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:

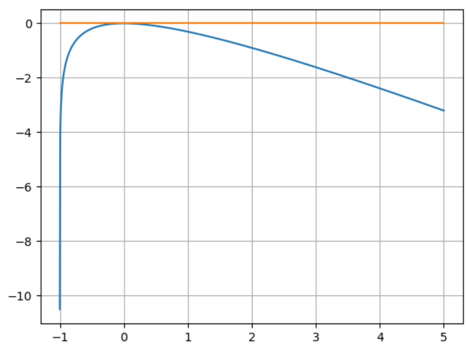

Figure 1: $f_1(x) = ln(x+1) - x$

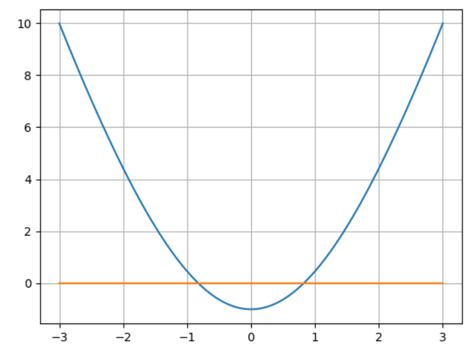

Figure 2: $f_2(x) = x^2 - cos(x)$

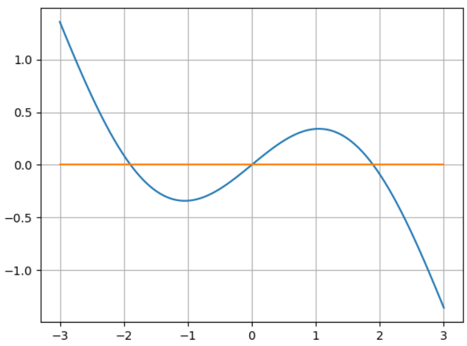

Figure 3: $f_3(x) = sin(x) - \frac{x}{2}$

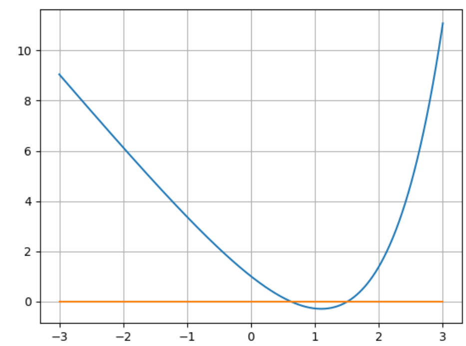

Figure 4: $f_4(x) = e^x - 3x$

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.

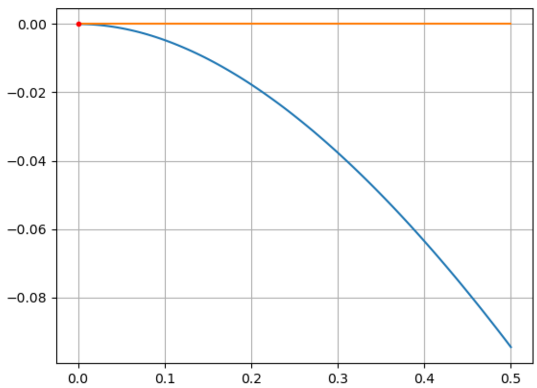

Figure 5: f_1: a=0, b=0.5

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

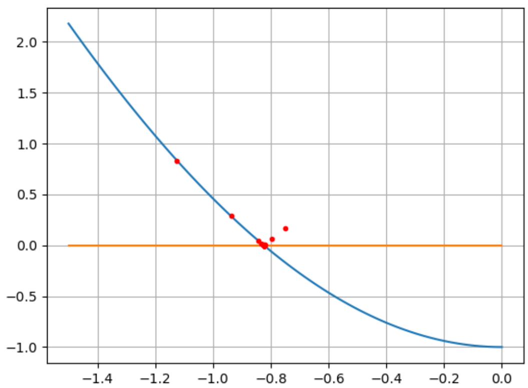

Figure 6: f_2: a=-1.5, b=0

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.

