E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Modelling and linearization of a drying plant

Master's Degree in
Industrial Engineering

Master's Thesis

Fermín Velázquez López

Director: Jorge Elso Torralba

Co-director: Irene Miquelez Madariaga

Pamplona, February 21st 2024

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Abstract

The present Master's Thesis has been done within the Dynamic and Control Systems group of the Engineering Department of the Public University of Navarre (UPNA). The main objective is to obtain a simulator of an industrial rotary dryer, to allow control strategies to be developed. To achieve this, the following steps were taken:

- Modelling of a rotary dryer based on selected literature.
- Implementation of a simulator in MATLAB/Simulink.
- Linearization of the model.
- Validation of the linear model.

Key Words

Rotary Dryers

Non-linear model

Linearization

Linear model

Validation

MATLAB

Simulink

Resumen

Este Trabajo de Fin de Máster se ha realizado en el grupo de Sistemas Dinámicos y de Control Automático del Departamento de Ingeniería de la Universidad Pública de Navarra. El objetivo principal es obtener un simulador de una secadora rotativa industrial, para permitir el desarrollo de estrategias de control. Para conseguirlo, se han llevado a cabo los siguientes pasos:

- Modelado de una secadora rotativa a partir de bibliografía seleccionada.
- Implementación de un simulador en MATLAB/Simulink.
- Linealización del modelo.
- Validación del modelo lineal.

Palabras Clave

Secadores rotativos

Modelo no lineal

Linealización

Modelo lineal

Validación

MATLAB

Simulink

Index

List of Figures

List of Tables

Notation

Symbol	Description	Units
A	Cross sectional area of the dryer	m^2
a	Constant in $\boldsymbol{\tau}$ equation	
a_w	Water activity	
C	Parameter in W_e equation	
$c_{p,p}$	Specific heat capacity of alfalfa	$kJ/kg \cdot °C$
$c_{p,v}$	Specific heat capacity of water vapor	$kJ/kg \cdot °C$
$c_{p,a}$	Specific heat capacity of air	$kJ/kg \cdot °C$
F_H	Drum loading factor	
G_a	Air mass flow rate	kg/s
G_p	Alfalfa mass flow rate	kg/s
h	heat loss coefficient	
K	Drying constant	min^{-1}
L	Dryer length	m
L_w	Vaporisation latent heat of pure water	kJ/kg
M_a	Mass hold-up of air	kg
M_p	Mass hold-up of alfalfa	kg
N	Dryer rotational speed	rpm
q_{lat}	Vaporisation latent heat of water in the product	kJ/kg
$\boldsymbol{Q}_p$	Heat loss through dryer shell	kJ/s
$Q_{st,n}$	Isosteric heat of desorption	kJ/kg
t	Time	s
T	Temperature	°C
$\boldsymbol{\tau}$	Residence time	min
U_va	Volumetric heat transfer coefficient	$kJ/m^3 \cdot s \cdot °C$

V	Volume	m^3
V_a	Air axial velocity	m/s
W	Alfalfa moisture content	kg H_2O/kg alfalfa
W_e	Alfalfa equilibrium moisture content	kg H_2O/kg alfalfa
X_m	Parameter in W_e equation	
Y	Absolute air moisture content	kg H_2O/kg air
$\boldsymbol{\alpha}$,S	Dryer slope	$^\circ$
R_w	Drying rate	kg/min
D_p	Particle size	μm

Subscripts

a	Air
e	Equilibrium
in	Inlet
out	Outlet
p	product (alfalfa)

1. Introduction

Alfalfa is the most cultivated crop from the legume family, it is cultivated worldwide and used as forage [1]. An important step in the harvesting process is the drying. High temperature rotary dryers (typically around 400 – 1000ºC) are used to reduce the moisture content from fresh alfalfa (usually containing about 70-80% of water) down to 10-12% water basis (w.b.) [2], that corresponds to **1.1**$\hat{1}$-13.64% dry basis (d.b). This is done to avoid spoilage due to microorganisms, germination, chemical and enzymatic processes, etc [3]. It also reduces the weight of the product, reducing the costs of transportation. The lower limit of 10% is set to avoid overheating, that can damage the crop. Besides, it is desirable to minimize the extent of the drying to the strictly necessary, as it is directly linked to the cost of heating.

1.1. Motivation

Despite the extended use of rotary dryers for many years, there exists little literature about its dynamics and control. The main reason for this is the complexity of the process, which combines thermal drying and product transport. Rotary dryers are highly non-linear systems, dependent both on time and space, and therefore their modelling is certainly difficult. Still, there exists literature about the drying process, mainly focused on the drying mechanisms rather than the control of it [4].

The control of the drying process is necessary due to several factors [5]:

1. maintenance of product quality (avoid spoilage due to high moisture content),
2. increase the product throughput while minimizing the energy consumption,
3. eliminate the effect of disturbances such as inlet product moisture.

The first two factors are relevant because a better control of the drying process means that less product is under-dried (leading to spoilage) and over-dried (which is costly mainly in terms of energy input). Additionally, optimal drying conditions should be ensured, even with disturbances affecting the output, such as variations in the inlet product moisture.

The impact of drying is very high in terms of energy consumption: it adds up to about 15% of the energy used by industries, thus making it desirable to improve the efficiency of the process [4]. The main costs of the dryer are related to the operation (burners that heat the air), so there is a benefit in reducing the energy consumption.

Although there is a clear incentive to apply automatic control, the most extended practice is still manual control, which leaves room for improvements in the process.

1.2. Objectives

In view of the lack of literature about the control of dryers, the objective of this work is to develop a simulator of a rotary dryer that allows the analysis of the dynamics of the system and most importantly, the design of controllers. For that purpose, a model of a rotary dryer is developed first, based on the available literature. Then, the model is implemented in the MATLAB/Simulink software. Finally, the model is linearized at several operating points, obtaining a linear model that facilitates the design of controllers using classic control theory. The validity of such model is then verified by comparing the open-loop behaviour against the previously developed non-linear model.

2. Theoretical foundations

In this chapter, the main mechanisms involved in the drying process of alfalfa (and rotary drying in general) are presented.

2.1. Rotary dryers

A rotary dryer is comprised of a cylindrical shell that rotates around its axis. The wet product is fed along with hot dry air[1], and the heat transfer is driven mainly by convection. The shell usually rests at a slight slope that facilitates the flow of the solid (Figure 1). Typically, the drums are equipped with 'lifting flights' (Figure 2), which lift the product during the rotation and then release it, allowing a higher contact area with the hot flowing air. Most of the drying results from this cascading period [3].

The models usually result in a set of simultaneous transport phenomena that are the reason of the complexity and highly nonlinear equations that govern the system. These equations model primarily momentum, heat, and mass transfer processes.

To model the flow of the solid product, the different ways of particle flow have to be considered: cascading, kiln (flow at the bottom of the shell caused by gravity) and bouncing or rolling. The residence time of the drying solid is an important parameter that affects the extent of the drying. The mean residence time (τ) is defined as the relation between total product mass hold-up (M_p) and product mass flow rate (G_p), as given by

$$\tau = \frac{M_p}{G_p}. \tag{1}$$

Mean reasidence time can be obtained empirically by measuring the mass hold-up at a given operation point. Analytical approximations of the mean residence time are provided by complex expressions that depend on dryer length (L), dryer diameter (D), rotational speed (N), slope angle (S and α), mass flow rate (G), air speed (V_a), and other coefficients (see Table 1).

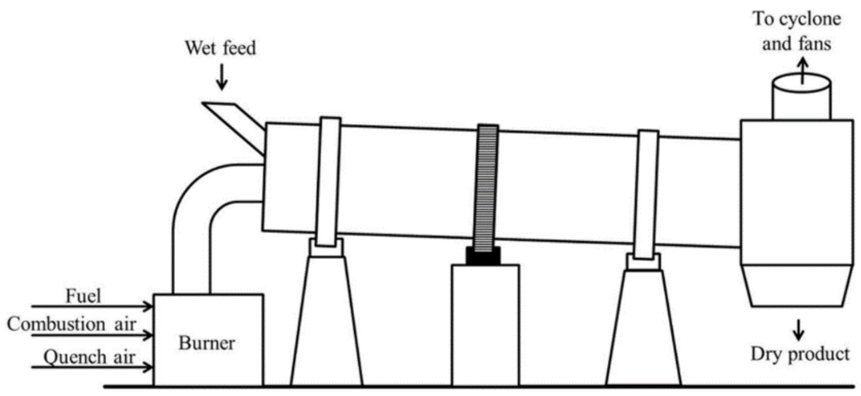


Figure 1. Diagram of a typical rotary dryer.[3]

[1] There are also counter current flow dryers, which will not be covered in this work.

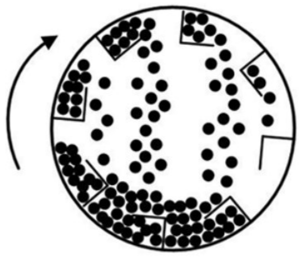


Figure 2. Section of a rotary dryer with lifting flights.[3]

Table 1. Residence time equations from literature

Source	Equation	
Obtained from [6], originally from Friedman and Marshall [7]	$\tau = \frac{0.23}{S \cdot N^{0.9}} \cdot \frac{L}{D} - 9.84 \cdot D_p^{-0.5} \cdot L \cdot \frac{G_a}{G_p}$	(2)
Obtained from [2], originally from [8] Saeman and Mitchell	$\tau = \frac{L}{F_H \cdot D \cdot N(\alpha \pm aV_a)}$	(3)

2.2. Moisture equilibrium – drying rate

The drying behaviour of solid particles can be separated in two main phases: the constant rate period and the falling rate period (Figure 3). The constant rate drying period corresponds to the evaporation of the surface water, while the falling rate period is determined by the transport of moisture from the interior to the surface of the particle. This is governed by several complex processes (diffusion, convection, capillarity, and adsorption [10]). It is not an easy task to predict this behaviour because of the biological variability, and empirical measurements must be made for each of the products to be dried. Drying of crops, such as alfalfa, takes place primarily during the falling rate period [3].

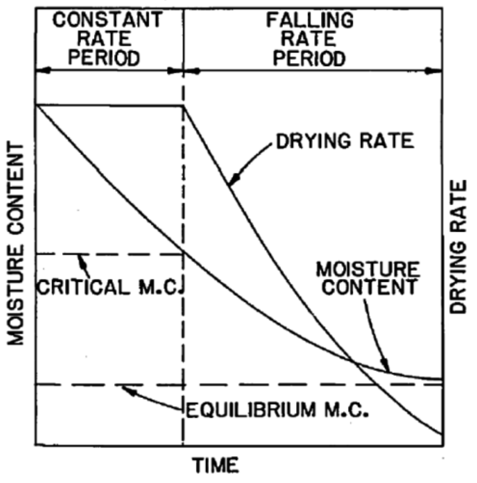


Figure 3. Constant and falling rate drying periods, from [10].

This drying stage can be expressed mathematically as

$$R_w = \frac{dW}{dt} = -\,K\ \ W - W_e\ . \tag{4}$$

The drying rate constant (K) is determined experimentally for each material, typically as a function of air temperature and moisture content. The drying rate (R_w) is proportional to the drying rate constant, and to the difference of current moisture content (W) and equilibrium moisture content (W_e). The latter is a function of air temperature and water activity[2].

Table 2. Drying rate constant equations from literature.

Product	Source	Equation	
Alfalfa chop	Sokhansanj and Patil [9]	$K = 0.0176 \cdot exp\,(-0.123 \cdot W + 0.044 \cdot T - 5.79 \cdot 10^{-5} \cdot T^2 - 3.24 \cdot 10^{-3} \cdot W \cdot T)$	(5)
Alfalfa	Obtained from [5], originally from Sokhansanj and Patil [9]	$K = 5.71 \cdot 10^4 \cdot exp\left(-\frac{73.99}{T}\right)$	(6)
Alfalfa stem	Bilanski et al. [11]	$K = 0.051 - 3.43 \cdot 10^{24} \cdot e^{-T}$	(7)
Alfalfa leaf	Bilanski et al. [12]	$K = 0.064 - 2.17 \cdot 10^{24} \cdot e^{-T}$	(8)
Alfalfa chop	Patil et al. [13]	$K = 0.133 \cdot L^{-0.48}$ *(where L = stem length (mm), obtained at T = 60°C)*	(9)

Equation (5) depends on both air temperature (T) and product moisture (W). The range of values for typical operating conditions of rotary dryers lies within 4.36×10^{-4} min^{-1} and around 0.7 min^{-1} (see Figure 4). The equations provided by [11] and [12] result in a constant values above temperatures of around 80°C (0.051 min^{-1} for alfalfa stems and 0.064 min^{-1} for leaves). Equation (9) was obtained with varying stem lengths from 10 mm to 75 mm, which result in values of drying rate constant ranging from around 0.01 min^{-1} to 0.05 min^{-1}. However, equation (6) provides values that are orders of magnitude away from the values presented by the rest of the literature (probably due to a mistake of interpretation), and for that reason are not considered from now on.

Apart from that last outlier, it is observable still that values differ from source to source, due to the unpredictable nature of the mechanisms involved in the drying process. A particular range of values (around 0.05 min^{-1}) is present in all of them, so it is considered as a representative value.

[2] Relative humidity of air is usually referred to as water activity.

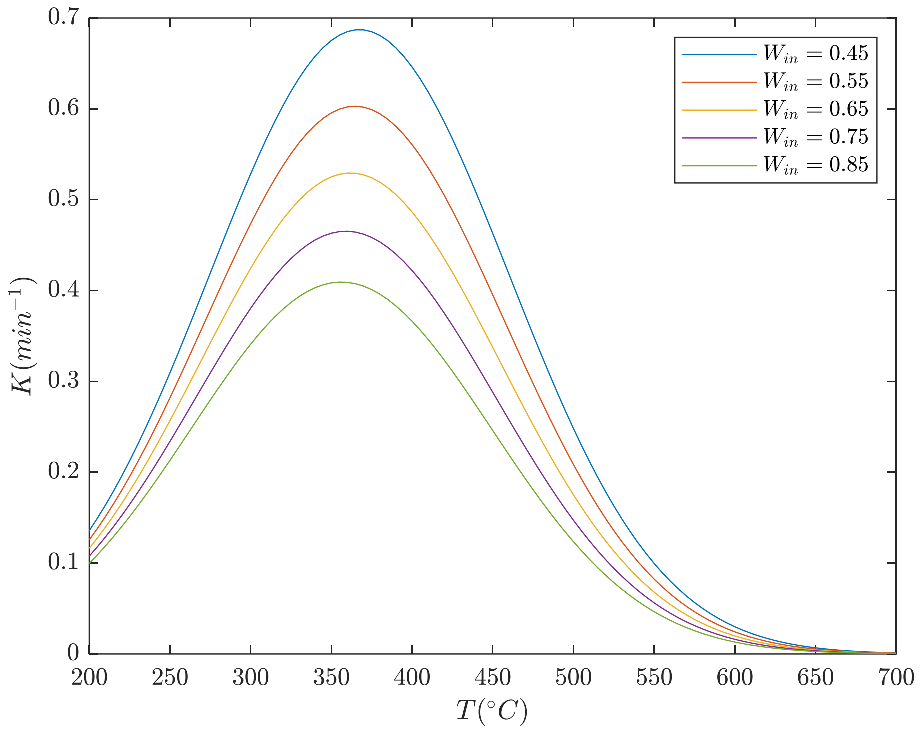


Figure 4. Drying rate constant (K) in min^{-1} from equation (5).

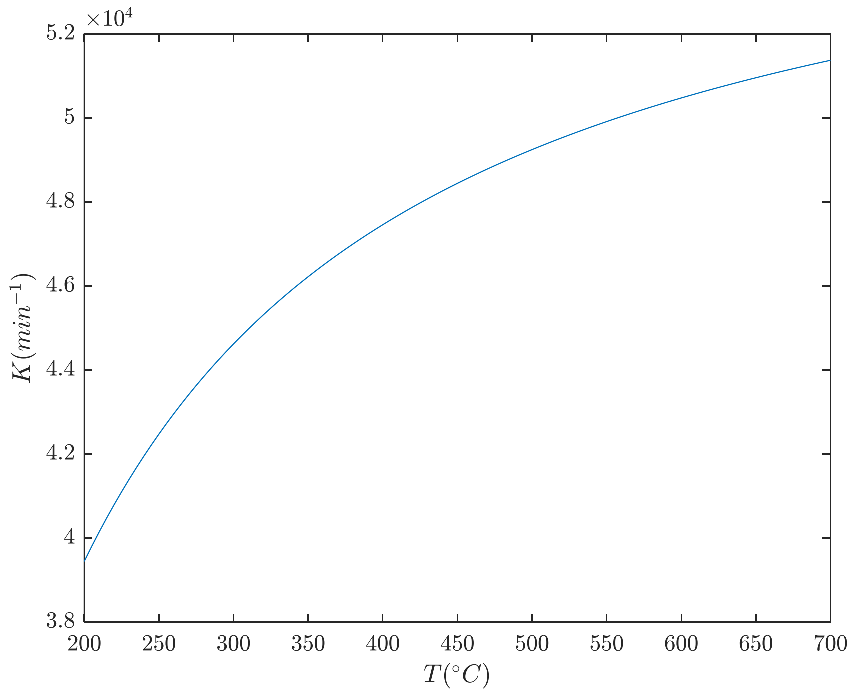


Figure 5. Drying rate constant (K) in min^{-} from equation (6).

The equilibrium moisture content (W_e) in equation (4), is calculated in [5] from the equilibrium isotherms modelled by the B.E.T.[3] equation

$$W_e = \frac{X_m C a_w}{(1-a_w)[1+(C-1)a_w]}. \tag{10}$$

The coefficients are expressed as

$$X_m = 3.9229 \cdot 10^{-4} \cdot exp\left(\frac{1858.8}{T+273.16}\right), \tag{11}$$

$$C = 323.1769 \cdot exp\left(-\frac{974.55}{T+273.16}\right), \tag{12}$$

and water activity is expressed as

$$a_w = \frac{p}{p_0}, \tag{13}$$

which is the relative humidity of the air, calculated as partial pressure of water vapor (p) divided by saturation pressure (p_0) at a given temperature.

This results in a set of isotherm curves that show the relationship between the alfalfa equilibrium moisture content and the water activity, as seen in Figure 6.

