# Sprawozdanie z przedmiotu Metody obliczeniowe

# Dział: Aproksymacja

Wykonał:
Imię i nazwisko Kacper Nalbach
Nr indeksu 80255
Grupa 4

Prowadzący Agnieszka Bołtuć

**1. Przykład nr 11**

$$\sqrt{3x^3 - x + 5} \qquad a=-1 \qquad b=1 \qquad x=0.25$$

**2. Rozwiązanie rachunkowe przykładu wybraną metodą**

Kacper Nalbach

Przykład 11. $\sqrt{3x^3-x+5}$ $a=-1$ $b=1$ $x=0{,}25$

$p(x)=1$

Aproksymacja średniokwadratowa (ciągła) $n=2$

$$\begin{bmatrix} \int_{-1}^{1} dx & \int_{-1}^{1} x^1\,dx & \int_{-1}^{1} x^2\,dx \\ \int_{-1}^{1} x^1\,dx & \int_{-1}^{1} x^1\cdot x^1\,dx & \int_{-1}^{1} x^1 x^2\,dx \\ \int_{-1}^{1} x^2\,dx & \int_{-1}^{1} x^2\cdot x^1\,dx & \int_{-1}^{1} x^2\cdot x^2\,dx \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \int_{-1}^{1} \sqrt{3x^3-x+5}\,dx \\ \int_{-1}^{1} x^1\cdot\sqrt{3x^3-x+5}\,dx \\ \int_{-1}^{1} x^2\cdot\sqrt{3x^3-x+5}\,dx \end{bmatrix}$$

$$\int_{-1}^{1} dx = x\Big|_{-1}^{1} = 1-(-1) = 2 \qquad \int_{-1}^{1} x\,dx = \frac{1}{2}x^2\Big|_{-1}^{1} = \frac{1}{2}-\frac{1}{2} = 0$$

$$\int_{-1}^{1} x^2\,dx = \frac{1}{3}x^3\Big|_{-1}^{1} = \frac{1}{3}-\left(-\frac{1}{3}\right) = \frac{2}{3} \qquad \int_{-1}^{1} x^3\,dx = \frac{1}{4}x^4\Big|_{-1}^{1} = \frac{1}{4}-\frac{1}{4} = 0$$

$$\int_{-1}^{1} x^4\,dx = \frac{1}{5}x^5\Big|_{-1}^{1} = \frac{1}{5}-\left(-\frac{1}{5}\right) = \frac{2}{5} \qquad \int_{-1}^{1} \sqrt{3x^3-x+5}\,dx = 4{,}88$$

$$\int_{-1}^{1} x\sqrt{3x^3-x+5}\,dx = -0{,}12 \qquad \int_{-1}^{1} x^2\sqrt{3x^3-x+5}\,dx = 1{,}73$$

$$\begin{bmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 4{,}88 \\ -0{,}12 \\ 1{,}73 \end{bmatrix}$$

$$\cancel{2a_0 + a_2 = 4{,}88}$$
$$\cancel{a_1 = -0{,}12}$$
$$\cancel{\tfrac{2}{3}a_1 + \tfrac{2}{5}a_2 = 1{,}73}$$

$$\cancel{\tfrac{2}{3}\cdot(-0{,}12) + \tfrac{2}{5}a_2 = 1{,}73}$$
$$\cancel{\tfrac{2}{5}a_2 = 1{,}73 - \tfrac{2}{3}\cdot(-0{,}12)}$$
$$\cancel{2a_2 = 5\cdot(1{,}73+0{,}08)}$$
$$\cancel{2a_2 = 8{,}89}$$

$$\cancel{2a_0 + 4{,}445 = 4{,}88}$$
$$\cancel{2a_0 = 0{,}435}$$
$$\cancel{a_0 = 0{,}2175}$$

$$\begin{cases} 2a_0 + \frac{2}{3}a_2 = 4{,}88 \\ \frac{2}{3}a_1 = -0{,}12 \\ \frac{2}{3}a_0 + \frac{2}{5}a_2 = 1{,}73 \end{cases}$$

$$a_1 = -0{,}18$$

$$a_0 = \frac{\left(4{,}88 - \frac{2}{3}a_2\right)}{2} \qquad a_0 = \frac{4{,}88 - \frac{2}{3}\cdot 0{,}195}{2} = 2{,}375$$

$$\frac{\frac{2}{3}\left(4{,}88 - \frac{2}{3}a_2\right)}{\cancel{2}} + \frac{2}{5}a_2 = 1{,}73$$

$$\frac{4{,}88}{3} - \frac{2}{3}\cdot\frac{1}{3}a_2 + \frac{2}{5}a_2 = 1{,}73$$

$$1{,}626 - \frac{2}{9}a_2 + \frac{2}{5}a_2 = 1{,}73$$

$$\frac{8}{45}a_2 = 0{,}104$$

$$a_2 = 0{,}195$$

$$W(x) = 2{,}375 - 0{,}18x + 0{,}195x^2$$

$$W(0{,}25) = 2{,}375 - 0{,}18\cdot 0{,}25 + 0{,}195\cdot(0{,}25)^2 = 2{,}3421$$

**3. Rozwiązania przykładu poszczególnymi programami**

***a) metoda Wielomianów Ortogonalnych***

```
public class WielomianOrtagonalny {
    double[] c_i;
    static int a;
    static int b;
    int n;

    static {
        a = -1;
        b = 1;
    }

    public static double wielomian(double x, double i) {
        if (i == 0) {
            return 1;
        } else if (i == 1) {
            return x;
        } else {
            return (1.0 / ((i - 1) + 1)) * ((2 * (i - 1)) + 1) * x * wielomian(x, i -
1) - ((i - 1) / ((i - 1) + 1)) * wielomian(x, i - 2);
```

```java
        }
    }

    public double oblicz(double x, int n) {
        this.n = n;
        oblicz_c_i(x);
        double suma = 0;
        for (int i = 0; i <= n; i++) {
            suma += c_i[i] * wielomian(x, i);
        }
        System.out.print("\nn = " + n + ", ");
        return suma;
    }

    private void oblicz_c_i(double x) {
        c_i = new double[n + 1];
        for (int i = 0; i <= n; i++) {
            c_i[i] = 1/lambda(i) * (new MetodaSimpsona(-1, 1, i, true).oblicz());
        }
    }

    private double lambda(int i) {
        return new MetodaSimpsona(-1, 1, i).oblicz();
    }

    public static void main(String[] args) {
        WielomianOrtagonalny wo = new WielomianOrtagonalny();
        double x = 0.25;
        System.out.println(wo.oblicz(x, 2));
        System.out.println(wo.oblicz(x, 4));
        System.out.println(wo.oblicz(x, 6));
        System.out.println(wo.oblicz(x, 8));
        System.out.println(wo.oblicz(x, 16));
        System.out.println("Funkcja = " + Math.sqrt(3 * Math.pow(x, 3) - x + 5));
    }
}
```

```java
import java.util.Arrays;

public class MetodaSimpsona {
    int n = 30;
    int a;
    int b;
    double h;
    double[] f_x;
    double[] t_i;
    int i_wo;
    boolean flag = false;

    MetodaSimpsona(int a, int b, int i_wo) {
        this.a = a;
        this.b = b;
        this.f_x = new double[n + 1];
```

```java
        this.t_i = new double[n];
        this.i_wo = i_wo;
    }

    MetodaSimpsona(int a, int b, int i_wo, boolean flag) {
        this.a = a;
        this.b = b;
        this.f_x = new double[n + 1];
        this.t_i = new double[n];
        this.i_wo = i_wo;
        this.flag = flag;
    }

    public double oblicz() {
        f_x_calculate();
        double sum = 0;
        for (int i = 1; i < n; i++) {
            sum += funkcjaPodcalkowa(f_x[i]) * 2;
        }
        for (int i = 0; i < n; i++) {
            sum += funkcjaPodcalkowa(t_i[i]) * 4;
        }
        sum += funkcjaPodcalkowa(f_x[0]) + funkcjaPodcalkowa(f_x[n]);
        sum *= h / 3;
        return sum;
    }

    private double funkcjaPodcalkowa(double f_x1) {
        if (!flag) {
            return Math.pow(WielomianOrtagonalny.wielomian(f_x1, i_wo), 2);
        } else {
            return WielomianOrtagonalny.wielomian(f_x1, i_wo) * Math.sqrt(3 *
Math.pow(f_x1, 3) - f_x1 + 5);
        }
    }

    void f_x_calculate() {
        for (int i = 1; i < n; i++) {
            f_x[i] = a + (i * 1.0 *(b - a)) / n; // 1 + 3 / 0.75
        }
        f_x[0] = a;
        f_x[n] = b;
        h = (f_x[2] - f_x[1]) / 2;
        for (int i = 0; i < n; i++) {
            t_i[i] = (f_x[i + 1] + f_x[i]) / 2;
        }
    }
}
```

```
n = 2, 2.283600088667846
n = 4, 2.1897408204304094
n = 6, 2.1905046611200527
n = 8, 2.1898029073977505
n = 16, 2.1861156423996393
Funkcja = 2.190176933491904
```

