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


## MÔ HÌNH HOÁ TOÁN HỌC (CO1007)

### Assignment (HK 201)

# "Mô hình hệ động lực dự báo tiểu khí hậu nhà kính"

Giáo viên ra đề và HD: Nguyễn Tiến Thịnh
Nguyễn An Khương
Trợ giảng: Trần Trung Hiếu
Students: Nguyễn Phúc Vinh - 1915940 - Nhóm trưởng.
Cao Thanh Bình - 1912715.
Nguyễn Hữu Kiệt - 1812729
Lê Thanh An - 1912523
Lê Hồ Long - 1913990

HO CHI MINH CITY, SEPTEMBER 2020

# Contents

# 1 Member list & Workload

| No. | Fullname | Student ID | Problems | Percentage of work |
|---|---|---|---|---|
| 1 | Nguyễn Văn A | 19181716 | - Relation & Counting: 1, 2, 3<br>Bonus: 1, 2, 3.<br>- Probability: 1, 2, 3. | 30% |
| 2 | Cao Thanh Bình | 1912715 | - Relation & Counting: 4, 5, 6<br>Bonus: 4, 5, 6.<br>- Graph: 1, 2, 3, Bonus: 1, 2, 3. | 20% |

# 2 Phần kiến thức chuẩn bị (Bài 1)

## 2.1 Trình bày định nghĩa, phân loại theo các tiêu chuẩn khác nhau, dạng tổng quát của hệ động lực nói chung, và đặc biệt là của hệ phương trình vi phân bậc nhất với điều kiện đầu tại thời điểm $t_0$ là mô hình hệ động lực liên tục được sử dụng trong BTL này.

- **Định nghĩa:** Hệ động lực bao gồm một tập hợp $P$ và ánh xạ $\phi_t$: $P \to P$, trong đó thời gian $t$ có thể không liên tục ($t \in \mathbb{Z}$) hoặc liên tục ($t \in \mathbb{R}$). Với mỗi vector $x \in P$ ta có:
  1. $\phi_0(x) = x$
  2. $\phi_t(\phi_s(x)) = \phi_{t+s}(x)\ \forall t, s \in \mathbb{R}$
- **Phân loại:**
  1. Hệ động lực liên tục (Time-continuous): Cho $P \subset 159\mathbb{R}^N$, $N \in \mathbb{N}$, $x = (x^1, x^2, ..., x^N) \in P$, $t \in \mathbb{R}$. Khi đó:
  $$F : P \to P, x' = F(x) = F(x(t))$$
  2. Hệ động lực rời rạc: Cho $P \subset \mathbb{R}^N$, $N \in \mathbb{N}$, $x_n \in P$, $n \in \mathbb{Z}$. Khi đó:
  $$M : P \to P, x_{n+1} = F(x_n)$$
- **Tổng quát hệ phương trình vi phân bậc nhất được sử dụng trong hệ động lực liên tục trong BTL này:**
  - Cho $a_{ij}(t)$, $b_j(t)$ là các hàm xác định với $i \in [1, n]$ và $j \in [1, n]$; $x_i(t)$ là các hàm cần tìm với $i \in [1, n]$

$$\begin{aligned}
x_1'(t) &= a_{11}(t)x_1(t) + a_{12}(t)x_2(t) + ... + a_{1n}(t)x_n(t) + b_1(t) \\
x_2'(t) &= a_{21}(t)x_1(t) + a_{22}(t)x_2(t) + ... + a_{2n}(t)x_n(t) + b_2(t) \\
x_3'(t) &= a_{31}(t)x_1(t) + a_{32}(t)x_2(t) + ... + a_{3n}(t)x_n(t) + b_3(t) \\
&\vdots \\
x_n'(t) &= a_{n1}(t)x_1(t) + a_{n2}(t)x_2(t) + ... + a_{nn}(t)x_n(t) + b_n(t)
\end{aligned}$$

  Với $\mathbf{x}(t)$ là cột vector gồm các hàm cần tìm $x_i(t)$, $i \in [1, n]$, $\mathbf{A}(t)$ gồm các $a_{ij}$ và $\mathbf{b}(t)$ gồm các $b_i(t)$, $i \in [1, n]$. Ta có:
  $$\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{b}(t)$$

– Khi n = 2, phương trình bậc nhất tuyến tính viết ở dạng ma trận là:

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \end{bmatrix} = \begin{bmatrix} a_{11}(t) & a_{12}(t) \\ a_{21}(t) & a_{22}(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} b_1(t) \\ b_2(t) \end{bmatrix}$$

## 2.2 Giới thiệu điều kiện cần và đủ để hệ phương trình vi phân trên tồn tại và duy nhất nghiệm.

- Hệ phương trình vi phân có nghiệm khi và chỉ khi hệ có điều kiện ban đầu.

## 2.3 Cho một số ví dụ về hệ phương trình vi phân bậc nhất giải được và công thức nghiệm chính xác.

**Ví dụ 1:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 7x + 3y$$

$$\frac{dy}{dt} = 6x + 4y$$

Với $x(0) = 1$, $y(0) = 4$; ta có công thức nghiệm chính xác:

$$x(t) = -e^t + 2e^{10t}$$

$$y(t) = 2e^t + 2e^{10t}$$

**Ví dụ 2:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 5x - y$$

$$\frac{dy}{dt} = x + 3y$$

Với $x(0) = 1$, $y(0) = 2$; ta có công thức nghiệm chính xác:

$$x(t) = e^{4t}(-t + 1)$$

$$y(t) = e^{4t}(-t + 2)$$

**Ví dụ 3:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 4x - 3y$$

$$\frac{dy}{dt} = 3x + 4y$$

Với $x(0) = 1$, $y(0) = 3$; ta có công thức nghiệm chính xác:

$$x(t) = e^{4t}(cos3t - 3sin3t)$$

$$y(t) = e^{4t}(sin3t + 3cos3t)$$

**Ví dụ 4:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = x + 8y$$

$$\frac{dy}{dy} = 2x + y$$

Với $x(0) = 3$, $y(0) = 1$; ta có công thức nghiệm chính xác:

$$x(t) = 2e^{-3t} + e^{5t}$$

$$y(t) = -e^{-3t} + 2e^{5t}$$

**Ví dụ 5:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 4x - 3y + e^{-t}$$

$$\frac{dy}{dt} = 2x - y$$

Với $x(0) = 0$, $y(0) = \frac{4}{3}$; ta có công thức nghiệm chính xác:

$$x(t) = 3e^{t} - 3e^{2t}$$

$$y(t) = 3e^{t} - 2e^{2t} + \frac{1}{3}e^{-t}$$

## 2.4 Giới thiệu và trình bày các bước xấp xỉ của giải thuật Explicit Euler và Explicit Runge–Kutta bậc 4 giải hệ phương trình vi phân bậc nhất tổng quát.

- **Explicit Euler**
  Phương pháp Euler là một phương pháp bậc một thường được sử dụng trong việc giải các phương trình vi phân thường (ODE - ordinary differential equation) . Phương pháp được đặt tên theo người giới thiệu nó - ông Leonhard Euler - trong cuốn sách Institutionum Calculi Integralis.
  Cho hệ phương trình vi phân:

$$\frac{dy}{dx} = f_1(x, y, z)$$

$$\frac{dz}{dx} = f_2(x, y, z)$$

  Với giá trị ban đầu $x_0$, $y_0$, $z_0$ thì giá trị mới của $y_1$ và $z_1$ là:

$$y_1 = y_0 + f_1(x_0, y_0, z_0).h$$

$$z_1 = z_0 + f_2(x_0, y_0, z_0).h$$

  Tổng quát:

$$y_{n+1} = y_n + f_1(x_n, y_n, z_n).h + O(h^2)$$
$$z_{n+1} = z_n + f_2(x_n, y_n, z_n).h + O(h^2)$$

- **Explicit Runge-Kutta bậc 4**

  – Phương pháp Runge-Kutta cũng như phương pháp Euler, là một phương pháp dùng để giải gần đúng hệ phương trình vi phân bậc nhất. Phương pháp Runge-Kutta được phát triển vào khoảng năm 1900 bởi hai nhà toán Đức C. Runge và M. W. Kutta. Phương pháp Explicit Runge-Kutta có nhiều bậc, bậc 1 chính là phương pháp Explicit Euler được trình bày ở trên. Ở đây chúng ta dùng bậc 4 vì bậc 4 là bậc cho kết quả chính xác nhất.

  – Với hệ phương trình vi phân như trên, ta có công thức tổng quát cho giải thuật Explicit Runge-Kutta bậc 4 là:

$$y_{i+1} = y_i + \frac{1}{6}(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}) + O(h^5)$$

$$z_{i+1} = z_i + \frac{1}{6}(t_{1i} + 2t_{2i} + 2t_{3i} + t_{4i}) + O(h^5)$$

với $\begin{cases} k_{1i} = h.f_1(x_i, y_i, z_i) \\ k_{2i} = h.f_1(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_{1i}, z_i + \frac{1}{2}k_{1i}) \\ k_{3i} = h.f_1(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_{2i}, z_i + \frac{1}{2}k_{2i}) \\ k_{4i} = h.f_1(x_i + h, y_i + k_{3i}, z_i + k_{3i}) \end{cases}$ và $\begin{cases} t_{1i} = h.f_2(x_i, y_i, z_i) \\ t_{2i} = h.f_2(x_i + \frac{1}{2}h, y_i + \frac{1}{2}t_{1i}, z_i + \frac{1}{2}t_{1i}) \\ t_{3i} = h.f_2(x_i + \frac{1}{2}h, y_i + \frac{1}{2}t_{2i}, z_i + \frac{1}{2}t_{2i}) \\ t_{4i} = h.f_2(x_i + h, y_i + t_{3i}, z_i + t_{3i}) \end{cases}$

## 2.5 Sử dụng công thức nghiệm xấp xỉ Explicit Euler và Explicit Runge–Kutta bậc 4, thực hiện tính xấp xỉ nghiệm chính xác của các ví dụ phía trên tại các thời điểm $t_0$, $t_0 + h$, $t_0 + 2h$,..., $t_0 + 5h$ với $h$ tự chọn.

**Ví dụ 1:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 7x + 3y$$

$$\frac{dy}{dt} = 6x + 4y$$

Với $t_0 = 0$, $x(0) = 1$, $y(0) = 4$, $h = 0.1$

- **Explicit Euler**

  Áp dụng công thức nghiệm xấp xỉ, ta có:

$$\begin{aligned} x_1 &= x_0 + f(t_0, x_0, y_0).h \\ &= x_0 + (7x_0 + 3y_0).h \\ &= 2.9 \end{aligned} \qquad \begin{aligned} y_1 &= y_0 + g(t_0, x_0, y_0).h \\ &= y_0 + (6x_0 + 4y_0).h \\ &= 6.2 \end{aligned}$$

$$\begin{aligned} x_2 &= x_1 + f(t_1, x_1, y_1).h \\ &= x_1 + (7x_1 + 3y_1).h \\ &= 6.79 \end{aligned} \qquad \begin{aligned} y_2 &= y_1 + g(t_1, x_1, y_1).h \\ &= y_1 + (6x_1 + 4y_1).h \\ &= 10.42 \end{aligned}$$

$$\begin{aligned} x_3 &= x_2 + f(t_2, x_2, y_2).h \\ &= x_2 + (7x_2 + 3y_2).h \\ &= 14.669 \end{aligned} \qquad \begin{aligned} y_3 &= y_2 + g(t_2, x_2, y_2).h \\ &= y_2 + (6x_2 + 4y_2).h \\ &= 18.662 \end{aligned}$$

$$\begin{aligned} x_4 &= x_3 + f(t_3, x_3, y_3).h \\ &= x_3 + (7x_3 + 3y_3).h \end{aligned} \qquad \begin{aligned} y_4 &= y_3 + g(t_3, x_3, y_3).h \\ &= y_3 + (6x_3 + 4y_3).h \end{aligned}$$

$= 30.5359$ $= 34.9282$

$x_5 = x_4 + f(t_0, x_0, y_0).h$ $y_5 = y_4 + g(t_4, x_4, y_4).h$

$= x_4 + (7x_4 + 3y_4).h$ $= y_4 + (6x_4 + 4y_4).h$

$= 62.38949$ $= 67.22102$

- **Explicit Runge-Kutta bậc 4**

  Áp dụng công thức nghiệm xấp xỉ, ta có:

  $x_1 = x_0 + \frac{1}{6}(k_{10} + 2k_{20} + 2k_{30} + k_{40})$ $y_1 = y_0 + \frac{1}{6}(t_{10} + 2t_{20} + 2t_{30} + t_{40})$

  $= 4.2458333333$ $= 7.7583333333$

  $x_2 = x_1 + \frac{1}{6}(k_{11} + 2k_{21} + 2k_{31} + k_{41})$ $y_2 = y_1 + \frac{1}{6}(t_{11} + 2t_{21} + 2t_{31} + t_{41})$

  $= 13.2992881944$ $= 17.4118402778$

  $x_3 = x_2 + \frac{1}{6}(k_{12} + 2k_{22} + 2k_{32} + k_{42})$ $y_3 = y_2 + \frac{1}{6}(t_{12} + 2t_{22} + 2t_{32} + t_{42})$

  $= 38.1265884693$ $= 42.9417015336$

  $x_4 = x_3 + \frac{1}{6}(k_{13} + 2k_{23} + 2k_{33} + k_{43})$ $y_4 = y_3 + \frac{1}{6}(t_{13} + 2t_{23} + 2t_{33} + t_{43})$

  $= 105.7272558832$ $= 111.3649507626$

  $x_5 = x_4 + \frac{1}{6}(k_{14} + 2k_{24} + 2k_{34} + k_{44})$ $y_5 = y_4 + \frac{1}{6}(t_{14} + 2t_{24} + 2t_{34} + t_{44})$

  $= 289.233969976$ $= 295.8347710639$

**Ví dụ 2:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 5x - y$$

$$\frac{dy}{dt} = x + 3y$$

Với $t_0 = 0$, $x(0) = 1$, $y(0) = 2$, $h = 0.1$
Tương tự như ví dụ 1, ta có:

- **Explicit Euler**

  $x_1 = 1.3$ $y_1 = 2.7$

  $x_2 = 1.68$ $y_2 = 3.64$

  $x_3 = 2.156$ $y_3 = 4.9$

  $x_4 = 2.744$ $y_4 = 6.5856$

  $x_5 = 3.45744$ $y_5 = 8.83568$

- **Explicit Runge-Kuta bậc 4**

  $x_1 = 1.3688$ $y_1 = 2.8605333333$

  $x_2 = 1.8585008356$ $y_2 = 4.0837691733$

  $x_3 = 2.4988279921$ $y_3 = 5.8183349472$

  $x_4 = 3.3195069551$ $y_4 = 8.2713261302$

  $x_5 = 4.3430755379$ $y_5 = 11.729869262$

**Ví dụ 3:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 4x - 3y$$

$$\frac{dy}{dy} = 3x + 4y$$

Với $t_0 = 0$, $x(0) = 1$, $y(0) = 3$, $h = 0.1$
Tương tự như ví dụ trên, ta có:

- **Explicit Euler**

  $x_1 = 0.5$ $y_1 = 4.5$
  $x_2 = -0.65$ $y_2 = 6.45$
  $x_3 = -2.845$ $y_3 = 8.835$
  $x_4 = -6.6335$ $y_4 = 11.5155$
  $x_5 = -12.74155$ $y_5 = 14.13165$

- **Explicit Runge-Kutta bậc 4**

  $x_1 = 0.4741458333$ $y_1 = 5.1689375$
  $x_2 = -0.95725331$ $y_2 = 8.3642433367$
  $x_3 = -3.998977149$ $y_3 = 12.7867377584$
  $x_4 = -9.715655783$ $y_4 = 18.4476674655$
  $x_5 = -19.6233399246$ $y_5 = 24.9029608335$

**Ví dụ 4:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = x + 8y$$

$$\frac{dy}{dt} = 2x + y$$

Với $t_0 = 0.x(0) = 3$, $y(0) = 1$, $h = 0.01$

- **Explicit Euler**

  $x_1 = 3.11$ $y_1 = 1.07$
  $x_2 = 3.2267$ $y_2 = 1.1429$
  $x_3 = 3.350399$ $y_3 = 1.218863$
  $x_4 = 3.48141203$ $y_4 = 1.29805961$
  $x_5 = 3.6200709191$ $y_5 = 1.3806684467$

- **Explicit Runge-Kutta bậc 4**

  $x_1 = 3.1151018413$ $y_1 = 1.0710605788$
  $x_2 = 3.2373565972$ $y_2 = 1.145179445$
  $x_3 = 3.3670951251$ $y_3 = 1.2225328709$
  $x_4 = 3.5046665$ $y_4 = 1.3033056352$
  $x_5 = 3.6504389176$ $y_5 = 1.3876914798$

**Ví dụ 5:** Cho hệ phương trình vi phân:

$$\frac{dx}{dt} = 4x - 3y + e^{-t}$$

$$\frac{dy}{dt} = 2x - y$$

Với $t_0 = 0$, $x(0) = 0$, $y(0) = \frac{4}{3}$, $h = 0.01$

- **Explicit Euler**

  $x_1 = -0.03$ $y_1 = 1.32$
  $x_2 = -0.060766611$ $y_2 = 1.3062$
  $x_3 = -0.0923452032$ $y_3 = 1.2919226678$
  $x_4 = -0.1247538264$ $y_4 = 1.277156537$
  $x_5 = -0.1580109005$ $y_5 = 1.2618898951$

- **Explicit Runge-Kutta bậc 4**

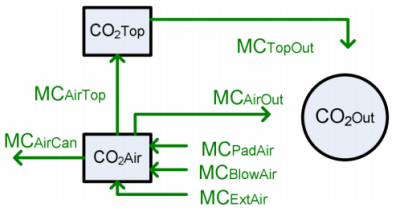
| | |
|---|---|
| $x_1 = -0.0302005017$ | $y_1 = 1.3199331106$ |
| $x_2 = -0.0613105636$ | $y_2 = 1.3060605221$ |
| $x_3 = -0.0933515185$ | $y_3 = 1.2917020328$ |
| $x_4 = -0.1263451586$ | $y_4 = 1.2768438148$ |
| $x_5 = -0.1603137449$ | $y_5 = 1.2614717412$ |

# 3 Phần ứng dụng

## 3.1 Bài 2

### 3.1.1 Chi tiết mô hình đối với nồng độ khí $CO_2$ trong nhà kính được sử dụng trong đề tài này

Dòng chuyển động của khí trong nhà kính được mô tả qua hình sau:




Hình 1: Dòng chuyển động của khí $CO_2$ bên trong và bê ngoài nhà kính

Sự thay đổi nồng độ của khí $CO_2$ ở gian trên và gian dưới được biểu diễn qua hệ gồm 2 phương trình sau đây:

$$\begin{cases} \text{cap}_{CO_{2Air}} \dot{CO_{2Air}} = MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} - MC_{AirCan} - MC_{AirTop} - MC_{AirOut}, \\ \text{cap}_{CO_{2Top}} \dot{CO_{2Top}} = MC_{AirTop} - MC_{TopOut} \end{cases}$$

Trong hệ này, một số giả thiết sau được đưa ra:

- Lượng khí $CO_2$ ở gian trên và gian dưới không bị ảnh hưởng bởi nguồn nào khác trong những nguồn ở hình 1.
- Nồng độ $CO_2$ phân bố đều nhau ở gian dưới và gian trên.

Sau đây chúng tôi sẽ trình bày một số công thức để tính $MC_{AB}$.
Đầu tiên là công thức tính lượng khí đi từ máy sưởi và gian dưới nhà kính:

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

Trong đó:

- $\eta_{HeatCO_2}$ là lượng khí$CO_2$ sinh ra khi 1 Joule nhiệt lượng (cảm nhận được) được sinh ra bởi máy sưởi ($mg_{CO_2} J^{-1}$).
- $U_{Blow}$ thể hiện mức cho phép lượng khí $CO_2$ được sinh ra đi vào gian dưới và được điều chỉnh trong khoảng $[0, 1]$.

