

Modeling the Spreading of Diseases

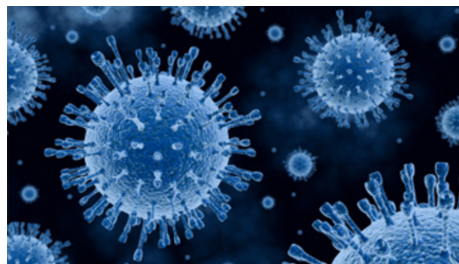
Joakim Sundnes^{1,2}

Hans Petter Langtangen^{1,2}

¹Simula Research Laboratory

²University of Oslo, Dept. of Informatics

Nov 8, 2020



0.1 Plan for week 46-47

Monday 9/11:

- Exercise E.29, E.49
- Modeling infectious diseases
- Comments on next week's project

Wednesday 11/11:

- No lecture (probably)

Week 47:

- Monday: Project lecture/questions
- Wednesday: Project lecture/questions (if needed)
- Thursday/Friday: Project help ("orakel"). (time and place TBD)

0.2 We shall model a complex phenomenon by simple math

Plan:

- Use simple intuition to derive a system of *difference equations* to model the spread of diseases
- Program the difference equations in the usual way (i.e. **for**-loops)
- Transform the difference equations to *ordinary differential equations*
- Explore possible model extensions

0.3 Assumptions:

- We consider a perfectly mixed population in a confined area
- No spatial transport, just temporal evolution
- We do not consider individuals, just a grand mix of them

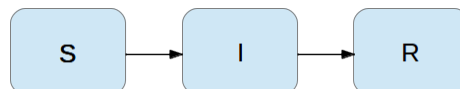
We consider very simple models, but these can be extended to full models that are used world-wide by health authorities. Typical diseases modeled are flu, measles, swine flu, HIV, SARS, ebola, Covid19, ...

0.4 We keep track of 3 categories in the SIR model

- **S**: susceptibles - who can get the disease
- **I**: infected - who have developed the disease and infect susceptibles
- **R**: recovered - who have recovered and become immune

Mathematical quantities: $S(t)$, $I(t)$, $R(t)$: no of people in each category

Goal: Find and solve equations for $S(t)$, $I(t)$, $R(t)$



0.5 The traditional modeling approach is very mathematical - our idea is to model, program and experiment

- Numerous books on mathematical biology treat the SIR model
- Quick modeling step (max 2 pages)
- Nonlinear differential equation model
- Cannot solve the equations, so focus is on discussing stability (eigenvalues), qualitative properties, etc.
- Very few extensions of the model to real-life situations

0.6 Dynamics in a time interval Δt : people move from S to I

S-I interaction:

- In a total population of N people, with S susceptibles and I infected, the chance of a single person in S meeting a person in I is proportional to I/N .
- The total number of such meetings will be proportional to SI/N . A certain fraction of these meetings leads to disease transmission.
- In a (small) time interval Δt , we assume that $\beta \Delta t SI/N$ meetings where the infected “successfully” infects the susceptible
- This gives a loss $\Delta t \beta SI/N$ in the S category and a corresponding gain in the I category

Remark. It is reasonable that the fraction depends on Δt (twice as many infected in $2\Delta t$ as in Δt). β is some unknown parameter we must measure, supposed to not depend on Δt , but maybe time t . β lumps a lot of biological and sociological effects into one number.

0.7 The equations describing S-I interaction become

Loss in $S(t)$ from time t to $t + \Delta t$:

$$S(t + \Delta t) = S(t) - \Delta t \beta \frac{S(t)I(t)}{N}$$

Gain in $I(t)$:

$$I(t + \Delta t) = I(t) + \Delta t \beta \frac{S(t)I(t)}{N}$$

0.8 Modeling the transition from I to R

I-R transition:

- After some days, the infected has recovered and moves to the R category
- A simple model: in a small time Δt (say 1 day), a fraction $\Delta t \nu$ of the infected are removed (ν must be measured)

We must subtract this fraction in the balance equation for I :

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

The loss $\Delta t \nu I$ is a gain in R :

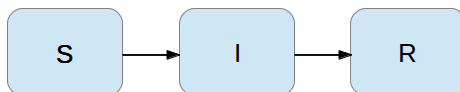
$$R(t + \Delta t) = R(t) + \Delta t \nu I(t)$$

0.9 We have three equations for S , I , and R

$$S(t + \Delta t) = S(t) - \Delta t \beta \frac{S(t)I(t)}{N} \quad (1)$$

$$I(t + \Delta t) = I(t) + \Delta t \beta \frac{S(t)I(t)}{N} - \Delta t \nu I(t) \quad (2)$$

$$R(t + \Delta t) = R(t) + \Delta t \nu I(t) \quad (3)$$



Before we can compute with these, we must

- know β and ν
- know $S(0)$ (many), $I(0)$ (few), $R(0)$ (0?)
- choose Δt

0.10 The computation involves just simple arithmetics

- Set $\Delta t = 0.1$ (= 6 minutes)
- Set $\beta = 0.06$, $\nu = 0.008333$
- Set $S(0) = 50$, $I(0) = 1$, $R(0) = 0$

$$\begin{aligned}
S(\Delta t) &= S(0) - \Delta t \beta \frac{S(0)I(0)}{N} \approx 49.99 \\
I(\Delta t) &= I(0) + \Delta t \beta \frac{S(0)I(0)}{N} - \Delta t \nu I(0) \approx 1.002 \\
R(\Delta t) &= R(0) + \Delta t \nu I(0) \approx 0.0008333
\end{aligned}$$

- We can continue, but quickly gets boring...
- Solve with a for-loop as usual

0.11 We use the standard notation

$$S^n = S(n\Delta t), \quad I^n = I(n\Delta t), \quad R^n = R(n\Delta t)$$

$$S^{n+1} = S((n+1)\Delta t), \quad I^{n+1} = I((n+1)\Delta t), \quad R^{n+1} = R((n+1)\Delta t)$$

The equations can now be written more compactly (and computer friendly):

$$S^{n+1} = S^n - \Delta t \beta S^n I^n \tag{4}$$

$$I^{n+1} = I^n + \Delta t \beta S^n I^n - \Delta t \nu I^n \tag{5}$$

$$R^{n+1} = R^n + \Delta t \nu I^n \tag{6}$$

0.12 Store S,I,R in arrays and solve with a loop

```

import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0, N*dt, N+1)
S = np.zeros(N+1)
I = np.zeros(N+1)
R = np.zeros(N+1)

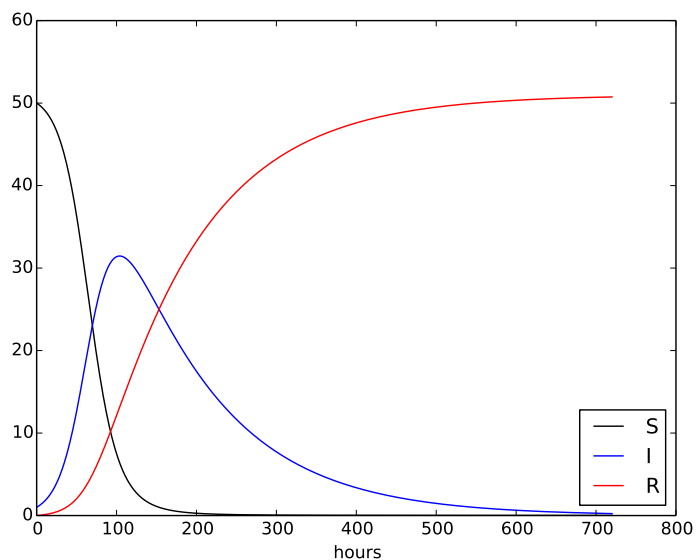
beta = 0.06
nu = 0.008333
dt = 0.1          # 6 min (time measured in hours)
D = 30            # simulate for D days
N = int(D*24/dt)  # corresponding no of hours
S[0] = 50
I[0] = 1

for n in range(N):
    S[n+1] = S[n] - dt*beta*S[n]*I[n]
    I[n+1] = I[n] + dt*beta*S[n]*I[n] - dt*nu*I[n]
    R[n+1] = R[n] + dt*nu*I[n]

# Plot the graphs
plt.plot(t, S, 'k-', t, I, 'b-', t, R, 'r-')
plt.legend(['S', 'I', 'R'], loc='lower right')
plt.xlabel('hours')
plt.show()

```

0.13 We have predicted a disease!



0.14 The standard mathematical approach: ODEs

We had from intuition established

$$\begin{aligned}
 S(t + \Delta t) &= S(t) - \Delta t \beta \frac{S(t)I(t)}{N} \\
 I(t + \Delta t) &= I(t) + \Delta t \beta \frac{S(t)I(t)}{N} - \Delta t \nu I(t) \\
 R(t + \Delta t) &= R(t) + \Delta t \nu I(t)
 \end{aligned}$$

The mathematician will now make *differential equations*. Divide by Δt and rearrange:

$$\begin{aligned}
 \frac{S(t + \Delta t) - S(t)}{\Delta t} &= -\beta \frac{S(t)I(t)}{N} \\
 \frac{I(t + \Delta t) - I(t)}{\Delta t} &= \beta \frac{S(t)I(t)}{N} - \nu I(t) \\
 \frac{R(t + \Delta t) - R(t)}{\Delta t} &= \nu I(t)
 \end{aligned}$$

0.15 A derivative arises as $\Delta t \rightarrow 0$

If we let $\Delta t \rightarrow 0$, we get derivatives on the left-hand side:

$$\begin{aligned}S'(t) &= -\beta \frac{S(t)I(t)}{N} \\I'(t) &= \beta \frac{S(t)I(t)}{N} - \nu I(t) \\R'(t) &= \nu R(t)\end{aligned}$$

This is a system of differential equations for the functions $S(t)$, $I(t)$, $R(t)$. For a unique solution, we need $S(0)$, $I(0)$, $R(0)$.

0.16 The ODE system cannot be solved analytically

Recall the Forward Euler method: Approximate the derivative with a *finite difference*, e.g.,

$$S'(t) \approx \frac{S(t + \Delta t) - S(t)}{\Delta t}$$

and rearrange to get formulas like

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

This brings us back to the first model, which we solved using a `for`-loop.

0.17 Or use a prebuilt solver like `ODESolver`

Implement the right hand side of the ODE system as a Python function:

```
def SIR_model(u,t):
    beta = 0.06
    nu = 0.008333
    S, I, R = u[0], u[1], u[2]
    N = S+I+R
    dS = -beta*S*I/N
    dI = beta*S*I/N - nu*I
    dR = nu*I
    return [dS,dI,dR]
```

0.18 Let us extend the model: no life-long immunity

Assumption. After some time, people in the R category lose the immunity. In a small time Δt this gives a leakage $\Delta t \gamma R$ to the S category. ($1/\gamma$ is the mean time for immunity.)

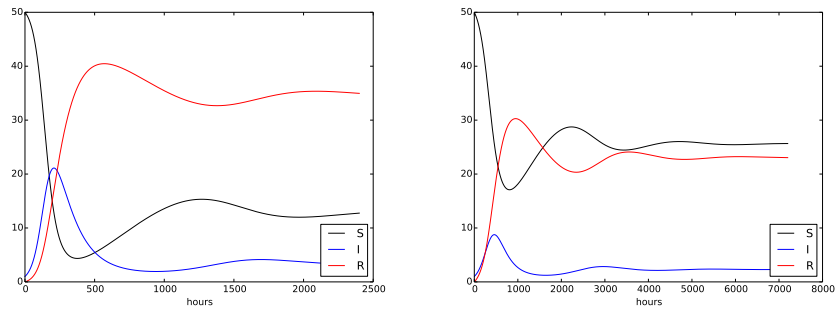


$$\begin{aligned}
S'(t) &= -\beta \frac{S(t)I(t)}{N} + \gamma R \\
I'(t) &= \beta \frac{S(t)I(t)}{N} - \nu I(t) \\
R'(t) &= \nu I(t) - \gamma R
\end{aligned}$$

No complications in the computational model!

0.19 The effect of loss of immunity

$1/\gamma = 50$ days. β reduced by 2 and 4 (left and right, resp.):



0.20 Adding more categories: the SEIR model

- Diseases have an *incubation period*, a delay from when a person gets infected until he/she has symptoms and can infect others
- For some applications, it is important to include the incubation period in the models
- Add a new category E (for exposed).
- People move from S to E as they are infected, then from E to I

$$\begin{aligned}
S'(t) &= -\beta SI/N + \gamma R, \\
E'(t) &= \beta SI/N - \mu E, \\
I'(t) &= \mu E - \nu I, \\
R'(t) &= \nu I - \gamma R.
\end{aligned}$$

0.21 The SEIR model implemented as a function

```
def SEIR(u,t):  
    S, E, I, R = u  
    N = S+I+R+E  
    beta=1.0; mu=1.0/5  
    nu=1.0/7; gamma=1.0/50  
    dS = -beta*S*I/N + gamma*R  
    dE = beta*S*I/N - mu*E  
    dI = mu*E - nu*I  
    dR = nu*I - gamma*R  
    return [dS,dE,dI,dR]
```

0.22 Parameter estimation is needed for predictive modeling

- Any small Δt will do
- One can reason about μ, ν, γ :
 - $1/\mu$ is the mean incubation time
 - $1/\nu$ is the mean recovery
 - $1/\gamma$ is mean duration of immunity
- β is more complex, since it depends both on the disease and how people behave

So, what if we don't know β ?

- Can still learn about the *dynamics* of diseases
- Can find the sensitivity to and influence of β
- Can apply *parameter estimation* procedures to fit β to data

0.23 A class is convenient for models with parameters

- The SEIR-function has all parameters explicitly defined in the code
- If we want to solve the model for multiple parameters, it is more convenient to implement it as a class
- A constructor (`__init__`) to set all the parameters, a `__call__` method to implement the ODE system

0.24 Class implementation of the SEIR model

```
class SEIR:
    def __init__(self, beta, mu, nu, gamma):
        self.beta = beta
        self.mu = mu
        self.nu = nu
        self.gamma = gamma

    def __call__(self, u, t):
        S, E, I, R = u
        N = S+I+R+E
        dS = -self.beta*S*I/N + self.gamma*R
        dE = self.beta*S*I/N - self.mu*E
        dI = self.mu*E - self.nu*I
        dR = self.nu*I - self.gamma*R
        return [dS, dE, dI, dR]
```