# Modelling and Simulation[1]

version 1.1

Peter Sloot

University of Amsterdam[2]

1 Based on lectures presented at the CERN school on computing, Sopron, Hongary August 1994. This text and the accomanying course material is available through WorldWideWeb: http://www.fwi.uva.nl/ in the home page of the research group Parallel Scientific Computing and Simulations under 'Publications.'

2 Dr. P.M.A. Sloot is an associate professor in parallel scientific computing and simulation at the University of Amsterdam and can be reached at: Kruislaan 403 1089 SJ Amsterdam, The Netherlands. Email: peterslo@fwi.uva.nl; Tel: +32 20 5257463; Fax: +31 20 5257490.

**Table of Contents:**

*'The real purpose of modelling and simulation is to discover that nature hasn't misled you into thinking you know something, you don't actually know'*

Adapted from R. Pirsig.

# I: Introduction[3]

## I.1 The computer experiment

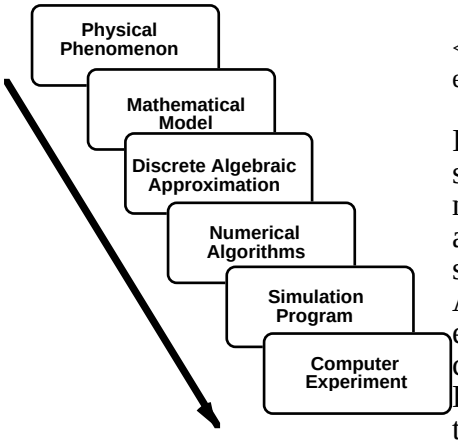
The use of computers in physics is becoming increasingly important. There is hardly any modern experiment thinkable where computers do not play an essential role one way or another. If we take a liberal approach to the concept of experimenting then we can make a fundamental distinction in the applicability of computers in physics. On the one hand computers are used in experimental set-ups such as a measuring/controlling/data analyses device, inevitable to the accurate measurement and data handling. On the other side we have the field where the computer is used to perform some *simulation* of a physical phenomenon. This is essentially the realm of *computational physics*. One of the crucial components of that research field is the correct abstraction of a physical phenomenon to a conceptual *model* and the translation into a computational model that can be validated. This leads us to the notion of a *computer experiment* where the model and the computer take the place of the 'classical' experimental set-up and where simulation replaces the experiment as such.
We can divide these type of computer experiments roughly into three categories [1]:

• Simulation of complex systems where many parameters need to be studied before construction can take place. Examples come from engineering: car-crash worthiness simulation, aircraft wing design, general complex safety systems.

• Simulations of phenomena with extreme temporal or spatial scales, that can not be studied in a laboratory. For instance: Astrophysical systems, mesoscopic systems, molecular design etc.

• 'Theoretical experiments' where the behaviour of isolated sub domains of complex systems is investigated to guide the theoretical physicist through the development of new theories. Typical research fields are: non-linear systems, plasma and quantum physics, light scattering from complex particles etc.

A classical approach to the different levels of abstractions required to build a reliable computer experiment is shown in Figure 1:




<- Figure 1: Functional stages in the development of a computer experiment.

In the sequel we will see that there are ways to shortcut this sequence of levels by the use of computational models that mimic the physical system and that can be mapped directly on a computer system. An example of such a short-cut is the simulation of the dynamics of a fluid flow via Cellular Automata, rather than via the formulation of the Navier-Stokes equations and the subsequent discretisation. Other examples can be found in the fields of Monte Carlo and Molecular Dynamics simulation, where the phase space of a thermodynamic system is explored via simulation. In some cases this implies that we can view the physical processes as

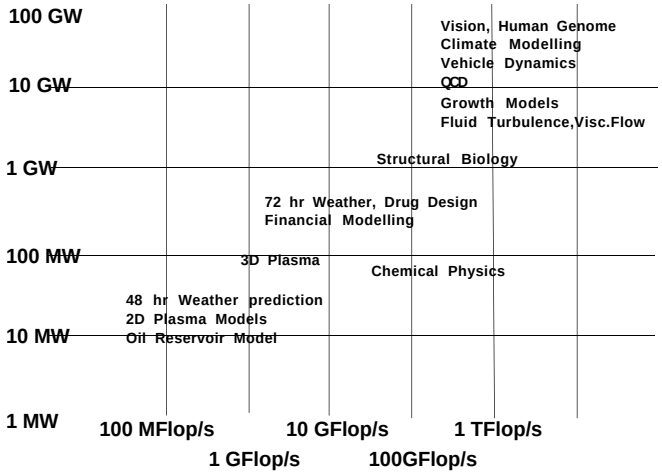
3 Parts of this text, and the accompanying hands-on training, have been used by the author in courses on (parallel) scientific computing, simulation and computational physics at the University of Amsterdam in the physics and computer science departments.

computations themselves (see for instance [2]).
The type of complex or chaotic dynamical systems we want to study are known or at least expected to be computationally irreducible, therefore their behaviour can only be studied by explicit simulation. The potential and the underlying difficulties of this field are generally recognised. We identify three major phases in the process of the development of a computer experiment, each with its own challenges:

• *The modelling phase*. The first step to simulation is the development of an abstract model of the physical system under study. Strange enough there are hardly any formal methods of modelling which support completeness, correctness and efficiency. The lack of such methods might harm the success of this research field significantly. There are complete journals and internet sites (see references) dedicated to the compilation of the consequences of using insufficient, incorrect and inefficient models. For instance recently some flight simulation experiments where reported where the underlying simulator (an explicit Finite Element solver) worked perfectly, but where deficiencies in the CAD/CAM model resulted in catastrophic misinterpretation of the simulations [3]. Some promising new approaches in this field are the use of 'structural interactive modelling methods' such as present in Stella ™ [4] and the concept of BondGraph modelling used in control systems design [5, 6] or object oriented modelling [7].

• *The simulation phase*. Solvers form the kernel of simulation systems. Here we refer to (mathematical) methods that make the underlying physical models discrete. A rough distinction can be made between solvers for Discrete Event systems and solvers for Continues systems. In the next paragraphs we will discuss these in more detail. The solvers that take a discrete event approach, rather than a continuous time approach, are relatively new in computational physics, but gain more and more impact. We will give some examples later on. The more conventional solvers are Finite Difference, Finite Element/Volume and a large class of linear algebra solvers.
Of special interest are the new results with 'particle methods'. Especially the recent results obtained with hierarchical particle methods and layered multipole expansion [8, 9]]. In these hierarchical solvers the computational complexity can be reduced dramatically, this is partly due to the efficient mapping of the conceptual model onto the computer specific model (see also figure 3).

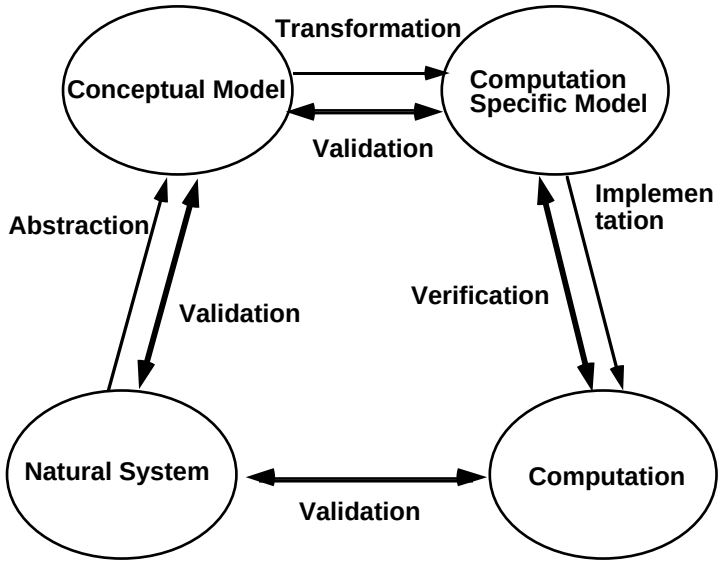
• *The computational phase*. In this phase we concentrate on the mapping of the different solvers to the machine architecture. Since the type of problems we are interested in are computationally very demanding, lots of research effort in the efficient use of modern architecture's for simulation is going on in this field. As an example Figure 2 shows the memory requirements and computational performance needed to investigate a number of challenging physical phenomena through simulation [10, 11, 12].




<- Figure 2: Examples of challenging computational physical research topics and their computational requirements.

We observe that the computational complexity of these problems is enormous. It is for this reason that since 1991 world-wide initiatives (the so-called High Performance Computing and Networking initiatives) are taken to push the limits of hard- and software to the extreme where, at least technologically speaking, these systems can be studied. Apart from these technological obstacles there are other, and maybe even more fundamental, questions that need to be addressed first. If for instance we can circumvent parts of the mathematical complexity by the use of models that directly map the physical system on the architecture, like in the cellular automata example, we might show up with more realistic computational requirements. Still we expect that -due to the irreducibility of some of the problems- high performance computer systems will remain extremely important in the field of computational physics and simulation.

In figure 3 we summarise the various conceptual steps required to map a natural system onto a computer [13]:




<-Figure 3: Summary of the various conceptual stages in the development of a computer experiment

Where the computation specific model consists of representing the derived conceptual model into a *language* that can be implemented. One of the things that often goes wrong in these stages is that the scientist tends to confuse his code with a conceptual model or -even worse- with the natural system itself, the researcher has falling in love with his model! This is a situation we should be prepared for and try to avoid at all cost. One way to maintain a critical attitude is to carefully design and test sub stages in the modelling and simulation cycle and to add primitives to the simulation that constantly monitor for 'unrealistic' results

I.2 A closer look at system models

In the next paragraphs we will use some concepts from the fields of (discrete event) computer simulations, the terminology used is briefly explained in the following intermezzo:

**Intermezzo: Simulation Lingo**

A summary of typical definitions in modelling and simulation used throughout this text.

• A *model* describes a *system* containing *components*

• *Components* can be 'lifeless' [Items, Objects] with characteristic *attributes* that cannot change the state of the *system*

• *Components* can be 'alive' [Entities, Subjects] with characteristic *attributes* that can change the state of the *system*

• The State of the system is built from:
Contents(t, elements)
Structure(relations, elements)
Attributes(elements)

• An *Event* describes (a number of) *changes* in the state of the system at a given time 't'

• A *change* is denoted by an *'action'*

• *Actions* can transform the *attributes* not changing the number of elements

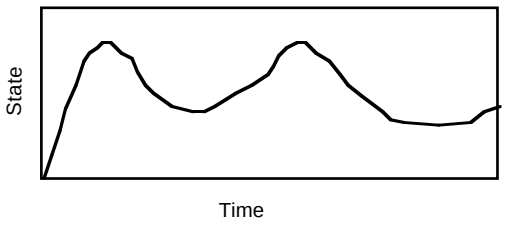
• *Actions* can split or merge the elements thus changing the number of elements

• A *process* is a sequence of state changes in a given time

• A *system* can be seen as a combination of *processes* and *interactions* (relations)

Although many attempts have been made throughout the years to categorise systems and models, no consensus has been arrived at. For the purpose of this text we limit ourselves to models sometimes referred to as 'Symbolic Models' where the attributes of the system are described by mathematical symbols and relations. It is convenient to make the following distinction between the different Symbolic Models [14]:

• Continuous-Time models: Here the state of a system changes continuously over time:




<- Figure 4: Trajectory of continuous-time model

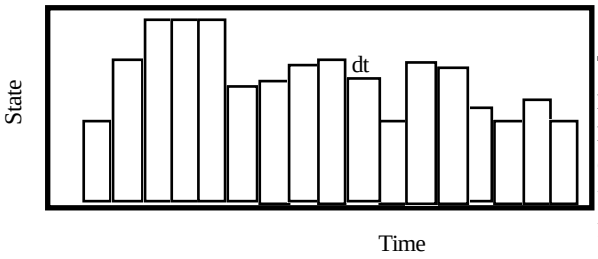
These type of models are usually represented by sets of differential equations. A further subdivision would be:

- Lumped Parameter models expressed in Ordinary Differential Equations (ODE's) and
- Distributed Parameter models expressed in Partial Differential Equations (PDE's):

$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{t})$ or $\dot{\overline{\mathbf{x}}} = \overline{\overline{\mathbf{A}}}\,\overline{\mathbf{x}} + \overline{\overline{\mathbf{B}}}\,\overline{\mathbf{u}}$

and $\frac{\partial \mathbf{u}}{\partial \mathbf{t}} = \sigma . \frac{\partial^2 \mathbf{u}}{\partial \mathbf{x}^2}$ respectively.

• Discrete-Time models: here the time axis is discretised.




<- Figure 5: Trajectory of discrete-time model

The state of a system changes are commonly represented by difference equations. These type of models are typical to engineering systems and computer-controlled systems. They can also arise from discretised versions of continuous-time models for example:

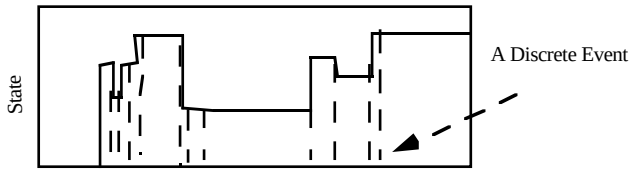
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{t})$$

$$\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k,\mathbf{t}_k) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta \mathbf{t}}$$

**or**

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + \Delta \mathbf{t}.\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k,\mathbf{t}_k)$$

• Discrete-Event models: Here the state is discretised and 'jumps' in time. Events can happen any time but happen only every now and then at (stochastic) time intervals. Typical examples come from 'event tracing' experiments, queuing models, Ising spin simulations, Image restoration, combat simulation, etc.




<- Figure 6: Trajectory of discrete-event model

In this paper we will concentrate on general methodologies applicable to a large variety of problems stemming from natural sciences.
We have organised this paper as follows. In Chapter II probabilistic models are introduced, these models provide us with a rigorous basis on which we cab build all type of simulation studies. Chapter III describes the use of Cellular Automata as an important modelling tool in physical simulations. Once a model is designed we need to look into the simulation method to probe the model. Concepts needed to implement a simulation method are introduced in chapter IV. Here we emphasise the (implementation) differences in event-driven and time-driven simulation. The last chapter deals to some extend with growth processes, a specific class problems that make use of the modelling and simulation strategies introduced in the preceding chapters.
Rather than presenting a complete formal overview of the underlying methodologies we will take a case-study approach throughout this text. I hope that the reader will enjoy working with this introduction to the modelling and simulation field as much as I have enjoyed writing it down!

*The fact that people subscribe to the Bayes-doctrine proves that miracles do happen...*
*this seems to be a contradiction.*

Sloot's free interpretation of Hogben's 'mathematics in the making'

## II: Stochastic Models

### II.1 Background

Describing phenomena that are difficult to cast into an analytical expression or that are computational intractable, inevitably require simulation. In this section we introduce some methods to formulate stochastic models and to analyse the transient behaviour of these models (as opposed to the steady-state or asymptotic behaviour that can be studied with simplifying analytical metamodels) [15].

The complete process of stochastic modelling and consequent simulation can summarised in the following design structure:

*Step 1*: Gather knowledge to decide what probabilistic distributions reflect the behaviour of the phenomenon and decide whether the Random Variables (RV) to be used are assumed to be independent.

*Step 2*: Generate Random Numbers (RN).

*Step 3*: Use the RN to create discrete and continuous distributions.

*Step 4*: Create a 'probability model' and track the system over continuous time by simulating the discrete events.

*Step 5*: Perform extensive I/O analyses and validation of the model

*Step 6*: Find indicators of the efficiency through variance estimators and obtain new estimators with reduced variances.

Step 1 to 3 rely heavily on parts of mathematical statistics theory (see for instance [16]).

Step 4 is the so called discrete event approach. Discussion on how to implement this method is still going on. An approach is to identify 3 regular methods of simulation based on the concept of locality [17]:

- Activity scanning: Exploiting the locality of state. For each event the induced activities are denoted in the simulation.

- Event scheduling: Exploits the locality of time: for each event an exact description of the state changes is recorded and subsequent events are scheduled by each event.

- Process interaction: Locality of Object. Here we focus on the interactions between elements (or process routines) in a model. Processes are handled semi parallel (co-routines).

Typical examples of problems where stochastic models are being used are:

• Queuing and Optimisation:
- Computer Aided Manufacturing and Logistic Simulation (design of a production process)
- Architecture and VLSI design

- Distributed Interactive Systems e.g. Person in loop models used for training (flight and combat simulators)

• Monte Carlo:
- Mutidimensional Integration
- Brownian motion and Random walk phenomena
- Ising spin problems
- Solving of huge linear systems

• Markov Chain:
- Compiler Optimisation
- Travelling Salesman type of problems
- Image restoration [18, 19]

In the next paragraphs we will apply the concepts introduced so far and take a look into one of the most important theorems in simulating probabilistic models.

## II.2 A Bayesian approach to simulation of a stochastic model:

Let P(x) be the probability that event 'x' has occurred, and P(y|x) the (conditional) probability that 'y' has occurred given the fact that event x has taken place. Then with the joint probability P(xy) (the probability that both occur) we have:

$$\mathbf{P}(\mathbf{x}|\mathbf{y}) = \frac{\mathbf{P}(\mathbf{xy})}{\mathbf{P}(\mathbf{y})}$$

If the occurrence of event 'x' is independent of the occurrence of event 'y' (and the other way around) then P(x|y) = P(x). Therefore the previous equation transforms into:

$$\mathbf{P}(\mathbf{x}|\mathbf{y}) = \frac{\mathbf{P}(\mathbf{y}|\mathbf{x})\mathbf{P}(\mathbf{x})}{\mathbf{P}(\mathbf{y})}$$

This is the well known Bayes theorem (reference [20] is worthwhile reading!). A simple formula that has had a enormous impact on a large variety of sciences. Interpreting this theorem in terms of simulation we see that if the user is able to express his beliefs about the values of the parameters under investigation in the form of a prior distribution function P(x), then experimental information based on the likelihood function P(y|x) (outcome of the simulation) can be used to convert these initial beliefs into a posterior distribution P(x|y). In other words given some idea's about the model we want to study, we can generate via simulation information that can be used to update our initial guesses. This can for instance be applied to complicated markov chain simulations, fuzzy logic models and Image recovery [18, 21].

