# Obliczenia Naukowe - Lista 3

Paweł Dychus (244941)

Listopad 2019

## Zadanie 1

### Opis problemu

Implementacja algorytmu rozwiązującego równanie $f(x) = 0$ metodą bisekcji. Metoda korzysta z twierdzenia Darboux, mówiącej o tym, że ciągła funkcja, która zmienia znak na przedziale [a,b] (tj. $f(a)*f(b) < 0$), musi mieć miejsce zerowe w tym przedziale (tj.$f(c) = 0, c \in (a, b)$). Po sprawdzeniu warunku początkowego tj. $f(a) * f(b) < 0$), metoda wyznacza iteracyjnie w każdym kroku środek przedziału $c = (a + b)/2$. Następnie podstawia za stary punkt 'a' wartość 'c', jeżeli mają ten sam znak (tj. $f(a) * f(c) > 0$, alternatywnie $f(b) * f(c) < 0$). W przeciwnym wypadku podstawia pod 'b'. W ten sposób algorytm dzieli w każdej iteracji przedział na kolejne, coraz to mniejsze połówki zawierające szukany punkt. Algorytm działa aż do osiągnięcia określonej precyzji, tudzież zera, co w praktyce jest rzadkością, ze względu na skończoną precyzję arytmetyki.

### Rozwiązanie

Implementacja algorytmu (pseudokod).

**Dane:**

f - zadana funkcja, f(x),

a,b - końce zadanego przedziału,

delta, epsilon - określona precyzja.

**Wyniki:**

(r,v,it,err) - gdzie:

r - przybliżenie pierwiastka równania,

v - wartość $f(r)$,

it - liczba iteracji,

err - flaga błędu
0 - brak błędu
1 - funkcja nie zmienia znaku

**Algorithm 1** Algoritm bisekcji

```
function MBISKECJI(f,a,b,delta,epsilon)
    u ← f(a)
    v ← f(b)
    e ← b − a
    it ← 0
    if sign(u)=sign(v) then
      return (err,err,err,1)
    while |e| > delta and |w| > epsilon do
        it ← it+1
        e ← e/2
        c ← a+e
        w ← f(c)
        if sign(w) = sign(f(a)) then
            a ← c
        else
            b ← c
return (c, w, it, 0)
```

Powyższa implementacja zawiera kilka usprawnień:

1. $c \leftarrow a + (b - a)/2$ - pozwala ominąć potencjalne błędy, związane z: $c \leftarrow (a + b)/2$
2. $sign(u) \neq sign(v)$ - pozwala ominąć potencjalne błędy, związane z: $f(a) * f(b) < 0$

Algorytm kończy się po uzyskaniu określonej precyzji, czy to delty (odległości na osi X), czy to epsilona (precyzja wartości).

# Zadanie 2

## Opis problemu

Implementacja algorytmu rozwiązującego równanie $f(x) = 0$ metodą Newtona. Metoda korzysta z twierdzenia Taylora. Czyli zastępuje funkcję wejściową dwoma pierwszymi składnikami ze wzoru Taylora, efektywnie linearyzując zadaną funkcje tj.($l(x) = f(c) + f'(c)(x - c)$). Metoda Newtona jest szybsza, ze względu na kwadratową zbieżność, ale ma kilka wad. Po pierwsze, metoda nie zawsze jest zbieżna, a po drugie koniecznie jest liczenie pochodnej.

## Rozwiązanie

Implementacja algorytmu (pseudokod).

**Dane:**

f,pf - zadana funkcja $f(x)$, i jej pochodna $f'(x)$

x0 - przybliżenie początkowe,

delta, epsilon - określona precyzja,

maxit - maksymalna ilość iteracji.

