ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

MÔ HÌNH HOÁ TOÁN HỌC (CO2011)

Báo cáo bài tập lớn

Mô hình Hệ động lực dự báo tiểu khí hậu nhà kính

GVHD: Nguyễn Tiến Thịnh
Nguyễn An Khương
Trần Trung Hiếu
Sinh viên: Thái Văn Nhật - 1813381
Nguyễn Trần Quang Minh - 1811083
Đoàn Tấn Lộc - 1812962
Văn Chấn Dương - 1811824

Tp.Hồ Chí Minh, Tháng 12/2020

Mục lục

Danh sách thành viên & Phân chia công việc

No.	Họ và tên	MSSV	Phần việc được phân công	Phần trăm hoàn thành công việc
1.	Thái Văn Nhật	1813381	Hiện thực các hàm **dx** và các hàm **rk4**, **euler**. Viết báo cáo liên quan đến việc kiểm thử mô hình.	25%
2.	Nguyễn Trần Quang Minh	1811083	Chuẩn bị nội dung phần áp suất hơi nước. Viết báo cáo liên quan đến nội dung và việc kiểm thử mô hình.	25%
3.	Văn Chấn Dương	1811824	Chuẩn bị nội dung cho phần kiến thức chuẩn bị, nội dung phần nồng độ CO_2. Viết các hàm hiện thực các phương trình liên quan đến mô hình nồng độ CO_2.	25%
4.	Đoàn Tấn Lộc	1812962	Viết các hàm hiện thực các phương trình liên quan đến mô hình áp suất hơi nước.	25%

File log chi tiết về tiến độ công việc và các nội dung thảo luận được đính kèm cùng với bài báo cáo này.

1 Kiến thức chuẩn bị

1.1 Phương trình vi phân thường

1.1.1 Định nghĩa

Phương trình vi phân là một phương trình liên quan tới một hàm và đạo hàm của hàm đó, mục tiêu của phương trình là để tìm một hàm thoả phương trình đã nêu. Dạng tổng quát như sau:

Giả sử ta được cho hàm $\phi(y, z)$ có hai biến với phương trình

$$\phi(u(x), u'(x)) = 0 \tag{1}$$

Mục tiêu ta cần tìm hàm $y = u(x)$ trên một miền thuộc trục x hay ta có thể nói là tìm hàm $u(x)$ với biến x (biến độc lập) sao cho phương trình (1) trở nên đồng nhất trên một miền nào đó thuộc trục x.

Từ "thường" ở đây được dùng để chỉ các phương trình vi phân có một biến độc lập, so với các phương trình có nhiều hơn một biến độc lập.

1.1.2 Dạng bậc 1, tổng quát

Có một dạng tổng quát hơn của phương trình vi phân so với phương trình (1). Giả sử $\phi = \phi(t, x, y)$ là một hàm được định nghĩa sẵn với 3 biến và ta cần tìm hàm $x(t)$ sao cho

$$\phi(t, x(t), \dot{x}(t)) = 0 \tag{2}$$

trên một miền thuộc trục t. Dạng tổng quát của phương trình này là

$$\frac{dx}{dt} = f(t, x) \tag{3}$$

Trong đó hàm f có được khi ta giải phương trình $\phi(t, x, y) = 0$ với y là một hàm theo t và x. Có một họ các phương trình vi phân có nghiệm ta có thể giải được trực tiếp. Giả sử hàm f trong phương trình (3) không phụ thuộc vào x: $f = f(t)$. Từ đây ta có thể thấy nghiệm của phương trình trên là

$$x(t) = \int^t f(s)ds + C \tag{4}$$

với C là một hằng số tuỳ ý.
Vậy để tìm được nghiệm ta chỉ cần thực hiện nguyên hàm một hàm đã biết.

1.2 Bài toán liên quan đến diều kiện ban đầu (Initial value problem - IVP)

Nghiệm giải bằng cách nguyên hàm một hàm đã biết sẽ bao gồm 1 hằng số C tuỳ ý và vì vậy nghiệm này không chỉ bao gồm 1 mà gồm 1 họ các nghiệm, mỗi nghiệm ứng với 1 giá trị của hằng số C. Ta có thể xác định nghiệm duy nhất của hệ phương trình nếu có thể xác định được cụ thể giá trị của hằng số C. Một cách để thực hiện việc này là ràng buộc hàm $x(t)$ sao cho $x(t)$ không chỉ thoả điều kiện của phương trình vi phân mà còn phải thoả điều kiện ban đầu x_0 tại một giá trị t_0 cho trước. Ta định nghĩa *bài toán điều kiện ban đầu* như sau:

$$\frac{dx}{dt} = f(t, x), x|_{t=t_0} = x_0 \tag{5}$$

Ở đây hàm f và điều kiện ban đầu t_0, x_0 được cho trước.

1.2.1 Điều kiện tồn tại và duy nhất nghiệm

1.2.1.a Điều kiện Lipschitz

Phương trình vi phân được xác định bởi các tính chất của hàm ở vế phải của phương trình

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

Ta cần phải biết được một số tính chất của hàm này để có thể đưa ra các kết luận liên quan đến nghiệm của phương trình. Các hàm $f(x, y)$ tối thiểu phải liên tục và thường các đạo hàm của chúng cũng là hàm liên tục. Ta giả thuyết rằng hàm f thoả *điều kiện Lipschitz* trên một miền thuộc mặt phẳng xy. Với hàm $f(y)$ không phụ thuộc vào biến độc lập x, ta định nghĩa điều kiện Lipschitz như sau:

Định nghĩa 1 *Hàm f được định nghĩa bởi y trên miền I thoả điều kiện Lipschitz nếu tồn tại một hằng số dương L sao cho*

$$|f(y_1) - f(y_2)| \leq L|y_1 - y_2| \tag{6}$$

với mọi y_1, y_2 *thuộc* I

Việc tổng quát hoá điều kiện Lipschitz là cần thiết cho các định lý về sự tồn tại và tồn tại duy nhất nghiệm được xây dựng dựa trên biến phụ thuộc y, hay còn được gọi là "điều kiện Lipschitz đối với y".

Ta có định nghĩa điều kiện Lipschitz cho hàm phụ thuộc vào cả hai biến x và y như sau:

Định nghĩa 2 *Hàm* f *thoả điều kiện Lipschitz đối với* y *thuộc miền* D *của mặt phẳng* xy *nếu tồn tại một hằng số dương L sao cho*

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2| \tag{7}$$

với mỗi cặp (x, y_1), (x, y_2) *thuộc* D.

Có một điều kiện đủ để hàm f thoả điều kiện Lipschitz đối với y trong miền D thuộc mặt phẳng xy. Một trong những yêu cầu của điều kiện này đó là miền D là một miền lồi.

Ta có định lý sau:

Định lý 1 *Giả sử hàm* f *và đạo hàm riêng phần đầu tiên của* f *là hàm liên tục trên miền lồi kín, bị giới hạn* D *thuộc mặt phẳng* xy. *Khi đó* f *thoả điều kiện Lipschitz đối với* y *ở miền ấy.*

Ta có định lý về tính tồn tại duy nhất trong bài toán liên quan đến điều kiện ban đầu:

Định lý 2 *Giả sử* $y = \varphi(x)$ *là một nghiệm của bài toán điều kiện ban đầu*

$$\frac{dy}{dx} = f(x, y), y|_{x=x_0} = y_0 \tag{8}$$

trên khoảng $c < x < b$, *với đồ thị của* $(x, \varphi(x))$ *nằm trong miền* D *thuộc mặt phẳng* xy *trong đó* f *là hàm liên tục và thoả điều kiện Lipschitz đối với* y. *Khi đó* φ *là nghiệm duy nhất.*

1.2.1.b Định lý về sự tồn tại nghiệm

Định lý về sự tồn tại duy nhất nghiệm (Định lý 2) khẳng định rằng không thể có nhiều hơn một nghiệm cho bài toán điều kiện ban đầu nhưng không khẳng định rằng là có tồn tại nghiệm. Thực chất, dưới những ràng buộc như ở định lý về sự tồn tại duy nhất nghiệm, phương trình (8) thực sự tồn tại nghiệm.

Định lý 3 *Giả sử hàm f của phương trình (8) là hàm liên tục và thoả điều kiện Lipschitz đối với y trên một miền D của mặt phẳng xy chứa điểm có toạ độ (x_0, y_0). Khi đó tồn tại một khoảng $a < x < b$ chứa x_0 mà tại đó nghiệm φ của (8) tồn tại; đồ thị $(x, \varphi(x))$ nằm trong D trên khoảng này.*

1.3 Hệ phương trình vi phân bậc nhất

1.3.1 Định nghĩa

Hệ phương trình vi phân thường bậc nhất có dạng:

$$\begin{cases} \dot{x}_1 &= f(x_1, x_2, ..., x_n, t) \\ \dot{x}_2 &= f(x_1, x_2, ..., x_n, t) \\ &\vdots \\ \dot{x}_n &= f(x_1, x_2, ..., x_n, t) \end{cases} \tag{9}$$

với t có giá trị thuộc khoảng I. Hệ phương trình vi phân bậc nhất với bài toán điều kiện ban đầu có dạng tương tự (9) và bổ sung thêm các điều kiện ban đầu $x_1(a) = c_1$, $x_2(a) = c_2$,..., $x_n(a) = c_n$.

Nghiệm của hệ phương trình vi phân bậc nhất là tập các hàm $x_1 = f_1(t)$, $x_2 = f_2(t)$,..., $x_n = f_n(t)$ thoả tất cả các phương trình của hệ. Nghiệm của hệ phương trình vi phân bậc nhất với bài toán điều kiện ban đầu ngoài ra còn phải thoả thêm các điều kiện ban đầu.

1.3.2 Điều kiện tồn tại và tồn tại duy nhất nghiệm

Vì đây là hệ các phương trình vi phân thường, dựa trên các định lý 2 và 3, thì hệ phương trình vi phân thường bậc nhất với bài toán điều kiện ban đầu cũng tồn tại nghiệm và nghiệm này là duy nhất. Tuy nhiên nghiệm này chỉ tồn tại trên một miền nhỏ xung quanh a.

1.3.3 Hệ tuyến tính

Hệ phương trình vi phân tuyến tính bậc nhất là hệ phương trình vi phân bậc nhất dưới dạng:

$$\begin{cases} \dot{x}_1 = a_{11}(t)x_1(t) + a_{12}(t)x_2(t) + ... + a_{1n}(t)x_n(t) + b_1(t) \\ \dot{x}_2 = a_{21}(t)x_1(t) + a_{22}(t)x_2(t) + ... + a_{2n}(t)x_n(t) + b_2(t) \\ \vdots \qquad\qquad\qquad \vdots \\ \dot{x}_n = a_{n1}(t)x_1(t) + a_{n2}(t)x_2(t) + ... + a_{nn}(t)x_n(t) + b_n(t) \end{cases} \tag{10}$$

1.3.4 Biểu diễn dạng ma trận cho hệ tuyến tính

Ma trận hệ số của hệ (10) gồm:

$$A(t) = \begin{bmatrix} a_{11}(t) & a_{12}(t) & \cdots & a_{1n}(t) \\ a_{21}(t) & a_{22}(t) & \cdots & a_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(t) & a_{n2}(t) & \cdots & a_{nn}(t) \end{bmatrix}, \vec{b}(t) = \begin{bmatrix} b_1(t) \\ b_2(t) \\ \vdots \\ b_n(t) \end{bmatrix}, \vec{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \vec{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix}$$

Khi đó hệ phương trình có thể được viết dưới dạng ma trận như sau:

$$\vec{x} = A(t)\vec{x} + \vec{b}(t) \tag{11}$$

Tại đây $A(t)$ là một ma trận với các phần tử chỉ phụ thuộc vào t và $\vec{b}(t)$ là vector cột với các phần tử của nó chỉ phụ thuộc vào t.

Vector $\vec{x}$ ở phương trình (11) phụ thuộc vào biến t nên được gọi là một *vector hàm*, tương tự cho $\vec{b}(t)$. Hệ (11) được gọi là *thuần nhất* nếu $\vec{b}(t) = \vec{0}$.

Bài toán điều kiện ban đầu cho hệ (11) là tìm vector hàm $\vec{x}(t) \in C^1$ thoả hệ trên một khoảng I và các điều kiện ban đầu $\vec{x}(t_0) = x_0 = (x_1(t_0), x_2(t_0), ..., x_n(t_0))^T$ trong đó $t_0 \in I$ và $x_0 \in \mathbb{R}^n$.

Ta có định lý về sự tồn tại và tồn tại duy nhất nghiệm cho hệ (11) như sau:

Định lý 4 *Giả sử $A(t)$ và $\vec{b}(t)$ là liên tục trên I và $t_0 \in I$. Khi đó, với mỗi $\vec{x}(t_0) = x_0 = (x_1(t_0), x_2(t_0), ..., x_n(t_0))^T \in \mathbb{R}^n$ tồn tại nghiệm duy nhất $\vec{x}(t)$ cho bài toán điều kiện ban đầu*

$$\vec{x} = A(t)\vec{x} + \vec{b}(t), \vec{x}(t_0) = x_0$$

trên toàn bộ miền I

1.3.5 Hệ phương trình vi phân bậc một thuần nhất

Ở đây ta xét hệ

$$\vec{x} = A(t)\vec{x} \tag{H}$$

Wronskian của n n-thành phần vector hàm $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$, ký hiệu là $W(\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n)(t)$ hay $W(t)$ hay $W(\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n)$ là định thức của ma trận $[\vec{u}_1 : \vec{u}_2 : \cdots : \vec{u}_n]$

n nghiệm $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$ độc lập tuyến tính với nhau của (H) được gọi là một *tập nghiệm cơ bản* của (H). Khi đó ma trận $X(t) = [\vec{u_1} : \vec{u_2} : ... : \vec{u_n}]$ nếu thoả điều kiện trên là một *ma trận cơ bản*.

Ta có một số định lý liên quan đến Wronskian và hệ phương trình vi phân bậc một thuần nhất như sau:

Định lý 5 *Nếu $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$ là các nghiệm của (H) trên một miền mở I, khi đó $W(\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n)(t) = 0$ với mọi $t \in I$; hoặc $W(\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n)(t) \neq 0$ với mọi $t \in I$.*

Định lý 6 *Giả sử $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$ là các nghiệm của (H) trên một miền mở I. Những điều sau đây là tương đương:*

(i) $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n$ *là tập nghiệm cơ bản của (H) trên I.*

(ii) $W(\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_n)(t) \neq 0$ *cho một số hoặc tất cả t trên I.*

(iii) $c_1\vec{u}_1 + c_2\vec{u}_2 + \cdots + c_n\vec{u}_n$ *là nghiệm tổng quát của (H).*

1.3.6 Hệ phương trình vi phân bậc một thuần nhất với hệ số hằng

Ở đây ta xét hệ

$$\vec{x} = A\vec{x} \tag{H}$$

với t thuộc miền I. A là một ma trận $n \times n$ với các phần tử là hằng số.

Nếu λ là trị riêng của A với vector riêng $\vec{v}$, đặt $\vec{x} = e^{\lambda t}\vec{v}$. Khi đó $\vec{x} = e^{\lambda t}\lambda\vec{v}$. Ta có:

$$A\vec{x} = e^{\lambda t}A\vec{v} = e^{\lambda t}\lambda\vec{v}$$

Nên $\vec{x} = A\vec{x}$; vậy $e^{\lambda t}\vec{v}$ là một nghiệm của (H)

Giả sử ma trận $n \times n$ A có n vector riêng độc lập tuyến tính $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$ và λ_k là trị riêng tương ứng với vector riêng $\vec{v}_k$. Khi đó nghiệm tổng quát là:

$$\vec{x} = c_1 e^{\lambda_1 t}\lambda\vec{v}_1 + c_2 e^{\lambda_2 t}\lambda\vec{v}_2 + \cdot c_n e^{\lambda_n t}\lambda\vec{v}_n.$$

với $c_1, c_2, \cdots, c_n$ là hằng số tuỳ ý.

Và khi đó, tập nghiệm cơ bản là $e^{\lambda_1 t}\vec{v}_1, e^{\lambda_2 t}\vec{v}_2, \cdots, e^{\lambda_n t}\vec{v}_n$. Ma trận cơ bản $X(t)$ là ma trận có các cột tương ứng với các phần tử trong tập nghiệm cơ bản.

Vậy với $X(t)$ là ma trận cơ bản, nghiệm tổng quát của phương trình (H) là $x_c(t) = X(t)\vec{c}$ với $\vec{c} = [c_1, c_2, \cdots, c_n]^T$ và nghiệm thoả điều kiện ban đầu $\vec{x}(a) = \vec{x}_0$ là $x(t) = X(t)X(a)^{-1}\vec{x}_0$

1.3.7 Hệ phương trình vi phân bậc một không thuần nhất

Ta xét hệ phương trình

$$\vec{x} = A(t)\vec{x} + \vec{b}(t) \tag{12}$$

Nghiệm cụ thể của hệ phương trình (12) đưa ra bởi công thức sau:

$$\vec{x}_p = X(t)\int_0^t X(s)^{-1}\vec{b}(s)ds$$

Trong đó $X(t)$ là ma trận cơ bản.

Nghiệm tổng quát của hệ phương trình (12) là:

$$\vec{x} = X(t)\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + X(t)\int_a^t X(s)^{-1}\vec{b}(s)ds$$

Ta có nghiệm cho bài toán điều kiện ban đầu của hệ phương trình vi phân bậc một không thuần nhất $\vec{x} = A(t)\vec{x} + \vec{b}(t), \vec{x}(a) = \vec{x}_0$ là:

$$\vec{x} = X(t)X(a)^{-1}\vec{x}_0 + \vec{x}_p(t)$$

Với $X(t)$ là ma trận cơ bản và $\vec{x}_p = X(t)\int_a^t X(s)^{-1}\vec{b}(s)ds$

1.4 Hệ động lực

1.4.1 Định nghĩa

Các mô hình toán học của các hệ thống khoa học thường cho ra các phương trình vi phân mà ở đó thời gian là biến độc lập. Các hệ này xuất hiện trong các lĩnh vực như thiên văn học, sinh học, hoá học, kinh tế học, kỹ thuật, vật lý học,... Hệ các phương trình vi phân

$$\dot{x} = f(x) \tag{13}$$

tạo ra một *hệ động lực học*. Trong đó cần chú ý những vấn đề sau:

1. Các hành vi liên quan đến tính chất của hệ được thể hiện trên một khoảng thời gian dài. Hay nói cách khác, cần xem xét đến tính động lực học khi biến thời gian $t \to +\infty$.
2. Sự thay đổi của nghiệm thu được từ hệ động lực khi các dữ liệu đầu vào thay đổi. Xét hệ (13) cho tất cả các phương trình vi phân với giá trị ban đầu $x(t_0) = a$, nghiệm không chỉ là một hàm theo biến t mà còn là một hàm theo biến a: $x = \phi(t, a)$
3. Họ các phương trình vi phân. Các phương trình của một họ thường được phân biệt bởi giá trị của một thông số, ví dụ như μ. Vế phải của hệ phương trình (13) được viết lại thành $f(x, \mu)$. Nghiệm của hệ giờ đây phụ thuộc thêm vào μ và (13) có thể được biểu diễn dưới dạng $x = \phi(x, a, \mu)$.

Những điều cần lưu ý trên thể hiện các vấn đề trong quá trình ứng dụng hệ động lực học, tại đây phụ thuộc vào các thông số, và cần phải được xem xét trên nhiều điều kiện ban đầu khác nhau.
Phương trình $\dot{x} = \mu x$ với giá trị ban đầu $x(0) = a$ có nghiệm $x = ae^{\mu t}$. Sự biến thiên của phương trình khi $t \to +\infty$ phụ thuộc đáng kể vào dấu của μ.

Như vậy, hệ động lực là một hệ thống thay đổi theo thời gian dựa vào một tập các ràng buộc nhất định để xác định cách chuyển trạng thái trong hệ thống.
Hay nói cách khác, hệ động lực là một *không gian pha (phase space)* hay *không gian trạng thái (state space)* cùng với cách biến đổi của không gian đó.

Một hệ động lực gồm 2 phần:

- *Vector trạng thái (State vector)* mô tả chính xác các trạng thái của hệ thống.
- Một hàm dùng để miêu tả cách chuyển trạng thái (định nghĩa trạng thái kế tiếp khi đang ở một trạng thái nhất định).

Vector trạng thái có thể được mô tả bởi:

$$\vec{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]$$

Hàm miêu tả có thể được biểu diễn bởi 1 hàm hoặc 1 tập gồm nhiều hàm

$$f_1(x_1, x_2, ..., x_n), f_2(x_1, x_2, ..., x_n), ..., f_n(x_1, x_2, ..., x_n)$$

Và hệ động lực giờ đây có thể được biểu diễn bằng một tập các phương trình vi phân như ở (13)

1.4.2 Phân loại

1.4.2.a Tự động và bất tự động

Hệ các phương trình vi phân như (13) được gọi là một hệ *tự động (autonomous)* vì vế phải của hệ không phụ thuộc vào biến thời gian. Hay nói cách khác, hệ tự động là một hệ phương trình vi phân thường không phụ thuộc vào biến độc lập.

Ngoài ra còn có một dạng hệ tổng quát hơn, *bất tự động (non-autonomous)*, như sau:

$$\dot{x} = f(t, x) \tag{14}$$

Từ hệ phương trình vi phân (14) gồm n phương trình này có thể đưa về một hệ tự động gồm $n + 1$ phương trình bằng cách thêm vào hệ phương trình $\dot{x}_{n+1} = 1$ và thay $t = x_{n+1}$ ở n phương trình còn lại.

1.4.2.b Rời rạc và Liên tục

Hệ động lực *liên tục* là hệ thay đổi trạng thái trong không gian trạng thái của hệ một cách liên tục. Hệ động lực *rời rạc* là hệ thay đổi trạng thái trong không gian trạng thái của hệ một cách rời rạc.

1.5 Một số ví dụ cho hệ phương trình vi phân bậc nhất

1.5.1 Ví dụ 1

Giải $\begin{cases} \dot{x}_1 = x_1 - 3x_2 \\ \dot{x}_2 = -2x_1 + 2x_2 \end{cases}$ với $\vec{x}(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

Ta có ma trận hệ số $A = \begin{bmatrix} 1 & -3 \\ -2 & 2 \end{bmatrix}$ Ta tìm trị riêng và vector riêng tương ứng của A:

$$det(A - \lambda I) = \begin{bmatrix} 1-\lambda & -3 \\ -2 & 2-\lambda \end{bmatrix} = (1-\lambda)(2-\lambda) - 6 = \lambda^2 - 3\lambda - 4 = (\lambda+1)(\lambda-4)$$

Vậy các trị riêng là $\lambda = -1$ và $\lambda = 4$.

Để tìm vector riêng tương ứng với trị riêng $\lambda = 4$, ta giải hệ $A\vec{x} = 4\vec{x}$:

$$\begin{bmatrix} 1 & -3 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ ? \end{bmatrix} = 4 \begin{bmatrix} 1 \\ ? \end{bmatrix}$$

Từ hàng đầu, ta có $1 \cdot 1 + (-3) \cdot ? = 4 \iff ? = -1$. Vậy vector riêng tương ứng với $\lambda = 4$ là $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ Tương tự, để tìm vector riêng tương ứng với $\lambda = -1$ ta giải phương trình:

$$\begin{bmatrix} 1 & -3 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ ? \end{bmatrix} = - \begin{bmatrix} 1 \\ ? \end{bmatrix}$$

Vậy $1 \cdot 1 + (-3) \cdot ? = -1 \iff ? = \frac{2}{3}$. Vậy vector riêng tương ứng với $\lambda = -1$ là $\begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$ hay $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$

Nghiệm tổng quát của hệ khi đó sẽ là $\vec{x} = c_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} e^{4t} + c_2 \begin{bmatrix} 3 \\ 2 \end{bmatrix} e^{-t}$, trong đó c_1, c_2 là các hằng số tuỳ ý.

Ta có thể viết lại thành $\vec{x} = \begin{bmatrix} c_1 e^{4t} + 3c_2 e^{-t} \\ -c_1 e^{4t} + 2c_2 e^{-t} \end{bmatrix}$

Ma trận cơ bản là:

$$X(t) = \begin{bmatrix} e^{4t} & 3e^{-t} \\ -e^{4t} & 2e^{-t} \end{bmatrix}$$

Ta tính được $X(0) = \begin{bmatrix} e^{4\cdot 0} & 3e^{-0} \\ -e^{4\cdot 0} & 2e^{-0} \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 2 \end{bmatrix}$ và $X(0)^{-1} = \begin{bmatrix} 0.4 & -0.6 \\ 0.2 & 0.2 \end{bmatrix}$

Vậy nghiệm $\vec{x} = X(t)X(0)^{-1}\vec{x}_0 = \begin{bmatrix} e^{4t} & 3e^{-t} \\ -e^{4t} & 2e^{-t} \end{bmatrix} \begin{bmatrix} 1.6 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 1.6e^{4t} - 0.6e^{-t} \\ -1.6e^{4t} - 0.4e^{-t} \end{bmatrix}$

1.5.2 Ví dụ 2

Giải $\begin{cases} \dot{x}_1 = 4x_1 - 3x_2 + t \\ \dot{x}_2 = 2x_1 - x_2 + e^t \end{cases}$ với $\vec{x}(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

Ta viết lại dưới dạng $\dot{\vec{x}} = A\vec{x} + \vec{b}(t)$ như sau:

$$\dot{\vec{x}} = \begin{bmatrix} 4 & -3 \\ 2 & -1 \end{bmatrix} \vec{x} + \begin{bmatrix} t \\ e^t \end{bmatrix}$$

Đầu tiên ta cần giải phương trình thuần nhất $\dot{\vec{x}} = A\vec{x}$. Kết quả thu được khi giải phương trình thuần nhất theo cách ở ví dụ 1 là:

$$\vec{x} = c_1 e^t \begin{bmatrix} 1 \\ 1 \end{bmatrix} + c_2 e^{2t} \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Từ đây ta có ma trận cơ bản $X(t) = \begin{bmatrix} e^t & 3e^{2t} \\ e^t & 2e^{2t} \end{bmatrix}$ Để tìm nghiệm cụ thể cho hệ phương trình ban đầu, ta cần tính $X(t) \int_0^t X(s)^{-1} \vec{b}(s) ds$

Ta có $X(s) = \begin{bmatrix} e^s & 3e^{2s} \\ e^s & 2e^{2s} \end{bmatrix}$. Định thức của $X(s)$ là $2e^{3s} - 3e^{3s} = -e^{3s}$. Khi đó:

$$X(s)^{-1} = \frac{1}{-e^{3s}} \begin{bmatrix} 2e^{2s} & -3e^{2s} \\ -e^s & e^s \end{bmatrix} = \begin{bmatrix} -2e^{-s} & 3e^{-s} \\ e^{-2s} & -e^{-2s} \end{bmatrix}$$

Khi đó

$$X(s)^{-1} \vec{b}(s) \begin{bmatrix} -2e^{-s} & 3e^{-s} \\ e^{-2s} & -e^{-2s} \end{bmatrix} \begin{bmatrix} s \\ e^s \end{bmatrix} = \begin{bmatrix} -2se^{-s} + 3 \\ se^{-2s} - e^{-s} \end{bmatrix}$$

$$\implies \int_0^t X(s)^{-1} \vec{b}(s) ds = \begin{bmatrix} \int_0^t (-2se^{-s} + 3) ds \\ \int_0^t (se^{-2s} - e^{-s}) ds \end{bmatrix} = \begin{bmatrix} 2e^{-1}(t+1) + 3t - 2 \\ e^{-2t}\left(\frac{-t}{2} - \frac{1}{4}\right) + e^{-t} - \frac{3}{4} \end{bmatrix}$$

Nghiệm cụ thể là

$$\begin{bmatrix} e^t & 3e^{2t} \\ e^t & 2e^{2t} \end{bmatrix} \begin{bmatrix} 2e^{-1}(t+1) + 3t - 2 \\ e^{-2t}\left(\frac{-t}{2} - \frac{1}{4}\right) + e^{-t} - \frac{3}{4} \end{bmatrix} = \begin{bmatrix} \frac{t}{2} + \frac{5}{4} + (3t+1)e^t - \frac{9}{4}e^{2t} \\ t + \frac{3}{2} + 3te^t - \frac{3}{2}e^{2t} \end{bmatrix} = \vec{x}_p(t)$$

Ta có

$$X(t)X(a)^{-1}(\vec{x}_0) = \begin{bmatrix} e^t & 3e^{2t} \\ e^t & 2e^{2t} \end{bmatrix} \begin{bmatrix} -2e^{-0} & 3e^{-0} \\ e^{-2\cdot 0} & -e^{-2\cdot 0} \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} e^t & 3e^{2t} \\ e^t & 2e^{2t} \end{bmatrix} \begin{bmatrix} -8 \\ 3 \end{bmatrix}$$

Vậy nghiệm chính xác của hệ phương trình ban đầu là

$$\vec{x}(t) = \begin{bmatrix} \frac{t}{2} + \frac{5}{4} + (3t+1)e^t - \frac{9}{4}e^{2t} \\ t + \frac{3}{2} + 3te^t - \frac{3}{2}e^{2t} \end{bmatrix} + \begin{bmatrix} e^t & 3e^{2t} \\ e^t & 2e^{2t} \end{bmatrix} \begin{bmatrix} -8 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{t}{2} + \frac{5}{4} + (3t-7)e^t + \frac{27}{4}e^{2t} \\ t + \frac{3}{2} + (3t-8)e^t + \frac{9}{2}e^{2t} \end{bmatrix}$$

1.6 Phương pháp số để giải bài toán liên quan đến điều kiện ban đầu

Ta xét bài toán điều kiện ban đầu sau và nghiệm giải theo phương pháp số của nó:

$$\frac{dy}{dt} = f(y,t), \quad y(t=0) = y_0. \tag{15}$$

Nếu $f(y,t) \equiv g(y)$, bài toán điều kiện ban đầu trên sẽ được gọi là tự động. Và khi $g(y) = ky$ với k là một hằng số, ta gọi bài toán điều kiện ban đầu trên là tuyến tính. Ta giả sử tồn tại nghiệm duy nhất và kí hiệu nghiệm đó là $y^e(t)$. Từ đây $y(t)$ sẽ đại diện cho nghiệm giải bằng phương pháp số, là một xấp xỉ của nghiệm chính xác $y^e(t)$.

