Intelligent Data Analysis

University of Science - Ho Chi Minh National University

Department of Computer Science

Final Project
A climate model

Author:
Ngoc-Duc Nguyen (email: nnduc@fit.hcmus.edu.vn)

Date: November 13, 2021

1 Background: Ordinary Differential Equation

Ordinary Differential Equation (ODE) can be used to describe a dynamic system. To some extent, we are living in a dynamic system, the weather outside of the window changes from dawn to dusk, the metabolism occurs in our body is also a dynamic system because thousands of reactions and molecules got synthesized and degraded as time goes.

More formally, if we define a set of variables, like the temperatures in a day, or the amount of molecule X in a certain time point, and it changes with the independent variable (in a dynamic system, usually it will be time t). ODE offers us a way to mathematically depict the dynamic changes of defined variables. The opposite system to that is called static system, thinking about taking a photo of the outside, this snapshot doesn't contain any dynamics, in another word, it is static.

Solving Ordinary Differential Equations means determining how the variables will change as time goes by, the solution, sometimes referred to as solution curve, provide informative prediction to the default behavior of any dynamic systems.

1.1 Example

Solving a sigmoidal signal-response curve

A sigmoidal signal-response curve describes a system like below:

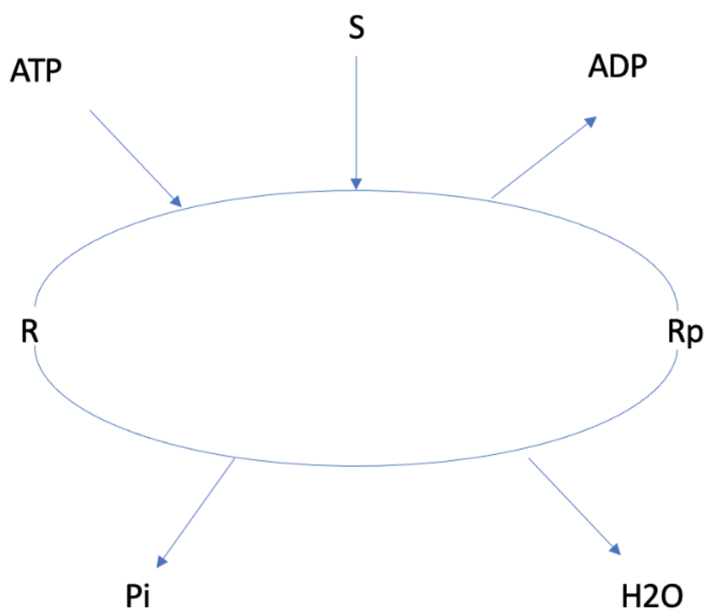

Figure 1: Phosphorylation process in the body (Source: Tyson et al. [2003])

It is the phosphorylation process that happened in our body, R is the substance, which will be converted to the phosphorylated form Rp to exert a lot of functions, this process is catalyzed by Signal S, and the ATP hydrolysis accompanying that. To describe this process, the original paper utilized the following ODE:

$$\frac{dR_p}{dt} = \frac{k_1 S(R_T - R_P)}{k_{m1} + R_T - R_P} - \frac{k_2 R_P}{k_{m2} + R_P}$$

Now we first want to solve this equation, meaning to obtain the solution curve that describes how Rp would change with time.

```
from scipy.integrate import odeint
import numpy
import matplotlib.pyplot as plt
def model(Rp,t,S):
    k1 = 1
    k2 = 1
    Rt = 1
    km1 = 0.05
    km2 = 0.05
    dRpdt = (k1*S*(Rt-Rp)/(km1+Rt-Rp)) - k2*Rp/(km2+Rp)
    return dRpdt
```

The above code is just to set up the model as we described using a mathematical equation, here some parameters we just used what has been provided in Tyson et al. [2003]

We set the signal strength as 1, but it is really just for illustration, feel free to change it to whatever value you prefer. Then we set the initial value of Rp to three different possibilities: 0, 0.3, and 1. It can show us how different initialization will finally converge to the steady-state. We set the simulation time window from 0 to 20 just for simplicity. Finally, we use `odeint` function to solve this ODE.

```
S = 1
Rp0 = [0,0.3,1]
t = np.linspace(0,20,200)
result = odeint(model,Rp0,t,args=(S,))
```

