Algorytmy Numeryczne - Faza 4

Bartłomiej Kruk, Mateusz Motyl, Szymon Lichota

26 maja 2021

1 Treść zadania

Równania $x^2-21=0$ oraz $e^x(x^2-21)=0$ mają te same pierwiastki. Sprawdzić czy dla różnych punktów startowych metoda stycznych Newtona potrzebuje tyle samo iteracji dla obu równań by osiągnąć dokładność 10^{-8}.

2 Teoretyczny opis metody

2.1 Teoria

Metoda stycznych Newtona
Problem znalezienia miejsca zerowego funkcji $f : [a, b] \to \mathbb{R}$ zamienimy na wyznaczenie miejsca zerowego stycznej do wykresu tej funkcji. Niech $x_0 \in [a, b]$ będzie ustalonym startowym przybliżeniem miejsca zerowego f. Za kolejne przybliżenie x_1 przyjmujemy miejsce zerowe stycznej w punkcie $(x_0, f(x_0))$ - w równaniu

$$y = f'(x_0)(x - x_0) + f(x_0)$$

wstawiamy $y = 0$ i otrzymujemy

$$0 = f'(x_0)(x - x_0) + f(x_0) \implies x := x_0 - \tfrac{f(x_0)}{f'(x_0)}.$$

Otrzymamy tak iteracje nazywane metodą Newtona (lub stycznych)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ k \geq 0.$$

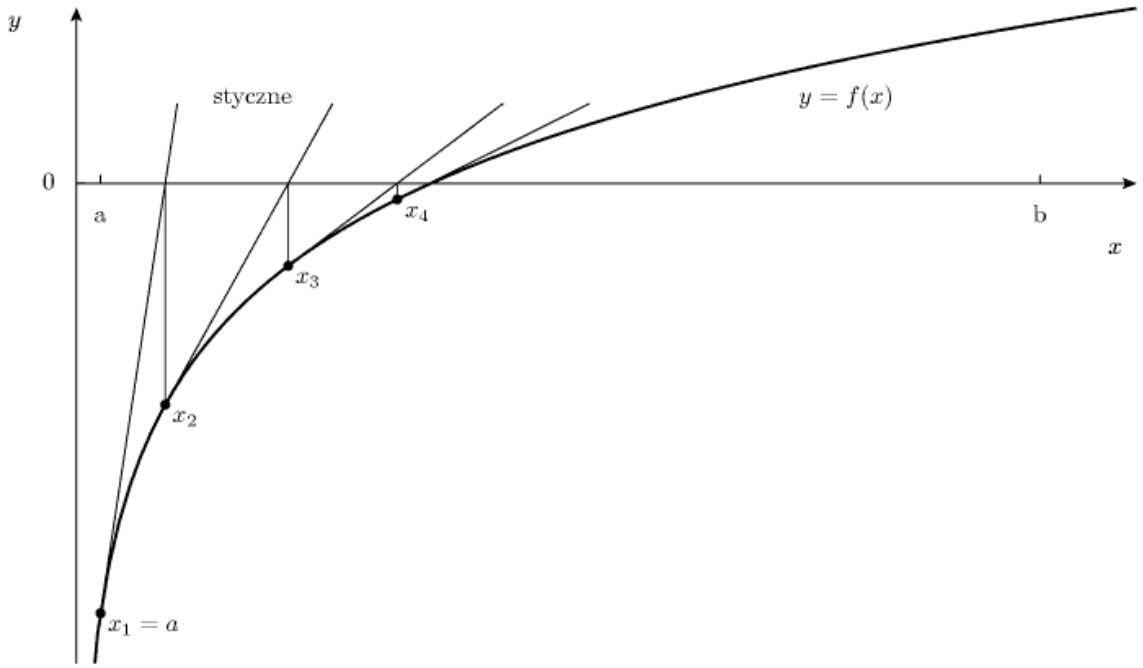
Założenia dla funkcji f :

- ma różne znaki na krańcach przedziału, tj. $f(a) \cdot f(b) < 0$,
- jest określona i ciągła w przedziale poszukiwań pierwiastka,
- w przedziale $[a, b]$ pierwsza pochodna $f'(x)$ jest różna od zera,
- pierwsza i druga pochodna funkcji mają stały znak w przedziale poszukiwań.