- $P_{Blow}$ là khả năng sinh ra $CO_2$ của máy sưởi (W).
- $A_{Flr}$ là diện tích nhà kính ($m^2$).

Lượng khí $CO_2$ được bơm vào nhà kính bởi bên thứ 3 chuyên cung cấp khí $CO_2$ được cho bởi công thức:

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

Trong đó:

- $U_{ExtCO_2}$ là tham số điều chỉnh tốc độ bơm khí $CO_2$ vào tronh nhà kính.
- $\phi_{ExtCO_2}$ là khả năng bơm $CO_2$ của bên thứ 3 ($mgs^{-1}$).

Lượng khí đi qua hệ thống thông gió vào nhà kính được tính bởi công thức sau:

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

Trong đó:

- Tốc độ đi qua tấm thông gió $f_{Pad}$ ($\mathrm{ms}^{-1}$).
- $U_{Pad}$ là mức cho phép lượng khí $CO_2$ đi qua tấm thông gió điều chỉnh được trong khoảng $[0, 1]$. tem $\phi_{Pad}$ là khả năng cho phép khí $CO_2$ đi qua tấm thông gió ($\mathrm{m}^3\mathrm{s}^{-1}$).

Lượng khí đi từ gian dưới lên gian trên nhà kính được tính bởi công thức:

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

Trong đó tốc độ lưu thông khí $CO_2$ qua màn chắn nhiệt $f_{ThScr}$ ($\mathrm{m\ s}^{-1}$) được tính bởi:

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

Trong đó:

- $U_{ThScr} \in [0, 1]$ là nơi có màn chắn nhiệt.
- $T_{Top}$ (K) và $T_{Air}$ (K) lần lượt là nhiệt độ bên trên và bên dưới màn chắn.
- $K_{ThScr}$ ($\mathrm{mK}^{-\frac{2}{3}}\mathrm{s}^{-1}$) là khả năng cho không khí thẩm thấu của màn chắn.
- $\rho_{Air}$ và $\rho_{Top}$ ($\mathrm{kgm}^{-3}$) lần lượt là mật độ không khí của tầng dưới và tầng trên màn chắn nhiệt.
- $g$ ($\mathrm{ms}^{-2}$) là gia tốc trọng trường.

Công thức Navier-Stokes xét mô hình lý thuyết về sự trao đổi không khí thông qua các vết nứt trên bề mặt màn chắn gây ra bởi sự chênh lệch nhiệt độ không khí:

$$\phi_{crack} = \frac{L \cdot SO}{\rho_{mean}} \left[ \frac{1}{2} \rho_{mean} \cdot SO \cdot g \cdot (\rho_1 - \rho_2) \right]^{\frac{1}{2}}$$

Trong đó:

- $\phi_{crack}$ ($\mathrm{m^3 s^{-1}}$) là lưu lượng không khí đi qua màn chắn.
- $L$ (m) là chiều dài khoản mở trên màn chắn.
- $SO$ (m) là khoản mở trên màn chắn.
- $\rho_1$ và $\rho_2$ ($\mathrm{kgm^{-3}}$) lần lượt là mật độ phía trên và phía dưới màn chắn.
- $\rho_{mean}$ ($\mathrm{kgm^{-3}}$) là mật độ trung bình của mật độ phía trên và phía dưới.
- $g$ ($\mathrm{ms^{-2}}$) là gia tốc trọng trường.

Lượng khí $CO_2$ từ bên trong ra bên ngoài nhà kính theo hai hướng từ gian dưới và từ gian trên qua các ô thông gió ta sử dụng các công thức như dưới đây:

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

Trong đó:

- $f_{VentSize}$ ($\mathrm{ms^{-1}}$) là tốc độ gió từ hệ thống quạt trên tường bao xung quanh nhà kính.
- $f_{VentForced}$ ($\mathrm{ms^{-1}}$) là tốc độ gió từ hệ thống quạt bên trong nhà kính.

Công thức tổng quát dưới đây $f_{VentRoofSide}$ ($\mathrm{ms^{-1}}$) được dùng để thiết lập công thức cho $f_{VentSide}$:

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

Trong đó:

- $C_d$ là lưu lượng gió
- $A_{Flr}$ ($\mathrm{m_2}$) diện tích nhà kính
- $A_{Roof}$ ($\mathrm{m_2}$) diện tích ô thông gió trên mái nhà
- $v_{Wind}$ ($\mathrm{ms^{-1}}$) là tốc độ gió tự nhiên
- $C_w$ là hệ số áp suất

Khi có lưới chắn côn trùng, tốc độ chuyển động của luồng khí qua các nơi thông gió sẽ giảm xuống với hệ số

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

Trong đó:

- $\zeta_{InsScr}$ là độ rỗ của lưới hay tỉ lệ diện tích các lỗ trên lưới và tổng diện tích của lưới

Tốc độ của trao đổi không khí xấp xỉ 50% tốc độ:

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

Ta có công thức tính tốc độ gió của hệ thống quạt trên tường bao xung quanh nhà kính:

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

Trong đó, $f''_{VentSide}$ là $f_{VentRoofSide}$ tính tại $A_{Roof} = 0$.

Tốc độ $f_{VentForced}$ bởi hệ thống quạt gió được tính bởi công thức:

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

Trong đó:

- $U_{VentForced}$ thể hiện sự điều chỉnh tốc độ gió cúa hệ thống và có giá trị trong khoảng $[0, 1]$.

Ta có $MC_{AirOut}$ là lượng khí $CO_2$ đi từ gian trên nhà kính ra ngoài thông qua ô mở trên mái nhà kính được tính bởi công thức:

$$MC_{TopOut} = f_{VentRoof}(CO_{2Top} - CO_{2Out}).$$

Trong đó, $f_{VentRoof}$ là tốc độ luồng khí đi qua ô mở mái nhà kính được tính bởi công thức:

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

Trong trường hợp $\eta_{Roof}$ vượt ngưỡng $\eta_{Roof\_Thr}$, $''f_{VentRoof}$ được tính như sau;

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

Lượng khí $CO_2$ bị hấp thụ vào trong tán lá thông qua quá trình quang hợp:

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

Trong đó:

- $M_{CH_2O}$ là khối lượng mol $CH_2O$ (mg$\mu$ mol$^{-1}$)

- $P$ là tốc độ quang hợp ($\mu$mol$\{CO_2\}$m$^{-2}$s$^{-1}$)
- $R$ là tốc độ hô hấp của cây ($\mu$mol$\{CO_2\}$m$^{-2}$s$^{-1}$)
- $h_{C_{Buf}}$ là hệ số được cho bởi:

$$h_{C_{Buf}} = \begin{cases} 0, & \mathrm{C}_{Buf} > C_{Buf}^{Max}, \\ 1, & \mathrm{C}_{Buf} \leq C_{Buf}^{Max}, \end{cases}$$

Quá trình quang hợp của lá được cho bởi định luật Fick:

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

Trong đó:

- $CO_{2Stom}$ là nồng độ khí $CO_2$ hấp thụ vào trong khí khổng ($\mu$mol m$^{-3}$).
- $Res$ là hệ số cản trở sự hấp thụ $CO_2$ và bên trong lá (s m$^{-1}$).

Quá trình quang hợp ở pha tối được biểu diễn qua mô hình động lực Michaelis–Menten, khi đó tốc độ phản ứng được cho bởi công thức:

$$P = \frac{P_{Max} \cdot CO_{2Stom}}{CO_{2Stom} + CO_{2\ 0.5}}$$

Trong đó:

- $CO_{2\ 0.5}$ là nồng độ khí $CO_2$ trong chất nền khi $P = P_{Max/2}$ ($\mu$mol$^{-3}$).

Giải tìm $CO_{2Stom}$ thì ta có tốc độ quang hợp P thỏa phương trình:

$$ResP^2 - (CO_{2\ Air} + CO_{2\ 0.5} + ResP_{Max})P + CO_{2\ Air}P_{Max} = 0$$

Đối với phương trình bậc 2 trên, ta chỉ quan tâm đến nghiệm $P$ sao cho $P \to P_{Max}$ khi $CO_{2\ Air} \to +\infty$.

Để giải phương trình trên, tốc độ quang hợp cực đại cần được xác định. Tốc độ đó được xác định bởi mô hình phản ứng hóa học Arrhenius

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

Trong đó:

- $k(T)$ là tốc độ phản ứng tại nhiệt độ $T$(K)
- $T_0$ là nhiệt độ tối ưu mà tốc độ phản ứng đã biết(K)
- $H_a$ là năng lượng hoạt hóa phản ứng (J mol$^{-1}$)
- $R$ là hằng số khí lí tưởng (J mol$^{-1}$K$^{-1}$)

Tuy nhiên đến một nhiệt độ càng cao đến một ngưỡng nào đó hoạt động của enzyme bị ức chế và tốc độ phản ứng sẽ giảm. Khi đó, mô hình Arrhenius không đủ để giải thích sự ức chế của enzyme và mô hình sau được xem như là mô hình cho sự hoạt động của enzyme trong quá trình quang hợp và phụ thuộc vào nhiệt độ của lá.

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

Trong đó:

- $f(T)$ đại diện cho sự hoạt động của enzyme ở nhiệt độ $T$(K)
- $H_d$ là năng lượng ức chế enzyme (J $\text{mol}^{-1}$)
- $S$ là một đại lượng entropy tương ứng (J $\text{mol}^{-1}\text{K}^{-1}$)

Kết hợp hai mô hình trên, tốc độ quang hợp tối đa trên mỗi đơn vị lá được cho bởi công thức

$$P_{Max}(T) = k(T)f(T)$$

Ta có định luật Beer về sự hấp thụ ánh sáng của tán lá dưới sự ảnh hưởng của chỉ số $LAI$ (leaf area index).

