ISBN: 978-972-9059-05-6

Systems Science European Union
Union Européenne de Systémique
UES

APOCOSIS

Spreadsheets, systemics and simulations

Jacky Legrand
Université Paris 2
legrand@u-paris2.fr

Abstract

Spreadsheets are especially dedicated for answering any "what-if" question and are considered mindtools for decision making. More, their simulation capabilities introduce necessarily a systemic dimension. Although spreadsheet simulation programs may revolutionize systems study, our systems science literature searches have revealed no studies paying them a little attention.
The aim of this paper is to redress the balance. Here is a case for spreadsheets as a practical support in systems thinking. We advocates for a widespread use of computing and graphical demonstration among non-programmers systemic researchers.
Some personal demonstrations suggest the power of simulation through a holistic behaviour emerging in a cells range from a distributed definition. Further, concrete situations of circularity (feedback) and recurrence (loops) are exemplified in an attempt to examine the most favoured fields of systems simulation.

Keywords: end user programming, simulation, spreadsheet, user interfaces, visualization systems.

1. Introduction

System science combines theoretical and practical approaches. Among practical displays, a system as such sets problems of observation and simulation. Within the constructivist approach, the simulation phase implies challenging the model, system-dynamics analysis, unexpected behaviour emerging, cheap scenarii building.

Spreadsheets have been used by accountants for hundreds of years. The first computerization dates from the early 1960's [1]. The microcomputer age spreadsheet of Daniel Bricklin and Robert Frankston dates from the late 1970's [2].

Since the introduction of VisiCalc, electronic spreadsheets turned out to be the highest uptake of personal computer usage in modern business and the most widely used decision support tool.

Of all end-user programming applications, spreadsheets are without a doubt the most pervasive and successful applications of personal computers.

In the rest of the paper, we use "spreadsheeting", for short, instead of "practice of building a spreadsheet model".

Spreadsheeting is an extremely popular and pervasive form of end-user programming in business as well as throughout a very broad range of applications. Three exotic sample applications may be mentioned to illustrate the diversity: football modelling [5], historico-military modelling [7], and water allocation for irrigation [1], [2]. This last example advocates for spreadsheet prototyping. Before the use of Agent-Based Models as a collective decision support system, a spreadsheet is laid out to raise discussions.

[1] Richard Mattessich, "Budgeting Models and System Simulation", *The Accounting Review*, July 1961, p. 384-397.
[2] The idea came to Bricklin while he was working on a MBA degree at the Harvard Business School in 1978 and the first advertisement for VisiCalc appeared in the May 1979 issue of Byte.

In spite of the presence of spreadsheets as a prevalent part of the decision in businesses they have received less research and a very little academic attention.

Are noticeable exceptions, the ongoing researches:

– about visualization from the visual language community [8];

– from the software engineering community attempting to provide tools and methodologies to enforce a systematic development of spreadsheets [4], [6], [12] [3];

– about Computer Assisted Teaching via spreadsheets [11];

– about human factors and human-computer interaction [10].

The main motivation for this paper arises from the fact that although spreadsheets are ubiquitous there is still no interest in the field of systems science.

Section 2 tries to impress upon systems scientists that spreadsheeting makes possible to play "what-if" simulations. In the rest of the paper, we show, using three sample applications of spreadsheets, how to take benefit from spreadsheet-based simulations.

Section 3 presents the "spreading-emerging" paradigm, section 4 presents the "circularity-feedback" paradigm and section 5 presents the "recurrence-loop" paradigm.

2. The case for spreadsheeting in systemics

Although end-user programming has become a widespread phenomenon, although spreadsheets have proven to be a highly versatile, user-friendly and successful tool, it is surprising that this new computational paradigm seems to have been neglected in the systems thinking community. Systems scientists shy away from spreadsheeting [4].

The paradox explanation may stay in the following dichotomy: either the results are systems thinking prospective (efficient simulation is not on the agenda) or the results are from fields where the systemic approach is practiced without being stated (efficient simulation has been done with usual tools of the field).

A spreadsheet has the following critical role: when the user alters cells, all affected dependent cells are redisplayed with new values computed during the "spread" updating.

The results of alternative configurations (effect of changes) are immediately visible. The user explorations are known as "what-if" (what would happen) calculations.

In addition, any change to the values is readily viewed when it results in an immediate change in the created graphs or charts.

A software for analyzing "what-if" situations deserves more in forecasting future events rather than in "accounting" for the past.

We believe that an unprecedented amount of simulation power, a direct browsing through the alternatives, the facility to display the data graphically and a graphical interface for input and immediate feedback are critical advantages for systems scientists.

Spreadsheeting is available on virtually every computer in any laboratory or at home and they have no need to learn programming in the conventional sense. They can develop a model in a layout, that is natural to them, more effectively, reliably and in shorter time. They may find it more convenient to progress in a step-wise fashion during simulation. They develop a more concrete understanding of the model through a realistic behaviour of an abstraction of the system. Adventurous colleagues will even experiment to exploit their understanding with modifying the formulae.

[3] See also the conferences of the European Spreadsheet Risks Interest Group : www.eusprig.org

[4] Only one paper claims its use since the 1rst System Science European Congress held in 1989 (Schwartz & al. CES6 Paris 2005).

important note : Although most commercial packages arrive equipped with a whole macro language which constitutes a complete programming language, the very meaning of this paper restricts spreadsheets to the original, inherent and essential paradigm.
The required competence for admission here is a basic understanding of copy-and-paste action implying formulae with fixed and relative cell references.

3. Spreading and Emergence

It was back in 1984 (the year of Excel for Macintosh) that Alan Kay [9] [5] points out that arguably spreadsheets might be used for a variety of novel programming and should take their place with the status they really have.

According to his academic non-numerical setting (a histogram using figurative cells and no graphical interface), spreadsheeting is not only a "wordprocessor for numbers". A range of cells is an aggregate of concurrently active agents. A global result may emerge from their immediate and simultaneous behaviour.

Although his example is a simple one, it illustrates how a range, filled with a unique formula describing a local behaviour, may exhibit a global emerging behaviour.

We suggest expanding the idea to a day-to-day schedule in a resource planning.

The parameters are : the production rate (quantity produced during one unit of time), the beginning of the labour day, the length of one time period as a quantity of time unit and the number of slices dividing one day (in practice, the number of cells used to represent the maximal work done in one day). Fours cells named respectively "production_rate", "begin", "time_period" and "slicing" are used for the parameters.

An intermediate cell (named "valcell") is used to compute the work that can be done during one period of time: production_rate * time_period.

Each day of the week is a column. The whole week is a range where one cell is representative of the activity of the period. Below the range, a row is used to quantify the targets of production (each day receives a target).

Each row of the range is preceded by a label explaining the period of the day. The text of the label is a formula since it depends on the number of periods and on the length of the period. The smart solution is to use 3 columns to get a label like X "to" Y, where X and Y are adequate formulae.

The formula uses the beginning of the day, the length of a period and the vertical number (relative coordinate) of the row of the label.

We can now generate the X column (from A1 to A13) using Fill Up copy with the following example formula (row 13 as the first row), A13:

```
=IF(13-CELL("row";A13)+1<=slicing;begin+(13-CELL("row";A13))*(time_period);" ")
```

In Excel, the "Cell" function is used to retrieve information about a cell, the "row" argument is used to get the vertical number (absolute coordinate) of the cell.

The Y column (from C1 to C13) is generated using Fill Up copy with the following example formula (row 13 as the first row), C13:

```
=IF(13-CELL("row";C13)+1<=slicing;begin+(13-CELL("row";C13))*(time_period)+(time_period);" ")
```

[5] This highly cited paper is not so far ahead of his time, see for example [13].

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10	17	to	20							
11	14	to	17							
12	11	to	14							
13	8	to	11							
14					MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
15	targets				110	199	42	70	200	20
16										
17				slicing	4					
18				time_period	3		valcell	30		
19				begin	8					
20				production_rate	10					

Figure 3.1: schedule template.

The dynamical labels as a copy of one and only one formula is by itself a noticeable introduction to the key result. The sheet is ready for the emergence demonstration.

The shadowed area (see Figure 3.1) receives the distributed definition of the schedule organized into a rectangular array of similar cells. Any cell will have the same behaviour i.e. the same formula. The formula will formalize how one cell "reacts" to its "neighbourhood": target below, period on left, eve and overnight workload.

A cell in the range devoted to the planning is "highlighted" when the period it represents is (totally or in part) "busy" (mandatory to produce target at hand).

The targets may exceed the number of periods allocated to one day.

A different pattern must be used when a cell applies to the work performed outside the normal working period (A.W. for Additional Work). The number of A.W. cells represents the number of missing labour periods.

Since it is a natural desire to move the extra work over an unoccupied period, we introduce an additional sophistication.

The "free" cells may detect the additional work in the next column (the day tomorrow). They code the opportunity to get ahead in the work with a different pattern: "ahead".

The "overloaded" cells may detect "free" cells in the previous column (the day before). They code the opportunity to get rid of extra work with a different pattern: "done".

In order to sum up the behaviour of our "agent-cell" we suggest the following decision tree

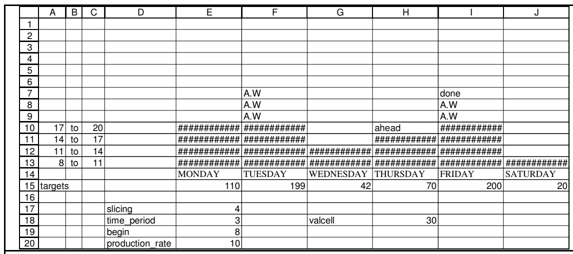
Is the cell a "working" cell?

yes	*no*
is it outside of the labour periods ?	is it inside of the labour periods ?
yes	*yes*
can get ahead the day before ?	additional work the day tomorrow ?
yes	*yes*
"done"	"ahead"
no	*no*
"A.W"	clear contents
no	*no*
standard pattern	clear contents

The described behaviour will be extended into as many subsequent cells as we wish using Fill Up and Fill Right copy (up a column and along the rows to the entire range) of the following example formula (row 13 as the first period of Monday), E13:

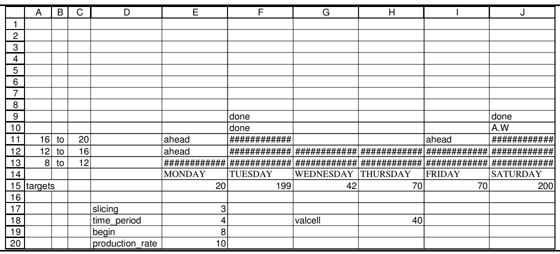
```
=IF((13-CELL("row";E13))*valcell<E$15;
        IF(13-CELL("row";E13)+1>slicing;
        IF(INT((E$15-1)/valcell)-(13-CELL("row";E13))+1<=slicing-INT((D$15-1)/valcell)-1;"done";"A.W");1);
        IF(13-CELL("row";E13)+1<=slicing;
        IF(13-CELL("row";E13)-INT((E$15-1)/valcell)<=INT((F$15-1)/valcell)-slicing+1;"ahead";"");""))
```

The output is the following (see Figure 3.2):



	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7						A.W			done	
8						A.W			A.W	
9						A.W			A.W	
10	17	to	20		############	############		ahead	############	
11	14	to	17		############	############		############	############	
12	11	to	14		############	############	############	############	############	
13	8	to	11		############	############	############	############	############	############
14					MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
15	targets				110	199	42	70	200	20
16										
17				slicing	4					
18				time_period	3		valcell	30		
19				begin	8					
20				production_rate	10					

Figure 3.2 :
schedule test in case of 4 periods of 3 hours



	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9						done				done
10						done				A.W
11	16	to	20		ahead	############			ahead	############
12	12	to	16		ahead	############	############	############	############	############
13	8	to	12		############	############	############	############	############	############
14					MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
15	targets				20	199	42	70	70	200
16										
17				slicing	3					
18				time_period	4		valcell	40		
19				begin	8					
20				production_rate	10					

Figure 3.3 :
schedule test in case of 3 periods of 4 hours

To change the day slicing, we modify E15 in the sheet. To lengthen or shorten the day, we modify E18 in the sheet. Targets may be also modified to get immediately a new output (see Figure 3.3).

Our academic realization allows a dynamically updated simulation of a global emerging solution, the definition of which is distributed all over the sensitive range.

4. Feedback and circularity

During spreadsheeting some cells depending on themselves (through direct or indirect references) are usually entered by mistake. Nevertheless, circular references may be created when simulating models that involve feedback loops. Circularity is desirable and the solution is computed iteratively**.**

For situations that require carrying on circularity anyway, a user-checkable option enforces the iteration while a user-selectable limit (maximum change between two consecutive iterations or automatically bounded iteration loop) enforces termination.

Our example shows a spreadsheet containing two circular cross-references. Its purpose is to iteratively compute the solution of a system of two equations. The equations are laid out so we can get the estimators (α, β) of a simple linear regression. We have no room here for a tutorial [6] about this analysis of the relationship between two observed variables, X and Y, that finds the best straight line ($Y = \alpha X + \beta$) through a scatter plot.
We choose an academic exercise because the key is outside of the solved problem. The key is the tool for solving.

We will focus on the two estimators (α, β) cells ("alpha", "beta") solving the equations that minimize the sum of the squares of the vertical distances of the observed values from the straight line.

As a preliminary step, it is necessary to enter the observed values, Y_i and X_i, two columns are used, the sums of which are named "sum*X*" and "sum*Y*". One of the columns is used to compute the number of values which is named "nbval". A sequence of intermediate formulae is introduced in the adjacent columns to compute:

– the $X_i * Y_i$, the sum of which is named "sumXY";
– the $X_i^{\,2}$, the sum of which is named "sumsqX".

The cells named "alpha" and "beta" receive the equations to be solved:

[6] The reader in need may see any undergraduate statistics textbook or grasp the concept in a glimpse with a full text search in the help menu of his(her) preferred "office suite" package.

– alpha = sumXY-beta*sumX)/sumsqX;
– beta = (sumY-alpha*sumX)/nbval.

With the estimators it is possible to compute in an additional column the estimated values, i.e. the $\alpha X_i + \beta$.

We recommend experiencing such coupled formulae in the manual calculation mode of the spreadsheet (Excel offers Tools-Options-Calculation Tab – Manual check box and F9 keystrokes to progress). The stepwise convergence is enhanced.

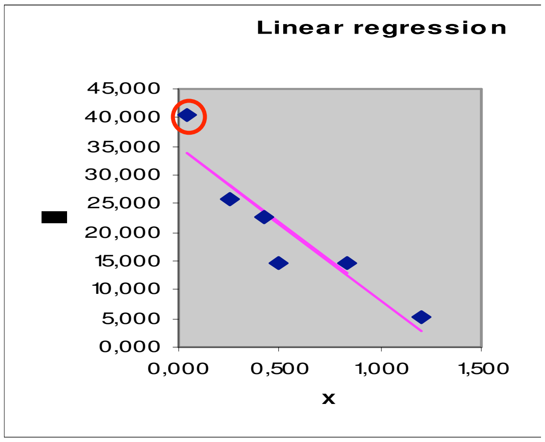


Figure 4.1: before drag and drop.

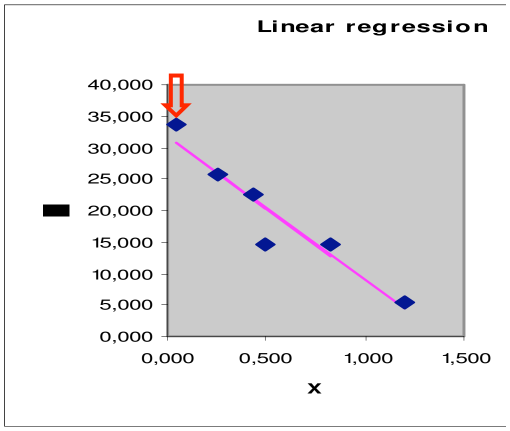


Figure 4.2: after drag and drop.

The visual demonstration is fully realized with the help of the chart wizard. The figure 4-1 presents the scatter plot of the Y_i as a function of X_i. The observed values are married, on a common graphical output, with the scatter plot connecting the estimated values with a straight line.

Since charts are dynamically linked, each iteration automatically redraws the straight line when the convergence of "alpha" and "beta" is experienced.

The actual data have been chosen to be a nasty sample with an adjustment of bad quality. The left top point is a good example of aberrant data. Since it is possible to update a cell's contents by using the mouse to change the position of a plotted point, we can drag the aberrant point and drop it on the straight line. The iteration process will adjust the new estimators and, consequently, the chart (see Figure 4.2).

5. Loops and Recurrence

We suggest here how to represent recurrences and to use a graphical output to experiment with solutions. By the way a system dynamical model is being calculated using a simple spreadsheet.

We choose the representation of the Lotka-Volterra predator-prey model [3], [14].

The model considers two populations: the preys (*Pro(t)* is the density of preys at *t*) and the predators (*Pre(t)* is the density of predators at *t*). The feeding of predators rests totally on the preys. The only augmentation and diminution factors are:

– growth of preys proportional to the number of preys, *Pro(t)* (independent of *Pre(t)*);
– destruction of preys proportional to the number of predators, *Pre(t)* (independent of *Pro(t)*);
– growth of predators proportional to the number of preys, *Pro(t)* (efficiency at turning food into offspring);

– mortality of predators proportional to the number of predators, *Pre(t)* (no internal competition).

The "progression" of the two populations against time may be formulated as a coupled pair of first order, non-linear, differential equations:

$$dPro(t) / dt = \alpha * Pro(t) - \beta * Pro(t) * Pre(t)$$
$$dPre(t) / dt = -\gamma * Pre(t) + \delta * Pro(t) * Pre(t)$$

where α is the growth rate of prey, β is the predation rate coefficient, γ is the predator mortality rate and δ is the rate at which predators increase by consuming prey.

We assume that we know the initial conditions of the equations at time t_0.

An implemented recurrent method is a numerical approximation *of* the differential equations. The Euler method is the simplest recurrence used for the discrete analytic solution of differential equations but least accurate (effects of the cumulative errors generated in each step). We prefer another recurrence: the fourth order Runge-Kutta for which the solution is not obvious.

Let Pro and Pre be the current values for, respectively, preys and predators populations, the next element of the prey sequence is computed with the formula:

(K1Pro + 2*K2Pro + 2*K3Pro + K4Pro) / 6

where K1Pro, K2Pro, K3Pro, K4Pro are computed with the formulae :

K1Pro = alpha*Pro – beta*Pro*Pre
K2Pro = alpha*(Pro + K1Pro/2) – beta*(Pro + K1Pro/2)*(Pre + K1Pre/2)
K3Pro = alpha*(Pro + K2Pro/2) – beta*(Pro + K2Pro/2)*(Pre + K2Pre/2)
K4Pro = alpha*(Pro + K3Pro) – beta*(Pro + K3Pro)*(Pre + K3Pre)

The next element of the predator sequence is computed with the formula:

(K1Pre + 2*K2Pre + 2*K3Pre + K4Pre) / 6

where K1Pre, K2Pre, K3Pre, K4Pre are computed with the formulae:

K1Pre = – gamma*Pre + delta*Pro*Pre
K2Pre = – gamma*(Pre + K1Pre/2) + delta*(Pre + K1Pre/2)*(Pro + K1Pro/2)
K3Pre = – gamma*(Pre + K2Pre/2) + delta*(Pre + K2Pre/2)*(Pro + K2Pro/2)
K4Pre = – gamma*(Pre + K3Pre) + delta*(Pre + K3Pre)*(Pro + K3Pro)

Let spreadsheet column A contains the values for discrete time. The first row is used for titles. The second row is used for initial values. The parameters, alpha, beta, gamma and delta are named cells anywhere, close to the initial values. The K1 to K4 coefficients may be hidden in 8 columns, from H to O. The predators and preys are placed down a column range, two cells wide, in columns B and C. We use row 3 as the first row and give the spreadsheet formulae in B3 and C3:

```
B3: =B2+(H2+2*I2+2*J2+K2)/6                     C3: =C2+(L2+2*M2+2*N2+O2)/6
```

With the following formulae in H2 to O2:

```
H2: =alpha*B2 - beta*B2*C2
I2: =alpha*(B2 + H2/2) - beta*(B2 + H2/2)*(C2 + L2/2)
J2: =alpha*(B2 + I2/2) - beta*(B2 + I2/2)*(C2 + M2/2)
K2: =alpha*(B2 + J2) - beta*(B2 + J2)*(C2 + N2)
L2: = -gamma*C2 + delta*B2*C2
M2: = -gamma*C2 + delta*B2*C2
N2: =-gamma*(C2 + M2/2) + delta*(C2 + M2/2)*(B2 + I2/2)
O2: =-gamma*(C2 + N2) + delta*(C2 + N2)*(B2 + J2)
```

Fill Down copy is an easy means to propagate formulae downward, we generate 500 rows. Each time-step is represented in a single row. This example show how much faster spreadsheeting can obtain the same results as a traditional programming

For concreteness, and as a hint to Volterra, we start with 1 hundred sharks as predators, 400 hundreds sardines as preys and the coefficients $\alpha = 0.05$, $\beta = 0.04$, $\gamma = 0.03$, $\delta = 0.0001$ are used.

The periodic solutions are plotted as successive points to observe the progression over time on Figure 5.1. The simultaneous graphs (a normalization facilitates a common graphical output), using shark and sardine ranges, emphasize the oscillatory nature of the population of the two species, the predator curve follows that of prey by π/4.

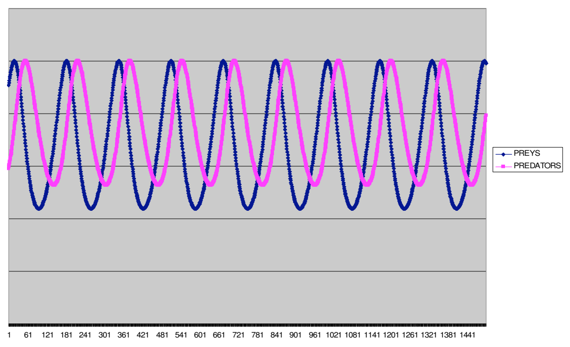


Figure 5.1: periodic evolution of the populations.

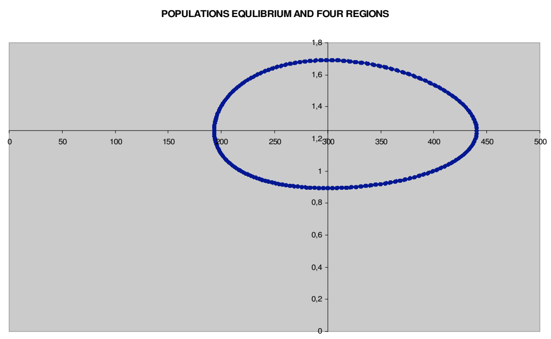


Figure 5.2: phase space diagram.

On Figure 5.2 we plot the shark population as a function of the sardine population with the changing variable being time. The output result is a phase space diagram. The trajectories are elliptical closed lines.

The phase space diagram is divided into four regions by the lines where each species is constant. The two lines intersect where both populations remain unchanged (*Pre = α / β, Pro = γ / δ)*.

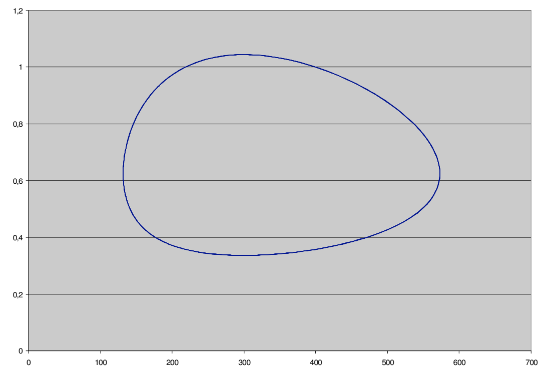


Figure 5.3:
To simulate a second species of predator
*β*2* is the new predation rate coefficient
Result : least environmental impact on preys, interesting!

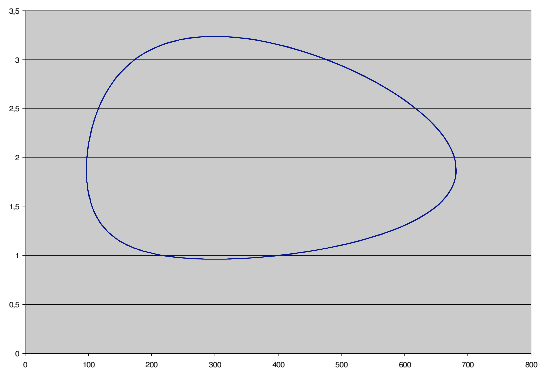


Figure 5.4:
To simulate an increase of live births for preys
*α*1.5* is the new growth rate of prey :
Result : average population of preys is constant while average population of predators increases, unlucky !

The workbook allows the user to generate different increases for populations, to observe the impact when formulae are recomputed and to view, at the same time, the graphical simulation when the chart is redrawn.

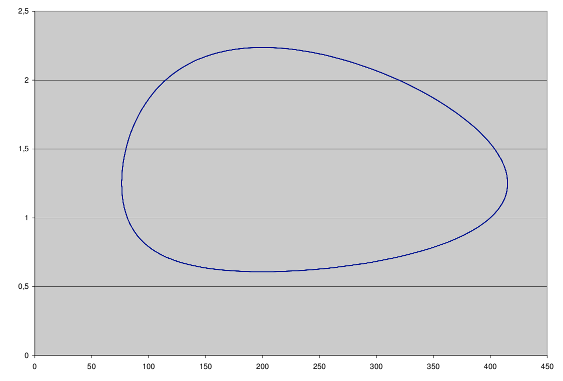
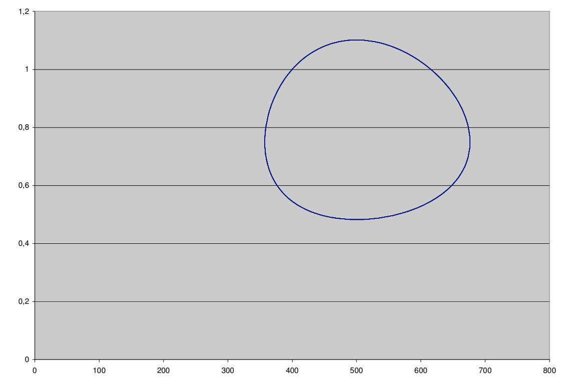
	

Figure 5.5: **To simulate a better reproduction of predators** *δ*1.5* is the new rate at which predators increase : **Result** : decrease of preys population but invariability of predators population, it missed !	Figure 5.6: **To simulate hunting of predators and disrupting reproduction of preys** *α/2* is the new growth rate of prey : *γ*2* is the new predator mortality rate **Result** : decrease of predators and increase of preys, exciting topic for system thinking !

The model is "animated" by spreadsheets. Our implementation displays intermediate results, the user manipulates the data and can instantly and graphically see what changes.

6. Remarks, Critics and Conclusion

When it would take far too long to create an application, the nature of the spreadsheet as a prototyping tool is a crucial advantage. However, can a spreadsheet program be considered as software? Does a spreadsheet program suffer from several limitations? Spreadsheets combine ideas from powerful programming languages. It is possible to compare the spreadsheet paradigm with the other programming paradigms.

We do not choose to take a theoretical perspective to argue that the pure spreadsheet language (as used in this paper) is equivalent to general purpose languages.

All programming paradigms are essentially equivalent, but who wants to use a Turing's machine? From this point of view, how spreadsheeting fundamentally differs from conventional programming is obvious at first glance: friendly user interface and instant modelling power are the special nature of spreadsheets.