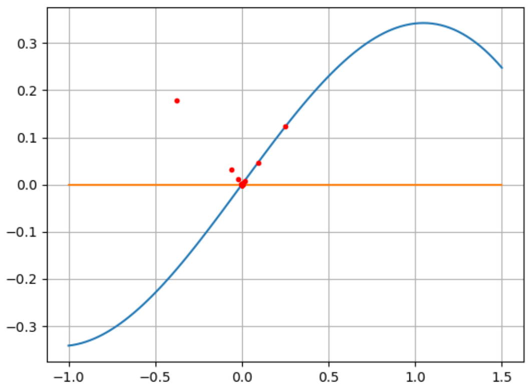

Figure 7: f_3: a=-1, b=1.5

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

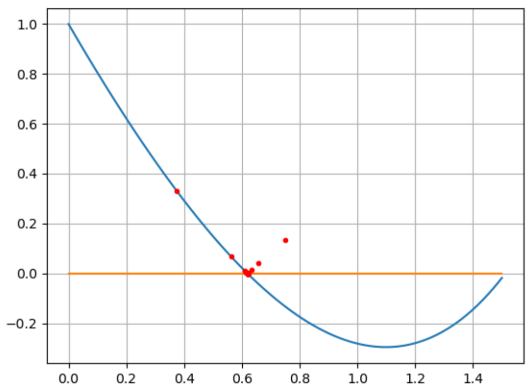

Figure 8: f_4: a=0, b=1.5

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:

Funzione	Test	maxIt	t1=t2	x0	x*	iters
f_1	1	10	1.e-6	0.9	0.03906334880516220	50
	2	500	1.e-6	0.9	0.00399648361001326	500
	3	50	1.e-2	0.9	0.1466607355161368	12
f_2	1	50	1.e-6	-0.2	0.824132740473451	17
	2	50	1.e-6	1	0.824132768268799	16
	3	50	1.e-10	1	0.824132312240288	27
f_3	1	50	1.e-6	0.5	1.89549515691576	29
	2	50	1.e-6	-1	1.895495083286574	28
f_4	1	50	1.e-6	1	0.619063024528459	27
	2	10	1.e-2	0.2	0.616850725376186	10