**Wyniki:**

(r,v,it,err) - gdzie:

r - przybliżenie pierwiastka równania,

v - wartość $f(r)$,

it - liczba iteracji,

err - flaga błędu

0 - metoda zbieżna

1 - nie osiągnieto wymaganej dokładności

2 - pochodna bliska zeru

**Algorithm 2** Algoritm Newtona

```
function MSTYCZNYCH(f,pf,x_0,delta,epsilon,maxit)
    v ← f(x0)
    if |v| < epsilon then
        return (x0, v, 0, 0)
    if |pf(v)| < epsilon then
        return (x0, v, 0, 2)                          ▷ Pochodna bliska zeru
    for k ← 1 to maxit do
        x1 ← x0 − (v/pf(x0))
        v ← f(x1)
        if |x1 − x0| ≤ delta or |v| ≤ epsilon then
            return (x0, v, it, 0)
        x0 ← x1
return (x0, v, 0, 1)                                  ▷ Nie osiągnięto określonej precyzji
```

Algorytm najpierw sprawdza warunki początkowe, czyli czy pochodna nie jest bliska zero i czy wybrany punkt nie jest już dostatecznie blisko. Następnie algorytm przechodzi do iteracji, która wykona się maksymalnie *maxit* razy. Podczas iteracji wyliczane są kolejne przybliżenia funkcji, zgodnie ze wzorem $x_{n+1} = x - \frac{f(x_0)}{f'(x_0)}$. Koniec następuje do osiągnięciu maksymalnej ilości iteracji(error $=$ 1), tudzież po osiągnięciu żądanej precyzji.

# Zadanie 3

## Opis problemu

Implementacja algorytmu rozwiązującego równanie $f(x) = 0$ metodą siecznych. Metoda jest bezpośrednim rozwinięciem metody Newtona. Polega na zastąpieniu pochodnej $f'(x)$, ilorazem różnicowym, tj. $f'(x) \approx \frac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}$. Metoda siecznych ma zbieżność $\alpha \approx (1+\sqrt{5})/2 \approx 1.618$. Czyli mniej niż w przypadku metody Newtona, ale w zamian, nie ma potrzeby liczenia dwóch osobnych wartości dla $f(x)$ i $f'(x)$, wystarczy $f(x)$.

## Rozwiązanie

Implementacja algorytmu (pseudokod).

**Dane:**

f - zadana funkcja $f(x)$,

x0.x1 - przybliżenia początkowe,

delta, epsilon - określona precyzja,

maxit - maksymalna ilość iteracji.

**Wyniki:**

(r,v,it,err) - gdzie:

r - przybliżenie pierwiastka równania,

v - wartość $f(r)$,

it - liczba iteracji,

err - flaga błędu

0 - metoda zbieżna

1 - nie osiągnieto wymaganej dokładności

**Algorithm 3** Algorytm siecznych

```
function MSIECZNYCH(f, x0, x1, delta, epsilon, maxit)
    f_x0 ← f(x0)
    f_x1 ← f(x1)
    for k ← 1 to maxit do
        if |f_x0| > |f_x1| then
            swap(x0, x1)
            swap(f_x0, f_x1)
        s ← (x1 − x0)/(f_x1 − f_x0)
        x1 ← x0
        f_x1 ← f_x0
        x0 ← x0 − (f_x0 · s)
        f_x0 ← f(x0)
        if |x1 − x0| ≤ delta or f_xx0 ≤ epsilon then
            return (x0, f_x0, it, 0)
return (x0, v, 0, 1)                          ▷ Nie osiągnięto określonej precyzji
```

W przedstawionej wersji algorytmu, kolejne wartości funkcji mają nierosnące moduły. Podczas iteracji wyliczane są kolejne przybliżenia funkcji, zgodnie ze wzorem $x_{n+1} = x - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$.
Koniec następuje do osiągnięciu maksymalnej ilości iteracji(error = 1), tudzież po osiągnięciu żądanej precyzji.

