# Firm's Profit Maximization and Cost Minimization Problems given Cobb Douglas Production Function

**back to Fan's Intro Math for Econon, Matlab Examples, or Dynamic Asset Repositories**

We have already solved the firm's maximization problem before given decreasing return to scale: Firm Maximization Problem with Capital and Labor (Decreasing Return to Scale)

Now, Let's solve the firm's problem with constraints. We can divide the profit maximization problem into two parts: 1, given a desired level of output, optimize over the optimal bundle of capital and labor; 2, given the result from the first part, optimize over the quantity of outputs. Here we focus on the first part, which can be thought of as a cost minimization or profit maximization problem.

## Profit Maximization with Constraint

Let's now write down the firm's cost minimization problem with the appropriate constraints, using the Cobb-Douglas production function.

We can state the problem as a profit maximization problem:

- $\max_{K,L} \{p \cdot AK^{\alpha}L^{\beta} - w \cdot L - r \cdot K\}$
- such that: $AK^{\alpha}L^{\beta} = q$, where *q* is some desired level of output

We can write down the lagrangian for this problem:

- $\mathscr{L} = \{p \cdot AK^{\alpha}L^{\beta} - w \cdot L - r \cdot K\} - \mu \cdot (AK^{\alpha}L^{\beta} - q)$

Now, the maximization problem has three choice variables, $L, K, \mu$, where $\mu$ is the lagrange multiplier.

***Step 1***: We can plug things into matlab's symbolic toolbox

```
% These are the parameters
syms p A alpha beta w r q
% These are the choice variables
syms K L m
% The Lagrangian
lagrangian = (p*A*(K^alpha)*(L^beta) - w*L - r*K) - m*(A*(K^alpha)*(L^beta) - q)
```

lagrangian = $m\ (q - A\ K^{\alpha}\ L^{\beta}) - L\ w - K\ r + A\ K^{\alpha}\ L^{\beta}\ p$

***Step 2***: As before, we can differentiate and obtain the gradient

```
d_lagrangian_K = diff(lagrangian, K);
d_lagrangian_L = diff(lagrangian, L);
d_lagrangian_m = diff(lagrangian, m);
GRADIENT = [d_lagrangian_K; d_lagrangian_L; d_lagrangian_m]
```

```
GRADIENT =
```

$$\begin{pmatrix} A\,K^{\alpha-1}\,L^{\beta}\,\alpha\,p - A\,K^{\alpha-1}\,L^{\beta}\,\alpha\,m - r \\ A\,K^{\alpha}\,L^{\beta-1}\,\beta\,p - A\,K^{\alpha}\,L^{\beta-1}\,\beta\,m - w \\ q - A\,K^{\alpha}\,L^{\beta} \end{pmatrix}$$

***Step 3***: We can solve the problem. Let's plug in some numbers (matlab in this case is unable to solve the problem with symbols):

```
% Given we have many symbols, type K, L, mu at the end to let matlab know what we are solving f
GRADIENT = subs(GRADIENT, {A,p,w,r,q,alpha,beta},{1,1,1,1,2,0.3,0.7});
solu = solve(GRADIENT(1)==0, GRADIENT(2)==0, GRADIENT(3)==0, K, L, m, 'Real', true);
solu_K = double(solu.K);
solu_L = double(solu.L);
solu_m = double(solu.m);
table(solu_K, solu_L, solu_m)
```

ans = 1×3 table

| | solu_K | solu_L | solu_m |
|---|---|---|---|
| 1 | 1.1052 | 2.5788 | -0.8420 |

***Step 4***: What is the gradient at the optimal choices?

These are almost all exactly zero, which is what we expect, at the optimal choices, gradient should be 0. (SB P460)

```
GRADIENT_at_optimum = double(subs(GRADIENT, {K,L,m}, {solu_K, solu_L, solu_m}))
```

```
GRADIENT_at_optimum = 3×1
10^-15 ×
   -0.0156
    0.0131
   -0.1296
```

