CN25 - Homework 2

Matteo Mazzetti 0001161552

1 Funzioni da analizzare

a) $f_a(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 2)^2$

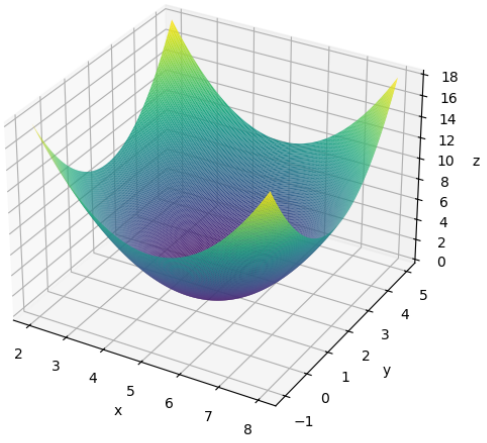

Figure 1: $f_a : \mathbb{R}^2 \to \mathbb{R}$

b) $f_b(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$

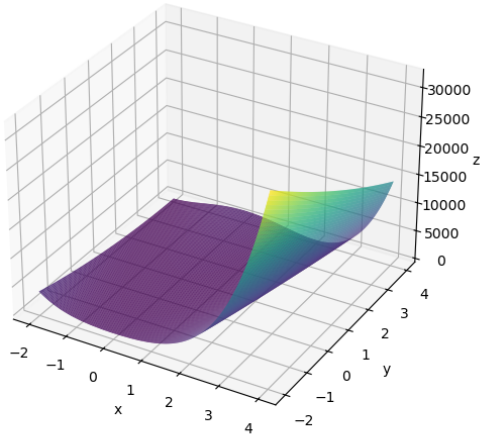

Figure 2: $f_b : \mathbb{R}^2 \to \mathbb{R}$

d) $f_d(x) = \frac{1}{2}\|Ax - b\|_2^2$ dove $A \in M_n(\mathbb{R})$ è una matrice con elementi casuali e $b \in \mathbb{R}^n$ si trova ponendo $\overline{x} = (1, 1, .., 1)^T$ e risolvendo $b = A\overline{x}$.
f) $f_f(x) = \sum_{i=1}^{n}(x_i - i)^2 - \sum_{i=1}^{n} ln(x_i)$ dove $x \in \mathbb{R}^n$ con $x_i > 0 \ \forall i = 1..n$

2 Metodo del gradiente

Per utilizzare il metodo del gradiente dobbiamo prima calcolare i gradienti delle quattro funzioni che stiamo studiando:
a) $\nabla f_a(x_1, x_2) = (2(x_1 - 5), 2(x_2 - 2))$
b) $\nabla f_a(x_1, x_2) = (-2(1 - x_1) - 400x_1(x_2 - x_1^2), 200(x_2 - x_1^2))$
d) $\nabla f_d(x) = A^T(Ax - b)$
f) $\partial_{x_i} f_f(x) = 2(x_i - i) - \frac{1}{x_i}$
Il metodo del gradiente è implementato in python così:

```
def GD(f,df,x0,alpha,maxIt,fToll,xToll,alphaConst):
    cont=0
    n=len(x0)
    dfNorm=np.linalg.norm(df(x0))
    while cont<maxIt and dfNorm>fToll:
        p=-df(x0)
        alpha=alpha if alphaConst else backtrackingFun(f,df,x0)
        xNew=x0+alpha*p
        if(np.linalg.norm(xNew-x0)<xToll):
            break
        x0=xNew
        dfNorm=np.linalg.norm(df(x0))
        cont+=1
    return x0,f(x0),cont
```

Dove l'algoritmo di backtracking ha il seguente codice:

```
def backtrackingFun(f,df,x,alpha=1,rho=0.5,c=1.e-4):
    while f(x-alpha*df(x))>f(x)-c*alpha*(np.linalg.norm(df(x))**2):
        alpha*=rho
    return alpha
```

Utilizziamo delle tabelle per visualizzare meglio i risultati.

