

# Continuous Dynamical Systems

## Computational Models for Complex Systems

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa

<http://pages.di.unipi.it/milazzo>

milazzo@di.unipi.it

Laurea Magistrale in Informatica

A.Y. 2018/2019

# Introduction

We will see how to define **ordinary differential equations** (ODEs) in order to model the dynamics of systems whose **state changes continuously**.

- focus on population models (birth/death of individuals)

See also:

- **Notes on a Short Course and Introduction to Dynamical Systems in Biomathematics** by Urszula Foryś  
Available on the course web page
- Chapter 2 of **A Guide to Numerical Modelling in Systems Biology** by Peter Deuffhard and Susanna Roblitz  
Freely accessible if you are within the UniPi subnet  
Link available on the course web page
- **Mathematical and Computer Modelling of Nonlinear Biosystems** by Urszula Foryś  
Available on the course web page

## Linear birth model, again

Let  $N(t)$  denote the **density of some population** at time  $t$ .

We want to construct a mathematical model able to **predict** the density of the same population at time  $t + \Delta t$ , that is  $N(t + \Delta t)$ .

Assume that:

- **all** individuals are **the same** (no distinction by gender, age, ...)
- there is **enough food and space** for every individual
- each individual has  $\lambda$  children every  $\sigma$  time units
- there is **no death** in the interval  $[t, t + \Delta t)$
- children **do not start reproducing** in the interval  $[t, t + \Delta t)$

## Linear birth model, again

We have seen that the model can be based on the following equation:

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

from which we derived the discretized model corresponding to the following recurrence equation (with step  $\Delta t$ ):

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

where  $r_d = \lambda \frac{\Delta t}{\sigma}$ .

We have seen that **discretization** can lead to **inaccuracies**

- Let's consider  $\Delta t \rightarrow 0$

## Linear birth model, continuous version

The equation

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

can be rewritten as

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

so that the left hand side turns out to be a **difference quotient**.

Now, let's consider the limit for  $\Delta t \rightarrow 0$

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

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

## Linear birth model, continuous version

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

The term on the left turns out to be the **derivative** of  $N(t)$

- denoted  $\dot{N}(t)$ , or  $\frac{dN}{dt}$

The term on the right does not depend on  $\Delta t$ . So, we obtain:

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

This is a so called **Ordinary Differential Equation (ODE)**

- relates the function  $N$  with its derivative  $\dot{N}$
- **time is continuous**:  $t$  can take any value in  $\mathbb{R}$

## Linear birth model, continuous version

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

Finding a **solution** of the differential equation corresponds to finding a **closed-form definition** of  $N(t)$  satisfying the equation

- a definition that depends only on  $t$  and some constants

For the linear growth model a **solution** can be found **analytically**.

Let's rewrite the equation as follows:  $\frac{\dot{N}(t)}{N(t)} = r_c$

Since  $\frac{\dot{N}(t)}{N(t)}$  is the derivative (w.r.t.  $t$ ) of  $\ln N(t)$  and  $r_c$  is the derivative of  $r_c t + c$  for any constant  $c$ , we obtain

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

that gives:

$$N(t) = C e^{r_c t}$$

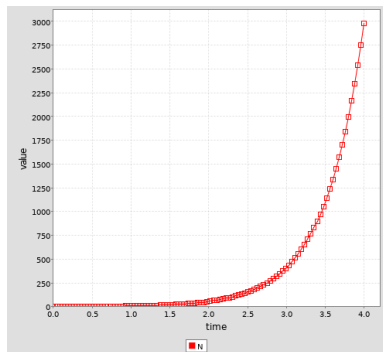
with  $e$  the Euler number and  $C = e^c$  (typically  $C = N(0)$ ).

## Linear birth model, continuous version

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

The solution of the ODE tells us that the population shows an exponential growth over time

- $r_c = 2$
- $C = N(0) = 1$





## Linear birth model, continuous version

This behaviour is **qualitatively the same** as that of the discretized model

- the population exhibits an **exponential growth**

What changes is the role of the growth rate:

**discrete model:**

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

general term:

$$N_t = r_d^t N_0$$

**continuous model:**

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

solution:

$$N(t) = C e^{r_c t}$$

The population grows if  $r_d > 1$

$$r_d = 1 + \lambda \frac{\Delta t}{\sigma}$$

$$\text{hence: } \frac{\lambda}{\sigma} > 0$$

The population grows if  $r_c > 0$

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

$$\text{hence: } \frac{\lambda}{\sigma} > 0$$

## “Radioactive” decay

This example describes spontaneous **decomposition** (or decay, degradation) of substances

- it is called “radioactive” since this has been considered initially for substances for which radioactivity can be measured

The idea is **each molecule decays at a constant rate**. So, the whole mass decreases with a rate which is proportional to the mass itself.

