Projekt z Algorytmów Numerycznych

Cyprian Lazarowski, Kacper Karwot, Kacper Muszczyński

5 maja 2021

1 Treść Zadania

Pobrać od użytkownika żądaną dokładność $0 < \varepsilon < 1$ oraz przedział, w którym szukamy pierwiastka równania:

$$\ln(x^2) - \sin(x) - 2 = 0$$

Porównać liczbę kroków potrzebnych metodzie siecznych Newtona by osiągnąć dokładność ε z liczbą kroków dla metody bisekcji dla kilku wybranych przedziałów zawierających dokładnie jeden pierwiastek. Znaleźć wszystkie pierwiastki równania z dokładnością 10^{-8}.

2 Teoretyczny opis metody

Zakładając, że posiadamy pewną funkcje $f(x)$, przedział $[a,b]$ oraz dokładność $0 < \varepsilon < 1$ wyznaczamy liczbę kroków metody połowienia oraz siecznych.

2.1 Metoda połowienia (bisekcji)

Dla przedziału $[a_0, b_0]$:=$[a, b]$, takiego że:

- Funkcja $f(x)$ jest ciągła na tym przedziale
- $f(a_0) \cdot f(b_0) < 0$

Wyznacza się środek określonego przedziału wzorem:

$$c_0 = \frac{a_0 + b_0}{2}$$

Jeżeli c_0:

- $f(c_0) = 0$, to c_0 jest rozwiązaniem równania
- $|x - c_0| \leq \frac{b_0 - a_0}{2} < \varepsilon$, to c_0 jest rozwiązaniem równania z dokładnością ε

W innym wypadku, wyznacza się kolejny podzbiór:

$$[a_1, b_1] := \begin{cases} [a_0, c_0] & \text{gdy } f(a_0) \cdot f(c_0) < 0 \\ [c_0, b_0] & \text{gdy } f(c_0) \cdot f(b_0) < 0 \end{cases}$$

Jeśli c_1 dla nowego podzbioru, nie jest rozwiązaniem, to powtarzamy proces przez wybranie przedziału $[a_2, b_2] \subset [a_1, b_1]$ jako ten z przedziałów $[a_1, c_1]$ lub $[c_1, b_1]$, w którym leży rozwiązanie równania. Punkt c_2 będzie kolejnym przybliżeniem rozwiązania.

Tym sposobem tworzymy, ciąg przedziałów $[a_k, b_k]$ oraz ciąg środków c_k. Gdzie k, jest liczbą kroków, które musi wykonać ta metoda, aby odszukać rozwiązanie równania, bądź jego przybliżenie z dokładnością ε.

2.2 Metoda Newtona siecznych

Dla $x_0 := a$ i $x_1 := b$ mamy

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

gdzie x_{k+1} jest miejsce zerowym prostej o równaniu

$$y = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k),$$

czyli siecznej wykresu $f(x)$ w punktach $(x_k, f(x_k)), (x_{k-1}, f(x_{k-1}))$. Powtarzamy działanie, wyznaczając kolejne x_{k+1} dopóki nie zachodzi

$$|x_{k+1} - x_k| \leq \epsilon,$$

a wtedy mówimy, że x_{k+1} jest rozwiązaniem z dokładnością ε.

2.3 Przykład Policzony Ręcznie

Do czytelnego przedstawienia wyników, jak i iteracji metody uproszczone zostaną dane startowe: $f(x)$=$\ln(x^2) - \sin(x) - 2 = 0$ i ε=0.1

Jeden z poprawnych podzbiorów, który posiada miejsce zerowe, przybiera wartości [-2,-1]

Metoda połowienia (bisekcji)

1. Krok
   $a_0 = -2$ i $f(a_0) \approx 0.2956$
   $b_0 = -1$ i $f(b_0) \approx -1.1585$
   $c_0 = -1.5$ i $f(c_0) \approx -0.1916$
   $$|\frac{b_0 - a_0}{2}| = |\frac{-1+2}{2}| = 0.5 > \varepsilon$$
   W takim wypadku $f(a_0) \cdot f(c_0) < 0$ więc kolejnym zbiorem jest [-2, -1.5]

