = 3 \\ 2x_1 - 3x_2 + 10x_3 = 1 \end{cases} \Longrightarrow \begin{cases} x_1 + 0.4x_2 + 0.1x_3 = 1.4 \\ -0.25x_1 + x_2 + 0.5x_3 = 0.75 \\ 0.2x_1 - 0.3x_2 + x_3 = 0.1 \end{cases}$$

O critério de Sassenfeld é tal que $\beta_i = \sum_{j=1}^{i-1} |a_{ij}|\beta_j + \sum_{j=i+1}^{n} |a_{ij}|$

$$\begin{aligned} \beta_1 &= |a_{11}| + |a_{12}| = 0.4 + 0.1 = 0.5 \\ \beta_2 &= |a_{21}|\beta_1 + |a_{23}| = 0.25 \times 0.5 + 0.5 = 0.625 \\ \beta_3 &= |a_{31}|\beta_1 + |a_{32}|\beta_2 = 0.2 \times 0.5 + 0.3 \times 0.625 = 0.2875 \end{aligned}$$

$\max\limits_{1 \leqslant i \leqslant n} \beta_i < 1$, portanto o critério de Sassenfeld é satisfeito e o método de Gauss-Siedel irá convergir:

```r
matriz <- matrix(
    c( 5,  2,  1, 7,
      -1,  4,  2, 3,
       2, -3, 10, 1),
    byrow = TRUE, ncol = 4
)

sistema(matriz, iterações = 25)
```

x1	x2	x3
0.0000000	0.0000000	0.0000000
1.4000000	0.7500000	0.1000000
1.0800000	1.0500000	0.0450000
0.9710000	0.9975000	0.1990000
0.9612000	0.8932500	0.2050500
1.0016900	0.8877750	0.1757350
1.0097430	0.9125550	0.1659945
1.0017791	0.9194385	0.1718179
0.9978610	0.9145358	0.1754757
0.9990905	0.9117274	0.1747885
1.0003513	0.9123784	0.1737001
1.0003086	0.9132378	0.1736432
0.9999762	0.9132555	0.1739096
0.9999159	0.9130393	0.1739814
0.9999880	0.9129883	0.1739286
1.0000190	0.9130327	0.1738989
1.0000071	0.9130553	0.1739060
0.9999967	0.9130488	0.1739152
0.9999975	0.9130416	0.1739153
1.0000003	0.9130417	0.1739130
1.0000007	0.9130436	0.1739125
1.0000001	0.9130440	0.1739129
0.9999998	0.9130436	0.1739132
0.9999999	0.9130434	0.1739131
1.0000000	0.9130434	0.1739130
1.0000000	0.9130435	0.1739130