An attempt to automate the analysis of complex system dynamics models: an example of WORLD 3

Pedro Retortillo, Margarita Mediavilla, Luis Javier Miguel, Carlos de Castro
University of Valladolid
Departamento de Ingeniería de Sistemas y Automática
E.T.S.I.I. Paseo del Cauce s/n
47011 Valladolid, Spain
Tel/fax: 34 983 423545/34 983 423355
marga@eis.uva.es

Abstract

Even the simple run of a medium size system dynamics model can be a cumbersome process, since the uncertainty of the parameters forces the modeller to consider many runs before being confident of how the model behaves. System dynamics simulation packages include some analysis tools, but in many occasions customized tools are desired. For example, one would like to be able to program iterative running of simulations and perform mathematical operations with the results, use analysis techniques such as PPM or screening, or even use fuzzy logic to automate the revision of graphs. In this paper we explore the possibilities of a programming language, Matlab, and its simulation tool, Simulink, for those possibilities mentioned. These languages come from the fields of engineering, but offer many interesting possibilities because of their programming ability. They enable the development of customized analysis tools at a very low programming cost. The World 3 model has been programmed in this languages and some examples of application programming runs, screening and fuzzy logic are given.

Keywords
Screening, analysis, uncertainty, World 3, fuzzy logic.

1. Introduccion

System dynamics modelling and simulation is an exciting process that helps the modeller to get an important understanding of the problems involved, but, when the model is built and run, the user does not have so many instruments to tests what that model is really showing. A handful of simulation runs do not give much information when one faces large models with many stock variables, nonlinear dynamics and a high degree of uncertainty in the parameters, and the tools for analyzing large scale models are not very developed.

Some rely on intuitive approaches based on experience or, for example, on the notion of system archetypes (Güneralp 2006, Senge 1990), others focus on bounding the structure of the model with the observed behaviour, using methods like the eigenvalue elasticity analysis described by Forrester and refined lately in Kampmann and Oliva 2006. This approach uses linear systems theory to decompose the behaviour into simple modes, each of which corresponds to an eigenvalue. Measuring how much a given eigenvalue changes with a small change in a link in the model gives an indication of how much that link contributes to that behaviour mode. The main drawback of this method is the fact that eigenvalues are meaningful only for linear models or valid linearizations, which, in many occasions, are not possible. The pathway participation metric (PPM), developed by Mojtahedzadeh, (Mojtahedzadeh et al. 2004), identifies the structure that is most influential in affecting the qualitative time path of a given variable. The main strength of PPM is that it does not require calculating eigenvalues.

These two approaches focus on linking the temporal behaviour observed in the simulations of the model with its structure, (focusing on what part of the systems structure contributes most to some pattern of behaviour) but do not pay so much attention to the uncertainties and parameter variations of the models, which may, significantly, change system behaviour.

Ford (Ford and Flynn 2004) uses a pragmatic approach called screening that also focuses on detecting what part of the model structure contributes to the observed behaviour, but does not use eigenvalues or dominant loops, and pays more attention to the uncertainty of parameters. Part of the screening process is done in Vensim using the sensibility tool, but another part must be done with customized software.

Knowing what part of the model structure contributes to the observed behaviours is interesting because that helps, as Ford describes, “creating system stories” or correct explanations of how influential pieces of structure give rise to behaviour, and helps managers to understand the systems they manage. But, prior to the construction of such explanations, the modeller would need to know how his/her system does really behave.

This is not an easy task, since the uncertainty of the parameters is high and the possible variations of the values of the parameters multiplied by the number of parameters gives rise to an enormous number of possible behaviours that the modeller would have to test in order to really know the model.

This is the reason why we have tried to explore some tools to ease the process of testing system dynamics models. We decided to work with World 3 (Meadows 1992) to gain insight into this model and become familiar with it, but soon we realized that experimenting with a model of such complexity, using current system dynamics software packages, was not easy. We felt the need to use a customize software to be able to program massive runs of simulations, extract the interesting features of the results and operating with them as desired. A programming language (MATLAB in this case) showed as a perfect platform for this task (some similar languajes are public such as Scilab). We, therefore, translated World 3 model to Simulink-Matlab platform. The task was a hard one, but the model is now ready, and testing may now be done at a very low programming cost, as we show in this paper.

Purpose and organization

This paper describes the first trials to analysing this model and exploring the strength of using a programming platform. Our aim is to ease the analysis of the model by automating the analysis process, so that the modeller does not have to be running several simulations and studying the results, but, after a brief programming stage, the computer does the work and shows the desired comprised results.

The paper makes a brief description of the World 3 model translation into Simulink-MATLAB code in section 2, then explores the possibilities of using this software to ease the analysis process of common tools such as sensibility analysis or screening in section 3, and finally, describes the use of fuzzy logic to automate the analysis of some of the results in section 4.

2. The World 3 model in Simulink

System Dynamics diagrams are designed to explain feedback relationships as clearly as possible, and Stella, Vensim and Powesim are, no doubt, the easiest software packages for system dynamics programming. Other simulation packages, such as the ones used in control engineering, do the same simulations but the graphical languages are different and more obscure, although they have other advantages, as we shall see.

For those who would like to explore the possibilities of these tools, or try our World 3 model, it could be useful to look at figure 1 where we compare the graphical implementation of a stock with two flows in Vensim and in Simulink. The main difference is in the representation of stocks.

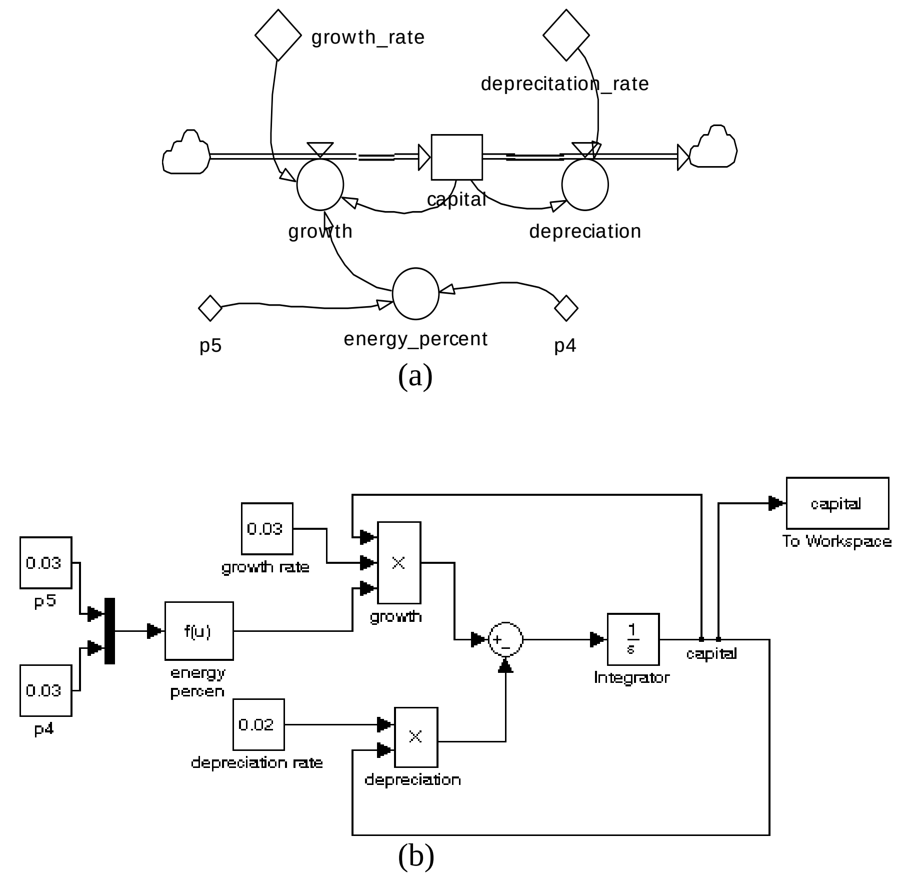


Figure 1: comparison of a Forrester diagram (a) and the equivalent block diagram (b). In the block diagram the variables “flow” between the blocks as they get operated. The block 1/s is the integrator, the output variable is the integral of the input.

Figure 2 shows the aspect of the interface of the World 3 model that we have built in Simulink. The complexity of the model is huge, and its aspect is less intuitive as in Stella or similar packages, but its modularity helps the integration and order. The best feature of this simulation is the interaction with the Matlab programming language, which enables the programming of experiments and analysis. The work presented in this paper explores some of the possibilities of analysis, it is only a first trial, further work should be devoted to a complete analysis of World 3, maybe introducing modifications such as the ones describe in Acharya and Saeed 1996.

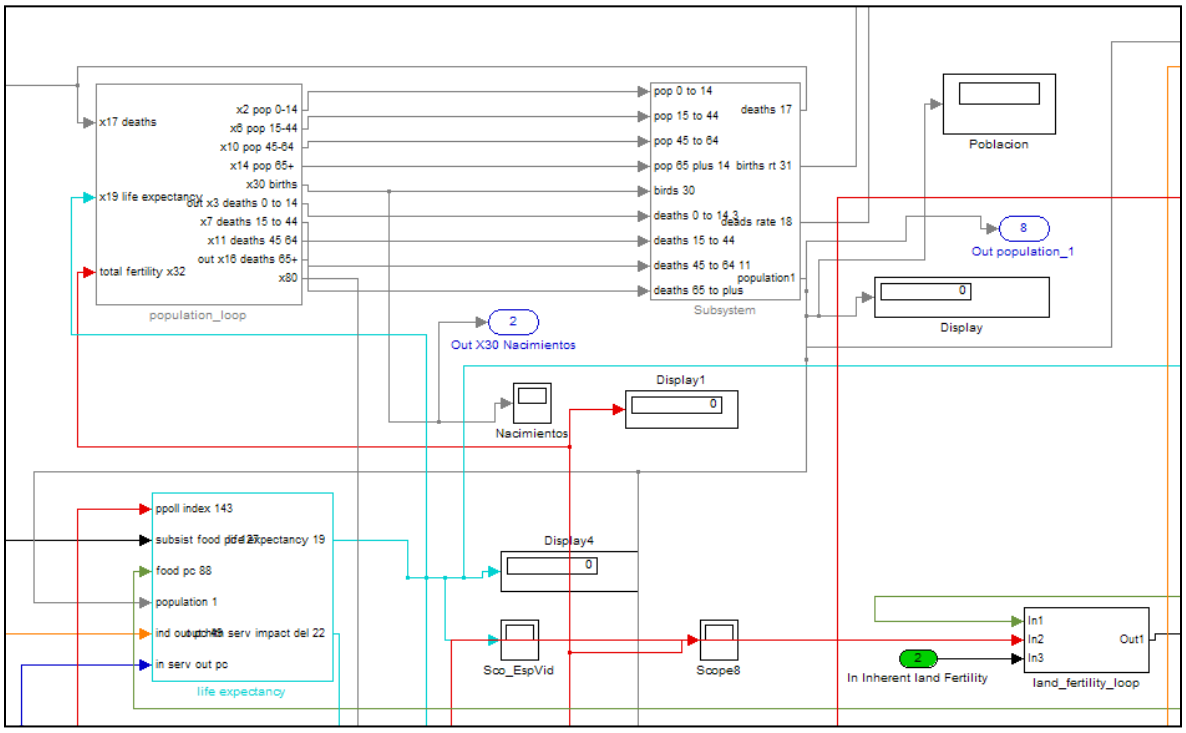


Figure 2: A partial view of the simulink interface with World 3 model.

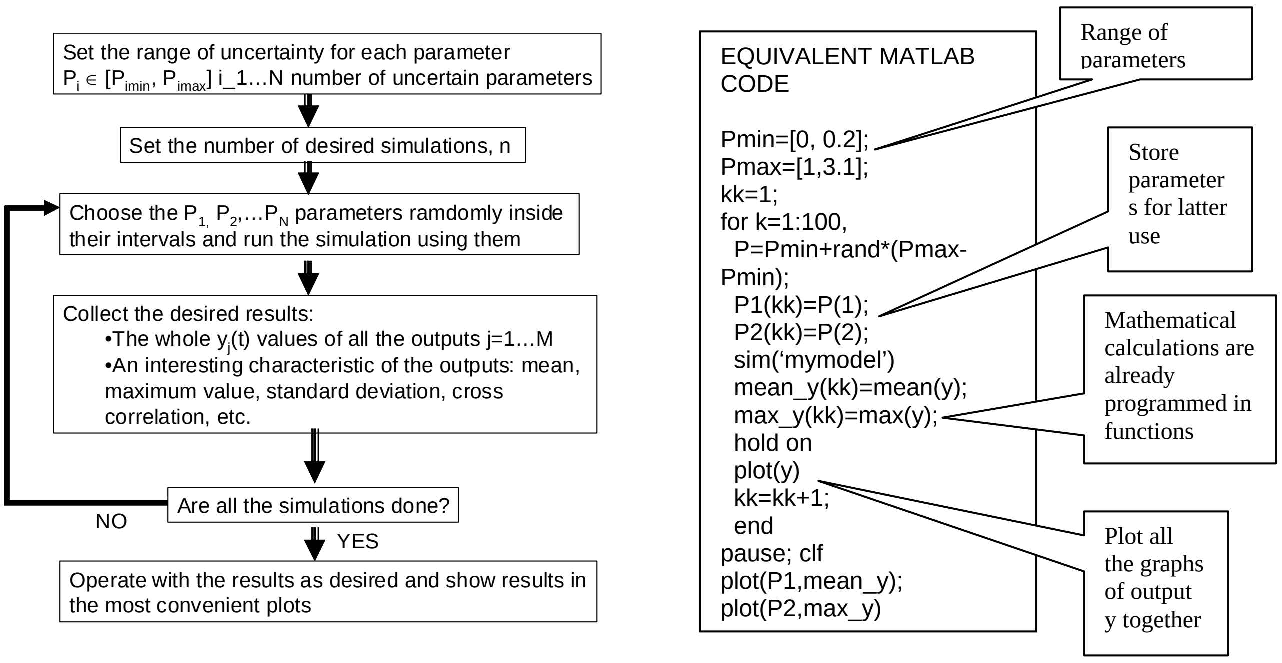
3. Exploring the possibilities of programming platforms for the analysis of models

When a modeller needs to run a model and see how it works, few runs are not enough to describe it. The uncertainty of most of the parameters of system dynamics models is high, or even very high (Ford and Flint 2004) and all of them might change at the same time. On the other hand, the nonlinearity of these models can make small changes in the parameters lead to big changes in the outputs. It is, thus, very desirable to program several runs of any model with randomly selected parameters and plot all the results. Some system dynamics software's enable this multiple runs, but one might like to be able to have more control on those runs. For example, one could decide to store all the output values to extract relevant statistical information of those outputs, or could be interested on plotting the results in a different way to observe the behaviour better. The fact that Matlab is a programming platform and the user programs the runs enables all kind of manipulations with the running of the simulation and with the data.

Some possibilities are shown in figure 3: run several simulations with randomly elected parameters, calculate several interesting characteristics out of those outputs and, for example, plot the results of the output variable of interest all together in one graph, or plot relevant features of the output versus some parameters. In figure 3 we can see the results of these kinds of experiments. We can see that the programming effort is very small using these programming platforms. We have simulated World 3 under Matlab randomly changing the following parameters in their intervals:

Parameter	**interval**
non renewable resources initial	$(1{\cdot}10^{12}, \ 2{\cdot}10^{12})$
inherent land fertility	(600, 1200)
index of absorption of pollution	(0.5 , 1)
average life on industrial capital	(12, 16)
year of stabilization of industrial capital	(2000, 2100)

The plot of the final value of population and the maximum population minus the final value can be seen in figure 4.


Set the range of uncertainty for each parameter
$P_i \in [P_{imin}, P_{imax}]$ i_1…N number of uncertain parameters
Set the number of desired simulations, n
Choose the $P_1, P_2, \ldots P_N$ parameters ramdomly inside their intervals and run the simulation using them
Collect the desired results:
•The whole $y_j(t)$ values of all the outputs j=1…M
•An interesting characteristic of the outputs: mean, maximum value, standard deviation, cross correlation, etc.
Are all the simulations done?
NO
YES
Operate with the results as desired and show results in the most convenient plots
EQUIVALENT MATLAB CODE
Range of parameters
Store parameter s for latter use
Mathematical calculations are already programmed in functions
Plot all the graphs of output y together

```
Pmin=[0, 0.2];
Pmax=[1,3.1];
kk=1;
for k=1:100,
  P=Pmin+rand*(Pmax-
Pmin);
  P1(kk)=P(1);
  P2(kk)=P(2);
  sim('mymodel')
  mean_y(kk)=mean(y);
  max_y(kk)=max(y);
  hold on
  plot(y)
  kk=kk+1;
  end
pause; clf
plot(P1,mean_y);
plot(P2,max_y)
```

Figure 3: Programming of several simulation runs with randomly selected parameters. All the variables involved are stored in vectors and matrices for later calculations and plots.

A sensibility analysis can be performed in a similar way, calculating the sensibility not only around one specific parameter set, but around several possible parameters sets, as shown in figure 5.

