PS4

1) (a) $f(x_1, x_2) = x_1^2 + x_2^2$

(a) Global minimum of f.

To find critical points, we equate derivatives to 0.

$$\frac{d}{dx_1} f(x_1, x_2) = f_{x_1}(x_1, x_2) = 2x_1$$

$$\frac{d}{dx_2} f(x_1, x_2) = f_{x_2}(x_1, x_2) = 2x_2$$

$$f_{x_1} = 0 \Rightarrow x_1 = 0 \qquad f_{x_2} = 0 \Rightarrow x_2 = 0$$

Critical point $(x_{1c}, x_{2c}) = (0, 0)$

To check if this is global minimum, we need to compute $D(x_{1c}, x_{2c})$ where,

$$D(x_{1c}, x_{2c}) = f_{x_1 x_1}(x_{1c}, x_{2c}) f_{x_2 x_2}(x_{1c}, x_{2c}) - \left[f_{x_1 x_2}(x_{1c}, x_{2c})\right]^2$$

$= 0$

$$f_{x_1 x_1} = 2 \; ; \; f_{x_2 x_2} = 2$$

$D(x_{1c}, x_{2c}) = 2(2) = 4$

As $D > 0$ and $f_{x_1x_1}(x_{1c}, x_{2c}) > 0$,

→ we have the global minimum of f at $(0,0)$.

The global minimum is 0 $f(0,0) = 0+0=0$

(b) what is the gradient, ∇f?

$$f(x_1, x_2) = x_1^2 + x_2^2$$

$$\frac{\partial f}{\partial x_1} = 2x_1 \qquad \frac{\partial f}{\partial x_2} = 2x_2$$

$$\nabla f = [2x_1, 2x_2]$$

(c) Gradient descent iteration:

$$(x_1^{k+1}, x_2^{k+1}) = (x_1^k, x_2^k) - \alpha \nabla f(x_1^k, x_2^k)$$

$$(x_1^{0+1}, x_2^{0+1}) = (3, 0.25) - 0.2\,(2x_1^0, 2x_2^0)$$

$$(x_1^1, x_2^1) = (3, 0.25) - 0.2(6, 0.5) = (1.8, 0.15)$$

(d) Values for (x_1^5, x_2^5) and (x_1^{10}, x_2^{10})

$$(x_1^5, x_2^5) = (0.2333, 0.0194)$$

$$(x_1^{10}, x_2^{10}) = (0.0181, 0.0015)$$

Check the python script for the process and outputs.

2) $f(x_1, x_2) = \frac{x_1^2}{16} + 4x_2^2$

(a)

$$f_{x_1}(x_1, x_2) = \frac{2x_1}{16} = \frac{x_1}{8}$$

$$f_{x_2}(x_1, x_2) = 4(2x_2) = 8x_2$$

For critical point, equate differentiation to zero ($f_{x_1} = 0$ & $f_{x_2} = 0$)

$\Rightarrow f_{x_1} = 0 \Rightarrow x_1 = 0$

$f_{x_2} = 0 \Rightarrow x_2 = 0$

$\Rightarrow$ Critical point = (0, 0)

$$D(x_{1c}, x_{2c}) = f_{x_1x_1} f_{x_2x_2} - [f_{x_1x_2}]^2$$

$$D = \left(\frac{1}{8}\right)(8) = 1$$

$\Rightarrow D = 1$ and $f_{x_1x_1}(x_{1c}, x_{2c}) = 1/8$

as $D > 0$ and $f_{x_1x_1} > 0$

$\Rightarrow$ The global minimum of f is at $(0,0)$.
and the value of global minimum $= 0$.

$$f(0,0) = 0$$

(b) Gradient, ∇f

$$f(x_1, x_2) = \frac{x_1^2}{16} + 4x_2^2$$

$$\frac{df}{dx_1} = \frac{x_1}{8} \qquad \frac{df}{dx_2} = 8x_2$$

Gradient $\nabla f = [x_1/8, \; 8x_2]$

(c) Value of:

$(x_1^1, x_2^1) = (2.925, -0.15)$

$(x_1^5, x_2^5) = (2.6433, -0.0194)$

$(x_1^{10}, x_2^{10}) = (2.3291, 0.0015)$

Check the python script for the process and outputs.

