Introduction to ODE (review)

Example 1

We will consider the following problem at first.

Problem: Solve the following equation:

$$y'(t) = y(t) \quad (t \geq 0),$$
$$y(0) = a, \ \ (a : given\ constant) \ .$$

It is well known that the solution by Math is written in the formula

$$y(t) = a\, e^t.$$

Here, we will solve above by numerical method. Here we introduce Euler method. Discrete equation in time will be

$$y'(t) \sim \frac{y(t+h)-y(t)}{h}$$

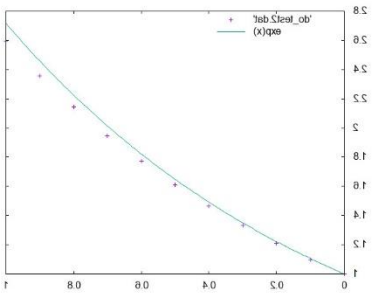
Then the equation will be:

$$\frac{y(t+h)-y(t)}{h} = y(t).$$

Thus, we get recurrence formula $y(t+h) = (1+h)y(t)$. Put $y_0 = a, \quad y_n = y(nh) \quad (n \geq 1)$ then

$$y_0 = a,$$
$$y_{n+1} = (1+h)y_n.$$

Example a = 0, h = 0.1, n = 10 t = 0 ~ 1.0



Example 2 (Lotka-Volterra type equation)

$$\begin{cases} \frac{dx}{dt}(t) = a\,x\,(t) - b\,x(t)y(t) \\ \frac{dy}{dt}(t) = c\,x(t)y(t)\ - d\,y(t) \end{cases} \quad a,\ b,\ c,\ d > 0$$

$$x(0) = x_0,\ y(0) = y_0$$

This system is well known as Pray–Predator model. (x : number of Pray, y : number of Predator)

Using Euler method

$$\begin{cases} \frac{x(t+\Delta t)-x(t)}{\Delta t} = a\,x\,(t) - b\,x(t)y(t) \\ \frac{y(t+\Delta t)-y(t)}{\Delta t} = c\,x(t)y(t)\ - d\,y(t) \end{cases}$$

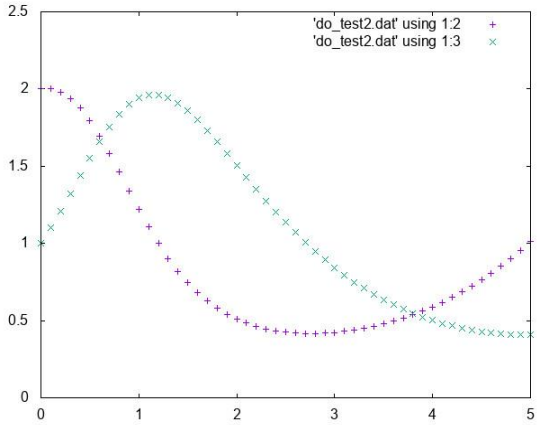
➔ $$\begin{cases} x(t+\Delta t) = (1 + a\,\Delta t\ - b\,\Delta t\,y(t))x\,(t) \\ y(t+\Delta t) = (1 + c\,\Delta t\,x(t)\ - d\,\Delta t\,)y(t) \end{cases}$$

(Recursion formula;) $\begin{cases} x_{i+1} = (1 + a\,\Delta t\ - b\,\Delta t\,y_i)x_i \\ y_{i+1} = (1 + c\,\Delta t\,x_i\ - d\,\Delta t\,)y_i \end{cases}$ $i = 0,\ 1,\ \ldots$

Here, $x_i \coloneqq x(i\,\Delta t),\ y_i \coloneqq y(i\,\Delta t)$

- ($a = b = c = d = 1.0,\ \Delta t = 0.1,\ n = 50,\ x_0 = 2.0,\ y_0 = 1.0$)

prey(x), predator(y)



Sample program by python (Lotka-Volterra)

```
-------------------------------- start program ------------------------------
import numpy as np                              #inport library 'numpy' to use arrays etc.
import matplotlib.pyplot as plt          #inport library 'matplotlib' to plot, for shortly 'plt'.
"""
Solve the following initil value problem for system of
first order differential equations:
x'(t) = a x(t) - b x(t) y(t)
y'(t) = c x(t) y(t) - d y(t)
x(0) = x0
y(0) = y0
"""
##### Parameters ###############
x0 = 2.0
y0 = 1.0
a = 1.0
b = 1.0
c = 1.0
d = 1.0
dt = 0.1
tmin = 0.0
tmax = 5.0 #simulate time
nt = int((tmax-tmin)/dt) + 1
###################################


x = np.zeros(nt)       # Initialization of x. Set x as array with size nt,
y = np.zeros(nt)
                                # and all components is zero.
########## Set x[0] ################
x[0] = x0
y[0] = y0
#########################################


############ Time developing ####################
for t in range(0, nt-1):
      x[t+1] = (1.0 + a * dt - b * dt * y[t]) * x[t]
```

```
    y[t+1] = (1.0 + c * dt * x[t] - d * dt) * y[t]
    t = t + 1
################################################

############## Plot ################################
ts = np.linspace(0, 1, nt)
plt.title("Solution with dt = 0.01")
plt.plot(ts, x)
plt.plot(ts, y)
plt.xlabel("time")
plt.show()

---------------------------------- end of program ------------------
```

Example 3　SIR model

SIR-model ([1]) was developed around one hundred years ago to analyze Spanish Flu spread. In this SIR model, all population is divided into three kinds:

$S(t)$: Susceptible,

$I(t)$: Infected

$R(t)$: Recovered　or Removed

We assume total population $N = S(0)$. The susceptible individuals (S) change into infected individuals (I) proportionate to $\frac{S}{N} \times I$ with ratio β. Thus, we can say:

β: the infection ratio,　It represents the ratio of transmitting disease from one infectious individual ([2]).　It can be translated number of new infected individuals which is transmitted from one infected person per day.

γ: the recovery factor (or rate).　It is the ratio an infected recovers and moves into the resistant phase. The infected individuals change into recovered individuals (R) and infected period follows exponential distribution. Roughly saying, it means the probability which infected individuals recovering in one certain period. It is also said the quarantine (or isolation) ratio. At the very beginning of epidemic, quarantine action work mainly on reducing inclement of $I(t)$.

If we use continuous number to describe person number, the model equations are the following:

$$\begin{cases} \frac{dS}{dt}(t) = -\frac{\beta}{N} S(t) I(t) \\ \frac{dI}{dt}(t) = \frac{\beta}{N} S(t) I(t) - \gamma I(t). \\ \frac{dR}{dt}(t) = \gamma I(t) \end{cases}$$

If one normalizes each unknown (i.e., $s(t) = \frac{S(t)}{N}$, $i(t) = \frac{I(t)}{N}$, $r(t) = \frac{R(t)}{N}$), we have

$$\begin{cases} \frac{ds}{dt}(t) = -\beta\, s(t) i(t) \\ \frac{di}{dt}(t) = \beta\, s(t) i(t) - \gamma\, i(t). \\ \frac{dr}{dt}(t) = \gamma\, i(t) \end{cases} \quad (1.1)$$

Please note that the equality:

$$\frac{d}{dt}(s(t) + i(t) + r(t)) = 0$$

holds. This means total population is constant (no one died or going out and no one birth or coming in).

Problem (Solving Equation)

We can solve the equation (1.1) numerically with the appropriate choice of β and γ and initial data $s(0) = s_0,\ i(0) = i_0,\ R(0) = r_0$.

The model has several important parameters (or quantity) which the behavior of the solution changes dramatically. For example, if one look at the second equation of (1.1), it is easy to see the sign of $\beta\, s(t)i(t) - \gamma\, i(t)$ plays an essential roll. If it is positive $i(t)$ increases, if negative it decreases.

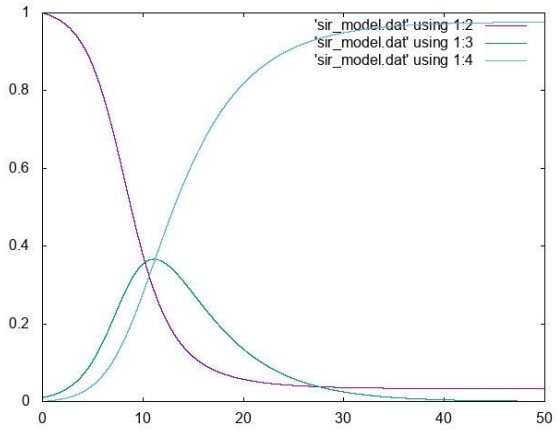
If, at some time $t = t_0$, $(\beta s(t_0) - \gamma)\, i(t_0)$, is positive, infected individuals are increasing. So, we call $\mathcal{R}_0 = \beta s(t_0)/\gamma$ effective reproduction number (基本再生産数).

基本再生産数は 1 が境目で感染者が増大するか減少するかどうか決まる。

So, the researchers are always taking care of this number and want to control it. They need to determine it by observing real data. Here we need to take care that the parameter γ is controllable by quarantine (or removing) some percentage of person artificially from the system. This is the reason why they request government to lockdown cities. Of course, the locking down action effects on $s(t_0)$, $i(t_0)$ and $r(t_0)$. (Note that they are not independent.)

- Example

$\beta = 0.5, \gamma = 0.2, S_0 = 1.0, I_0 = 0.01, R_0 = 0.0, \Delta t = 0.001, n = 50000.$



```
import numpy as np                          #inport library 'numpy' to use arrays etc.
import matplotlib.pyplot as plt          #inport library 'matplotlib' to plot, for shortly 'plt'.
##### Parameters ###############
beta = 0.7
gamma = 0.2
s0 = 1.0
infec0 = 0.01
r0 = 0.0
dt = 0.001
tmin = 0.0
tmax = 50.0 #simulate time
nt = int((tmax-tmin)/dt) + 1
##################################
s = np.zeros(nt)       # Initialization of x. Set x as array with size nt,
infec = np.zeros(nt)
r = np.zeros(nt)                            # and all components is zero.
########## Set x[0] ################
s[0] = s0
infec[0] = infec0
r[0] = r0
############ Time developing ####################
for t in range(0, nt-1):
    s[t+1] = (1.0 - beta * dt * infec[t]) * s[t]
    infec[t+1] = (1.0 + beta * dt * s[t] - gamma * dt) * infec[t]
    r[t+1] = r[t] + dt * gamma * infec[t]
    t = t + 1
############ Plot ################################
ts = np.linspace(0, 50, nt)
plt.title("Solution with beta = 0.7, gamma = 0.2")
plt.plot(ts, s, label = "S(t)")
plt.plot(ts, infec, label = "I(t)")
plt.plot(ts, r, label = "R(t)")
plt.xlabel("time")
plt.legend()
plt.show()
```