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

Trong đó:

- $I_0$ là năng lượng ánh sáng đi đến tán lá trước khi vào tán lá ($\mu$mol {photon} $\text{m}^{-2}\text{s}^{-1}$)
- $I$ là năng lượng ánh sáng sai khi đi xuyên qua tán lá ($\mu$mol {photon} $\text{m}^{-2}\text{s}^{-1}$)
- $K$ là hệ số tắt có giá trị từ 0.7 tới 1.0 nếu lá cây phân tầng ngang như cây cà chua và từ 0.3 đến 0.5 nếu lá cây nằm nghiêng như trong trường hợp cây lúa nước
- $m$ là hệ số truyền ánh sáng của lá cây và thường mặc định là 0.1

Khi đó năng lượng ánh sáng mà lá cây nhận được là sự chênh lệch năng lượng của tia trước khi vào tán là và tia ló sau khi vào tán lá và được tính bởi công thức

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

Trong đó:

- $L$ là lượng photon nhận được bởi lá cây ($\mu$mol {photon} $\text{m}^{-2}\text{s}^{-1}$)
- $L_0$ là lượng photon ban đầu phía trên tán lá ($\mu$mol {photon} $\text{m}^{-2}\text{s}^{-1}$)

Sau đây ta có công thức Arrhenius mở rộng thay cho công thức Arrhenius ở trên

$$k(T) = LAI.k(T_0).e^{-\frac{H_d}{R}\left(\frac{1}{T_0} - \frac{1}{\frac{H_d}{S}}\right)}$$

.

Khác với mô hình quang hợp cho một đơn vị lá, lượng năng lượng ánh sáng hấp thụ vào trong tán lá bị ảnh hưởng bởi LAI cần được thêm vào và ảnh hưởng đến tốc độ quang hợp cực đại $P_{Max}$. Do đó, ta xét mô hình sau cho $P_{Max}$, là hàm số phụ thuộc vào $L$ và $T$.

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

Trong đó:

- $L_{0.5}$ là lượng ánh sáng khi $P_{Max}(L,T) = P_{Max}(T/2)$ ($\mu$mol {photon} $m^{-2}s^{-1}$)
- $P_{MLT}$ là tốc độ quang hợp cực đại tại điểm bão hòa ánh sáng và nhiệt độ tối ưu $T$.

### 3.1.2 Code bài 2b

**Dữ liệu được đọc từ file** *data_in\data_CO2.xlsx*
Source trích từ file *source\code2_4.py*

```
import math
from math import sqrt
import pandas as pd
from xlsxwriter import Workbook

# formula 3
def cal_MCBlowAir(nHeatCO2, UBlow, PBlow, AFlr):
    return nHeatCO2 * UBlow * PBlow / AFlr


# formula 4
def cal_MCExtAir(UExtCO2, phiExtCO2, AFlr):
    return UExtCO2 * phiExtCO2 / AFlr


# formula 5
def cal_MCPadAir_1(fPad, CO2Out, CO2Air):
    return fPad * (CO2Out - CO2Air)


def cal_MCPadAir_2(UPad, phiPad, AFlr, CO2Out, CO2Air):
    fPad = UPad * phiPad / AFlr
    return fPad * (CO2Out - CO2Air)


# formula 6
def cal_MCAirTop(fThScr, CO2Air, CO2Top):
    return fThScr * (CO2Air - CO2Top)


# formula 7
def cal_fThScr(UThScr, KThScr, TAir, TTop, g, pAir, pTop):
    a = UThScr * KThScr * pow(abs(TAir - TTop), 2 / 3)
    PMean_Air = (pAir + pTop) / 2
    b = (1 - UThScr) * pow(g * (1 - UThScr) * abs(pAir - pTop) / (2 * PMean_Air), 1 / 2)
    return a + b
```

```python
# formula 9
def cal_MCAirOut(fVentSide, fVentForce, CO2Air, CO2Out):
    return (fVentSide + fVentForce) * (CO2Air - CO2Out)


# formula 10
def cal_ppfVentRoofSide(Cd, AFlr, URoof, USide, ARoof, ASide, g, hSideRoof, TAir, TOut, Cw, vWind
    a = Cd / AFlr
    b = pow(URoof * USide * ARoof * ASide, 2) / (pow(URoof * ARoof, 2) + pow(USide * ASide, 2))
    TMean_Air = (TAir + TOut) / 2
    c = 2 * g * hSideRoof * (TAir - TOut) / TMean_Air
    _d = (URoof * ARoof + USide * ASide) / 2
    d = pow(_d, 2) * Cw * pow(vWind, 2)
    return a * sqrt(b * c + d)


# formula 11
def cal_nInsScr(sInsScr):
    return sInsScr * (2 - sInsScr)


# formula 12
def cal_fleakage(cleakage, vWind):
    if vWind < 0.25:
        return 0.25 * cleakage
    else:
        return vWind * cleakage


# formula (**) calculate ppfVentSide
def cal_ppfVentSide(Cd, USide, ASide, vWind, AFlr, Cw):
    return Cd * USide * ASide * vWind * sqrt(Cw) / (2 * AFlr)


# formula 13, use formula 10 and formula (**)
def cal_fVentSide(nInsScr, ppfVentSide, fleakage, UThScr, ppfVentRoofSide, nSide, nSide_Thr):
    # nSide_Thr la nguong Stack
    # pp_fVentSide la f"VentSide tinh bang ppfVentRoofSide tai ARoof = 0
    if nSide >= nSide_Thr:
        return nInsScr * ppfVentSide + 0.5 * fleakage
    else:
        return nInsScr * (UThScr * ppfVentSide + (1 - UThScr) * ppfVentRoofSide * nSide) + 0.5 *


# formula 14
def cal_fVentForced(nInsScr, UVentForced, phiVentForced, AFlr):
    return nInsScr * UVentForced * phiVentForced / AFlr
```

```python
# formula 15
def cal_MCTopOut(fVentRoof, CO2Top, CO2Out):
    return fVentRoof * (CO2Top - CO2Out)


# formula 16
def cal_fVentRoof(nInsScr, fleakage, UThScr, ppfVentRoofSide, nRoof, nSide, nRoof_Thr, ppfVentRoo
    # nRoof_Thr la nguong Stack
    if nRoof >= nRoof_Thr:
        return nInsScr * ppfVentRoof + 0.5 * fleakage
    else:
        return nInsScr * (UThScr * ppfVentRoof + (1 - UThScr) * ppfVentRoofSide * nSide) + 0.5 *


# formula 17
def cal_ppfVentRoof(Cd, URoof, ARoof, AFlr, g, hVent, TAir, TOut, Cw, vWind):
    TMeanAir = (TAir + TOut) / 2
    part1 = Cd * URoof * ARoof / (2 * AFlr)
    part2 = g * hVent * (TAir - TOut) / 2 / TMeanAir + Cw * pow(vWind, 2)
    return part1 * sqrt(part2)


# formular 18 include 19
# tinh P
def cal_P(CO2Air, LAI):
    T_Can_K = 20 + 273
    J_POT = LAI * 210 * math.exp(37000 * (T_Can_K - 298.15) / (8.314 * T_Can_K * 298.15)) * (
                1 + math.exp((710 * 298.15 - 220000) / (8.314 * 298.15))) / (
                        1 + math.exp((710 * T_Can_K - 220000) / (8.314 * T_Can_K)))
    J = (J_POT + 38.5 - math.sqrt(math.pow(J_POT + 38.5, 2) - 2.8 * J_POT * 38.5)) / 1.4
    CO2Stom = 0.67 * CO2Air
    P = (J * (CO2Stom - 498.1)) / (4 * (CO2Stom + 2 * 498.1))
    return P


# tinh R
def cal_R(CO2Air, P):
    CO2Stom = 0.67 * CO2Air
    R = P * 498.1 / CO2Stom
    return R


def cal_MCAirCan(P, R, CBuf, CMaxBuf):
    MCH2O = 0.03
    hCBuf = 1
    if CBuf > CMaxBuf:
        hCBuf = 0
    return MCH2O * hCBuf * (P - R)
```

```
print("Input line of data")
i = int(input())
# Read data from excel file
data = pd.read_excel("../data_in/data_CO2.xlsx")
df = pd.DataFrame(data)
nHeatCO2 = float(df.at[i, "nHeatCO2"])
UBlow = float(df.at[i, "UBlow"])
PBlow = float(df.at[i, 'PBlow'])
AFlr = float(df.at[i, 'AFlr'])
UExtCO2 = float(df.at[0, 'UExtCO2'])
phiExtCO2 = float(df.at[0, 'phiExtCO2'])
UPad = float(df.at[0, 'UPad'])
phiPad = float(df.at[0, 'phiPad'])
CO2Out = float(df.at[0, 'CO2Out'])
LAI = float(df.at[0, 'LAI'])
CBuf = float(df.at[0, 'CBuf'])
CMax_Buf = float(df.at[0, 'CMax_Buf'])
UThScr = float(df.at[0, 'UThScr'])
KThScr = float(df.at[0, 'KThScr'])
TAir = float(df.at[0, 'TAir'])
TTop = float(df.at[0, 'TTop'])
g = float(df.at[0, 'g'])
pAir = float(df.at[0, 'pAir'])
pTop = float(df.at[0, 'pTop'])
cleakage = float(df.at[0, 'cleakage'])
vWind = float(df.at[0, 'vWind'])
Cd = float(df.at[0, 'Cd'])
URoof = float(df.at[0, 'URoof'])
USide = float(df.at[0, 'USide'])
ARoof = float(df.at[0, 'ARoof'])
ASide = float(df.at[0, 'ASide'])
hSideRoof = float(df.at[0, 'hSideRoof'])
TOut = float(df.at[0, 'TOut'])
Cw = float(df.at[0, 'Cw'])
nSide = float(df.at[0, 'nSide'])
nSide_Thr = float(df.at[0, 'nSide_Thr'])
sInsScr = float(df.at[0, 'sInsScr'])
UVentForced = float(df.at[0, 'UVentForced'])
phiVentForced = float(df.at[0, 'phiVentForced'])
capCO2Air = float(df.at[0, 'capCO2Air'])
hVent = float(df.at[0, 'hVent'])
nRoof = float(df.at[0, 'nRoof'])
nRoof_Thr = float(df.at[0, 'nRoof_Thr'])
capCO2Top = float(df.at[0, 'capCO2Top'])
CO2Air = float(df.at[0, 'CO2Air'])
CO2Top = float(df.at[0, 'CO2Top'])

# formular 1
```

```
def dxCO2Air(CO2Air, CO2Top):
    # TODO

    ######## Calculate MCBlowAir ########
    MCBlowAir = cal_MCBlowAir(nHeatCO2, UBlow, PBlow, AFlr)
    print("MCBlowAir = ", MCBlowAir)
    ######## Calculate MCExtAir ########
    MCExtAir = cal_MCExtAir(UExtCO2, phiExtCO2, AFlr)
    print("MCExtAir = ", MCExtAir)
    ######## Calculate MCPadAir ########
    MCPadAir = cal_MCPadAir_2(UPad, phiPad, AFlr, CO2Out, CO2Air)
    print("MCPadAir = ", MCPadAir)
    ######## Calculate MCAirCan ########
    P = cal_P(CO2Air, LAI)
    R = 0
    MCAirCan = cal_MCAirCan(P, R, CBuf, CMax_Buf)
    print("MCAirCan = ", MCAirCan)
    ######## Calculate MCAirTop ########
    fThScr = cal_fThScr(UThScr, KThScr, TAir, TTop, g, pAir, pTop)
    MCAirTop = cal_MCAirTop(fThScr, CO2Air, CO2Top)
    print("MCAirTop = ", MCAirTop)
    ######## Calculte MCAirOut ########
    # Calculate fleakage
    fleakage = cal_fleakage(cleakage, vWind)

    # Calculate ppfVentRoofSide
    ppfVentRoofSide = cal_ppfVentRoofSide(Cd, AFlr, URoof, USide, ARoof, ASide, g, hSideRoof, TAi

    # Calculate ppfVentSide
    ppfVentSide = cal_ppfVentSide(Cd, USide, ASide, vWind, AFlr, Cw)

    # Calculate fVentSide
    nInsScr = cal_nInsScr(sInsScr)
    fVentSide = cal_fVentSide(nInsScr, ppfVentSide, fleakage, UThScr, ppfVentRoofSide, nSide, nSi

    # Calculate fVentForce
    fVentForced = float(cal_fVentForced(nInsScr, UVentForced, phiVentForced, AFlr))

    MCAirOut = cal_MCAirOut(fVentSide, fVentForced, CO2Air, CO2Out)
    print("MCAirOut = ", MCAirOut)
    return (MCBlowAir + MCExtAir + MCPadAir - MCAirCan - MCAirTop - MCAirOut) / capCO2Air


# formula 2
def dxCO2Top(CO2Air, CO2Top):
    # TODO

    ######## Calculate MCAirTop ########
    fThScr = cal_fThScr(UThScr, KThScr, TAir, TTop, g, pAir, pTop)
```

```
    MCAirTop = cal_MCAirTop(fThScr, CO2Air, CO2Top)
    print("MCAirTop = ", MCAirTop)
    ######## Calculate MCTopOut ########
    # Calculate ppfVentRoofSide
    ppfVentRoofSide = cal_ppfVentRoofSide(Cd, AFlr, URoof, USide, ARoof, ASide, g, hSideRoof, TAi

    # Calculate ppfVentRoof
    ppfVentRoof = cal_ppfVentRoof(Cd, URoof, ARoof, AFlr, g, hVent, TAir, TOut, Cw, vWind)

    # Calculate fleakage
    fleakage = cal_fleakage(cleakage, vWind)

    # Calculate fVentRoof
    nInsScr = cal_nInsScr(sInsScr)
    fVentRoof = cal_fVentRoof(nInsScr, fleakage, UThScr, ppfVentRoofSide, nRoof, nSide, nRoof_Thr
    MCTopOut = cal_MCTopOut(fVentRoof, CO2Top, CO2Out)
    print("MCTopOut = ", MCTopOut)

    return (MCAirTop - MCTopOut) / capCO2Top
```

