Computational Models For Complex Systems (6 cfu) 23-24

Notes

University of Pisa

M.Sc. in Computer Science

Contents

Chapter 1

Introduction

1.1 What is a Model of a System?

A model is a simplified and approximate representation of a system, that allows reasoning on the systems properties. **So we construct models in order to understand some aspect of that system**. We include in the model only the aspect of the system that we consider essential, omitting details that would only complicate the analysis.

1.1.1 Understanding the model

After designing our model, then in order to acquire new knowledge, we have to **analyze the model** (like a child play with a toy to understand what it does), then we have to apply some **reasoning** regarding the result given during the analysis.

1.1.2 Errors

All the three steps listed above (Modelling, Analysis, and Reasoning), **can be sources of errors**. The model can be too abstract; the analysis can be too inaccurate; or the interpretation itself could be wrong. **Recall that the result that you get are showing you only something about the real system, you need to generalise what the result means.**

1.1.3 Examples of Models

Planimetries/project and scale models

In architecture planimetries and scale models are used to assest structural properties at design time, in order to evaluate the result in advance.

Life Science

In biology rats (in-vivo model) or a cell colture (in-vitro model) are used as model for the human being.

In Silico Models

In biology there are also computer-based techniques are usually faster and cheaper than in vivo e in vitro models. They simulate the interaction between proteins.

1.1.4 Mathematical Models

Mathematics provides tools for building abstract models of almost everything like **geometry** for Architecture or **differential equations** for Weather. Mathematical models have two advantages:

- They are formal model specification languages, meaning it is not an ambiguous model.
- there are a lot of analytical and numerical methods to analyze model of this type.

1.1.5 Computational Models

Computational Models is similar to Mathematical Models. A Computational Model is a mathematical representation of a dynamical system (systems which evolves over time) taking a computer-executable form.

in Computational Model we are restricting the class of systems of interest to dynamical systems (systems which evolve over time).

Dynamical Systems

Dynamical systems are systems that **evolve** by changing their state over time. Typically the state of a dynamical system is represented by a finite set of variables called **state variables**. In addition we will have some some law/rules/equations that describe how the state variables will vary over time. Our focus will be in prediction or analyze how the state of a dynamic system changes.

There are different types of dynamical systems depending on how the values of the variables change in either a discrete or a continuous way (or both).

Note: time can be interpreted as a discrete or continuous entity.

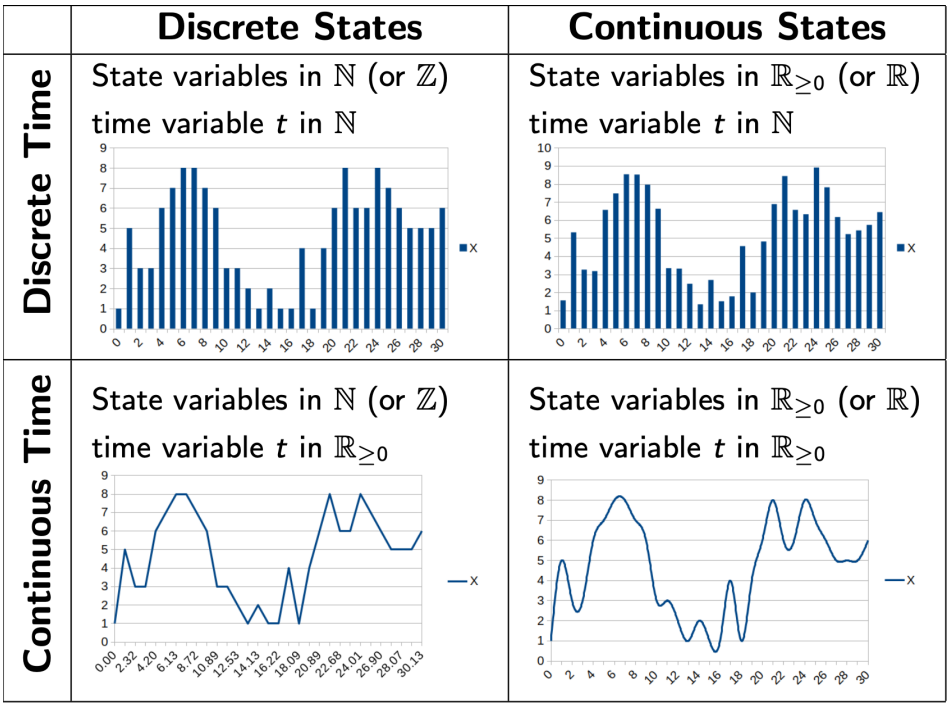


Figure 1.1: All possible types of dynamical systems.

1.1.6 Case of application

What kind of analysis could we do with models of dynamical systems?

- **Reachability of states:** predict the future state of the dynamical system.
- **Behavioral patterns:** the sequence of state that I pass through over time.
- **Effects of perturbations/ control strategies:** once I have a model of a system and I can simulate it, I can try to study how it will behave if I modify it.

1.1.7 How to build a computational models?

There are multiple ways

The Data-driven way

The Data-driven way (e.g. machine learning) consist in starting from the data of the system you want to model and then you apply some machine learning/optimization/whatever method to infer the model automatically. The generated model takes a form suitable for the inference method used. If enough data in available, it often works very well (good predictions), however inferred models are often very difficult to

be interpreted (meaning that we can do good prediction, but we can't explain why they are correct).

