Vilniaus Universitetas

Perkelties metodas, kubinis splainas

3 laboratorinis darbas

Skaitiniai metodai

Darbą atliko:

Dovydas Martinkus

Duomenų Mokslas 4 kursas 1 gr.

Vilnius, 2022

Turinys

1. Užduoties ataskaita

1.1 Perkelties metodas

Perkelties metodas skirtas spręsti tiesines lygčių sistemas, kurių sistemos matrica yra trįjstrižainė.

Reikalinga sudaryti programą, tiesines lygčių sistemas sprendžiančią perkelties metodu. Programos veikimo derinimui naudojama tiesinių lygčių sistema:

$$\begin{cases} 3x_1 + x_2 = 2 \\ -x_1 + 4x_2 + 3x_3 = -2 \\ 2x_2 + 4x_3 - x_4 = 1 \\ 2x_3 - 3x_4 = -1 \end{cases}$$

Tarkime, kad turime tiesinių lygčių sistemos matricą turinčią tokią (trįjstrižainę) formą:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_n & b_n \end{pmatrix}.$$

Žinoma, kad jei tiesinės lygčių sistemos įstrižainė yra vyraujanti, t. y. $|b_i| \geq |a_i| + |c_i|$, $i = 1, 2, \dots, n$ ir $|b_1| > |c_1|$, tai sprendžiant perkelties metodu, dalyba iš nulio yra negalima.

Nesunku pamatyti, kad ši sąlyga galioja ir anksčiau pateiktai tiesinių lygčių sistemai.

Paleidę programą matome, kad gaunamas tikslus lygčių sistemos sprendinys x_1 = 1, x_2 = -1, x_3 = 1, x_4 = 1.

1.2 Kubinis splainas

Jeigu turimos žinomos reikšmės $x_i,\ i = 1,\dots,n$, tai kubiniu splainu vadinama tolygi funkcija, nagrinėjamame intervale turinti tolydžias pirmos ir antros eilės išvestines ir formą:

$$S_i(x_i) = a_i x^3 + b_i x^2 + c_i x + d_i \qquad x_i \le x \le x_{i+1},\quad i = 0,1,\dots n-1$$

Jeigu papildomai galioja sąlygos $S_0(x_0) = 0,\quad S_{n-1}(x_n) = 0$, tai turimas splainas yra natūralusis.

Nežinomų koeficientų nustatymui natūraliajam splainui gali būti sprendžiama tiesinių lygčių sistema.

$$h_{(i-1)}g_{(i-1)} + 2\big(h_{(i-1)} + h_i\big)g_i + h_i g_{(i+1)} = 6\left(\frac{y_{(i+1)} - y_i}{h_i} - \frac{y_i - y_{(i-1)}}{h_{(i-1)}}\right),\quad \mathrm{i} = 1,\dots,\mathrm{n}-1$$

Tokia lygčių sistema yra trijstrižainė ir todėl gali būti sprendžiama naudojant perkelties metodą.

