Combining Mathematical and Simulation Approaches to Understand the Dynamics of Computer Models

Luis R. Izquierdo, Segismundo S. Izquierdo, José M. Galán & José I. Santos

PREPRINT

Authors' information

Luis R. Izquierdo
Departamento de Ingeniería Civil
Universidad de Burgos, E-09001, Burgos, Spain.
luis@izquierdo.name

Segismundo S. Izquierdo
Departamento de Organización de Empresas y C.I.M.
Universidad de Valladolid, E-47011, Valladolid, Spain.
segis@eis.uva.es

José M. Galán
Departamento de Ingeniería Civil
Universidad de Burgos, E-09001, Burgos, Spain.
jmgalan@ubu.es

José I. Santos
Departamento de Ingeniería Civil
Universidad de Burgos, E-09001, Burgos, Spain.
jisantos@ubu.es

Why would you want to read this chapter?

This chapter is about how to better understand the dynamics of computer models using both simulation and mathematical analysis. Our starting point is a computer model which is already implemented and ready to be run; our objective is to gain a thorough understanding of its dynamics. This chapter shows how computer simulation and mathematical analysis can be used together to provide a picture of the dynamics of the model that could not be drawn by using one of the two techniques only.

Abstract

This chapter shows how computer simulation and mathematical analysis can be used together to understand the dynamics of computer models. For this purpose, we show that it is useful to see the computer model as a particular implementation of a formal model in a certain programming language. This formal model is the abstract entity which is defined by the input-output relation that the computer model executes, and can be seen as a function that transforms probability distributions over the set of possible inputs into probability distributions over the set of possible outputs.

It is shown here that both computer simulation and mathematical analysis are extremely useful tools to analyse this formal model, and they are certainly complementary in the sense that they can provide fundamentally different insights on the same model. Even more importantly, this chapter shows that there are plenty of synergies to be exploited by using the two techniques together.

The mathematical analysis approach to analyse formal models consists in examining the rules that define the model directly. Its aim is to deduce the logical implications of these rules for any particular instance to which they can be applied. Our analysis of mathematical techniques to study formal models is focused on the theory of Markov Chains, which is particularly useful to characterise the dynamics of computer models.

In contrast with mathematical analysis, the computer simulation approach does not look at the rules that define the formal model directly, but instead tries to infer general properties of these rules by examining the outputs they produce when applied to particular instances of the input space. Thus, conclusions obtained with this approach may not be general. On a more positive note, computer simulation enables us to explore formal models beyond mathematical tractability, and we can achieve any arbitrary level of accuracy in our computational approximations by running the model sufficiently many times.

Bearing in mind the relative strengths and limitations of both approaches, this chapter explains three different ways in which mathematical analysis and computer simulation can be usefully combined to produce a better understanding of the dynamics of computer models. In doing so, it becomes clear that mathematical analysis and computer simulation should not be regarded as alternative –or even opposed– approaches to the formal study of social systems, but as complementary. Not only can they provide fundamentally different insights on the same model, but they can also produce hints for solutions for each other. In short, there are plenty of synergies to be exploited by using the two techniques together, so the full potential of each technique cannot be reached unless they are used in conjunction.

1 Introduction

This chapter is about how to better understand the dynamics of computer models using both simulation and mathematical analysis. Our starting point is a computer model which is already implemented and ready to be run; our objective is to gain a thorough understanding of its dynamics. Thus, this chapter is *not* about how to design, implement, verify, or validate a model; this chapter is about how to better understand its behaviour.

Naturally, we start by clearly defining our object of study: a computer model. The term 'computer model' can be understood in many different ways –i.e. seen from many different perspectives–, and not all of them are equally useful for every possible purpose. Thus, we start by interpreting the term 'computer model' in a way that will prove useful for our objective: to characterise and understand its behaviour. Once our object of study has been clearly defined, we then describe two techniques that are particularly useful to understand the dynamics of computer models: mathematical analysis and computer simulation.

In particular, this chapter will show that mathematical analysis and computer simulation should not be regarded as alternative –or even opposed– approaches to the formal study of social systems, but as complementary (Gotts et al. 2003a, 2003b). They are both extremely useful tools to analyse formal models, and they are certainly complementary in the sense that they can provide fundamentally different insights on the same model. Even more importantly, this chapter will show that there are plenty of synergies to be exploited by using the two techniques together, i.e. the full potential of each technique will not be reached until they are used in conjunction. The remaining of this introduction outlines the structure of the chapter.

Sections 2, 3 and 4 are devoted to explaining in detail what we understand by 'computer model', and they therefore provide the basic framework for the rest of the chapter. In particular, section 2 shows that a computer model can be seen as an implementation –i.e. an explicit representation– of a certain deterministic input-output function in a particular programming language. This interpretation is very useful since, in particular, it will allow us to abstract from the details of the modelling platform where the computer model has been programmed, and focus on analysing the formal model that the computer model implements. This is clarified in section 3, which explains that any computer model can be re-implemented in many different formalisms (in particular, in any sophisticated enough programming language), leading to alternative representations of the same input-output relation.

Most computer models in the Social Simulation literature make use of pseudo-random number generators. Section 4 explains that –for these cases and given our purposes– it is useful to abstract from the details of how pseudorandom numbers are generated, and look at the computer model as an implementation of a stochastic process. In a stochastic process, a certain input does not necessarily lead to one certain output only; instead, there are many different paths that the process may take with potentially different probabilities. Thus, in a stochastic process a certain input will generally lead to a particular probability distribution over the range of possible outputs, rather than to a single output only. Stochastic processes are used to formally describe how a system subjected to random events evolves through time.

Having explained our interpretation of the term 'computer model', section 5 introduces and compares the two techniques to analyse formal models that are assessed in this chapter: computer simulation and mathematical analysis. The following two sections sketch possible ways in which each of these two techniques can be used to obtain useful insights about the dynamics of a model. Section 8 is then focused on the *joint use* of computer simulation and mathematical analysis. It is shown here that the two techniques can be used together to provide a picture of the dynamics of the model that could not be drawn by using one of the two techniques only. Finally, our conclusions are summarised in section 9.

2 Computer models as input-output functions

At the most elementary level, a computer model can be seen as an implementation –i.e. an explicit representation– of a certain deterministic input-output function in a particular programming language. The word 'function' is useful because it correctly conveys the point that any particular input given to the computer model will lead to one and only one output[1]. (Obviously, different inputs may lead to the same output.) Admittedly, however, the word 'function' may also mislead the reader into thinking that a computer model is necessarily simple. The computer model may be as complex and sophisticated as the programmer wants it to be but, ultimately, it is just an entity that associates a specific output to any given input, i.e. a function. In any case, to avoid confusion, we will use the term 'formal model' to denote the function that a certain computer model implements[2]. To be sure, the 'formal model' that a particular computer model implements is the abstract entity which is defined by the input-output relation that the computer model executes[3].

Thus, running a computer model is just finding out the logical implications of applying a set of unambiguously defined formal rules (which are coded in the program and define the input-output function or formal model) to a set of inputs (Balzer et al. 2001). As an example, one could write the computer program "$y = 4 \cdot x$" and apply it to the input "$x = 2$" to obtain the output "$y = 8$". The output ($y = 8$), which is fully and unequivocally determined by the input ($x = 2$) and the set of rules coded in the program ($y = 4 \cdot x$), can be seen as a theorem obtained by pure deduction ($\{x = 2; y = 4 \cdot x\} \Longrightarrow y = 8$). Naturally, there is no reason why the inputs or the outputs should be numbers[4]; they could equally well be e.g. strings of characters. In the general case, a computer run is a logical theorem that reads: *the output obtained from running the computer simulation follows (with logical necessity) from applying to the input the algorithmic rules that define the model*. Thus, regardless of its inherent complexity, a computer run constitutes a perfectly valid sufficiency theorem (see e.g. Axtell 2000).

---

[1] Note that simulations of stochastic models are actually using pseudorandom number generators, which are deterministic algorithms that require a seed as an input.

[2] A formal model is a model expressed in a formal system (Cutland 1980). A formal system consists of a formal language and a deductive apparatus (a set of axioms and inference rules). Formal systems are used to derive new expressions by applying the inference rules to the axioms and/or previously derived expressions in the same system.

[3] The mere fact that the model has been implemented and can be run in a computer is a proof that the model is formal (Suber 2007).

[4] As a matter of fact, strictly speaking, inputs and outputs in a computer model are *never* numbers. We may interpret strings of bits as numbers, but we could equally well interpret the same strings of bits as e.g. letters. More importantly, a bit itself is already an abstraction, an interpretation we make of an electrical pulse that can be above or below a critical voltage threshold.

It is useful to realise that we could always apply the same inference rules ourselves to obtain –by logical deduction– the same output from the given input. While useful as a thought, when it comes to actually doing the job, it is much more convenient, efficient and less prone to errors to let computers derive the output for us. Computers are inference engines that are able to conduct many algorithmic processes at a speed that the human brain cannot achieve.

3 Different ways of representing the same formal model

A somewhat controversial issue in the Social Simulation literature refers to the allegedly unique features of some modelling platforms. It is important to realise that any formal model implemented in a computer model can be re-implemented in many different programming languages, leading to exactly the same input-output relation. Different implementations are just different ways of representing one same formal model, much in the same way that one can say 'Spain' or 'España' to express the same concept in different languages: same thing, different representation, that's all.

Thus, when analysing the dynamics of a computer model, it is useful to abstract from the details of the modelling platform that has been used to implement the computer model, and focus strictly on the formal model it represents, which could be re-implemented in any *sophisticated enough*[5] modelling platform. To be clear, let us emphasise that *any* computer model implemented in Objective-C (e.g. using Swarm) can be re-implemented in Java (e.g. using RePast or Mason), NetLogo, SDML, Mathematica© or Matlab©. Similarly, *any* computer model can be expressed as a well-defined mathematical function (Epstein 2006; Leombruni and Richiardi 2005; Richiardi et al. 2006).

Naturally, the implementation of a particular formal model may be more straightforward in some programming languages than in others. Programming languages differ in where they position themselves in the well-known trade-offs between ease of programming, functionality and performance; thus, different programming languages lead to more or less natural and more or less efficient implementations of any given formal model. Nonetheless, the important point is this: whilst we may have different implementations of the same formal model, and whilst each of these implementations may have different characteristics (in terms of e.g. code readability), ultimately they are all just different representations of the same formal model, and they will therefore return the same output when given the same input.

In the same way that using one or another formalism to represent a particular formal model will lead to more or less natural implementations, different formalisms also make more or less apparent certain properties of the formal model they implement. For

[5] A sufficient condition for a programming language to be "sophisticated enough" is to allow for the implementation of the following three control structures:

- Sequence (i.e. executing one subprogram, and then another subprogram),
- Selection (i.e. executing one of two subprograms according to the value of a boolean variable, e.g. IF[boolean == true]-THEN[subprogram1]-ELSE[subprogram2]), and
- Iteration (i.e. executing a subprogram until a boolean variable becomes false, e.g. WHILE[boolean == true]-DO[subprogram]).

Any programming language that can combine subprograms in these three ways can implement any computable function; this statement is known as the "structured program theorem"(Böhm and Jacopini 1966; Harel 1980; Wikipedia 2007).

example, we will see in this chapter that representing a computer model as a Markov chain, i.e. looking at the formal model implemented in a computer model through Markov's glasses, can make apparent various features of the computer model that may not be so evident without such glasses. In particular, as we will show later, Markov theory can be used to find out whether the initial conditions of a model determine its asymptotic dynamics or whether they are actually irrelevant in the long term. Also, the theory can reveal whether the model will sooner or later be trapped in an absorbing state.

4 'Stochastic' computer models as stochastic processes

Most computer models in the Social Simulation literature contain stochastic components. This section argues that, for these cases and given our purposes, it is convenient to revise our interpretation of computer models as *deterministic* input-output relations, abstract from the (deterministic) details of how pseudorandom numbers are generated, and reinterpret the term 'computer model' as an implementation of a stochastic process. This interpretation will prove useful in most cases and, importantly, does not imply any loss of generality: even if the computer model to be analysed does not contain any stochastic components, our interpretation will still be valid.

In the general case, the computer model to be analysed will make use of (what are meant to be) random numbers, i.e. the model will be stochastic. The word 'stochastic' requires some clarification. Strictly speaking, there does not exist a *truly stochastic* computer model, but one can approximate randomness to a very satisfactory extent by using pseudorandom number generators. The pseudorandom number generator is a deterministic algorithm that takes as input a value called the random seed, and generates a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by the value used to initialise the algorithm, i.e. the random seed. Therefore, if given the same random seed, the pseudorandom number generator will produce exactly the same sequence of (pseudorandom) numbers. (This fact is what made us define a computer model as an implementation of a certain *deterministic* input-output function in section 2.)

Fortunately, the sequences of numbers provided by current off-the-shelf pseudorandom number generators approximate randomness remarkably well. This basically means that, for most intents and purposes in this discipline, it seems safe to assume that the pseudorandom numbers generated in one simulation run will follow the intended probability distributions to a satisfactory degree. The only problem we might encounter appears when running several simulations which we would like to be statistically independent. As mentioned above, if we used the same random seed for every run, we would obtain the same sequence of pseudorandom numbers, i.e. we would obtain exactly the same results. How can we *truly randomly* select a random seed? Fortunately, for most applications in this discipline, the state of the computer system at the time of starting a new run can be considered a truly random variable; and, conveniently, if no seed is explicitly provided to the pseudorandom number generator, most platforms generate a seed from the state of the computer system (e.g. using the time). When this is done, the sequences of numbers obtained with readily available pseudorandom number generators approximate statistical randomness and independence remarkably well.