The Knowledge-driven way

(also called mechanistic models) consist in trying to reproduce through a mathematical model the internal mechanism of the system in order to reproduce the behaviour and to understand the internals of the systems. It requires limited data, but a good knowledge about the system functioning. Model construction usually requires some effort, and often predictions suffer from approximations but the method generated works also when few data are available; the model is interpretable: it contributes to understanding why a system behaves as observed; modelling allows validation hypotheses on the system functioning.

1.2 What is a complex system?

A complex system is a system consisting of many components (typically with a simple individual behaviours) interacting with each other, from these interaction emerges the global behaviour of the system.

Complex Networks

Complex Networks is a graph with complex structural properties. The dynamics of these networks and its evolution is a field of study (complex networks theory).

1.2.1 Modeling notations for complex systems

Many modeling languages are available for complex systems:

- **mathematics:** Recurrence relations and differential equations.
- **concurrency theory:** we can apply methods seen in the study of concurrent system like Petri nets. Rewrite rules (Multiset rewriting) that describe the different events that happens in a complex systems as rules.
- **artificial life:** approaches proposed to try to reproduce behaviour seen in life. Cellular automata that describe the population as a grid; agents based model in which you explicit as an agent (e.g. an algorithm or set of functions/procedures) and then you can put the agent in a virtual environment to see how they behave together.

1.3 Analyze the model

Modeling languages allow the modeler to express relationships between the state variables of a system and the rules/laws that determine the change of their values over time. The dynamics (or behaviour) of the systems (the actual sequences of states reached by the system over time) can be computed according to the semantics of the modeling language.

1.4 Analyze the behaviour

After the model has been specified, then there are multiple way to analyze the model.

1.4.1 Simulation

You can try to apply some simulation algorithm in order to try to execute that model to have a possible evolution of the system. If the system is deterministic you have only one possible evolution, instead if the system is stochastic-probabilistic you will repeat the simulation several time and get different behaviour of the system. **So simulations can give you only some possible behaviour, not all**. Instead of running simulations to construct some description of the overall behaviour of the systems, it can be done by using **transition systems**.

Transition system

A transition system is a graph (possible infinite) that describe the possible behaviour of a system. A possible transition system is the *Makov chain*

1.4.2 Model Checking

It is an approach that determine whether the whole transition system satisfies a given dynamical property.

1.5 Modeling vs programming

The approach that we will follow to analyze dynamical systems is similar to the approach used to analyze programs

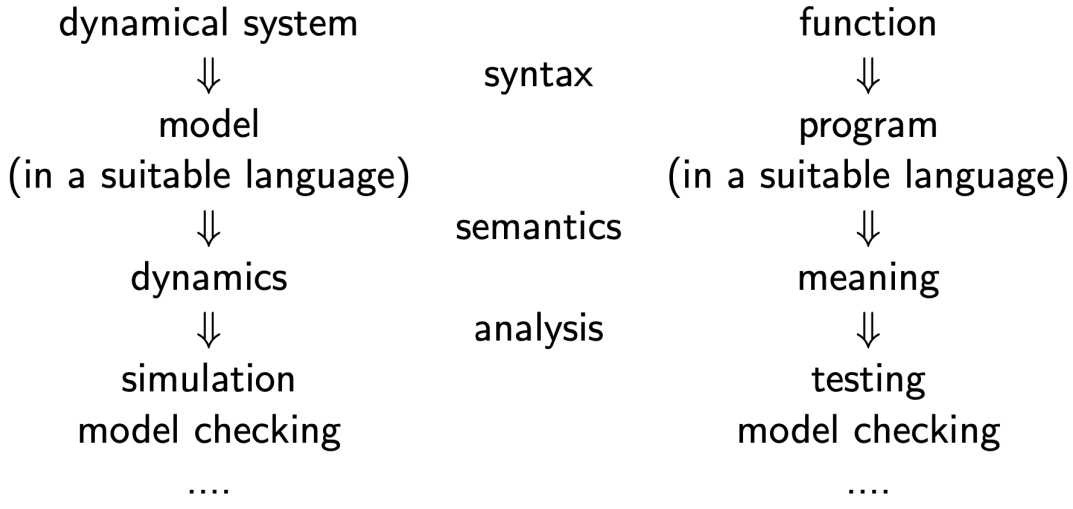


Figure 1.2: Modeling vs programming.

Chapter 2

Discrete Dynamical Systems

They are systems where the evolution is performed using discrete steps, in which the variables describing the state of the system are updated.

2.1 Recurrence relations (Difference equations)

Relations that tell you how from the current state of the discrete systems you obtain the next state of the systems, that is how from the original values the go to the new ones.

In this schema we will consider a generic system that is considered as a **population**(birth/death of individuals). Even the simplest form of interaction between individuals can lead to the emergence of **complex behaviors** (chaotic behavior) in the population.

2.2 Linear Birth Model

The linear birth model is the simplest model describing a recurrence relation.

Given a population, each individual is **indistinguishable** from each other. We denote with `N(t)` the **density of some population** at a time `t`, that is the variable that describe the number of individuals that are part of the population at time `t`.

Given this description we want to predict what will happen to the density of the same population at a future time $t^{'} = t + \Delta t$ assuming that:

- all individuals are indistinguishable from each other.
- there is enough food and space for every individual.
- each individual has λ children every σ time units.

- there is no death occurring in the interval $[t, t + \Delta t)$.
- children do not start reproducing in the interval $[t, t + \Delta t)$.

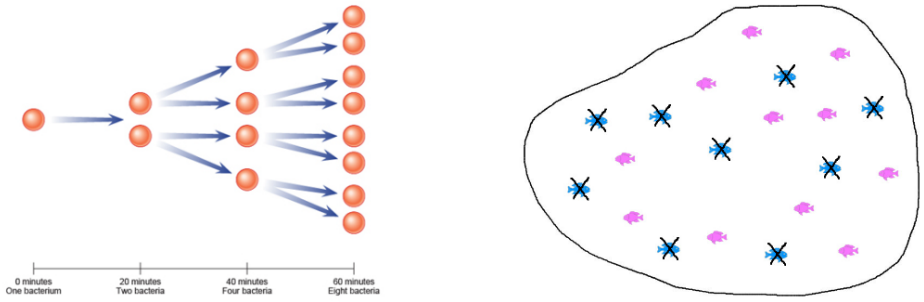


Figure 2.1: Example of populations satisfying the assumption. In the bacteria example in order to assume the no children duplication we set $\Delta t \leq 20 minutes$

2.2.1 Describing the birth process as a relation

The idea is that the size of the population at time $N(t + \Delta t)$ will be $\geq$ than $N(t)$ since we assume **that there is no death occurring in the interval** $[t, t + \Delta t]$. Additionally we know from our assumption that each *adult* individual has λ children every σ time units.

Then, the number of individuals at time $t + \Delta t$ corresponds to the number of individuals at time t, plus the number of newborns at time Δt:

$$N(t + \Delta t) = N(t) + \lambda \frac{\Delta t}{\sigma} N(t)$$

if we group for $N(t)$ then the equation can be rewritten as follows:

$$N(t + \Delta t) = N(t) * (1 + \lambda \frac{\Delta t}{\sigma})$$

Where $\frac{\Delta t}{\sigma}$ **describes the birth moments for every *adult* individual in the interval** $[t, t + \Delta t]$**.**

2.3 Constructing our algorithm

Defined our equation **we can derive a discrete model from it**.

We choose a **time step** (discretization step) that describe an update of the population, in our case **the time necessary for a newborns to be considered an adult so that it can reproduce**, and we set it as Δt.

Instead of representing the equation fully we rewrite it using the **notation of sequence theory** by instead of using the actual value of the time Δt, we just count the steps. Thus we obtain:

$$N_{t+1} = r_d N_t$$

where r stands for *rate*; d stands for *discrete*; r_d represents the **constant birth rate** s.t. $r_d = \lambda \frac{\Delta t}{\sigma}$.

2.3.1 Defining the general term of our equation

Given the recurrence relation we can *sometimes* calculate the **general term**, that is the solution of the recurrence relation. In the case of our linear birth model it is a **non-recursive definition of** N_t.

To do so we first calculate the first terms of N_t: $N_1, N_2, N_3, ...$

$$N_1 = r_d N_0$$

$$N_2 = r_d N_1 = r_d^2 N_0$$

$$N_3 = r_d N_2 = r_d^3 N_0$$

We notice how *it seems* that $N_t = r_d^t N_0$, **but we have to prove it**. To prove the formula, since we are in the realms of the natural numbers, we can use **mathematical induction**:

Base Case (t = 0): $N_0 = r_d^0 N_0$ which is true.

Induction Case (t = k + 1): we assume that the formula is correct for $t = 0$ to $t = k$ and check if it is true for $t = k+1$. We know from assumption that N_{k+1} will be a summation between the previous value for $t = k$ and the newborns in the current step. $N_{k+1} = r_d Nk$ and we know from **induction hypothesis** that $N_k = r_d^k N_0$, thus we can rewrite N_{k+1} as $N_{k+1} = r_d(r_d^k N_0) = r_d^{k+1} N_0$ proving the thesis.

Now having the general term $N_t = r_d^t N_0$ we can compute the solution of our model.

Phase portrait

A way to represent recurrence relations graphically is through **phase portrait**. It consists in putting in the *Cartesian plane*:

- **x-axis**: N_t
- **y-axis**: N_{t+1}

Then in the plane we plot the recurrent relation. First you draw the bisector and you draw the recurrence relation as a function, then by starting from point (N_0, N_0) on the bisector, the other point can be obtained by *"bouncing"* on the curve of the recurrence relation. **You can see how fast the quantity increases (or decreases) over time**.

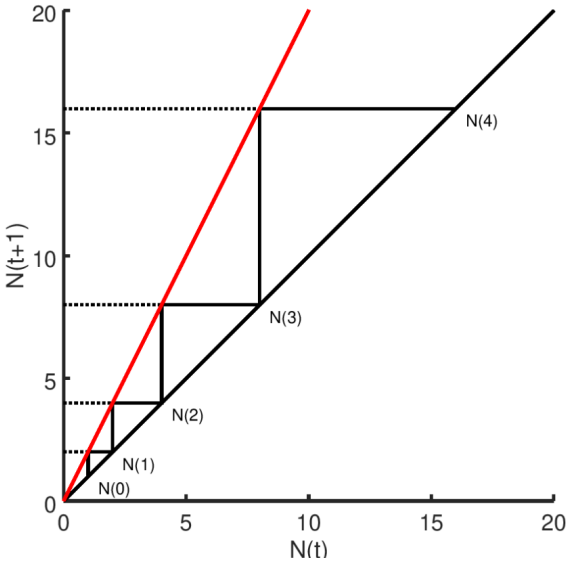


Figure 2.2:

Above an example of phase portrait in our linear birth model.
In red we draw the recurrence equation $N_{t+1} = 2N_t$. in **black** the bisector $N_{t+1} = N_t$.

2.4 Introducing death in our model

We complicate our recurrence relation by considering also deaths, so we are adding a negative term that decrease the number of individuals of our population over time.

Assume that, a constant fraction s_d **of adults that die in every time step** δt. Then our recurrence relation now is:

$$N_{t+1} = r_d N_t - s_d N_t$$

By grouping for N_t we can rewrite the equation as:

$$N_{t+1} = (r_d - s_d) N_t$$

Note: since the number of individuals which die cannot be greater than the whole population, then $0 \leq s_d \leq 1$

Since r_d and s_d are two constant, one positive and one negative, we can group them together in one single constant $\alpha_d = (r_d - s_d)$. We call α_d the **net growth rate**, that is the rate where you discard the individuals who dies. We can rewrite our equations as:

$$N_{t+1} = \alpha_d N_t$$

The difference in respect of the previous case is that α_d will be in general greater than 0, but not necessarily greater than 1.

General trend of our model

There could be many possible cases depending on the value of the constant α_d:

- $\alpha_d > 1$: the overall behaviour of the population is the same as before, **since the population at every steps will increase**.
- $\alpha_d = 1$: the population remains constant, **since the number of newborns always is equal to the number of dead ones**.
- $\alpha_d < 1$: if for instance we assume we have less newborns than the death of old individuals, **then at every step the population reduces**.

2.5 Introducing migration in our model

Independently from the size of the population, **we assume a fixed number of individuals arrive in our region**. We only consider people that arrives, so we model the migration using a constant and positive parameter declared as β.

Then our equation becomes:

$$N_{t+1} = \alpha_d N_t + \beta$$

with $\beta \geq 0$ representing the number of individuals entering the population every Δt time units.

By **mathematical induction** we can prove the general term is:

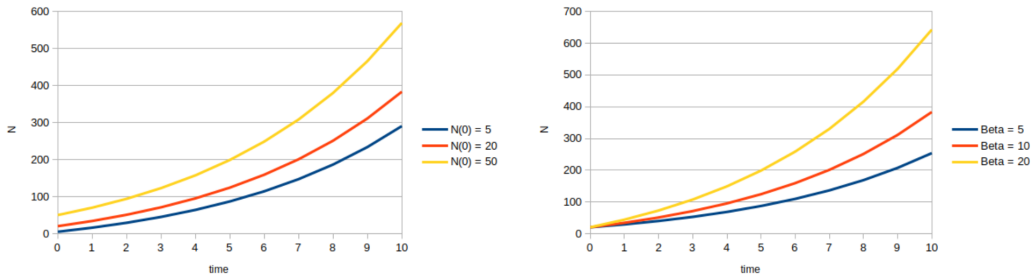
$$N_t = \alpha_d^t N_0 + \sum_{i=0}^{t-1} a_d^i \beta$$

where $\sum_{i=0}^{t-1} a_d^i \beta$ accumulates the β individuals that arrived in the previous step (from 0 to $t-1$) and the children produced by them at every step.

2.5.1 Simulating the migration

Having our general term, we can see what happens by varying α_d, N_0 and β :

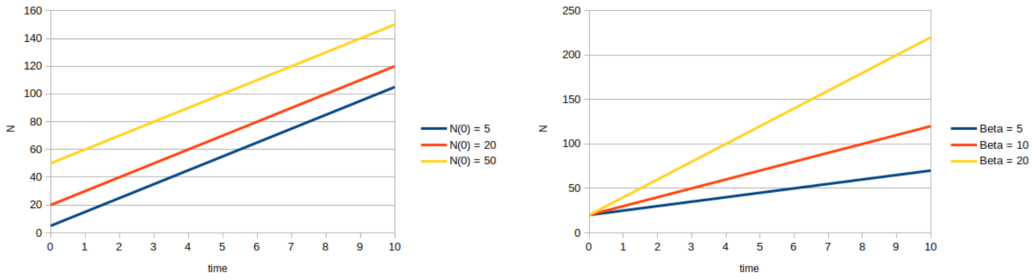
Case $\alpha_d > 1$



$N_0 = 5, 20, 50 \quad \beta = 10$ $\qquad$ $N_0 = 20 \quad \beta = 5, 10, 20$

Figure 2.3: The dynamics is dominated by the birth process (exponential growth).

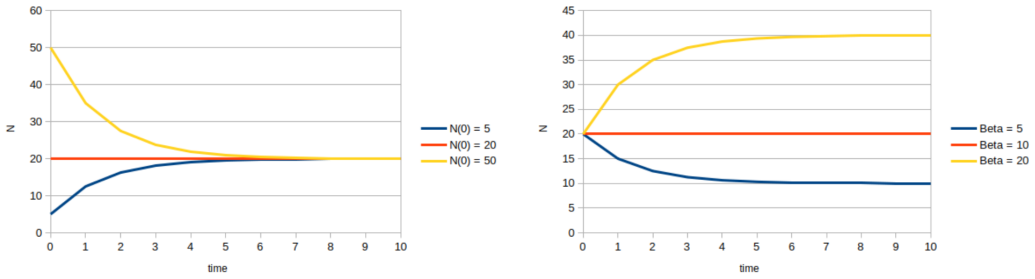
Case $\alpha_d = 1$



$N_0 = 5, 20, 50 \quad \beta = 10$ $\qquad$ $N_0 = 20 \quad \beta = 5, 10, 20$

Figure 2.4: The dynamics is dominated by the migration process (linear growth).

Case $\alpha_d < 1$



$N_0 = 5, 20, 50 \quad \beta = 10$ $N_0 = 20 \quad \beta = 5, 10, 20$

Figure 2.5: The population reaches a dynamic equilibrium: a stable state in which opposite phenomena compensate each other (migration compensates deaths). Note that it is independent from N_0

2.5.2 Computing the equilibrium point

The equilibrium point (also called saturation point) is defined as the moment $t = k+1$ where the computed size of the population is the same as the previous step, that is:

$$N_{t-1} = N_t$$

Knowing that $N_t = \alpha_d N_t + \beta$ we solve the equation in regards to N_t and obtain:

$$N_t = \frac{\beta}{1-\alpha_d}$$

2.5.3 Non-linear models

So far we have described a population where each individual behaves autonomously, which do not fit the definition of complex system (many components with very simple individual behaviour **that interact with each other and the environment**). Introducing interaction between the individuals **requires a non-linear model** (previously we used a linear one). In our (Non-linear) Birth Model now the environment has limited resources such as food and place, thus requiring that the individuals compete **(a form in interaction)** for survival.

2.6 Rewriting our equation

For simplicity assume we are in a closed environment (no migration), then we define K as the **carrying capacity of the environment**, meaning that in the environment there is enough food and space for K individuals.

The population is still governed by the birth rate, but now death is no more a constant, instead is negatively influenced by K:

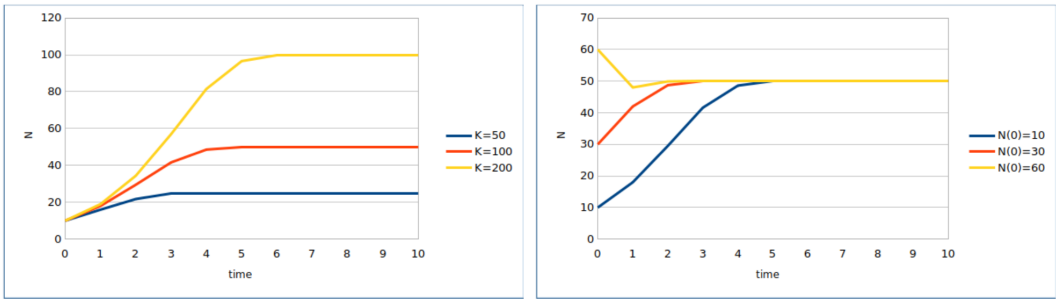
$$N_{t+1} = r_d N_t (1 - \frac{N_t}{K})$$

Note: this non-linear equation is called **logistic equation** (it is an alternative formulation)

So now the birth rate is modulated by the ratio of occupancy on the environment $\frac{N_t}{K}$:

- N_t close to 0 : we have a simple birth process with rate r_d (exponential growth).
- N_t increases: the growth tends to stop.

Eventually the population reaches a **dynamic equilibrium** representing the situation in which environment resources are fully exploited (**saturation**)



$N_0 = 10 \quad K = 50, 100, 200 \qquad N_0 = 10, 30, 60 \quad K = 100$

Figure 2.6: The first graph shows a case where the equilibrium point depends on K. The second graph shows me that the equilibrium point is independent from the initial number of individuals.

2.7 Removing homogeneity

For now we have considered an homogeneous population, but in realistic case we have different individuals with different features and we want to consider each group of individuals. In the system point of view this requires not just a single recurrence equation, but a **system of recurrence equations**.

consider a population of fishes that live in a pond. Each individual can either be a male fish, modelled as M_t or a female fish, modelled as F_t. We consider that a small part of males dies because of fights among them (death rate s_d).

then we construct a system of recurrence equations:

$$\begin{cases} F_{t+1} = r_d F_t (1 - \dfrac{F_t + M_t}{K}) \\ M_{t+1} = r_d F_t (1 - \dfrac{F_t + M_t}{K}) - s_d M_t \end{cases}$$

where:

- $r_d F_t$ represent the number of child born, note that they are generated by females.
- $F_t + M_t$ describes the whole population size (to be related with the carrying capacity K).

2.8 Limitation of discrete dynamical models

Discretization of the system dynamics may introduce inaccuracies: recurrence equations assume that nothing happens during the Δt time that occurs between N_t and N_{t+1}. Adjusting Δt to be smaller usually correspond to more accurate approximations, more precisely we should let Δt tend to 0.

Chapter 3

Continuous Dynamical Systems

3.1 Recap - Why we need to introduce ODEs

As we said in the previous chapter, using a discrete representation for the steps in our model can lead us to lose all the informations that happens between the step N_t and the step $N_t + 1$.

The simplest solution is to make the distance in time Δt between the to step very small (≈ 0), but it is not enough.

3.2 Reconsidering the population model

Recall in our population model that with $N(t)$ we denote the **density of some population** at time t. Our goal is to construct a mathematical model able to predict the density of the same population at a time $t^{'} = t + \Delta t$.

