Algoritmos en la Ciencia de Datos
Sección U

Eduardo Andrés Santizo Olivet
Carné: 21007361

Laboratorio No. 4

Gradient Descent Variants y Newton's Method

Problema 1 – GD Variants

En este primer problema del laboratorio, consideraremos el problema de ajustar una recta de regresión a un conjunto de datos. El modelo de regresión está dado por la ecuación $Ax = b$, en donde A representa el conjunto de entradas de dimensión $n \times d$ y b representa el conjunto de observaciones de dimensión $n \times 1$. El objetivo es determinar el vector de coeficientes x de dimensión $d \times 1$ y que minimiza la suma de errores al cuadrado:

$$\min_{x \in \mathbb{R}^d} f(x) \triangleq \sum_{i=1}^{n} f_i(x)$$

en donde, $f_i(x) = \left(x^T a_i - b_i\right)^2$ denota el cuadrado del error de la i-ésima observación $a_i \in \mathbb{R}^d$ y $b_i \in \mathbb{R}$. Se utilizarán los datos previamente explicados para estudiar cuatro formas diferentes de calcular el vector x^*, estas son: Solución cerrada, el algoritmo Gradient Descent (GD), el algoritmo de Stochastic Gradient Descent (SGD) y Mini Batch Gradient Descent (MBGD).

Parte 1 – Solución Cerrada

El problema puede resolverse de manera cerrada a través de la siguiente ecuación:

$$x^* = (A^T A)^{-1} A^T b$$

Utilizando la data generada con el código anterior, calcule el valor de x^* así como el valor de la función objetivo f. ¿Por qué en la práctica no se utiliza este método? Justifique su respuesta.

- En la práctica no se tiende a utilizar este método, ya que la carga computacional asociada a obtener la inversa de una matriz crece exponencialmente a medida que las dimensiones de la misma incrementan. Si este problema se tratara de una regresión lineal con un millón de muestras, por ejemplo, la matriz A se tornaría en una matriz de (1000000×100), por lo que al producto $A^T A$ le tomaría mucho tiempo ser invertido. Esto se puede probar empíricamente al invertir una matriz cuadrada aleatoria un total de 10 veces, cada vez utilizando dimensiones cada vez mayores. Si se mide el tiempo de ejecución para estas 10

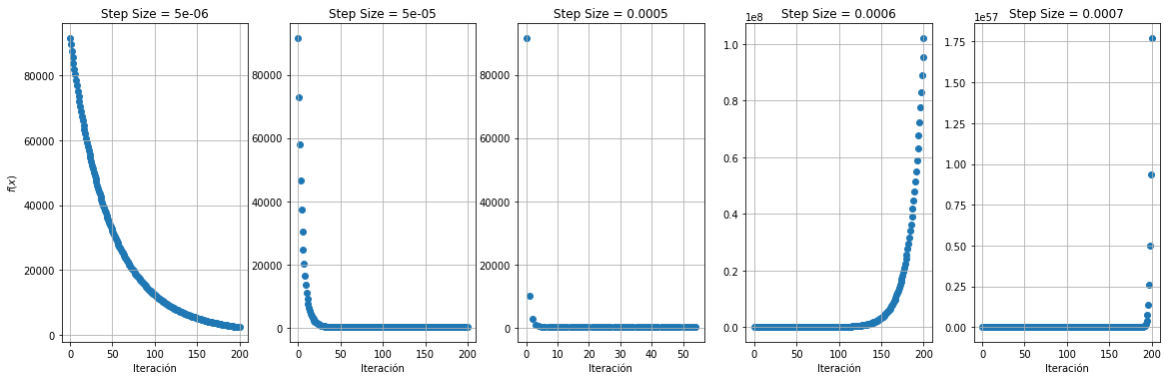
instrucciones, se podrá observar que el tiempo de ejecución incrementa exponencialmente.

```
Inv Matriz 1x1: 0.002221700007794425
Inv Matriz 10x10: 0.0025628999865148216
Inv Matriz 100x100: 0.1664452999830246
Inv Matriz 1000x1000: 3.0119156999862753
Inv Matriz 10000x10000: 373.6012941999943
```

Interfaz

Parte 2 – GD

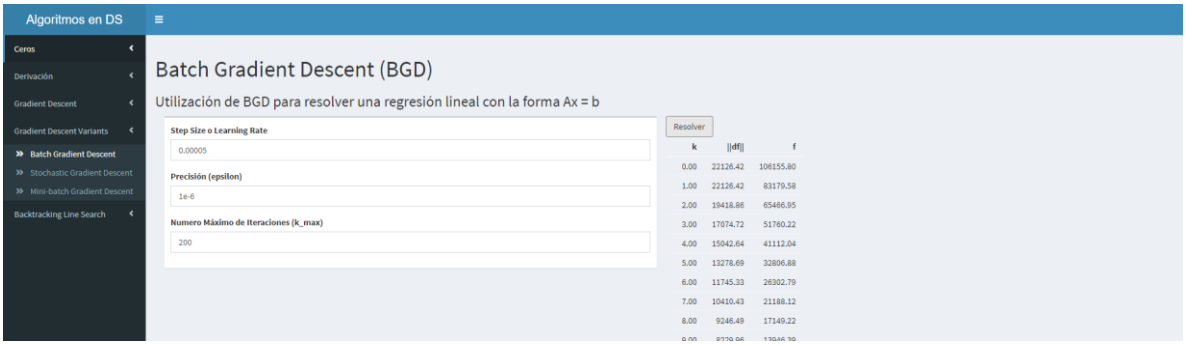
En esta parte se resolverá el problema planteado utilizando el algoritmo de gradient descent. Ejecute este algoritmo 3 veces, una para cada uno de los siguientes step sizes (o learning rates) constantes: 0.00005, 0.0005 y 0.0007. Inicialice el vector x con el vector nulo y seleccione un criterio adecuado para detener el algoritmo. Luego, realice una gráfica del valor de la función objetivo f del problema versus el índice de la iteración para cada uno de los 3 step sizes dados. Las tres gráficas deben estar en el mismo plano y mostrar los resultados de (al menos) las primeras 20 iteraciones. Finalmente, comente cómo el step size afecta la convergencia del algoritmo GD, puede experimentar con otros step sizes para justificar su respuesta. ¿Con cuál step size constante obtuvo el "mejor" resultado?



- Como se puede observar en las gráficas de arriba, el método de gradient descent es efectivo, pero únicamente bajo las condiciones adecuadas. El algoritmo se corrió con diferentes step sizes y este únicamente consiguió converger cuando se utilizaron valores muy específicos. Curiosamente, los step-sizes parecen contar con un "valor límite" a partir del cual ya no se debe incrementar o el método diverge. El método mejora su tasa de convergencia a medida que se incrementa el valor, pero una vez se alcanza un step-size de aproximadamente 0.0006, el método diverge. Por lo tanto, el mejor valor a emplear es el de 0.0005, el valor más grande posible antes de que el algoritmo comience a divergir.

  Tomando en cuenta todo lo anterior, se puede declarar que este método es útil, aunque probablemente sería adecuado primero obtener el step-size de forma exacta y seguido de esto emplearlo para ejecuciones posteriores. De esta manera ya no se debe pasar por una fase de experimentación previa para determinar bajo que situaciones el algoritmo diverge.

Interfaz Utilizada



Tablas de Resultados

Step Size = 0.00005

k	||df||	f
0.00	22126.42	106155.80
1.00	22126.42	83179.58
2.00	19418.86	65466.95
3.00	17074.72	51760.22
4.00	15042.64	41112.04
5.00	13278.69	32806.88
6.00	11745.33	26302.79
7.00	10410.43	21188.12
8.00	9246.49	17149.22
9.00	8229.96	13946.39
10.00	7340.64	11395.81
11.00	6561.23	9356.08
12.00	5876.88	7718.04
13.00	5274.85	6397.14
14.00	4744.19	5327.61
15.00	4275.50	4458.15
16.00	3860.69	3748.57
17.00	3492.79	3167.27
18.00	3165.81	2689.30
19.00	2874.58	2294.90
20.00	2614.64	1968.35

Step Size = 0.0005

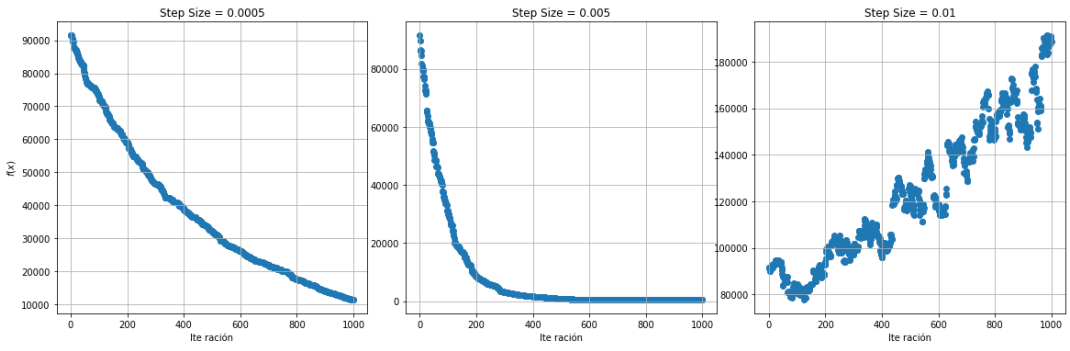
k	||df||	f
0.00	19271.86	83156.91
1.00	19271.86	11681.79
2.00	7726.66	2914.66
3.00	3935.74	1030.66
4.00	2239.51	505.25
5.00	1372.90	328.96
6.00	884.38	261.35
7.00	587.67	232.97
8.00	397.87	220.36
9.00	272.39	214.56
10.00	187.72	211.84
11.00	129.91	210.54
12.00	90.13	209.92
13.00	62.63	209.62
14.00	43.58	209.48
15.00	30.34	209.41
16.00	21.14	209.38
17.00	14.73	209.36
18.00	10.27	209.35
19.00	7.16	209.35
20.00	5.00	209.35

Step Size = 0.0007

k	||df||	f
0.00	23022.31	119306.62
1.00	23022.31	61937.28
2.00	18728.18	65807.32
3.00	19882.65	83053.52
4.00	22600.25	112810.23
5.00	26522.98	160250.63
6.00	31771.99	235373.13
7.00	38667.40	355659.98
8.00	47713.09	551727.49
9.00	59644.48	878034.49
10.00	75514.31	1433136.49
11.00	96820.10	2398310.84
12.00	125685.68	4111830.41
13.00	165117.43	7212804.46
14.00	219364.62	12921373.49
15.00	294426.77	23587091.76
16.00	398768.75	43766252.67
17.00	544330.10	82344730.44
18.00	747949.92	156731000.46
19.00	1033379.28	301153873.42
20.00	1434122.77	583107856.68

Parte 3 – SGD

En esta parte deberá implementar el algoritmo de SGD para resolver aproximadamente el problema. Ejecute este algoritmo 3 veces, una para cada uno de los siguientes step sizes (o learning rates) constantes: 0.0005, 0.005 y 0.01. Inicialice el vector x con el vector nulo y realice 1000 iteraciones para cada step size. Realice un gráfico de la función objetivo f versus el número de iteraciones para cada uno de los 3 step sizes dados. Las tres gráficas deben estar en el mismo plano. Finalmente, comente cómo el step size afecta la convergencia del SGD, puede experimentar con otros step sizes para justificar su respuesta. ¿Con cuál step size constante obtuvo el "mejor" resultado?



- Los resultados fueron similares a los observados en el caso del descenso gradiente simple: Mientras más grande es el step size, más rápida la convergencia. Sin embargo, existe un punto a partir del cual el algoritmo

diverge si se incrementa demasiado el step size. El "umbral de divergencia" en este caso parece consistir de un step size = 0.01. La única diferencia con respecto al descenso de gradiente tradicional es que el algoritmo converge más lento y de forma mucho más ruidosa debido a la introducción del elemento estocástico propio del algoritmo.

Interfaz Utilizada

Tablas de Resultados

Step Size = 0.0005

k	‖df‖	f
0.00	251.30	115449.35
1.00	46.78	115427.68
2.00	734.60	112868.95
3.00	151.03	112777.94
4.00	215.09	112560.91
5.00	116.51	112439.95
6.00	337.27	111975.69
7.00	91.52	111909.00
8.00	273.87	111411.87
9.00	0.22	111411.82
10.00	369.66	110891.25
11.00	186.23	110807.62
12.00	100.62	110826.30
13.00	145.13	110765.27
14.00	52.31	110742.37
15.00	109.23	110709.38
16.00	74.30	110674.56
17.00	306.62	110059.64
18.00	68.48	110019.20
19.00	625.28	108050.37
20.00	324.98	107715.96

Step Size = 0.005

k	‖df‖	f
0.00	145.19	107419.96
1.00	26.71	107345.56
2.00	5.11	107339.29
3.00	146.36	106885.98
4.00	92.06	106396.26
5.00	10.17	106356.07
6.00	3.26	106355.49
7.00	72.23	106643.58
8.00	77.77	105840.59
9.00	128.25	106775.42
10.00	95.71	106534.10
11.00	174.11	105443.21
12.00	47.72	105206.02
13.00	73.20	104588.71
14.00	290.22	101258.73
15.00	17.05	101352.93
16.00	127.92	100720.06
17.00	1.49	100716.80
18.00	248.16	99460.71
19.00	90.56	98631.08
20.00	27.29	98758.14

