Rozwiązanie rówaniania różniczkowego metodą elemntów skończonych

Łukasz Czarniekci
Grupa Poniedziałek 11:15

Celem pracy było zademonstrowanie rozwiązania zadanego równania różniczkowego
metodą elementów skończonych.

Postać równania :

$$\begin{cases} -\frac{d}{dx}(E(x)\frac{du(x)}{dx}) = 0 \\ u(2) = 0 \\ \frac{du(0)}{dx} + u(0) = 10 \\ E(x) = \begin{cases} 3\ dla\ x \in [0,1] \\ 5\ dla\ x\ \in\ (1,2] \end{cases} \end{cases}$$

Sformułowanie wariacyjne:

Do sformułowania wariacyjnego konieczna jest funkcja z przestrzeni testowej na zadanym przedziale równania. Różniczkowego. Dla danego równania jest zadany zerowy brzegowy warunek dirichleta dla argumentu x = 2. Dlatego funkcje testowe v zerują się dla tegoż argumentu. Na brzegu x=0 jest zadany warunek brzegowy Cauchy'ego, dlatego funkcje testowe nie mogą się zerować dla tego argumentu. Czyli mamy :

$$\begin{cases} v(x)\ \in\ [0,2] \\ v(2) = 0 \\ v(0)\ \neq 0 \end{cases}$$

Dla zadanego rówania sformułowanie wariacyjne z uwzgędnieniem warunku Cauchy'ego wygląda następująco :

$$\begin{cases} \int_0^2 Eu'v'dx - E(0)v(0)u(0) =- 10E(0)v(0) \\ B(u,v) = \int_0^2 Eu'v'dx - E(0)v(0)u(0) \\ L(v) =- 10E(0)v(0) \end{cases}$$

Za przestrzeń elementów skończonych, które podstawiamy dla zadanego równania w ramach funkcji testowych, oraz składników z których budujemy ostateczną funkcję u przyjmujemy n funkcji typu Bisplanck na zadanym przedziale.

$$\mathrm{h} = \frac{1}{\mathrm{n}} \qquad x_k = k * h$$

$$\mathrm{e}_1 = \begin{cases} \frac{\mathrm{x}_1 - \mathrm{x}}{\mathrm{h}}\ \mathrm{dla\ x}\ \in\ [\mathrm{x}_0, \mathrm{x}_1] \\ 0 \qquad \mathrm{w\,p.p.} \end{cases}$$

$$e_k = \begin{cases} \frac{x - x_{k-1}}{h} & \text{dla } x \in [x_{k-1}, x_x] \\ \frac{x_{k+1} - x}{h} & \text{dla } x \in [x_k, x_{k+1}] \\ 0 & \text{w p. p} \end{cases} \text{ dla k =2,...,n-1}$$

$$e_n = \begin{cases} \frac{x - x_n}{h} & \text{dla } x \in [x_{n-1}, x_{1n}] \\ 0 & \text{w p. p.} \end{cases}$$

Równanie różniczkowe przyjmuje postać równania macierzowego, gdzie niewiadomymi są wagi w_i przez jakie należy wymnożyć kolejne elementy naszej przestrzeni elementów skończonych. Przez wcześniej zadany warunek Diricheta wiemy, że funkcja zeruje się dla x_n dlatego możemy przyjąć, że w_n jest równe 0 i rozważać równanie macierzowe tylko dla n-1 elementów.

Nasze równanie macierzowe przyjmuje postać:

$$\begin{bmatrix} B(e_{1,}e_1) & \cdots & B(e_{n-1,}e_1) \\ \vdots & \ddots & \vdots \\ B(e_{1,}e_{n-1}) & \cdots & B(e_{n-1,}e_{n-1}) \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_{n-1} \end{bmatrix} = \begin{bmatrix} L(e_1) \\ \vdots \\ L(e_{n-1}) \end{bmatrix}$$

Zadany układ generuje skryptem napisanym w języku R. Napotkane całki rozwiązuje poleconą metodą kwadratury Gauss-Legendre (funkcja quad_Integra).

Skrypt wyznaczający funkcję u

Rozwiązywanie całek :

```
#Digital computing integrals
quad_Integral <- function(f,a,b)
{
 l<-(b-a)/2
 mid<-(b+a)/2
 return(l*( f(l/sqrt(3) +mid) +
          f(-l/sqrt(3) +mid) ))
}
```

Generowanie funkcji typu bisplanck :

