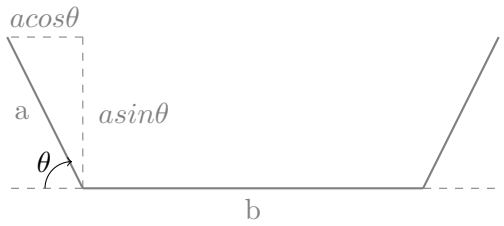
Unconstrained Optimization Assignment Report

CS164 - Professor Shekhar, R.

Ash Nguyen

2018/12/06



1. Expression And Estimation

The figure above illustrates the cross-sectional area of the drainage channel.

We can divide the cross-sectional area into three parts: the rectangle with dimension $asin\theta \times b$ and two right triangles with right angle sides of $asin\theta$ and $acos\theta$ as illustrated above. The total cross-sectional area is

$$A_{cross} = basin\theta + \frac{1}{2}2a^2 sin\theta cos\theta = basin\theta + a^2 sin\theta cos\theta$$

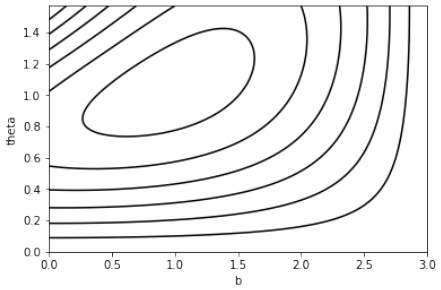


Figure 1: Contour Plot of A_{cross}

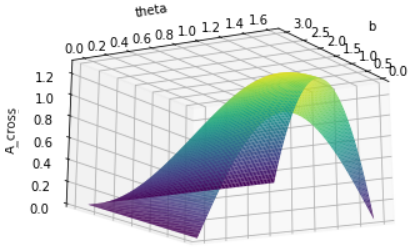


Figure 2: Surface Plot of A_{cross}

But we also have $2a + b = W = 3$ so $a = \frac{3-b}{2}$, so the above equation becomes:

$$A_{cross} = b\frac{3-b}{2}sin\theta + \left(\frac{3-b}{2}\right)^2 sin\theta cos\theta$$

Based on both the contour plot and the 3D surface plot of the expression for the cross-sectional, we can make a rough estimation that the values which will maximize the area is around $b = 1$ (unit) and $\theta = 1$ (radian). This optimal point appears to be unique over the considered domain, as the contour plot (of the function over the given domain) does not show any other critical point beside $(1, 1)$ (however, if we zoom out a little bit, we can see that (3,0) is also a critical point, in which we will find in the next section). Moreover, the surface plot clearly indicates the highest (maximum) points is global over the given domain, as there is no peak in the surface that is higher than the peak of the optimal point we estimated.

2. Analytical Solution

To find the values of b and θ that maximize the cross-sectional area, we need to find the critical point(s) of the function

$$A_{cross} = b\frac{3-b}{2}sin\theta + \left(\frac{3-b}{2}\right)^2 sin\theta cos\theta$$

over the given domain. But since we know that the optimal point is unique over the given domain, we will treat this as an unconstrained optimization problem, and then restrict our results and following analysis under the given constraints of b and θ for them to be physically possible.

The first partial derivatives of the objective function are given by:

$$\frac{\partial A_{cross}}{\partial b} = sin\theta\left(\frac{3}{2} - b - \frac{3}{2}cos\theta + \frac{1}{2}bcos\theta\right) \tag{1}$$

$$\frac{\partial A_{cross}}{\partial \theta} = \frac{3-b}{2}\left(bcos\theta + \frac{3-b}{2}cos^2\theta - \frac{3-b}{2}sin^2\theta\right) \tag{2}$$

For b_0 and θ_0 to be critical points of the objective function, they need to be roots of (1) and (2). Therefore, we have:

$$\begin{cases} sin\theta\left(\frac{3}{2} - b - \frac{3}{2}cos\theta + \frac{1}{2}bcos\theta\right) = 0 \\ \frac{3-b}{2}\left(bcos\theta + \frac{3-b}{2}cos^2\theta - \frac{3-b}{2}sin^2\theta\right) = 0 \end{cases}$$

Solving the first equation, we have $sin\theta = 0$ or $\frac{3}{2} - b - \frac{3}{2}cos\theta + \frac{1}{2}bcos\theta = 0$.

(*) For $sin\theta = 0$:

Over our domain $b \in [0, W]$ and $\theta \in [0, \pi/2]$, we must have $\theta = 0$ to satisfy $sin\theta = 0$. Plug $\theta = 0$ into the second equation, we have:

$$\begin{aligned} & \frac{3-b}{2}\left(bcos\theta + \frac{3-b}{2}cos^2\theta - \frac{3-b}{2}sin^2\theta\right) = 0 \\ \Leftrightarrow \quad & \frac{3-b}{2}\left(bcos(0) + \frac{3-b}{2}cos^2(0) - \frac{3-b}{2}sin^2(0)\right) = 0 \\ \Leftrightarrow \quad & \frac{3-b}{2}\left(b + \frac{3-b}{2}\right) = 0 \end{aligned}$$

The above expression yields $b = 3$ or $b = -3$, the latter not being in our domain. So in this case the root is $b = 3$, $\theta = 0$.