Given that –for most intents and purposes in this discipline– we can safely assume that pseudorandom numbers are random and independent enough, we dispense with the

qualifier ‘pseudo’ from now on for convenience. Since every random variable in the model follows a specific probability distribution, the computer model will indeed generate a particular probability distribution over the range of possible outputs. Thus, to summarise, a computer model can be usefully seen as the implementation of a stochastic process, i.e. a function that transforms any given input into a certain probability distribution over the set of possible outputs (Figure 1).

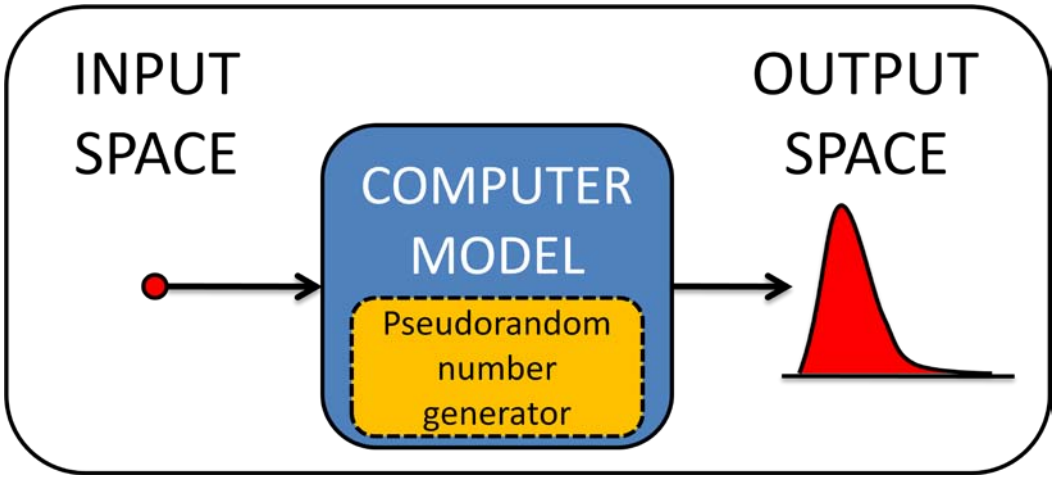

Figure 1. A computer model can be usefully seen as the implementation of a function that transforms any given input into a certain probability distribution over the set of possible outputs.

Having seen that we can satisfactorily simulate random variables, note that studying the behaviour of a model that has been parameterised stochastically does not introduce any conceptual difficulties. In other words, we can study the behaviour of a model that has been parameterised with probability distributions rather than certain values. An example would be a model where agents start at a random initial location.

To conclude this section, let us emphasise an important corollary of the previous paragraphs: *any statistic that we extract from a parameterised computer model follows a specific probability distribution* (even if the values of the input parameters have been expressed as probability distributions)[6]. Thus, a computer model can be seen as the implementation of a function that transforms probability distributions over the set of possible inputs into probability distributions over the set of possible outputs (Figure 2). The rest of the chapter is devoted to characterising this function.

[6] Note that statistics extracted from the model can be of any nature, as long as they are unambiguously defined. For example, they can refer to various time-steps, and only to certain agents (e.g. “average wealth of female agents in odd time-steps from 1 to 99”).

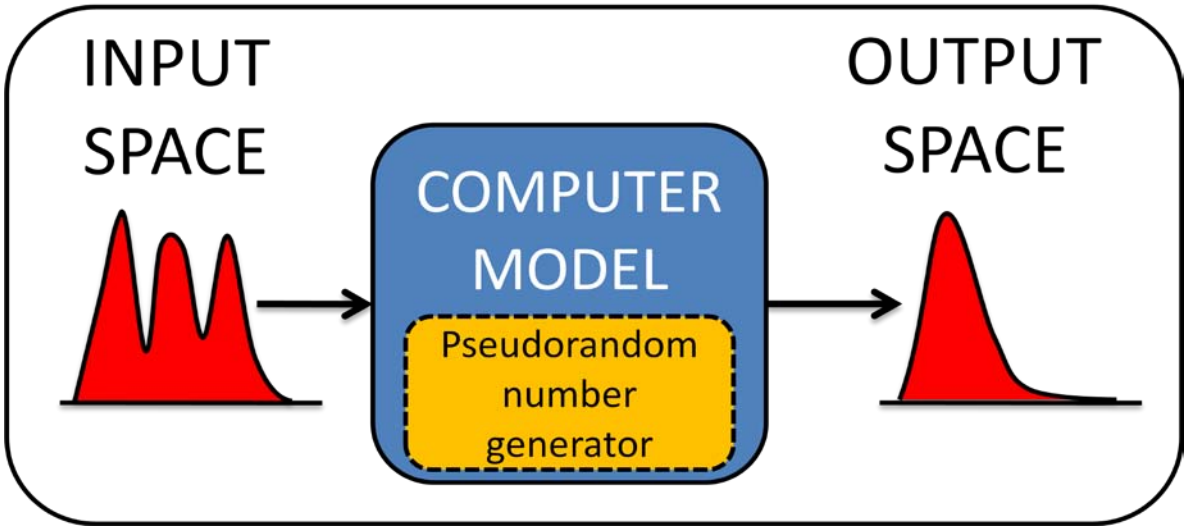

Figure 2. A computer model can be seen as the implementation of a function that transforms probability distributions over the set of possible inputs into probability distributions over the set of possible outputs.

5 Tools to understand the behaviour of formal models

Once it is settled that a computer model can be seen as a particular implementation of a (potentially stochastic) function in a certain programming language, let us refer to such a function as the 'formal model' that the computer model implements. As mentioned before, this formal model can be expressed in many different formalisms –in particular, it can always be expressed as a set of well defined mathematical equations (Leombruni and Richiardi 2005)– and our objective consists in understanding its behaviour. To do that, we count with two very useful tools: mathematical analysis[7] and computer simulation.

The advantages and limitations of these two tools to formally study social systems have been discussed at length in the literature (see e.g. Axtell 2000; Axtell and Epstein 1994; Edmonds 2005; Gilbert 1999; Gilbert and Troitzsch 1999; Gotts et al. 2003a; Holland and Miller 1991; Ostrom 1988). Here we only highlight the most prominent differences between these two techniques (see Figure 3).

[7] We use the term "mathematical analysis" in its broadest sense, i.e. we do not refer to any particular branch of mathematics, but to the general use of (any type of) mathematical technique to analyse a system.

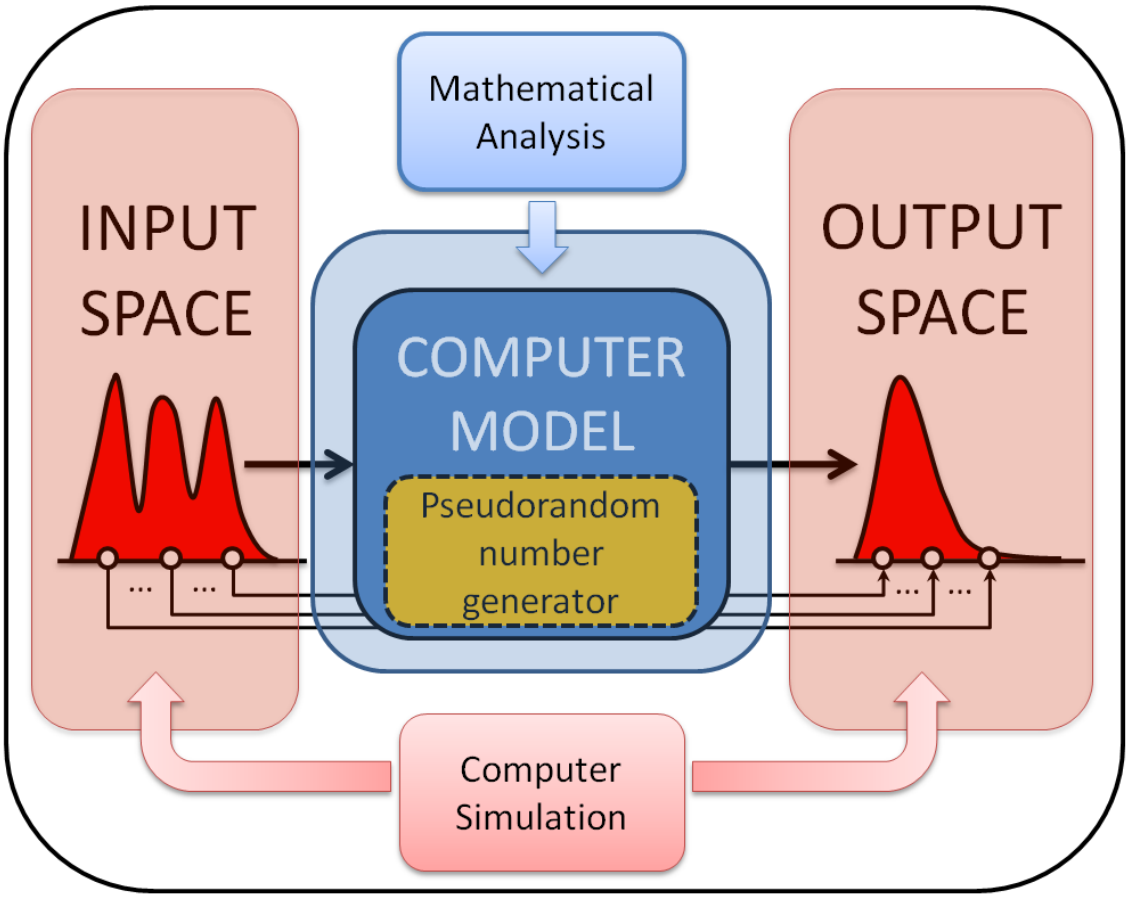

Figure 3. In general terms, mathematical analysis tends to examine the rules that define the formal model directly. In contrast, computer simulation tries to infer general properties of such rules by looking at the outputs they produce when applied to particular instances of the input space.

In broad terms, when using mathematical analysis, one examines the rules that define the formal model directly, and tries to draw general conclusions about these rules. These conclusions are obtained by using logical deduction; hence they follow with logical necessity from the premises of the formal model (and the axioms of the mathematics employed). The aim when using mathematical analysis is usually to "solve" the formal system (or, most often, certain aspects of it) by producing general closed-form solutions that can be applied to *any* instance of the whole input set (or, at least, to large portions of the input set). Since the inferences obtained with mathematical analysis pertain to the rules themselves, such inferences can be safely particularised to any specific parameterisation of the model, even if such a parameterisation was never explicitly contemplated when analysing the model mathematically. This greatly facilitates conducting sensitivity analyses and assessing the robustness of the model.

Computer simulation is a rather different approach to the characterisation of the formal model (Epstein 2006; Axelrod 1997a). When using computer simulation, one often treats the formal model as a black box, i.e. a somewhat obscure abstract entity that returns certain outputs when provided with inputs. Thus, the path to understand the behaviour of the model consists in obtaining many input-output pairs and –using generalisation by induction– inferring general patterns about how the rules transform the inputs into the outputs (i.e. how the formal model works).

Importantly, the execution of a simulation run, i.e. the logical process that transforms any (potentially stochastic) given input into its corresponding (potentially stochastic) output is pure deduction (i.e. strict application of the formal rules that define the model). Thus, running the model in a computer provides a formal proof that a particular input (together with the set of rules that define the model) is sufficient to generate the output that is observed during the simulation. This first part of the computer simulation approach is therefore, in a way, very "mathematical": outputs obtained follow with logical necessity from applying to the inputs the algorithmic rules that define the model.

In contrast, the second part of the computer simulation approach, i.e. inferring general patterns from particular instances of input-output pairs, can only lead to probable – rather than necessarily true– conclusions[8]. The following section explains how to rigorously assess the confidence we can place on the conclusions obtained using computer simulation, but the simple truth is irrefutable: inferences obtained using generalisation by induction can potentially fail when applied to instances that were not used to infer the general pattern. This is the domain of statistical extrapolation.

So why bother with computer simulation at all? The answer is clear: computer simulation enables us to study formal systems in ways that go beyond mathematical tractability. This role should not be underestimated: most models in the Social Simulation literature are mathematically intractable, and in such cases computer simulation is our only chance to move things forward. As a matter of fact, the formal models that many computer programs implement are often so complicated and cumbersome that the computer code itself is not that far from being one of the best descriptions of the formal model that can be provided.

Computer simulation can be very useful even when dealing with formal models that are mathematically tractable. Valuable uses of computer simulation in these cases include conducting insightful initial explorations of the model and presenting dynamic illustrations of its results.

And there is yet another important use of computer simulation. Note that understanding a formal model in depth requires identifying the parts of the model (i.e. the subset of rules) that are responsible for generating particular (sub)sets of results or properties of results. Investigating this in detail often involves changing certain subsets of rules in the model, so one can pinpoint which subsets of rules are necessary or sufficient to produce certain results. Importantly, changing subsets of rules can make the original model mathematically intractable and in such (common) cases, computer simulation is, again, our only hope. In this context, computer simulation can be very useful to produce counter-examples. This approach is very common in the literature of e.g. evolutionary game theory, where several authors (see e.g. Hauert and Doebeli 2004; Imhof et al. 2005; Izquierdo and Izquierdo 2006; Lieberman et al. 2009; Nowak and May 1992; Nowak and Sigmund 1992; Nowak and Sigmund 1993; Santos et al. 2006; Traulsen et al. 2006) resort to computer simulations to assess the implications of assumptions made in mathematically tractable models (e.g. the assumptions of "infinite populations" and "random encounters").

