Obliczenia Naukowe

Lista 3

Paweł Kędzierski

Zad 1

```
"""
Finds a root of a function using the method of bisection
    Arguments:
        f: input function f(x)
        a: beggining of an interval
        b: end of an interval
        delta: precision of r calculation
        epsilon: precision of f(r) calculation
    Returns:
        (r, w, it, err) a tuple of parameters
        r: found root
        w: f(r)
        it: number of iterations
        err: 0 if no errors occured, 1 otherwise
"""
function mbisekcji(f, a::Float64, b::Float64, delta::Float64, epsilon::Float64)

    e = b - a
    u = f(a)
    v = f(b)
    it = 0
    r = 0
    w = 0
    if sign(f(a)) == sign(f(b))
        return (r, w, it, 1)
    end

    while true
        it += 1
        e = e / 2
        r = (a + b) / 2
        w = f(r)

        if abs(e) < delta || abs(w) < epsilon
            return (r, w, it, 0)
        end
        if sign(w) != sign(u)
            b = r
            v = w
        else
            a = r
            u = w
        end
    end
end
```

Zad.2

```
"""
Finds a root of a function using the Newton's method
    Arguments:
        f: input function f(x)
        f: derivative of the input function
        x0: initial approximation
        delta: precision of r calculation
        epsilon: precision of f(r) calculation
        maxit: maximum number of iterations
    Returns:
        (r, w, it, err) a tuple of parameters
        r: found root
        v: f(r)
        it: number of iterations
        err: 0 if no errors occured, 1 otherwise
"""
function mstycznych(f, pf, x0::Float64, delta::Float64, epsilon::Float64, maxit::Int)
    v = f(x0)
    if abs(v) < epsilon
        return (x0, v, 0, 1)
    end
    it = 0
    for it = 1:maxit
        if pf(x0) == 0
            return (x1, v, it, 2)
        end
        x1 = x0 - v / pf(x0)
        v = f(x1)
        if abs(x1 - x0) < delta || abs(v) < epsilon
            return (x1, v, it, 0)
        end
        x0 = x1
    end
    return (x1, v, it, 1)
end
```

Zad. 3

```
"""
Finds a root of a function using a method of bisection
    Arguments:
        f: input function f(x)
        x0: initial approximation
        x1: initial approximation
        delta: precision of r calculation
        epsilon: precision of f(r) calculation
        maxit: maximum number of iterations
    Returns:
        (r, w, it, err) a tuple of parameters
        r: found root
        w: f(r)
        it: number of iterations
        err: 0 if no errors occured, 1 otherwise
"""
function msiecznych(f, a::Float64, b::Float64,
    delta::Float64, epsilon::Float64, maxit::Int)
    fa = f(a)
    fb = f(b)
    for it = 1:maxit
        if abs(fa) > abs(fb)
            temp = a
            a = b
            b = temp
            temp = fa
            fa = fb
            fb = temp
        end
        s = (b - a) / (fb - fa)
        b = a
        fb = fa
        a = a - fa * s
        fa = f(a)
        if(abs(b - a) < delta || abs(fa) < epsilon)
            return (a, fa, it, 0)
        end
    end
    return (a, fa, it, 1)
end
```