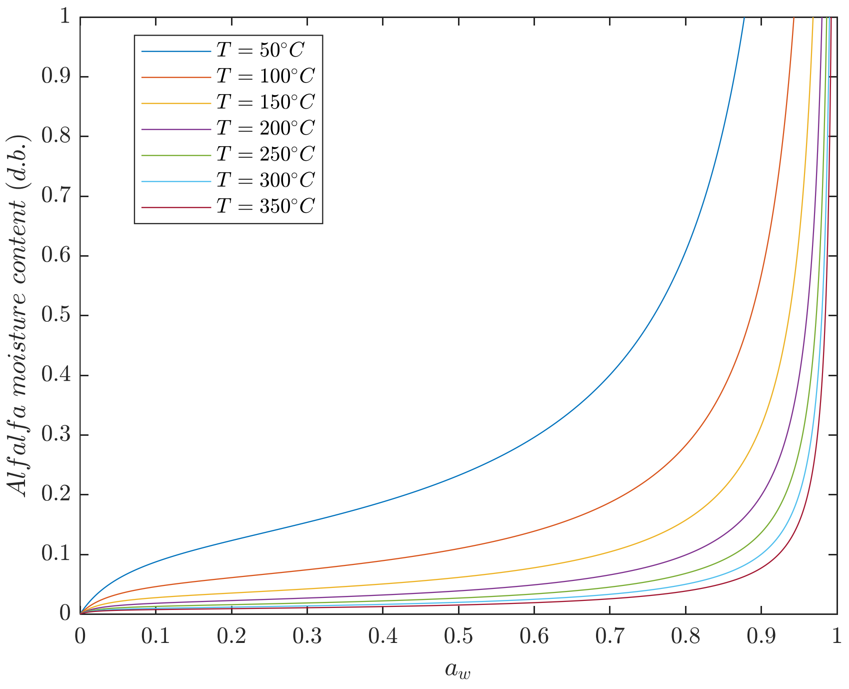


Figure 6. Equilibrium moisture isotherms calculated with the B.E.T. equation.

[3] Brauaner-Emmett-Teller theory models the adsorption mechanism.

In the regions of high water activity (close to 1), the air is close to saturation and cannot *accept* more water from the alfalfa, so equilibrium moisture content is high (even greater than 1, meaning each kg of alfalfa would hold more than 1 kg of water in equilibrium, as air is oversaturated). The greater the temperature, the more water it can hold before saturating, which can be seen as the hotter isotherms getting having lower moisture equilibrium content values.

The theoretical foundations of the modelling of rotary dryers have been established, based on different literature. In the next chapter, the model is developed by customizing the set of phenomena to be represented and by selecting the corresponding equations.

3. Modelling of the plant

In this section, the model of the rotary dryer is defined. Based on the literature review, expressions modelling the different drying mechanisms are selected, and the assumptions and simplifications are clarified. The values selected for the different parameters are also presented and justified. Then, the implementation in the MATLAB/Simulink software is performed.

3.1. Non-linear model

The rotary dryer is modelled as a 1-D model, of length L and divided into n different control volumes of length dL (Figure 7). At each control volume, all the properties are assumed equal and time dependent.

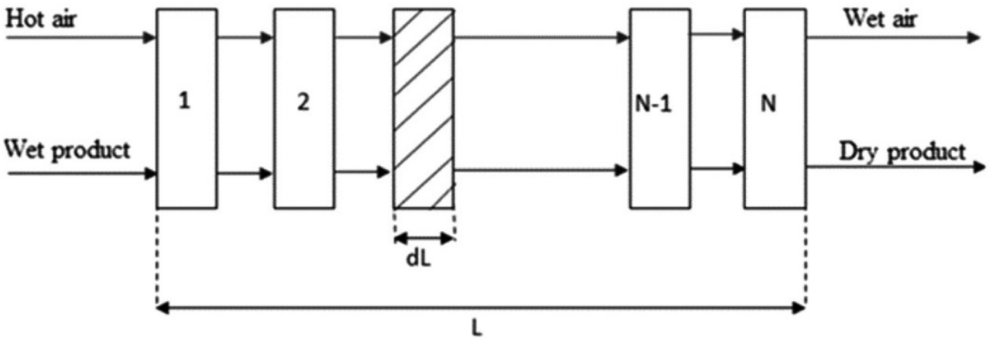


Figure 7. Diagram of the model, from [3].

The assumptions made are the following, based on the model proposed in [5]:

- All the drying occurs during the falling rate period (see Section 2.2 Moisture equilibrium – drying rate).
- There is no accumulation between elements: the inlet mass flow of alfalfa is equal to the outlet mass flow of alfalfa of the previous element.
- Air mass flow and axial velocity are constant.
- Heat transfer between air and alfalfa is calculated with a global volumetric coefficient, assumed constant throughout the dryer.
- Conduction heat transfer is neglected.

The equations for the mass and energy balances for the control volumes were extracted from [5]:

- Mass balance for alfalfa (change in water content present in alfalfa):

$$\frac{d(M_p W)}{dt} = M_p \frac{dW}{dt} + W \frac{dM_p}{dt} = G_{p,in} W_{in} - G_{p,out} W - R_w M_p. \tag{14}$$

- Mass balance for air (change in water content present in air):

$$\frac{d(M_a Y)}{dt} = M_a \frac{dY}{dt} + Y \frac{dM_a}{dt} = G_{a,in} Y_{in} - G_{a,out} Y + R_w M_p. \tag{15}$$

- Energy balance for alfalfa:

$$\begin{aligned}\frac{d(M_p c_{p,p} T_p)}{dt} &= M_p\ c_{p,p}\ \frac{dT_p}{dt} + M_p\ T_p\ \frac{dc_{p,p}}{dt} + T_p\ c_{p,p} \frac{dM_p}{dt} \\ &= G_{p,in}\ c_{p,pin}\ T_{p,in} - G_{p,out}\ c_{p,p}\ T_p + U_v aV(T_a - T_p) \\ &\quad - R_w\ M_p\ q_{lat} - R_w\ M_b\ c_{p,v}\ (T_a - T_p) - Q_p.\end{aligned} \tag{16}$$

- Energy balance for air:

$$\begin{aligned}\frac{d(M_a c_{p,a} T_a)}{dt} &= M_a\ c_{p,a}\ \frac{dT_a}{dt} + M_a\ T_a\ \frac{dc_{p,a}}{dt} + T_a\ c_{p,a} \frac{dM_a}{dt} \\ &= G_a\ c_{p,ain}\ T_{a,in} - G_a\ c_{p,a}\ T_a - U_v aV(T_a - T_p).\end{aligned} \tag{17}$$

Drying rate (R_w) is calculated using equation (4). The drying rate constant is assumed constant and is calculated as the mean between the values provided for alfalfa stem and alfalfa leaves in equations (7) and (8), for temperatures greater than 80°C, that yields a value of $K = 0.0575$ min^{-1}. The equilibrium moisture content was calculated with equation (10), and its coefficients with equations (11) and (12). To calculate the value of water activity, the saturation pressures at different temperatures, as well as specific heat capacity of air ($c_{p,a}$) and of water vapor ($c_{p,v}$) were obtained from [14]. The values for the specific heat of alfalfa were extracted from [15].

- Residence time was calculated with equation (3)

$$\tau = \frac{L}{F_H \cdot D \cdot N(\alpha \pm aV_a)},$$

where the drum loading factor $F_H = \pi$ and the constant $a = 0.026$ (from [2]).

- The dryer mass hold-up (M_p) is then calculated as

$$M_p = G_{p,in} \cdot \tau. \tag{18}$$

- The heat loss through the dryer shell ($\dot{Q}_p$) was assumed constant and equal to

$$Q_p = h \cdot A_{shell} \cdot (T_p - T_\infty). \tag{19}$$

where the heat loss coefficient (h) was obtained from [5]. An average value for temperature of alfalfa was selected (T_p = 87.5) and ambient temperature (T_∞) was set to 20°C.

- The global volumetric heat transfer coefficient ($U_v a$) was calculated with

$$U_v a = 0.52 \left(\frac{G_a}{A}\right)^{0.8}, \tag{20}$$

where A is the cross sectional area of the dryer.

3.2. Implementation in MATLAB/Simulink

The implementation of the model in Simulink is explained in this section, starting with a singular control volume, and building up to the full distributed model. The state space variables and equation are defined, as well as the inputs and outputs of the model. Finally, the MATLAB code for the start-up of the model is presented, together with the list of parameters and its values.

3.2.1. Implementation of a single control volume element

For the implementation of the model in Simulink, first the state space equations of the model are defined. We take the mass balance equations (equations (14)-(17)) as a starting point, and define the states as the water content in alfalfa and air, and the energy present in alfalfa and air. However, there are 4 equations and 5 variables (M_p, W, Y, T_p, T_a), so an extra state is defined as the alfalfa mass hold-up, and its corresponding equation (see equation (21)).

Table 3. States of the model

State	Description
x_1	Water in alfalfa (kg)
x_2	Wate in air (kg)
x_3	Energy in alfalfa
x_4	Energy in air
x_5	Alfalfa mass hold-up

The five state equations of the model are then the following:

- The mass balance for water in alfalfa:

$$x_1 = G_{p,in}(t)\ W_{in}(t) - G_{p,out}(t) W(t) - R_w(t)\ M_p(t).$$

- The mass balance for water in air:

$$x_2 = G_{a,in}\ Y_{in} - G_{a,out}\ Y + R_w(t)\ M_p.$$

- The energy balance for alfalfa:

$$x_3 = G_{p,in}(t)\ c_{p,pin}(t)\ T_{p,in}(t) - G_{p,out}(t)\ c_{p,p}(t)\ T_p(t) + U_v a V(T_a(t) - T_p(t)) -$$
$$- R_w(t)\ M_p(t)\ q_{lat}(t) - R_w(t)\ M_p(t)\ c_{p,v}(t)\ (T_a(t) - T_p(t)) - Q_p.$$

- The energy balance for air:

$$x_4 = G_a\ c_{p,ain}(t)\ T_{a,in}(t) - G_a\ c_{p,a}(t)\ T_a(t) - U_v a V(T_a(t) - T_p(t)).$$

The alfalfa mass hold-up balance:

$$x_5 = \frac{d(M_p)}{dt} = G_{p,in}(t) - G_{p,out}(t). \tag{21}$$

The implementation of the state equations in Simulink is performed as seen in Figure 8. The function blocks contain the state space equations, and the integrator blocks are set to an initial value (Figure 9).

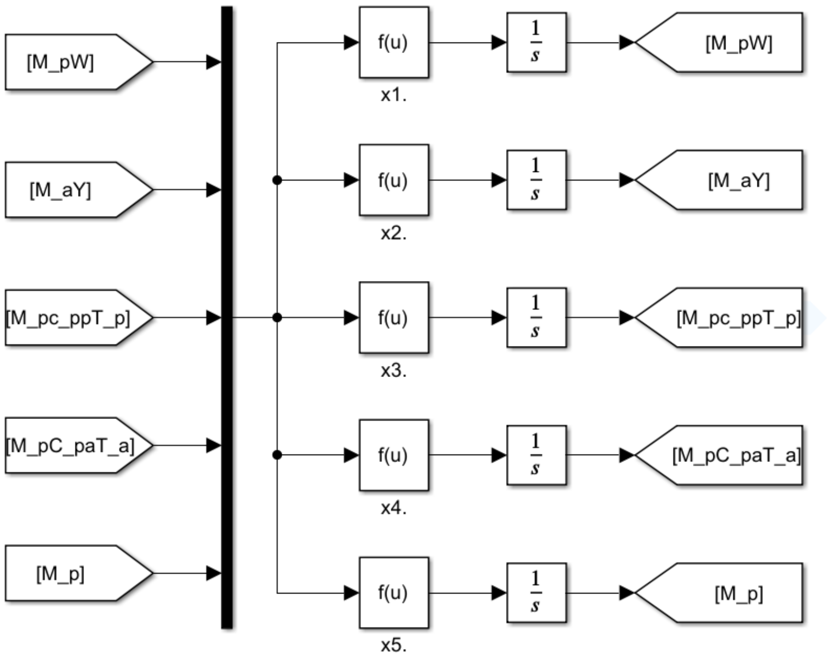
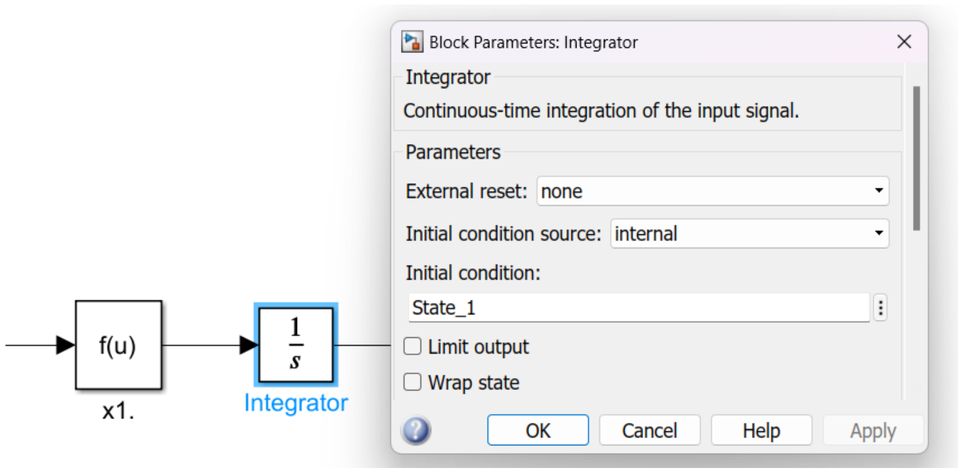


Figure 8. Implementation of state equations in Simulink



Block Parameters: Integrator

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

State_1

☐ Limit output

☐ Wrap state

Figure 9. Initial condition of integrator set to "State_1".

The input and output variables of each control volume are presented in Table 4.

Table 4. Inputs and outputs of the model

Inputs	Description	Outputs
W_{in}	Alfalfa moisture content	W_{out}
Y_{in}	Air moisture content	Y_{out}
$T_{p,in}$	Alfalfa temperature	$T_{p,out}$
$T_{a,in}$	Air temperature	$T_{a,out}$
$G_{p,in}$	Alfalfa mass flow rate	$G_{p,out}$
$G_{a,in}$	Air mass flow rate	$G_{a,out}$

The inputs and outputs are implemented in Simulink for each control volume, as they are then connected one to the next. The inputs are fed to a series of *Goto* blocks, where they are defined as local variables (Figure 10). The mass flow rate of air was assumed to be constant in section 3.1, and for that reason the input is fed directly into the output. The input mass flow rate of alfalfa, however, could change in this model. For the sake of simplicity of the linearization and control that will be covered later, it is left constant from now on and fed directly into the output too.

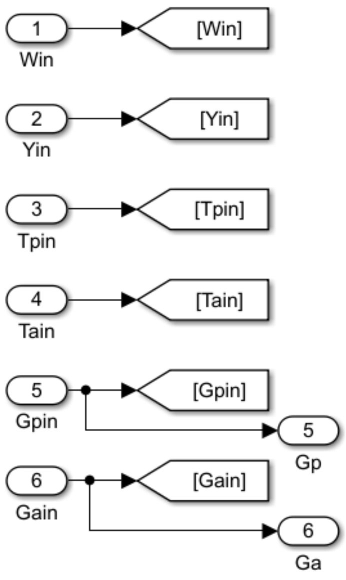


Figure 10. Implementation of inputs in Simulink.

The outputs are computed from the state variables, as shown in Figure 11.

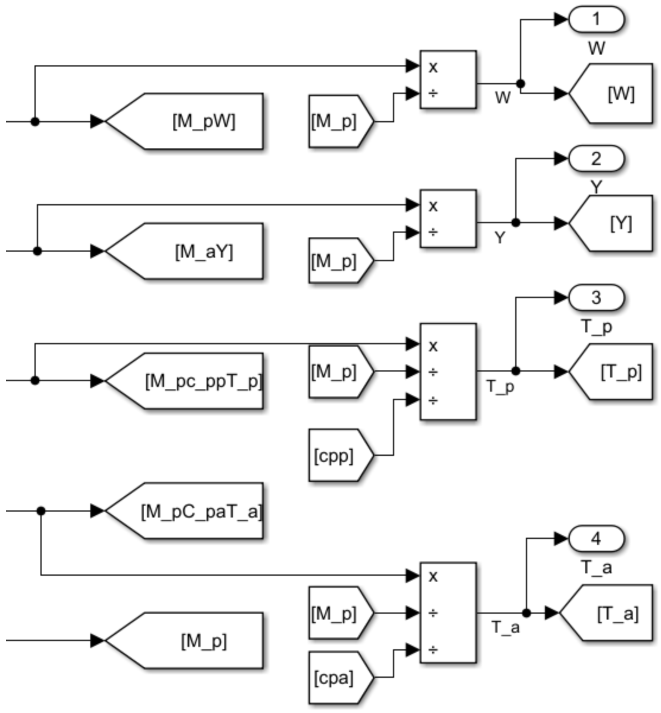


Figure 11. Implementation of outputs in Simulink.

The implementation of the drying rate (R_w) is done with equation (4). The blocks used to do so are represented in Figure 12.

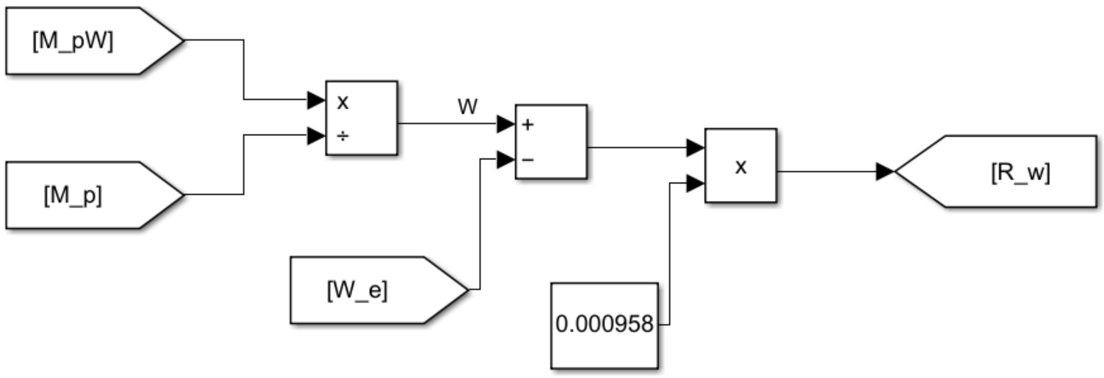


Figure 12. Implementation of drying rate equation

The implementation of the alfalfa equilibrium moisture content (W_e) calculation, defined in equation (10), is done as shown in Figure 13.

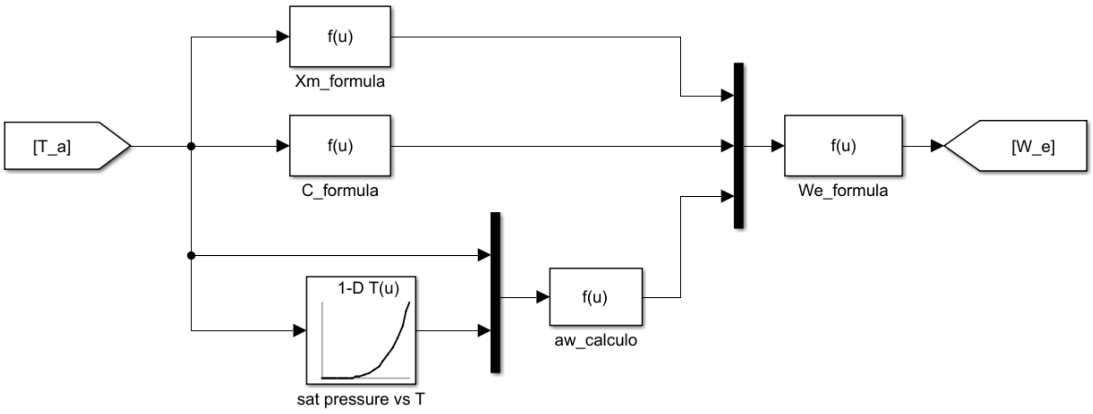


Figure 13. Implementation of alfalfa equilibrium moisture equation.

The function blocks *Xm_formula* and *C_formula* contain equations (11) and (12), that compute the parameters X_m and *C:*

$$X_m = 3.9229 \cdot 10^{-4} \cdot exp(\tfrac{1858.8}{T+273.16}), \text{ and}$$

$$C = 323.1769 \cdot exp\left(-\frac{974.55}{T+273.16}\right).$$

The saturation pressure needs to be calculated to obtain the value of water activity (a_w), as defined in equation (13). For that matter, a look-up table is introduced (Figure 14). Pressure is in kPa, temperature in °C.

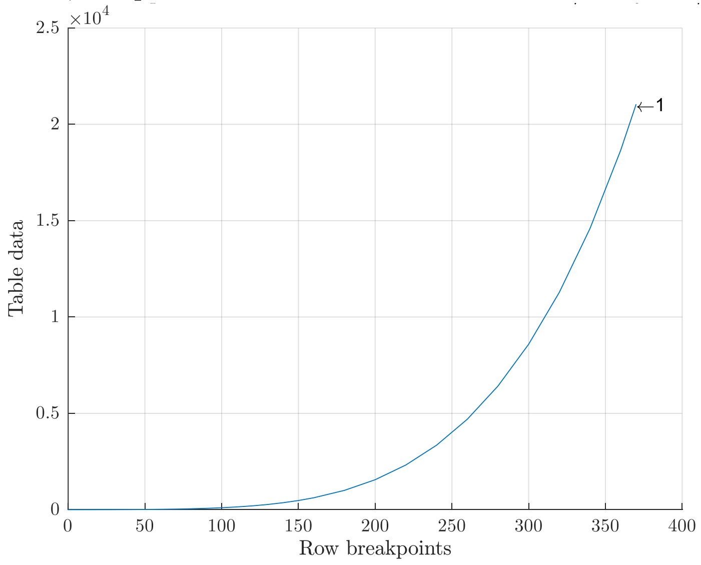


Figure 14. Plot 'saturation pressure (kPa) vs. air temperature (°C)'.

The function block called *aw_calculo* takes air temperature and saturation pressure and returns the water activity value (a_w). The formula is shown in equation (22).

$$a_w = \frac{M_p \cdot \mathrm{R}}{p_0 \cdot V \cdot 18 \cdot (T_a + 273.16)}, \tag{22}$$

where R is the ideal gas constant, in kJ/K · kmol, and 18 is the molar mass of water.

The *'We_formula'* function block computes equation (10), finally returning the W_e value:

$$W_e = \frac{X_m C a_w}{(1 - a_w)[1 + (C - 1)a_w]}. \tag{23}$$

The values of the specific heat for air ($c_{p,a}$), water vapor ($c_{p,v}$), and alfalfa ($c_{p,p}$), and the values for saturation pressure (p_0), are introduced as look-up tables as a function of temperature.

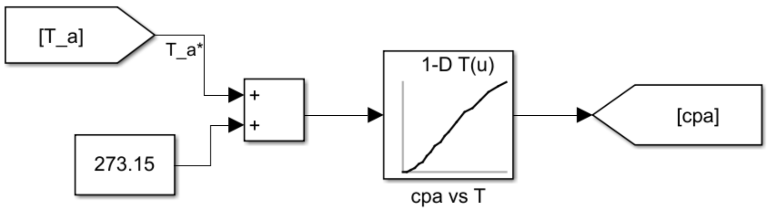


Figure 15. Implementation of specific heat of air.

The implementation of $c_{p,a}$ is done as shown in Figure 15. A look-up table is implemented with the values in kJ/kg · K for temperatures in K (Figure 16).

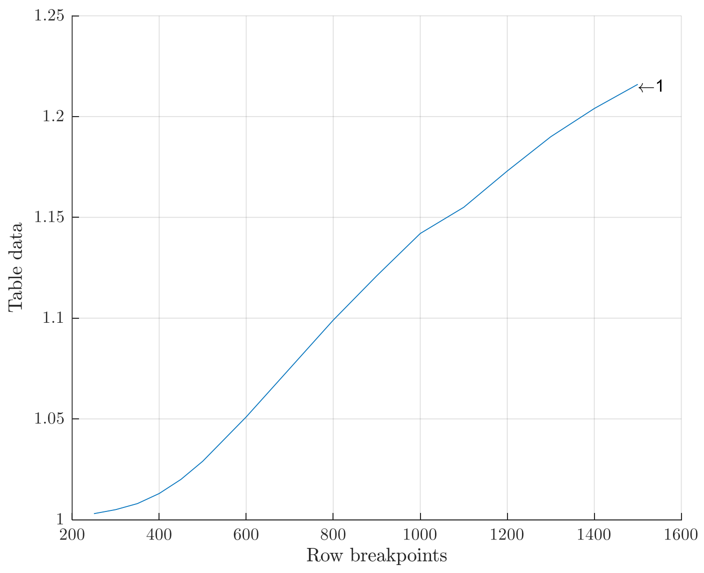


Figure 16. Plot 'specific heat of air (kJ/kg · K) vs air temperature (K)'.

For the temperature selected for water vapor $c_{p,v}$, the mean between the temperature of alfalfa and air was selected. This is an approximation, as water heats up since its vaporization at the temperature of alfalfa until it reaches the temperature of air (Figure 17).

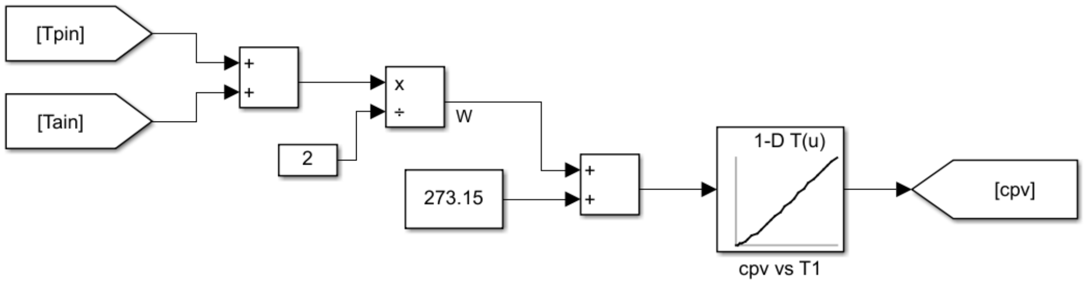


Figure 17. Implementation of specific heat of water vapor.

A look-up table is implemented with the values in kJ/kg · K for temperatures in K (Figure 18).

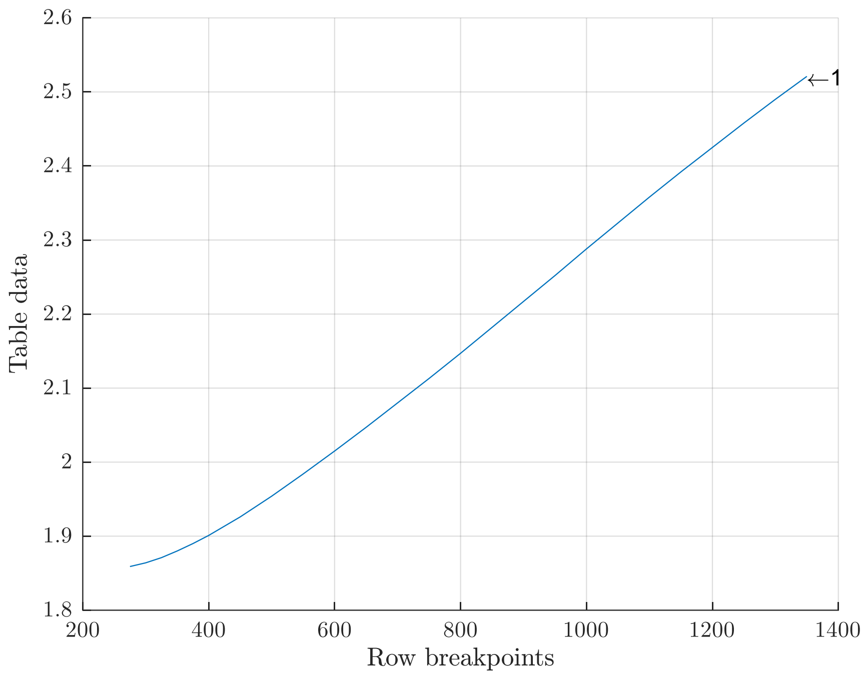


Figure 18. Plot 'specific heat of water vapor (kJ/kg · K) vs air temperature (K)'

For the implementation of the specific heat of alfalfa, a piecewise function was implemented to resemble the behavior shown in the source (Figure 19). To do so, a constant value of 0.3 cal/g · K is selected for moisture contents (in wet basis) from 0 to 4%. Then, for greater than 18% the specific heat is assumed constant too, with a value of 0.4731 cal/g · K. For values of moisture in between (from 4% to 18%), the specific heat is approximated with a straight line connecting both values (Figure 20).

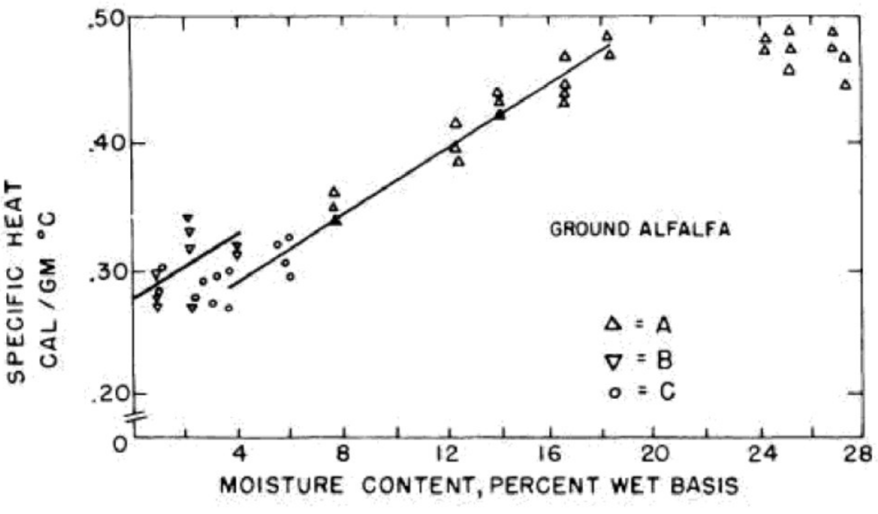


Figure 19. Specific heat of alfalfa, from [15].

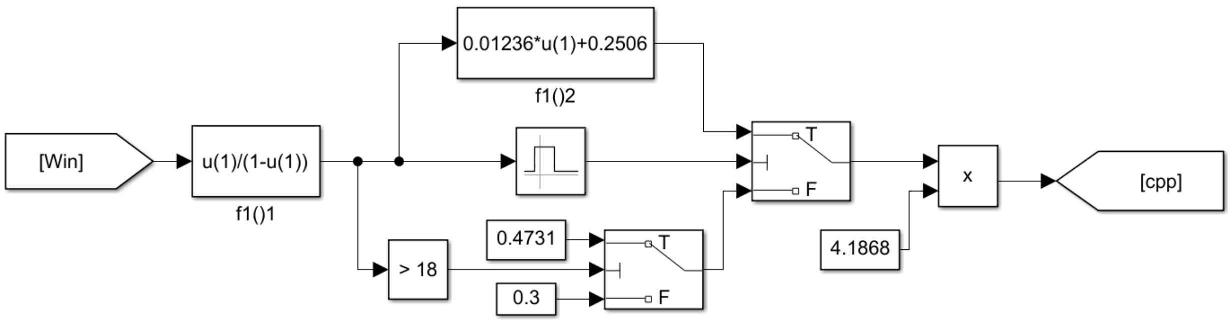


Figure 20. Implementation of specific heat of alfalfa

The implementation of equation (20), that computes the global volumetric heat transfer coefficient is done as indicated in Figure 21.

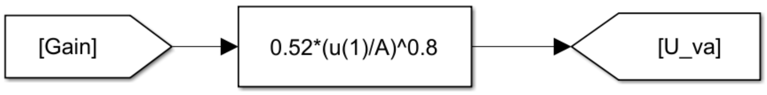


Figure 21. Implementation of global volumetric heat transfer coefficient.

The variables whose values were implemented with Simulink above (p_0, $c_{p,a}$, $c_{p,v}$, $c_{p,p}$, and U_va) are then fed together with the inputs and states to the state equations with a *Mux* block (Figure 22).

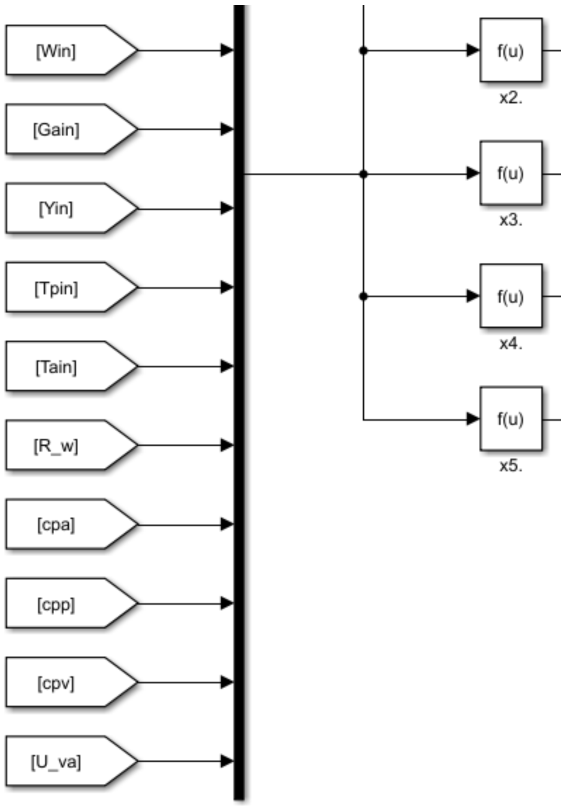


Figure 22. Mux block collecting states, inputs and variables feeding the state equation blocks.

The implementation of each of the single control volumes is encapsulated as a subsystem, with the six inputs and outputs (Figure 23).

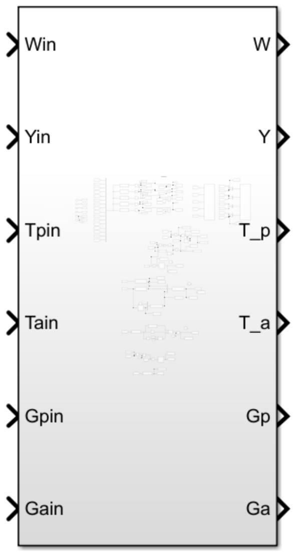


Figure 23. Simulink subsystem block modelling a single control volume.

3.2.2. Implementation of the distributed model

The subsystem blocks like the one in Figure 23 are then replicated and connected one after another, to create the distributed model with n control volumes. The inputs for alfalfa moisture content (W_{in}) and air temperature ($T_{a,in}$) are fed by a switch, that allows an input block to be connected to perform the linearization or a step to be introduced into the system. The rest of the inputs are imported form the MATLAB workspace directly. The model is implemented with four control volumes, as shown in Figure 24.

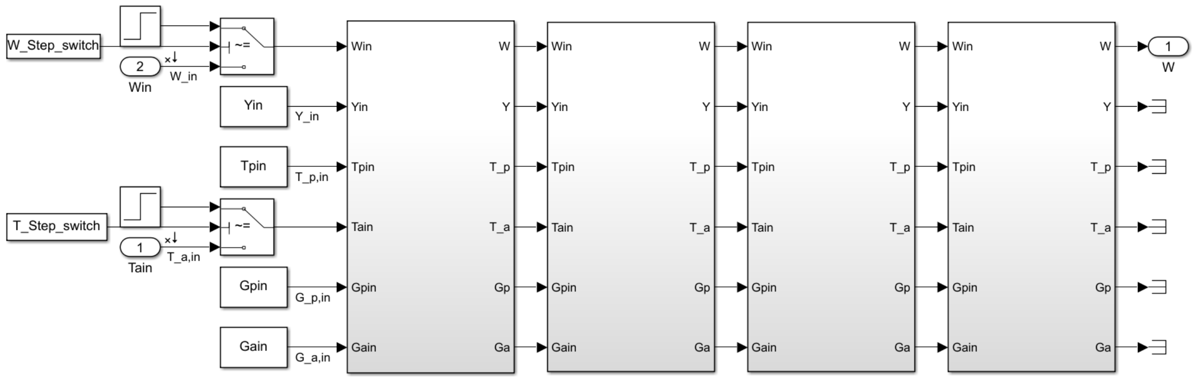


Figure 24. Simulink implementation of the distributed model with four control volumes.

3.2.3. Implementation of parameters

The values for the different parameters used are presented in Table 5. Air axial velocity (V_a) was selected from the range of 3-5 m/s, mentioned in [16]. Air inlet mass flow rate was calculated from the air volume rate in [5], and a value of density for air at 400°C. Alfalfa inlet mass flow rate was selected from the typical values of industrial rotary dryers of the dimensions used (20-75 t/h, from [17], and similar dimensioned rotary dryers from [18]). Alfalfa inlet temperature was selected at 15°C and the rest of values were extracted from the model presented in [5].

Table 5. Values of parameters

	Parameter	Value
D	Diameter	3.3 m
L	Length	20 m
V	Volume	171.06 m^3
α	Slope	0.005°
N	Rotational speed	10 rpm
Q_p	Heat loss through dryer shell	3.9188 kW
V_a	Air axial velocity	3 m/s
$G_{p,in}$	Alfalfa inlet mass flow rate	5.7 kg/s
$G_{a,in}$	Air inlet mass flow rate	17.1 kg/s
$T_{p,in}$	Alfalfa inlet temperature	15°C
Y_{in}	Air inlet absolute humidity	0.02 kg H_2O/kg air

4. Linearization of the model

In this chapter, the linearization of the model is covered. The system presents high non-linearity, and to analyze it and apply classic control theory to understand the system's properties, a linear model is needed. The result of the linearization will be a set of linear models that model the dynamics of the system at a set of operating points. This set is first selected based on the behaviour of the model and the typical values for rotary dryers found in the literature. Then, the linearization is performed with MATLAB. Finally, the validation is done by comparing the response of the linear model against the non-linear one to step inputs.