This is described by the following ODE:

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

whose solution is

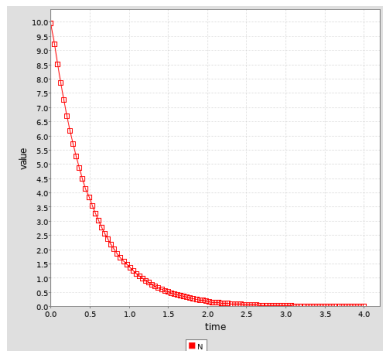
$$N(t) = N(0)e^{-d_c t}$$

## “Radioactive” decay

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

The negative exponent causes  $N(t)$  to tend to zero

- $d_c = 2$
- $C = N(0) = 10$



## Logistic equation, continuous version

A continuous version of the **logistic equation** can be defined as follows:

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

where  $r_c$  is the continuous growth rate and  $K$  is the **carrying capacity** of the environment

A **solution** of this ODE is

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

The solution tells us that  $N(t)$  tends to  $K$ , since  $e^{-r_c t}$  tends to 0

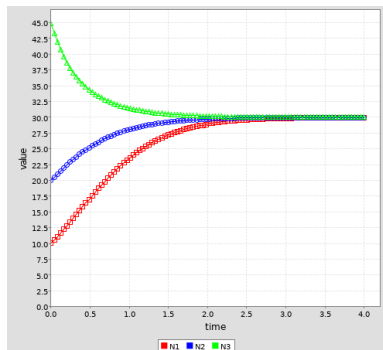
- **the population converges to the carrying capacity of the environment**

## Logistic equation, continuous version

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

The population converges to the carrying capacity of the environment

- $r_c = 2$
- $N(0) = 10$
- $K = 30$



## Systems of ODEs

We considered examples of systems described by a single variable  $N(t)$

When more than one variable has to be considered, we have to construct a **system of ODEs**

Let's consider a population of males and females, with fights among males

- $F(t)$  models females and  $M(t)$  models males
- assume a small part of males die because of **fights** among them (death rate  $s_c$ )

We obtain the following system of ODEs

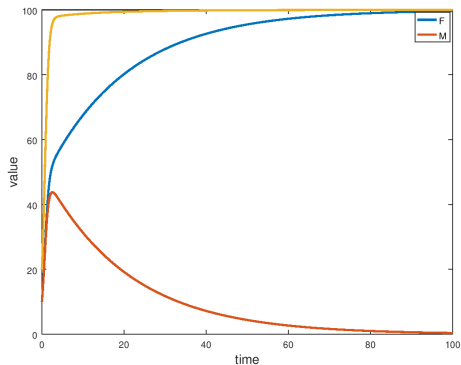
$$\begin{cases} \dot{F}(t) = r_c F(t) \left(1 - \frac{F(t)+M(t)}{K}\right) \\ \dot{M}(t) = r_c F(t) \left(1 - \frac{F(t)+M(t)}{K}\right) - s_c M(t) \end{cases}$$

where

- $r_d F(t)$  is used for both genders since both are generated by females
- $F(t) + M(t)$  describes the whole population size (to be related with the carrying capacity  $K$ )

# Systems of ODEs

## Dynamics of the systems of ODEs

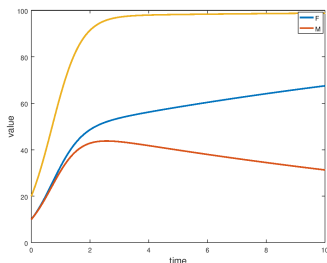
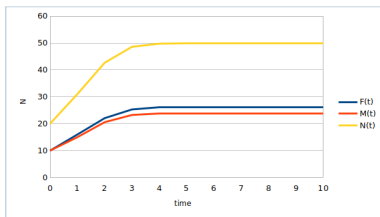


# Recurrence Equations vs ODEs

Why the dynamics is so different from that of the recurrence relations we have seen in the previous lesson?

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

$$\begin{cases} \dot{F}(t) = r_c F(t) \left(1 - \frac{F(t) + M(t)}{K}\right) \\ \dot{M}(t) = r_c F(t) \left(1 - \frac{F(t) + M(t)}{K}\right) - s_c M(t) \end{cases}$$





# Recurrence Equations vs ODEs

- **Recurrence relations** describe how to compute **the next state**
- **ODEs** describe derivatives: (the limit of) **the difference** between the current and the next state

Steady states are computed differently:

$$\begin{array}{l} \left\{ \begin{array}{l} F_t = r_d F_t \left(1 - \frac{F_t + M_t}{K}\right) \\ M_t = r_d F_t \left(1 - \frac{F_t + M_t}{K}\right) - s_d M_t \end{array} \right. \quad \left\{ \begin{array}{l} 0 = r_c F(t) \left(1 - \frac{F(t) + M(t)}{K}\right) \\ 0 = r_c F(t) \left(1 - \frac{F(t) + M(t)}{K}\right) - s_c M(t) \end{array} \right. \\ \Downarrow \\ M_t = F_t - s_d M_t \quad \quad \quad 0 = 0 - s_c M(t) \\ \Downarrow \\ F_t = (1 + s_d) M_t \quad \quad \quad \Downarrow \\ M(t) = 0 \end{array}$$

## Numerical solution of ODEs

Unfortunately, computing the solution of an ODE is not always possible/simple

Very often, ODEs are studied by using **numerical solvers** (or **numerical simulators**)

- Numerical solvers do not compute the general function obtained by “integrating” the ODE
- They usually solve the **initial value problem** (or Cauchy problem)

### Definition: Initial value problem

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

Actually, what we are usually interested in, are the values of  $F(t)$  for  $t \geq 0$

- hence, we want to perform a **numerical simulation** starting from  $t = 0$

# The Euler method

The **Euler method** is the simplest numerical simulation method

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

At each step, the **solution** of the differential equation is **approximated by its derivative** computed at the **beginning** of the time interval of length  $\tau$

