Assignment 8: Question 2

Saarthak Marathe | ME17B162

Animation GIF is added separately in the submission

In [1]:

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

In [2]:

```
def f(x):
    return (x[0]-5)**2 + (x[1]-5)**2

def g1(x):
    return x[0]+2*x[1]-15

def g2(x):
    return -x[0]+1

def g3(x):
    return -x[1]+1

def g4(x):
    return x[0]-10

def g5(x):
    return x[1]-10
```

In [30]:

```
def cont_score (g1,g2,g3,g4,g5):
    func_vals = [g1,g2,g3,g4,g5]
    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]-10
    diff_x1 = 2*x[1]-10
    return diff_x0**2+diff_x1**2

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

def centroid(x):
    return np.mean(x,axis=0)

def worst_no(points):
    max_no = 0
    max_val = f(points[0])
    for i in range(len(points)-1):
        if f(points[i+1]) > max_val:
            max_no = i
            max_val = f(points[i])
    return max_no

def complex_pts(points, pt_worst):
    alpha = 1
    while alpha>1e-4:
        pt_new = (1.0 + alpha)*np.array(centroid(points)) - alpha*np.array(pt_worst)
        if (f(pt_new)<f(pt_worst) and check(pt_new) == True):
            points.append(pt_new.tolist())
            return points
        else:
            alpha = alpha/2
    return points.append(pt_worst.tolist())
```

