

# Równanie Transportu Ciepła

Metoda Różnic Skończonych

Dominik Pilipczuk

Projekt obliczeniowy na przedmiot  
Równania Różniczkowe i Różnicowe

Akademia Górniczo Hutnicza w Krakowie  
21.01.2023

# 1 Wprowadzenie

Celem projektu było rozwiązanie równania różniczkowego, podanego przez prowadzącego, metodą elementów skończonych. Zadany problemem było równanie transportu ciepła w postaci:

$$-k(x)\frac{d^2u(x)}{dx^2} = 0 \quad (1)$$

gdzie:

$$k(x) = \begin{cases} 1 & \text{dla } x \in [0, 1] \\ 2 & \text{dla } x \in (1, 2] \end{cases}$$

Samo równanie opisuje przepływ ciepła w układzie, uwzględniające różnice temperatury i właściwości cieplne materiałów. Jest ono ważne w wielu dziedzinach, takich jak inżynieria cieplna czy fizyka.

Wraz z samym równaniem podane były warunki brzegowe:

$$u(2) = 0 \quad (2)$$

oraz:

$$\frac{du(0)}{dx} - u(0) = 20 \quad (3)$$

Łatwo zauważyć, że warunek (2) jest warunkiem Dirichleta dla  $x = 2$ , zaś warunek (3) jest warunkiem Cauchy'ego. Ponadto, dana jest dziedzina funkcji  $u(x)$ :

$$[0, 2] \ni x \rightarrow u(x) \in \mathbb{R}$$

## 2 Obliczenia

Przed wykorzystaniem narzędzi komputerowych do wyznaczenia funkcji  $u$ , należy uprościć równanie (1), obliczając wzór na składniki  $a$  i  $L$ :

$$-k \cdot u''(x) = 0$$

Dzielimy przez  $-k(x) \neq 0$

$$u'' = 0$$

Wprowadzamy funkcję  $v$ , taką, że  $v(2) = 0$ , oraz całkujemy całe równanie:

$$\int_0^2 u'' \cdot v \, dx = 0$$

Całkując przez części:

$$[u' \cdot v]_0^2 - \int_0^2 u' \cdot v' \, dx = 0$$

$$u'(2) \cdot v(2) - u'(0) \cdot v(0) - \int_0^2 u' \cdot v' \, dx = 0$$

Z (3) wyznaczamy wzór na  $u'$ , oraz odwracamy znaki:

$$\int_0^2 u' \cdot v' \, dx + 20 \cdot v(0) - u(0) \cdot v(0) = 0$$

$$\int_0^2 u' \cdot v' \, dx - u(0) \cdot v(0) = -20 \cdot v(0)$$

Wyznaczamy  $a(u, v)$  oraz  $L(v)$ :

$$a(u, v) = \int_0^2 u' \cdot v' \, dx - u(0) \cdot v(0) \tag{4}$$

$$L(v) = -20 \cdot v(0) \tag{5}$$

## 3 Implementacja

Implementacji algorytmu MRS dokonałem w języku Rust, ponieważ zapewnia on bezpieczeństwo, wydajność i elastyczność.

Przy pisaniu programu wspomagałem się trzema bibliotekami odpowiedzialnymi za: wyliczanie całek, rozwiązanie układu macierzy oraz za stworzenie wykresu.

Całkowanie odbywa się za pomocą metody Gaussa-Legendre'a z czterema punktami kwadratury.

Całość kodu znajduje się w repozytorium na Githubie (podanym w komentarzu do zadania).

### 3.1 Funkcje pomocnicze

Zadeklarowałem parę pomocniczych funkcji, które po krótkce opiszę.

```
1 fn get_a(u_d: impl Fn(f64) -> f64,  
2         v_d: impl Fn(f64) -> f64,  
3         u: impl Fn(f64) -> f64,  
4         v: impl Fn(f64) -> f64,  
5         a: f64, b: f64) -> f64 {  
6  
7     let quad: GaussLegendre =  
8         GaussLegendre::init(4);  
9  
10    quad.integrate(a, b,  
11                  |x| u_d(x)*v_d(x)) - u(0.)*v(0.)  
12 }
```

`get_a` - zwraca  $a(u, v)$

```
1 fn get_l(v: impl Fn(f64) -> f64) -> f64 {  
2     -20.0*v(0.0)  
3 }
```