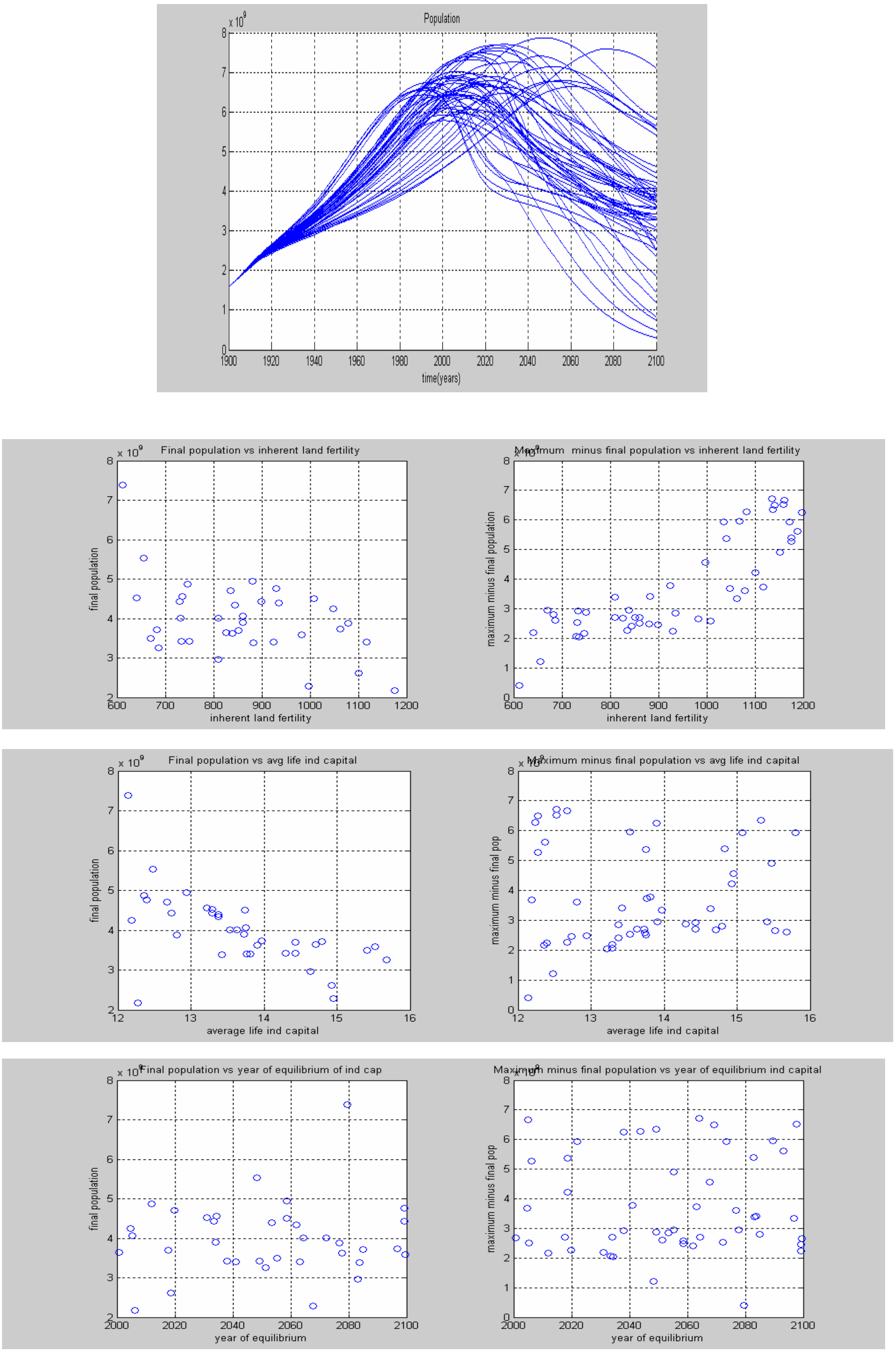


Figure 4: Results of 50 simulation of World 3 model with randomly selected parameters. (a) population, (b) final value of the population and difference between maximum and final value plotted against several parameters. Some correlations can be observed in inherent fertility and average life of industrial capital variables

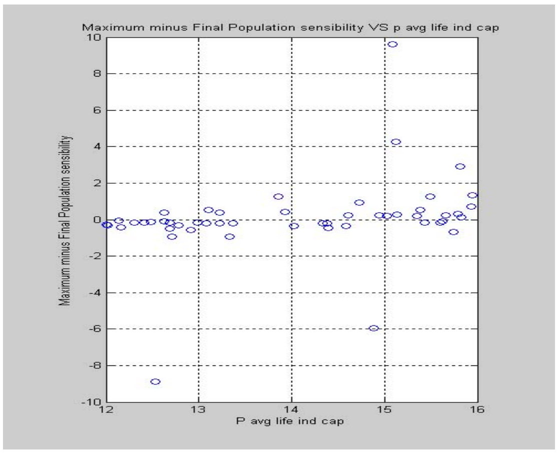
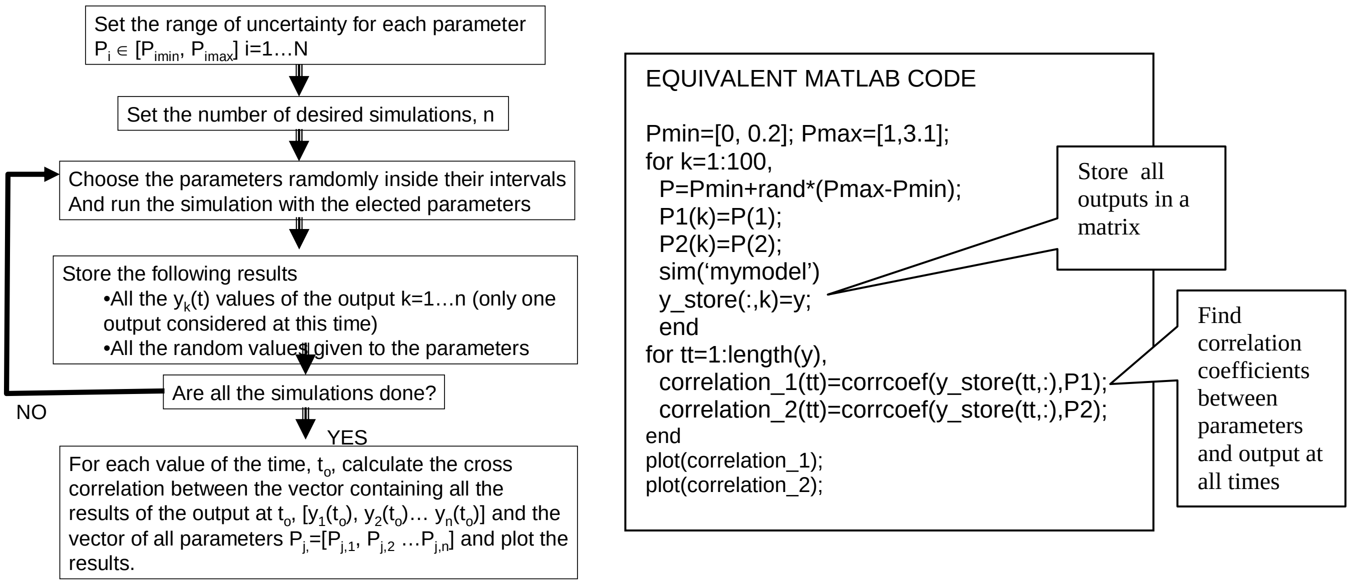


Figure 5: Sensitivity of the final population to average life of industrial capital with random parameters.