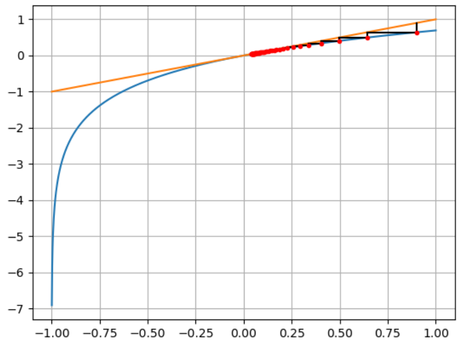

Figure 9: f_1: maxIt=50, t1=t2=1.e-6, x0=0.9

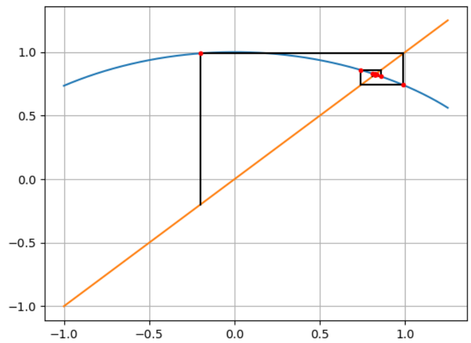

Figure 10: f_2: maxIt=50, t1=t2=1.e-6, x0=-0.2

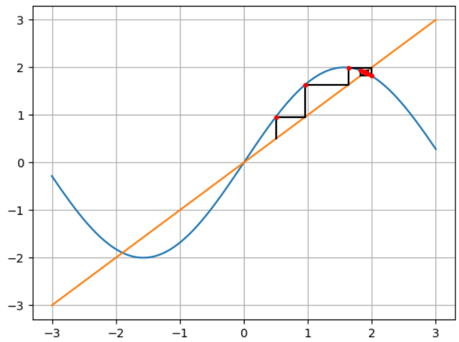

Figure 11: f_3: maxIt=50, t1=t2=1.e-6, x0=0.5

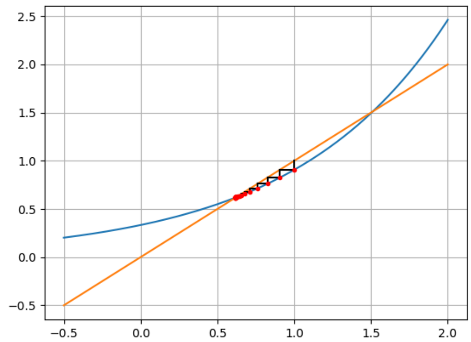

Figure 12: f_4: maxIt=50, t1=t2=1.e-6, x0=1

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:

Funzione	**Test**	**maxIt**	**t1=t2**	**x0**	**x***	**iters**
f_1	1	10	1.e-6	0.5	0.000741122575926130	9
	2	50	1.e-6	0.5	0.000741122575926130	9
	3	50	1.e-6	0.4	-0.00108684223674182	9
f_2	1	50	1.e-6	0.5	0.824132312409912	4
	2	50	1.e-10	0.5	0.824132312302522	5
	3	50	1.e-6	1	0.824132319050928	3
f_3	1	50	1.e-6	1.5	1.895494276472770	4
	2	50	1.e-6	0.5	-3.946500987334414.e-10	3
f_4	1	50	1.e-6	1	0.619061283355312	5
	2	500	1.e-12	1	0.619061286735945	6

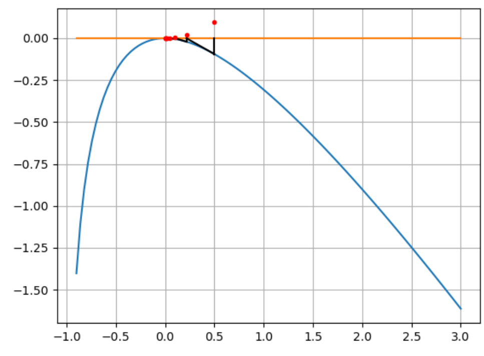

Figure 13: f_1: maxIt=50, t1=t2=1.e-6, x0=0.5

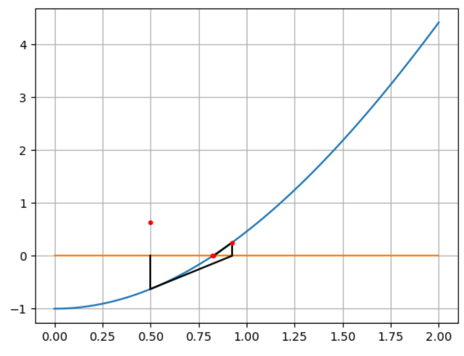

Figure 14: f_2: maxIt=50, t1=t2=1.e-6, x0=0.5

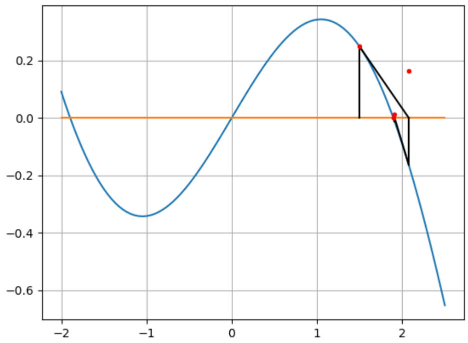

Figure 15: f_3: maxIt=50, t1=t2=1.e-6, x0=1.5

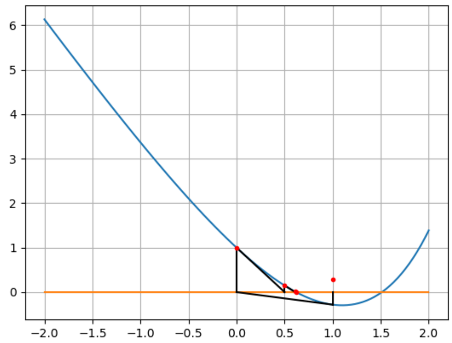

Figure 16: f_4: maxIt=50, t1=t2=1.e-6, x0=1