In [7]:
```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 26 21:39:15 2019

@author: jorge
"""

import sympy as sym
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
#get_ipython().run_line_magic('matplotlib', 'inline')
#https://scipy-lectures.org/packages/sympy.html
# ^^ how to use sympy ^^

height = 10
width = 10

mpl.rcParams['figure.figsize'] = (width, height)
mpl.rcParams['font.size'] = 18
mpl.rcParams['figure.titlesize'] = 'small'
mpl.rcParams['legend.fontsize'] = 'small'
mpl.rcParams['xtick.major.size'] = 12
mpl.rcParams['xtick.minor.size'] = 8
mpl.rcParams['xtick.labelsize'] = 18
mpl.rcParams['ytick.major.size'] = 12
mpl.rcParams['ytick.minor.size'] = 8
mpl.rcParams['ytick.labelsize'] = 18

HX ,HY  = 50,50 #number of x,y  points for countour
xmin,xmax = -1.5,1.5
ymin,ymax = -1.5,1.5
x1 = np.linspace(xmin,xmax,HX)
x2 = np.linspace(ymin,ymax,HY)
X1,X2 = np.meshgrid(x1,x2) # genertate mesh grid
w1=sym.Symbol('w1') # define symbols
w2=sym.Symbol('w2')

j=(1-w1)**2 + 100*(w2 - w1**2)**2# define equation

#compute gradient
j_grad1=sym.diff(j,w1)
j_grad2=sym.diff(j,w2)
#compute hessian
hess11=sym.diff(j_grad1,w1)
hess12=sym.diff(j_grad1,w2)
hess21=sym.diff(j_grad2,w1)
hess22=sym.diff(j_grad2,w2)

#generate contour map
ConMap=np.zeros((HX,HY))
for i in range(HX):
    for k in range(HY):
        ConMap[i,k]=j.subs({w1:x1[i],w2:x2[k]})


logetas_ls = np.arange(-5, 0, 0.5)
etas_ls = 10**logetas_ls
logetas = np.array([-4, -3, -2], dtype=np.float)
etas = 10**logetas
lim=50 # number of iterations
threshold = 1e-5

colors = ('lightgray', 'salmon', 'orange', 'black')
labels = [r'$\eta =$' + str(eta) for eta in etas]
labels.append('Exact Line Search')
```

In [8]:
```
plt.figure(figsize=(10,10))
plt.contour(X1,X2,ConMap)
for i in range(len(etas)+1):
    w=[-0.5,1] # starting point
    wStar=[1,1] # ending point
    ew=[]
    jw=[]
    count= 1
    line=[]
    while(True):
        #compute gradient matrix and hessian matrix
        g= np.array([float(j_grad1.subs({w1:w[0],w2:w[1]})),float(j_grad2.subs({w1:w
[0],w2:w[1]}))])
        H= np.array([[float(hess11.subs({w1:w[0],w2:w[1]})),float(hess12.subs({w1:w
[0],w2:w[1]}))],
                     [float(hess21.subs({w1:w[0],w2:w[1]})),float(hess22.subs({w1:w[0],
w2:w[1]}))]])

        # Exact Line Search
        dw = -g
        if i == (len(etas)):
            costs = np.zeros(len(etas_ls))
            for n, e in enumerate(etas_ls):
                costs[n] = j.subs({w1:w[0]+e*dw[0],w2:w[1]+e*dw[1]})
            eta = etas_ls[np.argmin(costs)]
        else:
            eta = etas[i]

        wnew =  w+eta*dw
        line.append(w)
        #loop check
        if(np.abs(wnew - w).mean() < threshold):
            print('Converge break')
            print(count)
            break
        elif( count>lim ):
            print('Count Break')
            break
        elif(np.isnan(g).any()):
            print('nan break')
            break
        else:
            count=count +1
            wprev=w.copy()

            w=wnew.copy()

            ew.append(np.linalg.norm(w-wStar))
            jw.append(j.subs({w1:w[0],w2:w[1]}))
    line=np.array(line)

    plt.plot(line[:,1],line[:,0], color=colors[i], label=labels[i])
    plt.scatter(line[:,1],line[:,0], 25, color=colors[i])

plt.plot(wStar[1],wStar[0], 'rx', markersize=8)
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.legend()
plt.tight_layout()
#plt.savefig('../prob5ab.eps', dpi=500)
```

```
Count Break
Count Break
nan break
Count Break
```