Sprawozdanie z przedmiotu Metody obliczeniowe

Dział: równania nieliniowe

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

Prowadzący Agnieszka Bołtuć

1. Przykład nr 11

11. (-4,-1) $12x^2 - 6x - 32$

2. Rozwiązanie rachunkowe przykładu wybraną metodą

Kacper Nalbach

$(-4,-1)$ $\quad 12x^2-6x-32$

Metoda stycznych

$a=-4 \quad b=-1 \quad \varepsilon=0{,}01$

$f(x)=12x^2-6x-32 \qquad f'(x)=24x-6 \qquad f''(x)=24$

$f'(a)\cdot f'(b)\geq 0$

$f'(-4)\cdot f'(-1)=(-102)\cdot(-30)=3060$

$f''(a)\cdot f''(b)\geq 0$

$f''(-4)\cdot f''(-1)=24\cdot 24=576$

Warunek koniecznej spełniony

$f(-4)=184 \qquad f'(-4)=24$

$x_0=a$

$x_0=-4$

$x_1=x_0-\frac{f(x_0)}{f'(x_0)}=-4-\frac{184}{-102}=-2{,}19607 \qquad f(-2{,}19607)=39{,}0491$

$39{,}0491>0{,}01$ oraz $|-2{,}19607+4|>0{,}01$

$x_2=-2{,}19607-\frac{39{,}0491}{-58{,}70568}=-1{,}53090 \qquad f(-1{,}53090)=5{,}30925$

$5{,}30925>0{,}01$ oraz $|-1{,}53090+2{,}19607|>0{,}01$

$x_3=-1{,}53090-\frac{5{,}30925}{-42{,}7416}=-1{,}40668 \qquad f(-1{,}40668)=0{,}18506$

$0{,}18506>0{,}01$ oraz $|-1{,}40668+1{,}53090|>0{,}01$

$x_4=-1{,}40668-\frac{0{,}18506}{-39{,}76032}=-1{,}40202 \qquad f(-1{,}40202)=0{,}000409648$

$0{,}000409648<0{,}01$

Warunek stopu spełniony, rozwiązanie: $-1{,}40202$

3. Rozwiązania przykładu poszczególnymi programami

a) metoda bisekcji

```
public class MetodaBisekcji {
    static double a;
    static double b;
    static double epsilon;
    double x_sr;
    int iteration = 1;

    static {
        a = -4;
        b = -1;
        epsilon = 0.05;
    }

    public void warunek() {
        if (function(a) * function(b) < 0) {
            System.out.println("Warunek konieczny spełniony");
        } else {
            System.out.println("Warunek konieczny niespełniony");
        }
    }

    public boolean warunek_x_sr() {
        return function(x_sr) * function(a) < 0;
    }

    public double function(double x) {
        return 12 * Math.pow(x, 2) - 6 * x - 32;
    }


    public void calc_x_sr() {
        x_sr = (a + b) / 2;
    }

    public void calc_przedzial() {
        if (warunek_x_sr()) {
            b = x_sr;
        } else {
            a = x_sr;
        }
    }

    public void oblicz() {
        warunek();
        calc_x_sr();
        if (x_sr == 0) {
            System.out.println("Wynik = " + x_sr);
```

```
        } else {
            iteration++;
        }
        while (!(Math.abs(function(x_sr)) < epsilon)) {
            calc_przedzial();
            calc_x_sr();
            iteration++;
        }

        System.out.println("Iteracja: " + iteration + " rozwiazanie: " + x_sr + ",
funkcja: " + function(x_sr));
    }

    public static void main(String[] args) {
        MetodaBisekcji mb = new MetodaBisekcji();
        mb.oblicz();
    }
}
```

```
Warunek konieczny spełniony
Iteracja: 11 rozwiazanie: -1.4013671875, funkcja: -0.025836944580078125
```