To get a feeling of the use(fullness) of this theorem consider the following head-or-tail experiment. N-times a coin is tossed with as result d-times head. What is the probability P(x), that the result of a toss gives a head?
Without a priori knowledge, P(x) ≈ d/N. If however preceding head-or-tail experiments were carried out, we could give a probability distribution for the random variable X:

$$\mathbf{p}(\mathbf{x}) = \mathbf{P}\{\mathbf{X} = \mathbf{x}\}$$

With Bayes theorem the best estimate for P(x) is such that:

$$\mathbf{P}(\mathbf{x}|\mathbf{d}) = \frac{\mathbf{P}(\mathbf{d}|\mathbf{x})\mathbf{P}(\mathbf{x})}{\mathbf{P}(\mathbf{d})} = \max.$$

Where the formula maximises the possibility that hypothesis X is the correct information given experimental data 'd'. This is clearly an experimentalist's interpretation of this theorem. In the sequel we will apply this notion of a experiment in different examples. In experiments we use:

$\mathbf{P}(\mathbf{x}|\mathbf{d}) \sim \mathbf{P}(\mathbf{d}|\mathbf{x})\mathbf{P}(\mathbf{x})$ since P(d) can be considered constant.

Consequently a simulation experiment consists of the following components:
- P(x): a_priori knowledge (hypothesis before experiment);
- P(d|x): experimental model (how data appears given X; the likelihood function);
- P(x|d): a_posteriori knowledge (= a priori knowledge + knowledge from experiment).

The p(x) and p(d|x) are established by 'common sense' or by means of 'educated guesses.'

Let's investigate these concepts by the following example. Suppose that:
- P(x): normal distribution with $\mu = 0.2$;
- P(d|x): N trials with d/N = 0.8.

We are interested in the influence of N and $\sigma$ on the a_posteriori distribution.
The a_priori knowledge P(x) has a normally distributed; notation $N(\mu, \sigma^2)$. In formula:

$$\mathbf{P}(\mathbf{x}) = \frac{1}{\text{Norm}} \frac{1}{\sigma\sqrt{2\pi}} \mathbf{e}^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu}{\sigma}\right)^2}$$

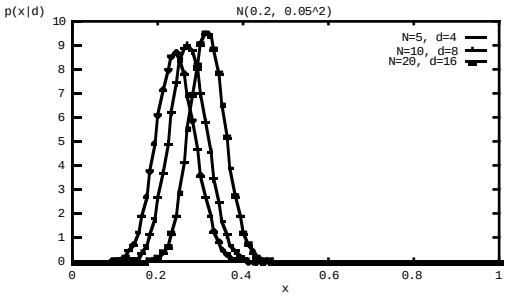
$$\text{Norm} = \int_0^1 \mathbf{P}(\mathbf{x})\,\mathbf{dx}$$

The experimental model P(d|x) is the probability distribution that the hypothesis about X is correct, given the data 'd' from the N trials. P(d|x) has a binomial probability distribution, notation $\text{bin}(\mathbf{N},\mathbf{x})$. In formula:

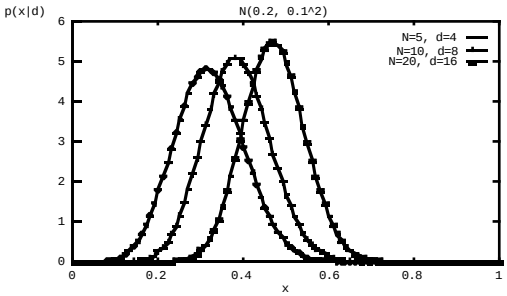
$$\mathbf{P}(\mathbf{d}|\mathbf{x}) = \binom{\mathbf{N}}{\mathbf{d}} \mathbf{x}^{\mathbf{d}} (1-\mathbf{x})^{\mathbf{N}-\mathbf{d}}$$

Remark: The binomial is not calculated as function of the number of successes, d, but rather as function of success per trial P(X = x).

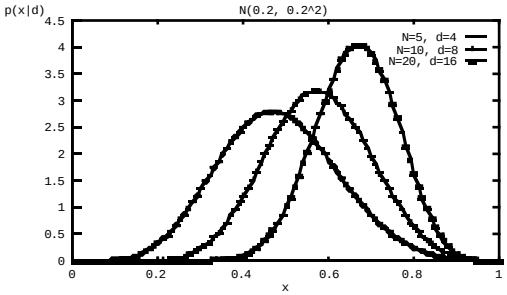
The next figures show the results of some of these experiments for different values of the parameters that model the system:




Experiment 1: mu = 0.2, sigma = 0.05




Experiment 2: mu = 0.2, sigma = 0.1




<- Experiment 3: mu = 0.2, sigma = 0.2

From these experiments we observe that if the prior information is accurate ($\sigma$ is small), the estimate for x is dominated by the value of $\mu$ (prior knowledge). We see that when the data set is large (N -> $\infty$) the prior information gets 'washed out'. Furthermore we observe that if no prior information is present, the estimate for x is data-driven. This should give us some idea's as to how to design and evaluate simulation experiments. The problem of course is that Bayes theorem does not prescribe how to find P(x). This information must be abstracted from the physical model and the

system knowledge we have. This inspires us to use the predictive capacity of Bayes theorem wherever possible. Especially in computational intractable problems where tedious simulation is the only way out.

## II.3 Monte Carlo Methods

In this paragraph one of the most interesting probabilistic simulation methods called 'Monte Carlo' (MC), after the famous casino in Mexico, is introduced. It can be applied to study statistical physics problems, crystallisation processes, phase transitions in systems with large degrees of freedom, evaluation of high dimensional integrals, solution to large linear systems, the size of cosmic showers, percolation of liquid through a solid etc. [22, 15].
The method is best explained by an example of its use. Suppose that we want to evaluate the integral:

$I = \int_0^1 f(x)dx$ or equivalently the associated quadrature formula $I = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ where f is evaluated for randomly chosen $\{x_i\}$ uniformly distributed over [0,1]. From the Central Limit theorem [16]we can estimate the uncertainty associated with this quadrature formula, for large N, to be:

$$\sigma_I^2 \approx \frac{1}{N}\sigma_f^2 = \frac{1}{N}\left[\frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} f_i\right)^2\right]$$ with $\sigma_f^2$ the variance in observed values f. As a consequence we see that the uncertainty in the estimate of the integral decreases as $\frac{1}{\sqrt{N}}$ and that the precision increases for smaller $\sigma_f^2$, implying a smooth function f. These observations are independent of the dimensionally of the problem, whereas for conventional numerical methods, such as Simpson's rule or the trapezoidal rule, the error increases with the dimensionallity. This independence of dimensionally is one of the outstanding characteristics of MC, explaining its wide-spread use.

Assume we want to evaluate $\int_0^1 e^{-x} dx$, than we simply evaluate for n-trials n times Exp[-X] with X random and uniform distributed over [0,1], then for n=N we can evaluate $F_N$ with the equations given above. Typical results are:
Fn = 0.62906, s = 0.18002 and error = -0.00306 for n = 3600
Fn = 0.63305, s= 0.18201, end error = 0.00092 for n = 7200.
From such experiments (of which more are in the hands-on documents) we observe that for n > 3000 the variance in the output remains unchanged, whereas the error in the estimation (here: |simulated - exact|) might still reduce for larger n. This is an experimental indication of a well known theoretical result that states that the error in the estimation resembles $\sigma_n \sim \sigma/\mathrm{sqrt}(n)$.

Consequently we note that we can reduce the error in MC integration by
- increasing the number of trials or
- using more efficient trials (i.e. more sampling in the neighbourhood of fast fluctuating values of the function f(x)).

This brings us to the notion of *Variance Reduction,* a very important general principle in Computer Simulations.

A typical example expressing the need of variance reduction techniques is shown in the next Intermezzo:

**Intermezzo: Need for Variance Reduction**

Consider a stochastic simulation where the observable $X_i$ is determined (with all X's independent and random). We model, for the sake of simplicity, the probabilistic behaviour by:

$$\mathbf{P}\{\mathbf{X_i} = 1\} = 1 - \mathbf{P}\{\mathbf{X_i} = 0\} = \mathbf{p}.$$

Now in order to allow for a reliable simulation we want to construct a 95% confidence interval of 'p' with a length of say: 2 x $10^{-6}$, or equivalently:

$$\overline{\mathbf{X}} \pm Z_{0.5/2} \times \frac{\mathbf{S}}{\sqrt{\mathbf{n}}}$$

Then from standard statistics we know that:

length of confidence interval $p = 2 \times Z_{0.5/2} \times \frac{\mathbf{S}}{\sqrt{\mathbf{n}}} = 2 \times 10^{-6}$

With unit normal $Z_{0.5/2} = 1.96$ and $\mathbf{S} = \sqrt{\mathbf{Var}\left(\mathbf{X_i}\right)} = \mathbf{p}(1-\mathbf{p})$ for the used random variables $X_i$, we find that

$$\sqrt{\mathbf{n}} = \frac{1.96 \times \sqrt{\mathbf{p}((1-\mathbf{p}))}}{2 \times 10^{-6}}$$

Therefore n = 3.8 $10^6$, that is more than a million simulation runs to obtain the required accuracy!

The most relevant Variance reduction Techniques are :

- The use of Control Variates
- Conditioning
- Stratified Sampling
- The use of Antithetic Variables
- Importance Sampling

We will briefly discuss Antithetic Variables and Importance Sampling techniques (for a more extensive treatment the reader is referred to for instance [16]).
In a typical simulation study we are interested in determining a parameter θ, connected to some stochastic model, by averaging the 'n' simulation runs with observed values $\{X_i\}$ with i = 1, n. The related Mean Square Error (MSE) is given by:

$$\mathbf{MSE} = \mathbf{E}\left[\left(\overline{\mathbf{X}} - \theta\right)^2\right] = \mathbf{Var}\left(\overline{\mathbf{X}}\right) = \mathbf{Var}\ \frac{(\mathbf{X})}{\mathbf{n}}$$

So if we can find a different (unbiased) estimate of θ with a smaller variance than $\overline{\mathbf{X}}$, then we have improved the estimator and made our simulation more efficient since we need less simulation runs to come to the same level of accuracy.
If possible the use of antithetic variables is very helpful here. Suppose we use simulation to obtain θ = E[X], than by generating $X_1$ and $X_2$, identically distributed variables having mean θ we obtain:

$\mathbf{Var}\left(\frac{\mathbf{X}_1 + \mathbf{X}_2}{2}\right) = \frac{1}{4}\left[\mathbf{Var}\left(\mathbf{X}_1\right) + \mathbf{Var}\left(\mathbf{X}_2\right) + 2\mathbf{Cov}\left(\mathbf{X}_1, \mathbf{X}_2\right)\right]$. Therefore if $X_1$ and $X_2$ are negatively correlated we get a reduction in the variance of θ (see also hands-on exercises). Moreover for all X's uniform random over [0,1], we only need to generate one X and calculate the other X of the pair from 1-X, this reduces the number of time-costly random number generations by a factor of 2.

Another variance reduction technique, often applied in Monte Carlo related simulations, is a method called Importance Sampling [23]:
We start by introducing a positive weight function $\mathbf{p(x)}$ such that

$$\int_a^b \mathbf{p(x)dx} = 1,$$

with $\mathbf{F} = \int_a^b \mathbf{f(x)dx}$ we get => $\int_a^b \frac{\mathbf{f(x)}}{\mathbf{p(x)}}\mathbf{p(x)dx}$ .

Next we evaluate the integral by 'sampling according to probability density function $\mathbf{p(x)}$' and constructing

$$\mathbf{F_n} = \frac{1}{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{n}} \frac{\mathbf{f(X_i)}}{\mathbf{p(X_i)}},$$

Choose p(x) such that the variance of the integrand f(x)/p(x) is minimised. For instance if p(x) mimics f(x) than the integrand will vary slowly and therefor $\sigma$ will be small (we use a a-posteriori estimator of $\sigma$).

In the next example we use antithetic variables together with importance sampling:

$$\mathbf{F} = \int_0^1 \mathbf{e}^{-\mathbf{x}^2}\mathbf{dx}$$

(with exact solution $\frac{\sqrt{\pi}}{2}\mathbf{Erf}[1] = 0.746824$)

| | $\mathbf{p(x)} = 1$(Uniform) | $\mathbf{p(x)} = \mathbf{Ae^{-x}}$(non Uniform) |
|---|---|---|
| n(trials) | 20000 | 1000 |
| Fn | 0.7452 | 0.7582 |
| s | 0.2010 | 0.0548 |
| s/sqrt(n) | 0.0016 | 0.0017 |
| CPU/trial (arbitrarily units) | 1 | 3 |
| Total CPU (arbitrarily units) | 20000 | 3000 |

From this simple experiment we see that in order to get the same accuracy we need approximately 7 times less computation time (for the specific computer system used)!

It is however often difficult or impossible to generalise this approach to sample complicated multidimensional weight functions, we need a different approach. A solution to this problem was given by Metropolis et al. [24, 22].

## II.4 Metropolis Monte Carlo

We used Importance Sampling to generate random variables according to a specific distribution (mimicking the integral we wanted to evaluate). Often it is difficult, or even impossible to generate variables with an arbitrary distribution. A solution to this problem was given by Metropolis [24]. With the Metropolis algorithm we generate a 'random walk' of points $\{X_i\}$ through a phase-space such that the asymptotic probability approaches the required p(x) [25].

We start by defining the 'random walk' with a transition probability $\Gamma_{ij}$ being the probability to get from $X_i$->$X_j$ such that the distribution of the points $\{X_i\}$ converges to p(x).
A sufficient condition is the so-called detailed balance condition:

$\mathbf{p}(\mathbf{X_i})\Gamma_{ij} = \mathbf{p}(\mathbf{X_j})\Gamma_{ji}$ . For instance: $\Gamma_{ij} = \min\left[1, \frac{\mathbf{p}(\mathbf{X_j})}{\mathbf{p}(\mathbf{X_i})}\right]$ so that $X_{n+1}$ can be generated from $X_n$ by the following pseudo code:

- Choose trial position $Xt = X_n + \delta_n$ with $\delta_n$ random over $[-\delta,\delta]$
- Calculate $\Gamma_{ij}$ , here $p(X_t)/P(X_n)$
- If $\Gamma_{ij} \geq 1$ : accept move and put $X_{n+1} = X_t$
- If $\Gamma_{ij} < 1$ : generate random number R
- If $R \leq \Gamma_{ij}$ : accept move, *else* put $X_{n+1} = X_n$

It is efficient to take $\delta$ such that roughly 50% if the trials are accepted and to start the walk with a value of X at which p(X) has a maximum.
It can be shown that this method guarantees that, for a large number of steps, all states are explored and an equilibrium will be found according to $\Gamma_{ij}$.

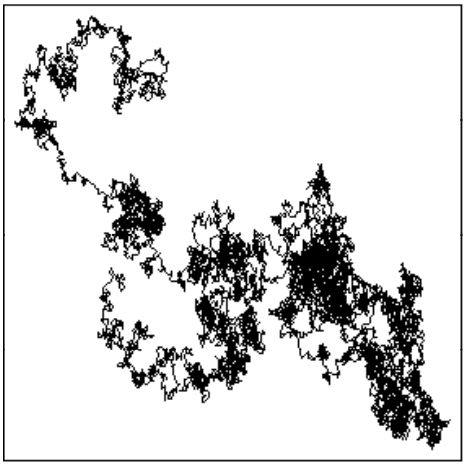
The next piece of Mathematica™ code generates a random walk based on the Metropolis algorithm, with a probability distribution $\mathbf{Exp}\left[-0.5\left(\mathbf{x}^2 + \mathbf{y}^2\right)\right]$ (a 2-dimensional normal distribution with mean {x,y}=0.0 and $\sigma_{xy} = \sigma_{yx} = 0$, and $\sigma_{xx}=\sigma_{yy}=1$):
(For an introduction to Mathematica™ see[ 26]).

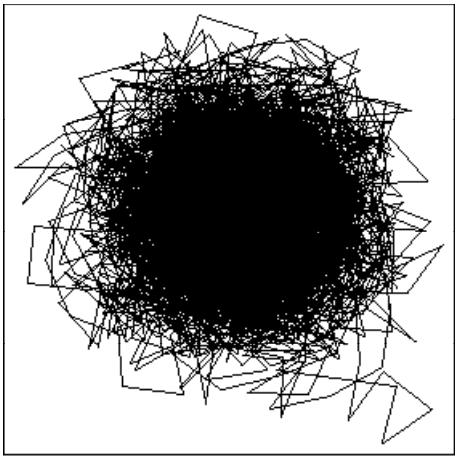
```
ProbDist[point_] := Exp[-0.5 * (point[[1]] * point[[1]]
                        + point[[2]] * point[[2]])]

MetropolisStep[oldpoint_] :=
     Module[{newpoint,Prob},
          newpoint=oldpoint+With[{dir = Random[Real, range]},
                    {Cos[dir], Sin[dir]}];
          Prob = ProbDist[newpoint] / ProbDist[oldpoint];
          If[Prob>=1,Return[newpoint],
               If[Prob>Random[Real,{0,1}],Return[newpoint],
               Return[oldpoint]
          ]
     ]
MetropolisWalk[n_Integer] :=
     Module[{points},
          points = NestList[ MetropolisStep[#]&, {0.0, 0.0}, n];
          Show[ Graphics[{Point[{0,0}], Line[points]}],
               AspectRatio->1.0, Frame->True,
               FrameTicks->None]
          ]
```

*Mathematica ™ code for generating a Random Walk:*

The result of the execution of this code for 10000 Metropolis steps with p(x,y) = 1 and p(x,y) $=\mathbf{Exp}\left[-0.5\left(\mathbf{x}^2 + \mathbf{y}^2\right)\right]$ respectively, is shown in the next figures:





MetropolisWalk[10000], with p(x,y) = 1 and with p(x,y) = $\mathbf{Exp}\left[-0.5\left(\mathbf{x}^2+\mathbf{y}^2\right)\right]$

From these figures we observe that for p(x,y)=1 we obtain a more or less random distribution, whereas the Z-normal distribution shows a normal distributed walk around the mean value (0,0). Closer inspection for different number of Metropolis steps nicely show the dependency of the density of the walk relative to the values of the co-variance matrix used (data not shown). The underlying simulation mechanism used here is the so-called Markov Chain simulation. Here a set of numbers Pij, i,j=1,...,N, is chosen such that whenever the process is in state i then, independent of the past states, the probability that the next state is j is Pij. Then the collection {Xn, n>=0} constitutes a Markov chain having transition probabilities Pij.
It is known that such a simulation strategy always produces a stationary distribution if the simulation is irreducible[4] and aperiodic[5] [16, 21] .
We will use this Markov Chain simulation method extensively in the next chapters.