(**) For $\frac{3}{2} - b - \frac{3}{2}cos\theta + \frac{1}{2}bcos\theta = 0$:

With some derivation, we can arrive at

$$cos\theta = \frac{3-2b}{3-b}$$

at which point, notice that $cos^2\theta - sin^2\theta = 2cos^2\theta - 1$, plug $cos\theta$ into the second equation in our initial set of equations, we have:

$$\frac{3-b}{2}\left[b\frac{3-2b}{3-b} + \frac{3-b}{2}\left(2\frac{(3-2b)^2}{(3-b)^2} - 1\right)\right] = 0$$

With $b = 3$, plug in the first equation and we will see that $sin\theta = 0$, as the previous root. With $b \neq 0$, we have:

$$\begin{aligned} & b\frac{3-2b}{3-b} + \frac{3-b}{2}\left(2\frac{(3-2b)^2}{(3-b)^2} - 1\right) = 0 \\ \Leftrightarrow \quad & \frac{3(b^2-4b+3)}{3-b} = 0 \\ \Leftrightarrow \quad & b - 1 = 0 \\ \Leftrightarrow \quad & b = 1 \end{aligned}$$

With $b = 1$, we plug into $cos\theta = \frac{3-2b}{3-b}$ and have

$$cos\theta = \frac{1}{2}$$

or

$$\theta = \frac{\pi}{3}$$

Overall, we have two roots: $b = 3$ and $\theta = 0$ or $b = 1$ and $\theta = \pi/3$. Evaluate the objective function at these points, we conclude that $b = 1$ and $\theta = \pi/3$ maximizes our function, which results in the value of $A_{cross} = \frac{\sqrt{3}}{2}$ (the other point being the minimum). These dimensions for the optimality are physically constructable.

Alternatively, we can evaluate the Hessian matrix of the objective function to see that the eigenvalues of the Hessian at the second critical points are negative, so the matrix is negatively definite, and therefore this constitutes a relative maximum. And because the function is periodic and we are looking only at a specific domain, by the surface plot we know that this is a global maximum in the given domain.

3. Numerical Solution

The gradient descent with exact line search converges in 230 steps, producing the optimality:

$$[b, \theta] = [1.0016746625052915, 1.0486471819458436]$$

which is indeed very close to our analytical solution (with θ in radian). The convergence plot is given by figure 3.

The gradient descent with fixed step $\alpha = 0.1$ produces the same optimality ($[b, \theta] = [1.0016745735256598, 1.0486471515530362]$), converging in 354 steps. This is slower than with line search, which is to be expected since line search will choose the best possible step size at each iteration, results in faster convergence. However, in actual implementation, the gradient descent with line search is slower, since it's more computational costly to compute the step size in each iteration than choosing a fixed one. The convergence plot for gradient descent with fixed step size is given by figure 4.