The `result` object is a NumPy array of the shape [200,3], 3 corresponds to three initialization conditions. Now we plot that:

```
fig,ax = plt.subplots()
ax.plot(t,result[:,0],label='R0=0')
ax.plot(t,result[:,1],label='R0=0.3')
ax.plot(t,result[:,2],label='R0=1')
ax.legend()
ax.set_xlabel('t')
ax.set_ylabel('Rp')
```

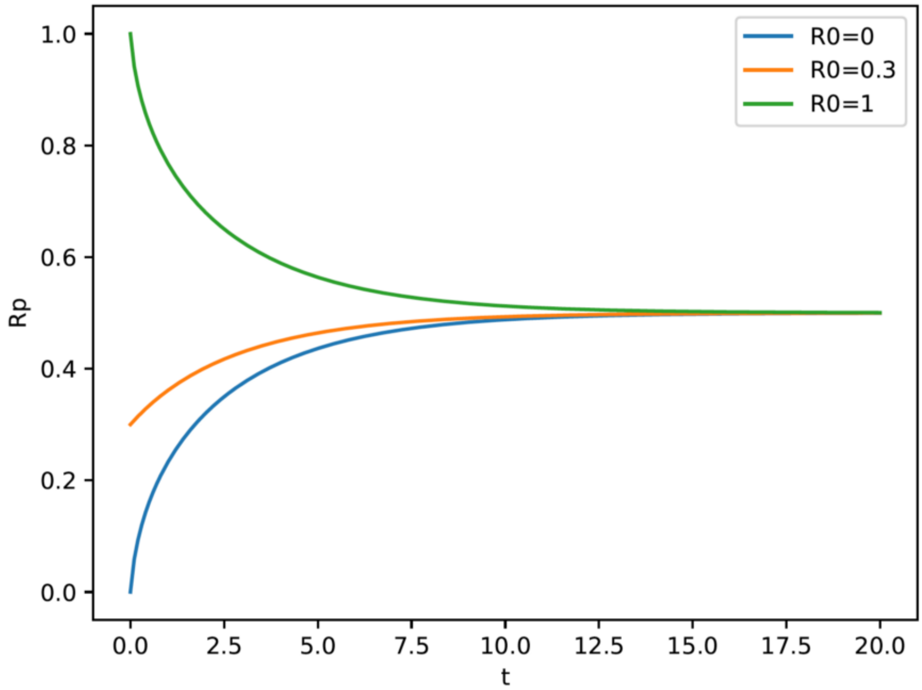

Figure 2: Sigmoidal response, solution curve

Here we can see, no matter where the Rp starts, they will converge to somewhere around 0.5, and this is called a steady-state. Steady-state is the gist of ODE because it defines the default behavior of a dynamic system. You may wonder why we call it a sigmoidal signal response curve? That is because apart from solving the solution curve, we are also interested in knowing how the signal strength will act upon the whole system, in ODE language, how signal strength will change the steady-state of the system? Now we are going to explore it!

Mathematically, a steady-state of a system is basically the root of the formula when setting $\frac{dR_p}{dt} = 0$. A better illustration is as below:

$$\frac{dR_p}{dt} = 0$$

$$\frac{k_1 S(R_T - R_P)}{k_{m1} + R_T - R_P} - \frac{k_2 R_P}{k_{m2} + R_P} = 0$$

So if we can solve the last equation with respect to Rp, we will have the value of Rp in the steady-state.

```
S_all = np.linspace(0,3,100)
def equation(Rp,S):
    k1 = 1
    k2 = 1
```

```
    Rt = 1
    km1 = 0.05
    km2 = 0.05
    return k1*S*(Rt-Rp)/(km1+Rt-Rp) - k2*Rp/(km2+Rp)

from scipy.optimize import fsolve
store = []
for S in S_all:
    Rp_ss = fsolve(equation,[1],args=(S,))[0]
    store.append(Rp_ss)
```

We first set the range of signal S from 0–3, then we use `fsolvefunction` from `scipy.optimize` to do the job. The result basically will be the Rp value when S is equal to the different values within 0–3.