The screening technique described by Ford and Flynn 2004, can also be implemented at minimum programming cost as can be seen in see figures 6 and 7, since most of the mathematical calculations are already programmed as toolboxes of Matlab:


Set the range of uncertainty for each parameter $P_i \in [P_{imin}, P_{imax}]$ i=1…N
Set the number of desired simulations, n
Choose the parameters ramdomly inside their intervals And run the simulation with the elected parameters
Store the following results •All the $y_k(t)$ values of the output k=1…n (only one output considered at this time) •All the random values given to the parameters
Are all the simulations done?
NO
YES
For each value of the time, t_o, calculate the cross correlation between the vector containing all the results of the output at t_o, $[y_1(t_o), y_2(t_o)… y_n(t_o)]$ and the vector of all parameters $P_j = [P_{j,1}, P_{j,2} …P_{j,n}]$ and plot the results.
Store all outputs in a matrix
Find correlation coefficients between parameters and output at all times

EQUIVALENT MATLAB CODE

```
Pmin=[0, 0.2]; Pmax=[1,3.1];
for k=1:100,
  P=Pmin+rand*(Pmax-Pmin);
  P1(k)=P(1);
  P2(k)=P(2);
  sim(‘mymodel’)
  y_store(:,k)=y;
  end
for tt=1:length(y),
  correlation_1(tt)=corrcoef(y_store(tt,:),P1);
  correlation_2(tt)=corrcoef(y_store(tt,:),P2);
end
plot(correlation_1);
plot(correlation_2);
```

Figure 6: Screening process. Several simulation runs with randomly selected parameters. The correlation coefficients between all the values of the output obtained (at each time instant) and the randomly selected parameters are calculated.

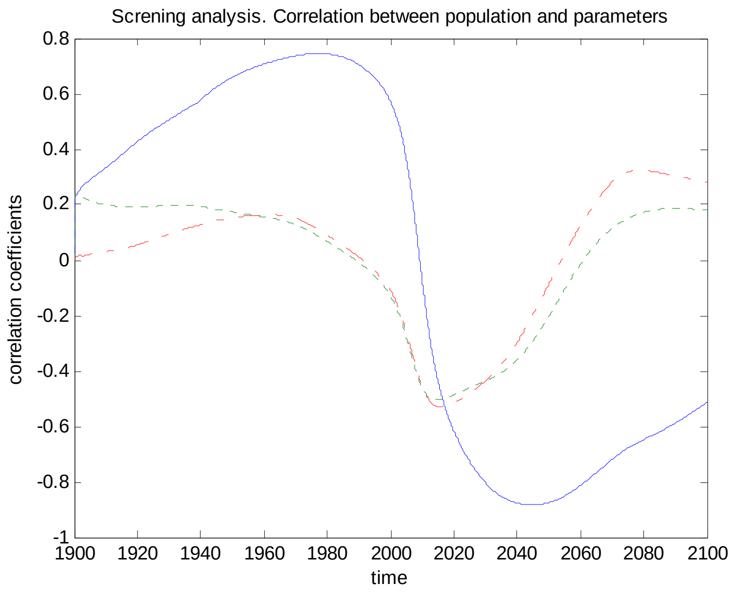


Figure 7: Results of the screening applied to the population and several parameters. Correlation coefficients between the population at each time';' parameters: average life of industrial capital (-), assimilation of pollution (...) and initial non renewable resources (.-.-)'

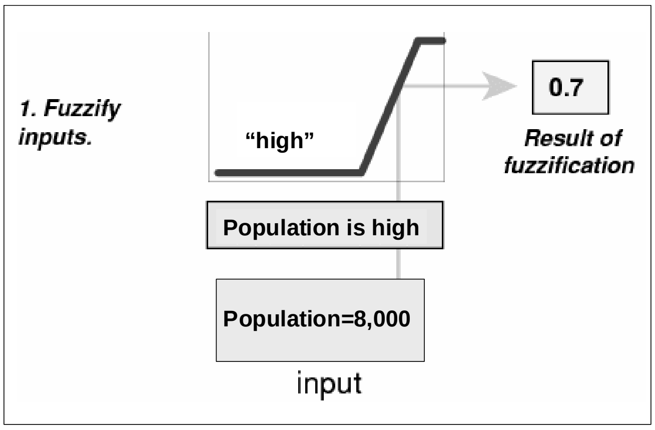
4. Application of fuzzy logic

Fuzzy logic is another interesting tool for analysis. Normally, system dynamics models are evaluated by looking at the graphs of the output variables. It is the human modeller the one who normally decides weather or not a curve shows a “good”, “poor” or “excellent” response. If the number of runs is high it would be desirable to count on an automatic procedure to classify the shape of a curve and automatically extract the desired information.

Fuzzy logic can help us on that, since it is a form of logic designed to imitate human reasoning and its vagueness. Fuzzy logic is based on the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. For example, in a simulation of the world population we could say that if human population is above 9,000 million inhabitants is clearly high and below 5,000 million is clearly low, but what about populations between 5,000 and 9,000? It would be “high” but not very “high”, or “high” only to a certain degree. Fuzzy logic describes variables in terms of fuzzy sets with a membership degree; in this case we would say that the population is 0.8 “high” and 0.2 “low”.

Fuzzy logic is divided in three stages: fuzzyfication, application of fuzzy rules and defuzzyfication.

Fuzzyfication is the process of assigning an input variable (numerical) to a fuzzy set with a membership function (see figure 8).



FUZZIFICATION PROCESS

Figure 8: fuzzy sets and fuzzyfication process. If the world population is 8,000 million, for example, it would be 0.2 "low" and 0.8 "high".