Given an ODE

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

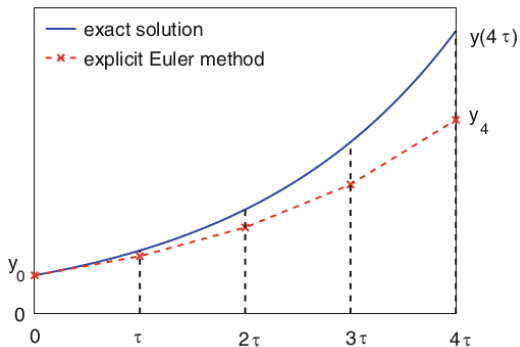
this corresponds to approximating its solution with the following **recurrence relation** (with  $N_0 = N(0)$ )

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

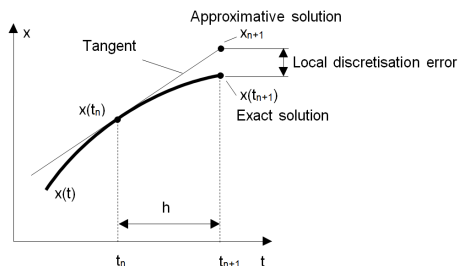
where  $N_k$  approximates  $N(k\tau)$

# The Euler method

Example of execution of the Euler method:



# The Euler method: errors



Each step of the Euler method give rise to an error

- **Local discretization error** (or local truncation error)  $|N(\tau) - N_1|$
- it is in the order  $O(\tau^2)$

Errors accumulate: after  $k$  steps, namely at time  $t = k\tau$ , we have

- **Global discretization error** (or global truncation error)  $|N(k\tau) - N_k|$
- it is in the order  $O(k\tau^2) = O(\tau)$ , since  $k\tau = t$  is constant

## Other “explicit” methods

A linear ( $O(\tau)$ ) global discretization error often imply that a very small discretization step  $\tau$  has to be used

- the computation becomes very slow (many steps)

Other methods have a global discretization error of a higher order (e.g.  $O(\tau^p)$  for some  $p$ ) which is better as long as  $\tau \rightarrow 0$  (hence, it is smaller than one)

A few examples of such methods:

- **Runge-Kutta methods:**  $p = 2$  in their 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.  $p \approx k$

## Other “explicit” methods

State-of-art methods can also:

- **self-determine** the step size  $\tau$  based on thresholds on local and global discretization errors
- **dynamically adjust** the step size  $\tau$  during their execution (e.g. Adaptive Runge-Kutta)

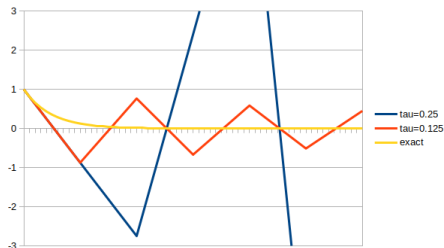
## Instability and stiff systems

In some cases, explicit methods may become **unstable**, unless a **very small** step size  $\tau$  is used.

Example:

$$\dot{N}(t) = -15N(t)$$

with  $N(0) = 1$



Euler method with  $\tau = 0.25$  and  $\tau = 0.125$   
compared with the exact solution



# Instability and stiff systems

This kind of problematic systems are called **stiff systems**

There is no precise definition of **stiffness**

- Intuitively, the system contains some **very fast** term which causes very small step sizes to be used with explicit methods
- Sometimes, the combination of fast terms with slow terms in a system of ODEs make explicit methods unstable although the shape of the solution is smooth

# Implicit Euler method

**Implicit methods** are often better suited for stiff systems.

Implicit variant of the Euler method:

At each step, the solution of the differential equation is approximated by its derivative computed **at the end** of the time interval of length  $\tau$

Given an ODE

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

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

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

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

# Implicit Euler method

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

In this case the value of  $N_{k+1}$  is not explicitly expressed in terms of  $N_k$ , but it is **implicitly expressed by means of an equation** which could be **difficult to solve**

Hence, the computation of the single step requires **more effort** in implicit methods

But, often the local discretization error is smaller

- **greater values of  $\tau$  can be used**
- with **stiff** systems it is **often convenient** to pay the extra time for the computation of each step

## Other “implicit” methods

A few examples of implicit methods:

- Implicit Runge-Kutta methods
- Multistep methods (e.g. BDF methods)

Implementations of implicit methods often require the modeler to provide the **Jacobian matrix** (partial derivatives) of the function  $f$

Also in these cases there are variants that can **self-determine** and **dynamically adjust** the step size  $\tau$  according to threshold on the local and global discretization errors.

There exist methods that are able to automatically **switch from explicit to implicit modes** (and vice-versa)

- LSODE: switches between Adams and BDF
- CVODE: switches between Adams and BDF

# Implementations

ODE solvers are available inside the main environments for numerical and mathematical computing (e.g. MatLab, Octave, Mathematica) and as libraries for the main programming languages (e.g. C, C++, Python)

Some implementations:

- `ode45`, `ode113`, `ode15s`, ... in MatLab  
see <https://www.mathworks.com/help/matlab/ordinary-differential-equations.html>
- `lsode` in Octave
- Sundials CVODE in C
- `odeint` in C++
- `scipy.integrate.odeint` and `scipy.integrate.ode` (interface to multiple solvers) in Python

## Using the Octave solver

Let's see, for example, the **Octave implementation** of this system of ODEs

$$\begin{cases} \dot{A}(t) = 320s(A(t) - A(t)B(t) + B(t) - qA(t)^2) \\ \dot{B}(t) = 320(C(t) - B(t) - A(t)B(t))/s \\ \dot{C}(t) = 320w(A(t) - C(t)) \end{cases}$$

with  $s = 77.27$ ,  $q = 8.375 \cdot 10^{-6}$ ,  $w = 0.161$ .