## 3.2 Bài 3: Chạy thử một số dữ liệu

```
print("dxCO2Air = ", dxCO2Air(CO2Air, CO2Top))
print("dxCO2Top = ", dxCO2Top(CO2Air, CO2Top))
```

- Với data đọc từ dòng 0 của data_CO2.xlsx trong thư mục data_in, ta có kết quả sau:
  Input line of data:
  0
  MCBlowAir = 0.20357142857142857
  MCExtAir = 4.628571428571429
  MCPadAir = 0.006727714285714286
  MCAirCan = -0.037749295986342614
  MCAirTop = 0.0
  MCAirOut = -0.04095714285714286
  dxCO2Air = 0.8195961683786762
  MCAirTop = 0.0
  MCTopOut = -0.027457624285367757
  dxCO2Top = 0.013728812142683879

- Với data đọc từ dòng 1 của data_CO2.xlsx trong thư mục data_in, ta có kết quả sau:
  Input line of data:
  1
  MCBlowAir = 1.8321428571428573
  MCExtAir = 4.628571428571429
  MCPadAir = 0.006727714285714286
  MCAirCan = -0.037749295986342614
  MCAirTop = 0.0
  MCAirOut = -0.04095714285714286
  dxCO2Air = 1.0910247398072477

MCAirTop = 0.0
MCTopOut = -0.027457624285367757
dxCO2Top = 0.013728812142683879

## 3.3 Bài 4

Source trích từ *source\code2_4.py*
**a)** Dựa trên cơ sở lý thuyết của phương phép Euler và Runge-Kutta bậc 4 đã nêu ở trên, chúng tôi đã tạo ra các Solver chạy bằng ngôn ngữ Python để dự đoán kết quả của mô hình.
Với các tham số đầu vào là *CO2Air_0*(giá trị tại thời điểm t của CO2Air), *CO2Top_0*(giá trị tại thời điểm t của CO2Top), *h*(kích thước bước sấp xỉ), *time*(thời điểm t cần tính giá trị của CO2Air và CO2Top)

*Note:* các biến như vận tốc gió(vWind), nhiệt độ(TAir, TTOP, TOut), URoof (tính thông qua ventWind và ventLee) thay đổi sau mỗi 5 phút nên cần cập nhật chúng trong quá trình tính toán . **b)** Với các Solver ở trên, ta tính được giá trị xấp xỉ của *CO2Air* và *CO2Top* tại thời điểm t đúng 5 phút, 10 phút, 15 phút,....

```
############### main ###############
```

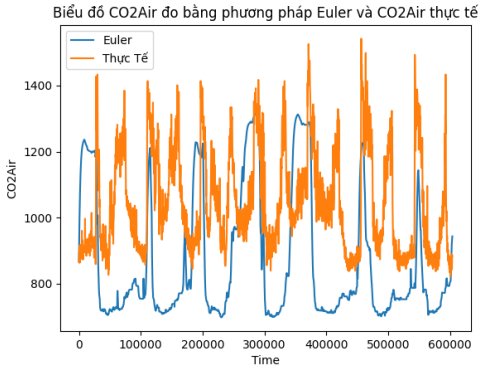
Các tham số đầu vào được đọc từ file *data_4.xslx*. Nhập kích thước bước sấp sỉ (h) là 1 (s) từ bàn phím. Nhập thời điểm t (time) là 603600 (s) tương ứng 1 tuần. Giá trị *CO2Air* và *CO2Top* tính được sau mỗi 300s sẽ được ghi vào file *Output_4.xslx*.

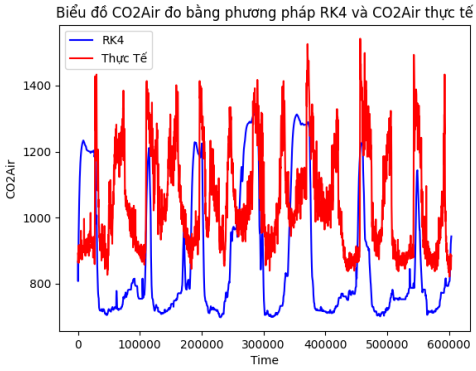
Với file *Output_4.xslx* ta tính độ chênh lệch của kết quả so với dữ liệu thật được lưu trong file *data_4.xslx* thông qua hàm *mse* (mean squared error) và vẽ đồ thị.

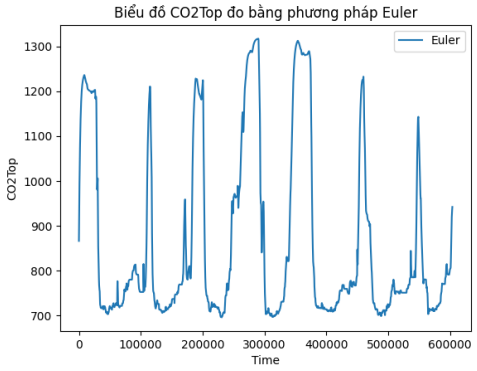
```
def mse(a):
    n = 0
    for idx in range(0, 2013):
        n += math.pow((float(df.at[idx, a]) - float(de.at[idx, 'CO2Air_Real'])),2)
    return n/2013

data = pd.read_excel("../data_out/Output_4.xlsx")
df = pd.DataFrame(data)
da = pd.read_excel("../data_in/data_4.xlsx")
de = pd.DataFrame(da)

print(mse('CO2Air_E'))
print(mse('CO2Air_RK4'))
```

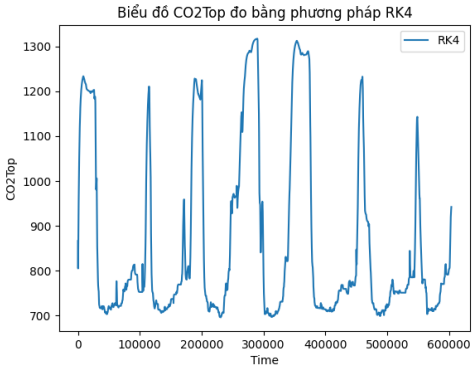
Kết quả là :
mse('CO2Air_E') = 80342.82022615442 (độ lệch của phương pháp Euler)
mse('CO2Air_RK4') = 80307.17721401696 (độ lệch của phương pháp Runge-Kutta 4)

Vì số liệu của các yếu tố bên trong nhà kính dùng để tính *CO2Air* và *CO2Top* không khớp với số liệu mà dữ liệu thật dùng để đo nên độ chính xác của mô hình không được cao.




