Przygotowali: Jan Kąkol i Maciej Kucharski

Zadanie 2.

Zbadać eksperymentalnie zbieżność metody Newtona dla układu

przy różnych wczytywanych z wejścia punktach startowych (.

Metoda obliczeniowa: Metoda Newtona dla układów równań

Dla układu z funkcją zmiennej definiujemy ciąg Newtona

gdzie jest ustalony.

Zakładamy, że jest klasy . Pochodna jest macierzą , a jest odwrotną. Ciąg Newtona będzie poprawnie zdefiniowany, gdy na każdym kroku macierz będzie odwracalna.

Uwaga

Wektor można wyznaczyć, jako rozwiązanie liniowego układu

Przykład:

Dane początkowe

Krótki opis programu.

wczytywanieDanych() – metoda odpowiedzialna za pobieranie informacji od użytkownika niezbędnych do obliczeń.

obliczenieFxy() – metoda wyliczająca wartość funkcji

obliczenieFprimXY() – metoda wyliczająca wartość funkcji

macierzOdwrotna() – metoda obliczająca macierz odwrotną

wyliczXY() – metoda wyliczająca wartość macierzy w punktach:

wyliczWynik() – metoda wyliczająca wynik

sprawdzWynik() – sprawdzenie czy otrzymany wynik mieści się w kryteriach podanych przez użytkownika

szukajDalej() – metoda decydująca czy program będzie wykonywał kolejne obliczenia, czy wynik jest już na tyle jednoznaczny, że można podać wynik

Opis wejścia wyjścia:

Program do działania potrzebuje kilku parametrów:

Podaj wartosc poczatkowa dla x[0]

Podajemy wartość początkową dla

Podaj wartosc poczatkowa dla y[0]

Podajemy wartość początkową dla

Teraz podaj maksymalną liczbe iteracji:

Podajemy maksymalną liczbę iteracji (tj.: maksymalną ilość obliczeń programu – tyle obliczeń program będzie wykonywał, jeżeli wcześniej nie znajdzie odpowiedniego wyniku)

Okresl dokladnosc: (zamiast kropki, daj przecinek)

Podajemy dokładność – jeżeli wartość bezwzględna wyniku zacznie być mniejsza niż ta podana wartość program przerwie obliczenia – funkcja dla tych zmienny jest zbieżna

Wyjście:

--------Twoje dane---------

x[0] 1.0

y[0] 3.0

Iteration count 20

Accuracy 0.3

Uklad jest rozbiezny.

Najważniejszym komunikatem wyjścia jest ostatnia linijka: wynik jest rozbieżny lub zbierzny.

Kilka przykładowych uruchomień programu:

--------Twoje dane---------

x[0] 320.0

y[0] 21.0

Iteration count 100

Accuracy 1.0

Uklad jest rozbiezny.

--------Twoje dane---------

x[0] 0.2

y[0] 0.9

Iteration count 100

Accuracy 0.9

Uklad jest zbiezny.

--------Twoje dane---------

x[0] 0.8

y[0] 0.2

Iteration count 1

Accuracy 0.9

Uklad jest rozbiezny.

--------Twoje dane---------

x[0] 23.0

y[0] 0.8

Iteration count 100

Accuracy 1.0

Uklad jest rozbiezny.

Program

**package** com.zbieznoscNewtona.service;

**import** java.util.Scanner;

// X[0] = startTab[0][0] Y[0] = startTab[0][1]

**public** **class** zbieznoscNewtona {

**static** Scanner *in* = **new** Scanner(System.*in*);

**static** **double** *startTab*[][] = **new** **double**[1][2];

**static** **double**[][] *bestXY* = **new** **double**[1][2];

**static** **int** *maxIterationCount* = 0;

**static** **double** *accuracy* = 0;

**final** **static** **double** *DECIMAL* = 1000.0;

**public** **static** **void** wczytywanieDanych() {

System.*out*

.println("Witaj w programie badajacym eksperymentalnie zbieznosc Newtona");

System.*out*.println("Podaj wartosc poczatkowa dla x[0]");

*startTab*[0][0] = *in*.nextDouble();

System.*out*.println("Podaj wartosc poczatkowa dla y[0]");

*startTab*[0][1] = *in*.nextDouble();

System.*out*.println("Teraz podaj maksymalną liczbe iteracji: ");

*maxIterationCount* = *in*.nextInt();

