Project in INF1100

2013

Nov 28, 2013

Disease Modeling

We shall in this project model epidemiological diseases such as measles or swine flu and eventually apply such models to simulate how humans can defend themselves against a zombie attack.

We start with simple disease modeling. Suppose we have three categories of people: susceptibles (S) who can get the disease, infected (I) who have developed the disease and who can infect susceptibles, and recovered (R) who have recovered from the disease and become immune. Let $S(t)$, $I(t)$, and $R(t)$ be the number of people in category S, I, and R, respectively. We have that $S + I + R = N$, where N is the size of the population, assumed constant here for simplicity.

When people mix in the population there are SI possible pairs of susceptibles and infected, and a certain fraction βSI per time interval meets with the result that the infected "successfully" infect the susceptible. During a time interval Δt, $\beta SI\Delta t$ get infected and move from the S to the I category:

$$S(t + \Delta t) = S(t) - \beta SI\Delta t\,.$$

We divide by Δt and let $\Delta \to 0$ to get the differential equation

$$S'(t) = -\beta SI\,. \tag{1}$$

A fraction νI of the infected will per time unit recover from the disease. In a time Δt, $\nu I\Delta t$ recover and move from the I to the R category. The quantity $1/\nu$ typically reflects the duration of the disease. In the same time interval, $\beta SI\Delta t$ come from the S to the I category. The accounting for the I category therefore becomes

$$I(t + \Delta t) = I(t) + \beta SI\Delta t - \nu I\Delta t,$$

which in the limit $\Delta t \to \infty$ approaches the differential equation

$$I'(t) = \beta SI - \nu I\,. \tag{2}$$

Finally, the R category gets contributions from the I category:

$$R(t+\Delta t) = R(t) + \nu I \Delta t\,.$$

The corresponding ODE for R reads

$$R'(t) = \nu I\,. \tag{3}$$

In case the recovered do not become immune, we do not need the recovered category, since the recovered go directly out of the I category to the S category again. This gives a contribution νI to the equation for S and we end up with an S-I system of only two equations for S and I.

The ODE system (1)-(3) is known as a *SIR model* in epidemiology (which is the name of the scientific field studying the spreading of epidemic diseases).

Solution of ODE systems. The SIR model and related models in the exercises below can be solved by numerical methods like the Forward Euler scheme, the iterated midpoint scheme, 2nd and 4th order Runge-Kutta methods, and so on. These methods can be implemented from scratch, or one can use read-made software like

- the `ODESolver` module from INF1100,
- `scipy.integrate.ode`
- `odespy`

The usage of the `ODESolver` module goes as follows.

```
from ODESolver import RungeKutta4
import numpy as np

class RHS:
    def __init__(self, ...):
        # Store parameters in the model

    def __call__(self, u, t):
        """Return right-hand side in ODE model u'=f(u,t)."""
        S, I, R = u
        return [..., ..., ...]

f = RHS(...)
solver = RungeKutta4(f)
U0 = [1500, 1, 0]
solver.set_initial_condition(U0)
n = 120
time_points = np.linspace(0, T=60, n+1)
u, t = solver.solve(time_points)
S = u[:,0]
I = u[:,1]
R = u[:,2]
```

Problem 1: Implement and experiment with the SIR model

Make a function for solving the differential equations in the SIR model (1)-(3) by any numerical method of your choice. Make a separate function for visualizing $S(t)$, $I(t)$, and $R(t)$ in the same plot.

Adding the equations shows that $S' + I' + R' = 0$, which means that $S + I + R$ must be constant. Perform a test at each time level for checking that $S + I + R$ equals $S_0 + I_0 + R_0$ within some small tolerance. If a subclass of `ODESolver` is used to solve the ODE system, the test can be implemented as a user-specified `terminate` function that is called by the `solve` method a every time level (simply return `True` for termination if $S + I + R$ is not sufficiently constant).

A specific population has 1500 susceptibles and one infected. We are interested in how the disease develops. Set $S(0) = 1500$, $I(0) = 1$, and $R(0) = 0$. Choose $\nu = 0.1$, $\Delta t = 0.5$, and $t \in [0, 60]$. Time t here counts days. Visualize first how the disease develops when $\beta = 0.0005$. Certain precautions, like staying inside, will reduce β. Try $\beta = 0.0001$ and comment from the plot how a reduction in β influences $S(t)$. (Put the comment as a multi-line string in the bottom of the program file.) Filename: `SIR.py`.