Warunek zakończenia obliczeń
W metodzie Newtona wykonuje się iteracyjnie obliczenia, aż do momentu gdy wartość funkcji w wyznaczonym punkcie jest bliska 0 (nie większa niż ϵ):

$$|f(x_k)| \leq \epsilon,$$

gdzie ϵ to dokładość obliczeń.



Rysunek 1: 4 pierwsze kroki działania metody Newtona

2.2 Przykład

Dla równania $x^3 + x + 1 = 0$ rozważanego na przedziale $[a, b] = [-1, 0]$ przybliżyć rozwiązanie metodą Newtona z $x_0 = a$ oraz $\epsilon = 0.025$.

Dla funkcji

$$f(x) = x^3 + x + 1$$

wyznaczamy jej pochodną

$$f'(x) = 3x^2 + 1.$$

Jako startowe przybliżenie miejsca zerowego ustalamy $x_0 = a = -1$.
Wyliczamy x_1 ze wzoru

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -1 - \frac{f(-1)}{f'(-1)} = -1 - (-\frac{1}{4}) = -\frac{3}{4}$$

Sprawdzamy warunek zakończenia iteracji

$$|f(x_1)| \leq \epsilon \;\Leftrightarrow\; \left|f(-\frac{3}{4})\right| \leq 0.025$$

Warunek nie został spełniony, więc iterujemy dalej. Wyliczamy x_2

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = -\frac{3}{4} - \frac{f(-\frac{3}{4})}{f'(-\frac{3}{4})} = -\frac{59}{86}$$

Sprawdzamy dokładność wyznaczonego x_2

$$\left|f(-\frac{59}{86})\right| \leq 0.025$$

Warunek zostaje spełniony, więc przerywamy dalsze poszukiwania, a szukanym przybliżeniem miejsca zerowego jest

$$x \approx x_2 = -\frac{59}{86}.$$

3 Opis implementacji

$x_0 \in [a, b]$ - ustalone startowe przybliżenie miejsca zerowego f

1. Pobieramy wartość pierwszego przybliżenia x_0. Obliczamy pochodną w punkcie x_0 i sprawczamy czy $f'(x_0) \neq 0$, jesli nie to podajemy inne przybliżenie. Sprawdzanie, czy pochodna jest różna od zera jest przygotowaniem do następnego kroku przy założeniu, że użytkownik od razu nie poda rozwiązania z założoną dokładnością.
2. Podajemy wartość ϵ, taką że $\epsilon > 0$. Przechodzimy do punktu 4.
3. Obliczamy pochodną w punkcie x_0 i sprawdzamy czy $f'(x_0) \neq 0$, jeśli nie to kończymy obliczenia.
4. Wyliczamy $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$.
5. Podstawiamy jako nowe $x_0 = x_1$ i zwiększamy licznik. Jeżeli licznik osiągnie wartość 32, będzie to znakiem dla naszego programu, że nie udało się znaleźć pierwiastka w 32 krokach.
6. Jeżeli $|f(x_0)| \leq \epsilon$ kończymy obliczenia, n jest liczbą kroków potrzebnych do znalezienia miejsca zerowego, a x_0 samym miejscem zerowym. W przeciwnym wypadku wracamy do kroku 3.

Algorithm 1: Implementacja metody Newtona

```
read x_0
while f'(x_0) = 0 do
    read x_0
read ε
while |f(x_0)| > ε do
    if f'(x_0) = 0 then
        break
    x_1 = x_0 − f(x_0)/f'(x_0)
    x_0 = x_1
    n = n + 1
    if n = 32 then
        break;
return n, x_0
```

4 Demonstracja

4.1 Opis programu

Specyfikacja wejścia:
W pierwszym wierszu standardowego wejścia podajemy liczbę od 1 do 4, w zależności od wyboru przykładu lub x w przypadku, gdy chcemy zakończyć wykonywanie programu. Jeśli chcemy wprowadzić inny przykład niż te, które są w standardowym programie, należy zmodyfikować kod źródłowy dodając do niego oczekiwaną funkcję oraz jej pochodną. W następnym wierszu podajemy wartość pierwszego przybliżenia x_0, takiego, że $f'(x_0) \neq 0$. Następnie wstawiamy wartość $\epsilon > 0$.
Specyfikacja wyjścia:
Program symuluje metodę stycznych Newtona; wyznacza pierwiastek funkcji za pomocą stycznych do wykresu funkcji $f : [a, b] \to \mathbb{R}$ i zwraca liczbę kroków potrzebnych do znalezienia miejsca zerowego z dokładnością do ϵ oraz samo miejsce zerowe.