2.1 Funzione (a)

Per la funzione $f_a(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 2)^2$ si ha che:

Test	x0	Alpha	AlphaConst	Iters	x*	f(x*)
1	(7,-1)	1	F	1	(5,2)	0
2	(7,-1)	0.001	T	3288	(5.00276853, 1.9958472)	2.491052370848344e-05
3	(5,0)	0.001	T	2993	(5,1.99500265)	2.4973541699900752e-05
4	(-100,100)	15	F	1	(5,2)	0

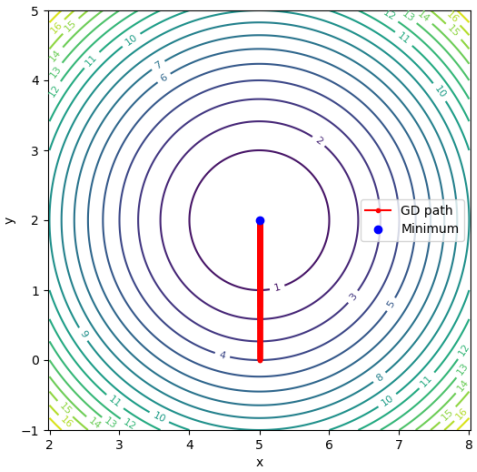

Figure 3: f_a Test 3

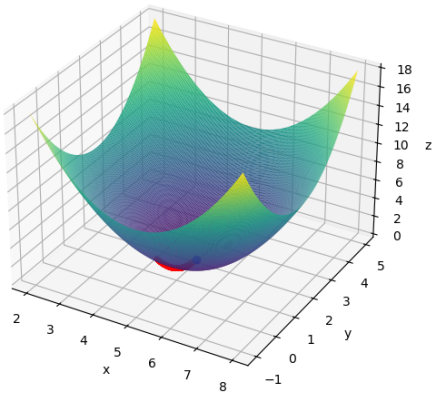

Figure 4: f_a Test 3

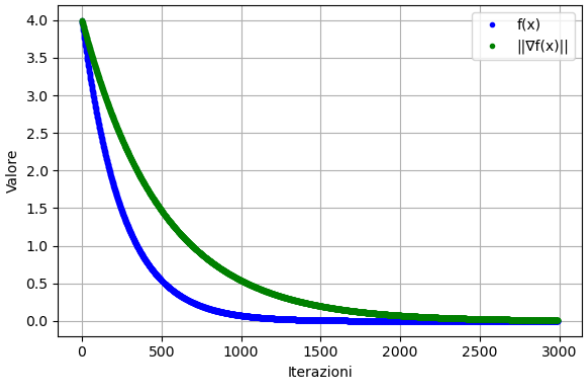

Figure 5: f_a Test 3

2.2 Funzione (b)

Per la funzione $f_b(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ nota come funzione di Rosenbrock si ha che:

Test	x0	Alpha	AlphaConst	Iters	x*	f(x*)
1	(0,2)	0.01	T	\	\	\
2	(0,2)	0.005	T	10000[1]	(0.53419238, 0.21465105)	0.7169736034126697
3	(0,2)	0.001	T	8270	(0.98892228, 0.97792265)	0.0001229150422919122
4	(0,2)	0.001	F	4375	(0.99616119, 0.99231348)	1.4792320012593029e-05

[1] L'algoritmo termina sempre per iterazioni anche con maxIt=30000 e 100000.