## 3.4 Bài 5

### 3.4.1 Bài 5.2

#### 3.4.1.a Bài 5.2.a: Chi tiết mô hình đối với áp suất hơi nước $VP$ trong nhà kính được sử dụng trong đề tài này

Công thức biểu diễn sự thay đổi áp suất hơi nước của hơi nước như sau:

$$\begin{cases} \text{cap}_{VP_{Air}} \dot{VP_{Air}} = MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BLowAir} \\ \qquad\qquad -\text{MV}_{AirThScr} - MV_{AirTop} - MV_{AirOut\_Pad} - MV_{AirMech}, \\ \text{cap}_{VP_{Top}} \dot{VP_{Top}} = MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut} \end{cases}$$

Hệ số chuyển đổi áp suất hơi nước từ khối lượng nước trong lượng khí nằm phía dưới màn chắn nhiệt được mô tả bởi công thức:

$$cap_{VP_{Air}} = \frac{M_{Water} h_{Air}}{R(T_{Air} + 273.5)}$$

Trong đó:

- $M_{Water}$ là khối lượng mol của nước (kg kmol $^{-1}$)
- $h_{Air}$ là chiều cao từ nền tới màn chắn nhiệt (m)

- $R$ là hằng số khí lí tưởng (J kmol$^{-1}$K$^{-1}$)

Tương tự ta có công thức biểu diễn cho $cap_{VP_{Top}}$

$$cap_{VP_{Top}} = \frac{M_{Water} h_{Top}}{R(T_{Top} + 273.5)}$$

Sự thoát hơi nước của tán lá được miêu tả bởi công thức:

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

Trong đó:

- $VEC_{CanAir}$ là hệ số chuyển đổi hơi nước giữa tán lá và không khí (kg Pa s$^{-1}$)
- $VP_{Can}$ là áp suất hơi bão hòa ở nhiệt độ tán cây

Theo Stanghellini thì hệ số chuyển đổi hơi nước giữa tán lá và không khí được tính bởi

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

Trong đó:

- $\rho_{Air}$ là mật độ không khí của nhà kính (kg m$^{-3}$)
- $c_{p,Air}$ là nhiệt riêng của không khí trong nhà kính (J K$^{-1}$kg$^{-1}$)
- $LAI$ là chỉ số diện tích lá
- $\Delta H$ là nhiệt độ ngầm của sự bay hơi nước (J kg$^{-1}$)
- $\gamma$ là hệ số psychometric (PA K$^{-1}$)
- $r_b$ là sức cản của mặt trên tán lá đối với sự bay hơi (s m$^{-1}$)
- $r_s$ là sức cản của khí khổng đối với sự bay hơi (s m$^{-1}$)

Sức cản trở bay hơi của mặt trên lá được tính như sau:

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

Trong đó:

- $r_{s,min}$ là khả năng cản trở tối thiểu (s m$^{-1}$)
- $rf$ là hệ số kháng mức bức xạ cao, mức $CO_2$ cao và độ chênh lệch áp suất hơi nước cao.

Theo Stanghellinin thì hệ số kháng được tính như sau:

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

Trong đó:

- $R_{Can}$ là bức xạ toàn cầu trên tán cây(W m$^{-2}$)

- $\eta_{mg\_ppm}$ là hệ chuyển đổi từ mg $m^{-3}$ sang ppm
- $c_{evap1}$ là hệ số kháng cho mức $CO_2$ cao và bằng 1.5 ($Wm^{-2}$)
- $c_{evap2}$ là hệ số kháng cho độ chênh lệch áp suất hơi nước ở mức cao và bằng 5.8 ($Wm^{-2}$)

Tham số $c_{evap3}$ và $c_{evap4}$ có sự thay đổi giữa ban ban ngày và ban đêm. Vì thế ta dùng một hệ số chuyển đổi để làm cho công thức có thể phân biệt được giữa ban đêm và ban ngày

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

Trong đó:

- $S_{r_s}$ là giá trị của biến chuyển
- $s_{r_s}$ là độ dốc cuả biến chuyển trong mô hình kháng khí khổng (m $W^{-2}$)
- $R_{Can\_SP}$ là bức xạ phía trên tán lá để xác định ngày hay đêm ($Wm^{-2}$)

Dùng giá trị chuyển đổi, thông số thoát hơi nước sau khi được làm mịn được mô tả bằng

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

Tham số $c_{evap4}$ cũng được tính tương tự như trên.

5mm

Dưới đây là công thức để tính $MV_{BlowAir}$ là lượng hơi nước đi từ máy sưởi vào gian dưới nhà kính

$$MV_{BlowAir} = \frac{\eta_{HeatVap} U_{Blow} P_{Blow}}{A_{Flr}}$$

Trong đó:

- $\eta_{HeatVap}$ là lượng hơi nước được tạo ra khi 1 Juole năng lượng được tiêu thụ hoàn toàn bởi máy sưởi (kg vapour $J^{-1}$).

Lượng hơi nước từ quạt gió vào gian dưới được tính bởi:

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

Trong đó:

- $f_{Air}$ là thông lượng gió của hệ thống quạt ($m^3 m^{-2} s^{-1}$)
- $\rho_{Pad}$ là độ mở của hệ thống quạt
- $x_{Pad}$ là tỉ lệ hơi nước trong không khí của cửa vào hệ thống quạt (kg water $kg^{-1}$ air)
- $x_{Out}$ là tỉ lệ hơi nước trong không khí của cửa ra hệ thống quạt (kg water $kg^{-1}$ air)

Dòng thông gió của hệ thống quạt được mô tả bằng

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

Trong đó:

- $U_{Pad}$ là độ mở của hệ thống quạt
- $\phi_{Pad}$ là sức chứa thông lượng của khí đi qua hệ thống quạt ($m^3s^{-1}$)

Lượng hơi nước từ nhà kính ra không khí được tính bởi công thức:

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

Lượng hơi nước từ hệ thông sương mù vào khoang dưới nhà kính phụ thuộc vào thông lượng hơi từ hệ thống sương mù đến không khí nhà kính được mô tả bằng:

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

Trong đó:

- $U_{Fog}$ là độ mở của hệ thống sương mù
- $\phi_{Fog}$

#### 3.4.1.b Bài 5.2.b: Code tính toán các hàm cần thiết

**Dữ liệu được đọc từ file** *data_in\data_VP.xlsx*
Source trích từ file *source\code5_2b.py*

```
import math
from math import sqrt
import pandas as pd

# f''VentRoof
def cal_ppfVentRoof(Cd, URoof, ARoof, AFlr, g, hVent, TAir, TOut, Cw, vWind):
    TMeanAir = (TAir + TOut) / 2
    part1 = Cd * URoof * ARoof / (2 * AFlr)
    part2 = g * hVent * (TAir - TOut) / 2 / TMeanAir + Cw * pow(vWind, 2)
    return part1 * sqrt(part2)

# formula 1
def cal_MVCanAir(VECCanAir, VPCan, VPAir):
    return VECCanAir * (VPCan - VPAir)

# formula 2
def cal_VECCanAir(pAir, c_p_Air, LAI, delta_H, y, rb, rs): #y is gamma
    return 2 * pAir * c_p_Air * LAI / (delta_H * y *(rb + rs))
def cal_rs(VPCan, VPAir, R_Can, CO2Air):
    rs_min = 82
    c_evap1 = 4.3
    c_evap2 = 0.54
    rf_RCan = (R_Can+c_evap1)/(R_Can+c_evap2)
    R_Can_SP = 5
```

```
    S_rs = 1/(1+math.exp(-1*(R_Can-R_Can_SP)))
    c_night_evap3 = 1.1*pow(10,-11)
    c_night_evap4 = 5.2*pow(10,-6)
    c_evap3 = c_night_evap3*(1-S_rs)+c_night_evap3*S_rs
    c_evap4 = c_night_evap4*(1-S_rs)+c_night_evap4*S_rs
    n_mg_ppm = 0.554
    rf_CO2Air = 1+c_evap3*math.pow(n_mg_ppm*CO2Air-200, 2)
    rf_VPCan_VPAir = 1+c_evap4*pow(VPCan - VPAir, 2)
    return rs_min*rf_RCan*rf_CO2Air*rf_VPCan_VPAir
# formula 3
def cal_MVPadAir(pAir, fPad, nPad, xPad, xOut):
    return pAir * fPad *(nPad * (xPad - xOut) + xOut)
def cal_fPad(UPad, phiPad, AFlr): # use for formula 3 & formula 10
    return UPad * phiPad / AFlr

# formula 4
def cal_MVFogAir(UFog, phiFog, AFlr):
    return UFog * phiFog / AFlr

# formula 5
def cal_MVBlowAir(nHeatVap, UBlow, PBlow, AFlr):
    return nHeatVap * UBlow * PBlow / AFlr

# formula 6
def cal_MVAirThScr(HECAirThScr, VPAir, VPThScr):
    if VPAir <= VPThScr:
        return 0
    else:
        return 6.4 * pow(10, -9) * HECAirThScr * (VPAir - VPThScr)
def cal_HECAirThScr(UThScr, TAir, TThScr): # use for formula 7
    return 1.7 * UThScr * pow(abs(TAir - TThScr), 0.33)

# formula 7
def cal_MVAirTop(MWater, R, fThScr, VPAir, VPTop, TAir, TTop):
    return (MWater / R) * fThScr * (VPAir / TAir - VPTop / TTop)
def cal_fThScr(UThScr, KThScr, TAir, TTop, g, pAir, pTop): # use for formula 6
    a = UThScr * KThScr * pow(abs(TAir - TTop), 2/3)
    PMean_Air = (pAir + pTop) / 2
    b = (1 - UThScr) * pow(g * (1 - UThScr) * abs(pAir - pTop) / (2 * PMean_Air), 1/2)
    return a + b

# formula 8
def cal_MVAirOut(MWater, R, fVentSide, fVentForced, VPAir, VPOut, TAir, TOut):
    return (MWater / R) * (fVentSide + fVentForced) * (VPAir / TAir - VPOut / TOut)
def cal_fVentSide(nInsScr, ppfVentSide, fleakage, UThScr, ppfVentRoofSide, nSide, nSide_Thr):  #
    if nSide >= nSide_Thr:
        return nInsScr * ppfVentSide + 0.5 * fleakage
    else:
        return nInsScr * (UThScr * ppfVentSide + (1 - UThScr) * ppfVentRoofSide * nSide) + 0.5 *
```

```
def cal_ppfVentSide(Cd, USide, ASide, vWind, AFlr, Cw):  # use for fVentSide
    return Cd * USide * ASide * vWind * sqrt(Cw) / (2 * AFlr)
def cal_ppfVentRoofSide(Cd, AFlr, URoof, USide, ARoof, ASide, g, hSideRoof, TAir, TOut, Cw, vWind
    a = Cd / AFlr
    b = pow(URoof * USide * ARoof * ASide, 2) / (pow(URoof * ARoof, 2) + pow(USide * ASide, 2))
    TMean_Air = (TAir + TOut) / 2
    c = 2 * g * hSideRoof * (TAir - TOut) / TMean_Air
    _d = (URoof * ARoof + USide * ASide) / 2
    d = pow(_d, 2) * Cw * pow(vWind, 2)
    return a * sqrt(b * c + d)
def cal_fVentForced(nInsScr, UVentForced, phiVentForced, AFlr):  # use for formula 8
    return nInsScr * UVentForced * phiVentForced / AFlr
def cal_nInsScr(sInsScr):
    return sInsScr * (2 - sInsScr)
def cal_fleakage(cleakage, vWind):
    if vWind < 0.25:
        return 0.25 * cleakage
    else:
        return vWind * cleakage

# formula 9
def cal_MVTopOut(MWater, R, fVentRoof, VPTop, VPOut, TTop, TOut):
    return (MWater / R) * fVentRoof * (VPTop / TTop - VPOut / TOut)
def cal_fVentRoof(nInsScr, fleakage, UThScr, ppfVentRoofSide, nRoof, nSide, nRoof_Thr, ppfVentRoo
    # nRoof_Thr la nguong Stack
    if nRoof >= nRoof_Thr:
        return nInsScr * ppfVentRoof + 0.5 * fleakage
    else:
        return nInsScr * (UThScr * ppfVentRoof + (1 - UThScr) * ppfVentRoofSide * nSide) + 0.5 *

# formula 10
def cal_MVAirOut_Pad(fPad, MWater, R, VPAir, TAir):
    return fPad * MWater / R * VPAir / TAir

# formula 11
def cal_MVAirMech(HECAirMech, VPAir, VPMech):
    if VPAir <= VPMech:
        return 0
    else:
        return 6.4 * pow(10, -9) * HECAirMech * (VPAir - VPMech)
def cal_HECAirMech(UMechCool, COPMechCool, PMechCool, AFlr, TAir, TMechCool, delta_H, VPAir, VPMe
    A1 = UMechCool * COPMechCool * PMechCool / AFlr
    A2 = -(TAir - TMechCool + 6.4 * pow(10, -9) * delta_H * (VPAir -VPMechCool))
    return A1 / A2

# formula 12
def cal_MVTopCov_in(HECTopCov_in, VPTop, VPCov_in):
    if VPTop <= VPCov_in:
        return 0
```

```
    else:
        return 6.4 * pow(10, -9) * HECTopCov_in * (VPTop - VPCov_in)
def cal_HECTopCov_in(cHECin, TTop, TCov_in, ACov, AFlr): #use for formula 12
    return cHECin * pow(TTop - TCov_in, 0.33) * ACov / AFlr

##########   START READING DATA   ##########
i = int(input())

data = pd.read_excel("../data_in/data_VP.xlsx")
df = pd.DataFrame(data)

rb = 275
CO2Air = float(df.at[i, 'CO2Air'])
pAir = float(df.at[i, 'pAir'])
LAI = float(df.at[i, 'LAI'])
VPCan = float(df.at[i, 'VPCan'])
R_Can = float(df.at[i, 'RCan'])
delta_H = 2450000
y = 65.8
c_p_Air = 1000

UPad = float(df.at[i, 'UPad'])
phiPad = float(df.at[i, 'phiPad'])
AFlr = float(df.at[i, 'AFlr'])
fPad = UPad*phiPad/AFlr
nPad = float(df.at[i, 'nPad'])
xPad = float(df.at[i, 'xPad'])
xOut = float(df.at[i, 'xOut'])

UFog = float(df.at[i, 'UFog'])
phiFog = 1.39

nHeatVap = 4.43*pow(10,-8)
UBlow = float(df.at[i, 'UBlow'])
PBlow = float(df.at[i, 'PBlow'])

UThScr = float(df.at[i, 'UThScr'])
TAir = float(df.at[i, 'TAir'])
TThScr = float(df.at[i, 'TThScr'])
HECAirThScr = 1.7*UThScr*pow(abs(TAir - TThScr), 0.33)
VPThScr = float(df.at[i, 'VPThScr'])

MWater = float(df.at[i, 'MWater'])
R = float(df.at[i, 'R'])
KThScr = float(df.at[i, 'KThScr'])
TOut = float(df.at[i, 'TOut'])
p_Mean_Air = float(df.at[i, 'p_Mean_Air'])
pOut = float(df.at[i, 'pOut'])
g = float(df.at[i, 'g'])
```

```
fThScr = UThScr*KThScr*pow(abs(TAir - TOut), 0.66)+(1-UThScr)/p_Mean_Air*pow((0.5*p_Mean_Air*(1-U
TTop = float(df.at[i, 'TTop'])
sInsScr = float(df.at[i, 'sInsScr'])
nInsScr = cal_nInsScr(sInsScr)
Cd = float(df.at[i, 'Cd'])
USide = float(df.at[i, 'USide'])
ASide = float(df.at[i, 'ASide'])
vWind = float(df.at[i, 'vWind'])
Cw = float(df.at[i, 'Cw'])
URoof = float(df.at[i, 'URoof'])
ARoof = float(df.at[i, 'ARoof'])
hSideRoof = float(df.at[i, 'hSideRoof'])
ppfVentSide = cal_ppfVentSide(Cd,USide,ASide,vWind,AFlr,Cw)
cleakage = float(df.at[i, 'cleakage'])
fleakage = cal_fleakage(cleakage, vWind)
ppfVentRoofSide = cal_ppfVentRoofSide(Cd,AFlr,URoof,USide,ARoof,ASide,g,hSideRoof,TAir,TOut,Cw,vW
nSide = float(df.at[i, 'nSide'])
nSide_Thr = float(df.at[i, 'nSide_Thr'])
fVentSide = cal_fVentSide(nInsScr, ppfVentSide, fleakage, UThScr, ppfVentRoofSide, nSide, nSide_T

UVentForced = float(df.at[i, 'UVentForced'])
phiVentForced = float(df.at[i, 'phiVentForced'])
fVentForced = cal_fVentForced(nInsScr, UVentForced, phiVentForced, AFlr)
UMechCool = float(df.at[i, 'UMechCool'])
COPMechCool = float(df.at[i, 'COPMechCool'])
PMechCool = float(df.at[i, 'PMechCool'])
TMechCool = float(df.at[i, 'TMechCool'])
VPAir = float(df.at[i, 'VPAir'])
VPTop = float(df.at[i, 'VPTop'])
VPOut = float(df.at[i, 'VPOut'])
VPMechCool = float(df.at[i, 'VPMechCool'])
HECAirMech = cal_HECAirMech(UMechCool,COPMechCool,PMechCool,AFlr,TAir,TMechCool,delta_H,VPAir,VPM
VPMech = float(df.at[i, 'VPMech'])
capVPAir = float(df.at[i, 'capVPAir'])
cHECin = float(df.at[i, 'cHECin'])
TCov_in = float(df.at[i, 'TCov_in'])
ACov = float(df.at[i, 'ACov'])
nRoof = float(df.at[i, 'nRoof'])
nRoof_Thr = float(df.at[i, 'nRoof_Thr'])
hVent = float(df.at[i, 'hVent'])
VPCov_in = float(df.at[i, 'VPCov_in'])
capVPTop = float(df.at[i, 'capVPTop'])
##########   END READING DATA   ##########

###### Calculate dx ######
def dxVPAir(VPAir, VPTop):
    rs = cal_rs(VPCan, VPAir, R_Can, CO2Air)
    VECCanAir = cal_VECCanAir(pAir,c_p_Air,LAI,delta_H,y,rb,rs)
    MVCanAir = cal_MVCanAir(VECCanAir, VPCan, VPAir)
```

```
    print("MVCanAir = ", MVCanAir)
    MVPadAir = cal_MVPadAir(pAir,fPad,nPad,xPad,xOut)
    print("MVPadAir = ", MVPadAir)
    MVFogAir = cal_MVFogAir(UFog, phiFog, AFlr)
    print("MVFogAir = ", MVFogAir)
    MVAirTop = cal_MVAirTop(MWater,R,fThScr, VPAir, VPTop,TAir,TTop)
    print("MVAirTop = ", MVAirTop)
    MVAirThScr = cal_MVAirThScr(HECAirThScr, VPAir, VPThScr)
    print("MVAirThScr = ", MVAirThScr)
    MVBlowAir = cal_MVBlowAir(nHeatVap, UBlow, PBlow, AFlr)
    print("MVBlowAir = ", MVBlowAir)
    MVAirOut = cal_MVAirOut(MWater,R,fVentSide,fVentForced,VPAir,VPOut,TAir,TOut)
    print("MVAirOut = ", MVAirOut)
    MVAirOut_Pad = cal_MVAirOut_Pad(fPad,MWater,R,VPAir,TAir)
    print("MVAirOut_Pad = ", MVAirOut_Pad)
    MVAirMech = cal_MVAirMech(HECAirMech,VPAir,VPMech)
    print("MVAirMech = ", MVAirMech)
    return (MVCanAir+MVPadAir+MVFogAir+MVBlowAir-MVAirThScr-MVAirTop-MVAirOut-MVAirOut_Pad-MVAirM

def dxVPTop(VPAir, VPTop):
    ppfVentRoof = cal_ppfVentRoof(Cd,URoof,ARoof,AFlr,g,hVent,TAir,TOut,Cw,vWind)
    fVentRoof = cal_fVentRoof(nInsScr,fleakage,UThScr,ppfVentRoofSide,nRoof,nSide,nRoof_Thr,ppfVe
    MVTopOut = cal_MVTopOut(MWater,R,fVentRoof,VPTop,VPOut,TTop,TOut)
    print("MVTopOut = ", MVTopOut)
    MVAirTop = cal_MVAirTop(MWater,R,fThScr, VPAir, VPTop,TAir,TTop)
    print("MVAirTop = ", MVAirTop)
    HECTopCov_in = cal_HECTopCov_in(cHECin,TTop,TCov_in,ACov,AFlr)
    MVTopCov_in = cal_MVTopCov_in(HECTopCov_in,VPTop,VPCov_in)
    print("MVTopCov_in = ", MVTopCov_in)

    return (MVAirTop-MVTopCov_in-MVTopOut)/capVPTop
```

