Long-Run Firm Equilibrium Notes

true

24 Mar 2023

Review: The Firm's Three Problems:

1. ~~Input cost minimization~~
2. ~~Output profit maximization~~
3. Long-Run Readjustment

Perfectly Competitive Input Markets

1. Many, many buyers
2. Homogeneous inputs
3. Perfect information
4. Free entry/exit
5. Firms are price takers (1-3)
6. Profits/lossesare temporary

The Producer's Input-Hiring Problem

1. Objective: Maximize Profit
2. Endogenous (choice) variable(s): Capital, Quantity, Labor
3. Exogenous variable(s): Technology, Market Price, Wage, Return to Capital

Initial Equilibrium (Short Run)

Set up the initial equilibrium with initial values of the market price (P) and quantity (Qm); planned levels of output (Q0), labor (L0), and capital (K); actual short-run labor (L), capital (K), and output (Q), and profit (Pi):

```
library(Rsolnp); library(nleqslv)
A = 100; a = 1/3; y1 = 3227.486; w = 10; r = 20
control = list(trace = 0)
P = NULL; Qm = NULL; Q0 = NULL; L0 = NULL; K = NULL; Q = NULL; L = NULL; Pi = NULL
```

When we last left our representative competitive firm, they operated in a market where $P = 0.5$ and maximized profit at $Q_i^* = \sqrt{\frac{31{,}250{,}000}{3}} \approx 3,227.5$ earning a total profit of \$575.8 (whew, whoever wrote this example could have come up with friendlier numbers!). Let's remind ourselves of how to solve the market equilibrium and set the first element of each market variable (`X[1]`) to the initial equilibrium.

```
supply <-function(x) 10000000*x[1] - x[2]
demand <- function(x) 6000000 - 2000000*x[1] - x[2]
marketEq <- nleqslv(c(2,2), function(x) c(supply(x), demand(x)))
P[1] = marketEq$x[1]; Qm[1] = marketEq$x[2]; Q0[1] = 2500
cbind(P, Qm, Q0)
```

```
##        P    Qm   Q0
## [1,] 0.5 5e+06 2500
```

The firm expected to produce 2,500 units and planned its capacity likewise. This meant building 25 units of capital and planning to hire 25 units of labor.

```
cost <- function(x) w*x[1] + r*x[2]
output <- function(x) A*x[1]^a * x[2]^(1 - a)
costMin <- solnp(pars = c(1, 1), fun = cost, ineqfun = output,
  ineqLB = Q0[1], ineqUB = Inf, LB = c(0, 0), control = control)
L0[1] = costMin$pars[1]; K[1] = costMin$pars[2]
cbind(L0, K)
```

```
##      L0  K
## [1,] 25 25
```

Based on this short run guess and the market price of $0.50, the firm chose its output (and, hence its short run variable input) to maximize profits.

```
labor <- function(x) x^3 / (K[1] ^ 2 * A^3)
loss <- function(x) -(P[1]*x - (w*labor(x) + r*K[1]))
piMax <- solnp(pars = 1, fun = loss, LB = 0)
```

```
##
## Iter: 1 fn: -575.8287     Pars:  3227.49095
## Iter: 2 fn: -575.8287     Pars:  3227.49095
## solnp--> Completed in 2 iterations
```

```
Q[1] = piMax$pars; L[1] = labor(piMax$pars); Pi[1] = -tail(piMax$values, 1)
cbind(Q, L, Pi)
```

```
##             Q        L       Pi
## [1,] 3227.491 53.79168 575.8287
```

From this outcome, the firm could lower its costs (and slightly increase profits) in the long run by increasing its capacity (choosing a level of capital that optimizes for 3,227.5 instead of 2,500 as initially expected).

Maybe this happens because independent startups enter a profit-making industry (exit a loss-making industry) or maybe this happens as capital pours into existing firms from investors looking for above-market returns - who knows!

Either way, let's solve it!

Long Run Convergence

The firm(s) will iteratively readjust levels of capital, readjust quantities (which shifts market supply), and adjust to changes in the market (shift in supply changes price) until industry profits equal zero.