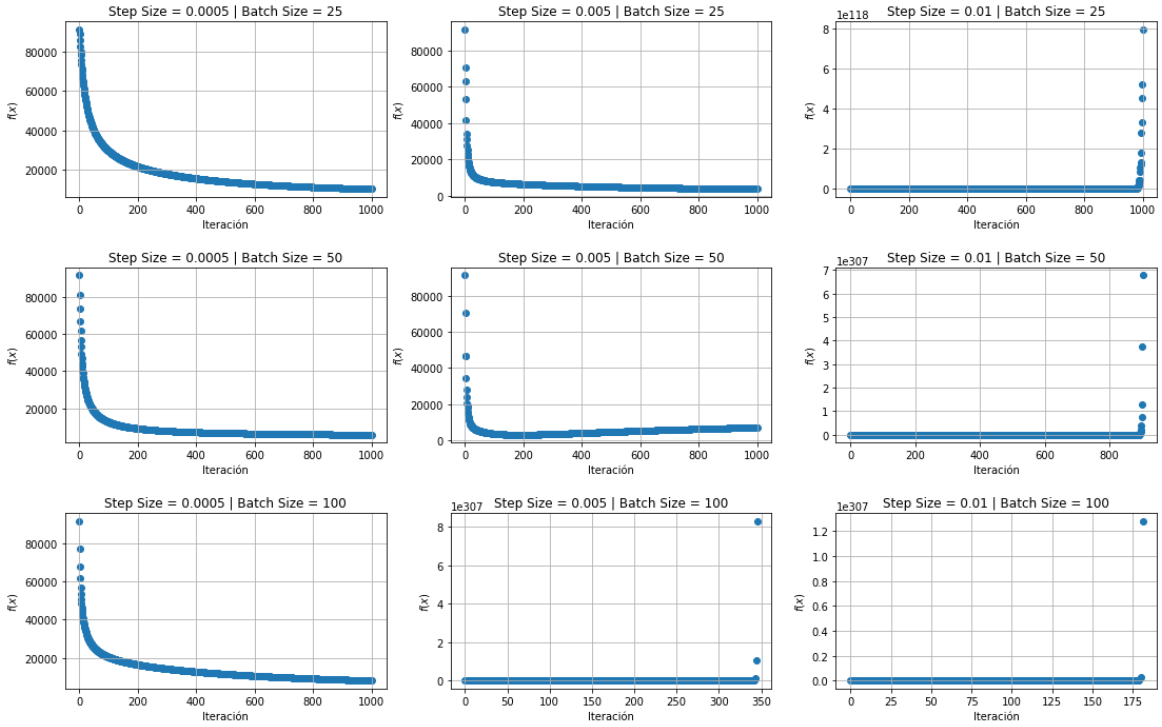
Step Size = 0.01

k	‖df‖	f
0.00	223.82	81107.71
1.00	57.86	80706.76
2.00	306.47	82171.35
3.00	49.17	81929.57
4.00	117.79	82386.25
5.00	257.04	80431.64
6.00	45.59	80549.61
7.00	16.51	80333.44
8.00	141.53	81189.07
9.00	57.47	80718.68
10.00	42.07	81128.68
11.00	22.49	81324.37
12.00	249.29	82733.03
13.00	85.94	82703.57
14.00	202.79	82974.18
15.00	78.17	82247.61
16.00	144.44	81107.59
17.00	30.25	81172.50
18.00	238.56	83611.29
19.00	151.15	83698.78
20.00	120.28	83655.23

Parte 4 – MBGD

Repita la parte 3, pero ahora implemente el algoritmo de MBGD para resolver aproximadamente el problema. Para ello, utilice batches de tamaño 25, 50 y 100, considerando para cada uno de estos casos los siguientes step sizes (o learning rates) constantes: 0.0005, 0.005 y 0.01. ¿Cómo el tamaño del batch afecta la

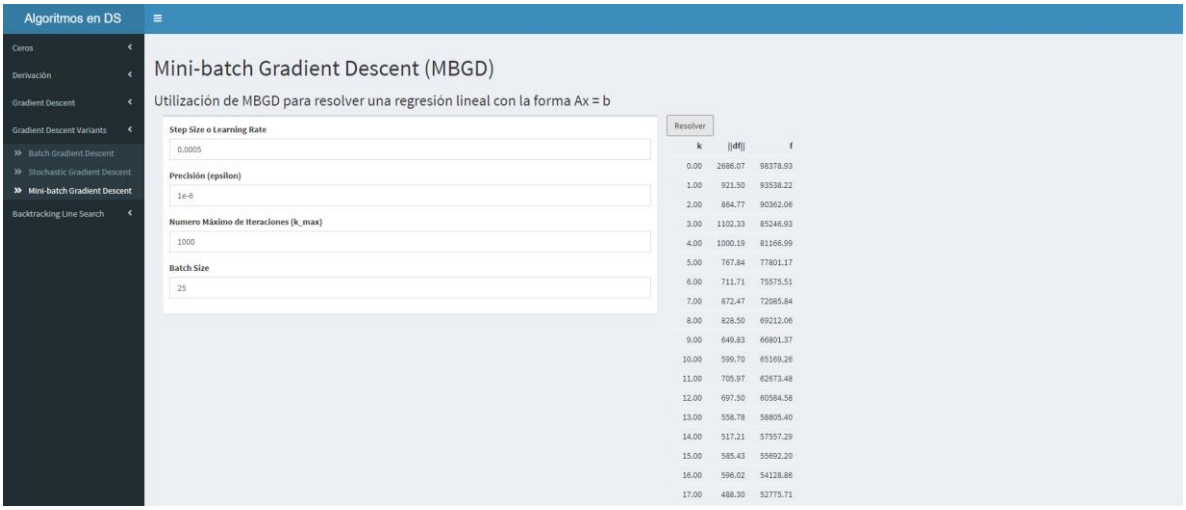
convergencia de este algoritmo? ¿Con cuál combinación de batch size y step size obtuvo el mejor resultado? Justifique su respuesta.



- Al igual que en los dos casos anteriores, el MBGD cuenta con un threshold superior a partir del cual el algoritmo comienza a diverger. En este caso particular, sin embargo, tanto la elección de step size, como de batch size influyen en la tasa de convergencia del mismo. Si tanto el batch size, como el step size se encuentran en un valor ideal, el algoritmo converge de forma súmamente rápida (por ejemplo, con la combinación de un step size = 0.005 y un batch size = 50).

  Sin embargo, si se incrementan demasiado los dos parámetros, el algoritmo comienza a diverger cada vez más rápido. A partir de las pruebas anteriores, se pudo llegar a determinar que un step size = 0.01 es demasiado agresivo, por lo que no importando el tamaño de batch, el algoritmo siempre diverge. Por lo tanto, esta variación de descenso de gradiente posee un desempeño comparable (si no mejor) que el del algoritmo tradicional, no obstante, se le agrega un parámetro adicional sobre el que se debe de tener control.

Interfaz Utilizada



k	‖df‖	f
0.00	2686.07	98378.93
1.00	921.50	93538.22
2.00	864.77	90362.06
3.00	1102.33	85246.93
4.00	1000.19	81166.99
5.00	767.84	77801.17
6.00	711.71	75575.51
7.00	872.47	72085.84
8.00	828.50	69212.06
9.00	649.83	66801.37
10.00	599.70	65169.26
11.00	705.97	62673.48
12.00	697.50	60584.58
13.00	558.78	58805.40
14.00	517.21	57557.29
15.00	585.43	55692.20
16.00	596.02	54128.86
17.00	488.30	52775.71

Tablas de Resultados

Step Size = 0.0005 | BS = 25

k	‖df‖	f
0.00	2686.07	98378.93
1.00	921.50	93538.22
2.00	864.77	90362.06
3.00	1102.33	85246.93
4.00	1000.19	81166.99
5.00	767.84	77801.17
6.00	711.71	75575.51
7.00	872.47	72085.84
8.00	828.50	69212.06
9.00	649.83	66801.37
10.00	599.70	65169.26
11.00	705.97	62673.48
12.00	697.50	60584.58
13.00	558.78	58805.40
14.00	517.21	57557.29
15.00	585.43	55692.20
16.00	596.02	54128.86
17.00	488.30	52775.71
18.00	455.82	51785.45
19.00	497.93	50337.17
20.00	516.29	49134.99