(d) Of the two problems, the first problem's gradient descent converges more rapidly than the second one to the global minimum. The minimum value is reached by the problem 1 in 20 iterations while problem 2 reaches the minimum at 286 iterations. The convergence rate of the second problem takes small steps in the descent and needs higher number of iterations to converge.

In comparision, the convergence rate of the first problem is more optimal. The '$x_t/8$' term in the gradient descent of problem 2 is reducing the stepsize and it is the reason for such a low convergence rate.

3)

In case of problem 1,

learning rate	Iterations
$\eta = 0.2$	1460
$\eta = 0.3$	816
$\eta = 0.4$	465
$\eta = 0.5$	2

As we increase the learning rate, the convergence is faster in problem 1.

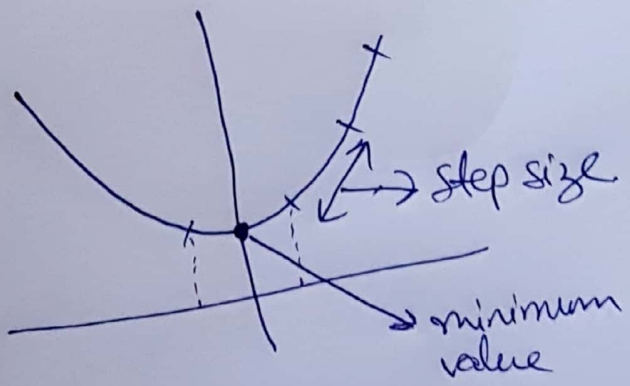
As learning rate increases, the step size increases and it reaches convergence at a faster pace.

In case of problem 2,

I have computed values upto 500 iterations and compared the convergence rates. I have used $\eta = 0.2, 0.3, 0.4$ and 0.5. With increase in the learning rate the rate of convergence is decreasing. There is no converge in problem 2 with learning rates of $\eta \geq 0.2$. This is due to the step size going beyond and skipping the minimum point.

I have attached the python scripts of both the problems to demonstrate the results.

The reason for non-convergence is because the step size being large, it passes over the minimum value and doesn't stop at that value.



4)

(c)

h	Point	Finite Diff.	Analytical
0.1	(1,2)	(2.1, 4.1)	(2, 4)
	(3,4)	(6.1, 8.1)	(6, 8)
	(5,6)	(10.1, 12.1)	(10, 12)
0.01	(1,2)	(2.01, 4.01)	(2, 4)
	(3,4)	(6.01, 8.01)	(6, 8)
	(5,6)	(10.01, 12.01)	(10, 12)

(d) Consider the method of finite differences,

For 2 dimensions

$$\frac{\partial f}{\partial x_1} = \frac{f(x_1+h, x_2) - f(x_1, x_2)}{h}$$

$$\frac{\partial f}{\partial x_2} = \frac{f(x_1, x_2+h) - f(x_1, x_2)}{h}$$

Extending to n dimensions

$$\frac{\partial f}{\partial x_n} = \frac{f(x_1, x_2, \ldots x_n+h) - f(x_1, x_2, \ldots x_n)}{h}$$

Each time, for 1 dimension, the function is called twice.
So, if we consider 'd' dimensions, we get '2d'.
Here, the function refers to the loss-function $f(x_1, \ldots x_n)$.

