CN25 - Homework 1

Matteo Mazzetti 0001161552

1 Funzioni da analizzare

Calcoleremo uno zero di queste funzioni scritte in seguito con i metodi di bisezione, del punto fisso e di Newton.
Rappresentiamo prima graficamente le varie funzioni:

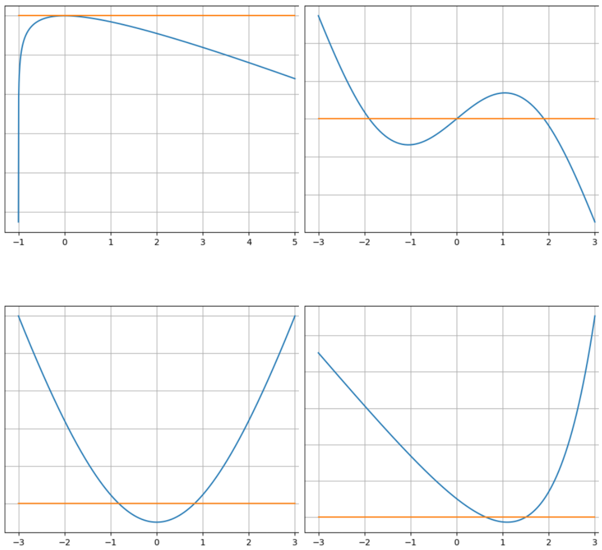

Figure 1:
Top Left: $f_1(x) = ln(x+1) - x$
Bottom Left: $f_2(x) = x^2 - cos(x)$
Top Right: $f_3(x) = sin(x) - \frac{x}{2}$
Bottom Right: $f_4(x) = e^x - 3x$

2 Metodo di bisezione

Utilizzeremo il codice di bisezione implementato così in Python:

```
def bisezione(f,a,b,maxIt,t):
    i=0
    if(np.abs(f(a))<t): return (a,f(a),i)
    elif(np.abs(f(b))<t): return (b,f(b),i)
    for i in range(maxIt):
        c=(a+b)/2
        fc=f(c)
        if(abs(fc)<t):
            return (c,fc,i)
        elif(f(a)*fc<0):
            b=c
        else:
            a=c
    return (c,fc,i)
```

Per f_1 prendendo come estremi $a = 0$ e $b = 0.5$ otteniamo $\overline{x} = 0$ in 0 iterazioni.

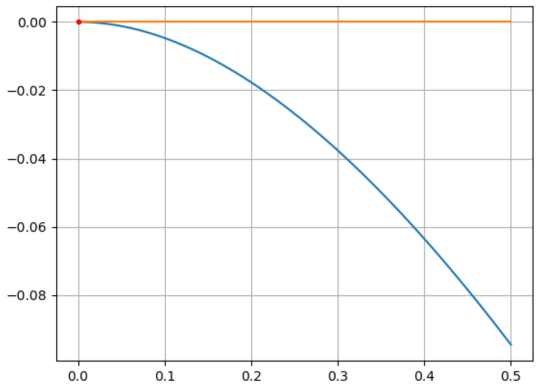

Figure 2: f_1: a=0, b=0.5

Per f_2 prendendo come estremi $a = -1.5$ e $b = 0$ otteniamo $\overline{x} = -0.8241323122638278$ in 32 iterazioni.

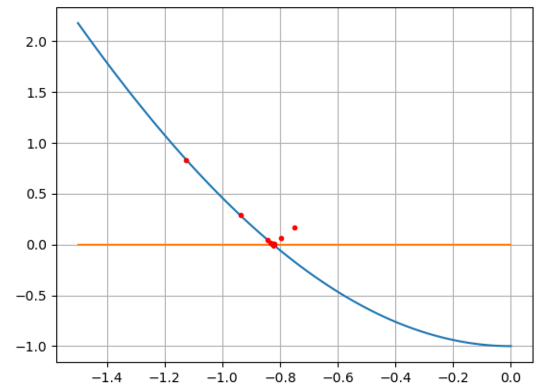

Figure 3: f_2: a=-1.5, b=0

Per f_3 prendendo come estremi $a = -1$ e $b = 1.5$ otteniamo $\overline{x} = 0$ in 32 iterazioni. Mentre se come estremi prendessimo $a = 1.5$ e $b = 2.5$ troveremmo uno zero in $\overline{x} = 1.8954942671116441$ in 31 iterazioni.

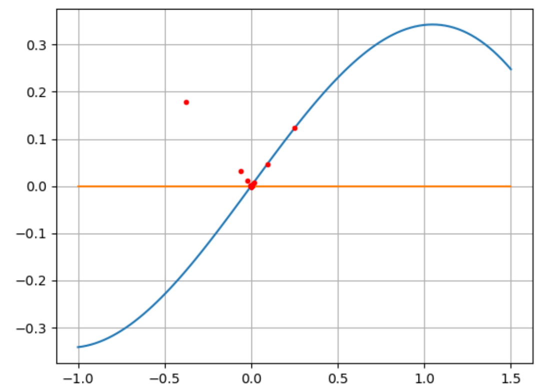

Figure 4: f_3: a=-1, b=1.5

Per f_4 prendendo come estremi $a = 0$ e $b = 1.5$ otteniamo $\overline{x} = 0.6190612867358141$ in 0 iterazioni.

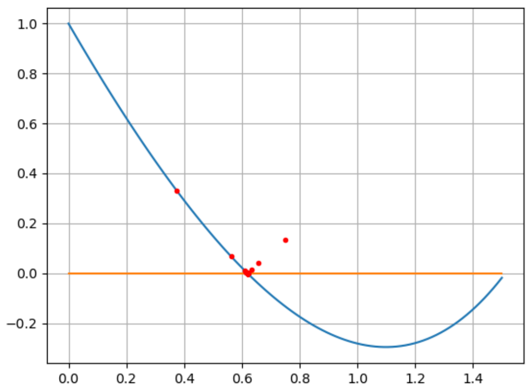

Figure 5: f_4: a=0, b=1.5

3 Metodo del Punto fisso

Vediamo invece adesso il codice del metodo iterativo del punto fisso:

```
def puntoFisso(f,g,x0,t1,t2,maxIt):
    cont=0
    while(np.abs(f(x0))>t1 and cont<maxIt):
        xNew=g(x0)
        delta=np.abs(x0-xNew)
        if(delta<t2):
            break
        x0=xNew
        cont+=1
    return (x0,f(x0),cont)
```

Per utilizzare questo metodo dobbiamo però definire le seguenti funzioni:
$g_1(x) = ln(x + 1)$
$g_2(x) = \sqrt{cos(x)}$
$g_3(x) = 2sin(x)$
$g_4(x) = \frac{1}{3}e^x$
Applichiamo l'algoritmo più volte ad ogni funzione facendone variare i parametri:
Per f_1 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=0.9` otteniamo $\overline{x} = 0.039063348805162205$ in 50 iterazioni;
2. se `maxIt=500, t1=t2=1.e-6, x0=0.9` otteniamo $\overline{x} = 0.003996483610013262$ in 500 iterazioni;
3. se `maxIt=50, t1=t2=1.e-2, x0=0.9` otteniamo $\overline{x} = 0.14666073551613687$ in 12 iterazioni;

Per f_2 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=-0.2` otteniamo $\overline{x} = 0.8241327404734513$ in 17 iterazioni;
2. se `maxIt=50, t1=t2=1.e-6, x0=1` otteniamo $\overline{x} = 0.8241327682687996$ in 16 iterazioni;
3. se `maxIt=50, t1=t2=1.e-10, x0=1` otteniamo $\overline{x} = 0.8241323122402889$ in 27 iterazioni;

Per f_3 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=0.5` otteniamo $\overline{x} = 1.895495156915764$ in 29 iterazioni;
2. se `maxIt=50, t1=t2=1.e-6, x0=-1` otteniamo $\overline{x} = 1.8954950832865742$ in 28 iterazioni;

Per f_4 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=1` otteniamo $\overline{x} = 0.6190630245284593$ in 27 iterazioni;
2. se `maxIt=10, t1=t2=1.e-2, x0=0.2` otteniamo $\overline{x} = 0.6168507253761869$ in 10 iterazioni;