3.2.1 Introducing the Ordinary Differential Equation

Given our equation which is based the model:

$$N(t + \Delta t) = N(t) + \lambda \frac{\Delta t}{\sigma} N(t)$$

And the corresponding **recurrence equation**:

$$N_{t+1} = r_d N_t$$

where $r_d = \frac{\Delta t}{\sigma}$ consider the case where $\Delta t \to 0$. This case cannot be done using discretization, because it can lead to inaccuracies.

To handle the case $\Delta t \to 0$ we make some transformation to our equation:

$$N(t+\Delta t) = N(t) + \lambda \frac{\Delta t}{\sigma} N(t) \rightarrow \frac{N(t+\Delta t)-N(t)}{\Delta t}$$

then by simplifying we get:

$$\frac{N(t+\Delta t)-N(t)}{\Delta t} = \frac{\lambda}{\sigma} N(t)$$

We can recognize that the left hand side can be traced back as the **difference quotient** $\frac{f(x+h)-f(x)}{h}$, **which when taken to the limit as h approaches 0 gives the derivative of the function f**.

Let's consider our equation for $\Delta t \rightarrow 0$:

$$\lim_{\Delta t \to 0} \frac{N(t+\Delta t)-N(t)}{\Delta t} = \lim_{\Delta t \to 0} r_c N(t)$$

with $r_c = \frac{\lambda}{\sigma}$

Note that on the right hand side Δt doesn't appear, thus we can remove the limit; the term on the left hand side is the derivative of $N(t)$, indicated as simply $\dot{N}(t)$. Thus we simply the equation as follows:

$$\dot{N}(t) = r_c N(t)$$

We have defined the dynamics of the system as the derivative equal to a constant multiplied by the value of the function at time t.

This equation is known as Ordinary Differential Equation (ODE).

An ODE has the following properties:

- it relates the function N with its derivative $\dot{N}$
- $t \in \mathbb{R}$, **so time is continuous**

We would like to use the Ordinary Differential Equation to run some simulation or to compute a general solution (like we did for the recurrence relation).

3.2.2 Using the ODE

In some very simple case, like our linear birth model, we can find the general solution by finding a closed-form definition of $N(t)$ satisfying the equation. **A closed-form definition is one that depends only on t and some constant**.

for the linear growth model a solution can be found analytically.

Recall our equation:

$$\dot{N}(t) = r_c N(t)$$

Now let's move $N(t)$ from the right to the left hand-side of the equation:

$$\frac{\dot{N}(t)}{N(t)} = r_c$$

Recall that $\frac{\dot{N}(t)}{N(t)} = \ln N(t)$ and r_c is the derivative of $r_c t + c$ for any constant c, obtaining:

$$\ln N(t) = r_c t + c$$

By resolving our equation in regards to $N(t)$ we finally get:

$$N(t) = Ce^{r_c t + c}$$

where $C = e^c$ (typically C is set to be equal to $N(0)$

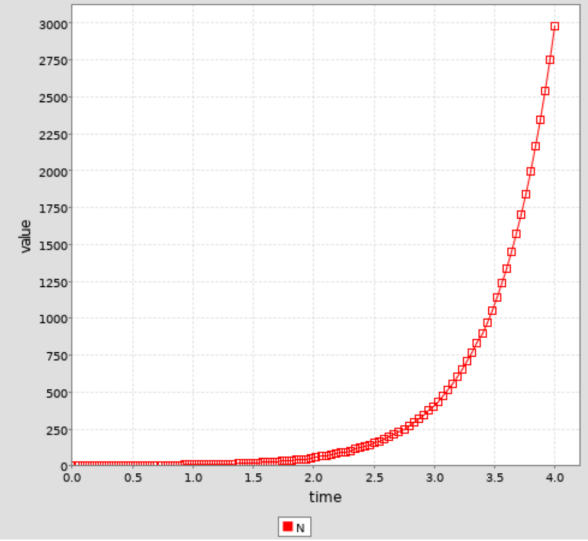


Figure 3.1: This graphs shows us that by setting in our population model $r_c = 2$ and $C = N(0) = 1$ then the population shows an exponential growth over time.

Differences between discrete and continuous model

This behaviour is qualitatively the same as for the discrete model, both showing exponential growth. **What changes is the meaning of the equation**:

- the recurrence relation tells you how to update the variable.
- in our new equation it defines the derivative, **meaning how fast it's changing**.

Note how both equations show rate ≥ 1 only if the birth rate $\frac{\lambda}{\sigma}$ is ≥ 1.

3.3 Example - Radioactive decay

We move from the example of the population model we have seen so far to another one.

Radioactive decay is a process where we have a negative evolution of the population. The idea is that **each molecule decays at a constant rate**, so the whole mass decreases with a rate which is proportional to the mass itself.