4.2 Przykłady

$$f(x) = x^3 + x + 1,\ x_0 = -1,\ \epsilon = 0.25$$

Wejście:
3
-1
$0,025$

Wyjście:
$2 : -0.686046511627907$

Inne przykłady:

- $f(x) = x^2 - 21,\ x_0 = -5,\ \epsilon = 1E - 8$

  Wyjście:
  $3 : -4.582575695074664$

- $f(x) = e^x(x^2 - 21),\ x_0 = -5,\ \epsilon = 1E - 8$

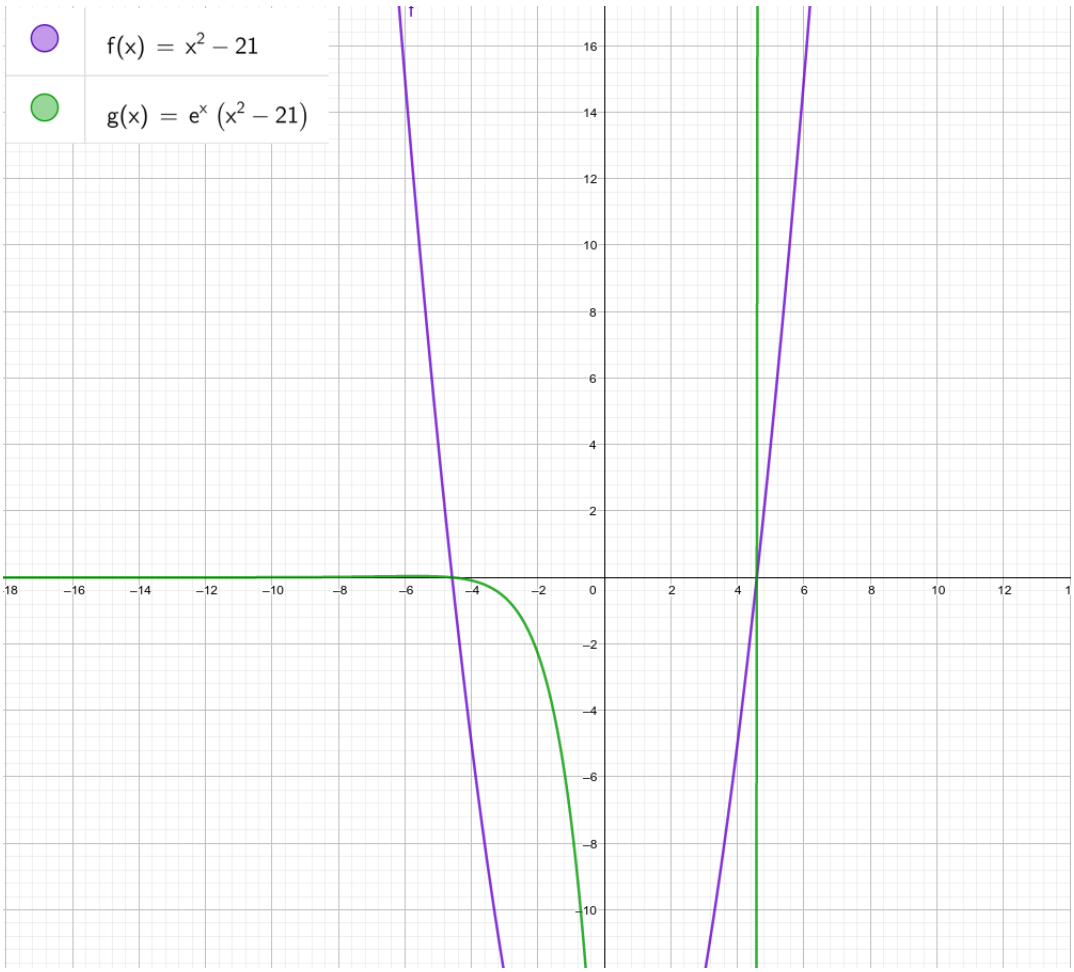
  Wyjście:
  $5 : -4.582575694949294$

- $f(x) = x^2 - 2,\ x_0 = -5,\ \epsilon = 1E - 8$

  Wyjście:
  $6 : -1.4142135623730951$

5 Podsumowanie programu

Dla różnych punktów startowych funkcje $f(x) = x^2 - 21$ i $g(x) = e^x(x^2 - 21)$ potrzebują różnych ilości iteracji, aby osiągnąć dokładność 10^{-8}.
Dla podanego przykładu weźmy 4 punkty startowe: 4, 15, 100, 1000. Aby osiągnąć żądaną dokładność funkcje potrzebują kolejno:
- 4, 6, 8, 12 iteracji dla funkcji $f(x)$;
- 8, 19, 32, $n > 32$ iteracji dla funkcji $g(x)$.
Z podanego przykładu można wywnioskować, że liczba iteracji rośnie wraz z wartością bezwzględną odległości podanego punktu x_0 od najbliższego rzeczywistego rozwiązania funkcji. Można też zauważyć, że ilość potrzebnych iteracji jest zależna od kąta nachylenia stycznej do funkcji w punkcie x względem OX.



Rysunek 2: Wykresy funkcji $f(x) = x^2 - 21$ oraz $g(x) = e^x(x^2 - 21)$.

6 Kod źródłowy

```
import java.util.Scanner;
import java.util.function.DoubleFunction;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Newton();
        input.close();
    }

    static void Newton(){
        DoubleFunction<Double> fun;
        DoubleFunction<Double> funDeriv;

        while(true){
            System.out.println("Wybierz przykład:");
            System.out.println("1) f(x) = x^2 - 21");
            System.out.println("2) f(x) = e^x(x^2 - 21)");
            System.out.println("3) f(x) = x^3 + x + 1");
            System.out.println("4) f(x) = x^2 + 2");
            System.out.println("x) end");

            Scanner input = new Scanner(System.in);
            String s = input.next();

            switch (s) {
                case "1":
                    fun = (x) -> x * x - 21;
                    funDeriv = (x) -> 2 * x;
                    break;
                case "2":
                    fun = (x) -> Math.exp(x) * (x * x - 21);