In [1]:
```
import numpy as np
```

In [4]:
```
#########################  Problem 1    ########################################

x_1=3
x_2=0.25

(A,B)=(x_1,x_2)

# Learning rate
LR=0.2
print("5 iterations")
for i in range(5):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    (A,B)=(round(A,4),round(B,4))
    print(A,B)
```

```
5 iterations
1.7999999999999998 0.15
1.0799999999999998 0.09
0.6479999999999999 0.054
0.3887999999999999 0.0324
0.23327999999999993 0.01944
```

In [3]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)

# Learning rate
LR=0.2
print("10 iterations")
for i in range(10):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    (A,B)=(round(A,4),round(B,4))
    print(A,B)
```

```
10 iterations
1.8 0.15
1.08 0.09
0.648 0.054
0.3888 0.0324
0.2333 0.0194
0.14 0.0116
0.084 0.007
0.0504 0.0042
0.0302 0.0025
0.0181 0.0015
```

In [1]:
```
#########################  Problem 2    ########################################

import numpy as np

x_1=3
x_2=0.25

(A,B)=(x_1,x_2)

# Learning rate
LR=0.2
print("1 iterations")
for i in range(1):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    (A,B)=(round(A,4),round(B,4))
    print(A,B)
```

```
1 iterations
2.925 -0.15
```

In [2]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)

# Learning rate
LR=0.2
print("5 iterations")
for i in range(5):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    (A,B)=(round(A,4),round(B,4))
    print(A,B)
```

```
5 iterations
2.925 -0.15
2.8519 0.09
2.7806 -0.054
2.7111 0.0324
2.6433 -0.0194
```

In [3]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)

# Learning rate
LR=0.2
print("10 iterations")
for i in range(10):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    (A,B)=(round(A,4),round(B,4))
    print(A,B)
```

```
10 iterations
2.925 -0.15
2.8519 0.09
2.7806 -0.054
2.7111 0.0324
2.6433 -0.0194
2.5772 0.0116
2.5128 -0.007
2.45 0.0042
2.3888 -0.0025
2.3291 0.0015
```

In [1]:
```
import numpy as np
```

In [71]:
```
#########################  Problem 1    ########################################

x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 1")
# Learning rate
print("Learning rate = 0.2")
LR=0.2
for i in range(1460):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    print(A,B)
```

```
3.7257e-320 3.103e-321
2.2356e-320 1.863e-321
1.3414e-320 1.117e-321
8.05e-321 6.7e-322
4.827e-321 4.05e-322
2.895e-321 2.4e-322
1.74e-321 1.43e-322
1.042e-321 8.4e-323
6.27e-322 5e-323
3.75e-322 3e-323
2.27e-322 2e-323
1.4e-322 1e-323
8.4e-323 5e-324
5e-323 5e-324
3e-323 5e-324
2e-323 5e-324
1e-323 5e-324
5e-324 5e-324
5e-324 5e-324
```

In [72]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 1")
# Learning rate
print("Learning rate = 0.3")
LR=0.3
for i in range(816):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    print(A,B)
```

```
1.1033534669133574e-302 9.194612224277986e-304
4.41341386765343e-303 3.6778448897111946e-304
1.765365547061372e-303 1.471137955884478e-304
7.0614621882454886e-304 5.884551823537912e-305
2.8245848752981954e-304 2.353820729415165e-305
1.1298339501192783e-304 9.41528291766066e-306
4.5193358004771133e-305 3.766113167064264e-306
1.8077343201908454e-305 1.5064452668257057e-306
7.230937280763382e-306 6.0257810673028224e-307
2.892374912305353e-306 2.410312426921129e-307
1.1569499649221412e-306 9.641249707684517e-308
4.627799859688565e-307 3.8564998830738067e-308
1.851119943875426e-307 1.542599953229523e-308
7.404479775501704e-308 6.170399812918093e-309
2.961791910200682e-308 2.468159925167237e-309
1.1847167640802727e-308 9.87263970066896e-310
4.73886705632109e-309 3.94905588026756e-310
1.895546822528435e-309 1.57962235210705e-310
7.58218729011373e-310 6.318489408428e-311
3.0328749160455e-310 2.5273957633713e-311
```