This is (a variant of) a well-known example of **stiff** system (the Oregonator)

# Using the Octave solver

```
% workaround for scripts starting with a function definition
1;

%% Function computing derivatives
function dX = dX(X,t)

    % model parameters
    s = 77.27;
    q = 0.000008375;
    w = 0.161;

    % variables used for the sake of readability
    A = X(1);
    B = X(2);
    C = X(3);

    % ODEs
    dA = 320 * s * (A - A*B + B - q*A*A);
    dB = 320 * (C-B-A*B)/s;
    dC = 320 * w * (A-C);

    dX(1) = dA;
    dX(2) = dB;
    dX(3) = dC;

endfunction

%(continues ...)
```

## Using the Octave solver

```
%%% setting the simulation time and the number of points
t=linspace(0,2,1000);

%%% (or the time distance between two consecutive points)
%t=0:0.001:2;

%%% initial state
X0 = [1 1 2];

%%% These can be used to set global and local error tolerance
%lsode_options("absolute tolerance",1e-5);
%lsode_options("relative tolerance",1e-5);

%%% These can be used to force non-stiff or stiff integration
%lsode_options("integration method","non-stiff");
%lsode_options("integration method","stiff");

%%% Call the solver
X = lsode ("dX",X0,t);

%(continues ...)
```



## Using the Octave solver

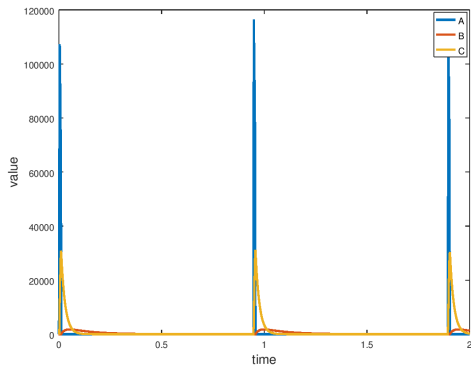
```
%%% Plot results
plot(t,X,"linewidth", 2);
xlabel ("time", "fontsize", 14);
ylabel ("value", "fontsize", 14);
legend("A","B","C");

%%% This would print the plot to a file
%print('graph.png','-dpng');

%%% Wait for the user to look at the graph
pause();
```

# Using the Octave solver

Result:



# Using the Octave solver

## Comments:

- The system is **stiff** (because of the “fast” peaks)
  - ▶ **try to force** the solver to work in **non-stiff mode**
- In the case of stiff systems, the solver uses the **Jacobian matrix** (partial derivatives) to approximate the system behaviour at each step
- An **approximate** Jacobian matrix is computed **automatically**...
- ... but a function computing it precisely **can be provided** to the solver
  - ▶ this increases both accuracy and performance

Let's modify the previous implementation by passing the Jacobian matrix to the solver

## Using the Octave solver

```
%%% Model parameters (global variables to be used in both the derivat
%%% and the jacobian functions)
global s = 77.27;
global q = 0.000008375;
global w = 0.161;

%%% Function computing derivatives
function dX = dX(X,t)

    global s q w; % use the global variables

    % variables used for the sake of readability
    A = X(1);
    B = X(2);
    C = X(3);

    % ODEs
    dA = 320 * s * (A - A*B + B - q*A*A);
    dB = 320 * (C-B-A*B)/s;
    dC = 320 * w * (A-C);

    dX(1) = dA;
    dX(2) = dB;
    dX(3) = dC;

endfunction

%(continues ...)
```

# Using the Octave solver

```
%% Function computing the Jacobian matrix
function J = jac(X,t)

    global s q w; % use the global variables

    % variables used for the sake of readability
    A = X(1);
    B = X(2);
    C = X(3);

    % partial derivatives
    dAdA = 320 * s * (1 - B - 2*q*A);
    dBdA = 320 * B / s;
    dCdA = 320 * w;
    dAdB = 320 * s * (-A + 1);
    dBdB = 320 * (-1 -A) / s;
    dCdB = 0;
    dAdC = 0;
    dBdC = 320/s;
    dCdC = 320 * w * (-1);

    J(1,1) = dAdA;    J(2,1) = dBdA;    J(3,1) = dCdA;
    J(1,2) = dAdB;    J(2,2) = dBdB;    J(3,2) = dCdB;
    J(1,3) = dAdC;    J(2,3) = dBdC;    J(3,3) = dCdC;

endfunction

%(continues ...)
```

## Using the Octave solver

```
%%% setting the simulation time and the number of points
t=linspace(0,2,1000);

%%% (or the time distance between two consecutive points)
%t=0:0.001:2;

%%% initial state
X0 = [1 1 2];

%%% These can be used to set global and local error tolerance
%lsode_options("absolute tolerance",1e-5);
%lsode_options("relative tolerance",1e-5);

%%% These can be used to force non-stiff or stiff integration
%lsode_options("integration method","non-stiff");
%lsode_options("integration method","stiff");

%%% Pass both the functions to the solver
X = lsode (@dX, @jac ,X0,t);

%(continues ...)
```

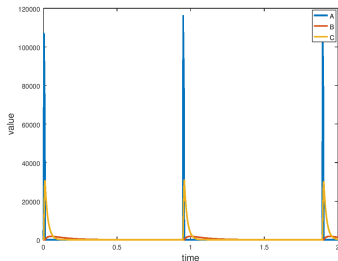
# Using the Octave solver

```
%% Plot results
plot(t,X,"linewidth", 2);
xlabel ("time", "fontsize", 14);
ylabel ("value", "fontsize", 14);
legend("A","B","C");

%% This would print the plot to a file
%print('graph.png','-dpng');

%% Wait for the user to look at the graph
pause();
```

Same result as before:



## RELEVANT EXAMPLES OF ODE MODELS



# Notation

It is no longer necessary to mention  $t$  in ODEs...

