Opis zadania

1 Zadanie

Proszę rozwiązać metodą elementów skończonych następujące równanie różniczkowe

$$(a(x)u'(x))' + b(x)u'(x) + c(x)u(x) = f(x) \tag{1}$$

Dokładne równanie do policzenia jest podane poniżej.
Rozwiązanie składa się z następujących etapów:

1. Proszę wyprowadzić sformułowanie wariacyjne dla przedstawionego układu równań liniowych
2. Proszę napisać procedurę generującą układ równań liniowych, rozwiązującą wygenerowany układ równań liniowych oraz rysujący wykres rozwiązania

2 Wymagania funkcjonalne

Proszę przyjąć zmienną n (ilość elementów) jako parametr uruchomieniowy aplikacji (nie dotyczy studentów wybierających opcję na ocenę 3.0). Dodatkowo proszę rysować wykres wyliczonego przybliżenia funkcji u - dopuszczalne jest rysowanie przez zewnętrzną aplikację - np. gnuplot, Excel. Programy napisane w języku Python będą ocenione na 0 punktów.

4.3 Odkształcenie sprężyste

$$-\frac{d}{dx}\left(E(x)\frac{du(x)}{dx}\right) = 0$$

$$u(2) = 0$$

$$\frac{du(0)}{dx} + u(0) = 10$$

$$E(x) = \begin{cases} 3 & \text{dla } x \in [0,1] \\ 5 & \text{dla } x \in (1,2] \end{cases}$$

Gdzie u to poszukiwana funkcja

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

Obliczenia potrzebne do napisania rozwiązania:

$$-\frac{d}{dx}\left(E(x)\frac{du(x)}{dx}\right)=0 \qquad u(2)=0 \quad \frac{du(0)}{dx}+u(0)=10$$

$$E(x)=\begin{cases}3, & \text{dla } x\in[0,1]\\ 5, & \text{dla } x\in(1,2]\end{cases}$$

$$-E(x)\cdot\frac{d^2u(x)}{dx^2}=0 \quad /\cdot v \qquad x\in[0,2]$$

$$-E(x)\cdot v\cdot\frac{d^2u(x)}{dx^2}=0 \quad /\int_0^2 \qquad u'(0)+u(0)=10$$

$$u'(0)=10-u(0)$$

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

$$u(2)=0 \Rightarrow v(2)=0$$

$$\int_0^2 E(x)v'u'\,dx + 10E(0)v(0) \quad E(0)u(0)v(0)=0$$

$$\int_0^2 E(x)v'u'\,dx - E(0)u(0)v(0) = -10E(0)v(0)$$

$$B(e_i,e_j)=\int_0^2 E(x)\,e_i'(x)\,e_j(x)\,dx - E(0)e_i(0)e_j(0)$$

$$L(e_j)=-10E(0)e_j(0)$$

Opis programu

```
public class LinearFunction {
    private  double a=0.0;
    private  double b=0.0;
    LinearFunction(){}

    LinearFunction (double fx1,
double fx2, double x1, double x2){
        this.a = (fx1-fx2)/(x1-
x2);
        this.b = fx1 - a*x1;
    }
    public double getDerivative(){
        return this.a;
    }
    public double getValue(double
x){
        return this.a*x + this.b;
    }
}
```

Klasa LinearFunction przechowuje oraz oblicza funkcję testującą, zwraca wartości tej funkcji oraz jej pochodnej w zależności od x. Aby rozważać inne funkcje testujące (np. kwadratową) należy dodać nowe metody związane z obliczaniem innej funkcji.

1. Funkcja calculate(n): oblicza wartości oraz argumenty (~200) potrzebnych do narysowania wykresu:

1. Tworzy macierz funkcji $B(e_i,e_j)$ i oblicza wartość wyrażenia dla x=0, gdzie:

$$B(e_i, e_j) = \int_0^2 E(x)e_i(x)e_j(x)dx - E(0)e_i(0)e_j(0)$$

$$\begin{bmatrix} B(e_1,e_1) & \cdots & B(e_n,e_1) \\ \vdots & \ddots & \vdots \\ B(e_0,e_n) & \cdots & B(e_n,e_n) \end{bmatrix} \begin{matrix} w_1 \\ \vdots \\ w_n \end{matrix} = \begin{matrix} L(e_1) \\ \vdots \\ L(e_n) \end{matrix}$$

```
public void calculate(int n) {
    RealMatrix B_uv = new Array2DRowRealMatrix(n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            B_uv.setEntry(i, j, B_ei_ej_x(i, j, 0.0, n));
        }
    }
```

2. Tworzy wektor z szukanymi wartościami w_i

```
RealVector L_v = new ArrayRealVector(n);
for (int i = 0; i < n; i++) {
    L_v.setEntry(i, L_v_x(i, 0.0, n));
}
```

3. Korzystając z funkcji solve z pakietu apache commos rozwiązującej układ AX=B, otrzymuje się tablicę z kolejnymi wartościami w_i

https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/linear/QRDecomposition.html

https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/linear/DecompositionSolver.html#solve(org.apache.commons.math3.linear.RealVector

```
RealVector w = new QRDecomposition(B_uv).getSolver().solve(L_v);
double[] ws = w.toArray();
```