Problem 2: Make a more flexible code for the SIR model

The parameters ν and β in the SIR model in Exercise 1 can be constants or functions of time. Now we shall make an implementation of the $f(u, t)$ function specifying the ODE system such that ν and β can be given as either a constant or a Python function. Introduce a class for $f(u, t)$, with the following code sketch:

```
class Problem:
    def __init__(self, nu, beta, S0, I0, R0, T):
        """
        nu, beta: parameters in the ODE system
        S0, I0, R0: initial values
        T: simulation for t in [0,T]
        """
        if isinstance(nu, (float,int)):  # number?
            self.nu = lambda t: nu       # wrap as function
        elif callable(nu):
            self.nu = nu

        # same for beta and self.beta
        ...

        # store the other parameters

    def __call__(self, u, t):
        """Right-hand side function of the ODE system."""
        S, I, R = u
        return [-self.beta(t)*S*I,    # S equation
                ...,                  # I equation
                self.nu(t)*I]         # R equation

# Example:
problem = Problem(beta=lambda t: 0.0005 if t <= 12 else 0.0001,
                  nu=0.1, S0=1500, I0=1, R0=0, T=60)
solver = ODESolver.ForwardEuler(problem)
```

Write the complete code for class `Problem` based on the sketch of ideas above. The ν parameter is usually not varying with time as $1/\nu$ is a characteristic size of the period a person is sick, but introduction of new medicine during the disease might change the picture such that time dependence becomes relevant. The construction `beta=lambda ...` is a quick way of defining a function as argument in a function call. The equivalent, more verbose, construction is `beta=mybeta`, where `mybeta` is an ordinary functin

```
def mybeta(t):
    if t <= 12:
        return 0.0005
    else:
        return 0.0001
```

We can also make a class `Solver` for solving the problem, here utilizing the `ODESolver` module for executing the numerical method:

```
class Solver:
    def __init__(self, problem, dt):
        self.problem, self.dt = problem, dt

    def solve(self, method=ODESolver.RungeKutta4):
        self.solver = method(self.problem)
        ic = [self.problem.S0, self.problem.I0, self.problem.R0]
        self.solver.set_initial_condition(ic)
        n = int(round(self.problem.T/float(self.dt)))
        t = np.linspace(0, self.problem.T, n+1)
        u, self.t = self.solver.solve(t)
        self.S, self.I, self.R = u[:,0], u[:,1], u[:,2]

    def plot(self):
        # plot S(t), I(t), and R(t)
```

After the breakout of a disease, authorities often start campaigns for decreasing the spreading of the disease. Suppose a massive campaign telling people to wash their hands more frequently is launched, with the effect that β is significantly reduced after a some days. For the specific case simulated in Problem 1, let

$$\beta(t) = \begin{cases} 0.0005, & 0 \leq t \leq 12, \\ 0.0001, & t > 12 \end{cases}$$

Simulate this scenario with the `Problem` and `Solver` classes. Report the maximum number of infected people and compare it to the case where $\beta(t) = 0.0005$. Filename: `SIR_class.py`.

Problem 3: SIR model with vaccination

We shall now extend the SIR model in Problems 1 and 2 with a vaccination program. If a fraction p of the susceptibles per time unit is being vaccinated, and we say that the vaccination is 100 percent effective, $pS\Delta t$ individuals will be removed from the S category in a time interval Δt. We place the vaccinated people in a new category V. The equations for S and V becomes

$$S' = -\beta SI - pS, \tag{4}$$
$$V' = pS \tag{5}$$

The equations for I and R are not affected. The initial condition for V can be taken as $V(0) = 0$. The resulting model is named SIRV.

a) Try the same parameters as in Problem 2 in combination with $p = 0.1$ and compute the evolution of $S(t)$, $I(t)$, $R(t)$, and $V(t)$. Comment on the effect of vaccination on the maximum number of infected.

Regarding the implementation, it is recommended to make classes as explained in Problem 2. The implementation in the next question will then be simpler.

b) Let the vaccination campaign start 6 days after the outbreak of the disease and let it last for 10 days,

$$p(t) = \begin{cases} 0.1, & 6 \leq t \leq 15 \\ 0, & \text{otherwise} \end{cases}$$

Plot the corresponding solutions $S(t)$, $I(t)$, $R(t)$, and $V(t)$.

c) Let the vaccination campaign last for V_T days:

$$p(t) = \begin{cases} 0.1, & 6 \leq t \leq 6 + V_T \\ 0, & \text{otherwise} \end{cases}$$

Compute the maximum number of infected people, $\max_t I(t)$, as a function of $V_T \in [0, 31]$, by running the model for $V_T = 0, 1, 2 \ldots, 31$. Plot this function. Determine from the plot the optimal V_T, i.e., the smallest vaccination period V_T such that increasing V_T has negligible effect on the maximum number of infected people.

Filename: `SIRV.py`.

0.1 Modeling of human-zombie interaction

Suppose the human population is attacked by zombies. This is quite a common happening in movies, and the "zombification" of humans acts much like the spreading of a disease. Let us make a differential equation model, inspired by the SIR model from Problem 1, to simulate how humans and zombies interact.

We introduce four categories of individuals:

- S: susceptible humans who can become zombies.
- I: infected humans, being bitten by zombies.
- Z: zombies.
- R: removed individuals, either conquered zombies or dead humans.

The corresponding functions counting how many individuals we have in each category are named $S(t)$, $I(t)$, $Z(t)$, and $R(t)$, respectively.

The type of zombies considered here is inspired by the standard for modern zombies set by the classic movie *The Night of the Living Dead*, by George A. Romero from 1968. Only a small extension of the SIR model is necessary to model the effect of human-zombie interaction mathematically. A fraction of the human susceptibles is getting bitten by zombies and moves to the infected category. A fraction of the infected is then turned into zombies. On the other hand, humans can conquer zombies.

Now we shall precisely set up all the dynamic features of the human-zombie populations we aim to model. Changes in the S category are due to three effects:

- Susceptibles are infected by zombies, modeled by a term $-\Delta t\beta SZ$, similar to the S-I interaction in the SIR model.
- Susceptibles die naturally or get killed and therefore enter the removed category. If the probability that one susceptible dies during a unit time interval is δ_S, the total expected number of deaths in a time interval Δt becomes $\Delta t\delta_S S$.
- We also allow new humans to enter the area with zombies, as this effect may be necessary to successfully run a war on zombies. The number of new individuals in the S category arriving per time unit is denoted by Σ, giving an increase in $S(t)$ by $\Delta t\Sigma$ during a time Δt.

We could also add newborns to the S category, but we simply skip this effect since it will not be significant over time scales of a few days.

The balance of the S category is then

$$S' = \Sigma - \beta SZ - \delta_S S,$$

in the limit $\Delta t \to 0$.

The infected category gets a contribution $\Delta t\beta SZ$ from the S category, but loses individuals to the Z and R category. That is, some infected are turned into zombies, while others die. Movies reveal that infected may commit suicide or that others (susceptibles) may kill them. Let δ_I be the probability of being killed in a unit time interval. During time Δt, a total of $\delta_I \Delta t I$ will die and hence be transferred to the removed category. The probability that a single infected is turned into a zombie during a unit time interval is denoted by ρ, so that a total of $\Delta t\rho I$ individuals are lost from the I to the Z category in time Δt. The accounting in the I category becomes

$$I' = \beta SZ - \rho I - \delta_I I\,.$$

The zombie category gains $-\Delta t\rho I$ individuals from the I category. We disregard the effect that any removed individual can turn into a zombie again, as we consider that effect as pure magic beyond reasonable behavior, at least according to what is observed in the Romero movie tradition. A fundamental

feature in zombie movies is that humans can conquer zombies. Here we consider zombie killing in a "man-to-man" human-zombie fight. This interaction resembles the nature of zombification (or the susceptible-infective interaction in the SIR model) and can be modeled by a loss $-\alpha SZ$ for some parameter α with an interpretation similar to that of β. The equation for Z then becomes

$$Z' = \rho I - \alpha SZ\,.$$

The accounting in the R category consists of a gain δS of natural deaths from the S category, a gain δI from the I category, and a gain αSZ from defeated zombies:

$$R' = \delta_S S + \delta_I I + \alpha SZ\,.$$

The complete SIZR model for human-zombie interaction can be summarized as

$$S' = \Sigma - \beta SZ - \delta_S S, \tag{6}$$
$$I' = \beta SZ - \rho I - \delta_I I, \tag{7}$$
$$Z' = \rho I - \alpha SZ, \tag{8}$$
$$R' = \delta_S S + \delta_I I + \alpha SZ\,. \tag{9}$$

The interpretations of the parameters are as follows:

- Σ: the number of new humans brought into the zombified area per unit time.
- β: the probability that a theoretically possible human-zombie pair actually meets physically, during a unit time interval, with the result that the human is infected.
- δ_S: the probability that a susceptible human is killed or dies, in a unit time interval.
- δ_I: the probability that an infected human is killed or dies, in a unit time interval.
- ρ: the probability that an infected human is turned into a zombie, during a unit time interval.
- α: the probability that, during a unit time interval, a theoretically possible human-zombie pair fights and the human kills the zombie.

Note that probabilities per unit time do not necessarily lie in the interval $[0, 1]$. The real probability, lying between 0 and 1, arises after multiplication by the time interval of interest.

Problem 4: Implement and test human-zombie interaction

Implement the SIZR model with problem and solver classes as explained in Problem 2. Allow parameters to vary in time. The time variation is essential to make a realistic model that can mimic what happens in movies. It becomes necessary to work with piecewise constant functions in time. Such functions are straightforwardly defined by:

```
from scitools.std import PiecewiseConstant

# Define f(t) as 1.5 in [0,3], 0.1 in [3,4] and 1 in [4,7]
f = PiecewiseConstant(domain=[0, 7],
                      data=[(0, 1.5), (3, 0.1), (4, 1)])
```

Test the implementation with the following data: $\beta = 0.0012$, $\alpha = 0.0016$, $\delta_I = 0.014$, $\Sigma = 2$, $\rho = 1$, $S(0) = 10$, $Z(0) = 100$, $I(0)$, $R(0) = 0$, and simulation time $T = 24$ hours. All other parameters can be set to zero. These values are estimated from the hysterical phase of the movie *The Night of the Living Dead*. The time unit is hours. Plot the S, I, Z, and R quantities.

Import problems with PiecewiseConstant.

If you get an `ImportError` when trying to import `PiecewiseConstant` you probably have a too old version of SciTools. The latest version can be obtained and installed by the following Unix commands:

```
hg clone https://hpl%40simula.no@code.google.com/p/scitools/
cd scitools
sudo python setup.py install
```

This will only work on machines where you are allowed to run the `sudo` command, i.e., install software in system directories. The command will in particular *not work* on UiO's machines. There, a more involved procedure is necessary:

```
mkdir $HOME/inf1100-software
python setup.py install --prefix=$HOME/inf1100-software
```

Add the following line at the end of your `.bashrc` file in your home directory:

```
export PYTHONPATH=$HOME/inf1100-software/lib/python2.7/site-packages:$
```

Make a new terminal window and see if you have the new SciTools version available (in this and all future terminal windows) by typing the command

```
python -c 'from scitools.std import PiecewiseConstant'
```

Filename: `SIZR.py`.

Problem 5: Simulate a zombie movie

The movie *The Night of the Living Dead* has three phases:

1. The initial phase, lasting for (say) 4 hours, where two humans meet one zombie and one of the humans get infected. A rough (and uncertain) estimation of parameters in this phase, taking into account dynamics not shown in the movie, yet necessary to establish a more realistic evolution of the S and Z categories later in the movie, is $\Sigma = 20$, $\beta = 0.03$, $\rho = 1$, $S(0) = 60$, and $Z(0) = 1$. All other parameters are taken as zero when not specified.

2. The hysterical phase, when the zombie treat is evident. This phase lasts for 24 hours, and relevant parameters can be taken as $\beta = 0.0012$, $\alpha = 0.0016$, $\delta_I = 0.014$, $\Sigma = 2$, $\rho = 1$.

3. The counter attack by humans, estimated to last for 5 hours, with parameters $\alpha = 0.006$, $\beta = 0$ (humans no longer get infected), $\delta_S = 0.0067$, $\rho = 1$.

Use the program from Problem 4 to simulate all three phases of the movie. Filename: `The_Night_of_the_Living_Dead.py`.

Problem 6: Simulate a war on zombies

A war on zombies can be implemented through large-scale effective attacks. A possible model is to increase α in the SIZR model from Exercise 4 by some additional amount $\omega(t)$, where $\omega(t)$ varies in time to model strong attacks at $m+1$ distinct points of time $T_0 < T_1 < \cdots < T_m$. Around these t values we want ω to have a large value, while in between the attacks ω is small. One possible mathematical function with this behavior is a sum of Gaussian functions:

$$\omega(t) = a \sum_{i=0}^{m} \exp\left(-\frac{1}{2}\left(\frac{t - T_i}{\sigma}\right)^2\right), \tag{10}$$

where a measures the strength of the attacks (the maximum value of $\omega(t)$) and σ measures the length of the attacks, which should be much less than the time between the points of attack: typically, 4σ measures the length of an attack, and we must have $4\sigma \ll T_i - T_{i-1}$ for $i = 1, \ldots, m$. We should choose a significantly larger than α to make the attacks in the war on zombies much stronger than the usual "man-to-man" killing of zombies.

Modify the model and the implementation from Problem 4 to include a war on zombies. We start out with 50 humans and 3 zombies and $\beta = 0.03$. This leads to a rapid zombification. Assume that there are some small resistances

against zombies from the humans, $\alpha = 0.2\beta$, throughout the simulation. In addition, the humans implement three strong attacks, $a = 50\alpha$, at 5, 10, and 18 hours after the zombification starts. The attacks last for about 2 hours ($\sigma = 0.7$). Let the `alpha` function in the code be $\alpha + \omega(t)$, set $\delta_S = \Delta_I = \Sigma = 0$, $\beta = 0.03$, and $\rho = 1$, simulate for $T = 20$ hours, and see if the war on zombies modeled by the suggested $\omega(t)$ is sufficient to save mankind. Filename: `war_on_zombies.py`.