In the ODEs that follow  
we will omit any explicit reference  
to the time variable  $t$

We will write:

- $X$  for  $X(t)$
- $\dot{X}$  for  $\dot{X}(t)$
- $X_0$  for  $X(0)$

# The Lotka-Volterra model of prey-predator interaction

Independently proposed by Lotka in 1925 and Volterra in 1926

- By **Lotka** as a description of an hypothetical biochemical oscillator
- By **Volterra** as a description of two interacting populations

Volterra introduced this model to explain a **strange phenomenon** observed in the Adriatic sea after the **First World War**

- Ecologists (and fisherman) observed an increase in the population of some species of fish
- They expected the war to cause all populations to decrease...

Volterra's **intuition** was that prey and predator species have different (but related) dynamics.

- **preys proliferate in the absence of predators**

# The Lotka-Volterra model of prey-predator interaction

Let us consider the following variables:

- $V$  describes the size (or density) of the population of **preys**
- $P$  describes the size (or density) of the population of **predators**

Basic observations on the **inner dynamics** of preys and predators (i.e. the dynamics of each population in isolation):

- In the absence of predators, preys can grow without any limitation.
- In the absence of preys, predators die.

From these observations, we obtain this preliminary system of ODEs:

$$\begin{cases} \dot{V} = rV & \text{exponential growth} \\ \dot{P} = -sP & \text{exponential decay} \end{cases}$$

where  $r$  is the growth rate of preys and  $s$  the death rate of predators.

# The Lotka-Volterra model of prey-predator interaction

If both species are present in the environment, we observe **hunting** of predators on preys.

Assuming that **meeting** between individuals of both species **is random**, the number of meetings per time unit is **proportional to both  $V$  and  $P$**

- namely it is proportional to the product  $VP$

Not all predator-prey meetings result in a hunting...

- let  $a$  be the portion of meetings resulting in a hunting

Now, the more the predators eat, the more they can survive and reproduce

- let  $b$  denote the number of offsprings produced for each hunting

# The Lotka-Volterra model of prey-predator interaction

We obtain the following system of ODEs (the **Lotka-Volterra model**):

$$\begin{cases} \dot{V} = rV - aVP \\ \dot{P} = -sP + abVP \end{cases}$$

that is

- preys decrease by a **hunting rate**  $aVP$
- predators increase by a **hunting and reproduction rate**  $abVP$

Predation is a **direct form** of interaction

- it is not mediated by the environment

# The Lotka-Volterra model: coefficients

Let's play with the model!

We consider the following coefficients:

- Birth of preys:  $r = 10$
- Death of predators:  $s = 10$
- Hunting of predators on preys:  $a = 0.01$
- Reproduction of predators:  $b = 1$

We obtain:

$$\begin{cases} \dot{V} = 10V - 0.01VP \\ \dot{P} = -10P + 0.01VP \end{cases}$$

Before using a numerical solver... is there any **steady state**?

## The Lotka-Volterra model: steady states

A **steady state** is a combination of values for the variables that remains unchanged over time

In a steady state, all differential equations are equal to **zero**

So, we can find steady states of the Lotka-Volterra model by solving the following system of equations:

$$\begin{cases} 0 = 10V - 0.01VP \\ 0 = -10P + 0.01VP \end{cases}$$

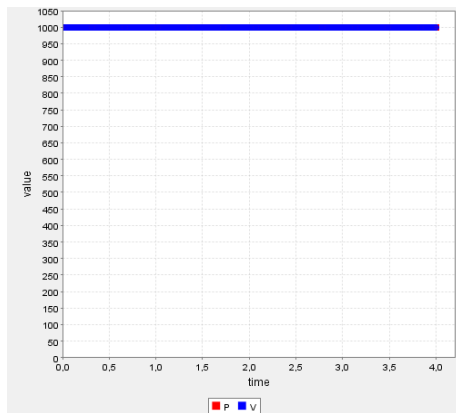
It has two solutions:

- $V = 0$  and  $P = 0$  (empty population)
- $V = 1000$  and  $P = 1000$

# The Lotka-Volterra model: numerical solution

Let's use a numerical solver with the following initial conditions:

- $V_0 = 1000$
- $P_0 = 1000$



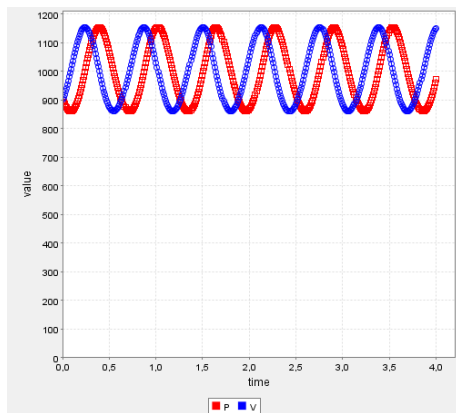
Both populations are actually stable



# The Lotka-Volterra model: numerical solution

Let's use a numerical solver with the following initial conditions:

- $V_0 = 900$
- $P_0 = 900$

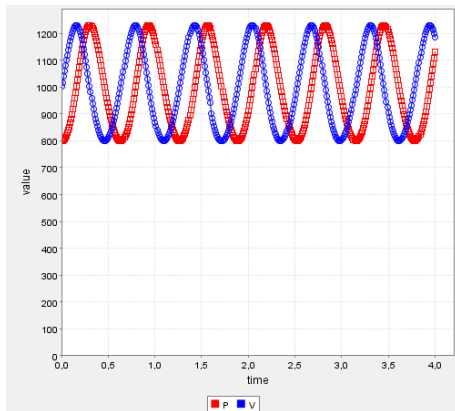


Oscillations around 1000 arise: first preys, then predators

# The Lotka-Volterra model: numerical solution

Let's use a numerical solver with the following initial conditions:

- $V_0 = 800$
- $P_0 = 1000$



Same dynamics: oscillations around 1000

# The Lotka-Volterra model: numerical solution

Back to the **initial question**:

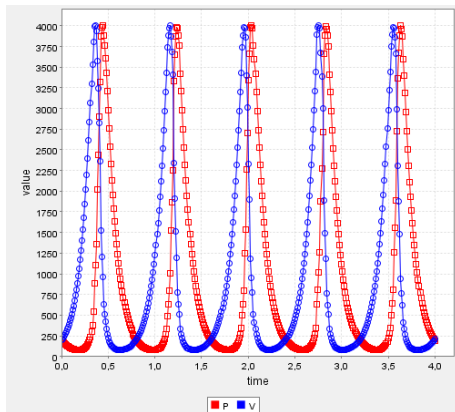
What happens if we significantly reduce the size of both preys and predators?

- **World war effect**

# The Lotka-Volterra model: numerical solution

Let's use a numerical solver with the following initial conditions:

- $V_0 = 200$
- $P_0 = 200$



Strong oscillations. At time 0.3 we have  $\sim 3500$  preys (much more than 1000) and only  $\sim 500$  predators.

# The SIR epidemic models and the effects of vaccination

Epidemic phenomena (spread of infectious diseases) are often studied by means of a **SIR model**

SIR stands for:

- **Susceptible**: individual that can be infected
- **Infected**: individual that has been infected and that can infect susceptible individuals
- **Recovered** (or Resistant): individual who passed the infection phase and cannot infect other individuals any more

First formulation of a SIR model was proposed by Kermack and McKendrick in 1927

# The SIR epidemic models and the effects of vaccination

The dynamics of epidemic phenomena is described by means of ODEs

- One equation for each type of individual: variables  $S, I, R$  describe the **ratios** of each class of individual in the population

Assumptions of the (initial) model:

- the **size** of the population is **constant** in time (and **normalized to 1**), so it always hold  $S + I + R = 1$
- only the transmission of infection is described: no reproduction, death, migration, etc...
- disease is transmitted by personal contacts between individuals of  $I$  and  $S$  classes (**horizontal transmission**)
- **contacts** between individuals **are random**, i.e. the number of infections is proportional to both  $I$  and  $S$
- after infection. individuals recover and become resistant to that disease

# The SIR epidemic models and the effects of vaccination

Therefore, the model is described by the system of equations:

$$\begin{cases} \dot{S} = -\beta SI \\ \dot{I} = \beta SI - \gamma I \\ \dot{R} = \gamma I \end{cases}$$

where:

- $\beta$  is the **infection coefficient**, describing probability of infection after the contact of a healthy individual with an infected one
- $\gamma$  is the **recovery coefficient**, describing the rate of recovery of each infected individual (in other words,  $1/\gamma$  is the **time** one individual requires for recovering)

## The SIR epidemic models: role of parameters

$$\begin{cases} \dot{S} = -\beta SI \\ \dot{I} = \beta SI - \gamma I \\ \dot{R} = \gamma I \end{cases}$$

### Note that:

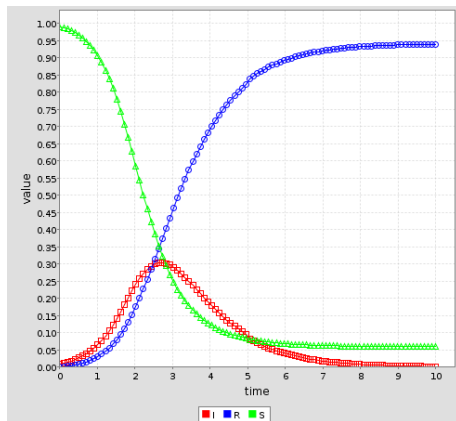
- $R$  is the ratio of the population which got the disease in the past
- $R$  is almost useless, it does not appear in the other equations and could be replaced by  $1 - S - I$
- $S$  can only decrease
- if  $\beta < \gamma$  (i.e.  $\beta/\gamma < 1$ ),  $I$  can only decrease (since  $S \leq 1$ )
- if  $\beta > \gamma$  (i.e.  $\beta/\gamma > 1$ ), the behavior of  $I$  depends on  $S$ . It initially increases if  $S > \gamma/\beta$ .



# The SIR epidemic models: numerical solution

First case:  $\beta/\gamma > 1$

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 3$
- $\gamma = 1$

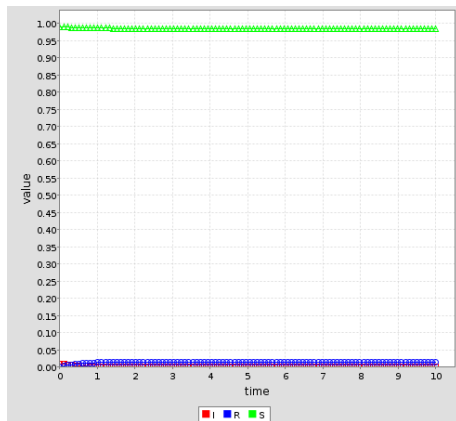


Spread of infection. 95% of the population gets the disease.