## II.5 Ising Spin Simulations [22, 23, 25, 27]

The 'Ising' system serves as one of the simplest models of interacting bodies in statistical physics. The model has been used to study ferromagnets, anti-ferromagnetism, phase separation in binary alloys, spin glasses and neural networks. It has been suggested [28] that the Ising model might be relevant to imitative behaviour in general, including such disparate systems as flying birds, swimming fish, flashing fireflies, beating heart cells and spreading diseases.

Essentially, the (2-D) model is comprised of an 'n by n' square lattice in which each lattice site has associated with it a value of 1 (up spin) or -1 (down spin). Spins on adjacent, nearest-neighbour sites interact in a pair-wise manner with a strength J (the exchange energy). When J is positive, the energy is lower when spins are in the same direction and when J is negative, the energy is lower when spins are in opposite directions. There may also be an external field of strength H (the magnetic field). The magnetisation of the system is the difference between the number of up and down spins on the lattice. The Energy of the system is given by:
$\mathbf{E} = -\mathbf{J}\sum_{<ij>}\mathbf{s_i}\mathbf{s_j} - \mu_0\mathbf{H}\sum_{\mathbf{i}}\mathbf{s_i}$, with first sum over all pairs of spins which are nearest neighbours and the second term the interaction energy of the magnetic moment with the external magnetic field.
In spin flipping, lattice sites are selected and either flipped or not, based on the energy change in the system that would result from the flip. The simulation is essentially a Markov Chain simulation.
In the (probabilistic) Ising model we assume a constant temperature condition (the canonical ensemble formulation). A lattice site is selected at random and a decision is made on whether or not to flip the site using the Metropolis Algorithm. Again the Metroplis method allows the system to reach a 'global' energy minimum rather than getting stuck in a 'local' minimum.

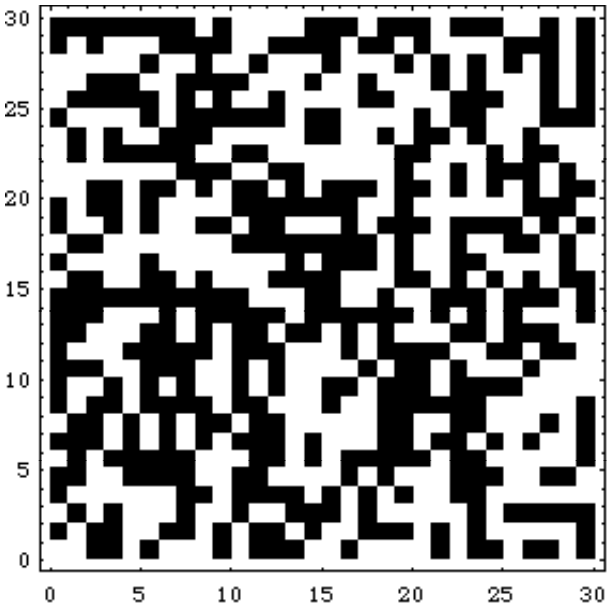
4 A Markov Chain is said to be irreducible if and only if for all pairs of system states (i,j) there is a positive probability of reaching j from i in a finite number of transitions.

5 A Markov Chain is said to be aperiodic if and only if for all system states i, the greatest common divisor of all integers n>=1, such that (Pn)ii>0, is equal to 1.

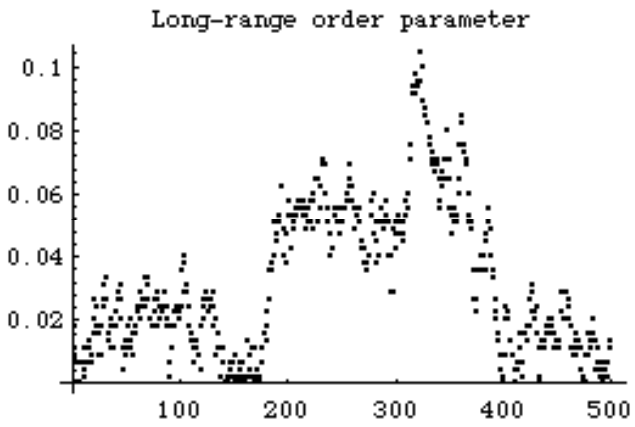
*Example of Ising spin system:*

In the next two simulations we study the natural phenomenon of ferromagnetism as an example [27]. Non-zero magnetism only occurs if the temperature is lower than a well-defined temperature known as the Curie critical temperature $T_C$. For $T > T_C$ the magnetisation vanishes. $T_C$ separates the disordered phase of the spin system for $T > T_C$ for the ordered ferromagnetic phase for $T < T_C$.

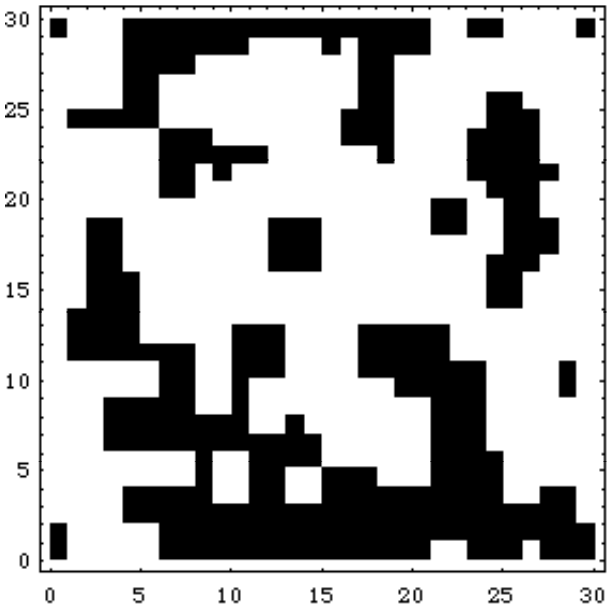
Explore the system for H=0 and $T>T_C$ or $T<T_C$.



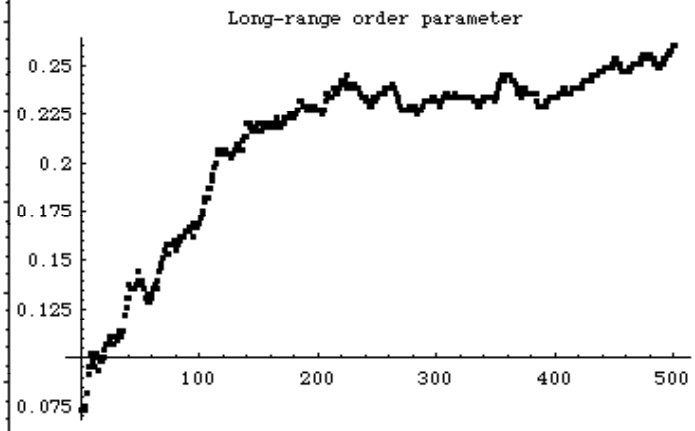

<- H=0; T>>Tc, final configuration on a 30x30 lattice.




Order in system versus number of iterations




<- H=0; T<<Tc, final configuration on a 30x30 lattice.




Order in system versus number of iterations

In this example we have performed more than 100000 Monte Carlo steps for each cell. To simulate a real system we would of course need $10^{23}$ cells instead of the $30^2$. Fortunately the computer experiments given here already give us some insight in the basic physics of such spin systems. Careful experiments with detailed statistical analyses can indeed reveal very interesting physical phenomena from this simple probabilistic model. Especially the study of phenomena like critical slowing and phase transitions in classical fluids are fields of active research. Using Bayesian methods we can reduce the amount of iterations required. An example will be given in the next session.

## II.6 Special Case: Simulated Annealing

Simulated Annealing is an optimisation algorithm that can be used for a wide range of applications. It is a stochastic algorithm that can find a global optimum out of a large number of candidate solutions. The process consists of first 'melting' the system, and then gradually lowering the temperature. At each temperature the simulation lasts long enough for the system to reach a steady state. The interesting part of this method, compared to deterministic optimisation schemes, is that it is able to detrap from local minima. The procedure is the Metropolis algorithm with an additional temperature as control parameter.

The general algorithm can be described as follows :

- The system to be optimised is stated at a high 'temperature' i.e. most of the solutions can be constructed (all system states can be reached).
- Then there are two possible constructions :

-*Homogeneous* Markov chains : A Markov chain is started at a temperature T. If the chain is 'finished' the temperature is decreased and a new chain is started.
-*Inhomogeneous* Markov chains : After each MC step a Temperature decrement takes place.

- After some simulation time the temperature is so low that only steps 'downhill' are accepted. If a predefined stop criteria is reached the simulation is ended.
- If one step results in a lower energy of the system we accept the change, if not then we accept with a probability $P=\exp(-\Delta E/kT)$.

*Homogeneous Markov Chains*

It is required that at each temperature a stationary distribution is reached. One way to guarantee this is by building a Markov Chain (see previous sections). In principle the Markov chains should be infinite long, only then equilibrium is achieved. In practice we go for a Heuristic approach where we trade the length of the chain 'L' against the cool-rate 'cr' [29]. We start at a high T, thus allowing for the exploration of the complete phase space, and slowly move to lower temperatures. For Homogeneous Markov chains the cooling procedure is usually: $T_k = cr * T_{k+1}$ known as simulated quenching [30]. In other well studied schemes we take information of the previous chains into account, this is in essence the Bayesian approach:

$$T_k = \frac{T_k}{1 + \frac{T_k * \ln(1+\delta)}{3 * \sigma_k}}$$

Here $\delta$ expresses the distance between the equilibrium distributions. If $\delta$ is small than nothing much will happen for distributions of subsequent chains. Large $\delta$ implies that the temperature differences will be large. $\sigma_k$ is a measure of the standard deviation in the cost function of chain 'k'.

*Inhomogeneous Markov Chains*

Now we allow for the temperature to change with each step. As a consequence we are not sure anymore whether we can explore the complete phase space: Ergodicity is said to be weak. A prove of the validity of this simulation method with a cooling schedule of $T(k) = \frac{T_0}{\ln(k)}$ can be found in [30, 29].

Different schedules and the consequences for parallel and vector implementations are discussed in [31, 32]

*The Bayesian cooling schedule for Simulated Annealing* [21]

We discussed two cooling schedules for the simulation with a homogeneous Markov chain. A third method can be used when incorporating the notion of a_priori information. In the Bayesian approach we investigate at each chain, using the information of the previous steps, whether to reduce the temperature or not (yet).

- First by applying Bayes theorem we estimate the stationary distribution of the next chain, based on the experimental results of the previous chains.
- Next we use this information to estimate the average value of the cost function.
- If a large $\delta$ is chosen than we also use a large chain length, such that the total amount of computational time remains the same for each step in temperature.

$\delta$ is determined by maximising the next equation as a function of $\delta$ [29].

$$\bar{C} - E[C_{k+1,j}] - \lambda \frac{\ln(1+\delta_j)}{\ln(1+\delta_1)} * T_1$$

The value of $\delta$ that maximises this function is used to determine the temperature en length of the next chain. $\bar{C}$ indicates the average value of the cost function over the complete phase-space, $E[C_{k+1,j}]$ indicates the Expectation of the cost function for chain k+1 with $\delta=\delta_j$. The expression preceded by $\lambda$ determines the length of the chain given a certain delta. To a first approximation we can take for $\lambda$:

$$\lambda = \frac{\bar{C} - C_{opt}}{K * n}$$

With Copt the optimal value of the cost function and K the total number of chains in a non-Bayesian schedule, n is the length of the chain.

Applying this Bayesian cooling to real problems seems to indicate a preference for a small $\delta$ and short chain lengths. The use of this algorithm is in part determined by the trade-off between the additional computing time for estimating the Expectation of the cost function of the next chain and a more ‘directed’ annealing scheme.

In one of the case studies discussed in the last two chapters we will apply the Simulated Annealing algorithm to mimic the process of crystal growth in equilibrium.

*'Don't fall in love with your model ...!'*

François Cellier

## III Modelling with Cellular Automata

Cellular Automata [33, 34] are used as a solving model for (highly parallel) information processing of (mainly) continuous systems. The model supports the so-called myopic algorithms where elementary operations take only one reference to bounded well defined subsets of data. Basically a Cellular Automaton is a collection of objects (the cellular automata) each of which have a state that is a function of the state of the other objects, and which is evaluated according to a certain set of rules. The system evolves in a set of discrete time-steps, with each object evaluating its new state at each step according to its rules, which may be different for different objects.

Definitions of simple cellular automata (CA)

∴ Let the domain of a CA be defined by 'a lattice' consisting of sites with cellular automaton rules.

∴ Each site of the lattice is in a state ("spin") pointing either up or down.

∴ The orientation of each spin at time t+1 is determined by that of its neighbours at time 't'. If the rules are probabilistic then the CA resembles[6] the so-called "Ising Spin Model" (see previous chapters).

Example (infection):
Initially the sites are occupied randomly with probability P. Then an empty site becomes occupied *if and only if* it has at least one occupied neighbour. (Occupied = spin up = true = 1; empty is spin down = false = 0 or = -1).
As a result the whole lattice will fill up with a specific system time (i.e. number of iterations) that depends logarithmically on the lattice size.

Simple Techniques

Take for instance a one-dimensional infection through a logical OR-rule:

```
For  i = 1 to L
      n(i) = n(i-1).OR.n(i+1)
```

With boundary conditions n(0) = n(L) and n(L+1) = n(1).
For two and three dimensions the sites are numbered consecutively from i=1 to $i=L^d$; this is the so-called 'Helical Boundary Condition'. There are buffer- lines and planes at the top and the bottom.

It is essential to note that we presumed an implicit updating scheme where the new n(i) depends on the new n(i-1), Traditionally one uses explicit updating schemes with the new n(i) depending on the old n(i-1). The latter is well suited for parallelisation.

### III.1 Formal Description and Classification (adapted from [35] and [33])

Let $\mathbf{a}_{i,j}^{(t)}$ denote the value of site *i*,*j* at time stamp 't' in a two dimensional square cellular automaton. The automaton then evolves according to

$$\mathbf{a}_{i,j}^{(t+1)} = \mathbf{F}\left[\mathbf{a}_{i-r,j-r}^{(t)}, \mathbf{a}_{i-r+1,j-r+1}^{(t)}, \cdots, \mathbf{a}_{i,j}^{(t)}, \cdots, \mathbf{a}_{i+r,j+r}^{(t)}\right]$$

6 A contradition: Compare the *probabilistic* Cellular *Automata* with a Sigarette Slotmachine that has a 20 % intrinsic probability of automatically taking your money without giving sigarettes in return!

Where **F** denotes the cellular automaton transformation rule and each site is specified by an integer $\mathbf{a}_{i,j}^{(t)}$ in the range 0 to k-1 at time 't'. The range parameter r specifies the region affected by a given site.

Although not restrictive, regular CA are specified by a number of generic features:

- Discrete in space, time, and state;
- Homogeneous (identical cells);
- Synchronous updating.

With a set of rules that posses the following characteristics:

- Deterministic;
- Spatially and temporally local.

Reformulation of this equation in terms of the base number 'k' results in:

$$\mathbf{a}_{i,j}^{(t+1)} = \tilde{\mathbf{F}}\left[\sum_{l=-r}^{r} \alpha_l \ \mathbf{a}_{i+l,j+l}^{(t)}\right],$$

where the transformation parameter set $\alpha_l$ consist of integer values coded by:

$$\alpha_l = \mathbf{k}^{r-l}.$$

This decoding results in a function $\tilde{\mathbf{F}}$ that simply transforms one integer value as argument into an updated site value. From the discreteness implicit in this description it is clear that only a countable limited number of unique transformation rules exist (number of rules = $\mathbf{k}^{\mathbf{k}^{(2r+1)}}$ i.e. 256 for k=2, r=1).

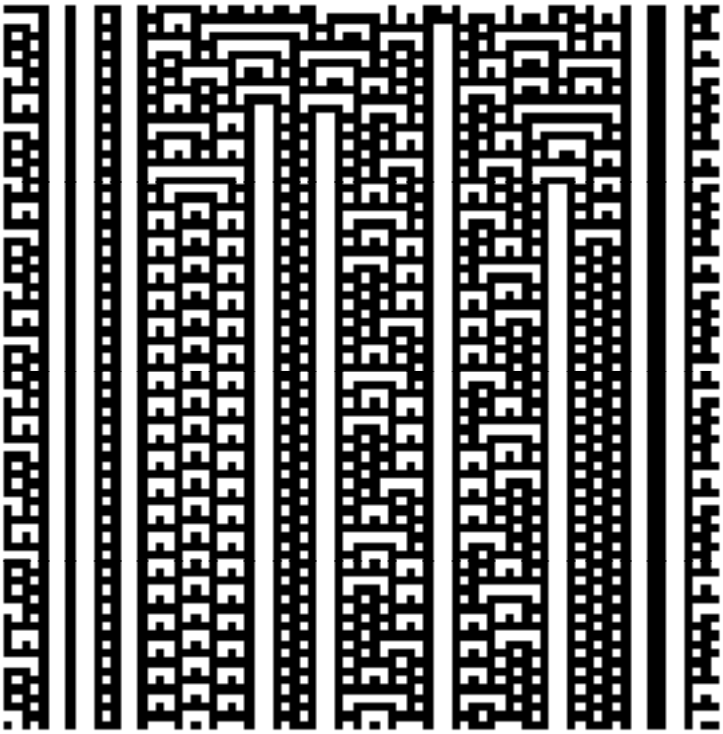
A 1D CA for instance, starts with a row of cells each of which has a value. The CA "evolves" by producing successive rows of sites, each new row representing a time step. The value of each site in a new row is determined by applying a specific rule to the site in the preceding row. The rule is homogeneous, which means that the same rule is used in determining the value of each new site, and the rule is local, which means that it is based the values of a "neighborhood" of sites, consisting of the site and/or its nearest-neighbor sites. The rule is applied to all of the sites in a row simultaneously. For a finite CA, the nearest-neighbor sites of the first and last sites in a row are taken from the other end of the row.