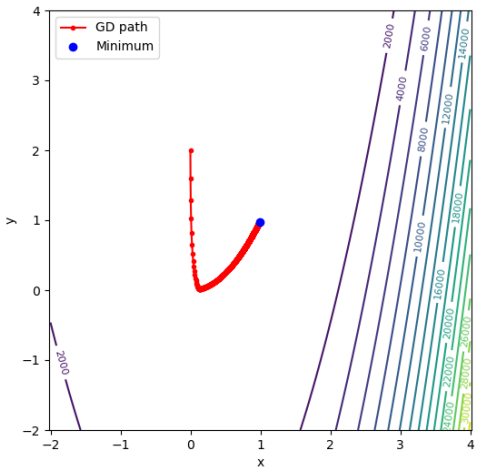

Figure 6: f_b Test 3

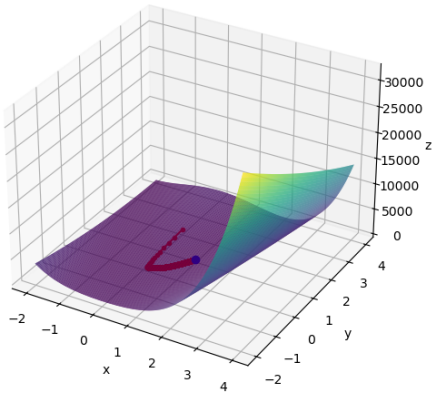

Figure 7: f_b Test 3

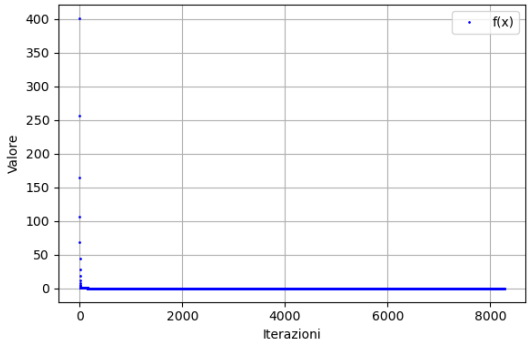

Figure 8: f_b Test 3

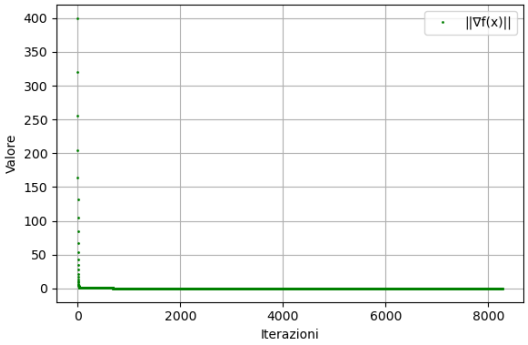

Figure 9: f_b Test 3

2.3 Funzione (d)

La funzione dei minimi quadrati lineari, ossia $f_d(x) = \frac{1}{2}\|Ax-b\|_2^2$ richiede che A sia una matrice quadrata casuale, nel nostro caso

$$A = \begin{pmatrix} 0.53632201 & 0.87712709 & 0.26605249 & 0.29726022 & 0.53965021 \\ 0.37268831 & 0.10548683 & 0.35747810 & 0.72337934 & 0.10020472 \\ 0.13209922 & 0.16335677 & 0.49748149 & 0.43642859 & 0.59638987 \\ 0.36028065 & 0.81779636 & 0.84351303 & 0.44975310 & 0.96032015 \\ 0.91500373 & 0.81463979 & 0.81824263 & 0.78417968 & 0.85253613 \end{pmatrix} \in M_5(\mathbb{R})$$

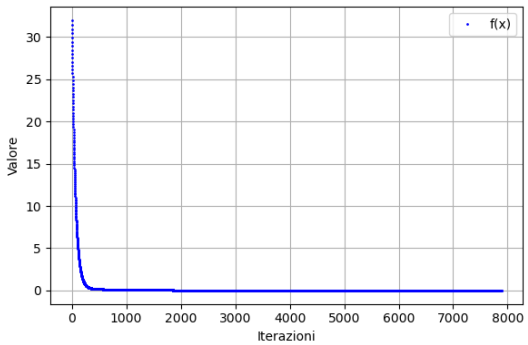

Figure 10: f_d Test 2

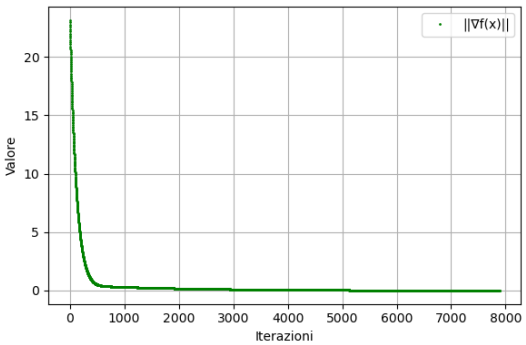

Figure 11: f_d Test 2

Test	x0	Alpha	AlphaConst	Iters	x*	f(x*)
1	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	F	9	(1.36602764, 2.22474487, 3.1583124, 4.12132034, 5.09807621)	-5.0727370374321525
2	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	1754	(1.36306408, 2.22214764, 3.15685415, 4.12074654, 5.09812989)	-5.072715912285671
3	(1.1, 2.1, 3.1, 4.1, 5.1)	0.005	T	485	(1.36550361, 2.22416712, 3.15796424, 4.12117936, 5.09808959)	-5.072736176747404
4	(1.1, 2.1, 3.1, 4.1, 5.1)	0.1	T	34	(1.36601339, 2.22471854, 3.15829316, 4.12131194, 5.09807704)	-5.072737036030224

2.4 Funzione (f)

La funzione che studieremo ora è $f_f(x) = \sum_{i=1}^{n}(x_i - i)^2 - \sum_{i=1}^{n} ln(x_i)$, dove $n = 5$ quindi $f_f : \mathbb{R}^5 \to \mathbb{R}$. Come al solito, utilizziamo una tabella per analizzare i vari test.

Test	x0	Alpha	AlphaConst	Iters	x*	f(x*)
1	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	F	9	(1.36602764, 2.22474487, 3.1583124, 4.12132034, 5.09807621)	-5.0727370374321525
2	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	1754	(1.36306408, 2.22214764, 3.15685415, 4.12074654, 5.09812989)	-5.072715912285671
3	(1.1, 2.1, 3.1, 4.1, 5.1)	0.005	T	485	(1.36550361, 2.22416712, 3.15796424, 4.12117936, 5.09808959)	-5.072736176747404
4	(1.1, 2.1, 3.1, 4.1, 5.1)	0.1	T	34	(1.36601339, 2.22471854, 3.15829316, 4.12131194, 5.09807704)	-5.072737036030224
5	(1, 1, 1, 1, 1)	0.001	T	3416	(1.36596667, 2.22413235, 3.15675625, 4.11869197, 5.09434125)	-5.072712746753138