# The SIR epidemic models: numerical solution

First case:  $\beta/\gamma < 1$

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 1$
- $\gamma = 3$



The infection **doesn't diffuse**.  $\sim 1\%$  of the population gets the disease.

# The SIR epidemic models: influenza

Let's try to **find parameters** to describe the **spread of influenza**:

Flu usually takes more or less one week (say, 8 days)

- $\gamma = 1/8 = 0.125$

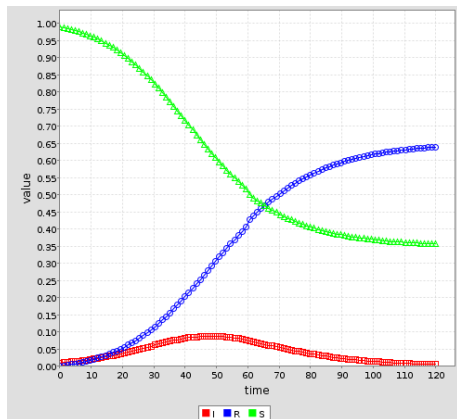
A person with flu usually infects more or less  $1/5$  of the persons he/she meets

- $\beta = 1/5 = 0.2$

Flu epidemics usually cover winter (say, 120 days)

# The SIR epidemic models: influenza

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 0.2$
- $\gamma = 0.125$



~ 65% of the population gets the disease, with a peak of infections at the end of the second month.

# The SIR epidemic models and vaccination

The SIR model can be used to study the **effects of vaccination**

Vaccinations may require **years** to show their effect

- **births and deaths cannot be ignored** if we consider a long time span

Let's **extend** the SIR model with birth and deaths

# The SIR epidemic models: births and deaths

## Assumptions:

- for the sake of simplicity we would like the population size **still to be constant over time** (not too wrong: the size of the population of a country does not change significantly over 10-20 years...)
- **No vertical transmission** of the disease (from parents to children)
- Newborns are susceptible

Constant population size can be obtained by using the **same coefficient**  $\mu$  for both birth and death

- This works since the population size is normalized to 1
- Let  $N = S + I + R$ , we have that  $\dot{N} = \mu - \mu N$  has  $N = 1$  as steady state

## The SIR epidemic models: births and deaths

This is the extended model:

$$\begin{cases} \dot{S} = \mu - \beta SI - \mu S \\ \dot{I} = \beta SI - \gamma I - \mu I \\ \dot{R} = \gamma I - \mu R \end{cases}$$

As in the previous case, the dynamics of the model is governed by the ratio between the positive and negative coefficients in the equation of  $I$

- if  $\beta < (\mu + \gamma)$  (i.e.  $\beta/(\mu + \gamma) < 1$ ),  $I$  can only decrease (since  $S \leq 1$ )
- if  $\beta > (\mu + \gamma)$  (i.e.  $\beta/(\mu + \gamma) > 1$ ), the behavior of  $I$  depends on  $S$ . It increases if  $S > (\mu + \gamma)/\beta$ .

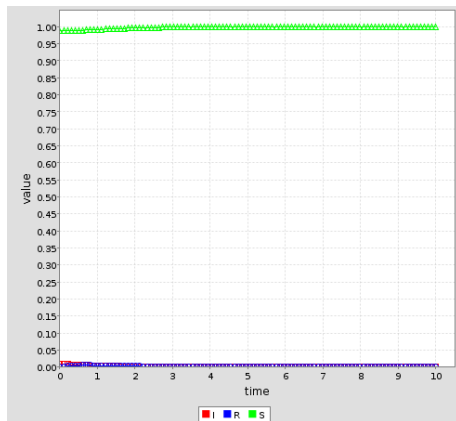
In the previous case  $S$  could only decrease, but this is no longer true, since we have births

- **births could maintain  $S$  above  $(\mu + \gamma)/\beta$ , which sustains infections (endemic state)**

# The SIR epidemic models: births and deaths

First case:  $\beta/(\mu + \gamma) < 1$

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 3$
- $\gamma = 2$
- $\mu = 2$



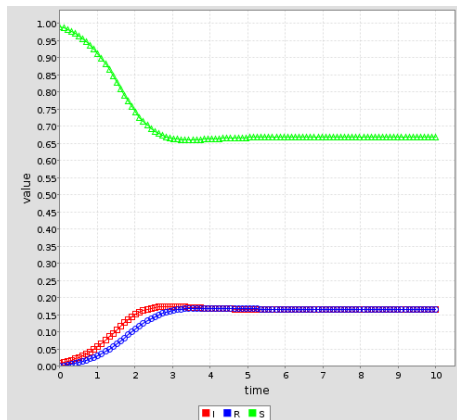
The infection **doesn't diffuse**. In the end, 100% of the population is susceptible (since recovered individuals die)



# The SIR epidemic models: births and deaths

Second case:  $\beta/(\mu + \gamma) > 1$

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 6$
- $\gamma = 2$
- $\mu = 2$



**Endemic state!** The population reaches a steady state in which  $\sim 17\%$  of the population is infected.

# The SIR epidemic models: vaccination

Let's **further extend** the model with **vaccinations**.

