Assignment 8: Question 1

Saarthak Marathe | ME17B162

Animation GIF is added separately in the submission

In [121]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [122]:

```
def f(x):
    return x[0]**2+x[1]**2-6*x[0]-8*x[1]+15

def g1(x):
    return 4*(x[0]**2)+x[1]**2-16

def g2(x):
    return 3*x[0]+5*x[1]-15

def g3(x):
    return -x[0]

def g4(x):
    return -x[1]
```

In [126]:

```
def cont_score (g1,g2,g3,g4):
    func_vals = [g1,g2,g3,g4]
    abv = np.array([max(i, 0) for i in func_vals])
    return np.sum(abv**2)

def err_diff_f(x):
    diff_x0 = 2*x[0]-6
    diff_x1 = 2*x[1]-8
    return diff_x0**2+diff_x1**2

def check(x):
    if g1(x)<=0 and g2(x)<=0 and x[0]>=0 and x[1]>=0:
        return True
    else:
        return False

max_iters = 4000
min_cost = 1e05
max_err = 1e-3
err=1
i=0
ini = np.array([0.0, 0.0])
x = ini
points = [x.tolist()]
while err > max_err:
    rand = np.random.uniform(-1, 1, size=x.shape)
    x_new = x + rand
    cost = f(x_new) + cont_score(g1(x_new),g2(x_new),g3(x_new),g4(x_new))
    err = err_diff_f(x_new)
    if (cost < min_cost and check(x_new)==True):
        min_cost = cost
        x = x_new
        if x.tolist() not in points:
            points.append(x.tolist())
    if i%int(0.025*max_iters) == 0:
        print('Iteration:',i,' - ', x,' - Cost:', min_cost)
    i = i + 1
    if i>max_iters:
        break
```

```
Iteration: 0  -  [0.22150665 0.25465339]  - Cost: 11.747646535375612
Iteration: 100  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 200  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 300  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 400  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 500  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 600  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 700  -  [1.40726789 2.13750768]  - Cost: -3.994326782008091
Iteration: 800  -  [1.5252775  2.05869121]  - Cost: -4.056513717849324
Iteration: 900  -  [1.65466486 1.98988591]  - Cost: -4.149514707585478
Iteration: 1000  -  [1.65466486 1.98988591]  - Cost: -4.149514707585478
Iteration: 1100  -  [1.65466486 1.98988591]  - Cost: -4.149514707585478
Iteration: 1200  -  [1.65466486 1.98988591]  - Cost: -4.149514707585478
Iteration: 1300  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 1400  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 1500  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 1600  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 1700  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
```

```
Iteration: 1800  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 1900  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2000  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2100  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2200  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2300  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2400  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2500  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2600  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2700  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2800  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 2900  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3000  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3100  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3200  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3300  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3400  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3500  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3600  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3700  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3800  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 3900  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
Iteration: 4000  -  [1.6922331  1.98337481]  - Cost: -4.222968586873751
```

In [127]:

```python
print('Unique Points during the iterations:')
points
```

```
Unique Points during the iterations:
```

Out[127]:

```
[[0.0, 0.0],
 [0.22150664953347565, 0.25465338896532574],
 [0.8483001232499685, 0.07353661723247673],
 [1.219367701264621, 0.43153080461950943],
 [1.9767467574271433, 0.2202207893021031],
 [1.5116435081532311, 0.48193833152432264],
 [1.329180523349121, 0.7424091489827529],
 [1.9341185600394863, 0.8869771075153641],
 [1.4339768792481062, 1.5224560344083393],
 [1.7733333386722614, 1.6934937239307353],
 [1.6689793956908499, 1.8990389951469413],
 [1.4072678897482789, 2.1375076797567703],
 [1.5252774970416676, 2.0586912091532197],
 [1.6546648588252921, 1.989885911114757],
 [1.6922331018701853, 1.9833748109075469]]
```

In [128]:

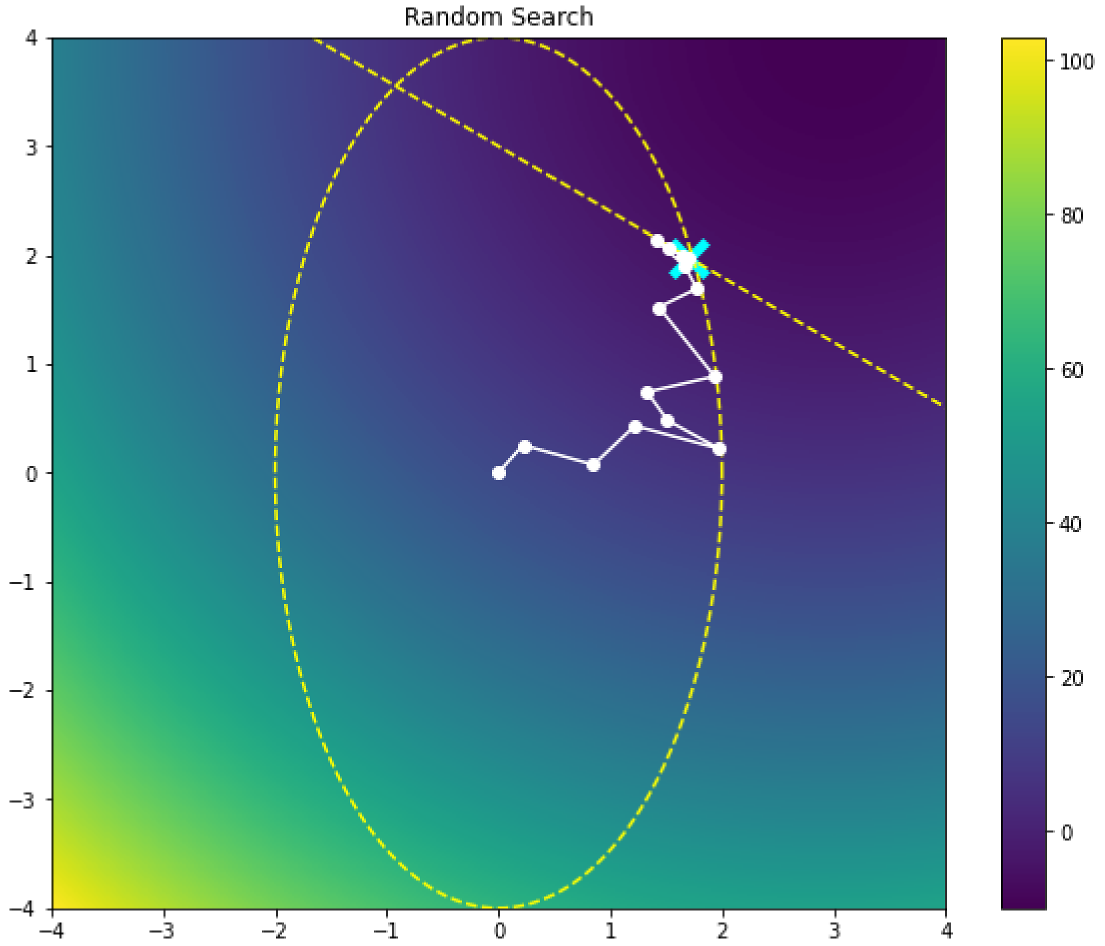
```python
print('Final point is:', points[-1], 'and optimal value is:', f(points[-1]))
```

```
Final point is: [1.6922331018701853, 1.9833748109075469] and optimal value i
s: -4.222968586873751
```

Final Plot

In [129]:

```
points = np.array(points)
fig = plt.figure(figsize=(10, 8))
x1 = np.linspace(-4, 4, 400)
x2 = np.linspace(-4, 4, 400)
X1, X2 = np.meshgrid(x1, x2)
Y = np.array([f(a) for a in np.c_[X1.ravel(), X2.ravel()]]).reshape(X1.shape)
plt.pcolormesh(X1, X2, Y)
plt.colorbar()
theta = np.linspace(0, 2*np.pi, 400)
x1 = 2*np.cos(theta)
y1 = 4*np.sin(theta)
x2 = np.linspace(-4, 4, 400)
y2 = (15-3*x2)/5
plt.plot(x1, y1, color='yellow', linestyle='--')
plt.plot(x2, y2, color='yellow', linestyle='--')
plt.xlim(-4, 4)
plt.ylim(-4, 4)
plt.plot(points[:, 0], points[:, 1], color='white', marker='o')
plt.scatter(points[-1, 0], points[-1, 1], c='cyan', marker='x', s=300, linewidth=5)
plt.title('Random Search')
plt.show()
```



For Animation

In [130]:

```
for i in range(len(points)):
    x1 = np.linspace(-4, 4, 400)
    x2 = np.linspace(-4, 4, 400)
    X1, X2 = np.meshgrid(x1, x2)
    Y = np.array([f(a) for a in np.c_[X1.ravel(), X2.ravel()]]).reshape(X1.shape)
    plt.pcolormesh(X1, X2, Y)
    plt.colorbar()
    theta = np.linspace(0, 2*np.pi, 400)
    x1 = 2*np.cos(theta)
    y1 = 4*np.sin(theta)
    x2 = np.linspace(-4, 4, 400)
    y2 = (15-3*x2)/5
    plt.plot(x1, y1, color='yellow', linestyle='--')
    plt.plot(x2, y2, color='yellow', linestyle='--')
    plt.xlim(-4, 4)
    plt.ylim(-4, 4)
    plt.plot(points[:i, 0], points[:i, 1], color='white', marker='o')
    if i == len(points)-1:
        plt.scatter(points[-1, 0], points[-1, 1], c='cyan', marker='x', s=300, linewidt
    plt.title('Random Search')
    plt.savefig(str(i)+'.png')
    plt.clf()
```

```
<Figure size 432x288 with 0 Axes>
```

In [131]:

```
import imageio
images = []
for i in range(len(points)):
    images.append(imageio.imread(str(i)+'.png'))
imageio.mimsave('movie_q1.gif', images)
```