Lecture 2: The Nature of Costs

ACCT 3210

Dr. Morris

We are interested in how costs respond to business decisions? Why?

- Costs are resources. Business decisions that have costs require resources so we need to make sure we have the resources when required.
- Cost, Volume, Profit analysis. Which we will talk about in the following lectures, is based on the relationship between a key decision—the volume decision (how much to produce)—and costs.
- We have to know where the resources are and where they need to them to be in order to understand the resource bargins that need to be negotiated.

Why do we care about costs?

Profit = Revenue - Total Cost

Total Cost = Fixed Cost + $\sum$ (Cost Per Unit $\times$ # Units)

Revenue = $\sum$ (Price Per Unit $\times$ # Units)

$\sum$*: Summed across all products.*

- Apple chooses how many devices to produce based projections of how costs and revenues will respond to this decision.
- Today we will focus on modelling the cost portion of this equation.

Notate Bene:

- Some of you will think that I use too many “U.S.” examples.
- Please note that most of my examples reference Apple’s *production*.
- If you think that this is a U.S. example, I have some very surprising news for you. :]

Why do we care about profit?

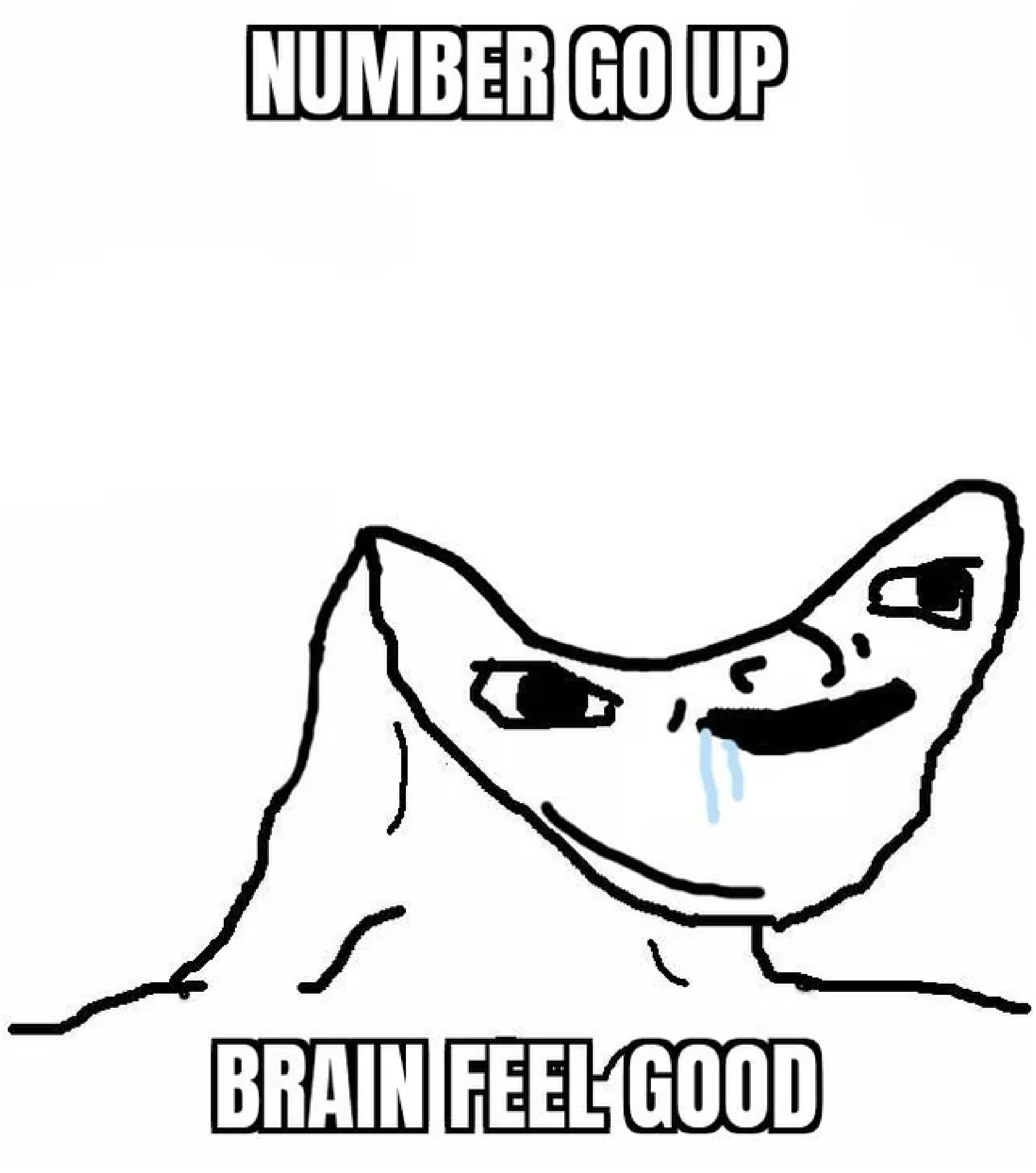


Figure 1: Zimmerman and Friedman's vision of the firm

Cost Functions

Models of the firm that we use to predict how cost will respond to various actions, which we express as variables in the model.

A simple example where everything is linear:

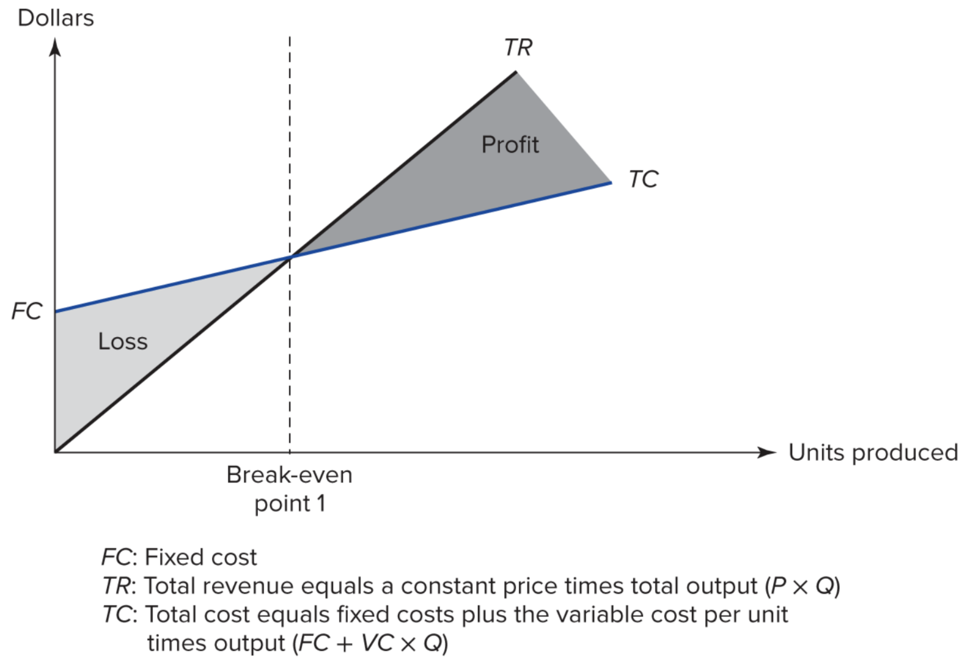


Figure 2: Linear Cost and Revenue Functions

Let's look at some cost terms in the context of a single-product firm.

- **Fixed cost (FC):** Cost at zero output. Also used to refer to costs that do not vary with output (or some other driver).
- **Variable cost (VC):** Cost that vary with output (or some other driver).
- **Marginal cost (MC):** The cost per unit at the margin (i.e. the point of interest). This is the rate of change of cost at the margin.
- **Incremental cost (IC):** The cost of producing the next unit. Often MC and IC have the same value, but they are slightly different things!

- **Average cost (ATC):** Total Cost of producing the output over the number of units of output. This is a simple average for single product firms. It is not simple at all for multi-product firms.

iPRS here.

- **Cost object:** An activity or item for which we want to measure cost.
- **Cost driver:** Any factor or activity whose change leads to a change in costs.

A simple example where everything is linear:

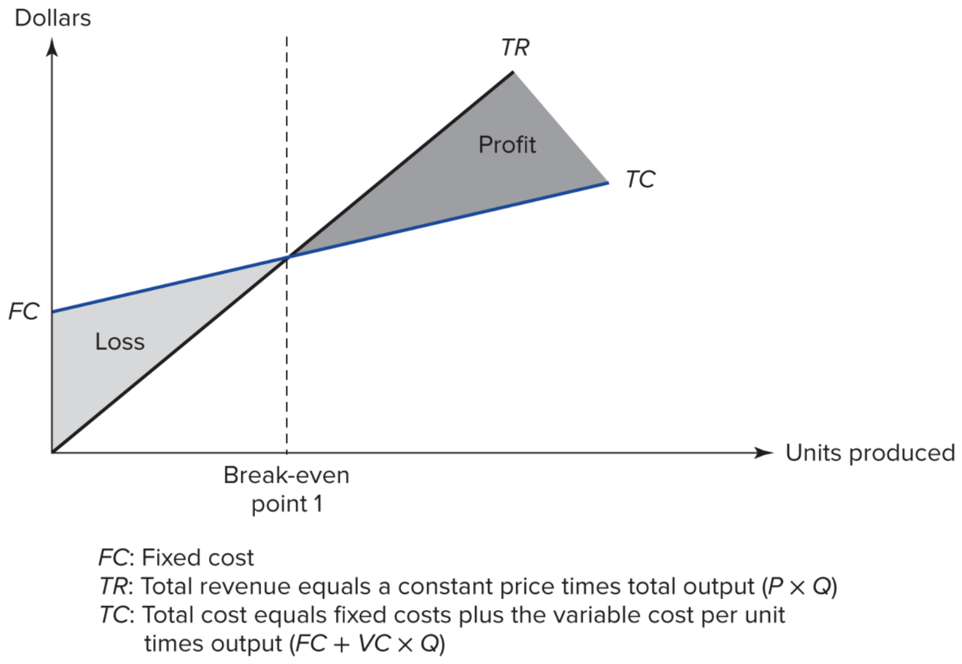


Figure 3: Linear Cost and Revenue Functions

Can you see anything unrealistic in this graph?

Most firms' costs are non-linear

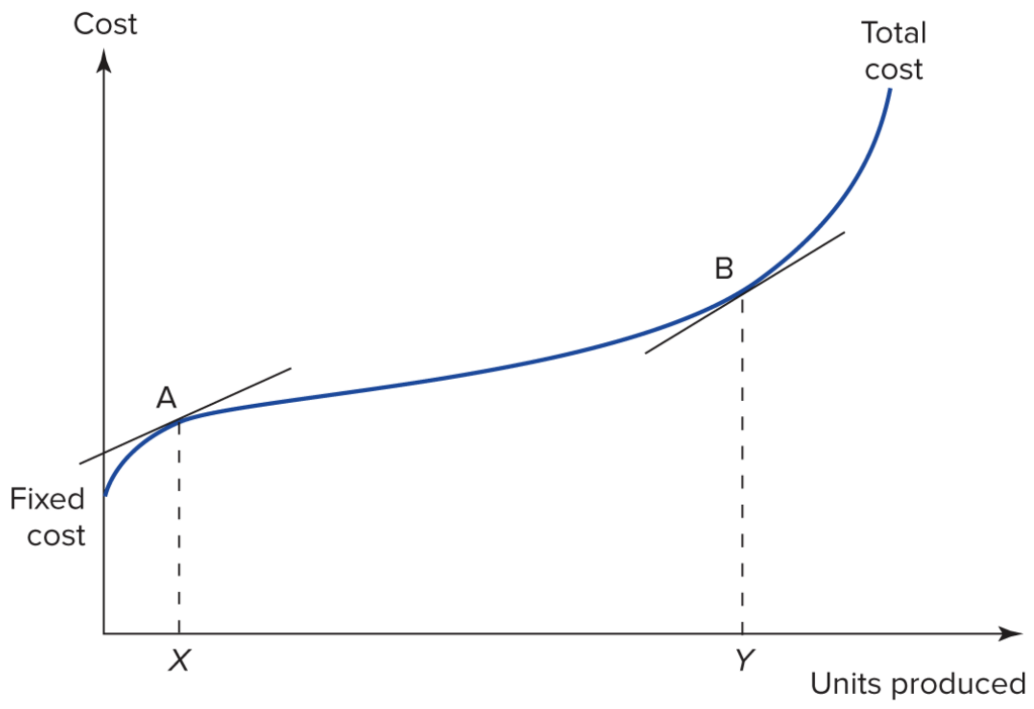


Figure 4: Non-linear Cost and Revenue Functions

Most firms' costs are non-linear

Questions about the previous slides:

1. What is the economic significance of the area to the left of the line from X to A?
2. What is the economic significance of the area between X->A and Y->B?
3. What is the economic significance of the area to the right of Y->B?

Note: I am not providing the answers here because I want you to ponder these questions.

Cost Types

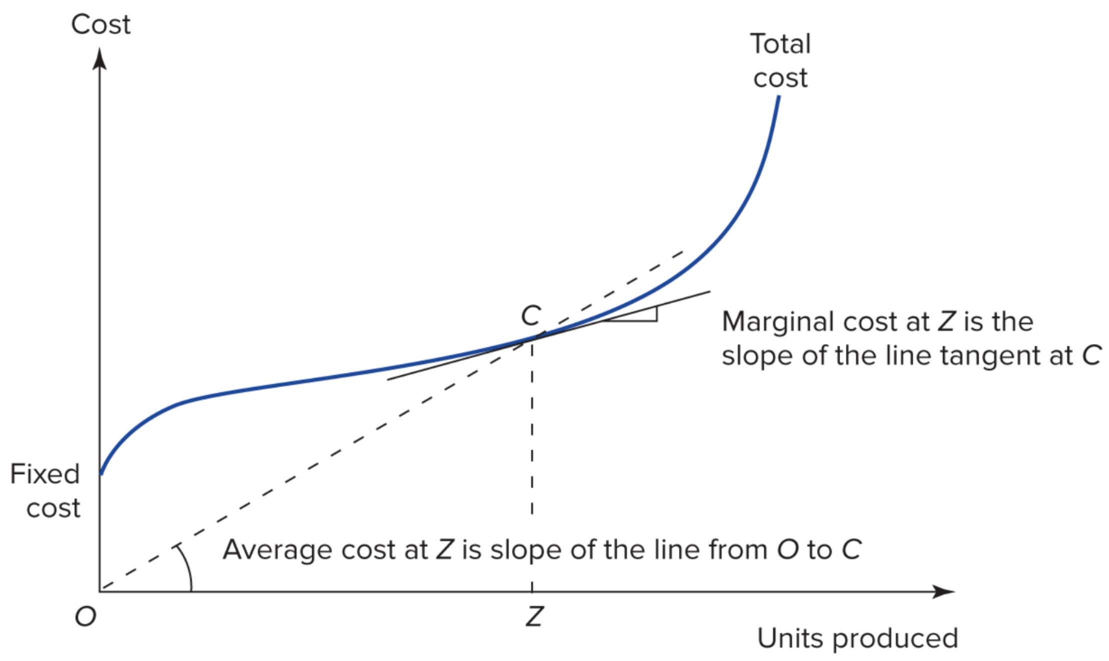


Figure 5: Marginal and Incremental Costs

1. If marginal cost is the slope of the tangent line and incremental cost is the cost of one unit, then when are they the same on this graph?
2. When are they different?

At any scale there is a range within which production is efficient

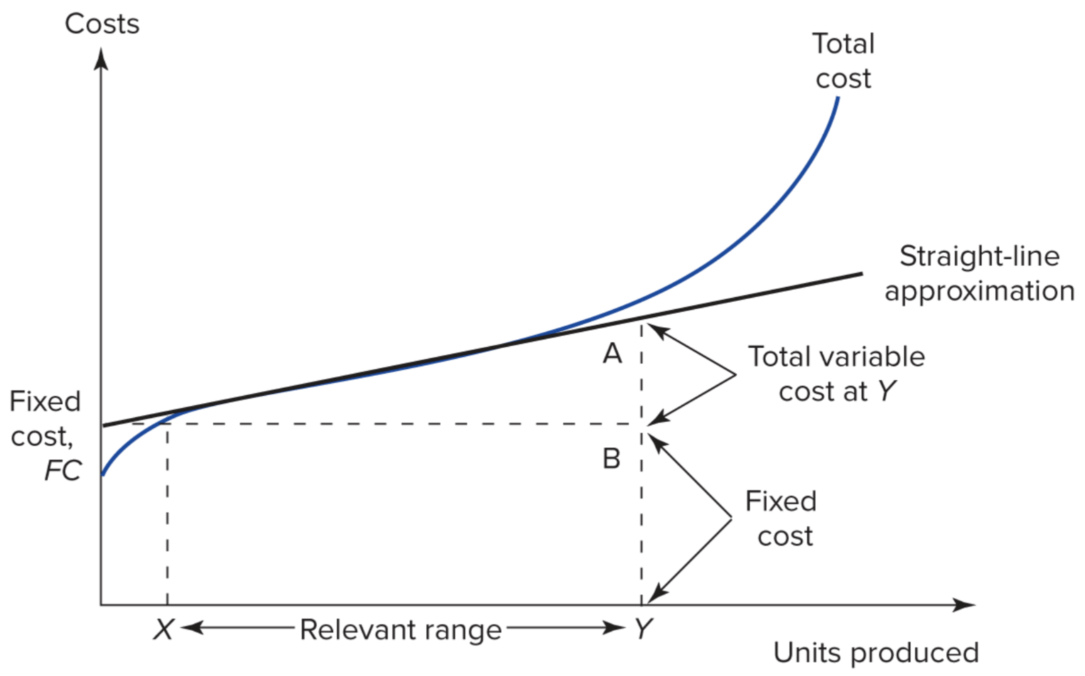


Figure 6: Producing outside of this range is less efficient, unless we change the scale of the firm.

Costs are not always smooth

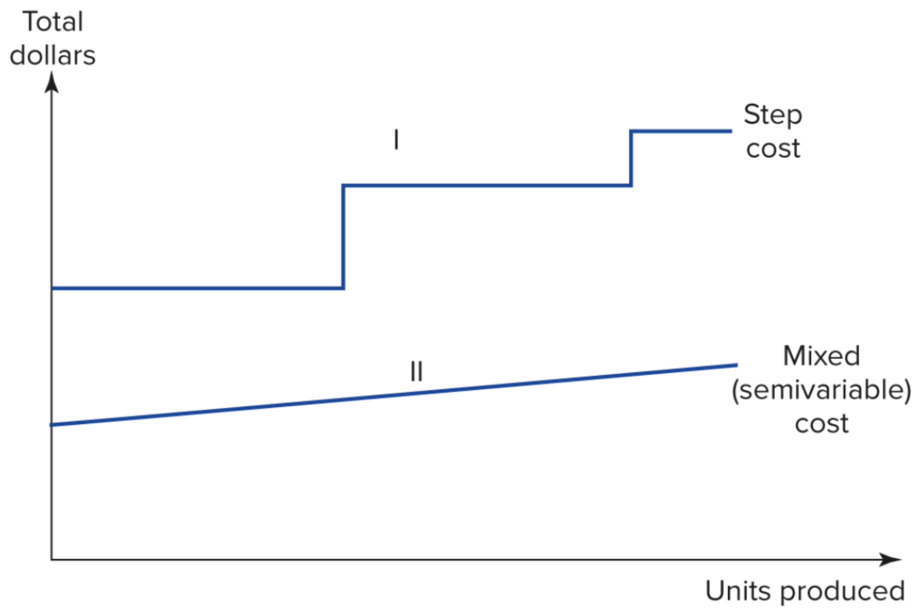


Figure 7: Most firms have a mix of these attributes. Steps occur when the scale of the firm changes (e.g. we add a new factory).

Let's talk about the homework assignment!

- Download the Excel File here
- **Please note that Python, Excel, or any other set of tools can be used on the homework!**

Cost in a Multiproduct Firm:

Consider three firms that produce two products with quantities denoted q_1 and q_2. The three distinct cost functions are:

- $C_1(q_1, q_2) = 10q_1 + 5q_2$
- $C_2(q_1, q_2) = 6q_1 + q_1^2 + 8q_2 + q_2^2$
- $C_3(q_1, q_2) = 7q_1 + 9q_2 + q_1 q_2$

Cost in a Multiproduct Firm:

1. Fill in the following table for each of the cost functions. (Incremental cost refers to the incremental cost of one additional unit of output.)

Output	Total Cost	Average Cost	Marginal Cost	Incremental Cost
q_1, q_2		q_1, q_2	q_1, q_2	q_1, q_2
100, 50				
60, 50				
40, 50				
30, 10				
30, 50				
30, 70				

Total cost:

- Plug the output data into each cost function!

Let's fill this out using Python (**don't panic**), Excel (also **don't panic**).

We'll start with Excel

Reference items:

- **Marginal cost (MC):** The cost per unit at the margin (i.e. the point of interest). This is the rate of change of cost at the margin.
- **Incremental cost (IC):** The cost of producing the next unit. Often MC and IC have the same value, but they are slightly different things!
- **Average cost (ATC):** Total Cost of producing the output over the number of units of output. This is a simple average for single product firms. It is not simple at all for multi-product firms.
- $C_1(q_1, q_2) = 10q_1 + 5q_2$
- $C_2(q_1, q_2) = 6q_1 + q_1^2 + 8q_2 + q_2^2$
- $C_3(q_1, q_2) = 7q_1 + 9q_2 + q_1 q_2$

Now Python

You can follow along in colab.

Set up: load libraries and data

First we need to load some data science libraries:

```python
# import pandas so we can put everything into a nice friendly data frame
import pandas as pd
import numpy as np

# lets put what we know into a dict (python dicts are POWERFUL use them when in doubt)
outputs = {
    "q1" : [100, 60, 40, 30, 30, 30],
    "q2" : [50, 50, 50, 10, 50, 70],
}
outputs
```

```
{'q1': [100, 60, 40, 30, 30, 30], 'q2': [50, 50, 50, 10, 50, 70]}
```

Table for Firm 1

```python
# use pandas to make that into a dataframe
cost_frame_1 = pd.DataFrame(outputs)
cost_frame_2 = pd.DataFrame(outputs)
cost_frame_3 = pd.DataFrame(outputs)
cost_frame_1
```

	q1	q2
0	100	50
1	60	50
2	40	50
3	30	10
4	30	50
5	30	70

Next write down our cost functions as... well... functions

```python
# write down our cost functions
## total cost
def cost_1(q1,q2):
  return 10 * q1 + 5 * q2
def cost_2(q1,q2):
  # note that we have to use ** in place of ^ here
  return 6 * q1 + q1**2 + 8 * q2 + q2**2
def cost_3(q1,q2):
  return 7 * q1 + 9 * q2 + q1 * q2

cost_1(1,2)
```

```
20
```

Notice how close this is to how you might type this on your phone!

Then we can use those functions to calculate average cost

- We can just pass q1,q2 as arguments to the cost functions

```python
cost_1(100,50)
```

```
1250
```

```python
cost_2(100,50)
```

```
13500
```

```python
cost_3(100,50)
```

```
6150
```

Now we need to do this to all the data in the data frames

slow simple way:

```
TotalCost1 = []
for q1,q2 in zip(outputs['q1'],outputs['q2']):
    TotalCost1.append(cost_1(q1,q2))
print(TotalCost1)
outputs
```

```
[1250, 850, 650, 350, 550, 650]
```

```
{'q1': [100, 60, 40, 30, 30, 30], 'q2': [50, 50, 50, 10, 50, 70]}
```

```
cost_1(100,50)
```

```
1250
```

less simple but faster way

```
TotalCost1 = [cost_1(q1,q2) for q1,q2 in zip(outputs['q1'],outputs['q2'])]
print(TotalCost1)
```

```
[1250, 850, 650, 350, 550, 650]
```

super fast way that scales to large datasets

```
cost_frame_1["Total Cost"] = np.vectorize(cost_1)(cost_frame_1['q1'],cost_frame_1['q2'])
cost_frame_1
```

	q1	q2	Total Cost
0	100	50	1250
1	60	50	850
2	40	50	650
3	30	10	350
4	30	50	550
5	30	70	650

```
# we can do this for the other to firms:
cost_frame_2["Total Cost"] = np.vectorize(cost_2)(cost_frame_2['q1'],cost_frame_2['q2'])
cost_frame_3["Total Cost"] = np.vectorize(cost_3)(cost_frame_3['q1'],cost_frame_3['q2'])
cost_frame_2
```

	q1	q2	Total Cost
0	100	50	13500
1	60	50	6860
2	40	50	4740
3	30	10	1260
4	30	50	3980
5	30	70	6540

Average cost

The average cost of a each product is the total cost for producing **that product alone** divided by the number of units produced.

For firm 1 & 2 this is straightforward, each firm has an AC for each product where we plug in zero for the other product:

- $AC_1(q_1) = (10q_1 + 0)/q_1$
- $AC_1(q_2) = (0 + 5q_2)/q_2$
- $AC_2(q_1) = (6q_1 + q_1^2 + 0 + 0)/q_1$
- $AC_2(q_2) = (0 + 0 + 8q_2 + q_2^2)/q_2$

What about firm 3?

$$C_3(q_1, q_2) = 7q_1 + 9q_2 + q_1 q_2$$

What does $q_1 \times q_2$ mean?

- when two products are multiplied like this we often refer to it as an “interaction”
- Plugging in zero no longer separates the costs.
- Calculating the average cost for each product requires us to separate the costs of the products.

- When there are interactions between products their costs are **inseparable**!!
- So "average cost" is no longer a meaningful number!

One way to think of this is that average cost requires us to pretend that the firm only produces one product. When we can separate costs then this pretend firm tells us something about the real firm. When we cannot separate costs, this pretend firm **does not tell us anything about the real firm**!!

One way to do this in python is to write a function

```python
# avg cost by product
def avg_cost(cost_function,q1=0,q2=0):
    """
    cost_function: the cost function you are averaging
    pass either q1 or q2 but not both to tell which product to use
    """
    if q1!=0 & q2!=0:
        print("only pass one nonzero argument")
        return None
    else:
        return cost_function(q1,q2) / (q1+q2)
```

Firm 1

```python
# average cost the fast way
cost_frame_1["AC q1"] = np.vectorize(avg_cost)(cost_1,q1=cost_frame_1['q1'])
cost_frame_1["AC q2"] = np.vectorize(avg_cost)(cost_1,q2=cost_frame_1['q2'])
cost_frame_1
```

	q1	q2	Total Cost	AC q1	AC q2
0	100	50	1250	10.0	5.0
1	60	50	850	10.0	5.0
2	40	50	650	10.0	5.0
3	30	10	350	10.0	5.0
4	30	50	550	10.0	5.0
5	30	70	650	10.0	5.0

Firm 2

```
# average cost the fast way
cost_frame_2["AC q1"] = np.vectorize(avg_cost)(cost_2,q1=cost_frame_2['q1'])
cost_frame_2["AC q2"] = np.vectorize(avg_cost)(cost_2,q2=cost_frame_2['q2'])
cost_frame_2
```

	q1	q2	Total Cost	AC q1	AC q2
0	100	50	13500	106.0	58.0
1	60	50	6860	66.0	58.0
2	40	50	4740	46.0	58.0
3	30	10	1260	36.0	18.0
4	30	50	3980	36.0	58.0
5	30	70	6540	36.0	78.0

Marginal Cost

The marginal cost is the derivative of the cost function wrt. the product.

$$C_1(q_1, q_2) = 10q_1 + 5q_2$$
$$MC_1(q_1) = 10$$
$$MC_1(q_2) = 5$$

$$C_2(q_1, q_2) = 6q_1 + q_1^2 + 8q_2 + q_2^2$$
$$MC_2(q_1) = 6 + 2q_1$$
$$MC_2(q_2) = 8 + 2q_2$$

$$C_3(q_1, q_2) = 7q_1 + 9q_2 + q_1 q_2$$
$$MC_3(q_1) = 7 + q_2$$
$$MC_3(q_2) = 9 + q_1$$

- This might help with the intuition for the average cost in this case!

Hate Calculus?

let's make python do the work

```
# we'll use symbolic python
import sympy as sp
# we need to tell it which symbols to use
q1,q2 = sp.symbols('q1 q2')
q1
```

q_1

```
# sympy can take the derivative for us
c1 = "10 * q1 + 5 * q2"
s_mcost_1_q1 = sp.diff(c1 , q1)
s_mcost_1_q2 = sp.diff(c1 , q2)
print(s_mcost_1_q1,s_mcost_1_q2)
```

```
10 5
```

```
# and we can convert that to a function
mcost_1_q1 = sp.lambdify(q1,s_mcost_1_q1)
mcost_1_q2 = sp.lambdify(q2,s_mcost_1_q2)
mcost_1_q1(100),mcost_1_q2(100)
```

```
(10, 5)
```

Firm 1

```
# marginal cost
cost_frame_1["MC q1"] = np.vectorize(mcost_1_q1)(cost_frame_1['q1'])
cost_frame_1["MC q2"] = np.vectorize(mcost_1_q2)(cost_frame_1['q2'])
cost_frame_1
```

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2
0	100	50	1250	10.0	5.0	10	5
1	60	50	850	10.0	5.0	10	5

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2
2	40	50	650	10.0	5.0	10	5
3	30	10	350	10.0	5.0	10	5
4	30	50	550	10.0	5.0	10	5
5	30	70	650	10.0	5.0	10	5

Firm 2

```
q1,q2 = sp.symbols('q1 q2')
# sympy can take the derivative for us
c2 = "6 * q1 + q1**2 + 8 * q2 + q2**2"
s_mcost_2_q1 = sp.diff(c2 , q1)
s_mcost_2_q2 = sp.diff(c2 , q2)
print(s_mcost_2_q1,s_mcost_2_q2)
# and we can convert that to a function
mcost_2_q1 = sp.lambdify(q1,s_mcost_2_q1)
mcost_2_q2 = sp.lambdify(q2,s_mcost_2_q2)
mcost_2_q1(100),mcost_2_q2(50)
```

```
2*q1 + 6 2*q2 + 8
```

```
(206, 108)
```

Firm 2 Table

```
# marginal cost
cost_frame_2["MC q1"] = np.vectorize(mcost_2_q1)(cost_frame_2['q1'])
cost_frame_2["MC q2"] = np.vectorize(mcost_2_q2)(cost_frame_2['q2'])
cost_frame_2
```

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2
0	100	50	13500	106.0	58.0	206	108
1	60	50	6860	66.0	58.0	126	108
2	40	50	4740	46.0	58.0	86	108
3	30	10	1260	36.0	18.0	66	28
4	30	50	3980	36.0	58.0	66	108

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2
5	30	70	6540	36.0	78.0	66	148

Firm 3

```
q1,q2 = sp.symbols('q1 q2')
# sympy can take the derivative for us
c3 = "7*q1 + 9*q2 + q1*q2"
s_mcost_3_q1 = sp.diff(c3 , q1)
s_mcost_3_q2 = sp.diff(c3 , q2)
print(s_mcost_3_q1,s_mcost_3_q2)
# and we can convert that to a function
# note tht we flip the inputs to match the function
mcost_3_q1 = sp.lambdify(q2,s_mcost_3_q1)
mcost_3_q2 = sp.lambdify(q1,s_mcost_3_q2)
mcost_3_q1(50),mcost_3_q2(100)
```

```
q2 + 7 q1 + 9
```

```
(57, 109)
```

Firm 3 Table

```
# marginal cost
cost_frame_3["MC q1"] = np.vectorize(mcost_3_q1)(cost_frame_3['q2'])
cost_frame_3["MC q2"] = np.vectorize(mcost_3_q2)(cost_frame_3['q1'])
cost_frame_3
```

	q1	q2	Total Cost	MC q1	MC q2
0	100	50	6150	57	109
1	60	50	3870	57	69
2	40	50	2730	57	49
3	30	10	600	17	39
4	30	50	2160	57	39
5	30	70	2940	77	39

Incremental Cost

$$IC(q_1) = C(q_1 + 1, q_2) - C(q_1, q_2)$$
$$IC(q_2) = C(q_1, q_2 + 1) - C(q_1, q_2)$$

Which I find a little easier to write than to say :)

In python we'll just write a little function for this

```python
# incremental cost if the cost of making the next unit by product
def inc_cost(cost_function,q1=q1,q2=q2,increment=str):
    '''
    cost_function: total cost function that you'd like to increment (over q1,q2)
    q1: the quantity you'd like to pass to the cost func as q1, defaults q1
    q2: same, default q2
    increment: the quantity you'd like to increment
    '''
    C_0 = cost_function(q1,q2)
    if increment == "q1":
        q1=q1+1
    elif increment == "q2":
        q2=q2+1
    else:
        print("increment must be one of q1,q2")
        return None
    C_1 = cost_function(q1,q2)
    return C_1 - C_0
```

Firm 1

```python
# Incremental cost
cost_frame_1["IC q1"] = np.vectorize(inc_cost)(
                                    cost_1,
                                    cost_frame_1['q1'],
                                    cost_frame_1['q2'],
                                    increment="q1"
                                    )
cost_frame_1["IC q2"] = np.vectorize(inc_cost)(
                                    cost_1,
                                    cost_frame_1['q1'],
                                    cost_frame_1['q2'],
                                    increment="q2"
                                    )
cost_frame_1
```

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2	IC q1	IC q2
0	100	50	1250	10.0	5.0	10	5	10	5
1	60	50	850	10.0	5.0	10	5	10	5
2	40	50	650	10.0	5.0	10	5	10	5
3	30	10	350	10.0	5.0	10	5	10	5
4	30	50	550	10.0	5.0	10	5	10	5
5	30	70	650	10.0	5.0	10	5	10	5

Firm 2

```
# Incremental cost
cost_frame_2["IC q1"] = np.vectorize(inc_cost)(
                                    cost_2,
                                    cost_frame_2['q1'],
                                    cost_frame_2['q2'],
                                    increment="q1"
                                    )
cost_frame_2["IC q2"] = np.vectorize(inc_cost)(
                                    cost_2,
                                    cost_frame_2['q1'],
                                    cost_frame_2['q2'],
                                    increment="q2"
                                    )
cost_frame_2
```

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2	IC q1	IC q2
0	100	50	13500	106.0	58.0	206	108	207	109
1	60	50	6860	66.0	58.0	126	108	127	109
2	40	50	4740	46.0	58.0	86	108	87	109
3	30	10	1260	36.0	18.0	66	28	67	29
4	30	50	3980	36.0	58.0	66	108	67	109

	q1	q2	Total Cost	AC q1	AC q2	MC q1	MC q2	IC q1	IC q2
5	30	70	6540	36.0	78.0	66	148	67	149

Firm 3

```
# Incremental cost
cost_frame_3["IC q1"] = np.vectorize(inc_cost)(
                                    cost_3,
                                    cost_frame_3['q1'],
                                    cost_frame_3['q2'],
                                    increment="q1"
                                    )
cost_frame_3["IC q2"] = np.vectorize(inc_cost)(
                                    cost_3,
                                    cost_frame_3['q1'],
                                    cost_frame_3['q2'],
                                    increment="q2"
                                    )
cost_frame_3
```

	q1	q2	Total Cost	MC q1	MC q2	IC q1	IC q2
0	100	50	6150	57	109	57	109
1	60	50	3870	57	69	57	69
2	40	50	2730	57	49	57	49
3	30	10	600	17	39	17	39
4	30	50	2160	57	39	57	39
5	30	70	2940	77	39	77	39

Let's make a 3d graph in Python!!!

First load libraries and make the data

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np # we already have np
```

```
# Create data for the plot
q1 = np.linspace(0, 1_000, 1_000)
q2 = np.linspace(0, 1_000, 1_000)
Q1, Q2 = np.meshgrid(q1, q2)
# calc costs
C1 = 10 * Q1 + 5 * Q2
C2 = 6 * Q1 + Q1**2 + 8 * Q2 + Q2**2
C3 = 7 * Q1 + 9 * Q2 + Q1 * Q2
```

Let’s look at what is in these variables:

```
print("q1")
print(q1)
print("q2")
print(q2)
print("Q1")
print(Q1)
print("Q2")
print(Q2)
```

```
q1
[   0.            1.001001      2.002002      3.003003      4.004004
    5.00500501    6.00600601    7.00700701    8.00800801    9.00900901
   10.01001001   11.01101101   12.01201201   13.01301301   14.01401401
   15.01501502   16.01601602   17.01701702   18.01801802   19.01901902
   20.02002002   21.02102102   22.02202202   23.02302302   24.02402402
   25.02502503   26.02602603   27.02702703   28.02802803   29.02902903
   30.03003003   31.03103103   32.03203203   33.03303303   34.03403403
   35.03503504   36.03603604   37.03703704   38.03803804   39.03903904
   40.04004004   41.04104104   42.04204204   43.04304304   44.04404404
   45.04504505   46.04604605   47.04704705   48.04804805   49.04904905
   50.05005005   51.05105105   52.05205205   53.05305305   54.05405405
   55.05505506   56.05605606   57.05705706   58.05805806   59.05905906
   60.06006006   61.06106106   62.06206206   63.06306306   64.06406406
   65.06506507   66.06606607   67.06706707   68.06806807   69.06906907
   70.07007007   71.07107107   72.07207207   73.07307307   74.07407407
   75.07507508   76.07607608   77.07707708   78.07807808   79.07907908
   80.08008008   81.08108108   82.08208208   83.08308308   84.08408408
   85.08508509   86.08608609   87.08708709   88.08808809   89.08908909
   90.09009009   91.09109109   92.09209209   93.09309309   94.09409409
```

```
 95.0950951    96.0960961    97.0970971    98.0980981    99.0990991
100.1001001   101.1011011   102.1021021   103.1031031   104.1041041
105.10510511  106.10610611  107.10710711  108.10810811  109.10910911
110.11011011  111.11111111  112.11211211  113.11311311  114.11411411
115.11511512  116.11611612  117.11711712  118.11811812  119.11911912
120.12012012  121.12112112  122.12212212  123.12312312  124.12412412
125.12512513  126.12612613  127.12712713  128.12812813  129.12912913
130.13013013  131.13113113  132.13213213  133.13313313  134.13413413
135.13513514  136.13613614  137.13713714  138.13813814  139.13913914
140.14014014  141.14114114  142.14214214  143.14314314  144.14414414
145.14514515  146.14614615  147.14714715  148.14814815  149.14914915
150.15015015  151.15115115  152.15215215  153.15315315  154.15415415
155.15515516  156.15615616  157.15715716  158.15815816  159.15915916
160.16016016  161.16116116  162.16216216  163.16316316  164.16416416
165.16516517  166.16616617  167.16716717  168.16816817  169.16916917
170.17017017  171.17117117  172.17217217  173.17317317  174.17417417
175.17517518  176.17617618  177.17717718  178.17817818  179.17917918
180.18018018  181.18118118  182.18218218  183.18318318  184.18418418
185.18518519  186.18618619  187.18718719  188.18818819  189.18918919
190.19019019  191.19119119  192.19219219  193.19319319  194.19419419
195.1951952   196.1961962   197.1971972   198.1981982   199.1991992
200.2002002   201.2012012   202.2022022   203.2032032   204.2042042
205.20520521  206.20620621  207.20720721  208.20820821  209.20920921
210.21021021  211.21121121  212.21221221  213.21321321  214.21421421
215.21521522  216.21621622  217.21721722  218.21821822  219.21921922
220.22022022  221.22122122  222.22222222  223.22322322  224.22422422
225.22522523  226.22622623  227.22722723  228.22822823  229.22922923
230.23023023  231.23123123  232.23223223  233.23323323  234.23423423
235.23523524  236.23623624  237.23723724  238.23823824  239.23923924
240.24024024  241.24124124  242.24224224  243.24324324  244.24424424
245.24524525  246.24624625  247.24724725  248.24824825  249.24924925
250.25025025  251.25125125  252.25225225  253.25325325  254.25425425
255.25525526  256.25625626  257.25725726  258.25825826  259.25925926
260.26026026  261.26126126  262.26226226  263.26326326  264.26426426
265.26526527  266.26626627  267.26726727  268.26826827  269.26926927
270.27027027  271.27127127  272.27227227  273.27327327  274.27427427
275.27527528  276.27627628  277.27727728  278.27827828  279.27927928
280.28028028  281.28128128  282.28228228  283.28328328  284.28428428
285.28528529  286.28628629  287.28728729  288.28828829  289.28928929
290.29029029  291.29129129  292.29229229  293.29329329  294.29429429
295.2952953   296.2962963   297.2972973   298.2982983   299.2992993
300.3003003   301.3013013   302.3023023   303.3033033   304.3043043
305.30530531  306.30630631  307.30730731  308.30830831  309.30930931
```

```
310.31031031  311.31131131  312.31231231  313.31331331  314.31431431
315.31531532  316.31631632  317.31731732  318.31831832  319.31931932
320.32032032  321.32132132  322.32232232  323.32332332  324.32432432
325.32532533  326.32632633  327.32732733  328.32832833  329.32932933
330.33033033  331.33133133  332.33233233  333.33333333  334.33433433
335.33533534  336.33633634  337.33733734  338.33833834  339.33933934
340.34034034  341.34134134  342.34234234  343.34334334  344.34434434
345.34534535  346.34634635  347.34734735  348.34834835  349.34934935
350.35035035  351.35135135  352.35235235  353.35335335  354.35435435
355.35535536  356.35635636  357.35735736  358.35835836  359.35935936
360.36036036  361.36136136  362.36236236  363.36336336  364.36436436
365.36536537  366.36636637  367.36736737  368.36836837  369.36936937
370.37037037  371.37137137  372.37237237  373.37337337  374.37437437
375.37537538  376.37637638  377.37737738  378.37837838  379.37937938
380.38038038  381.38138138  382.38238238  383.38338338  384.38438438
385.38538539  386.38638639  387.38738739  388.38838839  389.38938939
390.39039039  391.39139139  392.39239239  393.39339339  394.39439439
395.3953954   396.3963964   397.3973974   398.3983984   399.3993994
400.4004004   401.4014014   402.4024024   403.4034034   404.4044044
405.40540541  406.40640641  407.40740741  408.40840841  409.40940941
410.41041041  411.41141141  412.41241241  413.41341341  414.41441441
415.41541542  416.41641642  417.41741742  418.41841842  419.41941942
420.42042042  421.42142142  422.42242242  423.42342342  424.42442442
425.42542543  426.42642643  427.42742743  428.42842843  429.42942943
430.43043043  431.43143143  432.43243243  433.43343343  434.43443443
435.43543544  436.43643644  437.43743744  438.43843844  439.43943944
440.44044044  441.44144144  442.44244244  443.44344344  444.44444444
445.44544545  446.44644645  447.44744745  448.44844845  449.44944945
450.45045045  451.45145145  452.45245245  453.45345345  454.45445445
455.45545546  456.45645646  457.45745746  458.45845846  459.45945946
460.46046046  461.46146146  462.46246246  463.46346346  464.46446446
465.46546547  466.46646647  467.46746747  468.46846847  469.46946947
470.47047047  471.47147147  472.47247247  473.47347347  474.47447447
475.47547548  476.47647648  477.47747748  478.47847848  479.47947948
480.48048048  481.48148148  482.48248248  483.48348348  484.48448448
485.48548549  486.48648649  487.48748749  488.48848849  489.48948949
490.49049049  491.49149149  492.49249249  493.49349349  494.49449449
495.4954955   496.4964965   497.4974975   498.4984985   499.4994995
500.5005005   501.5015015   502.5025025   503.5035035   504.5045045
505.50550551  506.50650651  507.50750751  508.50850851  509.50950951
510.51051051  511.51151151  512.51251251  513.51351351  514.51451451
515.51551552  516.51651652  517.51751752  518.51851852  519.51951952
520.52052052  521.52152152  522.52252252  523.52352352  524.52452452
```

```
525.52552553  526.52652653  527.52752753  528.52852853  529.52952953
530.53053053  531.53153153  532.53253253  533.53353353  534.53453453
535.53553554  536.53653654  537.53753754  538.53853854  539.53953954
540.54054054  541.54154154  542.54254254  543.54354354  544.54454454
545.54554555  546.54654655  547.54754755  548.54854855  549.54954955
550.55055055  551.55155155  552.55255255  553.55355355  554.55455455
555.55555556  556.55655656  557.55755756  558.55855856  559.55955956
560.56056056  561.56156156  562.56256256  563.56356356  564.56456456
565.56556557  566.56656657  567.56756757  568.56856857  569.56956957
570.57057057  571.57157157  572.57257257  573.57357357  574.57457457
575.57557558  576.57657658  577.57757758  578.57857858  579.57957958
580.58058058  581.58158158  582.58258258  583.58358358  584.58458458
585.58558559  586.58658659  587.58758759  588.58858859  589.58958959
590.59059059  591.59159159  592.59259259  593.59359359  594.59459459
595.5955956   596.5965966   597.5975976   598.5985986   599.5995996
600.6006006   601.6016016   602.6026026   603.6036036   604.6046046
605.60560561  606.60660661  607.60760761  608.60860861  609.60960961
610.61061061  611.61161161  612.61261261  613.61361361  614.61461461
615.61561562  616.61661662  617.61761762  618.61861862  619.61961962
620.62062062  621.62162162  622.62262262  623.62362362  624.62462462
625.62562563  626.62662663  627.62762763  628.62862863  629.62962963
630.63063063  631.63163163  632.63263263  633.63363363  634.63463463
635.63563564  636.63663664  637.63763764  638.63863864  639.63963964
640.64064064  641.64164164  642.64264264  643.64364364  644.64464464
645.64564565  646.64664665  647.64764765  648.64864865  649.64964965
650.65065065  651.65165165  652.65265265  653.65365365  654.65465465
655.65565566  656.65665666  657.65765766  658.65865866  659.65965966
660.66066066  661.66166166  662.66266266  663.66366366  664.66466466
665.66566567  666.66666667  667.66766767  668.66866867  669.66966967
670.67067067  671.67167167  672.67267267  673.67367367  674.67467467
675.67567568  676.67667668  677.67767768  678.67867868  679.67967968
680.68068068  681.68168168  682.68268268  683.68368368  684.68468468
685.68568569  686.68668669  687.68768769  688.68868869  689.68968969
690.69069069  691.69169169  692.69269269  693.69369369  694.69469469
695.6956957   696.6966967   697.6976977   698.6986987   699.6996997
700.7007007   701.7017017   702.7027027   703.7037037   704.7047047
705.70570571  706.70670671  707.70770771  708.70870871  709.70970971
710.71071071  711.71171171  712.71271271  713.71371371  714.71471471
715.71571572  716.71671672  717.71771772  718.71871872  719.71971972
720.72072072  721.72172172  722.72272272  723.72372372  724.72472472
725.72572573  726.72672673  727.72772773  728.72872873  729.72972973
730.73073073  731.73173173  732.73273273  733.73373373  734.73473473
735.73573574  736.73673674  737.73773774  738.73873874  739.73973974
```

```
740.74074074  741.74174174  742.74274274  743.74374374  744.74474474
745.74574575  746.74674675  747.74774775  748.74874875  749.74974975
750.75075075  751.75175175  752.75275275  753.75375375  754.75475475
755.75575576  756.75675676  757.75775776  758.75875876  759.75975976
760.76076076  761.76176176  762.76276276  763.76376376  764.76476476
765.76576577  766.76676677  767.76776777  768.76876877  769.76976977
770.77077077  771.77177177  772.77277277  773.77377377  774.77477477
775.77577578  776.77677678  777.77777778  778.77877878  779.77977978
780.78078078  781.78178178  782.78278278  783.78378378  784.78478478
785.78578579  786.78678679  787.78778779  788.78878879  789.78978979
790.79079079  791.79179179  792.79279279  793.79379379  794.79479479
795.7957958   796.7967968   797.7977978   798.7987988   799.7997998
800.8008008   801.8018018   802.8028028   803.8038038   804.8048048
805.80580581  806.80680681  807.80780781  808.80880881  809.80980981
810.81081081  811.81181181  812.81281281  813.81381381  814.81481481
815.81581582  816.81681682  817.81781782  818.81881882  819.81981982
820.82082082  821.82182182  822.82282282  823.82382382  824.82482482
825.82582583  826.82682683  827.82782783  828.82882883  829.82982983
830.83083083  831.83183183  832.83283283  833.83383383  834.83483483
835.83583584  836.83683684  837.83783784  838.83883884  839.83983984
840.84084084  841.84184184  842.84284284  843.84384384  844.84484484
845.84584585  846.84684685  847.84784785  848.84884885  849.84984985
850.85085085  851.85185185  852.85285285  853.85385385  854.85485485
855.85585586  856.85685686  857.85785786  858.85885886  859.85985986
860.86086086  861.86186186  862.86286286  863.86386386  864.86486486
865.86586587  866.86686687  867.86786787  868.86886887  869.86986987
870.87087087  871.87187187  872.87287287  873.87387387  874.87487487
875.87587588  876.87687688  877.87787788  878.87887888  879.87987988
880.88088088  881.88188188  882.88288288  883.88388388  884.88488488
885.88588589  886.88688689  887.88788789  888.88888889  889.88988989
890.89089089  891.89189189  892.89289289  893.89389389  894.89489489
895.8958959   896.8968969   897.8978979   898.8988989   899.8998999
900.9009009   901.9019019   902.9029029   903.9039039   904.9049049
905.90590591  906.90690691  907.90790791  908.90890891  909.90990991
910.91091091  911.91191191  912.91291291  913.91391391  914.91491491
915.91591592  916.91691692  917.91791792  918.91891892  919.91991992
920.92092092  921.92192192  922.92292292  923.92392392  924.92492492
925.92592593  926.92692693  927.92792793  928.92892893  929.92992993
930.93093093  931.93193193  932.93293293  933.93393393  934.93493493
935.93593594  936.93693694  937.93793794  938.93893894  939.93993994
940.94094094  941.94194194  942.94294294  943.94394394  944.94494494
945.94594595  946.94694695  947.94794795  948.94894895  949.94994995
950.95095095  951.95195195  952.95295295  953.95395395  954.95495495
```

```
  955.95595596  956.95695696  957.95795796  958.95895896  959.95995996
  960.96096096  961.96196196  962.96296296  963.96396396  964.96496496
  965.96596597  966.96696697  967.96796797  968.96896897  969.96996997
  970.97097097  971.97197197  972.97297297  973.97397397  974.97497497
  975.97597598  976.97697698  977.97797798  978.97897898  979.97997998
  980.98098098  981.98198198  982.98298298  983.98398398  984.98498498
  985.98598599  986.98698699  987.98798799  988.98898899  989.98998999
  990.99099099  991.99199199  992.99299299  993.99399399  994.99499499
  995.995996    996.996997    997.997998    998.998999   1000.        ]
q2
[   0.            1.001001      2.002002      3.003003      4.004004
    5.00500501    6.00600601    7.00700701    8.00800801    9.00900901
   10.01001001   11.01101101   12.01201201   13.01301301   14.01401401
   15.01501502   16.01601602   17.01701702   18.01801802   19.01901902
   20.02002002   21.02102102   22.02202202   23.02302302   24.02402402
   25.02502503   26.02602603   27.02702703   28.02802803   29.02902903
   30.03003003   31.03103103   32.03203203   33.03303303   34.03403403
   35.03503504   36.03603604   37.03703704   38.03803804   39.03903904
   40.04004004   41.04104104   42.04204204   43.04304304   44.04404404
   45.04504505   46.04604605   47.04704705   48.04804805   49.04904905
   50.05005005   51.05105105   52.05205205   53.05305305   54.05405405
   55.05505506   56.05605606   57.05705706   58.05805806   59.05905906
   60.06006006   61.06106106   62.06206206   63.06306306   64.06406406
   65.06506507   66.06606607   67.06706707   68.06806807   69.06906907
   70.07007007   71.07107107   72.07207207   73.07307307   74.07407407
   75.07507508   76.07607608   77.07707708   78.07807808   79.07907908
   80.08008008   81.08108108   82.08208208   83.08308308   84.08408408
   85.08508509   86.08608609   87.08708709   88.08808809   89.08908909
   90.09009009   91.09109109   92.09209209   93.09309309   94.09409409
   95.0950951    96.0960961    97.0970971    98.0980981    99.0990991
  100.1001001   101.1011011   102.1021021   103.1031031   104.1041041
  105.10510511  106.10610611  107.10710711  108.10810811  109.10910911
  110.11011011  111.11111111  112.11211211  113.11311311  114.11411411
  115.11511512  116.11611612  117.11711712  118.11811812  119.11911912
  120.12012012  121.12112112  122.12212212  123.12312312  124.12412412
  125.12512513  126.12612613  127.12712713  128.12812813  129.12912913
  130.13013013  131.13113113  132.13213213  133.13313313  134.13413413
  135.13513514  136.13613614  137.13713714  138.13813814  139.13913914
  140.14014014  141.14114114  142.14214214  143.14314314  144.14414414
  145.14514515  146.14614615  147.14714715  148.14814815  149.14914915
  150.15015015  151.15115115  152.15215215  153.15315315  154.15415415
  155.15515516  156.15615616  157.15715716  158.15815816  159.15915916
  160.16016016  161.16116116  162.16216216  163.16316316  164.16416416
```

```
165.16516517  166.16616617  167.16716717  168.16816817  169.16916917
170.17017017  171.17117117  172.17217217  173.17317317  174.17417417
175.17517518  176.17617618  177.17717718  178.17817818  179.17917918
180.18018018  181.18118118  182.18218218  183.18318318  184.18418418
185.18518519  186.18618619  187.18718719  188.18818819  189.18918919
190.19019019  191.19119119  192.19219219  193.19319319  194.19419419
195.1951952   196.1961962   197.1971972   198.1981982   199.1991992
200.2002002   201.2012012   202.2022022   203.2032032   204.2042042
205.20520521  206.20620621  207.20720721  208.20820821  209.20920921
210.21021021  211.21121121  212.21221221  213.21321321  214.21421421
215.21521522  216.21621622  217.21721722  218.21821822  219.21921922
220.22022022  221.22122122  222.22222222  223.22322322  224.22422422
225.22522523  226.22622623  227.22722723  228.22822823  229.22922923
230.23023023  231.23123123  232.23223223  233.23323323  234.23423423
235.23523524  236.23623624  237.23723724  238.23823824  239.23923924
240.24024024  241.24124124  242.24224224  243.24324324  244.24424424
245.24524525  246.24624625  247.24724725  248.24824825  249.24924925
250.25025025  251.25125125  252.25225225  253.25325325  254.25425425
255.25525526  256.25625626  257.25725726  258.25825826  259.25925926
260.26026026  261.26126126  262.26226226  263.26326326  264.26426426
265.26526527  266.26626627  267.26726727  268.26826827  269.26926927
270.27027027  271.27127127  272.27227227  273.27327327  274.27427427
275.27527528  276.27627628  277.27727728  278.27827828  279.27927928
280.28028028  281.28128128  282.28228228  283.28328328  284.28428428
285.28528529  286.28628629  287.28728729  288.28828829  289.28928929
290.29029029  291.29129129  292.29229229  293.29329329  294.29429429
295.2952953   296.2962963   297.2972973   298.2982983   299.2992993
300.3003003   301.3013013   302.3023023   303.3033033   304.3043043
305.30530531  306.30630631  307.30730731  308.30830831  309.30930931
310.31031031  311.31131131  312.31231231  313.31331331  314.31431431
315.31531532  316.31631632  317.31731732  318.31831832  319.31931932
320.32032032  321.32132132  322.32232232  323.32332332  324.32432432
325.32532533  326.32632633  327.32732733  328.32832833  329.32932933
330.33033033  331.33133133  332.33233233  333.33333333  334.33433433
335.33533534  336.33633634  337.33733734  338.33833834  339.33933934
340.34034034  341.34134134  342.34234234  343.34334334  344.34434434
345.34534535  346.34634635  347.34734735  348.34834835  349.34934935
350.35035035  351.35135135  352.35235235  353.35335335  354.35435435
355.35535536  356.35635636  357.35735736  358.35835836  359.35935936
360.36036036  361.36136136  362.36236236  363.36336336  364.36436436
365.36536537  366.36636637  367.36736737  368.36836837  369.36936937
370.37037037  371.37137137  372.37237237  373.37337337  374.37437437
375.37537538  376.37637638  377.37737738  378.37837838  379.37937938
```

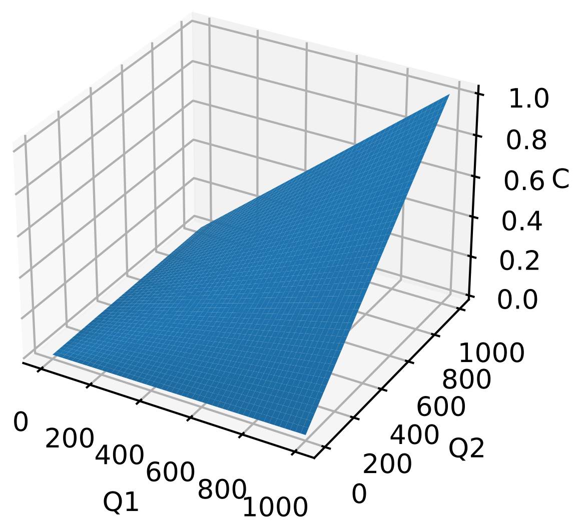
```
380.38038038  381.38138138  382.38238238  383.38338338  384.38438438
385.38538539  386.38638639  387.38738739  388.38838839  389.38938939
390.39039039  391.39139139  392.39239239  393.39339339  394.39439439
395.3953954   396.3963964   397.3973974   398.3983984   399.3993994
400.4004004   401.4014014   402.4024024   403.4034034   404.4044044
405.40540541  406.40640641  407.40740741  408.40840841  409.40940941
410.41041041  411.41141141  412.41241241  413.41341341  414.41441441
415.41541542  416.41641642  417.41741742  418.41841842  419.41941942
420.42042042  421.42142142  422.42242242  423.42342342  424.42442442
425.42542543  426.42642643  427.42742743  428.42842843  429.42942943
430.43043043  431.43143143  432.43243243  433.43343343  434.43443443
435.43543544  436.43643644  437.43743744  438.43843844  439.43943944
440.44044044  441.44144144  442.44244244  443.44344344  444.44444444
445.44544545  446.44644645  447.44744745  448.44844845  449.44944945
450.45045045  451.45145145  452.45245245  453.45345345  454.45445445
455.45545546  456.45645646  457.45745746  458.45845846  459.45945946
460.46046046  461.46146146  462.46246246  463.46346346  464.46446446
465.46546547  466.46646647  467.46746747  468.46846847  469.46946947
470.47047047  471.47147147  472.47247247  473.47347347  474.47447447
475.47547548  476.47647648  477.47747748  478.47847848  479.47947948
480.48048048  481.48148148  482.48248248  483.48348348  484.48448448
485.48548549  486.48648649  487.48748749  488.48848849  489.48948949
490.49049049  491.49149149  492.49249249  493.49349349  494.49449449
495.4954955   496.4964965   497.4974975   498.4984985   499.4994995
500.5005005   501.5015015   502.5025025   503.5035035   504.5045045
505.50550551  506.50650651  507.50750751  508.50850851  509.50950951
510.51051051  511.51151151  512.51251251  513.51351351  514.51451451
515.51551552  516.51651652  517.51751752  518.51851852  519.51951952
520.52052052  521.52152152  522.52252252  523.52352352  524.52452452
525.52552553  526.52652653  527.52752753  528.52852853  529.52952953
530.53053053  531.53153153  532.53253253  533.53353353  534.53453453
535.53553554  536.53653654  537.53753754  538.53853854  539.53953954
540.54054054  541.54154154  542.54254254  543.54354354  544.54454454
545.54554555  546.54654655  547.54754755  548.54854855  549.54954955
550.55055055  551.55155155  552.55255255  553.55355355  554.55455455
555.55555556  556.55655656  557.55755756  558.55855856  559.55955956
560.56056056  561.56156156  562.56256256  563.56356356  564.56456456
565.56556557  566.56656657  567.56756757  568.56856857  569.56956957
570.57057057  571.57157157  572.57257257  573.57357357  574.57457457
575.57557558  576.57657658  577.57757758  578.57857858  579.57957958
580.58058058  581.58158158  582.58258258  583.58358358  584.58458458
585.58558559  586.58658659  587.58758759  588.58858859  589.58958959
590.59059059  591.59159159  592.59259259  593.59359359  594.59459459
```

595.5955956	596.5965966	597.5975976	598.5985986	599.5995996
600.6006006	601.6016016	602.6026026	603.6036036	604.6046046
605.60560561	606.60660661	607.60760761	608.60860861	609.60960961
610.61061061	611.61161161	612.61261261	613.61361361	614.61461461
615.61561562	616.61661662	617.61761762	618.61861862	619.61961962
620.62062062	621.62162162	622.62262262	623.62362362	624.62462462
625.62562563	626.62662663	627.62762763	628.62862863	629.62962963
630.63063063	631.63163163	632.63263263	633.63363363	634.63463463
635.63563564	636.63663664	637.63763764	638.63863864	639.63963964
640.64064064	641.64164164	642.64264264	643.64364364	644.64464464
645.64564565	646.64664665	647.64764765	648.64864865	649.64964965
650.65065065	651.65165165	652.65265265	653.65365365	654.65465465
655.65565566	656.65665666	657.65765766	658.65865866	659.65965966
660.66066066	661.66166166	662.66266266	663.66366366	664.66466466
665.66566567	666.66666667	667.66766767	668.66866867	669.66966967
670.67067067	671.67167167	672.67267267	673.67367367	674.67467467
675.67567568	676.67667668	677.67767768	678.67867868	679.67967968
680.68068068	681.68168168	682.68268268	683.68368368	684.68468468
685.68568569	686.68668669	687.68768769	688.68868869	689.68968969
690.69069069	691.69169169	692.69269269	693.69369369	694.69469469
695.6956957	696.6966967	697.6976977	698.6986987	699.6996997
700.7007007	701.7017017	702.7027027	703.7037037	704.7047047
705.70570571	706.70670671	707.70770771	708.70870871	709.70970971
710.71071071	711.71171171	712.71271271	713.71371371	714.71471471
715.71571572	716.71671672	717.71771772	718.71871872	719.71971972
720.72072072	721.72172172	722.72272272	723.72372372	724.72472472
725.72572573	726.72672673	727.72772773	728.72872873	729.72972973
730.73073073	731.73173173	732.73273273	733.73373373	734.73473473
735.73573574	736.73673674	737.73773774	738.73873874	739.73973974
740.74074074	741.74174174	742.74274274	743.74374374	744.74474474
745.74574575	746.74674675	747.74774775	748.74874875	749.74974975
750.75075075	751.75175175	752.75275275	753.75375375	754.75475475
755.75575576	756.75675676	757.75775776	758.75875876	759.75975976
760.76076076	761.76176176	762.76276276	763.76376376	764.76476476
765.76576577	766.76676677	767.76776777	768.76876877	769.76976977
770.77077077	771.77177177	772.77277277	773.77377377	774.77477477
775.77577578	776.77677678	777.77777778	778.77877878	779.77977978
780.78078078	781.78178178	782.78278278	783.78378378	784.78478478
785.78578579	786.78678679	787.78778779	788.78878879	789.78978979
790.79079079	791.79179179	792.79279279	793.79379379	794.79479479
795.7957958	796.7967968	797.7977978	798.7987988	799.7997998
800.8008008	801.8018018	802.8028028	803.8038038	804.8048048
805.80580581	806.80680681	807.80780781	808.80880881	809.80980981

```
  810.81081081  811.81181181  812.81281281  813.81381381  814.81481481
  815.81581582  816.81681682  817.81781782  818.81881882  819.81981982
  820.82082082  821.82182182  822.82282282  823.82382382  824.82482482
  825.82582583  826.82682683  827.82782783  828.82882883  829.82982983
  830.83083083  831.83183183  832.83283283  833.83383383  834.83483483
  835.83583584  836.83683684  837.83783784  838.83883884  839.83983984
  840.84084084  841.84184184  842.84284284  843.84384384  844.84484484
  845.84584585  846.84684685  847.84784785  848.84884885  849.84984985
  850.85085085  851.85185185  852.85285285  853.85385385  854.85485485
  855.85585586  856.85685686  857.85785786  858.85885886  859.85985986
  860.86086086  861.86186186  862.86286286  863.86386386  864.86486486
  865.86586587  866.86686687  867.86786787  868.86886887  869.86986987
  870.87087087  871.87187187  872.87287287  873.87387387  874.87487487
  875.87587588  876.87687688  877.87787788  878.87887888  879.87987988
  880.88088088  881.88188188  882.88288288  883.88388388  884.88488488
  885.88588589  886.88688689  887.88788789  888.88888889  889.88988989
  890.89089089  891.89189189  892.89289289  893.89389389  894.89489489
  895.8958959   896.8968969   897.8978979   898.8988989   899.8998999
  900.9009009   901.9019019   902.9029029   903.9039039   904.9049049
  905.90590591  906.90690691  907.90790791  908.90890891  909.90990991
  910.91091091  911.91191191  912.91291291  913.91391391  914.91491491
  915.91591592  916.91691692  917.91791792  918.91891892  919.91991992
  920.92092092  921.92192192  922.92292292  923.92392392  924.92492492
  925.92592593  926.92692693  927.92792793  928.92892893  929.92992993
  930.93093093  931.93193193  932.93293293  933.93393393  934.93493493
  935.93593594  936.93693694  937.93793794  938.93893894  939.93993994
  940.94094094  941.94194194  942.94294294  943.94394394  944.94494494
  945.94594595  946.94694695  947.94794795  948.94894895  949.94994995
  950.95095095  951.95195195  952.95295295  953.95395395  954.95495495
  955.95595596  956.95695696  957.95795796  958.95895896  959.95995996
  960.96096096  961.96196196  962.96296296  963.96396396  964.96496496
  965.96596597  966.96696697  967.96796797  968.96896897  969.96996997
  970.97097097  971.97197197  972.97297297  973.97397397  974.97497497
  975.97597598  976.97697698  977.97797798  978.97897898  979.97997998
  980.98098098  981.98198198  982.98298298  983.98398398  984.98498498
  985.98598599  986.98698699  987.98798799  988.98898899  989.98998999
  990.99099099  991.99199199  992.99299299  993.99399399  994.99499499
  995.995996    996.996997    997.997998    998.998999   1000.        ]
Q1
[[   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.      ]
 [   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.      ]
 [   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.       ]
 ...
 [   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.       ]
 [   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.       ]
 [   0.           1.001001    2.002002 ...  997.997998  998.998999
  1000.       ]]
Q2
[[   0.           0.           0.        ...    0.           0.
     0.       ]
 [   1.001001     1.001001     1.001001 ...    1.001001     1.001001
     1.001001]
 [   2.002002     2.002002     2.002002 ...    2.002002     2.002002
     2.002002]
 ...
 [ 997.997998   997.997998   997.997998 ...  997.997998   997.997998
   997.997998]
 [ 998.998999   998.998999   998.998999 ...  998.998999   998.998999
   998.998999]
 [1000.        1000.        1000.        ... 1000.        1000.
  1000.       ]]
```

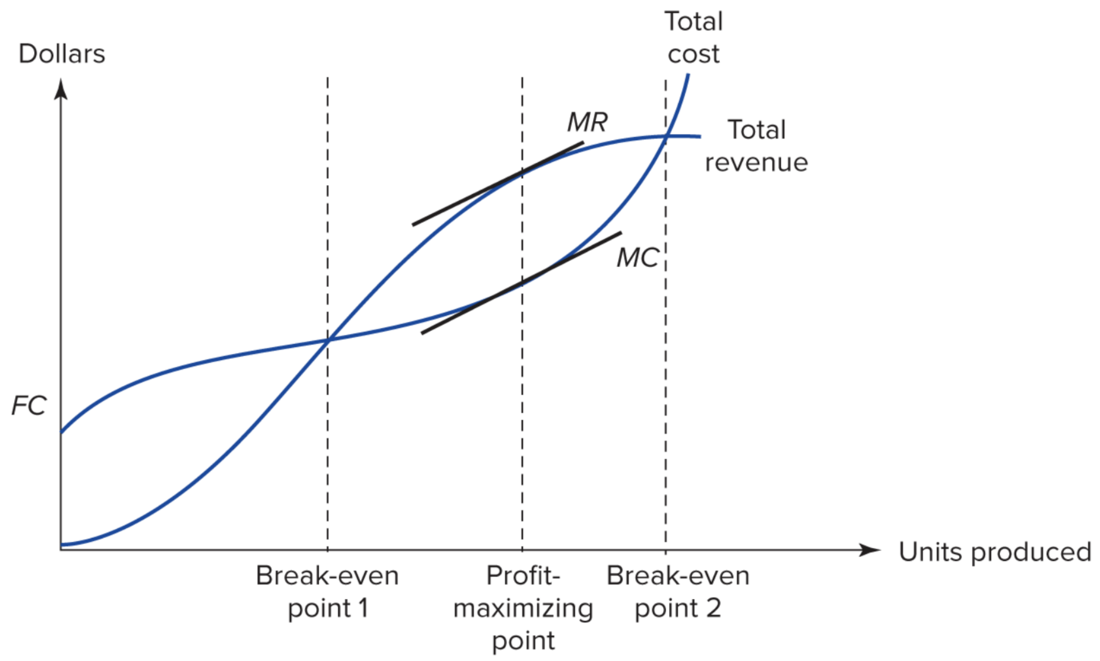
Make the plot:

```
# Create the figure and add a 3D axis
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot the data
# ax.plot_surface(Q1, Q2, C1)
# ax.plot_surface(Q1, Q2, C2)
ax.plot_surface(Q1, Q2, C3)

# Set axis labels and show the plot
ax.set_xlabel('Q1')
ax.set_ylabel('Q2')
ax.set_zlabel('Cost')
plt.show()
```



1.0
0.8
0.6
0.4
0.2
0.0
C
0
200
400
600
800
1000
Q1
0
200
400
600
800
1000
Q2

Turns out we are able to model non-linear functions pretty well!



MC: Marginal cost is the slope of the total cost curve.
MR: Marginal revenue is the slope of the total revenue curve.
MC and *MR* are equal at the profit-maximizing point.