System.*out*

.println("Okresl dokladnosc: (zamiast kropki, daj przecinek)");

*accuracy* = *in*.nextDouble();

System.*out*.println();

System.*out*.println("--------Twoje dane---------");

System.*out*.println("x[0] " + *startTab*[0][0]);

System.*out*.println("y[0] " + *startTab*[0][1]);

System.*out*.println("Iteration count " + *maxIterationCount*);

System.*out*.println("Accuracy " + *accuracy*);

}

**private** **static** **double**[][] obliczenieFxy(**double**[][] aktualnyWynik) {

**double** Fxy[][] = **new** **double**[1][2];

Fxy[0][0] = Math

.*round*(((2 \* aktualnyWynik[0][0] \* aktualnyWynik[0][1]) - 3)

\* *DECIMAL*)

/ *DECIMAL*;

Fxy[0][1] = Math

.*round*((((aktualnyWynik[0][0] \* aktualnyWynik[0][0]) - aktualnyWynik[0][1]) - 2)

\* *DECIMAL*)

/ *DECIMAL*;

/\*

\* System.out.println("\n------WYNIK F(x,y)---------");

\* System.out.println("| " + Fxy[0][0] + " |"); System.out.println("| "

\* + Fxy[0][1] + " |");

\*/

**return** Fxy;

}

**private** **static** **double**[][] obliczenieFprimXY(**double**[][] aktualnyWynik) {

**double** fPxy[][] = **new** **double**[2][2];

fPxy[0][0] = Math.*round*((2 \* aktualnyWynik[0][1]) \* *DECIMAL*) / *DECIMAL*;

fPxy[0][1] = Math.*round*((2 \* aktualnyWynik[0][0]) \* *DECIMAL*) / *DECIMAL*;

fPxy[1][0] = Math.*round*((2 \* aktualnyWynik[0][0]) \* *DECIMAL*) / *DECIMAL*;

fPxy[1][1] = -1;

/\*

\* System.out.println("\n\n-----WYNIK f`(x,y)-----------");

\* System.out.println(fPxy[0][0] + " " + fPxy[0][1]);

\* System.out.println(fPxy[1][0] + " " + fPxy[1][1]);

\*/

**return** fPxy;

}

**private** **static** **double**[][] macierzOdwrotna(**double**[][] fPxy) {

**double** wyznacznik = 0;

**double** fodwrotna[][] = **new** **double**[2][2];

wyznacznik = Math

.*round*(((fPxy[0][0] \* fPxy[1][1]) - (fPxy[1][0] \* fPxy[0][1]))

\* *DECIMAL*)

/ *DECIMAL*;

**if** (wyznacznik == 0) {

System.*out*.println("Wyznacznik rowny zero");

System.*exit*(0);

}

//System.out.println("Wyznacznik: " + wyznacznik);

fodwrotna[0][0] = Math.*round*(((1 / wyznacznik) \* fPxy[1][1]) \* *DECIMAL*)

/ *DECIMAL*;

fodwrotna[1][1] = Math.*round*(((1 / wyznacznik) \* fPxy[0][0]) \* *DECIMAL*)

/ *DECIMAL*;

fodwrotna[0][1] = Math.*round*(((1 / wyznacznik) \* (-1) \* fPxy[0][1])

\* *DECIMAL*)

/ *DECIMAL*;

fodwrotna[1][0] = Math.*round*(((1 / wyznacznik) \* (-1) \* fPxy[1][0])

\* *DECIMAL*)

/ *DECIMAL*;

/\*

\* System.out.println("---------Macierz odwrotna---------");

\* System.out.println(fodwrotna[0][0] + " " + fodwrotna[0][1]);

\* System.out.println(fodwrotna[1][0] + " " + fodwrotna[1][1]);

\*/

**return** fPxy;

}

**private** **static** **double**[][] wyliczXY(**double**[][] aktualnyWynik,

**double**[][] fOdwrotna, **double**[][] fxy) {

**double** wynik[][] = **new** **double**[1][2];

**double** tmp[][] = **new** **double**[1][2];

tmp[0][0] = Math

.*round*(((fOdwrotna[0][0] \* fxy[0][0]) + (fOdwrotna[1][0] \* fxy[0][1])

\* *DECIMAL*)

/ *DECIMAL*);

tmp[0][1] = Math

.*round*(((fOdwrotna[0][1] \* fxy[0][0]) + (fOdwrotna[1][1] \* fxy[0][1]))

