In [15]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from mpl_toolkits.mplot3d import Axes3D
```

In [14]:

```
#%matplotlib inline
#%matplotlib notebook
#%pylab
```

The Rosenbrock function

We will work with the Rosenbrock function,

$$f(x,y) = (x-1)^2 + b\,(y-x^2)^2$$

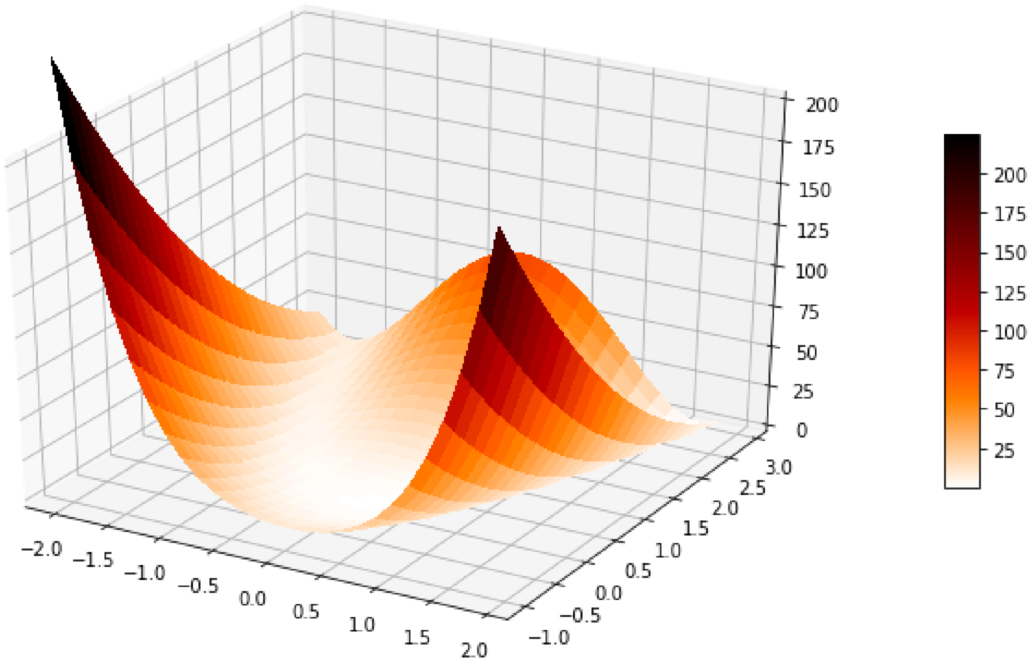
for the choice $b = 10$.

In [19]:

```
b = 10;
f = lambda x,y: (x-1)**2 + b*(y-x**2)**2;
```

In [74]:

```
# Initialize figure
figRos = plt.figure(figsize=(12, 7))
axRos = figRos.gca(projection='3d')

# Evaluate function
X = np.arange(-2, 2, 0.15)
Y = np.arange(-1, 3, 0.15)
X, Y = np.meshgrid(X, Y)
Z = f(X,Y)

# Plot the surface
surf = axRos.plot_surface(X, Y, Z, cmap=cm.gist_heat_r,
                          linewidth=0, antialiased=False)
axRos.set_zlim(0, 200)
figRos.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```



Gradient

The gradient of the Rosenbrock function is

$$\nabla f = \begin{pmatrix} 2(x-1) - 4b\,(y - x^2)\,x \\ 2b\,(y - x^2) \end{pmatrix}$$

In [20]:

```
df = lambda x,y: np.array([2*(x-1) - 4*b*(y - x**2)*x, \
                          2*b*(y-x**2)])
```

