In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
from collections import deque
```

In [2]:

```
def generate_direction(directions, i):
    return directions[i % 2]

def objective(x, q=1):
    if q == 1:
        return 100*(x[1] - x[0]**2)**2 + (1-x[0])**2
    elif q == 2:
        return (x[0] + 2*x[1] - 7)**2 + (2*x[0] + x[1] - 5)**2
    else:
        raise ValueError('Bad question number')

def find_best_lambda(x, lbd, q):
    sol = solve(diff(objective(x, q)), lbd)
    sol = np.array([float(re(i)) for i in sol])
    hess = diff(diff(objective(x, 1)))
    hess_vals = np.array([float(hess.evalf(subs={lbd: i})) for i in sol])

    if len(sol) == 1:
        return sol

    candidates = [i for i in range(len(sol)) if hess_vals[i] > 0]
    hess_vals = [hess_vals[i] for i in candidates]

    try:
        return sol[candidates[np.argmax(hess_vals)]]
    except:
        return 'None'


def powells_search(x_0, q):

    # Parameters to start off
    x_0 = np.asarray(x_0)
    x = x_0

    # Determine starting direction
    directions = deque(maxlen=2)
    x_mov, y_mov = x + np.array([0.01, 0]), x + np.array([0, 0.01])
    if objective(x_mov, q) < objective(y_mov, q):
        directions.extend([np.array([1, 0]), np.array([0, 1])])
    else:
        directions.extend([np.array([0, 1]), np.array([1, 0])])

    u = directions[0]

    # Deque to store visited points
    visited = deque(maxlen=3)

    # Other controllers
    count = 0
    feasible_count = 0
    done = False
    lbd = Symbol('lambda')

    # Movement of point
    points = []

    # Loop
    while not done:
```

```python

        # Find next symbolic point
        x_symb = x + lbd * u

        # Find best lambda and compute new vector
        step_size = find_best_lambda(x_symb, lbd, q)

        if isinstance(step_size, str):
            break
        else:
            x_new = x + step_size * u
            visited.append(x_new)

        if count >= 4:
            new_dir = visited[-1] - visited[0]
            directions.append(new_dir)

        # Decision
        x = x_new
        feasible_count += 1
        u = generate_direction(directions, count+1)

        if count % 50 == 0:
            print("Iteration {:3d} - x {} - f(x) {:.5f}".format(
                count, x, objective(x, q)
            ))
        points.append(x)

        count += 1
        if count > 2000:
            done = True

    return x, feasible_count, np.array(points)
```

In [18]:

```
opt, feasible_count, points = powells_search([0.0, 1.5], q=1)
```

```
Iteration   0 - x [1.22437075 1.5       ] - f(x) 0.05043
Iteration  50 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 100 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 150 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 200 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 250 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 300 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 350 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 400 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 450 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 500 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 550 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 600 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 650 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 700 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 750 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 800 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 850 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 900 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 950 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1000 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1050 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1100 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1150 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1200 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1250 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1300 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1350 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1400 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1450 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1500 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1550 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1600 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1650 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1700 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1750 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1800 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1850 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1900 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 1950 - x [0.88413459 0.78018583] - f(x) 0.01365
Iteration 2000 - x [0.88413459 0.78018583] - f(x) 0.01365
```

In [21]:

```
opt
```

Out[21]:

```
array([0.88413459, 0.78018583])
```

In [22]:

```
feasible_count
```

Out[22]:

2001

In [23]:

```
final_points = [[0.0, 1.5]]
for i in points.tolist():
    if i in final_points:
        continue
    else:
        final_points.append(i)

final_points = np.array(final_points)
```

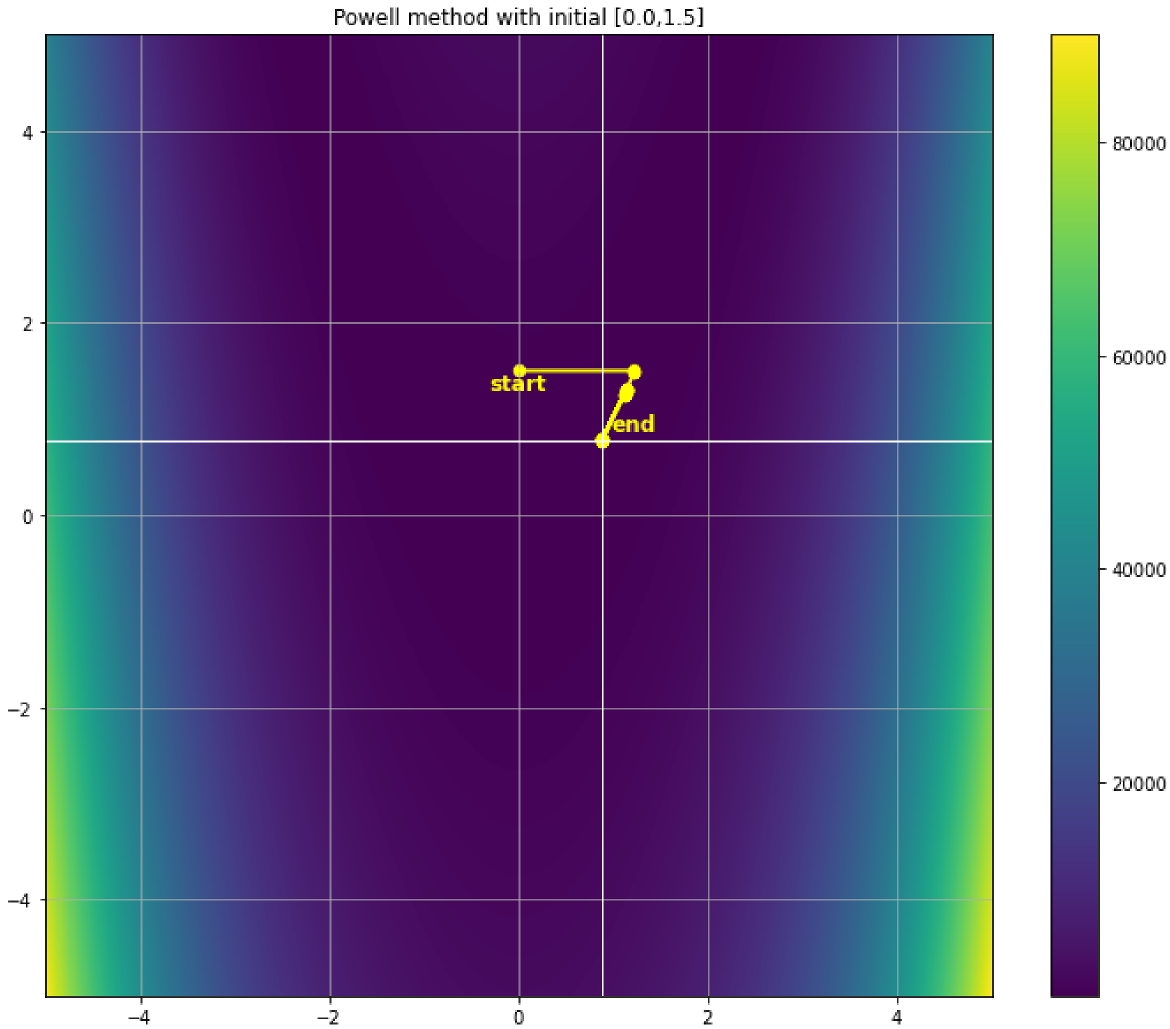
In [26]:

```
def track_movements(q, points):
    x = np.linspace(-5, 5, 1000)
    y = np.linspace(-5, 5, 1000)
    xx, yy = np.meshgrid(x, y)
    zz = np.array([objective(a, q) for a in np.c_[xx.ravel(), yy.ravel()]])
    zz = zz.reshape(xx.shape)

    plt.figure(figsize=(12, 10))
    plt.pcolormesh(xx, yy, zz)
    plt.colorbar()
    plt.plot(points[:, 0], points[:, 1], color='yellow', linewidth=2)
    plt.scatter(points[:, 0], points[:, 1], c='yellow', s=40, edgecolor='yellow')
    plt.text(points[0][0]-0.3, points[0][1]-0.2, 'start', color='yellow', fontsize=12,
    plt.text(points[-1][0]+0.1, points[-1][1]+0.1, 'end', color='yellow', fontsize=12,
    plt.axvline(points[-1][0], ymin=0, ymax=1, color='white', linewidth=1)
    plt.axhline(points[-1][1], xmin=0, xmax=1, color='white', linewidth=1)
    plt.grid(alpha=0.8)
    plt.title('Powell method with initial [0.0,1.5]')
    plt.show()
```

In [27]:

```
1 track_movements(1, final_points)
```



In [31]:

```
objective(opt, 2)
```

Out[31]:

```
0.0
```

In [28]:

```
opt, feasible_count, points = powells_search([0, 0], q=2)
```

```
Iteration   0 - x [0.  3.8] - f(x) 1.80000
```

In [29]:

```
opt
```

Out[29]:

```
array([1., 3.])
```

In [30]:

```
feasible_count
```

Out[30]:

```
9
```

In [111]:

```
final_points = [[0, 0]]
for i in points.tolist():
    if i in final_points:
        continue
    else:
        final_points.append(i)

final_points = np.array(final_points)
```

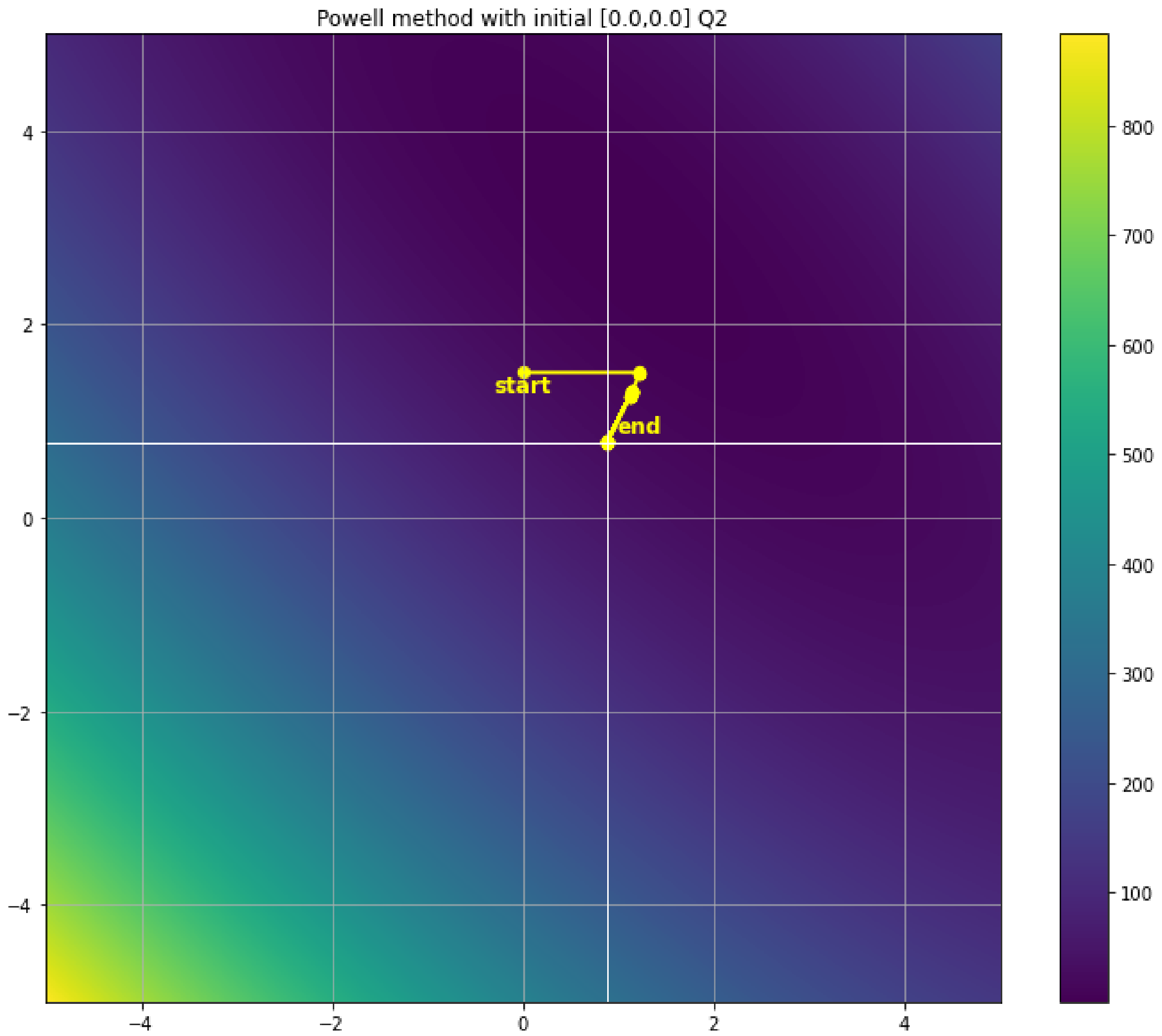
In [32]:

```
def track_movements(q, points):
    x = np.linspace(-5, 5, 1000)
    y = np.linspace(-5, 5, 1000)
    xx, yy = np.meshgrid(x, y)
    zz = np.array([objective(a, q) for a in np.c_[xx.ravel(), yy.ravel()]])
    zz = zz.reshape(xx.shape)

    plt.figure(figsize=(12, 10))
    plt.pcolormesh(xx, yy, zz)
    plt.colorbar()
    plt.plot(points[:, 0], points[:, 1], color='yellow', linewidth=2)
    plt.scatter(points[:, 0], points[:, 1], c='yellow', s=40, edgecolor='yellow')
    plt.text(points[0][0]-0.3, points[0][1]-0.2, 'start', color='yellow', fontsize=12,
    plt.text(points[-1][0]+0.1, points[-1][1]+0.1, 'end', color='yellow', fontsize=12,
    plt.axvline(points[-1][0], ymin=0, ymax=1, color='white', linewidth=1)
    plt.axhline(points[-1][1], xmin=0, xmax=1, color='white', linewidth=1)
    plt.grid(alpha=0.8)
    plt.title('Powell method with initial [0.0,0.0] Q2')
    plt.show()
```

In [33]:

```
track_movements(2, final_points)
```



In [41]:

```
x =[0.88413459, 0.78018583]
objective(x, 1)
```

Out[41]:

```
0.013652242836035321
```

In [ ]:

```

```