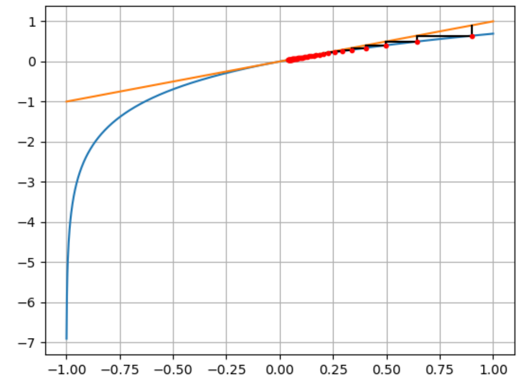

Figure 6: f_1: maxIt=50, t1=t2=1.e-6, x0=0.9

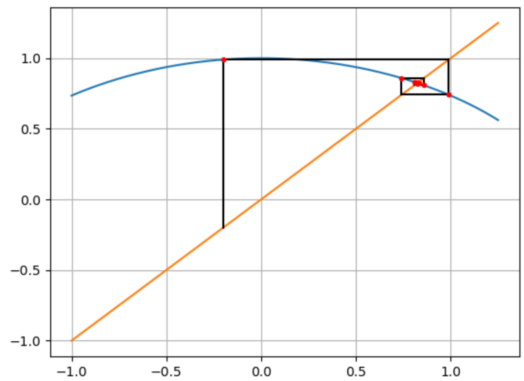

Figure 7: f_2: maxIt=50, t1=t2=1.e-6, x0=-0.2

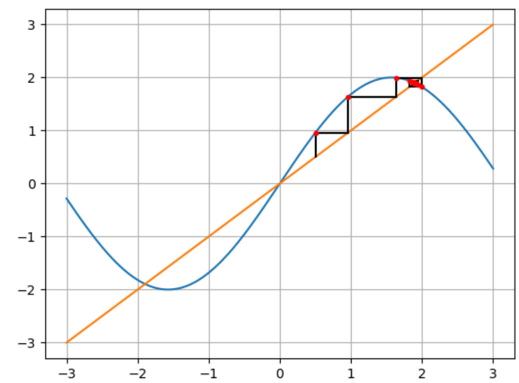

Figure 8: f_3: maxIt=50, t1=t2=1.e-6, x0=0.5

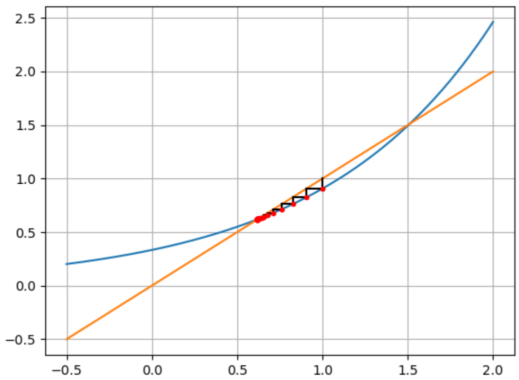

Figure 9: f_4: maxIt=50, t1=t2=1.e-6, x0=1

4 Metodo di Newton

Vediamo ora un altro metodo iterativo la cui implemantazione in python è:

```
def newton(f,Df,x0,t1,t2,maxIt,a,b):
    cont=0
    while (np.abs(f(x0))>t1 and cont<maxIt):
        fx0=f(x0)
        Dfx0=Df(x0)
        xNew=x0-fx0/Dfx0
        delta=np.abs(x0-xNew)
        if(delta<t2):
            break
        x0=xNew
        cont+=1
    return(x0,fx0,cont)
```

Notiamo che per questo metodo dobbiamo calcolare le derivate prime delle funzioni che stiamo studiando.
$f_1'(x) = \frac{1}{(x+1)} - 1$
$f_2'(x) = 2x + sin(x)$
$f_3'(x) = cos(x) - 1/2$
$f_4'(x) = e^x - 3$
Applichiamo l'algoritmo più volte ad ogni funzione facendone variare i parametri:
Per f_1 abbiamo che:

1. se `maxIt=10, t1=t2=1.e-6, x0=0.5` otteniamo $\overline{x} = 0.0007411225759261306$ in 9 iterazioni;
2. se `maxIt=50, t1=t2=1.e-6, x0=0.5` otteniamo $\overline{x} = 0.0007411225759261306$ in 9 iterazioni;
3. se `maxIt=50, t1=t2=1.e-6, x0=-0.4` otteniamo $\overline{x} = -0.001086842236741824$ in 9 iterazioni;