[8] Unless, of course, all possible particular instances are explored.

It is important to note that the fundamental distinction between mathematical analysis and computer simulation as presented here is *not* about whether one uses pen and paper or computers to analyse formal models. We can follow either approach with or without computers, and it is increasingly popular to do mathematical analysis with computers. Recent advancements in symbolic computation have opened up a new world of possibilities to conduct mathematical analyses (using e.g. Mathematica©). In other words, nowadays it is perfectly possible to use computers to *directly* examine the rules that define a formal model (see Figure 3).

Finally, as so often in life, things are not black or white, but involve some shade of grey. Similarly, most models are not tractable *or* intractable in mathematical terms; most often they are *partially* tractable. It is in these cases where an adequate combination of mathematical analysis and computer simulation is particularly useful. We illustrate this fact in section 8, but first let us look at each technique separately. The following two sections provide some guidelines on how computer simulation (section 6) and mathematical analysis (section 7) can be usefully employed to analyse formal models.

6 Computer simulation: Approximating the exact probability distribution by running the model

The previous sections have argued that *any* statistic obtained from a (stochastically or deterministically) parameterised model follows a specific probability distribution. The statistic could be anything as long as it is unambiguously defined; in particular, it could refer to one or several time-steps, and to one or various subcomponents of the model. Ideally, one would like to calculate the exact probability distribution for the statistic using mathematical analysis, but this will not always be possible. In contrast, using computer simulation we will always be able to approximate this probability distribution to any arbitrary level of accuracy; this section provides basic guidelines on how to do that.

The output probability distribution –which is fully and unequivocally determined by the input distribution– can be approximated to any degree of accuracy by running enough simulation runs. Note that any specific simulation run will be conducted with a particular certain value for every parameter (e.g. a particular initial location for every agent), and will produce one and only one particular certain output (see Figure 3). Thus, in order to infer the probability distribution over the set of outputs that a particular probability distribution over the set of inputs leads to, there will be a need to run the model many times (with different random seeds); this is the so-called Monte Carlo method.

The method is straightforward: obtain as many random samples as possible (i.e. run as many independent simulations as possible), since this will get us closer and closer to the exact distribution (by the law of large numbers). Having conducted a large number of simulation runs, the question that naturally comes to mind is: How close to the exact distribution is the one obtained by simulation?

To illustrate how to assess the quality of the approximation obtained by simulation, we use CoolWorld, a purpose-built agent-based model (Gilbert 2007) implemented in NetLogo 4.0 (Wilensky 1999). A full description of the model, an applet and the source code can be found at http://luis.izquierdo.name/models/coolworld. For our purposes, it suffices to say that in CoolWorld there is a population of agents called walkers, who

wander around a 2-dimensional grid made of square patches; some of the patches are empty whilst others contain a house (see Figure 4). Patches are at a certain predefined temperature, and walkers tend to walk towards warmer patches, staying for a while at the houses they encounter in their journey.

Let us assume that we are interested in studying the number of CoolWorld walkers staying in a house in time-step 50. Initial conditions (which involve 100 walkers placed at a random location) are unambiguously defined at http://luis.izquierdo.name/models/coolworld and can be set in the implementation of CoolWorld provided by clicking on the button "Special conditions". Figure 4 shows a snapshot of CoolWorld after having clicked on that button.

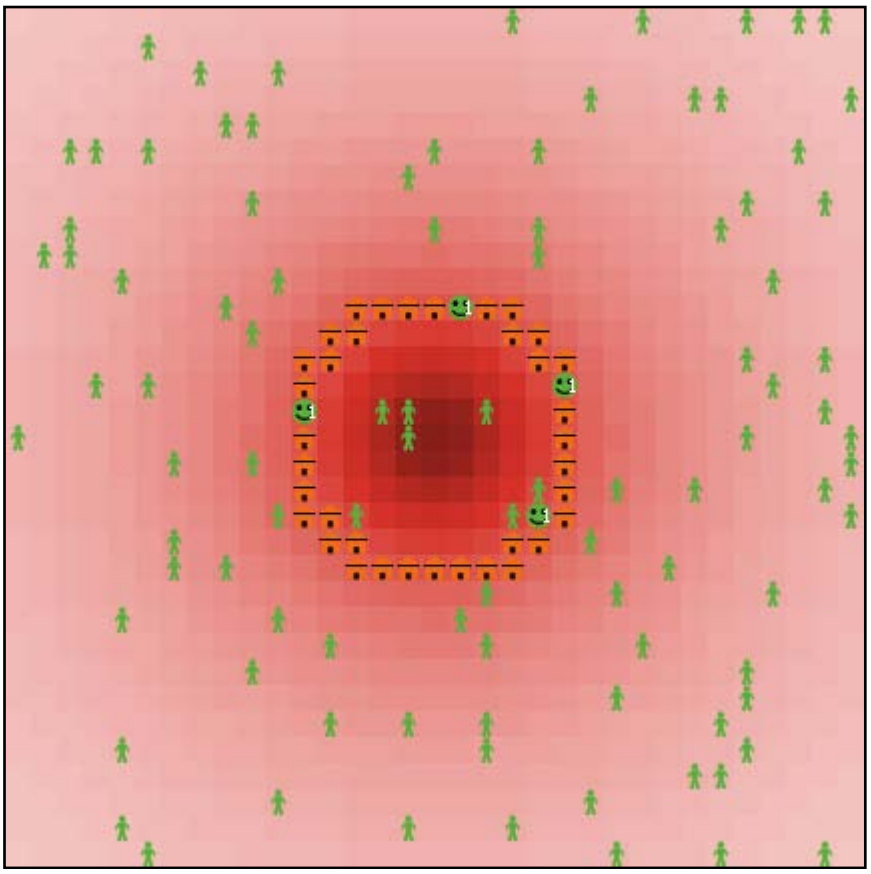

Figure 4. Snapshot of CoolWorld. Patches are coloured according to their temperature: the higher the temperature, the darker the shade of red. Houses are coloured in orange, and form a circle around the central patch. Walkers are coloured in green, and represented as a person if standing on a patch without a house, and as a smiling face if standing on a patch with a house. In the latter case, the white label indicates the number of walkers in the same house.

As argued before, given that the (stochastic) initial conditions are unambiguously defined, the number of CoolWorld walkers in a house after 50 time-steps will follow a specific probability distribution that we are aiming to approximate. For that, let us assume that we run 200 runs, and plot the relative frequency of the number of walkers in a patch with a house after 50 time-steps (see Figure 5).

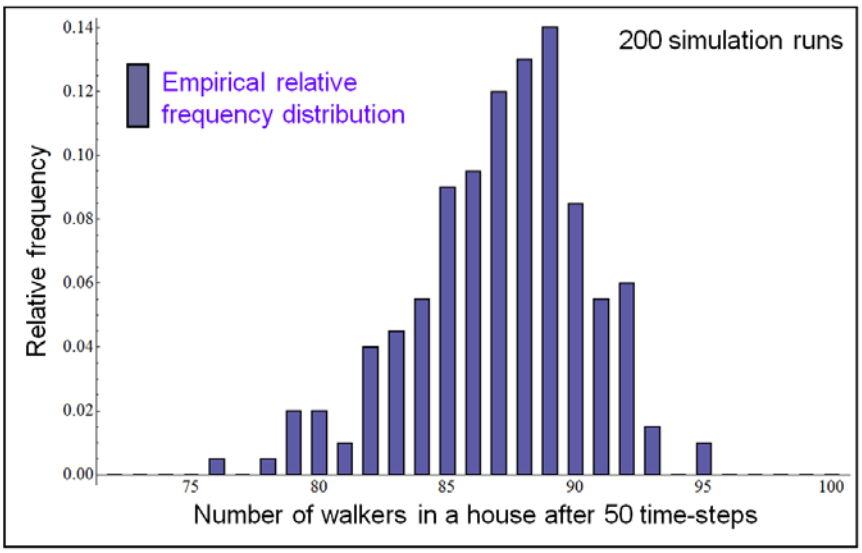

Figure 5. Relative frequency distribution of the number of walkers in a house after 50 time-steps, obtained by running CoolWorld 200 times, with initial conditions set by clicking on "Special conditions".

Figure 5 does not provide all the information that can be extracted from the data gathered. In particular, we can plot error bars showing the standard error for each calculated frequency without hardly any effort[9]. Standard errors give us information about the error we may be incurring when estimating the exact probabilities with the empirical frequencies. Another simple task that can be conducted consists in partitioning the set of runs into two batteries of approximately equal size and comparing the two distributions. If the two distributions are not similar, then there is no point in proceeding: we are not close to the exact distribution, so there is a need to run more simulations.

Figure 6 and Figure 7 show the data displayed in Figure 5 partitioned in two batteries of 100 simulation runs, including the standard errors. Figure 6 and Figure 7 also show the exact probability distribution we are trying to approximate, which has been calculated using mathematical methods that are explained later in this chapter.

[9] The frequency of the event "there are *i* walkers in a patch with a house" calculated over *n* simulation runs can be seen as the mean of a sample of *n* i.i.d. Bernouilli random variables where success denotes that the event occurred and failure denotes that it did not. Thus, the frequency *f* is the maximum likelihood (unbiased) estimator of the exact probability with which the event occurs. The standard error of the calculated frequency *f* is the standard deviation of the sample divided by the square root of the sample size. In this particular case, the formula reads:

$$\text{Std. error}(f, n) = (f\,(1 - f) / (n - 1))^{1/2}$$

Where *f* is the frequency of the event, *n* is the number of samples, and the standard deviation of the sample has been calculated dividing by $(n - 1)$.

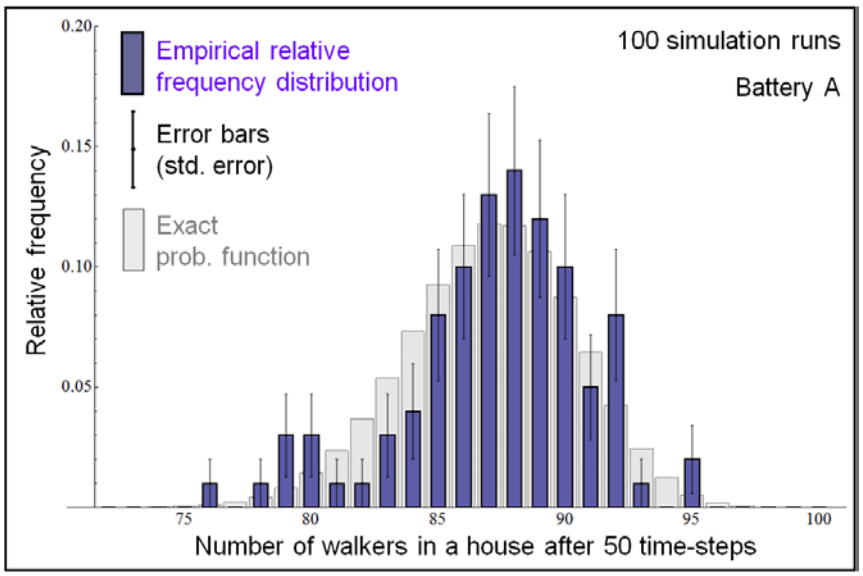

Figure 6. In blue: Relative frequency distribution of the number of walkers in a house after 50 time-steps, obtained by running CoolWorld 100 times (Battery A), with initial conditions set by clicking on "Special conditions". In grey: Exact probability distribution (calculated using Markov chain analysis).

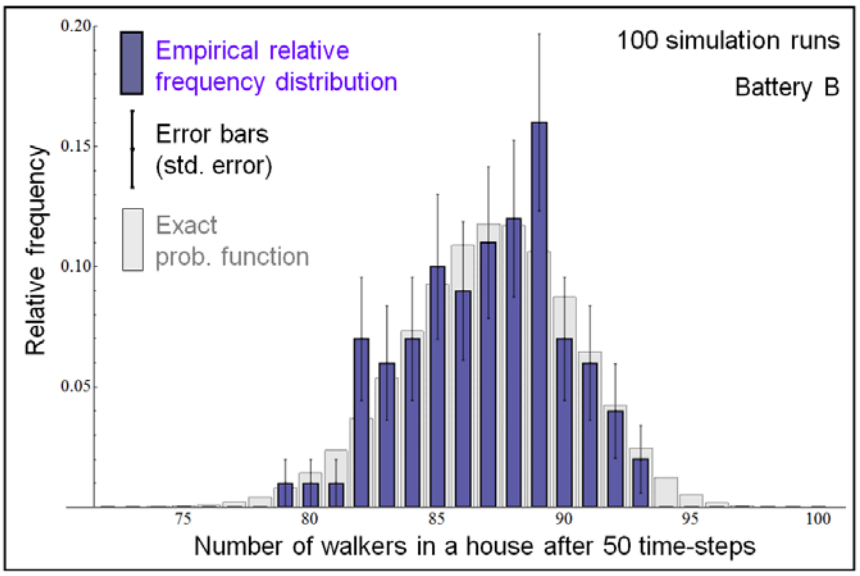

Figure 7. In blue: Relative frequency distribution of the number of walkers in a house after 50 time-steps, obtained by running CoolWorld 100 times (Battery B), with initial conditions set by clicking on "Special conditions". In grey: Exact probability distribution (calculated using Markov chain analysis).

Figure 6 and Figure 7 indicate that 100 simulation runs may not be enough to obtain a satisfactory approximation to the exact probability distribution. On the other hand, Figure 8 and Figure 9 show that running the model 50 000 times does seem to get us close to the exact probability distribution. The standard error, which is inversely proportional to the square root of the sample size (i.e. the number of runs), is naturally much lower in these latter cases.

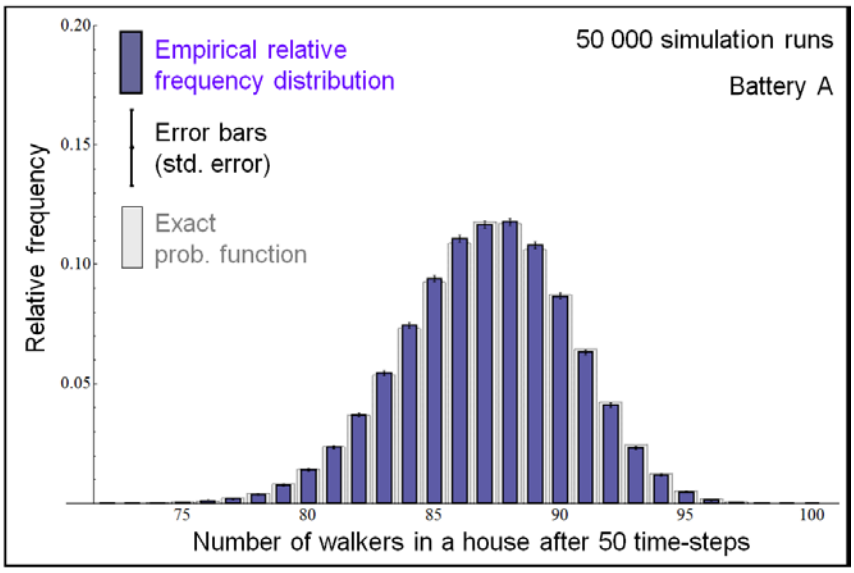

Figure 8. In blue: Relative frequency distribution of the number of walkers in a house after 50 time-steps, obtained by running CoolWorld 50 000 times (Battery A), with initial conditions set by clicking on "Special conditions". In grey: Exact probability distribution (calculated using Markov chain analysis).

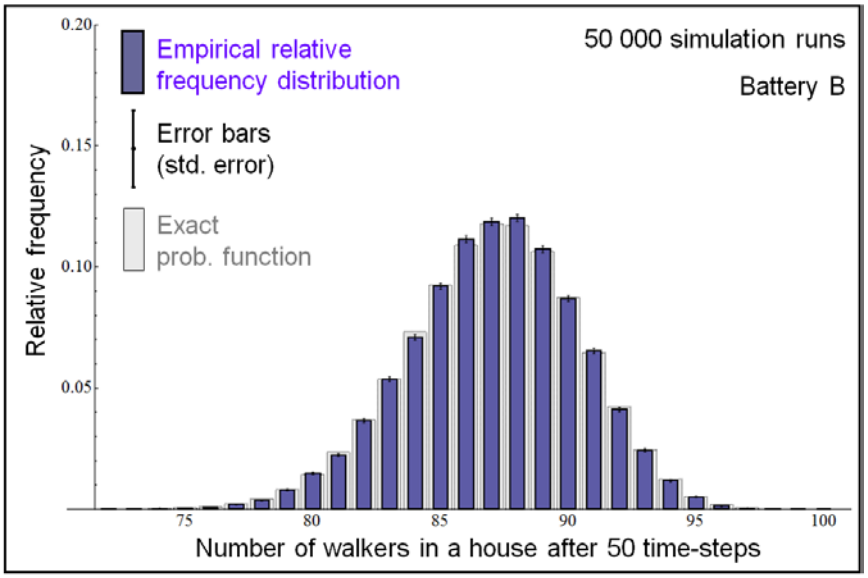

Figure 9. In blue: Relative frequency distribution of the number of walkers in a house after 50 time-steps, obtained by running CoolWorld 50 000 times (Battery B), with initial conditions set by clicking on "Special conditions". In grey: Exact probability distribution (calculated using Markov chain analysis).

When, like in this example, the space of all possible outcomes in the distribution under analysis is finite (the number of walkers in a house must be an integer between 0 and 100), one can go further and calculate confidence intervals for the obtained frequencies. This is easily conducted when one realises that the exact probability distribution is a multinomial. Genz and Kwong (2000) show how to calculate these confidence intervals.

To conclude this section, let us emphasise that all that has been written here applies to *any* statistic obtained from *any* computer model. In particular, the statistic may refer to predefined regimes (e.g. "number of time-steps between 0 and 100 where there are more than 20 walkers in a house") or to various time-steps (e.g. "total number of walkers in a house in odd time-steps in between time-steps 50 and 200"). These statistics, like any other one, follow a specific probability distribution that can be approximated to any degree of accuracy by running the computer model.

7 Mathematical analysis: Time-homogenous Markov chains.

The whole range of mathematical techniques that can be used to analyse formal systems is too broad to be reviewed here. Instead, we focus on one specific technique that seems to us particularly useful to analyse Social Simulation models: Markov chain analysis. Besides, there are multiple synergies to be exploited by using Markov chain analysis and computer simulation together, as we will see in the next section.

Our first objective is to learn how to represent a particular computer model as a time-homogeneous Markov chain. This alternative representation of the model will allow us to use several simple mathematical results that will prove useful to understand the dynamics of the model. We therefore start by describing time-homogeneous Markov chains.

7.1 What is a time-homogeneous Markov chain?

Consider a system that in time-step n={1,2,3,...} may be in one of a finite number of possible states $S = \{s_1, s_2,\ldots, s_M\}$. The set S is called the state space; in this chapter we only consider *finite* state spaces[10]. Let the sequence of random variables $X_n \in S$ represent the state of the system in time-step n. As an example, $X_3 = s_9$ means that at time $n = 3$ the system is in state s_9. The system starts at a certain initial state X_0 and moves from one state to another. The system is stochastic in that, given the present state, the system may move to one or another state with a certain probability (see Figure 10). The probability that the system moves from state i to state j in one time-step, $P(X_{n+1} = j \mid X_n = i)$, is denoted by $p_{i,j}$. As an example, in the Markov chain represented in Figure 10, $p_{4,6}$ equals 0 since the system cannot go from state 4 to state 6 in one single time-step. The system may also stay in the same state i, and this occurs with probability $p_{i,i}$. The probabilities $p_{i,j}$ are called transition probabilities and they are often arranged in a matrix, namely the transition matrix P.

[10] The term 'Markov chain' allows for countably infinite state spaces too (Karr 1990).

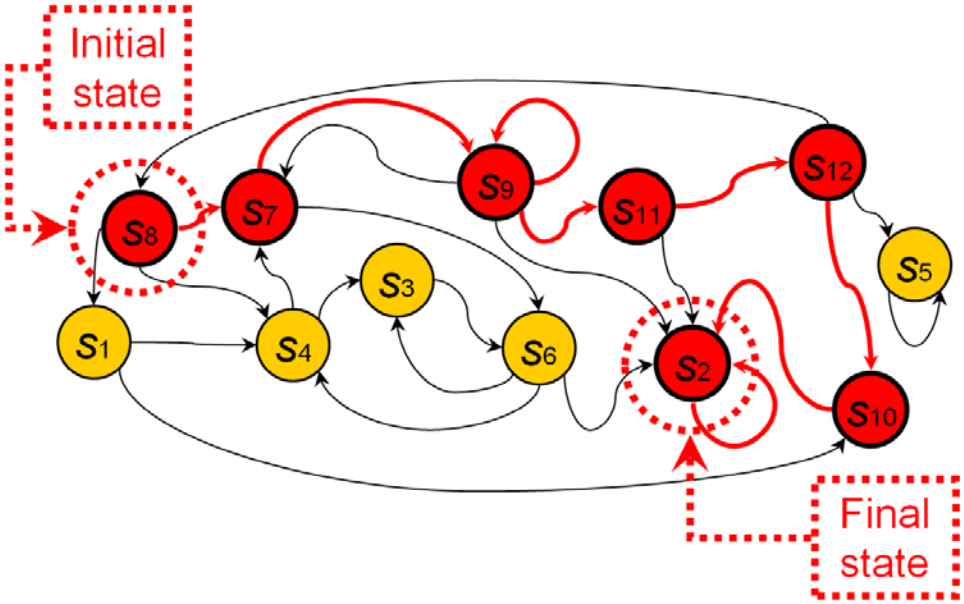

Figure 10. Schematic transition diagram of a Markov chain. Circles denote states and directed arrows indicate possible transitions between states. In this figure, circles and arrows coloured in red represent one possible path where the initial state X_0 is s_8 and the final state is s_2.

Implicitly, our definition of transition probabilities assumes two important properties about the system:

a. The system has the **Markov property**. This means that the present state contains all the information about the future evolution of the system that can be obtained from its past, i.e. given the present state of the system, knowing the past history about how the system reached the present state does not provide any additional information about the future evolution of the system. Formally,

$$\mathrm{P}(X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = \mathrm{P}(X_{n+1} = x_{n+1} \mid X_n = x_n)$$

b. In this chapter we focus on **time-homogeneous** Markov chains, i.e. Markov chains with time-homogeneous transition probabilities. This basically means that transition probabilities $p_{i,j}$ are independent of time, i.e. the one-step transition probability $p_{i,j}$ depends on i and j but is the same at all times n. Formally,

$$\mathrm{P}(X_{n+1} = j \mid X_n = i) = \mathrm{P}(X_n = j \mid X_{n-1} = i) = p_{i,j}$$

The crucial step in the process of representing a computer model as a time-homogeneous Markov chain (THMC) consists in identifying an appropriate set of state variables. A particular combination of specific values for these state variables will define one particular state of the system. Thus, the challenge consists in choosing the set of state variables in such a way that the computer model can be represented as a THMC. In other words, the set of state variables must be such that one can see the computer model as a transition matrix that unambiguously determines the probability of going from any state to any other state.

Example: A simple random walk

Let us consider a model of a simple 1-dimensional random walk and try to see it as a THMC. In this model –which can be run and downloaded at http://luis.izquierdo.name/models/randomwalk– there are 17 patches in line, labelled with the integers between 1 and 17. A random walker is initially placed on one of the patches. From then onwards, the random walker will move randomly to one of the spatially contiguous patches in every time-step (staying still is not an option). Space does not wrap around, i.e. patch 1's only neighbour is patch 2.

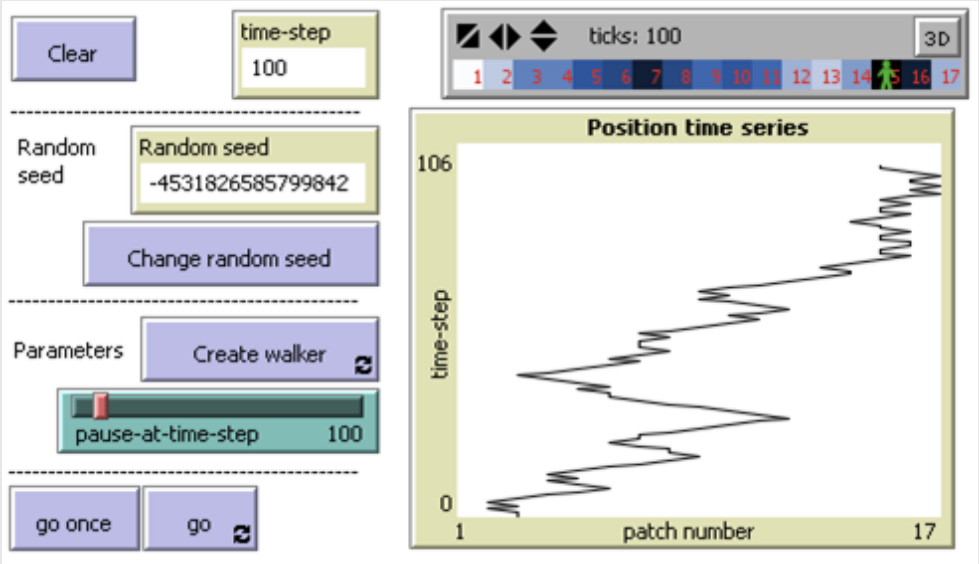

Figure 11. Snapshot of the 1-dimensional random walk applet. Patches are arranged in a horizontal line on the top right corner of the figure; they are labelled with red integers, and coloured in shades of blue according to the number of times that the random walker has visited them: the higher the number of visits, the darker the shade of blue. The plot beneath the patches shows the time series of the random walker's position.

This model can be easily represented as a THMC by choosing the agent's position (e.g. the number of the patch she is standing on) as the only state variable. To be sure, note that defining the state of the system in this way, it is true that there is a fixed probability of going from any state to any other state, independent of time. The transition matrix $P = [p_{i,j}]$ corresponding to the model is:

$$P = [p_{i,j}] = \begin{pmatrix} 0 & 1 & 0 & & & \cdots & 0 \\ 0.5 & 0 & 0.5 & 0 & & & \vdots \\ 0 & 0.5 & 0 & 0.5 & 0 & & \\ & 0 & 0.5 & 0 & 0.5 & 0 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & 0 & 0.5 & 0 & 0.5 & 0 \\ \vdots & & & & 0 & 0.5 & 0 & 0.5 \\ 0 & \cdots & & & & 0 & 1 & 0 \end{pmatrix} \quad [1]$$

Where, as explained above, $p_{i,j}$ is the probability $P(X_{n+1} = j \mid X_n = i)$ that the system will be in state j in the following time-step, knowing that it is currently in state i.

7.2 Transient distributions of finite THMCs

The analysis of the dynamics of THMCs is usually divided into two parts: transient dynamics (finite time) and asymptotic dynamics (infinite time). The transient behaviour is characterised by the distribution of the state of the system X_n for a fixed time-step $n \geq 0$. The asymptotic behaviour (see sections 7.3 and 7.4) is characterised by the limit of the distribution of X_n as n goes to infinity, when this limit exists.

This section explains how to calculate the transient distribution of a certain THMC, i.e. the distribution of X_n for a fixed $n \geq 0$. In simple words, we are after a vector $a^{(n)}$ containing the probability of finding the process in each possible state in time-step n. Formally, $a^{(n)} = [a_1{}^{(n)}, \ldots, a_M{}^{(n)}]$, where $a_i{}^{(n)} = \mathrm{P}(X_n = i)$, denotes the distribution of X_n for a THMC with M possible states. In particular, $a^{(0)}$ denotes the initial distribution over the state space, i.e. $a_i{}^{(0)} = \mathrm{P}(X_0 = i)$. Note that there is no problem in having uncertain initial conditions, i.e. probability functions over the space of possible inputs to the model.

It can be shown that one can easily calculate the transient distribution in time-step n, simply by multiplying the initial conditions by the n-th power of the transition matrix P.

Proposition 1. $a^{(n)} = a^{(0)} \cdot P^n$.

Thus, the elements $p^{(n)}{}_{i,j}$ of P^n represent the probability that the system is in state j after n time-steps having started in state i, i.e. $p^{(n)}{}_{i,j} = \mathrm{P}(X_n = j \mid X_0 = i)$. A straightforward corollary of Proposition 1 is that $a^{(n+m)} = a^{(n)} \cdot P^m$.

As an example, let us consider the 1-dimensional random walk again. Imagine that the random walker starts at an initial random location, i.e. $a^{(0)} = [1/17, \ldots, 1/17]$. The exact distribution of the walker's position in time-step 100 would then be $a^{(100)} = a^{(0)} \cdot P^{100}$. This distribution is represented in Figure 10, together with an empirical distribution obtained by running the model 50 000 times.

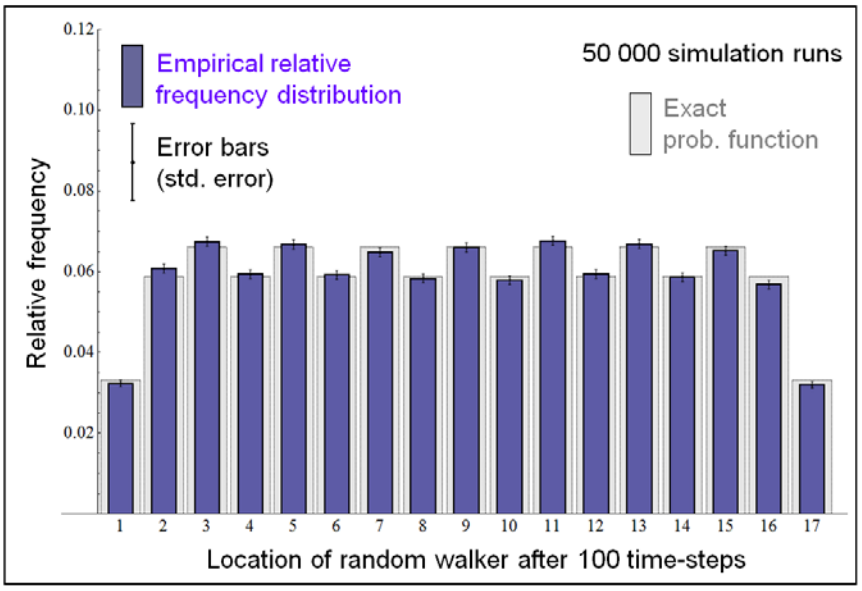

Figure 12. Probability function of the position of the 1-dimensional random walker in time-step 100, starting at an initial random location.

Having obtained the probability function over the states of the system for any fixed n, namely the probability mass function of X_n, it is then straightforward to calculate the distribution of *any* statistic that can be extracted from the model. As argued in the previous sections, the state of the system fully characterises it, so *any* statistic that we obtain about the computer model in time-step n must be, ultimately, a function of $\{X_0, X_1, \ldots, X_n\}$.

Admittedly, the transition matrix of most computer models cannot be easily derived, or it is unfeasible to operate with it. Nonetheless, this apparent drawback is not as important as one might expect. As we shall see below, it is often possible to infer many properties of a THMC even without knowing the exact values of its transition matrix, and these properties can yield useful insights about the dynamics of the associated process. Knowing the exact values of the transition matrix allows us to calculate the exact transient distributions using Proposition 1; this is desirable but not critical, since we can always approximate these distributions by conducting many simulation runs, as explained in section 6.

7.3 Important concepts

This section presents some basic concepts that will prove useful to analyse the dynamics of computer models. The notation used here follows the excellent book on stochastic processes written by Kulkarni (1995).

Definition 1: Accessibility
A state j is said to be accessible from state i if starting at state i there is a chance that the system may visit state j at some point in the future. By convention, every state is accessible from itself. Formally, a state j is said to be accessible from state i if for some $n \geq 0$, $p^{(n)}_{i,j} > 0$.

Note that j is accessible from $i \neq j$ if and only if there is a directed path from i to j in the transition diagram. In that case, we write $i \rightarrow j$. If $i \rightarrow j$ we also say that i leads to j. As an example, in the THMC represented in Figure 10, s_2 is accessible from s_{12} but not from s_5. Note that the definition of accessibility does not depend on the actual magnitude of $p^{(n)}_{i,j}$, only on whether it is exactly zero or strictly positive.

Definition 2: Communication
A state i is said to communicate with state j if $i \rightarrow j$ and $j \rightarrow i$.

If i communicates with j we also say that i and j communicate and write $i \leftrightarrow j$. As an example, note that in the simple random walk presented in section 7.1, every state communicates with every other state. It is worth noting that the relation "communication" is transitive, i.e.

$$i \leftrightarrow j \ , \ j \leftrightarrow k \quad \Longrightarrow \quad i \leftrightarrow k.$$

Definition 3: Communicating class
A set of states $C \subset S$ is said to be a communicating class if:

- Any two states in the communicating class communicate with each other. Formally,

$$i \in C, j \in C \quad \Longrightarrow \quad i \leftrightarrow j$$

- The set C is maximal, i.e. no strict superset of a communicating class can be a communicating class. Formally,

$$i \in C, i \leftrightarrow j \quad \Longrightarrow \quad j \in C$$

As an example, note that in the simple random walk presented in section 7.1 there is one single communicating class that contains all the states. In the THMC represented in Figure 10 there are 4 communicating classes: $\{s_2\}$, $\{s_5\}$, $\{s_{10}\}$, $\{s_1, s_3, s_4, s_6, s_7, s_8, s_9, s_{11}, s_{12}\}$.

Definition 4: Closed communicating class (i.e. absorbing class). Absorbing state.
A communicating class C is said to be closed if no state within C leads to any state outside C. Formally, a communicating class C is said to be closed if $i \in C$ and $j \notin C$ implies that j is not accessible from i.

Note that once a Markov chain visits a closed communicating class, it cannot leave it. Hence we will sometimes refer to closed communicating classes as "absorbing classes". This latter term is not standard in the literature, but we find it useful here for explanatory purposes. Note that if a Markov chain has one single communicating class, it must be closed.

As an example, note that the communicating classes $\{s_{10}\}$ and $\{s_1, s_3, s_4, s_6, s_7, s_8, s_9, s_{11}, s_{12}\}$ in the THMC represented in Figure 10 are not closed, as they can be abandoned. On the other hand, the communicating classes $\{s_2\}$ and $\{s_5\}$ are indeed closed, since they cannot be abandoned. When a closed communicating class consists of one single state, this state is called absorbing. Thus, s_2 and s_5 are absorbing states. Formally, state i is absorbing if and only if $p_{i,i} = 1$ and $p_{i,j} = 0$ for $i \neq j$.

Proposition 2. Decomposition Theorem (Chung, 1960)
The state space S of any Markov chain can be uniquely partitioned as follows:

$$S = C_1 \cup C_2 \cup \ldots \cup C_k \cup T$$

where C_1, C_2, …, C_k are closed communicating classes, and T is the union of all other communicating classes.

Note that we do not distinguish between non-closed communicating classes: we lump them all together into T. Thus, the unique partition of the THMC represented in Figure 10 is $S = \{s_2\} \cup \{s_5\} \cup \{s_1, s_3, s_4, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}\}$. The simple random walk model presented in section 7.1 has one single (closed) communicating class C_1 containing all the possible states, i.e. $S \equiv C_1$.

Definition 5: Irreducibility
A Markov chain is said to be irreducible if all its states belong to a single closed communicating class; otherwise it is called reducible. Thus, the simple random walk presented in section 7.1 is irreducible, but the THMC represented in Figure 10 is reducible.

Definition 6: Transient and recurrent states
A state i is said to be transient if, given that we start in state i, there is a non-zero probability that we will never return back to i. Otherwise, the state is called recurrent. A

Markov chain starting from a recurrent state will revisit it with probability 1, and hence revisit it infinitely often. On the other hand, a Markov chain starting from a transient state has a strictly positive probability of never coming back to it. Thus, a Markov chain will visit any transient state only finitely many times; eventually, transient states will not be revisited anymore.

Definition 7: Periodic and aperiodic states. Periodic and aperiodic communicating classes
A state i has period d if any return to state i must occur in multiples of d time-steps. If $d = 1$, then the state is said to be aperiodic; otherwise ($d > 1$), the state is said to be periodic with period d. Formally, state i's period d is the greatest common divisor of the set of integers $n > 0$ such that $p^{(n)}_{i,i} > 0$. For our purposes, the concept of periodicity is only relevant for recurrent states. As an example, note that every state in the simple random walk presented in section 7.1 is periodic with period 2.

An interesting and useful fact is that if $i \leftrightarrow j$, then i and j must have the same period (see theorem 5.2. in Kulkarni (1995)). In particular, note that if $p_{i,i} > 0$ for any i, then the communicating class to which i belongs must be aperiodic. Thus, it makes sense to qualify communicating classes as periodic with period d, or aperiodic. A closed communicating class with period d can return to its starting state only at times d, $2d$, $3d$, …

The concepts presented in this section will allow us to analyse the dynamics of any finite Markov chain. In particular, we will show that, given enough time, any finite Markov chain will necessarily end up in one of its closed communicating classes (i.e. absorbing classes).

7.4 Limiting behaviour of finite THMCs

This section is devoted to characterising the limiting behaviour of a THMC, i.e. studying the convergence (in distribution) of X_n as n tends to infinity. Specifically, we aim to study the behaviour of $a_i^{(n)} = P(X_n = i)$ as n tends to infinity. From Proposition 1 it is clear that analysing the limiting behaviour of P^n would enable us to characterise $a_i^{(n)}$. There are many introductory books in stochastic processes that offer clear and simple methods to analyse the limiting behaviour of THMCs when the transition matrix P is tractable (see e.g. chapter 5 in (Kulkarni 1999), chapters 2-4 in (Kulkarni 1995), chapter 3 in (Janssen and Manca 2006) or the book chapter written by (Karr 1990)). Nonetheless, we focus here on the general case, where operating with the transition matrix P may be computationally unfeasible.

7.4.1 General dynamics

The first step in the analysis of any THMC consists in identifying all the closed communicating classes, so we can partition the state space S as indicated by the decomposition theorem (see proposition 2). The following proposition (Theorems 3.7 and 3.8 in Kulkarni (1995)) reveals the significance of this partition:

Proposition 3. **General dynamics of finite THMCs**.
Consider a finite THMC that has been partitioned as indicated in proposition 2. Then:
(i) All states in T (i.e. not belonging to a closed communicating class) are transient.
(ii) All states in C_v (i.e. in any closed communicating class) are recurrent; $v \in \{1, 2, \ldots, k\}$.

Proposition 3 states that sooner or later the THMC will enter one of the absorbing classes and stay in it forever. Formally, for all $i \in S$ and all $j \in T$: $\lim_{n\to\infty} p_{i,j}^{(n)} = 0$, i.e. the probability of finding the process in a state belonging to a non-closed communicating class goes to zero as n goes to infinity. Naturally, if the initial state already belongs to an absorbing class C_v, then the chain will never abandon such a class. Formally, for all $i \in C_v$ and all $j \notin C_v$: $p^{(n)}_{i,j} = 0$ for all $n \geq 0$.

As an example of the usefulness of Proposition 3, consider the THMC represented in Figure 10. This THMC has only two absorbing classes: $\{s_2\}$ and $\{s_5\}$. Thus, the partition of the state space is: $S = \{s_2\} \cup \{s_5\} \cup \{s_1, s_3, s_4, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}\}$. Hence, applying Proposition 3 we can state that the process will eventually end up in one of the two absorbing states, s_2 or s_5. The probability of ending up in one or the other absorbing state depends on the initial conditions $a^{(0)}$ (and on the actual numbers $p_{i,j}$ in the transition matrix, of course). Slightly more formally, the limiting distribution of X_n exists, but it is not unique, i.e. it depends on the initial conditions.

7.4.2 Dynamics within absorbing classes

The previous section has explained that any simulation run will necessarily end up in a certain absorbing class; this section characterises the dynamics of a THMC that is already "trapped" in an absorbing class. This is precisely the analysis of irreducible Markov chains, since irreducible Markov chains are, by definition, Markov chains with one single closed communicating class (see definition 5). In other words, one can see any THMC as a set of transient states T plus a finite number of irreducible Markov sub-chains.

Irreducible THMCs behave significantly different depending on whether they are periodic or not. The following sections characterise these two cases.

Irreducible and aperiodic THMCs

Irreducible and aperiodic THMCs are often called ergodic. In these processes the probability function of X_n approaches a limit as n tends to infinity. This limit is called the **limiting distribution**, and is denoted here by π. Formally, the following limit exists and is unique (i.e. independent of the initial conditions $a_i^{(0)}$):

$$\lim_{n\to\infty} a_i^{(n)} = \pi_i > 0 \qquad i \in S$$

Thus, in ergodic THMCs the probability of finding the system in each of its states in the long run is strictly positive and independent of the initial conditions (Theorems 3.7 and 3.15 in Kulkarni (1995)). As previously mentioned, calculating such probabilities may be unfeasible, but we can estimate them sampling many simulation runs at a sufficiently large time-step.

Importantly, in ergodic THMCs the limiting distribution π coincides with the **occupancy distribution** π^*, which is the long-run fraction of the time that the THMC spends in each state[11]. Naturally, the occupancy distribution π^* is also independent of

[11] Formally, the occupancy of state i is defined as:

the initial conditions. Thus, in ergodic THMCs, running just one simulation for long enough (which enables us to estimate π^*) will serve to estimate π just as well.

The question that comes to mind then is: How long is long enough? i.e. when will I know that the empirical distribution obtained by simulation resembles the limiting distribution π? Unfortunately there is no answer for that. The silver lining is that knowing that the limiting and the occupancy distribution coincide, that they must be stable in time, and that they are independent of the initial conditions, enables us to conduct a wide range of tests that may tell us when it is certainly *not* long enough. For example, we can run a battery of simulations and study the empirical distribution over the states of the system across samples as time goes by. If the distribution is not stable, then we have not run the model for long enough. Similarly, since the occupancy distribution is independent of the initial conditions, one can run several simulations with widely different initial conditions, and compare the obtained occupancy distributions. If the empirical occupancy distributions are not similar, then we have not run the model for long enough. Many more checks can be conducted.

Admittedly, when analysing a computer model one is often interested not so much in the distribution over the possible states of the system, but rather in the distribution of a certain statistic. The crucial point is to realise that if the statistic is a function of the state of the system (and all statistics that can be extracted from the model are), then the limiting and the occupancy distributions of the statistic exist, coincide and are independent of the initial conditions.

Irreducible and periodic THMCs

In contrast with aperiodic THMCs, the probability distribution of X_n in periodic THMCs does not approach a limit as n tends to infinity. Instead, in an irreducible THMC with period d, as n tends to infinity, X_n will in general cycle through d probability functions depending on the initial distribution.

As an example, consider the simple random walk again (which is irreducible and periodic, with period 2), and assume that the random walker starts at patch number 1 (i.e. $X_0 = 1$). Given these settings, it can be shown that

$$\lim_{n\to\infty} a_i^{(2n)} = [\frac{1}{16}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{16}]$$

$$\lim_{n\to\infty} a_i^{(2n+1)} = [0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0]$$

In particular, the limits above show that the random walker cannot be at a patch with an even number in any even time-step, and he cannot be at a patch with an odd number in any odd time-step. In contrast, if the random walker started at patch number 2 (i.e. $X_0 = 2$), then the limits above would be interchanged.

$$\pi_i^* = \lim_{n\to\infty} \frac{E(N_i(n))}{n+1}$$

where $N_i(n)$ denotes the number of times that the THMC visits state i over the time span $\{0, 1, \ldots, n\}$.

Fortunately, every irreducible (periodic or aperiodic) THMC does have a unique occupancy distribution π^*, independent of the initial conditions (see Theorem 5.19 in Kulkarni (1999)). In our particular example, this is:

$$\pi^* = [\frac{1}{32}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{32}]$$

Thus, the long-run fraction of time that the system spends in each state in any irreducible THMC is unique (i.e. independent of the initial conditions). This is a very useful result, since any statistic which is a function of the state of the system will also have a unique occupancy distribution independent of the initial conditions. As explained before, this occupancy distribution can be approximated with one single simulation run, assuming it runs for long enough.

8 Synergies between Mathematical analysis and Computer simulation.

In this section we present various ways in which mathematical analysis and computer simulation can be combined to produce a better understanding of the dynamics of a model. Note that a full understanding of the dynamics of a model involves not only characterising (i.e. describing) them, but also finding out *why* such dynamics are being observed, i.e. identifying the subsets of rules that are necessary or sufficient to generate certain aspects of the observed dynamics. To do this, one often has to make changes in the model, i.e. build supporting models that differ only slightly from the original one and may yield useful insights about its dynamics. These supporting models will sometimes be more tractable (e.g. if heterogeneity or stochasticity are averaged out) and sometimes more complex (e.g. if interactions that were assumed to be global in the original model may only take place locally in the supporting model). Thus, for clarity, we distinguish three different cases and deal with them in turn (see Figure 13):

1. Characterisation of the dynamics of a model.
2. Moves towards greater mathematical tractability. This involves creating and studying supporting models that are simpler than the original one.
3. Moves towards greater mathematical complexity. This involves creating and studying supporting models that are less tractable than the original one.

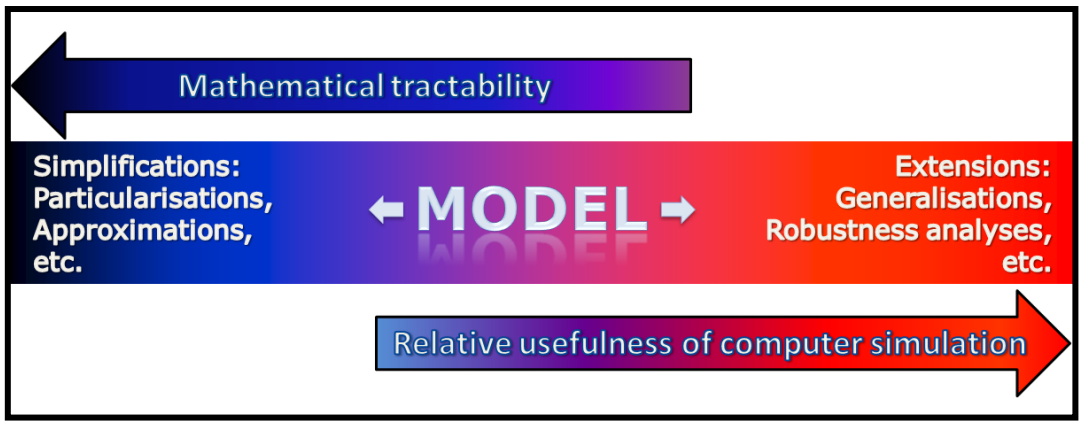

Figure 13. To fully understand the dynamics of a model, one often has to study supporting models that differ only slightly from the original one. Some of these supporting models may be more tractable whilst others may be more complex.

8.1 Characterising the dynamics of the model

There are many types of mathematical techniques that can be usefully combined with computer simulation to characterise the dynamics of a model (e.g. Stochastic Approximation Theory (Benveniste et al. 1990; Kushner and Yin 1997)), but for limitations of space we focus here on Markov chain analysis only.

When using Markov chain analysis to characterise the dynamics of a model it may happen that the transition matrix can be easily computed and we can operate with it, or it may not. In the former case –which is quite rare in Social Simulation models–, one can provide a full characterisation of the dynamics of the model just by operating with the transition matrix (see Proposition 1 and the beginning of section 7.4 for references). In general, however, deriving and operating with the transition matrix may be unfeasible, and it is in this common case where there is a lot to gain in using Markov chain analysis and computer simulation together. The overall method goes as follows:

- Use Markov chain analysis to assess the relevance of initial conditions and to identify the different regimes in which the dynamics of the model may end up trapped.
- Use the knowledge acquired in the previous point to design suitable computational experiments aimed at estimating the exact probability distributions for the relevant statistics (which potentially depend on the initial conditions).

The following describes this overall process in greater detail. Naturally, the first step consists in finding an appropriate definition of the state of the system, as explained in section 7.1. The next step is to identify all the closed communicating (i.e. absorbing) classes in the model C_v ($v \in \{1, 2, \ldots, k\}$). This allows us to partition the state space of the Markov chain as the union of all the closed communicating classes C_1, C_2, …, C_k in the model plus another class T containing all the states that belong to non-closed communicating classes. Izquierdo et al. (2009) illustrate how to do this in 10 well-known models in the Social Simulation literature.

In most cases, conducting the partition of the state space is not as difficult as it may seem at first. In particular, the following proposition provides some simple sufficient conditions that guarantee that the computer model contains one single aperiodic absorbing class, i.e. the finite THMC that the computer model implements is irreducible and aperiodic (i.e. ergodic).

<u>Proposition 4</u>. **Sufficient conditions for irreducibility and aperiodicity**.

(i) If it is possible to go from any state to any other state in one single time-step ($p_{i,j} > 0$ for all $i \neq j$) and there are more than 2 states, then the THMC is irreducible and aperiodic.

(ii) If it is possible to go from any state to any other state in a finite number of time-steps ($i \leftrightarrow j$ for all $i \neq j$), and there is at least one state in which the system may stay for two consecutive time-steps ($p_{i,i} > 0$ for some i), then the THMC is irreducible and aperiodic.

(iii) If there exists a positive integer n such that $p^{(n)}_{i,j} > 0$ for all i and j, then the THMC is irreducible and aperiodic (Janssen and Manca 2006, p. 107).

If one sees the transition diagram of the Markov chain as a (directed) network, the conditions above can be rewritten as:

(i) The network contains more than 2 nodes and there is a directed link from every node to every other node.
(ii) The network is strongly connected and there is at least one loop.
(iii) There exists a positive integer n such that there is at least one walk of length n from any node to every node (including itself).

Izquierdo et al. (2009) show that many models in the Social Simulation literature satisfy one of these sufficient conditions (e.g. Epstein and Axtell's (1996) Sugarscape, Axelrod's (1986) metanorms models, Takahashi's (2000) model of generalized exchange, and Miller and Page's (2004) standing ovation model with noise). This is important since, as explained in section 7.4.2, in ergodic THMCs the limiting and the occupancy distributions of any statistic exist, coincide and are independent of the initial conditions (so running just one simulation for long enough, which enables us to estimate the occupancy distribution, will serve to estimate the limiting distribution just as well).

Let us return to the general case. Having partitioned the state space, the analysis of the dynamics of the model is straightforward: all states in T (i.e. in any finite communicating class that is not closed) are transient, whereas all states in C_v (i.e. in any finite closed communicating class) are recurrent. In other words, sooner or later any simulation run will enter one of the absorbing classes C_v and stay in it forever.

Here computer simulation can play a crucial role again, since it allows us to estimate the probability of ending up in each of the absorbing classes for any (stochastic or deterministic) initial condition we may be interested in. A case-in-point would be a model that has only a few absorbing states, or where various absorbing states are put together into only a few groups. Izquierdo et al. (2009) analyse models that follow that pattern: Axelrod's (1997b) model of dissemination of culture, Arthur's (1989) model of competing technologies, and Axelrod and Bennett's (1993) model of competing bimodal coalitions. CharityWorld (Polhill et al. 2006; Izquierdo and Polhill 2006) is an example of a model with a unique absorbing state.

The following step consists in characterising the dynamics of the system within each of the absorbing classes. Once the system has entered a certain absorbing class C_v, it will remain in it forever exhibiting a unique conditional[12] occupancy distribution π_v* over the set of states that compose C_v. Naturally, the same applies to any statistic we may want to study, since all statistics that can be extracted from the model are a function of the state of the system.

The conditional occupancy distribution π_v* denotes the (strictly positive) long-run fraction of the time that the system spends in each state of C_v given that the system has entered C_v. Importantly, the conditional occupancy distribution π_v* is the same regardless of the specific state through which the system entered C_v. The role of simulation here is to estimate these conditional occupancy distributions for the relevant statistics by running the model for long enough.

Finally, recall that some absorbing classes are periodic and some are aperiodic. Aperiodic absorbing classes have a unique conditional limiting distribution π_v denoting

[12] Given that the system has entered the absorbing class C_v.

the long-run (strictly positive) probability of finding the system in each of the states that compose C_v given that the system has entered C_v. This conditional limiting distribution π_v coincides with the conditional occupancy distribution π_v* and, naturally, is also independent of the specific state through which the system entered C_v. (Again, note that this also applies to the distribution of any statistic, as they are all functions of the state of the system, necessarily.)