In [73]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 1")

# Learning rate
print("Learning rate = 0.4")
LR=0.4
for i in range(465):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    print(A,B)
```

```
8.517641300338355e-308 7.098034416948643e-309
1.7035282600676707e-308 1.41960688338973e-309

3.40705652013534e-309 2.83921376677946e-310
6.81411304027066e-310 5.678427533559e-311
1.3628226080541e-310 1.1356855067117e-311
2.725645216108e-311 2.271371013424e-312
5.451290432215e-312 4.54274202686e-313
1.090258086445e-312 9.085484054e-314
2.1805161729e-313 1.817096811e-314
4.3610323456e-314 3.63419362e-315
8.72206469e-315 7.26838726e-316
1.744412936e-315 1.45367744e-316
3.4888259e-316 2.907355e-317
6.9776516e-317 5.81471e-318
1.3955304e-317 1.16294e-318
2.79106e-318 2.32586e-319
5.5821e-319 4.6516e-320
1.11644e-319 9.303e-321
2.2327e-320 1.863e-321
```

In [74]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 1")

# Learning rate
print("Learning rate = 0.5")
LR=.5
for i in range(10):
    (A,B)=np.array((A,B))-(np.array((2*LR*A,2*LR*B)))
    print(A,B)
```

```
Problem 1
Learning rate = 0.5
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
0.0 0.0
```

In [75]:
```
#########################  Problem 2    ########################################

x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 2")

# Learning rate
LR=0.2
for i in range(500):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    print(A,B)
```

```
1.842082581016946e-05 1.74434622827354e-106
1.7960305164915223e-05 -1.0466077369641242e-106
1.7511297535792342e-05 6.279646421784745e-107
1.7073515097397533e-05 -3.7677878530708476e-107
1.6646677219962596e-05 2.2606727118425086e-107
1.6230510289463532e-05 -1.3564036271055051e-107
1.5824747532226945e-05 8.138421762633033e-108
1.542912884392127e-05 -4.883053057579821e-108
1.5043400622823239e-05 2.9298318345478934e-108
1.4667315607252657e-05 -1.757899100728736e-108
1.4300632717071341e-05 1.0547394604372419e-108
1.3943116899144558e-05 -6.328436762623451e-109
1.3594538976665943e-05 3.7970620575740714e-109
1.3254675502249294e-05 -2.2782372345444432e-109
1.2923308614693062e-05 1.366942340726666e-109
1.2600225899325736e-05 -8.201654044359999e-110
1.2285220251842592e-05 4.920992426615999e-110
1.1978089745546528e-05 -2.9525954559695997e-110
1.1678637501907865e-05 1.7715572735817602e-110
```

In [76]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 2")

# Learning rate
LR=0.3
for i in range(500):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    print(A,B)
```

```
4.0650992394342557e-08 4.598634785754457e+68
3.912658017955471e-08 -6.43808870005624e+68
3.765933342282141e-08 9.013324180078734e+68
3.6247108419465606e-08 -1.2618653852110226e+69
3.4887841853735645e-08 1.7666115392954315e+69
3.3579547784220555e-08 -2.4732561550136038e+69
3.232031474231229e-08 3.462558617019045e+69
3.1108302939475575e-08 -4.847582063826662e+69
2.994174157924524e-08 6.7866148893573265e+69
2.8818926270023543e-08 -9.501260845100256e+69
2.773821653489766e-08 1.330176518314036e+70
2.6698033414838997e-08 -1.8622471256396503e+70
2.5696857161782534e-08 2.6071459758955105e+70
2.473322501821569e-08 -3.650004366253714e+70
2.3805729080032604e-08 5.1100061127551996e+70
2.291301423953138e-08 -7.154008557857278e+70
2.2053776205548954e-08 1.0015611981000188e+71
2.1226759597840868e-08 -1.4021856773400263e+71
2.0430756112921834e-08 1.9630599482760366e+71
```

In [77]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 2")

# Learning rate
LR=0.4
for i in range(500):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    print(A,B)
```

```
0.001 5.085001857777038e+161
0.001 -1.1187004087109482e+162
0.001 2.461140899164086e+162
0.001 -5.41450997816099e+162
0.001 1.191192195195418e+163
0.001 -2.62062282942992e+163
0.001 5.765370224745825e+163
0.001 -1.2683814494440818e+164
0.001 2.79043918877698e+164
0.001 -6.138966215309357e+164
0.001 1.3505725673680584e+165
0.001 -2.971259648209729e+165
0.001 6.536771226061404e+165
0.001 -1.4380896697335088e+166
0.001 3.163797273413719e+166
0.001 -6.960354001510183e+166
0.001 1.5312778803322402e+167
0.001 -3.3688113367309287e+167
0.001 7.411384940808044e+167
```