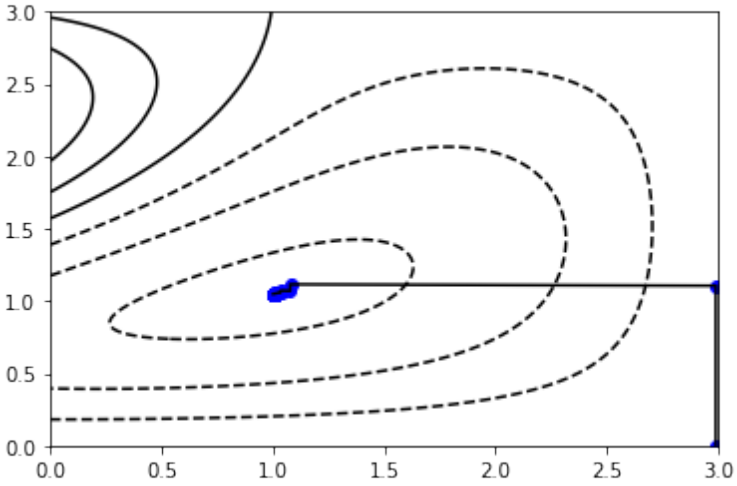


Figure 3: Convergence plot for gradient descent with exact line search

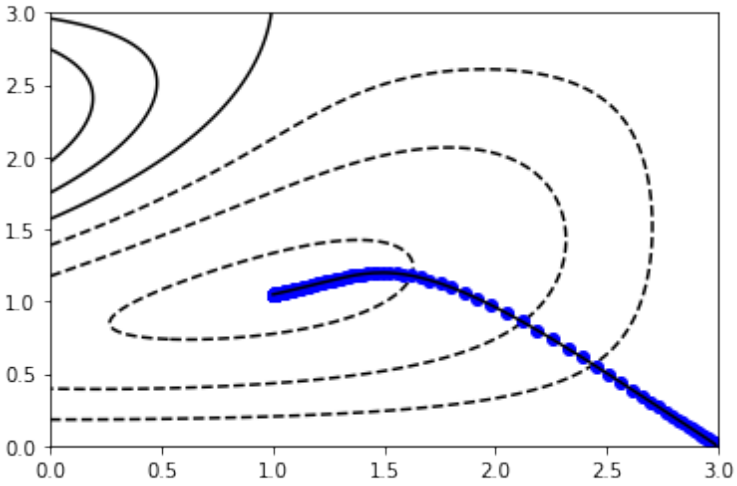


Figure 4: Convergence plot for gradient descent with fixed step size

A. Python Code For Surface Plotting

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import math

def f(x, y):
    return((3*x-x**2)/2*np.sin(y)+(3-x)**2/4*np.sin(y)*np.cos(y))



x = np.linspace(-3, 3, 100)
y = np.linspace(-3, math.pi/2, 100)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.set_xlabel('b')
ax.set_ylabel('theta')
ax.set_zlabel('A_cross');
ax.view_init(30,50)
```

B. Python Code For Contour Plotting

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

def f(x, y):
    return((3*x-x**2)/2*np.sin(y)+(3-x)**2/4*np.sin(y)*np.cos(y))


x = np.linspace(0, 3, 1000)
y = np.linspace(0, math.pi/2, 1000)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)


plt.contour(X, Y, Z, colors='black')
plt.xlabel("b")
plt.ylabel("theta")
plt.show()
```

C. Python Code For Gradient Descent With Exact Line Search

```
from scipy.misc import derivative
import matplotlib.pyplot as plt
import numpy as np
import math

def obfun(x1, x2):
    return(-((3*x1-x1**2)/2)*np.sin(x2)-((3-x1)**2/4)*np.sin(x2)*np.cos(x2))
def pd(fun, var=0, val=[]):
    arg = val[:]
    def wrap(x):
        arg[var] = x
        return fun(*arg)
    return(derivative(wrap, val[var], dx = 1e-1))
def funbi(alpha,vals,par):
    return(obfun(vals[0]+alpha*par[0],vals[1]+alpha*par[1]))
def exactlinesearch(vals,par):
    d=math.inf
    alpha=0.1
    while abs(d)>0.0001:
        d=-pd(funbi,0,[alpha,vals,par])
        al=0.1
        alpha=alpha+al*d
    return(alpha)
def gradient(inis):
    step=0
    d=[math.inf,math.inf]
    X1=[]
    X2=[]
    while abs(d[0])>0.000001 or abs(d[1])>0.000001:
        if (step % 10)==0:
            print('At step:', step)
        X1.append(inis[0])
        X2.append(inis[1])
        d=[-pd(obfun,0,inis),-pd(obfun,1,inis)]
        alpha=exactlinesearch(inis,d)
        inis[0],inis[1]=inis[0]+alpha*d[0],inis[1]+alpha*d[1]
        step+=1
    print("[x,y]=", inis)
    print("Converged in %d step(s)" % step)
    return(X1,X2,D)
```

```
X1,X2=gradient([2.99,0])
print(D)

x = np.linspace(0, 3, 1000)
y = np.linspace(0, 3, 1000)

X, Y = np.meshgrid(x, y)
Z = obfun(X, Y)
plt.contour(X, Y, Z, colors='black')

plt.plot(X1,X2,'bo',X1,X2,"k")
plt.show()
```

D. Python Code For Gradient Descent With Fixed Step Size

```
from scipy.misc import derivative
import matplotlib.pyplot as plt
import numpy as np
import math

def obfun(x1, x2):
    return(-((3*x1-x1**2)/2)*np.sin(x2)-((3-x1)**2/4)*np.sin(x2)*np.cos(x2))
def pd(fun, var=0, val=[]):
    arg = val[:]
    def wrap(x):
        arg[var] = x
        return fun(*arg)
    return(derivative(wrap, val[var], dx = 1e-1))
def gradient(inis):
    step=0
    d=[math.inf,math.inf]
    X1=[]
    X2=[]
    while abs(d[0])>0.000001 or abs(d[1])>0.000001:
        if (step % 10)==0:
            print('At step:', step)
        X1.append(inis[0])
        X2.append(inis[1])
        d=[-pd(obfun,0,inis),-pd(obfun,1,inis)]
        alpha=0.1
        inis[0],inis[1]=inis[0]+alpha*d[0],inis[1]+alpha*d[1]
        step+=1
    print("[x,y]=", inis)
    print("Converged in %d step(s)" % step)
    return(X1,X2)
X1,X2=gradient([2.99,0])
x = np.linspace(0, 3, 1000)
y = np.linspace(0, 3, 1000)

X, Y = np.meshgrid(x, y)
Z = obfun(X, Y)
plt.contour(X, Y, Z, colors='black')

plt.plot(X1,X2,'bo',X1,X2,"k")
plt.show()
```