1.6.1 Giải thuật Explicit Euler

Ta kí hiệu thời gian ở bước thời gian thứ nth là t_n và nghiệm tính được bằng phương pháp số ở bước thời gian thứ nth là y_n, hay $y_n \equiv y(t = t_n)$. Bước nhảy h (giả thiết là hằng số) được tính bởi công thức $h = t_n - t_{n-1}$. Với (t_n, y_n), phương pháp Explicit (Forward) Euler tính được y_{n+1} bằng:

$$y_{n+1} = y_n + hf(y_n, t_n) \tag{16}$$

Phương pháp Explicit Euler dựa trên công thức khai triển Taylor và bỏ đi phần dư, đó là nếu khai triển y trên lân cận $t = t_n$, ta được:

$$y(t_n + h) \equiv y_{n+1} = y(t_n) + h\frac{dy}{dt}|_{t_n} + \mathrm{O}(h^2) = y_n + hf(y_n, t_n) + \mathrm{O}(h^2). \tag{17}$$

Từ phương trình (17), ta có thể thấy được sai số dược tạo ra ở mỗi bước thời gian do việc lược bỏ đi phần dư của chuỗi Taylor, sai số này được gọi là *local truncation error* (LTE) của phương pháp. Ở phương pháp Explicit Euler, LTE là $\mathrm{O}(h^2)$. Vì vậy, phương pháp được gọi là phương pháp bậc nhất. Tổng quát, một phương pháp với LTE là $\mathrm{O}(h^{k+1})$ được gọi là phương pháp bậc k. Sai số của việc lược bỏ phần dư khác với sai số tổng quát g_n, với g_n được định nghĩa là trị tuyệt đối của hiệu giữa nghiệm chính xác và nghiệm được tính bằng phương pháp số, hay $g_n = |y^e(t_n) - y_n|$. Trong đa số trường hợp, ta không thể biết nghiệm chính xác vì vậy sai số tổng quát không thể tính toán được. Tuy nhiên, nếu ta bỏ qua sai số làm tròn, ta có thể giả định rằng sai số tổng quát ở bước thời gian thứ nth là n lần LTE, vì khi n tỉ lệ với $1/h$, g_n sẽ tỉ lệ với LTE/h. Điều này dẫn đến khi một phương pháp là phương pháp bậc k thì sai số tổng quát sẽ tăng với tỉ lệ h^k.

Một phương pháp số hội tụ (convergent numerical method) là phương pháp mà tại đó các nghiệm tính được tiến tới nghiệm chính xác khi độ rộng của các bước nhảy tiến về 0. Khi mà nghiệm chính xác không thể xác định được, ta có thể chọn, tuỳ vào yêu cầu của độ chính xác, nghiệm thu được với các bước thời gian đủ nhỏ là một nghiệm 'chính xác' để xác định các tính chất liên quan đến sự hội tụ.

Một điều cần chú ý ở phương pháp Explicit Euler là y_{n+1} được tính dựa trên các đại lượng đã biết trước đó như y_n và $f(y_n, t_n)$. Phương pháp Explicit tuy dễ hiện thực nhưng có hạn chế cần lưu ý là các bước thời gian cần phải được giới hạn sao cho đảm bảo được tính ổn định (numerical stability). Để thấy điều này rõ hơn, xét bài toán điều kiện ban đầu tuyến tính sau: $dy/dt = -ay, y(0) = 1$ với $a > 0$. Nghiệm chính xác cho bài toán này là $y^e = \exp(-at)$, là một nghiệm ổn định với $y^e(0) = 1$ và $y^e(\infty) = 0$. Bằng cách sử dụng phương trình (16), ta có được phương trình rời rạc hoá cho bài toán điều kiện ban đầu trên như sau:

$$y_{n+1} = y_n - ah \cdot y_n = (1-ah)y_n = (1-ah)^2 y_{n-1} = \cdots = (1-ah)^n y_1 = (1-ah)^{n+1} y_0 \tag{18}$$

Phương trình (18) chỉ ra rằng để hạn chế sự tăng vọt của sai số trong quá trình lặp, ta có điều kiện $|1-ah|<1$ hay để đảm bảo tính ổn định cho phương pháp Explicit Euler, ta phải có $h<\frac{2}{a}$

Hàm ổn định và miền ổn định của phương pháp có thể được định nghĩa như sau:

Xét phương trình:

$$x_{n+1}=\Phi(ah)x_n \tag{19}$$

Khi đó $\Phi=\Phi(z)$ với $z\in\mathbb{C}$ được gọi là hàm ổn định và miền:

$$\{z\in\mathbb{C}:|\Phi(z)|<1\} \tag{20}$$

được gọi là miền ổn định của phương pháp số.

1.6.2 Giải thuật Runge-Kutta

Ở phương pháp Explicit Euler, ta dùng đạo hàm của y tại một bước thời gian nhất định để ngoại suy kết quả cho bước thời gian kế tiếp. LTE của phương pháp này là $O(h^2)$, kết luận được đây là một phương pháp số bậc nhất. Runge-Kutta là một lớp các phương pháp dùng đạo hàm tại nhiều hơn một điểm để ngoại suy kết quả cho bước thời gian kế tiếp. Ta xét phương pháp Runge-Kutta bậc hai với LTE là $O(h^3)$ đầu tiên.

Với bài toán điều kiện ban đầu (15), bước thời gian h, nghiệm y_n ở bước thời gian thứ n, ta tính y_{n+1} bằng cách sau:

$$\begin{aligned}k_1&=hf(y_n,t_n)\\k_2&=hf(y_n+\beta k_1,t_n+\alpha h)\\y_{n+1}&=y_n+ak_1+bk_2\end{aligned} \tag{21}$$

Với các hằng số α, β, a và b được tính sao cho phương pháp có LTE là $O(h^3)$. Chú ý rằng nếu $k_2=0$ và $a=1$ thì phương trình (21) trở thành phương trình của phương pháp Explicit Euler.

Ta có khai triển Taylor của y ở lân cận t_n ứng với bậc của h^2 là:

$$y(t_{n+1})=y(t_n)+h\frac{dy}{dt}|_{t_n}+\frac{h^2}{2}\frac{d^2y}{dt^2}|_{t_n}+O(h^3) \tag{22}$$

Tuy nhiên, ta biết được $\frac{dy}{dt}=f(y,t)$ từ phương trình (15), cho nên ta có:

$$\frac{d^2y}{dt^2}=\frac{df(y,t)}{dt}=\frac{\partial f}{\partial t}+\frac{\partial f}{\partial y}\frac{dy}{dt}=\frac{\partial f}{\partial t}+f\frac{\partial f}{\partial y} \tag{23}$$

Từ 2 phương trình (22) và (23), ta có được:

$$y_{n+1}=y_n+hf(y_n,t_n)+\frac{h^2}{2}\left[\frac{\partial f}{\partial t}+f\frac{\partial f}{\partial y}\right](y_n,t_n)+O(h^3) \tag{24}$$

Tuy nhiên, k_2 ở phương trình (21) có thể được triển khai tới $O(h^3)$ như sau:

$$\begin{aligned}k_2&=hf(y_n+\beta k_1,t_n+\alpha h)\\&=h\left(f(y_n,t_n)+\alpha h\frac{\partial f}{\partial t}(y_n,t_n)+\beta k_1\frac{\partial f}{\partial t}(y_n,t_n)\right)+O(h^3)\end{aligned} \tag{25}$$

Thay k_2 ở phương trình (25) vào phương trình (21), ta được:

$$y_{n+1}=y_n+(a+b)hf(y_n,t_n)+bh^2(\alpha\frac{\partial f}{\partial t}+\beta f\frac{\partial f}{\partial y})(y_n,t_n)+O(h^3). \tag{26}$$

So sánh các số hạng có cùng hệ số ở 2 phương trình (26) và (24), ta có được hệ phương trình sau để xác định các hằng số:

$$\begin{aligned} a + b &= 1 \\ \alpha b &= \frac{1}{2} \\ \beta b &= \frac{1}{2}. \end{aligned} \tag{27}$$

Có vô số cách chọn a, b, α, và β để thoả hệ phương trình (27). Ta có thể chọn $\alpha = \beta = 1$ và $a = b = \frac{1}{2}$. Với cách chọn này, ta có phương pháp Runge-Kutta bậc 2 cổ điển (RK2) như sau:

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(yn + k_1, tn + h) \\ y_{n+1} &= y_n + (k_1 + k_2)/2 \end{aligned} \tag{28}$$

Bằng cách tương tự ta có thể xây dựng các phương pháp Runge-Kutta bậc cao hơn. Một trong các phương pháp được sử dụng rộng rãi để tìm nghiệm xấp xỉ cho các bài toán điều kiện ban đầu là Runge-Kutta bậc 4 (RK4). LTE của phương pháp này là $O(h^5)$. Phương pháp RK4 được thể hiện như dưới đây:

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(y_n + \frac{k_1}{2}, t_n + \frac{h}{2}) \\ k_3 &= hf(y_n + \frac{k_2}{2}, t_n + \frac{h}{2}) \\ k_4 &= hf(y_n + k_3, t_n + h) \\ y_{n+1} &= y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6. \end{aligned} \tag{29}$$

Cần lưu ý rằng các phương pháp Runge-Kutta đều là Explicit, vì vậy các phương pháp này chỉ ổn định ở các điều kiện nhất định.

1.7 Giải các ví dụ bằng phương pháp số

1.7.1 Ví dụ 1

$$\begin{cases} \dot{x_1} = x_1 - 3x_2 \\ \dot{x_2} = -2x_1 + 2x_2 \end{cases} \quad \text{với } \vec{x}(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Ta muốn tính $\vec{x}(0.25)$ với số lần lặp là 5. Vậy step size là $h = \frac{0.25 - 0}{5} = 0.05$

1.7.1.a Phương pháp Explicit Euler

Theo phương pháp Euler, với $\vec{x}_j$ cho sẵn, $\vec{x}_{j+1}$ sẽ được tính theo công thức sau:

$$\vec{x}_{j+1} = \vec{x}_j + hA\vec{x}_j$$

với $A = \begin{bmatrix} 1 & -3 \\ -2 & 2 \end{bmatrix}$

Bảng kết quả qua các lần lặp:

	x_1	x_2
$(t = 0)$	1	-2
$(t = 0.05)$	1.35	-2.3
$(t = 0.1)$	1.7625	-2.665
$(t = 0.15)$	$\frac{18003}{8000}$	$\frac{-12431}{4000}$
$(t = 0.2)$	$\frac{452649}{160000}$	$\frac{-58297}{16000}$
$(t = 0.25)$	$\frac{11254539}{3200000}$	$\frac{18003}{8000}$

Vậy $\vec{x}(0.25) \approx \begin{bmatrix} 3.5170 \\ -4.2908 \end{bmatrix}$

1.7.1.b Phương pháp Runge-Kutta bậc 4

Với $f(x, y, z) = y - 3z$ và $g(x, y, z) = -2y + 2z$ Ta tính theo công thức sau:

$$
\begin{aligned}
k_0 &= hf(x_i, y_i, z_i) \\
l_0 &= hg(x_i, y_i, z_i) \\
k_1 &= hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0) \\
l_1 &= hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0) \\
k_2 &= hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1) \\
l_2 &= hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1) \\
k_3 &= hf(x_i + h, y_i + k_2, z_i + l_2) \\
l_3 &= hg(x_i + h, y_i + k_2, z_i + l_2) \\
y_{i+1} &= y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3) \\
z_{i+1} &= z_i + \frac{1}{6}(l_0 + 2l_1 + 2l_2 + l_3)
\end{aligned}
$$

Bảng kết quả qua các lần lặp (các kết quả ở trung gian khi tính cho bước sau không làm tròn):

	x_1	x_2
$(t = 0)$	1	-2
$(t = 0.05)$	1.8440	-2.7488
$(t = 0.1)$	1.3835	-2.3347
$(t = 0.15)$	2.3989	-3.2597
$(t = 0.2)$	3.0696	-3.8883
$(t = 0.25)$	3.8819	-4.6607

So với nghiệm chính xác:

$$\vec{x}(0.25) = \begin{bmatrix} 1.6e^{4 \cdot 0.25} - 0.6e^{-0.25} \\ -1.6e^{4 \cdot 0.25} - 0.4e^{-0.25} \end{bmatrix} \approx \begin{bmatrix} 3.8820 \\ -4.6608 \end{bmatrix}$$

Phương pháp RK4 cho kết quả chính xác hơn.

1.7.2 Ví dụ 2

$$\begin{cases} \dot{x}_1 = 4x_1 - 3x_2 + t \\ \dot{x}_2 = 2x_1 - x_2 + e^t \end{cases} \quad \text{với } \vec{x}(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Ta muốn tính $\vec{x}(0.25)$ với số lần lặp là 5. Vậy step size là $h = \dfrac{0.25 - 0}{5} = 0.05$

1.7.2.a Phương pháp Explicit Euler

Theo phương pháp Euler, với $\vec{x}_j$ cho sẵn, $\vec{x}_{j+1}$ sẽ được tính theo công thức sau:

$$\vec{x}_{j+1} = \vec{x}_j + hA\vec{x}_j + h\vec{b}(j)$$

với $A = \begin{bmatrix} 4 & -3 \\ 2 & -1 \end{bmatrix}$ và $b = \begin{bmatrix} t \\ e^t \end{bmatrix}$

Bảng kết quả qua các lần lặp (các kết quả ở trung gian khi tính cho bước sau không làm tròn):

	x_1	x_2
$(t = 0)$	1	-2
$(t = 0.05)$	1.5	-1.75
$(t = 0.1)$	2.065	-1.460
$(t = 0.15)$	2.7020	-1.1252
$(t = 0.2)$	3.4178	-0.7406
$(t = 0.25)$	4.2235	-0.3007

1.7.2.b Phương pháp Runge-Kutta bậc 4

Với $f(x, y, z) = 4y - 3z + x$ và $g(x, y, z) = 2y - z + e^x$ Ta tính theo công thức sau:

$$\begin{aligned}
k_0 &= hf(x_i, y_i, z_i) \\
l_0 &= hg(x_i, y_i, z_i) \\
k_1 &= hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0) \\
l_1 &= hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0, z_i + \frac{1}{2}l_0) \\
k_2 &= hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1) \\
l_2 &= hg(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1, z_i + \frac{1}{2}l_1) \\
k_3 &= hf(x_i + h, y_i + k_2, z_i + l_2) \\
l_3 &= hg(x_i + h, y_i + k_2, z_i + l_2) \\
y_{i+1} &= y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3) \\
z_{i+1} &= z_i + \frac{1}{6}(l_0 + 2l_1 + 2l_2 + l_3)
\end{aligned}$$

Bảng kết quả qua các lần lặp (các kết quả ở trung gian khi tính cho bước sau không làm tròn):

	x_1	x_2
$(t = 0)$	1	-2
$(t = 0.05)$	1.5337	-1.7292
$(t = 0.1)$	2.1398	-1.4135
$(t = 0.15)$	2.8265	-1.0475
$(t = 0.2)$	3.6028	-0.6252
$(t = 0.25)$	4.4787	-0.1399

So với nghiệm chính xác:

$$\vec{x}(0.25) = \begin{bmatrix} \frac{0.25}{2} + \frac{5}{4} + (3 \cdot 0.25 - 7)e^{0.25} + \frac{27}{4}e^{2 \cdot 0.25} \\ 0.25 + \frac{3}{2} + (3t - 8)e^{0.25} + \frac{9}{2}e^{2 \cdot 0.25} \end{bmatrix} \approx \begin{bmatrix} 4.4787 \\ -0.1399 \end{bmatrix}$$

Phương pháp RK4 cho kết quả chính xác hơn.

2 Ứng dụng

2.1 Giới thiệu

Ngày nay, với sự hỗ trợ của công nghệ tiên tiến, nhiều giống hoa màu được trồng trọt, thu hoạch và kinh doanh trên thị trường dù không đúng mùa vụ chính. Các giống hoa màu này chủ yếu được trồng trong các nhà kính hiện đại với hệ thống điều khiển tự động hoặc bán tự động có chức năng điều chỉnh các yếu tố thời tiết và khí hậu bên trong nhà kính nhằm tạo điều kiện tốt nhất cho hoa màu phát triển.

Một số thành phần khí hậu chính bên trong nhà kính bao gồm: nhiệt độ, áp suất hơi nước, đặc biệt là nồng độ khí CO_2 ảnh hưởng trực tiếp đến năng suất cây trồng. Các thành phần này thông thường bị ảnh hưởng bởi một hay nhiều vật thể có trong nhà kính, cấu trúc nhà kính và sự chuyển động của dòng không khí bên trong nhà kính.

Trong bài báo cáo này, ta xét đến các nhân tố ảnh hưởng đến nồng độ khí CO_2 và áp suất hơi nước, xây dựng các mô hình động lực liên quan. Các nội dung của bài báo cáo tham khảo chủ yếu từ [Van11] và các tài liệu liên quan.

2.2 Sự trao đổi khí CO_2

Nồng độ khí CO_2 của khí hậu bên trong nhà kính sẽ được xem xét và mô tả một cách cụ thể. Mô hình nhà kính với *màn chắn nhiệt (thermal screen)* được xét để tổng quát hoá bài toán và có thể áp dụng mô hình này trong thực tế. Màn chắn nhiệt hay tấm chắn nhiệt được làm từ nhiều loại vật liệu khác nhau như kim loại hoặc nhựa dẻo có tính đàn hồi. Màn chắn nhiệt được sử dụng để bảo vệ cây trồng khỏi thiệt hại gây ra bởi ánh nắng trực tiếp từ mặt trời cũng như chống rét cho cây trồng vào mùa đông ở những nơi có khí hậu ôn đới.

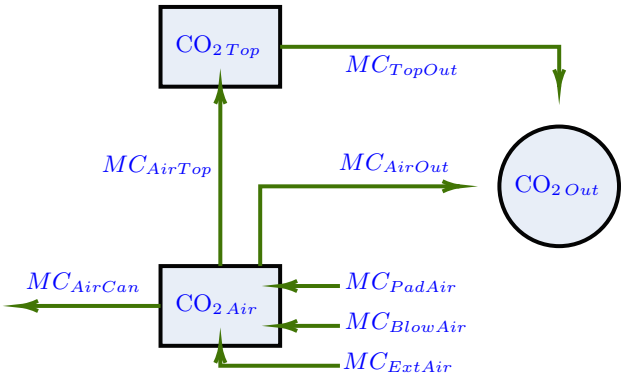
Màn chắn nhiệt chia nhà kính thành hai gian khác nhau, gồm gian nhà kính dưới màn chắn và gian nhà kính phía trên màn chắn. Gian phía trên thường hẹp hơn gian phía dưới. Điều này dẫn đến nồng độ CO_2 trong không khí ở gian trên và gian dưới nhà kính cũng khác nhau. Hình 1 tóm tắt sự lưu thông của lượng khí CO_2 trong nhà kính.

Đối với gian dưới của nhà kính, lượng khí CO_2 chủ yếu được đưa vào từ các nguồn như luồng gió tự nhiên thông qua hệ thống thông gió và thoát ra ngoài bởi hệ thống quạt. Ngoài ra lượng khí CO_2 ở gian này cũng nhận được bởi các máy sưởi không khí trong quá trình đốt nóng tạo nhiệt lượng và bởi bên thứ ba chuyên cung cấp khí CO_2. Một phần lượng khí CO_2 ở gian dưới nhà kính cũng thất thoát lên gian trên nhà kính dưới sự điều hướng của sự chênh lệch nhiệt độ và mật độ không khí giữa hai gian. Ngoài ra, một lượng lớn khí CO_2 cũng sẽ được hấp thụ vào trong cây trồng để thực hiện quá trình quang hợp. Đối với gian trên nhà kính, lượng khí CO_2 chủ yếu nhận tự sự trao đổi khí CO_2 với gian dưới và có thể thoát ra bên ngoài thông qua các ô thông gió ở trên mái nhà kính (nếu có).

2.3 Mô hình Hệ động lực biểu diễn chuyển động khí CO_2 và giả thiết

Phần này đề cập đến mô hình Hệ động lực biểu diễn dòng chuyển động của khí CO_2 bên trong và bên ngoài nhà kính sẽ được đề cập đến. Dựa trên sơ đồ quan sát được ở Hình 1, sự thay đổi của nồng độ khí CO_2 ở gian dưới và gian trên bên trong nhà kính được biểu diễn qua hệ gồm hai phương trình sau đây:

$$\begin{cases} cap_{\mathrm{CO}_{2Air}} \cdot \dot{\mathrm{C}}\mathrm{O}_{2Air} & = MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} \\ & \quad -MC_{AirCan} - MC_{AirTop} - MC_{AirOut}, \\ cap_{\mathrm{CO}_{2Top}} \cdot \dot{\mathrm{C}}\mathrm{O}_{2Top} & = MC_{AirTop} - MC_{TopOut} \end{cases} \tag{30}$$



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

Trong hệ phương trình vi phân này, một số giả thiết đã được xét đến như lượng khí CO_2 trong không khí ở gian dưới và gian trên của nhà kính không bị ảnh hưởng bởi nguồn nào khác ngoại trừ những nguồn đã được thể hiện trong sơ đồ ở Hình 1. Ngoài ra, nhà kính là một môi trường hoàn hảo theo nghĩa nồng độ CO_2 là một phân bố đều ở gian dưới và ở gian trên. Các ký hiệu cap_A, $CO_{2\,A}$ $\dot{CO}_{2\,A}$ và MC_{AB} lần lượt là khả năng chứa khí CO_2 trong A (m), nồng độ khí CO_2 trong A (mg m^{-3}), tốc độ thay đổi nồng độ khí CO_2 trong A (mg m^{-3} s^{-1}) và lưu lượng khí CO_2 đi từ A vào B (mg m^{-2} s^{-1}), trong đó Air và Top đại diện cho gian dưới và gian trên, $Blow$ đại diện cho máy sưởi, Ext đại diện cho bên thứ ba cung cấp khí CO_2, Pad đại diện cho hệ thống thông gió, Can đại diện cho tán lá cây trồng và Out đại diện cho không gian bên ngoài nhà kính.

Dưới đây là các công thức để tính MC_{AB}. Trước hết, lượng CO_2 đi từ máy sưởi vào gian dưới của nhà kính có thể được xác định thông qua công thức sau:

$$MC_{BlowAir} = \frac{\eta_{heatCO_2} \cdot U_{Blow} \cdot P_{Blow}}{A_{Flr}} \tag{31}$$

Trong đó η_{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}). Tham số U_{Blow} thể hiện mức cho phép lượng khí CO_2 sinh ra bởi máy sửa đi vào nhà kính có thể điều chỉnh được trong khoảng $[0, 1]$ và không có đơn vị. Hệ số P_{Blow} là khả năng sinh ra CO_2 của máy sưởi (W) và A_{Flr} là diện tích mặt nền nhà kính (m^2).

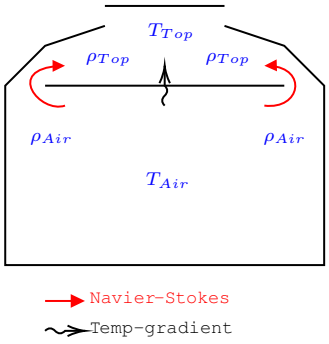
Tương tự, lượng khí CO_2 được bơm vào nhà kính bởi bên thứ ba chuyên cung cấp khí CO_2 được cho bởi công thức sau:

$$MC_{ExtAir} = \frac{U_{ExtCO_2} \cdot \phi_{ExtCO_2}}{A_{Flr}} \tag{32}$$

Trong đó U_{ExtCO_2} là tham số điều chỉnh tốc độ bơm khí CO_2 vào trong nhà kính (không có đơn vị) và ϕ_{ExtCO_2} là khả năng bơm CO_2 của bên thứ ba (mg s^{-1}).

Mặt khác, lượng khí CO_2 đi vào nhà kính thông qua hệ thống thông gió dựa trên sự chênh lệch của nồng độ khí CO_2 bên trong và bên ngoài nhà kính và khả năng cho dòng không khí đi qua của tấm thông gió. Hơn nữa, khả năng cho dòng không khí đi qua của tấm thông gió có thể điều chỉnh được. Ta dùng công thức sau để tính MC_{PadAir}:

$$MC_{PadAir} = f_{Pad}(CO_{2Out} - CO_{2\,Air}) = \frac{U_{Pad} \cdot \phi_{Pad}}{A_{Flr}}(CO_{2Out} - CO_{2\,Air}) \tag{33}$$



Hình 2: Chuyển động của khí CO_2 qua màn chắn nhiệt

Tốc độ của không khí đi qua tấm thông gió f_{Pad} (m s^{-1}) được tính bởi tích của tham số U_{Pad}, thể hiện mức cho phép lượng khí CO_2 đi qua tấm thông gió điều chỉnh được trong khoảng $[0, 1]$ (không có đơn vị) và ϕ_{Pad} là khả năng cho phép khí CO_2 đi qua của tấm thông gió (m^3 s^{-1}) chia cho diện tích nền nhà kính.

Đối với lượng khí CO_2 từ gian dưới lên gian trên nhà kính, quá trình này diễn ra phức tạp hơn và phụ thuộc vào độ chênh lệch nhiệt độ và độ chênh lệch mật độ của hai gian nhà kính thông qua màn chắn nhiệt.

$$MC_{AirTop} = f_{ThScr}(\mathrm{CO_2}_{Air} - \mathrm{CO_2}_{Top}) \tag{34}$$

trong đó tốc độ lưu thông khí CO_2 qua màn chắn nhiệt f_{ThScr} (m s^{-1}) là tổng của hai tốc độ gồm tốc độ thẩm thấu không khí qua màn chắn nhiệt và tốc độ lưu thông không khí tại những nơi không bị chắn bởi màn chắn nhiệt. (Hình 2)

$$f_{ThScr} = U_{ThScr} \cdot K_{ThScr}|T_{Air} - T_{Top}|^{\frac{2}{3}} + (1 - U_{ThScr})\left[\frac{g(1 - U_{ThScr})}{2\rho_{Air}^{mean}}|\rho_{Air} - \rho_{Top}|\right]^{\frac{1}{2}} \tag{35}$$

Và ρ_{Air}, ρ_{Out} và ρ_{Air}^{Mean} được tính bằng:

$$\rho_{Air} = \rho_{Air0} \exp\left(\frac{gM_{Air}h_{Air}^{Elevation}}{(273.15 + T_{Air})R}\right) \tag{36}$$

$$\rho_{Top} = \rho_{Air0} \exp\left(\frac{gM_{Air}h_{Top}^{Elevation}}{(273.15 + T_{Top})R}\right) \tag{37}$$

$$\rho_{Air}^{Mean} = \rho_{Air0} \exp\left(\frac{gM_{Air}h_{Mean\_Air}^{Elevation}}{(273.15 + T_{Air}^{Mean})R}\right) \tag{38}$$

Trong đó R là hằng số khí lý tưởng (J $kmol^{-1}$ K^{-1}), $h_A^{Elevation}$ là độ cao của phần không khi tại vị trí A so với mực nước biển, T_{Air} là nhiệt độ không khí của nhà kính phần dưới màn chắn nhiệt, T_{Top} là nhiệt độ không khí của phần không khí phía trên màn chắn nhiệt và T_{Air}^{Mean} là nhiệt độ trung bình của phần không khí phía dưới và phía trên màn chắn nhiệt.

