Optim-Mauritius-2023-CS2

January 11, 2023

In [1]:
```
# can problem

import numpy as np
# can function with penalization
def J(v):
    x = v[0]
    y = v[1]
    return x**2+x*y+1/2*rho*(x**2*y-2*V0)**2
def GradJ(v):
    x = v[0]
    y = v[1]
    gradJ = np.array([2*x+y+rho*(2*x*y)*(x**2*y-2*V0), x+rho*(x**2)*(x**2*y-2*V0)])
    return gradJ

# Steepest descent method
def F(J, GradJ, beta, alpha_init, tau, X0, N):
    x_k = X0
    for k in range(N):
        d_k = -GradJ(x_k)
        alpha = alpha_init
        while (not(J(x_k + alpha*d_k) <= J(x_k) + alpha*beta*np.dot(d_k, GradJ(x_k))))
            alpha *= tau
        x_k = x_k+alpha*d_k
        list_xk[k] = x_k
    return x_k
N = 30000
beta = 0.01
alpha_i = 0.1
tau = 0.3
rho=0.1    # be careful when rho is large
V0=1000
X0 = np.array([12,18])
list_xk = [0]*N

print("Approximation after {} iterations".format(N))
print(F(J, GradJ, beta, alpha_i, tau, X0, N))
```

```
Approximation after 30000 iterations
```

```
[ 10.02099593  19.9063193 ]
```

In [3]:
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

def can(x,y):
    return x**2+x*y

x, y = np.meshgrid(np.linspace(8,13, 200), np.linspace(14,24, 200))
z = can(x,y)
graphe = plt.contour(x,y,z,10)

xk_1 = [list_xk[k][0] for k in range(N)]
xk_2 = [list_xk[k][1] for k in range(N)]
plt.plot(xk_1, xk_2, "b:o")

plt.clabel(graphe,inline=1,fontsize=10,fmt='%3.2f')
plt.title("contour line for the can surface and sequence of approximations")
plt.show()
```