Optimization

In [21]:

```
F = lambda X: f(X[0],X[1])
dF = lambda X: df(X[0],X[1])
```

In [92]:

```
x0 = np.array([-1.4,1.1])
print(F(x0))
print(dF(x0))
```

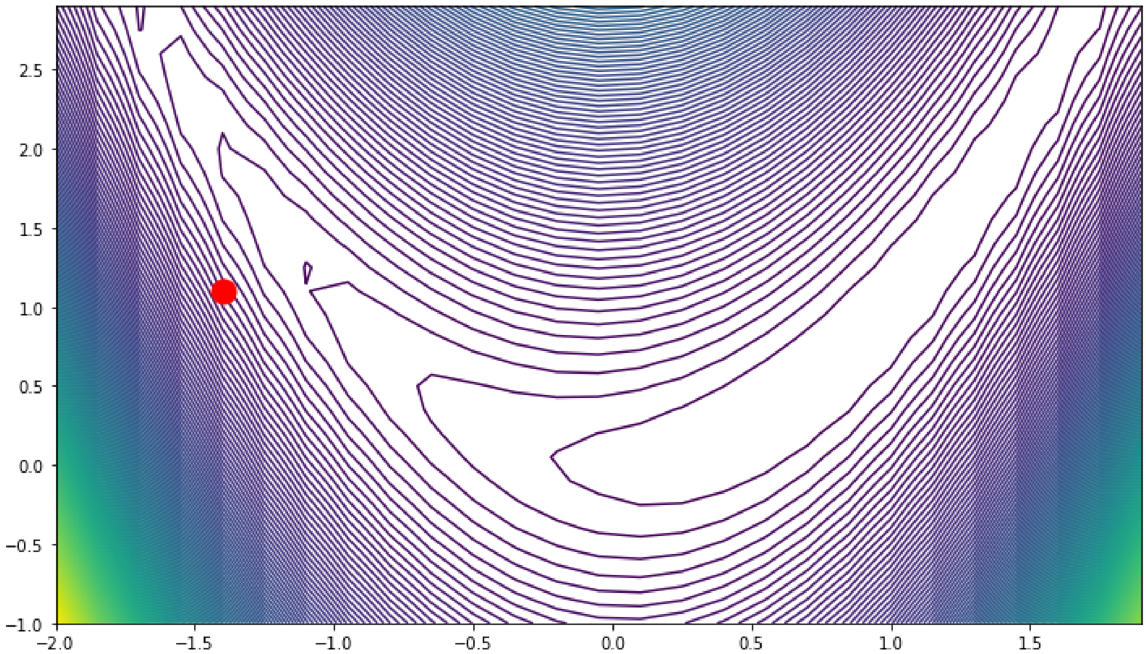
```
13.155999999999993
[-52.96 -17.2 ]
```

In [132]:

```
# Initialize figure
plt.figure(figsize=(12, 7))
plt.contour(X,Y,Z,200)
plt.plot([x0[0]],[x0[1]],marker='o',markersize=15, color ='r')
```

Out[132]:

```
[<matplotlib.lines.Line2D at 0x23be9ee1e50>]
```



Find a descent direction

In [150]:

```
fx = F(x0);
gx = dF(x0);
s = -gx;
print(s)
```

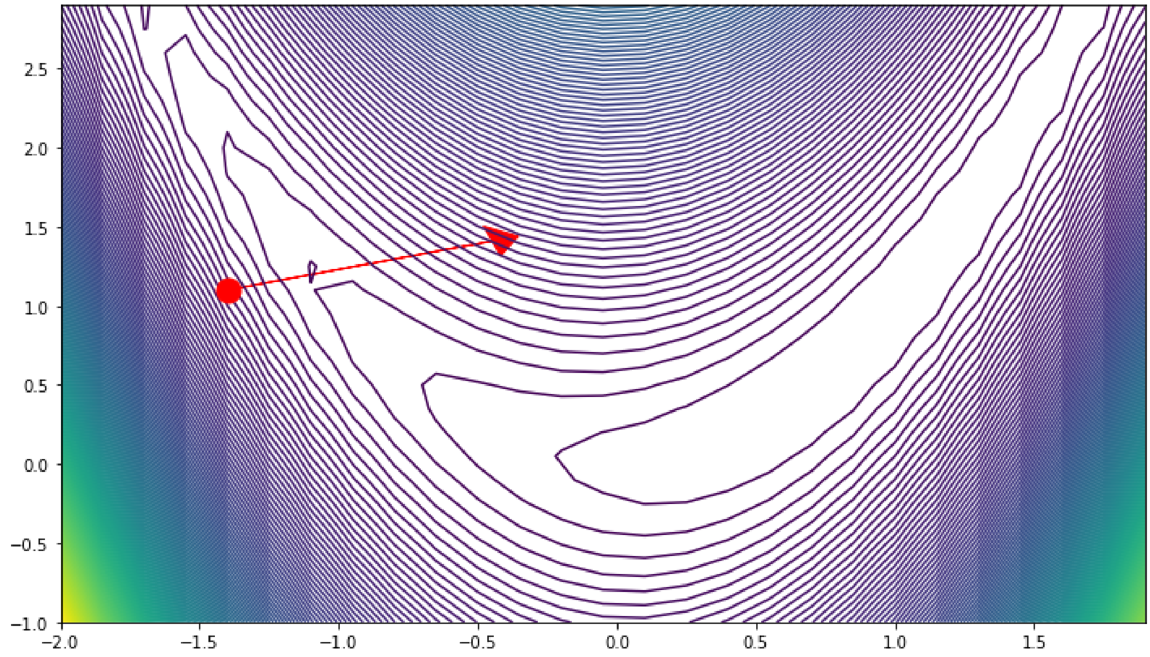
```
[52.96 17.2 ]
```

In [148]:

```
# Initialize figure
plt.figure(figsize=(12, 7))
plt.contour(X,Y,Z,200)
ns = np.sqrt(s[0]**2+s[1]**2);
plt.plot([x0[0]],[x0[1]],marker='o',markersize=15, color ='r')
plt.arrow(x0[0],x0[1],s[0]/ns,s[1]/ns, head_width=0.2, head_length=0.1, fc='r', ec='r')
```