***Step 5***: What is the hessian with respect to $K, L$ (excluding $\mu$) at the optimal choices?

The second derivative condition is a little bit more complicated. You can see details on P460 of SB. In practice, we find the hessian only with respect to the real choices, not the multipliers, and we check if the resulting matrix is **negative definite**. If it is, we have found a **local maximum**.

```
HESSIAN = [diff(GRADIENT(1), K), diff(GRADIENT(2), K);...
              diff(GRADIENT(1), L), diff(GRADIENT(2), L)];
HESSIAN_at_optimum = double(subs(HESSIAN, {K,L,m}, {solu_K, solu_L, solu_m}))
```

```
HESSIAN_at_optimum = 2×2
   -0.6334    0.2714
    0.2714   -0.1163
```

Is the Hessian Positive definite or negative definite? Let's prove by trial and try some random vectors and use the $xAx'$ rule:

```
% An empty vector of zeros
xAx_save = zeros(1,100);
% Try 100 random xs and see what xAx equal to
for i=1:100
```

```
    x = rand(1,2);
    xAx_save(i) = x*HESSIAN_at_optimum*x';
end
% Let's see the first 5 elements:
xAx_save(1:5)
```

```
ans = 1×5
   -0.2043   -0.1004   -0.0100   -0.0593   -0.0317
```

```
% OK the first 5 elements are negative, what about the rest?
% This command creates a new vector equal to FALSE (or 0) if above or equal 0, and TRUE (or 1)
is_negative = (xAx_save < 0);
is_negative(1:5)
```

```
ans = 1×5 logical array
   1   1   1   1   1
```

```
% This counts how many are negative, should be 100, because this is a maximum
sum(is_negative)
```

```
ans = 100
```

## Cost Minimization with Constraint

We can actually re-write the problem as a cost minimization problem, because the first term in the objective function actually is always equal to *q*, so that does not change regardless of the choices we make, so we can take it out, and say we are minimizing the cost. So we can re-write the problem as:

- $\min_{K,L} \{w \cdot L + r \cdot K\}$
- such that: $AK^{\alpha}L^{\beta} = q$, where *q* is some desired level of output

We can write down the lagrangian for this problem:

- $\mathscr{L} = \{w \cdot L + r \cdot K\} - \mu \cdot (AK^{\alpha}L^{\beta} - q)$

This problem looks a little different, will we get the same solution? Yes, we can call the solutions below as the solutions to the COO's problem.

## Cost Minimization Problem--Optimal Capital Labor Choices

Taking derivative of *L*, *K* and *μ* with respect to the lagrangian, and setting first order conditions to 0, we can derive the optimal constrained capital and labor choices using the first order conditions above, they are (they would be the same if we derived them using the constrained profit maximization problem earlier):

- $K^{*}(w,r,q) = \left(\frac{q}{A}\right)^{\frac{1}{\alpha+\beta}} \cdot \left[\frac{\alpha}{\beta} \cdot \frac{w}{r}\right]^{\frac{\beta}{\alpha+\beta}}$
- $L^{*}(w,r,q) = \left(\frac{q}{A}\right)^{\frac{1}{\alpha+\beta}} \cdot \left[\frac{\alpha}{\beta} \cdot \frac{w}{r}\right]^{\frac{-\alpha}{\alpha+\beta}}$

If you divide the optimal constrained capital and labor choice equations above, you will find the optimal ratio is the same as what we derived in the unconstrained profit maximization problem: Firm Maximization Problem with Capital and Labor (Decreasing Return to Scale):

- $\frac{K^*(r,w)}{L^*(r,w)} = \frac{w}{r} \cdot \frac{\alpha}{\beta}$

This means the constraint does not change the optimal capital and labor ratio.