Ở những nơi có màn chắn nhiệt với độ phủ $U_{ThScr} \in [0, 1]$ (không có đơn vị), tốc độ phụ thuộc vào sự chênh lệch giữa nhiệt độ bên gian trên T_{Top} (K) và bên gian dưới T_{Air} (K), cùng

với đó là khả năng cho không khí thẩm thấu của màn chắn K_{ThScr} (m $K^{-\frac{2}{3}}$ s^{-1}). Ở những nơi không có màn chắn nhiệt có độ phủ $1 - U_{ThScr}$ thì tốc độ lưu thông không khí được cho bởi phương trình *Navier - Stokes* phụ thuộc vào sự chênh lệch của mật độ không khí giữa 2 gian lần lượt là ρ_{Air} và ρ_{Top} có đơn vị là (kg m^{-3}). Hệ số $2/3$ trong công thức (35) đến từ thực nghiệm trong công trình của [Bal92]. Trong công trình đó, tác giả đã sử dụng dữ liệu đo đạc được về tốc độ trao đổi không khí qua các màn chắn làm từ 12 loại vật liệu khác nhau và sử dụng chúng để huấn luyện mô hình $K_{ThScr}|T_{Air} - T_{Top}|^m$ với m là tham số điều chỉnh được để tìm ra m gần bằng 0.66 hay $2/3$. Riêng công thức Navier - Stokes đến từ nghiên cứu của N.J.van de Braak, trong đó ông và các cộng sự đã 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 mật độ không khí có dạng:

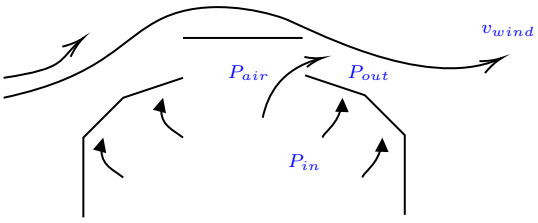
$$\phi_{crack} = \frac{L \cdot SO}{\rho_{mean}} \left[\frac{1}{2} \rho_{mean} \cdot SO \cdot g(\rho_1 - \rho_2)^{\frac{1}{2}} \right] \tag{39}$$

Trong đó ϕ_{crack} (m^3 s^{-1}) là lưu lượng không khí đi qua màn chắn, L(m) là chiều dài khoảng mở trên màn chắn, SO là khoảng mở trên màn chắn (m), ρ_{mean} (kg m^{-3}) là mật độ trung bình của mật độ không khí phía trên màn chắn ρ_1 (kg m^{-3}) và phía dưới màn chắn ρ_2 (kg m^{-3}) và g là gia tốc trọng trường (m s^{-2}). Các công trình sau đó và thực nghiệm cũng đã cho thấy công thức Navier - Stokes (39) cho kết quả tốt khi đối chiếu với dữ liệu đo đạc được.

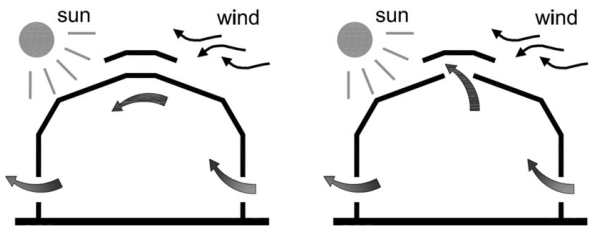
Tương tự, để biểu diễn lượng khí CO_2 từ bên trong ra bên ngoài nhà kính theo hai hướng từ gian dưới và từ gian trên thông qua các ô thông gió, ta sử dụng các công thức sau đây:

$$MC_{AirOut} = (f_{VentSide} + f_{VentForced})(\mathrm{CO}_{2Air} - \mathrm{CO}_{2Out}) \tag{40}$$

Trong đó, $f_{VentSide}$ là tốc độ gió của hệ thống quạt trên tường bao xung quanh nhà kính (m s^{-1}) và $f_{VentForced}$ là tốc độ gió từ hệ thống quạt bên trong nhà kính (m s^{-1}). Trong trường hợp này, nguyên lý *Bernoulli* đóng vai trò quan trọng biểu diễn bởi độ chênh lệch áp suất từ phía ngoài nhà kính gây ra bởi luồng gió tự nhiên và áp suất từ phía trong nhà kính gây ra do luồng không khí bên trong (Hình 3), hiệu ứng *Stack* (hay còn gọi là hiệu ứng *Chimney*) cũng cần được xem xét đến (Hình 4). Hiệu ứng Stack là hiệu ứng khi vào mùa đông, dòng không khí lạnh từ bên ngoài vào bên trong nhà kính bị làm nóng dần bởi hệ thống sưởi và có xu hướng đi lên phía trên mái nhà kính và thoát ra lại bên ngoài, vào mùa hè thì theo chiều ngược lại.



Hình 3: Dòng không khí qua ô mở mái nhà kính



Hình 4: Không có hiệu ứng Stack (trái) và có hiệu ứng Stack (phải)

Để tổng quát hoá mô hình cho nhiều loại nhà kính khác nhau, công thức tổng quát dưới đây $f_{VentRoofSide}$ (m s^{-1}) được dùng để thiết lập công thức cho $f_{VentSide}$ [Kit+96]:

$$f_{VentRoofSide} = \frac{C_d}{A_{Flr}} \left[\frac{U_{Roof}^2 \cdot U_{Side}^2 \cdot A_{Roof}^2 \cdot A_{Side}^2}{U_{Roof}^2 \cdot A_{Roof}^2 + U_{Side}^2 \cdot A_{Side}^2} \cdot \frac{2gh_{SideRoof}(T_{Air} - T_{Out})}{(T_{Air}^{Mean} + 273.15)} + \left(\frac{U_{Roof} \cdot A_{Roof} + U_{Side} \cdot A_{Side}}{2} \right)^2 C_w v_{Wind}^2 \right]^{\frac{1}{2}} \tag{41}$$

Trong đó U_{Roof} và U_{Side} lần lượt là mức cho phép sự thông khí từ mái và từ các nơi thông gió trên tường bao quanh nhà kính, A_{Roof} và A_{Side} lần lượt là tổng diện tích ô mở ở phần mái và tổng diện tích các nơi thông gió trên tường bao quanh nhà kính. Công thức (41) là tổng của hai thành phần nhân với tỷ lệ giữa hệ số lưu lượng gió C_d (không có đơn vị) và diện tích nền nhà kính A_{Flr}. Thành phần thứ nhất phụ thuộc vào độ chênh lệch nhiệt độ giữa bên ngoài và bên trong nhà kính (ở gian dưới màn chắn nhiệt) đại diện cho hiệu ứng Stack khi diện tích ô thông gió trên mái A_{Roof} (m^2) khác không. Thành phần thứ hai cho bởi độ chênh lệch áp suất bên trong và bên ngoài nhà kính, được tính bằng tổng diện tích các nơi thông gió trên nhà kính chia hai nhân với vận tốc gió tự nhiên v_{Wind} (m s^{-1}) và hệ số C_w (không có đơn vị). Các hệ số C_d và C_w là các hệ số lý thuyết phụ thuộc vào cấu trúc và hình dáng của nhà kính, có thể ước lượng thông qua các số liệu đo đạc được trên thực nghiệm.

Đối với mô hình này, C_d và C_w phụ thuộc vào quá trình sử dụng tấm che nắng. Công thức tính C_d và C_w như sau:

$$\begin{aligned} C_d &= C_d^{Gh}(1 - \eta_{ShScrC_d} U_{ShScr}) \\ C_w &= C_w^{Gh}(1 - \eta_{ShScrC_w} U_{ShScr}) \end{aligned} \tag{42}$$

Với C_d^{Gh} là hệ số lưu lượng gió cho nhà kính không có sử dụng tấm che nắng ngoài, η_{ShScrC_d} là một tham số xác định ảnh hưởng của tấm che nắng lên hệ số này, C_w^{Gh} là hệ số áp lực gió cho nhà kính không có tấm che nắng và η_{ShScrC_w} là tham số xác định ảnh hưởng của tấm che nắng lên áp lực gió này.

Ngoài ra, lưới chắn côn trùng gây hại trên các nơi thông gió và hệ số rò rỉ của nhà kính cũng được xét đến. Khi có lưới chắn côn trùng, tốc độ chuyển động của các luồng không khí qua các nơi thông gió sẽ giảm xuống với hệ số:

$$\eta_{InsScr} = \zeta_{InsScr}(2 - \zeta_{InsScr}) \tag{43}$$

Trong đó ζ_{InsScr} (không có đơn vị) là độ rổ của lưới, hay là tỷ lệ diện tích các lỗ trên lưới so với tổng diện tích lưới chắn. Với hệ số rò rỉ $c_{leakage}$ (không có đơn vị), tốc độ trao đổi không khí

thường được xấp xỉ khoảng 50% tốc độ rò rỉ $f_{leakage}$, $f_{leakage}$ được xác định như sau:

$$f_{leakage} = \begin{cases} 0.25 \cdot c_{leakage}, & v_{Wind} < 0.25, \\ v_{Wind} \cdot c_{leakage}, & v_{Wind} \geq 0.25 \end{cases} \tag{44}$$

Một cách ngầm hiểu, giả thiết về phân bố đều của sự rò rỉ của nhà kính đã được sử dụng.

Gọi η_{Roof} là tỷ lệ giữa diện tích ô mở trên mái nhà kính so với tổng diện tích các ô thông gió của nhà kính, η_{Roof} được xác định theo công thức:

$$\eta_{Roof} = \frac{U_{Roof} A_{Roof}}{U_{Roof} A_{Roof} + U_{Side} A_{Side}} \tag{45}$$

Khi tỷ lệ η_{Roof} giữa diện tích ô mở trên mái nhà kính so với tổng diện tích các ô thông gió trên nhà kính vượt ngưỡng Stack η_{Roof_Thr} thì hiệu ứng Stack không xảy ra và ngược lại. Gọi η_{Side} là tỷ lệ giữa diện tích các nơi thông gió trên tường bao quanh nhà kính so với diện tích của tất cả các nơi thông gió trên nhà kính. Ta cũng thiết lập được công thức tính η_{Side} tương tự dựa trên (45). Khi đó, $f_{VentSide}$ được tính bởi công thức sau:

$$f_{VentSide} = \begin{cases} \eta_{InsScr} \cdot f''_{VentSide} + 0.5 f_{leakage}, & \eta_{Roof} \geq \eta_{Roof\_Thr}, \\ \eta_{InsScr} \big[U_{ThScr} \cdot f''_{VentSide} & \\ \quad + (1 - U_{ThScr}) f_{VentRoofSide} \cdot \eta_{Side} \big] + 0.5 f_{leakage}, & \eta_{Roof} < \eta_{Roof\_Thr} \end{cases} \tag{46}$$

Trong đó $f''_{VentSide} = \frac{C_d U_{Side} A_{Side} v_{wind}}{2 A_{Flr}} \sqrt{C_w}$. Lưu ý, ở những nơi phủ bởi màn chắn nhiệt, hiệu ứng Stack cũng không xảy ra.

Tốc độ $f_{VentForced}$ tạo ra bởi hệ thống quạt gió bên trong nhà kính được tính bởi công thức sau:

$$f_{VentForced} = \frac{\eta_{InsScr} \cdot U_{VentForced} \cdot \phi_{VentForced}}{A_{Flr}} \tag{47}$$

$U_{VentForced}$ (không có đơn vị) thể hiện sự điều chỉnh $\phi_{VentForced}$ (tốc độ gió mà hệ thống có thể tạo ra, tính theo $m^3\ s^{-1}$), có giá trị trong khoảng $[0, 1]$.

Tương tự như MC_{AirOut}, lượng khí CO_2 đi từ gian trên của nhà kính ra bên ngoài thông qua ô mở trên mái nhà kính được cho bởi công thức:

$$MC_{TopOut} = f_{VentRoof}(\mathrm{CO}_{2Top} - \mathrm{CO}_{2Out}) \tag{48}$$

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

$$f_{VentRoof} = \begin{cases} \eta_{InsScr} \cdot f''_{VentRoof} + 0.5 f_{leakage}, & \eta_{Roof} \geq \eta_{Roof\_Thr}, \\ \eta_{InsScr} \big[U_{ThScr} \cdot f''_{VentRoof} & \\ \quad + (1 - U_{ThScr}) f_{VentRoofSide} \cdot \eta_{Roof} \big] + 0.5 f_{leakage}, & \eta_{Roof} < \eta_{Roof\_Thr} \end{cases} \tag{49}$$

Tuy nhiên, khác với $f_{VentSide}$ trong công thức (46), khi tỷ lệ η_{Roof} giữa diện tích ô mở trên mái nhà kính so với tổng diện tích các ô thông gió trên nhà kính vượt ngưỡng Stack η_{Roof_Thr} (hiệu ứng Stack không xảy ra), ta không thể sử dụng công thức $f_{VentRoofSide}$ trong công thức (41) với $A_{Side} = 0$ để tính $f''_{VentRoof}$ mà phải sử dụng công thức sau bởi [BB95]:

$$f''_{VentRoof} = \frac{C_d \cdot U_{Roof} \cdot A_{Roof}}{2 A_{Flr}} \left[\frac{g h_{Roof} (T_{Air} - T_{Out})}{2(T^{Mean}_{Air} + 273.15)} + C_w v^2_{Wind} \right]^{\frac{1}{2}} \tag{50}$$

Cuối cùng, ta cần mô tả lượng khí CO_2 bị hấp thụ vào trong tán lá thông qua quá trình quang hợp:

$$MC_{AirCan} = M_{\mathrm{CH_2O}} \cdot h_{C_{Buf}}(P - R_P) \tag{51}$$

Trong đó $M_{\mathrm{CH_2O}}$ là khối lượng mol của $\mathrm{CH_2O}$ (mg μmol^{-1}), P là tốc độ quang hợp (μmol $\{CO_2\}$ m^{-2} s^{-1}), R_P là tốc độ hô hấp của cây (μmol $\{CO_2\}$ m^{-2} s^{-1}) và hệ số

$$h_{C_{Buf}} = \begin{cases} 0, & C_{Buf} > C_{Buf}^{Max}, \\ 1, & C_{Buf} \leq C_{Buf}^{Max}. \end{cases} \tag{52}$$

thể hiện sự ngưng quá trình quang hợp khi lượng $\mathrm{CH_2O}$ là C_{Buf} (mg $\{\mathrm{CH_2O}\}$ m^{-2}) sinh ra đã vượt sức chứa của cây C_{Buf}^{Max} (mg $\{\mathrm{CH_2O}\}$ m^{-2}). Thông thường, tốc độ hô hấp của cây không đáng kể so với tốc độ quang hợp của cây và có thể được lược bỏ hoặc được tính vào khoảng 1% tốc độ quang hợp của cây.

Tốc độ quang hợp ở tán lá, P, được tính thông qua công thức:

$$P = \frac{J \cdot (\mathrm{CO_{2}}_{Stom} - \Gamma)}{4 \cdot (\mathrm{CO_{2}}_{Stom} + 2\Gamma)} \tag{53}$$

trong đó, J là tốc độ vận chuyển electron (μmol $\{\mathrm{e^-}\}$ m $^{-2}$ s^{-1}), 4 là số electron của mỗi phân tử CO_2 cố định. $\mathrm{CO_{2}}_{Stom}$ là nồng độ CO_2 ở khí khổng (μmol $\{CO_2\}$ mol^{-1} {air}) và Γ là điểm bù CO_2 (μmol $\{CO_2\}$ mol^{-1} air).

Tốc độ hô hấp sáng, R_P, được xác định thông qua công thức sau:

$$R_P = P \cdot \frac{\Gamma}{\mathrm{CO_{2}}_{Stom}} \tag{54}$$

Tốc độ vận chuyển electron, J, là một hàm theo tốc độ vận chuyển electron tiềm năng (potential rate of electron transport) và theo lượng bức xạ hoạt định quang hợp hấp thụ bởi tán lá:

$$J = \frac{J^{POT} + \alpha\, PAR_{Can} - \sqrt{(J^{POT} + \alpha\, PAR_{Can})^2 - 4\Theta \cdot J^{POT} \cdot \alpha\, PAR_{Can}}}{2\Theta} \tag{55}$$

Trong đó J^{POT} là tốc độ vận chuyển electron tiềm năng (μmol $\{\mathrm{e^-}\}$ m^{-2} s^{-1}), PAR_{Can} là bức xạ hoạt tính quang hợp hấp thụ (μmol {photons} m^{-2} s^{-1}), α là hằng số chuyển đổi từ electrons sang photons và Θ là bậc của đường cong thể hiện tốc độ vận chuyển electron. Ở đây để đơn giản hoá mô hình, ta cố định PAR_{Can} là hằng số có giá trị bằng 100 μmol {photons} m^{-2} s^{-1}

Tốc độ vận chuyển electron tiềm năng J^{POT} là một hàm phụ thuộc theo nhiệt độ:

$$J^{POT} = J_{25,Can}^{MAX} \cdot \exp\left(E_j \frac{T_{Can,K} - T_{25,K}}{R \cdot 10^{-3} \cdot T_{Can,K} \cdot T_{25,K}}\right) \cdot \frac{1 + \exp\left(\frac{S \cdot T_{25,K} - H}{R \cdot 10^{-3} \cdot T_{25,K}}\right)}{1 + \exp\left(\frac{S \cdot T_{Can,K} - H}{R \cdot 10^{-3} \cdot T_{Can,K}}\right)} \tag{56}$$

Với $J_{25,Can}^{MAX}$ (μmol $\{\mathrm{e^-}\}$ m $^{-2}$ s^{-1}) là tốc độ vận chuyển electron tối đa tại 25°C của tán lá, E_j là năng lượng hoạt hoá cho J^{POT} (J mol^{-1}), $T_{Can,K}$ là nhiệt độ của tán lá (K), $T_{25,K}$ là nhiệt độ mốc 25°C, R là hằng số khí lý tưởng (J kmol^{-1} K^{-1}), S là đại lượng entropy tương ứng và H là năng lượng bất hoạt (J mol^{-1}).

Tốc độ vận chuyển electron tối đa tại 25°C ở tán lá được tính bằng công thức:

$$J_{25,Can}^{MAX} = LAI \cdot J_{25,Leaf}^{MAX} \tag{57}$$

Trong đó $J^{MAX}_{25,Leaf}$ là tốc độ vận chuyển electron tối đa cho lá ở 25°C (μmol $\{e^-\}$ m^{-2} $\{leaf\}$ s^{-1}) và LAI là *chỉ số diện tích lá (leaf area index)* sẽ được đề cập rõ hơn ở phần 2.4.2.a.

Nồng độ CO_2 trong khí khổng CO_{2Stom} trong mô hình được giả định tỷ lệ với nồng độ CO_2 ở phần phía bên dưới tấm chắn nhiệt:

$$CO_{2Stom} = \eta_{CO_{2}Air\_Stom} \cdot CO_{2Air} \tag{58}$$

Với $\eta_{CO_{2}Air\_Stom}$ là hệ số chuyển đổi từ nồng độ CO_2 trong không gian bên dưới tấm chắn nhiệt, CO_{2Air}, qua CO_{2Stom}.

Điểm bù CO_2 (Γ) là một đại lượng ảnh hưởng đến tốc độ quang hợp của lá và phụ thuộc vào nhiệt độ:

$$\Gamma = c_\Gamma T_{Can} \tag{59}$$

Với c_Γ là hằng số thể hiện sự ảnh hưởng của nhiệt độ tán lá lên điểm bù CO_2.

Quan hệ giữa nhiệt độ của tán lá và điểm bù CO_2 chỉ phù hợp với việc tính toán tốc độ quang hợp của lá. Đối với cả tán lá ta cần sử dụng công thức sau:

$$\Gamma = \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}} \cdot c_\Gamma \cdot T_{Can} + 20 c_\Gamma \left(1 - \frac{J^{MAX}_{25,Leaf}}{J^{MAX}_{25,Can}}\right) \tag{60}$$

Phần 2.4 là mô hình tham khảo bổ sung để tính tốc độ quang hợp, dù không được hiện thực trong bài làm nhưng vẫn được trình bày trong bài báo cáo.

2.4 Sự quang hợp của thực vật nhóm C3

Trong đề tài này ta chỉ xét đến thực vật thuộc nhóm C3 gồm các giống cây trồng và hoa màu nào chỉ tồn tại duy nhất theo kiểu cố định cacbon C3 như cà chua, dưa leo,... Quang hợp là quá trình sử dụng khí CO_2, nước và năng lượng từ ánh sáng mặt trời để tạo thành các hợp chất hữu cơ nuôi cây. Quá trình này chủ yếu được thực hiện nhờ *diệp lục (chlorophyll)* chứa trong *lục lạp (chloroplast)*, một bào quan đặc biệt, của tế bào lá cây và cây. Sự quang hợp diễn ra theo hai pha: pha sáng và pha tối. Ở pha sáng, lá cây hấp thụ ánh sáng mặt trời và thực hiện quá trình chuyển hoá thành năng lượng ở thành phần *thylakoid* trên lục lạp nhằm cung cấp năng lượng cho pha tối. Sản phẩm của pha sáng là NADPH *(Nicotinamide Adenine Dinucleotide phosphate)* và ATP*(Adenosine Triphosphate)*. Ở pha tối, thông qua một *chu trình Calvin*, gồm một chuỗi các phản ứng hoá sinh cố định CO_2, khử CO_2, tái tạo chất nhận CO_2 là enzyme *Rubisco* xảy ra ở *chất nền (stroma)* của lục lạp mà không cần đến ánh sáng.

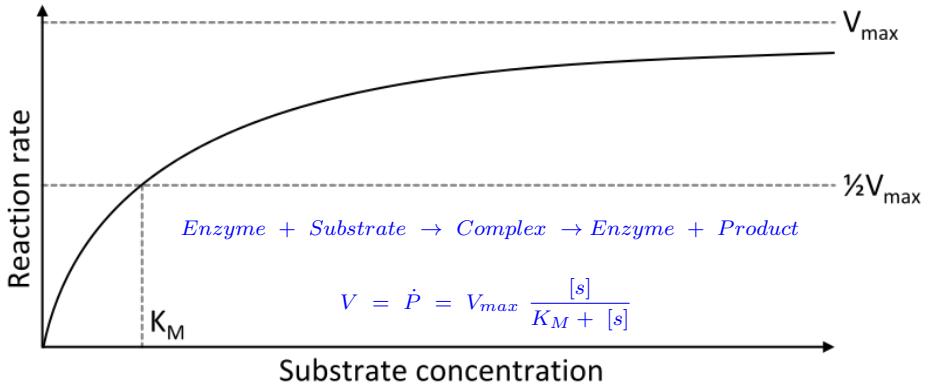
2.4.1 Mô hình quang hợp cho một đơn vị lá

Có nhiều cách để mô hình tốc độ quang hợp ở cây. Trong đề tài này ta kết hợp các mô hình lại với nhau (Các mô hình tham khảo trong [Lom+75], [Van11]).

2.4.1.a Sự khuếch tán CO_2 vào trong lá

Tốc độ quang hợp P của một đơn vị lá có thể được xem như tốc độ khí CO_2 khuếch tán từ không khí vào bên trong tế bào lá thông qua các *lỗ khí khổng (stomata)* nằm rải rác trên hai mặt lá. Quá trình khuếch tán được biểu diễn bởi định luật *Fick* cho bởi công thức:

$$P = \frac{CO_{2Air} - CO_{2Stom}}{Res} \tag{61}$$



Hình 5: Mô hình động lực Michaelis - Menten

Trong đó CO_{2Stom} là nồng độ khí CO_2 hấp thụ vào trong khí khổng (μmol m^{-3}) và Res là hệ số cản trở sự hấp thụ CO_2 vào trong tế bào lá (s m^{-1}). Hệ số cản trở này phụ thuộc vào nhiều yếu tố trong đó có tốc độ gió thổi qua lá cây.

2.4.1.b Quá trình sinh hoá ở pha tối

Các mô hình động lực *Michaelis - Menten* (được đặt tên theo nhà sinh học người Đức Leonor Michaelis và nhà vật lý người Canada Maud Menten) có thể được sử dụng để biểu diễn các quá trình sinh hoá ở pha tối của quá trình quang hợp. Quá trình này xảy ra trong chất nền của lục lạp để tạo thành phức hợp không bền và tiếp tục phân tách tái tạo lại thành enzyme và sinh ra các sản phẩm kèm theo. Một ví dụ về phản ứng hoá học ở pha tối diễn ra như sau:

- Ở bước đầu tiên, nước sẽ được tách ra thành 4 ion H^+ và 4 electron tự do cùng với sản phẩm kèm theo là khí O_2.

$$2H_2O \longrightarrow 4e^- + 4H^+ + O_2$$

- Ở bước thứ hai, CO_2 trong chất nền được kết hợp với các electron tự do và ion H^+ để tạo thành carbohydrate CH_2O và nước trở lại.

$$CO_2 + 4e^- + 4H^+ \longrightarrow CH_2O + H_2O$$

Michaelis và Menten đã nhận thấy rằng, tốc độ phản ứng cũng là tốc độ thay đổi của sản phẩm sinh ra bởi sự phân tách của phức hợp không bền đúng bằng tốc độ phản ứng ở điểm bão hoà (tốc độ tối đa mà phản ứng có thể đạt được) nhân với tỷ lệ giữa nồng độ chất tham gia phản ứng trong chất nền và tổng của chính nó với nồng độ của chất tham gia phản ứng khi tốc độ phản ứng bằng đúng 50% tốc độ phản ứng tại điểm bão hoà (Hình 5). Khi đó tốc độ quang hợp cho bởi công thức:

$$P = \frac{P_{Max} \cdot CO_{2Stom}}{CO_{2Stop} + CO_{2\,0.5}} \tag{62}$$

Trong đó $CO_{2\,0.5}$ là nồng độ khí CO_2 trong chất nền khi $P = P_{Max}/2$ (μmol m^{-3}). Giải tìm CO_{2Stom}, từ (61) và (62), tốc độ quang hợp P thoả phương trình:

$$ResP^2 - (CO_{2\,Air} + CO_{2\,0.5} + ResP_{Max})P + CO_{2\,Air} \cdot P_{Max} = 0 \tag{63}$$

Đối với phương trình bậc 2 trên, ta chỉ quan tâm đến nghiệm P sao cho $P \to P_{Max}$ khi $CO_{2\,Air} \to +\infty$. Lưu ý, lúc này tốc độ quang hợp P không còn phụ thuộc vào nồng độ CO_2 trong khí khổng nữa mà chỉ phụ thuộc vào nồng độ CO_2 trong không khí, hệ số cản trở Res và tốc độ quang hợp cực đại.

2.4.1.c Tốc độ quang hợp cực đại

Để giải phương trình (63), tốc độ quang hợp cực đại cần phải được xác định. Đối với mô hình cho sự quang hợp của một đơn vị lá, tốc độ quang hợp cực đại được xem như một hàm số phụ thuộc vào nhiệt độ của lá, năng lượng hoạt hoá và năng lượng ức chế enzyme. Thông thường, tốc độ đó sẽ được xác định bởi mô hình phản ứng hoá học *Arrhenius*:

$$k(T) = k(T_0) \exp\left(-\frac{H_a}{R}\left(\frac{1}{T} - \frac{1}{T_0}\right)\right) \tag{64}$$

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 hoá phản ứng (J mol^{-1}) và R là hằng số khí lý tưởng (J mol^{-1} K^{-1}).

Tuy nhiên, khi nhiệt độ càng cao, đến một ngưỡng nào đó, hoạt động của enzyme sẽ bị ức chế và làm giảm tốc độ của quá trình quang hợp. 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 Rubisco trong quá trình quang hợp và phụ thuộc vào nhiệt độ của lá:

$$f(T) = \frac{1 + \exp\left(-\frac{H_d}{R}\left(\frac{1}{T_0} - \left(\frac{H_d}{S}\right)^{-1}\right)\right)}{1 + \exp\left(-\frac{H_d}{R}\left(\frac{1}{T} - \left(\frac{H_d}{S}\right)^{-1}\right)\right)} \tag{65}$$

Trong mô hình (65), $f(T)$ đại diện cho sự hoạt động của enzyme ở nhiệt độ T (K), H_d là năng lượng ức chế enzyme (J mol^{-1}) và S là một đại lượng entropy tương ứng (J mol^{-1} K^{-1}).