In contrast with aperiodic absorbing classes, periodic absorbing classes do not generally have a unique limiting distribution; instead, they cycle through d probability functions depending on the specific state through which the system entered C_v (where d denotes the period of the periodic absorbing class). This is knowledge that one must take into account at the time of estimating the relevant probability distributions using computer simulation.

Thus, it is clear that Markov chain analysis and computer simulation greatly complement each other. Markov chain analysis provides the overall picture of the dynamics of the model by categorising its different dynamic regimes and identifying when and how initial conditions are relevant. Computer simulation uses this information to design appropriate computational experiments that allow us to quantify the probability distributions of the statistics we are interested in. As explained above, these probability distributions can always be approximated with any degree of accuracy by running the computer model several times.

There are several examples of this type of synergetic combination of Markov chain analysis and computer simulation in the literature. Galan and Izquierdo (2005) analysed Axelrod's (1986) agent-based model as a Markov chain, and concluded that the long-run behaviour of that model was independent of the initial conditions, in contrast to the initial conclusions of the original analysis. Galan and Izquierdo (2005) also used computer simulation to estimate various probability distributions. Ehrentreich (2002; 2006) used Markov chain analysis on the Artificial Stock Market (Arthur et al. 1997; LeBaron et al. 1999) to demonstrate that the mutation operator implemented in the model is not neutral to the learning rate, but introduces an upward bias[13]. A more positive example is provided by Izquierdo et al. (2007; 2008b), who used Markov chain analysis and computer simulation to confirm and advance various insights on reinforcement learning put forward by Macy and Flache (2002) and Flache and Macy (2002).

8.2 Moves towards greater mathematical tractability: Simplifications

There are at least two types of simplifications that can help us to better understand the dynamics of a model. One consists in studying specific parameterisations of the original model that are thought to lead to particularly simple dynamics, or to more tractable situations (Gilbert and Terna 2000; Gilbert 2007). Examples of this type of activity would be to run simulations without agents or with very few agents, explore the behaviour of the model using extreme parameter values, model very simple environments, etc. This activity is common practice in the field (see e.g. Gotts et al. 2003c, 2003d).

[13] This finding does not refute some of the most important conclusions obtained by the authors of the original model.

A second type of simplification consists in creating an abstraction of the original model (i.e. a model of the model) which is mathematically tractable. An example of one possible abstraction would be to study the *expected* motion of the dynamic system (see the studies conducted by Galan and Izquierdo (2005), Edwards et al. (2003), Castellano et al. (2000), Huet et al. (2007), Mabrouk et al. (2007), Vilà (2008) and Izquierdo et al. (2007; 2008b) for illustrations of mean-field approximations). Since these mathematical abstractions do not correspond in a one-to-one way with the specifications of the formal model, any results obtained with them will not be conclusive in general, but they may give us insights suggesting areas of stability and basins of attraction, clarifying assumptions, assessing sensitivity to parameters, or simply giving the option to illustrate graphically the expected dynamics of the original model. This approach can also be used as a verification technique to detect potential errors and artefacts (Galán et al. 2009).

8.3 Moves towards greater mathematical complexity: Extensions

As argued before, understanding the dynamics of a model implies identifying the set of assumptions that are responsible for particular aspects of the obtained results. Naturally, to assess the relevance of any assumption in a model, it is useful to replace it with other alternatives, and this often leads to greater mathematical complexity[14].

Ideally, the evaluation of the significance of an assumption is conducted by generalisation, i.e. by building a more general model that allows for a wide range of alternative competing assumptions, and contains the original assumption as a particular case. An example would be the introduction of arbitrary social networks of interaction in a model where every agent necessarily interacts with every other agent. In this case, the general model with arbitrary networks of interaction would correspond with the original model if the network is assumed to be complete, but any other network could also be studied within the same common framework. Another example is the introduction of noise in deterministic models.

Building models by generalisation is useful because it allows for a transparent, structured and systematic way of exploring the impact of various alternative assumptions that perform the same role in the model, but it often implies a loss in mathematical tractability (see e.g. Izquierdo and Izquierdo 2006). Thus, it is often the case that a rigorous study of the impact of alternative assumptions in a model requires being prepared to slide up and down the tractability continuum depicted in Figure 13 (Gotts et al. 2003b). In fact, all the cases that are mentioned in the rest of this section involved greater complexity than the original models they considered, and computer simulation had to be employed to understand their dynamics.

In the literature there are many examples of the type of activity explained in this section. For example, Klemm et al. studied the relevance of various assumptions in Axelrod's model of dissemination of culture (1997b) by changing the network topology (Klemm et al. 2003a), investigating the role of dimensionality (Klemm et al. 2003b, 2005), and introducing noise (Klemm et al. 2003c). Another example is given by Izquierdo and

[14] This is so because many assumptions we make in our models are, to some extent, for the sake of simplicity. As a matter of fact, in most cases the whole purpose of modelling is to build an abstraction of the world which is simpler than the world itself, so we can make inferences about the model that we cannot make directly from the real world (Edmonds 2001; Galán et al. 2009; Izquierdo et al. 2008a).

Izquierdo (2007), who analysed the impact of using different structures of social networks in the efficiency of a market with quality variability.

In the context of decision-making and learning, Flache and Hegselmann (1999) and Hegselmann and Flache (2000) compared two different decision-making algorithms that a set of players can use when confronting various types of social dilemmas. Similarly, Takadama et al. (2003) analysed the effect of three different learning algorithms within the same model.

Several authors, particularly in the literature of Game Theory, have investigated the effect of introducing noise in the decision-making of agents. This is useful not only to investigate the general effect of potential mistakes or experimentation, but also to identify the stochastic stability of different outcomes (see section 10 in Izquierdo et al. (2009)). An illustrative example is given by Izquierdo et al. (2008b), who investigate the reinforcement learning algorithm proposed by Bush and Mosteller (1955) using both mathematical analysis and simulation, and find that the inclusion of small quantities of randomness in players' decisions can change the dynamics of the model dramatically.

Another assumption investigated in the literature is the effect of different spatial topologies (see e.g. Flache and Hegselmann (2001), who generalised two of their cellular automata models by changing their –originally regular– grid structure). Finally, as mentioned in section 5, it is increasingly common in the field of evolutionary game theory to assess the impact of various assumptions using computer simulation (see e.g. Galan and Izquierdo 2005; Santos et al. 2006; Traulsen et al. 2006; Izquierdo and Izquierdo 2006).

9 Summary

In this chapter we have provided a set of guidelines to understand the dynamics of computer models using both simulation and mathematical analysis. In doing so, it has become clear that mathematical analysis and computer simulation should not be regarded as alternative –or even opposed– approaches to the formal study of social systems, but as complementary (Gotts et al. 2003a, 2003b). Not only can they provide fundamentally different insights on the same model, but they can also produce hints for solutions for each other. In short, there are plenty of synergies to be exploited by using the two techniques together, so the full potential of each technique cannot be reached unless they are used in conjunction.

To understand the dynamics of any particular computer model, we have seen that it is useful to see the computer model as the implementation of a function that transforms probability distributions over the set of possible inputs into probability distributions over the set of possible outputs. We refer to this function as the formal model that the computer model implements.

The mathematical approach to analyse formal models consists in examining the rules that define the model directly; the aim is to deduce the logical implications of these rules for any particular instance to which they can be applied. Our analysis of mathematical techniques to study formal models has been focused on the theory of Markov Chains. This theory is particularly useful for our purposes since many computer models can be meaningfully represented as time-homogenous Markov chains.

In contrast with mathematical analysis, the computer simulation approach does not look at the rules that define the formal model directly, but instead tries to infer general properties of these rules by examining the outputs they produce when applied to particular instances of the input space. Thus, in the simulation approach, the data is produced by the computer using strict logical deduction, but the general patterns about how the rules transform the inputs into the outputs are inferred using generalisation by induction. Thus, in the general case –and in contrast with mathematical analysis–, the inferences obtained using computer simulation will not be necessarily correct in a strict logical sense; but, on the other hand, computer simulation enables us to explore formal models beyond mathematical tractability, and the confidence we can place on the conclusions obtained with this approach can be rigorously assessed in statistical terms. Furthermore, as shown in this chapter, we can achieve any arbitrary level of accuracy in our computational approximations by running the model sufficiently many times.

Bearing in mind the relative strengths and limitations of both approaches, we have identified at least three different ways in which mathematical analysis and computer simulation can be usefully combined to produce a better understanding of the dynamics of computer models.

The first synergy appears at the time of characterising the dynamics of the formal model under study. To do that, we have shown how Markov chain analysis can be used to provide an overall picture of the dynamics of the model by categorising its different dynamic regimes and identifying when and how initial conditions are relevant. Having conducted such an analysis, one can then use computer simulation to design appropriate computational experiments with the aim of quantifying the probability distributions of the variables we are interested in. These probability distributions can always be approximated with any degree of accuracy by running the computer model several times.

The two other ways in which mathematical analysis and computer simulation can be combined derive from the fact that understanding the dynamics of a model involves not only characterising (i.e. describing) them, but also finding out *why* such dynamics are being observed (i.e. discover causality). This often implies building supporting models that can be simpler or more complex than the original one. The rationale to move towards simplicity is to achieve greater mathematical tractability, and this often involves studying particularly simple parameterisations of the original model, and creating abstractions which are amenable to mathematical analysis. The rationale to move towards complexity is to assess the relevance of specific assumptions, and it often involves building generalisations of the original model to explore the impact of competing assumptions that can perform the same role in the model but may lead to different results.

Let us conclude by encouraging the reader to put both mathematical analysis and computer simulation in their backpack, and be happy to glide up and down the tractability spectrum where both simple and complex models lie. The benefits are out there.

Acknowledgements

The authors have benefited from the financial support of the Spanish Ministry of Education and Science (projects DPI2005-05676 and TIN2008-06464-C03-02) and of the JCyL (projects VA006B09 and BU034A08). We are also very grateful to Nick Gotts, Bruce Edmonds and Gary Polhill for many extremely useful discussions.

Reading List

- EPSTEIN J M (2006) Remarks on the Foundations of Agent-Based Generative Social Science. In Judd K L and Tesfatsion L (Eds.) *Handbook of Computational Economics, Vol. 2: Agent-Based Computational Economics*, Chapter 34. Amsterdam, The Netherlands: North-Holland.
  - This chapter treats a variety of foundational and epistemological issues surrounding generative explanation in the social sciences, and discusses the role of agent-based computational models in generative social science.

- GRINSTEAD, C M and Snell, J L (1997) Introduction to Probability: Second Revised Edition. Chapter 11: Markov chains. *American Mathematical Society*. Available for download under the terms of the GNU Free Documentation License (FDL) at http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/book.html.
  - This chapter provides an excellent introduction to the theory of finite Markov Chains, with many examples and exercises.

- HÄGGSTRÖM, O (2002) *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press.
  - This book provides a clear and concise introduction to Probability theory and Markov Chain theory, and then illustrates the usefulness of these theories by studying a range of stochastic algorithms with important applications in optimisation and other problems in computing. One of the algorithms covered is the Markov chain Monte Carlo method.

- IZQUIERDO L R, Izquierdo, S S, Galán, J M and Santos, J I (2009) Techniques to Understand Computer Simulations: Markov Chain Analysis. Journal of Artificial Societies and Social Simulation, 12(1) 6 http://jasss.soc.surrey.ac.uk/12/1/6.html.
  - This paper analyses the dynamics of 10 well-known models in the Social Simulation literature using the theory of Markov Chains.

- KULKARNI, V G (1995). *Modeling and Analysis of Stochastic Systems*. Chapman & Hall/CRC.
  - This excellent book provides a rigorous analysis of many types of useful stochastic processes, e.g. discrete and continuous time Markov Chains, renewal processes, regenerative processes, and Markov regenerative processes.

- LEOMBRUNI, R and Richiardi, M (2005) Why are economists sceptical about agent-based simulations? *Physica A: Statistical Mechanics and its Applications* 355(1), pp. 103-109.
  - This paper discusses several issues surrounding the interpretation of simulation dynamics and the generalisation of the simulation results.

References

ARTHUR W B (1989) Competing technologies, increasing returns, and lock-in by historical events. *Economic Journal*, 99(394), pp. 116-131

ARTHUR W B, Holland J H, LeBaron B, Palmer R, and Tayler P (1997) Asset Pricing under Endogenous Expectations in an Artificial Stock Market. In Arthur W B, Durlauf S, and Lane D (Eds.) *The Economy as an Evolving Complex System II*: 15-44. Reading, MA: Addison-Wesley Longman

AXELROD R M (1986) An Evolutionary Approach to Norms. *American Political Science Review*, 80(4), pp. 1095-1111

AXELROD R M (1997a) Advancing the Art of Simulation in the Social Sciences. In Conte R, Hegselmann R, and Terna P (Eds.) *Simulating Social Phenomena, Lecture Notes in Economics and Mathematical Systems 456*: 21-40. Berlin: Springer-Verlag

AXELROD R M (1997b) The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, 41(2), pp. 203-226

AXELROD R M and Bennett D S (1993) A Landscape Theory of Aggregation. *British Journal of Political Science*, 23(2), pp. 211-233

AXTELL R L (2000) Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences. In Macal C M and Sallach D (Eds.) *Proceedings of the Workshop on Agent Simulation: Applications, Models, and Tools*: 3-24. Argonne, IL: Argonne National Laboratory

AXTELL R L and Epstein J M (1994) Agent Based Modeling: Understanding Our Creations. *The Bulletin of the Santa Fe Institute*, Winter 1994, pp. 28-32

BALZER W, Brendel K R, and Hofmann S (2001) Bad Arguments in the Comparison of Game Theory and Simulation in Social Studies. *Journal of Artificial Societies and Social Simulation*, 4(2)1. http://jasss.soc.surrey.ac.uk/4/2/1.html.