Although CA represent an elegant solving model for a large class of complex problems, quantitative research into the basic properties of the time- and state evolution of CA is still in its infancy. Some qualitative important results however have been obtained. Wolfram for instance finds that the majority of CA can be classified into 4 classes. The figures show 1-D examples of the classes with r=1, k=2 and the rule numbers indicated (respectively: 136, 73, 45, 110) and were created with Mathematica™.

Class I: Evolution from <u>all</u> initial states leads -after a finite time- to a homogeneous state where all sites have the same value, comparable to limit points.

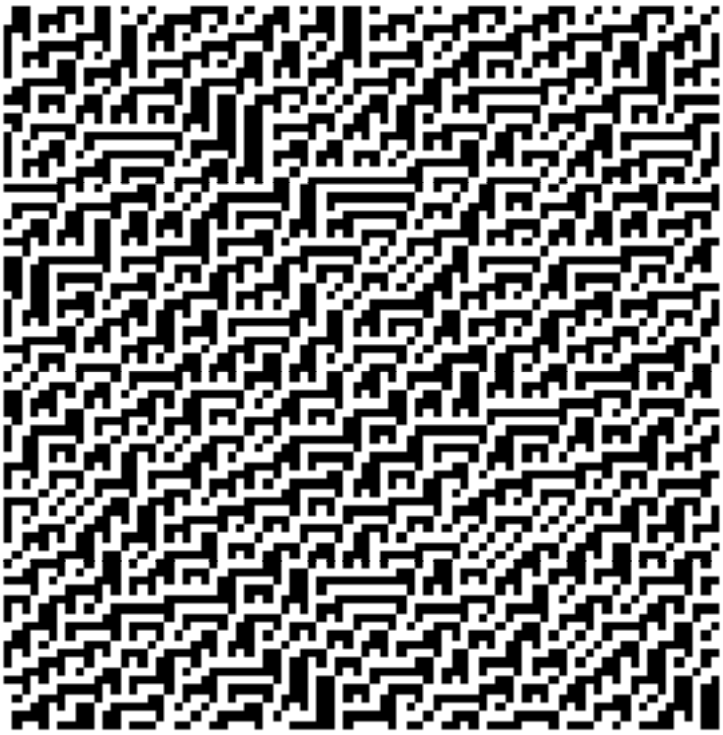
A k=2, r=1, Rule=136 1-Dimensional Cellular Automaton

Class II: Evolution leads to a set of separated simple stable or periodic structures. The position of these structures depends on the initial conditions. Local changes in the initial configuration propagate over a limited area only, comparable to limit cycles. (*Example*: A digital filter for image reconstruction).



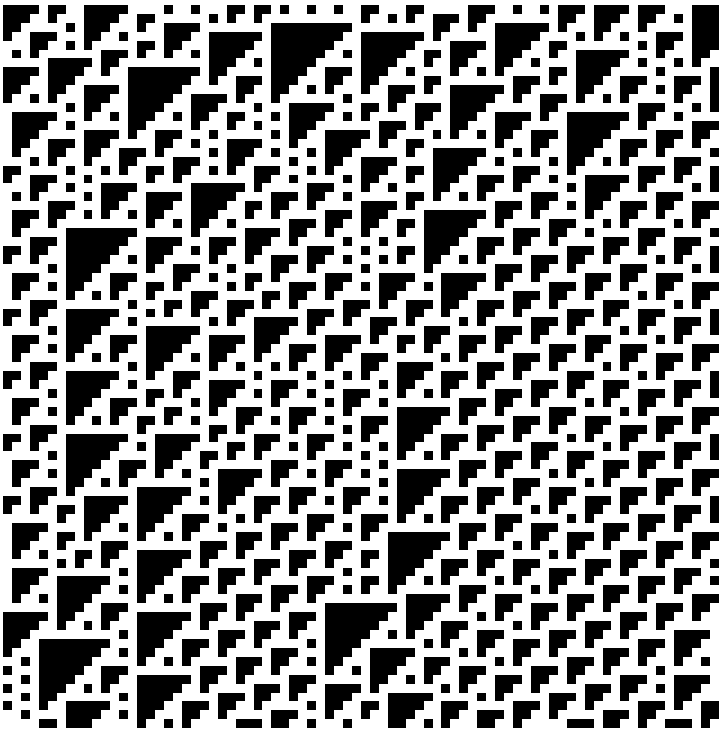
A k=2, r=1, Rule=73 1-Dimensional Cellular Automaton

Class III: Evolution leads to a chaotic pattern, both in time and space. These patterns have a close resemblance with the chaotic behaviour studied in the theory of dynamical systems under the influence of strange attractors. It is impossible to describe completely the initial configuration. These patterns tend to converge to invariant statistical properties such as fractal dimension, entropy etc. [36]. (*Example*: A model for many real-life physical systems possessing chaotic behaviour)



A k=2, r=1, Rule=45 1-Dimensional Cellular Automaton

Class IV: Evolution leads to complex (sometimes meta-stable) localised structures (sometimes long-lived). In contract to class 2, the information propagation is not limited. Cellular Automata capable of universal computing fall into this class.(*Example:* A model for the 'Game of Life' or a general purpose computer, Billiard Ball model)



A k=2, r=1, Rule=110 1-Dimensional Cellular Automaton

This notion of classes is important because it provides the researcher with an insight into the basic features of his CA with respect to a larger class of automata. As a consequence predictions about the temporal and spatial behaviour of his CA can be inferred. In later chapters we come back to this classification and discuss their relation to structural and behavioural complexity in system dynamics.

## III.2 Applications

Cellular automata are related to many models such as partial differential equations. Numerical approximations to partial differential equations typically involve a discrete grid of sizes, updated in discrete time steps according to definite rules. Such finite difference schemes differ from cellular automata in allowing sites to have *continues* values. Nevertheless, it is common to find that collections of sites yield average behaviour that can accurately mimic continuous variables. The cellular automaton evolution leads to effective randomisation of individual site values, but maintains smooth macroscopic average behaviour (see Wolfram [33]). For a few common classes of models related to CA the differences are listed below:

- **Partial Differential Equations:** Space, time and values are continuous.
- **Finite Difference Equations/Lattice Dynamical Systems:** Site values are continuous.
- **Dynamic Spin Systems:** Rules are probabilistic and updates may be asynchronous.
- **Direct Percolation:** Rules are intrinsically probabilistic.
- **Markov random fields:** Sites carry probabilities not definite values.
- **Particle Models:** Particles can have continuously variable positions and velocities.
- **Systolic Arrays:** Sites can store extensive information, boundary conditions can be different.

2 typical applications of CA:

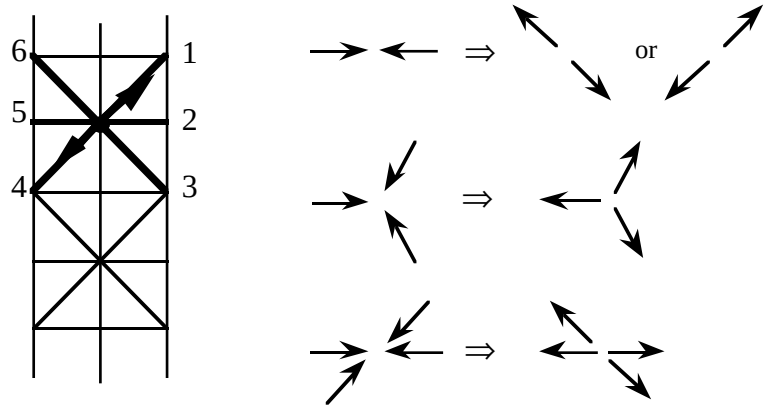
*Q2R Automata for Ising Model*
Rule: Flip a spin *if and only if* the energy does not change, i.e. if it has as many up as down neighbours. Updating: Sequential or lattice updating. Initially a random fraction of P spins is up.
Result: Curie point on square lattice at P=0.079551. Above this threshold (paramagnetic) the algorithm works nicely, but behaves poor below it. This phenomenon is still not completely understood. Apart from studying macroscopic behaviour, the model can be used to teach

spontaneous magnetisation in a course of electricity and magnetism before quantum mechanics and exp(-E/kT) is known. Thus a method to make Maxwell's equations more interesting.

*FHP hydrodynamic CA, "Lattice Gas"* [37]
The 'poor man's molecular dynamics'. Lattice gas models fill a niche between molecular dynamics and continuum methods. Here particles move with unit velocity on the bounds leading from one site of a triangular lattice to a neighbour site. For each direction, each bond carries either *one* or no particle.
At integer times particles can meet at the lattice sites and are scattered there subject to some collision rules. Hexagonal lattices allow for diffusion and can be used to study hydrodynamics in very complex geometry such as oil in North sea sand or flow through porous media. Notorious drawback of the simulations is the enormous finite size effects and the introduction of noise in the system and the problem of representing diffusion in 3D models. Lattice Boltzmann methods may provide a way out here (see for instance [38]).

Hexagonal CA
for Hydrodynamics

With Collision Rules:




One-bit implementation with numbering as in the figure and Xn and Yn site n before and after collision:

```
Y3 =   (X2.AND.Col.AND.Ang).OR.(X4.AND.Col.AND.NOT.Ang).
       OR.(X3.AND.NOT.Col)
```

Where Col is true if collision occurred and Ang 1 for counter clockwise and 0 for clockwise.
This results in very efficient implementations with approximately 1.6 GigaUpdates per second on a Cray-YMP-8 for 200 MegaSites in f77!
To get an impression of the state of art, recently Gerling et al. reported 14.2 updates/nsec on a single processor of the NEC-SX3 for rule 96 (XOR, one-dimensional k=2, r=1), and 0.9 updates/nsec for a 2-dimensional Kauffman model [39].[7]

## III.3 Implementation Aspects

Impressive speedup can be obtained by storing spins in single bits and treat all 64 or 32 spins in one word *concurrently* by one command. For instance: n(i) = ior(n(i-1),n(i+1). Where we implicitly consider only binary sites.[8] In order to avoid inefficiently shifting large amounts of data Rebbi and later Herrmann introduced the concept of sub lattices:
For example take L=96 and 32-bit integers 'n', where the three integers n(1), n(2), and n(3) contain all 96 spins in a mixed form given by:

7 Kauffman model: a random boolean network model where each site selects randomly at the beginning which rule it wants to follow in its later dynamics

8 Matheus [5,37] (With thanks to Prof. Dr. D. Stauffer for bringing this to my attention!).

n(1) = spins 1,4,7,...,94
n(2) = spins 2,5,8,...,95
n(3) = spins 3,6,9,...,96

and the boundary buffer words are: n(0) = SHIFT n(3) (equal to spins 96,3,6,..,93) and n(4) = SHIFT n(1). This domain scattering results in a update scheme that allows you to update hundreds of millions of sites in one second on a vector processor. State of the art is an update frequency of 1.4 $10^{14}$ sites per second!

*'I think the computer understands it, now I want to understand it!'*

Eugene Wigner

## IV: Event-driven versus time-driven simulation

We have seen that in continuous systems the state variables change continuously with respect to time, whereas in discrete systems the state variables change instantaneously at separate points in time. Unfortunately for the computational physicist there are but a few systems that are either completely discrete or completely continuous, although often one type dominates the other in such hybrid systems. The challenge here is to find a computational model that mimics closely the behaviour of the system, specifically the simulation time-advance approach is critical.
If we take a closer look into the dynamic nature of simulation models, keeping track of the simulation time as the simulation proceeds, we can distinguish between two *time-advance* approaches:

- *Time-driven* and *Event-driven*.

In time-driven simulation the time advances with a fixed increment, in the case of continuous systems. With this approach the simulation clock is advanced in increments of exactly $\Delta t$ time units. Then after each update of the clock, the state variables are updated for the time interval $[t, t+\Delta t)$. This is the most widely known approach in simulation of natural systems. Less widely used is the time-driven paradigm applied to discrete systems. In this case we have specifically to consider whether

- the time step $\Delta t$ is small enough to capture every event in the discrete system. This might imply that we need to make $\Delta t$ arbitrarily small, which is certainly not acceptable with respect to the computational times involved.

- the precision required can be obtained more efficiently through the event-driven execution mechanism. This primarily means that we have to trade efficiency against precision.

In event-driven simulation on the other hand, we have the next-event time advance approach. Here (in case of discrete systems) we have the following phases:

*Step 1:* The simulation clock is initialised to zero and the times of occurrence of future events are determined.

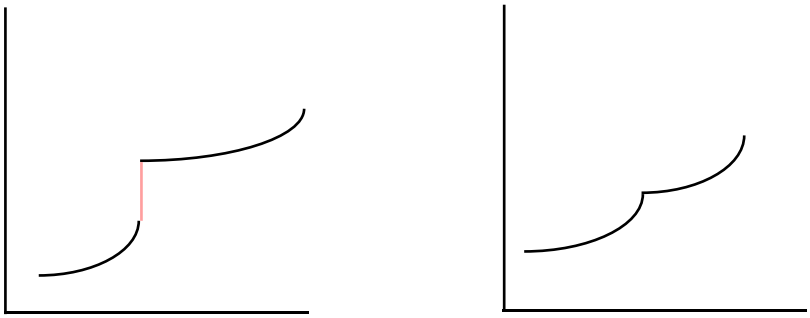
*Step 2:* The simulation clock is advanced to the time of the occurrence of the most imminent (i.e. first) of the future events.

*Step 3:* The state of the system is updated to account for the fact that an event has occurred.

*Step 4:* Knowledge of the times of occurrence of future events is updated and the first step is repeated.

The nice thing of this approach is that periods of inactivity can be skipped over by jumping the clock from event_time to the next_event_time. This is perfectly save since -per definition- all state changes only occur at event times. Therefore causality is guaranteed. The event-driven approach to discrete systems is usually exploited in queuing and optimisation problems. However, as we will see next, it is often also a very interesting paradigm for the simulation of continuous systems.

Consider a continuous system with where every now and then (possibly at irregular or probabilistic time steps) discontinuities occur, for instance in the temperature of a room where the heating is regulated in some feed-back loop:



Typical discontinuities in Time versus State trajectories of a continuous systems or its (higher order) derivative with respect to time.

The difficulty in the -here inefficient- time-driven simulation of such a system is in the integration method applied. Specifically multi-step integration methods to solve the underlying differential equations have might prove not to work in this case. The reason is that these methods use extrapolation to estimate the next time step, this would mean that they will try to adapt to the sudden change in state of the system thus reducing the time step to infinite small sizes.
Other examples are:

Continuous Ising spin systems where a configuration is defined by the spin variables $s(\nu) = \pm 1$, specified at the vertices $\nu$ of a two or three dimensional lattice and an independent Poisson process of attempted spin change arrivals are associated with the vertices of the lattice. If in such system an attempted change arrives at vertex $\nu$, the spin $s(\nu)$ is changed to $-s(\nu)$ with probability P, and with probability 1-P the spins remain unchanged.

Billiard ball models, used in gas dynamics, where N equal balls move frictionless in 2D and where each ball moves along a straight line with constant velocity, until it comes into contact with another ball. Upon such a contact a collision occurs whereupon the two participating balls instantly change their velocities and directions.
It is clear that time-driven simulation might not prove to be acceptable since this mechanism is slow and/or not sufficiently precise if $\Delta t$ is not chosen to be very small. An event-driven simulation where the state of the balls is updated from collision to collision of any ball, might prove much more accurate and efficient.
Let's explore this example in a little more detail (see for instance also: [40]):

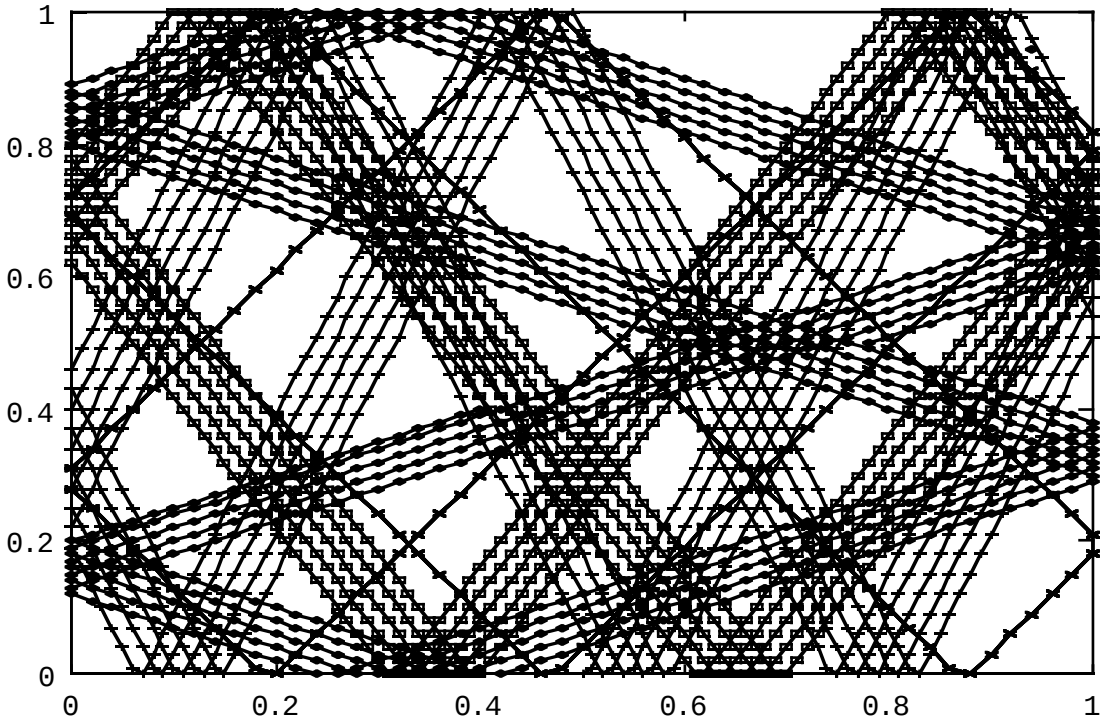
*Billiard Model Example*

Consider a Billiard Ball model to simulate the gas dynamics of 4 gas molecules (as long as we do not try to do any physics with this simple system it can perfectly well explain the concepts we are discussing). Assume that the molecules move in a straight line with constant velocity until they hit the wall of a 2D system (kinetic energy is conserved in this model). We can now investigate the consequences of the two paradigms; Time-driven versus Event-driven simulation:

```
input stop simulation time, Δt;
initialise balls with x,y position and velocity vx,vy in x,y
direction;
time = 0.0;
while (time < stop simulation) {
      for each ball do {
          x += Δt * vx; y += Δt * vy;
          if (x == 0) vx = -vx;
          if (y == 0) vy = -vy;
      }
      time += Δt;
}
```

Pseudo code for time-driven gas simulation

For a simulation time of 1.0 (arbitrarily units) with a ΔT of 0.001, implying 1000 state evaluations, this time-driven approach results in the following simulated behaviour of the 4 balls over time:



Time-driven state evaluation of gas model, 1000 state evaluations, 4 molecules.

