VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MATHEMATICAL MODELLING (CO2011)

Assignment (Semester 201)

Dynamical systems in forecasting Greenhouse Micro-climate

Advisor: Nguyễn Tiến Thịnh

HO CHI MINH CITY, JANUARY 2021

Contents

Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Nguyễn Đình Bảo Phúc	1852068	– Exercise 1 – Exercise 2 – Exercise 3 – Exercise 4 – Exercise 5	110%
2	Võ Nhật Thanh	1852738	– Exercise 1 – Exercise 2 – Exercise 3 – Exercise 4 – Exercise 5	110%
3	Trần Anh Dũng	1852306	– Exercise 1 – Exercise 2 – Exercise 3 – Exercise 4 – Exercise 5	110%
4	Nguyễn Minh Anh	1852008	– Exercise 1 – Exercise 2 – Exercise 3 – Exercise 4 – Exercise 5	110%
5	Lê Công Trường Thọ	1852768	– Exercise 2 – Pending – Pending – Pending	60%

1 Background

1.1 Introduction to dynamical systems

Dynamical system is a system that determine how one state of the system moves to another state of the set of fixed rules over time. This system can determine the present state in terms of past states. Besides that, it can predict the future laid on the basis for the analysis of nonlinear differential.

For instance, this system can store statistic and predict the evolution of population of bacteria.

The dynamical system has two parts: *The state space*, which is the set of all possible values of the state vector. *The state vector* consists of a set of variables whose values can be within certain interval.

This system is classified as continuous or discrete model. The biggest difference between two kinds of this system is about the state variable, which is changed only at countable number of points in time in discrete model while being changed in a continuous way in continuous system.

For the analysis, the system is considered as linear if the system behavior satisfies some basic properties or nonlinear system which does not satisfy previous properties. If the system does not depend on the independent variable, this is the autonomous one. We should consider the time variable, if the independent variable is time, we call it time-invariant system.

Another important characteristic of a dynamical system is whether it is time dependent or not. For time-dependent systems, the function that specifies $\dot{x}$ or Δx_n depends on the time itself, whereas in time-independent systems, this function does not change.

First order differential equation is the simplest dynamical model which has two phases: the initial state and function demonstrate the next state.

$$\begin{cases} \dot{y}(t) & = ay(x) \\ y(0) & = y_0 \end{cases}$$

$$a \in \mathbb{R}, \dot{y} = \frac{dy}{dx}$$

$$x_0 \in \mathbb{R}$$

In this assignment, we use the dynamical system especially the first order differential equation to predict the changes of CO_2 inside a greenhouse model (We consider the model of greenhouse in Texas, USA.)

1.2 Necessary and sufficient conditions for this dynamical system

The first order differential equations in the dynamical system with the initial value y_0 can be re-written as:

$$\begin{cases} \frac{dy}{dx} & = f(x, y) \\ y(x_0) & = y_0 \end{cases}$$

Suppose that we have a linear differential equation $y'(x) + p(x)y = g(x)$ with the initial value $y(x_0) = y_0$. In which, p and g are continuous functions on the open interval $I = (\alpha, \beta)$, and let $t_0 \in (\alpha, \beta)$. Then for each $t \in I$ there exists a unique solution $y = \phi(t)$ to the differential equation $\frac{dy}{dx} + p(x)y = g(x)$ that also satisfies the initial value condition that $y(x_0) = y_0$.

1.3 Examples of solvable first-order differential equations and their exact solutions

- First equation:

$$\begin{cases} \frac{dy}{dt} & = y - t^2 + 1 \\ y(0) & = 0.5 \end{cases}$$

This can be put into a form of a first-order linear differential equation: $\frac{dy}{dx} + P(x)y = Q(x)$ where P and Q are continuous functions on a given interval.

And general solution: $y = e^{-\int P(x)dx}(\int e^{\int P(x)dx}.Q(x)dx + C)$

We have, $\frac{dy}{dt} = y - t^2 + 1 \Rightarrow \frac{dy}{dt} - y = -t^2 + 1$ in which $P(t) = -1$ *and* $Q(t) = -t^2 + 1$

$$\begin{aligned} \Rightarrow \quad y &= e^{-\int P(t)dt}(\int e^{\int P(t)dt} \cdot Q(t)t + C) \\ y &= e^{-\int -1dt}(\int e^{\int 1dt} \cdot (-t^2 + 1)dt + C) \\ y &= e^{-}t(\int e^{t} \cdot (-t^2 + 1)dt + C) \\ y &= e^{-}t(e^{t} \cdot (t + 1)^2 + C) \\ y &= (t + 1)^2 + e^{t} \cdot C \end{aligned}$$

Combining the general solution with the given initial value, we obtain

$$\begin{aligned} 0.5 = y(0) &= (0 + 1)^2 + e^0 \cdot C = 1 + C \\ C &= -0.5 \end{aligned}$$

Thus, the particular solution is

$$y = (t + 1)^2 - 0.5e^t$$

- Second equation:

$$\begin{cases} \frac{dy}{dt} & = -1.2y + 7e^{-0.3t} \\ y(0) & = 3 \end{cases}$$

This can be put into a form of a first-order linear differential equation: $\frac{dy}{dx} + P(x)y = Q(x)$ where P and Q are continuous functions on a given interval.

And general solution: $y = e^{-\int P(x)dx}(\int e^{\int P(x)dx}.Q(x)dx + C)$

We have, $\frac{dy}{dt} = -1.2y + 7e^{-0.3t} \Rightarrow \frac{dy}{dt} + 1.2y = 7e^{-0.3t}$ in which $P(t) = 1.2$ *and* $Q(t) = 7e^{-0.3t}$

$$\Rightarrow \quad y = e^{-\int P(t)dt}(\int e^{\int P(t)dt} \cdot Q(t)t + C)$$

$$y = e^{-\int 1.2dt}(\int e^{\int 1.2dt} \cdot 7e^{-0.3t}dt + C)$$
$$y = 7e^{-1.2t}(\int e^{1.2t} \cdot e^{-0.3t}dt + C)$$
$$y = 7e^{-1.2t}(\int e^{0.9t}dt + C)$$
$$y = 7e^{-1.2t}(\frac{e^{0.9t}}{0.9} + C)$$
$$y = \frac{70}{9}e^{-0.3t} + 7e^{-1.2t} \cdot C$$

Combining the general solution with the given initial value, we obtain

$$3 = y(0) = \frac{70}{9}e^{-0.3\cdot 0} + 7e^{-1.2\cdot 0} \cdot C = \frac{70}{9} + 7C$$
$$7C = \frac{-43}{9}$$

Thus, the particular solution is

$$y = \frac{70}{9}e^{-0.3t} - \frac{43}{9}e^{-1.2t}$$

1.4 Explicit Euler and Explicit Runge–Kutta of order 4 algorithms

1.4.1 Explicit Euler algorithm

Given an initial value problem

$$\begin{cases} \frac{dy}{dt} = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

The operation which can not be evaluated numerically obviously is the limit h $\rightarrow$ 0, that defines the derivative

$$\frac{dy}{dt} = \lim_{h\to 0} \frac{y(t+h) - y(t)}{h}$$

However, for any positive (small) h, the finite difference $\frac{y(t+h)-y(t)}{h}$ can easily be evaluated. Thus,we can have the following approximation:

$$\frac{dy}{dt} \approx \frac{y(t+h) - y(t)}{h}$$

By definition, it is an approximation of the derivative y'(t). Let us therefore approximate the differential equation $\frac{dy}{dt} = f(t, y)$ by the difference equation

$$f(t, y(t)) \approx \frac{y(t+h) - y(t)}{h}$$

Given y at time t, one can compute y at the later time t + h, by solving the difference equation

$$y(t+h) \approx y(t) + h \cdot f(t, y(t))$$

This is exactly one step of the explicit Euler method. Introducing the notation $t_{n+1} = t_n + h$. Exact value of solution at t_k is denoted by $y(t_k)$, approximate value is denoted by y_k. We have Euler's formula

$$y(t_{n+1}) \approx y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

We can calculate $y(t_k)$ at any time t_k since we are provided with the initial condition $y(t_0) = y_0$ That is the approximation steps of the Explicit Euler algorithm to solve general first-order differential equations.

1.4.2 Explicit Runge–Kutta of order 4 algorithm

The formula for the fourth order Runge-Kutta method (RK4) is given below. Consider the problem

$$\begin{cases} \frac{dy}{dt} = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

Define h to be the time step size and $t_n = t_0 + n \cdot h$. Exact value of solution at t_n is denoted by $y(t_n)$, approximate value is denoted by y_n. Then the following formula

$$\begin{aligned} k_1 &= h \cdot f(t_n, y_n) \\ k_2 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= h \cdot f(t_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

computes an approximate solution, that is $y_n \approx y(t_n)$.

This method has local truncation error $O(h^4)$, provided the solution y(t) has five continuous derivatives. We introduce the notation k1, k2, k3, k4 into the method is to eliminate the need for successive nesting in the second variable of $f(t, y)$.

1.5 Applying Euler and Runge-Kutta algorithms, give approximate solution of the above examples

We use $h = 0.2$ for the first system of equations and $h = 0.5$ for second system of equations.

1.5.1 Explicit Euler

First system of equations:

$$\begin{cases} \frac{dy}{dt} & = y - t^2 + 1 \\ y(0) & = 0.5 \end{cases}$$

And its exact solution:

$$y = (t+1)^2 - 0.5e^t$$

With $t_0 = 0, h = 0.2 \quad \rightarrow \quad y_0 = 0.5$ and $t_k = t_0 + k \cdot h = 0.2k$.

Applying the explicit Euler's formula for approximate solution as stated in 1.4.1, we have:

$$\begin{aligned} y_1 &= y_0 + h \cdot (y_0 - t_0^2 + 1) \\ &= 0.5 + 0.2 \cdot (0.5 - 0^2 + 1) = 0.8 \\ y_2 &= y_1 + h \cdot (y_1 - t_1^2 + 1) \\ &= 0.8 + 0.2 \cdot (0.8 - 0.2^2 + 1) = 1.152 \\ y_3 &= y_2 + h \cdot (y_2 - t_2^2 + 1) \\ &= 1.152 + 0.2 \cdot (1.152 - 0.4^2 + 1) = 1.5504 \\ y_4 &= y_3 + h \cdot (y_3 - t_3^2 + 1) \\ &= 1.5504 + 0.2 \cdot (1.5504 - 0.6^2 + 1) = 1.98848 \\ y_5 &= y_4 + h \cdot (y_4 - t_4^2 + 1) \\ &= 1.98848 + 0.2 \cdot (1.98848 - 0.8^2 + 1) = 2.458176 \end{aligned}$$

Compare the appoximate solution (y_k) and exact solution ($y(t_k)$) we have the table for explicit Euler method as follows:

k	t_k	Approx.solution y_k	Exact solution $y(t_k)$	$\lvert y(t_k) - y_k \rvert$
0	0.0	0.5000000	0.5000000	0.0000000
1	0.2	0.8000000	0.8292986	0.0292986
2	0.4	1.1520000	1.2140877	0.0620877
3	0.6	1.5504000	1.6489406	0.0985406
4	0.8	1.9884800	2.1272295	0.1387495
5	1.0	2.4581760	2.6408591	0.1826831

Second system of equations:

$$\begin{cases} \frac{dy}{dt} & = -1.2y + 7e^{-0.3t} \\ y(0) & = 3 \end{cases}$$

And its exact solution:

$$y = \frac{70}{9}e^{-0.3t} - \frac{43}{9}e^{-1.2t}$$

With $t_0 = 0, h = 0.5 \quad \rightarrow \quad y_0 = 3$ and $t_k = t_0 + k \cdot h = 0.5k$.

Applying the explicit Euler's formula for approximate solution as stated in 1.4.1, we have:

$$\begin{aligned}
y_1 &= y_0 + h \cdot (-1.2 \cdot y_0 + 7e^{-0.3 \cdot t_0}) \\
&= 3 + 0.5 \cdot (-1.2 \cdot 3 + 7e^{-0.3 \cdot 0}) = 4.7 \\
y_2 &= y_1 + h \cdot (-1.2 \cdot y_1 + 7e^{-0.3 \cdot t_1}) \\
&= 4.7 + 0.5 \cdot (-1.2 \cdot 4.7 + 7e^{-0.3 \cdot 0.5}) = 4.892477917 \\
y_3 &= y_2 + h \cdot (-1.2 \cdot y_2 + 7e^{-0.3 \cdot t_2}) \\
&= 4.892477917 + 0.5 \cdot (-1.2 \cdot 4.892477917 + 7e^{-0.3 \cdot 1}) = 4.549854939 \\
y_4 &= y_3 + h \cdot (-1.2 \cdot y_3 + 7e^{-0.3 \cdot t_3}) \\
&= 4.549854939 + 0.5 \cdot (-1.2 \cdot 4.549854939 + 7e^{-0.3 \cdot 1.5}) = 4.051640506 \\
y_5 &= y_4 + h \cdot (-1.2 \cdot y_4 + 7e^{-0.3 \cdot t_4}) \\
&= 4.051640506 + 0.5 \cdot (-1.2 \cdot 4.051640506 + 7e^{-0.3 \cdot 2}) = 3.541496929
\end{aligned}$$

Compare the appoximate solution (y_k) and exact solution ($y(t_k)$) we have the table for explicit Euler method as follows:

k	t_k	Approx.solution y_k	Exact solution $y(t_k)$	$\lvert y(t_k) - y_k \rvert$
0	0.0	3.000	3.0000	0.0000
1	0.5	4.700	4.0720	0.6277
2	1.0	4.893	4.4323	0.5696
3	1.5	4.550	4.1700	0.3803
4	2.0	4.052	3.8350	0.2165
5	2.5	3.542	3.4360	0.1054

1.5.2 Explicit RK4

First system of equations:

$$\begin{cases} \frac{dy}{dt} & = y - t^2 + 1 \\ y(0) & = 0.5 \end{cases}$$

And its exact solution:

$$y = (t+1)^2 - 0.5e^t$$

With $t_0 = 0, h = 0.2 \quad \rightarrow \quad y_0 = 0.5$ and $t_k = t_0 + k \cdot h = 0.2k$.

From t = 0 to t = 1 with step size h = 0.2, it takes 5 steps: $t_0 = 0$, $t1 = 0.2$, $t2 = 0.4$, $t3 = 0.6$, $t4 = 0.8$, $t5 = 1$.

Step 0: $t_0 = 0$, $y_0 = 0.5$

Step 1: $t_1 = 0.2$

$k_1 = hf(t_0,\ y_0) = 0.2f(0, 0.5) = 0.3$

$k_2 = hf\left(t_0 + \frac{h}{2},\ y_0 + \frac{k_1}{2}\right) = 0.2f(0.1,\ 0.5 + \frac{0.3}{2}) = 0.328$

$k_3 = hf\left(t_0 + \frac{h}{2},\ y_0 + \frac{k_2}{2}\right) = 0.2f(0.1,\ 0.5 + \frac{0.328}{2}) = 0.3308$

$k_4 = hf(t_0 + h,\ y_0 + k_3) = 0.2f(0.2,\ 0.5 + 0.3308) = 0.35816$

$\rightarrow y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 0.5 + \frac{1}{6}(0.3 + 2 * 0.328 + 2 * 0.3308 + 0.35816)$

≈ 0.82929333333

Step 2: $t_2 = 0.4$

$k_1 = hf(t_1,\ y_1) = 0.2f(0.2, 0.82929333333) = 0.3578586667$

$k_2 = hf\left(t_1 + \frac{h}{2},\ y_1 + \frac{k_1}{2}\right) = 0.2f(0.3,\ 0.82929333333 + \frac{0.3578586667}{2}) = 0.3836445333$

$k_3 = hf\left(t_1 + \frac{h}{2},\ y_1 + \frac{k_2}{2}\right) = 0.2f(0.3,\ 0.82929333333 + \frac{0.3836445333}{2}) = 0.338622312$

$k_4 = hf(t_1 + h,\ y_1 + k_3) = 0.2f(0.4,\ 0.82929333333 + 0.338622312) = 0.4111032907$

$\rightarrow y_2 = y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 0.82929333333 + \frac{1}{6}\left(0.3578586667 + 2 * 0.3836445333 + 2 * 0.338622312 + 0.4111032907\right)$

≈ 1.214076211

Step 3: $t_3 = 0.6$

$k_1 = hf(t_2,\ y_2) = 0.2f(0.4, 1.214076211) = 0.4108152421$

$k_2 = hf\left(t_2 + \frac{h}{2},\ y_2 + \frac{k_1}{2}\right) = 0.2f(0.5,\ 1.214076211 + \frac{0.4108152421}{2}) = 0.4338967663$

$k_3 = hf\left(t_2 + \frac{h}{2},\ y_2 + \frac{k_2}{2}\right) = 0.2f(0.5,\ 1.214076211 + \frac{0.4338967663}{2}) = 0.4362049188$

$k_4 = hf(t_2 + h,\ y_2 + k_3) = 0.2f(0.6,\ 1.214076211 + 0.4362049188) = 0.4580562259$

$\rightarrow y_3 = y_2 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 1.214076211 + \frac{1}{6}\left(0.4108152421 + 2 * 0.4338967663 + 2 * 0.4362049188 + 0.4580562259\right)$

≈ 1.648922017

Step 4: $t_4 = 0.8$

$k_1 = hf(t_3,\ y_3) = 0.2f(0.6, 1.648922017) = 0.4577844034$

$k_2 = hf\left(t_3 + \frac{h}{2},\ y_3 + \frac{k_1}{2}\right) = 0.2f(0.7,\ 1.648922017 + \frac{0.4577844034}{2}) = 0.4775628437$

$k_3 = hf\left(t_3 + \frac{h}{2},\ y_3 + \frac{k_2}{2}\right) = 0.2f(0.7,\ 1.648922017 + \frac{0.4775628437}{2}) = 0.4795406878$

$k_4 = hf(t_3 + h,\ y_3 + k_3) = 0.2f(0.8,\ 1.648922017 + 0.4795406878) = 0.497692541$

$\rightarrow y_4 = y_3 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 1.648922017 + \frac{1}{6}\left(0.4577844034 + 2 * 0.4775628437 + 2 * 0.4795406878 + 0.497692541\right)$

≈ 2.127202685

Step 5: $t_5 = 1$

$k_1 = hf(t_4,\ y_4) = 0.2f(0.8, 2.127202685 = 0.497440537$

$k_2 = hf\left(t_4 + \frac{h}{2},\ y_4 + \frac{k_1}{2}\right) = 0.2f(0.9,\ 2.127202685 + \frac{0.497440537}{2}) = 0.5131845907$

$k_3 = hf\left(t_4 + \frac{h}{2},\ y_4 + \frac{k_2}{2}\right) = 0.2f(0.9,\ 2.127202685 + \frac{0.5131845907}{2}) = 0.5147589961$

$k_4 = hf(t_4 + h,\ y_4 + k_3) = 0.2f(1,\ 2.127202685 + 0.5147589961) = 0.5283923362$

$\rightarrow y_5 = y_4 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 2.127202685 + \frac{1}{6}\left(0.497440537 + 2 * 0.5131845907 + 2 * 0.5147589961 + 0.5283923362\right)$

≈ 2.640822693

Compare the appoximate solution (y_k) and exact solution $(y(t_k))$ we have the table for explicit RK4 method as follows:

t_k	Exact solution y_k	RK-4 solution $y(t_k)$	Error $\lvert y(t_k) - y_k \rvert$
0.0	0.5	0.5	0
0.2	0.829298620919915	0.829293333333333	0.000005287586582
0.4	1.214087651179365	1.214076210666667	0.000011440512698
0.6	1.648940599804746	1.648922017041600	0.000018582763146
0.8	2.127229535753766	2.127202684947944	0.000026850805823
1.0	2.640859085770477	2.640822692728752	0.000036393041726

Second system of equations:

$$\begin{cases} \frac{dy}{dt} & = -1.2y + 7e^{-0.3t} \\ y(0) & = 3 \end{cases}$$

And its exact solution: $y = \frac{70}{9}e^{-0.3t} - \frac{43}{9}e^{-1.2t}$
With $t_0 = 0, h = 0.5 \quad \rightarrow \quad y_0 = 3$ and $t_k = t_0 + kh = 0.5k$.

From t = 0 to t = 2.5 with step size h = 0.5, it takes 5 steps: $t_0 = 0$, $t1 = 0.5$, $t2 = 1.0$, $t3 = 1.5$, $t4 = 2.0$, $t5 = 2.5$.

Step 0: $t_0 = 0$, $y_0 = 3$

Step 1: $t_1 = 0.5$

$k_1 = hf(t_0, y_0) = 0.5f(0, 3) = 1.7$

$k_2 = hf\left(t_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 0.5f(0.25, 3 + \frac{1.7}{2}) = 0.9371022021$

$k_3 = hf\left(t_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 0.5f(0.25, 3 + \frac{0.9371022021}{2}) = 1.165971542$

$k_4 = hf(t_0 + h, y_0 + k_3) = 0.5f(0.5, 3 + 1.165971542) = 0.5128949926$

$\rightarrow y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= 3 + \frac{1}{6}(1.7 + 2 * 0.9371022021 + 2 * 0.9371022021 + 0.5128949926)$
≈ 4.069840413

Step 2: $t_2 = 1.0$

$$\begin{aligned} k_1 &= h \cdot f(t_1, y_1) && = 0.5 \cdot f(0.5, 4.069840413) && = 0.5705736695 \\ k_2 &= h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) && = 0.5 \cdot f(0.75, 4.069840413 + \frac{0.5705736695}{2}) && = 0.1817304168 \\ k_3 &= h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) && = 0.5 \cdot f(0.75, 4.069840413 + \frac{0.1817304168}{2}) && = 0.2983833926 \\ k_4 &= h \cdot f(t_1 + h, y_1 + k_3) && = 0.5 \cdot f(1.0, 4.069840413 + 0.2983833926) && = -0.02807051118 \end{aligned}$$

$$\begin{aligned} \rightarrow y_2 &= y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= 4.069840413 + \frac{1}{6}\left(0.5705736695 + 2 \cdot 0.1817304168 + 2 \cdot 0.2983833926 + -0.02807051118\right) \\ &\approx 4.320295543 \end{aligned}$$

Step 3: $t_3 = 1.5$

$$\begin{aligned} k_1 &= h \cdot f(t_1, y_1) && = 0.5 \cdot f(1, 4.320295543) && = 0.000686446 \\ k_2 &= h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) && = 0.5 \cdot f(1.25, 4.320295543 + \frac{0.000686446}{2}) && = -0.1868707839 \end{aligned}$$

$$k_3 = h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) = 0.5 \cdot f(1.25, 4.320295543 + \frac{-0.1868707839}{2}) = -0.1306036148$$
$$k_4 = h \cdot f(t_1 + h, y_1 + k_3) = 0.5 \cdot f(1.5, 4.320295543 + -0.1306036148) = -0.2821166262$$

$$\rightarrow y_3 = y_2 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$= 4.320295543 + \frac{1}{6}(0.000686446 + 2 \cdot -0.1868707839 + 2 \cdot -0.1306036148 + -0.2821166262)$$
$$\approx 4.167565713$$

Step 4: $t_4 = 2.0$

$$k_1 = h \cdot f(t_1, y_1) = 0.5 \cdot f(1.5, 4.167565713) = -0.2688408973$$
$$k_2 = h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) = 0.5 \cdot f(1.75, 4.167565713 + \frac{-0.2688408973}{2}) = -0.3494433835$$
$$k_3 = h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) = 0.5 \cdot f(1.75, 4.167565713 + \frac{-0.3494433835}{2}) = -0.3252626377$$
$$k_4 = h \cdot f(t_1 + h, y_1 + k_3) = 0.5 \cdot f(2, 4.167565713 + -0.3252626377) = -0.3845411191$$

$$\rightarrow y_4 = y_3 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$= 4.167565713 + \frac{1}{6}(-0.2688408973 + 2 \cdot -0.3494433835 + 2 \cdot -0.3252626377 + -0.3845411191)$$
$$\approx 3.833766704$$

Step 5: $t_5 = 2.5$

$$k_1 = h \cdot f(t_1, y_1) = 0.5 \cdot f(2.0, 3.833766704) = -0.3794192958$$
$$k_2 = h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) = 0.5 \cdot f(2.25, 3.833766704 + \frac{-0.3794192958}{2}) = -0.4043867613$$
$$k_3 = h \cdot f\left(t_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) = 0.5 \cdot f(2.25, 3.833766704 + \frac{-0.4043867613}{2}) = -0.3968965216$$
$$k_4 = h \cdot f(t_1 + h, y_1 + k_3) = 0.5 \cdot f(2.5, 3.833766704 + -0.3968965216) = -0.4088391746$$

$$\rightarrow y_5 = y_4 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$= 3.833766704 + \frac{1}{6}(-0.3794192958 + 2 \cdot -0.4043867613 + 2 \cdot -0.3968965216 + -0.3968965216)$$
$$\approx 3.435295864$$

Compare the appoximate solution (y_k) and exact solution ($y(t_k)$) we have the table for explicit RK4 method as follows:

t_k	Exact solution y_k	RK-4 solution $y(t_k)$	Error $\|y(t_k) - y_k\|$
0.0	3	3	0
0.5	4.072295333	4.069840413	0.002454920
1.0	4.322880482	4.320295543	0.002584939
1.5	4.169568713	4.167565713	0.002003000
2.0	3.835104726	3.833766704	0.001338022
2.5	3.436090528	3.435295864	0.000794664

2 Exercise 2

2.1 Restate the model of CO_2 concentration

2.1.1 CO_2 exchange

In nature, there are several factors that affect to the crop growth. In order to maximize the crop yield, it is necessary to control and predict these factors. The first factor we considered is CO_2 concentration. A greenhouse with thermal screen, which can help us not only on controlling CO_2 concentration but also light, temperature. Which means we can better control the climate to maximize the growth of crop.

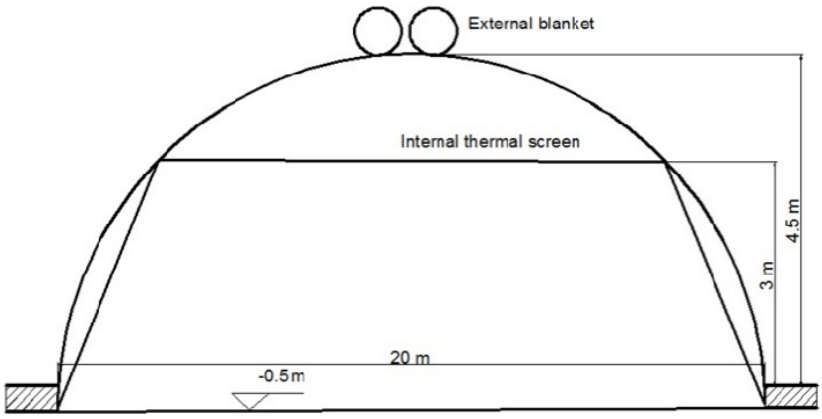


Figure 1: Greenhouse thermal screen

In figure 1, As you can see the greenhouse model is divided into two compartments by internal thermal screen. One part above the screen and the below one. The concentrations of CO_2 in two of the compartments are described as the figure below.

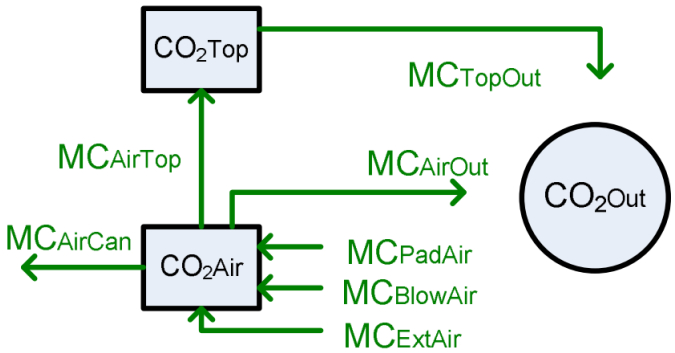


Figure 2: The CO2 flow inside and outside a greenhouse

- *Top*/*Air*: the compartment above/below the thermal screen
- *Out*/*Ext*: outside the greenhouse/external source

- *Blow/Pad*: the direct air heater/pad and fan
- *Can*: the canopy inside the greenhouse
- MC_{AB}: the net CO_2 flux from A to B.

2.1.2 Dynamical system models and assumptions

In this section, a dynamical system representing the CO_2 concentration in the greenhouse will be discussed. From figure 2, the fluctuation of CO_2 concentration in the lower and upper compartments of the greenhouse is represented by two differential equations.

$$\begin{cases} cap_{CO_2Air}\dot{CO_{2Air}} = MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} \\ \qquad\qquad\qquad\qquad -MC_{AirCan} - MC_{AirTop} - MC_{AirOut} \\ cap_{CO_2Top}\dot{CO_{2Top}} = MC_{AirTop} - MC_{TopOut} \end{cases} \tag{2.1}$$

- $cap_{CO_{2Top/Air}}$: capacity of the compartment above/below the thermal screen to store CO_2 (m)
- $CO_{2Top/Air}$: the rate change of CO_2 concentration in the compartment above/below the thermal screen in time ($mg\ m^{-3}\ s^{-1}$);
- MC_{AB}: the net CO_2 flux from A to B ($mg\ m^{-2}\ s^{-1}$)

In order to solve the equations, we must consider the formulas that calculate each MC_{AB} that was presented.

Here are formulas to calculate MC_{AB}. First, we consider the amount of CO_2 going from the heater into the greenhouse air as follows.

$$MC_{BlowAir} = \eta_{HeatCO_2} H_{BlowAir} = \frac{\eta_{HeatCO_2} U_{Blow}}{A_{Flr}} \tag{2.2}$$

- η_{HeatCO_2}: the amount of CO_2 released when 1 Joule sensible energy is produced by the direct air heater ($mg_{CO_2}\ J^{-1}$);
- $H_{BlowAir}$: the heat flux from the direct air heater to the greenhouse air ($W\ m^{-2}$);
- U_{Blow}: the control valve of the direct air heater ranging in $[0, 1]$;
- *Blow*: the heat capacity of the direct air heater W;
- *AFlr*: the area of the greenhouse floor m^2.

Similarly, the amount of CO_2 that is pumped into the greenhouse by the third party equals the third party's ability to pump CO_2 ϕ_{ExtCO_2} ($mg\ s^{-1}$) times the dimensionless parameter U_{ExtCO_2}, then divided by the area of the greenhouse.

$$MC_{ExtAir} = \frac{U_{ExtCO_2}\phi_{ExtCO_2}}{A_{Flr}} \tag{2.3}$$

On the other hand, the amount of CO_2 that enters the greenhouse through the pad system is calculated differently. It depends on the difference between the concentration of CO_2 inside and outside the greenhouse, and the ability of the pad system for the air to go through. Furthermore, the pad can be adjusted to let in more air. The following formula is used to calculate MC_{PadAir}.

$$\begin{aligned} MC_{PadAir} &= f_{Pad}(CO_{2Out} - CO_{2Air}) \\ &= \frac{U_{Pad}\phi_{Pad}}{A_{Flr}}(CO_{2Out} - CO_{2Air}) \end{aligned} \tag{2.4}$$

- f_{Pad}: the ventilation flux due to the pad and fan system (m^{-1})
- U_{Pad}: the control valve of the pad and fan system ranging in $[0, 1]$;
- ϕ_{Pad}: the capacity of the air flux through the pad $(m^3\ s^{-1})$

The net flux of CO_2 from the lower compartment to the upper compartment of the greenhouse is more complicated. It depends on the difference in temperature and air density between the two compartments and the airflow rate through the thermal screen f_{ThScr} $(m\ s^{-1})$.

$$MC_{AirTop} = f_{ThScr}(CO_{2Air} - CO_{2Top}) \tag{2.5}$$

Furthermore, f_{ThScr} is given by

$$f_{ThScr} = U_{ThScr}K_{ThScr}|T_{Air} - T_{Top}|^{\frac{2}{3}} + (1 - U_{ThScr})\left[\frac{g(1 - U_{ThScr})W}{2\rho_{Air}^{Mean}}|\rho_{Air} - \rho_{Top}|\right]^{\frac{1}{2}} \tag{2.6}$$

- f_{ThScr}: the air flux through the thermal screen$(m\ s^{-1})$.
- U_{ThScr}: the control of the thermal screen ranging in $[0, 1]$;
- K_{ThScr}: the screen flux coefficient determining the permeability of the screen $(m\ K^{-\frac{2}{3}}\ s^{-1})$.
- g: the gravitational acceleration $(m\ s^{-2})$.
- $\rho_{Air/Top}$:the density of the greenhouse air below/above the thermal screen $(kg\ m^{-3})$
- $2\rho_{Air}^{Mean}$: the mean density of the greenhouse air $(kg\ m^{-3})$
- $T_{Air/Top}$: the temperature below/above the thermal screen K

Similarly, for the net CO_2 flux from the inside to the outside of the greenhouse is given by the following formula

$$MC_{AirOut} = (f_{VentSide} + f_{VentForced})(CO_{2Air} - CO_{2Out}) \tag{2.7}$$

The flux $f_{VentSide}$ and $f_{VentForced}$ (ms^{-1}) are respectively the flux due to the fan system on the sidewalls and the fan system inside the greenhouse.

$$\begin{aligned} f_{VentRoofSide} = \frac{C_d}{A_{Flr}}\Bigg[&\frac{U_{Roof}^2U_{Side}^2A_{Roof}^2A_{Side}^2}{U_{Roof}^2A_{Roof}^2 + U_{Side}^2A_{Side}^2}. \times \frac{2gh_{SideRoof}(T_{Air} - T_{Out})}{T_{Air}^{Mean}} \\ &+ \left(\frac{U_{Roof}A_{Roof} + U_{Side}A_{Side}}{2}\right)^2 + C_W v_{Wind}^2\Bigg]^{\frac{1}{2}} \end{aligned} \tag{2.8}$$

- $f_{VentRoofSide}$: the ventilation rate through both the roof and side vents (ms^{-1})

- $C_{d/w}$: discharge/global wind pressure coefficient depending on the greenhouse shape and the use of an outdoor thermal screen $(-)$;
- $U_{Roof/Side}$: the control of the roof/side openings ranging in $[0, 1]$;
- A_{Side}: the roof/side opening area (m^2) .
- $h_{SideRoof}$: the vertical distance between mid-points of side wall and roof ventilation openings (m)
- T_{Air}^{Mean}: the mean temperature between the indoor and outdoor temperatures (K)
- v_{Wind}: wind speed $(m\ s^{-1})$

In the presence of an insect screen, the movement speed of the air currents through the ventilation areas will be reduced by a factor of η_{InsScr}, where ζ_{InsScr} is the porosity, the ratio of the area of the holes in the screen to the total area of the screen.

$$\eta_{InsScr} = \zeta_{InsScr}(2 - \zeta_{InsScr}) \tag{2.9}$$

Given the leakage coefficient $c_{leakage}$, which depends on the greenhouse type and is dimensionless, an amount of approximately half the leakage rate is added to the air-exchange rate, where leakage rate is calculated as below.

$$f_{leakage} = \begin{cases} 0.25 \times c_{leakage} & \text{if} v_{Wind} < 0.25 \\ v_{Wind} \times c_{leakage} & \text{if} v_{Wind} \geq 0.25 \end{cases} \tag{2.10}$$

Let η_{Side_Thr} be the Stack-effect threshold. If η_{Side}, the ratio between the sidewalls ventilation area and the total ventilation area, exceeds the threshold, the Stack effect does not occur and vice versa. Then, $f_{VentSide}$ is given by the following

$$f_{VentSide} = \begin{cases} \eta_{InsScr} f''_{VentSide} + 0.5 f_{leakage} & \text{if} \eta_{Side} \geq \eta_{Side\_Thr} \\ \eta_{InsScr}[U_{ThScr} f''_{VentSide} + \\ (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side}] + 0.5 f_{leakage} & \text{if} \eta_{Side} < \eta_{Side\_Thr} \end{cases} \tag{2.11}$$

In which, $f''_{VentSide}$ is $f_{VentSide}$ when $A_{Roof} = 0$. Moreover, the Stack effect does not occur where is covered by the thermal screen.

The flux $f_{VentForced}$ by the fan system inside the greenhouse is calculated as follows.

$$f_{VentForced} = \frac{\eta_{InsScr} U_{VentForced} \phi_{VentForced}}{A_{Flr}} \tag{2.12}$$

The dimensionless parameter $U_{VentForced} \in [0, 1]$ is to adjust the wind speed $\phi_{VentForced}$ due to the system $(m^3\ s^{-1})$.

Similarly to MC_{AirOut}, the net CO_2 flux from the greenhouse to outside the greenhouse through the roof openings is calculated by using the formula below, where $f_{VentRoof}$ is the flux rate through the roof openings.

$$MC_{TopOut} = f_{VentRoof}(CO_{2Top} - CO_{2Out}) \tag{2.13}$$

$f_{VentRoof}$ is the flux rate openings and is given by

$$f_{VentRoof} = \begin{cases} \eta_{InsScr} f''_{VentRoof} + 0.5 f_{leakage} & \text{if} \eta_{Roof} \geq \eta_{Roof\_Thr} \\ \eta_{InsScr}[U_{ThScr} f''_{VentRoof} + \\ (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side}] + 0.5 f_{leakage} & \text{if} \eta_{Roof} < \eta_{Roof\_Thr} \end{cases} \tag{2.14}$$

However, when the ratio of the roof-opening area to the total ventilation area exceeds the Stack-effect threshold, the Stack effect does not occur and we cannot reuse the formula 2.8 where $A_{Side} = 0$ to calculate $f''_{VentRoof}$.

$$f''_{VentRoof} = \frac{C_d U_{Roof} A_{Roof}}{2A_{Flr}} \left[\frac{g h_{Roof} (T_{Air} - T_{Out})}{2T^{Mean}_{Air}} + C_w v^2_{Wind} \right]^{\frac{1}{2}} \tag{2.15}$$

Finally, we need to consider the amount of CO_2 absorbed into the leaves due to photosynthesis.

$$MC_{AirCan} = M_{CH_2O} h_{C_{Buf}} (P - R) \tag{2.16}$$

Here,

- M_{CH_2O}:the molar mass of CH_2O $(mg\ \mu mol^{-1})$
- P: the photosynthetic rate $(\mu mol_{CO_2}\ m^{-2}\ s{-}1)$
- R: the respiration rate $(\mu mol_{CO_2}\ m^{-2}\ s^{-1})$
- $h_{C_{Buf}}$:shows the cessation of photosynthesis when CH_2O is C_{Buf} $(mg\ m^{-2})$ has reached C_{Max} $(mg\ m^{-2})$, which is the limit of the carbohydrates storage of the plants. The respiration rate during this process is usually as about 1% of the photosynthesis rate, and thus can be omitted during further calculation. The photosynthesis rate is described in more detail in Chapter 4.

2.1.3 Photosynthesis of C3 plants

Photosynthesis, the process by which green plants and certain other organisms transform light energy into chemical energy. During photosynthesis in green plants, light energy is captured and used to convert water, carbon dioxide, and minerals into oxygen and energy-rich organic compounds.

Photosynthesis has two phases consisting of the light-dependence phase and the light-independence (or dark) phase.

The photosynthetic rate P is defined as the diffusion of CO_2 from air into the leaf cells through stomata. From Fick's law for gas diffusion, we construct:

$$P = \frac{CO_{2Air} - CO_{2Stom}}{Res} \tag{2.20}$$

The notation CO_{2Stom} is the concentration of CO2 in the stomata $(\mu mol\ m^{-3})$ and Res is the resistance-to-absorption coefficient $(s\ m^{-1})$, this coefficient of resistance depends on many factors including the speed of the wind blowing through the leaves.

This way of calculating photosynthetic rate P is different from the model of Vanthoor 2011. In there the author uses:

$$P = \frac{J \cdot (CO_{2Stom} - \Gamma)}{4 \cdot (CO_{2Stom} + 2\Gamma)} \tag{2.21}$$

where J $(\mu mol\{e^-\}m^{-2}s^{-1})$ is the electron transport rate, 4 $(\mu mol\{e^-\}\mu mol^{-1}\{CO2\})$ is the number of electrons per fixed CO_2 molecule, CO_{2Stom} $(\mu mol\{CO_2\}\ mol^{-1}\{air\})$ is the CO_2 concentration in the stomata and Γ $(\mu mol\{CO_2\}mol^{-1}\{air\})$ is the CO_2 compensation point.

Following by the photorespiration R:

$$R = P \cdot \frac{\Gamma}{CO_{2Stom}} \tag{2.22}$$

But since we do not consider electron transport rate, we will use equation (20) constructed from Fick's law combined with Michaelis-Menten model.

In the dark phase, through Michaelis-Menten relationship describing enzyme-substrate reaction, the photosynthetic rate is given by:

$$P = \frac{P_{Max} \cdot CO_{2Stom}}{CO_{20.5} + CO_{2Stom}} \tag{2.23}$$

whereP_{Max} is the photosynthesis rate at saturating CO_{2Stom} and $CO_{20.5}$ is the concentration of CO_2 in the substrate when $P = P_{Max}/2$ ($\mu mol\ m^{-3}$).

We form a quadratic equation for the rate of photosynthetic P. The photosynthetic rate P then no longer depends on the concentration of CO_2 in the stomata but only on the concentration of CO_2 in the air, the resistance coefficient Res, and the maximum photosynthetic rate.

$$ResP^2 - (CO_{2Air} + CO_{20.5} + ResP_{Max})P + CO_{2Air}P_{Max} = 0 \tag{2.24}$$

Solving equation (22) requires the maximum rate of photosynthesis needs to be determined. For the model for the photosynthesis of one leaf unit, the maximum photosynthetic rate is usually be found through the Arrhenius model. P_{Max} and the Arrhenius model:

$$k(T) = k(T_0)e^{-\frac{H_a}{R}(\frac{1}{T} - \frac{1}{T_0})} \tag{2.25}$$

- T:the temperature of the leaf K
- T_0:a specific temperature of the leaf that we know the reaction rate K
- $k(T)$:the reaction rate $(-)$;
- H_a:the activation energy $(J\ mol^{-1})$,
- R:the ideal gas constant $(J\ mol^{-1})\ K^{-1}$,

Yet a problem is raised when the temperature increases to a certain threshold, the enzyme activity will be inhibited and the photosynthesis is slowed down and cease to advance. A model represents the activity of the Rubisco enzyme during photosynthesis with temperature as its parameter.

$$f(T) = \frac{1 + e^{-\frac{H_d}{R}(\frac{1}{T_0} - \frac{1}{\frac{H_d}{S}})}}{1 + e^{-\frac{H_d}{R}(\frac{1}{T} - \frac{1}{\frac{H_d}{S}})}} \tag{2.26}$$

In the model (24), $f(T)$ represents the enzyme activity at $T(K)$, H_d is the deactivation energy $(J\ mol^{-1})$, and S is the corresponding entropy quantity $(J\ mol^{-1}\ K^{-1})$.

Combining (23) and (24), we obtain the formula for maximum rate of photosynthetic rate.

$$P_{Max}(T) = k(T)f(T) \tag{2.27}$$

For a photosynthesis for the whole canopy, considering LAI (leaf area index), due to Beer's law, the intensity of the transmitted beam I with the initial state is I_0 ($\mu mol\{photons\}\ m^{-2}\ s^{-1}$) is equal to

$$I = \frac{I_0.K.e^{-K.LAI}}{1-m} \tag{2.28}$$

If the leaves are horizontally stratified such as in the case of tomato, the dimensionless extinction coefficient K will be between 0.7 and 1.0. Meanwhile, if the leaves are sloping as in the case of wet rice, K will be between 0.3 and 0.5. m is the transmittance coefficient of the leaves which is set as 0.1.

The amount of light absorbed by the canopy can be measured as the difference in the intensity of the light ray before entering the foliage and after passing through the foliage

$$L = L_0(1 - \frac{K.e^{-K.LAI}}{1-m}) \tag{2.29}$$

In this formula, L is luminous flux received by the leaves per unit area of the greenhouse floor ($\mu mol\{photons\}\ m^{-2}\ s^{-1}$) and L_0 is the initial value of L.

For calculating the maximum photosynthetic rate of all leaves in the greenhouse, we apply the modified Arrhenius model.

$$k(T) = LAI \cdot k(T_0) \cdot e^{-\frac{H_a}{R}(\frac{1}{T} - \frac{1}{T_0})} \tag{2.30}$$

- T:the temperature of the canopy K
- T_0: a specific temperature of the canopy that we know the reaction rate K
- $k(T)$: the reaction rate of the canopy at $T(-)$
- $k(T_0)$: the reaction rate in the stroma of a leaf at $T_0(-)$

Unlike the photosynthesis model for one leaf unit, the amount of light energy absorbed into the foliage in response to LAI needs to be added since it affects the maximum photosynthetic rate P_{Max}. Therefore, we consider the following formula of P_{Max}, which is a dependent function on L and T.

$$P_{Max}(L,T) = \frac{P_{MLT} \cdot P_{Max}(T) \cdot L}{L + L_{0.5}} \tag{2.31}$$

In which,

- L: the photosynthetically active radiation absorbed by the canopy($\mu mol\{photons\}\ m^{-2}\ s^{-1}$)
- $L_{0.5}$: the photosynthetically active radiation at which $P_{Max}(L,T) = P_{Max}(T)/2$ ($\mu mol\{photons\}\ m^{-2}\ s^{-1}$)
- P_{MLT}: the value of P_{Max} at saturation L and optimum T ($\mu mol\{CO_2\}\ m^{-2}\ s^{-1}$)

2.1.4 Vanthoor 2011 model of photosynthesis

For calculating photosynthesis rate and respiration rate, we we will use Equation (9.10) and relevant ones in reference [Van11].

Photosynthesis rate at canopy level, P, is described by (Farquhar, 1988):

$$P = \frac{J \cdot (CO_{2Stom} - \Gamma)}{4 \cdot (CO_{2Stom} + 2\Gamma)} \tag{2.32}$$

where J ($\mu mol\{e^-\}\ m^{-2}\ s^{-1}$) is the electron transport rate, 4 ($\mu mol\{e^-\}\ \mu mol^{-1}\{CO_2\}$) is the number of electrons per fixed CO_2 molecule, CO_{2Stom} ($\mu mol\{CO_2\}\ mol^{-1}\{air\}$) is the CO_2 concentration in the stomata and Γ ($\mu mol\{CO_2\}\ mol^{-1}\{air\}$) is the CO_2 compensation point.

The photorespiration, R, is described by (Farquhar & Von Caemmerer, 1982):

$$R = P \cdot \frac{\Gamma}{CO_{2Stom}} \tag{2.33}$$

The electron transport rate, J, is a function of the potential rate of electron transport and of the absorbed PAR by the canopy (Farquhar, 1988; Evans & Farquhar, 1991):

$$J = \frac{J^{POT} + \alpha PAR_{can} - \sqrt{(J^{POT} + \alpha PAR_{can})^2 - 4\Theta \cdot J^{POT} \cdot \alpha PAR_{can}}}{2\Theta} \tag{2.34}$$

where J^{POT} ($\mu mol\{e^-\}\ m^{-2}\ s^{-1}$) is the potential rate of electron transport, PAR_{Can} (PAR by the canopy) ($\mu mol\{photons\}\ m^{-2}\ s^{-1}$) is the absorbed PAR, α ($\mu mol\{e^-\}\ \mu mol^{-1}\{photons\}$) is the conversion factor from photons to electrons, including an efficiency term, and Θ ($-$) is the degree of curvature of the electron transport rate.

The potential rate of electron transport POT J , depends on temperature (Farquhar et al., 1980):

$$J^{POT} = J^{MAX}_{25,Can} \cdot \exp\left(E_j \frac{T_{Can,K} - T_{25,K}}{R \cdot T_{Can,K} \cdot T_{25,K}}\right) \cdot \frac{1 + \exp\left(\frac{S \cdot T_{25,K} - H}{R \cdot T_{25,K}}\right)}{1 + \exp\left(\frac{S \cdot T_{Can,K} - H}{R \cdot T_{Can,K}}\right)} \tag{2.35}$$

where $J^{MAX}_{25,Can}$ ($\mu mol\{e^-\}\ m^{-2}\ s^{-1}$) is the maximum rate of electron transport at 25℃ for the canopy, E_j ($J\ mol^{-1}$) is the activation energy for J^{POT}, $T_{Can,K}$ (K) is the canopy temperature, $T_{25,K}$ (K) is the reference temperature at 25℃, R ($J\ mol^{-1}\ K^{-1}$) is the molar gas constant, S ($J\ mol^{-1}\ K^{-1}$) is the entropy term and H ($J\ mol^{-1}$) is the deactivation energy.

The maximum rate of electron transport at 25°C for the canopy is calculated by (Evans & Farquhar, 1991):

$$J^{MAX}_{25,Can} = LAI \cdot J^{MAX}_{25,Leaf} \tag{2.36}$$

where $J^{MAX}_{25,Leaf}$, ($\mu mol\{e^-\}\ m^{-2}\{leaf\}\ s^{-1}$) is the maximum rate of electron transport for the leaf at 25℃.

In this exercise with assumption that PAR_Can is a constant

The total PAR absorbed by the canopy is the sum of the PAR transmitted by the greenhouse cover that is directly absorbed, and the PAR reflected by the greenhouse floor that is indirectly absorbed:

$$PAR_{Can} = PAR_{GhCan} + PAR_{FlrCan} \tag{2.37}$$

The PAR which is directly absorbed by the canopy is described by a negative exponential decay of light with LAI in a homogeneous crop (Ross, 1975):

$$PAR_{GhCan} = PAR_{Gh} \cdot (1 - \rho_{can}) \cdot (1 - \exp(-K_1 \cdot LAI)) \tag{2.38}$$

where PAR_{Gh} ($\mu mol\{photons\}\ m^{-2}\ s^{-1}$) is the PAR above the canopy, ρ_{can} ($-$) is the reflection coefficient of the canopy for PAR and K_1 is the extinction coefficient of the canopy for PAR ($-$).

The PAR above the canopy is described by:

$$PAR_{Gh} = \tau_{Gh} \cdot \eta_{Glob\_PAR} \cdot I_{Glob} \tag{2.39}$$

where τ_{Gh} ($-$) is the light transmission of the greenhouse cover η_{Glob_PAR} ($\mu mol\{photons\}\ J^{-1}$) is a conversion factor from global radiation to PAR and I_{Glob} ($W\ m^{-2}$) is the outside global radiation.

Absorption of PAR reflected by the greenhouse floor is described by:

$$PAR_{FlrCan} = \rho_{Flr} PAR_{Gh} \cdot (1 - \rho_{Can}) \cdot \exp(K_1 \cdot LAI) \cdot (1 - \exp(K_2 \cdot LAI)) \tag{2.40}$$

where ρ_{Flr} ($-$) is the reflection coefficient of the greenhouse floor and K_2 ($-$) is the extinction coefficient of the canopy when PAR is reflected from the floor to the canopy. We assumed K_2 to be equal to K_1.

The CO2-concentration inside the stomata, CO2Stom depends on the stomatal and mesophyl conductance, boundary layer resistance, the photosynthesis rate and the difference between the CO2-concentration in the stomata and the CO2-concentration of the greenhouse air. However, the CO2-concentration in the stomata is assumed to be a fixed fraction of the CO2-concentration in the greenhouse air (Evans & Farquhar, 1991):

$$CO_{2Stom} = \eta_{CO_{2Air\_Stom}} \cdot CO_{2Air} \tag{2.41}$$

where $\eta_{CO_{2Air\_Stom}}$ is conversion factor from the CO_2 concentration of the greenhouse air, CO_{2Air}, to the CO_2 concentration in the stomata.(Evans & Farquhar, 1991).

The CO_2 compensation point (Γ) affects the leaf photosynthesis rate and depends on temperature (Farquhar 1998) . To avoid unrealistically low optimal canopy temperatures, the sensitivity of the compensation point to temperature was adjusted by making the slope dependent of the ratio of $J^{MAX}_{25,Leaf}$ and $J^{MAX}_{25,Can}$:

$$\Gamma = \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}} c_\Gamma T_{Can} + 20 c_\Gamma \left(1 - \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}}\right) \tag{2.42}$$

where c_Γ ($\mu mol\{CO2\}\ mol^{-1}\{air\}\ K^{-1}$) determines the effect of canopy temperature on the CO_2 compensation point.

2.2 Implementing the code

- Two differential equations represents the fluctuation of $\{CO_2\}$ concentration:

$$cap_{CO_{2Air}} \dot{CO_{2Air}} = MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} - MC_{AirCan} - MC_{AirTop} - MC_{AirOut} \quad [mg\ m^{-2}\ s^{-1}]$$

$$cap_{CO_{2Top}} \dot{CO_{2Top}} = MC_{AirTop} - MC_{TopOut} \quad [mg\ m^{-2}\ s^{-1}]$$

All of the MC formula we need are demonstrated as below:

- $MC_{BlowAir} = \frac{\eta_{HeatCO_2} U_{Blow}}{A_{Flr}}$

```
def MC_blow_air():    #                (2.2)
    return (n_heat_Co2 * U_blow * P_blow)/A_flr
```

- $MC_{ExtAir} = \frac{U_{ExtCO_2}\phi_{ExtCO_2}}{A_{Flr}}$

```
def MC_ect_air():     #                (2.3)
    return (U_ext_Co2 * third_party_ability)/A_flr
```

- $MC_{PadAir} = f_{Pad}(CO_{2Out} - CO_{2Air})$

  $= \frac{U_{Pad}\phi_{Pad}}{A_{Flr}}(CO_{2Out} - CO_{2Air})$

```
def MC_pad_air():     #                (2.4)
    f_pad = (U_pad * ability_airflow)/A_flr
    return f_pad * (Co2_out - Co2_air)
```

- $MC_{AirTop} = f_{ThScr}(CO_{2Air} - CO_{2Top})$
  In order to calculate MC_{AirTop}, we need to find the value of f_{ThScr} first:

```
def p_air_average():
    temp = (g * M_air * h_elevation)/(293.15 * R)
    return (p_air_average_0 * math.exp(temp))
#?print(p_air_average())
def f_Th_Scr():       #                (2.6)
    diff_T = abs(T_air - T_top) ** (2.0/3.0)
    diff_p = abs(p_air - p_top)
    left_part = (((g * (1.0 - U_Th_Scr)) / (p_air_average() * 2.0)) *
    diff_p) ** (1.0/2.0)
    return (U_Th_Scr * K_Th_Scr * diff_T) + ((1.0 - U_Th_Scr) *
    left_part)
```

```
def MC_air_top():     #            (2.5)
    return f_Th_Scr() * (Co2_air - Co2_top)
```

- $MC_{AirOut} = (f_{VentSide} + f_{VentForced})(CO_{2Air} - CO_{2Out})$
  $f_{VentSide}$ is calculated in the separated function as below:

```
def f_Vent_roof_side(): #* equation 10            (2.8)
    first_part = ((U_roof * U_side * A_side * A_roof) **
    2.0)/(((U_roof**2.0) * (A_roof**2)) + ((U_side**2.0) *
    (A_side**2)))
    second_part = (2.0 * g * h_side_roof * (T_air-T_out))/T_average_air
    third_part = (((U_roof * A_roof) + (U_side * A_side))/2.0) ** 2.0
```

```
    return (C_d * (((first_part * second_part) + (third_part * C_w *
    (v_wind ** 2.0))) ** (1.0/2.0)))/A_flr
```

```
def n_ins_scr(): #*equation 11              (2.9)
    return dimensionless_ins_scr * (2.0 - dimensionless_ins_scr)
#?print(n_ins_scr())
```

```
def f_leak_age(): #*equation 12              (2.10)
    return 0.25 * c_leak_age if v_wind < 0.25 else v_wind * c_leak_age
#?print(f_leak_age())
```

```
def f_Vent_side(): #*equation 13              (2.11)
    if n_side >= n_side_thr :
        return (n_ins_scr() * f_2time_derivative_Vent_side() + 0.5 *
        f_leak_age())
    else :
        return (n_ins_scr() * (U_Th_Scr *
        f_2time_derivative_Vent_side() + (1.0 - U_Th_Scr) *
        f_Vent_roof_side()) + 0.5 * f_leak_age())
#?print(f_Vent_side())
```

The value of $f_{VentForced}$ is calculated as below:

```
def f_Vent_forced(): #*equation 14              (2.12)
    return (n_ins_scr() * U_Vent_forced * the_wind_speed)/A_flr
#?print(f_Vent_forced())
```

The value of two variable $f_{VentSide}$ and $f_{VentForced}$ are calculated so we can get calculate MC_{AirOut} through the formula:

```
def MC_air_out(): #* equation 9                (2.7)
    return ((f_Vent_side() + f_Vent_forced()) * (Co2_air - Co2_out))
#?print(MC_air_out())
```

- $MC_{TopOut} = f_{VentRoof}(CO_{2Top} - CO_{2Out})$
  Calculate the $f_{VentRoof}$:

```
def f_2time_derivative_Vent_roof(): #* equation 17        (2.15)
    first = (C_d * U_roof * A_roof) / (2.0 * A_flr)
    second = (g * h_Vent * (T_air - T_out)) / (2.0 * T_average_air)
    return first * ((second + C_w * (v_wind ** 2.0)) ** (1.0/2.0))
#?print(f_2time_derivative_Vent_roof())

def f_Vent_roof(): #* equation 16      (2.14)
    if n_roof >= n_roof_thr:
```

```
        return (n_ins_scr() * f_2time_derivative_Vent_roof() + 0.5 *
        f_leak_age())
    else:
        return (n_ins_scr() * (U_Th_Scr *
        f_2time_derivative_Vent_roof() + (1 - U_Th_Scr) *
        f_Vent_roof_side() * n_side) + 0.5 * f_leak_age())
#?print(f_Vent_roof())
```

MC_{TopOut} is calculated through the $f_{VentRoof}$

```
def MC_top_out(): #*equation 15                    (2.13)
    return (f_Vent_roof() * (Co2_top - Co2_out))
```

- $MC_{AirCan} = M_{CH_2O} h_{C_{Buf}} (P - R)$ All the variables that necessary to calculate P and R are determined as:

```
def J_Pot(): #* equation 9.15                (2.35)
    first_power =  (E_j * (T_can_k - T_25_k)) / (R * T_25_k * T_can_k)
    second_power = (S * T_25_k - H) / (R * T_25_k)
    third_power = (S * T_can_k - H) / (R * T_can_k)
    return (L_a_i * J_max_25_leaf * math.exp(first_power) * (1 +
    math.exp(second_power)))/(1 + math.exp(third_power))
#?print(J_Pot())

conversion_factor = 0.385
degree_of_curvature = 0.7
def J(): #* equation 9.14                   (2.34)
    sqrt_side = ((J_Pot() + (conversion_factor * PAR_can())) ** 2.0) -
    4.0 * degree_of_curvature * J_Pot() * PAR_can() * conversion_factor
    return (J_Pot() + conversion_factor * PAR_can() -
    math.sqrt(sqrt_side))/(2.0 * degree_of_curvature)
#?print(J())

n_Co2_air_stom = 0.67
c_Co2_compensation_point = 1.7

def Co2_stom(): #* equation 9.21             (2.41)
    return n_Co2_air_stom * Co2_air
def Co2_compensation_point(): #* equation 9.22
    return c_Co2_compensation_point * T_can_k

def P():                        (2.32)
    return (J() * (Co2_stom() - Co2_compensation_point()))/(4.0 *
    (Co2_stom() + (2.0 * Co2_compensation_point())))
def R():                       (2.33)
    return (P() * Co2_compensation_point())/Co2_stom()
```

After find all the variables, we calculate MC_{AirCan}

```
def MC_air_can(): #*equation 18        (2.16)
    return M_ch2o * h_C_Buf * (P() - R())
#?print(MC_air_can())
```

In the end, we find $\dot{CO_{2Air}}$ and $\dot{CO_{2Top}}$ as dx function with parameter x are CO_{2Air} and CO_{2Top}. The rate of concentration of CO_2 is calculated by dividing the right side of formula (sum of all MC stuff) by cap_{CO_2Air} and cap_{CO_2Top} respectively.

3 Implement the program

3.1 Data

In this assignment, we consider the statistic in the greenhouse model in Texas, USA. According to the statistic in table 8.2 in *A model based greenhouse design* reference, we have:

- cap_{CO2Air} equal to the value of height of the greenhouse compartment below the thermal screen, which is 4.7(m)
- cap_{CO2Top} equal to the value of height of the greenhouse compartment above the thermal screen, which means the subtraction of mean height of greenhouse and the height below the screen equal to 5.1 - 4.7 = 0.4 (m)
- $n_{heatCO2} = 0.057$ $(mgCO_2J^{-1})$
- U_{Blow} is a random value between 0 and 1.
- $P_{Blow} = 0$ (W)
- $A_{flr} = 7.8 * 10^4$ (m^2)
- U_{extCo2} is a random value between 0 and 1.
- $\phi_{ExtCO_2} = 4.3 * 10^5 (mgs^{-1})$.
- U_pad is a random value between 0 and 1.
- $\phi pad = 0$.
- U_{ThSrc} is a random value between 0 and 1.
- $K_{ThSrc} = 0.25 * 10^{-3} (m^3 m^{-2} K^{-0.66} s^{-1})$
- $h_{elevation} = 1470$ (m)
- $S_{Insscr} = 1.0$
- $c_{leakage} = 10^{-4}$
- $C_d = 0.65$
- $C_w = 0.09$
- U_{roof} is a random value between 0 and 1.
- $A_{roof} = 14040$

- $A_{side} = 0$
- $\eta_{roof} = 1.0$
- $\eta_{roofThr} = 0.9$

Some greenhouse parameters are constant value:

- g = $9.81(ms^{-1})$
- R = 8.3145 $(Jmol{-}1K{-}1)$

Table 4.2 demonstrate the statistic of average outdoor climate

- $v_{wind} = 2.9(ms^{-1})$
- T_{out} = 23.9 (Celcius) = 297.05K
- T_{aveAir} = 20 (Celcius) = 293.15K

Some parameters are collected from the statistic in table 9.1 (p.270) in Van11

- $M_{CH_2O} = 30 * 10^{-3}(mgCH2Omol^{-1}CH2O)$
- $h_C Buf = 1$
- $h_{Vent} = 0.97$
- $U_{Ventforce}$ is a random number between 0 and 1.

From table 8.1 in in Van11s' book, we have value of some parameters:

- p_{Air0} = 1.2 (kgm^{-3}). We consider the density at sea level
- M_{Air} = 28.9 $(kgkmol^{-1})$

According to Val11's book, we have value of some parameters:

-

*All the data collected in Van11's book (1991) are precise in times from day 157th to 162th. So the initial value of CO_2 Concentration is assumed based on the figure below from Van11 (page 97):

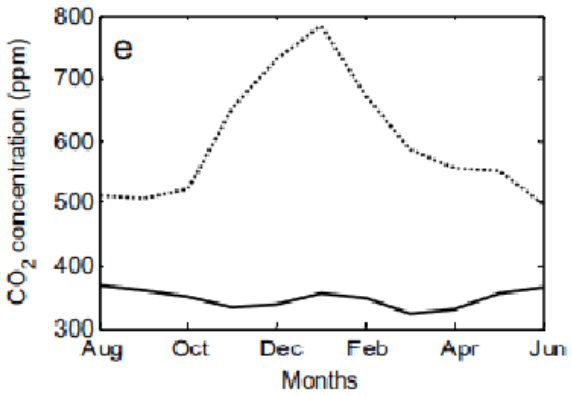


Figure 3: Concentration of CO_2

The picture above witness the CO2 concentration during day period and the in the low-tech greenhouse in Spain (solid line) and in the high-tech greenhouse in Texas, USA (dotted line) from August 1st to July 1st. Since we just considered the statistic in Texas during the time between 157th and 162th, means that its is between Apr and Jun on the graph.

So we assume the initial value of CO_{2Air} and CO_{2Top} below:

- CO_{2Air} is 550 (ppm) = 990.005(mgm^{-3})
- CO_{2Top} is 545 (ppm) = 981.005(mgm^{-3})

3.2 Result

- $MC_{BlowAir} = \frac{\eta_{HeatCO_2} U_{Blow} P_{Blow}}{A_{Flr}}$

$n_{HeatCO2}$	U_{Blow}	P_{Blow}	A_{flr}
0.057	0.52	0.0	7.8*10^4

$\rightarrow MC_{BlowAir} = 0.0$

- $MC_{ExtAir} = \frac{U_{ExtCO_2} \phi_{ExtCO_2}}{A_{Flr}}$

U_{ExtCO2}	ϕ_{ExtCO2}	A_{flr}
0.39	4.3*10^5	7.8*10^4

$\rightarrow MC_{ExtAir} = 2.15$

- $MC_{PadAir} = f_{Pad}(CO_{2Out} - CO_{2Air}) = \frac{U_{Pad}\phi_{Pad}}{A_{Flr}}(CO_{2Out} - CO_{2Air}) = 0$
The MC_{PadAir} equal to zero since greenhouse model in Texas, which do not have the cooling pad system.

- $MC_{AirTop} = f_{ThScr}(CO_{2Air} - CO_{2Top})$

$f_{ThScr} = U_{ThScr} K_{ThScr} |T_{Air} - T_{Top}|^{\frac{2}{3}} + (1 - U_{ThScr})[\frac{g(1-U_{ThScr})}{2\rho_{MeanAir}} |\rho_{Air} - \rho_{Top}|]^{\frac{1}{2}}$
$\rho_{Air} = \rho_{Air0} exp(\frac{gM_{Air}h_{Elevation}}{293.15R})$

p_{Air}	**g**	M_{Air}	$h_{elevation}$	**R**
1.2	9.81	28.96	1470	$8.3145 * 10^3$

$\rho_{Top} = \rho_{Air0} exp(\frac{gM_{Air}(h_{Elevation}+h_{Air})}{293.15R})$

p_{Air}	**g**	M_{Air}	$h_{elevation} + h_{Air}$	**R**
1.2	9.81	28.96	1470 + 4.7	$8.3145 * 10^3$

$\rho_{Air} = 1.424$
$\rho_{Top} = 1.425$

$\rho_{MeanAir} = \frac{\rho_{Air}+\rho_{Top}}{2} = 1.4245$

U_{ThScr}	K_{ThScr}	T_{Air}	T_{Top}	ρ_{Air}	ρ_{Top}	$\rho_{MeanAir}$
0.4	0.25*10^-3	291.15	295.15	1.424	1.425	1.4245

$f_{ThScr} = 0.024343607813780466$

f_{ThScr}	$CO2_{Air}$	$CO2_{Top}$
0.0243443	990.005	981.005

$\rightarrow MC_{AirTop} = 0.2190924703240242$

- $MC_{AirOut} = (f_{VentSide} + f_{VentForced})(CO_{2Air}$ - $CO_{2Out})$

$$\eta_{InsScr} = \zeta_{InsScr}(2 - \zeta_{InsScr})$$

We have $\zeta_{InsScr} = 1.0$. Assign to the equation, we get $\eta_{InsScr} = 1$

$$f_{leakage} = \begin{cases} 0.25 \cdot c_{leakage} & \text{if } v_{Wind} < 0.25 \\ v_{Wind} \cdot c_{leakage} & \text{if } v_{Wind} \geq 0.25 \end{cases}$$

Arcording to Van11 reference, we have $v_{wind} = 2.9$ and the value of $c_{leakage}$ equal to 10^{-4}. So the equation to calculate $f_{leakage} = v_{wind}.c_{leakage} = 0.00029$. Since we can not get the value of η_{Side_Thr}, we use the η_{Roof_Thr} instead.

$$f_{VentSide} = \begin{cases} \eta_{InsScr} f''_{VentSide} + 0.5 f_{leakage} & \text{if } \eta_{Roof} \geq \eta_{Roof\_Thr} \\ \eta_{InsScr}[U_{ThScr} f''_{VentSide} + \\ (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side}] + 0.5 f_{leakage} & \text{if } \eta_{Roof} < \eta_{Roof\_Thr} \end{cases}$$

The greenhouse model in Texas is the venlo types and is not equipped with side ventilation and forced ventilation, so the value of η_{roof} is always equal to 1, compare to $\eta_{roofThr} = 0.9$, so the value of

$$f_{VentSide} = \eta_{InsScr} f''_{VentSide} + 0.5 f_{leakage}$$

$$f''_{VentSide} = \frac{C_d U_{Side} A_{Side} v_{wind}}{2 A_{Flr}} \sqrt{(C_w)} \; when \; A_{Roof} = 0.$$

Due to the greenhouse model in Texas, which do not equipped side ventilation and forced ventilation so the value of $A_{Side} = 0$ in this assignment and the result of $f''_{VentSide}$ equal to 0. The value of $f_{VentSide}$ is assigned to $0.5 f_{leakage}$, where $f_{leakage} = 0.00029$; so we get $f_{VentSide} = 0.000145$

$$f_{Ventroof} = \eta_{InsScr} f''_{VentRoof} + 0.5 f_{leakage}$$

Similarly, $\phi_{VentForced} = 0$. We get:

$$f_{VentForced} = \frac{\eta_{InsScr} U_{VentForced} \phi_{VentForced}}{A_{Flr}} = 0$$

$f_{VentSide}$	$f_{Ventforced}$	CO_{2Air}	CO_{2Out}
0.000145	0	990.005	668

$\rightarrow MC_{AirOut} = 0.046690725$

- $MC_{TopOut} = f_{VentRoof}(CO_{2Top} - CO_{2Out})$ where $f_{VentRoof}$ is described as below:

$$f_{VentRoof} = \begin{cases} \eta_{InsScr} f''_{VentRoof} + 0.5 f_{leakage} & \text{if } \eta_{Roof} \geq \eta_{Roof\_Thr} \\ \eta_{InsScr}[U_{ThScr} f''_{VentSide} + \\ (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side}] + 0.5 f_{leakage} & \text{if } \eta_{Roof} < \eta_{Roof\_Thr} \end{cases}$$

Similar to $f_{VentSide}$, $f_{VentRoof} = \eta_{InsScr} f''_{VentRoof} + 0.5 f_{leakage}$

$$f''_{VentRoof} = \frac{C_d U_{Roof} A_{Side}}{2 A_{Flr}} \left[\frac{g h_{Vent} (T_{Air} - T_{Out})}{2 T_{Air}^{Mean}} + C_w v_{Wind}^2 \right]^{\frac{1}{2}}$$

C_d	0.65
U_{Roof}	0.5
A_{Roof}	14040
A_{flr}	$7.8 * 10^4$
h_{Vent}	0.97
T_{Air}	291.15
T_{Out}	297.05
T_{Air}^{Mean}	293.15
c_w	0.09
v_{wind}	2.9

$\rightarrow f''_{VentRoof} = 0.023783370636791715$

$f''_{VentRoof}$	$f_{leakage}$	CO_{2Top}	CO_{2Out}
0.023783370636791715	0.000145	981.005	668

$\rightarrow MC_{TopOut} = 7.489699651168991$

- $MC_{AirCan} = M_{CH_2O} h_{C_{Buf}} (P - R)$

$$P = \frac{J.(CO2_{Stom} - \Gamma)}{4.(CO2_{Stom} + 2\Gamma)}$$

$$J = \frac{J^{POT} + \alpha PAR_{Can} - \sqrt{(J^{POT} + \alpha PAR_{Can})^2 - 4\Theta . J^{POT} . \alpha PAR_{Can}}}{2\Theta}$$

$$J^{POT} = J^{MAX}_{25,Can} . e^{E_j \cdot \frac{T_{Can,K} - T_{25,K}}{R.T_{Can,K}.T_{25,K}}} . \frac{1 + e^{\frac{S.T_{25,K} - H}{R.T_{25,K}}}}{1 + e^{\frac{S.T_{Can,K} - H}{R.T_{Can,K}}}}$$

$$J^{MAX}_{25,Can} = J^{MAX}_{25,Leaf} \cdot LAI$$

From the statistic, we have LAI = 2. Beside, $J^{MAX}_{25,Leaf} = 210$. The value of $J^{MAX}_{25,Can} = 420$. Through the formula, we have the value of $J^{POT} = 329.47559370461505$

J_{Pot}	PAR_{Can}	α	θ
329.4755937	416.73118230	0.385	0.7

From this formula. We get the result of J = 133.27967070423793
The formula of CO_{2Stom}:

$$CO_{2Stom} = \eta_{CO_{2Air\_Stom}} \cdot CO_{2Air}$$

$\eta_{CO_{2Air\_Stom}}$	$\cdot CO_{2Air}$
0.67	990.005

$\rightarrow CO_{2Stom} = 663.30335$

- Equation of gamma: Since the model is not measured at 20 Celcius, so:

$$\Gamma = c_\Gamma T_{Can} = 497.845$$

Equation of P:

$$P = \frac{J \cdot (CO_{2Stom} - \Gamma)}{4 \cdot (CO_{2Stom} + 2\Gamma)}$$

J	CO_{2Stom}	Γ
3.27967	663.30335	497.845

$\rightarrow P = 3.3231348400622798$ Equation of photorespiration:

$$R = P \cdot \frac{\Gamma}{CO_{2Stom}}$$

P	CO_{2Stom}	Γ
3.3231348	663.30335	497.845

$\rightarrow R = 2.4941922341426523$

M_{CH_2O}	h_{cBuf}	P	R
$30 * 10^{-3}$	1	3.3231348	2.49419

$\rightarrow MC_{AirCan} = 0.024868278177588823$

- The function **dx** is defined as function dx_CO2() in the program, it is equivalent to equation:

$$\begin{cases} \dot{CO_{2Air}} = (MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} \\ \qquad\qquad -MC_{AirCan} - MC_{AirTop} - MC_{AirOut})/cap_{CO_2Air} \\ \dot{CO_{2Top}} = (MC_{AirTop} - MC_{TopOut})/cap_{CO_2Top} \end{cases}$$

. The result of the program:

```
[Running] python -u "/Users/trananhdung2720/Desktop/TestMM/1/for_the_system_1_and2.py"
('MV_blow_air', 0.0)
('MC_ext_air', 2.15)
('MC_pad_air', -0.0)
('p_air', 1.4242731260637598)
('p_top', 1.4250535878256156)
('f_Th_Scr', 0.024343607813777035)
('MC_air_top', 0.6572774109719799)
('f_Vent_roof_side', 0.0254475)
('n_ins_scr', 1.0)
('f_leak_age', 0.00029)
('f_2time_derivative_Vent_side', 0.0)
('f_Vent_side', 0.0154135)
('f_Vent_forced', 0.0)
('MC_air_out', 4.2696165675)
('f_2time_derivative_Vent_roof', 0.024895854045518605)
('f_Vent_roof', 0.010103341618207442)
('MC_top_out', 2.5258859212599516)
('PAR_Gh', 529.4811599999999)
('PAR_Gh_can', 370.9888235361396)
('PAR_flr_can', 45.74235877014021)
('PAR_can', 416.7311823062798)
('J_Pot', 329.47559370461505)
('J', 133.27967070423793)
('Co2_stom', 633.15335)
('Co2_compensation_point', 497.845)
('P()', 2.767892371530659)
('R_()', 2.1763785624836083)
('MC_air_can', 0.01774541427141152)
```

- This function will return two results: $\dot{CO_{2Air}} = 0.3956060694677419$ and $\dot{CO_{2Top}} = -18.176517952112416$.
  This conclude that the CO2 concentration in the greenhouse air is slightly increase and CO2 concentration in the greenhouse air above the thermal screen is decrease at the summer time from day 157th to 162th.

4 Calculating CO_2 concentration with Euler and Runge-Kutta algorithms

4.1 Euler and Runge-Kutta of order 4 algorithms with python language

- Euler algorithm:

```
@dataclass
class Euler:
    x_0: float = 0.0
    y_0: float = 0.0
    t: float = 0.0
    h: float = 0.0

    def __init__(self, x0, y0, t0, h):
        self.x_0 = x0
        self.y_0 = y0
        self.t = t0
        self.h = h

    def calculateNewValue(self, x, y):
        x_new = x + self.dx_dt(x, y, self.t) * h
        y_new = y + self.dy_dt(x, y, self.t) * h

        return x_new, y_new

    def calculateNext(self, x, y):
        self.t = self.t + h

        yield self.calculateNewValue(x, y)

    def dx_dt(self, x, y, t):
        rhs = (
            MC_blow_air() + MC_ext_air()
            + ((U_pad * phi_pad)/A_flr) * (668.0 - x) #! MC_pad_out
            - f_Th_Scr() * (x - y) #! MC_air_top
            - (f_Vent_forced() + f_Vent_side()) #!MC_air_out
            - ((M_ch2o * J())/4.0) * ((n_Co2_air_stom * x -
            Co2_compensation_point()) / (n_Co2_air_stom * x + (2.0 *
            Co2_compensation_point()))) * (1 - (Co2_compensation_point() /
            (n_Co2_air_stom * x)))#!MC_air_can
        )
        return rhs / 4.7

    def dy_dt(self, x, y, t):
        rhs = (
```

```
            f_Th_Scr() * (x - y) #!MC_air_top
            - f_Vent_roof() * (y - 668) #!MC_top_out
        )
        return rhs / 0.4


print("RUNNING EULER")
x = 990.005 #550
y = 981.005 #545
t = 157.0
h = 0.25
a = x/(0.0409*44.01)
b = y/(0.0409*44.01)
euler = Euler(x, y, t, h)  # x_0, y_0, t_0, h

list_x = [x]
list_y = [y]
list_t = [t]
list_a = [a]
list_b = [b]

for n in range(20):
    t = t + h
    gen_new_val = euler.calculateNext(x, y)
    x, y = next(gen_new_val)
    a = x/(0.0409*44.01)
    b = y/(0.0409*44.01)
    if n < 5:
        print(f"step: {n + 1}")
        print(f"new x: {a}\nnew y: {b}\n\n")
    if n == 0 or n == 4 or n == 9 or n == 14 or n == 19:
        list_x.append(x)
        list_y.append(y)
        list_a.append(a)
        list_b.append(b)
        list_t.append(t)

plt.plot(list_t, list_a, "-b", label = "CO2_air")
plt.plot(list_t, list_b, "-r", label = "CO2_top")
plt.title("The CO2 concentration below and above the thermal screen from
DOY 157-162")
plt.xlabel("time(days)")
plt.ylabel("CO2_concentration")
plt.legend()
#plt.subplot(1,1,1)
#plt.title("The CO2 concentration above the thermal screen from DOY
157-162")
#plt.xlabel("time(days)")
```

```
#plt.ylabel("CO2_top")

plt.show()
```

- Runge-Kutta of order 4 algorithm:

```
# in the program, the function is named 'runge_kutta4th'
@dataclass
class RK4:
    x_0: float = 0.0
    y_0: float = 0.0
    t: float = 0.0
    h: float = 0.0

    def __init__(self, x0, y0, t0, h):
        self.x_0 = x0
        self.y_0 = y0
        self.t = t0
        self.h = h

    def calculateNewValue(self, x, y):
        k0_x, k0_y = self.calculateFirstSlop(x, y)
        k1_x, k1_y = self.calculateMiddleSlop(x, y, k0_x, k0_y)
        k2_x, k2_y = self.calculateMiddleSlop(x, y, k1_x, k1_y)
        k3_x, k3_y = self.calculateLastSlop(x, y, k2_x, k2_y)

        x_new = x + (k0_x + 2 * k1_x + 2 * k2_x + k3_x) / 6.0
        y_new = y + (k0_y + 2 * k1_y + 2 * k2_y + k3_y) / 6.0

        return x_new, y_new

    def calculateFirstSlop(self, x, y):
        k0_x = h * self.dx_dt(x, y, self.t)
        k0_y = h * self.dy_dt(x, y, self.t)

        return k0_x, k0_y

    def calculateMiddleSlop(self, x, y, k_prev_x, k_prev_y):
        k_next_x = h * self.dx_dt(
            x + k_prev_x * 0.5, y + k_prev_y * 0.5, self.t + h * 0.5
        )
        k_next_y = h * self.dy_dt(
            x + k_prev_x * 0.5, y + k_prev_y * 0.5, self.t + h * 0.5
        )

        return k_next_x, k_next_y

    def calculateLastSlop(self, x, y, k_prev_x, k_prev_y):
```

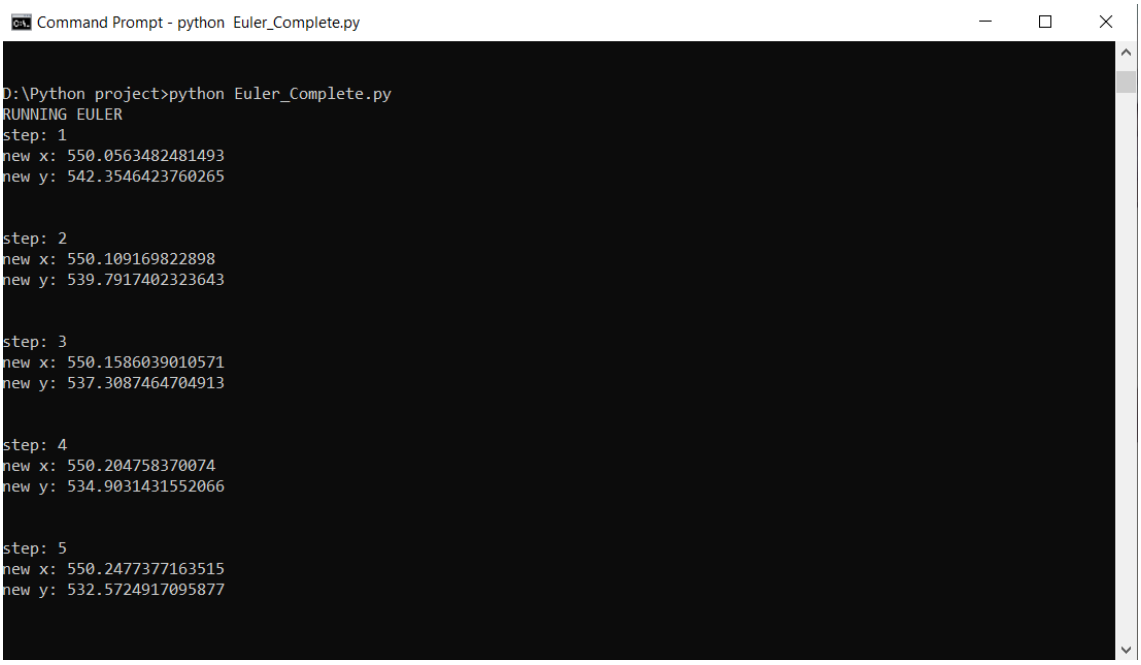
```
43          k_last_x = h * self.dx_dt(x + k_prev_x, y + k_prev_y, self.t + h)
44          k_last_y = h * self.dy_dt(x + k_prev_x, y + k_prev_y, self.t + h)
45
46          return k_last_x, k_last_y
47
48      def calculateNext(self, x, y):
49          self.t = self.t + h
50
51          yield self.calculateNewValue(x, y)
52
53      def dx_dt(self, x, y, t):
54          rhs = (
55              MC_blow_air() + MC_ext_air()
56              + ((U_pad * phi_pad)/A_flr) * (668.0 - x) #! MC_pad_out
57              - f_Th_Scr() * (x - y) #! MC_air_top
58              - (f_Vent_forced() + f_Vent_side()) #!MC_air_out
59              - ((M_ch2o * J())/4.0) * ((n_Co2_air_stom * x -
                Co2_compensation_point()) / (n_Co2_air_stom * x + (2.0 *
                Co2_compensation_point()))) * (1 - (Co2_compensation_point() /
                (n_Co2_air_stom * x)))#!MC_air_can
60          )
61          return rhs / 4.7
62
63      def dy_dt(self, x, y, t):
64          rhs = (
65              f_Th_Scr() * (x - y) #!MC_air_top
66              - f_Vent_roof() * (y - 668) #!MC_top_out
67          )
68          return rhs / 0.4
69
70
71  print("RUNNING RK4")
72  x = 990.005 #550
73  y = 981.005 #545
74  t = 157.0
75  h = 0.25
76  a = x/(0.0409*44.01)
77  b = y/(0.0409*44.01)
78  rk4 = RK4(x, y, t, h)  # x_0, y_0, t_0, h
79
80  list_x = [x]
81  list_y = [y]
82  list_t = [t]
83  list_a = [a]
84  list_b = [b]
85
86  for n in range(20):
87      t = t + h
```

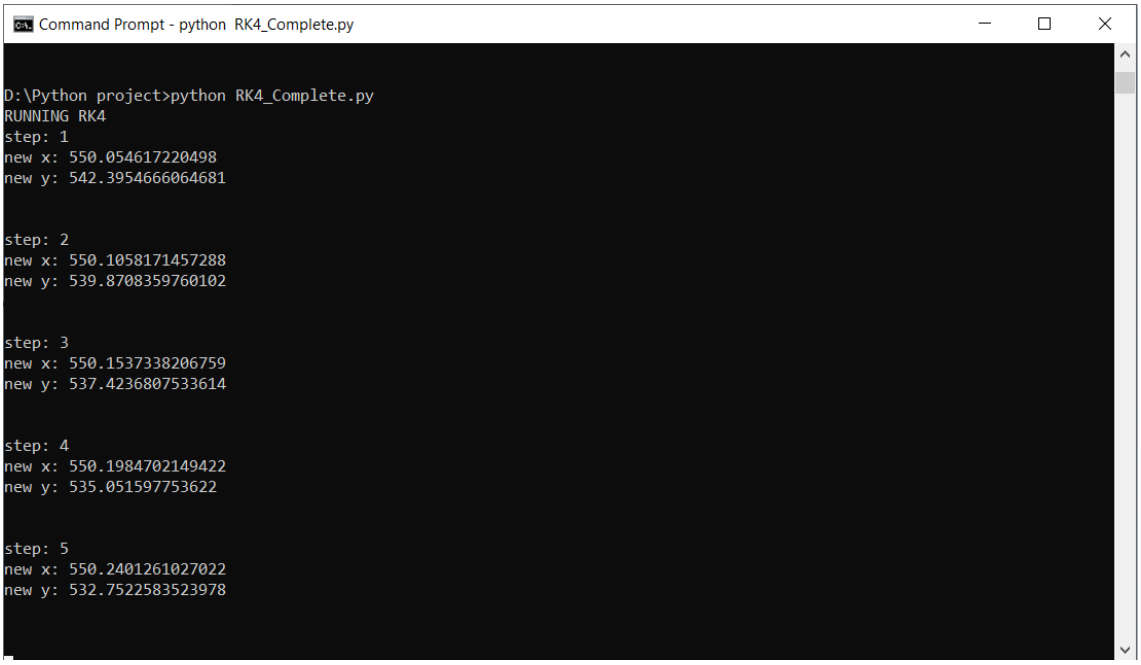
```
    gen_new_val = rk4.calculateNext(x, y)
    x, y = next(gen_new_val)
    a = x/(0.0409*44.01)
    b = y/(0.0409*44.01)
    if n < 5:
        print(f"step: {n + 1}")
        print(f"new x: {a}\nnew y: {b}\n\n")
    if n == 0 or n == 4 or n == 9 or n == 14 or n == 19:
        list_x.append(x)
        list_y.append(y)
        list_a.append(a)
        list_b.append(b)
        list_t.append(t)


#plt.subplot(1,1,1)
plt.plot(list_t, list_a, "-b", label = "CO2_air")
plt.plot(list_t, list_b, "-r", label = "CO2_top")
plt.title("The CO2 concentration below and above the thermal screen from
DOY 157-162")
plt.xlabel("time(days)")
plt.ylabel("CO2_concentration")
plt.legend()
#plt.subplot(1,1,1)
#plt.title("The CO2 concentration above the thermal screen from DOY
157-162")
#plt.xlabel("time(days)")
#plt.ylabel("CO2_top")

plt.show()
```

4.2 Comparing to actual data and comment on accuracy of the model

From initial state at which $t_0 = 157$, choosing $CO_{2Air} = 990.005\ mgm^{-3} (= 550\ ppm)$ and $CO_{2Top} = 981.005\ mgm^{-3} (= 545\ ppm)$ as initial values, we have the following approximation values of CO_{2Air}, CO_{2Top} in the next 5 minutes, 10 minutes, 20 minutes, 25 minutes, represented as the table below. The actual values of CO_{2Top} are not available for comparison.

- The result of the program: x is CO_{2Air} and y is CO_{2Top}


Command Prompt - python Euler_Complete.py
D:\Python project>python Euler_Complete.py
RUNNING EULER
step: 1
new x: 550.0563482481493
new y: 542.3546423760265
step: 2
new x: 550.109169822898
new y: 539.7917402323643
step: 3
new x: 550.1586039010571
new y: 537.3087464704913
step: 4
new x: 550.204758370074
new y: 534.9031431552066
step: 5
new x: 550.2477377163515
new y: 532.5724917095877


Command Prompt - python RK4_Complete.py
D:\Python project>python RK4_Complete.py
RUNNING RK4
step: 1
new x: 550.054617220498
new y: 542.3954666064681
step: 2
new x: 550.1058171457288
new y: 539.8708359760102
step: 3
new x: 550.1537338206759
new y: 537.4236807533614
step: 4
new x: 550.1984702149422
new y: 535.051597753622
step: 5
new x: 550.2401261027022
new y: 532.7522583523978

Time	CO_{2Air}(Euler)	CO_{2Air}(RK4)	**Actual data**	**Diff(Euler)**	**Diff(RK4)**
5	550.0563	550.0546	550.0500	0.0063	0.0046
10	550.1092	550.1058	550.1000	0.0092	0.0058
15	550.1586	550.1537	550.1300	0.0286	0.0237
20	550.2048	550.1985	550.1600	0.0448	0.0385
25	550.2477	550.2401	550.2000	0.0477	0.0401

Time	CO_{2Top}(Euler)	CO_{2Top}(RK4)	**Actual data**	**Diff(euler)**	**Diff(rk4)**
5	542.3546	542.3955	x	x	x
10	539.7917	539.8708	x	x	x
15	537.3087	537.4237	x	x	x
20	534.9031	535.0516	x	x	x
25	532.5725	532.7523	x	x	x

- The graph below is used to compare to actual data from Van11, the unit of CO_2 is changed to ppm to be similar to the references

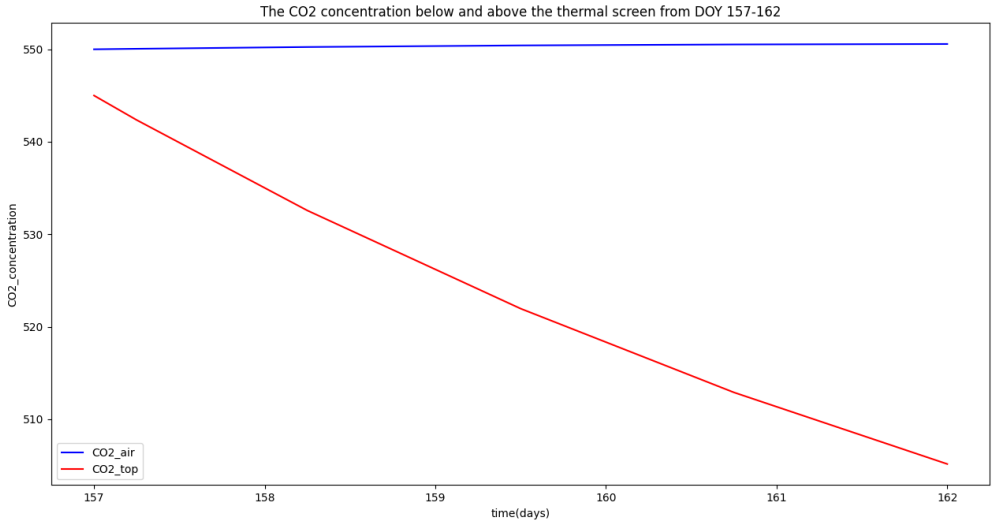


Figure 4: Result of Euler method

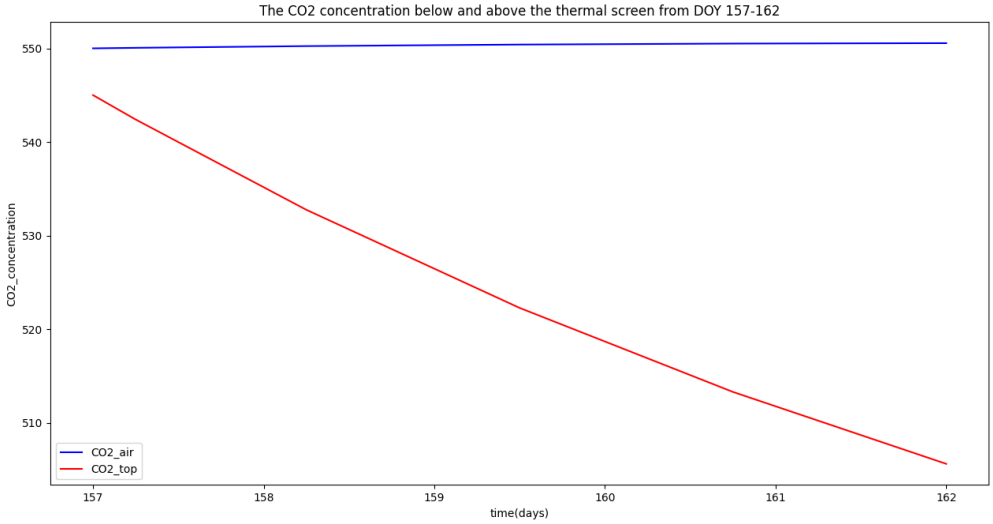


Figure 5: Result of RK4 method

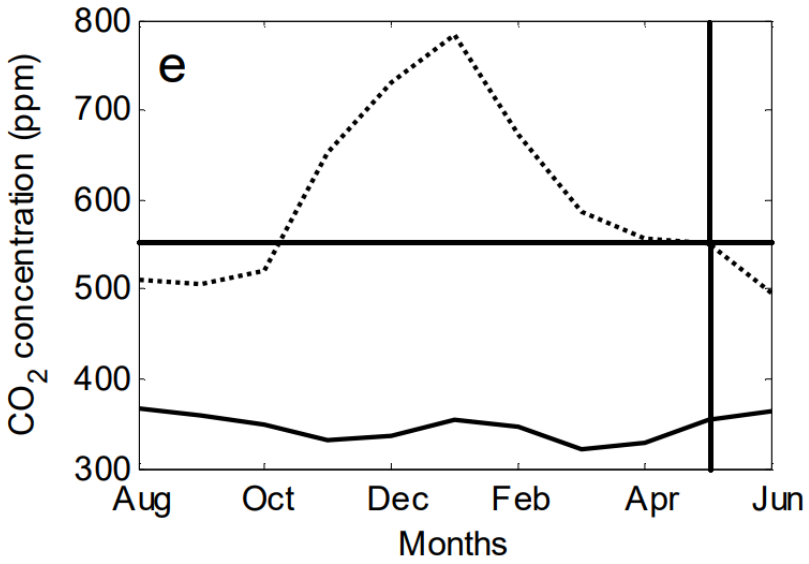


Figure 6: Actual Data from Van11 page 97

Conclusion, during the summer time between day 157th to 162th, the CO2 concentration in actual data was slightly changed and result on the program is nearly accurate.

5 Vapor pressure in greenhouse model

5.1 Theory

Beside CO_2 concentration, the vapour pressure is also a factor that affect to crop yield that we have to control and it can be predicted through the model. The vapour pressure of the greenhouse air VP_{Air} is described by:

$$\begin{aligned} cap_{VP_{Air}} \dot{VP_{Air}} = MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BlowAir} \\ - MV_{AirThScr} - MV_{AirTop} - MV_{AirOut} \\ - MV_{AirOut\_Pad} - MV_{AirMech} \quad [kg\ m^{-2}\ s^{-1}] \end{aligned}$$

- $cap_{VP_{Air}}$:the capacity of the air to store water vapor.
- MP_{CanAir}: the canopy
- MV_{PadAir}: The outlet air of the pad
- MV_{FogAir}: the fogging system.
- $MV_{BlowAir}$: the direct air heater.
- $MV_{AirThScr}$: the Thermal screen.
- MV_{AirTop}: the top compartment air
- MV_{AirOut}: the outdoor air.
- MV_{AirOut_Pad}: the air exchange caused by the pad and fan system.
- $MV_{AirMech}$: the mechanical cooling system

The vapour pressure of the air in the top compartment VP_{Top} is described by:

$$cap_{VP_{Top}} \dot{VP_{Top}} = MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut} \quad [kg\ \ m^{-2}\ \ s^{-1}]$$

- $cap_{VP_{Top}}$: the capacity of the top compartment to store water vapour.
- $MV_{TopCov,in}$: the vapour exchange between the top compartment and the internal cover.
- MV_{TopOut}: the vapour exchange between the top compartment and the outside air.

$$cap_{VPair} = \frac{M_{water}.h_{air}}{R(T_{Air+273.15})}[kgm^3J^{-1}]$$

$$cap_{VPtop} = \frac{M_{water}.h_{top}}{R(T_{Top+273.15})}[kgm^3J^{-1}]$$

- cap_{VPair}: water vapour capacity of the air compartment.
- M_{water}: the molar mass of water
- h_{air}: the height from the floor to the thermal screen.

- h_{top}: the height from the floor to the thermal screen.
- R: the molar gas constant.

The vapor pressure of the air in the top compartment VP_{Top} is described by:

$$cap_{VP_{Top}} \dot{VP_{Top}} = MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut} \qquad [kg \quad m^{-2} \quad s^{-1}]$$

- $cap_{VP_{Top}}$: the capacity of the top compartment to store water vapour.
- $MV_{TopCov,in}$: the vapor exchange between the top compartment and the internal cover layer.
- MV_{TopOut}: the vapor exchange between the top compartment and the outside air

The canopy transpiration is described by:

$$MV_{CanAir} = VEC_{CanAir}(VP_{Can} - VP_{Air})$$

- VEC_{CanAir}: the vapor exchange coefficient between the canopy and air
- VP_{Can}: the saturated vapour pressure at canopy temperature.

Based on Stanghellini (1987), the value of the vapor transfer coefficient of the canopy transpiration can be calculated by the equation below:

$$VEC_{CanAir} = \frac{2\rho_{Air} c_{p,Air} LAI}{\Delta H \gamma (r_b + r_s)}$$

- ρ_{Air}: the density of the greenhouse air
- $c_{p,Air}$: the specific heat capacity of the greenhouse air.
- LAI: the leaf area index.
- ΔH: the latent heat of evaporation of water.
- γ: the psychometric constant
- r_b: is the boundary layer resistance of the canopy for vapor transport
- r_s: the stomatal resistance of the canopy for vapor transport.

Arcording to [Sta87] The wind speed in the greenhouse and the temperature difference between the canopy and surrounding air are together affect the value of the boundary layer resistance for vapor transport. The stomatal resistance of the canopy is described by the formula below:

$$r_s = r_{s,\min} \cdot rf(R_{Can}) \cdot rf(CO_{2Air\_ppm}) \cdot rf(VP_{Can} - VP_{Air})$$

- $r_{s,\min}$: the minimum canopy resistance
- rf: the resistance factor for high radiation levels
- high CO_2: levels and large vapor pressure differences.

The resistance factors are described according to Stangellini (1987):

$$rf(R_{Can}) = \frac{R_{Can} + c_{evap1}}{R_{Can} + c_{evap2}}$$
$$rf(CO_{2Air}) = 1 + c_{evap3}(\eta_{mg\_ppm} CO_{2Air} - 200)^2$$
$$rf(VP_{Can} - VP_{Air}) = 1 + c_{evap4}(VP_{Can} - VP_{Air})^2$$

- R_{Can}: the global radiation above the canopy
- c_{evap1} ($W\ m^{-2}$), c_{evap2} ($W\ m^{-2}$), c_{evap3} (ppm^{-2}), c_{evap4} (Pa^{-2}) are empirically determined parameters
- η_{mg_ppm} ($ppm\ mg^{-1}\ m^3$) is the conversion factor from $mg\ m^{-3}\ CO_2$ to ppm

  Stanghellini limited the resistance factor for high CO_2 levels to 1.5 and the resistance factor for large vapor pressure differences to 5.8
- c_{evap3}: the transpiration variables
- c_{evap4} for day time and night time.

The values of the transpiration parameters c_{evap3} and c_{evap4} differed between the night period and day period which means that the accompanying equations are not differentiable at sunrise and sunset. Therefore the parameters c_{evap3} and c_{evap4} were smoothed using the differentiable switch function:

$$S_{r_s} = \frac{1}{1 + \exp\left(s_{r_s}(R_{Can} - R_{Can\_SP})\right)}$$

- S_{r_s}: the value of the differentiable switch

The vapor flux from the pad and fan to the greenhouse air is described by:

$$MV_{PadAir} = \rho_{Air} f_{Pad}(\eta_{Pad}(x_{Pad} - x_{Out}) + x_{Out})$$

- f_{Pad}: the ventilation flux due to the pad and fan system
- η_{Pad}: the efficiency of the pad and fan system
- x_{Pad}: the water vapor content of the pad
- x_{Out}: the water vapor content of the outdoor air.

The latent heat flux from the greenhouse air depends on the vapor flux from the fogging system to the greenhouse air which is described by:

$$MV_{FogAir} = \frac{U_{Fog}\phi_{Fog}}{A_{Flr}}$$

- U_{Fog}: the control valve of the fogging system
- ϕ_{Fog}: the capacity of the fogging system.

The vapor flux from the heat blower to the greenhouse air is proportional to the heat flux:

$$MV_{BlowAir} = \eta_{HeatVap} H_{BlowAir}$$

- $\eta_{HeatVap}$: the amount of vapor which is released when 1 Joule of sensible energy is produced by the direct air heater..

The vapor exchange coefficient between the air and an object is linearly related to the convective heat exchange coefficient between the air and the object. Therefore, the vapor flux from the air to an object by condensation is described by:

$$MV_{12} = \begin{cases} 0 & \text{if } VP_1 < VP_2 \\ 6.4 \cdot 10^{-9} HEC_{12}(VP_1 - VP_2) & \text{if } VP_1 > VP_2 \end{cases}$$

- MV_{12}: the vapor flux from air of location 1 to object 2.
- $6.4 \cdot 10^{-9}$ is the conversion factor relating the heat exchange coefficient ($W\ m^{-2}\ K^{-1}$) to the vapor exchange coefficient ($kg\ m^{-2}\ s^{-1}\ Pa^{-1}$), HEC_{12} ($W\ m^{-2}\ K^{-1}$) is the heat exchange coefficient between the air of location 1 to object 2 and VP_1 (Pa) is the vapor pressure of the air of location 1 and VP_2 is the saturated vapor pressure of object 2 at its temperature.

The vapor flux from the greenhouse air compartment to the thermal screen and the vapor flux from the top compartment to the interval cover layer are described:

$$MV_{AirThScr} = \begin{cases} 0 & \text{if } VP_{Air} < VP_{ThScr} \\ 6.4 \cdot 10^{-9} HEC_{AirThScr}(VP_{Air} - VP_{ThScr}) & \text{if } VP_{Air} > VP_{ThScr} \end{cases}$$

$$MV_{TopCov,in} = \begin{cases} 0 & \text{if } VP_{Top} < VP_{Cov,in} \\ 6.4 \cdot 10^{-9} HEC_{TopCov,in}(VP_{Top} - VP_{Cov,in}) & \text{if } VP_{Top} > VP_{Cov,in} \end{cases}$$

whereby their associated heat change coefficients are:

$$HEC_{AirThScr} = 1.7 U_{ThScr} |T_{Air} - T_{ThScr}|^{0.33}$$

$$HEC_{TopCov,in} = c_{HECin} (T_{Top} - T_{Cov,in})^{0.33} \frac{A_{Cov}}{A_{Flr}}$$

- $U_{ThScr} \in [0, 1]$ representing the percentage of places that are covered by the thermal screen.
- T_X (°C) is the temperature at location.
- X, c_{HECin}: the convective heat exchange parameter between cover and outdoor air depending on the greenhouse shape.
- A_{Cov}: the surface of the cover including the side-walls.
- A_{Flr}: the surface of the greenhouse floor.

The vapor flux from the greenhouse air to the outside air when using the pad and fan system is described by:

$$MV_{AirOut\_Pad} = f_{Pad} \frac{M_{Water}}{R} \left(\frac{VP_{Air}}{T_{Air} + 273.15} \right)$$

The vapor flux from the greenhouse air to the surface of mechanical cooling system.

$$MV_{AirMech} = \begin{cases} 0 & \text{if } VP_{Air} < VP_{Mech} \\ 6.4 \cdot 10^{-9} HEC_{AirMech}(VP_{Air} - VP_{Mech}) & \text{if } VP_{Air} > VP_{Mech} \end{cases}$$

with $HEC_{AirMech}$ is the heat exchange coefficient between the greenhouse air and the surface of the mechanical cooling unit:

$$HEC_{AirMech} = \frac{U_{MechCool} COP_{MechCool} P_{MechCool} / A_{Flr}}{T_{Air} - T_{MechCool} + 6.4 \cdot 10^{-9} \Delta H (VP_{Air} - VP_{MechCool})}$$

- $U_{MechCool}$: the control valve of the mechanical cooling mechanism.
- $COP_{MechCool}$: the coefficient of performance of the mechanical cooling system.
- $P_{MechCool}$: the electrical capacity of the mechanical cooling system.
- $T_{MechCool}$ (℃) is the temperature of the cooling surface.
- $VP_{MechCool}$ (Pa) is the saturated vapor pressure of the mechanical cooling mechanism.

5.2 Implement the program

- $cap_{VPair} = \frac{M_{water}.h_{air}}{R(T_{Air}+273.15)}$ $[kgm^3 J^{-1}]$

```
def cap_Vp_air(): #* equation 8.25
    return (M_water * h_air)/(R * (T_air + 273.15))
print("cap_Vp_air",cap_Vp_air())
```

M_{water}	h_{air}	R	T_{Air}
18	4.7	$8.314 * 10^3$	291.15

$\rightarrow cap_{VPAir} = 3.494970774 * 10^{-5}$

- $cap_{VPtop} = \frac{M_{water}.h_{top}}{R(T_{Top}+273.15)}$ $[kgm^3 J^{-1}]$

```
def cap_Vp_top(): #* equation 8.25
    return (M_water * h_top)/(R * (T_air + 273.15))
print("cap_Vp_top",cap_Vp_top())
```

M_{water}	h_{top}	R	T_{Top}
18	0.4	$8.314 * 10^3$	295.15

$\rightarrow cap_{VpTop} = 2.934132279 * 10^{-6}$

- $S_{r_s} = \frac{1}{1+\exp\left(s_{r_s}(R_{Can} - R_{Can\_SP})\right)}$

```
def S_r_s(): #* equation 8.51 (maybe this equation is redundant)
    return 1.0/(1.0 + math.exp(s_r_s * (R_can - R_can_sp)))
print("S_r_s",S_r_s())
```

The value of R_{can} is collected from the research of Michael E.Webber: "The Energy-Water Nexus: Spatially-Resolved Analysis of the Potential for Desalinating Brackish Groundwater by Use of Solar Energy" June 2015.

s_{r_s}	R_{can}	$R_{can_{sp}}$
-1.0	0.5439815	5.0

- $c_{evap3} = c_{evap3}^{night}(1 - S_{r_s}) + c_{evap3}^{night} S_{r_s}$
- $c_{evap4} = c_{evap4}^{night}(1 - S_{r_s}) + c_{evap4}^{night} S_{r_s}$

```
def c_evap_3(): #* equation 8.52
    return c_evap_night_3 #! explain in detail in report
print("c_evap_3",c_evap_3())
def c_evap_4(): #* equation 8.52
    return c_evap_day_4 #! read van11 on page 218
print("c_evap_4",c_evap_4())
```

e_{evap3}^{night}	e_{evap4}^{night}
$1.1 * 10^{-11}$	$5.2 * 10^{-6}$

- $rf(R_{Can}) = \frac{R_{Can} + c_{evap1}}{R_{Can} + c_{evap2}}$
- $rf(CO_{2Air}) = 1 + c_{evap3}(\eta_{mg\_ppm} CO_{2Air} - 200)^2$
- $rf(VP_{Can} - VP_{Air}) = 1 + c_{evap4}(VP_{Can} - VP_{Air})^2$

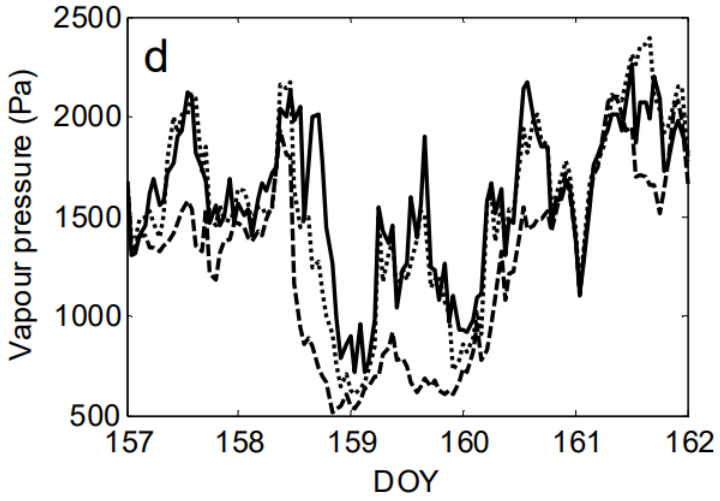
```
def rf_R_can(): #* equation 8.50
    return (R_can + c_evap_1)/(R_can + c_evap_2)
print("rf_R_can",rf_R_can())
def rf_Co2_air(): #* equation 8.50
    return 1.0 + c_evap_3() * ((n_mg_ppm * Co2_air - 200.0) ** 2.0)
print("rf_Co2_air",rf_Co2_air())
def rf_Vpcan_Vpair(): #* equation 8.50
    return 1.0 + c_evap_4() * ((Vp_can - Vp_air) ** 2.0)
print("rf_Vpcan_Vpair",rf_Vpcan_Vpair())
```

c_{evap1}	c_{evap2}
4.3	0.54

The value of $CO2_{Air}$ is the initial value of $CO2_{Air}$ which is assumed in question 3

c_{evap3}	η_{mg_ppm}	CO_{2Air}
$1.1 * 10^{-11}$	0.554	990.005

*All the data collected in Van11's book (1991) are precise in times from day 157th to 162th. So the initial value of VP_{air}, VP_{top} and the value of VP_{can} is assumed based on the figure below from Van11 (page 40):



c_{evap4}	VP_{Can}	VP_{Air}
$5.2 * 10^{-6}$	2300	1700

- $r_s = r_{s,\min} \cdot rf(R_{Can}) \cdot rf(CO_{2Air\_ppm}) \cdot rf(VP_{Can} - VP_{Air})$

```
def r_s(): #* equation 8.49
    return r_s_min * rf_R_can() * rf_Co2_air() * rf_Vpcan_Vpair()
print("r_s",r_s())
```

r_{smin}	rf_{Rcan}	rf_{CO2Air}	$rf_{VpcanVpair}$
82	2.4434182609901476	1.0000011514079858	2.872

$\rightarrow r_s = 575.4354366964179$

- $VEC_{CanAir} = \frac{2\rho_{Air}c_{p,Air}LAI}{\Delta H\gamma(r_b+r_s)}$

```
def VEC_can_air(): #* equation 8.48
    return (2.0 * p_air() * c_p_air * L_a_i)/(delta_H * gamma * (r_b +
    r_s()))
print("VEC_can_air",VEC_can_air())
```

The value of p_{air} is already calculated in question 3, so we reused it

p_{air}	c_{pair}	LAI	δH	γ	r_b	r_s
1.4242731260637596	10^3	2.0	$2.45 * 10^6$	65.8	275	575.4354366964179

$\rightarrow VEC_{CanAir} = 4.1554680236317574 * 10^{-08}$

- $MV_{CanAir} = VEC_{CanAir}(VP_{Can} - VP_{Air})$

```
def MV_can_air(): #* equation 8.47
    return (VEC_can_air() * (Vp_can - Vp_air))
print("MV_can_air",MV_can_air())
```

VEC_{canair}	VP_{can}	VP_{air}
$4.1554680236317574 * 10^{-08}$	2300	1700

$\rightarrow MV_{CanAir} = 2.4932808141790545 * 10^{-5}$

- $MV_{BlowAir} = \eta_{HeatVap}H_{BlowAir}$

```
def MV_blow_air(): #* equation 8.55
    return (n_heat_vap * U_blow * P_blow)/A_flr
print("MV_blow_air",MV_blow_air())
```

$H_{BlowAir} = \frac{U_{Blow}.P_{Blow}}{A_flr}$ Since $P_{Blow} = 0$, we get $H_{BlowAir} \rightarrow MV_{BlowAir} = 0$

- $f_{Pad} = \frac{U_{Pad}\phi_{Pad}}{A_{Flr}}$

```
def f_pad(): #* to help equation 8.58
    return (U_pad * phi_pad)/A_flr
print("f_pad",f_pad())
```

Since the greenhouse model in Texas does not have pad system, $\phi Pad = 0$, we get $f_{Pad} = 0$. So:

- $MV_{PadAir} = \rho_{Air}f_{Pad}(\eta_{Pad}(x_{Pad} - x_{Out}) + x_{Out}) = 0$

```
def MV_pad_air(): #* equation 8.58
    return p_air() * f_pad() * (n_pad * (x_pad - x_out) + x_out)
print("MV_pad_air",MV_pad_air())
```

Similarity, we get the value of MV_{AirOut_Pad} equal to 0.

- $MV_{AirOut\_Pad} = f_{Pad}\frac{M_{Water}}{R}\left(\frac{VP_{Air}}{T_{Air}+273.15}\right) = 0$

```
def MV_air_out_pad(): #* equation 8.62
    return (f_pad() * M_water * Vp_air)/(R * (T_air + 273.15))
print("MV_air_out_pad",MV_air_out_pad())
```

Since the greenhouse model in Texas does not have fogging system pad, $\phi_{Fog} = 0$, we get $f_{Fog} = 0$.

- $MV_{FogAir} = \frac{U_{Fog}\phi_{Fog}}{A_{Flr}} = 0$

```
def MV_frog_air(): #* equation 8.64
    return (U_frog * phi_frog)/A_flr
print("MV_frog_air",MV_frog_air())
```

- $MV_{AirTop} = \frac{M_{Water}}{R} f_{ThScr}\left(\frac{VP_{Air}}{T_{Air}+273.15} - \frac{VP_{Top}}{T_{Top}+273.15}\right)$

```
def MV_air_top(): #* equation 8.45 and according to van11 on page 216,
f_air_top = f_Th_Scr
    return (M_water/R) * f_Th_Scr() * (Vp_air/(T_air + 273.15) -
    Vp_top/(T_top + 273.15))
print("MV_air_top",MV_air_top())
```

The value of f_{ThScr} is already calculated in question 3, so we reused it

M_{water}	R	f_{ThScr}	VP_{Air}	T_{Air}	VP_{Top}	T_{Top}
18	8314	0.0243443	1700	291.15	1650	295.15

$\rightarrow MV_{AirTop} = 1.3098215855448786 * 10^{-5}$

- $MV_{AirOut} = \frac{M_{Water}}{R}(f_{VentSide} + f_{VentForced})\left(\frac{VP_{Air}}{T_{Air}+273.15} - \frac{VP_{Out}}{T_{Out}+273.15}\right)$

```
def MV_air_out(): #* equation 8.45 and according to van11 on page 216,
f_air_out = f_vent_side + f_vent_forced
    return (M_water/R) * (f_Vent_side() + f_Vent_forced()) * (Vp_air/(T_air
    + 273.15) - Vp_out/(T_out + 273.15))
print("MV_air_out",MV_air_out())
```

$f_{VentSide}$	$f_{VentForced}$	VP_{Out}	T_{Out}
0.000145	0	1000	297.05

$\rightarrow MV_{AirOut} = 8.100860849924075 * 10^{-5}$

- $MV_{TopOut} = \frac{M_{Water}}{R} f_{VentRoof} \left(\frac{VP_{Top}}{T_{Top}+273.15} - \frac{VP_{Out}}{T_{Out}+273.15} \right)$

```
def MV_top_out(): #* equation 8.45 and according to van11 on page 216, f_top_out = f_vent_roof
    return (M_water/R) * f_Vent_roof() * (Vp_top/(T_top + 273.15) - Vp_out/(T_out + 273.15))
print("MV_top_out",MV_top_out())
```

$f_{VentRoof} = f''_{VentRoof} + 0.5 f_{leakage} = 0.023783370636791715 + 0.5 * 0.00029 = 0.02392837064$
$\rightarrow MV_{TopOut} = 4.766389152705058 * 10^{-5}$

- $HEC_{TopCov,in} = c_{HECin}(T_{Top} - T_{Cov,in})^{0.33} \frac{A_{Cov}}{A_{Flr}}$
- $HEC_{AirThScr} = 1.7 U_{ThScr} |T_{Air} - T_{ThScr}|^{0.33}$
- $HEC_{AirMech} = \frac{U_{MechCool} COP_{MechCool} P_{MechCool}/A_{Flr}}{T_{Air} - T_{MechCool} + 6.4 \cdot 10^{-9} \Delta H (VP_{Air} - VP_{MechCool})}$

```
def HEC_top_cov_in(): #* to cal MV_top_cov_in
    return (c_HEC_in * A_cov * ((T_top - T_cov_in) ** 0.33))/A_flr #! be careful of negative number power a float number
print("HEC_top_cov_in",HEC_top_cov_in())

def HEC_air_Th_Scr(): #* to cal MV_air_Th_Scr
    return 1.7 * U_Th_Scr * (abs(T_air - T_Th_Scr) ** 0.33)
print("HEC_air_Th_Scr",HEC_air_Th_Scr())

def HEC_air_mech():#*equation 8.63 to cal MV_air_mech
    numerator = (U_mech_cool * Cop_mech_cool * P_mech_cool)/A_flr
    denominator = T_air - T_mech_cool + 6.4 * (10.0 **(-9.0)) * delta_H * (Vp_air - Vp_mach_cool)
    return numerator/denominator
print("HEC_air_mech",HEC_air_mech())
```

c_{HECin}	T_{Top}	$T_{Cov,in}$	A_{Cov}	A_{Flr}
1.86	295.15	294.05	90000	78000

$\rightarrow HEC_{TopCov,in} = 2.2147282078336135$

U_{ThScr}	T_{Air}	T_{ThScr}
0.4	291.15	294.05

$\rightarrow HEC_{AirThScr} = 0.966273906746953$

- $MV_{AirMech} = \begin{cases} 0 & \text{if } VP_{Air} < VP_{Mech} \\ 6.4 \cdot 10^{-9} HEC_{AirMech}(VP_{Air} - VP_{Mech}) & \text{if } VP_{Air} > VP_{Mech} \end{cases}$

```
Vp_mech = 0.0
def MV_air_mech(): #* equation 8.43 page 221
    if Vp_air <= Vp_mech :
        return 0
    else :
        return 6.4 * (10.0 ** (-9.0)) * HEC_air_mech() * (Vp_air - Vp_mech)
print("MV_air_mech",MV_air_mech())
```

Since the greenhouse model in Texas does not have cooling system, we get $MV_{AirMech} = 0$.

- $MV_{AirThScr} = \begin{cases} 0 & \text{if } VP_{Air} < VP_{ThScr} \\ 6.4 \cdot 10^{-9} HEC_{AirThScr}(VP_{Air} - VP_{ThScr}) & \text{if } VP_{Air} > VP_{ThScr} \end{cases}$

```
Vp_Th_Scr = 0.0
def MV_air_Th_Scr(): #* equation 8.43 page 215
    if Vp_air <= Vp_Th_Scr :
        return 0
    else :
        return 6.4 * (10.0 ** (-9.0)) * HEC_air_Th_Scr() * (Vp_air -
        Vp_Th_Scr)
```

$HEC_{AirThScr}$	VP_{Air}	VP_{ThScr}
0.966273906746953	1700	2400

$\rightarrow MV_{AirThScr} = 0$

- $MV_{TopCov,in} = \begin{cases} 0 & \text{if } VP_{Top} < VP_{Cov,in} \\ 6.4 \cdot 10^{-9} HEC_{TopCov,in}(VP_{Top} - VP_{Cov,in}) & \text{if } VP_{Top} > VP_{Cov,in} \end{cases}$

```
def MV_top_cov_in(): #* equation 8.43 page 215
    if Vp_top <= Vp_cov_in :
        return 0
    else :
        return 6.4 * (10.0 ** (-9.0)) * HEC_top_cov_in() * (Vp_air -
        Vp_cov_in)
```

```
print("MV_top_cov_in",MV_top_cov_in())
```

$HEC_{TopCov,in}$	VP_{Top}	$VP_{Cov,in}$
2.2147282078336135	1650	2400

$\rightarrow MV_{TopCov,in} = 0$

$$\rightarrow cap_{VP_{Air}} \dot{VP}_{Air} = MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BlowAir} - MV_{AirThScr} - MV_{AirTop} - MV_{AirOut} - MV_{AirOut\_Pad} - MV_{AirMech} \quad [kg\ m^{-2}\ s^{-1}]$$

dx_{VpAir} is calculated by dividing the right side by cap_VP

```
def dx_Vp():
    return (MV_can_air() + MV_pad_air() + MV_frog_air() + MV_blow_air() -
    MV_air_Th_Scr() - MV_air_top() - MV_air_out() - MV_air_out_pad() -
    MV_air_mech())/cap_Vp_air(), (MV_air_top() - MV_top_cov_in() -
    MV_top_out())/cap_Vp_top()
```

This function will return two results: $\dot{VP}_{Air} = -1.9793635132871816$ and $\dot{VP}_{Top} = -11.781252917521945$.

5.3 Calculating vapor pressure with Euler and Runge-Kutta algorithms

- Euler algorithm:

```
  # ! CLASSES
@dataclass
class Euler:
    x_0: float = 0.0
    y_0: float = 0.0
    t: float = 0.0
    h: float = 0.0

    def __init__(self, x0, y0, t0, h):
        self.x_0 = x0
        self.y_0 = y0
        self.t = t0
        self.h = h

    def calculateNewValue(self, x, y):
        x_new = x + self.dx_dt(x, y, self.t) * h
        y_new = y + self.dy_dt(x, y, self.t) * h

        return x_new, y_new

```

```
    def calculateNext(self, x, y):
        self.t = self.t + h

        yield self.calculateNewValue(x, y)

    def dx_dt(self, x, y, t):
        rhs = (
            VEC_can_air() * (Vp_can - x) #! MV_can_air
            + MV_pad_air()
            + MV_frog_air()
            + MV_blow_air()
            - 6.4 * (10.0 ** (-9.0)) * HEC_air_Th_Scr() * (x - Vp_Th_Scr)
            #! MV_air_Th_scr
            - (M_water/R) * f_Th_Scr() * (x/T_air - y/T_top) #! MV_air_top
            - (M_water/R) * (f_Vent_side() + f_Vent_forced()) * (x/T_air -
            Vp_out/T_out) #! MV_air_out
            - MV_air_out_pad()
            - MV_air_mech()
        )

        return rhs / cap_Vp_air()

    def dy_dt(self, x, y, t):
        rhs = (
            (M_water/R) * f_Th_Scr() * (x/T_air - y/T_top) #! MV_air_top
            - (M_water/R) * f_Vent_roof() * (y/T_top - Vp_out/T_out) #!
            MV_top_out
            - 6.4 * (10.0 ** (-9.0)) * HEC_top_cov_in() * (x - Vp_cov_in)
            #! MV_top_cov_in
        )

        return rhs / cap_Vp_top()


print("RUNNING EULER")
x = 1700.0 #! starting point of VPair
y = 1650.0 #! starting point of VPtop
t = 157.0
h = 1
euler = Euler(x, y, t, h)  # x_0, y_0, t_0, h

list_x = [x]
list_t = [t]
for n in range(5):
    t = t + h
    list_t.append(t)
list_y = [y]
for n in range(5):
```

```
    gen_new_val = euler.calculateNext(x, y)
    x, y = next(gen_new_val)

    print(f"step: {n + 1}")
    print(f"new x: {x}\nnew y: {y}\n\n")
    list_x.append(x)
    list_y.append(y)


plt.plot(list_t, list_x, "-b", label = "VP_air")
plt.plot(list_t, list_y, "-r", label = "VP_top")
plt.title("The Vapour pressure below and above the thermal screen from DOY
157-162")
plt.xlabel("time(days)")
plt.ylabel("Vapour pressure")
plt.legend()
plt.show()
```

- Runge-Kutta of order 4 algorithm:

```
  # ! CLASSES
@dataclass
class RK4:
    x_0: float = 0.0
    y_0: float = 0.0
    t: float = 0.0
    h: float = 0.0

    def __init__(self, x0, y0, t0, h):
        self.x_0 = x0
        self.y_0 = y0
        self.t = t0
        self.h = h

    def calculateNewValue(self, x, y):
        k0_x, k0_y = self.calculateFirstSlop(x, y)
        k1_x, k1_y = self.calculateMiddleSlop(x, y, k0_x, k0_y)
        k2_x, k2_y = self.calculateMiddleSlop(x, y, k1_x, k1_y)
        k3_x, k3_y = self.calculateLastSlop(x, y, k2_x, k2_y)

        x_new = x + (k0_x + 2 * k1_x + 2 * k2_x + k3_x) / 6.0
        y_new = y + (k0_y + 2 * k1_y + 2 * k2_y + k3_y) / 6.0

        return x_new, y_new

    def calculateFirstSlop(self, x, y):
        k0_x = h * self.dx_dt(x, y, self.t)
        k0_y = h * self.dy_dt(x, y, self.t)
```

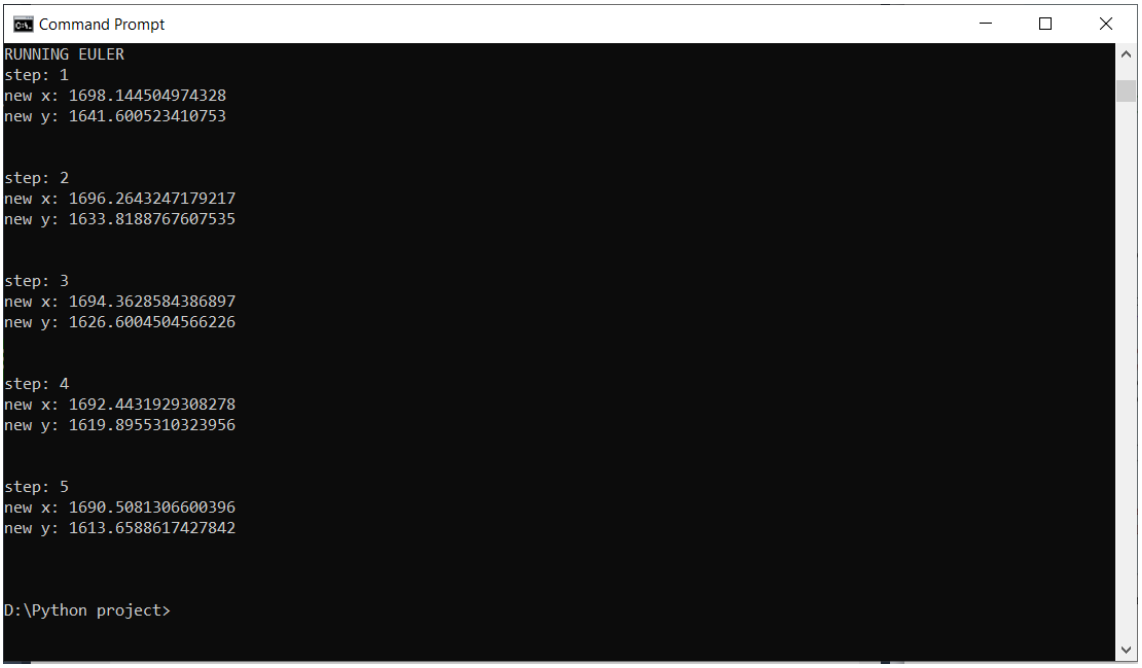
```
29
30          return k0_x, k0_y
31
32      def calculateMiddleSlop(self, x, y, k_prev_x, k_prev_y):
33          k_next_x = h * self.dx_dt(
34              x + k_prev_x * 0.5, y + k_prev_y * 0.5, self.t + h * 0.5
35          )
36          k_next_y = h * self.dy_dt(
37              x + k_prev_x * 0.5, y + k_prev_y * 0.5, self.t + h * 0.5
38          )
39
40          return k_next_x, k_next_y
41
42      def calculateLastSlop(self, x, y, k_prev_x, k_prev_y):
43          k_last_x = h * self.dx_dt(x + k_prev_x, y + k_prev_y, self.t + h)
44          k_last_y = h * self.dy_dt(x + k_prev_x, y + k_prev_y, self.t + h)
45
46          return k_last_x, k_last_y
47
48      def calculateNext(self, x, y):
49          self.t = self.t + h
50
51          yield self.calculateNewValue(x, y)
52
53      def dx_dt(self, x, y, t):
54          rhs = (
55              VEC_can_air() * (Vp_can - x) #! MV_can_air
56              + MV_pad_air()
57              + MV_frog_air()
58              + MV_blow_air()
59              - 6.4 * (10.0 ** (-9.0)) * HEC_air_Th_Scr() * (x - Vp_Th_Scr)
                #! MV_air_Th_scr
60              - (M_water/R) * f_Th_Scr() * (x/T_air - y/T_top) #! MV_air_top
61              - (M_water/R) * (f_Vent_side() + f_Vent_forced()) * (x/T_air -
                Vp_out/T_out) #! MV_air_out
62              - MV_air_out_pad()
63              - MV_air_mech()
64          )
65
66          return rhs / cap_Vp_air()
67
68      def dy_dt(self, x, y, t):
69          rhs = (
70              (M_water/R) * f_Th_Scr() * (x/T_air - y/T_top) #! MV_air_top
71              - (M_water/R) * f_Vent_roof() * (y/T_top - Vp_out/T_out) #!
                MV_top_out
72              - 6.4 * (10.0 ** (-9.0)) * HEC_top_cov_in() * (x - Vp_cov_in)
                #! MV_top_cov_in
```

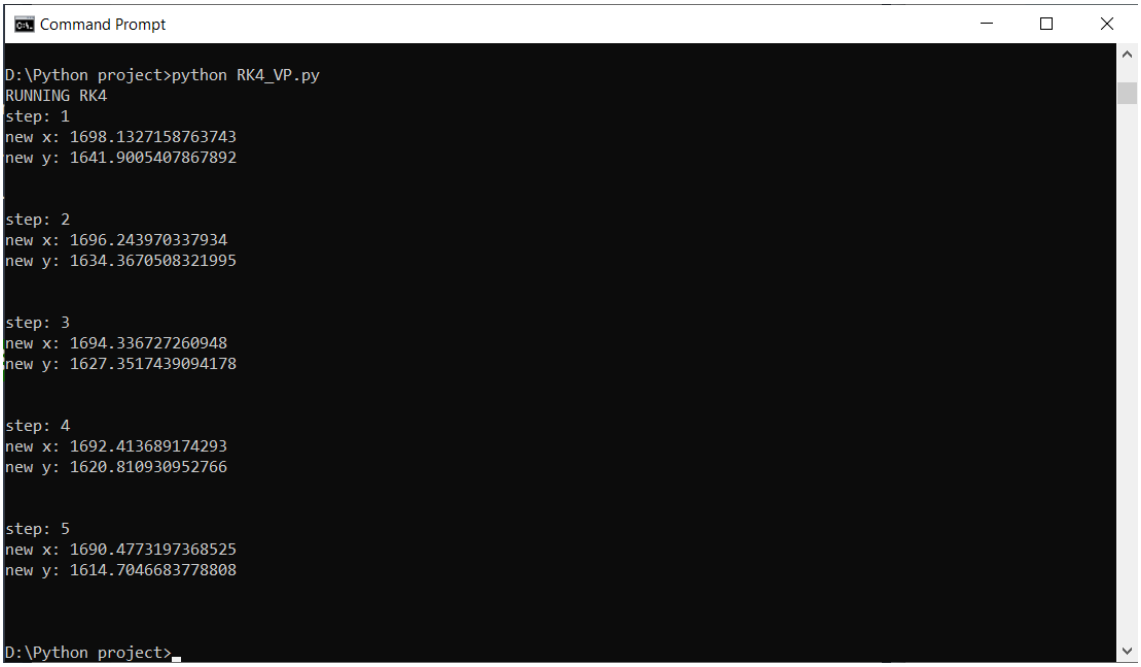
```
        )

        return rhs / cap_Vp_top()


print("RUNNING RK4")
x = 1700.0 #! starting point of VPair
y = 1650.0 #! starting point of VPtop0
t = 157.0
h = 1
rk4 = RK4(x, y, t, h)  # x_0, y_0, t_0, h

list_x = [x]
list_t = [t]
for n in range(5):
    t = t + h
    list_t.append(t)
list_y = [y]
for n in range(5):
    gen_new_val = rk4.calculateNext(x, y)
    x, y = next(gen_new_val)

    print(f"step: {n + 1}")
    print(f"new x: {x}\nnew y: {y}\n\n")
    list_x.append(x)
    list_y.append(y)


plt.plot(list_t, list_x, "-b", label = "VP_air")
plt.plot(list_t, list_y, "-r", label = "VP_top")
plt.title("The Vapour pressure below and above the thermal screen from DOY
157-162")
plt.xlabel("time(days)")
plt.ylabel("Vapour pressure")
plt.legend()
plt.show()
```

5.4 Comparing to actual data and comment on accuracy of the model

From initial state at which $t_0 = 157$, choosing $VP_{Air} = 1700Pa$ and $VP_{Top} = 1650Pa$ as initial values, we have the following approximation values of VP_{Top}, VP_{Air} in the next 5 minutes, 10 minutes, 20 minutes, 25 minutes, represented as the table below. The actual values of VP_{Air} and VP_{Top} are not available for comparison.

The result of the program: x is VP_{Air} and y is VP_{Top}





The graph below is used to compare to actual data from Van11.

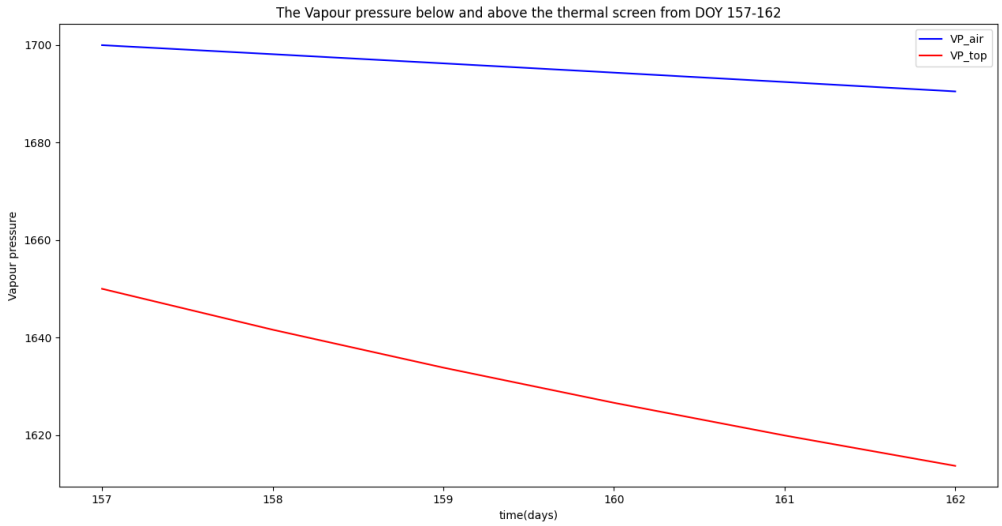


Figure 7: Result of Euler method

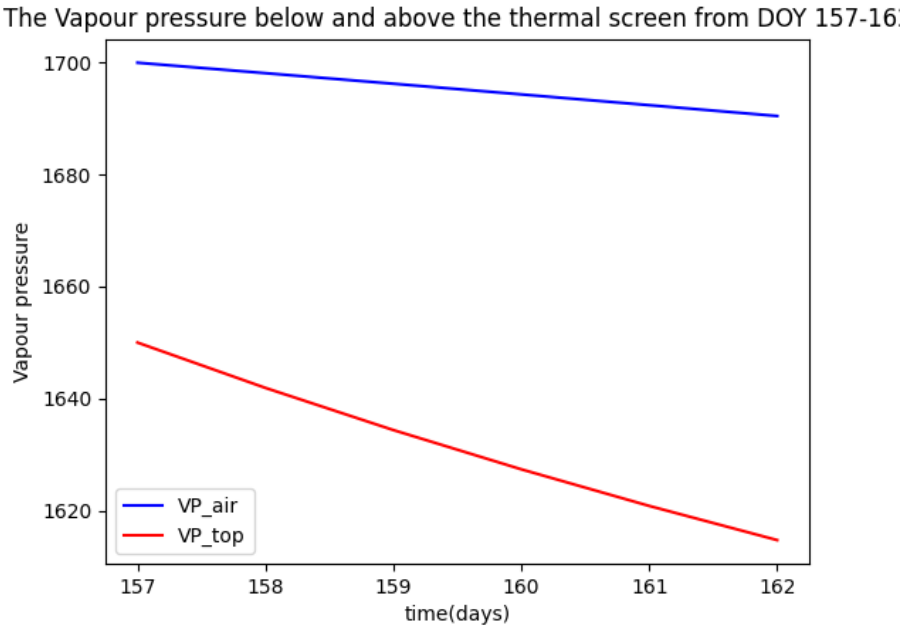


Figure 8: Result of RK4 method

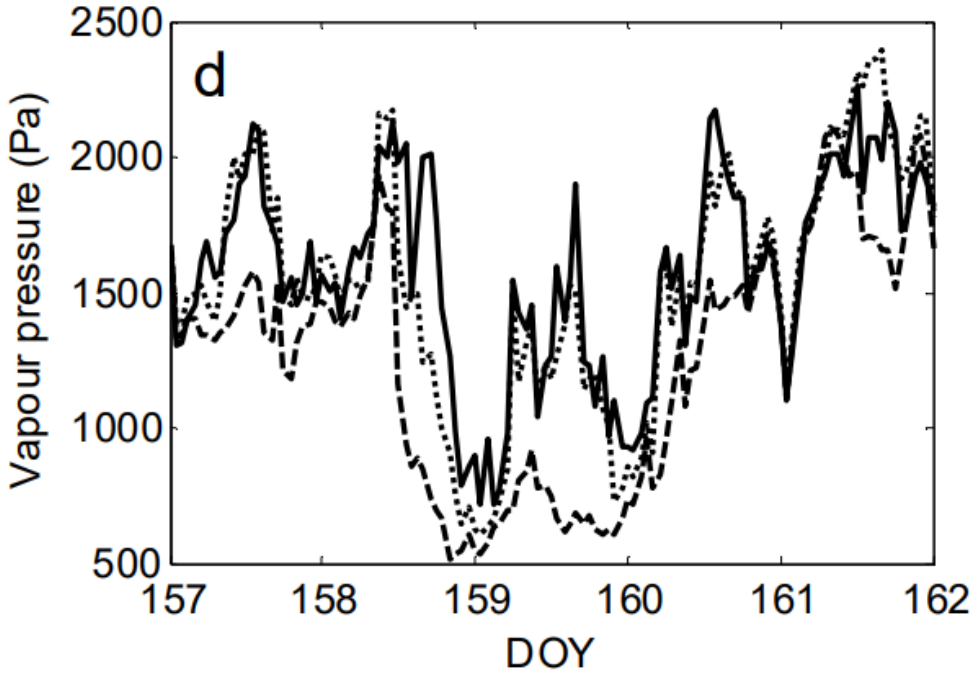


Figure 9: Actual Data from Van11 page 40

Conclusion, even though the predictions indeed resemble the general trends in actual data, they are not close to the actual data as the difference is fairly large and is clearly noticeable in the graph.This can stem from the fact that we lack data and our model is chosen arbitrarily(arbitrary components, the climate of Texas in the data set, and the photosynthetic rate of tomatoes) despite the unawareness of the actual greenhouse model used. So,our model is not cut out for simulating the change in vapour pressure compared to the given model.

References

[Sib75] Konstantin Sergeevich Sibirsky. *Introduction to topological dynamics*. Leiden: Noordhoff International Publishing, 1975.

[Sta87] Cecilia Stanghellini. "Transpiration of greenhouse crops: an aid to climate management". PhD thesis. IMAG, 1987.

[Bal92] LJ Balemans. "Assessment of criteria for energetic effectiveness of greenhouse screens.". In: (1992).

[Bap07] Fátima Baptista. "Modelling the climate in unheated tomato greenhouses and predicting Botrytis cinerea infection". In: (2007).

[Van11] Bram HE Vanthoor. *A model-based greenhouse design method*. 2011.

[Kje15] Jill B. Kjellsson. "The Energy-Water Nexus: Spatially-Resolved Analysis of the Potential for Desalinating Brackish Groundwater by Use of Solar Energy.". In: (2015).

[Tim] Padfield Tim. *Calculator for atmospheric moisture*. URL: https://conservationphysics.org/atmcalc/atmoclc2.pdf.