Now let's take a look at result:

```
fig,ax = plt.subplots()
ax.plot(S_all,store,c='k')
ax.set_xlim(0,3)
ax.set_xlabel('Signal(S)')
ax.set_ylim(0,1.1)
ax.set_ylabel('Response(R_ss)')
```

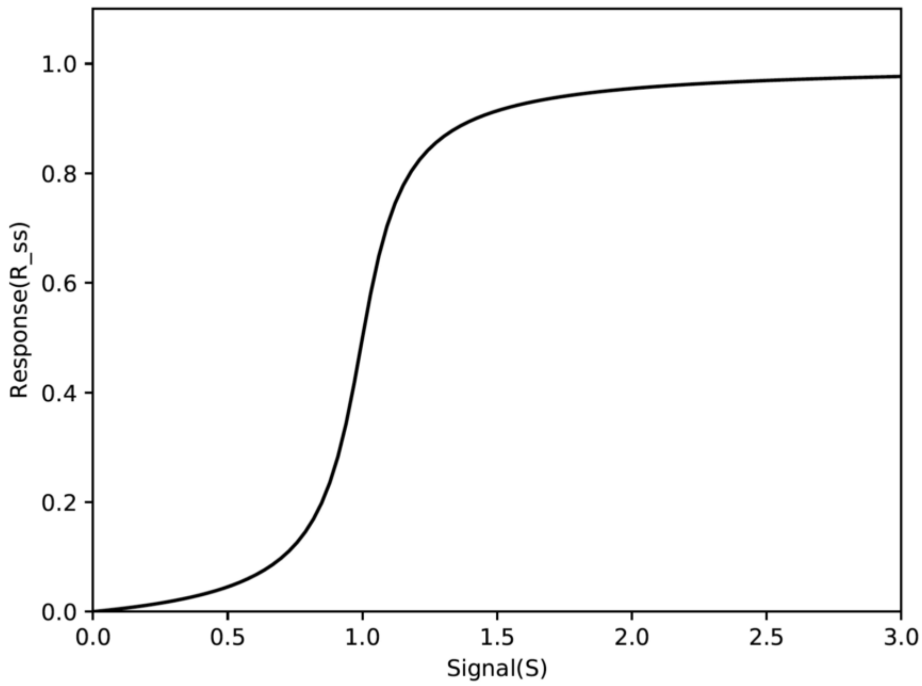

Figure 3: Signal response curve

Now it becomes clear why it is called the sigmoidal signal-response curve because as the strength of the signal changes, the steady-state of the system will respond in a sigmoidal manner.

2 Background: climate physics

The simplest climate model can be conceptualized as:

$$\text{change in heat content} = + \text{absorbed solar radiation (energy from the Sun's rays)} \tag{1}$$

$$- \text{outgoing thermal radiation (i.e. blackbody cooling to space)} \tag{2}$$

$$+ \text{human-caused greenhouse effect (trapped outgoing radiation)} \tag{3}$$

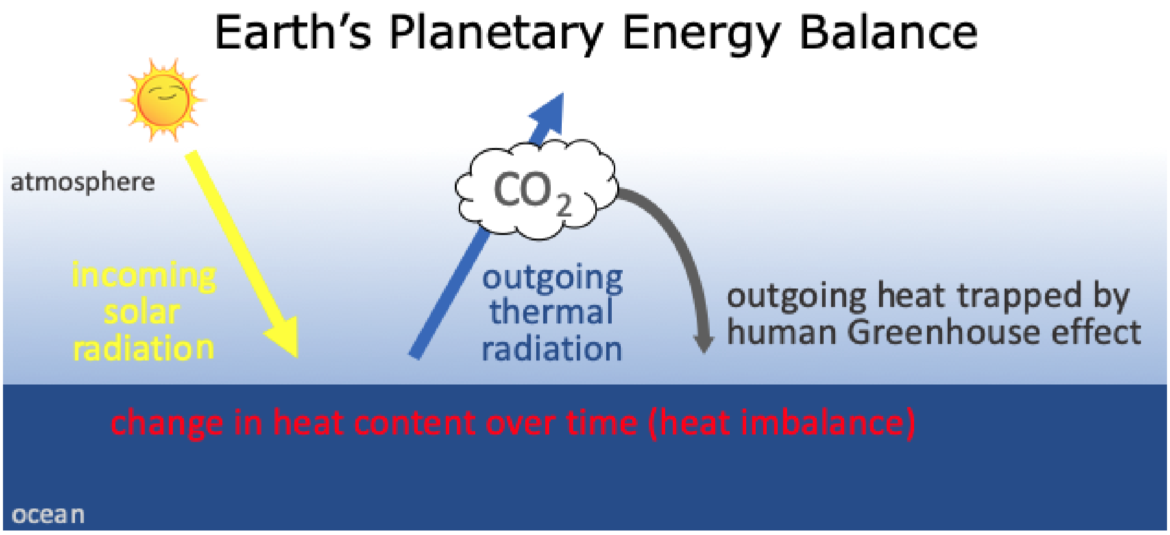

Figure 4: Earth's Planetary Energy Balance

2.1 Absorted solar radiation

In the physical world there are physical variables to identify.
In our baking the earth example, we will identify the following quantities:

- Industrial Revolution Start: 1850
- Avg Temperature in 1850: 14.0 °C
- Solar Insolation $S = 1368 W/m^2$: energy from the sun
- Albedo or plentary reflectivity:$\alpha = 0.3$
- atmosphere and upper-ocean heat capacity: $C = 51 J/m^2/C$

Earth Baking Formula: $C\ \text{temp}'(t) = S(1-\alpha)/4$.

At Earth's orbital distance from the Sun, the power of the Sun's rays that intercept the Earth is equal to $S = 1368$. A small fraction $\alpha = 0.3$ of this incoming solar radiation is reflected back out to space (by reflective surfaces like white clouds, snow, and ice), with the remaining fraction $(1-\alpha)$ being absorbed.

Since the incoming solar rays are all approximately parallel this far from the Sun, the cross-sectional area of the Earth that intercepts them is just a disc of area πR^2. Since all of the other terms we will consider act on the entire surface area $4\pi R^2$ of the spherical Earth, the absorbed solar radiation *per unit surface area* (averaged over the entire globe) is reduced by a factor of 4..

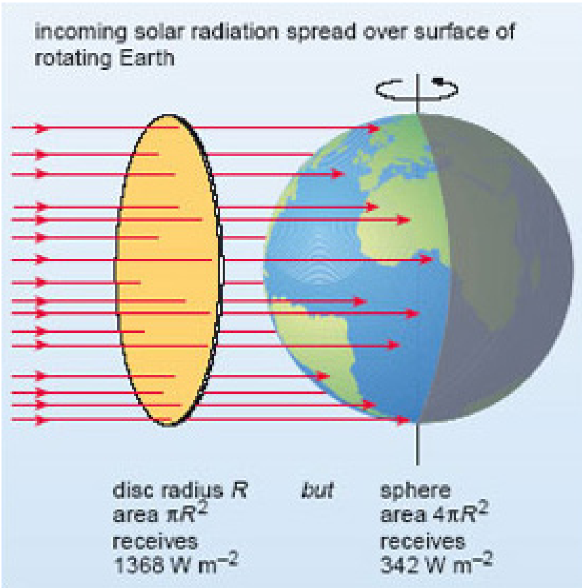

Figure 5: Incoming solar radiation

The absorbed solar radiation per unit area is thus absorbed solar radiation $\equiv \frac{S(1-\alpha)}{4}$ The heat content $Ctemp$ is determined by the temperature $temp$ (in Kelvin) and the heat capacity of the climate system. While we are interested in the temperature of the atmosphere, which has a very small heat capacity, its heat is closely coupled with that of the upper ocean, which has a much larger heat capacity of.

The *change in heat content over time* is thus simply given by $\frac{d(CT)}{dt}$. Since the heat capacity of sea water hardly changes with temperature, we can rewrite this in terms of the change in temperature with time as:

$$\text{change in heat content} = C\frac{dtemp}{dt}$$

Your work here is modeling ODE with only absorbed solar radiation to predict the change in temperature for 200 years from 1850 and visualizing your model

2.2 Outgoing: thermal radiation

The outgoing thermal radiation term $\mathcal{G}(T)$ (or "blackbody cooling to space") represents the combined effects of *negative feedbacks that dampen warming*, such as **blackbody radiation**, and *positive feedbacks that amplify warming*, such as the **water vapor feedback**.

Since these physics are too complicated to deal with here, we *linearize* the model combining the incoming and the outgoing.

We assume that the preindustrial world was in energy balance, and thus the equilibrium temperature is the preindustrial temperature.

Thus we assume temp'(t) = B(temp(0)-temp(t)) for some value of B. The minus sign in front of temp(t) indicating it restores equilibrium. The climate feedback parameter has been chosen is $B = 1.3W/m^2/°C$.

Extending your model with thermal radiation and visualizing it

2.3 Greenhouse: Human-caused greenhouse effect

Empirically, the greenhouse effect is known to be a logarithmic function of gaseous carbon dioxide (CO_2) concentrations

$$\text{human-caused greenhouse effect} = (\text{forcing\_coef}) \ln\left(\frac{[\text{CO}_2]}{[\text{CO}_2]_{\text{PreIndust}}}\right),$$

where

- CO_2 forcing coefficient is $5.0W/m^2$
- preindustrial CO_2 concentration is 280.

How this depends on time into the future depends on human behavior! Time is not modelled in the above equation. Through observations to 2020 (figure 6), we can assume that the CO_2 concentration (ppm) increase each t year from 1850 by $1 + \left(\frac{t}{220}\right)^3$ times.

Extending your model with Greenhouse Effects and visualizing it

3 Experiments

1. Building a model for temparature prediction using **Python** programming language.
2. Predicting the temperature change in 2030.
3. Comparing your model predictions with NASA observations.

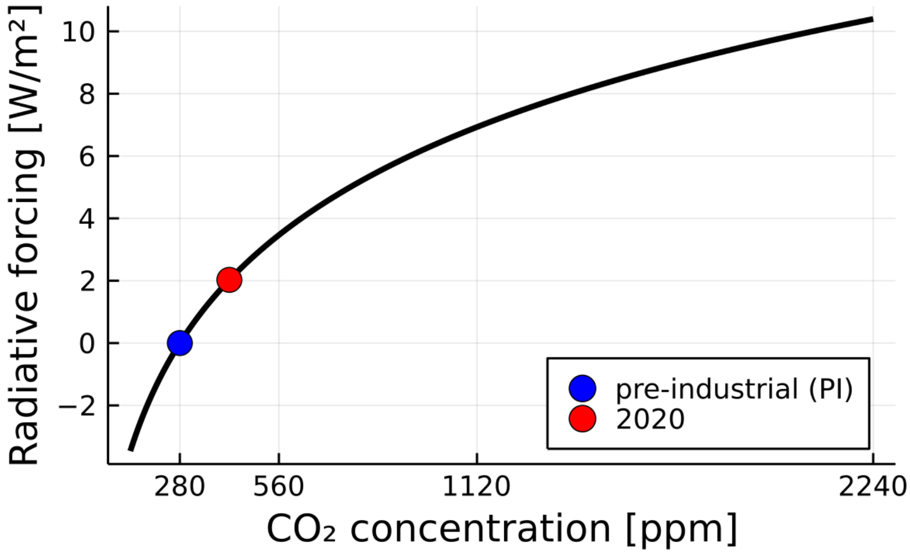

Figure 6: The change of CO_2 concentration from PI to 2020

4 Submission

Each group (2-3 members) submits a compressed file (GroupID.zip), including:

- An ipython notebook (GroupID.ipynb) containing the analysis. Be sure the results are reproducible, meaning they remain unchanged after rerunning all cells. You can use any tool you know to conduct the analysis.
- A Data directory containing all relevant data files. If any file in the data set is too large, you may replace it with a text file containing the link to the file (be sure it is publicly accessible).

Improving your model for additional points.

References

Alan Edelman, D. P. Sanders, and C. E. Leiserson. **Introduction to Computational Thinking**. URL `https://computationalthinking.mit.edu/Spring21`. pages

F. Li. **Ordinary Differential Equation in Python**. URL `https://towardsdatascience.com/ordinal-differential-equation-ode-in-python-8dc1de21323b`. pages

J. J. Tyson, K. C. Chen, and B. Novak. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. **Current Opinion in Cell Biology**, 15(2):221–231, 2003. ISSN 0955-0674. doi: 10.1016/S0955-0674(03)00017-6. URL `https://www.sciencedirect.com/science/article/pii/S0955067403000176`. pages 2, 3