Assumption:

- vaccination is done on **a fraction  $p$  of the newborns**
- this corresponds to assuming that a fraction  $p$  of newborns are already in the **recovered** state  $R$

This results in the following model:

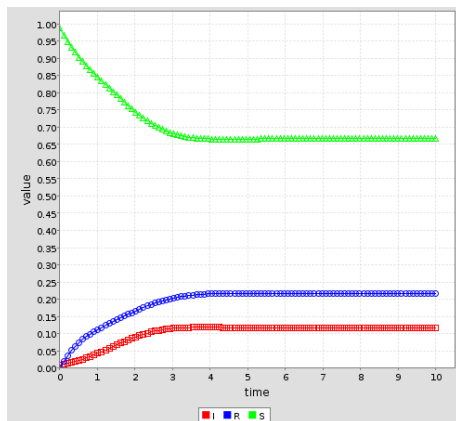
$$\begin{cases} \dot{S} = (1 - p)\mu - \beta SI - \mu S \\ \dot{I} = \beta SI - \gamma I - \mu I \\ \dot{R} = p\mu + \gamma I - \mu R \end{cases}$$

Let's perform some simulations by **varying  $p$** .

# The SIR epidemic models: births and deaths

First case:  $p$  small

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 6$
- $\gamma = 2$
- $\mu = 2$
- $p = 0.1$

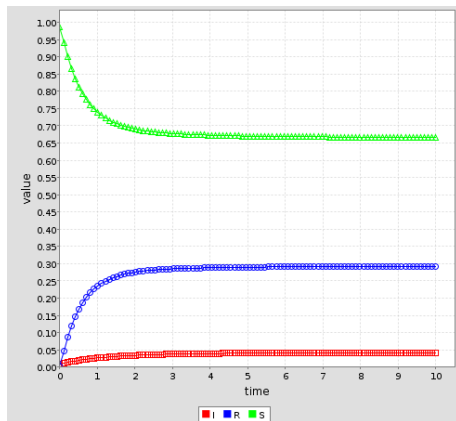


Still endemic state!

# The SIR epidemic models: births and deaths

First case:  $p$  medium

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 6$
- $\gamma = 2$
- $\mu = 2$
- $p = 0.25$

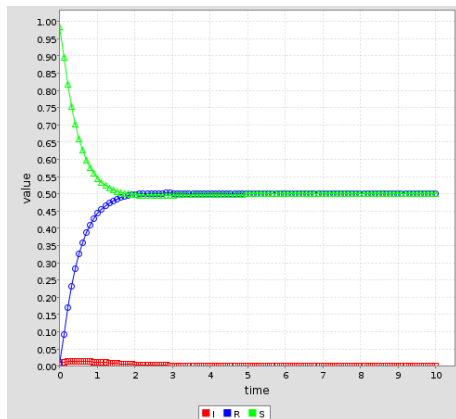


Endemic state, but with less infected.

# The SIR epidemic models: births and deaths

First case: **higher  $p$**

- $S_0 = 0.99$
- $I_0 = 0.01$
- $R_0 = 0$
  
- $\beta = 6$
- $\gamma = 2$
- $\mu = 2$
- $p = 0.5$



**Healthy population!** The disease has been **eradicated**.

## The SIR epidemic models: vaccination threshold

So, which ratio of the newborns should be vaccinated in order to obtain disease eradication?

- $p_c$ : threshold value of vaccination

The value of  $p_c$  can be computed analytically (see lecture notes) or simply by performing numerical simulations varying  $p$ .

- result:

$$p_c = 1 - \frac{\mu + \gamma}{\beta}$$

In the previous example  $p_c = 1 - (2 + 2)/6 = 1/3$

## The SIR epidemic models: application to measles

Let's consider **measles** (“morbillo”)

For measles the following parameter values were estimated:

$$\mu = 0.02 \quad \beta = 1800 \quad \gamma = 100$$

which give the critical ratio of vaccine

$$p_c = 1 - \frac{0.02 + 100}{1800} \simeq 0.95$$

meaning that 95% of the newborns should be vaccinated in order to eradicate the disease

# Lessons learnt

Summing up:

- ODEs can be used to describe dynamical systems whose state updates continuously
  - ▶ **continuous dynamical systems**
- Solving ODEs analytically is often difficult/impossible
  - ▶ **numerical solution** of initial value problems helps
  - ▶ numerical solvers approximate ODEs by discretizing them (recurrence equations)
  - ▶ **stiffness** causes problems: implicit methods perform better in these cases
- ODEs are used extensively
  - ▶ Lotka-Volterra and SIR models are well-established examples



# Limitations of continuous dynamical models

Continuous dynamical models based on ODEs describe **THE** behavior of the system, starting from an initial state

- often system may exhibit **different behaviors** starting from the **same initial state**
- this happens when some events happening within the system are somehow **random**
- in order to properly model these system, probabilisties have to be included in the model (**probabilistic/stochastic model**)

ODEs are **not very modeler-friendly**

- When the number of equations is high and each equation has many terms, understanding the model and manipulating it may become difficult
- **more intuitive notations** should be found

# Exercises

- Define a **variant of the Lotka-Volterra model** with 3 kinds of individual: **predator**, **prey** and **vegetation**. Predators eat preys, preys eat vegetation, and vegetation (in the absence of preys) grows exponentially
  - ▶ compute the steady states
  - ▶ try to perform numerical simulation by varying coefficient values and initial quantities of predators, preys and vegetation
- Extend the **“influenza”** model with **vaccination**
  - ▶ the vaccination rate could be constant in time or (better) could be described by a **logistic function** with  $K$  corresponding to the number of individuals who choose to vaccinate (typically within few weeks)
- A lot of **variants of the SIR** model exist
  - ▶ look at the **Wikipedia page** “Compartmental models in epidemiology”