b) metoda stycznych

```
public class MetodaStycznych {
    static double a;
    static double b;
    static double epsilon;
    int iteration = 1;
    int n = 1;

    static {
        a = -4;
        b = -1;
        epsilon = 0.01;
    }

    public double func(double x) {
        return 12 * Math.pow(x, 2) - 6 * x - 32;
    }

    public double func_prim(double x) {
        return 24 * x - 6;
    }

    public double func_bis(double x) {
        return 24;
    }

    public void warunek() {
        if (func(a) * func(b) < 0 && func_prim(a) * func_prim(b) >= 0 && func_bis(a)
```

```
* func_bis(b) >= 0) {
            System.out.println("Warunek konieczny spełniony");
        } else {
            System.out.println("Warunek konieczny niespełniony");
        }
    }

    public double xn1(int n) {
        if (n == 0) {
            return x0();
        } else {
            return xn1(n - 1) - func(xn1(n - 1)) / func_prim(xn1(n - 1));
        }
    }

    public double x0() {
        if (func_bis(a) * func(a) >= 0) {
            return a;
        } else {
            return b;
        }
    }

    public void oblicz() {
        warunek();
        double xn1_func = func(xn1(n));
        double xn1_n1 = xn1(n);
        double xn1_n = xn1(n - 1);
        while(Math.abs(xn1_func) >= epsilon && Math.abs(xn1_n1 - xn1_n) >= epsilon) {
            n++;
            iteration++;
            xn1_func = func(xn1(n));
            xn1_n1 = xn1(n);
            xn1_n = xn1(n - 1);
        }
        System.out.println("Iteracja: " + iteration + " rozwiazanie: " + xn1_n1 + ",
funkcja: " + func(xn1_n1));
    }

    public static void main(String[] args) {
        MetodaStycznych ms = new MetodaStycznych();
        ms.oblicz();
    }
}
```

```
Warunek konieczny spełniony
Iteracja: 4 rozwiazanie: -1.4020255314547025, funkcja: 2.602789383132631E-4
```

c) metoda siecznych

```
public class MetodaSiecznych {
```

```java
    static double a;
    static double b;
    static double epsilon;
    int iteration = 1;
    int n = 1;

    static {
        a = -4;
        b = -1;
        epsilon = 0.01;
    }

    public double func(double x) {
        return 12 * Math.pow(x, 2) - 6 * x - 32;
    }

    public double func_prim(double x) {
        return 24 * x - 6;
    }

    public double func_bis(double x) {
        return 24;
    }

    public void warunek() {
        if (func(a) * func(b) < 0) {
            System.out.println("Warunek konieczny spełniony");
        } else {
            System.out.println("Warunek konieczny niespełniony");
        }
    }

    public boolean nieruchoma_a() {
        return func(a) * func_bis(a) >= 0;
    }

    public double xn1_a(int n) {
        if (n == 0) {
            return b;
        } else {
            return xn1_a(n - 1) - (func(xn1_a(n - 1)) / (func(xn1_a(n - 1)) -
func(a))) * (xn1_a(n - 1) - a);
        }
    }

    public double xn1_b(int n) {
        if (n == 0) {
            return a;
        } else {
            return xn1_b(n - 1) - (func(xn1_b(n - 1)) / (func(b) - func(xn1_b(n -
1)))) * (b - xn1_b(n - 1));
        }
    }
```

```java
    public void oblicz() {
        warunek();
        if (nieruchoma_a()) {
            while(Math.abs(func(xn1_a(n))) > epsilon && Math.abs(xn1_a(n) - xn1_a(n -
1)) > epsilon) {
                n++;
                iteration++;
            }
            System.out.println("Iteracja: " + iteration + " rozwiazanie: " + xn1_a(n)
+ ", funkcja: " + func(xn1_a(n)));
        } else {
            while(Math.abs(func(xn1_b(n))) > epsilon && Math.abs(xn1_b(n) - xn1_b(n -
1)) > epsilon) {
                n++;
                iteration++;
            }
            System.out.println("Iteracja: " + iteration + " rozwiazanie: " + xn1_b(n)
+ ", funkcja: " + func(xn1_b(n)));
        }
    }

    public static void main(String[] args) {
        MetodaSiecznych ms = new MetodaSiecznych();
        ms.oblicz();
    }
}
```

```
Warunek konieczny spełniony
Iteracja: 5 rozwiazanie: -1.3944719545108155, funkcja: -0.2985438899285384
```

4. Analiza

	E = 0.5	E = 0.05	E = 0.005	E = 0.005	E = 0.0005
Metoda Bisekcji	-1.3984375, 8 iteracji	-1.4013671875, 12 iteracji	-1.402099609375, 15 iteracji	-1.402008056640625, 19 iteracji	-1.4020195007324219, 23 iteracje
Metoda Stycznych	-1.406682774365051, 3 iteracje	-1.4020255314547025, 4 iteracje	-1.4020255314547025, 4 iteracje	-1.4020255314547025, 4 iteracje	-1.4020189668129603, 5 iteracji
Metoda Siecznych	-1.2121212121212122, 1.3634359559103002.	-1.3986926307115066	-1.4017349968586483, 9 iteracji	-1.4019947446525887, 12 iteracji	

	1 iteracja	3 iteracje	, 6 iteracji		

5. Wnioski

Metoda stycznych najszybciej oraz najdokładniej znalazła rozwiązanie dla mojego przykładu, gdzie już przy dokładności 0.5 wynik był najbliższy prawdziwemu w porównaniu do innych metod. Najwięcej iteracji dla dokładności 0.0005 potrzebowała metoda bisekcji.