1. Input cost minimization
2. Output profit maximization
3. Long-Run Readjustment
4. Repeat until Zero Profit ($P = ATC$)

Market Adjustment

The problem with this inflow of capital, is that it happens industry-wide. The firm could sit by and not do this, but it would just mean other firms (new or existing) will do it instead, so failing to do so leaves $$ on the table.

```
supply <-function(x) (Q[1]/Q0[1])*(10000000*x[1]) - x[2]   # Scale up market supply
demand <- function(x) 6000000 - 2000000*x[1] - x[2]
marketEq <- nleqslv(c(2,2), function(x) c(supply(x), demand(x)))
P[2] = marketEq$x[1]; Qm[2] = marketEq$x[2]
cbind(P, Qm)
```

```
##              P      Qm
## [1,] 0.5000000 5000000
## [2,] 0.4024155 5195169
```

Cost (Re-)Minimization

Given this new market reality, firms will want to adjust their capital input to (conditionally) minimize costs.

Solve for the cost-minimizing level of capital for the previous short-run level of output.

```
Q0[2] = Q[1]
costMin <- solnp(pars = c(1, 1), fun = cost,
  ineqfun = output, ineqLB = Q0[2], ineqUB = Inf, LB = c(0,0), control = control)
L0[2] = costMin$pars[1]; K[2] = costMin$pars[2]
cbind(Q0, L0, K)
```

```
##            Q0       L0        K
## [1,] 2500.000 25.00000 25.00000
## [2,] 3227.491 32.27491 32.27491
```

Profit (Re-Maximization)

```
labor <- function(x) x^3 / (K[2] ^ 2 * A^3)           # Note the change in index!
loss <- function(x) -(P[2]*x - (w*labor(x) + r*K[2])) # New index here, too!
piMax <- solnp(pars = 1, fun = loss, LB = 0, control = control)
Q[2] = piMax$pars; L[2] = labor(piMax$pars); Pi[2] = -tail(piMax$values, 1)
cbind(Q, L, Pi)
```

```
##             Q        L       Pi
## [1,] 3227.491 53.79168 575.8287
## [2,] 3738.017 50.14109 357.3268
```

The representative firm in our market increased its output from 3227.5 to 4166.7, or about 29%