# Zadanie 4

## Problem

Znaleźć pierwiastki równania $sinx - (\frac{1}{2}x)^2 = 0$, używając

- bisekcji na $[1.5, 2.0]$
- Newtona dla $x_0 = 1.5$
- siecznych dla $x_0 = 1, x_2 = 2$

i $\delta = \frac{1}{2}10^{-5}$ , $\epsilon = \frac{1}{2}10^{-5}$

## Rozwiązanie

Zadanie rozwiązane z użyciem zadanych metod, w dołączonym pliku Zadanie4.jl. Przyjęto maksymalną ilość iteracji równą 64.

## Wyniki

Poniższa tabela przedstawia zebrane wyniki.

| Algorytm | $x_0$ | $f(x_0)$ | Iteracji | Flaga |
|---|---|---|---|---|
| bisekcja | 1.9337539672851562 | -2.7027680138402843$\cdot 10^{-7}$ | 16 | 0 |
| Newtona | 1.933930573929843 | -2.2423316314856834$\cdot 10^{-8}$ | 4 | 0 |
| sieczne | 1.9337539405015145 | -2.3487103129049558$\cdot 10^{-7}$ | 5 | 0 |

Tablica 1 – Miejsca zerowe $f(x) = sinx - (\frac{1}{2}x)^2$

### Wnioski

Przedstawione wyniki nie dziwią. Liczba iteracji odzwierciedla zbieżność metod, najszybsza kwadratowa dla algorytmu Newtona skończyła się już po 4 krokach, metoda siecznych po 5, a najwolniejsza liniowa bisekcja potrzebowała 16 iteracji. Widać również, że metoda Newtona tak szybko zbiega do 0, że znaleziony pierwiastek odstaje od zadanej precyzji. Na tej podstawie możemy powiedzieć, że o ile metoda bisekcji nie jest najszybsza, jest najstabliniejsza i generuje dostatecznie dokładne wyniki. Ponadto metodę bisekcji można stosować globalnie.

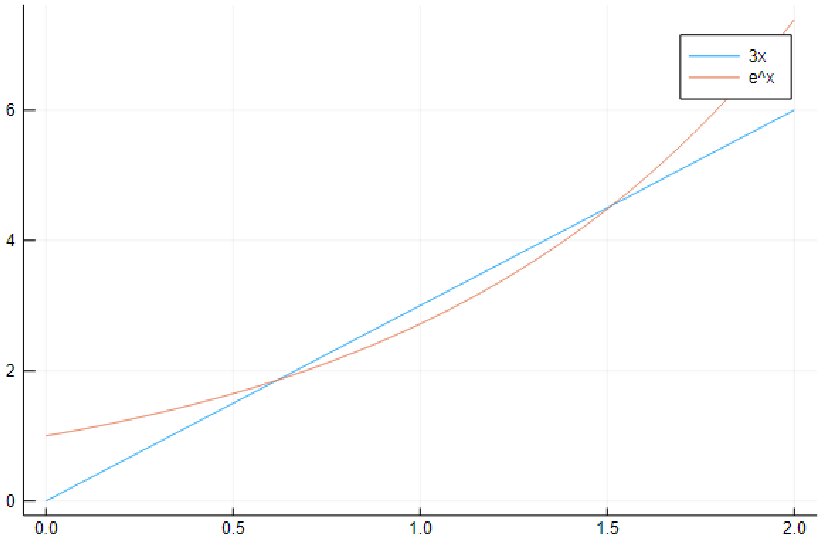
## Zadanie 5

### Problem

Znaleźć miejsca przecięć funkcji: $y_1 = 3x, y_2 = e^x$, czyli pierwiastki równania $g(x) = e^x - 3x = 0$, używając metody bisekcji. O zadanej dokładności: $\delta = 10^{-4}$ , $\epsilon = 10^{-4}$.

### Rozwiązanie

Znajdziemy pierwiastki funkcji $g(x) = 3x - e^x = 0$. Miejsca poszukiwania zostały wyznaczone graficznie z pomocą poniższego wykresu funkcji.




Rysunek 1 – Wykres danych funkcji

Jak widać, mamy dwóch kandydatów jeden pomiędzy $[0, 1]$, a drugi $[1, 2]$. Następnie zastosujemy metodę bisekcji.