Any problem that can be solved with a spreadsheet can also be solved with another computer program. "what–if" explorations are perhaps more efficient according to the conventional programming wisdom. What it lacks is speed in obtaining results.

Conversely, when is solved on a spreadsheet, how clear is it that a viable solution is obtainable? What it lacks is guaranty of effectiveness and verifiability [7].

Spreadsheets also have their share of problems which typically go unnoticed. The first significant deficiency is a perceived simplicity with which large tasks can be achieved. The second one is the "buried in the formula" logic of the creator, the documentation of the model is largely inaccessible.

A spreadsheet program is obviously software and thus should enforce a systematic development and then meticulous tests.

Nonetheless, it is to the systems scientists' advantage to rule out some particular simulation through simple conventional programming code (VBA or any else). A blend of pure

[7] See for example: Kellie B. Keeling , Numerical accuracy issues in using Excel for simulation studies, *Proceedings of the 36th conference on Winter simulation*, 2004, p. 1513-1518.

spreadsheets with macros language facilities is not excluded. Spreadsheets are also connected to databases and can import or export the data.

There remain a whole number of interesting research questions, some of which we have addressed. We would be delighted to hear from systemics researchers thinking about spreadsheeting.

References

[1] Géraldine Abrami, «Niveaux d'organisation dans la modélisation multi-agent pour la gestion de ressources renouvelables», PhD *thesis ENGREF*, Montpellier, France, 29 novembre 2004.

[2] Olivier Barreteau, «Modèles et processus de décision collective: entre compréhension et facilitation de la gestion concertée de la ressource en eau», *HDR Thesis Dauphine*, Paris, 2007.

[3] Nicholas F. Britton, *Essential Mathematical Biology*, Springer, 2004, 370 p.

[4] Martin Erwig, Robin Abraham, Irene Cooperstein, Steve Kollmansberger, «Automatic generation and maintenance of correct spreadsheets», *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO, 2005, p. 136-145.

[5] Alain Fernandez, «Représentation des actions en jeux sportifs collectifs», C*olloque international scientifique DREEPS IUFM & UFR-STAPS*, Nice, France, 16-19 décembre 1998. *http://www.unice.fr/ufrstaps/colloque_antibes/Fernandez/Ferna2.htm*

[6] Marc Fisher, Gregg Rothermel, Darren Brown, Mingming Cao, Curtis Cook, Margaret Burnett, «Integrating automated test generation into the WYSIWYT spreadsheet testing methodology», *ACM Trans. on Software Engineering and Methodology*, Issue 2(15), April 2006, p. 150-194.

[7] Dean S. Hartley, «The Oak Ridge Spreadsheet Battle Model», *Proceedings of the 22nd conference on Winter simulation*, New Orleans, LA, 1990, p. 863-869.

[8] T. J. Jankun-Kelly, Kwan-Liu Ma, «A spreadsheet interface for visualization exploration», *Proceedings of the conference on Visualization*, Salt Lake City, UT, October 2000, p. 69-76.

[9] Alan Kay, «Computer Software», *Scientific American*, Issue 251(3), September 1984, p. 41-47.

[10] Albert Napier, Richard Batsell, David Lane, Norman Guadagno, «Knowledge of command usage in a spreadsheet program», *ACM SIGMIS Database*, Issue 1(23), Winter 1992, p. 13-21.

[11] Jeff Parker, «Teaching complexity via spreadsheets», *Journal of Computing Sciences in Colleges*, Issue 5(23), May 2008, p. 83-89.

[12] Kamalasen Rajalingham, David Chadwick, Brian Knight, Dilwyn Edwards, «Quality Control in Spreadsheets: A Software Engineering-Based Approach to Spreadsheet Development», *Proc. of the 33th Hawaii Int. Conference on System Sciences*, Maui HI, January 2000, p. 4006-4015.

[13] Richard M. Reese, «Simulation programming using spreadsheet software», *Proceedings of the 16th conference on Winter simulation*, Dallas, Tx, 1984, p. 538-542.

[14] Yasuhiro Takeuchi, *Global Dynamical Properties of Lotka-Volterra Systems*, World Scientific Publishing Company, 1996, 302 p.