`get_l` - zwraca  $L(v)$

```

1 fn u_i(i_: usize) -> impl Fn(f64) -> f64{
2     let n: f64 = N as f64;
3     let i = i_ as f64;
4     move |x| {
5         if x > x_i(i - 1.) && x <= x_i(i) {
6             return n/2.0*x - i + 1.0
7         }
8         if x > x_i(i) && x < x_i(i+1.) {
9             return -n/2.0*x + i + 1.0
10        };
11        return 0.0;
12    }
13 }

```

`u_i` - zwraca  $u_i(x)$

```

1 fn ud_i(i_: usize) -> impl Fn(f64) -> f64 {
2     let n = N as f64;
3     let i = i_ as f64;
4     move |x| {
5         if x > x_i(i-1.) && x <= x_i(i) {
6             return n/2.0;
7         }
8         if x > x_i(i) && x < x_i(i+1.) {
9             return -n/2.0;
10        };
11        return 0.0;
12    }
13 }

```

`ud_i` - zwraca  $u'_i(x)$

```

1 fn x_i(i: f64) -> f64 {
2     2.0*(i as f64)/(N as f64)
3 }

```

`x_i` - zwraca  $x_i$

Ponadto, znajdują się funkcję tworzące wykres, w które szczegóły nie będę się zagłębiał.

## 3.2 Funkcja główna

```
1 fn main() {
2     // Macierz wypelniona zerami
3     let mut a: Array2<f64> =
4         Array2::<f64>::zeros((N+1, N+1));
5
6     let n = N as f64;
7     // Wypelniamy macierz zgodnie z
8     // przykladem na zajeciach
9     for i in 0..N {
10         for j in 0..=N {
11             let s: f64;
12             let e: f64;
13             let diff = i.abs_diff(j);
14             if diff > 1 { continue; }
15             if diff == 1 {
16                 s = 2. * f64::max(0.,
17                     f64::min(i as f64, j as f64) / n);
18
19                 e = 2. * f64::min(1.,
20                     f64::max(i as f64, j as f64) / n);
21             } else {
22                 s = 2. * f64::max(0., (i as f64 - 1.) / n);
23                 e = 2. * f64::min(1., (i as f64 + 1.) / n);
24             }
25
26             a[[i, j]] = get_a(ud_i(j), ud_i(i),
27                             u_i(j), u_i(i), s, e);
28         }
29     }
30
31     a[[N, N]] = 1.;
32
33 }
```

```

34 // Macierz B
35 let mut b: Array1<f64> =
36     Array1::<f64>::zeros(N+1);
37
38 for i in 0..N {
39     b[i] = get_l(u_i(i));
40 }
41 b[N] = 0.;
42
43 // Rozwiązujemy układ macierzy
44 let res = a.solve_into(b).unwrap();
45
46 // tworzymy punkty
47 let x: Vec<f64> = (0..=2000)
48     .map(|x| x as f64*0.001).
49     collect::<Vec<f64>>();
50
51 let mut y = vec![0f64; x.len()];
52
53 for i in 0..x.len() {
54     for j in 0..res.len() {
55         let e = u_i(j);
56         y[i] = y[i] + res[j] * e(x[i])
57     }
58 }
59
60 // rysujemy wykres
61 plot(x, y);
62 }

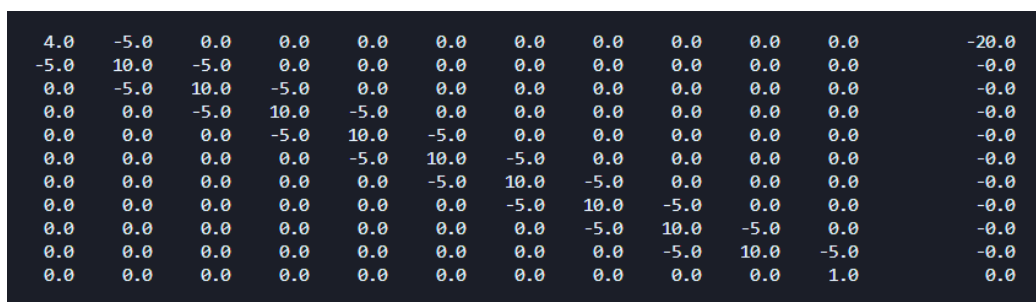
```

Funckja *main()* działa analogicznie do przykładów podanych na zajęciach. Tworzy i wypełnia macierz *A* i *B* po czym rozwiązuje układ równań  $A \cdot X = B$  i ukazuje wyniki na wykresie.

