Pregunta 4

Solucion de minimos cuadrados con el metodo de Broyden

In [44]:

```
from numpy import dot,sqrt,array,eye
from numpy.linalg import solve
def broyden(f,x0,A0):
    TOL=1E-5
    MAXITER=10
    k=0
    xk=x0
    Ak=A0
    terminar=False
    while not terminar:
        fk=f(xk)
        absfk = sqrt(dot(fk.T,fk))
        if absfk<TOL or k>MAXITER :
            terminar=True
        else:
            sk = solve(Ak,-fk)
            xk1 = xk + sk
            fk1 = f(xk1)
            yk = fk1 - fk
            Ak += (1/dot(sk.T,sk)) * (yk - Ak@sk)@sk.T
            k +=1
            fk = fk1
            xk = xk1
            print (f"{k:<8d}{xk[0][0]:<8.5f} {xk[1][0]:<8.5f} {xk[2][0]:<8.5f} {xk[3][0]:<8.5f}")
    print(f"Metodo termino en {k} iteraciones, |f(x)|={absfk}")
    return xk
```

In [ ]:

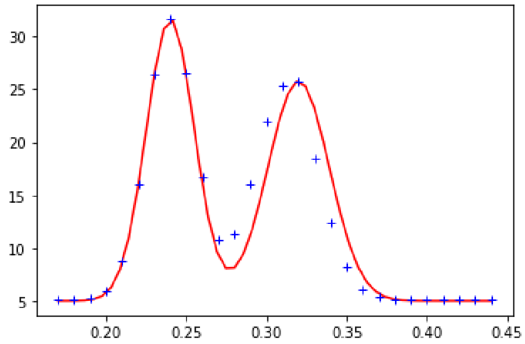
```
import numpy as np
def f(x):
    z = np.zeros_like(x)
    k = x[0];    a = x[1]
    t = x[2];    b = x[3]
    expk2 = np.exp(-k**2*(xx-a)**2)
    expt2 = np.exp(-t**2*(xx-b)**2)
    vv = yy-5-(k*expk2+t*expt2)/(np.pi**0.5)
    z[0] = np.dot(vv , ( (2*k**2)*((xx-a)**2) - 1)*expk2 )
    z[1] = np.dot(vv , (xx-a)* expk2 )
    z[2] = np.dot(vv , ( (2*t**2)*((xx-b)**2) - 1)*expt2 )
    z[3] = np.dot(vv , (xx-b)*expt2  )
    return z
def g(z,x):
    k = x[0];    a = x[1]
    t = x[2];    b = x[3]
    expk2 = np.exp(-k**2*(z-a)**2)
    expt2 = np.exp(-t**2*(z-b)**2)
    return 5+(k*expk2+t*expt2)/(np.pi**0.5)
```

In [ ]:
```
data = np.array([[0.17 , 5.10] ,
[0.18 , 5.10 ],
[0.19 , 5.20 ],
[0.20 , 5.87 ],
[0.21 , 8.72 ],
[0.22 , 16.04],
[0.23 , 26.35],
[0.24 , 31.63],
[0.25 , 26.51],
[0.26 , 16.68],
[0.27 , 10.80],
[0.28 , 11.26],
[0.29 , 16.05],
[0.30 , 21.96],
[0.31 , 25.31],
[0.32 , 25.79],
[0.33 , 18.44],
[0.34 , 12.45],
[0.35 , 8.22],
[0.36 , 6.12],
[0.37 , 5.35],
[0.38 , 5.15],
[0.39 , 5.10],
[0.40 , 5.10],
[0.41 , 5.09],
[0.42 , 5.09],
[0.43 , 5.09],
[0.44 , 5.09]] )
xx = data[:,0]
yy = data[:,1]
```

Aproximacion inicial

In [58]:
```
x0 = np.array([[(31.63-5)*np.pi**(0.5),0.24,(25.79-5)*np.pi**(0.5),0.32]]).T
import matplotlib.pyplot as plt
tt =np.linspace(xx[0],xx[-1],50)
plt.plot(tt,g(tt,x0),'r')
plt.plot(xx,yy,'b+')
```

Out[58]:
```
[<matplotlib.lines.Line2D at 0x7f025e27f550>]
```



Aproximacion del jacobiano

In [59]:
```
A0 = np.zeros((4,4), dtype='f4')
A0[0,0] = ((f(x0+np.array([1E-5,0,0,0]).reshape(4,1))-f(x0))/1E-5)[0,0]
A0[1,1] = ((f(x0+np.array([0,1E-5,0,0]).reshape(4,1))-f(x0))/1E-5)[1,0]
A0[2,2] = ((f(x0+np.array([0,0,1E-5,0]).reshape(4,1))-f(x0))/1E-5)[2,0]
A0[3,3] = ((f(x0+np.array([0,0,0,1E-5]).reshape(4,1))-f(x0))/1E-5)[3,0]

xk = broyden(f,x0,A0)
```

```
1        45.64301 0.24045  34.30439 0.31179
2        46.42729 0.24017  35.78734 0.31220
3        46.76573 0.24014  37.02009 0.31233
4        46.79409 0.24010  37.12587 0.31245
5        46.77598 0.24011  37.12652 0.31244
6        46.77689 0.24011  37.12704 0.31244
7        46.77672 0.24011  37.12729 0.31244
8        46.77673 0.24011  37.12730 0.31244
Metodo termino en 8 iteraciones, |f(x)|=[[2.34276684e-06]]
```

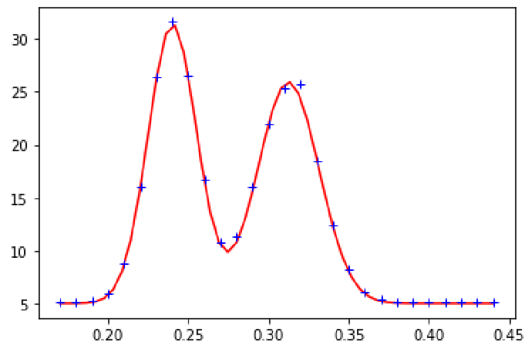
Grafica del ajuste

In [60]:

```
plt.plot(tt,g(tt,xk),'r')
plt.plot(xx,yy,'b+')
```

Out[60]:

```
[<matplotlib.lines.Line2D at 0x7f025e266890>]
```



Parametros ajustados

In [63]:

```
print(f"k={xk[0][0]:5.3f} a={xk[1][0]:5.3f} t={xk[2][0]:5.3f} b={xk[3][0]:5.3f}")
```

```
k=46.777 a=0.240 t=37.127 b=0.312
```

In [65]:

```
print(f"distancia entre picos : {abs(xk[3][0]-xk[1][0]):5.3f}")
```

```
distancia entre picos : 0.072
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: