Replicator Dynamics Model

In this lesson we will start to implement a model with an economic interpretation.

Though the model is extremely simple, it still can generate results not totally immediate. We will exploit Lsd to implement the model, observe the results, and find a convincing representation of the results to reply to an apparently difficult property of the model.

Replicator Dynamics Model

Let's consider a model made of a market and several firms. Firms are assigned a constant level of quality. Firms' sales are computed according to the equation:

$$Sales_t = Sales_{t-1}\left[1 + a\left(\frac{Quality - AvQuality_t}{AvQuality_t}\right)\right]$$

That is, sales change of $\frac{a * Sales_{t-1}}{100}$ for each percentage of difference between the firms' own quality and the market average quality.

Replicator Dynamics Model

The average quality must be a *weighted* average of firms' qualities, with the sales weighting the different firms.

A weighted average is like the average but the units computed are not of the same importance. For example, consider that in Italy there are only Ferrari and Fiat 500. Suppose that the quality of Ferrari is 1000 and the quality of Fiat is 10. The average quality of a car in Italy is not $\frac{1000+10}{2} = 505$, that is, 50 times that of a Fiat car. We need to consider that there are far more Fiat's than Ferrari's. Suppose that there are 20,000,000 Fiat in Italy, and 2,000 Ferrari. Then the weighted average of quality for the cars in Italy is:

$$AvQuality_{Italy} = \frac{20,000,000 * 10 + 2,000 * 1000}{20,000,000 + 2,000} = 10.099$$

Very close to that of the most frequently used Fiat car.

Replicator Dynamics Model

In general, if we have N firms on which we have to compute the average of the quality weighted with the sales, the equation will be:

$$AvQuality_t = \frac{\sum_{i=1}^{N} Sales_t^i \times Quality^i}{\sum_{i=1}^{N} Sales_t^i}$$

The only problem is that this equation, together with the former equation for sales, cannot work. To see why, let's make Lsd discover the error.

Replicator Dynamics Model

Let's create a new model in LMM. Call it **RD**. Insert the first equation as:

```
EQUATION("Sales")
/*
Level of sales, computed as the past sales plus a share of the
difference of quality in respect of the average quality
*/
v[0]=V("Quality");
v[1]=V("a");
v[2]=VL("Sales",1);
v[3]=V("AvQuality");
v[4]=v[2]*(1+v[1]*(v[0]-v[3])/v[3]);
RESULT(v[4] )
```

Replicator Dynamics Model

The equation for average quality is must be the computational equivalent for $AvQuality_t = \frac{\sum_{i=1}^{N} Sales_t^i \times Quality^i}{\sum_{i=1}^{N} Sales_t^i}$. In computational terms it must contain the following steps:

1. Set to 0 two variables where to store the dividend and the divisor;
2. take the first object **Firm**;
3. compute in this object the values of **Quality** and and **Sales**;
4. increase the dividend with the product of the two values;
5. increase the divisor with the value for **Sales**;
6. change the currently used **Firm** with the next copy and goto 3

Replicator Dynamics Model

The Lsd code equivalent is the following:

```
EQUATION("AvQuality")
/*
Average quality, weighted with the value of sales
*/
v[0]=0;
v[1]=0;
CYCLE(cur, "Firm")
 {
  v[2]=VS(cur,"Quality");
  v[3]=VS(cur,"Sales");
  v[0]=v[0]+v[2]*v[3];
  v[1]=v[1]+v[3];
 }
RESULT(v[0]/v[1] )
```

Replicator Dynamics Model

Let's comment upong the code.

```
...
v[0]=0;
v[1]=0;
...
```

This two lines set to 0 two local variables. These variables will be used as containers for the cumulated values of sales and sales times quality. In other terms, they will act as boxes. Firstly they are made empty (e.g. equal to 0). Then they are filled in putting into them all the values computed by each firm.

Replicator Dynamics Model

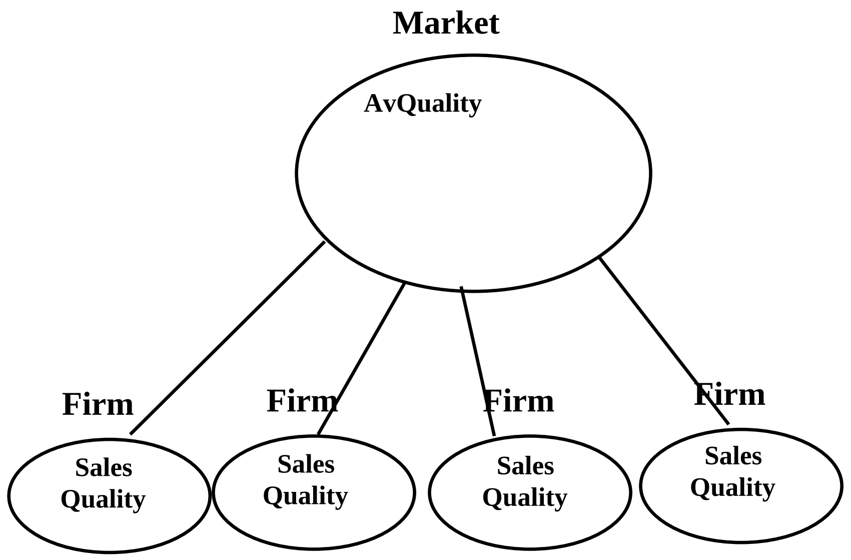
```
CYCLE(cur, "Firm")
 {
  v[2]=VS(cur,"Quality");
  v[3]=VS(cur,"Sales");
  ...
 }
```

This code implements a cycle. That is, the lines within the cycle are repeated again and again for as many times as many copies of object **Firm** are present in the model. At each repetition, the local variable **cur** refers to a different copy of a **Firm**. So, for example, each time `v[2]` contains the quality of a different Firm. The local variable **cur** is the equivalent of one of the v[..], but, instead of accepting numerical values, can be assigned a copy of an object.

The Lsd command `VS(cur, "VarLabel")` is like `V(''VarLabel'')` but instead of searching in the model an object containing **VarLabel** it searches in the object contained in **cur**.

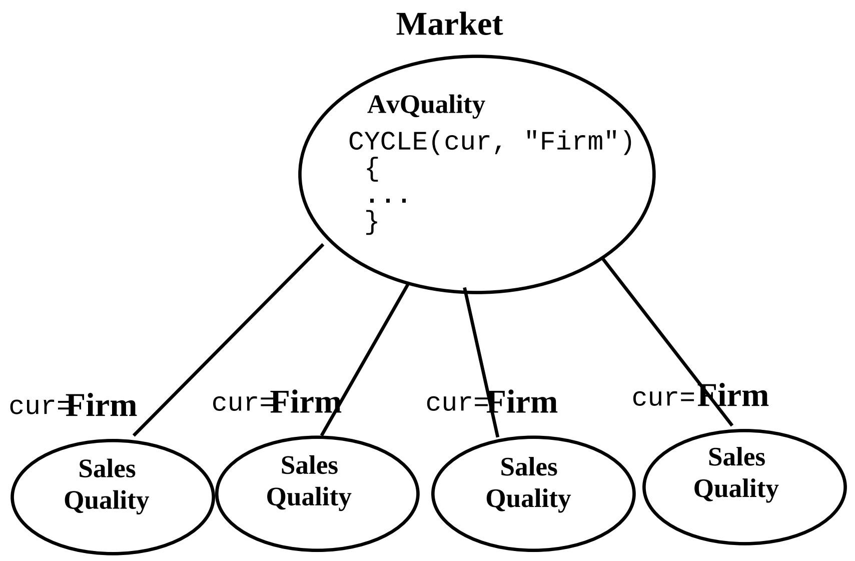
Replicator Dynamics Model

The model is composed by **Market** containing several **Firm**. The equation for **AvQuality** is located in **Market**. When its code requests one value for **Quality** the system has many copies to be used, and not indication on which one to get. By default it gets the first.



Replicator Dynamics Model

The solution is to use the *pointer* `cur`. This is a sort of variable where the modeller can store objects instead of numbers. When the command line `CYCLE(cur,''Firm'')` is used, the system repeats the following lines as many times as many objects **Firm** are found in **Market**. During each repetition `cur` will contain a different copy of **Firm**.



Replicator Dynamics Model

```
CYCLE(cur, "Firm")
 {
  v[2]=VS(cur,"Quality");
  v[3]=VS(cur,"Sales");
  v[0]=v[0]+v[2]*v[3];
  v[1]=v[1]+v[3];
 }
RESULT(v[0]/v[1] )
```

The lines for v[0] and v[1] in the cycles fill in these two local variables. For example, v[1] is assigned its own value (the amount cumulated so far) plus the level of sales for the **Firm** currently used in the cycle.

In the end of the cycles v[1] will contain the sum of all the sales for each **Firm** in the model.

Replicator Dynamics Model

We can now compile the model: menu **Model/Run** will create the Lsd model program for this model, or give messages on the grammar errors preventing the compilation. In case, check the line numbers on the messages of error to fix the equations.

Create a model using the commands in menu **Model** to create variables, parameters and objects. Beware to have the Lsd Browser showing the correct object before including new items.

Replicator Dynamics Model

The model must be composed by:

- Object **Market** containing variable **AvQuality** (0 lags), parameter **a** and object **Firm**.
- Object **Firm** containing variable **Sales** (0 lags) and parameter **Quality**.

Finally, generate 10 copies of object **Firm**; assign to each **Quality** an increasing value from 10 to 19; assign value 0.2 to parameter **a**. We can now run the model, but for the fact that the model will issue a fatal error...

Dead lock errors

If no other errors appeared, the model has shown a small window communicating the existence of a fatal error, preventing the continuation of the simulation. See Log window, and you will likely see the following text:

```
Dead lock! Variable:
AvQuality
requested by Object:
Firm

Fatal error detected at time 1.
Offending code contained in the equation for Variable: Sales
... (deleted text) ...
Level Variable Label
2 Sales
1 AvQuality
0 Lsd Simulation Manager
```

Dead-lock errors

The dead-lock error is an example of a logical error. The two equations we have implemented have nothing wrong on their own, but they cannot work in the same model. To see why let's consider again the equations.

$$AvQuality_t = \frac{\sum_{i=1}^{N} Sales_t^i \times Quality^i}{\sum_{i=1}^{N} Sales_t^i} \tag{1}$$

$$Sales_t = Sales_{t-1} \left[1 + a \left(\frac{Quality - AvQuality_t}{AvQuality_t} \right) \right] \tag{2}$$

Dead-lock errors

Suppose that at the beginning of the simulation 1 the model starts to compute equation (1). In order to execute the code for the equation it needs the values of **Sales** and **Quality** for each firm. Concerning **Quality** there is no problem, since it is a parameter. But **Sales** is not computed yet. In this case the equation for **AvQuality** is interrupted and the system begins the execution of the equation (2) to update the first **Sales**. This procedure is normal, since the modeller in Lsd does not tell the system explicitly the order of execution of the equations.

The problem is that in order to compute equation (2) the system needs, among other values, also the value of **AvQuality** at time $t = 1$. But this is not available, because its execution has been interrupted.

Even if we start from equation (2) we have the same problem. None of the two equations can be computed before the other.

Dead-lock errors

Dead-locks can appear only in computational models, not in mathematical ones. In mathematics it is perfectly normal to have a system of equations like:

$$X_t = 2 + 3Y_t \tag{1}$$

$$Y_t = 4 - 2X_t \tag{2}$$

The mathematical sense of the system is: *find the values of X_t and Y_t such that the conditions (1) and (2) are satisfied*. Any high-school students is taught smart tricks to solve different types of these systems.

In computation terms the system above is an error. The computer reads the equations as: *to compute X first you need to compute Y. To compute Y first you need to compute X*. Locked.

Mathematics vs. Computers

Mathematical models and computational models are very different, as we have seen above. The main difference is that math expressions are time-less statements, while computational models are programs executing one step at a time. Which modelling style is better in Economics? Depends on the phenomenon you want to represent.

- Mathematical models represent universal laws. For example, a demand curve $P = \frac{1}{Q}$
- Computational models represent dynamical phenomena. For example, the development of a market.

Learning by coding

Implementing a model teaches the modeller what is relevant in the reality. Our two equations were well adapted to represent the meaning of their variables, but the model does not work because we did not consider the problem of time precedence. For real economic phenomena time is relevant, and the computational models teach us to consider it properly.

The need to implement a model in computational terms forces modellers to think correctly to the reality under investigation.

Solving dead-lock errors

How do we solve a dead-lock error? We need to force the system to complete first one equation and then the other. We have two options: first equation (1) and then equation (2), or the other way around. Let's have the model to compute first **AvQuality**. That is, our model becomes:

$$AvQuality_t = \frac{\sum_{i=1}^{N} Sales^i_{\mathbf{t\text{-}1}} \times Quality^i}{\sum_{i=1}^{N} Sales^i_{\mathbf{t\text{-}1}}} \tag{1}$$

$$Sales_t = Sales_{t-1}\left[1 + a\left(\frac{Quality - AvQuality_t}{AvQuality_t}\right)\right] \tag{2}$$

In this way, equation (1) can be computed because it does need to compute $\mathbf{Sales}_t$. When (1) produced $\mathbf{AvQuality}_t$ then we can compute (2).

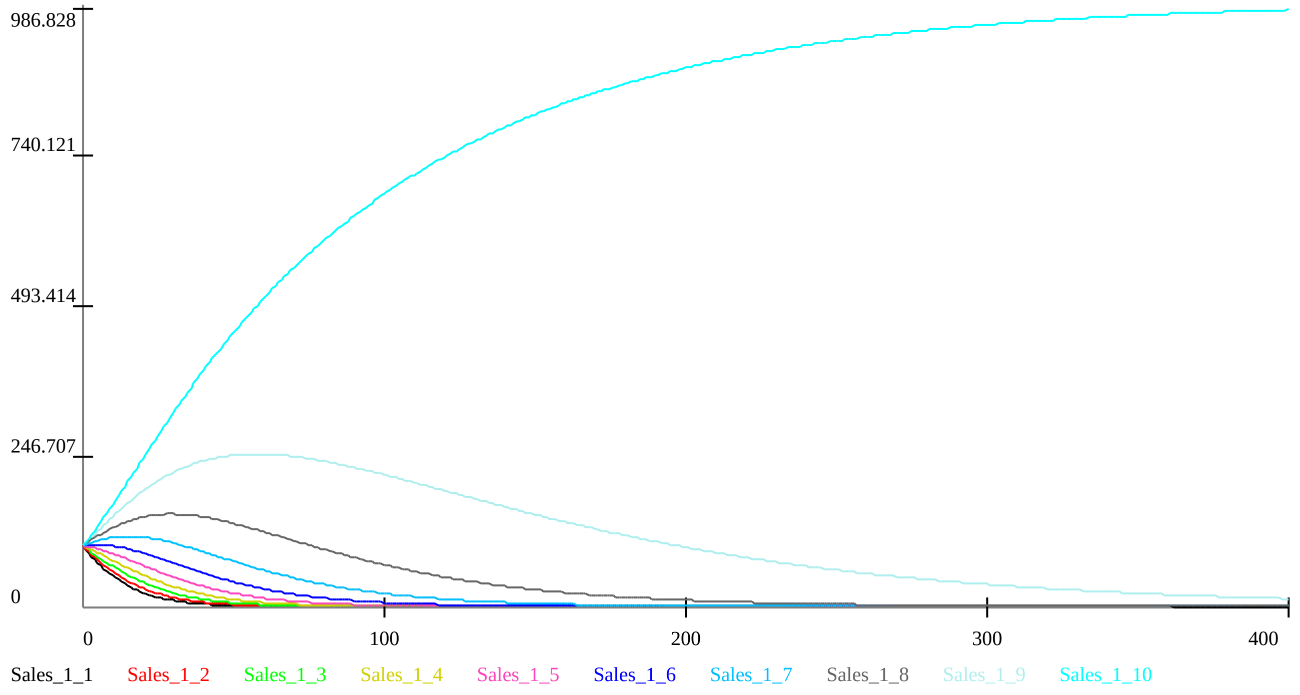
Fixing the dead-lock error

The equation for average quality becomes:

```
EQUATION("AvQuality")
/*
Average quality, weighted with the value of sales
*/
v[0]=0;
v[1]=0;
CYCLE(cur, "Firm")
 {
  v[2]=VS(cur,"Quality");
  v[3]=VLS(cur,"Sales",1); //use lagged values
  v[0]=v[0]+v[2]*v[3];
  v[1]=v[1]+v[3];
 }
RESULT(v[0]/v[1] )
```

Running the model

Now the model works. We have that the sales of 10 firms with qualities ranging from 10 to 19 and initial sales at 100 produce the following result.



Interpretation and Question

The mode can be interpreted as expressing the fact that high-quality firms eventually gain shares at the expenses of low quality ones.

It needs to be noted that his model is *not* a model of competition, at least in the long term. It is a model representing the path leading to a monopoly, since it is impossible to prevent the highest quality firm to dominate the market.

But we may also consider a potential puzzle. If the sales' dynamics is driven by relative quality, a constant parameter, why is that the some firms (e.g. the 9^{th}), have their sales increasing and then falling?

Interpretation and Question

Simulation models only produce the consequences of their content. Therefore, the question above may be easily answered, especially by gifted mathematicians.

However, simulation models frequently surprise, and supporting the motivation of unexpected results is crucial to exploit the model. As an exercise, besides answering the question, find a convincing way to present the results so that the answer can be easily understood.