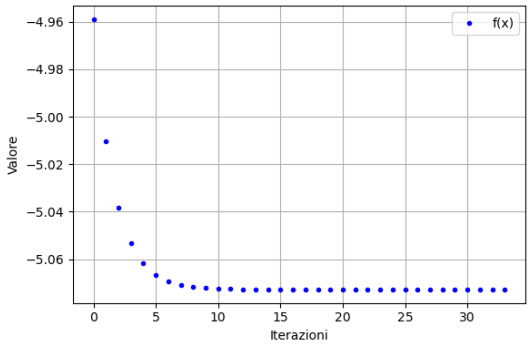

Figure 12: f_f Test 4

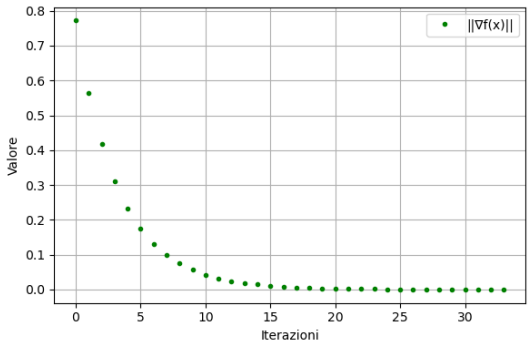

Figure 13: f_f Test 4

3 Metodo del gradiente stocastico

Utilizziamo ora il metodo del gradiente stocastico sull'ultima funzione che abbiamo visto: $f_f(x) = \sum_{i=1}^{n}(x_i - i)^2 - \sum_{i=1}^{n} ln(x_i)$ dove $x \in \mathbb{R}^n$ con $x_i > 0 \ \forall i = 1..n$ per $n = 5$.
Possiamo farlo poiché f_f è una funzione che è somma, a sua volta, di altre funzioni.

```
def SGD (f,df,x0,maxEpoche,fToll,xToll,alpha,k):
    n=len(x0)
    S_k = np.arange(0,n,1)
    cont=0
    batch = S_k[:k]
    dfVals = [np.sum(df(x0, j) for j in batch)]
    dfNorm=np.linalg.norm(dfVals[-1])
    for epoca in range(0,maxEpoche):
        np.random.shuffle(S_k)
        for i in range(0,n,k):
            batch=S_k[i:i+k]
            p=-np.sum([df(x0, j) for j in batch])
            xNew=x0+alpha*p
            if (np.linalg.norm(xNew-x0)<xToll):
                return x0,f(x0),epoca,cont
            x0=xNew
            cont+=1
            dfVals.append(p)
        dfNorm=np.linalg.norm(dfVals[-1])
        if(dfNorm<fToll):
            break
    return x0,f(x0),epoca,cont
```

Confrontiamo l'errore relativo a seconda della dimensione dei mini-batch(k). Confrontiamolo anche con il metodo del gradiente non stocastico. Notiamo che i Test 1-3 sono stati fatti tutti con maxEpoche=100. Sull'asse delle ascisse abbiamo il numero di iterazioni mentre su quello delle ordinate ci sono i valori dell'errore relativo, dove abbiamo preso $x_i^* = \frac{i+\sqrt{i^2+2}}{2}$.

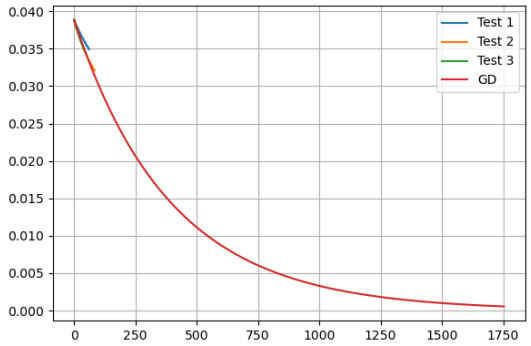

Figure 14: f_f Confronto

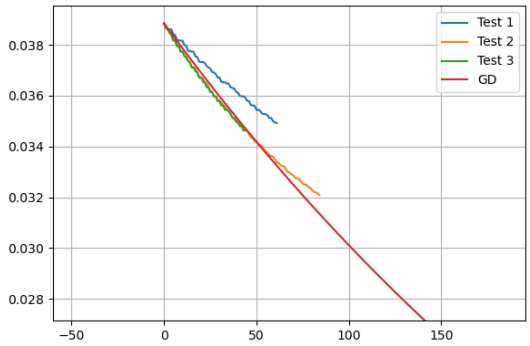

Figure 15: f_f Zoom del confronto

Test	**x0**	**Alpha**	**AlphaConst**	**Iters**	**k**	**x***	**f(x*)**
GD	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	1754	\	(1.36306408, 2.22214764, 3.15685415, 4.12074654, 5.09812989)	-5.072715912285671
SGD 1	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	62	2	(1.12152188, 2.12152188, 3.12152188, 4.12152188, 5.12152188)	-4.980977230740051
SGD 2	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	85	3	(1.13923524, 2.13923524, 3.13923524, 4.13923524, 5.13923524)	-4.995267877673674
SGD 3	(1.1, 2.1, 3.1, 4.1, 5.1)	0.001	T	45	4	(1.12334525, 2.12334525, 3.12334525, 4.12334525, 5.12334525)	-4.982610612476134