Step Size = 0.005 | BS = 25

k	‖df‖	f
0.00	3359.29	102578.47
1.00	1481.68	82122.94
2.00	1224.69	60223.40
3.00	970.94	48831.39
4.00	705.03	37690.49
5.00	753.08	31664.89
6.00	801.00	25433.21
7.00	435.43	20902.13
8.00	582.05	17673.23
9.00	252.29	16466.41
10.00	241.83	14941.85
11.00	278.49	13511.29
12.00	287.28	12265.02
13.00	173.05	11675.52
14.00	180.77	11219.64
15.00	172.83	11135.27
16.00	190.29	10343.06
17.00	127.30	9844.84
18.00	131.07	9323.75
19.00	117.43	8954.55
20.00	105.55	8486.95

Step Size = 0.01 | BS = 25

k	‖df‖	f
0.00	2492.95	76309.06
1.00	872.67	87678.25
2.00	793.39	96200.10
3.00	1030.71	105683.35
4.00	834.91	109433.32
5.00	1110.83	124781.11
6.00	1170.90	147129.21
7.00	994.05	169185.66
8.00	1660.84	257351.62
9.00	2150.23	351676.33
10.00	1806.68	378525.38
11.00	2587.19	582240.49
12.00	2772.42	741629.95
13.00	2774.35	926089.16
14.00	3706.88	1277026.18
15.00	3958.18	1446715.28
16.00	4061.58	1888596.11
17.00	4322.03	2170042.37
18.00	4626.16	2416790.92
19.00	5669.90	3091418.64
20.00	4623.70	3233533.51

Step Size = 0.0005 | BS = 50

k	‖df‖	f
0.00	2760.08	83680.39
1.00	1733.02	73254.65
2.00	1549.63	65442.37
3.00	1372.21	58587.45
4.00	1229.69	53485.20
5.00	1108.63	48789.40
6.00	996.11	45318.71
7.00	913.42	41972.64
8.00	823.94	39515.45
9.00	766.71	37041.53
10.00	695.44	35234.76
11.00	654.78	33342.88
12.00	598.07	31967.59
13.00	568.10	30476.40
14.00	522.97	29396.86
15.00	499.98	28189.79
16.00	463.94	27319.47
17.00	445.67	26319.58
18.00	416.63	25601.72
19.00	401.76	24756.90
20.00	378.01	24153.22

Step Size = 0.005 | BS = 50

k	‖df‖	f
0.00	2506.04	94881.67
1.00	1563.14	78021.77
2.00	1524.13	62761.40
3.00	1267.14	48782.26
4.00	1497.71	52146.84
5.00	1000.92	41527.80
6.00	1392.73	40803.59
7.00	945.31	32632.30
8.00	1137.39	35588.83
9.00	925.15	27924.94
10.00	902.00	23897.88
11.00	785.14	24083.70
12.00	793.23	23656.95
13.00	669.94	19625.49
14.00	708.99	17874.01
15.00	513.16	17288.15
16.00	615.37	17062.26
17.00	412.17	14271.01
18.00	469.29	13469.11
19.00	315.59	12572.30
20.00	351.81	12261.80

Step Size = 0.01 | BS = 50

k	‖df‖	f
0.00	3020.16	97384.27
1.00	1690.80	169635.91
2.00	1837.41	290348.45
3.00	2970.20	629253.47
4.00	3930.74	982720.57
5.00	4596.05	1592305.54
6.00	6845.89	3314079.70
7.00	9926.60	6658847.38
8.00	12455.32	11577226.65
9.00	16637.50	17779104.68
10.00	27844.89	54884009.50
11.00	31869.09	89857972.98
12.00	66736.86	357693762.39
13.00	82033.35	450736121.35
14.00	111356.06	1041396313.32
15.00	187863.62	2470294574.09
16.00	196232.54	2997469286.62
17.00	309769.20	6352083671.16
18.00	431774.14	15706861142.89
19.00	553340.25	24552829316.46
20.00	1029285.37	82435933673.36

Step Size = 0.0005 | BS = 100

k	‖df‖	f
0.00	2859.41	90403.18
1.00	2859.41	71767.52
2.00	2111.43	60168.55
3.00	1596.31	52406.51
4.00	1238.14	46851.92
5.00	986.06	42644.76
6.00	805.95	39311.68
7.00	674.86	36579.40
8.00	577.40	34281.84
9.00	503.25	32312.62
10.00	445.47	30600.05
11.00	399.39	29093.61
12.00	361.82	27756.20
13.00	330.59	26559.69
14.00	304.17	25482.20
15.00	281.50	24506.28
16.00	261.79	23617.84
17.00	244.48	22805.30
18.00	229.15	22059.03
19.00	215.46	21371.00
20.00	203.17	20734.39

Step Size = 0.005 | BS = 100

k	‖df‖	f
0.00	2876.00	85138.98
1.00	2876.00	102327.21
2.00	4625.50	298932.16
3.00	9774.84	1483697.22
4.00	23745.56	9373183.60
5.00	62843.28	68817280.66
6.00	174870.63	546182180.37
7.00	500782.56	4540955533.07
8.00	1458416.23	38791926525.56
9.00	4291075.58	337108111372.98
10.00	12708158.92	2962243640714.08
11.00	37797597.27	26225794482349.93
12.00	112747776.30	233392514056474.59
13.00	336995448.33	2084655845580295.25
14.00	1008678204.17	18669034691012116.00
15.00	3022155695.61	167509069431255424.00
16.00	9061369766.99	1505095948123844096.00
17.00	27182992740.66	13537619407376869376.00
18.00	81576589901.62	121859340706827354112.00
19.00	244880648653.49	1097565554721231077466.00
20.00	735244901787.28	9889988237488195371288.00

Step Size = 0.01 | BS = 100

k	‖df‖	f
0.00	2375.32	98714.92
1.00	2375.32	350164.00
2.00	9228.78	6780472.16
3.00	47551.20	194106927.58
4.00	271055.07	6573968919.52
5.00	1625571.96	242839610588.61
6.00	10020947.84	9401097156393.41
7.00	62745365.01	373502925785902.50
8.00	396531057.83	15063491952275164.00
9.00	2520537477.81	613119043886868736.00
10.00	16083388714.83	25104995551876046848.00
11.00	102902929251.42	1032227450008147853382.00
12.00	659681909532.38	42571073218340888512044.00
13.00	4235451756009.89	1759865594284709173504268.00
14.00	27226415493938.25	728909859546893321206060608.00
15.00	175191954684436.69	30238615569025734380820826886.00
16.00	1128245751890327.25	12561617580445910326668642466.00
17.00	7271245457917356.00	52245735338785677484882608686660.00
18.00	46891121141112240.00	217530460193034125062840240424462.00
19.00	302564806870746048.00	9065819521087342854406066662022024.00
20.00	1953292895086214144.00	378158604035526541086046044868820826.00

