Optim-Mauritius-2023-CS5

January 12, 2023

In [13]:
```
# simulated annealing for Rosenbrock
import math
import numpy as np

def SA(f,x0,Niter):
    n=len(x0)
    T=0.1
    alpha=0.9
    y0=f(x0)
    x=x0
    y=y0
    s=0.01
    icount=0
    for k in range (20):
        T=alpha*T
        accept=0
        for l in range (np.int(Niter/20)):
            icount=icount+1
            list_x[icount-1] = x0
            x1=x0+s*np.random.uniform(-0.5,0.5,size=n)
            y1=f(x1)
            dy=y1-y0
            if (np.random.uniform(0.,1.,size=1)<np.exp(-dy/T)):
                x0=x1
                y0=y1
                accept=accept+1
                if (y0<y):
                    x=x0;y=y0
        if (accept<25/100*Niter/20):
            s=s/2
        if (accept>75/100*Niter/20):
            s=2*s
    return x,icount
```

In [14]:
```
def Rosenbrock(X):
    x = X[0]
    y = X[1]
    return 100*(y-x**2)**2+(x-1)**2
```

```
In [15]: N=20000
         list_x = [0]*N
         X0=np.array([-1.2,1])
         n=2
         SA(Rosenbrock,X0,N)

Out[15]: (array([ 1.00100233,  1.00180691]), 20000)

In [17]: import matplotlib as mpl
         import matplotlib.pyplot as plt

         def RosenbrockB(x,y):
             return 100*(y-x**2)**2+(x-1)**2

         x, y = np.meshgrid(np.linspace(-0.5,1.5, 200), np.linspace(-0.5,1.5, 200))
         z = RosenbrockB(x,y)
         graphe = plt.contour(x,y,z,[1, 4,20,100])

         xk_1 = [list_x[k][0] for k in range(N)]
         xk_2 = [list_x[k][1] for k in range(N)]
         plt.plot(xk_1, xk_2, "b:o") #x_k pour methode pattern search

         plt.clabel(graphe,inline=1,fontsize=10,fmt='%3.2f')
         plt.title("Contour lines for Rosenbrock with the simulated annealing method")
         plt.show()
```