Out[148]:

```
<matplotlib.patches.FancyArrow at 0x23bed73a220>
```

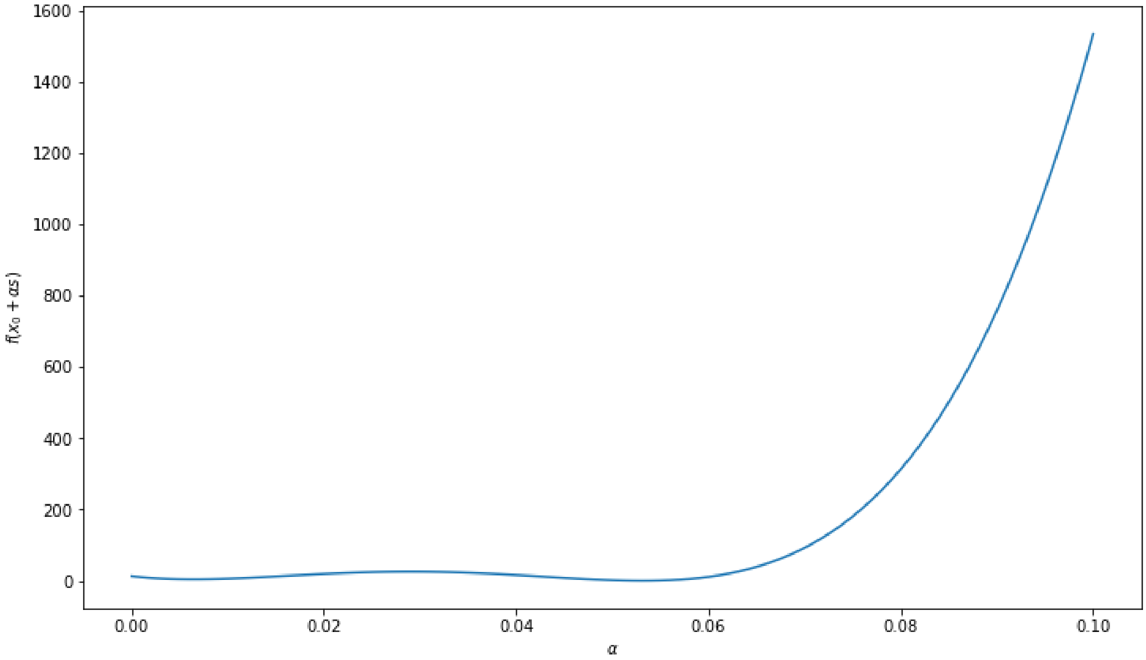


How far should we go along this direction?

Find α that minimizes $f(x_0 + \alpha s)$

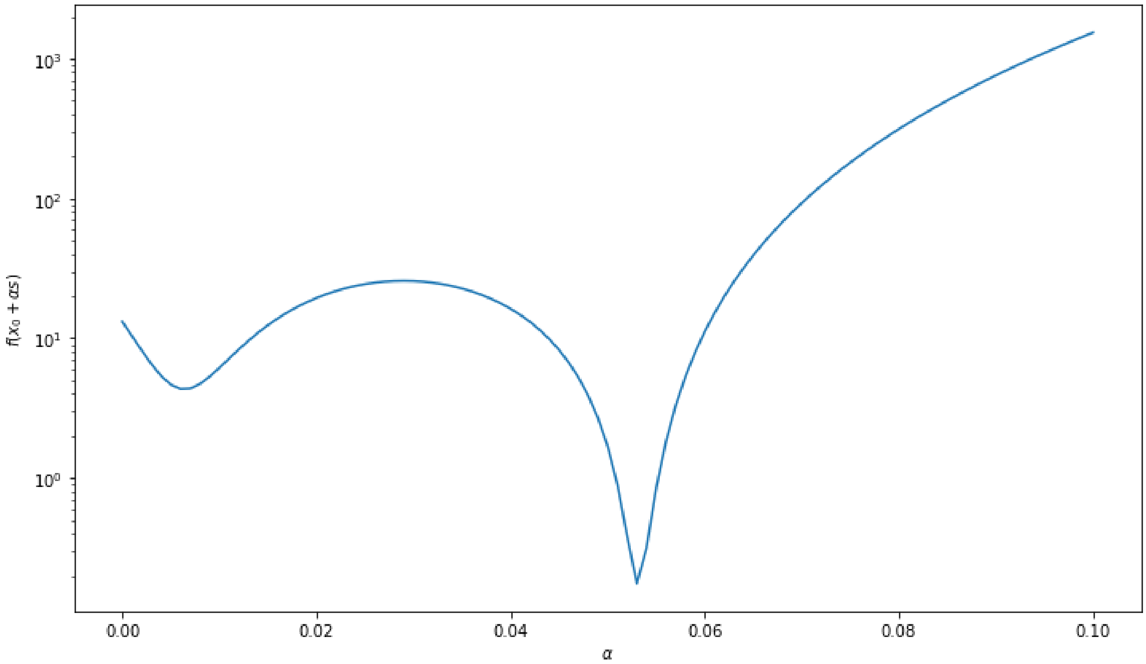
In [192]:

```
al = np.linspace(0,0.1,101)
z = [F(x0+a*s) for a in al]
figLS = plt.figure(figsize=(12, 7))
plt.plot(al,z)
plt.ylabel('$f(x_0+ \\alpha s)$')
plt.xlabel('$\\alpha$')
plt.show()
```