4.1. Operating points

To analyze the model implemented in MATLAB/Simulink, a set of simulations are performed at fixed input air temperature ($T_{a,in}$) and alfalfa moisture content (W_{in}). The objective is to obtain the output alfalfa moisture content (W_{out}) when steady-state conditions are reached. The simulation time was set high enough accordingly, as it can be seen in the sample of simulations represented in Figure 25, with each of the signals representing a W_{out} for a given $T_{a,in}$ and W_{in}. As it can be seen, steady-state conditions are reached.

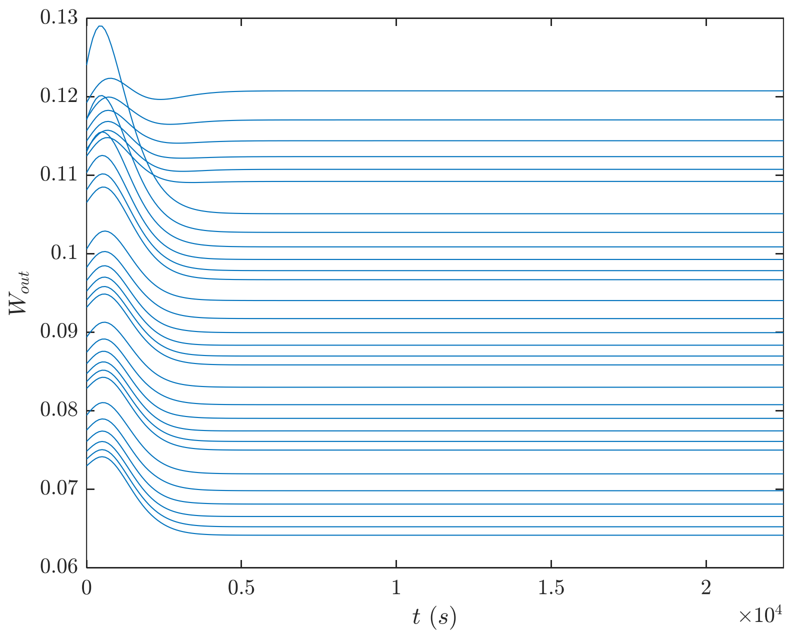


Figure 25. Sample of the simulations performed to obtain W_{out}.

The values are presented in Table 6, in kg of water per kg of alfalfa × 100 (the notation is in percentage, although it represents a ratio of kilograms). The W_{in} and $T_{a,in}$ values are selected to be similar to the ones used in [5], where they use the set 0.44, 0.54, 0.64, 0.74 and 0.84 for the first variable, and perform a step of input temperature starting from 480°C.

Table 6. W_{out} values obtained after simulations at different $T_{a,in}$ and W_{in}.

	$T_{a,in}$ (°C)													
		460	465	470	475	480	485	490	495	500	505	510	515	520
W_{in}	0.85	328.8%	125.6%	28.6%	17.1%	14.3%	13.0%	12.3%	11.7%	11.3%	11.1%	10.9%	10.7%	10.5%
	0.75	75.7%	21.2%	14.8%	12.7%	11.7%	11.0%	10.5%	10.2%	9.9%	9.7%	9.6%	9.4%	9.3%
	0.65	59.8%	18.7%	13.4%	11.5%	10.6%	9.9%	9.4%	9.1%	8.9%	8.7%	8.5%	8.4%	8.3%
	0.55	46.9%	16.6%	12.0%	10.4%	9.5%	8.8%	8.4%	8.1%	7.8%	7.7%	7.5%	7.4%	7.2%
	0.45	37.0%	14.6%	10.7%	9.2%	8.4%	7.7%	7.3%	7.0%	6.8%	6.6%	6.5%	6.3%	6.2%

The values of W_{out} are within the expected range, as they lay around the 10-12% w.b. (corresponding to $11.\widehat{1}$-13.64% d.b.), that is the desired setpoint range to avoid underdrying and overdrying (see chapter 1). In [5], at 480°C the values of W_{out} at steady state range from around 8.5% to 17.5% d.b., so the model implemented in this work shows similar results, from 8.4% to 14.3% d.b., that reflect the similarities in the models used, and the use of many similar parameter values.

4.2. Linearization

To linearize the plant, it is decided to center the operating points around the setpoint for W_{out}, that is selected to be 13%. This is within the setpoint range, and close to the maximum moisture to avoid spoilage, corresponding to the minimum heating use and thus minimum operating cost. As the system will then operate around those conditions, a set of five W_{out} values are chosen: 10%, 11.5%, 13%, 14.5% and 16%. The W_{in} values remain the ones used in the previous section: 0.45, 0.55, 0.65, 0.75 and 0.85. Now, the $T_{a,in}$ values that satisfy the steady state at those operating points need to be obtained and are presented in Table 7.

Table 7. $T_{a,in}$ values corresponding to the selected steady-state operating points.

	W_{out}					
		0.1	0.115	0.13	0.145	0.16
W_{in}	0.85	500	499	484	479	476
	0.75	498	479	472	469	467
	0.65	486	475	471	468	467
	0.55	477	471	468	467	465
	0.45	472	468	467	466	465

The model to be linearized consists of two inputs, $T_{a,in}$ and W_{in}, and one output, W_{out}. To perform the linearization, first the MATLAB function *findop* is used (Figure 26). It takes a Simulink model and an array of operating points as inputs and returns the steady-state operating points.

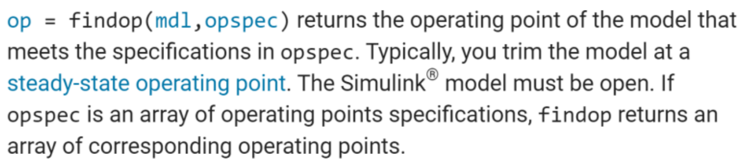

`op = findop(mdl,opspec)` returns the operating point of the model that meets the specifications in `opspec`. Typically, you trim the model at a steady-state operating point. The Simulink® model must be open. If `opspec` is an array of operating points specifications, `findop` returns an array of corresponding operating points.

Figure 26. The 'findop' function, from the MATLAB Documentation.

Although the steady state operating points had been manually obtained, this step is necessary to have the precise steady-state operating point conditions to feed the linearization MATLAB function. As the *findop* tool shows difficulties finding the steady state operating points while fixing the W_{out} values, in the array of operating points opspec the closest conditions to the desired ones are fed from the ones in Table 6. The function *newspec* is used to set the states to the desired values and to put boundaries to the $T_{a,in}$ values in the steady-state search, and to fix the W_{out} values. The *findop* function then returns the steady-state operating point conditions.

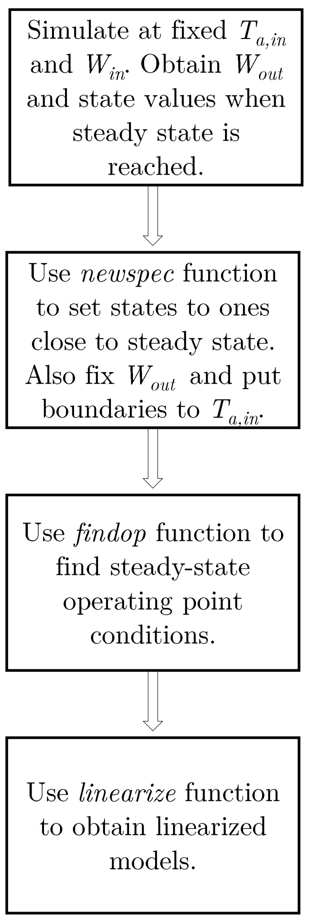
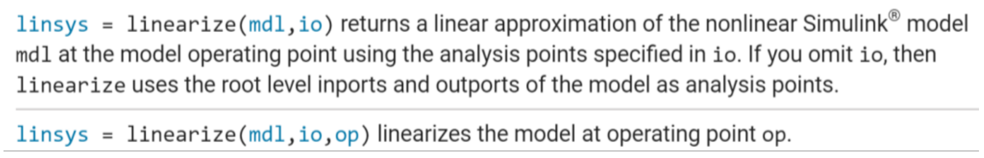

Simulate at fixed $T_{a,in}$ and W_{in}. Obtain W_{out} and state values when steady state is reached.
Use *newspec* function to set states to ones close to steady state. Also fix W_{out} and put boundaries to $T_{a,in}$.
Use *findop* function to find steady-state operating point conditions.
Use *linearize* function to obtain linearized models.

Figure 27. Step by step process to linearize the model.

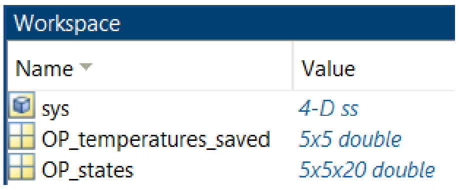
The next step is to linearize the model at those operating points. The MATLAB function called *linearize* is used (Figure 28). It takes the Simulink model, the analysis points and the operating points as inputs and returns a linear approximation of the model at those operating points.


`linsys = linearize(mdl,io)` returns a linear approximation of the nonlinear Simulink® model `mdl` at the model operating point using the analysis points specified in `io`. If you omit `io`, then `linearize` uses the root level inports and outports of the model as analysis points.

`linsys = linearize(mdl,io,op)` linearizes the model at operating point `op`.

Figure 28. The 'linearize' function, from the MATLAB Documentation.

The analysis points are obtained via the *getlinio* function. It returns the inputs and outputs of the model. The operating points are first obtained with the function *operpoint*, and then are iteratively modified to match the ones obtained previously with *findop*. Finally, the *linearize* function is called iteratively for all the operating points and the linearized models are obtained. The step by step process diagram that sums up the linearization process can be seen in Figure 27. The temperatures, the state values for the steady state operating points and the linearized models are saved as the result of the linearization process, as seen in Figure 29.


Workspace

Name	Value
sys	4-D ss
OP_temperatures_saved	5x5 double
OP_states	5x5x20 double

Figure 29. Output of the linearization process.

The OP_temperatures_saved.mat file is a 5x5 double containing the values of the $T_{a,in}$ at the operating points, where the rows and columns coincide with the input W_{in} and output W_{out} values that define the operating point, as in Table 7.

The OP_states.mat file contains in a similar way the states of the operating points, where the first two dimensions represent the input W_{in} and output W_{out} values, and the third dimension allows the 20 states of the non-linear model to be stored.

The sys.mat file is a ss object, where the first dimension refers to the output of the system, in this case only W_{out}, the second dimension refers to the inputs, T_{ain} and W_{in}, and the third dimension is used to access each of the operating points. They are numbered as shown in Table 8.

Table 8. Numbering of the linearized models.

		W_{out}				
		0.1	0.115	0.13	0.145	0.16
	0.85	1	2	3	4	5
	0.75	6	7	8	9	10
W_{in}	0.65	11	12	13	14	15
	0.55	16	17	18	19	20
	0.45	21	22	23	24	25

As an example, the linearized system number 1 (corresponding to $W_{in} = 0.85$ and $W_{out} = 0.1$) from input $T_{a,in}$ to output W_{out} is analyzed. As it can be observed in Table 9, there are twelve states, three per control volume. This is due to the fact that in the model there were initially five states per control volume, but later the variables $G_{p,in}$ and $G_{p,in}$ were left fixed, to further analyze the system and develop control techniques only with the two inputs $T_{a,in}$ and $G_{p,in}$. The function *linearize* returns then a three state space model per control volume, and a total of twelve state for the full model.

Table 9. State space linear model at operating point W_{in} = 0.85 and W_{out} = 0.1, A matrix from input $T_{a,in}$.

A											
-2.14E-03	0	-2.99E-08	0	0	0	0	0	0	0	0	0
-3.34E-02	-3.25E-02	3.13E-02	0	0	0	0	0	0	0	0	0
0	3.22E-02	-3.57E-02	0	0	0	0	0	0	0	0	0
1.18E-03	0	0	-2.12E-03	0	-9.63E-08	0	0	0	0	0	0
0	1.18E-03	-3.82E-07	-1.09E-02	-3.23E-02	3.11E-02	0	0	0	0	0	0
0	0	3.26E-03	0	3.16E-02	-3.48E-02	0	0	0	0	0	0
0	0	0	1.16E-03	0	0	-2.09E-03	0	-9.65E-07	0	0	0
0	0	0	0	1.16E-03	1.04E-06	6.47E-02	-3.18E-02	3.07E-02	0	0	0
0	0	0	0	0	3.23E-03	0	3.09E-02	-3.41E-02	0	0	0
0	0	0	0	0	0	1.14E-03	0	0	-2.08E-03	0	-1.12E-06
0	0	0	0	0	0	0	1.14E-03	-6.22E-07	-7.50E-02	-3.15E-02	3.03E-02
0	0	0	0	0	0	0	0	3.28E-03	0	3.05E-02	-3.37E-02

The B matrix shows that only the first control volume is directly affected by the input $T_{a,in}$. The C matrix shows accordingly that only states from the last control volume affect the output. Finally, the D matrix shows that input and output are not affected directly. Matrices B, C and D can be seen in

Table 10. State space linear model at operating point W_{in} = 0.85 and W_{out} = 0.1, matrices 'B', 'C' and 'D' from input '$T_{a,in}$'.

B	C'	D
0	0	0
-4.03E-03	0	
18.04	0	
0	0	
0	0	
0	0	
0	0	
0	0	
0	0	
0	1.97E-04	
0	0	
0	0	

The linearized system number '1' now from input $T_{a,in}$ to output W_{out} is analyzed. Similar observations can be made as for the input W_{in}, where only the first control volume is affected by the input, and only the last control volume affects the output (Table 11).

$$\frac{-1.806\cdot10^{-16}\mathrm{s}^{7}-2.568\cdot10^{-17}\mathrm{s}^{6}-1.623\cdot\mathbf{10}^{-18}\mathrm{s}^{5}-5.293\cdot\mathbf{10}^{-20}\mathrm{s}^{-4}-8.127\cdot10^{-22}\mathrm{s}^{3}-3.903\cdot10^{-24}\mathrm{s}^{2}-7.45\cdot10^{-27}\mathrm{s}-4.994^{-30}}{\begin{array}{c}s^{12}+0.2749\,s^{11}+0.02946\,s^{10}+0.001532\,s^{9}+3.904\cdot10^{-5}\,s^{8}+4.511\cdot10^{-7}\,s^{7}+2.888\cdot10^{-9}\,s^{6}+1.14\cdot10^{-11}\,s^{5}+\\+\,2.907\cdot10^{-14}\,s^{4}+4.831^{-17}\,s^{3}+5.07^{-20}\,s^{2}+3.06^{-23}\,s+8.116^{-27}\end{array}}$$

Table 11. State space linear model at operating point W_{in} = 0.85 and W_{out} = 0.1, from input 'W_{in}'.

A

-2.14E-03	0	-2.99E-08	0	0	0	0	0	0	0	0	0
-3.34E-02	-3.25E-02	3.13E-02	0	0	0	0	0	0	0	0	0
0	3.22E-02	-3.57E-02	0	0	0	0	0	0	0	0	0
1.18E-03	0	0	-2.12E-03	0	-9.63E-08	0	0	0	0	0	0
0	1.18E-03	-3.82E-07	-1.09E-02	-3.23E-02	3.11E-02	0	0	0	0	0	0
0	0	3.26E-03	0	3.16E-02	-3.48E-02	0	0	0	0	0	0
0	0	0	1.16E-03	0	0	-2.09E-03	0	-9.65E-07	0	0	0
0	0	0	0	1.16E-03	1.04E-06	6.47E-02	-3.18E-02	3.07E-02	0	0	0
0	0	0	0	0	3.23E-03	0	3.09E-02	-3.41E-02	0	0	0
0	0	0	0	0	0	1.14E-03	0	0	-2.08E-03	0	-1.12E-06
0	0	0	0	0	0	0	1.14E-03	-6.22E-07	-7.50E-02	-3.15E-02	3.03E-02
0	0	0	0	0	0	0	0	3.28E-03	0	3.05E-02	-3.37E-02

B	C'	D
5.7	0	0
0	0	
0	0	
0	0	
0	0	
0	0	
0	0	
0	0	
0	0	
0	1.97E-04	
0	0	
0	0	

The transfer functions modelling the linear relationship between the two inputs and the output are, in the case of the first operating point, corresponding to operating point $W_{\text{in}} = 0.85$ and $W_{\text{out}} = 0.1$, from $T_{\text{a,in}}$ to W_{out} is the following:

It can be observed that it has seven zeros and twelve poles (matching the number of states). Similarly, the transfer function corresponding to operating point $W_{\text{in}} = 0.85$ and $W_{\text{out}} = 0.1$, from W_{in} to W_{out} has the same poles (as expected) and there are eight zeros.

$$\frac{1.76\cdot 10^{-12}\text{s}^8 + 4.68\cdot 10^{-13}\text{s}^7 + 2.57\cdot 10^{-14}\text{s}^6 + 2.27\cdot 10^{-15}\text{s}^5 + 4.81\cdot 10^{-17}\text{s}^4 + 3.24\cdot 10^{-19}\text{s}^3 + 9.73\cdot 10^{-22}\text{s}^2 - 1.37\cdot 10^{-24}\text{s} - 7.41^{-28}}{\begin{array}{c} s^{12} + 0.2749\, s^{11} + 0.02946\, s^{10} + 0.001532\, s^9 + 3.904\cdot 10^{-5}\, s^8 + 4.511\cdot 10^{-7}\, s^7 + 2.888\cdot 10^{-9}\, s^6 + 1.14\cdot 10^{-11}\, s^5 + \\ + 2.907\cdot 10^{-14}\, s^4 + 4.831^{-17}\, s^3 + 5.07^{-20}\, s^2 + 3.06^{-23}\, s + 8.116^{-27} \end{array}}$$

To analyze the frequency response of the system, the Bode plots are obtained. First, the ones corresponding to the input $T_{a,in}$ are shown in Figure 30. As it can be observed in the magnitude plot between 1 rad/s and 10 rad/s, the slope is -100 dB/dec, that is expected as the transfer function has five more poles than zeros (at -20 dB/dec, they add up to -100 dB/dec). The DC gains are quite low, as it can be also seen by the small coefficients in the transfer function. In a future iteration, ideally the variables and parameters should be scaled to obtain numbers closer to 0, as there can be problems later on in the control design phases due to poor resolution while using certain functions and tools.