### 3.4.2 Bài 5.3: Chạy thử một số dữ liệu

```
print("dxCO2Air = ", dxCO2Air(CO2Air, CO2Top))
print("dxCO2Top = ", dxCO2Top(CO2Air, CO2Top))
```

- Với data đọc từ dòng 0 của data_VP.xlsx trong thư mục data_in, ta có kết quả sau:
  MVCanAir = 4.3319155181017255e-07
  MVPadAir = 0.0
  MVFogAir = 4.964285714285714e-05
  MVAirTop = 5.9285343200552286e-05
  MVAirThScr = 4.602078262871478e-10
  MVBlowAir = 0.0
  MVAirOut = -4.861513877462847e-07
  MVAirOut_Pad = 4.561991328414344e-06
  MVAirMech = 0
  dxVPAir = -2.2142657757298863e-06

  MVTopOut = -1.1827325860913288e-07

MVAirTop = 5.9285343200552286e-05
MVTopCov_in = 0
dxVPTop = 2.970180822958071e-05

### 3.4.3 Bài 5.4

Source trích từ *source\code5.py*
**a)** Với các tham số đầu vào là $VPAir\_0$(giá trị tại thời điểm t của VPAir), $VPTop\_0$(giá trị tại thời điểm t của VPTop), $h$(kích thước bước sấp xỉ), $time$(thời điểm t cần tính giá trị của VPAir và VPTop)

```
def euler(VPAir, VPTop, h, time):
    n = int(time/h)
    VPAir_0 = VPAir
    VPTop_0 = VPTop

    for idx in range(1, n + 1):
        k = h * dxVPAir(VPAir_0, VPTop_0)
        t = h * dxVPTop(VPAir_0, VPTop_0)

        VPAir_0 += k
        VPTop_0 += t

        if idx % 300 == 0:
            row = idx // 300
            worksheet.write(row+1, 0, idx)
            worksheet.write(row + 1, 1, VPAir_0)
            worksheet.write(row + 1, 2, VPTop_0)
            global TAir, TTop, TOut, vWind, URoof, CO2Air
            TAir = float(df.at[row, 'TAir'])
            TTop= float(df.at[row, 'TTop'])
            TOut = float(df.at[row, 'TOut'])
            vWind = float(df.at[row, 'vWind'])
            URoof= float(df.at[row, 'URoof'])
            CO2Air = float(df.at[row, 'CO2Air'])

    return VPAir_0, VPTop_0


# Explicit Runge-Kutta 4th order
def rk4(VPAir, VPTop, h, time):
    n = int(time/h)
    VPAir_0 = VPAir
    VPTop_0 = VPTop

    for idx in range(1, n + 1):
        k1 = h * dxVPAir(VPAir_0, VPTop_0)
        t1 = h * dxVPTop(VPAir_0, VPTop_0)
        k2 = h * dxVPAir(VPAir_0 + 0.5 * k1, VPTop_0 + 0.5 * k1)
        t2 = h * dxVPTop(VPAir_0 + 0.5 * t1, VPTop_0 + 0.5 * t1)
```

```
        k3 = h * dxVPAir(VPAir_0 + 0.5 * k2, VPTop_0 + 0.5 * k2)
        t3 = h * dxVPTop(VPAir_0 + 0.5 * t2, VPTop_0 + 0.5 * t2)
        k4 = h * dxVPAir(VPAir_0 + k3, VPTop_0 + k3)
        t4 = h * dxVPTop(VPAir_0 + t3, VPTop_0 + t3)

        VPAir_0 += (1.0 / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4)
        VPTop_0 += (1.0 / 6.0) * (t1 + 2 * t2 + 2 * t3 + t4)

        if idx % 300 == 0:
            row = idx // 300
            worksheet.write(row + 1, 4, VPAir_0)
            worksheet.write(row + 1, 5, VPTop_0)
            global TAir, TTop, TOut, vWind, URoof, CO2Air
            TAir = float(df.at[row, 'TAir'])
            TTop= float(df.at[row, 'TTop'])
            TOut = float(df.at[row, 'TOut'])
            vWind = float(df.at[row, 'vWind'])
            URoof= float(df.at[row, 'URoof'])
            CO2Air = float(df.at[row, 'CO2Air'])

    return VPAir_0, VPTop_0
```

*Note:* các biến như vận tốc gió(vWind), nhiệt độ(TAir, TTOP, TOut), URoof (tính thông qua ventWind và ventLee), CO2Air thay đổi sau mỗi 5 phút nên cần cập nhật chúng trong quá trình tính toán .

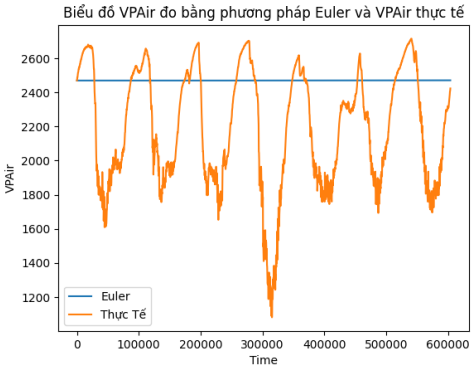
**b)** Với các Solver ở trên, ta tính được giá trị xấp xỉ của $VPAir$ và $VPTop$ tại thời điểm t đúng 5 phút, 10 phút,.....
Các tham số đầu vào được đọc từ file $data_in\backslash data\_5.xlsx$. Nhập kích thước bước sấp sỉ (h) là 1 (s) từ bàn phím. Nhập thời điểm t (time) là 603600 (s) tương ứng 1 tuần. Giá trị $VPAir$ và $VPTop$ tính được sau mỗi 300s sẽ được ghi vào file $Output\_5.xslx$.

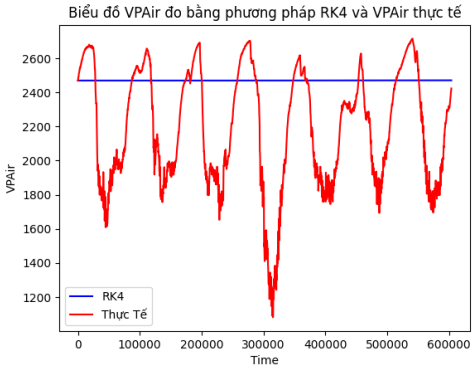
Với file $Output\_5.xslx$ ta tính độ chênh lệch của kết quả so với dữ liệu thật được lưu trong file $data\_5.xslx$ thông qua hàm $mse$ (mean squared error).

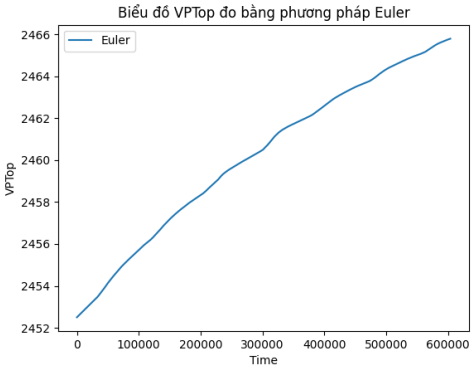
```
  def mse(a):
    n = 0
    for idx in range(0, 2013):
        n += math.pow((float(df.at[idx, a]) - float(de.at[idx, 'VPAir_Real'])),2)
    return n/2013

data = pd.read_excel("../data_out/Output_5.xlsx")
df = pd.DataFrame(data)
da = pd.read_excel("../data_in/data_5.xlsx")
de = pd.DataFrame(da)
print(mse('VPAir_euler'))
print(mse('VPAir_rk4'))
```

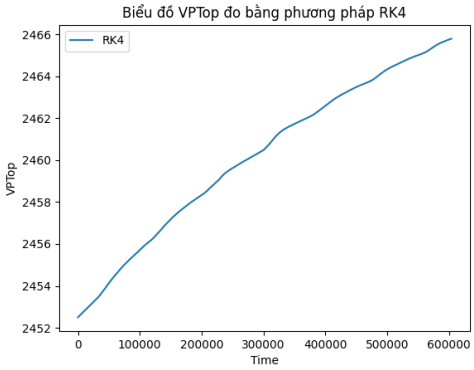
Kết quả là :
mse('VPAir_E') = 201403.0585496551 (độ lệch của phương pháp Euler)

mse('VPAir_RK4') = 201403.0854646318 (độ lệch của phương pháp Runge-Kutta 4)



# References

[1] **Link source code của nhóm:** https://github.com/phucvinh57/Mathematical_Modelling_Assignment