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

MATHEMATICAL MODELING (CO2011)

Assignment

Dynamical systems in forecasting Greenhouse Micro-climate

Advisor:	- Nguyen An Khuong.
Students:	- Phan Tuấn Khải 1952780.
	- Ngô Minh Đại 1913008.
	- Nguyễn Thanh Hùng 1952272.
	- Lưu Chấn Hưng 1952063.
	- Đinh Hoàng Anh 1952553.

HO CHI MINH CITY, SEPTEMBER 2020

Contents

1 Member list & Workload **2**

2 Background section: **3**

2.1 Exercise 1: 3

2.1.1 1.a 3

2.1.2 1.b 4

2.1.3 1.c 5

2.1.4 1.d 7

2.1.5 1.e 10

3 Application section: **13**

3.1 Exercise 2: 13

3.1.1 2.a 13

3.1.2 2.b 18

3.2 Exercise 3: 21

3.3 Exercise 4: 23

3.3.1 4.a 23

3.3.2 4.b 24

3.4 Exercise 5: 26

3.4.1 5.2a 26

3.4.2 5.2b 29

3.4.3 5.3 31

3.4.4 5.4a 33

3.4.5 5.4b 34

1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Luu Chan Hung	1952063	- Exercise 1d - Exercise 5...	20%
2	Dinh Hoang Anh	1952553	- Exercise 1e - Exercise 5...	20%
3	Ngo Minh Dai	1913008	- Exercise 1b,c - Exercise 2a - Exercise 5.2a	20%
4	Phan Tuan Khai	1952780	- Exercise 1a - Exercise 3 - Exercise 4	20%
5	Nguyen Thanh Hung	1952272	- Exercise 2b - Exercise 3 - Exercise 4	20%

2 Background section:

2.1 Exercise 1:

2.1.1 1.a

Present the definition and classification according to different criteria, the general form of dynamical systems, and especially first-order differential equations systems with initial condition at time t0, which are continuous dynamical systems used in this assignment. • Definition:
Mathematical models of scientific systems often lead to differential equations in which the independent variable is time. Such systems arise in astronomy, biology, chemistry, economics, engineering, physics and other disciplines. It is common to speak of a system of differential equations

$$\dot{x} = f(x)$$

as governing a dynamical system. A formal definition of dynamical system that is sometimes used in abstract studies is that of a family of maps $\phi(t, \cdot) : \Omega \to \Omega$, defined on a domain Ω parametrized by a variable t on an interval (a,b) and satisfying, for each p $\in \Omega$ the following three conditions:
(i) $\phi(0, p) = p$
(ii) $\phi \in C[(a, b) \times \Omega]$
(iii) $\phi(t + \alpha, p) = \phi(t, \phi(\alpha, p))$.
• Classification according to different criteria:
Consider the three mentioned conditions:
If in item (ii) the C is replaced by C^k with $k \geqslant 1$, the dynamical system is said to be differentiable. These studies can be viewed as abstractions of flows of autonomous differential equations. When the mapping ϕ is generated by an initial-value problem

$$\dot{x} = f(x), x(t_0) = p$$

it is a homeomorphism, i.e., a continuous map with a continuous inverse. If the vector field f is C^k, it is a diffeomorphism (which is differentiable map that possesses a differential inverse.)
There are also Discrete dynamical systems for which t takes on discrete values, but we shall consider continuous dynamical systems generated by autonomous system like equation $\dot{x} = f(x)$ - for which t takes on values in an interval of the real axis - unless otherwise specified.
• The general form of dynamical systems:
In the most general sense, a dynamical system is a tuple (T,M Φ) where T is a monoid, written additively, M is a non-empty set and Φ is a function
$\Phi : U \subseteq (T \times M) \to M$
with $proj_2$(U) = M (where $proj_2$ is the 2nd projection map)
I(x) = $\{t \in T : (t, x) \in U\}$
$\Phi(0, x) = x$
$\Phi(t_2, \Phi(t_1, x)) = \Phi(t_2 + t_1, x)$, for $t_1, t_2 + t_1 \in I(x)$ and $t_2 \in I(\Phi(t_1, x))$
The function Φ(t,x) is called the evolution funnction of the dynamical system: it associates to every point in the set M a unique image, depending on the variable t, called the evolution parameter. M is called phase space or state space. while the variable x represents an initial state of the system.
We often write
$\Phi_x(t) \equiv \Phi(t, x)$
$\Phi^t(x) \equiv \Phi(t, x)$

if we take one of the variables as constant.
$\Phi_x : I(x) \to M$
is called the flow through x and its graph trajectory through x. The set
$\gamma_x \equiv \{\Phi(t,x) : t \in I(x)\}$
is called the orbit through x. Note that the orbit through x is the image of the flow through x.
A subset S of the state space M is called **ϕ-invariant** if for all x in S and all t in T
$\Phi(t,x) \in S$
Thus, in particular, if S is **ϕ-invariant**, I(x) = T for all x in S. That is, the flow through x must be defined for all time for every element of S

• **The general form of first-order differential equations system:**
There is a general form of differential equation. Suppose $\phi = \phi(t,x,y)$ is a prescribed function of three variables and the problem is that of finding a function x(t) such that

$$\phi(t, x(t), \dot{x}(t)) = 0$$

on an interval of the t axis. The standard form of this equation would be

$$\frac{dx}{dt} = f(t,x)$$

where the function f is obtained by solving the equation $\phi(t,x,y) = 0$ for y as a function of t and x.

• **The initial-value problem**
We can only have a unique solution of the differential equation if we somehow specify this constant. One way to do this is to require that the function x(t) not only satisfy the differential equation but also satisfy the condition that it take on a prescribed value x_0 at a prescribed point t_0. This lack of uniqueness turns out to be characteristic of more general differential equations as well, and the remedy suggested above to render the solution unique turns out to be appropriate in a variety of naturally occurring circumstances. For this reason we define the initial-value problem:

$$\frac{dx}{dt} = f(t,x), x|_{t=t_0} = x_0$$

Here the function f and the initial data t_0, x_0 are prescribed.

• **First-order differential equations system in this assignment.**

$\dot{CO}_{2Air} = \frac{MC_{BlowAir}+MC_{ExtAir}+MC_{PadAir}-MC_{AirCan}-MC_{AirTop}-MC_{AirOut}}{cap_{CO_{2Air}}}$
$\dot{CO}_{2Top} = \frac{MC_{AirTop}-MC_{TopOut}}{cap_{CO_{2Top}}}$
With innitial data is $cap_{CO_{2Air}}$ and $cap_{CO_{2Top}}$

2.1.2 1.b

Introduce a necessary and sufficient condition for the above systems of differential equations to exist and have unique solutions. Solution:
We have the formula:
$\frac{dy}{dx}$ = *f(x,y)*
The problem how to solve $y = y(x)$ satisfied that $y(x_0) = y_0$ is called the cosi problem, and x_0, y_0 is called the initial value (the initial condition).
The problem is in which condion:

- The cosi problem has the solution.
- The solution is unique.

Solving the mentioned above issues means that the theorem of the unique solution.

Theorem: We have the formula $f(x, y, y^{'})$ with the initial value (x_0, y_0). Supposed that:

- $f(x, y)$ (x, y) is a two-variable continuous function in the inner bounded domain G

$$\begin{cases} x_0 - a \leq x \leq x_0 + a \\ y_0 - b \leq x \leq y_0 + b \end{cases}$$

  Since f is continuous in the inner bounded domain G, there exists $M > 0$ satisfied that $|f(x, y)| \leq M \ \forall (x, y) \in G$

- $f(x, y)$ is a function satisfying the condition Lipsit according to variable y in the inner bounded domain G

  Then there exists only one solution $y = y(x)$ of the equations $f(x, y, y^{'})$ defined and continuous for the values of x in the segment $x_0 - h \leq x \leq x_0 + h$
  Where, $h = min(a, \frac{b}{M})$ satisfied that when $x = x_0$, $y(x_0) = y_0$

2.1.3 1.c

Give some examples of solvable first-order differential equations and their exact solutions. • Example 1: Suppose that a bread is removed from a 100^oC oven and placed in a room with a temperature of 25^oC. In 20 min the bread has a temperature of 60^oC. We want to determine the time when the bread at a temperature of 89^oC.

Solution:
The rate of change of the temperature T(t) is the derivative $\frac{dT}{dt}$.
According to Newton's law of cooling, the rate at which the temperature T(t) changes in a cooling body is proportional to the difference between the temperature of the body and the constant temperature T_s of the surrounding medium. Symbolically, the statement is expressed as
$\frac{dT}{dt} = k(T - T_s)$
Where k is a constant of proportionality.
Separating the variables we have
$\frac{dT}{T - T_s} = kdt \iff \frac{dT}{T - 25} = kdt$

$\Rightarrow \int \frac{dT}{T - 25} = \int kdt \Rightarrow ln/T - 25/ = kt + ln\ C$

$\Rightarrow T - 25 = e^{kt+lnC} = Ce^{kt}$

To find k we use the condition: if t = 20min then T = 60^oC. So

$60 - 25 = 75e^{k.20} \Rightarrow e^k = (\frac{7}{15})^{\frac{1}{20}}$

$k = \frac{1}{20}.ln(\frac{7}{15})$

Therefore

$T = 75(\frac{7}{15})^{\frac{t}{20}} + 25$

We want to know when T(t) = 89^oC. Thus we solve

$89 = 75(\frac{7}{15})^{\frac{t}{20}} + 25 \Rightarrow t \approx 4\ min$

• Example 2: Find the solution of the initial-value problem and calculate y(1).
y' + y = x, y(0) = 1.

Solution:
P(x)= 1 ,Q(x)=x.
We have the formula for this solution.

$\frac{dy}{dx} + P(x)y = Q(x)$

Mutiplying both sides of the aboved formula to $e^{\int P(x)dx}$, we have:

$e^{\int P(x)dx}.\frac{dy}{dx} + e^{\int P(x)dx}.P(x)y = e^{\int P(x)dx}.Q(x)$

Taking the primitive of both sides:

$e^{\int P(x)dx}.y = \int Q(x).e^{\int P(x)dx}dx + C$

$y = e^{-\int P(x)dx}.\left[\int Q(x).e^{\int P(x)dx}dx + C\right]$

Apply the formula mentioned above, we have

$y = e^{-\int 1dx}.\left[\int x.e^{\int 1dx}dx + C\right] = e^{-x}.[\int x.e^{x}dx + C]$

$= e^{-x}.[(x - 1).e^{x}dx + C] = (x - 1) + Ce^{-x}$

Since $x = 0,\ y = 1 \Rightarrow 1 = (0 - 1) + Ce^{0} \Rightarrow C = 2$. Therefore the solution to the initial-value problem is $y = x - 1 + 2e^{-x}$.
Finally, it's easy to find $y(1) = 1 - 1 + 2e^{-1} = 2e^{-1}$.

• Example 3: We will solve the example 2 with another method: Picard's Iteration Scheme.

Picard's Theorem: is proved by applying Picard's iteration scheme, which we now introduce. We begin by noticing that any solution to the initial value problem of Equations:

$\frac{dy}{dx} = f(x,y),\ y(x_o) = y_o$ (1)

It must also satisfy the integral equation:
$y(x) = y_o + \int_{x_o}^{x} f((t,y(t))dt$ (2)
because
$\int_{x_o}^{x} \frac{dy}{dt}dt = y(x) - y(x_o)$.
The converse is also true: if y(x) satisfies Equation (2), then y' = ((t,y(t)) and y(x_o) = y_o. So Equation (1) may be replaced by Equation (2). This sets the stage for Picard's interation

method: In the integrand in Equation (2), replace y(t) by the constant y_o, then integrate and call the resulting right-hand side of Equation (2) y_1(x):

$y_1(x) = y_o + \int\limits_{x_o}^{x} f(t,y_o)\,dt$ (3)

This starts the process. To keep it goint, we use the iterative formulas

$y_{n+1}(x) = y_o + \int\limits_{x_o}^{x} f(t,y_n(t))\,dt$ (4)

Solution:
Return to Example 2:
y' = x - y, y(0) = 1
For the problem at hand, *f(x,y) = x - y*. And Equation (3) becomes
$y_2(x) = 1 + \int\limits_{0}^{x} f\left(t - 1 + t - x^2 + \frac{1}{6}t^3\right)dt = 1 - x + x^2 - \frac{1}{3}x^3 + \frac{1}{4!}\,x^4$

In this example it is possible to find the exact solution because
$\frac{dy}{dx} + y = x$

is a first-order differential equation that is linear in y. We have already shown how to find the general solution

$y = x - 1 + Ce^{-x}$

in the previous example. The solution of the initial value problem is then

$y = x - 1 + 2e^{-x}$

If we substitute the Maclaurin series for e^{-x} in this particular solution, we get

$y = x - 1 + 2\left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - ...\right) = 1 - x + x^2 - \frac{x^3}{3!} + 2\left(\frac{x^4}{4!} - \frac{x^5}{5!} + ...\right)$

The following example illustrate the Picard iteration scheme, but in most practical cases the computations soon become too burdensome to continue. However, it is one way we could get an idea of how the solution behaves near the initial point.

2.1.4 1.d

Euler's method Given a differential equation $\frac{dy}{dx} = f(x, y)$ and an initial condition $y(x_0) = y_0$, we can approximate the solution $y = y(x)$ by its linearization:

$$L(x) = y(x_0) + y^{'}(x_0)(x - x_0)$$

or

$$L(x) = y_0 + f(x_0, y_0)(x - x_0)$$

The function $L(x)$ gives a good approximation to the solution $y(x)$ in a short interval about x_0. The basis of Euler's method is to path together a string of linearization to approximate the curve over a longer stretch.

- **Step 1:** We know the point (x_0, y_0) lies on the solution curve. So from this point which lies exactly on the solution curve, we have obtained the point (x_1, y_1), which lies very close to the point $(x_1, y(x_1))$ on the solution curve by the equation:

$$y_1 = L(x_1) = y_0 + f(x_0, y_0)dx$$

- **Step 2:** Using the point (x_1, y_1) and the slope $f(x_1, y_1)$ on the solution curve through (x_1, y_1), we take the second step. Setting $x_2 = x_1 + dx$, we use the linearization of the solution curve through (x_1, y_1) to calculate:

$$y_2 = L(x_2) = y_1 + f(x_1, y_1)dx$$

- **Step 3:** Continuing in this fashion, we take the third step and so on. We build literally an approximation to one of the solution by following the direction of the slope field of the differential equation.

The step (dx) would be small enough to calculate a good approximation. Euler's method is easy to implement on a computer or calculator. A computer generates a table of numerical solutions to an initial value problem, allowing us to input x_0 and y_0, the number of step n, and the step size dx. It then calculates the approximate solution value $y_1, y_2...y_n$ in iterative fashion, as just described.

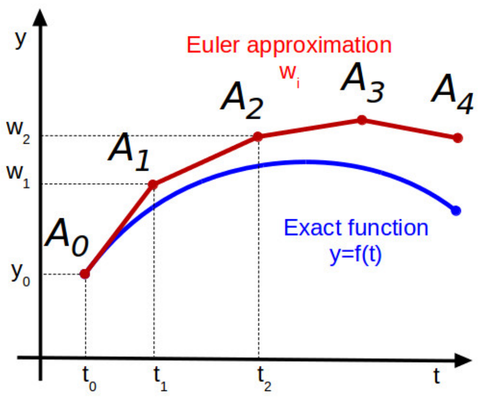


Figure 1: Visualization of Euler's method

Explicit Runge-Kutta of order 4 algorithms While Euler's method is simple to program on a computer, it is inherently unstable, only first order accurate, and requires very small Δt in order to achieve reasonable results.

Mathematicians and engineers have developed clever algorithms to modify the slope such that information is used from one or more values of t between t_n and t_{n+1}. These algorithms include the implicit Euler method and various kinds of predictor-corrector techniques, which can be formulated to first-, second-, or higher-order accuracy.

The Runge-Kutta technique is fourth-order accurate, and can be thought of as a kind of predictor-corrector technique in that the final value of y_{n+1} at $t = t_{n+1}$ is calculated as:

$$y_{n+1} = y_n + \Delta y_{final}$$

where increment Δy_{final} is a weighted average of four "trial increments," namely Δy_1, Δy_2, Δy_3, and Δy_4, evaluated from slopes calculated at $t = t_n$, $t_{n+\frac{1}{2}}$, $t_{n+\frac{1}{2}}$, and t_{n+1}, respectively. The final predicted value of y_{n+1} at $t = t_{n+1}$ is calculated as:

$$\Delta y_{final} = \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

- **Step 1:** Compute the slope at x_n:

$$k_1 = f(x_n, y_n)$$

- **Step 2:** Compute the slope at the midpoint from k_1:

$$k_2 = f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_1)$$

- **Step 3:** Compute the slope at the midpoint from k_2:

$$k_3 = f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_2)$$

- **Step 4:** Compute the slope at the endpoint from k_3:

$$k_4 = f(x_n + h, y_n + hk_3)$$

- **Step 5:** Compute y_{n+1}:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

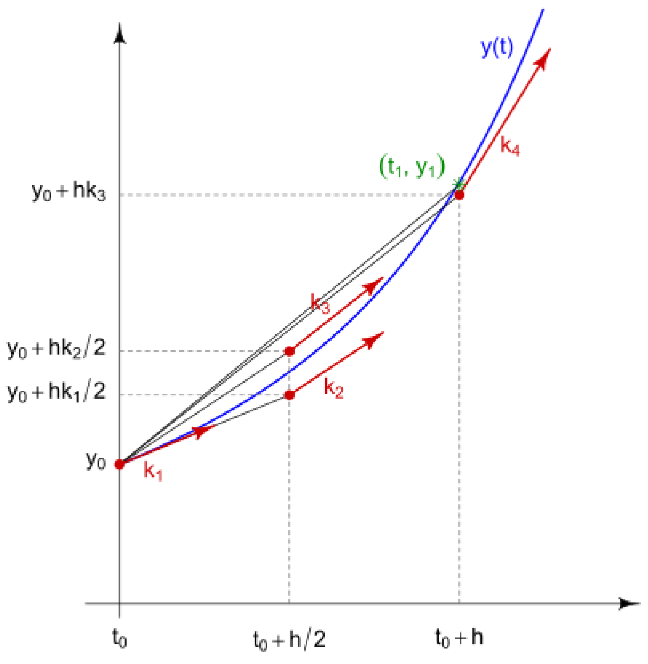


Figure 2: Visualization of Runge-Kutta method

2.1.5 1.e

Using Explicit Euler and Explicit Runge-Kutta, give approximate values of the exact solutions of the above examples at time t_0, $t_0 + h$, $t_0 + 2h$, ..., $t_0 + 5h$ with optional h. Example 1: Suppose that a bread is removed from a 100^oC oven and placed in a room with a temperature of 25^oC. In 20 min the bread has a temperature of 60^oC. We want to determine the temperature after 4 min
We have:

$$\frac{dT}{dt} = \frac{1}{20}ln\frac{7}{15}(T - 25)$$

• **Euler's method:**
The initial point is $t_0 = 0$ and $T_0 = 100$ with step $h = dt = 0.8$ (min)

$$T_1 = T_0 + \frac{1}{20}ln\frac{7}{15}(T - 25)dt = 100 + \frac{1}{20}ln\frac{7}{15}(100 - 25)0.8 = 97.714$$

$$T_2 = T_1 + \frac{1}{20}ln\frac{7}{15}(T - 25)dt = 97.714 + \frac{1}{20}ln\frac{7}{15}(97.714 - 25)0.8 = 95.497$$

$$T_3 = T_2 + \frac{1}{20}ln\frac{7}{15}(T - 25)dt = 95.497 + \frac{1}{20}ln\frac{7}{15}(95.497 - 25)0.8 = 93.348$$

$$T_4 = T_3 + \frac{1}{20}ln\frac{7}{15}(T - 25)dt = 93.348 + \frac{1}{20}ln\frac{7}{15}(93.348 - 25)0.8 = 91.264$$

$$T_5 = T_4 + \frac{1}{20}ln\frac{7}{15}(T - 25)dt = 91.264 + \frac{1}{20}ln\frac{7}{15}(91.264 - 25)0.8 = 89.244$$

As we can see the approximation is 89.244 compare with the exact number is 89.3966. The error is 0.17%
• **Runge-Kutta's method:**

Step 1

$$k_1 = f(0, 100) = \frac{1}{20}ln\frac{7}{15}(100 - 25) = -2.858$$

$$k_2 = f(0 + \frac{1}{2} * 0.8, 100 + \frac{1}{2} * 0.8 * (-2.858)) = -2.8145$$

$$k_3 = f(0 + \frac{1}{2} * 0.8, 100 + \frac{1}{2} * 0.8 * (-2.8145)) = -2.8151$$

$$k_4 = f(0 + 0.8, 100 + 0.8 * (-2.8151)) = -2.7722$$

$$y(0.8) = 100 + \frac{0.8}{6}(-2.858 + 2 * (-2.8145) + 2 * (-2.8151) - 2.7722) = 97.7481$$

Step 2

$$k_1 = f(0.8, 97.7481) = \frac{1}{20}ln\frac{7}{15}(97.7481 - 25) = -2.7722$$

$$k_2 = f(0.8 + \frac{1}{2} * 0.8, 97.7481 + \frac{1}{2} * 0.8 * (-2.7722)) = -2.73$$

$$k_3 = f(0.8 + \frac{1}{2} * 0.8, 97.7481 + \frac{1}{2} * 0.8 * (-2.73)) = -2.7306$$

$$k_4 = f(0.8 + 0.8, 97.7481 + 0.8 * (-2.7306)) = -2.689$$

$$y(1.6) = 97.7481 + \frac{0.8}{6}(-2.7722 + 2 * (-2.73) + 2 * (-2.7306) - 2.689) = 95.5638$$

Step 3

$$k_1 = f(1.6, 95.5638) = \frac{1}{20}ln\frac{7}{15}(95.5638 - 25) = -2.689$$

$$k_2 = f(1.6 + \frac{1}{2} * 0.8, 95.5638 + \frac{1}{2} * 0.8 * (-2.689)) = -2.648$$

$$k_3 = f(1.6 + \frac{1}{2} * 0.8, 95.5638 + \frac{1}{2} * 0.8 * (-2.648)) = -2.649$$

$$k_4 = f(1.6 + 0.8, 95.5638 + 0.8 * (-2.649)) = -2.608$$

$$y(2.4) = 95.5638 + \frac{0.8}{6}(-2.689 + 2 * (-2.648) + 2 * (-2.649) - 2.608) = 93.449$$

Step 4

$$k_1 = f(2.4, 93.449) = \frac{1}{20}ln\frac{7}{15}(93.449 - 25) = -2.6084$$

$$k_2 = f(2.4 + \frac{1}{2} * 0.8, 93.449 + \frac{1}{2} * 0.8 * (-2.6084)) = -2.5686$$

$$k_3 = f(2.4 + \frac{1}{2} * 0.8, 93.449 + \frac{1}{2} * 0.8 * (-2.5686)) = -2.5692$$

$$k_4 = f(2.4 + 0.8, 93.449 + 0.8 * (-2.5692)) = -2.53$$

$$y(3.2) = 93.449 + \frac{0.8}{6}(-2.6084 + 2 * (-2.5686) + 2 * (-2.5692) - 2.53) = 91.4438$$

Step 1

$$k_1 = f(3.2, 91.4438) = \frac{1}{20}ln\frac{7}{15}(91.4438 - 25) = -2.532$$

$$k_2 = f(3.2 + \frac{1}{2} * 0.8, 91.4438 + \frac{1}{2} * 0.8 * (-2.532)) = -2.4934$$

$$k_3 = f(3.2 + \frac{1}{2} * 0.8, 91.4438 + \frac{1}{2} * 0.8 * (-2.4934)) = -2.494$$

$$k_4 = f(3.2 + 0.8, 91.4438 + 0.8 * (-2.494)) = -2.4469$$

$$y(4) = 91.4438 + \frac{0.8}{6}(-2.532 + 2 * (-2.4934) + 2 * (-2.494) - 2.4469) = 89.45$$

As we can see the approximation is 89.45 compare with the exact number is 89.3966. The error is 0.06%

Example 2: Find the solution of the initial-value problem and calculate $y(1)$.

$$y' + y = x,\ y(0) = 1$$

We have:

$$\frac{dy}{dx} = x - y$$

• **Euler's method:**
The initial point is $x_0 = 0$ and $y_0 = 1$ with step $h = dt = 0.2$

$$y_1 = y_0 + (x_0 - y_0)dx = 1 + (0 - 1) * 0.2 = 0.8$$

$$y_2 = y_1 + (x_1 - y_1)dx = 0.8 + (0.2 - 0.8) * 0.2 = 0.68$$

$$y_3 = y_2 + (x_2 - y_2)dx = 0.68 + (0.4 - 0.68) * 0.2 = 0.624$$

$$y_4 = y_3 + (x_3 - y_3)dx = 0.628 + (0.6 - 0.628) * 0.2 = 0.6192$$

$$y_5 = y_5 + (x_5 - y_5)dx = 0.6192 + (0.8 - 0.6192) * 0.2 = 0.65032$$

As we can see the approximation is 0.65032 compare with the exact number is $2e^{-1}$ The error is 11.61%

• **Runge-Kutta's method:**

Step 1

$$k_1 = f(0, 1) = 0 - 1 = -1$$

$$k_2 = f(0 + \frac{1}{2} * 0.2, 1 + \frac{1}{2} * 0.2 * (-1)) = -0.8$$

$$k_3 = f(0 + \frac{1}{2} * 0.2, 1 + \frac{1}{2} * 0.2 * (-0.8)) = -0.82$$

$$k_4 = f(0 + 0.2, 1 + 0.2 * (-0.82)) = -0.64$$

$$y(0.2) = 1 + \frac{0.2}{6}(-1 + 2 * (-0.8) + 2 * (-0.82) - 0.64) = 0.84$$

Step 2

$$k_1 = f(0.2, 0.84) = 0.2 - 0.84 = -0.64$$

$$k_2 = f(0.2 + \frac{1}{2} * 0.2, 0.84 + \frac{1}{2} * 0.2 * (-0.64)) = -0.48$$

$$k_3 = f(0.2 + \frac{1}{2} * 0.2, 0.84 + \frac{1}{2} * 0.2 * (-0.48)) = -0.49$$

$$k_4 = f(0.2 + 0.2, 0.84 + 0.2 * (-0.49)) = -0.34$$

$$y(0.4) = 0.84 + \frac{0.2}{6}(-0.64 + 2 * (-0.48) + 2 * (-0.49) - 0.34) = 0.74$$

Step 3

$$k_1 = f(0.4, 0.74) = 0.4 - 0.74 = -0.34$$

$$k_2 = f(0.4 + \frac{1}{2} * 0.2, 0.74 + \frac{1}{2} * 0.2 * (-0.34)) = -0.21$$

$$k_3 = f(0.4 + \frac{1}{2} * 0.2, 0.74 + \frac{1}{2} * 0.2 * (-0.21)) = -0.22$$

$$k_4 = f(0.4 + 0.2, 0.74 + 0.2 * (-0.22)) = -0.1$$

$$y(0.6) = 0.74 + \frac{0.2}{6}(-0.34 + 2 * (-0.21) + 2 * (-0.22) - 0.1) = 0.7$$

Step 4

$$k_1 = f(0.6, 0.7) = 0.6 - 0.7 = -0.1$$

$$k_2 = f(0.6 + \frac{1}{2} * 0.2, 0.7 + \frac{1}{2} * 0.2 * (-0.1)) = 0.01$$

$$k_3 = f(0.6 + \frac{1}{2} * 0.2, 0.7 + \frac{1}{2} * 0.2 * (-0.01)) = -0.001$$

$$k_4 = f(0.6 + 0.2, 0.7 + 0.2 * (-0.001)) = 0.1002$$

$$y(0.6) = 0.7 + \frac{0.2}{6}(-0.1 + 2 * 0.01 + 2 * (-0.001) + 0.1002) = 0.7$$

Step 5

$$k_1 = f(0.8, 0.7) = 0.8 - 0.7 = 0.1$$

$$k_2 = f(0.8 + \frac{1}{2} * 0.2, 0.7 + \frac{1}{2} * 0.2 * 0.097) = 0.19$$

$$k_3 = f(0.8 + \frac{1}{2} * 0.2, 0.7 + \frac{1}{2} * 0.2 * 0.19) = 0.18$$

$$k_4 = f(0.8 + 0.2, 0.7 + 0.2 * 0.18) = 0.26$$

$$y(0.4) = 0.703 + \frac{0.2}{6}(0.1 + 2 * 0.19 + 2 * 0.18 + 0.26) = 0.74$$

As we can see the approximation is 0.74 compare with the exact number is $2e^{-1}$ The error is 0.58%

3 Application section:

3.1 Exercise 2:

3.1.1 2.a

Restate the model for the CO_2 concentration in greenhouses in detail as described in Sections 2, 3, and 4 of this assignment and write to the report. • The concentration of CO2 in the greenhouse air will be described in more detail.
We have the figure:
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.

CO_2 concentration of the greenhouse air and the air in the top compartment
In the range of the report, the greenhouse air CO_2 concentration CO_{2Air} is described by:
$cap_{CO_{2Air}}\dot{C}O_{2Air} = MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir}$ - MC_{AirCan} - MC_{AirTop} - MC_{AirOut}
The CO_2 concentration of the top compartment air CO_{2Top} is described by:
$cap_{CO_{2Top}}\dot{C}O_{2Top} = MC_{AirTop}$ - MC_{TopOut}
where:
• $cap_{CO_{2Top/Air}}$ is the capacity of the compartment above and below the thermal screen to store CO_2 respectively (m).
• $\dot{C}O_{2Top/Air}$ is the rate of change of CO_2 concentration in the compartment above and below the thermal screen in time (mg m^{-3} s^{-1}).
• Carbon dioxide is exchanged between the greenhouse air and surrounding elements i.e. the direct air heater $MC_{BlowAir}$, the external CO_2 source MC_{ExtAir}, the pad and fan system MC_{PadAir}, and the outdoor air MC_{AirOut}, MC_{AirCan} is the CO_2 flux between the greenhouse air and the canopy (mg m^{-2} s^{-1}).

Direct air heater
The head flux from the direct air heater to the greenhouse air is described by:
$H_{BlowAir} = U_{Blow}\ P_{Blow}\ /\ A_{Flr}$ [W m^{-2}]
where:

• U_{Blow} is the control value of the direct air heater ranging in [0, 1].
• P_{Blow} (W) is the heat capacity of the direct air heaters.
• A_{Flr} (m^2) is the surface of the greenhouse floor.

The CO_2 flux from the heat blower to the greenhouse air is proportional to the heat flux:
$MC_{BlowAir} = \eta_{HeatCO2}\ H_{BlowAir} = \frac{\eta_{HeatCO2}\ U_{Blow}\ P_{Blow}}{A_{Flr}}$ [mg m^{-2} s^{-1}]
where $\eta_{HeatCO2}$ (mg CO_2 J^{-1}) is the amount of CO_2 which is released when 1 Joule of sensible energy is produced by the direct air heater.

External CO_2 source
The CO_2 added to the greenhouse by an external CO_2-source is decribed by:
$MC_{ExtAir} = U_{ExtCO2}\ \phi_{ExtCO2}\ /\ A_{Flr}$ [mg m^{-2} s^{-1}]
where:
• U_{ExtCO2} is the control value of the external CO_2 source ranging in [0, 1].
• ϕ_{ExtCO2} (mg s^{-1}) is the capacity of the external CO_2 source.

On the other hand, the amount of CO_2 that enters the greenhouse through the pad system is due to the difference in the concentration of CO_2 inside and outside the greenhouse. It will be shown clearly in the formula:
$MC_{PadAir} = f_{Pad}(CO_{2Out} - CO_{2Air}) = \frac{U_{Pad}\ \phi_{ExtCO2}}{A_{Flr}}(CO_{2Out} - CO_{2Air})$
where:
• U_{Pad} is the control value of the pad and fan system ranging in [0, 1].
• f_{Pad} (m s^{-1}) is the ventilation flux due to the pad and fan system.
• ϕ_{Pad} (m^3 s^{-1}) is the capacity of the air flux through the pad.

Movement of air through the thermal screen
The net flux of CO_2 from the lower compartment to the upper compartment of the greenhouse is more complicated and it depends on the difference in the temperature and air density between the two compartments.
$MC_{AirTop} = f_{ThScr}(CO_{2Air} - CO_{2Top})$

The airflow rate through the thermal screen f_{ThScr} (m s^{-1}) is the sum of the penetration rate through the screen and the rate at the open regions that are not covered by the thermal screen.
$f_{ThScr} = U_{ThScr}K_{ThScr}|T_{Air} - T_{Top}|^{\frac{2}{3}} + (1 - U_{ThScr})\left[\frac{g(1 - U_{ThScr})}{2p^{Mean}_{Air}}|p_{Air} - p_{Top}|\right]^{\frac{1}{2}}.$
where:
• f_{ThScr} is the air flux through the thermal screen (m s^{-1});
• U_{ThScr} is the control of the thermal screen ranging in [0, 1];
• K_{ThScr} is the screen flux coefficient determining the permeability of the screen (m $K^{-\frac{2}{3}}$ s^{-1});
• g is the gravitational acceleration (m s^{-2});
• $p_{Air/Top}$ is the density of the greenhouse air below/above the thermal screen (kg m^{-3});
• p^{Mean}_{Air} is the mean density of the greenhouse air (kg m^{-3});
• $T_{Air/Top}$ is the temperature below and above the thermal screen (K);

CO_2 flux from the inside to the outside of the greenhouse
Let consider the formula:
$MC_{AirOut} = (f_{VentSide} + f_{VentForced})(CO_{2Air} - CO_{2Out})$
where:
• $f_{VentSide}$ is the rate for the sidewall ventilation system (m s^{-1});

• $U_{VentForced}$ is the the rate for the force ventilation system (m s^{-1});

To generalize the model for many different types of greenhouses, the following general formula $f_{VentRoofSide}$ (m s^{-1}) is used:

$$f_{VentRoofSide} = \frac{C_d}{A_{Flr}} \left[\frac{U_{Roof}^2 U_{Side}^2 A_{Roof}^2 A_{Side}^2}{U_{Roof}^2 A_{Roof}^2 + U_{Side}^2 A_{Side}^2} . \frac{2gh_{SideRoof}(T_{Air} - T_{Out})}{T_{Air}^{Mean}} + \left(\frac{U_{Roof}A_{Roof} + U_{Side}A_{Side}}{2} \right)^2 C_w v_{wind}^2 \right]^{\frac{1}{2}}$$

where:
• $f_{VentRoofSide}$ is the ventilation rate through both the roof and side vents (m s^{-1});
• $C_{d/w}$ is the discharge/global wind pressure coefficient depending on the greenhouse shape and the use of an outdoor thermal screen;
• $U_{Roof/Side}$ is the control of the roof/side openings ranging in [0,1];
• $A_{Roof/Side}$ is the roof/side opening area m^2;
• $h_{SideRoof}$ is the vertical distance between mid-points of side wall and roof ventilation openings (m);
• T_{Air}^{Mean} is the mean temperature between the indoor and outdoor temperature (K);
• v_{wind} is the speed of wind (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:
$\eta_{InsScr} = \zeta_{InsScr}(2 - \zeta_{InsScr})$.
where:
• η_{InsScr} is the reducing factor;
• ζ_{InsScr} is the screen porosity i.e. the area of holes per unit area of the insect screen.

The greenhouse leakage is also mentioned in this assignment:

$$f_{leakage} = \begin{cases} 0.25 \times c_{leakage},\ v_{wind} < 0.25 \\ v_{lwind} \times c_{leakage},\ v_{wind} \geq 0.25 \end{cases}.$$

where:
• $f_{leakage}$ is the leakage rate depending on wind speed (m s^{-1});
• $c_{leakage}$ is the leakage coefficient depending on the greenhouse type;

The total ventilation rates for roof and side vents are calculated by:

$$f_{VentRoof} = \begin{cases} \eta_{InsScr}\ f''_{VentRoof} + 0.5 f_{leakage},\ \eta_{Side} \geq \eta_{Side_Thr} \\ \eta_{InsScr}\ \left[U_{ThScr} f''_{VentRoof} + (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side} \right] + 0.5 f_{leakage},\ \eta_{Side} < \eta_{Side_Thr} \end{cases}.$$

$$f_{VentSide} = \begin{cases} \eta_{InsScr}\ f''_{VentSide} + 0.5 f_{leakage},\ \eta_{Side} \geq \eta_{Side_Thr} \\ \eta_{InsScr}\ \left[U_{ThScr} f''_{VentSide} + (1 - U_{ThScr}) f_{VentRoofSide} \eta_{Side} \right] + 0.5 f_{leakage},\ \eta_{Side} < \eta_{Side_Thr} \end{cases}.$$

where:
• $f''_{VentSide}$ is $f_{VentRoofSide}$ when $A_{Roof} = 0$;
• η_{Side} is the ratio between the side vents area and total ventilation area;
• η_{Side_Thr} is the threshold value above which no chimney effect is assumed to occur;
• $f''_{VentRoof}$ is calculate by the formula:

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

• h_{Roof} is the vertical dimension of a single ventilation opening (m).

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

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

where:
• $U_{VentForced}$ is the control value of the forced ventilation ranging in [0, 1].
• $\phi_{VentForced}$ is the air flow capacity of the forced ventilation system ($m^3\ s^{-1}$).

The MC_{TopOut} is the net CO_2 flux from the greenhouse to outside through the roof openings is calculated by the formula:
$MC_{TopOut} = f_{VentRoof}(CO_{2Top} - CO_{2Out})$

The net photosynthesis rate
The net photosynthesis rate equals the gross photosoynthesis rate minus photorespiration:
$MC_{AirCan} = M_{CH_2O}\ h_{C_{Buf}}^{MC_{AirCan}}(P - R)$
where:
• M_{CH_2O} is the molar mass of CH_2O (mg $\mu\ mol^{-1}$);
• $h_{C_{Buf}}$ is the inhibition of the photosynthesis rate by saturation of the leaves with carbohydrates, where
$h_{C_{Buf}} = \begin{cases} 0,\ C_{Buf} > C_{Buf}^{Max} \\ 1,\ C_{Buf} \leq C_{Buf}^{Max} \end{cases}$.
• C_{Buf}/C_{Buf}^{Max} is the capacity/maximum of carbonhydrates storage in the canopy buffer (mg $\{CH_2O\}\ m^{-2}$);
• P/R is the photosynthesis/photorespiration rate of the canopy during the photosynthesis process (μmol $\{CO_2\}\ m^{-2}\ s^{-1}$);

Photosynthesis rate at canopy level, P, is described by:
$P = \frac{J\ .\ (CO_{2Stom} - \Gamma)}{4\ .\ (CO_{2Stom} + 2\ \Gamma)}$
where:
• J (μmol$\{e^-\}\ m^{-2}\ s^{-1}$) is the electron transport rate.
• 4 (μmol $\{e^-\}\ \mu\ mol^{-1}\ \{CO_2\}$) is the number of elections per fixed CO_2 molecule.
• CO_{2Stom} (μmol $\{CO_2\}\ mol^{-1}$ {air}) is the CO_2-concentration in the stomata.
• Γ (μmol $\{CO_2\}\ mol^{-1}$ {air}) is the CO_2 compensation point.

The photorespiration, R, is described by:
$R = P\ .\ \frac{\Gamma}{CO_{2Stom}}$

The electron transport rate
The electron transport rate, J, is a function of the potential rate of electron transport and of the absorbed PAR by the canopy:
$J = \frac{J^{POT} + \alpha\ PAR_{Can} - \sqrt{(J_{POT} + \alpha\ PAR_{Can})^2 - 4\Theta.J_{POT}.\alpha\ PAR_{Can}}}{2\Theta}$
where:
• J^{POT} (μmol $\{e^-\}\ m^{-2}\ s^{-1}$) is the potential rate of electron transport.
• PAR_{Can} (μmol {photons} $m^{-2}\ s^{-1}$) is the absorbed PAR.
• α (μmol $\{e^-\}\ \mu\ mol^{-1}$ {photons}) is the conversion factor from photons to electrons, including an efficency term.
• Θ is the degree of curvature of the electron transport rate.

The potential rate of electron transport J^{POT}, depends on temperature (Farquhar et al., 1980):
$J^{POT} = J_{25,Can}^{POT}\ .\ e^{E_j\ .\ \frac{T_{Can,K} - T_{25,K}}{R.T_{Can,K}.T_{25,K}}}\ .\ \frac{1 + e^{\frac{S.T_{25,K} - H}{R.T_{25,K}}}}{1 + e^{\frac{S.T_{Can,K} - H}{R.T_{Can,K}}}}\ \left[\mu\text{mol}\ \{e^-\}\ m^{-2}\ s^{-1}\right]$

where:
• $J_{25,Can}^{MAX}$ (μmol $\{e^-\}$ m^{-2} s^{-1}) is the maximum rate of electron transport at 25^oC 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^oC.
• R_g (J mol^{-1} K^{-1}) is the molar gas constant.
• S (J mol^{-1} K^{-1}) is the entropy term.
• H (J mol^{-1}) is the deactivation energy.

The maximum rate of electron transport at 25^oC for the canopy is calculated by:
$J_{25,Can}^{MAX} = LAI \, . \, J_{25,Leaf}^{MAX}$ $\left[\mu\text{mol } \{e^-\}\ m^{-2}\ s^{-1}\right]$
where $J_{25,Can}^{MAX}$ (μ $\{e^-\}$ m^{-2} {leaf} s^{-1}) is the maximum rate of electron transport for the leaf at 25^oC.

The absorbed PAR by the canopy
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:
$PAR_{Can} = PAR_{GhCan} + PAR_{FlrCan}$ $\left[\mu\text{mol } \{\text{photons}\}\ m^{-2}\ s^{-1}\right]$

The PAR which is directly absorbed by the canopy is described by a negative exponential decay of light with LAI in a homogeneous crop:
$PAR_{GhCan} = PAR_{Gh} \, . \, (1 - p_{Can}) \, . \, (1 - exp\{-K_1 \, . \, LAI\})$ $\left[\mu\text{mol } \{\text{photons}\}\ m^{-2}\ s^{-1}\right]$
where:
• PAR_{Gh} $\left(\mu\text{mol } \{\text{photons}\}\ m^{-2}\ s^{-1}\right)$ is the PAR above the canopy.
• p_{Can} (W) is the reflection coefficient of the canopy for PAR.
• K_1 is the extinction coefficient of the canopy for PAR.

The PAR above the canopy is described by:
$PAR_{Gh} = \tau_{Gh} \, . \, \eta_{Glob_PAR} \, . \, I_{Glob}$ $\left[\mu\text{mol } \{\text{photons}\}\ m^{-2}\ s^{-1}\right]$
where:
• τ_{Gh} is the light transmission of the greenhouse cover.
• η_{Glob_PAR} $\left(\mu\text{mol } \{\text{photons}\}\ J^{-1}\right]$ is a conversion factor from global radiation to PAR.
• I_{Glob} (W m^{-2})is the outside global radiation.

Absorption of PAR reflected by the greenhouse floor is described by:
$PAR_{FlrCan} = p_{Flr} \, . \, PAR_{Gh} \, . \, (1 - p_{Can}) \, . \, exp(-K_1 \, . \, LAI) \, . \, (1 - exp(-K_2 \, . \, LAI))$ $\left[\mu\text{mol } \{\text{photons}\}\ m^{-2}\ s^{-1}\right]$
where:
• p_{Flr} is the reflection coefficient of the greenhouse floor.
• 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.

CO_2-relationships in the photosynthetic issue

The CO_2-concentration inside the stomata, CO_{2Stom} depends on the stomatal and mesophyl conductance, boundary layer resistance, the photosynthesis rate and the difference between the CO_2-concentration in the stomata and the CO_2-concentration of the greenhouse air. However, the CO_2-concentration in the stomata is assumed to be a fixed fraction of the CO_2-concentration

in the greenhouse air:
$CO2_{Stom} = \eta_{CO2Air_Stom} \cdot CO2_{Air}$ $[\mu\text{mol } \{CO_2\}\ mol^{-1}\ \{\text{air}\}]$
where η_{CO2Air_Stom} is conversion factor from the CO_2-concentration of the greenhouse air, $CO2_{Air}$, to the CO_2-concentration in the stomata.

The CO_2 compensation point (Γ) affects the leaf photosynthesis rate and depends on temperature:
$\Gamma = \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}} c_\Gamma\ T_{Can} + 20c_\Gamma \left(1 - \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}}\right)$ $[\mu\text{mol } \{CO_2\}\ mol^{-1}\ \{\text{air}\}]$
where:
• c_Γ $(\mu\text{mol } \{CO_2\}\ mol^{-1}\ \{\text{air}\}\ K^{-1})$ determines the effect of canopy temperature on the CO_2 compensation point.

3.1.2 2.b

Write programs that calculate the net CO_2 flux from one place to an other place using the formulas (3)-(7), (9)-(19). Each formula is a function with input parameters being the coefficients and variables involved in each respective formula. Then write a program that returns the right side of the system of (1) and (2) divided by $cap_{CO_{2Air}}$ and $cap_{CO_{2Top}}$ respectively and named this function dx. Present carefully in the report.

Formula 3: The amount of CO_2 going from the heater into the greenhouse air.

```
def MC_BlowAir(n_HeatCO2, U_Blow, P_Blow, A_Flr):
  return (n_HeatCO2 * U_Blow * P_Blow) / A_Flr
```

Formula 4: TThe amount of CO_2 that is pumped into the greenhouse by the third party that supplies CO_2.

```
def MC_ExtAir(U_ExtCO2, phi_ExtCO2, A_Flr):
  return (U_ExtCO2 * phi_ExtCO2) / A_Flr
```

Formula 5: TThe amount of CO_2 that is pumped into the greenhouse by the pad system.

```
def MC_PadAir(CO2_Air, U_Pad, phi_Pad, A_Flr, CO2_Out):
  return (U_Pad * phi_Pad * (CO2_Out - CO2_Air)) / A_Flr
```

Formula 6: The net flux of CO_2 from the lower compartment to the upper compartment.

```
def MC_AirTop(CO2_Air, CO2_Top, U_ThScr, K_ThScr, T_Air, T_Top, P_MeanAir, P_Air,
    P_Top, g):
  return f_ThScr(U_ThScr, K_ThScr, T_Air, T_Top, P_MeanAir, P_Air, P_Top, g) * (
    CO2_Air - CO2_Top)
```

Formula 7: The airflow rate through the thermal screen.

```
def f_ThScr(U_ThScr, K_ThScr, T_Air, T_Top, P_MeanAir, P_Air, P_Top, g):
  return U_ThScr * K_ThScr * (abs(T_Top - T_Air) ** (2 / 3)) + (1 - U_ThScr) * (g
    * (1 - U_ThScr) * abs(P_Air - P_Top) / (2 * P_MeanAir)) ** (1 / 2)
```

Formula 9: the net CO_2 flux from the inside to the outside of the greenhouse.

```
def MC_AirOut(CO2_Air, CO2_Out, n_Side, n_SideTHr, U_VentForced, phi_VentForced,
    U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, A_Flr, U_Roof, U_Side, A_Roof,
    A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir, Cw, g):
  return (f_VentForced(U_VentForced, phi_VentForced, A_Flr, phi_InsScr) +
    f_VentSide(n_Side, n_SideTHr, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd,
    A_Flr, U_Roof, U_Side, A_Roof, A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir, Cw
    , g)) * (CO2_Air - CO2_Out)
```

Formula 10: General formula used to set the formula for $f_{VentSide}$.

```python
def f_VentRoofSide(Cd, A_Flr, U_Roof, U_Side, A_Roof, A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir, Cw, v_Wind, g):
  i = ((U_Roof * A_Side * U_Side * A_Roof) ** 2) * 2 * g * h_SideRoof * (T_Air - T_Out) / (((U_Roof * A_Roof) ** 2 + (U_Side * A_Side) ** 2) * T_MeanAir)
  j = (((U_Roof * A_Roof + U_Side * A_Side) ** 2) / 4) * Cw * (v_Wind ** 2)
  return (Cd / A_Flr) * (i + j) ** (1 / 2)
```

Formula 11:

```python
def n_InsScr(phi_InsScr):
  return phi_InsScr * (2 - phi_InsScr)
```

Formula 12: leakage rate.

```python
def f_leakage(v_Wind, c_leakage):
  if v_Wind < 0.25:
    return 0.25 * c_leakage
  else:
    return v_Wind * c_leakage
```

Formula 13:

```python
def f_VentSide(n_Side, n_SideTHr, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, A_Flr, U_Roof, U_Side, A_Roof, A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir, Cw, g):
  if n_Side < n_SideTHr:
    return n_InsScr(phi_InsScr) * ((U_ThScr * (Cd * U_Side * A_Side * v_Wind * math.sqrt(Cw)) / (2 * A_Flr)) + (1 - U_ThScr) * n_Side * f_VentRoofSide(Cd, A_Flr, U_Roof, U_Side, A_Roof, A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir, Cw, v_Wind, g)) + 0.5 * f_leakage(v_Wind, c_leakage)
  else:
    return n_InsScr(phi_InsScr) * (U_ThScr * (Cd * U_Side * A_Side * v_Wind * math.sqrt(Cw)) / (2 * A_Flr)) + 0.5 * f_leakage(v_Wind, c_leakage)
```

Formula 14:

```python
def f_VentForced(phi_InsScr, U_VentForced, phi_VentForced, A_Flr):
  return n_InsScr(phi_InsScr) * U_VentForced * phi_VentForced / A_Flr
```

Formula 15: The net CO2 flux from the greenhouse to outside the greenhouse through the roof openings.

```python
def MC_TopOut(CO2_Top, CO2_Out, n_Roof, n_Side, n_RoofThr, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir, Cw, g):
  return f_VentRoof(n_Roof, n_Side, n_RoofThr, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir, Cw, g) * (CO2_Top - CO2_Out)
```

Formula 16: The flux rate through the roof openings. through the roof openings.

```python
def f_VentRoof(n_Roof, n_Side, n_RoofThr, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir, Cw, g):
  if n_Roof < n_RoofThr:
    return n_InsScr(phi_InsScr) * (U_ThScr * f_Prime_VentRoof(Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir, Cw, v_Wind, g) + (1 - U_ThScr) * n_Side * f_Prime_VentRoof(Cd, U_Roof, A_Roof, A_Flr, h_Roof, T_Air, T_Out, T_MeanAir, Cw, v_Wind, g)) + 0.5 * f_leakage(v_Wind, c_leakage)
  else:
    return n_InsScr(phi_InsScr) * f_Prime_VentRoof(Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir, Cw, v_Wind, g) + 0.5 * f_leakage(v_Wind, c_leakage)
```

Formula 17: The flux rate through the roof openings. through the roof openings.

```python
def f_Prime_VentRoof(Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out, T_MeanAir,
    Cw, v_Wind, g):

  return Cd * U_Roof * A_Roof * ((g * h_Vent * (T_Air - T_Out) / (2 * T_MeanAir) +
     Cw * v_Wind ** 2) ** (1 / 2)) / (2 * A_Flr)
```

Formula 18: The amount of CO2 that is absorbed into the leaves due to photosynthesis

```python
def MC_AirCan(CO2_Air, n_CO2_Air_Stom, LAI, T_Air, c_gamma, Theta_Gh, n_Glob_PAR,
    I_Glob, rho_Flr, rho_Can, K1, K2, J_Max25Can, T_Can, T_25K, alpha, phi, CBuf,
    C_MaxBuf, Ej, S, H, R, T_CanK):
  CO2_Stom = n_CO2_Air_Stom * CO2_Air
  gamma = (1 / LAI) * c_gamma * T_Can + 20 * c_gamma * (1 - (1 / LAI))
  PAR_Gh = Theta_Gh * n_Glob_PAR * I_Glob
  PAR_FlrCan = rho_Flr * PAR_Gh * (1 - rho_Can) * math.exp((-K1) * LAI) * (1 -
    math.exp((-K2) * LAI))
  PAR_GhCan = PAR_Gh * (1 - rho_Can) * (1 - math.exp((-K1) * LAI))
  PAR_Can = PAR_GhCan + PAR_FlrCan
  J_Pot = J_Max25Can * math.exp(Ej * ((T_CanK - T_25K) / (R * T_CanK * T_25K))) *
    (1 + math.exp((S * T_25K - H) / (R * T_25K))) / (1 + math.exp((S * T_CanK - H)
     / (R * T_CanK)))
  J = (J_Pot + alpha * PAR_Can - math.sqrt((J_Pot + alpha * PAR_Can) ** 2 - 4 *
    phi * J_Pot * alpha *PAR_Can)) / (2 * phi)
  P = (J * (CO2_Stom - gamma)) / (4 * (CO2_Stom + 2 * gamma))
  R = P * (gamma / CO2_Stom)
  return 30 * h_C_Buf(CBuf, C_MaxBuf) * (P - R)
```

Formula 19: $h_{C_{Buf}}$ coefficent.

```python
def h_C_Buf(CBuf, C_MaxBuf):
  if CBuf > C_MaxBuf:
    return 0
  else:
    return 1
```

The rate of change of CO2 concentration in lower and upper compartments:

```python
def dx(CO2_Air, CO2_Top):
  MCBlowAir = MC_BlowAir(n_HeatCO2, U_Blow, P_Blow, A_Flr)
  MCExtAir = MC_ExtAir(U_ExtCO2, phi_ExtCO2, A_Flr)
  MCPadAir = MC_PadAir(CO2_Air, U_Pad, phi_Pad, A_Flr, CO2_Out)
  MCAirCan = MC_AirCan(CO2_Air, n_CO2_Air_Stom, LAI, T_Air, c_gamma, Theta_Gh,
    n_Glob_PAR, I_Glob, rho_Flr, rho_Can, K1, K2, J_Max25Can, T_Can, T_25K, alpha,
     phi, CBuf, C_MaxBuf, Ej, S, H, R, T_CanK)
  MCAirOut = MC_AirOut(CO2_Air, CO2_Out, n_Side, n_SideTHr, U_VentForced,
    phi_VentForced, U_ThScr, phi_InsScr, v_Wind, c_leakage, Cd, A_Flr, U_Roof,
    U_Side, A_Roof, A_Side, h_SideRoof, T_Air, T_Out, T_MeanAir5, Cw, g)
  MCAirTop = MC_AirTop(CO2_Air, CO2_Top, U_ThScr, K_ThScr, T_Air, T_Top, P_MeanAir
    , P_Air, P_Top=, g)
  dx_CO2_Air = (MCBlowAir + MCExtAir + MCPadAir - MCAirCan - MCAirTop - MCAirOut)
    / cap_CO2_Air

  MCAirTop = MC_AirTop(CO2_Air, CO2_Top, U_ThScr, K_ThScr, T_Air, T_Top, P_MeanAir
    , P_Air, P_Top, g)
  MCTopOut = MC_TopOut(CO2_Top, CO2_Out, n_Roof, n_Side, n_RoofThr, U_ThScr,
    phi_InsScr, v_Wind, c_leakage, Cd, U_Roof, A_Roof, A_Flr, h_Vent, T_Air, T_Out
    , T_MeanAir, Cw, g)
  dx_CO2_Top = (MCAirTop - MCTopOut) / cap_CO2_Top
  return dx_CO2_Air, dx_CO2_Top
```

3.2 Exercise 3:

Assuming constant temperature difference and constant air density difference, study from [Van11] and related citations to find specific and reasonable values for each input parameter of the function dx including the difference in temperature and in air density except for the variables CO2 Air and CO2 T op. Noting that these values need to match the units considered in the report. Check if the programs work properly with the values found and specific CO2 Air and CO2 T op values. Present details in the report.

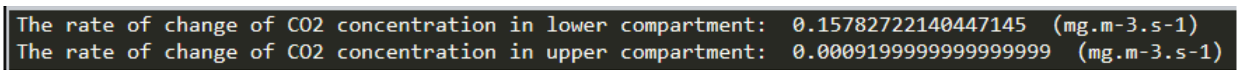
We choose region The Netherlands on Van11 and data on Cucumber/MicrosoftResearch

We choose constant temperature difference equals 1.

Name	Value	Unit	Source/Reason
$\eta_{HeatCO2}$	0.057	$mg\{CO_2\}J^{-1}$	Van11
U_{blow}	0	-	because P_blow is x in Netherlands
P_{blow}	0	W	Van11
A_{Flr}	1.4 $\cdot 10^4$	m^2	Van11
U_{ExtCO_2}	1	-	On title
ϕ_{ExtCO_2}	$7.2 \cdot 10^4$	mgs^{-1}	Van11
U_{Pad}	0	-	Because ϕ_Pad is x in Netherlands
ϕ_{Pad}	x	$m^3 \cdot s^{-1}$	Van11
CO_{2out}	668	mgm^{-3}	Van11
U_{ThScr}	0.95	-	On dataset
K_{ThScr}	$0.05 \cdot 10^{-3}$	$m^3m^{-2}K^{-0.66}s^{-1}$	Van11
$T_{air} - T_{top}$	1	K	Constant temperature difference
ρ_{Air}	$\frac{353}{291.85}$	$kg \cdot m^{-3}$	$\rho_{Air} = \frac{353}{T_{air}} with T_{Air} = 291.85K$
ρ_{Top}	$\frac{353}{290.85}$	$kg \cdot m^{-3}$	$\rho_{Air} = \frac{353}{T_{top}} with T_{Air} = 290.85K$
${\rho_{air}}^{mean}$	1.21	$kg \cdot m^{-3}$	${\rho_{air}}^{mean} = \frac{\rho_{Air}+\rho_{Top}}{2}$
η_{Side}	1	-	Random
η_{Side_Thr}	0.9	-	η_{Roof_Thr} in Van11
ζ_{InsScr}	1	-	Van11
U_{Roof}	0	-	$\rho_{Air} = \frac{ventlee+ventside}{2}$ with ventlee = ventside = 0
U_{Side}	0	-	Random
A_{Roof}	$0.13 \cdot 10^4$	m^2	Van11: $\frac{A_{Roof}}{A_{Flr}} = 0.1$
A_{Side}	0	m^2	Van11: $\frac{A_{Side}}{A_{Flr}} = 0$
g	9.81	ms^{-2}	Van11
$h_{SideRoof}$	x	m	Van11
T_{Air} - T_{Out}	1	K	Constant temperature difference
${T_{Air}}^{mean}$	291.35	K	${T_{Air}}^{mean} = \frac{T_{Air}+T_{Out}}{2}$ with $T_{Air} = 291.85 and T_{Out} = 290.85$
C_w	0.09	-	Van11
v_{wind}	4.6	ms^{-1}	Van11
$U_{VentForced}$	0	-	No $\phi_{VentForced}$ in the dataset we choose
$\phi_{VentForced}$	x	m^3s^{-1}	Van11
η_{Roof}	1	-	Random
η_{Roof_Thr}	0.9	-	Van11
C_d	0.75	-	Van11
h_{Vent}	0.68	m	Van11
M_{CH_2O}	30	g/mol	
C_{Buf}^{Max}	$20 \cdot 10^3$	mg m^{-2}	Van11
C_{Buf}	$15 \cdot 10^3$	mg m^{-2}	Random
α	0.385	$\mu mol\{e-\}\mu mol^{-1}\{photons\}$	Van11
Θ	0.7	-	Van11
$J_{25,Can}^{Max}$	210	$\mu mol\{e-\}m^2\{leaf\}s^{-1}$	Van11
E_j	$37 \cdot 10^3$	$Jmol^{-1}$	Van11

Name	Value	Unit	Source/Reason
$T_{Can,K}$	292.85	K	$T_{Can,K} = T_{Air} + 1$
$T_{25,K}$	298.15	K	Van11
S	710	$Jmol^{-1}$	Van11
H	$22 \cdot 10^4$	$Jmol^{-1}$	Van11
R	8.314	$Jmol^{-1}$	Van11
ρ_{Can}	0.07	-	Van11
K_1	0.7	-	Van11
LAI	2.5	m^2m^{-2}	Van11
τ_{Gh}	0.78	-	Van11
η_{Glob_PAR}	2.3	$\mu mol\{photons\}J^{-1}$	Van11
I_{Glob}	8.5	Wm^{-1}	Van11
ρ_{Flr}	0.5	-	Van11
K_2	0.7	-	Van11
η_{CO2Air_Stom}	0.67	$\mu mol\{CO_2\}mol^{-1}\{air\}$	Van11
c_Γ	1.7	$\mu mol\{CO_2\}mol^{-1}\{air\}K^{-1}$	Van11
T_{Can}	292.85	K	$T_{Can,K} = T_{Air} + 1$
cap_{CO2Air}	3	m	Random
cap_{CO2Top}	1.5	m	Random

With the specific data as above we can calculate as follows:


```
The rate of change of CO2 concentration in lower compartment:  0.15782722140447145  (mg.m-3.s-1)
The rate of change of CO2 concentration in upper compartment:  0.0009199999999999999  (mg.m-3.s-1)
```

3.3 Exercise 4:

3.3.1 4.a

Study explicit Euler and Runge–Kutta of order 4 algorithms for solving first-order differential equations (see references [ESG93; EG96]). Write two programs performing the two algorithms and named them euler and rk4 respectively with inputs: a callable function as dx, the values at time t of the two variables CO2 Air and CO2 T op, the time step h. These solvers return the approximate values of CO2 Air and CO2 T op at time t+h. Present details in the report.

• With the Runge–Kutta of order 4 algorithm found in problem 1. We implement the function rk4.The input parameters: callable, h, time_start, time_end, CO2_Air, CO2_Top, they are a function callable (dx), h step, initial time, end time and CO2 concentration above and below the greenhouse at the inception respectively.

```
def rk4(callable, CO2_Air, CO2_Top, h):
  if callable == True:
    temp = dx(CO2_Air, CO2_Top)
    K1_Air = h * temp[0]
    K1_Top = h * temp[1]

    temp = dx(CO2_Air + K1_Air / 2, CO2_Top + K1_Top / 2)
    K2_Air = h * temp[0]
    K2_Top = h * temp[1]

    temp = dx(CO2_Air + K2_Air / 2, CO2_Top  + K2_Top / 2)
    K3_Air = h * temp[0]
    K3_Top = h * temp[1]

    temp = dx(CO2_Air + K3_Air, CO2_Top + K3_Top)
    K4_Air = h * temp[0]
    K4_Top = h * temp[1]

    CO2_Air = CO2_Air + (K1_Air + 2 * K2_Air + 2 * K3_Air + K4_Air) * (1 / 6)
    CO2_Top = CO2_Top + (K1_Top + 2 * K2_Top + 2 * K3_Top + K4_Top) * (1 / 6)
    return CO2_Air, CO2_Top
```

- *With Euler algorithm.*

```
def euler(callable, CO2_Air, CO2_Top, h):
  if callable == True:

    temp = dx(CO2_Air, CO2_Top)

    CO2_Air = CO2_Air + h * temp[0]
    CO2_Top = CO2_Top + h * temp[1]
    return CO2_Air, CO2_Top
```

3.3.2 4.b

Select specific values of CO2 Air and CO2 Top at time t from the data set as initial values to run the solvers and find the approximate values of CO2 Air and CO2 T op in the next5 minutes, 10 minutes, 20 minutes, ... and calculate the difference of the result from the actual data. Comment on the accuracy of the model and present details in the report. With the following parameters in lesson 3 and the input parameters of the rk4 and euler functions:

• CO2_Top and CO2_Air : 673 ($mgm-3$).
• h : 10 (s)
• callable: callable(dx)

Runge–Kutta of order 4 algorithm

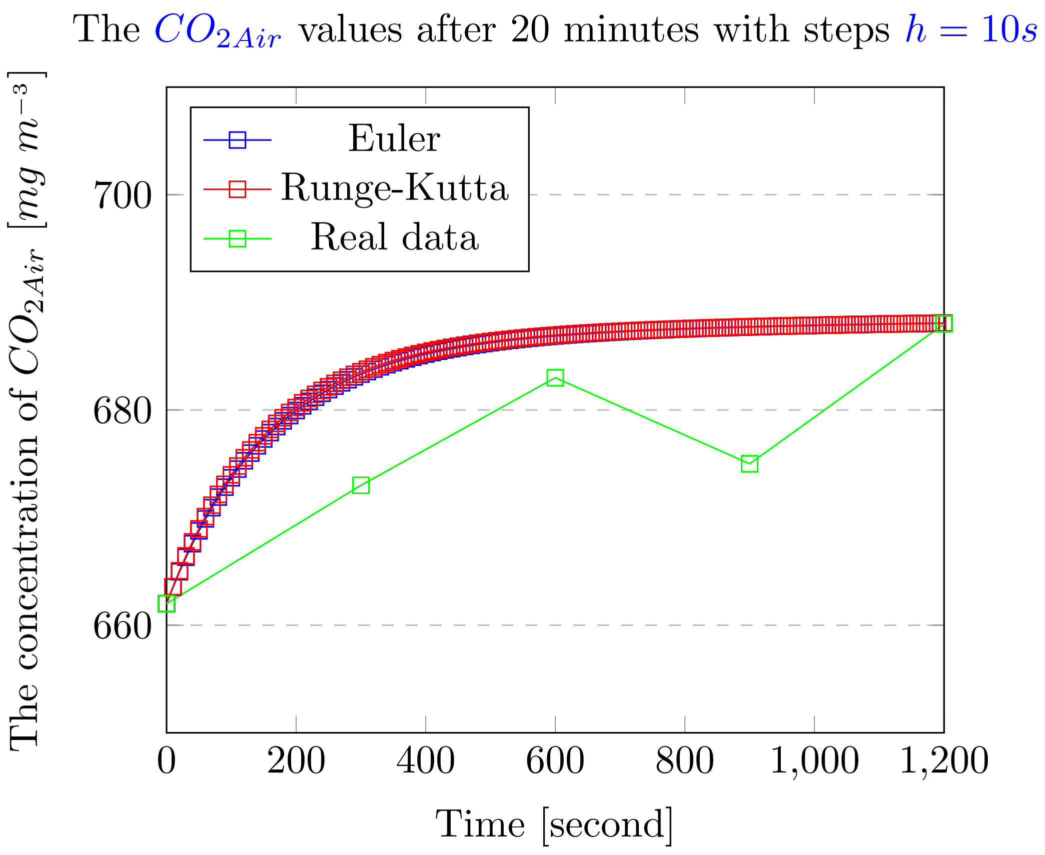
```
t = 0
h = 10
end_time = 1200
CO2_Air = 673
CO2_Top = 673
while t < end_time:
  res = rk4(callable(dx), h, CO2_Air, CO2_Top)
  CO2_Air = res[0]
  CO2_Top = res[1]
  t += h
  print(res[0])
```

Euler algorithm

```
t = 0
h = 10
end_time = 1200
CO2_Air = 673
CO2_Top = 673
while t < end_time:
  res = euler(callable(dx), h, CO2_Air, CO2_Top)
  CO2_Air = res[0]
  CO2_Top = res[1]
  t += h
  print(res[0])
```

Time (minute)	Runge-Kutta data	Euler data	Real data
0	662	662	662
5	683.33	683.55	673
10	686.88	686.94	683
15	687.71	687.73	675
20	688.07	688.07	688

The difference values of CO_{2Air} concentration between Euler and Runge-Kutta method after a period of 20 minutes with a step $h = 10s$



Comment:

• The values found are increasing toward the horizontal asymptote line. Because according to the problem, when the environmental-affected variables are set to constants, the formula show that the concentration CO2Air is affected by the function (CO2Air - CO2Out), where the absolute values of this functions decrease over time, so the value of CO2Air will gradually increase to a constant value, this time-dependent variable function is also called a continuous dynamic system (flow)

• In fact, variables values are always affected by the environment, there is no constant (e.g when the CO2 concentration changes, the air pressure as well as the temperature will change

according to the condition of the environment). This makes the values of CO2Air always fluctuate irregularly known as discrete dynamic system (maps)
• Since step h is rather small (10s) compared to the execution time of the function, the outputs of the function rk4 and the euler functions are not different too much.

3.4 Exercise 5:

3.4.1 5.2a

Restate the model for the Vapour Pessure in greenhouses and write to the report.

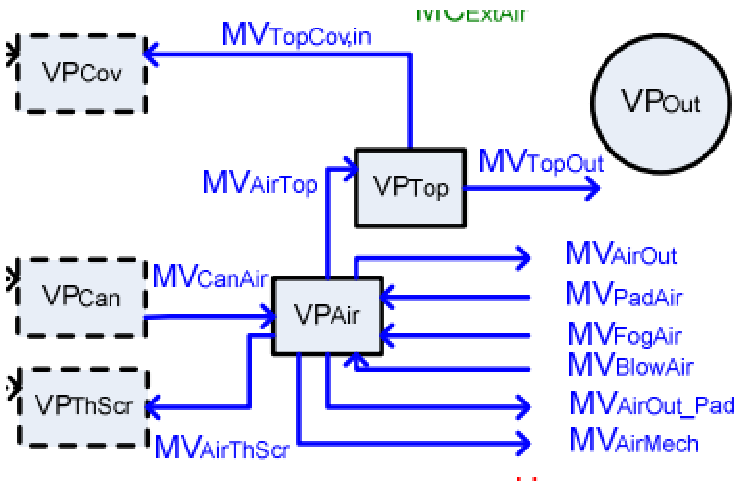


Figure 3: Visualization of Euler's method

Vapour pressure of the greenhouse air and the air in the top compartment **The vapour pressure of the greenhouse air VP_{Air} is described by:**

$$(1): cap_{VP_{Air}}\dot{VP}_{Air} = MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BlowAir} - MV_{AirThScr} - MV_{AirTop} - MV_{AirOut} - MV_{AirOut_Pad} - MV_{AirMech}\ [kgm^{-2}s^{-1}]$$

where:
• $cap_{VP_{Air}}$ is the capacity of the air to store water vapour.
Vapour is exchanged between the air and surrounding elements, i.e.
• MV_{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 outdoor air due to 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:

$$(2) : cap_{VP_{Top}}\dot{VP}_{Top} = MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut} \; [kgm^{-2}s^{-1}]$$

where:

• $cap_{VP_{Top}}$ is the capacity of the top compartment to store water vapour.
Vapour is exchanged between the air and surrounding elements, i.e.

• $MV_{TopCov,in}$ is the vapour exchange between the top compartment and the internal cover layer.

• MV_{TopOut} is the vapour exchange between the top compartment and the outside air.

Vapour Fluxes The vapour flux from location to object:

$$(3) : MV_{12} = \frac{1}{1+exp(S_{MV_{12}}(VP_1 - VP_2)}6.4 * 10^{-9}HEC_{12}(VP_1 - VP_2) \; [kgm^{-2}s^{-1}]$$

where:

• MV_{12} is the vapour flux from location 1 to object 2.

• $S_{MV_{12}} = -0.1$ is the slope of the differentiable switch function for vapour pressure differences.

• HEC_{12} is the heat exchange coefficient between the air of location 1 to object 2 .

$$(4) : HEC_{TopCov,in} = c_{HECin}(T_{Top} - T_{Cov,in})^{0.33} \text{ with } c_{HECin} = 1.86$$

$$(5) : HEC_{AirThScr} = 1.7U_{ThScr}|T_{air} - T_{ThScr}|^{0.33}$$

• VP_1 is the vapour pressure of the air of location 1 .

• VP_2 is the vapour pressure of the air of object 2 .

$MV_{AirThScr}$, $MV_{TopCov,in}$ and $MV_{AirMech}$ are described analogously to Equation (3)
The vapour flux from location to location:

$$(6) : MV_{12} = \frac{M_{water}}{R}f_{12}(\frac{VP_1}{T_1+273.15} - \frac{VP_2}{T_2+273.15}) \; [kgm^{-2}s^{-1}]$$

where:

• $M_{water} = 18$ is the vapour flux from location 1 to object 2.

• $R = 8.314 * 10^3$ is the Molar gas constant.

• f_{12} is the air flux from location 1 to location 2.

• T_1 is the temperature of the air of location 1 .

• T_2 is the temperature of the air of location 2 .

MV_{AirTop}, MV_{AirOut} and MV_{TopOut} are described analogously to Equation (6) whereby their accompanying air fluxes are f_{ThScr} (the flux through the thermal screen), $f_{VentSide} + f_{VentForced}$ (the flux due to natural ventilation through the side windows or forced ventilation) and $f_{VentRoof}$ (flux due to roof ventilation) respectively

• $f_{ThScr} = U_{ThScr}K_{ThScr}|T_{Air} - T_{Out}|^{0.66} + \frac{1-U_{ThScr}}{p_{air}^{mean}}(0.5p_{air}^{mean}(1 - U_{ThScr})g|p_{Air} - p_{Out}|)^{0.5}$ with $g = 9.81$ and $K_{ThScr} = 0.05 * 10^{-3}$ and $p_{Air} = \frac{353}{T_{Air}}$ and $p_{Out} = \frac{353}{T_{Out}}$

• $f_{Ventside} = \eta_{InsScr}f''_{Ventside} + 0.5f_{leakage}$ with $\eta_{InsScr} = 1$ and $f''_{Ventside}$ is $f_{VentRoofSide}$ when $A_{Roof} = 0$ and $f_{leakage} = v_{wind} * 10^{-4}$ (We use this formula beacause $\eta_{Roof} = 1.0 > \eta_{Roof_Thr} = 0.9$)

$$f_{VentRoofSide} = \frac{C_d}{A_{Flr}}$$

$$\left[\frac{U_{Roof}^2 \; U_{Side}^2A_{Roof}^2A_{Side}^2}{U_{Roof}^2A_{Roof}^2 \; + U_{Side}^2A_{Side}^2} . \frac{2gh_{SideRoof}(T_{Air} \text{ - } T_{Out})}{T_{Air}^{Mean}} + \left(\frac{U_{Roof}A_{Roof} \; + U_{Side}A_{Side}}{2}\right)^2 C_w v_{wind}^2\right]^{\frac{1}{2}} \text{ with}$$

$$C_d = 0.75, C_w = 0.09, \; A_{Flr} = 1.4 * 10^4, \; A_{roof} = 1.4 * 10^3, A_{side}, h_{sideroof} = 0.$$

• $f_{Ventroof} = \eta_{InsScr} f''_{Ventroof} + 0.5 f_{leakage}$ with $\eta_{InsScr} = 1$ and $f_{leakage} = v_{wind} * 10^{-4}$ (We use this formula beacause $\eta_{Roof} = 1.0 > \eta_{Roof_Thr} = 0.9$)

$$f''_{Ventroof} = \frac{C_d U_{Roof} A_{Roof}}{2A_{Flr}} \left(\frac{g h_{Roof}(T_{Air} - T_{Out})}{2T^{Mean}_{Air}} + C_w v^2_{wind}\right)^{\frac{1}{2}} \text{ with } C_d = 0.75, C_w = 0.09,$$
$$A_{Flr} = 1.3 * 10^4,\ A_{Roof} = 1.4 * 10^3,\ h_{Roof} = 1.6$$

• $f_{Ventforced} = 0$ because we can not find the values of related coefficients.
Canopy transpiration The canopy transpiration is described by:

$$(7) : MV_{CanAir} = VEC_{CanAir}(VP_{Can} - VP_{Air})$$

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

$$VP_{Can} = 610.78 * exp(\frac{T_{Can}}{T_{Can} + 238.3} * 17.2694) \text{ with } T_{Can} = T_{Air} + 1$$

• $VP_{Air} = RelativeHumidity * VP_{Can}$
The vapour transfer coefficient of the canopy transpiration can be calculated by:

$$(8) : VEC_{CanAir} = \frac{2 p_{Air} c_{p,Air} LAI}{\Delta h \gamma (r_b + r_s)}$$

where:
• $p_{Air} = \frac{353}{T_{Air}}$ is the density of the greenhouse
• $c_{p,Air} = 10^3$ is the specific heat capacity of the air.
• $\Delta h = 2.45 * 10^6$ is the latent heat of evaporation.
• $\gamma = 65.8$ is the psychometric constant
• $r_b = 275$ is the boundary layer resistance of the canopy for vapour transport.
• $r_s = r_{s,min} = 82$ is the stomatal resistance of the canopy for vapour transport.
• $LAI = 2$ is the leaf area index.
Direct air heater The vapour flux from the heat blower to the greenhouse air is proportional to the heat flux:

$$(9) : MV_{BlowAir} = \frac{\eta_{HeatVap} U_{Blow} P_{Blow}}{A_{Flr}}$$

where:
• $\eta_{HeatVap}$ is the amount of vapour which is released when 1 Joule of sensible energy is produced by the direct air heater
• $U_{Blow} = 0$ **Pad and fan cooling The vapour flux from the pad and fan to the greenhouse air is described by:**

$$(10) : MV_{PadAir} = p_{Air} f_{Air} (\eta_{Pad}(x_{Pad} - x_{Out}) + x_{Out})$$

where:
• $x_{Pad} = x_{Out} = \eta_{Pad} = 0$
• $f_{Pad} = \frac{U_{Pad} \phi_{Pad}}{A_{Flr}}$ is the ventilation flux due to the pad and fan system.
The sensible heat flux from the greenhouse air to the outside air $H_{AirOut_{Pad}}$, when using the pad and fan system is described by:

$$(10) : MV_{AirOut\_Pad} = f_{Pad} \frac{M_{water}}{R} \left(\frac{VP_{Air}}{T_{Air} + 273.15}\right)$$

where:
• $f_{Pad} = \frac{U_{Pad} \phi_{Pad}}{A_{Flr}}$ with $U_{Pad} = 0$
Fogging The latent heat flux from the greenhouse air depends on the vapour flux from the fogging system to the greenhouse air which is described by:

$$(11) : MV_{FogAir} = \frac{U_{Fog}\eta_{Fog}}{A_{Flr}}$$

where:
• $U_{fog} = 0$ is the control valve of the fogging system
• η_{Fog} is the capacity of the fogging system.

3.4.2 5.2b

Write programs that calculate the net vapor pressure from one place to an other place). Each formula is a function with input parameters being the coefficients and variables involved in each respective formula. Then write a program that returns the right side of the system of (1) and (2) divided by $cap_{VP_{Air}}$ and $cap_{VP_{Top}}$ respectively and named this function dx. Present carefully in the report.

The vapour flux from location to object:

```
def MV_location_to_object(HEC, VP_1, VP_2):
      MV_1_2 =((6.4 * 10**-9 * HEC*(VP_1-VP_2))/(1+math.exp(-0.1*(VP_1-VP_2))))
      return MV_1_2
```

• **To find** $HEC_{TopCov,in}$

```
def HEC_Top_Cov_in(T_Top, T_Cov_in):
      return 1.86*(T_Top - T_Cov_in)**0.33
```

• **To find** $HEC_{AirThScr}$

```
def HEC_Air_ThScr(T_Air, T_ThScr):
    U_ThScr = 1
    return 1.7*U_ThScr*abs(T_Air - T_ThScr)**0.33
```

The vapour flux from location to location:

```
def MV_location_to_location(f, T_1, T_2, VP_1, VP_2):
      MV_1_2 = 18 * (1/(8.314*(10**3))) * f * (((VP_1)/(T_1+273.15)) - ((VP_2)/(T_2+273.15)))
      return MV_1_2
```

• **To find** f_{ThScr}

```
def f_ThScr(U_ThScr, K_ThScr, T_Air, T_Out, p_mean_air, p_Air, p_Out, g):
      return U_ThScr * K_ThScr * abs(T_Air-T_Out)**0.66 + (1-U_ThScr)*((0.5*p_mean_air*(1-U_ThScr)*g*abs(p_Air-p_Out))**0.5)/p_mean_air
```

• **To find** $f_{VentSide}$

```
def f_Vent_Side(n_Ins_Scr, f_prime_Vent_side, f_leakage):
      return n_Ins_Scr*f_prime_Vent_side + 0.5*f_leakage
```

• **To find** $f_{VentRoofSide}$

```
def f_prime_Vent_Roof_side(C_d, A_Flr, U_roof, U_side, A_roof, A_side, g, h_roof_side, T_Air, T_Out, T_Mean_Air, C_w, v_wind):
      return C_d*(((U_roof**2 * U_side**2 * A_roof**2 * A_side**2 * 2 * g * h_roof_side * (T_Air - T_Out))/((U_roof**2 * A_roof**2 + U_side**2 +A_side**2)*T_Mean_Air)
              + (C_w*v_wind**2 *((U_roof*A_roof+U_side*A_side)/2)**2)))**0.5 / A_Flr
```

• **To find** $f_{VentRoof}$

```
def f_Vent_Roof(n_Ins_Scr, f_prime_Vent_roof, f_leakage):
      return n_Ins_Scr * f_prime_Vent_roof + 0.5 * f_leakage
```

• **To find** $f_{leakage}$

```
  def Cal_f_leakage(Windspeed): #c_leakage = 10^-4
        if (Windspeed < 0.25): return 0.25*10**-4
        else: return Windspeed*10**-4
```

Canopy transpiration:

```
def Cal_MV_Can_Air(VEC, VP_Can, VP_Air):
    return VEC*(VP_Can-VP_Air)
```

• **To find** VEC_{CanAir}

```
  def VEC_Can_Air(p_Air, c_p_Air, LAI, delta_h, gamma, r_b, r_s):
        return (2 * p_Air * c_p_Air * LAI)/(delta_h*gamma*(r_s+r_b))
```

Direct air heater:

```
def Cal_MV_Blow_Air(n_Heat_Vap, H_Blow_Air):
    return n_Heat_Vap*H_Blow_Air
```

• **To find** H_{Blow}

```
  def Cal_MV_Blow_Air(n_Heat_Vap, H_Blow_Air):
        return n_Heat_Vap*H_Blow_Air
```

Pad and fan cooling:

```
def Cal_MV_Pad_Air(p_Air, f_Pad, n_Pad, x_Pad, x_Out):
    return p_Air*f_Pad*(n_Pad*(x_Pad-x_Out)+x_Out)

def Cal_MV_Air_Out_Pad(f_pad, M_Water, R, VP_Air, T_Air):
    return f_pad*M_Water/R*(VP_Air/(T_Air+273.15))
```

• **To find** f_{Pad}

```
  def f_Pad(U_Pad, phi_Pad, Air_Flr):
        return U_Pad*phi_Pad/Air_Flr
```

Fogging

```
def Cal_MV_Fog_Air(U_Fog, phi_Fog, Air_Flr):
    return U_Fog*phi_Fog/Air_Flr
```

Function dx to calculate VP_{Air} **and** VP_{Top}

```
def dx(VP_Air, VP_saturated, T_Air, U_ThScr, U_Roof,temp, Windspeed):
    p_Air = Calculate_density(VP_Air,VP_saturated, T_Air)
    p_Other = Calculate_density(VP_Air,VP_saturated, T_Air+temp)
    f_leakage = Cal_f_leakage(Windspeed)

    MV_Blow_Air = Cal_MV_Blow_Air(4.43*10**-8, H_Blow(1, 0, 1.4 * 10**4))
                  #0
    MV_Pad_Air  = Cal_MV_Pad_Air(p_Air, f_Pad(0, 0, 1.4*10**4), 0, 0, 0)
           #0
    MV_Air_Out_Pad = Cal_MV_Air_Out_Pad(f_Pad(0, 0, 1.4*10**4), 18, 8.314*10**3,
    VP_Air, T_Air)      #0
    MV_Fog_Air = Cal_MV_Fog_Air(0, 0, 1.4*10**4)
                  #0
    MV_Can_Air = Cal_MV_Can_Air(VEC_Can_Air(p_Air, 10**3, 2, 2.45*10**6, 65.8,
    275, 82.0), VP_saturated, VP_Air)
    MV_Air_Top =  MV_location_to_location(f_ThScr(U_ThScr, 0.05* 10**-3, T_Air,
    T_Air+temp, (p_Air+p_Other)/2, p_Air, p_Other, 9.81), T_Air, T_Air+temp,
    VP_Air, VP_saturated)
    MV_Air_Out =  MV_location_to_location(f_Vent_Side(1, f_prime_Vent_Roof_side
    (0.75, 1.4*10**4, U_Roof, 0, 0.1*1.4*10**4, 0, 9.81, 0, T_Air, T_Air+temp,
    T_Air+temp/2, 0.09, Windspeed), f_leakage)+f_Vent_Forced(), T_Air, T_Air+temp,
     VP_Air, VP_saturated)
    MV_Top_Out =  MV_location_to_location(f_Vent_Roof(1, f_prime_Vent_roof(0.75,
    U_Roof, 0.1*1.4*10**4, 1.4*10**4, 9.81, 0.68, T_Air, T_Air+temp, T_Air+temp/2,
     0.09, Windspeed), f_leakage), T_Air, T_Air+temp, VP_Air, VP_saturated)
    MV_Air_ThScr = MV_location_to_object(HEC_Air_ThScr(T_Air, T_Air+temp), VP_Air,
     VP_saturated)      #0
    MV_Top_Cov_in = MV_location_to_object(HEC_Top_Cov_in(T_Air+temp, T_Air+temp),
    VP_Air, VP_saturated)   #0
    MV_Air_Mech = MV_location_to_object(HEC_Air_Mech(), VP_Air, VP_saturated) #0

    cap_VP_Air = Cal_cap_VP(18, 4, 8.314*10**3, T_Air)
    cap_VP_Top = Cal_cap_VP(18, 4.8, 8.314*10**3, T_Air+temp)

    VP_Air = (MV_Can_Air + MV_Pad_Air + MV_Fog_Air + MV_Blow_Air - MV_Air_ThScr -
    MV_Air_Top - MV_Air_Out - MV_Air_Out_Pad - MV_Air_Mech) / cap_VP_Air
    VP_Top = (MV_Air_Top - MV_Top_Cov_in - MV_Top_Out) / cap_VP_Top
    return VP_Air, VP_Top
```

- **To find cap_{VPAir} and cap_{VPTop}**

```
  def Cal_cap_VP(M_Water, h, R, T):
        return M_Water*h/(R*(T+273.15))
```

Some other functions: • **To find VP and $VP_{Saturated}$**

```
  def Cal_VP_Air(Relative_Humidity, P_ws):
        return Relative_Humidity * P_ws
    def Cal_VP_saturated(T_Air):
        return 610.78 * math.exp(T_Air * 17.2694 / (T_Air + 238.3))
```

- **To find Air Density**

```
  def pressure_at_height(h):
        return 101325*(1-2.25577*(10**-5)*h)**5.25588

    def Calculate_density(VP_Air, VP_saturated, T):
        return ((pressure_at_height(0) - VP_saturated) / (287.058*(T + 273.15))) +
     (VP_Air/(461.495*(T+273.15)))
```

3.4.3 5.3

Check the program Assuming constant temperature difference and constant air density difference, study from [Van11] and related citations to find specific and reasonable values for each input parameter of the function dx including the difference in temperature and in air density except for the variables CO2 Air and CO2 T op. Noting that these values need to match the units considered in the report. Check if the programs work properly with the values found and specific CO2 Air and CO2 T op values. Present details in the report.

We choose region The Netherlands on Van11 and data on Cucumber/MicrosoftResearch and constant temperature difference equals 1.

Name	Value	Unit	Source/Reason
T_{Can}	$T_{Air} - 1$	K	Constant temperature difference
T_{Top}	$T_{Air} - 1$	K	Constant temperature difference
$T_{TopCov,in}$	$T_{Air} - 1$	K	Constant temperature difference
T_{ThScr}	$T_{Air} - 1$	K	Constant temperature difference
T_{Out}	$T_{Air} - 1$	K	Constant temperature difference
U_{ThScr}	0.95	-	Dataset
C_{HECin}	1.86	$Wm^{-2}K^{-1}$	Van11
M_{Water}	18	$kgkmol^{-1}$	Van11
$S_{MV_{12}}$	-0.1	Pa^{-1}	Van11
R	$8.314 * 10^3$	$Jkmol^{-1}K^{-1}$	Van11
K_{ThScr}	$0.05 * 10^{-3}$	$m^3m^{-2}K^{-0.66}s^{-1}$	Van11
g	9.81	ms^{-2}	Van11
ρ_{Air}	$\frac{353}{291.85}$	$kg \cdot m^{-3}$	$\rho_{Air} = \frac{353}{T_{air}} with T_{Air} = 291.85K$
ρ_{Top}	$\frac{353}{290.85}$	$kg \cdot m^{-3}$	$\rho_{Air} = \frac{353}{T_{top}} with T_{Air} = 290.85K$
${\rho_{air}}^{mean}$	1.21	$kg \cdot m^{-3}$	${\rho_{air}}^{mean} = \frac{\rho_{Air}+\rho_{Top}}{2}$
η_{Side}	1	-	Random
η_{Side_Thr}	0.9	-	η_{Roof_Thr} in Van11
ζ_{InsScr}	1	-	Van11
U_{Roof}	0	-	$\rho_{Air} = \frac{ventlee+ventside}{2}$ with ventlee = ventside = 0
U_{Side}	0	-	Random
A_{Roof}	$0.13 \cdot 10^4$	m^2	Van11: $\frac{A_{Roof}}{A_{Flr}} = 0.1$
A_{Side}	0	m^2	Van11: $\frac{A_{Side}}{A_{Flr}} = 0$
$h_{SideRoof}$	x	m	Van11
${T_{Air}}^{mean}$	291.35	K	${T_{Air}}^{mean} = \frac{T_{Air}+T_{Out}}{2}$ with $T_{Air} = 291.85 and T_{Out} = 290.85$
C_w	0.09	-	Van11
v_{wind}	4.6	ms^{-1}	Van11
$U_{VentForced}$	0	-	No $\phi_{VentForced}$ in the dataset we choose
$\phi_{VentForced}$	x	m^3s^{-1}	Van11
η_{Roof}	1	-	Random
η_{Roof_Thr}	0.9	-	Van11
C_d	0.75	-	Van11
h_{Vent}	0.68	m	Van11
M_{CH_2O}	30	g/mol	

Name	Value	Unit	Source/Reason
$T_{Can,K}$	292.85	K	$T_{Can,K} = T_{Air} + 1$
$T_{25,K}$	298.15	K	Van11
S	710	$Jmol^{-1}$	Van11
H	$22 \cdot 10^4$	$Jmol^{-1}$	Van11
R	8.314	$Jmol^{-1}$	Van11
ρ_{Can}	0.07	-	Van11
K_1	0.7	-	Van11
LAI	2	m^2m^{-2}	Van11

For the particular values of dataset, we choose the data on Cucumber/MicrosoftResearch

Name	Value
T_{Air}	18.8
T_{Top}	18.7
Rh (Relative Humidity)	82
U_{ThScr}	0.95
U_{Roof}	e^{-8}
WindSpeed	0
VP_{Air}	1770.61
VP_{Top}	1770.61
$VP_{Saturated}$	2158.28
p_{Air}	1.1964

The result is $\dot{V}P_{Air} = 1.1369$ and $\dot{V}P_{Top} = -0.0369$

3.4.4 5.4a

Study explicit Euler and Runge–Kutta of order 4 algorithms for solving first-order differential equations (see references [ESG93; EG96]). Write two programs performing the two algorithms and named them euler and rk4 respectively with inputs: a callable function as dx, the values at time t of the two variables CO2 Air and CO2 T op, the time step h. These solvers return the approximate values of CO2 Air and CO2 T op at time t+h. Present details in the report.

- *With Euler algorithm.*

```python
def Euler(func, VP_Air, VP_Top, h):
    time = updateTime()
    VP_saturated = updateSaturated(time)
    T_Air = updateTAir(time)
    U_ThScr = updateUThScr(time)
    U_Roof = updateURoof(time)
    windSpeed = updateVWind(time)

    k = func(VP_Air, VP_saturated, T_Air, U_ThScr, U_Roof)

    VP_Air += CO2_Air + k[0]*h
    VP_Top += CO2_Top + k[1]*h

    return VP_Air, VP_Top
```

- *Runge-Kutta's algorithaKR*

```python
def rk(func, VP_Air, VP_Top, h):
    time = updateTime()
    VP_saturated = updateSaturated(time)
    T_Air = updateTAir(time)
    U_ThScr = updateUThScr(time)
    U_Roof = updateURoof(time)
    k = func(VP_Air, VP_saturated, T_Air, U_ThScr, U_Roof)

    k1 = func(VP_Air, VP_saturated, T_Air, U_ThScr, U_Roof, temp, Windspeed)
      k2 = func(VP_Air + h*k1[0]*0.5, VP_saturated, T_Air, U_ThScr, U_Roof, temp
  , Windspeed)
      k3 = func(VP_Air + h*k2[0]*0.5, VP_saturated, T_Air, U_ThScr, U_Roof, temp
  , Windspeed)
```

```
        k4 = func(VP_Air + h*k3[0], VP_saturated, T_Air, U_ThScr, U_Roof, temp,
    Windspeed)

        VP_Air += h * (k1[0] + 2*k2[0] + 2*k3[0] + k4[0]) * (1 / 6)
        VP_Top += h * (k1[1] + 2*k2[1] + 2*k3[1] + k4[1]) * (1 / 6)

    return VP_Air, VP_Top
```

3.4.5 5.4b

Select specific values of CO2 Air and CO2 Top at time t from the data set as initial values to run the solvers and find the approximate values of CO2 Air and CO2 T op in the next5 minutes, 10 minutes, 20 minutes, ... and calculate the difference of the result from the actual data. Comment on the accuracy of the model and present details in the report.

Step 1: Calculate and save the data

```
  t = 0
h = 0.1
temp = -0.1
end_time = 5


filename = "Data_after_sort.csv"
filename_result_VP_Air = "result_VP_Air_version_2.csv"
filename_result_VP_Top = "result_VP_Top_version_2.csv"

field = []
rows = []


rows_result_VP_Air = []
rows_result_VP_Top = []
field_result_VP_Air = ['Timestamp', 'Real_VP_Air', 'Euler_Vp_Air','
    Euler_Difference', 'Rk_VP_Air', ' Rk_Difference']
field_result_VP_Top = ['Timestamp', 'Euler_VP_Top', 'Rk_VP_Top']

#field_out = ['Timestamp', 'Temp', 'Rh', 'VentLee', 'VentWind', 'EnergyCurtain', '
    OutsideTemp', 'OutsideRh', 'WindSpeed']
with open(filename, mode= 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        rows.append(row)
    print("Total no. of rows: %d"%(csv_reader.line_num))
initial = Cal_VP_Air(float(rows[0]['Rh'])/100, Cal_VP_saturated(float(rows[0]['
    Temp'])))

rows_result_VP_Air.append([rows[0]['Timestamp'], initial, 0, 0, 0, 0])
rows_result_VP_Top.append([rows[0]['Timestamp'], initial, initial])

for i in range(100):

    if ( rows[i]['Temp'] != '') :
        T_Air = float(rows[i]['Temp'])
    else : T_Air = float(rows[i-1]['Temp'])

    if ( rows[i]['Rh'] != '') :
        RH = float(rows[i]['Rh'])/100
    else: RH = float(rows[i-1]['Rh'])/100

```

```python
    if ( rows[i]['EnergyCurtain'] != ''):
        U_ThScr = float(rows[i]['EnergyCurtain'])/100
    else: U_ThScr = float(rows[i-1]['EnergyCurtain'])/100
    if ( rows[i]['VentLee'] != '' and rows[i]['VentWind'] != '') :
        U_Roof = Cal_U_Roof(float(rows[i]['VentLee']), float(rows[i]['VentWind']))
        if U_Roof == 0:
            U_Roof = math.exp(-8)
    else:
        U_Roof = math.exp(-8)

    Windspeed = float(rows[i]['WindSpeed'])

    VP_saturated = Cal_VP_saturated(T_Air)
    VP_Air_rk = Cal_VP_Air(RH, VP_saturated)
    VP_Top_rk = VP_Air_rk
    VP_Air_el = VP_Air_rk
    VP_Top_el = VP_Top_rk

    if ( i > 0 ):
        rows_result_VP_Air.append([rows[i]['Timestamp'], VP_Air_rk, result_el[0],
    result_el[0] - VP_Air_rk, result_rk[0], result_rk[0] - VP_Air_rk])
        rows_result_VP_Top.append([rows[i]['Timestamp'], result_el[1], result_rk
    [1]])

    while t < end_time:
        result_rk = rk4(dx, VP_Air_rk, VP_Top_rk, h, VP_saturated, T_Air, U_ThScr,
     U_Roof, temp, Windspeed)
        result_el = euler(dx, VP_Air_el, VP_Top_el, h, VP_saturated, T_Air,
    U_ThScr, U_Roof, temp, Windspeed)
        t = round(t + h, 1)
        VP_Air_rk = result_rk[0]
        VP_Top_rk = result_rk[1]
        VP_Air_el = result_el[0]
        VP_Top_el = result_el[1]

    t = 0

with open(filename_result_VP_Air, 'w', newline='') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
    # writing the fields
    csvwriter.writerow(field_result_VP_Air)
    csvwriter.writerows(rows_result_VP_Air)

with open(filename_result_VP_Top, 'w', newline='') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
    # writing the fields
    csvwriter.writerow(field_result_VP_Top)
    csvwriter.writerows(rows_result_VP_Top)
```

Illustrating Data in Graphs:

The following table shows the data of VP_{Air} from t = 0 to t = 90(min) with step h = 0.1 (min)

Time (minute)	Real data	Euler data	Runge-Kutta data
0	1770.613	0	0
5	1777.321	1776.255	1776.254
10	1772.976	1783.068	1783.067
15	1770.803	1778.799	1778.797

20	1768.631	1776.657	1776.656
25	1759.817	1774.516	1774.515
30	1775.291	1765.635	1765.634
35	1753.339	1781.276	1781.275
40	1768.631	1776.657	1776.656
45	1751.18	1759.246	1759.245
50	1742.453	1757.105	1757.104
55	1753.339	1748.308	1748.308
60	1751.18	1759.231	1759.230
65	1744.702	1757.118	1757.117
70	1742.543	1750.729	1750.728
75	1744.702	1748.591	1748.590
80	1746.861	1750.719	1750.718
85	1736.065	1752.847	1752.845
90	1736.065	1742.207	1742.206

As you can see, the values of Euler's method and Rybge-Kutta's method are almost the same. Therefore, the graph can not demonstrate the different between data of two methods. For more details, the following link contains all the data during the duration from t = 0 to t = 500 (min)

https://drive.google.com/drive/folders/1EHZswAuSg9qqUgGkQ6D8HeXFyRNbhF8q?usp=sharing

The following graph shows the data of VP_{Air} from t = 0 to t = 500(min) with step h = 0.1 (min)

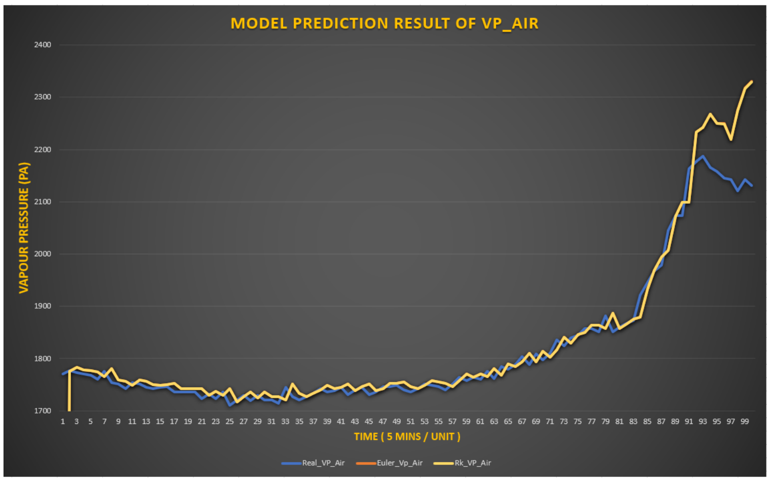


Figure 4: The data of VP_{Air}

The following table shows the data of VP_{Air} from t = 0 to t = 90(min) with step h = 0.1 (min)

Time (minute)	Euler data	Runge-Kutta data
0	0	0

5	1770.613	1770.613
10	1770.430	1770.430
15	1777.140	1777.140
20	1772.804	1772.804
25	1770.630	1770.631
30	1768.457	1768.457
35	1759.645	1759.645
40	1775.114	1775.114
45	1753.160	1753.160
50	1751.000	1751.000
55	1744.514	1744.514
60	1742.346	1742.346
65	1744.506	1744.506
70	1746.667	1746.667
75	1735.865	1735.865
80	1735.865	1735.865
85	1735.865	1735.865
90	1735.865	1735.865

As you can see, the values of Euler's method and Rybge-Kutta's method are almost the same. Therefore, the graph can not demonstrate the different between data of two methods. For more details, the following link contains all the data during the duration from t = 0 to t = 500 (min)

https://drive.google.com/drive/folders/1EHZswAuSg9qqUgGkQ6D8HeXFyRNbhF8q?usp=sharing

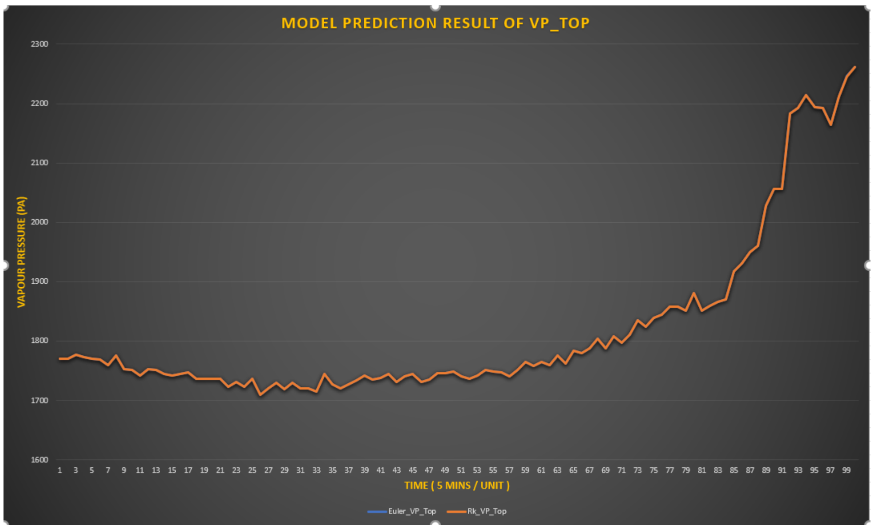


Figure 5: The data of VP_{Top}

References

[Bal89] J Balemans. Assessment of criteria for energetic effectiveness of greenhouse screens 1989.

[BB95] Thierry Boulard and Alain Baille. “Modelling of air exchange rate in a greenhouse equipped with continuous roof vents”. In: Journal of Agricultural Engineering Research 61.1 (1995), pp. 37–47.

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

[Kit+96] C Kittas et al. “Wind induced air exchange rates in a greenhouse tunnel with continuous side openings”. In: Journal of Agricultural Engineering Research 65.1 (1996), pp. 37–49.

[Lom+75] Paul W Lommen et al. “Photosynthetic model”. In: Perspectives of Biophysical Ecology. Springer, 1975, pp. 33–43.

[De 96] HF De Zwart. Analyzing energy-saving options in greenhouse cultivation using a simulation model. 1996.

[EG96] Hairer Ernst and Wanner Gerhard. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer, 1996.

[ESG93] Hairer Ernst, P. Nørsett Syvert, and Wanner Gerhard. Solving Ordinary Differential Equations I: Nonstiff Prolemns. Springer, 1993.

[MG] N.J.van de Braak MIGUEL A.F. and 1995 G.P.ABot. “Mass flow through materials with pores and openings: II-natural convection”. In: submitted for publication in International Journal of Heat and Mass Transfer ().