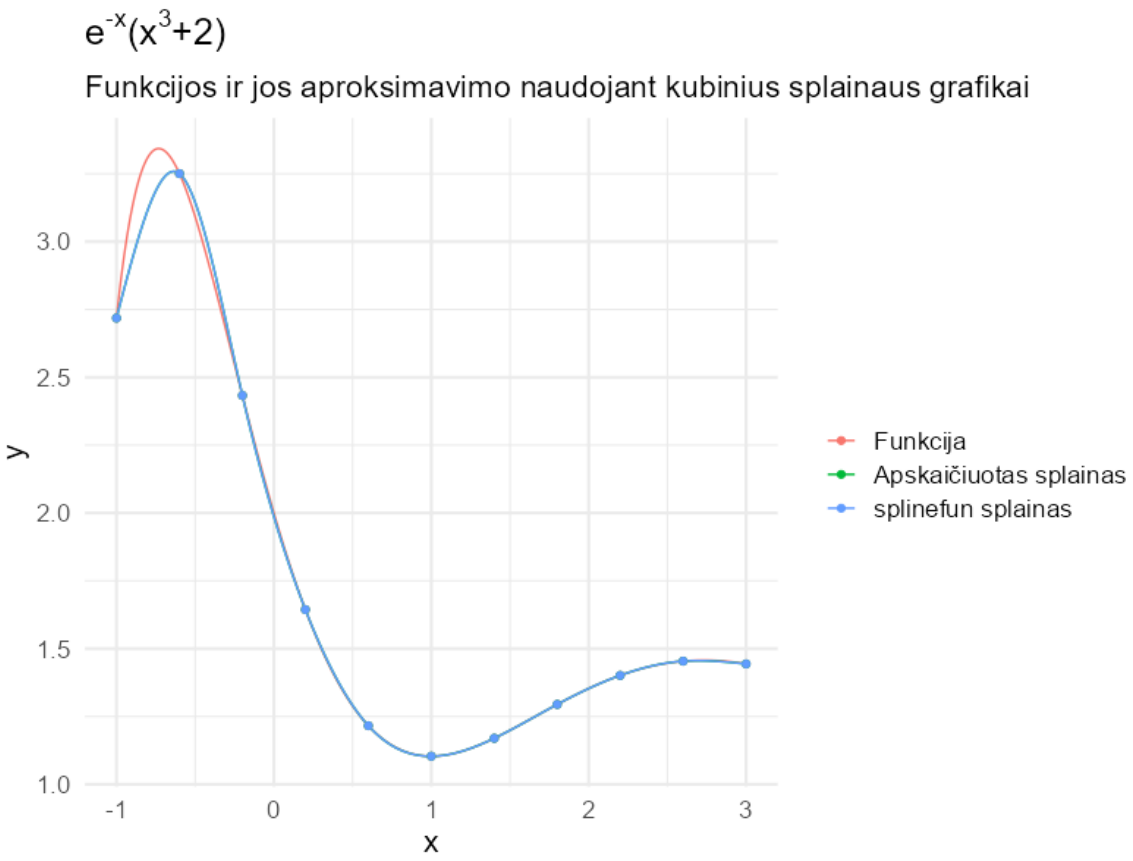
Reikalinga sudaryti (naturalųjį) kubinį splainą funkcijos $e^{-x}(x^3 + 2)$ aproksimavimui intervale [-1,3]. Intervalas dalijimas į 10 vienodo ilgio intervalų taip gaunant 11 vienodai nutolusių interpoliavimo mazgų.

1 lentelėje pateiktos funkcijos reikšmės interpoliavimo mazguose (3 skaičių po kablelio tikslumu):

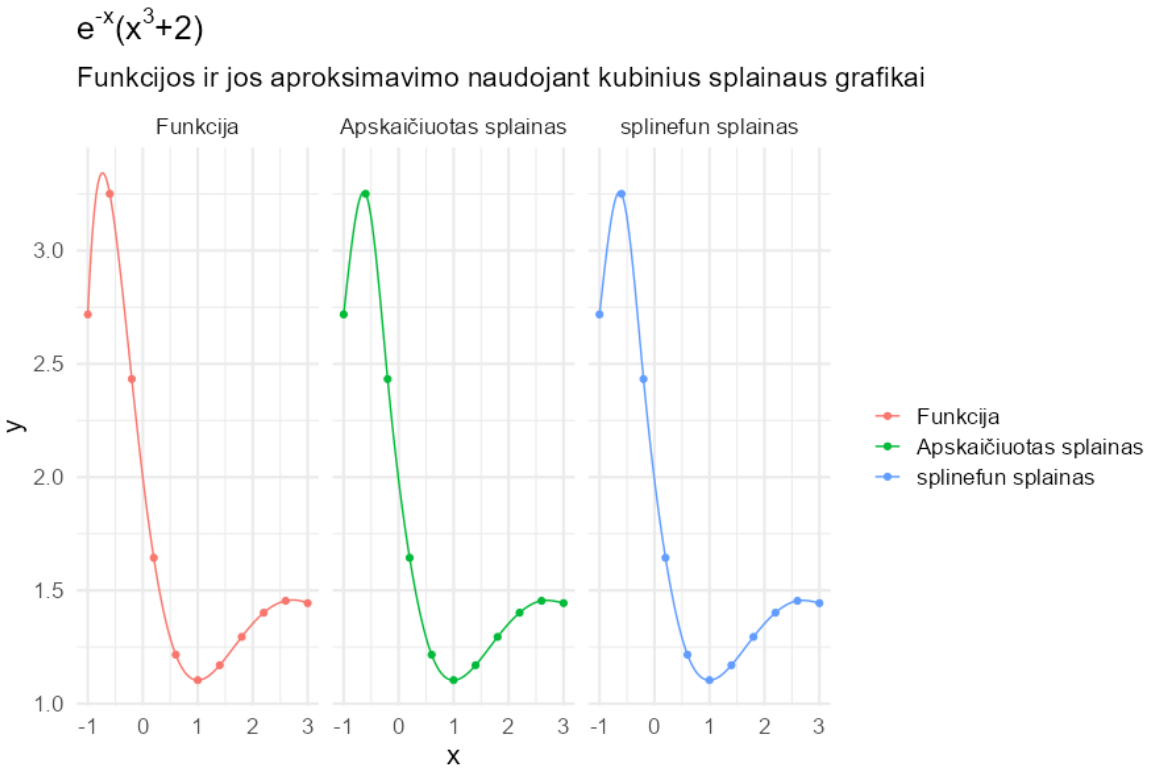
1 lentelė Funkcijos reikšmių lentelė interpoliavimo mazguose