***b) Metoda Najmniejszych Kwadratów***

```
import java.util.Arrays;

public class MetodaNajmniejszychKwadratow {
    static double[] x;
    static double[] y;
    int n;
    int m;
    static double[][] skj;
    static double[] tk;
    static double[] a;

    static {

//        {-1, -0.5, 0, 0.5, 1

//        x[0] = -1;
//        x[1] = -0.5;
//        x[2] = 0;
//        x[3] = 0.5;
//        x[4] = 1;

//        y[0] = Math.sqrt(3);
//        y[1] = 2.26384;
//        y[2] = Math.sqrt(5);
//        y[3] = 2.20794;
//        y[4] = Math.sqrt(7);
    }

    public void calc_skj() {
        for (int i = 0; i <= m ; i++) {
            for (int j = 0; j <= m; j++) {
                double suma = 0;
                for (int k = 0; k < n; k++) {
                    suma += Math.pow(x[k], i + j);
                }
                skj[i][j] = suma;
            }
        }
    }

    public void calc_tk() {
        for (int k = 0; k <= m; k++) {
            double suma = 0;
            for (int i = 0; i < n; i++) {
                suma += Math.pow(x[i], k) * y[i];
            }
```

```java
            tk[k] = suma;
        }
    }

    public void calc_points() {
        x = new double[5];
        y = new double[n];

        x[0] = -1;
        x[1] = -0.5;
        x[2] = 0;
        x[3] = 0.5;
        x[4] = 1;

        for (int i = 0; i < n; i++) {
            y[i] = Math.sqrt(3 * Math.pow(x[i], 3) - x[i] + 5);
        }
    }

    public void calc(double x, int m, int n) {
        this.n = n;
        this.m = m;
        calc_points();

        skj = new double[m + 1][m + 1];
        tk = new double[m + 1];
        a = new double[m + 1];
        calc_skj();
        calc_tk();
        a = Gauss.lsolve(skj, tk);

        System.out.print(a[0]);
        double wx = a[0];
        for (int i = 1; i <= m; i++) {
            if (i > 1) {
                System.out.print(" + " + a[i] + "x" + "^" + i);
            } else {
                System.out.print(" + " + a[i] + "x");
            }
            wx += Math.pow(x, i) * a[i];
        }

        System.out.println("\nW(" + x + ") = " + wx + " dla n = " + m);
    }

    public static void main(String[] args) {
        MetodaNajmniejszychKwadratow mnk = new MetodaNajmniejszychKwadratow();
        for (int i = 1; i < 6 ; i++) {
            try {
                mnk.calc(0.25, i, 5);
            } catch (Exception ignored) {
            }
        }
```

```
    }
}
```

```
2.2171314 + 0.354299x
W(0.25) = 2.30570615 dla n = 1
2.244034 + 0.354299x + -0.0538052x^2
W(0.25) = 2.329245925 dla n = 2
2.244034 + -0.2268248x + -0.0538052x^2 + 0.6836751x^3
W(0.25) = 2.1946473984375 dla n = 3
2.236068 + -0.2268248x + 0.0147904x^2 + 0.6836751x^3 + -0.0619573x^4
W(0.25) = 2.190726602734375 dla n = 4
```