In [44]:

```
ini = [[1.01,1.01],[1.1,1.1],[1.1,1.01],[1.01,1.1]]
mid = centroid(ini)
err = err_diff_f(mid)
points = ini
i = 0
max_iters = 4000
max_err = 1e-4
mid_points = [mid.tolist()]

while err>max_err:
    worst_no = worst_no(points)
    pt_worst = points[worst_no]
    points.remove(pt_worst)
    points = complex_pts(points, pt_worst)
    if points[worst_no(points)] != pt_worst:
        mid = centroid(points)
        mid_points.append(mid.tolist())
        err = err_diff_f(mid)
        i = i+1
    else:
        break
    if i%int(0.0005*max_iters) == 0:
        print('Iteration:',i,' - Centroid Point ', mid,' - Cost:', f(mid))
    if i>max_iters:
        break
```

```
Iteration: 2  - Centroid Point  [1.075 1.135]  - Cost: 30.34385
Iteration: 4  - Centroid Point  [1.13055556 1.19722222]  - Cost: 29.43371913
580247
Iteration: 6  - Centroid Point  [1.17820988 1.26339506]  - Cost: 28.56829621
2467608
Iteration: 8  - Centroid Point  [1.19206447 1.34646776]  - Cost: 27.84867078
5374097
Iteration: 10  - Centroid Point  [1.22133135 1.42183432]  - Cost: 27.0816063
5656894
Iteration: 12  - Centroid Point  [1.28097301 1.4820135 ]  - Cost: 26.2073908
19468337
Iteration: 14  - Centroid Point  [1.35773573 1.51363213]  - Cost: 25.4208498
9609765
Iteration: 16  - Centroid Point  [1.43960231 1.52714329]  - Cost: 24.7371654
47628772
Iteration: 18  - Centroid Point  [1.50321362 1.57323887]  - Cost: 23.9702068
2199426
Iteration: 20  - Centroid Point  [1.56577781 1.63020094]  - Cost: 23.1494277
1582229
Iteration: 22  - Centroid Point  [1.64682996 1.65216382]  - Cost: 22.4517564
15439277
Iteration: 24  - Centroid Point  [1.7248208 1.6710378]  - Cost: 21.808788163
720575
Iteration: 26  - Centroid Point  [1.78500397 1.7255974 ]  - Cost: 21.0579119
0242077
Iteration: 28  - Centroid Point  [1.85220571 1.77531576]  - Cost: 20.3071973
89890625
Iteration: 30  - Centroid Point  [1.93565543 1.79042433]  - Cost: 19.6915836
55935084
Iteration: 32  - Centroid Point  [1.95418034 1.87322275]  - Cost: 19.0537533
73343355
Iteration: 34  - Centroid Point  [2.00708206 1.93881707]  - Cost: 18.3283987
```

```
2382863
Iteration: 36  - Centroid Point  [2.06297371 1.99570223]  - Cost: 17.6519285
10072345
Iteration: 38  - Centroid Point  [2.08382413 2.07172685]  - Cost: 17.0788653
06386135
Iteration: 40  - Centroid Point  [2.10947155 2.15220743]  - Cost: 16.4650772
01801027
Iteration: 42  - Centroid Point  [2.16909421 2.21255833]  - Cost: 15.7838586
49304147
Iteration: 44  - Centroid Point  [2.24944263 2.23936823]  - Cost: 15.1866536
19800431
Iteration: 46  - Centroid Point  [2.3242829  2.26020307]  - Cost: 14.6659492
57542621
Iteration: 48  - Centroid Point  [2.37962144 2.31590669]  - Cost: 14.0707407
0870029
Iteration: 50  - Centroid Point  [2.44480185 2.36983676]  - Cost: 13.4467962
84205721
Iteration: 52  - Centroid Point  [2.5272969  2.38929216]  - Cost: 12.9300560
38321652
Iteration: 54  - Centroid Point  [2.54127267 2.47279904]  - Cost: 12.4320847
69386417
Iteration: 56  - Centroid Point  [2.58971344 2.54013567]  - Cost: 11.8604137
97139863
Iteration: 58  - Centroid Point  [2.6441849 2.6016311]  - Cost: 11.302038198
604098
Iteration: 60  - Centroid Point  [2.66285119 2.68059482]  - Cost: 10.8419049
45447805
Iteration: 62  - Centroid Point  [2.74363383 2.70170884]  - Cost: 10.3733305
62696115
Iteration: 64  - Centroid Point  [2.80629111 2.75808276]  - Cost: 9.83855159
6942954
Iteration: 66  - Centroid Point  [2.87105023 2.80466179]  - Cost: 9.35193700
1507014
Iteration: 68  - Centroid Point  [2.95346762 2.81815756]  - Cost: 8.94873122
1982968
Iteration: 70  - Centroid Point  [3.02985912 2.84880005]  - Cost: 8.50911628
2776393