Parte 5 – Comparación

Compare el desempeño de cada uno de los métodos implementados. Realice una lista de posiciones ordenando los métodos de mejor a peor desempeño. Para ello, considere el valor óptimo de la función objetivo f^* alcanzado, el número de iteraciones requeridas y por supuesto, el error en el x^* obtenido versus el x real (x_{true}).

```python
# Se crean los diccionarios donde se alamacenarán datos
costs = {}
iters = {}
X_exp = {}
MSE = {}

# Se obtienen los costos más bajos obtenidos por cada método en su mejor configuración
costs["GD"] = runs_GD[0.0005].iloc[-1,:]["f"]
costs["SGD"] = runs_SGD[0.005].iloc[-1,:]["f"]
costs["MBGD"] = runs_MBGD["0.005, 25"].iloc[-1,:]["f"]

# Se obtiene el número de iteración para cada método en su mejor configuración
iters["GD"] = runs_GD[0.0005].iloc[-1,:]["k"]
iters["SGD"] = runs_SGD[0.005].iloc[-1,:]["k"]
iters["MBGD"] = runs_MBGD["0.005, 25"].iloc[-1,:]["k"]

# Se obtiene la última aproximación de X* para cada método
X_exp["GD"] = runs_GD[0.0005].iloc[-1,:]["Xk"]
X_exp["SGD"] = runs_SGD[0.005].iloc[-1,:]["Xk"]
X_exp["MBGD"] = runs_MBGD["0.005, 25"].iloc[-1,:]["Xk"]

# Se obtiene el MSE entre el X* obtenido y el X_true
MSE["GD"] = np.mean((X_exp["GD"] - X_true)**2)
MSE["SGD"] = np.mean((X_exp["SGD"] - X_true)**2)
MSE["MBGD"] = np.mean((X_exp["GD"] - X_true)**2)

# Se presentan los resultados en una tabla
df_comparacion = pd.DataFrame([iters, costs, MSE], ["Iters", "Cost", "MSE"])
df_comparacion
```

	GD	SGD	MBGD
Iters	54.000000	1000.000000	1000.000000
Cost	216.393813	443.417785	3781.285444
MSE	0.000273	0.002576	0.000273

Al comparar cada uno de los métodos empleados, se puede observar que, al menos de forma experimental, cada uno cuenta con sus ventajas y sus desventajas. El método de descenso de gradiente tradicional parece ser el mejor método, al obtener tanto el costo como el MSE más bajos y todo utilizando un número de iteraciones mucho menor. La desventaja del mismo, es que este carece de escalabilidad. Todas las ventajas anteriormente mencionadas provienen del hecho que el descenso de gradiente procesa todas las muestras de entrenamiento a la vez. Esto es sencillo para un "dataset" pequeño como el utilizado, pero en casos

donde los datos alcanzan los millones o billones (como comúnmente ocurre aplicaciones de data science), esto rápidamente se torna muy computacionalmente costoso.

Seguido de esto tenemos al descenso de gradiente estocástico. Este requiere de muchas más iteraciones (casi 950 más) a comparación del descenso de gradiente tradicional, pero este consigue alcanzar un valor de costo similar, utilizando una fracción de la carga computacional que utiliza el GD. La desventaja de este método es que toma más tiempo en computarse (cada muestra se procesa individualmente) y cuenta con una convergencia más "ruidosa" (debido a la elección aleatoria de muestras).

Finalmente se encuentra el mini batch gradient descent (MBGD). Este es un método interesante en términos de sus resultados. Dado que el mismo combina elementos de tanto GD como SGD, el mismo converge a una velocidad similar a la del SGD, pero alcanza una precisión similar a la del GD (denotado por sus MSE idénticos). Este emplea un poco más de capacidad computacional a comparación del SGD, pero nunca tanto como el GD y debido a que toma pequeños "batches" del dataset completo a la vez, su trayectoria de convergencia es mucho más suave. La única desventaja de este método es que su costo o valor de la función objetivo es peor al de ambos métodos, aunque esto probablemente se deba a un error en programación.

Tomando todo lo anterior en cuenta, y que estos algoritmos se utilizarán específicamente para "data science", se establece el siguiente "ranking" experimental (de mejor a peor):

1. Mini-batch gradient descent (MBGD)
2. Batch gradient descent (BGD)
3. Stochastic gradient descent (SGD)

El MBGD probó consistir del mejor método entre ambos, ya que ofrece el mejor tradeoff entre requisitos computacionales y precisión. Le sigue el BGD, ya que ningún otro algoritmo pudo llegar alcanzar la misma precisión con la misma rapidez. No se colocó en primero, debido a la seria limitante de términos de capacidad computacional requerida. Finalmente, se encuentra el SGD. Este no es malo y de hecho podría colocarse como un segundo "segundo lugar", pero este no solo converge más lento, sino que también lo hace de forma ruidosa, por lo que se arriesga a que se dificulte la convergencia sin razón aparente.

Problema 2 – Método de Newton

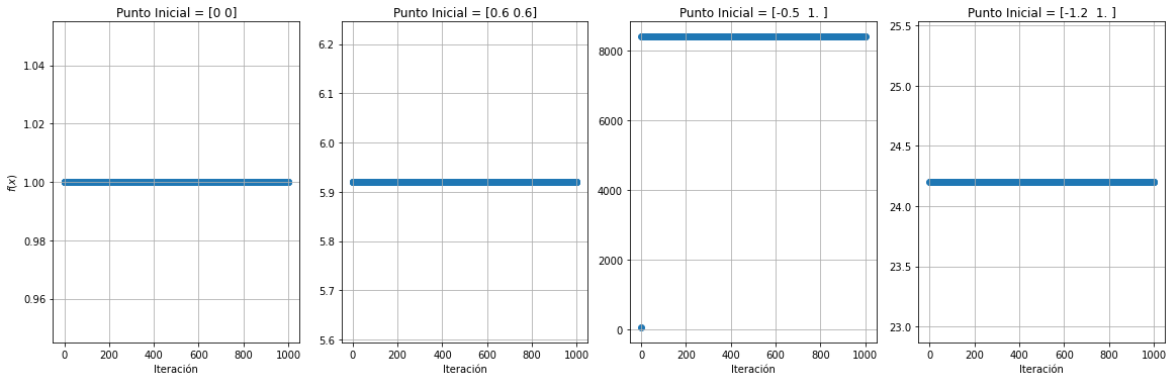
Considere de nuevo el problema de optimización

$$\min_{(x_1,x_2)\in\mathbb{R}^2} f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Recuerde que la función objetivo de este problema es conocida como Rosenbrock's Function y es utilizada como benchmark en la evaluación de algoritmo.

