

Your grade: 80%

Your latest: 80% • Your highest: 80% • To pass you need at least 80%. We keep your highest score.

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

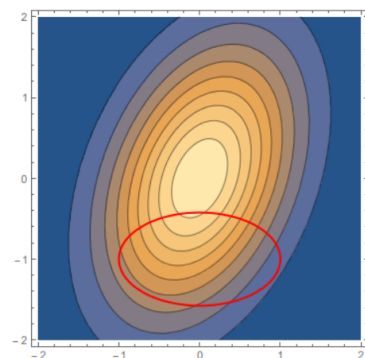
Let's consider the example of finding the minimum of the function,

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

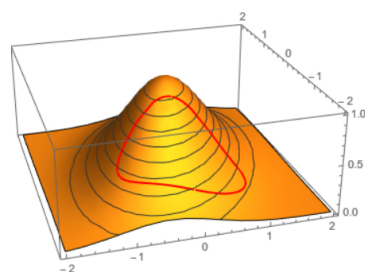
along the curve (or, subject to the constraint),

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

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.

marks in the following code.

```
1 # First we define the functions,
2 def f(x, y):
3     return np.exp(-(2*x*x + y*y - x*y) / 2)
4
5 def g(x, y):
6     return x*x + 3*(y+1)**2 - 1
7
8 # Next their derivatives,
9 def dfdx(x, y):
10    return 1/2 * (-4*x + y) * f(x, y)
11
12 def dfdy(x, y):
13    return 1/2 * (x - 2*y) * f(x, y)
14
15 def dgdx(x, y):
16    return 2*x * g(x, y)
17
18 def dgdy(x, y):
19    return 6*(y+1)
```

Run

Reset

✓ 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 method in scipy to solve.

1 / 1 point

```
1 from scipy import optimize
2
3 def DL(xyλ):
4     [x, y, λ] = xyλ
5     return np.array([
6         dfdx(x, y) - λ * dgdx(x, y),
7         dfdy(x, y) - λ * dgdy(x, y),
8         -g(x, y)
9     ])
10
11 # Initial guesses for finding multiple solutions
12 initial_guesses = [(-1, -1, 0), (1, 1, 0), (-1, 1, 0), (1, -1, 0)]
13 solutions = []
14
15 # Solve for each initial guess
16 for guess in initial_guesses:
17     sol = optimize.root(DL, guess)
18     x, y, λ = sol.x
19     solutions.append((x, y, f(x, y))) # Store (x, y, f(x, y))
20     print(f'Solution from initial guess {guess}: x = {x:.2f}, y = {y:.2f}, f(x, y) = {f(x, y):.2f}')
21
22 # Find the solution with the minimum f(x, y)
23 global_min_solution = min(solutions, key=lambda sol: sol[2])
24 x_min, y_min, f_min = global_min_solution
25 print(f'\nGlobal minimum of f(x, y): f(x, y) = {f_min:.2f} at x = {x_min:.2f}, y = {y_min:.2f}')
```

Run

Reset

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.

-1.21

✓ 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
Exactly 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.

0 / 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.

```

8 # Define the constraint function g(x, y)
9 def g(x, y):
10     return np.cosh(y) + x - 2
11
12 # Define the partial derivatives of f
13 def dfdx(x, y):
14     return -np.exp(x - y**2 + x*y) * (1 + y)
15
16 def dfdy(x, y):
17     return -np.exp(x - y**2 + x*y) * (-2*y + x)
18
19 # Define the partial derivatives of g
20 def dgdx(x, y):
21     return 1
22
23 def dgdy(x, y):
24     return np.sinh(y)
25
26 # Define the system of equations for the Lagrange multiplier problem
27 def DL(xyλ):
28     x, y, λ = xyλ
29     return np.array([
30         dfdx(x, y) - λ * dgdx(x, y),
31         dfdy(x, y) - λ * dgdy(x, y),
32         -g(x, y)
33     ])
34
35 # Initial guess for [x, y, λ]
36 initial_guess = [-1.0, -1.0, 0]
37
38 # Solve the system using scipy.optimize.root
39 solution = root(DL, initial_guess)
40 x, y, λ = solution.x
41
42 # Output the results
43 print("x = %g" % x)
44 print("y = %g" % y)
45 print("λ = %g" % λ)
46 print("f(x, y) = %g" % f(x, y))
47

```

Run

Reset

✗ Incorrect

Check your code carefully, particularly your function and derivatives, you're not reproducing the correct values.