\* *DECIMAL*)

/ *DECIMAL*;

wynik[0][0] = Math.*round*((aktualnyWynik[0][0] - tmp[0][0]) \* *DECIMAL*)

/ *DECIMAL*;

wynik[0][1] = Math.*round*((aktualnyWynik[0][1] - tmp[0][1]) \* *DECIMAL*)

/ *DECIMAL*;

/\*

\* System.out.println("\n------WYNIK---------"); System.out.println("| "

\* + wynik[0][0] + " |"); System.out.println("| " + wynik[0][1] + " |");

\*/

**return** wynik;

}

**private** **static** **double**[][] wyliczWynik(**double**[][] aktualnyXY) {

**double** wynik[][] = **new** **double**[1][2];

wynik[0][0] = Math

.*round*(((2 \* aktualnyXY[0][0] \* aktualnyXY[0][1]) - 3)

\* *DECIMAL*)

/ *DECIMAL*;

wynik[0][1] = Math

.*round*((((aktualnyXY[0][0] \* aktualnyXY[0][0]) - aktualnyXY[0][1]) - 2)

\* *DECIMAL*)

/ *DECIMAL*;

/\*

\* System.out.println("Wwynik pierwszego rownania = " + wynik[0][0]);

\* System.out.println("Wynik drugiego rownania = " + wynik[0][1]);

\* System.out.println("Dla x = " + aktualnyXY[0][0] + " , y = " +

\* aktualnyXY[0][1]);

\*/

**return** wynik;

}

**private** **static** **double**[][] sprawdzWynik(**double**[][] wynik,

**double**[][] bestResult) {

**if** (bestResult == **null**)

**return** wynik;

**if** (wynik[0][0] < bestResult[0][0] && wynik[0][1] < bestResult[0][1]) {

*bestXY*[0][0] = wynik[0][0];

*bestXY*[0][1] = wynik[0][1];

**return** wynik;

} **else**

**return** bestResult;

}

**private** **static** **boolean** szukajDalej(**double**[][] bestResult) {

String odpowiedz = **null**;

**if** ((bestResult[0][0] < *accuracy*) && (bestResult[0][1] < *accuracy*)) {

System.*out*.println("Uklad jest zbiezny.");

System.*out*.println("\n------WYNIK---------");

System.*out*

.println("x = " + *bestXY*[0][0] + " , y = " + *bestXY*[0][1]);

System.*out*.println("Czy chcesz kontynuowac szukanie?(T/N)");

odpowiedz = *in*.next();

**if** ("T".equalsIgnoreCase(odpowiedz))

**return** **false**;

**else**

**return** **true**;

} **else**

System.*out*.println("Uklad jest rozbiezny.");

**return** **true**;

}

**public** **static** **void** main(String[] args) {

**double** Fxy[][] = **new** **double**[1][2];

**double** fPxy[][] = **new** **double**[2][2];

**double** fodwrotna[][] = **new** **double**[2][2];

**double** aktualnyXY[][] = *startTab*;

**double** wynik[][] = **new** **double**[1][2];

**double** bestResult[][] = **new** **double**[1][2];

*wczytywanieDanych*();

**int** i = 0;

**boolean** czyKontynuowac = **false**;

**while** (!czyKontynuowac && i <= *maxIterationCount*) {

Fxy = *obliczenieFxy*(aktualnyXY);

fPxy = *obliczenieFprimXY*(aktualnyXY);

fodwrotna = *macierzOdwrotna*(fPxy);

aktualnyXY = *wyliczXY*(aktualnyXY, fodwrotna, Fxy);

wynik = *wyliczWynik*(aktualnyXY);

**if** (i == 0) {

bestResult[0][0] = wynik[0][0];

bestResult[0][1] = wynik[0][1];

*bestXY*[0][0] = aktualnyXY[0][0];

*bestXY*[0][1] = aktualnyXY[0][1];

}

bestResult = *sprawdzWynik*(wynik, bestResult);

**if** ((bestResult[0][0] < *accuracy*) && (bestResult[0][1] < *accuracy*)) {

czyKontynuowac = *szukajDalej*(bestResult);

*maxIterationCount* = *maxIterationCount* + 50;

}

**if** (i == *maxIterationCount*) {

czyKontynuowac = *szukajDalej*(bestResult);

*maxIterationCount* = *maxIterationCount* + 50;

}

i++;

}

}

}