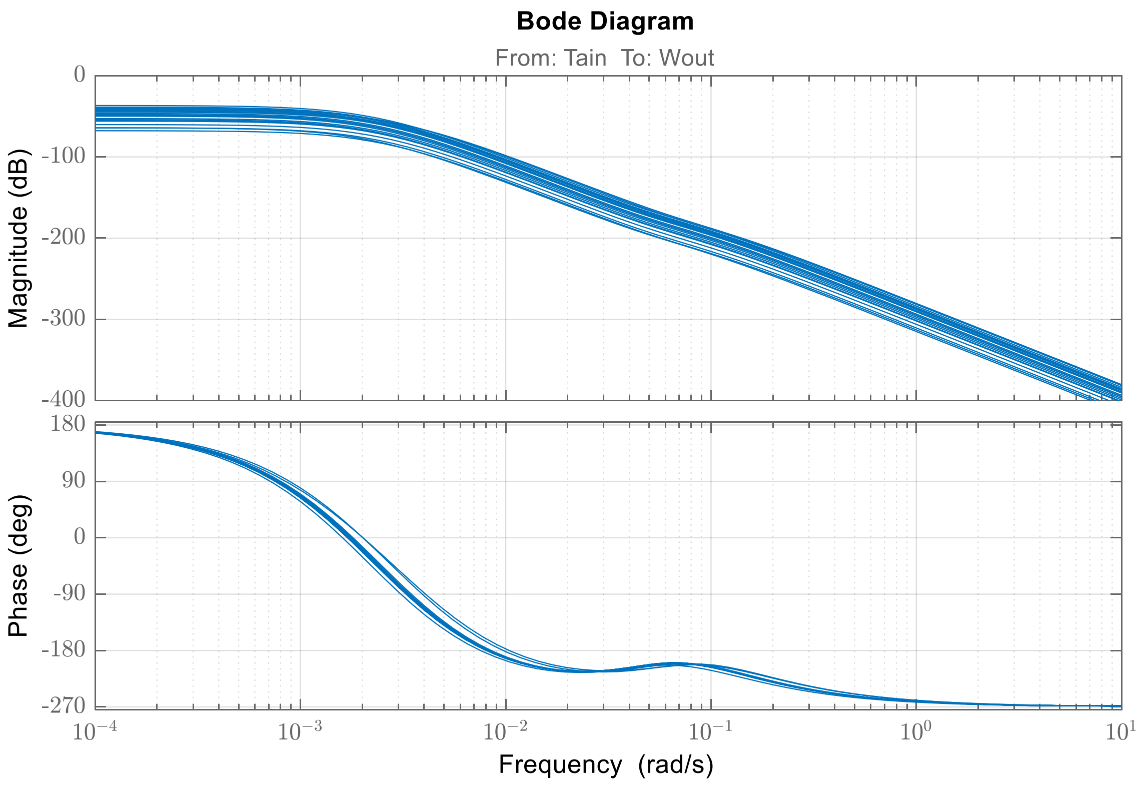


Figure 30. Bode plots of the systems from $T_{a,in}$ to W_{out}.

The Bode plots corresponding to the input W_{in} are shown in Figure 31. It can be checked that the slope at high frequencies is -80 dB/dec, corresponding to the extra four poles than zeros. It is observed that all operating points show a similar response qualitatively, showing some variation in magnitude and phase but maintaining the same overall behavior.

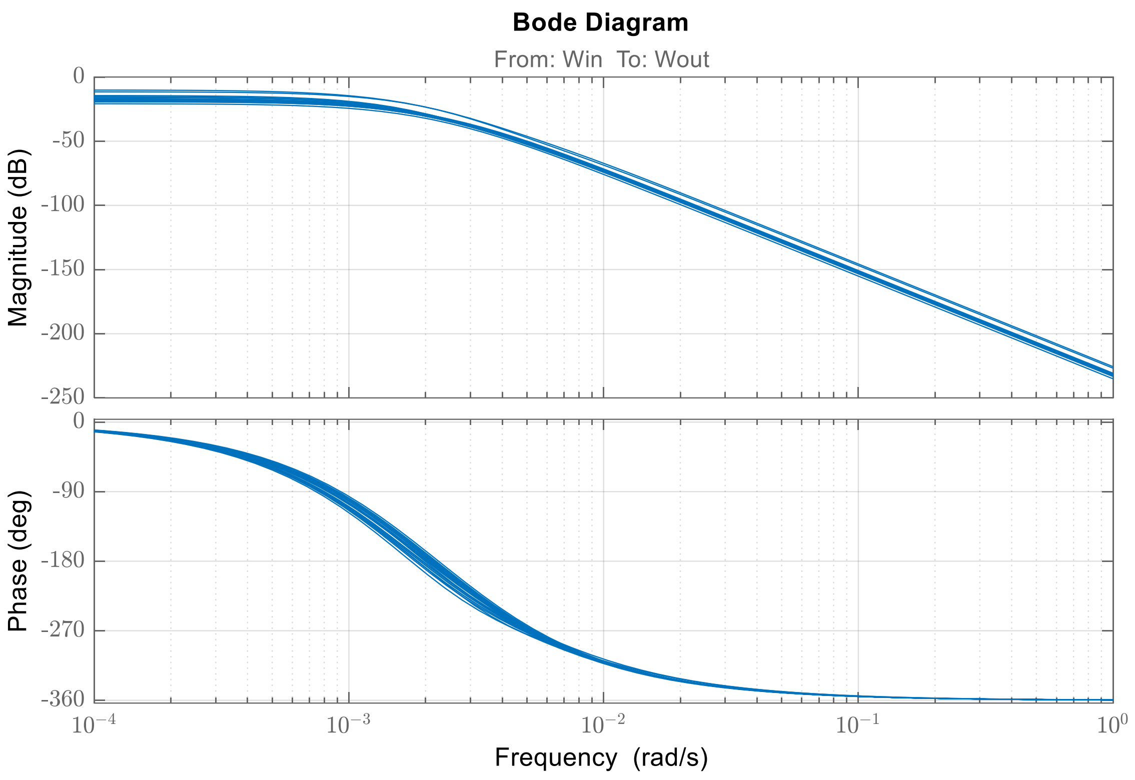


Figure 31. Bode plots of the systems from W_{in} to W_{out}.

4.3. Linear model open-loop validation

To validate the linear model, a set of steps is introduced in both the linear and non-linear system and the responses are compared. First, a step input of 1°C is introduced at $T_{a,in}$. The responses of both linear and non-linear systems are represented in the figures from Figure 32 to Figure 36, showing in the same plot the models with same W_{in}. It can be observed that the linear model is very accurate for these small steps, and only show appreciable differences at the scales shown in the operating points with higher W_{out} setpoint values.

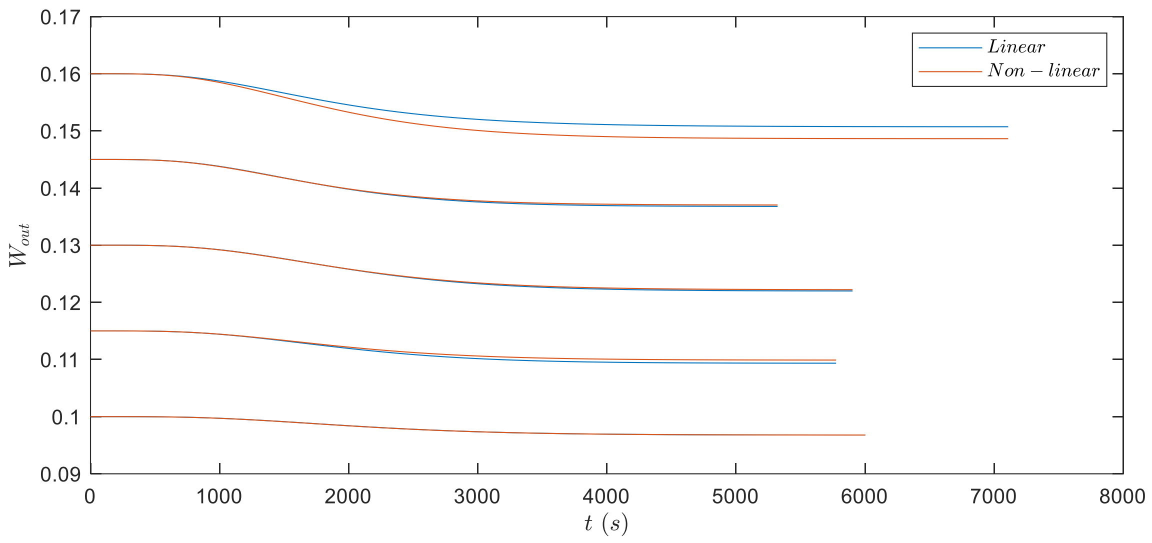


Figure 32. Responses to a step input of 1°C, W_{in}=0.45.

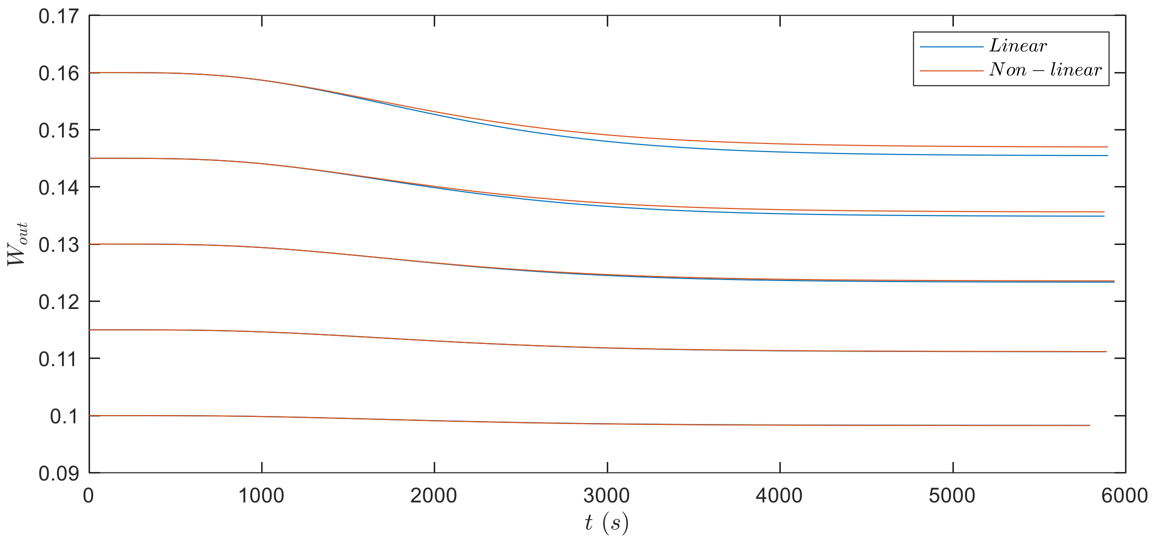


Figure 33. Responses to a step input of 1°C, W_{in}=0.55

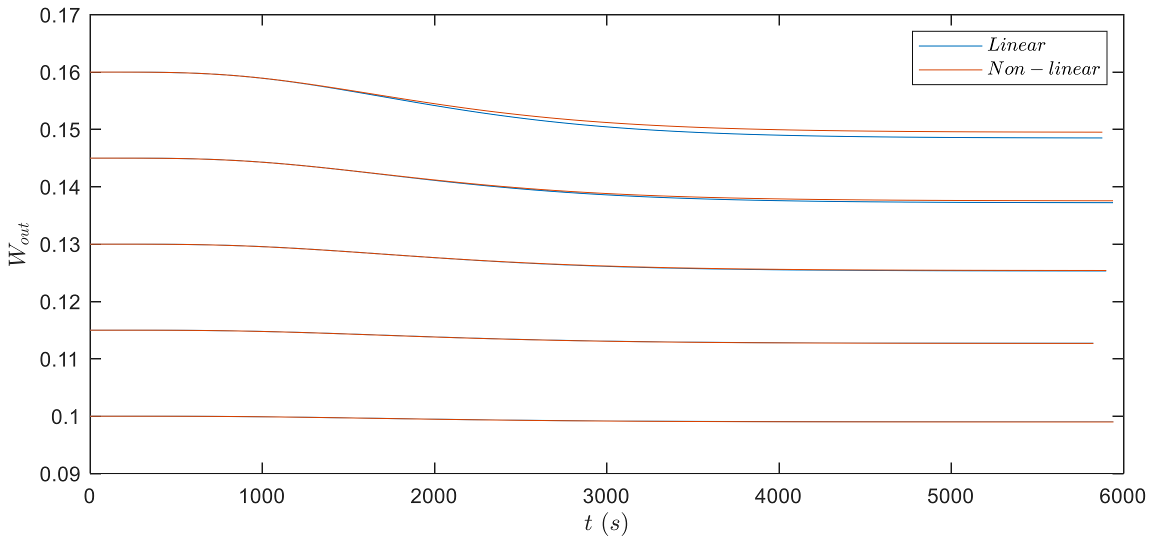


Figure 34. Responses to a step input of 1°C, W_{in}=0.65

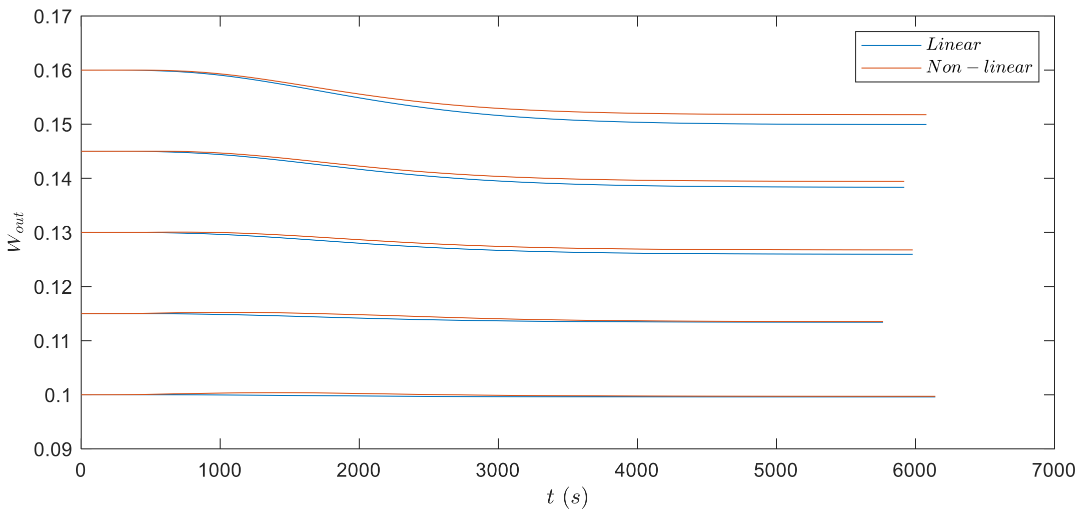


Figure 35. Responses to a step input of 1°C, W_{in}=0.75

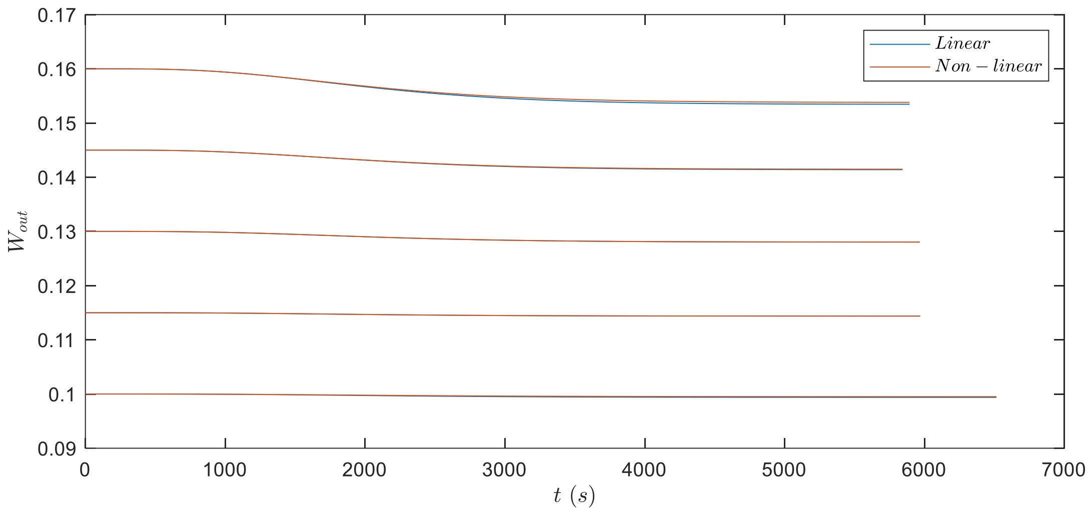


Figure 36. Responses to a step input of 1°C, W_{in}=0.85.

A step of 3°C is introduced to the models at $W_{in} = 0.75$, and the responses show little difference in the operating points with $W_{out} = 0.1$ and $W_{out} = 0.115$. The response differs more as W_{out} increases, and the one corresponding to the operating point of $W_{out} = 0.16$ shows a difference of around 1% moisture content (see Figure 37).

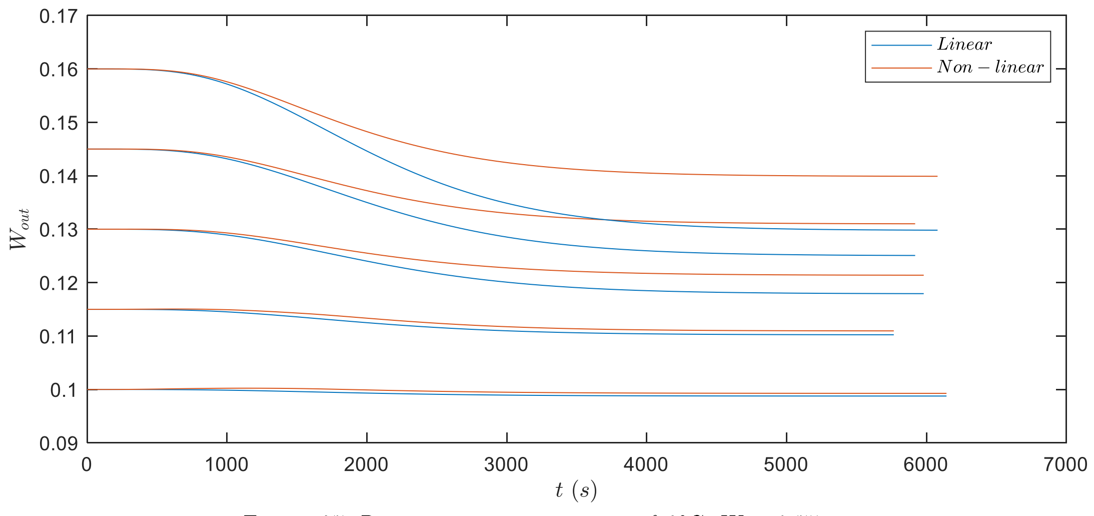


Figure 37. Responses to a step input of 3°C, W_{in}=0.75.

Now, the step responses of all the plants to 1°C are represented in the same plot. First, all the linear model ones (*Figure 38*), and then the non-linear ones (Figure 39). It is observed that all operating points show similar responses qualitatively, and that settling time is very similar. Only amplitude shows a significant difference among plants of different operating point conditions. This is true both for the linear and the non-linear models.

