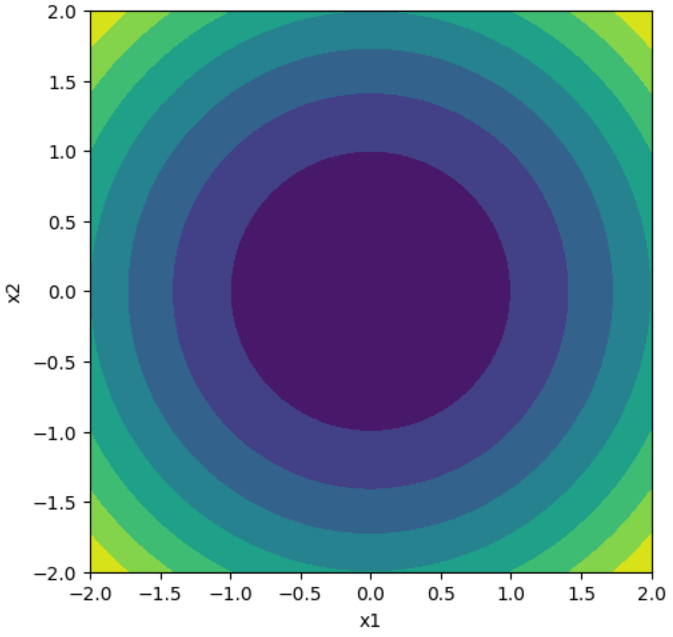
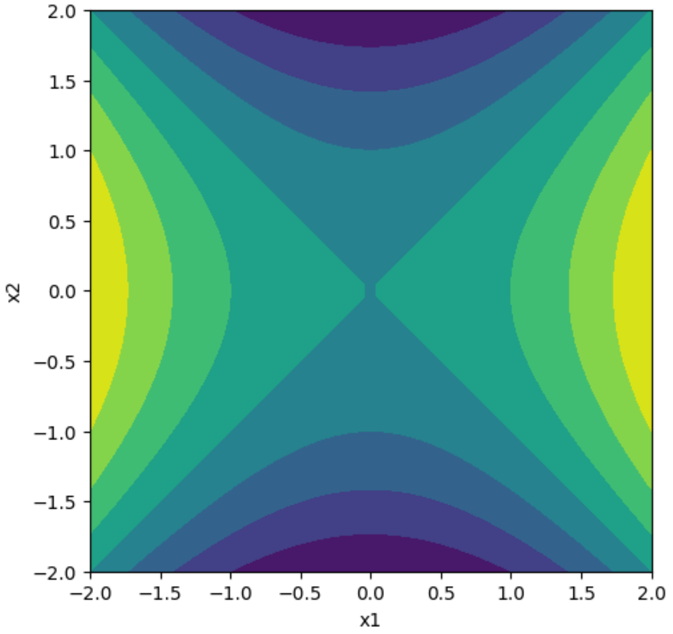
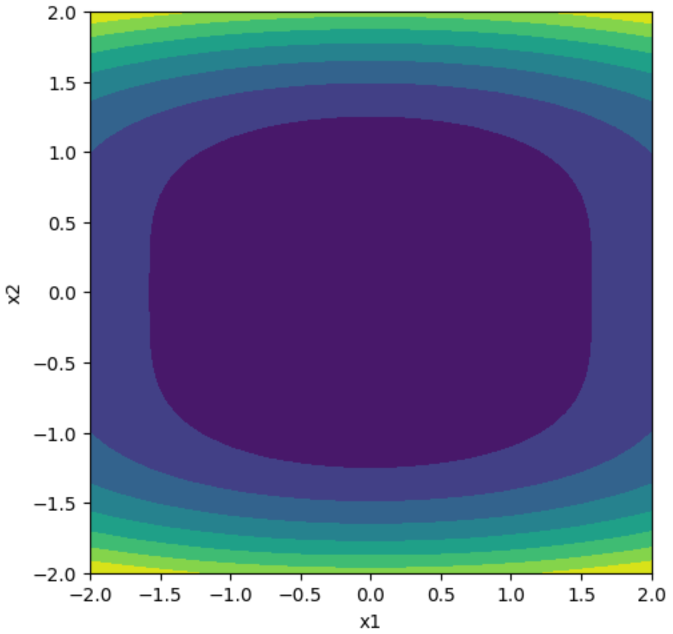
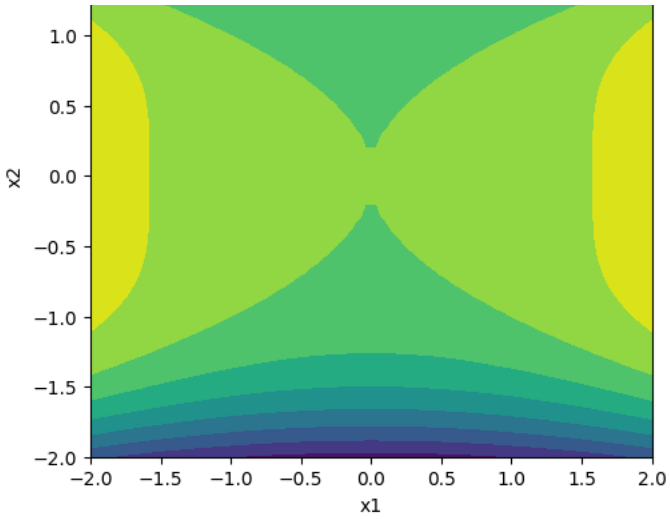
```
import numpy as np
import matplotlib.pyplot as plt


import numpy as np
import matplotlib.pylab as plt

def contour_plot(fobj):
    # Generate the data for a contour plot
    n = 50
    x1 = np.linspace(-2, 2, n)
    x2 = np.linspace(-2, 2, n)
    X1, X2 = np.meshgrid(x1, x2)
    f = np.zeros((n, n))

    # Query the function at the specified locations
    for i in range(n):
        for j in range(n):
            f[i, j] = fobj([X1[i, j], X2[i, j]])

    fig, ax = plt.subplots(1, 1)
    ax.contourf(X1, X2, f)
    ax.set_aspect('equal', 'box')
    fig.tight_layout()
    plt.xlabel('x1')
    plt.ylabel('x2')

    return

def f1(x):
    return x[0]**2 + x[1]**2

def f2(x):
    return x[0]**2 - x[1]**2

def f3(x):
    return x[0]**2 + x[1]**4

def f4(x):
    return x[0]**2 - x[1]**4

def f5(x):
    return x[1]**2

def f6(x):
    return x[0] + x[1]**2

def f7(x):
    return 2*x[0]**2 + 2*x[0]*x[1] + 0.5*x[1]**2

def f8(x):
    return 2*x[0]**2 + 2*x[0]*x[1] + 0.5*x[1]**2 + 2*x[0] + x[1]

def f9(x):
    return 2*x[0]**2 + 2*x[0]*x[1] + 0.5*x[1]**2 - x[0] + 2*x[1]

def f10(x):
    return 2*x[0]**2 + 2*x[0]*x[1] + x[1]**2 - x[0] + 2*x[1]

def f11(x):
    return -x[0]**2 -x[1]**2

def f12(x):
    return -x[0]**2

funcs = [f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12]

for fobj in funcs:
    contour_plot(fobj)

plt.show()
```

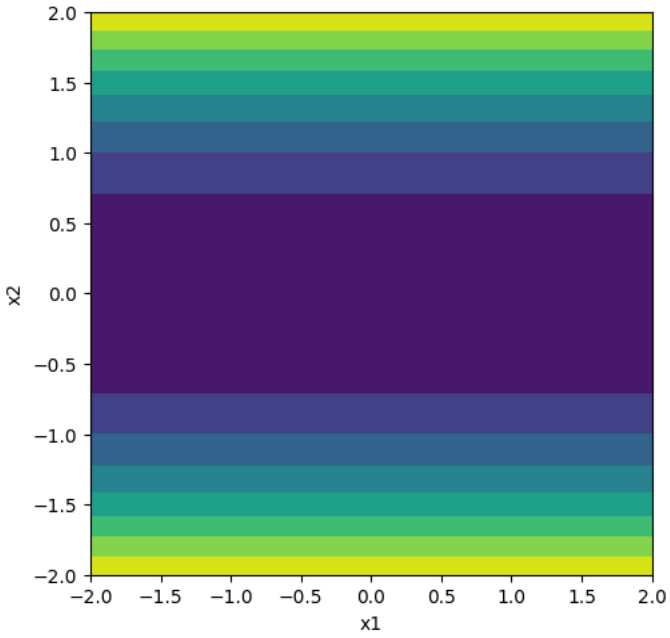

x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1

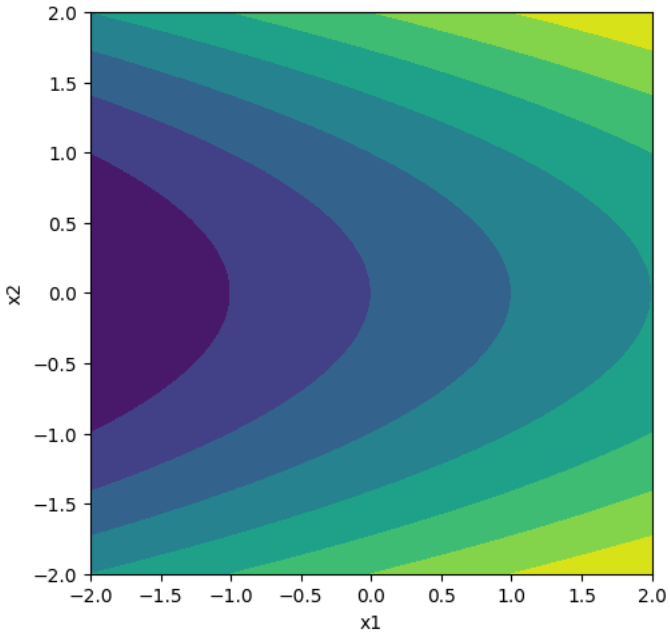

x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1


x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1

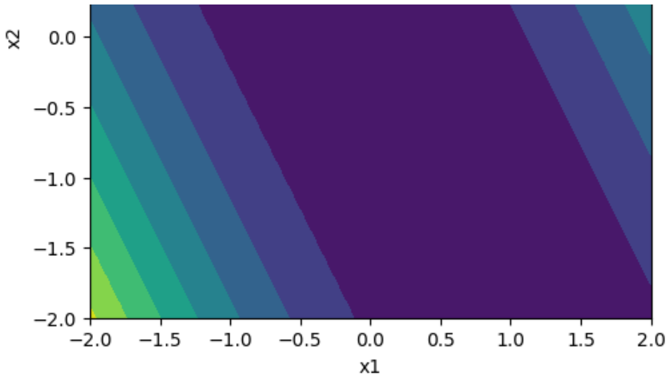

2.0
1.5

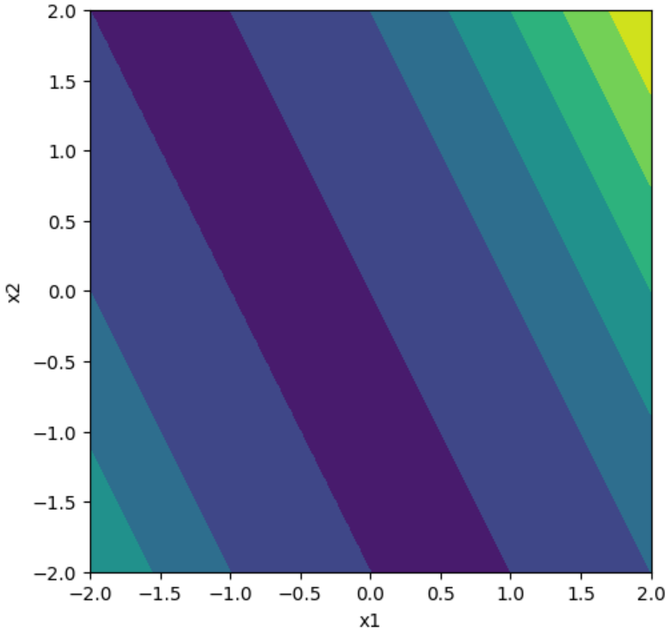

x2
x1
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

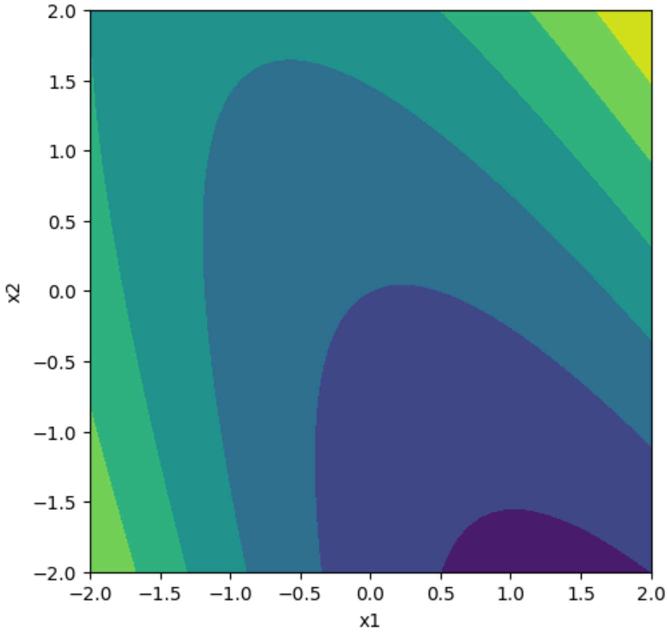

x2
x1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


x2
x1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

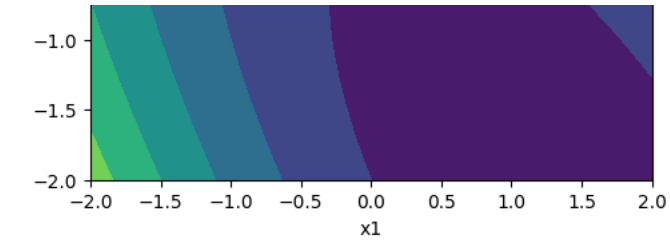

2.0
1.5
1.0
0.5

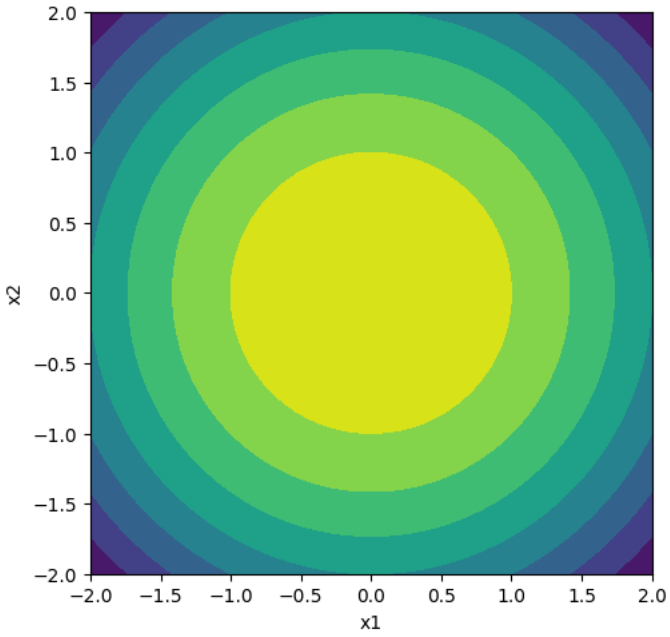

x2
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1

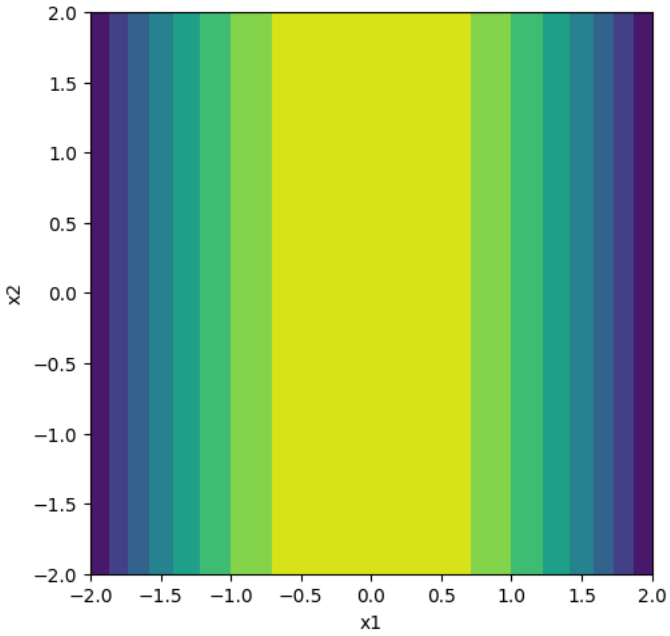

x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1


x2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
x1


x2
2.0
1.5
1.0
0.5
0.0
−0.5








```
def approx_gradient(x,fobj,h=1e-6):
  x=np.array(x)
  e1=np.array([1,0])
  e2=np.array([0,1])
  g=np.array([0.5*(fobj(x + h*e1)-fobj(x-h*e1))/h,
              0.5*(fobj(x + h*e2)-fobj(x-h*e2))/h])
  return g

def approx_hessian(x, fobj, h=1e-6):
  x=np.array(x)
  e1=np.array([1,0])
  e2=np.array([0,1])
  H =np.array([[(fobj(x + h*e1)-2*fobj(x)+fobj(x-h*e1))/h**2,
                0.25*(fobj(x + h*(e1 + e2)) -
                      fobj(x + h*(e1 - e2)) -
                      fobj(x + h*(e2 - e1)) +
                      fobj(x + h*(e1 + e2)))/h**2],