The application of fuzzy rules is based on an imitation of the reasoning logic of human languages. For example, we could apply the logic describe in the following sentence: "if the final value of the population is low and population curve has a very negative slope then the result of the run is a catastrophe" (see figure 9)

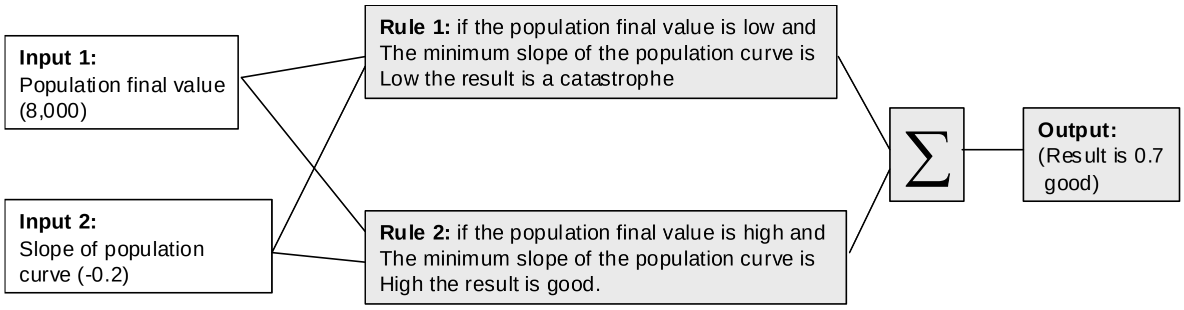

Input 1: Population final value (8,000)
Input 2: Slope of population curve (-0.2)
Rule 1: if the population final value is low and The minimum slope of the population curve is Low the result is a catastrophe
Rule 2: if the population final value is high and The minimum slope of the population curve is High the result is good.
∑
Output: (Result is 0.7 good)

Figure 9: fuzzy rules

The logical sentences are translated into mathematical operations of the fuzzy sets. We can electr any of these two options for implementation of AND and OR operators.

AND operator	Minimum of the membership funcions
	product of the membership functions
OR operator	Maximum of memebership funcions
	Probor operator (probor(a,b)=a+b-a*b)

The defuzzyfication process assigns a numerical output value to the output fuzzy sets and membership functions. In simple rules as the ones we are using in this paper the result of the logical operation is enough.

The entire process is resumed in figure 10.

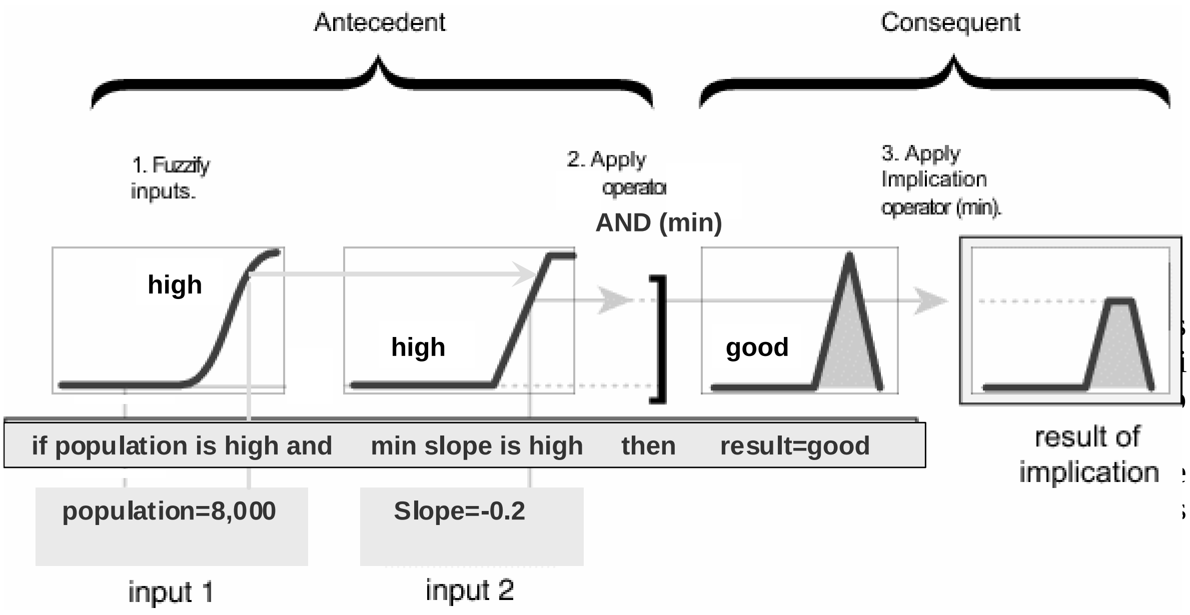


Figure 10: exmaple of aplication of fuzzy rules

We have used fuzzy logic to evaluate the response of the output values of some simulations of World 3. The population curve is the output that we select as most significant. We would say that the curve is a bad result (catastrophe) if the final value of the population is low, being this feature the most significant. But, the result is also bad if the peak of the population curve occurs very soon or if the final slope of the population is steep, which means that the population tends to decrease even more.
Therefore, the following values of the population curve are treated: the final value, the year when the maximum is reached and the final slope of the curve.

And the fuzzy logic is the one of the sentence:

IF final population == low OR
0.5*year of peak of population == low
OR 0.7*final slope of population curve == low
→ result is a collapse

In figure 11, 12 and 13 we can see several population curves and the results of the application of this sentence. We can see that those results classified with a 1 result in the “collapse” set correspond to curves with a very bad result: very low final population and steep declines. Those results with a value lower than zero have different degrees of “badness”, which pretty much reflect how close they are to the characteristics expressed in the equation. Figure 14 describes the code needed to implement this program, which is very short.