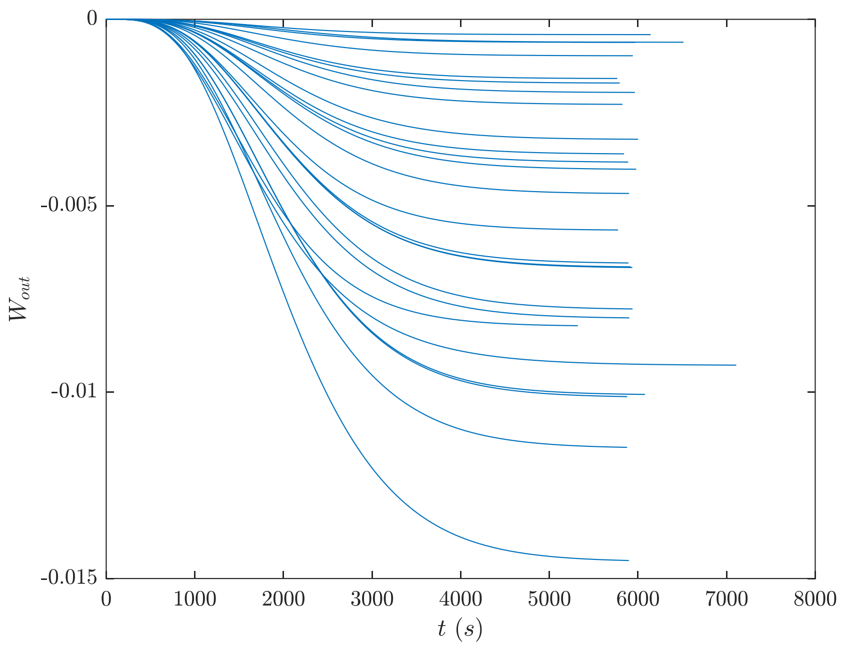


Figure 38. Step responses to 1°C, linear models

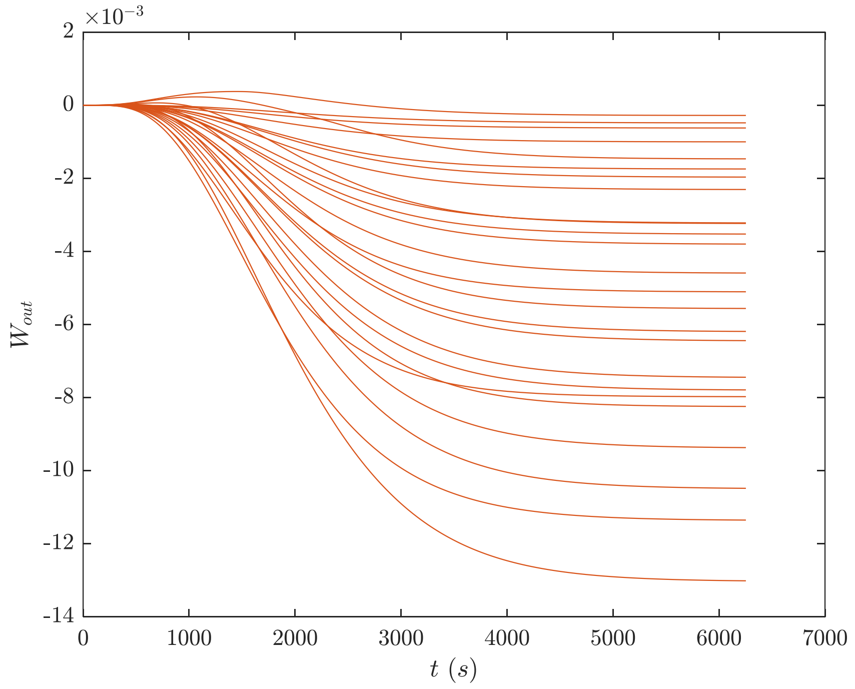


Figure 39. Step responses to 1°C, non-linear models.

Next, a set of steps is introduced now to the input W_{in}. First, the responses of the operating points corresponding to initial W_{in} of 0.45, 0.55 and 0.65 are shown (Figure 40, Figure 41, and Figure 42).

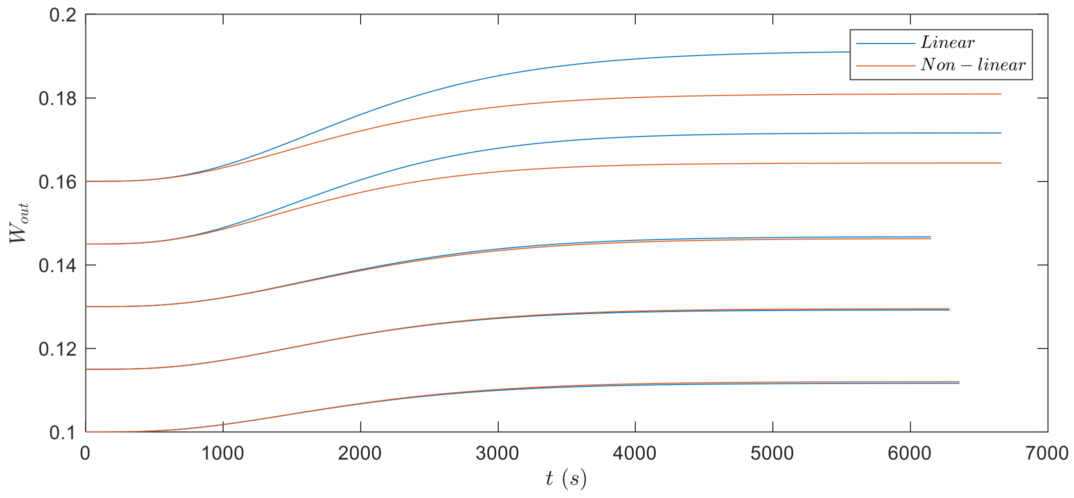


Figure 40. Response to a W_{in} step input of +0.1, W_{in}=0.45

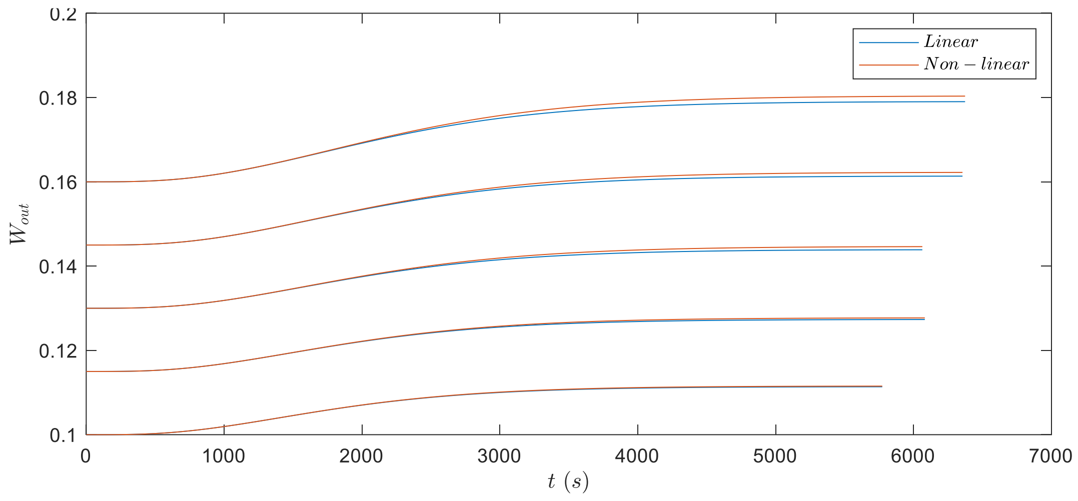


Figure 41. Response to a W_{in} step input of +0.1, W_{in}=0.55

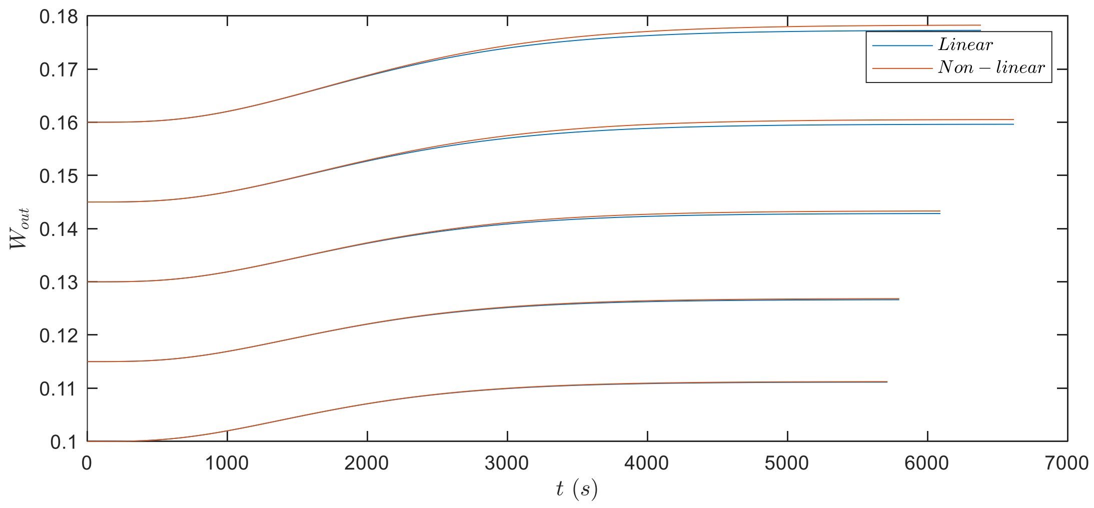


Figure 42. Response to a W_{in} step input of +0.1, W_{in}=0.65

However, when $W_{in} = 0.75$, the non-linearity starts to affect the model, and the linear approximation is no longer able to predict the output (Figure 43).

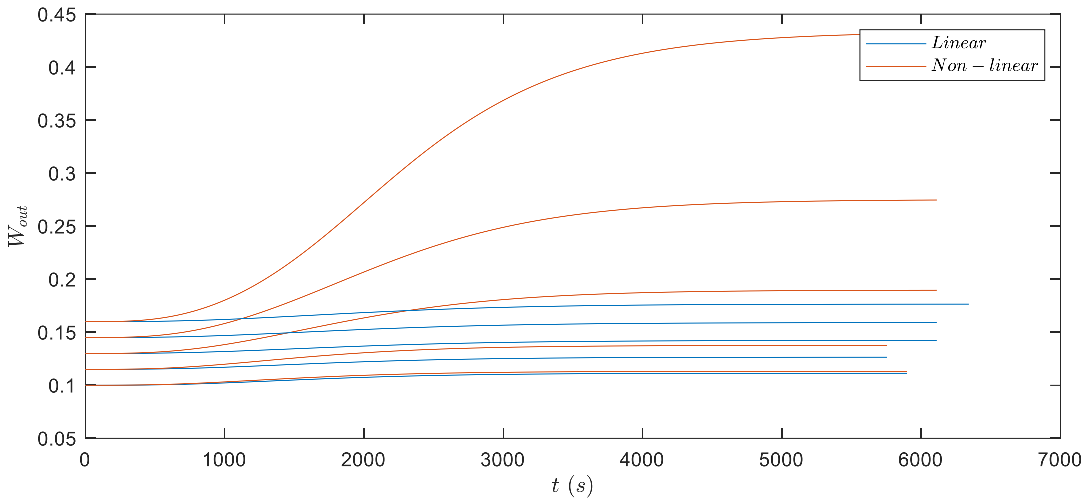


Figure 43. Response to a W_{in} step input of +0.1, W_{in}=0.75

If a smaller step, of +0.05 W_{in} is introduced, the linear model is still valid (Figure 44).

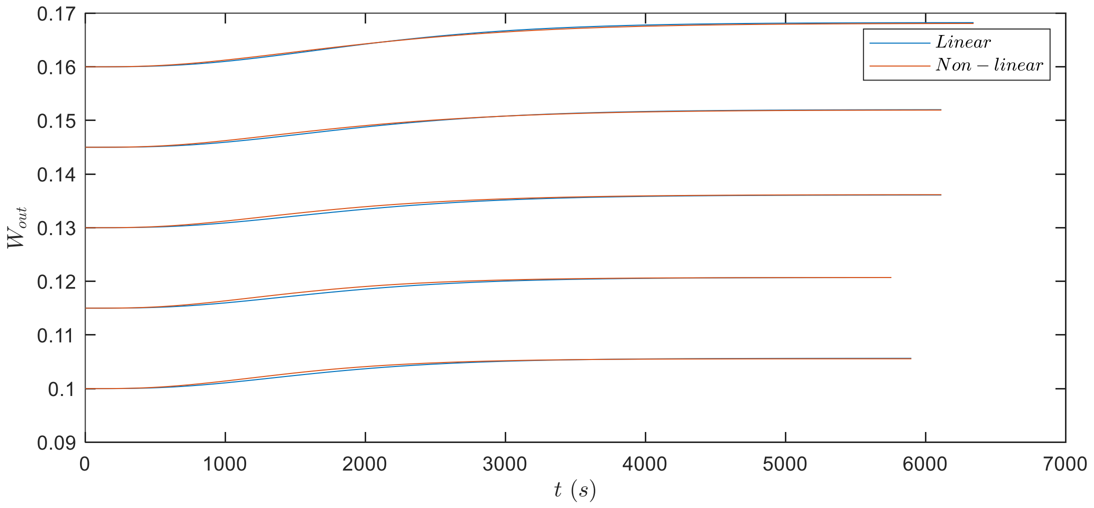


Figure 44. Response to a W_{in} step input of +0.05, W_{in}=0.75

Simulating for initial values of $W_{in} = 0.85$, with a step of +0.025 W_{in} and -0.025 W_{in} (Figure 45 and Figure 46), the non-linearities are even more notable.

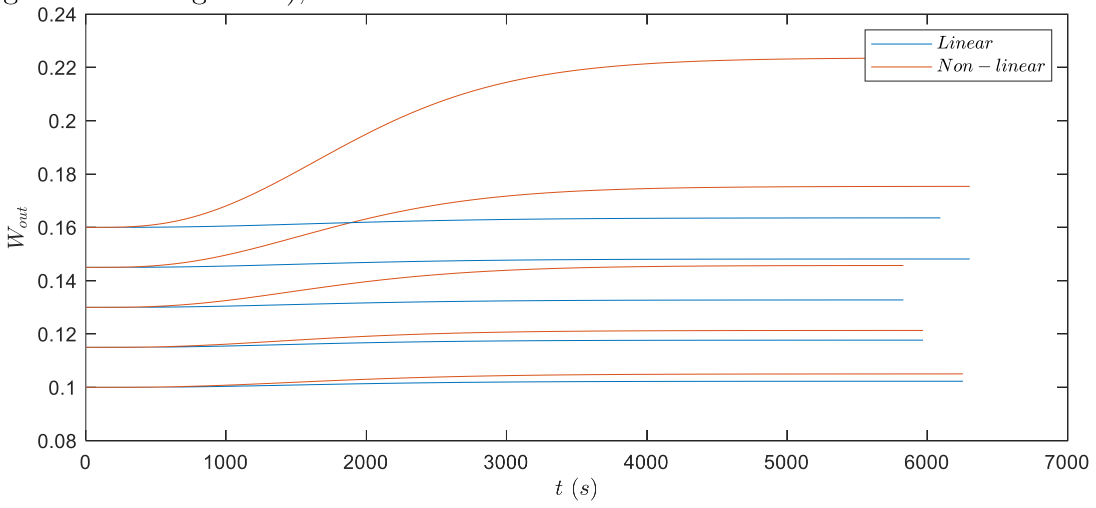


Figure 45. Response to a W_{in} step input of +0.025, W_{in}=0.85

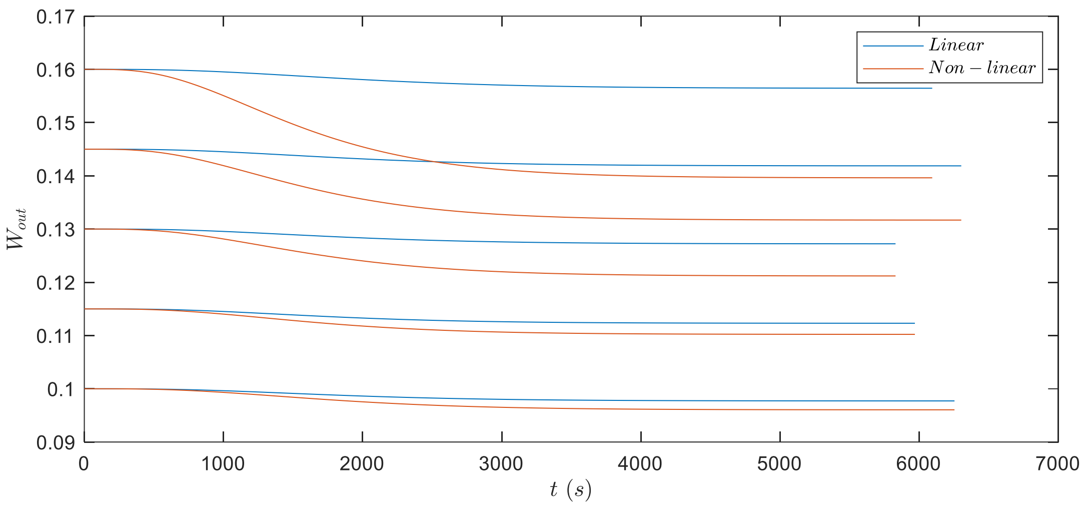


Figure 46. Response to a W_{in} step input of -0.025, W_{in}=0.85

The deviation from the non-linear model suggests that a control strategy where only model inversion was used, without feedback loop correcting the error is not enough.

Now, the step responses of all the plants to a W_{in} step input of +0.1 are represented in the same plot. The linear model ones are represented in Figure 47, while the non-linear ones are represented in Figure 48 and Figure 49.

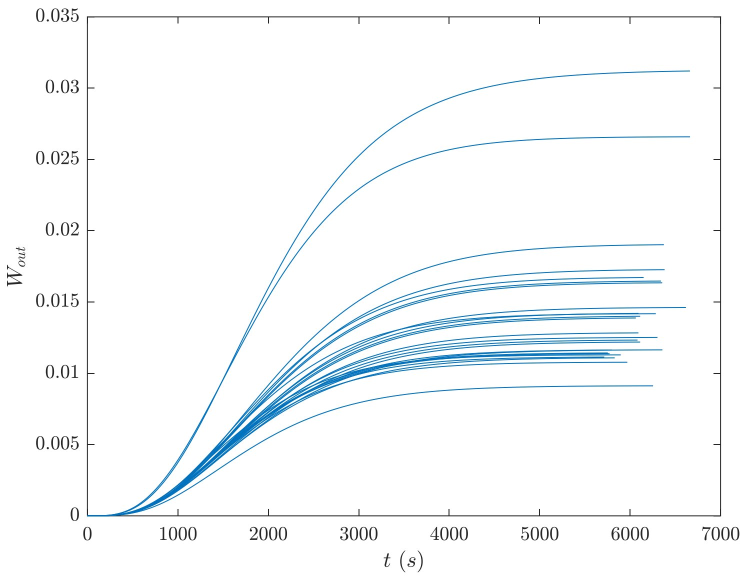


Figure 47. Step responses to a W_{in} step input of +0.1, linear models.