               [0, (fobj(x + h*e2) - 2*fobj(x) + fobj(x - h*e2))/h**2]])
  H[1,0] = H[0,1]
  return H
```

```
x0 = [0, 0]
for index, fobj in enumerate(funcs):
    print('Testing function %d at the point' % index, x0)
    g = approx_gradient(x0, fobj)

    if np.sqrt(np.dot(g, g)) < 1e-4:
        print('First-order necessary condition satisfied for', fobj)

        H = approx_hessian(x0, fobj)
        eig, Q = np.linalg.eigh(H)

        print('Eigenvalues:', eig)
        if eig[0] > 1e-4:
            print('Second order sufficient conditions satisfied for', fobj)

        elif eig[0] < -1e-4:
            print('Second order sufficient conditions violated for', fobj)

        else:
            print('Indefinite or numerical problems?', eig[0])

    else:
        print('First-order necessary condition *violated* for', fobj)
    print('')
```

```
Testing function 0 at the point [0, 0]
First-order necessary condition satisfied for <function f1 at 0x7ab96a455120>
Eigenvalues: [2. 2.]
Second order sufficient conditions satisfied for <function f1 at 0x7ab96a455120>

Testing function 1 at the point [0, 0]
First-order necessary condition satisfied for <function f2 at 0x7ab96a455870>
Eigenvalues: [-2.  2.]
Second order sufficient conditions violated for <function f2 at 0x7ab96a455870>

Testing function 2 at the point [0, 0]
First-order necessary condition satisfied for <function f3 at 0x7ab96a455480>
Eigenvalues: [2.e-12 2.e+00]
Indefinite or numerical problems? 1.9999999999999996e-12