4. Szukane w (ud+u=w ale ud=0) $w \approx \sum_{i=0}^{n} w_i e_i$

5. Przygotowuję dane do wykresu: obliczam wartości u(x) dla kolejnych x. Zapisuje w tablicach wartości oraz argumenty.

```
    double acc = 0.1;
    yi = new double[(int) (domain / acc) + 2];
    xi = new double[(int) (domain / acc) + 2];
    double x = 0.0;
    int y = 0;
    while (x <= domain) {
        for (int i = 0; i < n; i++) {
            yi[y] += e_i(i, x, n) * ws[i];
        }
        xi[y] = x;
        y++;
        x += acc;
    }
}
```

2. Funkcja L_v_x(i,x,n): oblicza wartość $L_i(x)$ dla macierzy

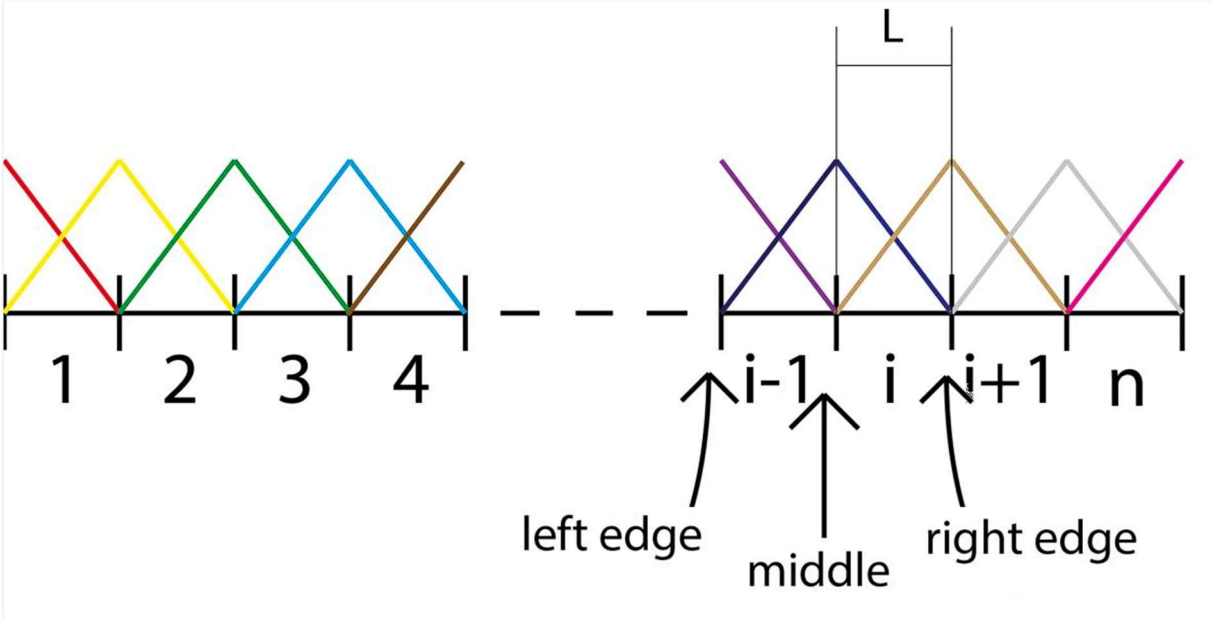
```
private double L_v_x(int i, double x, int n) {
    return -10 * E(x) * e_i(i, x, n);
}
```

3. Funkcja B_e_i_e_j_x(i,j,x,n): oblicza wartość $B(e_i,e_j)(x)$ dla macierzy

```
private double B_ei_ej_x(int i, int j, double x, int n) {
    double integral = 0.0;
    if (Math.abs(j - i) <= 1) {
        integral = getIntegral(i, j, n);
    }
    return integral - E(x) * e_i(i, x, n) * e_i(j, x, n);
}
```

4. Funkcja E(x): zwraca wartość funkcji E(x) zdefiniowanej w poleceniu w zależności od argumentu

```
private static double E(double x) {
    if (x >= 0 && x <= 1.0) {
        return 3.0;
    } else if (x > 1 && x <= 2) {
        return 5.0;
    } else throw new IllegalArgumentException("out of domain");
}
```



5. Funkcja $e_i(i,x,n)$: oblicza wartość funkcji nad i-tym przedziałem w zleżności od podanego x;

```
private static double e_i(int i, double x, int n) {
    double l = domain / n;
    double middle = l * i;
    double left_edge = l * (i - 1);
    double right_edge = l * (i + 1);
    if (x < left_edge || x > right_edge) {
        return new LinearFunction().getValue(x);
    }
    if (x >= middle) {
        return new LinearFunction(1, 0, middle, right_edge).getValue(x);
    } else {
        return new LinearFunction(0, 1, left_edge, middle).getValue(x);
    }
}
```

6. Funkcja e_i_dx(i,x,n): oblicza wartość pochodnej funkcji nad i-tym przedziałem w zleżności od podanego x;

```
private static double e_i_dx(int i, double x, int n) {
    double l = domain / n;
    double middle = l * i;
    double left_edge = l * (i - 1);
    double right_edge = l * (i + 1);
    if (x < left_edge || x > right_edge) {
        return new LinearFunction().getDerivative();
    }
    if (x >= middle) {
        return new LinearFunction(1, 0, middle,right_edge).getDerivative();
    } else {
        return new LinearFunction(0, 1, left_edge, middle).getDerivative();
    }
}
```