Bằng cách kết hợp mô hình (64) và (65), tốc độ quang hợp tối đa trên mỗi đơn vị lá được xác định bởi công thức:

$$P_{Max}(T) = k(T)f(T) \tag{66}$$

2.4.2 Mô hình quang hợp cho cả tán lá

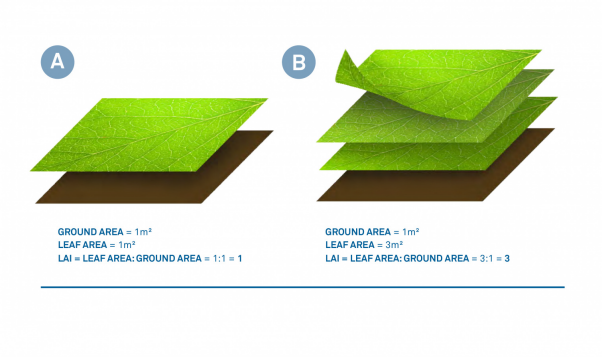
Trong phần này, ta triển khai mô hình cho cả tán lá bên trong nhà kính.

2.4.2.a Chỉ số diện tích lá

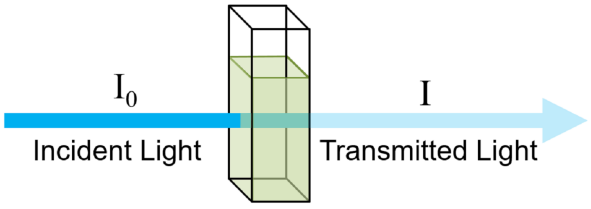
Trước hết ta cần xét đến khái niệm chỉ số diện tích lá (left area index - LAI). Chỉ số LAI được tính bởi tổng mật độ lá trên một đơn vị diện tích đất trong nhà kính. Khi đó, nếu tán lá càng dày thì chỉ số LAI càng cao (Hình 6). Chỉ số này rất quan trọng đối với mô hình quang hợp cho cả tán lá vì độ hấp thụ ánh sáng phụ thuộc chặt chẽ vào LAI. Thông qua LAI và định luật

Beer, năng lượng ánh sáng đi đến tán lá trước khi vào tán lá I_0 (μmol {photons} m^{-2} s^{-1}) và sau khi xuyên qua tán lá I (μmol {photons} m^{-2} s^{-1}) đúng bằng

$$I = \frac{I_0 \cdot K \cdot \exp(-K \cdot LAI)}{1 - m} \tag{67}$$



Hình 6: Chỉ số diện tích lá



Hình 7: Định luật Beer, cường độ tia tới giảm sau khi đi qua một lượng dung dịch

trong đó K là hệ số tắt có giá trị từ 0.7 đến 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 thường mặc định là 0.1. Khi đó, năng lượng ánh sáng lá cây nhận được là sự chênh lệch của lượng năng lượng của tia tới trước khi vào tán lá và năng lượng của tia ló sau khi đi qua tán lá và được tính bởi công thức:

$$L = L_0 \left(1 - \frac{K \cdot \exp(-K \cdot LAI)}{1 - m}\right) \tag{68}$$

Ký hiệu L là lượng photon nhận vào bởi lá cây (μmol {photons} m^{-2} s^{-1}) và L_0 là lượng photon ban đầu phía trên tán lá. Lượng photon ban đầu trên tán lá được tính theo công thức sau:

$$L_0 = \tau_{Gh} \cdot \eta_{Glob\_PAR} \cdot I_{Glob} \tag{69}$$

Trong đó, τ_{Gh} là lượng ánh sáng xuyên qua tấm chắn của nhà kính. η_{Glob_PAR} là hệ số chuyển từ bức xạ toàn phần sang bức xạ hoạt tính quang hợp. (μmol {photons} J^{-1}) và I_{Glob} là bức xạ toàn phần ở bên ngoài (W m^{-2}).

Công thức (68) chưa xét đến yếu tố phản xạ ánh sáng và sự hấp thụ bức xạ từ nền nhà kính và các vật dụng khác.

2.4.2.b Công thức Arrhenius mở rộng

Để tính giá trị P_{Max} (tốc độ quang hợp tối đa của toàn bộ là cây trong nhà kính), công thức mở rộng sau của Arrhenius được sử dụng trong công thức (66) thay cho (64):

$$k(T) = LAI \cdot k(T_0) \exp\left(-\frac{H_a}{R}\left(\frac{1}{T} - \frac{1}{T_0}\right)\right) \tag{70}$$

Ở đây, $k(T)$ là tốc độ phản ứng cho toàn bộ lá cây ở nhiệt độ T (K) và $k(T_0)$ là tốc độ phản ứng ở điều kiện tối ưu T_0 (K) của một đơn vị lá và H_a cũng là năng lượng hoạt hoá cho một đơn vị lá cây

2.4.2.c Mô hình động lực Michaelis - Menten cho P_{Max}

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.

$$P_{Max}(L,T) = \frac{P_{MLT} \cdot P_{Max}(T) \cdot L}{L + L_{0.5}} \tag{71}$$

Trong đó, $L_{0.5}$ là năng lượng ánh sáng khi $P_{Max}(L,T) = P_{Max}(L)/2$ [Lom+75] (μmol {photons} m^{-2} s^{-1}), $P_{Max}(T)$ được tính bởi công thức (66) với $k(T)$ tính theo công thức (70) và P_{MLT} là tốc độ quang hợp cực đại tại điểm bão hoà ánh sáng và nhiệt độ tối ưu T. Thông thường P_{MLT} được xác định dựa trên các nghiên cứu trước đó và thực nghiệm.

Đối với $L_{0.5}$, dựa theo [Lom+75], giá trị L_0 điển hình là 1×10^5 erg cm^{-2} s^{-1}. 1 photon có năng lượng là $2.46 \cdot 10^5$ J, thông qua các phép biến đổi, ta tính được $L_{0.5} \approx 406$ (μmol {photons} m^{-2} s^{-1}).

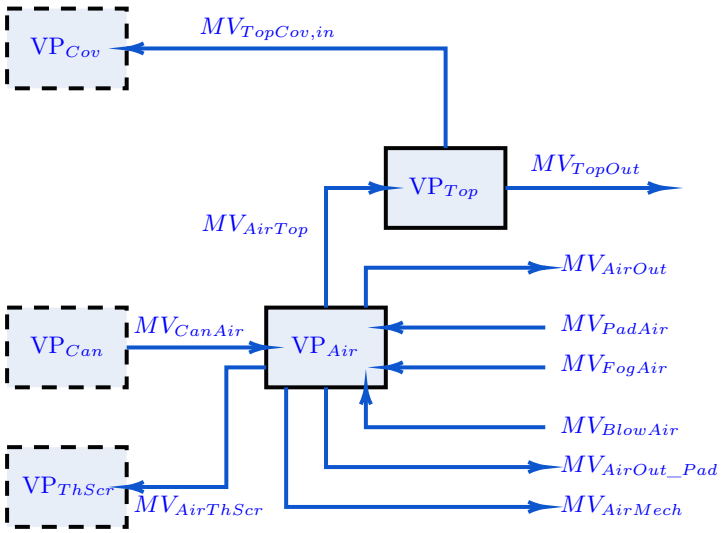
2.5 Áp suất hơi nước (Vapor Pressure)

2.5.1 Mô hình đối với áp suất hơi nước:

2.5.1.a Mô tả dòng chuyển động hơi nước:

Mô hình thứ hai được xét tới trong báo cáo này là mô hình áp suất hơi nước mà cụ thể là áp suất hơi nước ở gian trên và gian dưới nhà kính. Cũng giống như mô hình đối với lượng khí CO_2, gian trên và gian dưới của nhà kính được phân cách bởi một tấm chắn nhiệt. Hơi nước ở gain trên và gian dưới nhà kính được hòa lẫn vào nhau nếu tấm chắn nhiệt được mở ra hoặc tháo dỡ hoàn toàn.

Nói cách khác, trong hầu hết thời gian, gian trên và gian dưới nhà kính cũng có áp suất hơi nước khác nhau. Sơ đồ tóm tắt sự lưu thông của hơi nước được miêu tả qua hình sau.



Hình 8: Dòng chuyển động của hơi nước bên trong và bên ngoài nhà kính

Đối với gian dưới của nhà kính, tương tự như mô hình CO_2, hơi nước lưu thông được đưa vào qua các hệ thống hệ thống thông gió và thoát ra ngoài qua hệ thống quạt (MV_{PadAir}, MV_{AirOut}). Một phần hơi nước cũng thẩm thấu ra bên ngoài theo hệ thông thông gió (MV_{AirOut_Pad}). Ngoài ra, lượng hơi nước bên trong còn được cung cấp bởi quá trình thoát hơi nước của lá cây (MV_{CanAir}) và từ luồng khí thổi vào bởi máy sưởi($MV_{BlowAir}$). Thất thoát hơi nước còn diễn ra ở gian dưới do hơi nước cô đọng tại màn chắn nhiệt và được loại bỏ ở các hệ thống nhà kính hiện đại ($MV_{AirThScr}$) và dòng khí từ gian dưới lên gian trên (MV_{AirTop}), cùng với hệ thống làm mát cơ học (MV_{Mech}). Khác với mô hình cho CO_2, hơi nước trong nhà kính còn được cung cấp bởi hệ thống phun sương (MV_{FogAir}). Hệ thống này có tác dụng cung cấp đủ độ ẩm cho nhà kính qua việc phun trực tiếp hơi nước vào, đây là nguồn chính thay đổi lượng hơi nước bên trong nhà kính.

Hình 9: Hệ thống phun sương điều tiết độ ẩm

Đối với gian trên nhà kính, hơi nước chỉ đến từ một nguồn duy nhất là từ gian dưới đi lên theo dòng lưu thông của không khí (MV_{AirTop}). Sự thất thoát hơi nước tại dây bao gồm thất thoát qua các lỗ thông hơi ra bên ngoài (MV_{TopOut}) và ngưng đọng hơi nước ở bề mặt lớp che phủ ($MV_{TopCov,in}$).

2.6 Mô hình hệ động lực biểu diễn chuyển động hơi nước và giả thiết:

Trong mục này mô hình Hệ động lực biểu diễn dòng chuyển động của hơi nước bên trong và bên ngoài nhà kính sẽ được đề cập tới. Dựa trên sơ đồ đã nêu, sự thay đổi áp suất hơi nước ở gian trên và gian dưới bên trong nhà kính được biểu diễn qua hệ gồm hai phương trình sau đây:

$$\begin{cases} cap_{VP_{Air}} \dot{VP_{Air}} = & MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BlowAir} - MV_{AirThScr} \\ & -MV_{AirTop} - MV_{AirOut} - MV_{AirOut\_Pad} - MV_{Mech} \\ cap_{VP_{Top}} \dot{VP_{Top}} = & MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut} \end{cases} \tag{72}$$

Với các giá trị capacities:

$$cap_{VP_{Air}} = \frac{M_{Water} h_{Air}}{R(T_{Air} + 273.15)} \tag{73}$$

$$cap_{VP_{Top}} = \frac{M_{Water} h_{Top}}{R(T_{Top} + 273.15)} \tag{74}$$

Tương tự như hệ phương trình đối với CO_2, các giả thiết đã được xét đến bao gồm không gain bên trong nhà kính không chịu ảnh hưởng bởi bất kỳ yếu tố nào ngoài trừ những đại lượng được đề trong hệ phương trình, hơn nữa hơi nước được xem như phân bố đều. Về các đại lượng: cap_{VP_A} là áp suất hơi nước tối đa trong A (kg m^3 J^{-1}), VP_A là áp suất hơi nước tại A (kg m^{-1} s^{-2}), $\dot{VP_A}$ là tốc độ thay đổi áp suất hơi nước trong A (kg m^{-2} s^{-1}), MV_{AB} là lưu lượng hơi nước từ A sang B (kg m^{-2} s^{-1}).

Dựa theo Apenndix H, trang 208 [De 96], áp suất hơi bão hòa VP_A được tính như sau:

$$VP_A = 610.78 \cdot exp(t/(T_A + 238.3) * 17.2694) \tag{75}$$

Dưới đây là một số công thức tính MV_{AB}. Trước hết, xét lượng hơi nước từ máy sưởi vào gian dưới nhà kính như sau:

$$MV_{BlowAir} = \frac{\eta_{HeatVap} U_{Blow} P_{Blow}}{A_{Flr}} \tag{76}$$

Trong đó, $\eta_{HeatVap}$ (kg {vapour} J^{-1}) là tổng lượng hơi nước khi 1 Joule năng lượng (cảm nhận được) sinh ra bởi máy sưởi. U_{Blow} là độ mở của van máy sưởi nằm trong đoạn $[0, 1]$ và không có đơn vị, P_{Blow} (W) là công suất sinh hơi nước của máy sưởi

Tương tự, lượng hơi nước cung cấp từ hệ thống phun sương được cho theo công thức:

$$MV_{FogAir} = \frac{U_{Fog}\phi_{Fog}}{A_{Flr}} \tag{77}$$

Trong đó, U_{Fog} là van điều khiển của hệ thống phun sương, có giá trị trong đoạn $[0, 1]$ và không có đơn vị, ϕ_{Fog} (kg water s^{-1}) là mức cung cấp tối đa của hệ thống phun sương.

Với sự chênh lệch áp suất hơi nước bên trong và bên ngoài nhà kính, khả năng cho dòng khí đi qua tấm thông gió, ta có công thức tính MV_{PadAir}

$$MV_{PadAir} = \rho_{Air} \frac{U_{Pad}\phi_{Pad}}{A_{Flr}} (\eta_{Pad}(x_{Pad} - x_{Out}) + x_{Out}) \tag{78}$$

Với U_{Pad} là van điều khiển của tấm thông gió, có giá trị trong đoạn $[0, 1]$ và không có đơn vị, ϕ_{Pad} ($m^3 s^{-1}$) là lượng khí tối đa thông qua tấm thông gió, η_{Pad} là hệ số tương quan giữa hệ thống quạt và tấm thông gió, x_{Pad} (kg water kg^{-1} air) là lượng nước có trong không khí ở tấm thông gió và x_{Out} (kg water kg^{-1} air) là lượng nước có trong không khí ở ngoài trời.

Cũng tại tấm thông gió, một lượng hơi nước cũng bị thất thoát ra ngoài MV_{AirOut_Pad}

$$MV_{AirOut\_Pad} = \frac{U_{Pad}\phi_{Pad}}{A_{Flr}} \frac{M_{Water}}{R} \frac{VP_{Air}}{T_{Air} + 273.15} \tag{79}$$

Trong đó, M_{Water} là khối lượng mol của nước, R là hằng số phân tử khí, VP_{Air} là áp suất hơi nước ngoài trời và T_{Air} là nhiệt độ ngoài trời.

Khác với mô hình CO_2, lượng hơi nước trong không khí còn bị ảnh hưởng theo hệ thống làm mát cơ học, miêu tả theo công thức sau:

Hình 10: Hệ thống làm mát cơ học

$$MV_{AirMech} = 6.04{\cdot}10^{-9}\frac{U_{MechCool}COP_{MechCool}P_{MechCool}/A_{Flr}}{T_{Air} - T_{MechCool} + 6.4 \cdot 10^{-9}\Delta H(VP_{Air} - VP_{MechCool})}(VP_{Air}{-}VP_{Mech}) \tag{80}$$

Trong đó, $U_{MechCool}$ là van điều khiển của hệ thống làm mát, $COP_{MechCool}$ là hệ số hiệu năng của hệ thống, $P_{MechCool}$ là công suất tối đa của hệ thống. $T_{MechCool}$ là nhiệt độ bề mặt làm mát. $VP_{MechCool}$ là áp suát hơi nước bão hòa của hệ thống.

Sự ngưng tụ của hơi nước từ không khílên các bề mặtđược mô tả theo công thức sau:

$$\begin{cases} MV_{12} = 0 & VP_1 < VP_2 \\ MV_{12} = 6.49 \cdot 10^{-9} HEC_{12}(VP_1 - VP_2) & VP_1 > VP_2 \end{cases} \tag{81}$$

Để các hàm này có đạo hàm tại mọi điểm, một hàm chuyển đổi được sử dụng:

$$MV_{12} = \frac{1}{1 + exp(s_{MV_{12}}(VP_1 - VP_2))} 6.49 \cdot 10^{-9} HEC_{12}(VP_1 - VP_2) \tag{82}$$

Do đó lượng hơi nước mất đi do hiện tượng ngưng tụ tại tấm chắn nhiệt cũng được miêu tả theo công thức:

$$MV_{AirThSCr} = \frac{1}{1 + exp(s_{MV_{12}}(VP_{Air} - VP_{AirThSCr}))} 6.04{\cdot}10^{-9}{\cdot}HEC_{AirThScr}(VP_{Air}{-}VP_{AirThSCr}) \tag{83}$$

Với giá trị hệ số trao đổi nhiệt $HEC_{AirThScr}$:

$$HEC_{AirThScr} = 1.7U_{ThScr}|T_{Air} - T_{ThScr}| \tag{84}$$

Trong đó, U_{ThScr} độ mở của tấm chắn nhiệt, tức vùng che phủ của nó trong nhà kính.
Và cũng như vậy đối với lớp mái che phía trên nhà kính:

$$MV_{TopCov,in} = \frac{1}{1 + exp(s_{MV_{12}}(VP_{Air} - VP_{Topcov,in}))} 6.04{\cdot}10^{-9} \cdot HEC_{TopCov,in}(VP_{Air} - VP_{Topcov,in}) \tag{85}$$

Với giá trị hệ số trao đổi nhiệt$HEC_{TopCov,in}$:

$$HEC_{TopCov,in} = c_{HECin}(T_{Top} - T_{Cov,in})^{0.33} \frac{A_{Cov}}{A_{Flr}} \tag{86}$$

Trong đó, giá trị c_{HECin} là hệ số trao đổi nhiệt giữa lớp che phủ và không khí ở môi trường bên ngoài.
Đối với lượng hơi nước đi từ gian dưới lên gian trên nhà kính, ta cũng có công thức như sau:

$$MV_{AirTop} = \frac{M_{Water}}{R} f_{ThScr} \left(\frac{VP_{Air}}{T_{Air} + 273.15} - \frac{VP_{Top}}{T_{Top} + 273.15} \right) \tag{87}$$

Với f_{ThScr} tính theo (35)
Tương tự, để biểu diễn luồng hơi nước tử bên trong hai gian nhà kính ra ngoài qua các ô thông gió, ta sử dụng các công thức như sau đây:

$$MV_{AirOut} = \frac{M_{Water}}{R} (f_{VentSide} + f_{VentForced}) \left(\frac{VP_{Air}}{T_{Air} + 273.15} - \frac{VP_{Out}}{T_{Out} + 273.15} \right) \tag{88}$$

Với $f_{VentSide}$ và $f_{VentForced}$ tính theo (46) và (47)
Và với luồng hơi nước từ gian nhà trên ra ngoài thông qua ô mở trên mái:

$$MV_{TopOut} = \frac{M_{Water}}{R} f_{VentRoof} \left(\frac{VP_{Top}}{T_{Top} + 273.15} - \frac{VP_{Out}}{T_{Out} + 273.15} \right) \tag{89}$$

Với $f_{VentRoof}$ tính theo (49)
Cuối cùng, chúng ta cần xét đến sự thoát hơi nước của tán lá:

$$MV_{CanAir} = VEC_{CanAir}(VP_{Can} - VP_{Air}) \tag{90}$$

Với giá trị VEC_{CanAir}:

$$VEC_{CanAir} = \frac{2\rho_{Air} c_{p,Air} LAI}{\Delta H \gamma (r_b + r_s)} \tag{91}$$

Trong đó, ρ_{Air} (kg m^{-3}) là khối lượng riêng của không khí, $c_{p,Air}$ (J K^{-1} kg^{-1}) là nhiệt dung của không khí, ΔH (J kg^{-1}) là nhiệt hóa hơi của hơi nước, γ (Pa K^{-1}) là hằng số độ ẩm, r_b (s

m^{-1}) biểu diễn sự cản trở thoát hơi nước ở lớp bao ngoài tán lá, r_s (s m^{-1}) là sự cản trở của khí khổng trong quá trình thoát hơi nước.

Sự cản trở thoát hơi nước của lớp bao ngoài được mô tả phụ thuộc vào tốc độ gió bên trong nhà kính theo mô hình thiết lập trong [Sta87]. Tuy nhiên ở mô hình nhà kính được xem xét trong bài báo cáo này, ta không thực hiện việc đo hay mô phỏng tốc độ gió bên trong nhà kình và do đó ta sẽ xét giá trị r_b là một hằng số (275 sm^{-1}). Thay vào đó, với mô hình của [Sta87], sự cản trở trao đổi hơi nước ở khí khổng được xét theo công thức:

$$r_s = r_{s,min} \cdot rf_{RCan} \cdot rf(CO_{2Air\_ppm}) \cdot rf(VP_{Can} - VP_{Air}) \tag{92}$$

Với $r_{s,min}$ (s m^{-1}) là độ cản trở tối thiểu của khí khổng và rf là các nhân tố thay đổi giá trị r_s liên quan đến năng lượng bức xạ cao, lượng CO_2 cao và độ chênh lệch áp suất hơi lớn giữa tán lá và không khí trong nhà kính. Các giá trị rf được mô tả trong [Sta87] như sau:

$$rf(R_{can}) = \frac{R_{can} + c_{evap1}}{R_{can} + c_{evap2}} \tag{93}$$

$$rf(CO_{2Air}) = 1 + c_{evap3}(CO_{2Air} - 200)^2 \tag{94}$$

$$rf(VP_{can} - VP_{Air}) = 1 + c_{evap4}(VP_{can} - VP_{Air}) \tag{95}$$

Trong đó, R_{can} (W m^{-2}) là bức xạ mặt trời phía trên tán lá, c_{evap1} (W m^{-2}), $c_{evap2)}$ (W m^{-2}), c_{evap3} (ppm^{-2}), c_{evap4} (Pa^{-2}) là các giá trị quan sát được trong quá trình vận hành nhà kính. Trong mô hình trao đổi hơi nước này, Stanghellini đã giới hạn $rf(CO_{2Air})$ và $rf(VP_{can} - VP_{Air})$ lại lần lượt là 1.5 và 5.8. Trong khi đó, R_{can} được lấy xấp xỉ 65% tổng bức xạ mặt trời mà nhà kính nhận được.

Do các giá trị c_{evap3} và c_{evap4} thay đổi khác nhau vào buổi sáng và tối nên ở các thời điểm bình minh và hoàng hôn, các hàm (94) và (95) không có đạo hàm. Do đó, ta sử dụng hàm chuyển đổi để các hàm này liên tục và có đạo hàm tại tất cả các điểm:

$$S_{r_s} = \frac{1}{1 + exp(s_{r_s}(R_{Can} - R_{Can\_SP}))} \tag{96}$$

Trong đó, s_{r_s} (mW^{-2}) là độ dốc hàm, R_{Can_SP} (Wm^{-2}) là bức xạ phía trên tán lá để định nghĩa bình minh và hoàng hôn. Với giá trị chuyển đổi, hàm định nghĩa c_{evap3} và c_{evap4} như sau:

$$c_{evap3} = c_{evap3}^{night}(1 - S_{r_s}) + c_{evap3}^{day} S_{r_s} \tag{97}$$

$$c_{evap4} = c_{evap4}^{night}(1 - S_{r_s}) + c_{evap4}^{day} S_{r_s} \tag{98}$$

2.7 Hiện thực chương trình tính các công thức bằng Python

2.7.1 Các công thức liên quan đến mô hình nồng độ CO_2 và mô hình áp suất hơi nước

Việc hiện thực các công thức dựa trên các công thức đã trình bày ở trong báo cáo, với các tham số là các hệ số và biến số tham gia vào từng công thức tương ứng. Các tham số được chú thích

bên trong hàm để thuận tiện hơn cho việc sử dụng. Chi tiết các công thức đã hiện thực trên `Python` script được đính kèm với bài báo cáo và được liệt kê ở phần phụ lục A và B.

2.7.2 Các giả thiết

1. Nồng độ CO_2 ở bên ngoài nhà kính có nồng độ không đổi 668 mg m^{-3}.
2. Hệ thống nhà kính chỉ sử dụng thông gió ở mái, không sử dụng thông gió ở 2 bên tường.
3. Hệ thống nhà kính không sử dụng tấm che nắng.
4. Hệ thống nhà kính không sử dụng hệ thống quạt bên trong.
5. Hệ thống nhà kính sử dụng hệ thống bơm khí CO_2 có capacity 72000 mg s^{-1}.
6. Bức xạ mặt trời mà tán lá cây nhận được lấy xấp xỉ 65% lượng bức xạ đo từ trạm thời tiết.
7. Nhiệt độ gian trên nhà kính cao hơn gian dưới 1°C.
8. Nhiệt độ của tán cây cao hơn nhiệt độ phần gian dưới nhà kính 15°C.

2.7.3 Danh sách ký hiệu các thông số và giá trị của chúng:

Sử dụng dữ liệu từ [Van11] và từ trang web https://github.com/CEAOD/Data/tree/master/GH_Cucumber/AutonomousGreenhouseChallengeFirstEdition(2018), danh sách các thông số và giá trị sẽ được sử dụng để tính toán trong mô hình như sau: (*Các thông số không được dùng trong thí nghiệm sẽ có giá trị ký hiệu bởi dấu* $\times$*, khi tính toán sẽ có giá trị = 0)*

Ký hiệu	Giá trị	Đơn vị
$cap_{CO_{2Air}}$	3.8	m
$cap_{CO_{2Top}}$	0.4	m
η_{HeatCO_2}	0.057	mg $\{CO_2\}$ J^{-1}
P_{Blow}	$\times$	W
U_{Blow}	[0,1]	
A_{flr}	$1.4 \cdot 10^4$	m^2
g	9.81	m s^{-2}
U_{ExtCO_2}	[0,1]	
ϕ_{ExtCO_2}	$7.2 \cdot 10^4$	mg s^{-1}
U_{Pad}	[0,1]	
ϕ_{Pad}	$\times$	m^3 s^{-1}
U_{ThScr}	[0,1]	
K_{ThScr}	$0.05 \cdot 10^{-3}$	m^3 $K^{-\frac{2}{3}}$ s^{-1}
T_{Air}	Thay đổi theo thời gian	°C
T_{Top}	Thay đổi theo thời gian	°C
T_{Out}	Thay đổi theo thời gian	°C
M_{Air}	28.96	kg $kmol^{-1}$
h_{Air}	3.8	m
h_{Top}	0.4	m
$h_{Elevation}$	0	
ρ_{Air0}	1.20	kg m^{-3}
C_d^{Gh}	0.75	
C_w^{Gh}	0.09	
η_{ShScrC_w}	$\times$	
η_{ShScrC_d}	$\times$	
U_{ShScr}	[0,1]	

CO_{2Out}	668 (Giả thiết theo [Van11])	mg m^{-3}
U_{Roof}	[0,1]	
U_{Side}	[0,1]	
A_{Roof}/A_{Flr}	0.1	
A_{Side}/A_{Flr}	0	
$h_{SideRoof}$	×	m
h_{Roof}	0.68	m
v_{Wind}	Thay đổi theo thời gian	m s^{-1}
ζ_{InsScr}	1	
$c_{leakage}$	$1{\cdot}10^{-4}$	
η_{Side}	0	
η_{Roof_Thr}	0.9 (Giả sử theo [Van11])	
$U_{VentForced}$	[0,1]	
$\phi_{VentForced}$	×	
M_{CH_2O}	$30{\cdot}\ 10^{-3}$	mg {CH_2O} $\mu$$mol^{-1}$
$h_{C_{buf}}$	1	
C_{Buf}^{Max}	$20{\cdot}10^{3}$	mg {CH_2O} m^{-2}
Θ	0.7	
α	0.385	μmol {e^-} $\mu$$mol^{-1}$ {photons}
J^{POT}	Tính theo (56)	
PAR_{Can}	100 (Giả thiết)	μmol {photons} m^{-2} s^{-1}
E_j	$37{\cdot}10^{3}$	J mol^{-1}
$T_{25,K}$	298.15	K
R	$8.314{\cdot}10^{3}$	J $kmol^{-1}$ K^{-1}
H	$22{\cdot}10^{4}$	J mol^{-1}
S	710	J mol^{-1} K^{-1}
$J_{25,Leaf}^{MAX}$	210	μmol {e^-} m^{-2} {leaf} s^{-1}
LAI	Giá trị trung bình: 2.5	
c_Γ	1.7	μmol {CO_2} $\mu$$mol^{-1}$ {air} K^{-1}
$\eta_{HeapVap}$	$4.43{\cdot}\ 10^{-8}$	kg {vapour} J^{-1}
U_{Fog}	[0,1]	
ϕ_{Fog}	×	kg {water} s^{-1}
η_{Pad}	×	
M_{Water}	18	kg $kmol^{-1}$
$U_{MechCool}$	[0,1]	
$COP_{MechCool}$	×	
$P_{MechCool}$	×	
$T_{MechCool}$	Thay đổi theo thời gian	
ΔH	$2.45{\cdot}\ 10^{6}$	J kg^{-1} {water}
$s_{MV_{12}}$	-0.1	Pa^{-1}
c_{HECin}	1.86	W m^{-2} K^{-2}
$c_{p,Air}$	10^{3}	J K^{-1} kg^{-1}
γ	65.8	Pa K^{-1}
r_b	275	s m^{-1}
r_s	$r_{s,min} = 82$	s m^{-1}
R_{canSP}	5	W m^{-2}
c_{evap1}	4.30	W m^{-2}
c_{evap2}	0.54	W m^{-2}
c_{evap3}^{day}	$6.1{\cdot}\ 10^{-7}$	ppm^{-2}

c_{evap3}^{night}	$1.1 \cdot 10^{-11}$	ppm^{-2}
c_{evap4}^{day}	$4.3 \cdot 10^{-6}$	Pa^{-2}
c_{evap4}^{night}	$5.2 \cdot 10^{-6}$	Pa^{-2}
x_{pad}	0	kg {water} kg^{-1} {vapour}
A_{cov}	$1.8 \cdot 10^{4}$	m^2
h_{ThScr}	$0.35 \cdot 10^{-3}$	m

2.7.4 Dataset và các số liệu đã sử dụng

Sử dụng bộ Dataset trong của đội `Sonoma` (https://github.com/CEAOD/Data/tree/master/GH_Cucumber/AutonomousGreenhouseChallengeFirstEdition(2018)/Sonoma) với 2 file `Greenhouse_climate.csv` và `meteo.csv` để lấy giữ liệu ban đầu và làm dữ liệu để so sánh.

Ở file `Greenhouse_climate.csv`, các field đã được sử dụng và mô tả của chúng như sau:

Cột	Mô tả	Đơn vị	Interval time
`Tair`	Nhiệt độ không khí của nhà kính	°C	5 phút
`RHair`	Độ ẩm tương đối của nhà kính	%	5 phút
`CO2air`	Nồng độ CO2 bên trong nhà kính	ppm	5 phút
`VentLee`	Độ thông gió ở hướng khuất gió	%(0 - 100%)	5 phút
`VentWind`	Độ thông gió ở hướng đón gió	%(0 - 100%)	5 phút

Do nhà kính sử dụng để validate mô hình không có sử dụng thông gió ở hai bên tường, cho nên chênh lệch của độ thông gió ở hướng khuất gió và hướng đón gió được sử dụng để làm giá trị cho biến điều khiển U_{Roof}. Các field `CO2air` và `RHair` được dùng để kiểm tra mô hình.

Ở file `meteo.csv`, các field đã được sử dụng và mô tả của chúng như sau:

Cột	Mô tả	Đơn vị	Interval time
`Tout`	Nhiệt độ không khí ở bên ngoài	°C	5 phút
`RHout`	Độ ẩm tương đối ở bên ngoài	%	5 phút
`Windsp`	Tốc độ gió	$m\ s^{-1}$	5 phút

3 Chạy thử nghiệm với mô hình CO_2

3.1 Bài toán 3

Ở bài toán này, chúng ta sẽ xác định giá trị của $\dot{CO}_{2Air}$ và $\dot{CO}_{2Top}$ tại thời điểm t thông qua hàm **dx** đã được định nghĩa ở bài toán 2. Dữ liệu được tham khảo từ trang web https://github.com/CEAOD/Data/tree/master/GH_Cucumber/AutonomousGreenhouseChallengeFirstEdition(2018). Các biến số thay đổi theo thời gian T_{out}, v_{wind} được trích từ file `meteo.csv`. Đây là file dữ liệu môi trường tại nơi các đội tham gia cuộc thi Autonomous Greenhouse Challenge 2018 được tổ chức tại Hà Lan. Các biến số T_{Air}, U_{Roof} và giá trị đối chiếu CO_{2Air} được trích dẫn từ file `Greenhouse_climate.csv` của đội **Sonoma** - đội quán quân của cuộc thi. Giá trị U_{Roof} được tính bằng công thức $U_{Roof} = \frac{\texttt{ventLee+ventWind}}{2}\%$, với hai thông số `VentLee` và `VentWind` được trích từ tập dữ liệu của đội. Về mặt khách quan, các thông số từ file `Greenhouse_climate.csv` trên được điều chỉnh bởi các thành viên trong đội **Sonoma** và họ ghi nhận lại kết quả thông qua các cảm biến có sẵn, xử lý và tính toán dữ liệu. Việc đánh giá mô hình toán học đang được xây dựng có tính đúng đắn hay không sẽ phụ thuộc vào kết quả xấp xỉ giá trị CO_{2Air} và giá trị CO_{2Top} tại mỗi thời điểm có tiệm cận với dữ liệu thực tế hay không. Và hướng thay đổi của đồ thị có giống với hướng thay đổi của dữ liệu thực tế hay không.

Các hằng số có liên quan đã được định nghĩa trong mục 2.7.3. Trích trong tập dữ liệu `meteo.csv` và `Greenhouse_climate.csv`, dòng **4**, ta lấy được các giá trị biến số thay đổi theo thời gian như sau:

- v_{Wind} = `Windsp` = 3.2 m s^{-1}
- `Tout` = 17.7 °C
- `VentLee` = 52.9 %
- `VentWind` = 2.9 %
- U_{ThScr} = 0
- `Tair` = 19.8999999966472 °C
- CO_{2air} = 427 ppm
- $CO_{2SetPoint}$ = `NaN`

Lúc này, do không ghi nhận được giá trị của $CO_{2SetPoint}$ nên giá trị U_{extCO_2} sẽ được chọn random trong khoảng `[0, 0.3]`. Trong trường hợp này, ta chọn $U_{extCO_2} = 0.3$.

Tuy nhiên, có một vài biến số cũng có thay đổi theo thời gian nhưng không được ghi nhận trong tập dữ liệu. Trong trường hợp này, các giá trị sau được lựa chọn theo nguyên nhân khách quan (để đạt tính chính xác cao hơn):

- $U_{Blow} = 0.2$
- $U_{VentForced} = 0.5$
- $U_{Pad} = 0.5$
- $U_{Roof} = \frac{52.9 + 2.9}{2}\% = 0.279$

- $U_{Side} = 0.5$
- $U_{ShScr} = 0.5$
- $T_{Top} = \texttt{Tair} + 1 = 20.8999999966472$ °C
- $T_{Can} = \texttt{Tair} + 15 = 34.8999999966472$ °C
- $C_{Buf} = 20{\cdot}10^{-3}$ mg $\{CH_2O\}$ m^{-2}

Vì CO_{2Top} không có số liệu để thử nghiệm nên trong trường hợp này, ban đầu, nồng độ CO_{2Top} sẽ bằng với nồng độ của CO_{2Air} và có giá trị là 427 ppm. Lưu ý rằng, vì đơn vị của tốc độ thay đổi nồng độ khí CO_2 là mg m^{-3} s^{-1} nên đơn vị thời gian t trong bài toán này là **giây**

Sau khi có các giá trị cho từng tham số đầu vào của hàm **dx**, sử dụng công thức đã được hiện thực bằng Python ở phụ lục A, chạy thử nghiệm với các giá trị trên, ta thu được kết quả sau:

```
CO2_air_diff = 0.14144976328287065 (ppm/s)
CO2_top_diff = -1.4020894430178459 (ppm/s)
```

Như vậy, tại thời điểm t_0 đang xét, tốc độ thay đổi nồng độ của khí CO_2 trong nhà kính là **0.14144976328287065 (ppm/s)** và bên ngoài nhà kính là **-1.4020894430178459 (ppm/s)**. Kết quả này cho thấy nồng độ khí CO_2 có xu hướng tăng khi ở điều kiện có tham số đầu vào như trên. Đối chứng với số liệu của CO_{2Air} trong tập dữ liệu, ta thấy trong 5 phút tiếp theo, nồng độ CO_{2Air} ở mức 443 (ppm), tăng 16 (ppm) so với thời gian trước đó. Giả sử rằng tốc độ thay đổi nồng độ khí CO_2 trong nhà kính là không thay đổi trong khoảng thời gian các cảm biến cập nhật dữ liệu mới (tức là trong vòng 5 phút), nồng độ CO_{2Air} lúc này tăng một lượng bằng $0.14144976328287065 \times 300 \approx 42.42$ (ppm). Ta thấy kết quả chạy thực nghiệm bị sai lệch so với thực tế là 26.42 (ppm). Nguyên nhân cho việc dẫn đến sai số là mô hình toán học đang xây dựng có nhiều giả thiết không đúng với thực tế cũng như có nhiều thông số bị bỏ qua. Tuy nhiên, việc tính toán được xu hướng tăng của mật độ CO_{2Air} trong trường hợp này cũng thể hiện được tính đúng đắn một phần của mô hình. Do CO_{2Top} không có giá trị thực tế để đối chứng nên chúng ta không thể đánh giá được tính đúng đắn của tham số này.

Ở trường hợp thứ hai, chúng ta sử dụng dữ liệu ở dòng **100** trong tập dữ liệu.

- $v_{Wind} = \texttt{Windsp} = 6.3$ m s^{-1}
- $\texttt{Tout} = 17.8$ °C
- $\texttt{VentLee} = 28.4$ %
- $\texttt{ventWind} = 1.6$ %
- $U_{ThScr} = 0$
- $\texttt{Tair} = 21.8999999966472$ °C
- $CO_{2Air} = 457$ ppm
- $CO_{2SetPoint} = \texttt{NaN}$

Lúc này, giá trị $U_{Roof} = \dfrac{28.4 + 1.6}{2}\% = 0.3$. Cũng tương tự như trường hợp trên, giá trị U_{ExtCO_2} có giá trị là 0.3. Các giá trị biến số có thay đổi theo thời gian còn lại vẫn giữ giá trị như trong trường hợp trên. Giá trị nồng độ CO_{2Top} sau 97 lần lấy dữ liệu sẽ có sự khác biệt so

với nồng độ CO_{2Air}. Trong trường hợp này, giá trị CO_{2Top} bằng 410 (ppm). Chạy chương trình, ta thu kết quả như sau:

```
CO2_air_diff = -0.06791671473551529 (ppm/s)
CO2_top_diff = -0.010278697335806134 (ppm/s)
```

So với dữ liệu thực tế từ tập dữ liệu, giá trị CO_{2Air} trong 5 phút tiếp theo là 443 (ppm), giảm một lượng là 13 (ppm). Giả sử rằng tốc độ thay đổi của nồng độ khí CO_2 trong nhà kính là không thay đổi trong 5 phút tiếp theo, nồng độ CO_{2Air} lúc này giảm một lượng $-0.06791671473551529 \times 300 \approx -20.37$ (ppm). So sánh với dữ liệu thực tế, sai số trong trường hợp này là 7.37 (ppm). Kết quả cho thấy nồng độ CO_2 trong nhà kính đang có xu hướng giảm với tập dữ liệu trên và khớp với xu hướng thay đổi nồng độ khí CO_2 so với thực tế.

3.2 Bài toán 4

3.2.1 Các thuật giải cho hệ các phương trình vi phân thường bậc nhất

3.2.1.a Phương pháp Explicit Euler

Tham khảo từ [EG96] và phần giới thiệu giải thuật Explicit Euler ở phần 1.6.1, từ công thức (30), ta có hệ phương trình vi phân thường bậc nhất như sau:

$$\begin{cases} \dot{CO_{2Air}} &= \dfrac{MC_{BlowAir} + MC_{ExtAir} + MC_{PadAir} - MC_{AirCan} - MC_{AirTop} - MC_{AirOut}}{cap_{CO_{2Air}}} \\ \dot{CO_{2Top}} &= \dfrac{MC_{AirTop} - MC_{TopOut}}{cap_{CO_{2Top}}} \end{cases} \tag{99}$$

Trong hệ phương trình này, giá trị nồng độ CO_{2Air} và CO_{2Top} là các biến số. Theo lý thuyết, 2 biến số này phải là một hàm số được biểu diễn theo thời gian vì $\dot{CO}_{2Air} = \dfrac{\Delta CO_{2Air}}{\Delta t}$ mô tả tốc độ thay đổi của nồng độ khí CO_{2Air} và có đơn vị là (mg m^{-3} s^{-1}). Tương tự với CO_{2Top}. Đặt hàm số f bằng vế phải của phương trình vi phân thứ nhất, hàm số g bằng vế phải của phương trình vi phân thứ hai. Ta có được hệ phương trình vi phân tổng quát như sau:

$$\begin{cases} \dot{CO_{2Air}}(t) = f(t, CO_{2Air}(t), CO_{2Top}(t)) \\ \dot{CO_{2Top}}(t) = g(t, CO_{2Air}(t), CO_{2Top}(t)) \\ CO_{2Air}(t_0) = \alpha, CO_{2Top}(t_0) = \beta \end{cases} \tag{100}$$

Từ công thức (16), ta có thể suy ra được công thức explicit Euler trong bài toán này:

$$\begin{cases} CO_{2Air}(t+1) = CO_{2Air}(t) + hf(t, CO_{2Air}(t), CO_{2Top}(t)) \\ CO_{2Top}(t+1) = CO_{2Top}(t) + hg(t, CO_{2Air}(t), CO_{2Top}(t)) \end{cases} \tag{101}$$

Các biến thay đổi theo thời gian, nếu có dữ liệu trong tập dữ liệu, hoặc có công thức chuyển đổi từ các dữ liệu có sẵn được xem như là hằng số tại thời gian t đang xét. Nếu tại thời điểm t, các thông số chưa được cập nhật mới, giá trị các biến số sẽ là giá trị cũ trước đó, ngược lại, giá trị các biến số sẽ là giá trị mới cập nhật được. Việc trích xuất các giá trị này có thể thực hiện một cách chủ quan. Do đó, trong hệ phương trình này, chúng ta không sử dụng biến số t mà sẽ được mặc định hiểu có sự thay đổi dữ liệu thông qua sự thay đổi của các biến số. Ta kí hiệu ở bước thời gian thứ n là t_n và nghiệm ở bước thứ n là CO_{2Air_n}, CO_{2Top_n}. Khi đó, nghiệm bài toán

được tính bằng:

$$\begin{cases} CO_{2Air_{n+1}} = CO_{2Air_n} + hf(CO_{2Air_n}, CO_{2Top_n}) \\ CO_{2Top_{n+1}} = CO_{2Top_n} + hg(CO_{2Air_n}, CO_{2Top_n}) \end{cases} \tag{102}$$

Chương trình cho giải thuật explicit Euler được viết bằng ngôn ngữ `Python` như sau:

```
def dx(x0, y0, flag):
    '''
    Calculate the rate of change in CO2 concentration in greenhouse and top of greenhouse

    :param x0: CO2-concentration of the greenhouse air at time start t
    :param y0: CO2-concentration of the top greenhouse air at time start t
    :param flag: To be used for printing result to log file and saving to data for plotting
    '''
    CO2_air = f_func(x0, y0, flag)
    CO2_top = g_func(x0, y0, flag)
    return CO2_air, CO2_top


def euler(dx, x0, y0, h, flag):
    '''
    Calculate the approximate value of CO2-concentration of the greenhouse air
    and CO2-concentration of the top greenhouse air

    :param dx: the callable function of the ODE system
    :param x0: CO2-concentration of the greenhouse air at time start t
    :param y0: CO2-concentration of the top greenhouse air at time start t
    :param h: step approximation
    :param flag: To be used for printing result to log file and saving to data for plotting
    '''

    Kx, Ky = dx(x0, y0, flag)

    Kx = h * Kx
    Ky = h * Ky

    x0 = x0 + Kx
    y0 = y0 + Ky

    return x0, y0
```

Trong đoạn code trên, hàm **dx** trả về 2 giá trị $\dot{CO_{2Air}}$ và $\dot{CO_{2Top}}$ như đã định nghĩa ở bài toán 2; **flag** là biến dùng để bật cờ báo hiệu trong phép in kết quả ra file log. **x0, y0** là 2 tham số đầu vào giá trị nồng độ khí CO_2 trong và bên ngoài nhà kính. h là kích thước bước xấp xỉ. Hàm **euler** sẽ trả về giá trị xấp xỉ của CO_{2Air} và CO_{2Top} tại thời điểm $t + h$

3.2.1.b Phương pháp Runge-Kutta bậc 4

Tham khảo từ [EG96] và phần giới thiệu giải thuật Runge-Kutta ở mục 1.6.2, ta có hệ phương trình vi phân thường bậc nhất tổng quát của mô hình toán học đang xây dựng chính là hệ phương trình vi phân ở công thức (100). Áp dụng công thức (29) đã được giới thiệu, ta có thể

dễ dàng tìm được nghiệm xấp xỉ như sau:

$$\begin{cases} K_{1x} = hf(\mathrm{CO}_{2Air_n}, \mathrm{CO}_{2Top_n}) \\ K_{1y} = hg(\mathrm{CO}_{2Air_n}, \mathrm{CO}_{2Top_n}) \\ K_{2x} = hf\left(\mathrm{CO}_{2Air_n} + \frac{K_{1x}}{2}, \mathrm{CO}_{2Top_n} + \frac{K_{1y}}{2}\right) \\ K_{2y} = hg\left(\mathrm{CO}_{2Air_n} + \frac{K_{1x}}{2}, \mathrm{CO}_{2Top_n} + \frac{K_{1y}}{2}\right) \\ K_{3x} = hf\left(\mathrm{CO}_{2Air_n} + \frac{K_{2x}}{2}, \mathrm{CO}_{2Top_n} + \frac{K_{2y}}{2}\right) \\ K_{3y} = hg\left(\mathrm{CO}_{2Air_n} + \frac{K_{2x}}{2}, \mathrm{CO}_{2Top_n} + \frac{K_{2y}}{2}\right) \\ K_{4x} = hf(\mathrm{CO}_{2Air_n} + K_{3x}, \mathrm{CO}_{2Top_n} + K_{3y}) \\ K_{4y} = hg(\mathrm{CO}_{2Air_n} + K_{3x}, \mathrm{CO}_{2Top_n} + K_{3y}) \\ \mathrm{CO}_{2Air_{n+1}} = \mathrm{CO}_{2Air_n} + \frac{1}{6}(K_{1x} + 2K_{2x} + 2K_{3x} + K_{4x}) \\ \mathrm{CO}_{2Top_{n+1}} = \mathrm{CO}_{2Top_n} + \frac{1}{6}(K_{1y} + 2K_{2y} + 2K_{3y} + K_{4y}) \end{cases} \tag{103}$$

Cần chú ý rằng, biến thời gian t trong phương pháp này không được sử dụng vì các biến số thay đổi theo thời gian, ngoại trừ 2 biến số cần xấp xỉ CO_{2Air} và CO_{2Top}, đều được cập nhật từ tập dữ liệu mỗi khi các cảm biến ghi nhận giá trị dữ liệu mới. Ngoài ra, chúng ta mong muốn rằng giá trị xấp xỉ gần bằng với giá trị thực tế, tức là sai số của phương pháp này phải rất bé và tiệm cận về 0. Do LTE của giải thuật Runge-Kutta bậc 4 là $O(h^5)$, do đó khi $h \to 0$, sai số càng tiến gần về 0. Trong bài toán này, bước xấp xỉ h được lựa chọn là 0.1. Mặt khác, từ công thức (29), ta thấy các giá trị của $k1, k2, k3, k4$ có phụ thuộc vào biến số thời gian, tuy nhiên, do h là rất nhỏ và tại thời điểm $t + h$ không ảnh hưởng đến quá trình lấy dữ liệu từ cảm biến, do đó biến thời gian t không được sử dụng ở công thức (103) như trên.

Chương trình cho giải thuật Runge-Kutta bậc 4 được viết bằng ngôn ngữ `Python` như sau:

```
def rk4(dx, x0, y0, h, flag):
    '''
    Calculate the approximate value of CO2-concentration of the greenhouse air
    and CO2-concentration of the top greenhouse air

    :param dx: the callable function of the ODE system
    :param x0: CO2-concentration of the greenhouse air at time start t
    :param y0: CO2-concentration of the top greenhouse air at time start t
    :param h: step approximation
    :param flag: To be used for printing result to log file and saving to data for plotting
    '''

    K1x, K1y = dx(x0, y0, flag)
    K1x = h * K1x
    K1y = h * K1y

    K2x, K2y = dx(x0 + K1x/2, y0 + K1y/2, flag)
    K2x = h * K2x
    K2y = h * K2y

    K3x, K3y = dx(x0 + K2x/2, y0 + K2y/2, flag)
    K3x = h * K3x
    K3y = h * K3y

    K4x, K4y = dx(x0 + K3x, y0 + K3y, flag)
```

```
    K4x = h * K4x
    K4y = h * K4y

    Kx = (1 / 6) * (K1x + 2 * K2x + 2 * K3x + K4x)
    x0 = x0 + Kx

    Ky = (1 / 6) * (K1y + 2 * K2y + 2 * K3y + K4y)
    y0 = y0 + Ky

    return x0, y0
```

3.2.2 Sử dụng các solver đã xây dựng để chạy thử nghiệm

Sau khi xây dựng được hai hàm **euler** và **rk4**, ta bắt đầu tính giá trị xấp xỉ của nồng độ $CO_{2\,Air}$ và $CO_{2\,Top}$ tại các thời điểm $t+h$, $t+2h$, ... với h là kích thước bước xấp xỉ và t là mốc thời gian ban đầu. Trong phần trình bày dưới dưới đây, để phù hợp với timestep có trong tập dữ liệu, nhóm chọn giá trị $h = 5$ (phút) (do cứ mỗi 5 phút, giá trị của cảm biến lại cập nhật 1 lần). Như vậy, chúng ta cần 2 biến để đánh dấu các mốc thời gian, một là để cập nhật dữ liệu sau mỗi 300 giây và biến còn lại dùng để đánh dấu mốc ghi nhận và lưu trữ giá trị xấp xỉ tại các thời điểm sau mốc thời gian h, $2h$, ..., $(n)h$ (phút). Ngoài ra, chúng ta còn có bước xấp xỉ $step$ là tham số đầu vào của 2 hàm **euler** và **rk4** tương ứng với giá trị h trong các hàm đó. Do đó, để có thể tìm được giá trị xấp xỉ tại thời điểm sau h phút, chúng ta cần phải lặp lại các bước $step$ k lần (với $k = 1/step$, $step \leq 1$). Trong bài toán này, giá trị $step = 0.1$ và cần phải thực hiện bước lặp $10 \times 300 = 3000$ lần thì mới lấy được giá trị xấp xỉ tại các bước thời gian h.

Như đã đề cập trong mục 3.1, các biến số thay đổi theo thời gian sẽ được lấy dữ liệu từ 3 file `meteo.csv`, `Greenhouse_climate.csv` và `vip.csv`. Do đó, quá trình lấy dữ liệu được thực hiện thông qua các tác vụ file I/O trong ngôn ngữ `Python`. Cần chú ý rằng, các thông số v_{Wind} và T_{Out} được trích từ file `meteo.csv`, cột thứ 9 và 11; $CO_{2\,Air}, T_{Air}, U_{ThScr}$, `ventLee`, `ventWind` được trích xuất từ file `Greenhouse_climate.csv`, cột thứ 2, 3, 8, 9, 10; $CO_{2\,Air\,SetPoint}$ trích từ file `vip.csv`, cột 1. Giá trị $U_{roof} = \dfrac{\texttt{ventLee} + \texttt{ventWind}}{2} \div 100$, trong khi giá trị $U_{ThScr} = \texttt{EnergyScreen} \div 100$. Đối với U_{extCO_2}, giá trị này sẽ phụ thuộc vào set point của nồng độ $CO_{2\,Air}$, nếu $CO_{2\,Air} < CO_{2\,Air\,SetPoint}$, U_{extCO_2} bằng **1**, ngược lại sẽ có giá trị là **0**. Trong trường hợp dữ liệu của set point là `NaN`, U_{extCO_2} sẽ có giá trị random trong khoảng `[0, 0.3]`. Sau khi có đầy đủ các giá trị biến số thay đổi theo thời gian, ta gọi các solver `euler` và `rk4` để thực hiện công việc tính toán. Giá trị xấp xỉ $CO_{2\,Air}$ và $CO_{2\,Top}$ được cập nhật liên tục qua mỗi vòng lặp, và khi lặp đủ tới kích thước xấp xỉ h, 2 giá trị này được dùng để xuất ra file log và lưu trữ cho quá trình vẽ đồ thị thông qua biến `flag` đã được định nghĩa ở mục 3.2.1.