Testing function 3 at the point [0, 0]
First-order necessary condition satisfied for <function f4 at 0x7ab96a455510>
Eigenvalues: [-2.e-12  2.e+00]
Indefinite or numerical problems? -1.9999999999999996e-12

Testing function 4 at the point [0, 0]
First-order necessary condition satisfied for <function f5 at 0x7ab96a455360>
Eigenvalues: [0. 2.]
Indefinite or numerical problems? 0.0

Testing function 5 at the point [0, 0]
First-order necessary condition *violated* for <function f6 at 0x7ab96a455630>

Testing function 6 at the point [0, 0]
First-order necessary condition satisfied for <function f7 at 0x7ab96a4556c0>
Eigenvalues: [0. 5.]
Indefinite or numerical problems? 0.0

Testing function 7 at the point [0, 0]
First-order necessary condition *violated* for <function f8 at 0x7ab96a455000>

Testing function 8 at the point [0, 0]
First-order necessary condition *violated* for <function f9 at 0x7ab96a455bd0>

Testing function 9 at the point [0, 0]
First-order necessary condition *violated* for <function f10 at 0x7ab96a455b40>

Testing function 10 at the point [0, 0]
First-order necessary condition satisfied for <function f11 at 0x7ab96a455750>
Eigenvalues: [-2. -2.]
Second order sufficient conditions violated for <function f11 at 0x7ab96a455750>

Testing function 11 at the point [0, 0]
First-order necessary condition satisfied for <function f12 at 0x7ab96a455d80>
Eigenvalues: [-2. -0.]
Second order sufficient conditions violated for <function f12 at 0x7ab96a455d80>
```