Congratulations! You passed!

Grade received 100% To pass 80% or higher

Go to next item

1. In this quiz we will consider Lagrange multipliers as a technique to find a minimum of a function subject to a constraint, i.e. solutions lying on a particular curve. 1 / 1 point

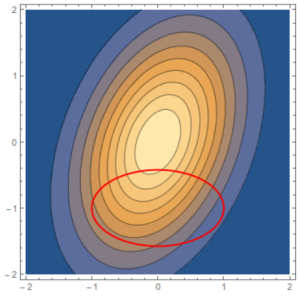
Let's consider the example of finding the minimum of the function,

$$f(\mathbf{x}) = \exp\left(-\frac{2x^2 + y^2 - xy}{2}\right)$$

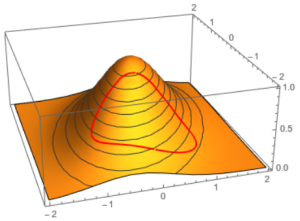
along the curve (or, subject to the constraint),

$$g(\mathbf{x}) = x^2 + 3(y+1)^2 - 1 = 0\ .$$

The functions themselves are fairly simple, on a contour map they look as follows,



However, their combination can become quite complicated if they were computed directly, as can be inferred from the shape of the constraint on the surface plot,



Do note, in this case, the function $f(\mathbf{x})$ does not have any minima itself, but along the curve, there are two minima (and two maxima).

A situation like this is where Lagrange multipliers come in. The observation is that the maxima and minima on the curve, will be found where the constraint is parallel to the contours of the function.

Since the gradient is perpendicular to the contours, the gradient of the function and the gradient of the constraint will also be parallel, that is,

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$$

If we write this out in component form, this becomes,

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \lambda \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix}$$

This equation, along with $g(\mathbf{x}) = 0$ is enough to specify the system fully. We can put all this information into a single vector equation,

$$\nabla\mathcal{L}(x, y, \lambda) = \begin{bmatrix} \frac{\partial f}{\partial x} - \lambda\frac{\partial g}{\partial x} \\ \frac{\partial f}{\partial y} - \lambda\frac{\partial g}{\partial y} \\ -g(\mathbf{x}) \end{bmatrix} = 0$$

and then solve this equation to solve the system.

Let's reflect on what we have done here, we have converted from a question of finding a minimum of a 2D function constrained to a 1D curve, to finding the zeros of a 3D vector equation.

Whereas this may sound like we have made the problem more complicated, we are exactly equipped to deal with this kind of problem; we can use the root finding methods, such as the Newton-Raphson method that we've discussed previously.

Let's set up the system,

The function and two of the derivatives are defined for you. Set up the other two by replacing the question marks in the following code.

```
# First we define the functions,
def f (x, y) :
    return np.exp(-(2*x*x + y*y - x*y) / 2)

def g (x, y) :
    return x*x + 3*(y+1)**2 - 1

# Next their derivatives,
def dfdx (x, y) :
    return 1/2 * (-4*x + y) * f(x, y)

def dfdy (x, y) :
    return 1/2 * (x - 2*y) * f(x, y)

def dgdx (x, y) :
    return 2 * x

def dgdy (x, y) :
    return 6 * (y+1)
```

Run

Reset

No Output

Correct

Well done.

2. Next let's define the vector, $\nabla\mathcal{L}$, that we are to find the zeros of; we'll call this "DL" in the code. Then we can use a pre-written root finding methodin scipy to solve. **1 / 1 point**

```
from scipy import optimize

def DL (xyλ) :
    [x, y, λ] = xyλ
    return np.array([
            dfdx(x, y) - λ * dgdx(x, y),
            dfdy(x, y) - λ * dgdy(x, y),
            - g(x, y)
        ])

(x0, y0, λ0) = (-1, -1, 0)
x, y, λ = optimize.root(DL, [x0, y0, λ0]).x
print("x = %g" % x)
print("y = %g" % y)
print("λ = %g" % λ)
print("f(x, y) = %g" % f(x, y))
```

Run

Reset

```
x = -0.958963
y = -1.1637
λ = -0.246538
f(x, y) = 0.353902
```

Here, the first two elements of the array are the x and y coordinates that we wanted to find, and the last element is the Lagrange multiplier, which we can throw away now it has been used.

Check that (x, y) does indeed solve the equation $g(x, y) = 0$.

You should be able to use the code find the other roots of the system.

Re-use the code above with different starting values to find the other stationary points on the constraint.

There are four in total. Give the y coordinate of any of the other solutions to two decimal places.

-0.43

Correct
This is one of the maxima or minima.

3. In the previous question, you gave the y coordinate of *any* of the stationary points. In this part, give the x coordinate of the *global minimum* of $f(x)$ on $g(x) = 0$. **1 / 1 point**

Give your answer to 2 decimal places.

0.93

Correct
Eactly this is the global minimum, subject to the constraint.

4. You may be wondering about why the vector $\nabla\mathcal{L}$ gets a funny symbol. This is because it can be written as the gradient (over $x, y,$ and λ) of a scalar function $\mathcal{L}(x, y, \lambda)$. **1 / 1 point**

Based on your knowledge of derivatives, what function $\mathcal{L}(x, y, \lambda)$ would give the expected form of $\nabla\mathcal{L}$?

- (●) $\mathcal{L}(x, y, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$
- ( ) $\mathcal{L}(x, y, \lambda) = f(\mathbf{x}) + g(\mathbf{x}) + \lambda$
- ( ) $\mathcal{L}(x, y, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}) + \lambda$
- ( ) $\mathcal{L}(x, y, \lambda) = f(x, y) + g(x, y, \lambda)$

Correct
Exactly, this is also why the last component of $\nabla\mathcal{L}$ has a minus sign.

5. Hopefully you've now built up a feeling for how Lagrange multipliers work. Let's test this out on a new function and constraint. **1 / 1 point**

Calculate the minimum of

$$f(x, y) = -\exp(x - y^2 + xy)$$

on the constraint,

$$g(x, y) = \cosh(y) + x - 2 = 0$$

Use the code you've written in the previous questions to help you.

```python
# Import libraries
import numpy as np
from scipy import optimize

# First we define the functions, YOU SHOULD IMPLEMENT THESE
def f (x, y) :
    return -np.exp(x - y*y + x*y)

def g (x, y) :
    return np.cosh(y)+x-2

# Next their derivatives, YOU SHOULD IMPLEMENT THESE
def dfdx (x, y) :
    return (-np.exp(x - y*y + x*y))*(1+y)

def dfdy (x, y) :
    return (-np.exp(x - y*y + x*y))*(-2*y+x)

def dgdx (x, y) :
    return 1

def dgdy (x, y) :
    return np.sinh(y)

# Use the definition of DL from previously.
def DL (xyλ) :
    [x, y, λ] = xyλ
    return np.array([
            dfdx(x, y) - λ * dgdx(x, y),
            dfdy(x, y) - λ * dgdy(x, y),
            - g(x, y)
        ])

```

```python
# To score on this question, the code above should set
# the variables x, y, λ, to the values which solve the
# Langrange multiplier problem.

# I.e. use the optimize.root method, as you did previously.
(x0, y0, λ0) = (0, 0, 0)
x, y, λ = optimize.root(DL, [x0, y0, λ0]).x
```

Run

Reset

```
x = 0.957782
y = 0.289565
λ = -4.07789
f(x, y) = -3.16222
```

Correct

Well done. You've constucted and run your own Lagrange multiplyer solver for a new function.