We can describe this model by using the following Ordinary Differential Equation:

$$\dot{N}(t) = -d_c N(t)$$

then if we solve it in regards to $N(t)$ we get:

$$N(t) = N(0)e^{-d_c t}$$

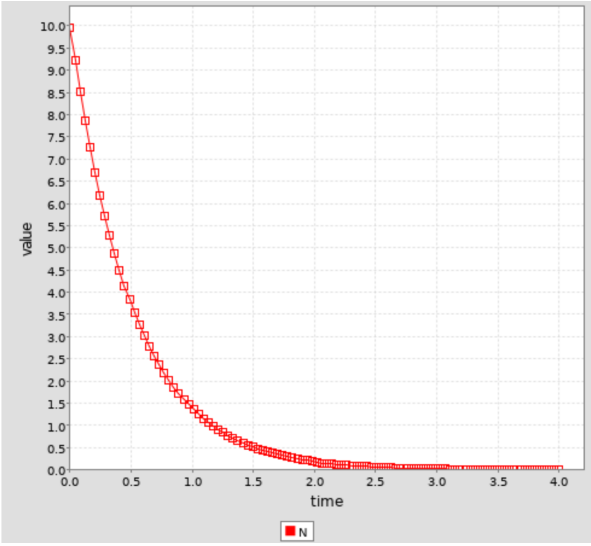


Figure 3.2: Example of applying the ODE of the radioactive decay. Note that by setting $d_c = 2$ and $C = N(0) = 10$ we get $N(t)$ to tend to zero. This is caused by the negative exponent in the ODE.

3.4 Continuous version of the logistic equation

Given the non-linear logistic equation, we can define as follows its continuous version:

$$\dot{N}(t) = r_c N(t)(1 - \frac{N(t)}{K})$$

where:

- r_c is the **continuous growth rate**.

- K is the **carrying capacity** of the environment.

Resolving the ODE in regards to $N(t)$ we get:

$$N(t) = \frac{K}{1+(\frac{K}{N(0)}-1)e^{-r_c t}}$$

We notice that when $N(t) \to K$ **the population converges to the carrying capacity of the environment**.

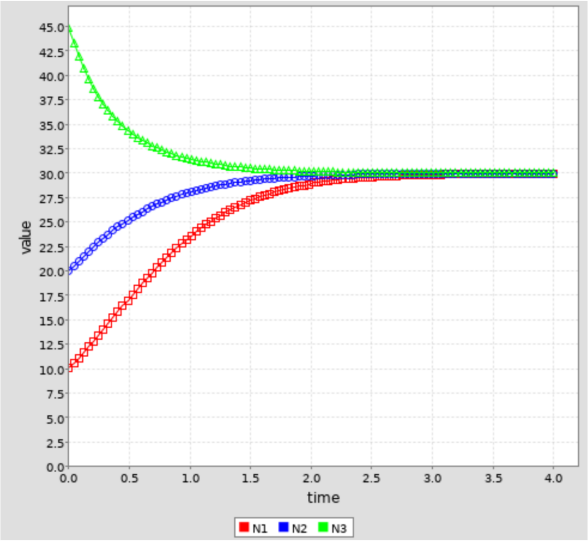


Figure 3.3: Example of the continuous logistic equation. Note how by putting $r_c = 2$, $N(0) = 10$, $K = 30$, the population will eventually converge to the carrying capacity of the environment.

The trend is the same as for the discrete case.

3.5 Systems of ODE

Now let's consider a population of males, indicated as $M(t)$, and females, indicated as $F(t)$. Assume that males fight with each other, so a small part of them dies because of it with a death rate of s_c.

Thus we have to expand our model considering a system of ODEs:

$$\begin{cases} \dot{F}_t = r_c F_t(1 - \dfrac{F_t + M_t}{K}) \\ \dot{M}_t = r_c F_t(1 - \dfrac{F_t + M_t}{K}) - s_c M_t \end{cases}$$

where:

- $r_c F(t)$ is used for both genders, **since both are generated by females**.

- $F(t) + M(t)$ describes the whole population size.

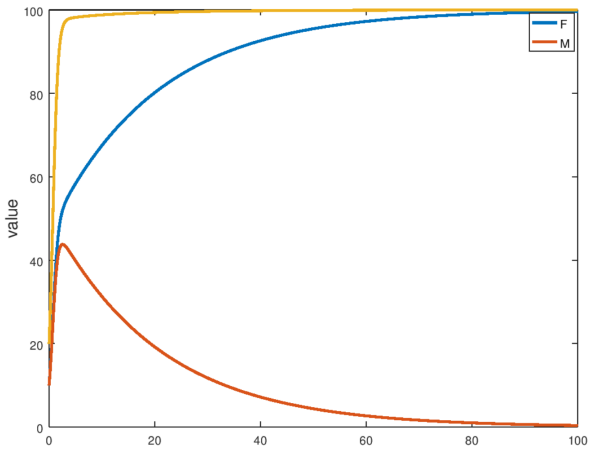


Figure 3.4: Example of the system of ODEs.

What happens in this scenario, after some times you have only females in the population. This shows a completely different behaviour from the discrete case, **because the meaning of their equations is different**:

- The recurrence relations indicates us the size of the population.
- The system of ODEs describes how fast is changing.

3.6 Numerical Solution of ODEs

Recall that we said that computing the solution of an ODE is not always possible. For this reason we prefer working using an approximation of the solution, indicated as **numerical solver** (or numerical simulator). This approximation doesn't compute the general function, **instead it solves the initial value problem, also called Cauchy problem**.

Initial Value Problem

Given an ODE in the form $\dot{N}(t) = f(N(t))$ and an initial value N_0 s.t. $N(0) = N_0$, compute a function $F(t)$ that is a solution of the ODE and s.t. $F(0) = N_0$.