x	$e^{-x}(x^3 + 2)$
-1	2.718
-0.6	3.251
-0.2	2.433
0.2	1.644
0.6	1.216
1	1.104
1.4	1.17
1.8	1.295
2.2	1.401
2.6	1.454
3	1.444

Gauti rezultatai pateikti grafiškai (1 ir 2 pav.). Lygintas pats funkcijos grafikas, apskaičiuotas natūralusis kubinis splainas ir naudojant R funkciją *splinefun* gautas natūralusis kubinis splainas. Kaip matome iš grafiko, apskaičiuoto ir *splinefun* funkcija gautų splainų reikšmės sutapo funkcijos apibrėžimo srityje.



1 pav. Duotosios funkcijos ir abiejų splainų grafikai su pažymėtais interpoliavimo taškais



2 pav. Duotosios funkcijos ir abiejų splainų grafikai su pažymėtais interpoliavimo taškais

Priedas

Žemiau pateiktas naudotas programinis kodas:

```
# Dovydas Martinkus
# Duomenų Mokslas 4k. 1gr.
# 3 uzduotis

###


# Perkelties metodas


## Funkciju aprasymas


vyraujanti <- function(A) {
  result <- TRUE
  A <- abs(A)

  if (A[1,1] <= A[1,2]) {
    result <- FALSE
  }

  for ( i in 2:(nrow(A)-1) ) {
    if(A[i,i] < A[i,i-1] + A[i,i+1]) {
      result <- FALSE
    }
  }
  n <- nrow(A)
  if (A[n,n] < A[n,n-1]) {
    result <- FALSE
  }

  return(result)
}



triistrizaine <- function(A) {
  result <- FALSE

  result <- all(A[abs(row(A) - col(A)) > 1] == 0)

  return(result)
}




perkelties <- function(A,B) {

  if (!triistrizaine(A)) {
    print("TLS nera triistrizaine")
  }

  if (!vyraujanti(A)) {
    print("TLS isstrizaine nera vyraujanti")
```

```
  }

  p <- -1 * A[1,2] / A[1,1]
  q <- B[1] / A[1,1]

  for ( i in 2:(nrow(A)-1) ) {
    p_i <- -1* A[i,1+i] / (A[i,i] + A[i,i-1]*p[i-1])
    q_i <- (B[i]-A[i,i-1]*q[i-1]) / (A[i,i] + A[i,i-1]*p[i-1])

    p <- c(p,p_i)
    q <- c(q,q_i)
  }
  n <- nrow(A)
  q_n <- (B[n] - A[n,n-1]*q[n-1]) / (A[n,n] + A[n,n-1]*p[n-1])


  x <- numeric(n)
  x[n] <- q_n

  for (i in seq(n-1,1)) {
    x[i] <- p[i]*x[i+1] + q[i]
  }

  return(x)
}




A <- matrix(c(3, 1, 0, 0,
              -1, 4, 3, 0,
              0, 2, 4, -1,
              0, 0, -7, -3),
            ncol=4,nrow=4,byrow=TRUE)

B <- matrix(c(2,-2,1,-1),ncol=1)




x <- perkelties(A,B)


## gauto sprendinio patikrinimas istatant i lygciu sistema
palyginimas <- cbind(A %*% matrix(x,ncol=1),B)
colnames(palyginimas) <- c("Gautas B","Norimas B")
t(palyginimas)









# Kubinis splainas
```

```
funkcija <- function(x) {
  exp(x)*(x^3+2)
}


interpoliavimo_taskai <- function(func,n,a,b) {
  step <-(b-a)/n
  x <- a + step*(0:10)
  y <- funkcija(x)
  return(data.frame(x=x,y=y))
}




kubinis_splainas <- function(x,y) {

  # 1 dalis konstruojama triistrizaine lygciu sistema
  n <- length(x)-1
  h <- diff(x)
  y_diff <- diff(y)
  B <- numeric(n-1)
  A <- matrix(nrow=n-1,ncol=n-1)


  for ( i in 1:(n-1) ) {

    if (i == 1) {
      row <- c(2*(h[i]+h[i+1]),h[i+1],rep(0,n-1-i-1))