                    funDeriv = (x) -> Math.exp(x) * (x * x + 2*x - 21);
                    break;
                case "3":
                    fun = (x) -> x * x * x + x + 1;
                    funDeriv = (x) -> 3 * x * x + 1;
                    break;
                case "4":
                    fun = (x) -> x * x + 2;
                    funDeriv = (x) -> 2 * x;
                    break;
                case "x":
                    System.exit(0);
                default:
                    System.out.println("Podaj jeszcze raz: \n");
                    continue;
            }

            System.out.println("Podaj wartości pierwszego przybliżenia x0: ");
            double x0 = input.nextDouble();

            while(funDeriv.apply(x0) == 0){
                System.out.println("Podaj prawidlowe przyblizenie:");
                x0 = input.nextDouble();
            }
```

```
            System.out.println("Dokładność (epsilon > 0):");
            double epsilon = input.nextDouble();
            while(epsilon <= 0){
                System.out.println("Podaj prawidlową dokładność:");
                epsilon = input.nextDouble();
            }


            int n = 0;
            double x1;

            while(Math.abs(fun.apply(x0)) > epsilon){
                if(funDeriv.apply(x0) == 0){
                    System.out.println("Wartosc pochodnej równa 0.");
                    break;
                }

                x1 = x0 - fun.apply(x0) / funDeriv.apply(x0);
                x0 = x1;
                n++;

                if(n == 32){
                    System.out.println("Nie udalo sie znalezc rozwiazania w 32 krokach.");
                    System.exit(0);
                }
            }

            System.out.println(n + ": " + x0 + "\n");
        }
    }
}
```

Literatura

[1] Jaruszewska-Walczak Danuta. wykład nr 5: aproksymacja wielomianem Taylora, metody: Newtona (stycznych), Newtona z ustaloną pochodną, siecznych i ich interpretacje; Lista zadań nr 2 - równania nieliniowe i ich układy

[2] Metoda Newtona

https://en.wikipedia.org/wiki/Newton%27s_method

[3] Algorytm

https://mrostkow.oeiizk.waw.pl/mojapraca/nmn.htm