Next we consider the case of a event-driven simulation of this 4 molecule gas. The pseudo code shown takes into account all the typical aspects of detecting, handling and processing events within the correct time frame.

```
input stop_simulation_time;
initialise balls with x,y position and velocity vx,vy in x,y direction;
for each ball do {
        impact_time = collision_time(ball);
        schedule(MOVE, impact_time, ball);
}
prev_time = 0.0;
while (time() < stop_simulation_time) {
        next_event(&event, &ball);
        switch(event) {
        case MOVE:
                update_positions(time() - prev_time);
                impact_time = collision_time(ball);
                schedule(MOVE, impact_time, ball);
                prev_time = time();
        break;
}

collision_time(ball)
{
        if (vx >= 0) {
                t0 = (1 - x) / xv;
        } else {
                t0 = -x / xv;
        }
        if (vy >= 0) {
                t1 = (1 - y) / yv;
                } else {
```

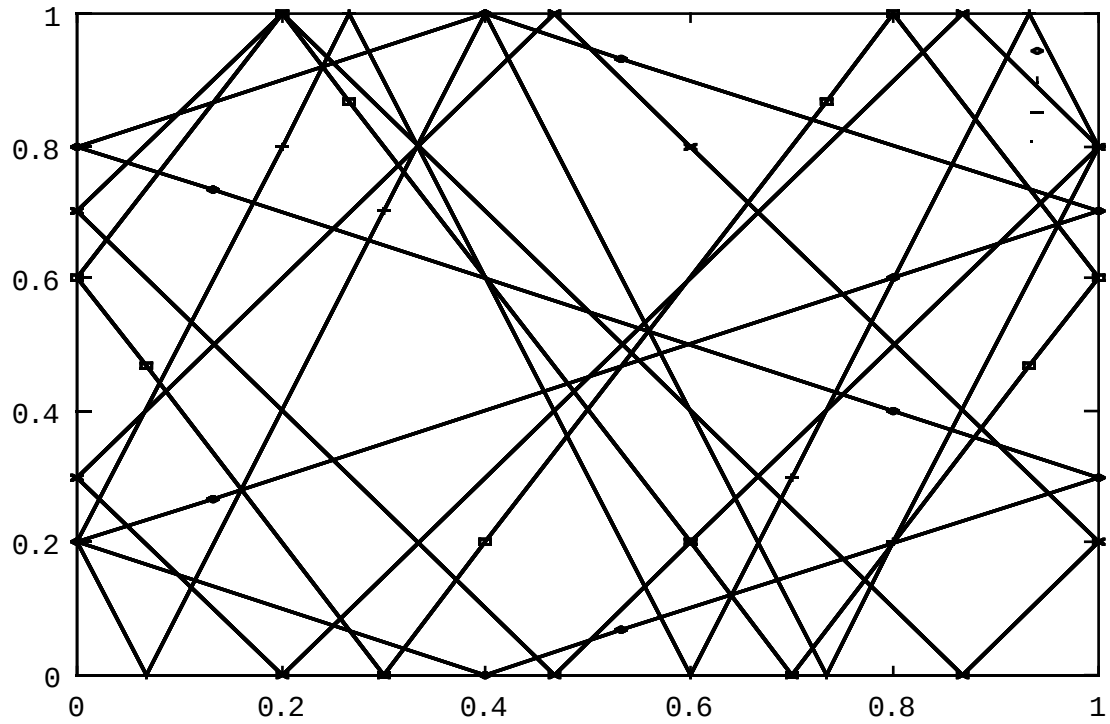
```
                t1 = -y / yv;
        }
        return min(t0, t1);
}

update_positions(Δt)
{
        for each ball do {
                x += Δt * vx;
                y += Δt * vy;
                if (x == 0 || x == 1)
                        vx = -vx;
                if (y == 0 || x == 1)
                        vy = -vy;
        }

        }
}
```

Pseudo code for an event-driven gas simulation

For a simulation time of 1.0 (arbitrarily units) with 60 state evaluations, this event-driven approach results in the following simulated behaviour of the 4 balls over time:




From these examples we see that the cumulated error over time in the time-driven simulation, even for a Δt as small as 0.001, results in a violation of the conservation of momentum. The gain we have in event-driven simulations is however partly frustrated by the growth in the (complexity) of the simulation code. If we would like to go to modern (parallel) architecture's, event-driven would become increasingly more difficult to code due to the complicated synchronisation protocols [41]. There is always the necessity to find a balance between the pro's and con's of time-driven versus event-driven simulation.
In general we observe that:

- If the time between the succeeding events become small, time driven is preferred (in the example this situation occurs for a system of billions of molecules to be simulated). This is the *frequency* parameter.

- The overhead per event (i.e. the state update) is larger with event-driven simulation. This is referred to as the *event overhead*.
- The amount of work between any two events plays an important role. This we call the *granularity* of the system.

The factors: frequency, overhead and granularity, together with the consequences for implementation on modern architecture's and the required accuracy, must be studied carefully before making a decision on what simulation mechanism should be used.

In view of the discussion in the previous paragraphs on cellular automata and computability, it is interesting to note that the Billiard-ball model was first introduced as *model of computation* by Fredkin in 1982 [42]. He showed that by using only bumping balls and mirrors, and restricting initial configurations so that no head-on collisions occur, any digital computation can be constructed. The rationale behind this is that every place where a collision of hard spheres within a finite diameter occurs can be seen as a Boolean logic gate. It was Margolus who in 1984 showed that this model can be coded in a cellular automata with 6 simple rules [43].

In the last two chapters we will look into one research field where many of the ideas discussed sofar come together.

*'The Universe is the most efficient simulation of the Universe in the eye of God'*

E. Fredkin

## V: Modelling microscopic growth processes

In this chapter models for growth processes and patterns in physics and biology and some simulation strategies will be discussed. These examples use many of the techniques and principles discussed in the previous sections, are illustrative and are challenging topics of active research. First we introduce techniques common in modelling *microscopic* processes, then we apply these to crystallisation in equilibrium and non-equilibrium growth.

Many growth processes in nature can be described as aggregation processes, where the growth form emerges from the addition of particles to a preceding growth stage. This type of growth process is found in a wide range of objects in nature, such as many crystals and bacteria colonies. The growth form consists of aggregates of 'small' particles, in the example of a crystal and a bacteria colony the particles are molecules and cells respectively. In the growth process of these aggregates often objects are formed which are characterised by a high degree of irregularity and non-smoothness. It is not easy to describe the shape of these objects using the 'normal' elements (lines, circles etc.) from Euclidean geometry. An important method to model the morphology of such irregular objects from nature is fractal geometry. For this purpose the next section explains concepts of fractals and self-similarity. In the other sections two types of growth processes are discussed. The first type is growth in equilibrium, a growth process which can be described with a deterministic cellular automaton and is for example found in the formation of a perfect crystal. The second type is growth in non-equilibrium, which is present in many growth patterns in physics and biology and can be modelled with a probabilistic cellular automaton.

### V.1 Fractal dimension and self-similarity

#### V.1.1 Regular fractals

A well known object from fractal geometry [44] is the quadric Koch curve (Fig. V.1). The construction of this curve starts with the square in Fig. V.1a. In each construction step an edge is replaced by a set of 8 equal-sized new edges. In Figs. V.1b and c the first and the second stage in the construction are shown. This process results in the curve shown in Fig. V.1d which is known, in the limit case, as the quadric Koch curve. This type of curve was considered in the past as a pathological case for which certain mathematical properties cannot be determined. The curve is characterised by three remarkable properties: it is an example of a continuous curve for which there is no tangent defined in any of its points, it is locally self-similar on each scale: an enlargement of the object will yield the same details, and the total length of the curve is infinite. The quadric Koch curve is an example of an object with a fractal dimension. A fractal dimension differs from an ordinary dimension in the sense that it is, in many cases, not an integer but a fraction. The value of the fractal dimension $D$ can, in this special case, be determined analytically: the value is 1.5 exactly. The value of $D$ may be calculated for this self-similar curve made of $N$ equal sides of length $r$ by using Eq. (V.1) from Mandelbrot [44].

$$\mathbf{D} = \frac{\log(\mathbf{N})}{\log\left(\frac{1}{\mathbf{r}_{\mathbf{sim}}}\right)} \qquad \text{(V.1)}$$

The ratio $\mathbf{r}_{\mathbf{sim}}$ of the length of an edge in a construction stage and the preceding construction stage is known as the *similarity ratio*.

Fig V.1 : a,b,c: First three construction stages of the quadric Koch curve (d).

The physical properties of an object with a fractal dimension can be related to the intuitive idea of dimension by using a mass-length scaling relation [45]:

$$M(R) \sim R^D \qquad \text{(V.2)}$$

To calculate the fractal dimension *D* of an object we have to determine the 'mass' *M(R)* within a circle or sphere with radius *R*, somewhere centred on the object. The number of fundamental units triples when *R* is tripled in the case of a line (Fig. V.2a) leading to a value *D*=1. In the case of a plane the number of fundamental units is multiplied with a factor 9 when *R* is tripled, leading to *D*=2. In the example of Fig. 2c the number of fundamental units increases with a factor 5 when *R* is tripled. The value of *D*=log(5)/log(3) is now intermediate between a line and a plane and can be easily determined since the object in Fig. V.2c is completely self-similar.

Fig V.2 : a,b,c: Determination of the fractal dimension *D* using the mass-length scaling relation in Eq. (V.2) (after Sander [45]).

The fractal dimension can be used to measure the plane-filling or space-filling properties of a curve. A value of $D \approx 2$ of a curve indicates that the curve is nearly space-filling.

### V.1.2: Fractal growth patterns

In the examples of Fig. V.1d and V.2c it is possible to determine the fractal dimension *D* analytically. Many objects from nature share the same peculiar properties with mathematical constructions within in a certain scale range. Many growth forms from nature show a high degree of non-smoothness and a (statistical) property of self-similarity. In Fig. V.3 the outline of a digitised photograph of a hydro-coral is shown. The fractal dimension of this outline can be approximated experimentally by estimating the length of the outline. The length of the perimeter of the object can be estimated by mapping the picture of Fig. V.3 onto a grid with grid spacing $\varepsilon$. The number $N(\varepsilon)$ of grid sites, the boxes, in which a part of the outline of the object is present, is

counted for various values of $\varepsilon$. The value of the estimated fractal box dimension $\mathbf{D_{box}}$ [46] can be estimated by using the relation:

$$\mathbf{N}(\varepsilon) \sim \varepsilon^{\mathbf{D_{box}}} \qquad \text{(V.3)}$$

In Fig. V.4 two plots are shown of $\mathbf{N}(\varepsilon)$ for various values of ε. The value of $\mathbf{D_{box}}$ is approximated by determining the slope of the line through the measurements plotted in Fig. V.4. In Fig. V.4a this method is calibrated by using the quadric Koch curve from Fig. V.1d. In Fig. V.4b the box dimension is estimated for the outline of the hydro-coral depicted in Fig. V.3, resulting in a value of $D_{box}$ of about 1.6. In experiments $D_{box}$ can be used to estimate the degree of irregularity or space-filling properties of actual and simulated objects. This morphological property can be used to compare actual and simulated objects and is furthermore useful to compare a range of gradually changing objects. What we do here is look for properties that allow us to validate the conceptual model against the natural model, taking the approach outlined in Chapter I (examples will follow in the section on diffusion limited aggregation).

Fig V.3 : Outline of a digitised photograph of hydro-coral.

Fig V.4 : Plots of $N(\varepsilon)$ for various values of $\varepsilon$. In the measurements are plotted for the quadric Koch curve from Fig. V.1d, in (b) for the outline of the hydro-coral in Fig. V.3.

## V.2 Growth in equilibrium

In a growth process in equilibrium, as for example found in a perfect crystal where the growth process is near or in equilibrium, molecules are ‘exploring’ various sites of the crystal and are added to the crystal until the most stable configuration is found. In this type of growth process a continuous rearrangement of particles takes place, the process is relatively slow and the resulting objects are very regular [45]. In some cases the growth form which emerges is a ‘normal’ object from Euclidean geometry whereas in other cases objects are formed that resemble regular fractal objects.
An aggregation process, with growth in equilibrium, can be modelled with a deterministic cellular automaton. An example of the construction of the crystal-like form depicted in Fig. V.2c, using a deterministic cellular automaton, is shown in Fig. V.5. The object is represented by a cluster of occupied sites in a lattice. In the growth process unoccupied sites are added to the cluster. The rules which are applied in the cellular automaton are very simple: an unoccupied site is added to the cluster when it neighbours to an occupied site and to three unoccupied sites in the lattice. The result of this construction, after many iteration steps, is the regular fractal from Fig. V.2c.

Fig. V.5: Growth in equilibrium modelled by a deterministic cellular automaton

### V.2.1 Crystallisation on a sphere [47].

In this section a case study is given of the emergence of a highly regular crystal. In this case crystallisation with spherical boundary conditions is studied. The particles are allowed to move over the sphere until the most stable configuration is found. For the simulation of crystallisation the spherical space affects the properties of the crystalline state in an essential way, for example on a flat surface a hexagonal lattice will be formed, on a spherical surface such lattices can not exist without defects. These simulations can provide important information for the study of closed 2D systems. Especially the ordering of the particles and defects are of interest in this close packing problem.
The idea for these kind of simulations comes from a remarkable phenomenon that was observed during experiments with fragmented biomembranes. This fragmented biomaterial spontaneously form vesicles (see Fig. V.6), spherical shells consisting of a bilayer of lipid molecules [48].

Fig V.6 : A schematic representation of a vesicle.

The surprising result was that only very specific sizes of those vesicles are formed. To explain this result the following hypothesis was made. The lipid molecules on the inner layer of the vesicle pose stringent packing constraints and therefore the molecules will search for optimal packings that have a minimal energy. From physics it is known that symmetric arrangements are good candidates for the lowest energy states. From mathematics it is known that on a spherical surface only specific numbers of points can be distributed with high symmetry, more specifically the platonic solids with 4,6,8,12 and 20 points (tetrahedron, octahedron, cube, icosahedron (see Fig. V.7) and dodecahedron respectively) are the only perfectly regular arrangements. An other highly symmetric arrangement is the much talked about Buckyball with 60 particles.

Fig V.7: The icosahedral platonic solid.

The formation of these very regular arrangements is an example of the formation of a form in equilibrium, which results in one of the 'normal' objects of Euclidean geometry.

The discrete sizes, found in the experiments with fragmented biovesicles, might be explained by the (symmetric) low energy arrangements on a spherical surface. To test this hypothesis we have designed a simulation that gives lowest energy arrangements of particles on the spherical surface [47, 49]. This problem of crystallisation of particles can, like many problems originating from physics, chemistry and mathematics, be formulated as an optimisation problem. A vast majority of these problems involve the determination of the absolute minimum of a underlying multidimensional function. Usually optimisation of these complex systems is far from trivial since the solution must be attained from a very large irregular candidate space, containing many local extrema. As a consequence the computational effort required for an exact solution grows more rapidly than a polynomial function of the problem size, the problem is said to be NP (non-polynomial time) complete. Because it is impossible to examine all solution candidates, even in principle, approximation methods are required. A well established computational scheme is the Simulated Annealing (SA) method. In physical annealing a material is heated to a high temperature and then allowed to cool slowly. At high temperature the molecules move freely with respect to one another. If the liquid is cooled slowly, thermal mobility is lost. The molecules search for the lowest energy consistent with the physical constraints. The potential energy between the molecules is described by the Lennard-Jones potential. The formula for the Lennard-Jones potential is :

$$V(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$$

with r the distance between two particles, $\sigma$ a measure of the width of the potential and $\varepsilon$ the well depth.

In terms of the crystallisation problem at hand the procedure works as follows. First N particles are randomly placed on a spherical surface. The annealing begins by creating a Markov chain, of given length, at a certain temperature. The Markov chain grows by randomly displacing particles and calculating the corresponding change in energy of the system and deciding on acceptance of the displacement. The moves are accepted with probability $P(\Delta E,T)$ at temperature T according to the following scheme :

$$P(\Delta E, T) = \exp(-\Delta E / T) \qquad \text{if } \Delta E > 0$$

$$P(\Delta E, T) = 1 \qquad \text{if } \Delta E \leq 0$$

$$\text{with } \Delta E = E_{new} - E_{old}$$

Unaccepted moves are undone. This Metropolis choice of P(ΔE,T) guarantees that the system evolves into the equilibrium distribution [24].

The Lennard-Jones potential has a minimum at distance $r = 2^{1/6}\sigma$, this distance is energetically favoured for neighbouring particles. This means that the radius of the sphere is dependent on the number of particles. Therefore during the annealing the radius is also optimised, this is done by calculating the energy of the system at a randomly generated new radius and subtracting the current energy. Acceptance is also decided according to the probability scheme given above.
After a chain has ended the temperature is lowered by multiplying the temperature with the cool-rate, which is a number slightly less than 1 (typically 0.9) after which a new chain is started. This process continues until a stop criterion is met. The stop criterion in our implementation is met when the standard deviation in the final energies of the last ten chains falls below a certain value (typically $10^{-6}$).
In theory SA guarantees the finding of the global minimum. In practice the Markov chains can not be of infinite length, and therefore it is not guaranteed that the global optimum is found. The Markov chains should however be of sufficient length to be reasonably sure that within a few trials the optimum is found. The length of the Markov chains that are chosen in practice are a function of the number of particles on the sphere. The length increases so fast as a number of particles that calculation of thousands of particles is not yet possible. Therefore we have also looked at vectorised and parallel implementations, these are discussed elsewhere [31,32].
One of the first things that one looks for is if the simulation reproduces theoretical predictions. For example it can be checked if the platonic solids are also found in the experiments. It turns out that the triangular platonic solids, N=4, 6 and 12, are indeed optimal solutions while the non triangular solids, N=8 and 20, are both not optimal. An other example is N=32, there we find an arrangement that can be described as a dodecahedron and an icosahedron merged together, while its dual with N=60, the Buckyball, is no optimal solution. The above described results are also found if the Coulomb potential instead of the L.J. potential is used.