    }

    else if (i == n-1) {
      row <- c(rep(0,n-1-2),h[i],2*(h[i]+h[i+1]))
    }

    else {
      row <- c(c(rep(0,i-2),h[i],2*(h[i]+h[i+1]),h[i+1],rep(0,n-1-i-1)))
    }

    b_row <- 6*((y[i+2]-y[i+1])/h[i+1] - (y[i+1]-y[i])/h[i])

    A[i,] <- row
    B[i] <- b_row
  }

  if (!triistrizaine(A)) {
    print("TLS nera triistrizaine")
  }

  if (!vyraujanti(A)) {
    print("TLS isstrizaine nera vyraujanti")
  }

  # 2 dalis grazinamas kubinis splainas
  g <- c(0,perkelties(A,B),0)

  G <- g[1:length(g)-1] / 2
```

```
  e <- y_diff / h - 1/6*g[2:length(g)]*h - 1/3*g[1:length(g)-1]*h

  H <- diff(g) / (6 * h)


  func <- function(z) {
    results <- c()
    for (zz in z) {
      for ( i in 0:(n-1) ) {
        if (x[i+1] <= zz & zz <= x[i+2]) {
          results <- c(results,y[i+1] + e[i+1]*(zz - x[i+1]) + G[i+1]*(zz - x[i+1])^2 + H[i+1]*(zz -
x[i+1])^3)
          break
        }
        if (i == n-1) {
          print("Funkcijos argumentas ne is tinkamo intervalo")
          results <- c(results,NA)
        }
      }
    }
    return(results)
  }

}




a <- -1
b <- 3
n <- 10

lentele <- interpoliavimo_taskai(funkcija,n,a,b)



# analogiskai R splinefun() grazinama funkcija
gautas_splainas <- kubinis_splainas(lentele$x,lentele$y)

r_splainas <- splinefun(lentele$x,lentele$y,method='natural')
library(tidyverse)
library(latex2exp)


rezultatai <- tibble(x = seq(-1, 3, 0.01),
           `Funkcija` = funkcija(x),
           `Apskaičiuotas splainas` = gautas_splainas(x),
           `splinefun splainas` = r_splainas(x))


rezultatai2 <- tibble(x = lentele$x,
                      `Funkcija` = funkcija(x),
                      `Apskaičiuotas splainas` = gautas_splainas(x),
                      `splinefun splainas` = r_splainas(x))

rezultatai <- rezultatai %>% pivot_longer(2:4,names_to = " ",values_to="y")
rezultatai$` `<- factor(rezultatai$` `,levels=c("Funkcija","Apskaičiuotas splainas","splinefun
splainas"))
```

```
rezultatai2 <- rezultatai2 %>% pivot_longer(2:4,names_to = " ",values_to="y")
rezultatai2$` `<- factor(rezultatai2$` `,levels=c("Funkcija","Apskaičiuotas splainas","splinefun
splainas"))

ggplot(rezultatai, aes(x,y,color=` `)) +
  geom_line() + geom_point(data = rezultatai2, aes(x,y,color=` `)) +
  labs(title=TeX("e^{-x}(x^3+2)"),
       subtitle = "Funkcijos ir jos aproksimavimo naudojant kubinius splainaus grafikai") +
  theme_minimal(base_size = 16)



ggplot(rezultatai, aes(x,y,color=` `)) +
  geom_line() +  geom_point(data = rezultatai2, aes(x,y,color=` `)) +
  labs(title=TeX("e^{-x}(x^3+2)"),
       subtitle = "Funkcijos ir jos aproksimavimo naudojant kubinius splainaus grafikai") +
  theme_minimal(base_size = 16) + facet_wrap(vars(` `))
```