Iteration: 72  - Centroid Point  [3.08993144 2.90871542]  - Cost: 8.02183310
3135465
Iteration: 74  - Centroid Point  [3.12033515 2.98458914]  - Cost: 7.59502086
9598549
Iteration: 76  - Centroid Point  [3.13433011 3.06725384]  - Cost: 7.21623185
0377756
Iteration: 78  - Centroid Point  [3.18164006 3.13245108]  - Cost: 6.79417183
8040764
Iteration: 80  - Centroid Point  [3.23795543 3.194739  ]  - Cost: 6.36376833
6950585
Iteration: 82  - Centroid Point  [3.25895461 3.2751409 ]  - Cost: 6.00637798
5290124
Iteration: 84  - Centroid Point  [3.2787964  3.35442678]  - Cost: 5.67045302
6762626
Iteration: 86  - Centroid Point  [3.33421159 3.4158467 ]  - Cost: 5.28439270
6024013
Iteration: 88  - Centroid Point  [3.38287896 3.48235284]  - Cost: 4.91833336
9152984
Iteration: 90  - Centroid Point  [3.39747246 3.56559467]  - Cost: 4.62561316
6805344
Iteration: 92  - Centroid Point  [3.48003983 3.5842065 ]  - Cost: 4.31475012
8780188
Iteration: 94  - Centroid Point  [3.54580638 3.63766135]  - Cost: 3.97064566
8086627
```

```
Iteration: 96  - Centroid Point  [3.60098666 3.69217857]  - Cost: 3.66763520
53969277
Iteration: 98  - Centroid Point  [3.67532028 3.71161669]  - Cost: 3.41470792
16964356
Iteration: 100  - Centroid Point  [3.75598386 3.73778779]  - Cost: 3.1407558
05456286
Iteration: 102  - Centroid Point  [3.81608746 3.79754178]  - Cost: 2.8475546
909272844
Iteration: 104  - Centroid Point  [3.8422747  3.87759998]  - Cost: 2.6001096
737728724
Iteration: 106  - Centroid Point  [3.86440364 3.95406332]  - Cost: 2.3835626
30636919
Iteration: 108  - Centroid Point  [3.92135077 4.0110213 ]  - Cost: 2.1415630
23882541
Iteration: 110  - Centroid Point  [3.97461883 4.07590867]  - Cost: 1.9053513
252153496
Iteration: 112  - Centroid Point  [3.99388627 4.15853931]  - Cost: 1.7203209
2371764
Iteration: 114  - Centroid Point  [4.07750412 4.17293756]  - Cost: 1.5350309
237517137
Iteration: 116  - Centroid Point  [4.14541207 4.22232866]  - Cost: 1.3350932
399548783
Iteration: 118  - Centroid Point  [4.20557635 4.27581591]  - Cost: 1.1555515
41785677
Iteration: 120  - Centroid Point  [4.28317134 4.29340004]  - Cost: 1.0131268
232061497
Iteration: 122  - Centroid Point  [4.36463652 4.3148665 ]  - Cost: 0.8730946
630176277
Iteration: 124  - Centroid Point  [4.42781094 4.37137333]  - Cost: 0.7225718
134864023
Iteration: 126  - Centroid Point  [4.45865583 4.45036346]  - Cost: 0.5951538
343227707
Iteration: 128  - Centroid Point  [4.48245188 4.5234693 ]  - Cost: 0.4949375
721740499
Iteration: 130  - Centroid Point  [4.54014385 4.57539945]  - Cost: 0.3917533
002635205
Iteration: 132  - Centroid Point  [4.59735225 4.63791961]  - Cost: 0.2932274
1812759767
Iteration: 134  - Centroid Point  [4.62077218 4.71926211]  - Cost: 0.2226275
0116062485
Iteration: 136  - Centroid Point  [4.70473393 4.72896558]  - Cost: 0.1606417
081989809
Iteration: 138  - Centroid Point  [4.77399728 4.77367735]  - Cost: 0.1022991
699674756
Iteration: 140  - Centroid Point  [4.83853481 4.82538225]  - Cost: 0.0565623
6669501064
Iteration: 142  - Centroid Point  [4.91882092 4.84052481]  - Cost: 0.0320223
7748730252
Iteration: 144  - Centroid Point  [4.94358275 4.91988161]  - Cost: 0.0096018
61734657553
Iteration: 146  - Centroid Point  [4.99706006 4.96482101]  - Cost: 0.0012462
043593770467
Iteration: 148  - Centroid Point  [5.02245754 4.98247788]  - Cost: 0.0008113
656106519422
Iteration: 150  - Centroid Point  [4.98344862 4.99490338]  - Cost: 0.0002999
2365089809937
Iteration: 152  - Centroid Point  [5.0081254  4.98680561]  - Cost: 0.0002401
1404168766996
Iteration: 154  - Centroid Point  [4.99834023 4.99531126]  - Cost: 2.4739134
842279976e-05
```