c) metoda średniokwadratowa ciągła

```
import java.util.Arrays;

public class SrednioKwadratowa {
    static int a;
    static int b;
    static int p;
    int n;
    double[][] matrix;
    double[] matrix_bi;
    double[] niewiadome;

    static {
        a = -1;
        b = 1;
        p = 1;
    }

    public double funkcja(double x) {
        return Math.sqrt(3 * Math.pow(x, 3) - x + 5);
    }

    public double wielomian(double x, int n){
        this.n = n;
        matrix = new double[n + 1][n + 1];
        matrix_bi = new double[n + 1];
        oblicz_matrix();
        double sum = 0;
        for (int i = 0; i <= n; i++) {
            sum += niewiadome[i] * Math.pow(x, i);
        }
        return sum;
    }

    public void oblicz_matrix() {
        MetodaSimpsona mt = new MetodaSimpsona(a, b);
        MetodaSimpsona mt2 = new MetodaSimpsona(a, b, true);
        for (int i = 0; i <= n; i++) {
            for (int j = 0; j <= n; j++) {
```

```
                matrix[i][j] = mt.oblicz(i, j);
            }
            matrix_bi[i] = mt2.oblicz(i, 0);
        }
        niewiadome = new double[n + 1];
        niewiadome = Gauss.lsolve(matrix, matrix_bi);
    }

    public static void main(String[] args) {
        SrednioKwadratowa sk = new SrednioKwadratowa();
        System.out.println("Funkcja = " + sk.funkcja(0.25) + " wielomian = " +
sk.wielomian(0.25, 2) + " n = " + 2);
        System.out.println("Funkcja = " + sk.funkcja(0.25) + " wielomian = " +
sk.wielomian(0.25, 4) + " n = " + 4);
        System.out.println("Funkcja = " + sk.funkcja(0.25) + " wielomian = " +
sk.wielomian(0.25, 6) + " n = " + 6);
        System.out.println("Funkcja = " + sk.funkcja(0.25) + " wielomian = " +
sk.wielomian(1 0.25 8) + " n = " + 8);
        System.out.println("Funkcja = " + sk.funkcja(0.25) + " wielomian = " +
sk.wielomian(0.25, 16) + " n = " + 16);
    }
}
```

```
Funkcja = 2.190176933491904 wielomian = 2.2836001500000003 n = 2
Funkcja = 2.190176933491904 wielomian = 2.1897386601562494 n = 4
Funkcja = 2.190176933491904 wielomian = 2.1904733982910156 n = 6
Funkcja = 2.190176933491904 wielomian = 2.1901289031036386 n = 8
Funkcja = 2.190176933491904 wielomian = 2.1901770566830976 n = 16
```

## 4. Analiza

***Funkcja = 2.190176933491904***

| Metoda | N = 2 | N = 4 | N = 6 | N = 8 | N = 16 |
|---|---|---|---|---|---|
| ***Wielomianów Ortogonalnych*** | ***2.283600088667846*** | ***2.1897408204304094*** | ***2.1905046611200527*** | ***2.1898029073977505*** | ***2.1861156423996393*** |
| **średniokwadratowa ciągła** | 2.2836001500000003 | 2.1897386601562494 | 2.1904733982910156 | 2.1901289031036386 | 2.1901770566830976 |

| Metoda | N = 1 | N = 2 | N = 3 | N = 4 |
|---|---|---|---|---|
| ***Najmniejszych Kwadratów*** | **2.30570615** | **2.329245925** | **2.1946473984375** | **2.190726602734375** |

## 5. Wnioski

Metoda średniokwadratowa ciągła na moim przykładzie działa najlepiej, zwłaszcza dla większych n, metoda wielomianów ortogonalnych dla większych n zaczyna oddalać się od wyniku. Metoda najmniejszych kwadratów nie działa dla n większych niż 4, ponieważ wtedy mamy macierz o wyznaczniku 0.