In [85]:

```
figLS = plt.figure(figsize=(12, 7))
plt.plot(al,z)
plt.yscale('log')
plt.ylabel('$f(x_0+ \\alpha s)$')
plt.xlabel('$\\alpha$')
plt.show()
```



In [180]:

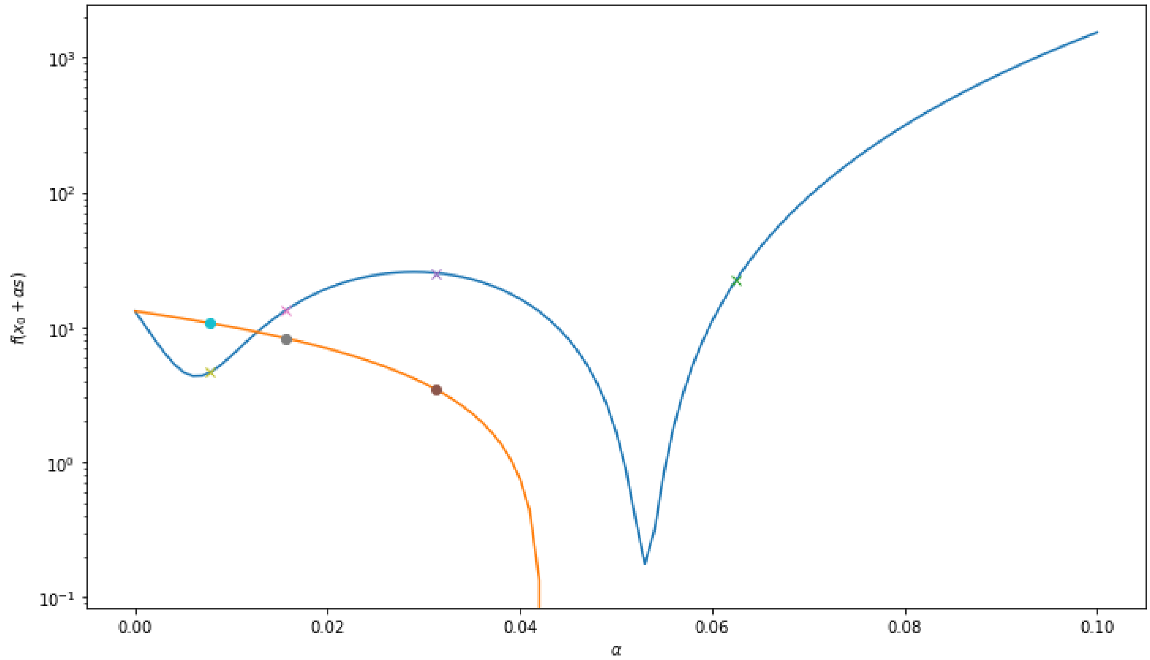
```
[fx,fx+0.01*d]
```

Out[180]:

```
[13.155999999999993, 10.055398399999996]
```

In [193]:

```
theta = 0.1
alpha = 1
tol = 1e-10
d = theta*np.dot(gx,s)

figLS1 = plt.figure(figsize=(12, 7))
plt.plot(al,z)
plt.plot(al,[fx+a*d for a in al])

for i in range(10):

    if (alpha<=0.1):
        plt.plot(alpha,F(x0+alpha*s),marker='x');
        plt.plot(alpha,fx + alpha*d,marker='o')

    if F(x0+alpha*s) < (fx + alpha*d):
        break;
    alpha = alpha/2;

plt.yscale('log')
plt.ylabel('$f(x_0+ \\alpha s)$')
plt.xlabel('$\\alpha$')
plt.show()
```



In [54]:

```
alpha
```

Out[54]:

```
0.0078125
```

In [ ]:

```

```