### Wyniki

Otrzymane wyniki ilustruje poniższa tabela.

| | $x_0$ | $x_1$ |
|---|---|---|
| Przedział | $[0, 1]$ | $[1, 2]$ |
| Punkt | 0.619140625 | 1.5120849609375 |
| Wartość | -9.066320343276146·$10^5$ | -7.618578602830439·$10^5$ |
| Iteracje | 9 | 13 |

Tablica 2 – Miejsca zerowe $g(x) = 3x - e^x = 0$

### Wniosek

Istotą problemu jest poprawne wyznaczenie przedziałów poszukiwania. W tym przypadku najlepiej narysować sobie przebieg początkowy danych funkcji, lub mając odpowiednie zdolności numeryczne, wydedukować prawdopodobne miejsca zerowe. Bowiem dobranie niepoprawnego przedziału, prowadziło by do niekompletnego, lub błędnego rozwiązania.

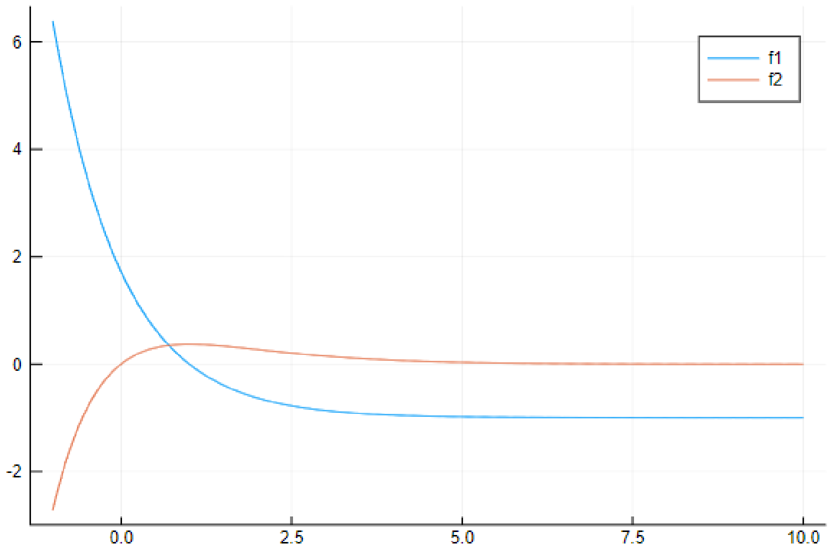
## Zadanie 6

### Problem

Znaleźć pierwiastki równania: $f_1(x) = e^{1-x} - 1$ oraz $f_2(x) = xe^{-x}$ z użyciem metod bisekcji, siecznych i Newtona. O zadanej dokładności: $\delta = 10^{-5}$ , $\epsilon = 10^{-5}$.

### Rozwiązanie

Miejsca poszukiwania zostały znalezione graficznie z pomocą poniższego wykresu funkcji. Ponadto przeprowadzono analize problematycznych miejsc.




Rysunek 2 – Wykres danych funkcji

W pierwszym przypadku ($f_1(x)$) mamy miejsce zerowe dla 1, w drugim ($f_2(x)$) dla 0. Zobaczmy co wyznaczą algorytmy.

### Wyniki

W tabeli 3 wyznaczono, rozwiązania poprawne algorytmu bisekcji. W tabeli 4 algorytmem Newtona, a w tabeli 5 algorytmem siecznych.