Chương trình cho quá trình tính toán bằng giải thuật `explicit Euler` được viết bằng ngôn ngữ `Python` như sau:

```
def euler_loop(x0, y0, h, n):
    '''
        Calculate the approximate value of CO2-concentration of the greenhouse air
        and CO2-concentration of the top greenhouse air at t + h, t + 2h, ..., t + n*h

        :param x0: CO2-concentration of the greenhouse air at time start t
        :param y0: CO2-concentration of the top greenhouse air at time start t
        :param h: step approximation
        :param n: the number of times you want to get data (means you can still get data after
                  n*h (minutes))
    '''
    print("------EULER RESULT------")
```

```
    file_meteo = open("meteo.csv")
    file_meteo.readline()
    file_ghc = open("Greenhouse_climate.csv")
    file_ghc.readline()
    file_vip = open("vip.csv")
    file_vip.readline()

    CO2_air = x0
    CO2_top = y0
    step_next = 1
    step = 0.1
    n *= 300
    i = 0
    t0 = 0
    getData = False
    CO2_air_real_data = 427

    while i < n:
        if i // 300 >= step_next:
            step_next += 1

            line_meteo = file_meteo.readline()
            line_ghc = file_ghc.readline()
            line_vip = file_vip.readline()
            elements_vip = line_vip.split(',')

            while "NaN" in line_meteo or "NaN" in line_ghc or elements_vip[1] == "NaN":
                line_meteo = file_meteo.readline()
                line_ghc = file_ghc.readline()
                line_vip = file_vip.readline()
                elements_vip = line_vip.split(',')

            elements_vip = line_vip.split(',')
            elements_meteo = line_meteo.split(',')
            elements_ghc = line_ghc.split(',')

            init.v_wind = float(elements_meteo[10])
            init.T_out = float(elements_meteo[8])
            init.T_air = float(elements_ghc[9])
            init.U_thscr = float(elements_ghc[3]) / 100
            init.U_roof = (float(elements_ghc[10]) + float(elements_ghc[11])) / 200
            init.T_can = init.T_air + 15
            init.T_top = init.T_air + 1

            if elements_vip[0] == "NaN":
                init.U_extCO2 = np.random.uniform(0, 0.3)
            elif init.convert_mgm3_to_ppm(CO2_air) < float(elements_vip[0]):
                init.U_extCO2 = 1
            else:
                init.U_extCO2 = 0

            CO2_air_real_data = float(elements_ghc[2])

            getData = True

        if getData:
            randomU()
            getData = False

        CO2_air_step, CO2_top_step = euler(dx, CO2_air, CO2_top, step, getData)
        CO2_air = CO2_air_step
```

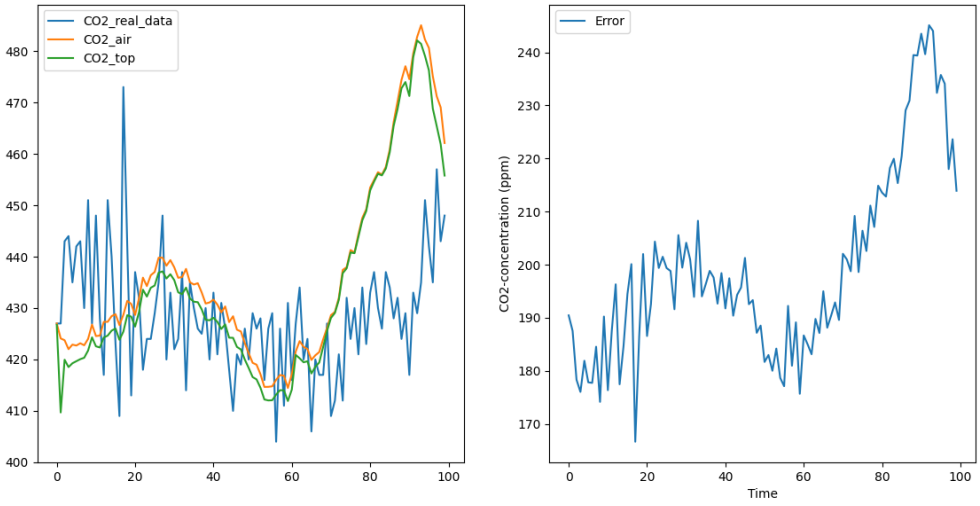
```
        CO2_top = CO2_top_step

        if t0 % (h * 60) == 0:

            print("CO2_real: ", CO2_air_real_data)
            list_real_CO2_air_data.append(CO2_air_real_data)

            err = abs(CO2_air_real_data - CO2_air)
            list_err.append(err)

            list_CO2_air_euler.append(init.convert_mgm3_to_ppm(CO2_air))
            list_CO2_top_euler.append(init.convert_mgm3_to_ppm(CO2_top))
            print("CO2_air at t +", t0 // 60, "=", init.convert_mgm3_to_ppm(CO2_air))
            print("CO2_air at t +", t0 // 60, "=", init.convert_mgm3_to_ppm(CO2_top))
            print("------------------")

        i += 1
        t0 += 1

    file_meteo.close()
    file_ghc.close()
    file_vip.close()
```

Tương tự với giải thuật Runge-Kutta bậc 4, lúc này ta chỉ cần thay hàm `euler` thành `rk4`

```
def rk4_loop(x0, y0, h, n):
    '''
        Calculate the approximate value of CO2-concentration of the greenhouse air
        and CO2-concentration of the top greenhouse air at t + h, t + 2h, ..., t + n*h

        :param x0: CO2-concentration of the greenhouse air at time start t
        :param y0: CO2-concentration of the top greenhouse air at time start t
        :param h: step approximation
        :param n: the number of times you want to get data (means you can still get data after
                  n*h (minutes))
    '''

    print("------RUNGE-KUTTA 4 RESULT------")

    file_meteo = open("meteo.csv")
    file_meteo.readline()
    file_ghc = open("Greenhouse_climate.csv")
    file_ghc.readline()
    file_vip = open("vip.csv")
    file_vip.readline()

    CO2_air = x0
    CO2_top = y0
    step_next = 1
    step = 0.1
    n *= 300
    i = 0
    t0 = 0
    getData = False
    CO2_air_real_data = 427

    while i < n:
        if i // 300 >= step_next:
            step_next += 1

            line_meteo = file_meteo.readline()
```

```
            line_ghc = file_ghc.readline()
            line_vip = file_vip.readline()
            elements_vip = line_vip.split(',')

            while "NaN" in line_meteo or "NaN" in line_ghc or elements_vip[1] == "NaN":
                line_meteo = file_meteo.readline()
                line_ghc = file_ghc.readline()
                line_vip = file_vip.readline()
                elements_vip = line_vip.split(',')

            elements_vip = line_vip.split(',')
            elements_meteo = line_meteo.split(',')
            elements_ghc = line_ghc.split(',')

            init.v_wind = float(elements_meteo[10])
            init.T_out = float(elements_meteo[8])
            init.T_air = float(elements_ghc[9])
            init.U_thscr = float(elements_ghc[3]) / 100
            init.U_roof = (float(elements_ghc[10]) + float(elements_ghc[11])) / 200
            init.T_can = init.T_air + 15
            init.T_top = init.T_air + 1

            if elements_vip[0] == "NaN":
                init.U_extCO2 = np.random.uniform(0, 0.3)
            elif init.convert_mgm3_to_ppm(CO2_air) < float(elements_vip[0]):
                init.U_extCO2 = 1
            else:
                init.U_extCO2 = 0

            CO2_air_real_data = float(elements_ghc[2])

            getData = True

        if getData:
            randomU()
            getData = False

        CO2_air_step, CO2_top_step = rk4(dx, CO2_air, CO2_top, step, getData)
        CO2_air = CO2_air_step
        CO2_top = CO2_top_step

        if t0 % (h * 60) == 0:

            print("CO2_real: ", CO2_air_real_data)
            list_real_CO2_air_data.append(CO2_air_real_data)

            err = abs(CO2_air_real_data - CO2_air)
            list_err_rk4.append(err)

            list_CO2_air_rk4.append(init.convert_mgm3_to_ppm(CO2_air))
            list_CO2_top_rk4.append(init.convert_mgm3_to_ppm(CO2_top))
            print("CO2_air at t +", t0 // 60, "=", init.convert_mgm3_to_ppm(CO2_air))
            print("CO2_air at t +", t0 // 60, "=", init.convert_mgm3_to_ppm(CO2_top))
            print("------------------")

        i += 1
        t0 += 1

    file_meteo.close()
    file_ghc.close()
    file_vip.close()
```

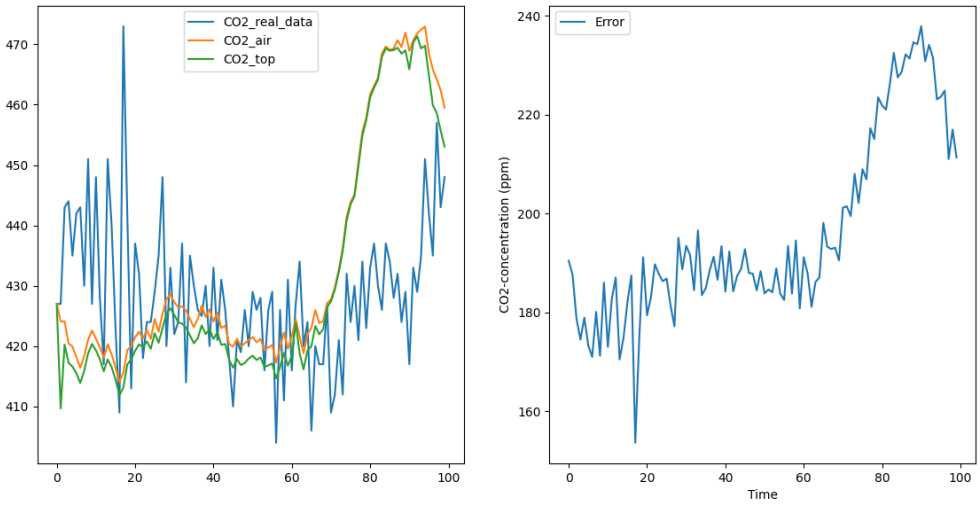
Chạy thử nghiệm với $n = 100$ và giải thuật `Explicit Euler`, lúc này ta sẽ tìm được các giá trị xấp xỉ của CO_{2Air} và CO_{2Top} tại thời điểm $t + 5$, $t + 10$, ..., $t + 500$ và kết quả được ghi vào Log file 1. Để so sánh với dữ liệu thực tế và sai số, ta vẽ đồ thị biểu diễn sự thay đổi của nồng độ CO_2 và dữ liệu thực tế để có hướng nhìn tổng quát hơn về kết quả.



Hình 11: Kết quả chạy bằng phương pháp `Euler` với $n = 100$

Quan sát đồ thị, ta thấy giá trị xấp xỉ của CO_{2Air} trong 25 phút đầu thay đổi không đúng theo hướng của đường dữ liệu thực tế, tuy nhiên khoảng thời gian sau, giá trị thay đổi theo hướng ổn định so với dữ liệu thực tế. Giá trị sai số dao động trong khoảng từ 245(ppm) đến 165(ppm), giá trị sai số trung bình `Err Mean` bằng 193.74835753193125 (ppm). Điều này cho thấy sai số trong 500 phút đầu tiên vẫn còn khá lớn và chưa ổn định so với thực tế (ít dao động, một vài thời điểm tăng/giảm đột biến). Đối với biến số CO_{2Top}, do không có giá trị đối chứng nên không thể đánh giá được trong bài toán này.

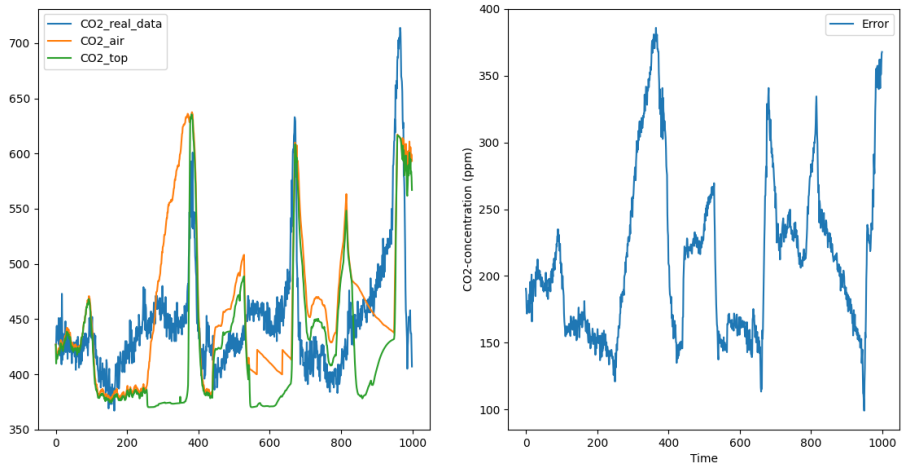
Cũng với $n = 100$, chạy thử nghiệm với giải thuật `Runge-Kutta` bậc 4, kết quả được ghi nhận trong Log file 2. Đồ thị của kết quả được thể hiện dưới đây:



Hình 12: Kết quả chạy bằng phương pháp `Runge-Kutta` bậc 4 với $n = 100$

Quan sát đồ thị, ta thấy kết quả đường biểu diễn của CO_{2Air} và CO_{2Top} tương đồng với nhau. Xét về mặt sai số của kết quả, về tổng quan, giải thuật `Runge-Kutta` bậc 4 cho kết quả chính xác hơn khi các giá trị sai số nằm trong khoảng 155 (ppm) đến 240 (ppm). Tuy nhiên, về sai số trung bình, phương pháp `Runge-Kutta` bậc 4 có sai số trung bình bằng 197.5474763284454 lớn hơn so với sai số trung bình của phương pháp `Euler`. Do đó, trong 500 phút đầu tiên, phương pháp `Euler` cho kết quả tốt hơn.

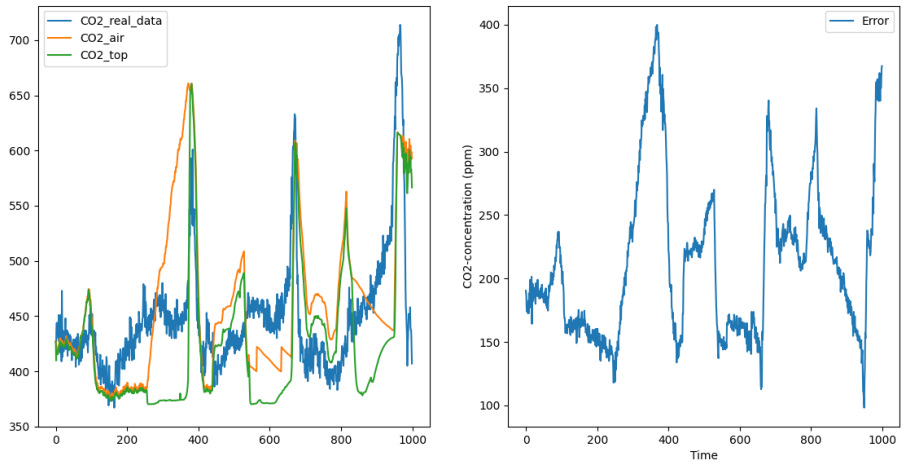
Chạy thử nghiệm với $n = 1000$ và giải thuật `Explicit Euler`, thu được kết quả trong Log file 3. Đồ thị của kết quả được thể hiện như sau:



Hình 13: Kết quả chạy bằng phương pháp `Euler` với $n = 1000$

Từ đồ thị, ta có thể nhận thấy giá trị CO_{2Air} phần lớn thay đổi đúng theo hướng thay đổi của dữ liệu thực tế. Tuy nhiên còn một vài khoảng thời gian chạy sai lệch kết quả ($n = 250 - 380$, $n = 830 - 950$). Giá trị sai số dao động trong khoảng 120 (ppm) đến 385 (ppm) và sai số trung bình `Err Mean` bằng 212.41997669580064 (ppm).

Cũng với $n = 1000$, với giải thuật `Runge-Kutta` bậc 4, ta thu được kết quả các giá trị xấp xỉ của CO_{2Air} và CO_{2Top} trong File log 4. Đồ thị của kết quả được thể hiện như sau:



Hình 14: Kết quả chạy bằng phương pháp `Runge-Kutta` bậc 4 với $n = 1000$

Quan sát đồ thị ta thấy dạng đồ thị của phương pháp `Runge-Kutta` cũng tương tự với `Explicit Euler`. Tuy nhiên lúc này sai số trung bình của phương pháp **Runge-Kutta** bằng 215.47870111346336 lớn hơn so với phương pháp **Euler**. Như vậy qua 2 trường hợp trên, ta có thể thấy phương pháp `Explicit Euler` cho kết quả tốt hơn trong mô hình toán học dang xây dựng.

Bình luận về tính chính xác của mô hình:

- Sai số ở hai phương pháp áp dụng trong mô hình tương đối lớn.
  Nguyên nhân phần lớn do mô hình còn có khá nhiều giả thiết đặt ra, thiếu dữ kiện và chưa sát so với mô hình được xây dựng ngoài thực tế. Tham khảo từ [Van11], ta có thể thấy còn khá nhiều yếu tố quyết định đến nồng độ của CO_{2Air} và CO_{2Top} nhưng lại không được xây dựng trong mô hình này.

  Mặt khác, do kết quả ở mỗi bước đều là các giá trị xấp xỉ, tức là nó đã có 1 lượng sai số so với kết quả thực tế. Khi kết quả xấp xỉ đó được sử dụng lại để tính toán ở các bước tiếp theo, kết quả ở những lần sau chắc chắc sẽ dẫn đến sai lệch và điều này dẫn đến sai số tăng một cách đột biến nếu số lần lặp lớn.

- Chiều hướng thay đổi của đồ thị biểu diễn nồng độ CO_{2Air} phần lớn thay đổi theo như dữ liệu thực tế (chính xác khoảng 85%), điều đó cũng thể hiện cho thấy các giá trị biến thiên theo thời gian được sử dụng trong mô hình có tác động đến kết quả nồng độ CO_2 giống với thực tế

4 Chạy thử nghiệm với mô hình VP

4.1 Bài toán 3

Tương tự với mô hình CO_2, ta áp dụng lại hàm **dx** để xác định các giá trị $V\dot{P}_{Air}$ và $V\dot{P}_{Top}$ với cùng nguồn data như mô hình trên. Các biến số thay đổi bao gồm T_{out}, v_{wind}, RH_{out} được lấy từ file`meteo.csv`. Trong khí đó các biến số T_{Air}, U_{roof} và U_{ThScr} cùng với giá trị tham khảo RH_{Air} được lấy từ file `Greenhouse_climate.csv`

Giá trị U_{roof} được tính bằng công thức $U_{roof} = \frac{\texttt{VentLee}+\texttt{VentWind}}{2}\%$. Giá trị U_{ThScr} chính là `EnScr`, giá trị U_{ShScr} là `BlackScr`. Tất cả các số liệu đều từ file `Greenhouse_climate.csv`.

Với giá trị $r_{s_{min}}$ qua chạy thử nghiệm nhóm phát hiện giá trị$r_{s_{min}} = 82$ không thích hợp với mô hình thực tế do nhóm **Sonoma** do đó, giá trị này đã được tăng lên 200 để có kết quả sát với thực tế hơn.

Ta sẽ đánh giá mô hình này qua hai giá trị xấp xỉ VP_{Air} và VP_{Top} cùng với hướng thay đổi của dữ liệu so với dữ liệu thật được đo bởi đội **Sonoma**.

Các hằng số có liên quan đã được định nghĩa trong mục 2.7.3. Trích trong tập dữ liệu `meteo.csv` và `Greenhouse_climate.csv`, dòng **6**, ta lấy được các giá trị biến số thay đổi theo thời gian như sau:

- v_{Wind} = `Windsp` = 3.2 m s^{-1}
- `Tout` = 17.89999999 °C
- `RHout` = 87.3 %
- `Rcan` = 0.65*`Iglob` = 0 Wm^{-2}
- `VentLee` = 51.1 %
- `VentWind` = 2.9 %
- `Tair` = 19.8999999966472 °C

Tuy nhiên, có một vài biến số cũng có thay đổi theo thời gian nhưng không được ghi nhận trong tập dữ liệu. Trong trường hợp này, các giá trị sau được lựa chọn theo nguyên nhân khách quan (để đạt tính chính xác cao hơn):

- $U_{Blow} = 0.2$
- $U_{Ext_{CO_2}} = 0.2$
- $U_{Thscr} = 0.5$
- $U_{VentForced} = 0.5$
- $U_{Pad} = 0.5$
- $U_{Roof} = |2.9 + 52.9| \div 200 = 0.279$
- $U_{Side} = 0.5$
- $U_{ShScr} = 0.5$
- T_{Top} = `Tair` + 1 = 20.8999999966472 °C
- T_{Can} = `Tair` + 15 = 34.8999999966472 °C

- T_{ThScr} = `Tair` + 1 = 20.8999999966472 °C
- $T_{MechCool}$ = `Tair` - 1 = 18.8999999966472 °C
- $T_{Cov,in}$ = `Tair` - 2 = 17.8999999966472 °C

Vì VP_{Top} không có số liệu `Relative humidity` để thử nghiệm nên trong trường hợp này, ban đầu, độ ẩm tương đối RH_{Top} sẽ bằng với của VP_{Air} và có giá trị là 80.4%.

Sau khi có các giá trị cho từng tham số đầu vào của hàm **dx**, sử dụng công thức đã được hiện thực bằng Python ở phụ lục B, chạy thử nghiệm với các giá trị trên, ta thu được kết quả sau:

```
Relative humidity air diff =  1.4480934896960872e-06
Relative humidity top diff =  1.5006028751304253e-06
```

Như vậy, tại thời điểm $t = 20$ đang xét, tốc độ thay đổiđộ ẩm tương đối của không khí trong nhà kính là **1.4480934896960872e-06** và bên ngoài nhà kính là **1.5006028751304253e-06**. Kết quả này cho thấy áp suất không khí có xu hướng tăng khi ở điều kiện có tham số đầu vào như trên. Đối chứng với số liệu của `Relative humidity` trong tập dữ liệu, ta thấy trong 5 phút tiếp theo, RH_{Air} ở mức 81.1%, tăng 0.2% so với thời gian trước đó. Giả sử rằng tốc độ thay đổi áp suất hơi trong nhà kính là không thay đổi trong khoảng thời gian các cảm biến cập nhật dữ liệu mới (tức là trong vòng 5 phút), độ ẩm tương đối lúc này tăng một lượng bằng $1.4480934896960872e-06 \times 300 \approx 0.04\%$. Ta thấy kết quả chạy thực nghiệm bị sai lệch so với thực tế khá nhiều . Nguyên nhân cho việc dẫn đến sai số là mô hình toán học đang xây dựng có nhiều giả thiết không đúng với thực tế cũng như có nhiều thông số bị bỏ qua. Tuy nhiên, việc tính toán được xu hướng tăng của áp suất hơi trong trường hợp này cũng thể hiện được tính đúng đắn một phần của mô hình. Do VP_{Top} không có giá trị thực tế để đối chứng nên chúng ta không thể đánh giá được tính đúng đắn của tham số này.

Ở trường hợp thứ hai, chúng ta sử dụng dữ liệu ở dòng **100** trong tập dữ liệu.

- v_{Wind} = `Windsp` = 6.3 m s^{-1}
- `Tout` = 17.8 °C
- `RHout` = 78.3 %
- `Rcan` = 0.65*`Iglob` = 61.1 Wm^{-2}
- `VentLee` = 28.4 %
- `ventWind` = 1.6 %
- `Tair` = 21.8999999966472 °C
- $CO_{2\,Air}$ = 457 ppm

Lúc này, giá trị $U_{Roof} = |28.4 + 1.6| \div 200 = 0.15$. Các giá trị biến số có thay đổi theo thời gian còn lại vẫn giữ giá trị như trong trường hợp trên. Giá trị nồng độ RH_{Air} sau 97 lần lấy dữ liệu sẽ có sự khác biệt:

```
Relative humidity air diff =  -4.665834408809069e-05
Relative humidity top diff =  -4.792895476348124e-05
```

So với dữ liệu thực tế từ tập dữ liệu, giá trị RH_{Air} trong 5 phút tiếp theo là 71.1%, giảm một lượng là 0.3%. Giả sử rằng tốc độ thay đổi củaấp suất hơi trong nhà kính là không thay đổi trong 5 phút tiếp theo, áp suất hơi lúc này giảm một lượng $4.665834408809069e-05 \times 300 \approx 1.4\%$. So sánh với dữ liệu thực tế, sai số trong trường hợp này cũng khá to. Kết quả cho thấy áp suất hơi trong nhà kính đang có xu hướng giảm với tập dữ liệu trên có cùng xu hướng thay đổi áp suất hơi so với thực tế.

4.2 Bài toán 4

4.2.1 Các thuật giải cho hệ các phương trình vi phân thường bậc nhất

4.2.1.a Phương pháp Explicit Euler

Tham khảo từ [EG96] và phần giới thiệu giải thuật Explicit Euler ở phần 1.6.1, từ công thức (30), ta có hệ phương trình vi phân thường bậc nhất như sau:

$$\begin{cases} \dot{VP_{Air}} = & (MV_{CanAir} + MV_{PadAir} + MV_{FogAir} + MV_{BlowAir} - MV_{AirThScr} \\ & -MV_{AirTop} - MV_{AirOut} - MV_{AirOut\_Pad} - MV_{Mech}) \cdot (cap_{VP_{Air}})^{-1} \\ \dot{VP_{Top}} = & (MV_{AirTop} - MV_{TopCov,in} - MV_{TopOut}) \cdot (cap_{VP_{Top}})^{-1} \end{cases} \tag{104}$$

Trong hệ phương trình này, giá trị nồng độ VP_{Air} và VP_{Top} là các biến số. Theo lý thuyết, 2 biến số này phải là một hàm số được biểu diễn theo thời gian vì $\dot{VP_{Air}} = \dfrac{\Delta VP_{Air}}{\Delta t}$ mô tả tốc độ thay đổi của áp suất hơi và có đơn vị là kg m^{-2} s^{-1} . Tương tự với VP_{Top}. Đặt hàm số f bằng vế phải của phương trình vi phân thứ nhất, hàm số g bằng vế phải của phương trình vi phân thứ hai. Ta có được hệ phương trình vi phân tổng quát như sau:

$$\begin{cases} \dot{VP_{Air}}(t) = f(t, VP_{Air}(t), VP_{Top}(t)) \\ \dot{VP_{Top}}(t) = g(t, VP_{Air}(t), VP_{Top}(t)) \\ VP_{Air}(t_0) = \alpha, VP_{Top}(t_0) = \beta \end{cases} \tag{105}$$

Từ công thức (49), ta có thể suy ra được công thức explicit Euler trong bài toán này:

$$\begin{cases} VP_{Air}(t+1) = VP_{Air}(t) + hf(t, VP_{Air}(t), VP_{Top}(t)) \\ VP_{Top}(t+1) = VP_{Top}(t) + hg(t, VP_{Air}(t), VP_{Top}(t)) \end{cases} \tag{106}$$

Các biến thay đổi theo thời gian, nếu có dữ liệu trong tập dữ liệu, hoặc có công thức chuyển đổi từ các dữ liệu có sẵn được xem như là hằng số tại thời gian t đang xét. Nếu tại thời điểm t, các thông số chưa được cập nhật mới, giá trị các biến số sẽ là giá trị cũ trước đó, ngược lại, giá trị các biến số sẽ là giá trị mới cập nhật được. Việc trích xuất các giá trị này có thể thực hiện một cách chủ quan. Do đó, trong hệ phương trình này, chúng ta không sử dụng biến số t mà sẽ được mặc định hiểu có sự thay đổi dữ liệu thông qua sự thay đổi của các biến số. Ta kí hiệu ở bước thời gian thứ n là t_n và nghiệm ở bước thứ n là VP_{Air_n}, VP_{Top_n}. Khi đó, nghiệm bài toán được tính bằng:

$$\begin{cases} VP_{Air_{n+1}} = VP_{Air_n} + hf(VP_{Air_n}, VP_{Top_n}) \\ VP_{Top_{n+1}} = VP_{Top_n} + hg(VP_{Air_n}, VP_{Top_n}) \end{cases} \tag{107}$$

Chương trình cho giải thuật explicit Euler được viết bằng ngôn ngữ `Python` như sau:

```
def dx(x0, y0):
    '''
    Calculate the rate of change in CO2 concentration in greenhouse and top of greenhouse

    :param x0: Vapour Pressure of the greenhouse air at time start t
    :param y0: Vapour Pressure of the top greenhouse air at time start t
    '''
    init.VP_air = x0
    init.VP_top = y0
    VP_air = f_func()
    VP_top = g_func()

    return VP_air, VP_top
def euler(x0, y0, h):
    '''
    Calculate the approximate value of Vapour Pressure of the greenhouse air
    and Vapour Pressure of the top greenhouse air

    :param dx: the callable function of the ODE system
    :param x0: Vapour Pressure of the greenhouse air at time start t
    :param y0: Vapour Pressure of the top greenhouse air at time start t
    :param h: step approximation
    '''
    Kx, Ky = dx(x0, y0)
    Kx = h * Kx
    Ky = h * Ky

    x0 = x0 + Kx
    y0 = y0 + Ky

    return x0, y0
```

Trong đoạn code trên, hàm **dx** trả về 2 giá trị $V\dot{P}_{Air}$ và $V\dot{P}_{Top}$ như đã định nghĩa ở bài toán 2. **x0, y0** là 2 tham số đầu vào giá trị nồng độ khí CO_2 trong và bên ngoài nhà kính. h là kích thước bước xấp xỉ. Hàm **euler** sẽ trả về giá trị xấp xỉ của VP_{Air} và VP_{Top} tại thời điểm $t + h$

4.2.1.b Phương pháp Runge-Kutta bậc 4

Tham khảo từ [EG96] và phần giới thiệu giải thuật Runge-Kutta ở mục 1.6.2, ta có hệ phương trình vi phân thường bậc nhất tổng quát của mô hình toán học đang xây dựng chính là hệ phương trình vi phân ở công thức (105). Áp dụng công thức (29) đã được giới thiệu, ta có thể

dễ dàng tìm được nghiệm xấp xỉ như sau:

$$\begin{cases} K_{1x} = hf(VP_{Air_n}, VP_{Top_n}) \\ K_{1y} = hg(VP_{Air_n}, VP_{Top_n}) \\ K_{2x} = hf\left(VP_{Air_n} + \frac{K_{1x}}{2}, VP_{Top_n} + \frac{K_{1y}}{2}\right) \\ K_{2y} = hg\left(VP_{Air_n} + \frac{K_{1x}}{2}, VP_{Top_n} + \frac{K_{1y}}{2}\right) \\ K_{3x} = hf\left(VP_{Air_n} + \frac{K_{2x}}{2}, VP_{Top_n} + \frac{K_{2y}}{2}\right) \\ K_{3y} = hg\left(VP_{Air_n} + \frac{K_{2x}}{2}, VP_{Top_n} + \frac{K_{2y}}{2}\right) \\ K_{4x} = hf(VP_{Air_n} + K_{3x}, VP_{Top_n} + K_{3y}) \\ K_{4y} = hg(VP_{Air_n} + K_{3x}, VP_{Top_n} + K_{3y}) \\ VP_{Air_{n+1}} = VP_{Air_n} + \frac{1}{6}(K_{1x} + 2K_{2x} + 2K_{3x} + K_{4x}) \\ VP_{Top_{n+1}} = VP_{Top_n} + \frac{1}{6}(K_{1y} + 2K_{2y} + 2K_{3y} + K_{4y}) \end{cases} \tag{108}$$

Cần chú ý rằng, biến thời gian t trong phương pháp này không được sử dụng vì các biến số thay đổi theo thời gian, ngoại trừ 2 biến số cần xấp xỉ VP_{Air} và VP_{Top}, đều được cập nhật từ tập dữ liệu mỗi khi các cảm biến ghi nhận giá trị dữ liệu mới. Ngoài ra, chúng ta mong muốn rằng giá trị xấp xỉ gần bằng với giá trị thực tế, tức là sai số của phương pháp này phải rất bé và tiệm cận về 0. Do LTE của giải thuật Runge-Kutta bậc 4 là $O(h^5)$, do đó khi $h \to 0$, sai số càng tiến gần về 0. Trong bài toán này, bước xấp xỉ h được lựa chọn là 0.1. Mặt khác, từ công thức 29, ta thấy các giá trị của $k1, k2, k3, k4$ có phụ thuộc vào biến số thời gian, tuy nhiên, do h là rất nhỏ và tại thời điểm $t + h$ không ảnh hưởng đến quá trình lấy dữ liệu từ cảm biến, do đó biến thời gian t không được sử dụng ở công thức (107) như trên.