Per f_2 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=0.5` otteniamo $\overline{x} = 0.8241323124099124$ in 4 iterazioni;
2. se `maxIt=50, t1=t2=1.e-10, x0=0.5` otteniamo $\overline{x} = 0.8241323123025224$ in 5 iterazioni;
3. se `maxIt=50, t1=t2=1.e-6, x0=1` otteniamo $\overline{x} = 0.8241323190509289$ in 3 iterazioni;

Per f_3 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=1.5` otteniamo $\overline{x} = 1.8954942764727707$ in 4 iterazioni;
2. se `maxIt=50, t1=t2=1.e-6, x0=0.5` otteniamo $\overline{x} = -3.946500987334414 \cdot 10^{-10}$ in 3 iterazioni;

Per f_4 abbiamo che:

1. se `maxIt=50, t1=t2=1.e-6, x0=1` otteniamo $\overline{x} = 0.6190612833553127$ in 5 iterazioni;
2. se `maxIt=500, t1=t2=1.e-12, x0=1` otteniamo $\overline{x} = 0.6190612867359452$ in 6 iterazioni;

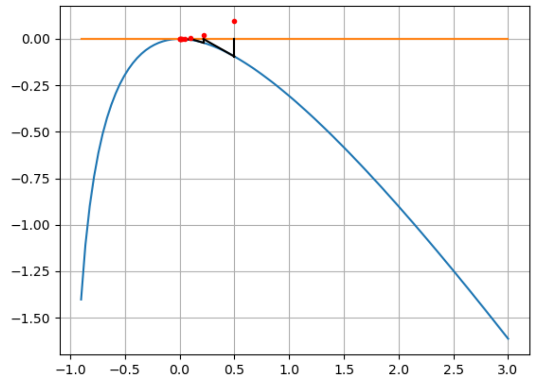

Figure 10: f_1: maxIt=50, t1=t2=1.e-6, x0=0.5

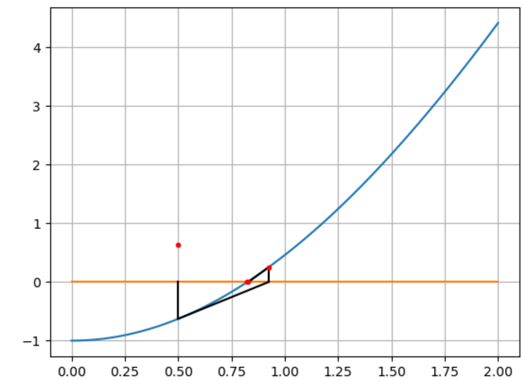

Figure 11: f_2: maxIt=50, t1=t2=1.e-6, x0=0.5

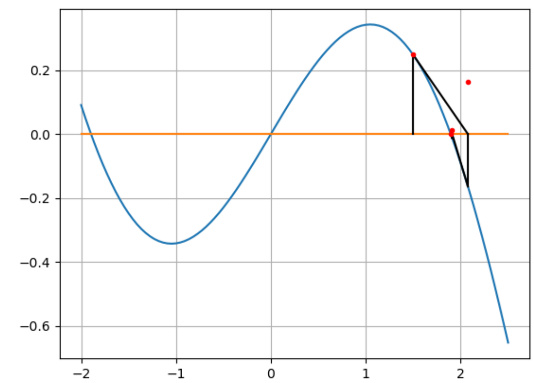

Figure 12: f_3: maxIt=50, t1=t2=1.e-6, x0=1.5

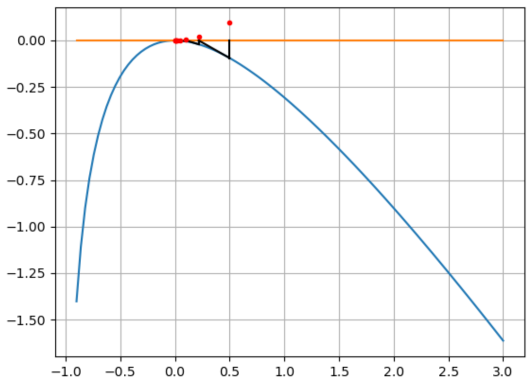

Figure 13: f_4: maxIt=50, t1=t2=1.e-6, x0=1