In [78]:
```
x_1=3
x_2=0.25

(A,B)=(x_1,x_2)
print("Problem 2")

# Learning rate
LR=0.5
for i in range(1564):
    (A,B)=np.array((A,B))-(np.array(((LR*A)/8,(LR*B)*8)))
    print(A,B)
```

```
1.39545092773245e-43 nan
1.3082352447491717e-43 nan
1.2264705419523486e-43 nan
1.1498161330803269e-43 nan
1.0779526247628065e-43 nan
1.010580585715131e-43 nan
9.474192991079353e-44 nan
8.882055929136894e-44 nan
8.326927433565838e-44 nan
7.806494468967973e-44 nan

7.318588564657474e-44 nan
6.861176779366382e-44 nan
6.432353230655983e-44 nan
6.030331153739984e-44 nan
5.653435456631235e-44 nan
5.300095740591783e-44 nan
4.968839756804797e-44 nan
4.658287272004498e-44 nan
4.3671443175042167e-44 nan
```

In [1]:
```
### Problem 4 A ###############

import numpy as np
def lossfunction(x_1,x_2):
    return (x_1**2,x_2**2)
```

In [2]:
```
a=lossfunction(2,3)
a
```

Out[2]:
```
(4, 9)
```

In [3]:
```
################ Problem 4 B #########

def finiteDiff(x_1,x_2,h):
    fx1=np.array(np.array(lossfunction(x_1+h,x_2))-np.array(lossfunction(x_1,x_2
    fx2=np.array(np.array(lossfunction(x_1,x_2+h))-np.array(lossfunction(x_1,x_2
    fx1=(round(fx1[0],4))
    fx2=(round(fx2[1],4))
    return (fx1,fx2)
```

In [4]:
```
def AnalyticalGradient(x_1,x_2):
    return (2*x_1,2*x_2)
```

In [5]:
```
################ Problem 4 C #########


q=finiteDiff(1,2,0.1)
w=finiteDiff(3,4,0.1)
e=finiteDiff(5,6,0.1)
print(q)
print(w)
print(e)
```

```
(2.1, 4.1)
(6.1, 8.1)
(10.1, 12.1)
```

In [6]:
```
r=AnalyticalGradient(1,2)
t=AnalyticalGradient(3,4)
y=AnalyticalGradient(5,6)
print(r)
print(t)
print(y)
```

```
(2, 4)
(6, 8)
(10, 12)
```

In [7]:
```
q=finiteDiff(1,2,0.01)
w=finiteDiff(3,4,0.01)
e=finiteDiff(5,6,0.01)
print(q)
print(w)
print(e)
```

```
(2.01, 4.01)
(6.01, 8.01)
(10.01, 12.01)
```

In [8]:
```
r=AnalyticalGradient(1,2)
t=AnalyticalGradient(3,4)
y=AnalyticalGradient(5,6)
print(r)
print(t)
print(y)
```

```
(2, 4)
(6, 8)
(10, 12)
```