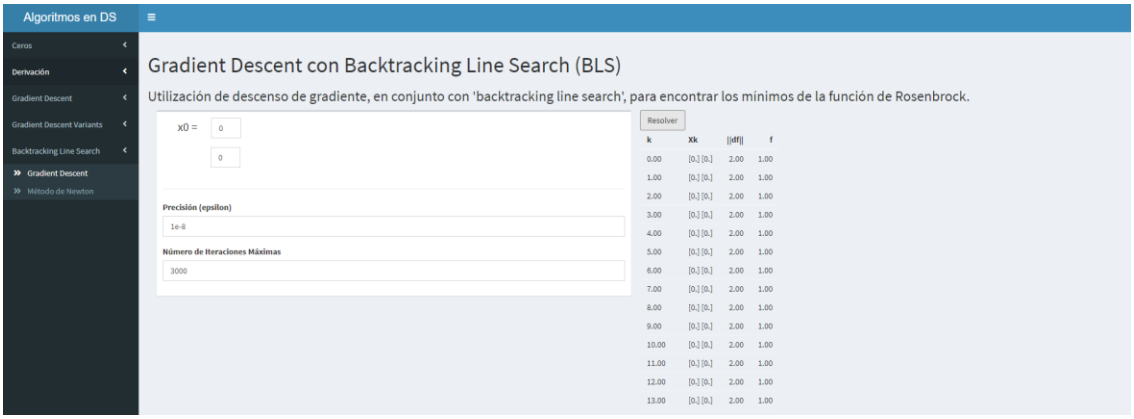
Parte 1 - GD con Backtracking Line Search

En el laboratorio anterior, se aplicó el método de GD (con un step size constante) para resolver el problema. Ahora, utilice "Backtracking Line Search" para determinar el step size. Fije los parámetros de este algoritmo de acuerdo a los valores dados en clase. Compare los resultados de ambas formas de determinar el step size, para ello utilice $x_0 = (0,0)^T$, $(0.6,0.6)^T$, $(-0.5,1)^T$ y $(-1.2,1)^T$.



- Las gráficas generadas parecen indicar que el algoritmo de BLS no está funcionando correctamente. No obstante, al investigar un poco más sobre las operaciones internas del algoritmo, se pudo llegar a observar que el mismo si está en funcionamiento. Sin embargo, debido a que este se basa en reducir el step size hasta eventualmente cumplir con las condiciones de Armijo y las mismas no se cumplen durante un gran número de iteraciones, el valor del step size termina tomando un valor en el orden de 1x10^-17. Esto implica que cada cambio al vector inicial x_k es virtualmente nulo y por lo tanto, el mismo permanece constante a lo largo de casi todas las iteraciones. Esto es un indicador que, para métodos que no incluyen el valor de la segunda derivada en la formulación de su modelo, no es apta la inclusión del BLS. La utilización de un valor constante o de algún otro método basado en momentos, por ejemplo, resulta en la mejor opción.

Interfaz



Tablas de Datos

Punto Inicial = (0,0)

k	Xk	||df||	f
0.00	[0.] [0.]	2.00	1.00
1.00	[0.] [0.]	2.00	1.00
2.00	[0.] [0.]	2.00	1.00
3.00	[0.] [0.]	2.00	1.00
4.00	[0.] [0.]	2.00	1.00
5.00	[0.] [0.]	2.00	1.00
6.00	[0.] [0.]	2.00	1.00
7.00	[0.] [0.]	2.00	1.00
8.00	[0.] [0.]	2.00	1.00
9.00	[0.] [0.]	2.00	1.00
10.00	[0.] [0.]	2.00	1.00
11.00	[0.] [0.]	2.00	1.00
12.00	[0.] [0.]	2.00	1.00
13.00	[0.] [0.]	2.00	1.00
14.00	[0.] [0.]	2.00	1.00
15.00	[0.] [0.]	2.00	1.00
16.00	[0.] [0.]	2.00	1.00
17.00	[0.] [0.]	2.00	1.00
18.00	[0.] [0.]	2.00	1.00
19.00	[0.] [0.]	2.00	1.00
20.00	[0.] [0.]	2.00	1.00

Punto Inicial = (0.6,0.6)

k	Xk	||df||	f
0.00	[0.6] [0.6]	75.59	5.92
1.00	[0.6] [0.6]	75.59	5.92
2.00	[0.6] [0.6]	75.59	5.92
3.00	[0.6] [0.6]	75.59	5.92
4.00	[0.6] [0.6]	75.59	5.92
5.00	[0.6] [0.6]	75.59	5.92
6.00	[0.6] [0.6]	75.59	5.92
7.00	[0.6] [0.6]	75.59	5.92
8.00	[0.6] [0.6]	75.59	5.92
9.00	[0.6] [0.6]	75.59	5.92
10.00	[0.6] [0.6]	75.59	5.92
11.00	[0.6] [0.6]	75.59	5.92
12.00	[0.6] [0.6]	75.59	5.92
13.00	[0.6] [0.6]	75.59	5.92
14.00	[0.6] [0.6]	75.59	5.92
15.00	[0.6] [0.6]	75.59	5.92
16.00	[0.6] [0.6]	75.59	5.92
17.00	[0.6] [0.6]	75.59	5.92
18.00	[0.6] [0.6]	75.59	5.92
19.00	[0.6] [0.6]	75.59	5.92
20.00	[0.6] [0.6]	75.59	5.92

Punto Inicial = (-0.5, 1)

k	Xk	||df||	f
0.00	[-0.5] [1.]	210.02	58.50
1.00	[-2.7969] [-1.3438]	210.02	8416.45
2.00	[-2.7969] [-1.3438]	10424.81	8416.45
3.00	[-2.7969] [-1.3438]	10424.81	8416.45
4.00	[-2.7969] [-1.3438]	10424.81	8416.45
5.00	[-2.7969] [-1.3438]	10424.81	8416.45
6.00	[-2.7969] [-1.3438]	10424.81	8416.45
7.00	[-2.7969] [-1.3438]	10424.81	8416.45
8.00	[-2.7969] [-1.3438]	10424.81	8416.45
9.00	[-2.7969] [-1.3438]	10424.81	8416.45
10.00	[-2.7969] [-1.3438]	10424.81	8416.45
11.00	[-2.7969] [-1.3438]	10424.81	8416.45
12.00	[-2.7969] [-1.3438]	10424.81	8416.45
13.00	[-2.7969] [-1.3438]	10424.81	8416.45
14.00	[-2.7969] [-1.3438]	10424.81	8416.45
15.00	[-2.7969] [-1.3438]	10424.81	8416.45
16.00	[-2.7969] [-1.3438]	10424.81	8416.45
17.00	[-2.7969] [-1.3438]	10424.81	8416.45
18.00	[-2.7969] [-1.3438]	10424.81	8416.45
19.00	[-2.7969] [-1.3438]	10424.81	8416.45
20.00	[-2.7969] [-1.3438]	10424.81	8416.45