2. Krok
   $a_1 = -2$ i $f(a_1) \approx 0.2956$
   $b_1 = -1.5$ i $f(b_1) \approx -0.1916$
   $c_1 = -1.75$ i $f(c_1) \approx 0.1032$
   $$|\frac{b_1 - a_1}{2}| = |\frac{-1.5+2}{2}| = 0.25 > \varepsilon$$
   W takim wypadku $f(c_1) \cdot f(b_1) < 0$ więc kolejnym zbiorem jest [-1.75, -1.5]

3. Krok
   $a_2 = -1.75$ i $f(a_2) \approx 0.1032$
   $b_2 = -1.5$ i $f(b_2) \approx -0.1916$
   $c_2 = -1.625$ i $f(c_2) \approx -0.0305$
   $$|\frac{b_2 - a_2}{2}| = |\frac{-1.5+1.75}{2}| = 0.125 > \varepsilon$$
   W takim wypadku $f(a_2) \cdot f(c_2) < 0$ więc kolejnym zbiorem jest [-1.75, -1.625]

4. Krok
   $a_3 = -1.75$ i $f(a_3) \approx 0.1032$
   $b_3 = -1.625$ i $f(b_3) \approx -0.0305$
   $c_3 = -1.6875$ i $f(c_3) \approx 0.0397$
   $$|\frac{b_3 - a_3}{2}| = |\frac{-1.625+1.75}{2}| = 0.0625 < \varepsilon$$
   Miejsce zerowe znajduje się z dokładnością 0.0625 od c_3=-1.6875

Metoda siecznych

1. Krok $x_0 = -2$ i $x_1 = -1$

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = -1 - \frac{f(-1) \cdot 1}{f(-1) - f(-2)} \approx -1.7967$$

$|x_2 - x_1| = |-1.7967 + 1| = 0.7967 > \epsilon$

2. Krok $x_1 = -1$ i $x_2 = -1.7967$

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} = -1.7967 - \frac{f(-1.7967) \cdot -0.7967}{f(-1.7967) - f(-1)} \approx -1.7073$$

$|x_3 - x_2| = |-1.7073 + 1.7967| = 0.0894 < \epsilon$

Miejsce zerowe znajduje się z dokładnością 0.0894 od x_3=-1.7073

Dla podanego zbioru [-2,-1] metoda siecznych wykonała 2 kroki, a metoda połownienia już 4.

3 Opis implementacji

Zgodnie z treścią zadania i późniejszą korektą słowną wprowadzoną na pierwszym spotkaniu zaimplementowano trzy funkcje odpowiadające trzem sposobom aproksymacji miejsca zerowego funkcji:

1. metodą siecznych
2. metodą połowienia(bisekcji)
3. metodą siecznych zmodyfikowaną w ten sposób, by argumenty były zawsze różnych znaków

Poza obliczeniami na zadanym przez użytkownika przedziale, jest przeprowadzana prosta statystyka. Badanie przebiega na zbiorze wszystkich par liczb (x, y) z przedziału w pobliżu miejsc zerowych badanej funkcji takich, że:

$$y > x \ \wedge \ x, y \in \left\{ \frac{k}{10}, \ k \in \mathbb{Z} \right\}$$

Każda metoda zawiera licznik *time-to-kill*[1], ustawiony na początku na 64 i zmniejszający się o jeden co krok, więc jeżeli pierwiastek nie zostanie odnaleziony w 64 krokach, metoda kończy działanie. Stan licznika jest zwracany jako reprezentacja ilości kroków potrzebnej metodzie do znalezienia miejsca zerowego, 0 jest interpretowane jako porażka.

Wyniki przechowywane są w tablicy obiektów klasy Wynik, będącej efektywnie krotką o nazwanych polach.

Funkcja której miejsca zerowe są przybliżane jest przekazywana jako argument, może więc być zmieniona.