In [45]:

```
print('Points at each step will be:')
mid_points
```

```
Points at each step will be:
```

Out[45]:

```
[[1.0550000000000002, 1.0550000000000002],
 [1.085, 1.0850000000000002],
 [1.0750000000000002, 1.135],
 [1.1183333333333336, 1.1583333333333334],
 [1.1305555555555558, 1.1972222222222224],
 [1.1309259259259263, 1.242037037037037],
 [1.1782098765432103, 1.263395061728395],
 [1.174794238683128, 1.3101028806584363],
 [1.1920644718792874, 1.3464677640603566],
 [1.2324531321444907, 1.3712734339277548],
 [1.2213313519280606, 1.4218343240359703],
 [1.255771731951431, 1.4496141340242852],
 [1.2809730054700343, 1.4820134972649837],
 [1.3318558880477802, 1.4715720559761105],
 [1.3577357317181034, 1.5136321341409493],
 [1.3912713515391808, 1.5285309908970772],
 [1.4396023087333418, 1.5271432900777748],
 [1.460550373279304, 1.5746322207677568],
 [1.5032136239831146, 1.5732388670207897],
 [1.544306185791326, 1.5881452610138034],
 [1.5657778133024878, 1.6302009424571253],
 [1.6149813754386484, 1.619757826226722],
 [1.6468299590385267, 1.6521638194443107],
 [1.6740865793951156, 1.6799364644049686],
 [1.724820795945706, 1.671037797593542],
 [1.7488435141475835, 1.7156675614018255],
 [1.7850039672870766, 1.7255973961559135],
 [1.831692272191795, 1.7282653723625523],
 [1.8522057064719308, 1.7753157556866521],
 [1.897091116486285, 1.7704514547349197],
 [1.9356554294795971, 1.7904243257001693],
 [1.9582758961000806, 1.8291956517186083],
 [1.9541803391615904, 1.873222750659467],
 [2.001649993341227, 1.8914436973172433],
 [2.007082056255668, 1.9388170740967097],
 [2.0169990297389093, 1.9731266963303382],
 [2.0629737137289457, 1.9957022278367273],
 [2.0563865398077885, 2.0469869681919404],
 [2.083824132594761, 2.0717268541426272],
 [2.118457227682087, 2.1031506704505247],
 [2.1094715529941452, 2.152207434020001],
 [2.1514487357062073, 2.171069670883495],
 [2.169094211660199, 2.21255832942672],
 [2.2178283254588433, 2.20373549612839],
 [2.2494426288893545, 2.2393682302724027],
 [2.2727947082993905, 2.2660383663348465],
 [2.3242828967715266, 2.260203065730373],
 [2.346518497181338, 2.306670945430024],
 [2.379621439278816, 2.3159066880577592],
 [2.42748718052173, 2.322482099810591],
 [2.4448018478830633, 2.369836756468543],
 [2.4880884812744015, 2.3654794174612386],
```

```
[2.5272969005073134, 2.389292161102489],
[2.5459709725881217, 2.427256790210923],
[2.5412726667392636, 2.4727990443846846],
[2.5882718786153975, 2.494085913041595],
[2.589713444787875, 2.540135670630689],
[2.600201020840236, 2.577423628468766],
[2.6441848960897425, 2.601631097017725],
[2.6344610291965047, 2.6520410177406273],
[2.662851185963114, 2.6805948248540568],
[2.6941303866593382, 2.7120876646061736],
[2.7436338251944647, 2.7017088417936357],
[2.765949236014774, 2.7442198697619498],
[2.8062911126159373, 2.758082759253783],
[2.8497857292241124, 2.757253315933405],
[2.871050226708751, 2.8046617881724565],
[2.9188021430177606, 2.8024453724778136],
[2.953467620017812, 2.818157558468667],
[2.9790942639387707, 2.859589830145886],
[3.029859124607478, 2.848800052555788],
[3.0561451960249464, 2.881919588302414],
[3.089931436362985, 2.908715422200724],
[3.079214783531281, 2.959448779554114],
[3.120335152671997, 2.9845891408157135],
[3.136842385685896, 3.0199159734112877],
[3.1343301115631315, 3.067253840320018],
[3.181790316416069, 3.0883905234772326],
[3.181640056438067, 3.132451083989978],
[3.194997937258949, 3.1721476584468653],
[3.237955428512258, 3.1947390036226992],
[3.2279386317234477, 3.2445013072291298],
[3.2589546085583696, 3.275140895542484],
[3.2882345086037676, 3.3041064791493437],
[3.2787964040781317, 3.3544267843246063],
[3.3227183824367312, 3.3779481321151597],
[3.334211588187384, 3.415846701516922],
[3.3355830745310633, 3.4613745994884484],
[3.3828789593586537, 3.482352837755747],
[3.3790640322813363, 3.5284346270589193],
[3.3974724559266516, 3.565594674685155],
[3.437360557180031, 3.5895468268447663],
[3.480039834189812, 3.5842065040654925],
[3.4975178659163273, 3.6311307100046895],
[3.5458063822641277, 3.6376613525914774],
[3.5782154977334795, 3.645785550929674],
[3.600986663086144, 3.6921785716184012],
[3.65248782947284, 3.6859529400883453],
[3.675320277930848, 3.711616689166136],
[3.707647682593074, 3.7473799163189145],
[3.755983863578364, 3.737787792097196],
[3.7734800532620163, 3.7785699916331525],
[3.8160874550247885, 3.797541777533372],
[3.8078203350037825, 3.846969892045875],
[3.842274698615362, 3.877599982044404],
[3.8706416058339395, 3.902837776115949],
[3.864403637943934, 3.9540633226041133],
[3.9103929599250407, 3.9760308284637684],
[3.9213507705199144, 4.011021302744817],
[3.926789973091987, 4.057905859759184],
[3.974618831080694, 4.0759086713744],
[3.9714467565366895, 4.120526394121831],
[3.993886269952999, 4.158539314092126],
```

```
 [4.033177931954935, 4.178743726633054],
 [4.077504123444436, 4.172937558893105],
 [4.098265460364891, 4.219620672290359],
 [4.145412073889843, 4.222328657785553],
 [4.1809431731778455, 4.2311808660129575],
 [4.205576348177283, 4.275815905166141],
 [4.256355603131757, 4.266596280352742],
 [4.283171342434748, 4.293400043235673],
 [4.31579235598468, 4.3260272864900795],
 [4.364636519523508, 4.314866501552854],
 [4.386044542163534, 4.356266273832996],
 [4.427810936013067, 4.37137333134828],
 [4.421083050133946, 4.420205336149122],
 [4.458655832683524, 4.450363459334078],
 [4.485655337056824, 4.471695144054657],
 [4.482451877236462, 4.523469295010292],
 [4.53009231451726, 4.543479929450228],
 [4.540143853190173, 4.575399453009373],
 [4.5494700262391055, 4.623203974258605],
 [4.5973522520612295, 4.637919609468513],
 [4.5945517731430785, 4.680868761707432],
 [4.620772181105437, 4.719262110613659],
 [4.6589807779673915, 4.735496346934463],
 [4.7047339267684185, 4.728965584846899],
 [4.7284394840844195, 4.774947266555917],
 [4.77399727810805, 4.773677354944527],
 [4.812466348006533, 4.782897123963766],
 [4.838534813364249, 4.825382245462574],
 [4.888226142234801, 4.8130238830246626],
 [4.918820924295673, 4.840524813356141],
 [4.951254905256615, 4.869723503931818],
 [4.94358275173394, 4.9198816136411665],
 [4.987546245289351, 4.940396070928088],
 [4.9970600569153385, 4.964821013459225],
 [4.99587429867255, 4.986195794152514],
 [5.0224575391819775, 4.9824778841305655],
 [5.001768241554984, 4.990017968978629],
 [4.983448620588659, 4.994903384409245],
 [4.9956617012328755, 4.991646440788834],
 [5.008125404292066, 4.986805613057889],
 [4.999816268919273, 4.990032831545186],
 [4.998340231433445, 4.995311257828825]]
```