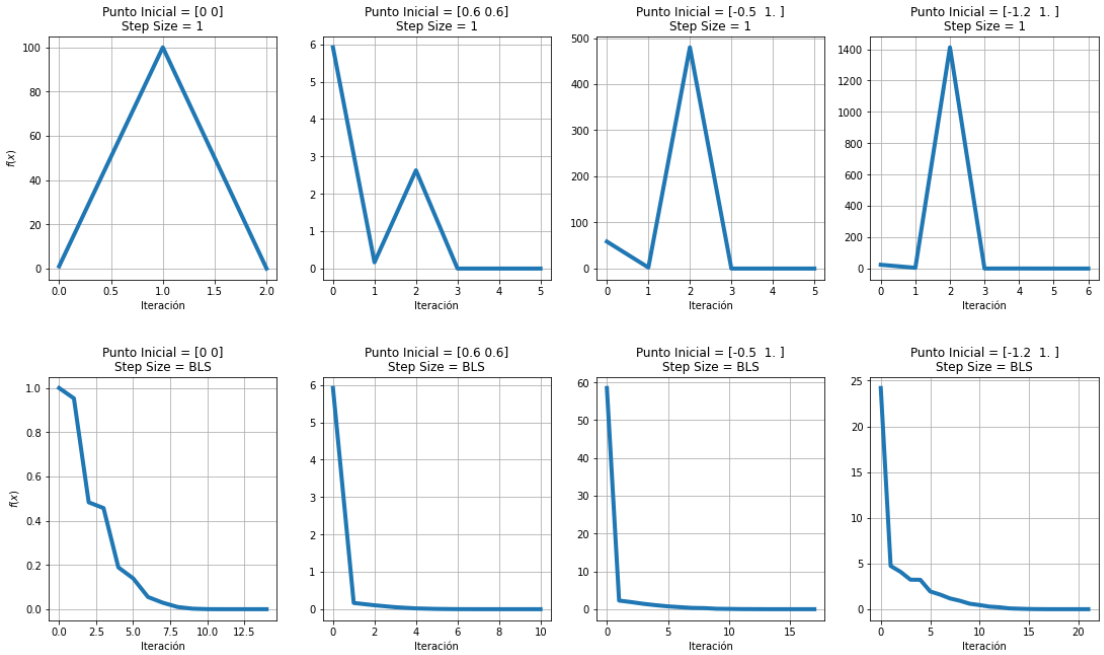
Punto Inicial = (-1.2, 1)

k	Xk	||df||	f
0.00	[-1.2] [1.]	232.87	24.20
1.00	[-1.2] [1.]	232.87	24.20
2.00	[-1.2] [1.]	232.87	24.20
3.00	[-1.2] [1.]	232.87	24.20
4.00	[-1.2] [1.]	232.87	24.20
5.00	[-1.2] [1.]	232.87	24.20
6.00	[-1.2] [1.]	232.87	24.20
7.00	[-1.2] [1.]	232.87	24.20
8.00	[-1.2] [1.]	232.87	24.20
9.00	[-1.2] [1.]	232.87	24.20
10.00	[-1.2] [1.]	232.87	24.20
11.00	[-1.2] [1.]	232.87	24.20
12.00	[-1.2] [1.]	232.87	24.20
13.00	[-1.2] [1.]	232.87	24.20
14.00	[-1.2] [1.]	232.87	24.20
15.00	[-1.2] [1.]	232.87	24.20
16.00	[-1.2] [1.]	232.87	24.20
17.00	[-1.2] [1.]	232.87	24.20
18.00	[-1.2] [1.]	232.87	24.20
19.00	[-1.2] [1.]	232.87	24.20
20.00	[-1.2] [1.]	232.87	24.20

Parte 2 – Método de Newton con Backtracking Line Search

Aplique el método de Newton para resolver este problema de optimización. Al igual que en el problema, utilice $x_0 = (0,0)^T$ y un step-size unitario. ¿Qué sucede si varía el punto inicial a $(0.6,0.6)^T$, $(-0.5,1)^T$ y $(-1.2,1)^T$ y si el step size lo calcula utilizando "Back tracking Line Search"? Experimente con estos parámetros y reporte sus hallazgos. Detenga la ejecución del algoritmo cuando $\|\nabla f(x_k)\| < 10^{-8}$ o bien cuando el número de iteraciones exceda 3000. Su output debe ser mostrado en una tabla con las cuatro columnas siguientes:

1. El número k de la iteración
2. x_k
3. La dirección p_k
4. $||\nabla f(x_k)||$

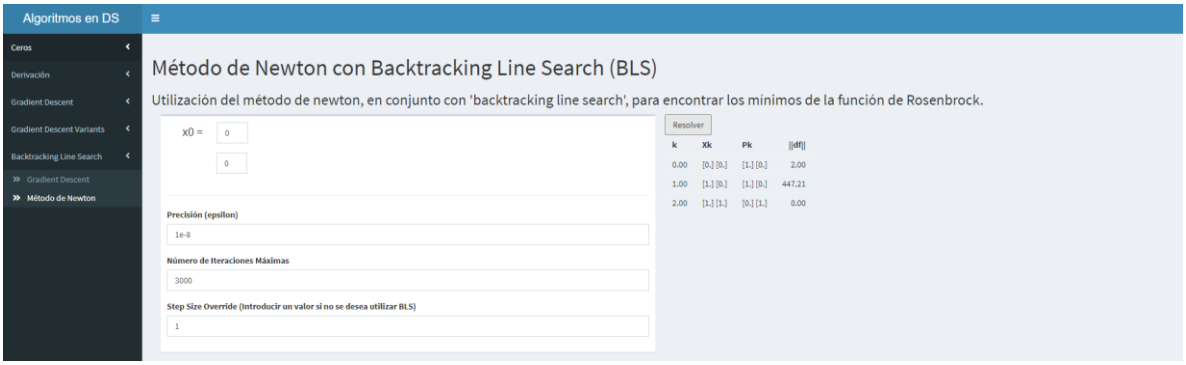


- Al emplear un step size unitario, todos los algoritmos convergieron en una cantidad increíblemente pequeña de iteraciones (6 o menos), esto a pesar de que se le haya dado al algoritmo un máximo de 3000 iteraciones y que su punto inicial se haya variado. Esto demuestra la gran capacidad de los modelos que incluyen la curvatura de la función objetivo dentro de su formulación (como es el caso del método de Newton). A pesar de esta ventaja significativa, es importante mencionar que el valor de la función objetivo fluctúa agresivamente mientras es minimizada, aunque el método de Newton es capaz de autocorregirse para regresar al punto en el que se encontraba.

  Al emplear un step size encontrado a través de back tracking line search, el algoritmo tendió a tomar un poco más de tiempo en converger (entre 10 a 20 iteraciones), pero presentó una trayectoria un poco más "limpia" en términos de la minimización de su función objetivo. Otra característica particular de las curvas de minimización al utilizar BLS, es que, en la mayoría de las posiciones iniciales, el algoritmo fue capaz de llegar a un valor "suficientemente bueno" de su función objetivo de forma casi inmediata y seguido de esto, únicamente se reguló para continuar minimizando hacia el mínimo buscado. Por lo tanto, al acoplarle BLS al método de Newton, este se torna ligeramente más lento en términos de su convergencia, pero el step size deja de consistir de un hiperparámetro a calibrar por parte del usuario.

Para determinar cuál de los dos métodos es superior, se deben de tomar en cuenta las preferencias del experimentador: Se valora más un tiempo de convergencia bajo, pero que requiere de mayor experimentación (Step Size Constante) o la conveniencia de no necesitar que experimentar con un parámetro adicional y que devuelve resultados casi iguales a los anteriores (BLS).