In a similar way as in the case of the $T_{a,in}$ step of 1°C, it is observed that all operating points show similar responses qualitatively, and that their settling time is very similar. Amplitude shows again a significant difference among plants of different operating point conditions. This is true both for the linear and the non-linear models. It is observed that the non-linearities are high for several operating points, as the dryer saturates and can no longer dry the alfalfa. That is the reason why the output moisture content of alfalfa reaches values above the unit, more precisely around W_{out} = 11 (kg of water per kg of alfalfa).

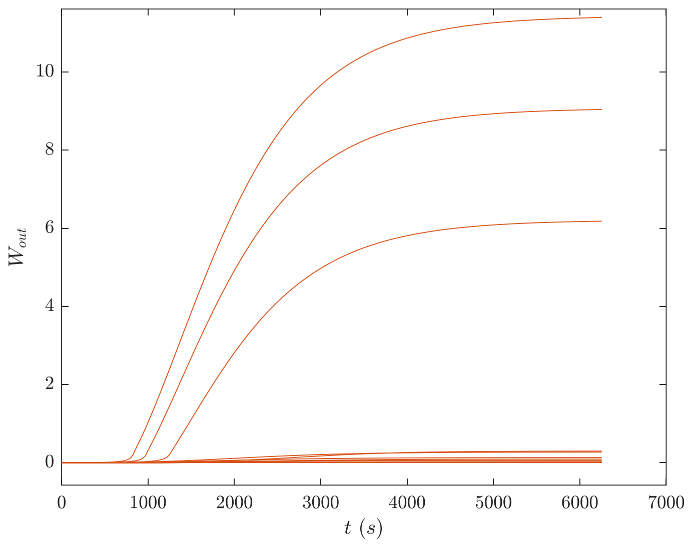


Figure 48. Step responses to a W_{in} step input of +0.1, non-linear models.

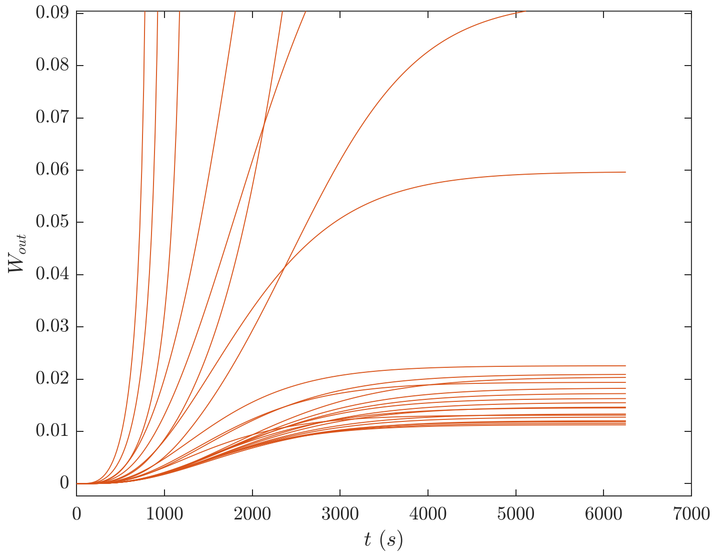


Figure 49. Step responses to a W_{in} step input of +0.1, non-linear models, zoomed in.

4.3.1. Validation final remarks

The linear model accurately predicts the non-linear model behavior for steps of 1°C, and for a step of 3°C the maximum magnitude difference was of 0.5% W_{out}. In the case of step inputs of W_{in}, the non-linearity was appreciated but only was exceptionally high for the operating points where initial $W_{in} = 0.85$. The linear model is then considered to be able to represent the behavior of the system, and it allows the design of controllers using linear methods typical from classic control theory.

5. Conclusion

In this work, a model of an industrial rotary dryer for alfalfa was developed, based on a collection of previous models and equations that describe the different mechanisms involved in drying. Some simplifications were made respective to the dynamics of the drying, and others regarding the operation of the dryer. Then, the model was implemented in the MATLAB/Simulink software. The code for the initialization of the model, as well as the Simulink structures used to model the drying mechanisms were described. The model was implemented with a set of parameters, whose values were selected to match an exemplary industrial rotary dryer. The model can be adapted to alternative dimensions, as well as operating conditions, by slightly changing the parameter values. As a result, simulations can be performed to predict the behaviour of a rotary dryer approximately.

The model was then linearized at a set of operating points, and the state space representation of the linear models was analyzed for an operating point. The newly obtained linear models allow the application of classic control theory, both to further analyze the system and to apply control methods that rely on linear models. The models were validated against the non-linear model, and showed a similar behaviour, only differing notably in a few cases where the high non-linearity of the system was evident. This suggest that closed-loop validation needs to be considered when designing controllers base just on the linear models.

The conclusions that can be extracted from this work are:

- The non-linear model that was developed and the simulator implemented in MATLAB/Simulink allow to predict the behaviour of an industrial rotary dryer of alfalfa.
- The system was confirmed to be highly non-linear due to the effect of the inputs depends highly on the operating point.
- The linear model that was obtained at 25 operating points and later validated against the non-linear model was shown to be capable of correctly predicting the behaviour of the system in the vicinity of the operating points.

5.1. Future work

There are many ways in which this work could be further extended. Firstly, to improve the accuracy of the model, the number of control volumes could be increased, to n=10 or higher, as it is still not very demanding computationally. More importantly, the non-linear model should be validated against a real rotary dryer. This would first allow the model to take parameter values of a real model, which in this work was done by selecting arbitrarily from ranges found in the literature. Then, it would allow to check if the group of equations selected that model the drying mechanisms, and the resulting model, are enough to make accurate predictions of a real dryer.

Regarding the linearization, more inputs could be considered as variable (for example, the input mass flow of alfalfa). This would allow more complex controllers to be developed, turning the two input one output MISO system obtained in this work into a three input or higher model.

6. References

[1] R. O. Ilao *et al.*, *Recent Advances in Biofertilizers and Biofungicides (PGPR) for Sustainable Agriculture*. Newcastle upon Tyne: Cambridge Scholars Publishing, 2014.

[2] E. A. Arinze, G. J. Schoenau, S. Sokhansanj, and P. Adapa, "Aerodynamic Separation and Fractional Drying of Alfalfa Leaves and Stems—A Review and New Concept," *Drying Technology*, vol. 21, no. 9, pp. 1669–1698, Dec. 2003, doi: 10.1081/DRT-120025503.

[3] M. A. Delele, F. Weigler, and J. Mellmann, "Advances in the Application of a Rotary Dryer for Drying of Agricultural Products: A Review," *Drying Technology*, vol. 33, no. 5, pp. 541–558, Apr. 2015, doi: 10.1080/07373937.2014.958498.

[4] F. B. Freire, G. N. A. Vieira, J. T. Freire, and A. S. Mujumdar, "Trends in Modeling and Sensing Approaches for Drying Control," *Drying Technology*, vol. 32, no. 13, pp. 1524–1532, Oct. 2014, doi: 10.1080/07373937.2014.925471.

[5] A. Iguaz, H. Budman, and P. L. Douglas, "MODELLING AND CONTROL OF AN ALFALFA ROTARY DRYER," *Drying Technology*, vol. 20, no. 9, pp. 1869–1887, Jan. 2002, doi: 10.1081/DRT-120015419.

[6] F. Castaño, F. R. Rubio, and M. G. Ortega, "Modelado de Secaderos Rotatorios en Isocorriente," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 6, no. 4, pp. 32–43, Oct. 2009, doi: 10.1016/S1697-7912(09)70106-2.

[7] S. J. Friedman and W. R. Jr. Marshall, "Studies in Rotary Dryer -Part I. Holdup and Dusting," in *Chem. Eng. Progress*, vol. 45(8), (1949a), pp. 482-493.

[8] W. C. Saeman, "Air-solids interaction in rotary dryers and coolers," in *Chemical Engineering Progress*, vol. 58(36), 1962, pp. 49–55.

[9] S. Sokhansanj and R. T. Patil, "Kinetics of dehydration of green alfalfa," in *Drying technology*, vol. 14(5), 1996, pp. 1197–1234.

[10] C. D. Chukwunonye, N. R. Nnaemeka, O. V. Chijioke, and N. C. Obiora, "Thin Layer Drying Modelling for Some Selected Nigerian Produce: A Review".

[11] W. K. Bilanski, J. H. A. Lee, and R. M. Halyk, "HIGH-TEMPERATURE DRYING OF ALFALFA STEMS," *Can. J. Plant Sci.*, vol. 45, no. 5, pp. 471–476, Sep. 1965, doi: 10.4141/cjps65-091.

[12] W. K. Bilanski, J. H. Lee, and R. M. Halyk, "HIGH-TEMPERATURE DRYING OF ALFALFA LEAVES," *CANADIAN AGRICULTURAL ENGINEERING*, 1965.

[13] R. T. Patil, S. Sokhansanj, E. A. Arinze, and G. Schoenau, "Thin layer drying of components of fresh," vol. 34, no. 4, 1992.

[14] M. J. Moran and H. N. Shapiro, "Appendix Tables, Figures, and Charts," in *Fundamentals of Engineering Thermodynamics*, 5th ed., 2006, pp. 718–765.

[15] N. N. Mohsenin, "Specific heat capacity of ground alfalfa," in *Thermal Properties of Food and Agricultural Materials*, 1980, pp. 49–50.

[16] A. S. Mujumdar, Ed., *Handbook of industrial drying*, Second edition, Revised and Expanded. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2019.

[17] A. S. Mujumdar, "CLASSIFICATION AND SELECTION OF INDUSTRIAL DRYERS," *Mujumdar's Practical Guide to Industrial Drying: Principles, Equipment and New Developments*, pp. 23–36, 2000.

[18] Qingdao palet machinery co,.ltd, "Rotary dryers specification and parameter." [Online]. Available: https://www.biopelletmachines.com/rotary-dryer/

ANEXO A: Drying rates

```
k_chop = zeros(5,101);
k_2007 = zeros(5,101);
k_2007_kelvin = zeros(5,101);

w_values = (0.45:0.1:0.85);
t_values = (200:5:700);

for i=1:5
    for j=1:101
w = w_values(i);
t = t_values(j);

k_chop(i,j) =  0.0176*exp(-0.123*w+0.044*t-5.79*10^-5*t^2-3.24*10^-3*w*t)/60;
k_2007(i,j) =  5.71*10^4*exp(-73.99/(t));
k_2007_kelvin(i,j) =  5.71*10^4*exp(-73.99/(273+t));

k_Bilanski_stem(i,j) = (0.051-3.43*10^24*exp(-t));
k_Bilanski_leaf(i,j) = (0.064-2.17*10^24*exp(-t));

    end
end
%
% figure
% for j=1:81
% plot(w_values,k_chop(:,j))
% hold on
% end

figure
for i=1:5
plot(t_values,k_chop(i,:))
hold on
end
legend('$ W_{in} = 0.45$','$ W_{in} = 0.55$','$ W_{in} = 0.65$', ...
    '$ W_{in} = 0.75$','$ W_{in} = 0.85$')
xlabel('$ T (^{\circ}C)$','Interpreter','latex')
ylabel('${\it} K (min^{-1})$','Interpreter','latex')


figure
plot(t_values,k_2007(1,:))
hold on
% plot(t_values,k_2007_kelvin(1,:))
% legend('$ T \ in \ ^{\circ}C$')
xlabel('$ T (^{\circ}C)$','Interpreter','latex')
ylabel('${\it} K (min^{-1})$','Interpreter','latex')
k_Bilanski_stem = zeros(5,128);


k_Bilanski_leaf = zeros(5,128);


t_values = (65:5:700);


for i=1:5
    for j=1:128
w = w_values(i);
t = t_values(j);

k_Bilanski_stem(i,j) = (0.051-3.43*10^24*exp(-t));
k_Bilanski_leaf(i,j) = (0.064-2.17*10^24*exp(-t));

    end
end
```

```
figure
plot(t_values,k_Bilanski_leaf(1,:))
hold on
plot(t_values,k_Bilanski_stem(1,:))
% plot(t_values,k_2007_kelvin(1,:))
legend('$ Stem$','$ Leaf$')
xlabel('$ T (^{\circ}C)$','Interpreter','latex')
ylabel('${\it} K (min^{-1})$','Interpreter','latex')
%
% [W,T] = meshgrid(w_values,t_values);
% figure
% surf(W,T,k_chop);

% zlim([0 0.0001])
% figure
% surf(W,T,K_stem)
% grid on
% zlim([0 0.001])
% figure
% surf(W,T,K_leaf)
% grid on
% zlim([0 0.001])
```

ANEXO B: BET_curves

```
a_w = (0:0.0001:1);

Temp = (50:50:350);

Moisture = zeros(size(a_w,2),size(Temp,2));



figure

for j=1:size(Temp,2)
    for i=1:size(a_w,2)

        X_m = 3.9229*10^-4*exp(1858.8/(Temp(j)+273.16));
        C = 323.1769*exp(-974.55/(Temp(j)+273.16));
        Moisture(i,j) = X_m*C*a_w(i)/((1-a_w(i))*(1+(C-1)*a_w(i)));

    end
    plot (a_w(1,:),Moisture(:,j));
    hold on
end

ylim([0 1])

legend('$ T  =  50^{\circ}C$', ...
    '$ T  =  100^{\circ}C$', ...
    '$ T  =  150^{\circ}C$', ...
    '$ T  =  200^{\circ}C$', ...
    '$ T  =  250^{\circ}C$', ...
    '$ T  =  300^{\circ}C$', ...
    '$ T  =  350^{\circ}C$')

xlabel('$ a_{w}$','Interpreter','latex')
ylabel('$ Alfalfa \ moisture \ content \ (d.b.)$','Interpreter','latex')
```

ANEXO C: Startup

```
opengl software
load('sys_lin')
load('OP_temperatures');
load('OP_states');
load('Controller_22');
load('G_fbk3');
load('G_ff3');
load('G_fbk4');
load('G_ff4');
load('W_dist3');
load('W_dist8');

W_ref_step = 0.01; % Step de la referencia Win

W_ref_track_switch = 0; % 1=ON 0=OFF tracking (dist rej o white noise)
W_dist_noise_switch = 0; % 1:white noise 0:step
W_dist_step = 0.1;

Sample_time_sim = 100;
Sample_time_signal = 100;
W_dist_step_time = 10000;
stop_time = 60*350;
```

Parametros

```
% # de bloques
n = 4;

% Constantes
R = 8.31; % Constante de los gases ideales [kJ / (K kmol)]
M_molar_agua = 18; % [kg/kmol]

% Parameters
D = 3.3;            %Effective dryer diameter (m)
L = 20;             %Dryer length (m)
V = 171.06;         %Volume (m^3)
alpha = 0.005;      %Dryer slope (degrees)
N = 10;             %Dryer rotational speed (rpm)
A = pi/4*D^2;       %Cross sectional area of the dryer (m2)
V_a = 3;            %Air axial velocity (m/s)
Gain = 34.2*0.5;    % Flujo masico aire
Gpin = 5.7;         % Flujo masico producto [kg/s]
Win = 0.85;              %kg agua / kg producto seco
Yin = 0.02;         %kg agua / kg aire seco (deberia ser 0.02-0.03)
Tpin = 15;          %Temperatura de alfalfa de entrada, valor inicial -
Arbitrario [°C]
Tain = 469;         %Temperatura de aire a la entrada - 200-800°C [°C]

% Calculo de masa acumulada (agua y alfalfa)

t_r_OG_Saeman = L/(pi*D*N*(tand(alpha)+0.0026*V_a));

M_holdup_Saeman = t_r_OG_Saeman*Gpin*60/n; % [min] * [kg/s] * [s/min] = [kg]

M_holdup_alfalfa_y_agua = M_holdup_Saeman;
M_holdup_agua = (Win/(1+Win))*M_holdup_alfalfa_y_agua; % [kg]
M_air = Gain*L/V_a;
M_holdup_alfalfa = M_holdup_alfalfa_y_agua-M_holdup_agua;
tr = M_holdup_alfalfa/Gpin;

F = 34.2;           %Inlet air flowrate (m^3/s)

h = 0.28;           %Heat loss coefficient (kJ/m^2*s*°C)
```

```
T_ambiente = 20;    %Temperatura ambiente ?hacer igual a T_pin_0?
T_media = (160+Tpin)/2;
Q_p_total = h*pi*D*L*(T_media-T_ambiente);   %Heat loss through dryer shell
(J/s)
Q_p = Q_p_total/n;  %Heat loss through dryer shell (J/s)

T_inc = 0;
T_Step_switch = 0;
Win_inc = 0;
W_Step_switch = 0;

Tain_f = Tain+T_inc;
Win_f = Win+Win_inc;

Win_values = [0.85 0.75 0.65 0.55 0.45]; % Valores de humedad de producto de
entrada (b.s.)
Wout_values = [0.1 0.115 0.13 0.145 0.16]; % Valores de humedad de producto de
entrada (b.s.)
```

ANEXO D: W_out_recollection

```
Win_values = [0.85,0.75,0.65,0.55,0.45]; % Valores de humedad de producto de
entrada (b.s.)
Tain_values = (460:5:520); % Valores de temperatura de aire de entrada

W_out = zeros(5,13,1);
Sample_time_signal = 100;
opengl software

T_inc = 0;
T_Step_switch = 0;
Win_inc = 0;
W_Step_switch = 0;

mdl = 'modelo_no_lineal';
OP_state_val = zeros(5,13,20);
```

W_out recollection

```
T_fixed_switch=1;




for i=1:5
    for j=1:13
Win = Win_values(i);
Tain = Tain_values(j);
stop_time = 60*400;

Simout = sim(mdl); % Lanzar la simulacion del modelo
length = size(Simout.x1); % Obtener longitud vector de datos de salida para
luego coger el valor final (asegurandose de que se haya estabilizado)

% Recoger valores de variables de estado en los ptos de operacion en un
% struct
OP_state_val(i,j,:) = [Simout.x1(length(1));
                    Simout.x2(length(1));
                    Simout.x3(length(1));
                    Simout.x4(length(1));
                    Simout.x5(length(1));
                    Simout.x6(length(1));
                    Simout.x7(length(1));
                    Simout.x8(length(1));
                    Simout.x9(length(1));
                    Simout.x10(length(1));
                    Simout.x11(length(1));
                    Simout.x12(length(1));
                    Simout.x13(length(1));
                    Simout.x14(length(1));
                    Simout.x15(length(1));
                    Simout.x16(length(1));
                    Simout.x17(length(1));
                    Simout.x18(length(1));
                    Simout.x19(length(1));
                    Simout.x20(length(1))];

W_out(i,j,:) = Simout.W_out(length(1));

Message = ['Simulation ',num2str(i),',',num2str(j),' done.'];
disp(Message);

t = [0:Sample_time_sim:Sample_time_sim*size(Wref_signal)-1];
```

```
    end
end

save('W_out','W_out');
```