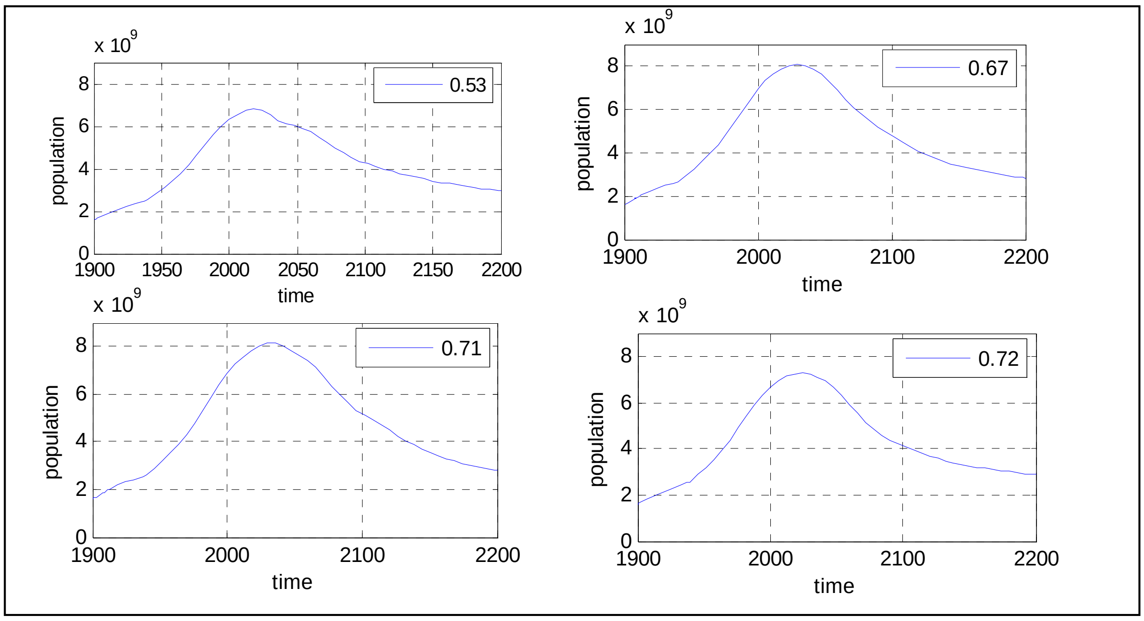


Figure 11: Results of the use of fuzzy logic to the analysis of the population curve. The curve whose result of 0.53 is clearly the best result, the final population if not too low and it is more stable than in other results. The other results are similar, there are slight differences, for example, the curve with the result 0.71 has a steeper slope of the population but the curve with 0.72 has got a earlier peak.

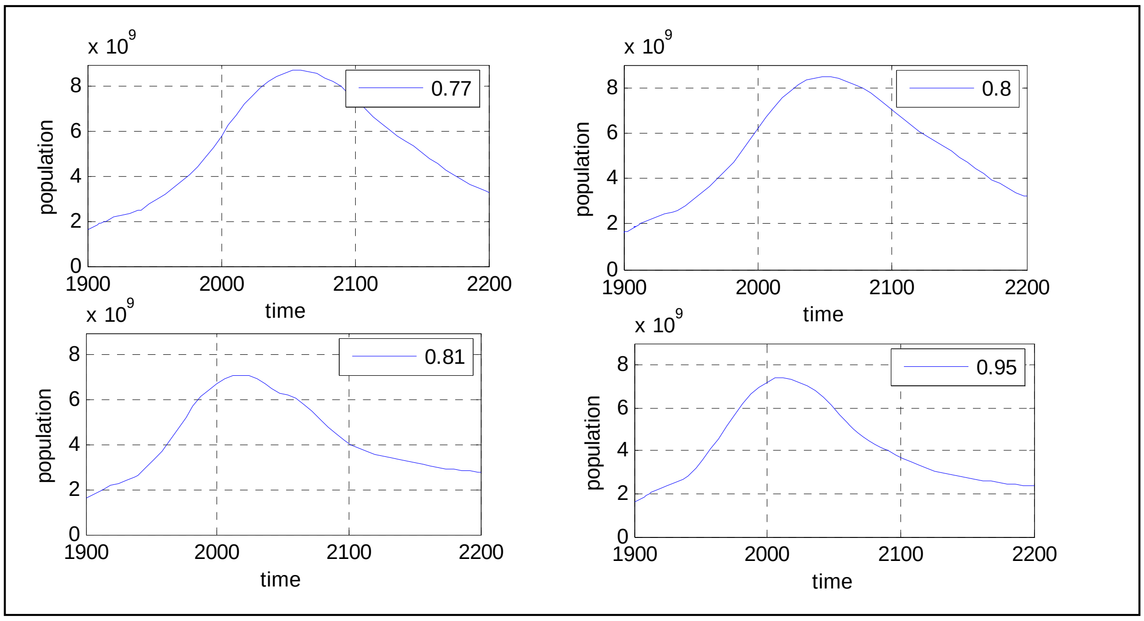


Figure 12: Results of the use of fuzzy logic to the analysis of the population curve. These curves are clearly worse than the ones of figure 11, and the result of the fuzzy logic shows it clearly with a higher value in the “catastrophe” set.