Chương trình cho giải thuật Runge-Kutta bậc 4 được viết bằng ngôn ngữ `Python` như sau:

```
def rk4(x0,y0,h):
    '''
    Calculate the approximate value of CO2-concentration of the greenhouse air
    and CO2-concentration of the top greenhouse air

    :param dx: the callable function of the ODE system
    :param x0: Vapour Pressure of the greenhouse air at time start t
    :param y0: Vapour Pressure of the top greenhouse air at time start t
    :param h: step approximation
    '''

    K1x, K1y = dx(x0, y0)
    K1x = h * K1x
    K1y = h * K1y

    K2x, K2y = dx(x0 + K1x/2, y0 + K1y/2)
    K2x = h * K2x
    K2y = h * K2y

    K3x, K3y = dx(x0 + K2x/2, y0 + K2y/2)
    K3x = h * K3x
    K3y = h * K3y

    K4x, K4y = dx(x0 + K3x, y0 + K3y)
    K4x = h * K4x
```

```
    K4y = h * K4y

    Kx = (1 / 6) * (K1x + 2 * K2x + 2 * K3x + K4x)
    x0 = x0 + Kx

    Ky = (1 / 6) * (K1y + 2 * K2y + 2 * K3y + K4y)
    y0 = y0 + Ky

    return x0, y0
```

4.2.2 Sử dụng các solver đã xây dựng để chạy thử nghiệm

Sau khi xây dựng được hai hàm **euler** và **rk4**, ta bắt đầu tính giá trị xấp xỉ của VP_{Air} và VP_{Top} tại các thời điểm $t+h$, $t+2h$, ... với h là kích thước bước xấp xỉ và t là mốc thời gian ban đầu. Trong phần trình bày dưới dưới đây, để phù hợp với timestep có trong tập dữ liệu, nhóm chọn giá trị $h = 5$ (phút) (do cứ mỗi 5 phút, giá trị của cảm biến lại cập nhật 1 lần). Như vậy, chúng ta cần 2 biến để đánh dấu các mốc thời gian, một là để cập nhật dữ liệu sau mỗi 300 giây và biến còn lại dùng để đánh dấu mốc ghi nhận và lưu trữ giá trị xấp xỉ tại các thời điểm sau mốc thời gian h, $2h$, ..., $(n)h$ (phút). Ngoài ra, chúng ta còn có bước xấp xỉ $step$ là tham số đầu vào của 2 hàm **euler** và **rk4** tương ứng với giá trị h trong các hàm đó. Do đó, để có thể tìm được giá trị xấp xỉ tại thời điểm sau h phút, chúng ta cần phải lặp lại các bước $step$ k lần (với $k = 1/step$, $step \leq 1$). Trong bài toán này, giá trị $step = 0.1$ và cần phải thực hiện bước lặp $10 \times 300 = 3000$ lần thì mới lấy được giá trị xấp xỉ tại các bước thời gian h.

Như đã đề cập trong mục 3.1, các biến số thay đổi theo thời gian sẽ được lấy dữ liệu từ 3 file `meteo.csv`, `Greenhouse_climate.csv` và `vip.csv`. Do đó, quá trình lấy dữ liệu được thực hiện thông qua các tác vụ file I/O trong ngôn ngữ `Python`. Cần chú ý rằng, các thông số I_{glob} và RH_{Out}, T_{Out}, v_{Wind} được trích từ file `meteo.csv`, cột thứ 2 và 8, 9, 10, 11; $CO_{2\,Air}, U_{ThScr}, RH_{Air}, T_{Air}$, `ventLee`, `ventWind` được trích xuất từ file `Greenhouse_climate.csv`, cột thứ 3,4,9,10,11,12. Giá trị $U_{roof} = \dfrac{\texttt{ventLee} + \texttt{ventWind}}{2} \div 100$, trong khi giá trị $U_{ThScr} = \texttt{EnergyScreen} \div 100$. Giá trị R_{Can} lấy xấp xỉ bằng 65% I_{glob}.

Sau khi có đầy đủ các giá trị biến số thay đổi theo thời gian, ta gọi các solver `euler` và `rk4` để thực hiện công việc tính toán. Giá trị xấp xỉ VP_{Air} và VP_{Top} được cập nhật liên tục qua mỗi vòng lặp, và khi lặp đủ tới kích thước xấp xỉ h, 2 giá trị này được dùng để xuất ra file log và lưu trữ cho quá trình vẽ đồ thị.

Để thuận tiện cho việc kiểm tra kết quả và so sánh với số liệu thực tế, ta sẽ sử dụng đồ thị đường với số liệu là **Relative Humidity** được tính sẵn trong giải thuật.

Chương trình cho quá trình tính toán bằng giải thuật `explicit Euler` được viết bằng ngôn ngữ `Python` như sau:

```
def param_update(elements,elements2):
    init.v_wind = float(elements[10])
    init.T_out = float(elements[8])
    init.R_can = 0.65 *float(elements[2])
    init.T_air = float(elements2[9])
    init.U_ThScr = float(elements2[3])/100
    init.U_thscr = float(elements2[3])/100
    init.U_roof = abs((float(elements2[10])+float(elements2[11])))/200
    init.CO2_air = float(elements2[2])
    init.T_top = init.T_air - 1
    init.T_thscr = init.T_air + 1
    init.T_mech_cool = init.T_air - 1
    init.T_top_cov_in = init.T_top - 1
```

```
    init.T_can = init.T_air+15
    init.T_cov_in = init.T_top-1
    init.T_mean_air =0.5*(init.T_air+init.T_top)
    init.VP_out = float(elements[7])*init.VP_Out()/100

def euler_solve(x0, y0, h):
    file = open("meteo.csv")
    file.readline()
    n=h//5  #Tính số lần xấp xỉ để đủ 5 phút
    file2 = open("Greenhouse_climate.csv")
    file2.readline()
    for i in range(n):
        line = file.readline()
        while "NaN" in line:
            line = file.readline()
        elements = line.split(',')


        line2 = file2.readline()
        while "NaN" in line2:
            line2 = file2.readline()
        elements2 = line2.split(',')

        #Update các giá trị mỗi 5 phút
        param_update(elements,elements2)

        #Xấp xỉ cho 1 khoảng thời gian 5 phút
        for j in range(int(300//step)):
            x0,y0=euler(x0,y0,step)

        #Giới hạn giá trị RH lại nếu lớn hơn 100%
        if x0>init.VP_Air():
            x0 = init.VP_Air()
        if y0>init.VP_Top():
            y0 = init.VP_Top()

        print("--------------------------------")
        print("VP Air at t +", i * 5, "= ", x0/init.VP_Air()*100)
        print("VP Top at t +", i * 5, "= ", y0/init.VP_Top()*100)


    file.close()
    file2.close()
    return x0, y0
```

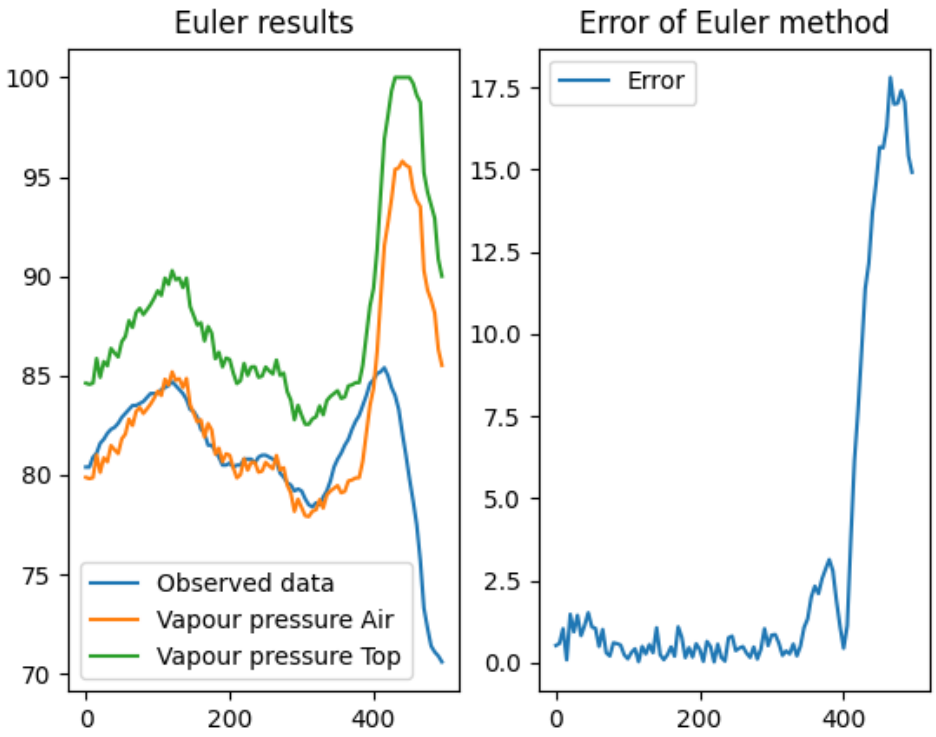
Tương tự với giải thuật `Runge-Kutta` bậc 4, lúc này ta chỉ cần thay hàm `euler` thành `rk4`

```
def rk4_solve(x0, y0, h):
    file = open("meteo.csv")
    file.readline()
    n=h//5 #Tính số lần xấp xỉ để đủ 5 phút
    file2 = open("Greenhouse_climate.csv")
    file2.readline()
    for i in range(n):

        line = file.readline()
        while "NaN" in line:
            line = file.readline()
        elements = line.split(',')

        line2 = file2.readline()
```

```
        while "NaN" in line2:
            line2 = file2.readline()
        elements2 = line2.split(',')

        #Update các giá trị mỗi 5 phút
        param_update(elements,elements2)

        #Xấp xỉ cho mỗi khoảng thời gian 5 phút
        for j in range(int(300//step)):
            x0,y0=rk4(x0,y0,step)

        #Giới hạn giá trị RH lại nếu lớn hơn 100%
        if x0>init.VP_Air():
            x0 = init.VP_Air()
        if y0>init.VP_Top():
            y0 = init.VP_Top()

        print("--------------------------------")
        print("VP Air at t +", i * 5, "= ", x0/init.VP_Air()*100)
        print("VP Top at t +", i * 5, "= ", y0/init.VP_Top()*100)
        rk_air.append(x0/init.VP_Air()*100)
        rk_top.append(y0/init.VP_Top()*100)
    file.close()
    return x0, y0
```

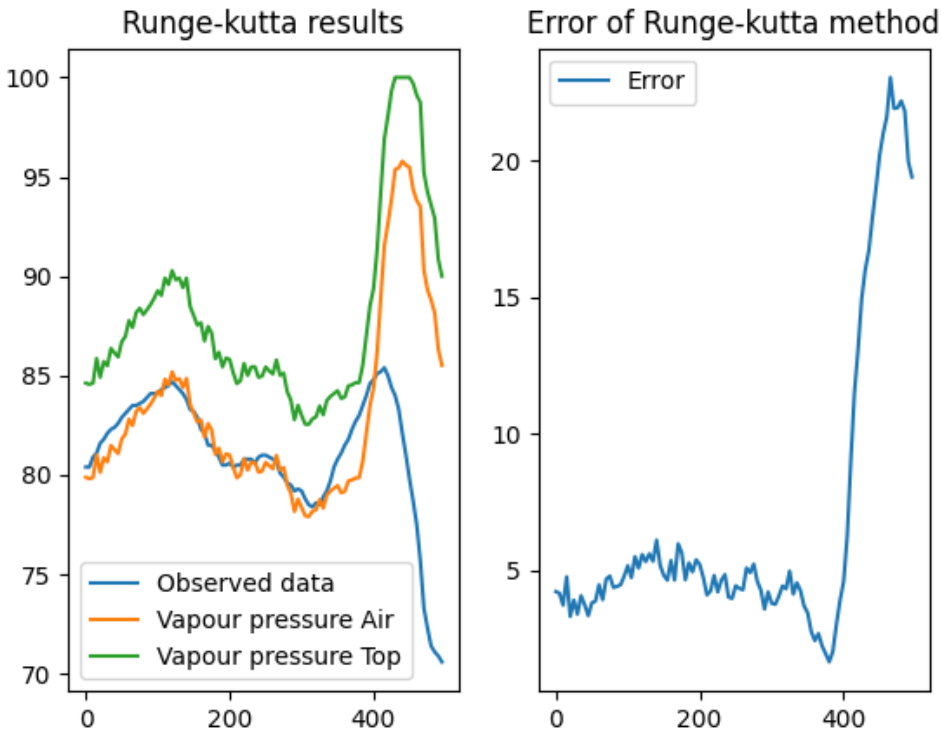
Chạy thử nghiệm với $n = 100$ và giải thuật `Explicit Euler`, lúc này ta sẽ tìm được các giá trị xấp xỉ của VP_{Air} và VP_{Top} tại thời điểm $t + 5$, $t + 10$, ..., $t + 500$ và kết quả được ghi vào Log file 5. Để so sánh với dữ liệu thực tế và sai số, ta vẽ đồ thị biểu diễn sự thay đổi của nồng độ VP_{Air} và dữ liệu thực tế để có hướng nhìn tổng quát hơn về kết quả.

Quan sát đồ thị, ta thấy giá trị xấp xỉ của VP_{Air} trong 400 phút đầu thay đổi đúng theo hướng của đường dữ liệu thực tế,cho ra kết quả hội tụ về đúng đường thực tế với sai số luôn ít hơn 2.5%. Tuy nhiên khoảng thời gian sau, giá trị thay đổi theo hướng tăng vọt và không ổn định so với dữ liệu thực tế. Giá trị sai số tăng lên đến 17.5%, giá trị sai số trung bình `Err Mean` bằng 3.038%. Nhận xét chung với 500 phút đầu tiền, mô hình chạy tương đối ổn định với sai số tạm chấp nhận đượ . Đối với biến số VP_{Top}, do không có giá trị đối chứng nên không thể đánh giá được trong bài toán này.



Hình 15: Kết quả chạy bằng phương pháp `Euler` với $n = 100$

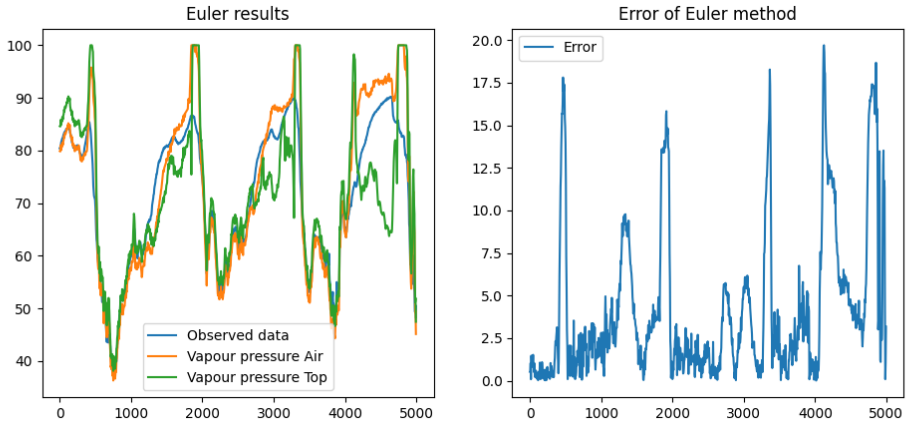
Cũng với $n = 100$, chạy thử nghiệm với giải thuật `Runge-Kutta` bậc 4, kết quả được ghi nhận trong Log file 6. Đồ thị của kết quả được thể hiện dưới đây:



Hình 16: Kết quả chạy bằng phương pháp `Runge-Kutta` bậc 4 với $n = 100$

Với đồ thị này, ta không thể nhận biết rõ ràng khác biệt giữa hai phương pháp **Euler** và **Runge-Kutta**. Dạng của 2 đường đều tương tự nhau khi bám sát đường biểu diễn dữ liệu thực ở 400 phút đầu tiên và tăng vọt lên sau đó. Về mặt số liệu, sai số của giải thuật **Runge-Kutta** về tổng thể lớn hơn so với giải thuật **Euler** khi ở 400 phút đều xấp xỉ 5% và sai số trung bình `Err Mean` bằng 6.87%.

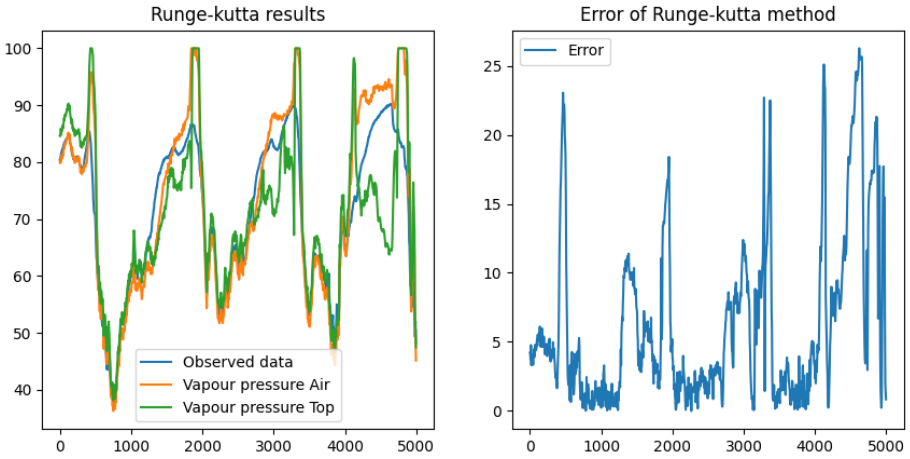
Chạy thử nghiệm với $n = 1000$ và giải thuật `Explicit Euler`, thu được kết quả trong Log file 7. Đồ thị của kết quả được thể hiện như sau:



Hình 17: Kết quả chạy bằng phương pháp `Euler` với $n = 1000$

Từ đồ thị, ta có thể nhận thấy giá trị VP_{Air} phần lớn thay đổi đúng theo hướng thay đổi của dữ liệu thực tế. Tuy nhiên còn một vài khoảng thời gian chạy sai lệch kết quả, điển hình là các đoạn từ phút thứ 3300 đến 3400 và thứ 4000 đến 4200. Giá trị sai số dao động thường xuyên ở mức $2 - 5\%$ và sai số trung bình `Err Mean` bằng 4.21%.

Cũng với $n = 1000$, với giải thuật `Runge-Kutta` bậc 4, ta thu được kết quả các giá trị xấp xỉ của VP_{Air} và VP_{Top} trong File log 8. Đồ thị của kết quả được thể hiện như sau:



Hình 18: Kết quả chạy bằng phương pháp `Runge-Kutta` bậc 4 với $n = 1000$

Quan sát đồ thị ta thấy dạng đồ thị của phương pháp **Runge-Kutta** cũng tương tự với **Explicit Euler**. Tuy nhiên lúc này sai số trung bình của phương pháp **Runge-Kutta** bằng 6.3% lớn hơn so với phương pháp **Euler**. Như vậy qua 2 trường hợp trên, ta có thể thấy phương pháp **Explicit Euler** cho kết quả tốt hơn trong mô hình toán học dang xây dựng.

Bình luận về tính chính xác của mô hình:

- Sai số ở hai phương pháp áp dụng trong mô hình áp suất hơi này tương đối nhỏ so với mô hình $CO2_2$, có thể tạm chấp nhận được.
  Nguyên nhân gây nên sai số phần lớn do mô hình còn có khá nhiều giả thiết đặt ra, thiếu dữ kiện và chưa sát so với mô hình được xây dựng ngoài thực tế. Tham khảo từ [Van11], ta có thể thấy còn khá nhiều yếu tố quyết định đến áp suất hơi nhưng lại không được xây dựng trong mô hình này.

  Mặt khác, do kết quả ở mỗi bước đều là các giá trị xấp xỉ, tức là nó đã có 1 lượng sai số so với kết quả thực tế. Khi kết quả xấp xỉ đó được sử dụng lại để tính toán ở các bước tiếp theo, kết quả ở những lần sau chắc chắc sẽ dẫn đến sai lệch và điều này dẫn đến sai số tăng một cách đột biến nếu số lần lặp lớn.

- Việc mô đường biểu diễn giá trị xấp xỉ có những chu kỳ thay đổi tương tự và giá trị gần với thực tế chứng tỏ mô hình này vẫn có giá trị tham khảo và có thể được dùng để xấp xỉ tương đối. Điều này cũng thể hiện các biến số được đưa vào mô hình có tác động tương đối lớn và là các tác nhân chính duy trì sự cân bằng của hệ này.