In our case we are interested only in studying the values of $F(t)$ where $t \geq 0$, **hence we perform a numerical simulation starting at** $t = 0$.

3.7 The Euler method

The Euler method is the simplest numerical simulation method available. The main concept behind this approach is to approximate the given continuous system with a recurrence relation specifically designed to approximate the continuous differential equation. **The idea is that, since we know the derivative, we use it as it was the function**.

It is based on the idea of discretizing the dynamics of differential equations by time steps of constant length $\tau = \Delta t$.

Given an Ordinary Differential Equation in the form:

$$\dot{N}(t) = f(N(t))$$

this corresponds to approximating its solution with the following recurrence relation (assuming $N_0 = N(0)$)

$$N_{k+1} = N_k + \tau f(N_k)$$

where $N_k \approx N(\tau k)$

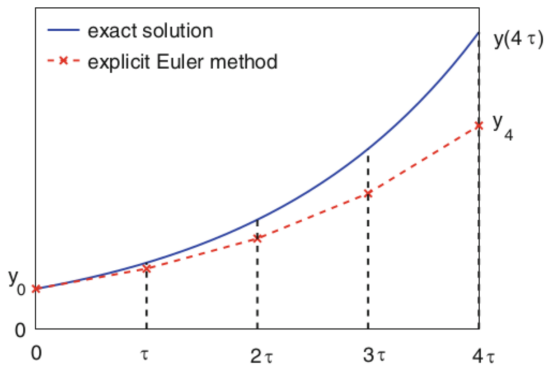


Figure 3.5: A graph showing the correct solution compared to the approximation generated by the Euler method.

3.7.1 Errors in the Euler method

By approximating in each step the Euler method makes local errors that will contribute to a global error at the end of the whole simulation.

The local discretization error is computed as $|N(\tau) - N_1|$ and it is in the order of $O(\tau^2)$ *(the motivation is because we truncate the Taylor Series at the first step).*

The Global discretization error is obtained by accumulate all the local discretization errors after k steps, namely at time $t = k\tau$. The global error is computes as $|N(k\tau) - N_k)$ and is in the order $O(k\tau^2) = O(\tau)$ since $k\tau = t$ is constant.

3.8 Other numerical simulation methods

A linear error of $O(\tau)$) is quite annoying, requiring us to set $\tau \approx 0$ in order to have something acceptable, making the computation very slow (it requires a lot of steps).

To overcome this other methods have been proposed, having a global discretization error of a higher order (e.g. $O(\tau^p)$ for some p) which is better as long as $\tau \to 0$ (hence $\tau < 1$). To work these method use more than one point to approximate the value of the function, making a single step require more time, but maintain the error inside some defined boundaries.

A few examples of such methods are:

- **Runge-Kutta methods:** $p = 2$ in the original formulation, but can be higher
- **Multistep methods (e.g. Adams methods):** extrapolate the value of the next step from the values of the previous k steps obtaining $p \approx k$.

State-of-art methods can also:

- **s**elf-determine the step size τ based on thresholds on local and global discretization errors.
- **d**ynamically adjust the step size τ during their executing (e.g. Adaptive Runge-Kutta)

3.9 Instability and stiff systems

There is another problem that could arise: if you have a derivative which cause the value of a variable in a very fast way, if τ is not small enough you not only pay some error but you could have that your computed solution is unstable.

What we mean for unstable is that we lose informations and your points start to oscillate creating chaos.

This kind of problematic systems are called **stiff systems**. There is no clear definition of stiffness: intuitively contains a very fast term that cannot be captured by our method, worse if we have in our systems also slow terms to take into account.

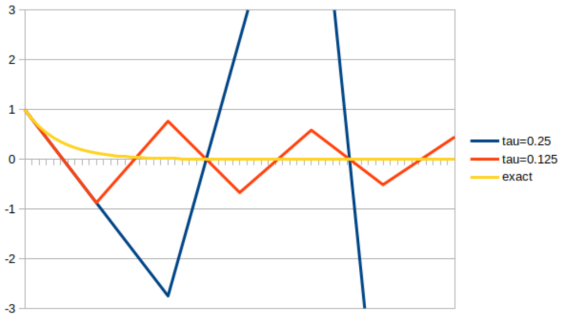


Figure 3.6: An example of a stiff system. In orange is shown the real solution, while in red and blue the one obtained by setting τ to different values

In this cases you need to consider alternative numerical simulation methods called **implicit numerical simulation methods**. All the methods we have seen so far have and implicit version, which requires more computation time for step.

3.10 Implicit Euler method

The idea of the implicit version of the Euler method is that, like the previous version we approximate the function by using its derivative, **but the derivative is not computed on the value N at step k, but at the step $k+1$.**

By using this approach we are no longer defining N_{k+1} in terms of N_k, but as an equation where N_{k+1} is in both sides:

$$N_{k+1} = N_k + \tau f(N_{k+1})$$

where $N_k \approx N(k\tau)$.

There are methods from numerical analysis that are able to solve these type of equations. This method requires more effort for computing a single step, but **often** the local discretization error is smaller permitting us to use greater values of τ.

3.11 Other implicit methods

Implementations of implicit methods that use more than one variables require the modeler to provide the **Jacobian matrix** (partial derivatives) of the function f. Most are able to compute the Jacobian matrix autonomously by doing some approximation.

There exist methods that are able to automatically switch from explicit to implicit methods by determining if the system is stiff.

Bibliography