This simulation procedure allows for a more closer look at the influence of the anneal temperature on the configuration. As explained before in simulated annealing we start with a system at a high temperature, such that most of the phase space can be sampled. In Fig. V.8a we see a sample arrangement in which the particles are not completely ordered, this is more clear if we look at the radial distribution function in Fig. V.8b. The radial distribution function is taken as an average over a number of arrangements at that temperature. This function shows what the probability is of finding a particle at a certain distance from a reference particle. If we have a 2D system of infinite size we would expect no correlation between particles with an infinite distance, therefore the radial distribution function has a value of 1 at large distances which means that we just expect to find the average density without any structure such as with short distances.

<- Fig V.8a : Sample arrangement for N=32 at a high temperature.

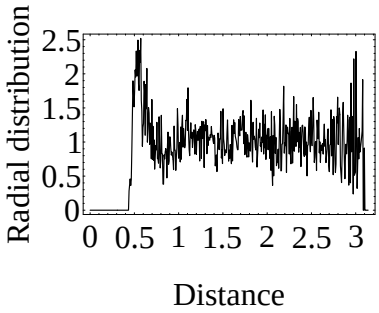



Fig. V.8b : Radial distribution function for N=32 at a high temperature.

If the temperature is lowered and approaches zero, the particles will search for the lowest energy state(s). The SA approach makes sure that the system is able to get out of local minima and in principle guarantees finding of the lowest energy state(s). At very low temperatures the system is highly ordered, see Fig. V.9a for the arrangement and Fig. V.9b for the radial distribution function. In the arrangement it is clear that all particles have a set of 5 or 6 particles around it. From the radial distribution function it can be seen that the distances between the particles are discrete, a sign of crystallisation.

<- Fig. V.9a : Sample arrangement for N=32 at low temperature.

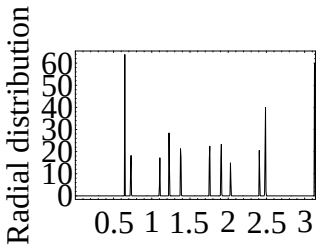



Fig. V.9b : Radial distribution function for N=32 at a very low temperature.

After a number of runs with different numbers of particles (and some extreme amount of computing time) we can study the potential energy of the system versus the number of particles. In [47] it was shown that at some specific N there are configurations of lower potential energy than configurations with neighbouring values of N. This might be indicative for some sort of global discretisation of this type of spherical systems.

## V.3 Growth in non-equilibrium

### V.3.1 Diffusion limited aggregation using cellular automata

Many growth processes in nature are not in equilibrium. An extreme example is an aggregation process of particles where as soon as a particle is added to the growth form, it stops trying other sites an no further rearrangement takes place. The local probabilities that the object grows are not everywhere equal on the aggregate and an instable situation emerges. The growth process in non-equilibrium is relatively fast and often irregular objects, characterised by a fractal dimension, are formed [50, 45, 51].
An example of a growth process, in non-equilibrium from physics, is viscous fingering. The phenomenon can be demonstrated in a experiment where air displaces a high-viscosity fluid between two glass plates. In Fig. V.10 a diagram is shown of an experiment where air is injected between two glass plates at $y$=0 and displaces a high viscosity fluid, which is only removed at the top of the plates (both sides are closed). The pressure $P$ will be the highest at $y$=0 and the lowest at $y$=$L$, where $L$ represents the length of the glass plates. In the fluid the pressure is given by the Laplace equation [46]:

$$-\nabla^2 P = 0 \qquad \text{(V.6)}$$

In the air the pressure is everywhere equal, since its viscosity can be ignored. The pressure in the air equals to the input pressure $P$($y$=0) and the consequence is that the largest pressure gradients occur at the tips of the fingers in Fig. V.10, while the lowest gradients occur below the tips. The probability that the fingers continue to grow will be the highest at the tips and in a next growth stage the pressure gradients in the tips are still more amplified, resulting in an instable situation. In Fig. V.11 an example of the resulting growth pattern is shown, it is an irregular shaped object, known in the literature as viscous fingering.

Fig. V.10 (left): Diagram of a viscous fingering experiment
Fig. V.11 (right): Example of a viscous fingering growth pattern.

Another example of growth in non-equilibrium is growth of a bacteria colony (for example *Bacillus subtilus*) on a petri-dish [52]. The colony consumes nutrients from its immediate environment and the distribution of nutrients is determined by diffusion. When it is assumed that the concentration $c$ is zero at the colony and that the diffusion process is fast compared to the growth process, the concentration field will attain a steady state in which the diffusion equation

$$\frac{\mathbf{dc}}{\mathbf{dt}} = \mathbf{D}_{\mathbf{diffusion}} \nabla^2 \mathbf{c} \qquad \text{(V.7)}$$

equals zero. In this equation $D_{diffusion}$ is the diffusion coefficient. The nutrient source may be, for example, a circle around the colony or a linear source where the concentration is maximal. The local nutrient concentration at sites between the colony and the source can be described with the Laplace equation:

$$\nabla^2 \mathbf{c} = 0 \qquad \text{(V.8)}$$

The growth process of a bacteria colony, viscous fingering, and various other growth patterns from physics as for example electric discharge patterns and growth forms of electro deposits, can be simulated with one model: the Diffusion Limited Aggregation model [50]. At the heart of all these growth patterns there is one Partial Differential Equation, the Laplace equation, which describes the distribution of the concentration (Eq. V.8), pressure (Eq. V.6), electric potential etc. in the environment of the growth pattern. The DLA-model is a probabilistic cellular automaton which resides on a square two-dimensional or three dimensional lattice. The growth pattern can be constructed using the following Monte Carlo approach. The first step in the construction is to occupy a lattice site with a seed. After that, particles are released from a circle (using the seed as a centre) or from line at a large distance from the seed. The particle starts a random walk, the walk stops when the particles leaves the circle or reaches a perimeter site of the seed and sticks. Then more random walkers are released from the source and are allowed to walk until the distance with respect to the cluster with occupied sites becomes too large or it reaches a perimeter site, neighbouring to one of the previous particles, and it sticks. In Fig. V.12 the first stages are shown of the formation of the growth pattern. When this procedure is repeated many times an irregular growth pattern as shown in Fig. V.13 is generated. The cluster depicted in Fig. V.13 was generated on a 1000 x 1000 lattice, using a linear source at the top and a seed positioned at the bottom of the lattice. The fractal dimension $D_{box}$ of this growth form can be determined by applying the box-counting method (Eq. V.3). In this case the $D_{box}$ is about 1.7. The underlying Laplace equation in this Monte Carlo approach is solved by mimicking a diffusion process using random moving particles.

Fig. V.12 (left): First steps in the construction of the DLA-cluster. Sites which are part of the cluster are visualised as black circles, sites which are possible candidates to be added to the cluster in next iteration steps are indicated with open circles.
Fig. V.13 (right): DLA-cluster generated on a 1000 x 1000 lattice using a linear source of nutrient, located at the top row of the lattice

To obtain more insight in the 'nutrient' distribution around the DLA-cluster the underlying Laplace equation can be solved numerically and a DLA cluster can, alternatively, be constructed using the nutrient distribution over the lattice. The cluster is initialised with a seed and the following boundary conditions are applied: $c$=0 on the cluster itself and $c$=1 at the nutrient source, which may be circular, linear etc. The cluster is constructed using the following rules in the probabilistic cellular automaton:

- 1: solve the Laplace equation (Eq. V.8), using the boundary conditions $c$=0 on the cluster and $c$=1 at the nutrient source.
- 2: new sites are added to the cluster are added to the cluster with probability $p$ (Eq. V.9 ).
- 3: goto 1

The probability $p$ that a perimeter site (the sites indicated with an open circle in Fig. V.12) with index $k$ will be added to the DLA-cluster (black circles in Fig. V.12) is determined by

$$\mathbf{p}(\mathbf{k} \in (\mathbf{open\_circle}) \rightarrow \mathbf{k} \in (\mathbf{black\_circle})) = \frac{(\mathbf{c_k})^{\eta}}{\sum_{\mathbf{j} \in \mathbf{open\_circle}} (\mathbf{c_j})^{\eta}} \qquad \text{(V.9)}$$

The exponent $\eta$ applied in (V.9) describes the relation between the local field and the probability. This exponent usually ranges in experiments from 0.0 to 2.0. The sum in the denominator represents the sum of all local concentrations of the possible growth candidates (the open circles in Fig. V.12).

The Laplace equation can be solved, using the boundary conditions mentioned above, by finite differencing and the successive over-relaxation method:

$$\mathbf{c}_{\mathbf{i,j}}^{\mathbf{n+1}} = \frac{\omega}{4}(\mathbf{c}_{\mathbf{i-1,j}}^{\mathbf{n+1}} + \mathbf{c}_{\mathbf{i,j-1}}^{\mathbf{n+1}} + \mathbf{c}_{\mathbf{i+1,j}}^{\mathbf{n}} + \mathbf{c}_{\mathbf{i,j+1}}^{\mathbf{n}}) + (1-\omega)\mathbf{c}_{\mathbf{i,j}}^{\mathbf{n}} \qquad \text{(V.10)}$$

In this method the new local nutrient concentration in the lattice $c_{i,j}^{n+1}$, at a site with lattice co-ordinates $i,j$ , is determined in an iterative procedure which converges as soon as the difference between the new and old local nutrient concentration ($c_{i,j}^{n+1}$ -$c_{i,j}^{n}$ ) is below a certain tolerance level. The $\omega$ in Eq. V.10 is the over-relaxation parameter, which in general lies within the range $1 \le \omega < 2$. With this alternative construction of the DLA-cluster it becomes possible to visualise the basins of equal nutrient ranges around the growth pattern in Fig. V.13. The nutrient concentration decreases when the black or white basin is situated close to the object. The concentration in the white basin that surrounds the growth pattern is almost zero. In the example of Fig. V.13 a linear source of nutrient is used, positioned at the top row of the lattice and the exponent $\eta$ in Eq. V.9 is set to unity. From Fig. V.13 it can be observed that the probability that new sites will be added to the cluster will be the highest at the tips of the cluster, where the steepest nutrient gradients occur, and the lowest in the bays between the branches. In successive growth steps the nutrient gradients at the tips will even become steeper and a comparable instable situation is encountered as in the viscous fingering example (Figs. V.10 and V.11).

The effect of changing the exponent $\eta$ in Eq. V.9 is that the overall shape of the cluster changes. For the value $\eta$=0 the shape changes in a compact cluster and it can be demonstrated that the DLA-model for this special case transforms into the Eden model. This model is one of the earliest probabilistic cellular automata to simulate growth. In the Eden model each possible growth candidate in Fig. V.12 has the same probability to become occupied. For the value $\eta$=1 the 'normal' DLA-cluster is obtained, while for higher values more dendritic shapes are generated [53]. With the parameter $\eta$ the effect of nutrient gradients on the growth process can be controlled, where the Eden model is an extreme example in which gradients have no effect on the local probability that a new site will be added to the growth form. The objects which can be generated by varying $\eta$ in the range 0.0-2.0, can be compared to each other using $D_{box}$ from Eq. V.3. The value of $D_{box}$ in experiments where $\eta$ is set to the values 0.0, 0.5, 1.0, and 2.0; becomes respectively 2.0, 1.8, 1.7, and 1.5. This decrease in $D_{box}$ in this range indicates a decrease in irregularity or space-filling properties of the perimeters of the objects.

The DLA-model is useful as a simulation model for a wide range of growth patterns from nature. The fractal dimension $D_{box}$ = 1.7 of the DLA-cluster with $\eta$=1 seems to correspond quite well with the $D_{box}$ of several real growth patterns. It should be noted that $D_{box}$ is only one aspect of the morphology, it is possible to generate objects with a very different overall morphology with equal fractal dimensions. A very interesting property of the DLA-model is that it includes a model of the physical environment on the growth process. Several phenomena that can be observed in experiments with the bacteria *Bacillus subtilus* can be predicted with this model. It can for example be predicted that the shape of the nutrient source (linear, circular etc.) has no influence on the degree of irregularity, as expressed in the value of $D_{box}$, of the colony.

### V.3.2 Diffusion limited growth in continuous space

Diffusion limited growth of aggregates which consist of loose particles can be simulated with the Diffusion Limited Aggregation model. In the DLA model the aggregate is represented in a discrete space by a 2D or 3D lattice. This model is for example very well applicable in simulation models of bacteria colonies, where the individual bacteria can be considered as unconnected particles. In many diffusion limited growth processes structures are formed that can not easily be described as an aggregate of unconnected particles. Examples are radiate accretive growth in sponges and stony-corals [54, 55], growth of ammonium chloride crystals in a aqueous solution [56]. In these cases growth of the object occurs by the addition of new layers on top of previous growth stages, while the previous stages remain unchanged during the growth process. In this type of growth process a layered structure is formed that consists of connected elements, the most natural way of representing this structure is in continuous space. The tip-splitting phenomenon which can be observed in these objects can be modelled by using the local radius of curvature in combination with the local nutrient gradient for determining the growth velocities. In crystal growth the radius of curvature can be related to the surface tension [57]. In biological objects the local radius of curvature can be related to the amount of contact with the environment. When the local radius of curvature becomes too high the amount of contact decreases and locally a shortage in supply of nutrients may occur in the tissue, leading to a decrease in growth velocity.

In this section a 2D and a 3D model is presented in which growth is diffusion limited and where layered structures are formed in an iterative geometric construction and in which the layered objects are represented in continuous space. A more detailed description of the iterative geometric constructions for simulating radiate accretive growth can be found elsewhere [54, 55], in this section only a short outline is given of the 2D and 3D constructions.
In the 2D iterative geometric construction the layered object is represented by tangential edges that are situated on the growth lines of the object and by longitudinal edges which connect the successive layers. The basic construction is shown in Fig. V.14. The tangential elements are equal-sized with length *s* and the length of the longitudinal elements *l* is determined by the growth function:

$$\mathbf{l} = \mathbf{s} \cdot \mathbf{k}(\mathbf{c}) \cdot \mathbf{h}(\mathbf{rad\_curv}) \qquad \text{(V.11)}$$

Both components in the growth function $k(c)$ and $h(rad\_curv)$ will be discussed below. The 2D construction starts with a number of tangential elements arranged in hemi-circle and in each iteration a new layer $j$+1 of tangential and longitudinal elements is constructed on top of the preceding layer $j$. In the construction new longitudinal elements are set perpendicular with respect to a previous tangential element. The neighbouring tangential elements are arranged on a continuous curve. New tangential elements are inserted and deleted when the distance between adjacent tangential elements becomes too large or too small. The length *l* of a new longitudinal element depends on the $h(rad\_curv)$ component in Eq. V.11. In this function a normalised version of the local radius of curvature $rad\_curv$ is returned:

$$\mathbf{h}(\mathbf{rad\_curv} = 1.0 - (\mathbf{rad\_curv} - \min\_\mathbf{curv}) / (\max\_\mathbf{curv} - \min\_\mathbf{curv}) \text{ for } \min\_\mathbf{curv} \le \mathbf{rad\_curv} \le \min\_\mathbf{curv} \qquad \text{(V.12)}$$
$$\mathbf{h}(\mathbf{rad\_curv}) = 1.0 \text{ for } \mathbf{rad\_curv} < \min\_\mathbf{curv}$$
$$\mathbf{h}(\mathbf{rad\_curv}) = 0.0 \text{ for } \mathbf{h}(\mathbf{rad\_curv}) = 0.0$$

The local radius of curvature $rad\_curv$ is estimated from points situated on neighbouring tangential elements. As soon as a certain maximum allowed value of the radius of curvature $\max\_curv$ is exceeded, the growth function in Eq. V.11 returns values below *s* and locally the distance between successive layers *j* and *j*+1 decreases.

Fig. V.14 The 2D construction of a new layer *j*+1 of tangential and longitudinal elements on top of the preceding layer *j*

In the 3D iterative geometric construction the layered object consists of layers of triangles. The tangential elements are represented by the edges of the triangles. In the basic construction (see Fig. 15) a new layer *j*+1 of triangles and longitudinal elements is constructed on top of the preceding layer *j*. The longitudinal elements connect a vertex from layer *j* with the corresponding vertex in layer *j*+1. The direction of a new longitudinal element $(V_{i,j}, V_{i,j+1})$ is determined by the mean value of the normal vectors of the triangles that surround the vertex $V_{i,j}$: the consequence is that the new longitudinal element is set perpendicular with respect to the tangential elements in layer *j*. The length *l* of the longitudinal element is determined by the growth function :

$$\mathbf{l} = \mathbf{s} \cdot \mathbf{k}(\mathbf{c}) \cdot \mathbf{h}_2(\mathbf{low\_norm\_curv}, \mathbf{av\_norm\_curv}) \qquad \text{(V.13)}$$

In the $h_2$ component a series of estimations are made of the local radius of curvature in vertex $V_{i,j}$ in several tangential directions. The estimations are normalised using Eq. V.12. The product of the mean value of the normalised radius of curvature $av\_norm\_curv$ and the lowest value of the radius of curvature $low\_norm\_curv$ determines the final value of $h_2$ in Eq. V.13. The 3D construction starts with a number of tangential elements situated on a sphere and analogous to the 2D model insertion and deletion rules are applied when the distances between adjacent vertices becomes too large or too small.