BENVENISTE A, Métivier M, and Priouret P (1990) *Adaptive Algorithms and Stochastic Approximations*. Berlin: Springer-Verlag.

BÖHM C and Jacopini G (1966) Flow diagrams, turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5), pp. 366-371

BUSH R R and Mosteller F (1955). *Stochastic Models for Learning*. New York: John Wiliey & Son.

CASTELLANO C, Marsili M, and Vespignani A (2000) Nonequilibrium phase transition in a model for social influence. *Physical Review Letters*, 85(16), pp. 3536-3539

CUTLAND N (1980) *Computability: An Introduction to Recursive Function Theory*. Cambridge: Cambridge University Press.

EDMONDS B (2001) The Use of Models - making MABS actually work. In Moss S and Davidsson P (Eds.) *Multi-Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1979*: 15-32. Berlin: Springer-Verlag

EDMONDS B (2005) Simulation and Complexity - how they can relate. In Feldmann V and Mühlfeld K (Eds.) *Virtual Worlds of Precision - computer-based simulations in the sciences and social sciences*: 5-32. Münster, Germany: Lit-Verlag

EDWARDS M, Huet S, Goreaud F, and Deffuant G (2003) Comparing an individual-based model of behaviour diffusion with its mean field aggregate approximation. *Journal of Artificial Societies and Social Simulation*, 6(4)9. http://jasss.soc.surrey.ac.uk/6/4/9.html.

EHRENTREICH N (2002) The Santa Fe Artificial Stock Market Re-Examined - Suggested Corrections. *Economics Working Paper Archive at WUSTL*. http://econwpa.wustl.edu:80/eps/comp/papers/0209/0209001.pdf.

EHRENTREICH N (2006) Technical trading in the Santa Fe Institute Artificial Stock Market revisited. *Journal of Economic Behavior & Organization*, 61(4), pp. 599-616

EPSTEIN J M (2006) Remarks on the Foundations of Agent-Based Generative Social Science. In Judd K L and Tesfatsion L (Eds.) *Handbook of Computational Economics, Vol. 2: Agent-Based Computational Economics*: Amsterdam, The Netherlands: North-Holland

EPSTEIN J M and Axtell R L (1996) *Growing Artificial Societies. Social Science From the Bottom Up*. Cambridge, MA: Brookings Institution Press-MIT Press.

FLACHE A and Hegselmann R (1999). Rationality vs. Learning in the Evolution of Solidarity Networks: A Theoretical Comparison. *Computational & Mathematical Organization Theory* 5(2), pp. 97-127.

FLACHE A and Hegselmann R (2001) Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics. *Journal of Artificial Societies and Social Simulation*, 4(4)6. http://jasss.soc.surrey.ac.uk/4/4/6.html.

FLACHE A and Macy M W (2002) Stochastic Collusion and the Power Law of Learning. *Journal of Conflict Resolution*, 46(5), pp. 629-653

GALAN J M and Izquierdo L R (2005) Appearances Can Be Deceiving: Lessons Learned Re-Implementing Axelrod's 'Evolutionary Approach to Norms'. *Journal of Artificial Societies and Social Simulation*, 8(3)2. http://jasss.soc.surrey.ac.uk/8/3/2.html.

GALÁN J M, Izquierdo L R, Izquierdo S S, Santos J I, del Olmo R, López-Paredes A, and Edmonds B (2009) Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation*, *Journal of Artificial Societies and Social Simulation*, 12(1)1 http://jasss.soc.surrey.ac.uk/12/1/1.html.

GENZ A and Kwong K S (2000) Numerical evaluation of singular multivariate normal distributions. *Journal of Statistical Computation and Simulation*, 68(1), pp. 1-21

GILBERT N (1999) Simulation: A new way of doing social science. *The American Behavioral Scientist*, 42(10), pp. 1485-1487

GILBERT, N (2007) *Agent-Based Models*. Series: Quantitative Applications in the Social Sciences. Sage Publications: London.

GILBERT N and Terna P (2000) How to build and use agent-based models in social science. *Mind and Society*, 1(1), pp. 57-72

GILBERT N and Troitzsch K G (1999) *Simulation for the social scientist*. Buckingham, UK: Open University Press.

GOTTS N M, Polhill J G and Law A N R (2003a) Agent-based simulation in the study of social dilemmas. *Artificial Intelligence Review*, 19 (1), pp. 3-92

GOTTS N M, Polhill J G, and Adam W J (2003b) Simulation and Analysis in Agent-Based Modelling of Land Use Change. *Online proceedings of the First Conference of the European Social Simulation Association, Groningen, The Netherlands, 18-21 September 2003*. http://www.uni-koblenz.de/~essa/ESSA2003/proceedings.htm.

GOTTS N M, Polhill J G, and Law A N R (2003c) Aspiration levels in a land-use simulation. *Cybernetics and Systems* 34 (8), pp. 663-683.

GOTTS N M, Polhill J G, Law A N R, and Izquierdo, L.R. (2003d). Dynamics of imitation in a land use simulation. In: *Proceedings of the Second International Symposium on Imitation in Animals and Artefacts*, University of Wales, Aberystwyth, 7th-11th April 2003, pp 39-46.

HAREL D (1980) On folk theorems. *Communications of the ACM*, 23(7), pp. 379-389

HAUERT C and Doebeli M (2004) Spatial structure often inhibits the evolution of cooperation in the snowdrift game. *Nature*, 428(6983), pp. 643-646

HEGSELMANN R and Flache A (2000). Rational and Adaptive Playing. *Analyse & Kritik* 22(1), pp. 75-97.

HOLLAND J H and Miller J H (1991) Artificial adaptive agents in economic theory. *American Economic Review*, 81(2), pp. 365-370

HUET S, Edwards M, and Deffuant G (2007) Taking into Account the Variations of Neighbourhood Sizes in the Mean-Field Approximation of the Threshold Model on a Random Network. *Journal of Artificial Societies and Social Simulation*, 10(1)10. http://jasss.soc.surrey.ac.uk/10/1/10.html.

IMHOF L A, Fudenberg D, and Nowak M A (2005) Evolutionary cycles of cooperation and defection. *Proceedings of the National Academy of Sciences of the United States of America*, 102(31), pp. 10797-10800

IZQUIERDO L R, Izquierdo S S, Galán J M, and Santos J I (2009) Techniques to understand computer simulations: Markov chain analysis . *Journal of Artificial Societies and Social Simulation*, 12(1)6 http://jasss.soc.surrey.ac.uk/12/1/6.html.

IZQUIERDO L R, Galán J M, Santos, J I and Olmo R. (2008a). Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. *Empiria* 16, pp. 85-112

IZQUIERDO L R, Izquierdo S S, Gotts N M, and Polhill J G (2007) Transient and Asymptotic Dynamics of Reinforcement Learning in Games. *Games and Economic Behavior*, 61(2), pp. 259-276

IZQUIERDO L R and Polhill J G (2006) Is your model susceptible to floating point errors? *Journal of Artificial Societies and Social Simulation*, 9(4)4. http://jasss.soc.surrey.ac.uk/9/4/4.html.

IZQUIERDO S S, Izquierdo L R, and Gotts N M (2008b) Reinforcement learning dynamics in social dilemmas. *Journal of Artificial Societies and Social Simulation*, 11(2)1. http://jasss.soc.surrey.ac.uk/11/2/1.html.

IZQUIERDO S S and Izquierdo L R (2007) The Impact on Market Efficiency of Quality Uncertainty without Asymmetric Information. *Journal of Business Research*, 60(8), pp. 858-867

IZQUIERDO S S and Izquierdo L R (2006) On the Structural Robustness of Evolutionary Models of Cooperation. In Corchado E, Yin H, Botti V J, and Fyfe C (Eds.) *Intelligent Data Engineering and Automated Learning - IDEAL 2006. Lecture Notes in Computer Science 4224*: 172-182. Berlin Heidelberg: Springer

JANSSEN J and Manca R (2006) *Applied Semi-Markov Processes*. New York, NY: Springer.

KARR A F (1990) Markov Processes. In Heyman D P and Sobel M J (Eds.) *Stochastic Models. Handbooks in Operations Research and Management Science 2*: 95-123. Elsevier Science Publishers B.V. (North-Holland)

KLEMM, K, Eguíluz, V M, Toral, R and San Miguel, M (2003a) Nonequilibrium Transitions in Complex Networks: A Model of Social Interaction. *Physical Review E* 67(2), Article 026120.

KLEMM, K, Eguíluz, V M, Toral, R and San Miguel, M (2003b) Role of Dimensionality in Axelrod's Model for the Dissemination of Culture. *Physica A* 327, pp. 1-5.

KLEMM, K, Eguíluz, V M, Toral, R and San Miguel, M (2003c) Global Culture: A Noise-Induced Transition in Finite Systems. *Physical Review E* 67(4), Article 045101.

KLEMM, K, Eguíluz, V M, Toral, R and San Miguel, M (2005) Globalization, Polarization and Cultural Drift. *Journal of Economic Dynamics and Control* 29(1-2), pp. 321-334.

KULKARNI V G (1995) *Modelling and Analysis of Stochastic Systems*. Boca Raton, Florida: Chapman & Hall/CRC.

KULKARNI V G (1999) *Modeling, Analysis, Design, and Control of Stochastic Systems*. New York: Springer-Verlag.

KUSHNER H J and Yin G G (1997) *Stochastic Approximation Algorithms and Applications*. New York, NY: Springer-Verlag.

LEBARON B, Arthur W B, and Palmer R (1999) Time series properties of an artificial stock market. *Journal of Economic Dynamics & Control*, 23(9-10), pp. 1487-1516

LEOMBRUNI R and Richiardi M (2005) Why are economists sceptical about agent-based simulations? *Physica A*, 355, pp. 103-109

LIEBERMAN E, Havlin S, and Nowak M A (2009) Evolutionary dynamics on graphs. *Nature*, 433(7023), pp. 312-316

MABROUK N, Deffuant G, and Lobry C (2007) Confronting macro, meso and micro scale modelling of bacteria dynamics. *M2M 2007: Third International Model-to-Model Workshop, Marseille, France, 15-16 March 2007*. http://m2m2007.macaulay.ac.uk/M2M2007-Mabrouk.pdf.

MACY M W and Flache A (2002) Learning Dynamics in Social Dilemmas. *Proceedings of the National Academy of Sciences of the United States of America*, 99(3), pp. 7229-7236

MILLER J H and Page S E (2004) The standing ovation problem. *Complexity*, 9(5), pp. 8-16

NOWAK M A and May R M (1992) Evolutionary Games and Spatial Chaos. *Nature*, 359(6398), pp. 826-829

NOWAK M A and Sigmund K (1992) Tit for tat in heterogeneous populations. *Nature*, 355(6357), pp. 250-253

NOWAK M A and Sigmund K (1993) A strategy of win-stay, lose-shift that outperforms tit for tat in the Prisoner's Dilemma game. *Nature*, 364(6432), pp. 56-58

OSTROM T (1988) Computer simulation: the third symbol system. *Journal of Experimental Social Psychology*, 24(5), pp. 381-392

POLHILL J G, Izquierdo L R, and Gotts N M (2006) What every agent based modeller should know about floating point arithmetic. *Environmental Modelling & Software*, 21(3), pp. 283-309

RICHIARDI M, Leombruni R, Saam N J, and Sonnessa M (2006) A Common Protocol for Agent-Based Social Simulation. *Journal of Artificial Societies and Social Simulation*, 9(1)15. http://jasss.soc.surrey.ac.uk/9/1/15.html.

SANTOS F C, Pacheco J M, and Lenaerts T (2006) Evolutionary dynamics of social dilemmas in structured heterogeneous populations. *Proceedings of the National Academy of Sciences of the United States of America*, 103(9), pp. 3490-3494

SUBER P (2007) Formal Systems and Machines: An Isomorphism. *Hand-out for "Logical Systems"*: Earlham College

TAKADAMA K, Suematsu Y L, Sugimoto N, Nawa N E, and Shimohara K (2003) Cross-element validation in multiagent-based simulation: Switching learning mechanisms in agents. *Journal of Artificial Societies and Social Simulation*, 6(4)6. http://jasss.soc.surrey.ac.uk/6/4/6.html .

TAKAHASHI N (2000) The emergence of generalized exchange. *American Journal of Sociology*, 10(4), pp. 1105-1134

TRAULSEN A, Nowak M A, and Pacheco J M (2006) Stochastic dynamics of invasion and fixation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 74(1), pp. 011909

VILÀ X (2008) A Model-To-Model Analysis of Bertrand Competition. *Journal of Artificial Societies and Social Simulation*, 11(2)11. http://jasss.soc.surrey.ac.uk/11/2/11.html.

WIKIPEDIA (2007). Structured program theorem. http://en.wikipedia.org/w/index.php?title=Structured_program_theorem&oldid=112885072.

WILENSKY U (1999) *NetLogo*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University.