| Funkcja | przedział | $x_0$ | $f(x_0)$ | # Iteracji | Flaga |
|---|---|---|---|---|---|
| $f_1(x)$ | $[0.0, 1.5]$ | 1.0000076293945312 | -7.6293654275305656$\cdot 10^{-6}$ | 16 | 0 |
| $f_1(x)$ | $[0.0, 10.0]$ | 1.0000076293945312 | -3.814689989667386$\cdot 10^{-6}$ | 19 | 0 |
| $f_1(x)$ | $[-100.0, 1000.0]$ | 1.0000020265579224 | -2.026555868894775$\cdot 10^{-6}$ | 26 | 0 |
| $f_2(x)$ | $[-0.5, 1.0]$ | -7.62939453125$\cdot 10^{-6}$ | -7.629452739132958$\cdot 10^{-6}$ | 16 | 0 |
| $f_2(x)$ | $[-0.5, 10.0]$ | -1.9073486328125$\cdot 10^{-6}$ | -1.9073522707947765$\cdot 10^{-6}$ | 18 | 0 |
| $f_2(x)$ | $[-5.0, 1000.0]$ | 497.5 | -4.318056675122998$\cdot 10^{-214}$ | 1 | 0 |

Tablica 3 – Rozwiązania z użyciem metody bisekcji

| $x_0$ | $r$ | $f(r)$ | # Iteracji | Flaga |
|---|---|---|---|---|
| $f_1(x) = e^{1-x} - 1$ | | | | |
| 0.5 | 0.9999850223207645 | $1.1216494399945987 \cdot 10^{-10}$ | 4 | 0 |
| 5 | err | err | 32 | 1 |
| 100 | err | err | 0 | 2 |
| $f_2(x) = x \cdot e^{-x}$ | | | | |
| -0.5 | -0.0005537098560354364 | $-3.0642502806087233 \cdot 10^{-7}$ | 4 | 0 |
| 1 | err | err | 0 | 2 |
| 10 | 13.299218178860919 | $8.173205649825561 \cdot 10^{-6}$ | 4 | 0 |

Tablica 4 – Rozwiązania z użyciem metody Newtona

| punkt a,b | $r$ | $f(r)$ | # Iteracji | Flaga |
|---|---|---|---|---|
| $f_1(x) = e^{1-x} - 1$ | | | | |
| $(-1.0, 2.0)$ | 1.0000009310146594 | $-9.310142259355558 \cdot 10^{-7}$ | 7 | 0 |
| $(-1.0, 5.0)$ | 3.4954649211875086 | -0.9175418940419083 | 32 | 1 |
| $f_2(x) = x \cdot e^{-x}$ | | | | |
| $(-1.0, 1.0)$ | $2.9415790318691894 \cdot 10^{-8}$ | $2.9415789453403187 \cdot 10^{-8}$ | 11 | 0 |
| $(-3.0, 3.0)$ | 14.694632087522688 | $6.1004389854849775 \cdot 10^{-6}$ | 16 | 0 |
| $(10.0, 20.0)$ | 20.00090808104888 | $4.1187525544928226 \cdot 10^{-8}$ | 1 | 0 |

Tablica 5 – Rozwiązania z użyciem metody siecznych

## Wniosek

Jak można było przewidzieć metoda biskecji działa poprawnie i stabilnie. Zwiększa się jedynie ilość iteracji, dla większych przedziałów, co ma oczywisty sens. Uważać trzeba jednak na $f_2$, gdzie dla dostatecznie dużych przedziałów, otrzymamy po pierwszej iteracji wartość bliską 0, ale nie jemu równą.

W przypadku z metodą Newtona, trzeba uważać na początkową wartość $x_0$. W $f_1$ dla wartości $x > 1$, ilość iteracji zaczyna gwałtownie rosnąć, aż do osiągniecia wartości, gdzie pochodna jest na tyle blisko 0, żeby terminować metodę. W $f_2$, dla $x = 0$, pochodna się zeruje, co powoduje terminacje metody, dla $x > 1$, funkcja zbiega do nieprawidłowego miejsca zerowego będącą asymptotą $x = 0$.

Metoda stycznych działa analogicznie do metody Newtona. Dla większych przedziałów wzrasta szybko ilość iteracji. A w $f_2$, dla dostatecznie dużych punktów, będziemy otrzymywać wartości bliskie asymptoty równej 0.