A Hiện thực các công thức liên quan đến mô hình nồng độ CO_2 bằng Python

```
def MC_blow_air(eta_heatCO2, U_blow, P_blow, A_flr):
    '''
    Calculate the flow of CO2 from blower to the greenhouse air

    :param eta_heatCO2: Amount of CO2 generated when 1J of sensible heat is generated
    by the direct air heater
    :param U_blow: The control valve of the direct air heater
    :param P_blow: The heat capacity of the direct air heater
    :param A_flr: Greenhouse floor area
    '''
    return (eta_heatCO2*U_blow*P_blow)/A_flr
#---------------------------------------------------------------------


def MC_ext_air(U_extCO2, phi_extCO2, A_flr):
    '''
    Calculate the CO2 supply rate (from the provider)

    :param U_extCO2: The control valve of the external CO2 source
    :param phi_extCO2: The capacity of the external CO2 source
    :param A_flr: Greenhouse floor area
    '''
    return (U_extCO2*phi_extCO2)/A_flr
#---------------------------------------------------------------------


def MC_pad_air(U_pad, phi_pad, A_flr, CO2_out, CO2_air):
    '''
    Calculate the flow of CO2 from the pad and fan system to greenhouse air

    :param U_pad: Pad and fan control
    :param phi_pad: Capacity of the air flux through the pad and fan system
    :param CO2_out: Outdoor CO2 concentration
    :param CO2_air: CO2-concentration of the greenhouse air
    '''
    return ((U_pad*phi_pad)/A_flr)*(CO2_out - CO2_air)
#---------------------------------------------------------------------


def MC_air_top(f_thscr, CO2_air, CO2_top):
    '''
    Calculate the flow of CO2 from the greenhouse air to the top compartment air

    :param f_thscr: Air flux rate through the thermal screen
    :param CO2_air: CO2-concentration of the greenhouse air
    :param CO2_top: CO2-concentration of the top compartment air
    '''
    return f_thscr*(CO2_air - CO2_top)

def f_thscr(U_thscr, K_thscr, T_air, T_top, g, rho_mean_air, rho_air, rho_top):
    '''
    Calculate the air flux rate through the thermal screen

    :param U_thscr: Control of the thermal screen
    :param K_thscr: The thermal screen flux coefficient
    :param T_air: Greenhouse air temperature
```

```
    :param T_top: Greenhouse top compartment air temperature
    :param g: Gravitational acceleration
    :param rho_mean_air: The mean density of the greenhouse and the top compartment air
    :param rho_air: The density of greenhouse air
    :param rho_top: The density of top compartment air
    '''
    Thscr = U_thscr*K_thscr*(abs(T_air - T_top))**(2/3)
    NotThscr = (1 - U_thscr)*( (g*(1-U_thscr))/(2*rho_mean_air)*abs(rho_air - rho_top) )**(1/2)

    return Thscr + NotThscr

def rho_air(rho_air0, g, M_air, h_elevation, T_air, R):
    '''
    Calculate the density of the greenhouse air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational acceleration
    :param M_air: The molar mass of air
    :param h_elevation: The altitude of the greenhouse above sea level
    :param T_air: Greenhouse air temperature
    :param R: The molar gas constant
    '''
    return rho_air0*math.exp( (g*M_air*h_elevation)/((273.15+T_air)*R) )

def rho_top(rho_air0, g, M_air, h_elevation_top, T_top, R):
    '''
    Calculate the density of the top compartment air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational acceleration
    :param M_air: The molar mass of air
    :param h_elevation: The altitude of the greenhouse above sea level
    :param T_top: Top compartment temperature
    :param R: The molar gas constant
    '''
    return rho_air0*math.exp( (g*M_air*h_elevation_top)/((273.15+T_top)*R) )

def rho_mean_air(rho_air0, g, M_air, h_elevation_mean_air, T_air, T_top, R):
    '''
    Calculate the mean density of the greenhouse air and outdoor air

    :param rho_air: The density of the greenhouse air
    :param rho_out: The density of the outdoor air
    '''
    return rho_air0*math.exp( (g*M_air*h_elevation_mean_air)/((273.15 + (T_air + T_top)/2 )*R) )
#-----------------------------------------------------------------


def MC_air_out(f_vent_side, f_vent_forced, CO2_air, CO2_out):
    '''
    Calculate the flow of CO2 from the greenhouse air to the outdoor

    :param f_vent_side: Natural ventilation rate for side window
    :param f_vent_fourced: Forced ventilation rate for side window
    :param CO2_air: CO2-concentration rate of the greenhouse air
    :param CO2_out: CO2-concentration rate of the outdoor
    '''
    return (f_vent_side + f_vent_forced)*(CO2_air - CO2_out)

def f_vent_roof_side(C_d, A_flr, U_roof, U_side, A_roof, A_side, g, h_side_roof,
        T_air, T_out, T_mean_air, C_w, v_wind):
```

```
    '''
    Calculate the ventilation rate through both the roof and the side vents

    :param C_d: Discharge coefficient
    :param A_flr: Greenhouse floor area
    :param U_roof: Control of the aperture of the roof vent
    :param U_side: Control of the side ventilators
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    :param g: Gravitational acceleration
    :param h_side_roof: Vertical distance between mid-points of side wall and
    roof ventilation openings
    :param T_air: Greenhouse air temperature
    :param T_out: Outdoor temperature
    :param T_mean_air: Mean temperature of the greenhouse air and the outside air
    :param C_w: Global wind pressure coefficent
    :param v_wind: The wind speed
    '''
    if (U_side*A_side == 0 and U_roof*A_roof == 0):
        return 0
    else:
        return (C_d/A_flr)* \
            (((U_roof**2)*(U_side**2)*(A_roof**2)*(A_side**2)) / \
             ((U_roof**2)*(A_roof**2)+(U_side**2)*(A_side**2)) \
             * (2*g*h_side_roof*(T_air - T_out)) / (T_mean_air + 273.15) \
             + (((U_roof*A_roof + U_side*A_side)/2)**2) * C_w * v_wind ** 2)**(1/2)

def T_mean_air(T_air, T_out):
    '''
        Calculate the mean temperature

        :param T_air: The temperature of the greenhouse air
        :param T_out: The temperature of the outdoor air
    '''
    return (T_air + T_out) / 2

def C_d(C_d_Gh, eta_shscrCd, U_shscr):
    '''
    Calculate the discharge coefficent

    :param C_d_Gh: the discharge coefficient determined for a greenhouse
    without an external shading screen
    :param eta_shscrCd: parameter that determines the effect of the
    shading screen on the discharge coefficient
    :param U_shscr: Control of the external shading screen
    '''
    return C_d_Gh*(1 - eta_shscrCd*U_shscr)

def C_w(C_w_Gh, eta_shscrCw, U_shscr):
    '''
    Calculate the global wind pressure coefficent

    :param C_w_Gh: the global wind pressure coefficient for a greenhouse
    without an external shading screen
    :param eta_shscrCw: parameter that determines the effect of the
    shading screen on the global wind pressure coefficient
    :param U_shscr: Control of the external shading screen
    '''
    return C_w_Gh*(1 - eta_shscrCw*U_shscr)

def eta_insscr(zeta_insscr):
```

```
    '''
    Calculate the ventilation rate's reduce factor created by insect screens

    :param zeta_insscr: The screen porosity
    '''
    return zeta_insscr*(2 - zeta_insscr)

def f_leakage(c_leakage, v_wind):
    '''
    Calculate the leakage rate

    :param c_leakage: The leakage coefficent
    :param v_wind: The wind speed
    '''
    if v_wind < 0.25:
        return 0.25*c_leakage
    else:
        return v_wind*c_leakage

def f_vent_side(eta_insscr, f2_vent_side, f_leakage, U_thscr, f_vent_roof_side,
        eta_side, eta_roof, eta_roof_thr):
    '''
    Calculate the ventilation rate through the side vents

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param f2_vent_side: The ventilation rate for sidewall ventilation only
    :param f_leakage: The leakage rate
    :param U_thscr: Control of the thermal screen
    :param f_vent_roof_side: The ventilation rate through both the roof and the side vents
    :param eta_side: The ratio between the side vents area and total ventilation area
    :param eta_roof: The ratio between the roof vents area and total ventilation area
    :param eta_roof_thr: The threshold value for which there is no chimney effect
    '''
    if eta_roof < eta_roof_thr:
        return eta_insscr*( U_thscr*f2_vent_side \
            +(1 - U_thscr)*f_vent_roof_side*eta_side ) + 0.5*f_leakage
    else:
        return eta_insscr*f2_vent_side + 0.5*f_leakage

def f2_vent_side(C_d, U_side, A_side, v_wind, A_flr, C_w):
    '''
    Calculate the ventilation rate for sidewalls ventilation only

    :param C_d: Discharge coefficent
    :param U_side: Control of the side ventilators
    :param A_side: Maximum sidewall ventilation area
    :param v_wind: The wind speed
    :param C_w: Global wind pressure coefficent
    :param A_flr: Greenhouse floor area
    '''
    return ((C_d*U_side*A_side*v_wind)/(2*A_flr))*np.sqrt(C_w)

def f_vent_forced(eta_insscr, U_vent_forced, phi_vent_forced, A_flr):
    '''
    Calculate the forced ventilation

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param U_vent_forced: The control of the forced ventilation
    :param phi_vent_forced: The air flow capacity of the forced ventilation system
    :param A_flr: Greenhouse floor area
    '''
```

```
    return (eta_insscr*U_vent_forced*phi_vent_forced)/A_flr
#-------------------------------------------------------------------


def MC_top_out(f_vent_roof, CO2_top, CO2_out):
    '''
    Calculate the CO2 exchange between the top compartment air and the outside air

    :param f_vent_roof: The ventilation rate through roof openings
    :param CO2_top: CO2-concentration of the top compartment air
    :param CO2_out: CO2-concentration of the outside air
    '''
    return f_vent_roof*(CO2_top - CO2_out)

def f_vent_roof(eta_insscr, f2_vent_roof, f_leakage, U_thscr, f_vent_roof_side,
        eta_roof, eta_roof_thr):
    '''
    Calculate the ventilation rate through roof openings

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param f2_vent_roof: The ventilation rate for roof vent ventilation only
    :param f_leakage: The leakage rate
    :param U_thscr: Control of the thermal screen
    :param f_vent_roof_side: The ventilation rate through both the roof and the side vents
    :param eta_roof: The ratio between the roof vents area and total ventilation area
    :param eta_roof_thr: The threshold value for which there is no chimney effect
    '''
    if eta_roof < eta_roof_thr:
        return eta_insscr*( U_thscr*f2_vent_roof \
            +(1 - U_thscr)*f_vent_roof_side*eta_roof ) + 0.5*f_leakage
    else:
        return eta_insscr*f2_vent_roof + 0.5*f_leakage

def f2_vent_roof(C_d, U_roof, A_roof, A_flr, g, h_roof, T_air, T_out, T_mean_air, C_w, v_wind):
    '''
    Calculate the ventilation rate due to roof ventilation

    :param C_d: Discharge coefficent
    :param U_roof: Control of the aperture of the roof vent
    :param A_roof: Maximum roof ventilation area
    :param A_flr: Greenhouse floor area
    :param g: Gravitational accleration
    :param h_roof: The vertical dimension of a single ventilation opening
    :param T_air: Greenhouse air temperature
    :param T_out: Outdoor temperature
    :param T_mean_air: Mean temperature of the greenhouse air and the outside air
    :param C_w: Global wind pressure coefficent
    :param v_wind: The wind speed
    '''
    return ( (C_d*U_roof*A_roof)/(2*A_flr) ) \
        *( (g*h_roof*(T_air - T_out))/(2*(T_mean_air + 273.15)) + C_w*v_wind**2 )**(1/2)

def eta_roof(U_roof, U_side, A_roof, A_side):
    '''
    Calculate the ratio between the roof vents area and total ventilation area
    :param U_roof: Control of the aperture of the roof vent
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    '''
    if (A_roof*U_roof + A_side*U_side == 0):
        return 0
```

```
    return (A_roof*U_roof)/(A_roof*U_roof + U_side*A_side)

def eta_side(U_roof, U_side, A_roof, A_side):
    '''
    Calculate the ratio between the roof vents area and total ventilation area
    :param U_roof: Control of the aperture of the roof vent
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    '''
    if (A_roof*U_roof + A_side*U_side == 0):
        return 0
    return (A_side*U_side)/(A_roof*U_roof + U_side*A_side)
#-----------------------------------------------------------------------


def MC_air_can(M_CH2O, h_Cbuf, P, R_P):
    '''
    Calculate the CO2 flux from the air to the canopy

    :param M_CH2O: The molar mass of CH2O
    :param h_Cbuf: The inhibition of the photosynthesis rate by saturation of the leaves
    with carbonhydrates
    :param P: The photonsynthesis rate
    :param R: The photorespiration during the photosynthesis process
    '''
    return M_CH2O*h_Cbuf*(P - R_P)

def h_Cbuf(C_buf, C_max_buf):
    '''
    Calculate the inhibition of the photosynthesis rate by saturation of the leaves
    with carbonhydrates

    :param C_buf: The buffer capacity
    :param C_max_buf: The maximum buffer capacity
    '''
    if C_buf > C_max_buf:
        return 0
    else:
        return 1

def P(J, CO2_stom, Gamma):
    '''
    Calculate the canopy photosynthesis rate

    :param J: The electron transport rate
    :param CO2_stom: CO2-concentration in the stomata
    :param Gamma: CO2 compensation point
    '''
    return (J*(CO2_stom - Gamma))/(4*(CO2_stom + 2*Gamma))

def R_P(P, Gamma, CO2_stom):
    '''
    Calculate the hotorespiration during photosynthesis processes

    :param P: the canopy photosynthesis rate
    :param Gamma: CO2 compensation point
    :param CO2_stom: CO2-concentration in the stomata
    '''
    return P*(Gamma/CO2_stom)

def J(J_POT, alpha, PAR_can, Theta):
```

```
    '''
    Calculate the electron transport rate
    :param J_POT: The potential electron transport rate
    :param alpha: The conversions factor from photons to electrons
    :param PAR_can: The absorbed PAR
    :param Theta: The degree of curvature of electron transport rate
    '''
    return (J_POT + alpha*PAR_can - \
        np.sqrt( (J_POT + alpha*PAR_can)**2 - 4*Theta*J_POT \
            *alpha*PAR_can ))/ (2*Theta)

def J_POT(J_MAX_25can, Ej, T_can, T_25, R, S, H):
    '''
    Calculate the potential electron transport rate

    :param J_MAX_25can: The maximum rate of electron transport at 25oC for the canopy
    :param Ej: The activation energy for J_POT
    :param T_can: Canopy temperature (K)
    :param T_25: The reference temperature at 25oC (K)
    :param R: The molar gas constant
    :param S: The entropy term
    :param H: The deactivation energy
    '''
    return J_MAX_25can*np.exp(Ej*( (T_can + 273.15) -T_25)/(R*(10**-3)* (T_can + 273.15) *T_25)) \
        * (1 + np.exp((S*T_25 - H)/(R*(10**-3)*T_25))) \
        /(1 + np.exp((S* (T_can + 273.15) - H)/(R*(10**-3)* (T_can + 273.15) )))

def J_MAX_25can(LAI, J_MAX_25leaf):
    '''
    Calculate the maximum rate of electron transport at 25oC for the canopy

    :param LAI: leaf area index
    :param J_MAX_25leaf: The maximum rate of electron transport for the
    leaf at 25oC
    '''
    return LAI*J_MAX_25leaf

def CO2_stom(eta_CO2_air_stom, CO2_air):
    '''
    Calculate the CO2-concentration in the stomata

    :param eta_CO2_air_stom: conversion factor
    :param CO2_air: CO2-concentration of the greenhouse air
    '''
    return eta_CO2_air_stom*CO2_air

def Gamma(J_MAX_25can, J_MAX_25leaf, c_Gamma, T_can):
    '''
    Calculate the CO2 compensation point

    :param J_MAX_25can: The maximum rate of electron transport at 25oC for the canopy
    :param J_MAX_25leaf: The maximum rate of electron transport for the
    :param c_Gamma: the effect of canopy temperature on the CO2 compensation point
    :param T_can: The canopy temperature
    '''
    return (J_MAX_25leaf/J_MAX_25can)*c_Gamma*(T_can + 273.15) \
        + 20*c_Gamma*(1 - (J_MAX_25leaf/J_MAX_25can))
#-----------------------------------------------------------------------
```

B Hiện thực các công thức liên quan đến mô hình áp suất hơi nước bằng Python

```
def rho(h):
    '''
    Compute the saturated Vapour Pressure

    :param h: Altitude of place need to be computed
    '''
    return 101325 *(1 - 2.25577* 10**(-5)* h)** 5.25588
def rho_air():
    '''
    Calculte the density of the greenhouse air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational accleration
    :param M_air: The molar mass of air
    :param h_elevation: The altitude of the greenhouse above sea level
    :param T_air: Greenhouse air temperature
    :param R: The molar gas constant
    '''
    return M_air*rho(h_elevation)/((273.15+T_air)*R)

def rho_out():
    '''
    Calculate the density of the outdoor air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational accleration
    :param M_air: The molar mass of air
    :param h_elevation: The altitude of the greenhouse above sea level
    :param T_out: Outdoor temperature
    :param R: The molar gas constant
    '''
    return M_air*rho(h_elevation)/((273.15+T_out)*R)

def rho_top():
    '''
    Calculate the density of the outdoor air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational accleration
    :param M_air: The molar mass of air
    :param h_elevation_top: The altitude of the top compartment above sea level
    :param T_top: Top compartment temperature
    :param R: The molar gas constant
    '''
    return M_air*rho(h_elevation)/((273.15+T_top)*R)

def rho_mean_air():
    '''
    Calculate the mean density of the top and below compartment of the greenhouse air

    :param rho_air0: The density of the air at sea level
    :param g: Gravitational accleration
    :param M_air: The molar mass of air
    :param h_elevation_mean_air: The mean altitude of both compartment
    :param T_mean_air: Mean temperature
    :param R: The molar gas constant
```

```
    '''
    return M_air*rho(h_elevation)/((273.15+T_mean_air)*R)

''' (7) '''
def f_thscr():
    '''
    Calculate the air flux rate through the thermal screen

    :param U_thscr: Control of the thermal screen
    :param K_thscr: The thermal screen flux coefficent
    :param T_air: Greenhouse air temperature
    :param T_top: Greenhouse top compartment air temperature
    :param g: Gravitational accleration
    :param rho_mean_air: The mean density of the greenhouse and the top compartment air
    :param rho_air: The density of greenhouse air
    :param rho_top: The density of top compartment air
    '''
    return U_thscr*K_thscr*(abs(T_air - T_top))**(2/3) + \
        (1 - U_thscr)/rho_mean_air()*( (g*(1-U_thscr))/(2*rho_mean_air())*abs(rho_air() - rho_top() )** 1/2


def C_d():
    '''
    Calculate the discharge coefficent

    :param C_d_Gh: the discharge coefficient determined for a greenhouse
    without an external shading screen
    :param eta_shscrCd: parameter that determines the effect of the
    shading screen on the discharge coefficient
    :param U_shscr: Control of the external shading screen
    '''
    return C_d_Gh*(1 - eta_shscrCd*U_shscr)

def C_w():
    '''
    Calculate the global wind pressure coefficent

    :param C_w_Gh: the global wind pressure coefficient for a greenhouse
    without an external shading screen
    :param eta_shscrCw: parameter that determines the effect of the
    shading screen on the global wind pressure coefficient
    :param U_shscr: Control of the external shading screen
    '''
    return C_w_Gh*(1 - eta_shscrCw*U_shscr)

''' (10) '''
def f_vent_roof_side():
    '''
    Calculate the ventilation rate through both the roof and the side vents

    :param C_d: Discharge coefficent
    :param A_flr: Greenhouse floor area
    :param U_roof: Control of the aperture of the roof vent
    :param U_side: Control of the side ventilators
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    :param g: Gravitational accleration
    :param h_side_roof: Vertical distance between mid-points of side wall and
    roof ventilation openings
    :param T_air: Greenhouse air temperature
    :param T_out: Outdoor temperature
```

```
    :param T_mean_air: Mean temperature of the greenhouse air and the outside air
    :param C_w: Global wind pressure coefficent
    :param v_wind: The wind speed
    '''
    if (U_roof*A_roof==0):
        return 0
    return (C_d()/A_flr)* \
        ( ((U_roof**2)*(U_side**2)*(A_roof**2)*(A_side**2))/ \
        ((U_roof**2)*(A_roof**2)+(U_side**2)*(A_side**2)) \
        *(2*g*h_side_roof*(T_air - T_out))/(T_mean_air+273.15) \
        + (((U_roof*A_roof + U_side*A_side)/2)**2)*C_w()*v_wind**2  )**(1/2)
''' (11) '''
def eta_insscr():
    '''
    Calculate the ventilation rate's reduce factor created by insect screens

    :param zeta_insscr: The screen porosity
    '''
    return zeta_insscr*(2 - zeta_insscr)
''' (12) '''
def f_leakage():
    '''
    Calculate the leakage rate

    :param c_leakage: The leakage coefficent
    :param v_wind: The wind speed
    '''
    if v_wind < 0.25:
        return 0.25*c_leakage
    else:
        return v_wind*c_leakage
''' (13) '''
def f_vent_side():
    '''
    Calculate the ventilation rate through the side vents

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param f2_vent_side: The ventilation rate for sidewall ventilation only
    :param f_leakage: The leakage rate
    :param U_thscr: Control of the thermal screen
    :param f_vent_roof_side: The ventilation rate through both the roof and the side vents
    :param eta_side: The ratio between the side vents area and total ventilation area
    :param eta_roof: The ratio between the roof vents area and total ventilation area
    :param eta_roof_thr: The threshold value for which there is no chimney effect
    '''
    if eta_roof() < eta_roof_thr:
        return eta_insscr()*( U_thscr*f2_vent_side() \
            +(1 - U_thscr)*f_vent_roof_side()*eta_side()) + 0.5*f_leakage()
    else:
        return eta_insscr()*f2_vent_side() + 0.5*f_leakage()
def f2_vent_side():
    '''
    Calculate the ventilation rate for sidewalls ventilation only

    :param C_d: Discharge coefficent
    :param U_side: Control of the side ventilators
    :param A_side: Maximum sidewall ventilation area
    :param v_wind: The wind speed
    :param C_w: Global wind pressure coefficent
    :param A_flr: Greenhouse floor area
    '''
```

```
    return ((C_d()*U_side*A_side*v_wind)/(2*A_flr))*np.sqrt(C_w())

''' (14) '''
def f_vent_forced():
    '''
    Calculate the forced ventilation

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param U_vent_forced: The control of the forced ventilation
    :param phi_vent_forced: The air flow capacity of the forced ventilation system
    :param A_flr: Greenhouse floor area
    '''
    return (eta_insscr()*U_vent_forced*phi_vent_forced)/A_flr

def eta_roof():
    '''
    Calculate the ratio between the roof vents area and total ventilation area
    :param U_roof: Control of the aperture of the roof vent
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    '''
    if (A_roof + A_side == 0):
        return 0
    return (A_roof*U_roof)/(A_roof + A_side)

def eta_side():
    '''
    Calculate the ratio between the roof vents area and total ventilation area
    :param U_roof: Control of the aperture of the roof vent
    :param A_roof: Maximum roof ventilation area
    :param A_side: Maximum sidewall ventilation area
    '''
    if (A_roof + A_side == 0):
        return 0
    return (A_side*U_side)/(A_roof+ A_side)

''' (16) '''
def f_vent_roof():
    '''
    Calculate the ventilation rate through roof openings

    :param eta_insscr: The ventilation rate's reduce factor created by insect screens
    :param f2_vent_roof: The ventilation rate for roof vent ventilation only
    :param f_leakage: The leakage rate
    :param U_thscr: Control of the thermal screen
    :param f_vent_roof_side: The ventilation rate through both the roof and the side vents
    :param eta_roof: The ratio between the roof vents area and total ventilation area
    :param eta_roof_thr: The threshold value for which there is no chimney effect
    '''
    if eta_roof() < eta_roof_thr:
        return eta_insscr()*( U_thscr*f2_vent_roof() \
            +(1 - U_thscr)*f_vent_roof_side()*eta_roof() ) + 0.5*f_leakage()
    else:
        return eta_insscr()*f2_vent_roof() + 0.5*f_leakage()
''' (17) '''
def f2_vent_roof():
    '''
    Calculate the ventilation rate due to roof ventilation

    :param C_d: Discharge coefficent
    :param U_roof: Control of the aperture of the roof vent
```

```
    :param A_roof: Maximum roof ventilation area
    :param A_flr: Greenhouse floor area
    :param g: Gravitational accleration
    :param h_roof: The vertical dimension of a single ventilation opening
    :param T_air: Greenhouse air temperature
    :param T_out: Outdoor temperature
    :param T_mean_air: Mean temperature of the greenhouse air and the outside air
    :param C_w: Global wind pressure coefficent
    :param v_wind: The wind speed
    '''
    return ( (C_d()*U_roof*A_roof)/(2*A_flr) ) \
        *( (g*h_vent*(T_air - T_out))/(2*(T_mean_air+273.15)) + C_w()*v_wind**2 )**(1/2)



''' (6) '''
def MV_blow_air():
    '''
    Calculate the flow of vapour from blower to the greenhouse air

    :param eta_heatvap: Amount of vapour which is released when 1 Joule of sensible energy is produced by the heat blow
    :param U_blow:                The control value of the direct air heater
    :param P_blow:                The heat capacity of the direct air heater
    :param A_flr:                Greenhouse floor area
    '''
    return (eta_heatVap*U_blow*P_blow)/A_flr

''' (7) '''
def MV_fog_air():
        '''
        Caculate amount of vapour supplied from fogging system
    :param U_fog:         The control value of the direct fogging system
    :param phi_fog: fogging system maximum supply
    :param A_flr:         Greenhouse floor area
        '''
        return (U_fog*phi_fog)/A_flr

''' (8) '''
def MV_pad_air():
        '''
        Caculate the ability of air crossing the pad and fan system
        :param rho_air:        The density of air
        :param U_pad:        The control value of pad and fan system
        :param phi_pad: Pad and fan system maximum supply
        :param A_flr:         Greenhouse floor area
        :param eta_pad: Efficiency of the pad and fan system
        :param x_pad:        Water vapour content in pad and fan system
        :param x_out:        Outdoor water vapour content
        '''
        return (rho_air()*U_pad*phi_pad*(eta_pad*(x_pad - x_out)+x_out))/A_flr

''' (9) '''
def MV_air_out_pad():
        '''
        Caculate amount of vapour flux loss out
        :param U_pad:        The control value of pad and fan system
        :param phi_pad:        Pad and fan system maximum supply
        :param M_water: Molar mass of water
        :param VP_air:        Air vapour pressure
        :param T_air:        Air temperature
        :param A_flr:        Greenhouse floor area
```

```
        :param R:                FIR flux density
        '''
        return (U_pad*phi_pad*M_water*VP_air)/(A_flr*R*(T_air+273.15))

''' (10)'''
def MV_air_mech():
        '''
        Caculate The amount of vapour in the air is also affected by system mechanical cool
        :param U_mech_cool:          Control of the mechanical cooling
        :param COP_mech_cool: Coefficient of performance of the mechanical cooling system
        :param P_mech_cool:   Electrical capacity of the mechanical cooling system
        :param A_flr:                  Greenhouse floor area
        :param T_air:                  Air temperature
        :param T_mech_cool:          The temperature of the cool surface of the mechanical cooling system
        :param VP_air:                   Air vapour pressure
        :param VP_mech_cool:  The mechanical cooling system vapour pressure
        :param xic_ma:                   Stefan Boltzmann constant
        :param H:                      Sensible heat flux density
        '''
        return (6.04*pow(10,-9)*((U_mech_cool*COP_mech_cool*P_mech_cool)/A_flr)*(VP_air-VP_mech()) /(T_air-T_mech_cool+
'''(11)'''
def MV_air_thscr():
    '''
    Caculate the amount of vapour lost due to condensation forming a heat shield
    :param HEC_air_thscr: Heat exchange coefficient
    :param VP_air:                  Air vapour pressure
    :param VP_mech:                  Mechanical cooling vapour pressure
    '''

    return 6.04*pow(10,-9)*HEC_air_thscr()*(VP_air-VP_ThScr())/(1+np.exp(s_mv_12*(VP_air-VP_ThScr())))

'''(12)'''
def HEC_air_thscr():
        '''
        Caculate heat exchange coefficient
        :param U_thscr:        Control of the thermal screen
        :param T_air:        Air temperature
        :param T_thscr:        Thermal screen temperature
        '''
        return 1.7*U_thscr*abs(T_air-T_thscr)

'''(13)'''
def MV_top_cov_in():
    '''
    Caculate mass vapour flux cover internal side
    :param VP_air:                  Air vapour pressure
    :param VP_mech:                  Mechanical cooling vapour pressure
    '''
    return 6.04*pow(10,-9)*HEC_top_cov_in()*(VP_air-VP_Top_cov_in())/(1+np.exp(s_mv_12*(VP_air-VP_Top_cov_in())))

'''(14)'''
def HEC_top_cov_in():
        '''
        Caculate heat exchange coefficient top cover internal side
        :param c_HEC_in: Coefficient of heat exchange between the cover and the air in the environment outside
        :param T_top:         Tempature of compartment above the thermal screen
        :param T_cov_in: Tempature of cover internal side
        :param A_cov:          Greenhouse cover area
        :param A_flr:          Greenhouse floor area
        '''
        return (c_HEC_in*pow(T_top - T_cov_in,0.33)*A_cov)/A_flr
```

```
'''(15)'''
def MV_air_top():
        '''
        Caculate the amount of vapour going from air to top
        :param M_water:        Molar mass of water
        :param R:                FIR flux density
        :param f_thscr: Caculate form f_thscr funtion
        :param VP_air:         Air vapour pressure
        :param T_air:         Air temperature
        :param VP_top:         Compartment above the thermal screen vapour pressure
        :param T_top:         Tempature of compartment above the thermal screen
        '''
        return (M_water*f_thscr()*((VP_air)/(T_air+273.15) - (VP_top)/(T_top+273.15)))/R

'''(16)'''
def MV_air_out():
        '''
        Caculate the amount of vapour going from air to outside
        :param M_water:                 Molar mass of water
        :param R:                         FIR flux density
        :param f_vent_side:   Caculate form f_vent_side funtion
        :param f_vent_forced: Caculate form f_vent_forced funtion
        :param VP_air:                  Air vapour pressure
        :param T_air:                  Air temperature
        :param VP_out:                  Outside vapour pressure
        :param T_out:                  Tempature of outside
        '''
        return (M_water*(f_vent_side()+f_vent_forced())*((VP_air)/(T_air+273.15) - (VP_out)/(T_out+273.15)))/R

'''(17)'''
def MV_top_out():
        '''
        Caculate the amount of vapour going from top to outside
        :param M_water:                 Molar mass of water
        :param R:                         FIR flux density
        :param f_vent_roof:   Caculate form f_vent_roof funtion
        :param VP_top:                  Top vapour pressure
        :param T_top:                  Top temperature
        :param VP_out:                  Outside vapour pressure
        :param T_out:                  Tempature of outside
        '''
        return (M_water*f_vent_roof()*((VP_top)/(T_top+273.15) - (VP_out)/(T_out+273.15)))/R


'''(18)'''
def VEC_can_air():
        '''
        Caculate vapour exchange coefficient
        :param rho_air: The density of greenhouse air
        :param c_p_air: Heat capacity of air
        :param LAI:               The leaf area index
        :param delta_h: Latent heat of evaporation
        :param gamma:               Psychrometric constant
        :param r_b:               Boundary layer resistance of the canopy for vapour transport
        :param r_s:               The canopy resistance for transpiration
        '''
        return (2*rho_air()*c_p_air*LAI)/(delta_H*gamma*(r_b+r_s()))
def MV_can_air():
        '''
        Caculate the evaporation of the foliage
        :param VEC_can_air: Caculate from VEC_can_air funtion
```

```
        :param VP_can:                Canopy vapour pressure
        :param VP_air:               Air vapour pressure
        '''
        return VEC_can_air()*(VP_can()-VP_air)

def r_s():
    '''
        Caculate the transpiration resistance
        :param r_s_min:    Minimum canopy resistance
        '''
    # return 82
    return r_s_min*rf_R_can()*rf_CO2()*rf_VP()
def rf_R_can():
    '''
        Caculate the canopy resistance factor
        :param R_can:       Radiation above the canopy
        :param c_evap1:         Observed param 1
        :param c_evap2:                Observed param 1
        '''
    return ((R_can+c_evap1)/(R_can+c_evap2))
def rf_CO2():
    '''
        Caculate the resistance factor of Co2 in the lower compartment
        :param c_evap3:         Observed param 3
        :param CO2_air:                CO2 amount in the lower compartment
        '''
    temp = 1+c_evap3()*(CO2_air-200)**2
    return temp
    if temp <= 1.5:
        return temp
    return 1.5
    # return temp
def rf_VP():
    '''
        Caculate the resistance factor of vapour pressire in the lower compartment
        :param c_evap3:         Observed param 4
        :param VP_air:                Vapour pressure in the lower compartment
    :param VP_can:                Vapour pressure at the canopy
        '''
    temp = 1+c_evap4()*(VP_can()-VP_air)**2
    return temp
    if temp <=5.8:
        return temp
    return 5.8
    # return temp
def S_rs():
    '''
        Caculate the switch function
        :param s_r_s:             Slope of the function
        :param R_can:                Radiation above canopy
    :param R_can_SP:        Radiation abobe canopy at night
        '''
    return 1/(1+np.exp(srs*(R_can-R_can_SP)))

def c_evap3():
    '''
        Caculate the resistance factor of vapour pressire in the lower compartment
        :param c_night_evap3:         c_evap at night
        :param c_day_evap3:                c_evap at day
        '''
    return c_night_evap3*(1-S_rs())+c_day_evap3*S_rs()
```

```
def c_evap4():
    '''
        Caculate the resistance factor of vapour pressire in the lower compartment
        :param c_night_evap4:         c_evap at night
        :param c_day_evap4:               c_evap at day
        '''
    return c_night_evap4*(1-S_rs())+c_day_evap4*S_rs()

def cap_VP_air():
    return (M_water*h_air)/(R*(T_air+273.15))
def cap_VP_top():
    return (M_water*h_top)/(R*(T_top+273.15))
def VP_sat(t):
    '''
        Caculate saturated vapour pressure
        :param t: Temperature
        '''
    return 610.78 * np.exp( t / ( t + 238.3 ) * 17.2694 )

def VP_ThScr():
    '''
        Caculate saturated vapour pressire on the thermal screen
        :param T_thscr: Temperature of thermal screen
        '''
    return VP_sat(T_thscr)

def VP_mech():
    '''
        Caculate saturated vapour pressire on the mechanical cooling surface
        :param T_mech_cool: Temperature of mechanical cooling surface
        '''
    return VP_sat(T_mech_cool)

def VP_Top_cov_in():
    '''
        Caculate saturated vapour pressire on the top covering layer
        :param T_cov_in: Temperature of thermal screen
        '''
    return VP_sat(T_cov_in)

def VP_can():
    '''
        Caculate saturated vapour pressire in the canopy
        :param T_can: Temperature of canopy
        '''
    return VP_sat(T_can)
def VP_Out():
    '''
        Caculate saturated vapour pressire outside
        :param T_out: Temperature of canopy
        '''
    return VP_sat(T_out)

def VP_Top():
    '''
        Caculate saturated vapour pressire on the top compartment
        :param T_top: Temperature of top compartment
        '''
    return VP_sat(T_top)
def VP_Air():
    '''
```

```
        Caculate saturated vapour pressire in the lower compartment
        :param T_air: Temperature of lower compartment
        '''
    return VP_sat(T_air)
```

Tài liệu tham khảo

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

[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.

[De 96] H.F. De Zwart. "Analyzing energy-saving options in greenhouse cultivation using a simulation model". English. WU thesis 2071 Proefschrift Wageningen. PhD thesis. 1996. ISBN: 9789054855330.

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

[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*. Ed. by David M. Gates and Rudolf B. Schmerl. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 33–43. ISBN: 978-3-642-87810-7. DOI: 10.1007/978-3-642-87810-7_2. URL: https://doi.org/10.1007/978-3-642-87810-7_2.

[Sta87] C. Stanghellini. *Transpiration of greenhouse crops : an aid to climate management*. 1987.

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