Steady-state operating ponint states

```
opspec = operspec(mdl)




OP_states_not_found = zeros(5,13,20);


OP_states_saved = zeros(5,13,20);


Op_not_found = [0,0];




T_fixed_switch=1;




% Bucle para pasar por todos los casos de Win y Tain
for i=1:5
    for j=1:13

% Establecer parametros
Win = Win_values(i);
Tain = Tain_values(j);

opspec.States(1).x = OP_state_val(i,j,1);
opspec.States(1).x = OP_state_val(i,j,1);
opspec.States(2).x = OP_state_val(i,j,2);
opspec.States(3).x = OP_state_val(i,j,3);
opspec.States(4).x = OP_state_val(i,j,4);
opspec.States(5).x = OP_state_val(i,j,5);
opspec.States(6).x = OP_state_val(i,j,6);
opspec.States(7).x = OP_state_val(i,j,7);
opspec.States(8).x = OP_state_val(i,j,8);
opspec.States(9).x = OP_state_val(i,j,9);
opspec.States(10).x = OP_state_val(i,j,10);
opspec.States(11).x = OP_state_val(i,j,11);
opspec.States(12).x = OP_state_val(i,j,12);
opspec.States(13).x = OP_state_val(i,j,13);
opspec.States(14).x = OP_state_val(i,j,14);
opspec.States(15).x = OP_state_val(i,j,15);
opspec.States(16).x = OP_state_val(i,j,16);
opspec.States(17).x = OP_state_val(i,j,17);
opspec.States(18).x = OP_state_val(i,j,18);
opspec.States(19).x = OP_state_val(i,j,19);
opspec.States(20).x = OP_state_val(i,j,20);

[op,opreport] = findop(mdl,opspec);

% Check if operating point successfully met
if size(opreport.TerminationString)==[1,53]
    OP_states_saved(i,j,:) = [op.States(1, 1).x;
                        op.States(2, 1).x  ;
                        op.States(3, 1).x  ;
                        op.States(4, 1).x  ;
                        op.States(5, 1).x  ;
                        op.States(6, 1).x  ;
                        op.States(7, 1).x  ;
                        op.States(8, 1).x  ;
                        op.States(9, 1).x  ;
                        op.States(10, 1).x  ;
                        op.States(11, 1).x  ;
                        op.States(12, 1).x  ;
                        op.States(13, 1).x  ;
                        op.States(14, 1).x  ;
                        op.States(15, 1).x  ;
                        op.States(16, 1).x  ;
                        op.States(17, 1).x  ;
                        op.States(18, 1).x  ;
                        op.States(19, 1).x  ;
                        op.States(20, 1).x ] ;
else
    Op_not_found( end+1,  :) = [i;j];
    OP_states_not_found(i,j,:) = [op.States(1, 1).x;
                        op.States(2, 1).x  ;
                        op.States(3, 1).x  ;
                        op.States(4, 1).x  ;
                        op.States(5, 1).x  ;
                        op.States(6, 1).x  ;
                        op.States(7, 1).x  ;
                        op.States(8, 1).x  ;
                        op.States(9, 1).x  ;
                        op.States(10, 1).x  ;
                        op.States(11, 1).x  ;
                        op.States(12, 1).x  ;
                        op.States(13, 1).x  ;
                        op.States(14, 1).x  ;
                        op.States(15, 1).x  ;
                        op.States(16, 1).x  ;
                        op.States(17, 1).x  ;
                        op.States(18, 1).x  ;
                        op.States(19, 1).x  ;
                        op.States(20, 1).x ] ;
end
Message = ['Op search ',num2str(i),',',num2str(j),' done.'];
disp(Message);
    end
end

save('OP_state_approach','OP_states_saved')
```

W_out recollection, at steady-state

```
for i=1:5
    for j=1:13
Win = Win_values(i);
Tain = Tain_values(j);

% Poner estados al valor del pto de operacion
State_1 = OP_states_saved(i,j,1);
State_2 = OP_states_saved(i,j,2);
State_3 = OP_states_saved(i,j,3);
State_4 = OP_states_saved(i,j,4);
State_5 = OP_states_saved(i,j,5);
State_6 = OP_states_saved(i,j,6);
State_7 = OP_states_saved(i,j,7);
State_8 = OP_states_saved(i,j,8);
State_9 = OP_states_saved(i,j,9);
State_10 = OP_states_saved(i,j,10);
State_11 = OP_states_saved(i,j,11);
State_12 = OP_states_saved(i,j,12);
State_13 = OP_states_saved(i,j,13);
```

```
State_14 = OP_states_saved(i,j,14);
State_15 = OP_states_saved(i,j,15);
State_16 = OP_states_saved(i,j,16);
State_17 = OP_states_saved(i,j,17);
State_18 = OP_states_saved(i,j,18);
State_19 = OP_states_saved(i,j,19);
State_20 = OP_states_saved(i,j,20);

Simout = sim(mdl); % Lanzar la simulacion del modelo
length = size(Simout.x1); % Obtener longitud vector de datos de salida para
luego coger el valor final (asegurandose de que se haya estabilizado)


W_out(i,j,:) = Simout.W_out(length(1));

Message = ['Simulation ',num2str(i),',',num2str(j),' done.'];
disp(Message);
    end
end

save('W_out','W_out');
```

ANEXO E: Linearization

```
Stability_grid = zeros(5,5);

step_time=0;



Win_values = [0.85 0.7 0.65 0.55 0.45]; % Valores de humedad de producto de
entrada (b.s.)
Wout_values = [0.1 0.115 0.13 0.145 0.16]; % Valores de humedad de producto de
entrada (b.s.)
Sample_time_signal = 100;
opengl software
load('OP_state_approach.mat');
Proxy_states = zeros(5,5,2);
Proxy_states(:,:,1)= [1,1,1,1,1;
                      2,2,2,2,2;
                      3,3,3,3,3;
                      4,4,4,4,4;
                      5,5,5,5,5];

Proxy_states(:,:,2)= [13,11,7,5,5;
                      12,4,5,4,3;
                       5,4,4,4,3;
                       4,4,4,3,3;
                       5,5,3,4,3];

T_inc = 0;
T_Step_switch = 0;
Win_inc = 0;
W_Step_switch = 0;

mdl = 'modelo_no_lineal';
OP_state_val = zeros(5,5,20);

OP_states_not_found = zeros(5,5,20);
OP_states = zeros(5,5,20);
Op_not_found = [0,0];
```

Steady-state operating ponint states and temperatures

```
T_fixed_switch=0;

W_fixed_switch=0;

% Bucle para pasar por todos los casos de Win y Tain
for i=1:5
    for j=1:5

% Establecer parametros
Win = Win_values(i);

opspec = operspec(mdl);
newspec = addoutputspec(opspec,'modelo_no_lineal/Subsystem3',1);

ii = Proxy_states(i,j,1);
jj = Proxy_states(i,j,2);

newspec.States(1).x = OP_states_saved(ii,jj,1);
newspec.States(1).x = OP_states_saved(ii,jj,1);
newspec.States(2).x = OP_states_saved(ii,jj,2);
newspec.States(3).x = OP_states_saved(ii,jj,3);
newspec.States(4).x = OP_states_saved(ii,jj,4);
newspec.States(5).x = OP_states_saved(ii,jj,5);
newspec.States(6).x = OP_states_saved(ii,jj,6);
newspec.States(7).x = OP_states_saved(ii,jj,7);
newspec.States(8).x = OP_states_saved(ii,jj,8);
newspec.States(9).x = OP_states_saved(ii,jj,9);
newspec.States(10).x = OP_states_saved(ii,jj,10);
newspec.States(11).x = OP_states_saved(ii,jj,11);
newspec.States(12).x = OP_states_saved(ii,jj,12);
newspec.States(13).x = OP_states_saved(ii,jj,13);
newspec.States(14).x = OP_states_saved(ii,jj,14);
newspec.States(15).x = OP_states_saved(ii,jj,15);
newspec.States(16).x = OP_states_saved(ii,jj,16);
newspec.States(17).x = OP_states_saved(ii,jj,17);
newspec.States(18).x = OP_states_saved(ii,jj,18);
newspec.States(19).x = OP_states_saved(ii,jj,19);
newspec.States(20).x = OP_states_saved(ii,jj,20);

if ((i==1)&&(j==2)||(i==2)&&(j==1))
    newspec.Inputs(1).Min = 440 ; %440
    newspec.Inputs(1).Max = 500; %600
    newspec.Inputs(1).u = 465.2; %450
else
    if (i==5)
        if (j==2)
            newspec.Inputs(1).Min = 467.5 ; %440
            newspec.Inputs(1).Max = 468.4; %600
            newspec.Inputs(1).u = 468.2; %450
        else
            if (j==3)
                newspec.Inputs(1).Min = 466.5 ; %440
                newspec.Inputs(1).Max = 467.4; %600
                newspec.Inputs(1).u = 467.2; %450
            else
                if (j==4)
                    newspec.Inputs(1).Min = 465.5 ; %440
                    newspec.Inputs(1).Max = 466.4; %600
                    newspec.Inputs(1).u = 466.2; %450
                else
                    if (j==5)
                        newspec.Inputs(1).Min = 463.5 ; %440
                        newspec.Inputs(1).Max = 465.4; %600
                        newspec.Inputs(1).u = 466.2; %450
                    else
                        newspec.Inputs(1).Min = 440 ; %440
                        newspec.Inputs(1).Max = 500; %600
                        newspec.Inputs(1).u = 465.2; %450
                    end
                end
            end
        end
    else
        newspec.Inputs(1).Min = 440 ; %440
        newspec.Inputs(1).Max = 505; %600
        newspec.Inputs(1).u = 465.2; %450
    end
end

    newspec.Inputs(2).Known = 1;
    newspec.Inputs(2).u = Win_values(i);

newspec.Outputs(1).Known = 1;
newspec.Outputs(1).y = Wout_values(j);

[op,opreport] = findop(mdl,newspec);

% Check if operating point successfully met
```

```
if size(opreport.TerminationString)==[1,53]
    OP_states(i,j,:) = [op.States(1, 1).x;
                        op.States(2, 1).x  ;
                        op.States(3, 1).x  ;
                        op.States(4, 1).x  ;
                        op.States(5, 1).x  ;
                        op.States(6, 1).x  ;
                        op.States(7, 1).x  ;
                        op.States(8, 1).x  ;
                        op.States(9, 1).x  ;
                        op.States(10, 1).x  ;
                        op.States(11, 1).x  ;
                        op.States(12, 1).x  ;
                        op.States(13, 1).x  ;
                        op.States(14, 1).x  ;
                        op.States(15, 1).x  ;
                        op.States(16, 1).x  ;
                        op.States(17, 1).x  ;
                        op.States(18, 1).x  ;
                        op.States(19, 1).x  ;
                        op.States(20, 1).x ] ;
    OP_temperatures_saved(i,j) = op.Inputs.u;
else
    Op_not_found( end+1,  :) = [i;j];
    OP_states_not_found(i,j,:) = [op.States(1, 1).x;
                        op.States(2, 1).x  ;
                        op.States(3, 1).x  ;
                        op.States(4, 1).x  ;
                        op.States(5, 1).x  ;
                        op.States(6, 1).x  ;
                        op.States(7, 1).x  ;
                        op.States(8, 1).x  ;
                        op.States(9, 1).x  ;
                        op.States(10, 1).x  ;
                        op.States(11, 1).x  ;
                        op.States(12, 1).x  ;
                        op.States(13, 1).x  ;
                        op.States(14, 1).x  ;
                        op.States(15, 1).x  ;
                        op.States(16, 1).x  ;
                        op.States(17, 1).x  ;
                        op.States(18, 1).x  ;
                        op.States(19, 1).x  ;
                        op.States(20, 1).x ] ;
end
Message = ['Op search ',num2str(i),',',num2str(j),' done.'];
disp(Message);
    end
end
save('OP_states',"OP_states")
save('OP_temperatures_saved','OP_temperatures_saved')
```

Linearize

```
for i=1:5
    for j=1:5

Win = Win_values(i);
Tain = OP_temperatures_saved(i,j);

io = getlinio(mdl)
op = operpoint(mdl)

op.States(1).x = OP_states(i,j,1);
op.States(1).x = OP_states(i,j,1);
op.States(3).x = OP_states(i,j,3);
op.States(4).x = OP_states(i,j,4);
op.States(5).x = OP_states(i,j,5);
op.States(6).x = OP_states(i,j,6);
op.States(8).x = OP_states(i,j,8);
op.States(9).x = OP_states(i,j,9);
op.States(10).x = OP_states(i,j,10);
op.States(11).x = OP_states(i,j,11);
op.States(13).x = OP_states(i,j,13);
op.States(15).x = OP_states(i,j,15);
op.States(16).x = OP_states(i,j,16);
op.States(17).x = OP_states(i,j,17);
op.States(19).x = OP_states(i,j,19);
op.States(20).x = OP_states(i,j,20);

sys_provisional(:,:,(i-1)*5+j) = linearize(mdl,io,op);

if isstable(sys_provisional(:,:,(i-1)*5+j))==1
    sys(:,:,(i-1)*5+j)=sys_provisional(:,:,(i-1)*5+j);
end

Message = ['Model ',num2str(i),',',num2str(j),' linearized.'];
disp(Message);
    end
end

save('sys_saved_final_V','sys')
```

ANEXO F: Validation

```
Win_values = [0.85 0.75 0.65 0.55 0.45]; % Valores de humedad de producto de
entrada (b.s.)
Wout_values = [0.1 0.115 0.13 0.145 0.16]; % Valores de humedad de producto de
entrada (b.s.)
step_time = 0;
Sample_time_signal = 100;
opengl software
load('sys_lin.mat')
load('OP_states_new.mat');
load('OP_temperatures_new.mat');
```

Validation linear vs non linear

```
T_fixed_switch=1;

figure

for i=1:1 % 1:1 2:2 3:3 4:4 5:5 (fix Win = 0.85 0.75 0.65 0.55 0.45)
    for j=1:5

T_inc = 3;          % Set step T magnitude (°C)
T_Step_switch = 1;  % 1=ON 0=OFF
Win_inc = -0.025;   % Set step W magnitude (kg/kg)
W_Step_switch = 0;  % 1=ON 0=OFF

W_out_sp = Wout_values(j);

[W_out_inc,t]=step(sys(:,'Tain',i*5-5+j));  % T step
% [W_out_inc,t]=step(sys(:,'Win',i*5-5+j)); % W step

mdl = 'modelo_no_lineal';

t_final = size(t);
stop_time = t(t_final(1));
Sample_time_signal = t(2);
Tain = OP_temperatures_saved(i,j);
Tain_f = Tain+T_inc;
Win = Win_values(i);
Win_f = Win+Win_inc;

% Poner estados al valor del pto de operacion
State_1 = OP_states(i,j,1);
State_2 = OP_states(i,j,2);
State_3 = OP_states(i,j,3);
State_4 = OP_states(i,j,4);
State_5 = OP_states(i,j,5);
State_6 = OP_states(i,j,6);
State_7 = OP_states(i,j,7);
State_8 = OP_states(i,j,8);
State_9 = OP_states(i,j,9);
State_10 = OP_states(i,j,10);
State_11 = OP_states(i,j,11);
State_12 = OP_states(i,j,12);
State_13 = OP_states(i,j,13);
State_14 = OP_states(i,j,14);
State_15 = OP_states(i,j,15);
State_16 = OP_states(i,j,16);
State_17 = OP_states(i,j,17);
State_18 = OP_states(i,j,18);
State_19 = OP_states(i,j,19);
State_20 = OP_states(i,j,20);
```

```
Simout = sim(mdl); % Lanzar la simulacion del modelo
W_out_nl = Simout.W_out_signal;

% Comparacion lineal vs no lineal
% plot(t,W_out_sp+Win_inc*W_out_inc(:,1,1),'Color','#0072BD');  % W step
plot(t,W_out_sp+T_inc*W_out_inc(:,1,1),'Color','#0072BD');  % T step
hold on
plot(t,W_out_nl,'Color','#D95319'); % Non linear

    end
end

xlabel('$ t \ (s)$','Interpreter','latex')
ylabel('$ W_{out}$','Interpreter','latex')
legend('$ Linear$','$ Non-linear$','Interpreter','latex')
```

Validation all steps same model

```
figure




for i=1:5
    for j=1:5

T_inc = 1;           % Set step T magnitude (°C)
T_Step_switch = 1;   % 1=ON 0=OFF
Win_inc = 0.1;    % Set step W magnitude (kg/kg)
W_Step_switch = 0;   % 1=ON 0=OFF

W_out_sp = Wout_values(j);
% [W_out_inc,t]=step(sys(:,'Tain',i*5-5+j)); % T step
[W_out_inc,t]=step(sys(:,'Win',i*5-5+j));    % W step

t_final = size(t);
stop_time = t(t_final(1));
Sample_time_signal = t(2);
Tain = OP_temperatures_saved(i,j);
Tain_f = Tain+T_inc;
Win = Win_values(i);
Win_f = Win+Win_inc;

% Poner estados al valor del pto de operacion
State_1 = OP_states(i,j,1);
State_2 = OP_states(i,j,2);
State_3 = OP_states(i,j,3);
State_4 = OP_states(i,j,4);
State_5 = OP_states(i,j,5);
State_6 = OP_states(i,j,6);
State_7 = OP_states(i,j,7);
State_8 = OP_states(i,j,8);
State_9 = OP_states(i,j,9);
State_10 = OP_states(i,j,10);
State_11 = OP_states(i,j,11);
State_12 = OP_states(i,j,12);
State_13 = OP_states(i,j,13);
State_14 = OP_states(i,j,14);
State_15 = OP_states(i,j,15);
State_16 = OP_states(i,j,16);
State_17 = OP_states(i,j,17);
State_18 = OP_states(i,j,18);
State_19 = OP_states(i,j,19);
State_20 = OP_states(i,j,20);

Simout = sim('modelo_no_lineal'); % Lanzar la simulacion del modelo
```

```
W_out_nl = Simout.W_out_signal;


% plot(t,T_inc*W_out_inc(:,1,1),'Color','#0072BD'); % T step
plot(t,Win_inc*W_out_inc(:,1,1),'Color','#0072BD'); % W step
hold on
% plot(t,W_out_nl-W_out_sp,'Color','#D95319'); % Non linear
% hold on

    end
end

xlabel('$ t \ (s)$','Interpreter','latex')
ylabel('$ W_{out}$','Interpreter','latex')
```