Fig. V.15 The 3D construction of a new layer *j*+1 of tangential and longitudinal elements on top of the preceding layer *j*

The nutrient distribution model is obtained by mapping the 2D and 3D geometric model onto respectively a lattice of 1000 x 1000 and 100 x 100 sites. In the nutrient distribution model the following assumptions are used: the object consumes nutrient, resulting in a nutrient concentration zero on the object itself and there is a linear source of nutrient by setting the top lattice row of the 2D model and the top plane of the 3D model to the concentration $c = 1.0$. In both lattices the boundary condition $c = 0.0$ is used for the bottom of the lattice. After each step the lattices are used to obtain a solution of the Laplace equation Eq. 8 by finite differencing and successive over relaxation Eq. V.10. In the 2D and 3D model a normal vector, of constant length, is constructed on the surface that 'probes' the nutrient distribution and the local nutrient gradient $k(c)$ along this vector is estimated. The function $k(c)$ is used in both growth functions (Eqs. V.11 and V.13) to determine the growth velocities over the surfaces of the objects. The exponent $\eta$ in Eq. V.14 describes the relation between the local field and the nutrient concentration $c$ and is set to value 1.0.

An example of an object generated in the 2D construction within 180 iteration steps is shown in Fig. V.16. In this figure the successive growth lines can be seen. The nutrient distribution around the object is visualised by displaying the basins with equal nutrient concentration ranges in alternating black and white. The basins situated closer to the object have a lower $c$ range than the basins towards the top of the lattice.

Fig. V.16 Example of an object generated with the 2D construction using the growth function in Eq. V.11 and the basic construction shown in Fig. V.14.

In Fig. V.17 four successive construction stages, generated with the 3D construction, are shown.

Fig. V.17 Four successive construction stages (10, 20, 20, and 40 iteration steps) generated with the 3D construction using the growth function in Eq. V.13 and the basic construction from Fig. V.15.

In the last chapter I will discuss *macrocopic* growth processes that can be observed for instance in population dynamics.

*'The answer to the Ultimate Question of Life, the Universe, and Everything is ... Forty-two,*
*... but what was the question?'*

Douglas Adams

## VI: Modelling macroscopic growth processes (population dynamics)

One of the open questions in modelling and simulation theory is to what extend macroscopical relations can be inferred from systems modelled by microscopic entities. It is not the purpose of this paper to address that, undoubtedly very interesting, question. Some remarks however can be made at this point. We have seen in the chapter on MC simulation and Ising spin systems that by simply modelling a two dimensional world of dynamically flipping spins we can study macroscopic systems such as magnetisation and processes in simple liquids. In later chapters we briefly studied particle models with which we can estimate global thermodynamical properties such as the formation and stable configurations of vesicles, or morphological structures arising from simple non-equilibrium rules applied to small aggregating entities. We note two things: First, it seems that fairly simple microscopical (i.e. particle) systems with simple interaction rules allow for the modelling of macroscopic characteristics and secondly, that the behavioural complexity of a system is often much larger than its structural complexity. The latter is easily seen in systems modelled by simple (linear) differential equations (DE's), where the solutions often are relatively complex. The DE's describe the structure of the system, whereas the solutions describe its (state) behaviour.
Conventionally we identify two paradigms in the modelling of natural systems. One is from the inside out, where we start with fine grain structures, apply some rules and study the large scale behaviour (as in the examples above). This is the most well known paradigm. The other is from the outside in, where we start with the modelling the coarse grain behaviour and simulate the system to get an idea about its finer structures. This is for instance done in behavioural, economical and 'classical' biological and morphological studies. Due to the enormous increase in computational power over the last years, we might expect that these two paradigms will merge in the near future. Fine grain systems can be simulated to the extreme of coarse grain behaviour, and coarse grain models can be simulated down to extreme small time and length scales.
Examples of intermediate models are for example, interacting DLA models, diffusion through porous structures, and lattice gas models filling the gab between molecular dynamics and continuous models.
In this paragraph we will briefly investigate an 'outside in' example. We start with some basic assumptions on simple ecosystem models and simulate the dynamics over time. This approach to modelling is prone to model errors since it is difficult to analyse its constituting parts separately, as is possible with the 'inside out' approach.

The behaviour of the systems we will be studying can either be:

-Continuous steady-state, for t -> ∞ every variable in the system assumes a constant value. (Compare with Class I Cellular Automaton, chapter III)

-Periodic steady-state, some variables remain oscillating.
(Compare with Class II Cellular Automaton)

-No steady-state, transients grow without bounds, the system is unstable.

-Chaotic state, transients do not die out but stay bounded.
(Compare with Class III Cellular Automaton)

VI.1 Predator-Prey models

In this section we look into a classical example of population dynamics modelling as a representative of continuous models. We explore growth and decay processes in simple two species models and look into modelling of crowding, grouping and coorporation effects [14].

Assume we have a population of predators $x_{pred}$ that foray on a population of preys $x_{prey}$. When a predator eats a prey the predator population gets an increment in calories whereas the prey populations is reduced in calories. Furthermore we assume that the predator population dies out when left without prey, while the prey population feeds on an abundant available species (compare to heat bath in thermo dynamics).

The system of differential equations describing this model is the so-called Lotka-Volterra system:

$$\dot{x}_{pred} = -a.x_{pred} + k.b.x_{pred}.x_{prey}$$

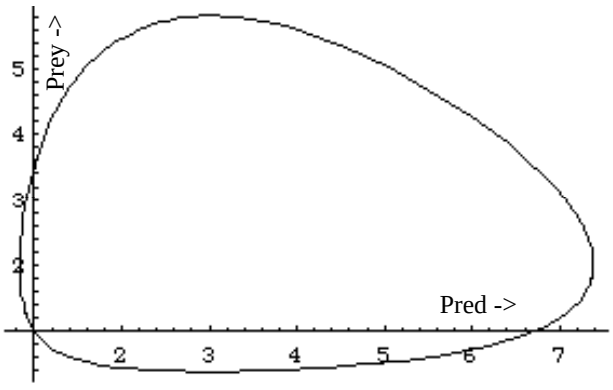
$$\dot{x}_{prey} = c.x_{prey} - b.x_{pred}.x_{prey}$$

With excess death rate of the predator population a > 0, the excess birth rate of the prey population c > 0, the grazing factor
b > 0 and the efficiency factor 0 < k < 1. The latter describes the efficiency of migrating calories by grazing from the prey to the predator population. Understanding the behavioural complexity of this simple model prompts us to look into its transient and steady state behaviour.

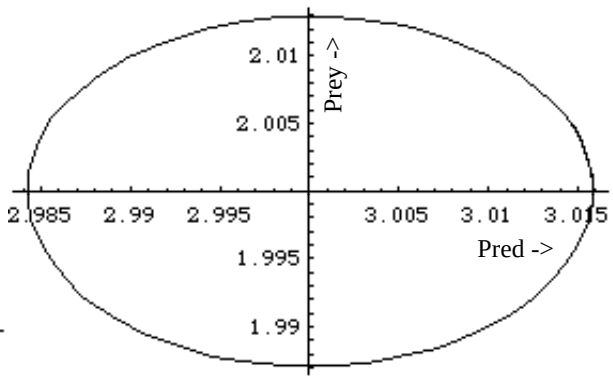
We explore the behaviour of the system by looking into the vicinity of the steady state points (SS). These can be derived by putting all derivatives to zero, resulting in SS= {{0,0} and {3,2}}. Then we can either do some metamodelling by linearising the model, or go directly into more detailed analyses. The next Mathematica™ code supports this more detailed analyses for two different initial conditions (a = 3, b = 1.5, c = 4.5, k = 1).

**Pop_Dyn**[x0_,y0_] = NDSolve**[**{$x_{pred}$'[t] == -3 $x_{pred}$[t] + 1.5 $x_{pred}$[t] $x_{prey}$[t],
$x_{prey}$'[t] == 4.5 $x_{prey}$[t] - 1.5 $x_{pred}$[t] $x_{prey}$[t],
$x_{pred}$[0]== x0, $x_{prey}$[0]==y0},{$x_{pred}$, $x_{prey}$},{t, 0, 2}
**]**

Note that simulation of the Lotka-Volterra model results in periodic steady state values rather than a continuous state value. Furthermore we see that for different initial conditions we obtain different trajectories. Note particularly the behaviour around the Steady State points (0,0) and (3,2) where the system seems to be quenched to a very small region in phase space.




ParametricPlot for [**Pop_Dyn**[1.0, 1.0]




ParametricPlot for [**Pop_Dyn**[3.01, 2.01]

This analyses indicates the so characteristic periodic behaviour of such systems. Next we make the model slightly more advanced by introducing competition and coorporation. When several species compete for the same food source we have a population density dependent competition. This is modelled by a cross product of the competing populations preceded by a negative sign:

$$\dot{\mathbf{x}}_{\mathbf{pred}} = \mathbf{a}.\mathbf{x}_{\mathbf{pred}} - \mathbf{b}.\mathbf{x}_{\mathbf{pred}}.\mathbf{x}_{\mathbf{prey}}$$
$$\dot{\mathbf{x}}_{\mathbf{prey}} = \mathbf{c}.\mathbf{x}_{\mathbf{prey}} - \mathbf{d}.\mathbf{x}_{\mathbf{pred}}.\mathbf{x}_{\mathbf{prey}}$$

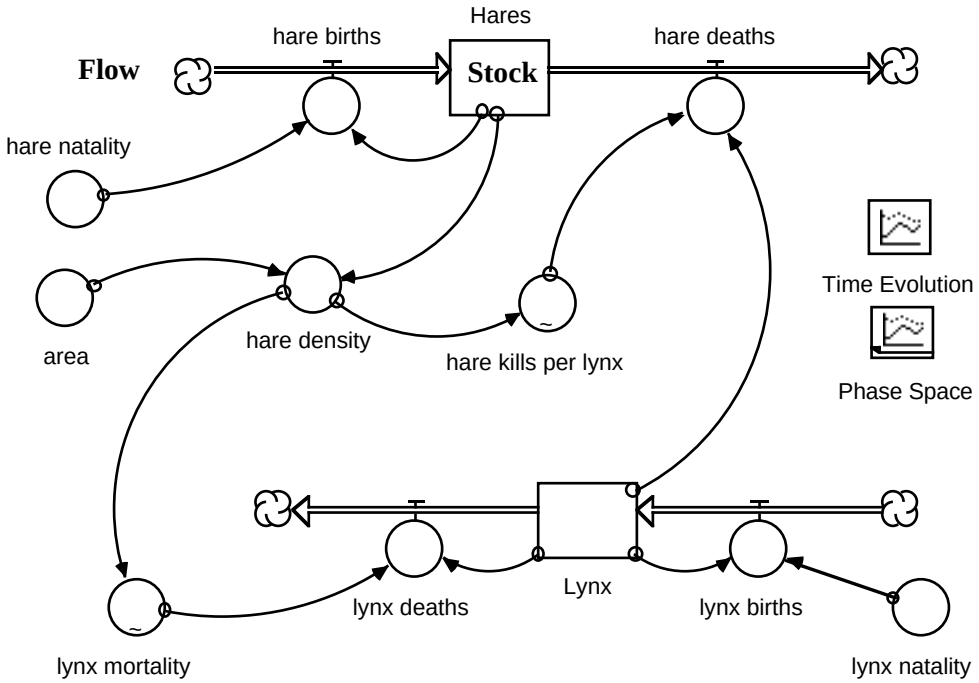
As a consequence the growth of both populations is exponentially but bounded by the competition. An opposite situation arises in case of symbioses among species, each species needs the other. This is expressed by including coorporation in the Lotka-Volterra model:

$$\dot{\mathbf{x}}_{\mathbf{pred}} = -\mathbf{a}.\mathbf{x}_{\mathbf{pred}} + \mathbf{b}.\mathbf{x}_{\mathbf{pred}}.\mathbf{x}_{\mathbf{prey}}$$
$$\dot{\mathbf{x}}_{\mathbf{prey}} = -\mathbf{c}.\mathbf{x}_{\mathbf{prey}} + \mathbf{d}.\mathbf{x}_{\mathbf{pred}}.\mathbf{x}_{\mathbf{prey}},$$

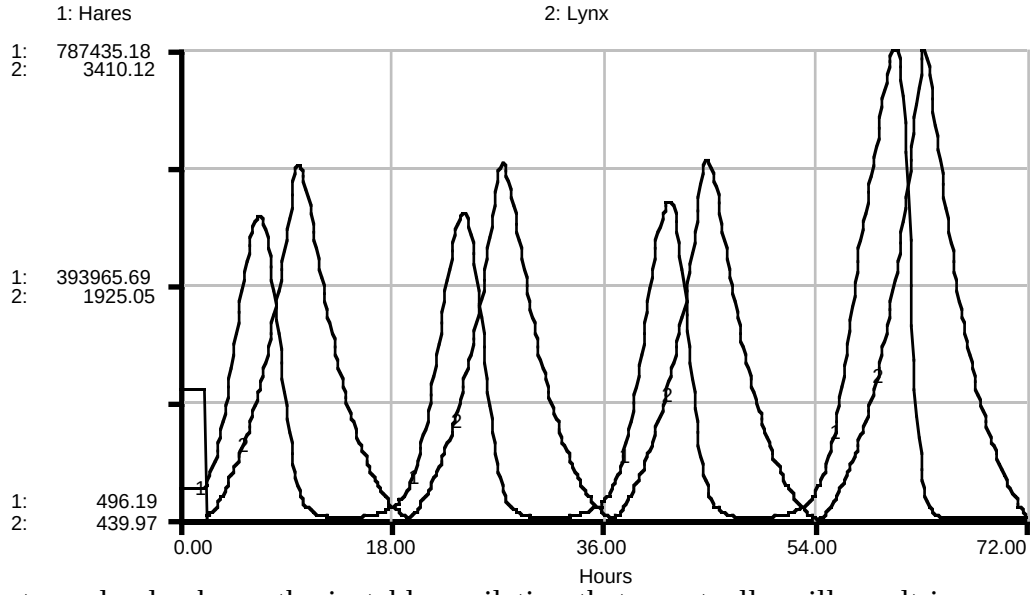
where both populations tend to decay, but through coorporation we allow for a stable equilibrium to be reached.

Finally we can mimic grouping, as an inverse mechanism to crowding. This mimics for instance the positive effects of travelling in herds. A simple model would be: $\dot{\mathbf{x}} = -\mathbf{a}.\mathbf{x} + \mathbf{b}.\mathbf{x}^2$
It is convenient to combine these effects into one population dynamics representation of n-species:
$\dot{\mathbf{x}}_{\mathbf{i}} = \left(\mathbf{a}_{\mathbf{i}} + \sum_{\mathbf{j}=1}^{\mathbf{n}} \mathbf{b}_{\mathbf{ij}}\mathbf{x}_{\mathbf{j}}\right)\mathbf{x}_{\mathbf{i}}, \quad \forall_{\mathbf{i}} \in [1,\mathbf{n}]$, hence all model parameters are expressed in 'a' (birth and death rate) and $b_{ij}$ (grouping, crowding for i=j and competition, coorporation for i≠j).

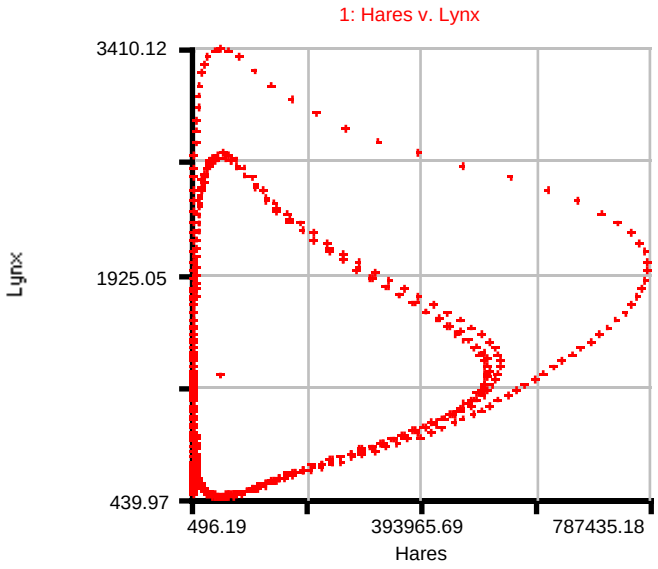
Before we move into models exhibiting more complex behaviour, we first look at a typical simulation of the discussed predator_prey model processed with the system dynamics modelling tool STELLA™ [4].
STELLA™ primarily consists of a object oriented graphical modelling tool that allows for very intuitive modelling of problems from system dynamics. It forces the modeller to think in concepts of flows, stocks and converters. After the modelling phase STELLA™ simulates the model with (relatively simple) integrators that progress the model through time. Nice, instructive animation's allow for monitoring the simulation.
As an instructive example the next figure shows a schematic predator-prey model designed in STELLA™, is highly self-explanatory.



Reading this structure diagram from top to bottom, we start by modelling the influx of hares stimulated by the ‘hare births’, whereas the outflux is determined by the ‘hare deaths’. The arrows indicate relationships such as the dependence of the ‘area’, ‘hare kills per lynx’ and the ‘lynx mortality’ on the ‘hare density’. Note the feedback loop indicated by the arrow from the Hares stock back into the ‘hare births’ flow.
The evolution of the system is driven by a simple rule: `Hares(t) = Hares(t - dt) + (hare_births - hare_deaths) * dt`. With ‘INFLOWS’: `hare_births = Hares*hare_natality`. Where a compounding process is used to depict hare births. The births flow is defined as the product of Hares and their natality. This process works just like compounding interest in a savings account.
‘OUTFLOWS’: `hare_deaths = Lynx*hare_kills_per_lynx`. Where lynx are the resource underwriting the consumption of hares and each lynx has a productivity, given by hare kills per lynx.
An identical description -for the lynx- can be applied to the bottom part of the diagram.
Simulation with this model of a period of 72 hours simulation time, results in the following trajectory:




This trajectory clearly shows the instable oscilation that eventually will result in an exponential unbounded growth pattern.
The behaviour of the two competing species in phase-space is shown in the next figure. Note the tendency to instability.



Next we'll look into a slightly more complicated system. Although it can easily be modelled with Stella™, we will use the primitives of Mathematica™ once again to get a feeling of the rich behavioural characteristics of the relatively simple structural equations describing the system.