Zad 4

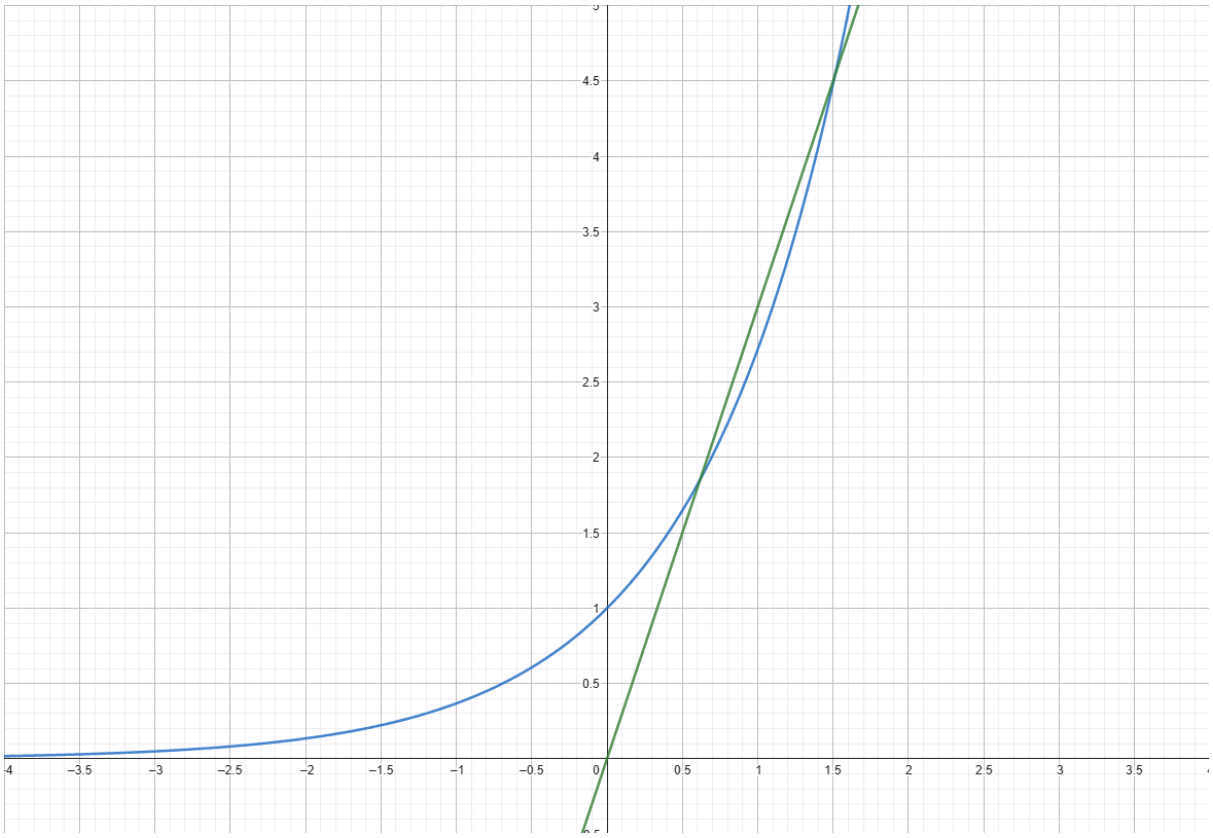
$$f(x) = \sin(x) - \frac{x^2}{4}$$

Metoda	r	f(r)	Liczba iteracji
Bisekcji	1.9337539672851562	-2.7027680138402843e-7	16
Newtona	1.933753779789742	-2.2423316314856834e-8	4
Siecznych	1.933753644474301	1.564525129449379e-7	4

Możemy zauważyć że metoda bisekcji potrzebowała najwiekszą ilość iteracji. Metody Newtona oraz siecznych posiadają mniejszą liczbe iteracji

Zad 5

Prawidłowe miejsca przecięć widzimy na tym wykresie:



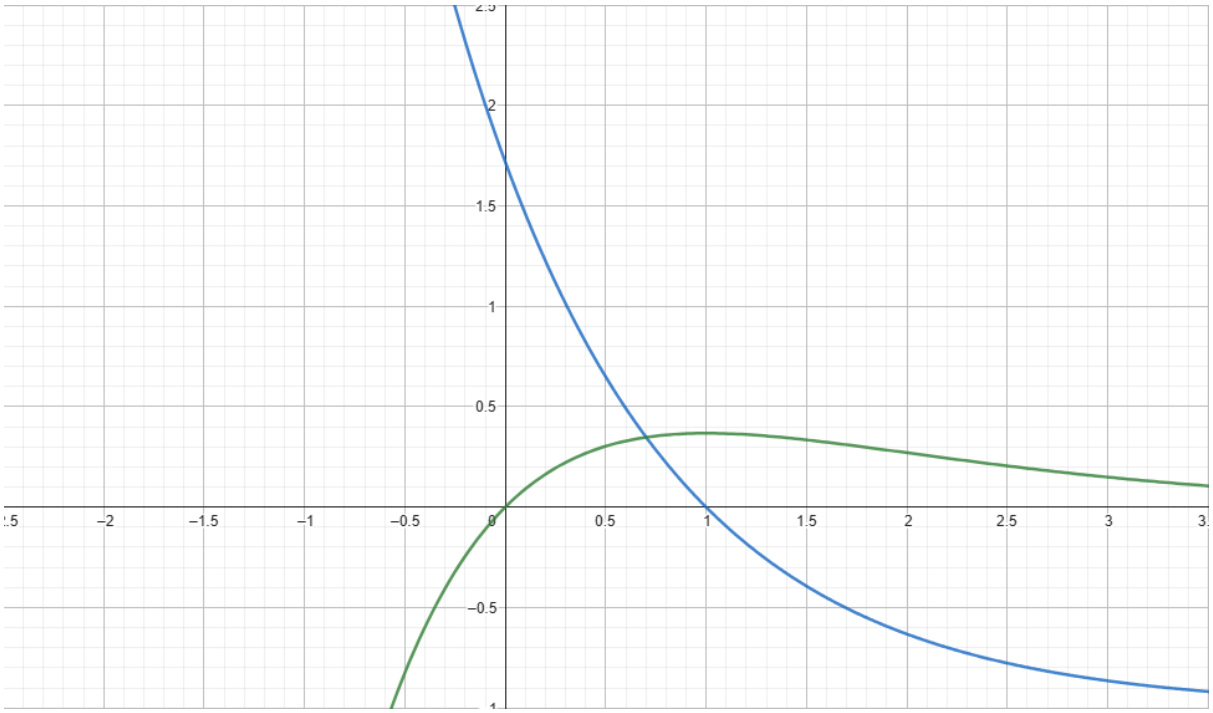
$$f(x) = 3x - e^x$$

Przedział	r	f(r)	Liczba iteracji
[0, 1]	0.6190643310546875	3.4790879874790903e-6	16
[1, 2]	1.5121307373046875	5.86035312810651e-6	16

W tym zadaniu aby wyznaczyć poprawny punkty przecięcia funkcji musiałem dobrać odpowiedni przedział. W przypadku złego dobrania punktów przecięcia wychodziły mi niepoprawne wyniki

Zad 6

Prawidłowe miejsca zerowe widzimy na tym wykresie:



Metoda Bisekcji

$$f(x) = e^{1-x} - 1$$

Przedział	x	f(x)	Liczba iteracji
[0, 1]	0.9999923706054688	7.629423635080457e-6	17
[-1, 1]	0.9999923706054688	7.629423635080457e-6	18
[-3, 3]	1.0000076293945312	-7.6293654275305656e-6	9

$$f(x) = xe^{-x}$$

Przedział	x	f(x)	Liczba iteracji
[0, 1]	7.62939453125e-6	7.629423635080457e-6	17
[-1, 1]	0	7.629423635080457e-6	1

W przypadku tej metody ważne było dobranie odpowiedniego przedziału w celu znalezienia miejsc zerowych

Metoda Newtona (przyjąłem maxit = 100)

$$f(x) = e^{1-x} - 1$$

Początkowe x0	x	f(x)	Liczba iteracji
0	0.9999984358892101	7.629423635080457e-6	4
1	1	0	0
2.0	0.9999999810061002	1.8993900008368314e-8	5

$$f(x) = xe^{-x}$$

Początkowe x0	x	f(x)	Liczba iteracji
-1	-3.0642493416461764e-7	7.629423635080457e-6	17
0	0	0	0
5	15.194283983439147	3.827247505782993e-6	9
1	ERROR	ERROR	ERROR

W przypadku tej metody ważne było dobranie odpowiedniego początkowego x0 w celu znalezienia miejsc zerowych. W momencie wybrania x0= 1 program nie jest w stanie znaleźć miejsca zerowego. Dzieję się tak ponieważ wartość pochodnej w tym punkcie jest równa 0, a więc styczna jest równoległa do osi OX co uniemożliwia wyznaczenie kolejnego punktu przecięcia.

Metoda Siecznych

$$f(x) = e^{1-x} - 1$$

Początkowe x0	Początkowe x1	x	f(x)	Liczba iteracji
0	0.5	0.9999998133327657	1.8666725165594755e-7	5
1	1.05	1	0	1

$$f(x) = xe^{-x}$$

Początkowe x0	Początkowe x1	x	f(x)	Liczba iteracji
-1	-0.5	-1.2229958402039555e-7	-1.2229959897758473e-7	6
0	0.5	0	0	1

W przypadku tej metody ważne było dobranie odpowiedniego początkowego x0 oraz x1 w celu znalezienia miejsc zerowych