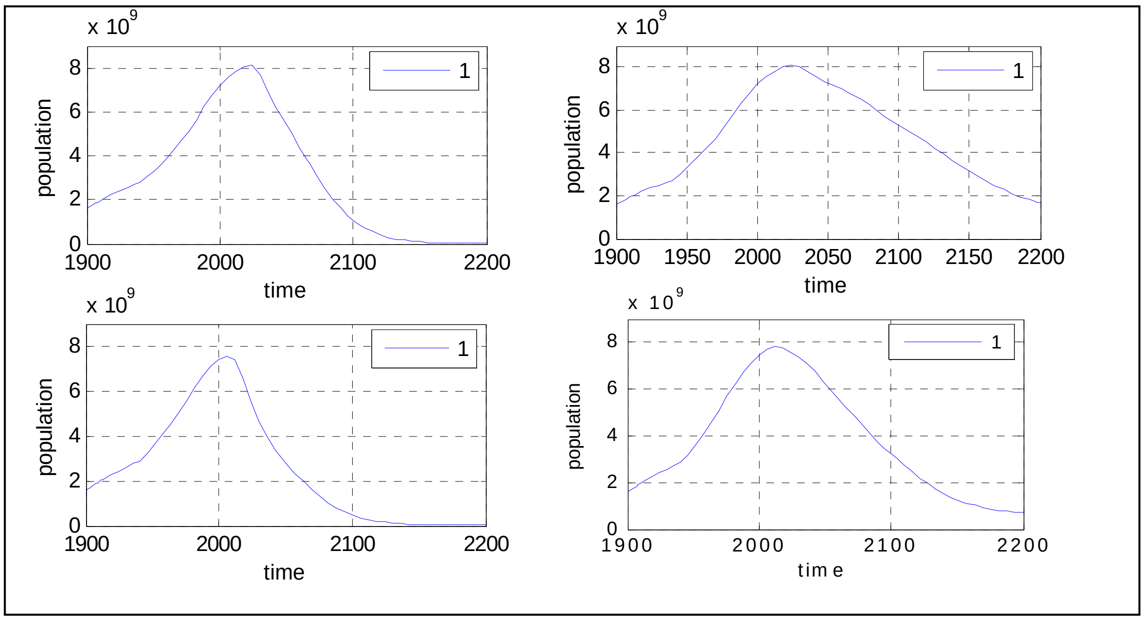
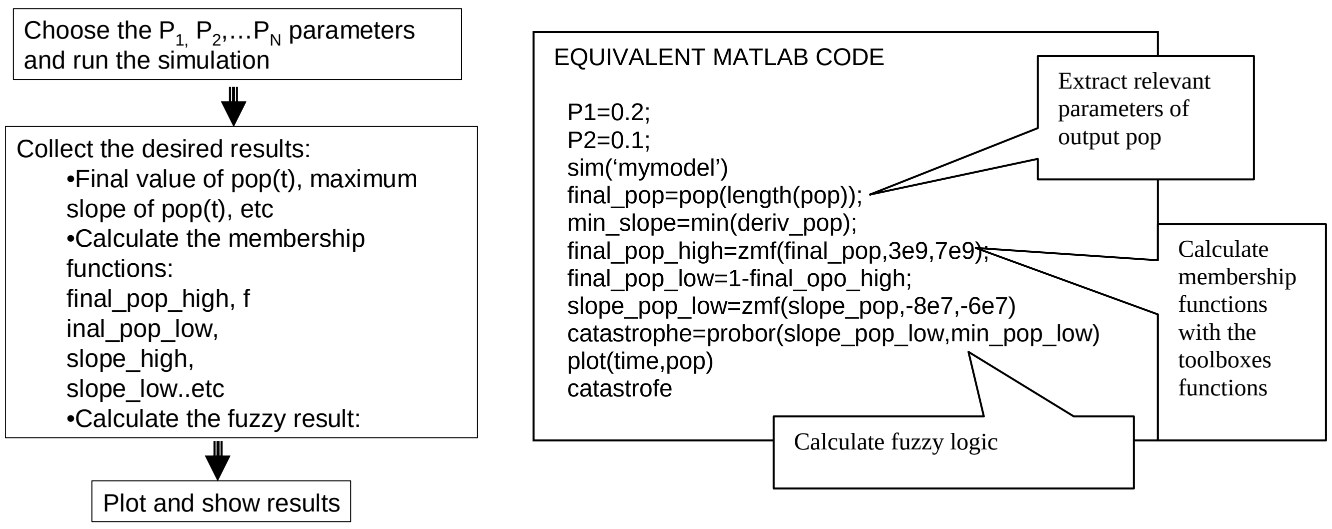


Figure 13: Results of the use of fuzzy logic to the analysis of the population curve. The curves that obtain the value 1 are clearly the worse results, with very low final population and steep declines.


Choose the P1, P2,…PN parameters and run the simulation
Collect the desired results:
•Final value of pop(t), maximum slope of pop(t), etc
•Calculate the membership functions:
final_pop_high, f
inal_pop_low,
slope_high,
slope_low..etc
•Calculate the fuzzy result:
Plot and show results
EQUIVALENT MATLAB CODE
Extract relevant parameters of output pop
Calculate membership functions with the toolboxes functions
Calculate fuzzy logic

```
P1=0.2;
P2=0.1;
sim('mymodel')
final_pop=pop(length(pop));
min_slope=min(deriv_pop);
final_pop_high=zmf(final_pop,3e9,7e9);
final_pop_low=1-final_opo_high;
slope_pop_low=zmf(slope_pop,-8e7,-6e7)
catastrophe=probor(slope_pop_low,min_pop_low)
plot(time,pop)
catastrofe
```

Figure 14: Fuzzy logic process. One simulation run, extraction of the desired characteristics of the output and calculation of the fuzzy logic.

5. Conclusions

This paper has explored the possibilities that a programming language (Matlab-simulink in this case) gives us for programming and analysis of system dynamic models. Several analysis have been applied to the World 3 model programmed in Simulink: several runs with random parameters, collection of results of several simulation and calculation of interesting statistical indicators, screening, and also application of fuzzy logic for automatic characterization of output functions.

All this enables a systematic analysis and programming of simulations and offers the possibility to analyse complex models as World 3 in a systematic way. The complete Simulink code of World 3 is available for those who would like to explore of modify this model in Matlab environment.

The results of this paper show very promising possibilities. The use of fuzzy logic could ease the classification of results and enable a better global understanding of complex models.

References

Acharya, S. R. and Saeed, K. 1996. An attempt to operationalize the recommendations of the 'Limits to growth' study to sustain the future of mankind. *System Dynamics Review* Volume 12 Number 4,pp:281-304.

Ford, A. and Flynn, H. 2004. Statistical screening of system dynamics models. *System Dynamics Review,* 21(4): 273 – 303

Güneralp, B. 2006. Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review* 22,(3): 263–289

Kampmann, C. E. and Oliva, R. 2006. Loop eigenvalue elasticity analysis: three case studies. *System Dynamics Review,* 22(2):

Meadows, D. H., Meadows, D. L., Randers, J., and Berhens 111, W. W. 1972. The Limits to Growth. New York: Universe Books. [Republished by Productivity Press, Portland, OR].

Meadows, D. H., Meadows, D. L., Randers, J. 1992. Beyond the Limits. Post Mills, Vermont: Chelsea Green Publishing Company

Mojtahedzadeh M., Andersen D. and Richardson GP. 2004. Using Digest to implement the pathway participation method for detecting influential system structure. *System Dynamics Review* 20(1): 1–20

Scilab http://www.scilab.org

Wolstenholme, E. Using generic system archetypes to support thinking and modelling *System Dynamics Review* 20(4): 341-356