3.1 Metoda siecznych

Metoda siecznych przyjmuje cztery argumenty, kolejno: funkcję, dwa wstępne przybliżenia pierwiastka (zamiennie nazywane krańcami przedziału, przez konieczność porównania wyników z metodą bisekcji) oraz żądaną dokładność.

Algorithm 1: Metoda siecznych

Data: f, x, y, eps
Result: przybliżone miejsce zerowe f, w okolicy x, y, z dokładnością eps
ttk ← 64;
while *ttk > 0 and |x-y| > eps* **do**
  x ← y;
  y ← x - (f(x) · (x - y)) / (f(x) - f(y));
  ttk ← ttk - 1

[1]Ponieważ metoda siecznych ma szansę być rozbieżna i wykonywać się potencjalnie w nieskończoność, konieczne było wprowadzenie sposobu na terminowanie jej w takim wypadku, dlatego zaimplementowano licznik *time-to-kill* zamiast wprost liczyć wykonane kroki

3.2 Metoda bisekcji

Metoda bisekcji jest skonstruowana tak samo jak poprzednia. Poza metodą wyznaczania kolejnego kroku różni się tym, że przed rozpoczęciem obliczeń następuje kontrola znaku funkcji dla argumentów.

Algorithm 2: Metoda bisekcji

```
Data: f, x, y, eps
Result: przybliżone miejsce zerowe f, w okolicy x, y, z dokładnością eps
if f(x) · f(y) > 0 then
    return;
x, y ← ( f(x) < 0 ) ? x, y : y, x;
ttk ← 64;
while ttk > 0 and |x-y| > eps do
    środek ← (x+y)/2;
    if f(środek) < 0 then
        x ← środek;
    else
        y ← środek;
    ttk ← ttk - 1
```

3.3 Metoda siecznych zmodyfikowana

Metoda siecznych zmodyfikowana o dodatkową kontrolę znaku funkcji dla argumentów na podobieństwo metody bisekcji.

Algorithm 3: Metoda siecznych zmodyfikowana

```
Data: f, x, y, eps
Result: przybliżone miejsce zerowe f, w okolicy x, y, z dokładnością eps
if f(x) · f(y) > 0 then
    return;
x, y ← ( f(x) < 0 ) ? x, y : y, x;
ttk ← 64;
while ttk > 0 and |x-y| > eps do
    środek ← x - (f(x) · (x - y)) / (f(x) - f(y));
    if f(środek) < 0 then
        x ← środek;
    else
        y ← środek;
    ttk ← ttk - 1
```

3.4 Pozostałe funkcje i klasy pomocnicze

Następujące funkcje i klasy zostały wprowadzone dla polepszenia czytelności kodu i wymagają szczegółowego omówienia.

Klasa wyniki przechowuje string określający typ metody i ewentualną informację o tym z jakiego powodu nie udało się znaleźć miejsca pierwiastka, krańce przedziału oraz wartości funkcji w ostatnim kroku, stan licznika *time-to-kill* oraz punkty od których rozpoczęto przybliżanie.

Dwie pomocnicze funkcje który_mniejszy(x,y) oraz różne_znaki(x,y) użyte są we wstępie metody połowienia, funkcja(x) przechowuje przybliżaną funkcję, a sieczne_krok(f,x,y) wzór na kolejne przybliżenie w metodzie siecznych.

4 Instrukcja obsługi programu

4.1 Uruchomienie

Do poprawnego działania programu niezbędne są:

1. Python: https://www.python.org/downloads/
2. Biblioteka tk_html_widgets: https://pypi.org/project/tk-html-widgets/
3. Plik program.py uruchamiający program
4. Plik dane.txt w tym samym folderze z programem. Ten powinien już się wygenerować podczas pierwszego uruchomienia opcji Edytuj plik. Jednak w przypadku nieznanego błędu trzeba go dodać ręcznie

4.2 Działanie

Użytkownikowi pokazuje się proste okienko z czterema guzikami:

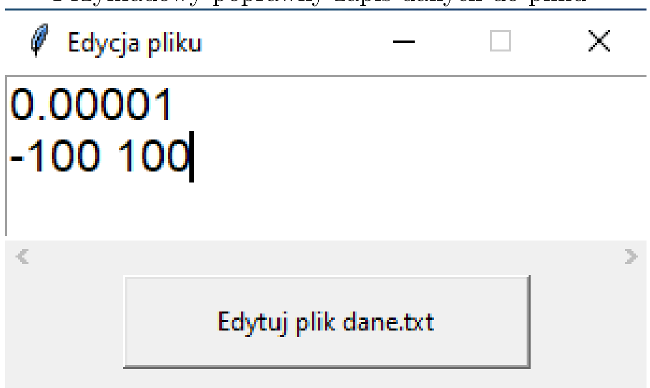
- Edytuj plik - uruchamia on nowe okno z polem tekstowym, które umożliwia edytowanie pliku tekstowego dane.txt. Postęp zapisujemy, klikając guzik Edytuj plik dane.txt. Przy zapisaniu powinna, pojawić się wiadomość informująca o poprawnym zapisaniu wartości z pola tekstowego.
- Oblicz dla podanego - guzik ten otwiera nowe okno, które liczy i wyświetla wyniki dla trzech metod, które wykorzystują dane podane w dane.txt.
- Wygeneruj dla podanego - guzik ten odpowiada za wygenerowanie dla określonego w pliku przykładu zestawów zbiorów, na których później porównywane są trzy operacje.
- Dodatkowe informacje - informuje o autorach projektu, oraz daje linka do dokumentacji.

4.3 Wprowadzanie danych do pliku dane.txt

Wprowadzanie danych do pliku dane.txt polega na wpisywaniu liczb w określone linijki. Przejście między danymi wykonujemy używając spacji. Liczby zmiennoprzecinkowe piszemy z kropką np 3.14. W określoną linijkę wprowadzamy:

- Pierwsza: wartości ε, oznaczającą dokładność
- Druga: wartości $x1$ i wartość $x2$, tworzące przedział

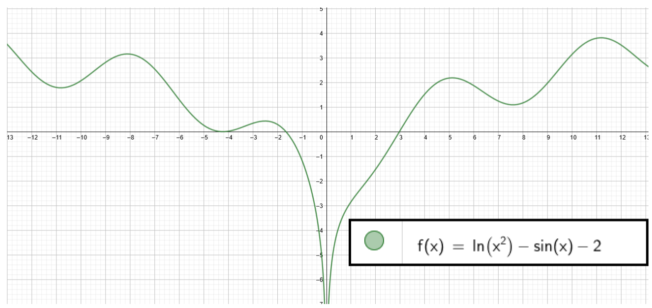
Przykładowy poprawny zapis danych do pliku[2]



Liczba danych w drugiej linijce powinna być różna od siebie, jak i ich odległość na osi x powinna być większa od ε w innym przypadku program poinformuje o błędzie.

[2] Dane są wybrane jedynie w celu zaprezentowania ich użytkownikowi. Sam dobór liczb może być błędny w niektórych przykładach.

5 Raport z demonstracji



Badanie w miejscach wybranych na podstawie wykresu w pobliżu miejsc w których można zaobserwować przecięcia wykresu z osią OX.

Przedział: (-4.2, -4.1) Dokładność: 10^{-8}

	Pierwiastek w pobliżu x=	z dokładnością	ilość kroków
Bisekcja	-4.154700380563735	5.9^{-9}	24
Sieczne	-4.154700380563578	1^{-10}	7
Sieczne++	-4.154700380563583	6.2^{-15}	25

Przedział: (-2, -1) Dokładność: 10^{-8}

	Pierwiastek w pobliżu x=	z dokładnością	ilość kroków
Bisekcja	-1.651399902999401	7.4^{-9}	27
Sieczne	-1.651399902900191	3.9^{-10}	7
Sieczne++	-1.651399902900190	4.4^{-16}	35

Przedział: (2, 3) Dokładność: 10^{-8}

	Pierwiastek w pobliżu x=	z dokładnością	ilość kroków
Bisekcja	2.9661586657166480	7.4^{-9}	27
Sieczne	2.9661586695118833	2.8^{-11}	4
Sieczne++	2.9661586695118830	4.4^{-16}	6

Przedział: (1, 10) Dokładność: 10^{-8}

	Pierwiastek w pobliżu x=	z dokładnością	ilość kroków
Bisekcja	2.9661586759611964	$8,3^{-9}$	30
Sieczne	-4.281256222626640	7.9^{-11}	15
Sieczne++	2.9661586695118830	4.4^{-16}	9

Łatwo widać, że w ogólności najmniejszą ilość kroków wykonuje metoda siecznych. Natomiast na co warto zwrócić uwagę, to fakt, że zmodyfikowana metoda siecznych daje wyniki z dwukrotnie wyższą niż zadana dokładnością, co w zasadzie nie powinno mieć miejsca. Dzieje się tak przez połączenie sposobu w który zaimplementowano obliczanie dokładności oraz wprowadzenia warunku, że przedział musi zawierać pierwiastek, na podobieństwo metody bisekcji. Dokładność jest liczona jako odległość między krańcami przedziału dla metody bisekcji i siecznych zmodyfikowanej, natomiast jako różnica między dwoma najnowszymi przybliżeniami dla metody siecznych. W przypadku metody siecznych zmodyfikowanej, jeden z krańców dość szybko "przysuwa się" do pierwiastka, a mimo to odległość między krańcami pozostaje znacząca (o ile jeden z krańców nie jest bardzo blisko pierwiastka), i taki stan utrzymuje się do momentu kiedy "ruchomy" kraniec przedziału znajdzie się dokładnie na pierwiastku; dopiero wtedy zostaje "przesunięty" drugi kraniec. W czasie debugowania (obrazy niezamieszczone) zaobserwowano gwałtowny skok odległości między krańcami w ostatnim kroku metody.

Jeden z przedziałów wybrano tak, żeby zaprezentować, że metoda siecznych może "znaleźć" inny pierwiastek niż pozostałe dwie, przy tych samych początkowych argumentach, przez różnice w sposobie działania.

Podsumowanie badania statystycznego wspomnianego w sekcji trzeciej:

1. algorytmy wywołano na 9801 parach liczb
2. metoda siecznych była zbieżna w 78% przypadków
3. 35% z nich spełniało warunek konieczny do wykonania metody bisekcji lub zmodyfikowanej siecznych
4. średnia ilość kroków potrzebna metodom do znalezienia pierwiastka z zadaną dokładnością:
   (a) 9.34 dla siecznych (21 uwzględniając przedziały w których jest rozbieżna [3])
   (b) 29.16 dla bisekcji
   (c) 31.54 dla siecznych zmodyfikowanej

Na podstawie przebiegu ćwiczenia i powyższej statystyki sformułowano następujące wnioski: Niepoprawnym było założenie jakoby metoda siecznych miała być wykonywana wyłącznie na parach liczb dla których badana funkcja przyjmuje przeciwne znaki, wprowadzenie takiej kontroli zmienia sposób jej działania w nieoczekiwany sposób. Dalej, zgodnie z intuicją metoda siecznych średnio szybciej (przez "szybciej" rozumiane jest "w mniejszej ilości kroków") wyznacza pierwiastek z żądaną dokładnością (niż metoda połowienia), nie wymagając przy tym sprawdzania znaku funkcji na każdym kroku.

[3] Liczba ta jest w pewien sposób niedokładna ponieważ, w przeciwieństwie do pozostałych, zależy od początkowej wartości licznika *time-to-kill*.

6 Kod programu

Wyjątek z kodu źródłowego programu zawierający wszystkie metody, lecz nie wystarczający do odtworzenia pliku z kodem źródłowym. Dla zmniejszenia bloku tekstu w dokumentacji wycięto polecenia odpowiadające za okienka i generowanie danych na kilku podzbiorach, które znajdują się w oryginalnym pliku.

```
import math

class Wynik:
    def __init__(self, typ, x=None, dok=None, ttk=None):
        self.typ = typ
        self.x = x
        self.dok = dok
        self.ttk = ttk

def funkcja(x):
    return math.log(x**2) - math.sin(x) - 2

def który_mniejszy(x, y):
    if x < y:
        return x, y
    else:
        return y, x

def różne_znaki(x, y):
    if x * y < 0:
        return True
    else:
        return False

def sieczne_krok(f, x, y):
    return x - (f(x) * (x - y)) / (f(x) - f(y))

#Metoda Siecznych
def sieczne(f, x, y, eps):
    ttk = 64
    while ttk > 0 and abs(x-y) > eps:
        temp = sieczne_krok(f, x, y)
        x = y
        y = temp
        ttk = ttk - 1
        dok=abs(x-y)
    if ttk==0: return Wynik("sieczne 64")
    else: return Wynik("sieczne", temp, dok, ttk)
```

```python
#Metoda Bisekcji
def bisekcja(f, x, y, eps):
    if not różne_znaki(f(x), f(y)):
        return Wynik("bisekcja - znak")
    ttk = 64
    while ttk > 0 and abs(x - y) > eps:
        srodek = (x + y) / 2
        if różne_znaki(f(y), f(srodek)):
            x = srodek
        else:
            y = srodek
        ttk = ttk - 1
    dok=abs(x - y)
    if ttk==0: return Wynik("bisekcja 64")
    else: return Wynik("bisekcja", srodek, dok, ttk)

#Metoda Siecznych+ - sieczne + reguły bisekcji
def sieczne_plus(f, x, y, eps):
    if not różne_znaki(f(x), f(y)):
        return Wynik("sieczne+ - znak")
    ttk = 64
    while ttk > 0 and abs(x-y) > eps:
        kolejny = sieczne_krok(f, x, y)
        if różne_znaki(f(y), f(kolejny)):
            x = kolejny
        else:
            y = kolejny
        ttk = ttk - 1
        dok=abs(x-y)
    if ttk==0: return Wynik("sieczne+ 64")
    else: return Wynik("sieczne+", kolejny, dok, ttk)

def wypisz_wynik(f, x, y, eps):
    BW=bisekcja(f,x,y,eps)
    print("Bisekcja Wyniki:")
    if BW.typ=="bisekcja - znak": print("W tym przedziale nie ma miejsca zerowego, metoda nie zadziała!\n")
    if BW.typ=="bisekcja 64": print("Metoda dla podanego przykładnu nie zmieściła się w 64 krokach\n")
    if BW.typ=="bisekcja": print("Miejsce zerowe występuje w pobliżu x={} z dokladnością: {}, metoda wykonała {} kroków\n".format(BW.x, BW.dok, 64-BW.ttk))

    SW=sieczne(f,x,y,eps)
    print("Sieczne Wyniki: ")
```

```python
    if SW.typ=="sieczne 64": print("Metoda dla podanego przykładnu nie zmieściła się w 64 krokach\n")
    if SW.typ=="sieczne": print("Miejsce zerowe występuje w pobliżu x={} z dokladnością: {}, metoda wykonała {} kroków\n".format(SW.x, SW.dok, 64-SW.ttk))

    SPW=sieczne_plus(f,x,y,eps)
    print("Sieczne+ Wyniki: ")
    if SPW.typ=="sieczne+ - znak": print("W tym przedziale nie ma miejsca zerowego, metoda nie zadziała!\n")
    if SPW.typ=="sieczne+ 64": print("Metoda dla podanego przykładnu nie zmieściła się w 64 krokach\n")
    if SPW.typ=="sieczne+": print("Miejsce zerowe występuje w pobliżu x={} z dokladnością: {}, metoda wykonała {} kroków\n".format(SPW.x, SPW.dok, 64-SPW.ttk))

wypisz_wynik(funkcja, -8, -1, 0.00001)
```