Interfaz Utilizada



Tablas de Datos

Punto Inicial = (0,0)
Step Size = 1

k	Xk	Pk	||df||
0.00	[0.] [0.]	[1.] [0.]	2.00
1.00	[1.] [0.]	[1.] [0.]	447.21
2.00	[1.] [1.]	[0.] [1.]	0.00

Punto Inicial = (0.6,0.6)
Step Size = 1

k	Xk	Pk	||df||
0.00	[0.6] [0.6]	[-0.0085] [-0.2502]	75.59
1.00	[0.5915] [0.3498]	[-0.0085] [-0.2502]	0.80
2.00	[0.9942] [0.8262]	[0.4027] [0.4764]	72.17
3.00	[0.9943] [0.9887]	[0.0002] [0.1625]	0.01
4.00	[1.] [1.]	[0.0057] [0.0113]	0.01
5.00	[1.] [1.]	[0.] [0.]	0.00

Punto Inicial = (-0.5, 1)
Step Size = 1

k	Xk	Pk	||df||
0.00	[-0.5] [1.]	[-0.0101] [-0.7399]	210.02
1.00	[-0.5101] [0.2601]	[-0.0101] [-0.7399]	3.04
2.00	[0.97] [-1.2497]	[1.4801] [-1.5098]	956.17
3.00	[0.9701] [0.941]	[0.0001] [2.1907]	0.06
4.00	[1.] [0.9991]	[0.0299] [0.0581]	0.40
5.00	[1.] [1.]	[0.] [0.0009]	0.00

Punto Inicial = (-1.2, 1)
Step Size = 1

k	Xk	Pk	||df||
0.00	[-1.2] [1.]	[0.0247] [0.3807]	232.87
1.00	[-1.1753] [1.3807]	[0.0247] [0.3807]	4.64
2.00	[0.7631] [-3.175]	[1.9384] [-4.5557]	1370.79
3.00	[0.7634] [0.5828]	[0.0003] [3.7579]	0.47
4.00	[1.] [0.944]	[0.2366] [0.3612]	25.03
5.00	[1.] [1.]	[0.] [0.056]	0.00
6.00	[1.] [1.]	[0.] [0.]	0.00

Punto Inicial = (0,0)
Step Size = 1

k	Xk	Pk	||df||
0.00	[0.] [0.]	[1.] [0.]	2.00
1.00	[0.25] [0.]	[1.] [0.]	13.37
2.00	[0.3056] [0.0903]	[0.0556] [0.0903]	1.19
3.00	[0.5203] [0.223]	[0.4294] [0.2655]	13.07
4.00	[0.5658] [0.3181]	[0.0456] [0.0951]	0.58
5.00	[0.7192] [0.4927]	[0.3068] [0.3492]	8.15
6.00	[0.7667] [0.5856]	[0.0475] [0.0929]	0.50
7.00	[0.8471] [0.71]	[0.1608] [0.2488]	2.73
8.00	[0.9078] [0.8204]	[0.0607] [0.1105]	1.37
9.00	[0.9609] [0.9205]	[0.0531] [0.1]	1.15
10.00	[0.9859] [0.9714]	[0.025] [0.0509]	0.25
11.00	[0.9984] [0.9967]	[0.0125] [0.0253]	0.07
12.00	[1.] [0.9999]	[0.0015] [0.0032]	0.00
13.00	[1.] [1.]	[0.] [0.0001]	0.00
14.00	[1.] [1.]	[0.] [0.]	0.00

Punto Inicial = (0.6,0.6)
Step Size = 1

k	Xk	Pk	||df||
0.00	[0.6] [0.6]	[-0.0085] [-0.2502]	75.59
1.00	[0.5915] [0.3498]	[-0.0085] [-0.2502]	0.80
2.00	[0.6922] [0.4689]	[0.4027] [0.4764]	3.00
3.00	[0.7935] [0.6194]	[0.1013] [0.1505]	3.51
4.00	[0.8611] [0.7369]	[0.0676] [0.1176]	1.59
5.00	[0.9337] [0.8665]	[0.0725] [0.1295]	2.11
6.00	[0.966] [0.9321]	[0.0323] [0.0656]	0.40
7.00	[0.9941] [0.9875]	[0.0281] [0.0554]	0.34
8.00	[0.9992] [0.9984]	[0.0051] [0.0109]	0.01
9.00	[1.] [1.]	[0.0008] [0.0016]	0.00
10.00	[1.] [1.]	[0.] [0.]	0.00

Punto Inicial = (-0.5, 1)
Step Size = 1

k	Xk	Pk	||df||
0.00	[-0.5] [1.]	[-0.0101] [-0.7399]	210.02
1.00	[-0.5101] [0.2601]	[-0.0101] [-0.7399]	3.04
2.00	[-0.3251] [0.0713]	[1.4801] [-1.5098]	9.88
3.00	[-0.1565] [-0.0039]	[0.1685] [-0.0752]	7.00
4.00	[0.0166] [-0.0297]	[0.1732] [-0.0258]	6.25
5.00	[0.1572] [0.0049]	[0.1406] [0.0347]	3.98
6.00	[0.3274] [0.0782]	[0.1702] [0.0733]	6.29
7.00	[0.4264] [0.172]	[0.099] [0.0938]	2.03
8.00	[0.6202] [0.3471]	[0.1939] [0.1751]	11.40
9.00	[0.6648] [0.44]	[0.0446] [0.0929]	0.42
10.00	[0.7847] [0.6004]	[0.2398] [0.3209]	5.36
11.00	[0.8376] [0.6987]	[0.0528] [0.0983]	0.83
12.00	[0.9418] [0.8761]	[0.1042] [0.1774]	4.53
13.00	[0.9601] [0.9215]	[0.0183] [0.0454]	0.08
14.00	[0.9975] [0.9936]	[0.0373] [0.072]	0.62
15.00	[0.9995] [0.9989]	[0.002] [0.0053]	0.00
16.00	[1.] [1.]	[0.0005] [0.0011]	0.00
17.00	[1.] [1.]	[0.] [0.]	0.00

Punto Inicial = (-1.2, 1)
Step Size = 1

k	Xk	Pk	||df||
0.00	[-1.2] [1.]	[0.0247] [0.3807]	232.87
1.00	[-1.1753] [1.3807]	[0.0247] [0.3807]	4.64
2.00	[-0.933] [0.8112]	[1.9384] [-4.5557]	28.55
3.00	[-0.7825] [0.5897]	[0.1504] [-0.2215]	11.57
4.00	[-0.46] [0.1076]	[0.3225] [-0.4822]	30.33
5.00	[-0.393] [0.15]	[0.067] [0.0424]	3.60
6.00	[-0.2094] [0.0068]	[0.7345] [-0.5729]	9.25
7.00	[-0.0657] [-0.0163]	[0.1437] [-0.0231]	4.92
8.00	[0.142] [-0.023]	[0.2078] [-0.0067]	8.66
9.00	[0.2311] [0.0455]	[0.0891] [0.0685]	1.78
10.00	[0.3797] [0.1181]	[0.2973] [0.1453]	5.88
11.00	[0.4796] [0.22]	[0.0999] [0.1019]	2.18
12.00	[0.6534] [0.3967]	[0.1738] [0.1767]	9.40
13.00	[0.7026] [0.4913]	[0.0492] [0.0945]	0.49
14.00	[0.8028] [0.6332]	[0.2003] [0.2839]	3.92
15.00	[0.8635] [0.7419]	[0.0607] [0.1087]	1.24
16.00	[0.9421] [0.8813]	[0.0786] [0.1394]	2.53
17.00	[0.968] [0.9363]	[0.0259] [0.055]	0.24
18.00	[0.9962] [0.9916]	[0.0282] [0.0553]	0.35
19.00	[0.9995] [0.9989]	[0.0033] [0.0073]	0.00
20.00	[1.] [1.]	[0.0005] [0.001]	0.00
21.00	[1.] [1.]	[0.] [0.]	0.00