## Cost Minimization Problem--Solving on Matlab

***Step 1***: We can plug things into matlab's symbolic toolbox

```
clear all
% These are the parameters
syms p A alpha beta w r q
% These are the choice variables
syms K L m
% The Lagrangian
lagrangian_min = (w*L + r*K) - m*(A*(K^alpha)*(L^beta) - q)
```

lagrangian_min = $K\,r + L\,w + m\,(q - A\,K^{\alpha}\,L^{\beta})$

***Step 2***: As before, we can differentiate and obtain the gradient

```
d_lagrangian_min_K = diff(lagrangian_min, K);
d_lagrangian_min_L = diff(lagrangian_min, L);
d_lagrangian_min_m = diff(lagrangian_min, m);
GRADIENT = [d_lagrangian_min_K; d_lagrangian_min_L; d_lagrangian_min_m]
```

GRADIENT =

$$\begin{pmatrix} r - A\,K^{\alpha-1}\,L^{\beta}\,\alpha\,m \\ w - A\,K^{\alpha}\,L^{\beta-1}\,\beta\,m \\ q - A\,K^{\alpha}\,L^{\beta} \end{pmatrix}$$

***Step 3***: We can solve the problem. Let's plug in some numbers:

```
% Given we have many symbols, type K, L, mu at the end to let matlab know what we are solving f
GRADIENT = subs(GRADIENT, {A,p,w,r,q,alpha,beta},{1,1,1,1,2,0.3,0.7});
solu = solve(GRADIENT(1)==0, GRADIENT(2)==0, GRADIENT(3)==0, K, L, m, 'Real', true);
solu_K = double(solu.K);
solu_L = double(solu.L);
solu_m = double(solu.m);
table(solu_K, solu_L, solu_m)
```

ans = 1×3 table

| | solu_K | solu_L | solu_m |
|---|---|---|---|
| 1 | 1.1052 | 2.5788 | 1.8420 |

***Step 4***: What is the gradient at the optimal choices?

These are almost all exactly zero, which is what we expect, at the optimal choices, gradient should be 0. (SB P460)

```
GRADIENT_at_optimum = double(subs(GRADIENT, {K,L,m}, {solu_K, solu_L, solu_m}))
```

```
GRADIENT_at_optimum = 3×1
10^-15 ×
    0.0156
   -0.0131
   -0.1296
```

***Step 5***: What is the hessian with respect to $K, L$ (excluding $\mu$) at the optimal choices?

The second derivative condition is a little bit more complicated. You can see details on P460 of SB. In practice, we find the hessian only with respect to the real choices, not the multipliers, and we check if the resulting matrix is **positive definite**. If it is, we have found a **local minimum**.

```
HESSIAN = [diff(GRADIENT(1), K), diff(GRADIENT(2), K);...
           diff(GRADIENT(1), L), diff(GRADIENT(2), L)];
HESSIAN_at_optimum = double(subs(HESSIAN, {K,L,m}, {solu_K, solu_L, solu_m}))
```

```
HESSIAN_at_optimum = 2×2
    0.6334   -0.2714
   -0.2714    0.1163
```

Is the Hessian Positive definite or negative definite? Let's prove by trial and try some random vectors and use the $xAx'$ rule:

```
% An empty vector of zeros
xAx_save = zeros(1,100);
% Try 100 random xs and see what xAx equal to
for i=1:100
    x = rand(1,2);
    xAx_save(i) = x*HESSIAN_at_optimum*x';
end
% Let's see the first 5 elements:
xAx_save(1:5)
```

```
ans = 1×5
    0.0118    0.0684    0.3003    0.4626    0.0017
```

```
% OK the first 5 elements are positive, what about the rest?
% This command creates a new vector equal to FALSE (or 0) if below or equal 0, and TRUE (or 1)
is_positive = (xAx_save > 0);
is_positive(1:5)
```

```
ans = 1×5 logical array
   1   1   1   1   1
```

```
% This counts how many are postiive, should be 100, because this is a minimum
sum(is_positive)
```

```
ans = 100
```