## VI.2 Chaotic Behaviour in simple systems

In this section we take the model one step further, to investigate the chaotic state where transients do not die out but still stay bounded. We start by investigating the Gilpin Model [58, 14].
Gilpins equations model one predator 'z' foraying on two different preys 'x' and 'y'. The preys suffer from crowding and competition. The differential system describing the model is given below:

$$\dot{x} = x - 0.001x^2 - 0.001xy - 0.01xz$$

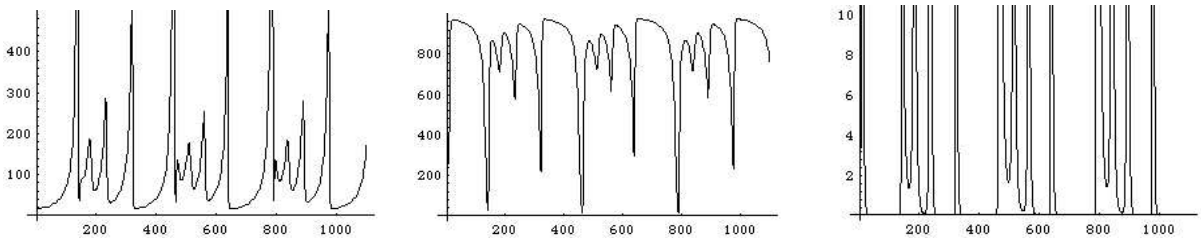
$$\dot{y} = y - 0.0015xy - 0.001y^2 - 0.001yz$$

$$\dot{z} = -z + 0.005xz + 0.005yz$$

Expressed in Mathematica™, for start population sizes of 16, we have for 1100 time steps:

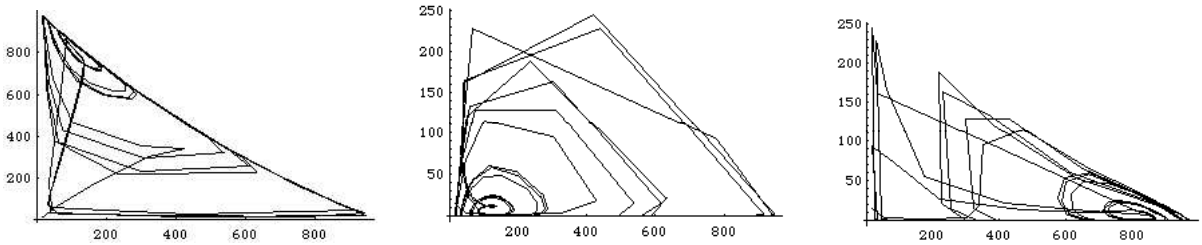
```
Gilpin = NDSolve[{
                x'[t] == x[t] - 0.001 x[t]^2 - 0.001 x[t] y[t] - 0.01 x[t] z[t],
                y'[t] == y[t] - 0.001 y[t]^2 - 0.0015 x[t] y[t] - 0.001 y[t] z[t],
                z'[t] == - z[t] + 0.005 x[t] z[t] + 0.0005 y[t] z[t],
                x[0] == y[0] == z[0] == 16
                }
                , {x[t], y[t], z[t]},{t,0,1100},MaxSteps ->3000
        ]
```

The next figures depict the changes of population x, y, z versus time of a simulation run of this model over 1100 time steps.



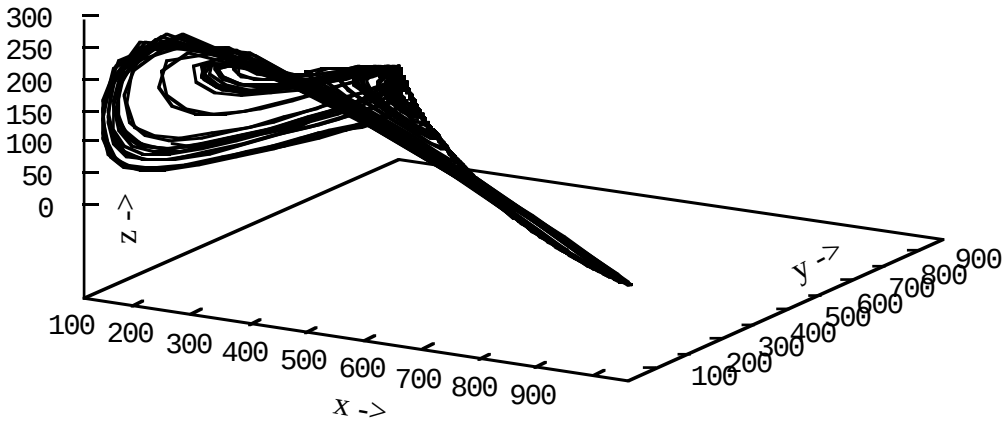

The result of executing 'Gilpin' over 1100 time steps. From Left to right for x[t]. y[t], z[t] versus 't' respectively.
(Mathematica™ instruction: Plot[Evaluate[n[t] /. Gilpin],{t,0,1100}], for n is x,y,z respectively)

From these figures we observe that a strong periodicity occurs for each population, but that after each oscillation the system is in a slightly different state. As mentioned before, this is typical for chaotic behaviour.
The 2-parameter phase portraits indicate regions to which the system most of the time is bounded. Every now a short visit to different parts of the phase space takes place. We observe a strange attractor for small x, large y and small z. A spontaneous growth of z directly leads to much smaller y and smaller x. This is consistent with the choice of the parameters in the Gilpin model.




Phase portraits of respectively x versus y; x versus z and y versus z.

Finally, a closer look into the phase space of the complete system (containing all 3 populations) reveals the combined effects more clearly:




Complete phase space of the execution of ‘Gilpin’ over 5000 time steps.

Inspecting this phase portray we observe that most of the time there seems to be an abundant number of population ‘y’. Every now and then however the predator ‘z’ seems to explode thus reducing ‘y’ dramatically and allowing for ‘x’ to grow fast due to a lack in competition from ‘y’. As a consequence the amount of food for ‘z’, in terms of ‘y’ is diminished and ‘z’ will reduce in size. Then the population of ‘y’ recovers to the dominant species again. It is important to note that although the system moves in cycles, each period is slightly different from the previous one. Here we observe the typical chaotic behaviour where transients do not die out or explode but remain stable and bounded.
Care should be taking when we perform such numerical intensive simulations since we could be observing a numerical artefact where the observed behaviour comes from the integration technique rather than from the implicit characteristics of the system. There are many detailed studies necessary to gain some confidence in the reliability (and numerical stability) of the underlying simulation engine. This is one of the reasons why I hesitated to show this example in Stella™, since the numerical simulation possibilities in that system are limited.

## Final Remarks and Acknowledgements

With this text I tried to share some of the enthusiasm doing research and education in this field. Unfortunately I could only briefly touch upon some of the major topics, and -undoubtedly- missed other relevant and inspiring idea's and work. Especially the possibilities in the use of particle models or, more general, complex system models, in relation to the mapping to massive parallel systems is one of the challenges of the near future. In the end we might even be breaking the intractability for some of the problems that can be expressed is these models [15, 59]. Clearly a fast amount of research has to be initiated and conducted to open up this field. Recent results on the use of hierarchical particle methods in physics however are very promising [[8], [9]]. The mere fact that universities are tuning there education and research programs to allow for more training and exploration in the fields joining physics (or more general: natural sciences) with computer science is a hopeful sign on the wall (e.g. the Physics Computing of Complex Systems programme at Caltech, and numerous other comparable initiatives throughout the world).

I sincerely wish to thank Benno Overeinder, Jeroen Voogd and Jaap Kaandorp for their suggestions and contributions to this document and their assistance given to the accompanying hands-on documents and exercises.

## References

•*Suggested Reading:*

-S.M. Ross, A course in Simulation, Maxwell Macmillian, 1991. (Basic textbook for theory behind Discrete Event Simulation, strong inProbabilistic modelling and simulation, not so strong in languages, 202 pp, ~ $40)
-F. E. Cellier, Continuous System Modelling. New York: Springer-Verlag, 1991.(Modelling techniques for nonlinear systems. 755 pp ~ $80 )
-B. Zeigler, Theory of Modelling and Simulation. New York: John Wiley & Sons, 1976. (Classical text on simulation theory. Not a programming oriented text.)
-A. M. Law and W. D. Kelton, Simulation Modelling and Analysis. New York: McGraw-Hill, first edition 1982; second edition U89/U90. (Monte Carlo and discrete-event simulation. Good textbook on simulation methodology.)
-P. Bratley, B. C. Fox, and L. E. Schrage, A Guide to Simulation. New York: Springer-Verlag, 1983. (Monte Carlo and discrete-event simulation. Good coverage of statistical methodology.)
-H. Gould et al., Computer Simulation Methods I + II, Addison Wesley, 1988
-S. Wolfram, Theory and Application of Cellular Automata, World-Scientific, 1986
-R.W. Hockney et al., Computer Simulations using Particles, Adam Hilger, 1989

•*Journals of Interest:*

TOMACS: Transactions of Modelling And Computer Simulations
The Mathematica Journal
Computers in Physics
Int. Journal of Physics C
Complex Systems
(Transactions of) Simulation

•*Internet sites:*

comp.simulation
comp.theory.cell-automata
mathsource.wri.com (ftp-site)
Cpc@Queens-Belfast.AC.UK (mailer)
risk@csl.sri.com (mailer)

•*References used in this text:*


1] R.W. Hockney and J.W. Eastwood, *Computer Simulations Using Particles* (IOP Publishing Ltd, 1988).
2] S. Wolfram, *Cellular Automata and Complexity* (Addison-Wesley, 1994).
3] P.G. Neumann, "Modeling and Simulation," Communications of the ACM **36**, 124 (1993).
4] H.P. Systems, *Stella II: An Introduction to Systems Thinking*, HighPerformance Systems, 45 Lyme Road Hanover NH 03755, 1992, AppleLink: X0858.
5] F.E. Cellier, "Bond Graphs -The Right Choice for Educating Students in Modeling Continuous-Time Physical Systems," in *International Conference on Simulation in Engineering Education*, Ed. H. Vakilzadian, SCS, 1992, pp. 123-127.
6] M. Weiner and F.E. Cellier, "Modeling and Simulation of a Solar Energy System by Use of Bond Graphs," in *International Conference on Bond Graph Modeling ICBGM '93* , Ed. J.J. Granda and F.E. Cellier, SCS, 1993, pp. 301-306.
7] F.E. Cellier, B.P. Zeigler, and A.H. Cutler, "Object-oriented modeling: Tools and Techniques for capturing properties of physical systems in computer code," in *Computer Aided Design in Control Systems*, InternationalFederation of Automatic Control, Pergamon press, Preprint, 1991.
8] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (MIT Press, Cambridge, MA, 1988).
9] J. Barnes and P. Hut, "A Hierarchical O(NlogN) Force-Calculation Algorithm," Nature **324**, 446-449 (1986).
10] M. Committee on Physical, F.C.S.E. Engineering Sciences, and O.T. Technology, *Grand Challenges: High Performance Computing and Communications*, Document , 1992, Obtain via: Committee on Physical, Mathematical and Engineering Sciences c/o NSF. Computer and Information Science and Engineering 1800G Street, N.W. Washington D.C. 20550, .
11] M. Committee on Physical, F.C.S.E. Engineering Sciences, and O.T. Technology, *Grand Chalanges 1993: High Performance Computing and Communications*, Document, 1993, To Supplement the President's Fiscal Year 1993 Budget.Obtain via: Committee on Physical, Mathematical and Engineering Sciences c/o NSF. Computer and Information Science and Engineering 1800G Street, N.W. Washington D.C. 20550, .
12] C. Rubbia, *Report of the EEC working group on High-Performance Computing*, E.C. Document, feb 1991, .
13] M.R. Garzia, "Discrete event simulation methodologies and formalisms," Simulation Digest **21**, 3-13 (1990).
14] F.E. Cellier, *Continuous System Modeling* (Springer-Verlag, 1991).
15] J.F. Traub and H. Wozniakowski, "Breaking Intractability," Scientific American, 90B-93 (1994).
16] S.M. Ross, *A course in Simulation* (Maxwell Macmillian New York, 1991).
17] R.E. Nance, "A history of discrete event simulation programming languages," ACM SIGPLAN **28**, 1-53 (1993).
18] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," IEEE Trans. Patt. Anal. Mach. Intell. **6**, 721-741 (1984).

19] J.B. Cole, "The statistical mechanics of image recovery and pattern recognition," Am. J. Phys. **59**, 839-842 (1991).
20] T. Bayes, "An essay toward solving a problem in the doctrine of changes," Philosophical Transactions of the Royal Society, 370-418 (1763).
21] P.J.M. Laarhoven, C.G.E. Boender, E.H.L. Aarts, and A.H.G. Rinnooy Kan, *A Bayesian Approach to Simulated Annealing*, Tech. Rept. 8839/A Philips, 1988.
22] S.E. Koonin and D.C. Meredith, *Computational Physics* (Addison Wesley, 1989).
23] H. Gould and J. Tobochnik, *Computer Simulation Methods part I and II* (Addison Wesley, 1987).
24] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," J. of. chem. physics **21**, 1087-1092 (1953).
25] R.J. Gaylord, "Catastrophes in Complex Systems," Mathematica in Education **2**, 19-23 (1992).
26] S. Wolfram, *Mathematica: A system for doing mathematics by computer* (Addison Wesley, 1991).
27] R.J. Gaylord, "The Ising Model," Mathematica in Education **3** (1994).
28] E. Callen and D. Shapero, "A theory of social imitation," Physics Today **27**, 23-28 (1974).
29] P.J.M. Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications* ( D.Reidel Publishing Company, Dordrecht, Holland , Mathematics and its applications, 1987).
30] L. Ingber, "Simulated Annealing: Practice versus Theory," Mathematical Computer Modelling **18**, 29-58 (1993).
31] P.M.A. Sloot, J.M. Voogd, D. de Kanter, and L.O. Hertzberger, *Simulated Annealing: Comparison of Vector and Parallel Implementations*, Tech. Rept. CS-93-06 Computer Systems Technical Report, F.W.I. University of Amsterdam, Kruislaan 403 Amsterdam, October, 1993.
32] A. ter Laak, L.O. Hertzberger, and P.M.A. Sloot, "NonConvex Continuous Optimization Experiments on a Transputer System," in *Transputer Systems-Ongoing Research*, Ed. A.R. Allen, IOS Press, Amsterdam, 1992, pp. 251 -.
33] S. Wolfram, *Theory and Applications of Cellular Automata* (World Scientific, 1986).
34] G. Weisbuch, *Complex System Dynamics* (Addison Wesley, 1991).
35] B. Overeinder and P.M.A. Sloot, "Time Warp on a Transputer Platform: Experiments with Asynchronous Cellular Automata," in *Proceedings of PACTA 92*, 1992.
36] P. Hogeweg, "Cellular Automata for Ecological Modeling," Applied Mathematics and Computation, 81-100 (1988).
37] U. Frisch, B. Hasslachir, and V. Pomeau, "Lattice gas automata for the Navier-Stokes equations," Phys. Rev. Lett. **56**, 1505 (1986).
38] H. Chen, "Discrete Boltzmann systems and fluid flows," Computers in Physics **7**, 632-637 (1993).
39] R.W. Gerling and D. Stauffer, "High speed simulations of ordered and disordered Cellular Automata," Int. J. Mod. Phys. C **2**, 799-803 (1991).
40] P.A. Fishwick, *Building Digital Worlds* (In preparation, 1994).
41] B. Overeinder, L.O. Hertzberger, and P.M.A. Sloot, "Parallel Discrete Event Simulation," in *Proceedings of the Third Workshop Computersystems*, Ed. W.J. Withagen, Facultyof El. Eng., Eindhoven University, The Netherlands, 1991, pp. 19-30.
42] E. Fredkin and T. Toffoli, "Conservative logic," Internat. Theoret. Phys. **21**, 219-253 (1982).
43] N. Margolus, "Physics-like models of computation," Phys. D. **10D:81**, 95- (1984).
44] B.B. Mandelbrot, *The fractal geometry of nature* (Freeman, San Francisco, 1983).
45] L.M. Sander, "Fractal Growth," Scientific American **256**, 82-88 (1987).
46] J. Feder, *Fractals* (Plenum Press New York, 1988).
47] P.M.A. Sloot, A. ter Laak, P. Pandolfi, R. van Dantzig, and D. Frenkel, "Crystallization on a Sphere," in *Proceedings of the 4th international conference Physics Computing '92* , Ed. R.A. de Groot and E.J. Nadrachal, World Scientific Singapore, ISBN: 981-02-1245-3, 1992, pp. 471-472.
48] P.M.A. Sloot, R. van Dantzig, and W.S. Bont, *Stability of spherical bilayer vesicles*, Submitted for publication.
49] J.M. Voogd, P.M.A. Sloot, and R. van Dantzig, "Simulated Annealing for N-body Systems," in *Proceedings of the HPCN 1994*, 1994.
50] L.M. Sander, "Fractal Growth processes," Nature **322**, 789-793 (1986).
51] E. Sander, L.M. Sander, and R.M. Ziff, "Fractals and fractal correlations," Computers in Physics **8**, 420-425 (1994).
52] H. Fujikawa and M. Matsushita, "Bacterial fractal growth in the concentration field of nutrient," J. Phys. Soc. Japan **60**, 88-94 (1991).
53] P. Meakin, "A new model for biological pattern formation," J. Theor. Biol **118**, 101-113 (1986).
54] J.A. Kaandorp, "A formal description of the radiate accretive growth," J. Theor. Biology **166**, 149-161 (1994).
55] J. Kaandorp, *Fractal modelling: Growth and form in biology* (Springer Verlag Berlin, New York, 1994).
56] E. Brener, K. Kassner, and H. Muller-Krumbaar, "," Int. J. mod. Physics C **3**, 825-851 (1992).
57] R. Almgren, "," J. Comput. Phys. **106**, 337-357 (1993).
58] M.E. Gilpin, "Spiral chaos in predator-prey model," The Americal Naturalist **113**, 306-308 (1979).
59] J.H. Reif and R.S. Tate, "The complexity of N-body simulation," in *Automata, Languages and Programming, Proceedings of ICALP 93 , Sweden*, Ed. A. Lingas, R. Karlsson, and S. Carlsson, ICALP, 1993, pp. 162-174.