## 4 Wyniki

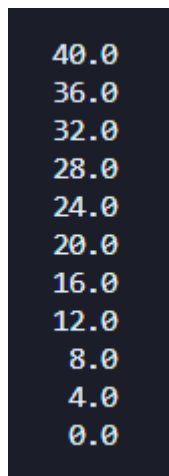
Po uruchomieniu programu w konsoli powinny wyświetlić się przybliżone wartości macierzy (zaokrąglone do pierwszego miejsca po przecinku), oraz okno z wyznaczonym wykresem.

Warto zaznaczyć, że zaokrąglenia mają jedynie na celu ułatwić czytanie, zatem nie mają wpływu na przeprowadzane obliczenia.



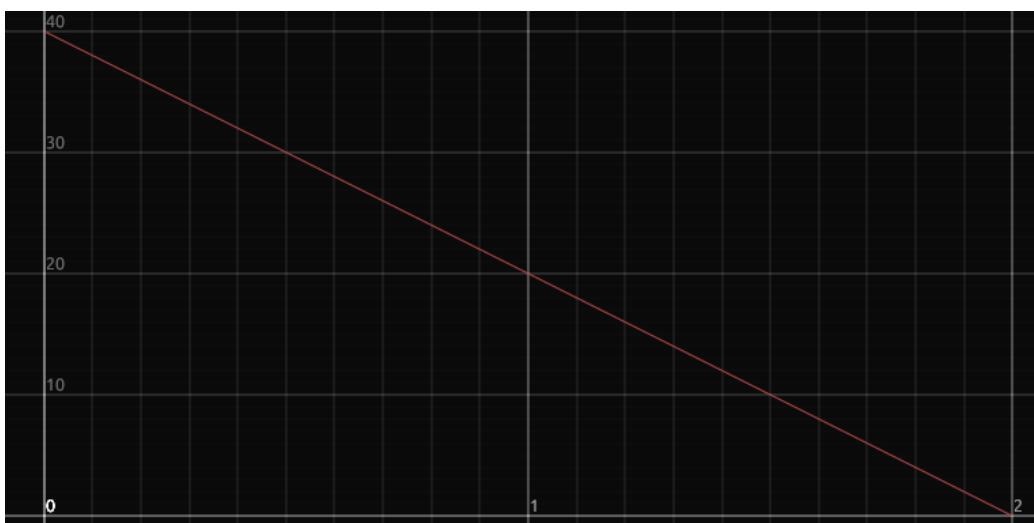
4.0	-5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-20.0
-5.0	10.0	-5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.0
0.0	-5.0	10.0	-5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.0
0.0	0.0	-5.0	10.0	-5.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.0
0.0	0.0	0.0	-5.0	10.0	-5.0	0.0	0.0	0.0	0.0	0.0	-0.0
0.0	0.0	0.0	0.0	-5.0	10.0	-5.0	0.0	0.0	0.0	0.0	-0.0
0.0	0.0	0.0	0.0	0.0	-5.0	10.0	-5.0	0.0	0.0	0.0	-0.0
0.0	0.0	0.0	0.0	0.0	0.0	-5.0	10.0	-5.0	0.0	0.0	-0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.0	10.0	-5.0	0.0	-0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.0	10.0	-5.0	-0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.0	10.0	-0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Rysunek 1. Układ równań.



Rysunek 2. Rozwiązanie układu równań (1).





**Rysunek 3.** Wykres wyliczonej funkcji.

## 5 Wnioski

Po przeanalizowaniu wykresu funkcji  $u(x)$  możemy wyznaczyć jej wzór:

$$u(x) = -20x + 40 \tag{6}$$

### 5.1 Dowód

Dla pewności, możemy podstawić wyznaczoną funkcję pod otrzymane równości i udowodnić jej poprawność.

Ze wzoru (1) wiemy, że funkcja  $u$  jest ciągła (bądź nie zależy w ogóle od  $x$ ), ponieważ jej druga pochodna  $u''(x)$  jest równa zero.

Ponadto, funkcja  $u(x) = -20x + 40$  spełnia warunek Dirichleta (2) i warunek Cauchy'ego (3).