Źródła:
https://en.wikipedia.org/wiki/Gaussian_quadrature

```
#returns ei for n elements on given interval [a,b]
base_Function <- function(a,b, n,i, derivative)
{
 interval=(b-a)/n

 if (!derivative)
 {
  if (i==1)
  {
   e <- function(x)
   {
    if (x>=a && x<=a+interval)
     return((a+interval-x)/interval)
    return(0)
   }
  }
  else if (i>1 && i<n)
  {
   i=i-1
   e <- function(x)
   {
    if (x>=a + (i-1)*interval && x<=a + i*interval)
     return((x- a-(i-1)*interval)/interval)
    else if (x>=a + i*interval && x<=a + (i+1)*interval)
     return((a+(i+1)*interval -x)/interval)
    return(0)
   }
  }
  else if (i==n)      e <- function(x) 0
 }
 else
 {
  if (i==1)
  {
   e <- function(x)
   {
    if (x>=a && x<=a+interval)
     return(-1/interval)
    return(0)
   }
  }
  else if (i>1 && i<n)
  {
   i=i-1
   e <- function(x)
   {
    if (x>=a + (i-1)*interval && x<=a + i*interval)
     return(1/interval)
    else if (x>=a + i*interval && x<=a + (i+1)*interval)
     return(-1/interval)
    return(0)
   }
  }
  else if (i==n)
  {
   e<- function(x) 0
  }
 }
 return(e)
}
```

Źródła:
https://en.wikipedia.org/wiki/Gaussian_quadrature

```
E <- function(x)
{
 if (x>=0 && x<=1)
  return(3)
 if (x>1&&x<=2)
  return(5)
```

Konstrukcja Macierzy :

```
construct_Matrix <- function(n,a,b)
{
 m<-matrix(nrow=n-1,ncol=n-1)
 for (i in 1:(n-1))
 {
  func=function(x) E(x)*base_Function(a,b,n,i,TRUE)(x)*base_Function(a,b,n,1,TRUE)(x)
  B=quad_Integral(func,a,a+(b-a)/n)
  m[1,i]=B-E(0)*base_Function(a,b,n,1,FALSE)(0)*base_Function(a,b,n,i,FALSE)(0)
  m[i,1]=m[1,i]
 }

 for (i in 2:(n-1))
 {
  for (j in 2:(n-1))
  {
   func=function(x) E(x)*base_Function(a,b,n,j,TRUE)(x)*base_Function(a,b,n,i,TRUE)(x)
   B=quad_Integral(func,a+(i-2)*(b-a)/n,a+(i)*(b-a)/n)
   m[i,j] <- B-E(0)*base_Function(a,b,n,j,FALSE)(0)*base_Function(a,b,n,i,FALSE)(0)
  }
 }
 return(m)
}
```

Konstrukcja wektora z prawej strony równania macierzowego:

```
constructVector <- function(n,a,b)
{
 v=matrix(nrow=n-1,ncol=1)
 for(i in 1:(n-1))
  v[i,1]=-10*E(0)*base_Function(a,b,n,i,FALSE)(0)
 return(v)
}
```

Rozwiązanie równanie macierzowego i złożenie z uzyskanego równanie estymacji poszukiwanej funkcji :

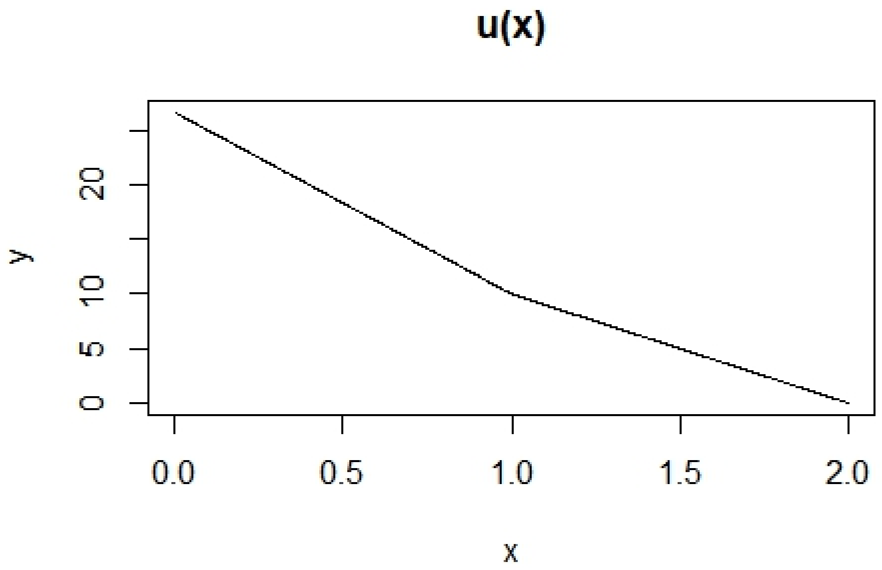
```
constructFunction <- function(n,a,b)
{
 B=construct_Matrix(n,a,b)
 v=constructVector(n,a,b)
 solution=solve(B,v)

 answer=function(x)
 {
  s=0
  for (i in 1:(n-1))
    s=s+base_Function(a,b,n,i,FALSE)(x)*solution[i,1]
  return(s)
 }
 return(answer)
}
```

Wykres funkcji u dla 1000 elementów :

```
f=constructFunction(1000,0,2)

x=1:1000/500
y=1:1000
for (i in 1:1000)
  y[i]=f(x[i])
plot(x,y,type="l",main="u(x)")
```



Źródła:
https://en.wikipedia.org/wiki/Gaussian_quadrature