In [47]:

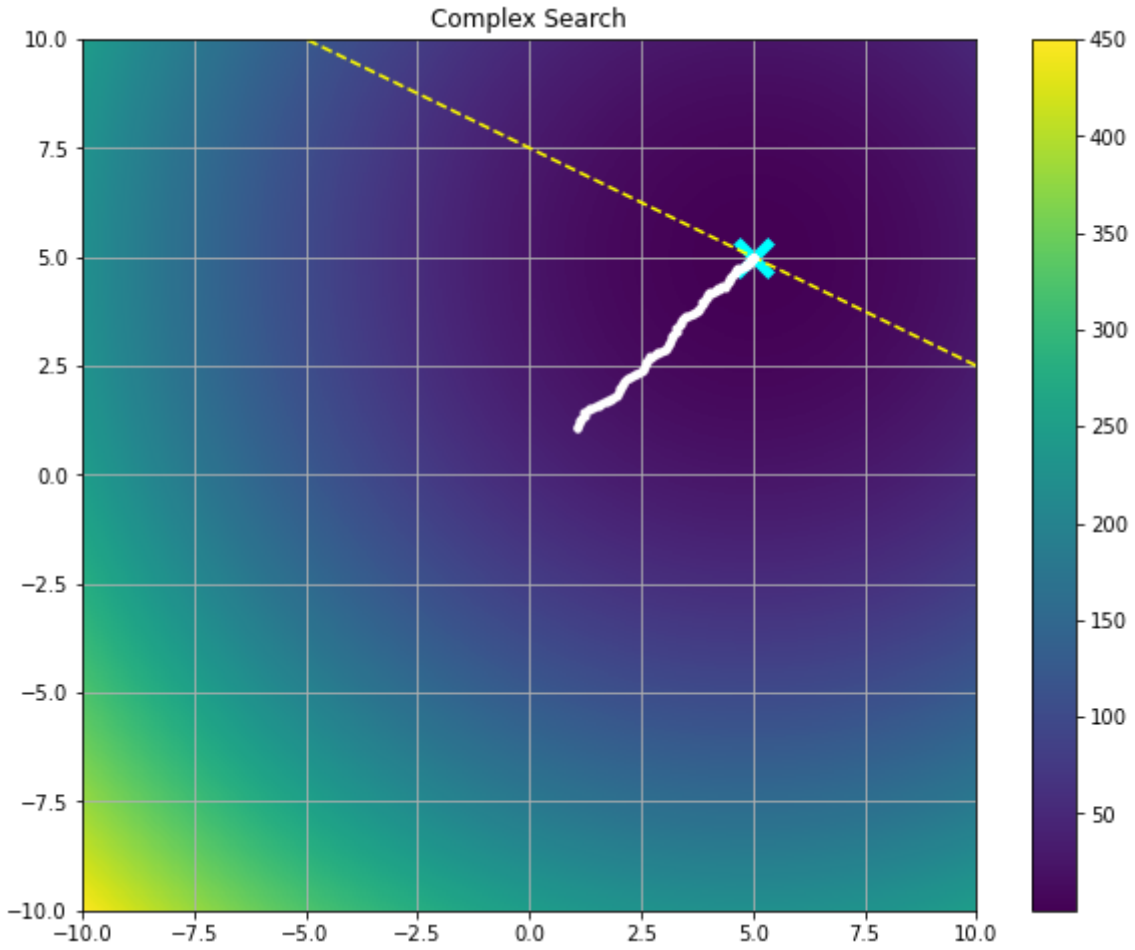
```
print('Final centroid point is:', mid_points[-1], 'and optimal value is:', f(mid_points
```

```
Final centroid point is: [4.998340231433445, 4.995311257828825] and optimal
value is: 2.4739134842279976e-05
```

Final Plot

In [53]:

```
mid_points = np.array(mid_points)
fig = plt.figure(figsize=(10, 8))
x1 = np.linspace(-10, 10, 400)
x2 = np.linspace(-10, 10, 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()
x2 = np.linspace(-10, 10, 400)
y2 = (15-x2)/2
plt.plot(x2, y2, color='yellow', linestyle='--')
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.plot(mid_points[:, 0], mid_points[:, 1], color='white', marker='.')
plt.scatter(mid_points[-1, 0], mid_points[-1, 1], c='cyan', marker='x', s=300, linewidt
plt.title('Complex Search')
plt.grid()
plt.show()
```



For Animation

In [61]:

```
for i in range(len(mid_points)):
    x1 = np.linspace(-10, 10, 400)
    x2 = np.linspace(-10, 10, 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()
    x2 = np.linspace(-10, 10, 400)
    y2 = (15-x2)/2
    plt.plot(x2, y2, color='yellow', linestyle='--')
    plt.xlim(-10, 10)
    plt.ylim(-10, 10)
    plt.plot(mid_points[:i, 0], mid_points[:i, 1], color='white', marker='o')
    if i==len(mid_points)-1:
        plt.scatter(mid_points[-1, 0], mid_points[-1, 1], c='cyan', marker='x', s=300,
    plt.grid()
    plt.title('Complex Search')
    plt.savefig(str(i)+'q2.png')
    plt.clf()
```

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

In [63]:

```
images = []
for i in range(len(mid_points)):
    images.append(imageio.imread(str(i)+'q2.png'))
imageio.mimsave('movie_q2.gif', images)
```