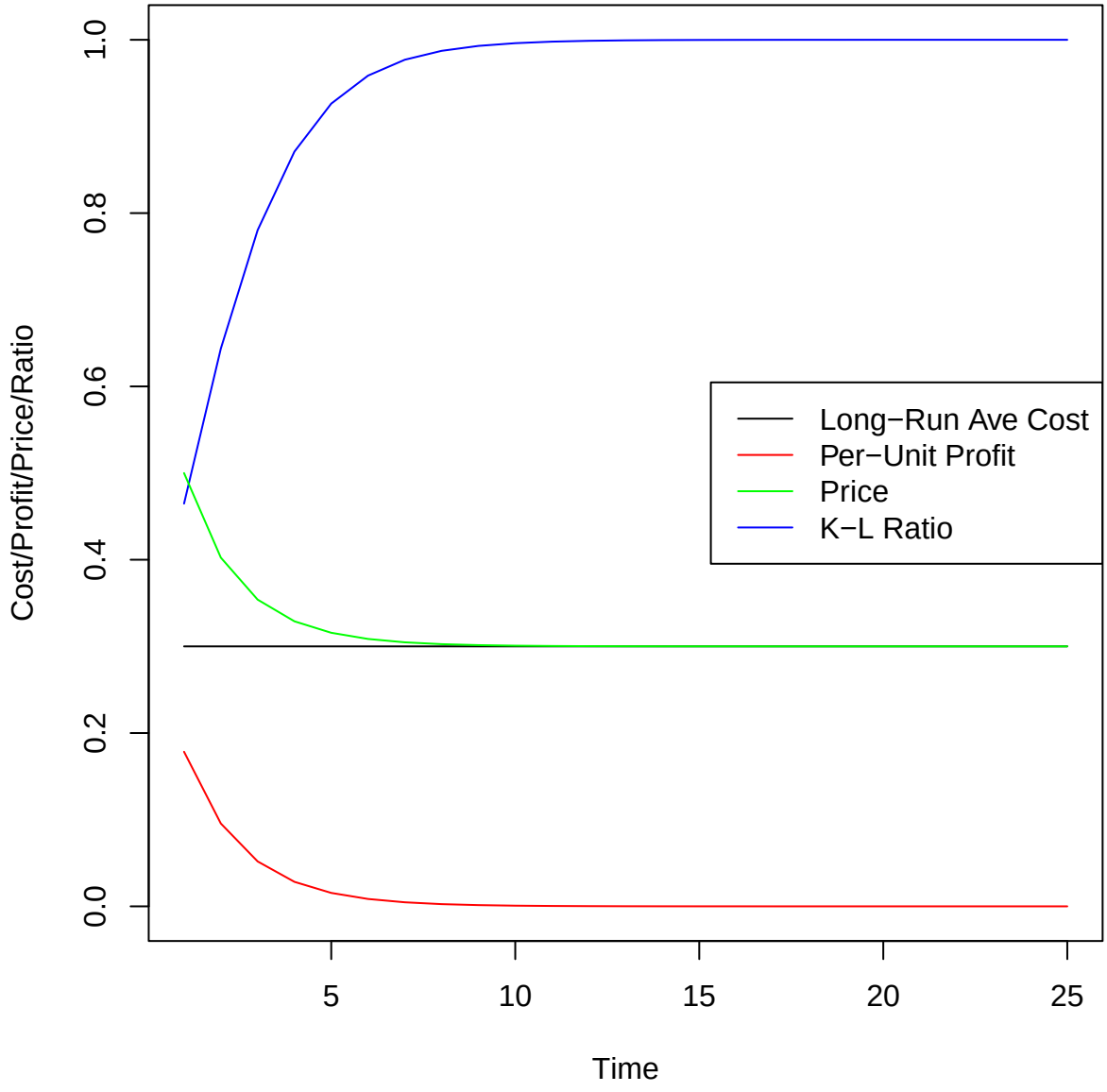
Iterating the solution

Step 4 tells us makes this problem ideal for the computer to solve using a "for" or "while" loop! I've done this for you, but with come careful *thinking* (not coding) we can set it up to calculate the full convergence to the long run equilibrium!

```
tol = 1e-8; i = 2
while(Pi[i-1] > tol) {
  supply <-function(x) (Q[i-1]/Q0[1])*(10000000*x[1]) - x[2]
  demand <- function(x) 6000000 - 2000000*x[1] - x[2]
  marketEq <- nleqslv(c(2,2), function(x) c(supply(x), demand(x)))
  P[i] = marketEq$x[1]; Qm[i] = marketEq$x[2]; Q0[i] = Q[i-1]
  costMin <- solnp(pars = c(1, 1), fun = cost, ineqfun = output, ineqLB = Q0[i],
                   ineqUB = Inf, LB = c(0,0), control = control)
  L0[i] = costMin$pars[1]; K[i] = costMin$pars[2]
  labor <- function(x) x^3 / (K[i] ^ 2 * A^3)
  loss <- function(x) -(P[i]*x - (w*labor(x) + r*K[i]))
  piMax <- solnp(pars = 1, fun = loss, LB = 0, control = control)
  Q[i] = piMax$pars; L[i] = labor(piMax$pars); Pi[i] = -tail(piMax$values, 1)
  i = i + 1 # `while()` requires us to reset i each time if you want to store them.
}
result <- data.frame(Time = c(1:length(P)), Price = P, Q.Market = Qm,
    Q.Planned = Q0, L.Planned = L0, K = K, Q = Q, L = L, Profit = Pi)
attach(result)      # This attaches the "result" data frame to reference its names.
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     K, L, Q
```

```
plot(Time, rep(0.3, nrow(result)), ylim = c(0, 1), type = 'l',
     ylab = 'Cost/Profit/Price/Ratio')
lines(Time, K / L, col = 'blue')
lines(Time, Profit / Q, col = 'red')
lines(Time, Price, col = 'green')
legend('right', c('Long-Run Ave Cost', 'Per-Unit Profit', 'Price', 'K-L Ratio'),
  lty = 1, col = c('black', 'red', 'green', 'blue'))
```



Like I said: