International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 12 (2019), pp. 2143-2158
© International Research Publication House. http://www.irphouse.com

Nonlinear Dynamic Simulation of a Magnetic Levitation System Using MATLAB/Simulink

Ayokunle A. Awelewa[1]; Olawale Popoola[2]; Abdulkareem Ademola[3]

[1]*Postdoctoral Fellow, Department of Electrical Engineering, Tshwane University of Technology, South Africa.*

[2]*Director, CEEP, epartment of Electrical Engineering, Tshwane University of Technology, South Africa.*

[1,3]*Lecturers, Department of Electrical and Information Engineering, Covenant University, Nigeria.*

ORCID: 0000-0002-4409-6628 (Ayokunle)

Abstract

Nonlinear control has witnessed a resurgence of interest due to the availability of fast, powerful, low-cost computer resources and advanced electronics technology, thereby spawning analysis and design based on extensive computer simulations and system prototypes. This study considers and treats dynamic simulation of a nonlinear magnetic levitation system under the control action of three nonlinear control laws, which are constructed using the concept of state feedback linearization. Whereas the first law is a direct state feedback controller, the other two are based on state homogeneity control. Of particular interest is the third controller, which is an improved and modified version of the direct state homogeneity-based controller. To reveal the ramified dynamics of the system, its performance with respect to two output functions is assessed during normal and abnormal operating conditions. New general MATLAB functions are provided to aid the study, and further serve as analysis tools to support subsequent research in related areas.

Keywords: Control action, controllable normal form, output functions, relative degree, state feedback, transformation function

I. INTRODUCTION

The theory of linear and nonlinear control systems has received different attention in the control engineering community. Linear control design has been effectively and numerously applied to formulate controllers for simple and large systems [1, 2], because it offers both a wide range of well established tools that can be systematically put to use for controller realization and a simple and generally applicable procedure founded on the well known indirect method of Lyapunov. The limited operational range of this design approach notwithstanding, a very useful control algorithm based on this linear design is the PID controller [3, 4], which has gained universal acceptance owing to the amenability of its simple structure to tuning for various operating conditions. Conversely, in the past, nonlinear control theory witnessed slow progress, because of its limited analysis tools and lack of a generally applicable design principle [5], although, due to their different characteristics, many nonlinear systems lend themselves more to specific analysis and design principles than others. But the tempo has switched to higher gear, as the continuous development in computer architecture and low-cost microcontroller technology has encouraged further and focused nonlinear control research—resulting in a great deal of simulations of complex and nonlinear systems as well as the use of nonlinear control algorithms in real-life systems [6, 7].

Many practical and experimental systems—such as inverted pendulums, ball-on-a-beam systems, magnetic levitation systems, etc.—have attracted attention in the literature and been used in prototypical forms in engineering departments at various universities and research institutions globally, because they provide good frameworks and ingredients for analyzing and testing control algorithms. They have been employed over the years as platforms [8, 9] to educate and train students and lab personnel alike in control and control-related subjects. But of particular focus in this paper is a magnetic levitation system, which is one of the interesting test beds for control laws. In the literature, various linear controllers [10, 11], particularly PID controllers and their variants, have been developed to control the dynamic characteristics of a magnetic levitation system, which happens to be inherently unstable. These controllers used linearized models of the system, and they performed acceptably well. Nevertheless, the restriction of the operating ranges of linearized system models to very small neighbourhoods in state space constrains the performance of many controllers built on these models. Similarly, other controllers based on sliding mode and intelligent techniques available [12, 13, 14] for controlling magnetic levitation systems.

The focal point of this study is to perform simulations of a nonlinear magnetic levitation system based on compact nonlinear state feedback controllers developed to stabilize and improve its dynamic performance. MATLAB/Simulink is used greatly to aid the study. The rest of the paper is organized as follows. Section 2 lays the foundation for the system design and implementation, providing information on materials and system representation. In Section 3 details about the construction and implementation of the control laws are highlighted. This section is central to the study, as it offers the overall framework, simulation tools, and description of the controller design. The results of the simulation are presented in Section 4, and Section 5 gives a concluding note.

II. MATERIALS AND SYSTEM REPRESENTATION

II.I Background to Simulink

Simulink is an integral part of MATLAB, which is a Mathworks product for scientific and engineering computation with a very broad of area of applications. Simulink is a powerful and robust programming environment in which representative models and functional characteristics of systems are created, tested, verified, and evaluated by interconnecting blocks to form correct input-output relations. Since it is a graphical environment for model-based simulation and design, it is very useful for investigating and/or compensating for behavioral characteristics of dynamic systems.

There are many blocks available in the Simulink development environment, with each of these as well as similar others grouped under an appropriate library [15]. For instance, "Sources" library contains blocks for creating input signals of diverse shapes and forms; "Sinks" library contains blocks for outputting/displaying/visualizing simulation results; "Commonly Used Blocks" library comprises blocks that are used to perform common tasks such as addition, multiplication, integration, multiplexing/de-multiplexing of signals, etc.; "Continuous" library has blocks for carrying out continuous-time operations. Other libraries are "Discrete", "Signal Routing", "Math Operation", "Discontinuities", etc. There are customized blocks, which are specifically designed and prepackaged to solve problems in specific disciplines or subject areas. Few examples are Simulink Control Design, Simscape, Simulink Design Optimization, Simulink 3D animation, Simulink Extras, Aerospace Blockset, etc. Models are created by simply moving blocks from the relevant libraries in the library browser into the development environment.

Again, since MATLAB and Simulink are integrated, information can be readily transferred between them. For example, the use of "To Workspace" block from the Sinks library, "Save data to Workspace" box in the scope parameter window, and "Data Import/Export" tool in the configuration parameter window are possible ways [15] by which MATLAB/Simulink information exchange can be realized. Even code written in other platforms, such as C and C^{++}, can be imported into Simulink.

General Steps for Constructing System Models in Simulink

The following are the steps required for developing a Simulink-based system model.

i) Obtain the physical description/representation of the system. This must be as sufficient as possible in order to capture the required system behaviour to be investigated. Although the purpose of the model will influence to a great measure the degree of detail of system representation, it is notable that the results generated by Simulink cannot "precede" the level of accuracy included in the model.

ii) Open the Simulink library browser. Type the word "simulink" (all in lower case) in the MATLAB command prompt and wait for a while for the library browser window to open.

iii) Click on the "New Model" button on the library browser toolbar, and a new model window (also called the Simulink Editor) will be created. Save this window with an appropriate name.

iv) Drag all necessary blocks, including Scope blocks, from the Simulink library browser into the new Simulink Editor. Note that all blocks in the Editor can be moved from one point to another by a simple click-and-drag or select-and-shift method.

v) Connect all the blocks together, with the output port of a block joined to the input port of another until the blocks are fully joined according to the defined relations among the variables of the system. Alternatively, two blocks can also be joined together by selecting the block whose output is to be connected to the input of the second block, pressing and holding the "ctrl" key, and then clicking the second block.

vi) Set the simulation and block parameters. The parameters of each block are set in the block's parameter window, which can be opened by double-clicking the block. The simulation parameters, such as simulation time (start and stop), minimum step size, maximum step size, solver, solver type, etc., are also set from the "Configuration parameters" window under the "Simulation" menu in the Simulink Editor.

vii) Run the simulation. This is done by clicking the "Run" button on the Simulink Editor toolbar or by selecting "Run" from the "Simulation" menu in the Simulink Editor.

II.II Magnetic Levitation System Model

The schematic of a magnetic levitation system is shown in Fig. 1 [16]. It employs an electromagnetic circuit to suspend a ferromagnetic ball against the action of gravity.

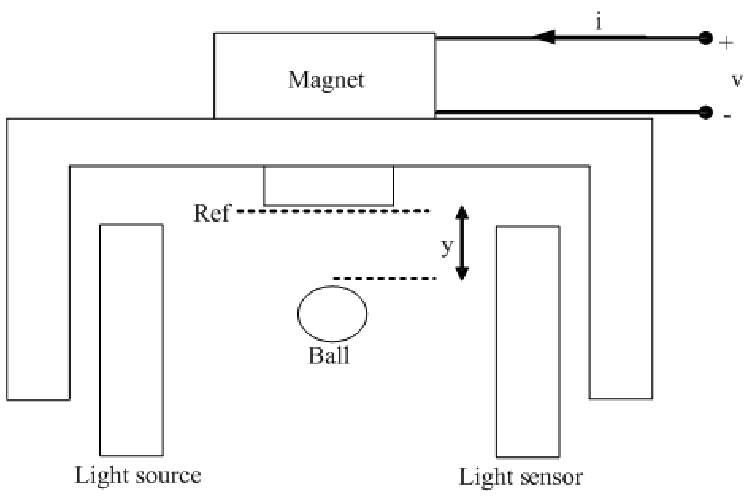

Fig. 1: Layout of a Magnetic Levitation System

The model of the system can be presented in conformance with the general representation

$$\dot{\mathrm{x}} = f(\mathrm{x}) + g(\mathrm{x})u \qquad (1)$$

of a nonlinear dynamic model, where x is a n-dimensional vector of state variables; f(x) and g(x) are nonlinear vector functions; and u is the control input.

Following the complete derivation given in Appendix, equation (1) is expanded into

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix} x_2 \\ g - \frac{K}{m}\left(\frac{x_3}{x_1}\right)^2 \\ -\left(\frac{R}{Lc} - \frac{2K}{Lc}\frac{x_2}{{x_1}^2}\right)x_3 \end{bmatrix} + \begin{bmatrix}0\\0\\ \frac{1}{Lc}\end{bmatrix}u, \qquad (2)$$

where x_1 is the position of the ball from the reference point, x_2 is the speed, x_3 is the current flowing in the coil of the electromagnet, m is the ball mass, g is the acceleration due to gravity, K is the magnetic force constant, R is the coil resistance of the electromagnet, Lc is the fixed inductance of the electromagnet, and u is the input voltage v applied across the electromagnet.

For a specific set of parameter values (see Appendix), equation (2) becomes

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix} x_2 \\ 9.81 - 0.0623\left(\frac{x_3}{x_1}\right)^2 \\ 0.0121\frac{x_2 x_3}{{x_1}^2} - 2.8532x_3 \end{bmatrix} + \begin{bmatrix}0\\0\\9.1743\end{bmatrix}u. \qquad (3)$$

To determine the stability of the nonlinear system, it is essential to first compute its equilibrium point, and then observe its behaviour about that point. The indirect method of Lyapunov states that a nonlinear system is asymptotically stable about an equilibrium point if every state trajectory originating from a small neighborhood around the equilibrium point at an initial time t_0 terminates at the point at a future time $t_f \leq \infty$ [17]. Since an equilibrium point represents a point in state space where the derivative of the state vector is zero, analytically, the equilibrium point or state x_{me} for equation (3) gives

$$\mathrm{x}_{me} = \begin{bmatrix} x_{1me} \\ 0 \\ (12.5476x_{1me}) \end{bmatrix}, \qquad (4)$$

where x_{1me} is the equilibrium value for the first state variable, which is the ball position. To completely determine x_{me}, the model of the system is set up in Simulink, as shown in Fig. 2, and ran from the MATLAB editor using the script in Table 1.

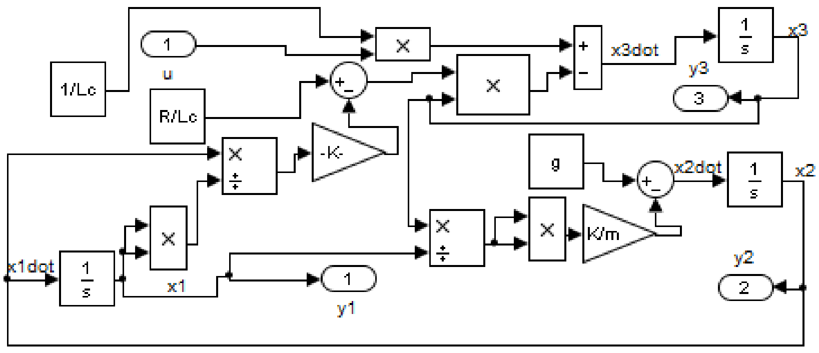

Fig. 2: Nonlinear magnetic levitation Simulink model

Note that equation (4) is not unique, as it depends on the chosen equilibrium value of the ball position. A number of initial guess values for the state vector make the "trim" function converge to corresponding solutions. For instance, the guess value x0 = [2; 1; 3] yields the point

$$\mathrm{x}_{me} = \begin{bmatrix} 0.01 \\ 0 \\ 0.125 \end{bmatrix}. \qquad (5)$$

The stability of the point in equation (5) can be easily inferred by examining the signs of the poles of the system linearized about that point. The same script in Table 1 serves this purpose as well. The magnetic levitation system has one unstable pole, and is said to be unstable. The next task is to design nonlinear controllers to stabilize the system and improve its overall dynamic characteristics.

Table 1: MATLAB script for the equilibrium point of a magnetic levitation system

Magnetic levitation script

```
clc,clear % Clear the workspace and command window
%% Define system parameters
R=31.1; Lc=0.109; g=9.80671;
K=0.0006590;m=0.01058;
%% Define guess values for x, u, and y
x0=[2;1;3];  % Initial state vector guess
u0=1;        % Initial input guess
y0=[0;0;0];  % Initial output guess
%% Define other arguments used to constrain x, u, and y
ix=[];       % No state variable is constrained
iu=[];       % The input is not constrained
iy=[];       % The output is not constrained
%% Find the equilibrium point xe using the MATLAB function "trim". The first argument is the Simulink model
% name
[xme,ume,yme,dx]=trim('maglev_eq_sim1',x0,u0,y0,ix,iu,iy);
xme=[-xme(1);-xme(2);xme(3)];
%% Determine the stability of the equilibrium point xme by examining the poles of the equivalent linearized
%  system (about the point) using the MATLAB functions "eig" and "linmod".
[A,B,C,D]=linmod('maglev_eq_sim1',xme,ume);
sys1_poles=eig(A);
sys1_sign=sign(sys1_pole);
if sys1_sign(1)>0 || sys1_sign(2)>0 || sys1_sign(3)>0
   disp('The system is unstable')
elseif sys1_sign(1)<0 && sys1_sign(2)<0 && sys1_sign(3)<0
   disp('The system is asymtotically stable')
else
   disp('The system stability cannot be inferred')
end
```

III. SYSTEM DESIGN

There are a number of techniques [18] for handling nonlinearities in control systems. The most common technique involves approximating nonlinearities using the Taylor series method. Although this approach normally leads to a less complicated linearized equivalent model, which is only valid in a small neighborhood around the system operating point, it has been used extensively in the literature for control system design—especially when it is known, or assumed, that system parameter variations are inconsiderable. Other techniques as well, such as state-feedback linearization, robust control, passivity-based design, backstepping control, etc., have their benefits and drawbacks. The underlying technique used in this paper is state-feedback linearization, because the controllers designed based on it have the ability to compensate nonlinearity as well as furnish the system with stability and good dynamic performance. The complete design framework is shown in Fig. 3.

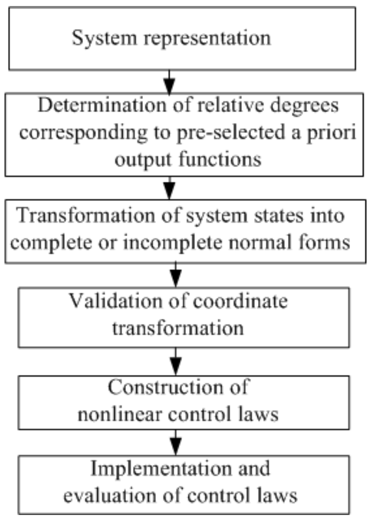
System representation
Determination of relative degrees corresponding to pre-selected a priori output functions
Transformation of system states into complete or incomplete normal forms
Validation of coordinate transformation
Construction of nonlinear control laws
Implementation and evaluation of control laws

Fig. 3: Design framework for compensating nonlinear dynamics

III.I Determination of System Relative Degrees

The relative degree of a system with respect to an output function determines principally whether an affine nonlinear system can be completely or partly transformed into a controllable form. If the relative degree is the same as the order of the system, then complete transformation is possible—otherwise, only partial transformation with associated system internal dynamics can be attained.

In general, the relative degree of an affine nonlinear system given in equation (1) with respect to an output function *c(x)* is the positive integer m such that

$$L_g c(\mathrm{x}) = L_g L_f c(\mathrm{x}) = \cdots = L_g L_f^{m-2} c(\mathrm{x}) = 0 \qquad (6)$$

and

$$L_g L_f^{m-1} c(\mathrm{x}) \neq 0 \qquad (7)$$

at the operating point x = x_0, where $L_g L_f^k c(x)$, $k = 1, 2, \cdots, m-1$, is given by [19]

$$L_g L_f^k c(\mathrm{x}) = \frac{\partial L_f^k c(\mathrm{x})}{\partial \mathrm{x}} g(\mathrm{x}) \qquad (8)$$

and called the Lie derivative of $L_f^k c(\mathrm{x})$ along g(x). Table 2 presents a MATLAB function for determining the relative degree of a general single-input single-output (SISO) affine nonlinear system. The script for specifically finding the relative degree of the magnetic levitation system given in equation (2) is provided in the table as well. Table 3 gives the relative degrees of the magnetic levitation system with respect to various output functions. The output functions are selected a priori based on ease of measurability (by direct sensing) or computability (through virtual sensing); their connotations can as well be easily inferred.

Table 2: MATLAB programs for finding the relative degree of an affine nonlinear system

MATLAB function
%MATLAB function sysreldeg(f,g,c,x) computes the %relative
degree of a general SISO affine nonlinear %system defined as
dX/dt=f(X) + g(X)u, y=c(X).
% f, g, and c are all defined as symbolic variables in the %calling
script. As well, the state variables are defined as %symbolic
variables with names x1, x2, x3,..., xn.
%The order of the system must be at least 2.
function Rdeg=sysreldeg(f,g,c,x)
order_ofsys=max(size(f));
% Initialize necessary system arrays
LfCx_s=zeros(1,order_ofsys);
LgLfcx_s=zeros(1,order_ofsys);
LfCx= sym(LfCx_s);LgLfCx=sym(LgLfcx_s);
% Determine the entries of LfCx
LfCx(1)=c;
if order_ofsys==2
jacob_1= jacobian(LfCx(order_ofsys-1),x);
LfCx(order_ofsys)=jacob_1*f;
else
for i=2:order_ofsys
jacob_2=jacobian(LfCx(i-1),x);
LfCx(i)=jacob_2*f;
end
end
% Determine the entries of LgLfCx
for i=1:order_ofsys
LgLfCx(i)=jacobian(LfCx(i),x)*g;
end
% Determine and output the relative degree of the system
input('Supply the operating point x0 in the format: x1 = ; x2 = ;
x3 = ; ... ; xn = ; ')
LgLfCx_numval=subs(LgLfCx);
LgLfCx_ind=find(LgLfCx_numval);
Rdeg=LgLfCx_ind(1);

MATLAB script
%This script finds the relative degree of a magnetic %
levitation system.
clear
clc
% Define necessary symbolic variables
syms x1 x2 x3 f g c
% Provide system parameters
R=31.1;Lc=0.109;gv=9.81;
K=0.0006590;
m=0.01058;
% Provide the entries of functions f and g
x=[x1 x2 x3];
f_1=x2;
f_2=(gv-K/m*(x3/x1)^2);
f_3=(-(R/Lc-2*K/Lc*(x2/(x1^2)))*x3);
f=[f_1 f_2 f_3]';
g=[0 0 1/Lc]';
%Define the output function c and call the MATLAB
%function sysreldeg(f,g,c,x)
c=x1;
RD=sysreldeg(f,g,c,x);
Rdegree=['The relative degree m of the system is: '
num2str(RD)];
disp(Rdegree)

Table 3: Relative degrees of a magnetic levitation system

Output function *c(x)*	**Relative degree** m
$x_1 - 0.01$	3
${x_1}^2 - 0.01^2$	3
${x_2}^2$	3
x_2	2
$x_1 + x_2$	2

III. II. Derivation and Validation of Normal Forms of System State Equations

With a number of output functions chosen, as given previously, the system can now be transformed into controllable forms using the coordinate transformation function W = Ψ(x). The entire flowchart for this transformation is shown in Fig. 4. This study only considers a case in which the relative degree is equal to the order of the system, because the incomplete or partial transformation does result in the presence of zero or internal dynamics, which may be stable or unstable.

A general SISO affine nonlinear system model given in equation (1) can be converted, using the transformation function W = Ψ(x), into the fully controllable normal form if the relative degree of the system to an output function c(x) is the same as the order of the system, and the jacobian matrix of the transformation function is invertible at the system operating point x0 [17]. The transformation function is defined as [19]

$$\begin{aligned} w_1 &= c(\mathrm{x}) \\ w_2 &= L_f c(\mathrm{x}) \\ w_3 &= {L_f}^2 c(\mathrm{x}) \\ &\vdots \\ w_n &= {L_f}^{n-1} c(\mathrm{x}), \end{aligned} \tag{13}$$

and the transformed states as

$$\begin{aligned} \frac{dw_1}{dt} &= w_2 \\ \frac{dw_2}{dt} &= w_3 \\ \frac{dw_3}{dt} &= w_4 \\ &\vdots \\ \frac{dw_n}{dt} &= u_{eq} \end{aligned} \tag{14}$$

where u_{eq} is called the equivalent control signal, which is expressed as

$$u_{eq} = L_f^n c(\mathrm{x}) + L_g L_f^{n-1} c(\mathrm{x}) u. \tag{15}$$

The MATLAB function for computing and validating equation (13) as well as $L_f^n c(\mathrm{x})$ and $L_g L_f^{n-1} c(\mathrm{x})$ is given in Table 4. A calling script for the magnetic levitation system is provided in the table. Table 5 shows the various components of equations (13) and (14) with respect to the output functions (of the magnetic levitation system) that yield m = n. Note that the system components when $c(\mathrm{x}) = {x_2}^2$ are invalid, because the transformation resulting from this output fails to meet the requirement for its validity.

III.III Determination and Implementation of Control Laws

Three different control signals are constructed based on the transformed state equations given in equation (14): direct state feedback control, state homogeneity-based control, and modified homogeneity-based control. From equation (15), the overall control law is defined as

$$u = \frac{u_{eq} - L_f^m c(\mathrm{x})}{L_g L_f^{m-1} c(\mathrm{x})}. \tag{16}$$

The equivalent control takes the form

$$u_{eq} = -k_1[\sigma_1(w)] - k_2[\sigma_2(w)] - k_3[\sigma_3(w)], \tag{17}$$

where *σ1*, *σ2*, and *σ3* are functions of transformed state vector $w = [w_1\ w_2\ w_3]^{\mathrm{T}}$; *k1*, *k2*, and *k3* are the controller gains. Rewriting equation (17) in terms of the original state vector $\mathrm{x} = [x_1\ x_2\ x_3]^{\mathrm{T}}$ gives

$$u_{eq} = \left[-k_1[\sigma_1(w)] - k_2[\sigma_2(w)] - k_3[\sigma_3(w)]\right]\Big|_{W=\Psi(\mathrm{x})} \tag{18}$$

Direct state feedback control

By applying a direct state feedback control technique [20, 21] to the transformed state equations, equation (18) becomes

$$u_{eq} = [-k_1(w_1) - k_2(w_2) - k_3(w_3)]|_{W=\Psi(\mathrm{x})}, \tag{19}$$

or, according to Table 5,

$$u_{eq} = -k_1[c(\mathrm{x})] - k_2[L_f c(\mathrm{x})] - k_3[L^2{}_f c(\mathrm{x})]. \tag{20}$$

The only controller parameters are *k1*, *k2*, and *k3*. The values of these parameters are chosen based on simulation-guided pole assignment.

State homogeneity-based control

In terms of a dilated state homogeneity-based control [22], the equivalent control also becomes

$$u_{eq} = [-k_1 sign(w_1)|\tau w_1|^{\tau_1} - k_2 sign(w_2)|\tau w_2|^{\tau_2} - k_3 sign(w_3)|\tau w_3|^{\tau_3}]|_{W=\Psi(x)} \quad (21)$$

where τ is a dilation gain; τ_1, τ_2, and τ_3 are additional controller gains defined as

$$\tau_{r-1} = \frac{\tau_r \tau_{r+1}}{2\tau_{r+1} - \tau_r}, \qquad \forall\, r = 2, 3, 4, \cdots, m \quad (22)$$

with $\tau_{r+1} = 1$, $\tau_m \in (1-\epsilon, 1)$, and $\epsilon \in (0,1)$. Note that *sign(.)* is a sign function whose output is 0, 1, or -1, depending on whether its argument is zero, positive, or negative, respectively, and the symbol | . | stands for the absolute value function.

The controller parameters are k_1, k_2, k_3, τ_1, τ_2, τ_3, τ, and ϵ. Their values are all chosen by simulation. Here k_1, k_2, and k_3 represent the coefficients of a stable characteristic polynomial in order of the increasing power of the polynomial variable.

Modified state homogeneity-based control

This is a modified form of equation (21) introduced as a way to reduce the number of controller parameters, and still maintain an appreciably good dynamic performance. It is expressed as

$$u_{eq} = [-k_1 sat(w_1, \epsilon_1) - k_2 sat(w_2, \epsilon_2) - k_3 sat(w_3, \epsilon_3)]|_{W=\Psi(x)} \quad (23)$$

where *sat(.)* is the saturation function

$$sat(f_1, f_2) = min\big(1,\ max(-1,\ f_1/f_2)\big). \quad (24)$$

The parameters of the controller are k_1, k_2, k_3, ϵ_1, ϵ_2, and ϵ_3. Often $\epsilon_1 = \epsilon_2 = \epsilon_3$.

Combining the system model with the three control inputs leads to the overall implementation schemes shown in Fig. 5, Fig. 6, and Fig. 7, respectively. (These schemes are developed specifically for a case with output $c(x) = x_1 - 0.01$. Similar ones can be produced for the other case with $c(x) = x_1^2 - 0.01^2$.)

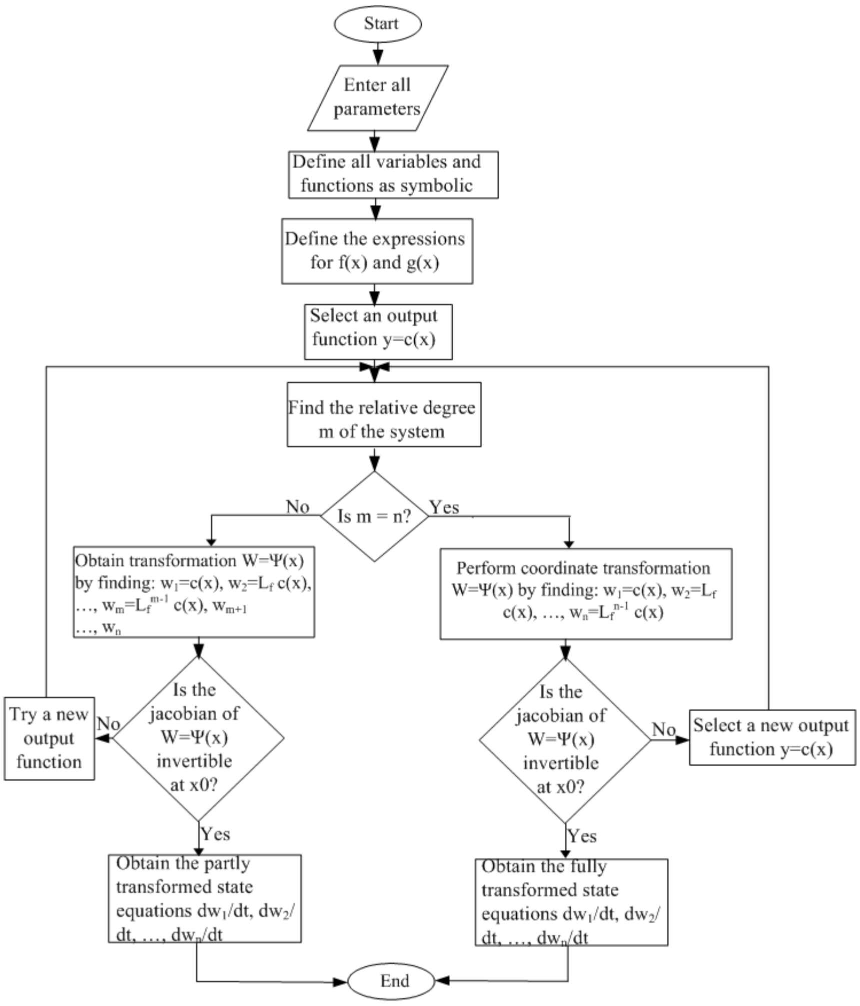
Start
Enter all parameters
Define all variables and functions as symbolic
Define the expressions for f(x) and g(x)
Select an output function y=c(x)
Find the relative degree m of the system
Is m = n?
No
Yes
Obtain transformation W=Ψ(x) by finding: w1=c(x), w2=Lf c(x), …, wm=Lf^m-1 c(x), wm+1 …, wn
Perform coordinate transformation W=Ψ(x) by finding: w1=c(x), w2=Lf c(x), …, wn=Lf^n-1 c(x)
Is the jacobian of W=Ψ(x) invertible at x0?
No
Try a new output function
Select a new output function y=c(x)
Yes
Obtain the partly transformed state equations dw1/dt, dw2/dt, …, dwn/dt
Obtain the fully transformed state equations dw1/dt, dw2/dt, …, dwn/dt
End

Fig. 4: Flowchart for transformation of SISO affine nonlinear state equations

Table 4: MATLAB programs for computing the transformation function W = Ψ(x)

MATLAB function	**MATLAB script**

%MATLAB function full_lin(f,g,c,x) computes the
% function for transforming a general SISO affine
%nonlinear system defined as dX/dt=f(X) + g(X)u,
%y=c(X), into the FULLY cotrollable normal forms.
%It also verifies if the transformation is valid.
% f, g, and c are all defined as symbolic variables in the calling script.
% As well, the state variables are defined as symbolic
%variables with names x1, x2, x3,..., xn. The order of the
%system must be at least 2.
function [LfCx, LgLfCx]=full_lin(f,g,c,x)
order_ofsys=max(size(f));
% Initialize necessary system arrays
LfCx_s=zeros(1,order_ofsys);
LgLfcx_s=zeros(1,order_ofsys);
jac_mat_s=zeros(sysorder,sysorder);
LfCx= sym(LfCx_s);LgLfCx=sym(LgLfcx_s);
jac_matrix=sym(jac_mat_s);
% Determine the entries of LfCx
LfCx(1)=c;
if order_ofsys==2
jacob_1= jacobian(LfCx(order_ofsys-1),x);
LfCx(order_ofsys)=jacob_1*f;
jacob_2= jacobian(LfCx(sysorder),x);
LfCx(order_ofsys+1)=jacob_2*f;
else
for i=2:order_ofsys+1
jacob_2=jacobian(LfCx(i-1),x);
LfCx(i)=jacob_2*f;
end
end
% Determine the entries of LgLfCx
for i=1:order_ofsys
LgLfCx(i)=jacobian(LfCx(i),x)*g;
end
% Validate the transformation
for ii=1:order_ofsys
jac_matrix(ii,:)=jacobian(LfCx(ii),x);
end
input('Enter all the n steady-state values as : x1 = ; x2 = ; x3 = ; ... ; xn = ; ')
jac_matrix_comp=subs(jac_matrix);
jj=det(jac_matrix_comp);
jj=single(jj);
if jj~=0;
disp('The transformation is valid, and the returned function is correct')
else
disp('The transformation is NOT valid, and the returned function is NOT correct')
end
	%This script finds the components of equations (13)
	%and (14) and validate them.
	clear
	clc
	% Define necessary symbolic variables
	syms x1 x2 x3 f g c
	% Provide system parameters
	R=31.1;Lc=0.109;gv=9.81;
	K=0.0006590;
	m=0.01058;
	% Provide the entries of functions f and g
	x=[x1 x2 x3];
	f_1=x2;
	f_2=(gv-K/m*(x3/x1)^2);
	f_3=(-(R/Lc-2*K/Lc*(x2/(x1^2)))*x3);
	f=[f_1 f_2 f_3]';
	g=[0 0 1/Lc]';
	%Define the output function c and call the MATLAB
	%function full_lin(f,g,c,x)
	c=x1;
	[lfh, lglfh]=full_lin(f,g,h,x);

Table 5: Components of equations (13) and (14) with respect to c(x) that gives m = n

Components of W = Ψ(x)	**Output function *c(x)***		
	$\boldsymbol{x_1 - 0.01}$	$\boldsymbol{x_1{}^2 - 0.01^2}$	$\boldsymbol{x_2{}^2}$
$w_1 = c(\mathrm{x})$	$x_1 - 0.01$	$x_1{}^2 - 0.01^2$	Not valid
$w_2 = L_f c(\mathrm{x})$	x_2	$2x_1x_2$	Not valid
$w_3 = L^2{}_f c(\mathrm{x})$	$9.81 - 0.0623\dfrac{x_3{}^2}{x_1{}^2}$	$19.62x_1 + 2x_2{}^2$ $-0.1246\dfrac{x_3{}^2}{x_1}$	Not valid
$L^3{}_f c(\mathrm{x})$	$0.1246\dfrac{x_2x_3{}^2}{x_1{}^3}$ $-0.0015\dfrac{x_2x_3{}^2}{x_1{}^4}$ $-35.5438\dfrac{x_3{}^2}{x_1{}^2}$	$58.86x_2$ $+71.0876\dfrac{x_3{}^2}{x_1}$ $-0.1246\dfrac{x_2x_3{}^2}{x_1{}^2}$ $-0.0030\dfrac{x_2x_3{}^2}{x_1{}^3}$	Not valid
$L_g L_f^2 c(\mathrm{x})$	$-1.1429\dfrac{x_3}{x_1{}^2}$	$-2.2858\dfrac{x_3}{x_1}$	Not valid

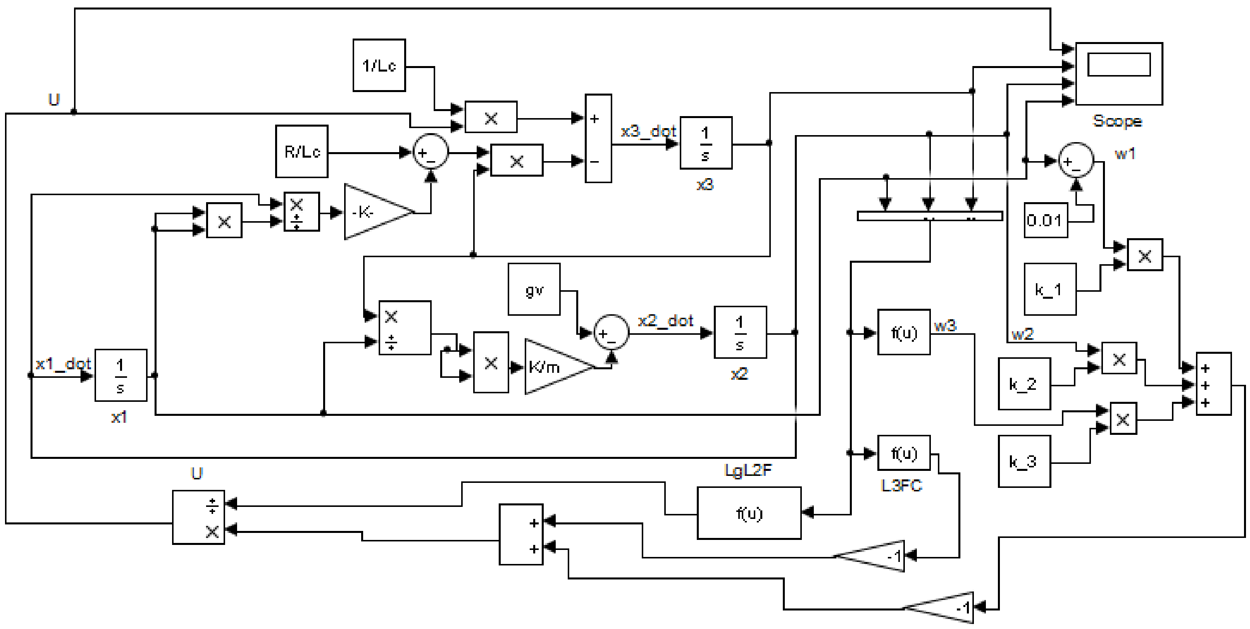

Fig. 5: Simulink model for a direct nonlinear state feedback control scheme

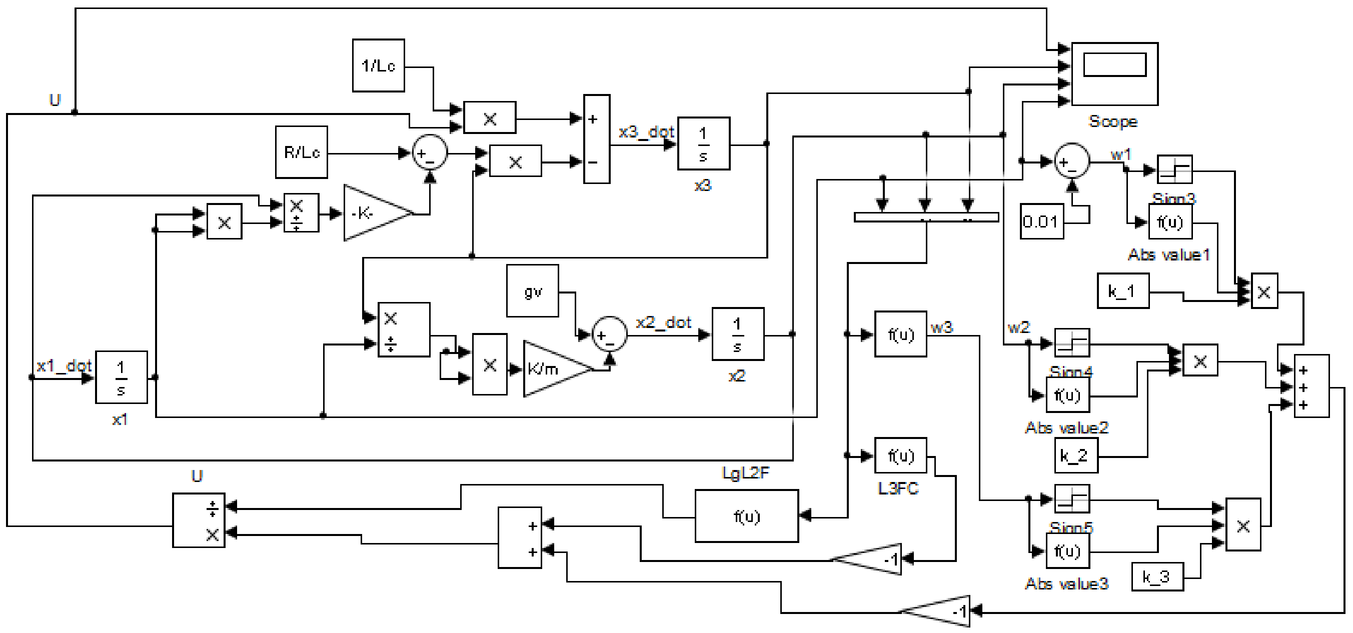

Fig. 6: Simulink model for a state homogeneity-based control scheme

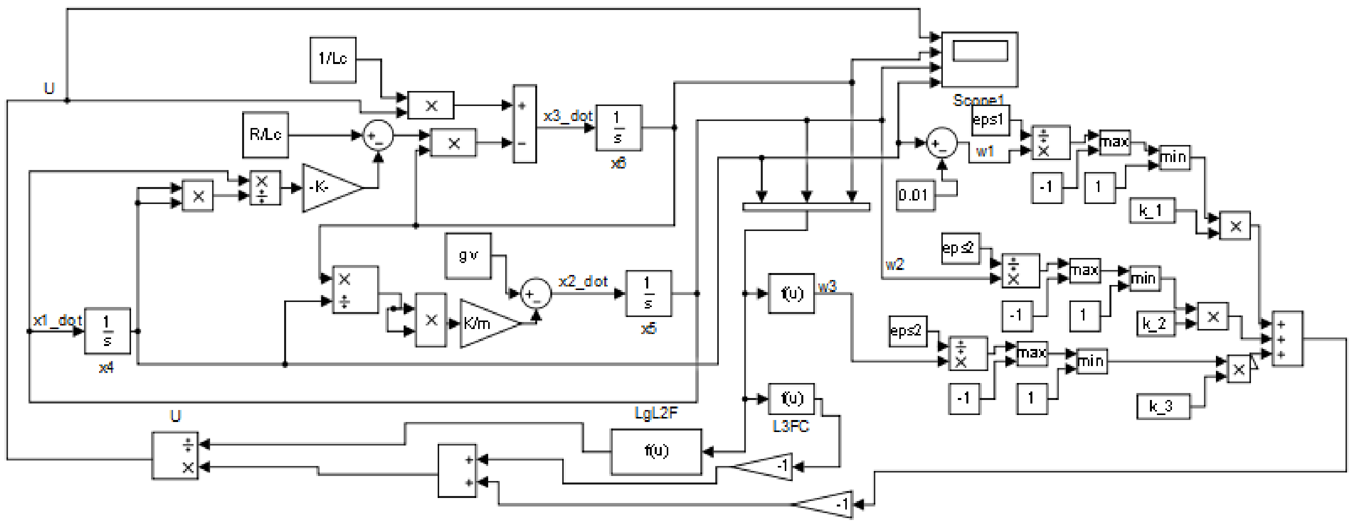

Fig. 7: Simulink model for a modified state homogeneity-based control scheme

IV. RESULTS AND DISCUSSION

The magnetic levitation system, as pointed out previously, has inherently unstable dynamics. This is reflected in the diverging free response of the system state vector, as portrayed in Fig. 8. This response depicts expected behaviour of the system when the input signal (i.e., voltage) is zero—the current is zero, and the ball position and velocity grow infinitely without bounds. In addition, subjecting the system to the control action of each of the three control laws, in an attempt to improve its dynamics, reveals further performance characteristics vis-à-vis the system output function or signal employed in developing the control laws.

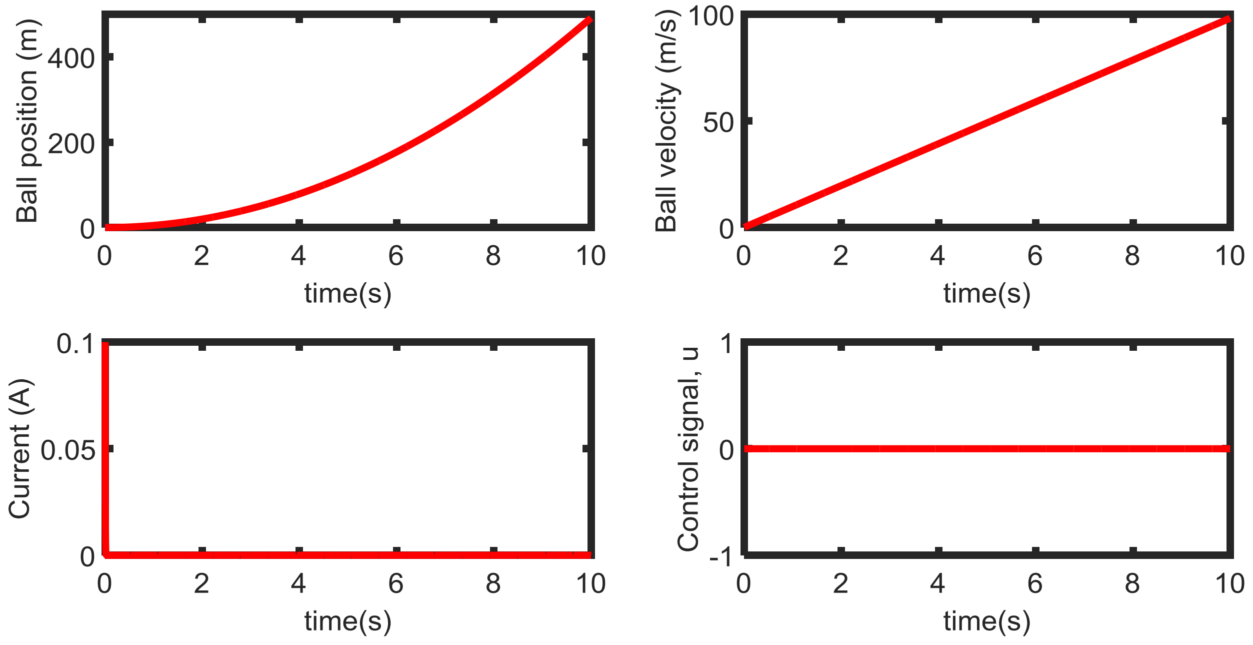

Fig. 8: Magnetic levitation system open-loop behaviour

IV.I System Response under Direct State Feedback Control

Two different system responses under direct state feedback control are shown in Fig. 9 and Fig. 10. Whereas Fig. 9 shows system behaviour under a normal operating condition, Fig. 10 depicts a condition when the system is subjected to a unit step change (2s duration) in the ball position from its steady-state value. The pre-selected pole set is [-15, -10, -7].

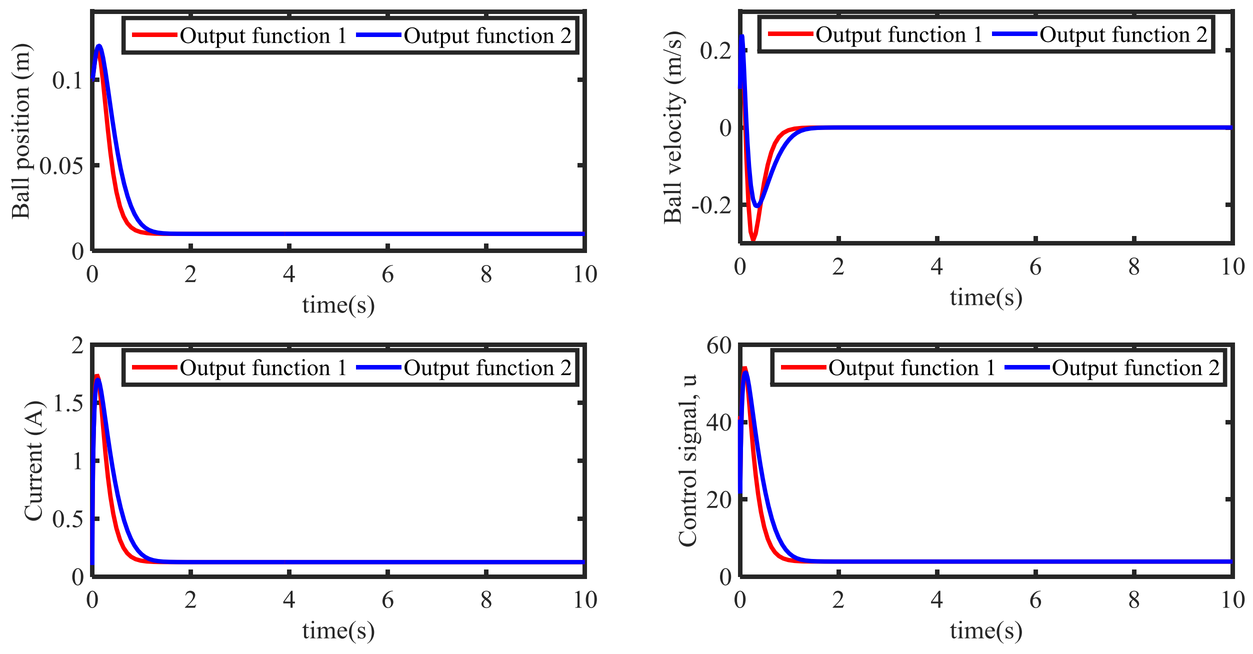

Fig. 9: Normal system operating condition under direct state feedback control

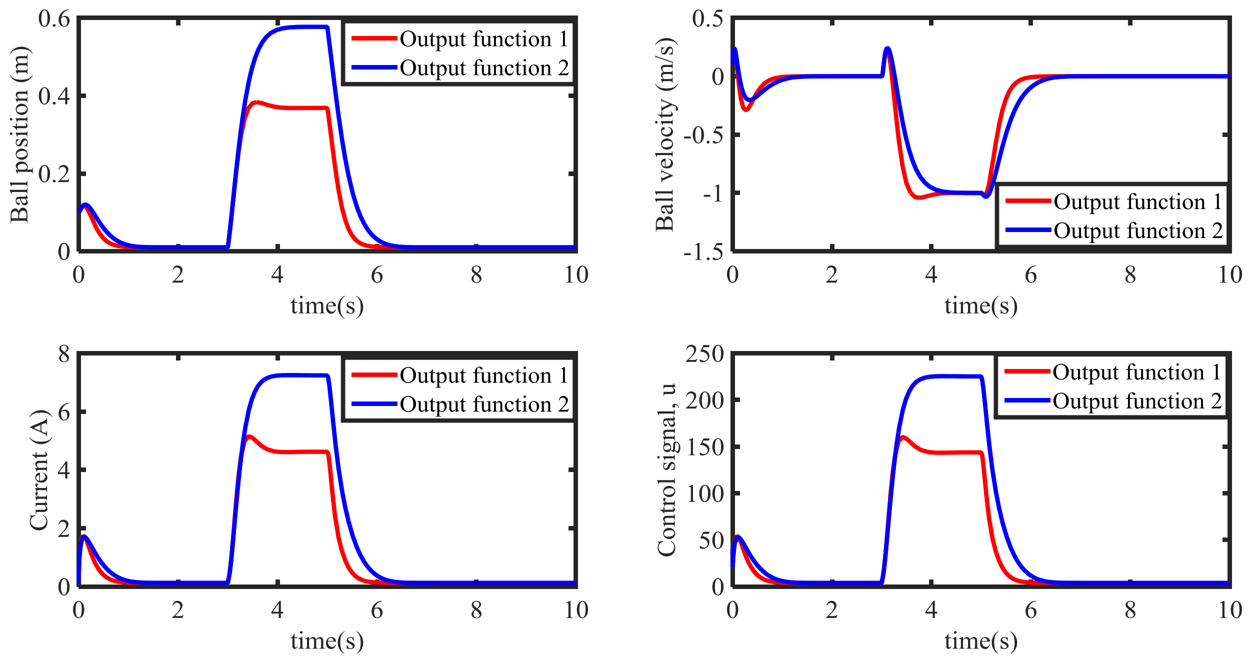

Fig. 10: Abnormal system operating condition under direct state feedback control

From many simulations scenarios carried out, it is observed that the greater the amplitude and/or duration of the disturbance signal, the greater the control effort required to bring the system back to its equilibrium point. Generally, under normal operating condition, the initial control effort for stable and good system dynamics depends on the location of the pre-selected closed-loop poles. Although the effort is larger when the poles are farther from the origin, closer poles could also result in high control energy for this particular nonlinear system. Again, for this control strategy, the first output function, $\boldsymbol{c(\mathbf{x}) = x_1 - 0.01}$, seems more desirable (as clearly demonstrated in Fig. 10).

IV.II System Response under State Homogeneity-based Control

The parameter values of the control law are chosen based on extensive simulations to determine the set that results in appreciable dynamic performance. These values are k_1 = 84, k_2 = 61, k_3 = 14, τ_1 = 17/20, τ_2 = 17/19, τ_3 = 17/18, and τ = 22. Fig. 11 shows the system performance during a normal operating condition, while Fig. 12 is a result of application of a step disturbance signal of magnitude 0.16 and duration 1s. From the simulations of several sets of parameter values carried out, it is observed that the system performance deteriorates for the output function, $c(\mathrm{x}) = x_1{}^2 - 0.01^2$, when a unit step disturbance signal of duration 2s is applied (see Fig. 13). As pointed out earlier, a substantial amount of control effort is required to counteract the effect of disturbance on the system performance.

IV.III System Response under Modified Homogeneity-based Control

For this case, system responses under normal and abnormal conditions are displayed in Fig. 14 and Fig. 15, respectively. The controller parameters are k_1 = 150, k_2 = 95, k_3 = 18, and $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$ = 0.85. As depicted in the figures, the system response with respect to the first output function displays higher initial oscillations in contrast to the response in terms of the second output function. This control strategy could accommodate different system output functions and tolerate disturbance more than the direct homogeneity-based control.

Generally, although the control signals for the three control strategies settle to a steady-state value of between 5 and 10v, a good amount of control effort is required initially to sustain the dynamic performance of the system. Further, because various output functions lead to different system dynamic performance under these control laws, the system behaviour should always be investigated with respect to those functions that meet the transformation conditions highlighted before in Section 3.

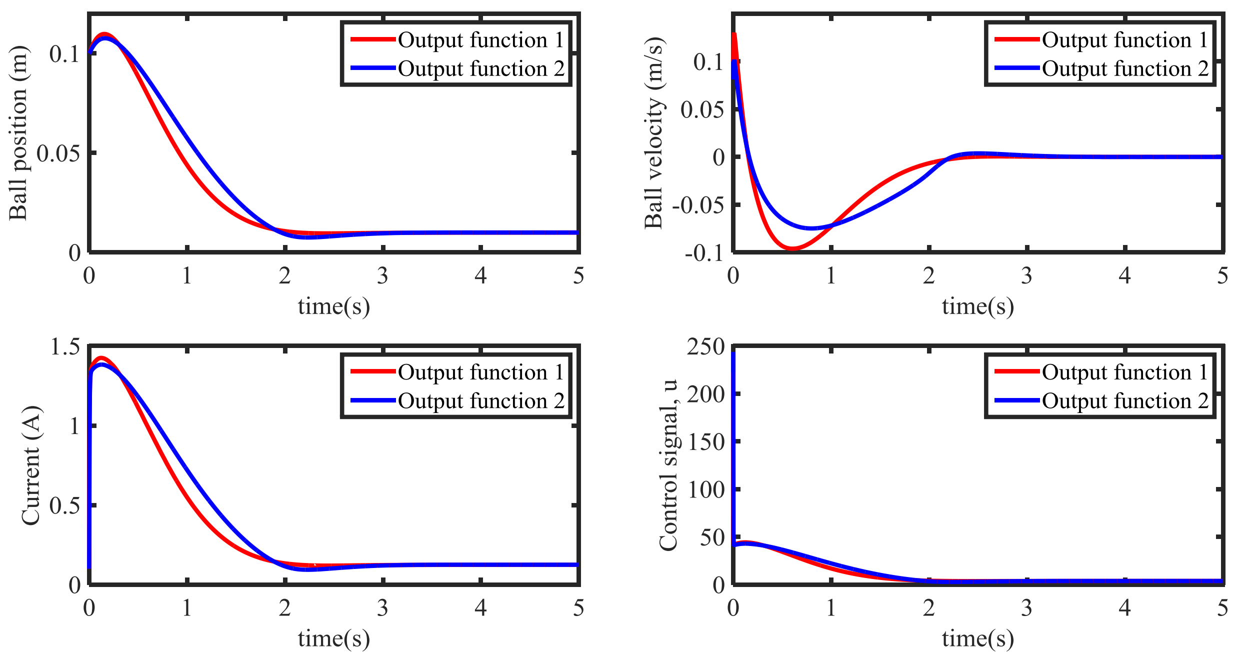

Fig. 11: Normal system operating condition under state homogeneity control

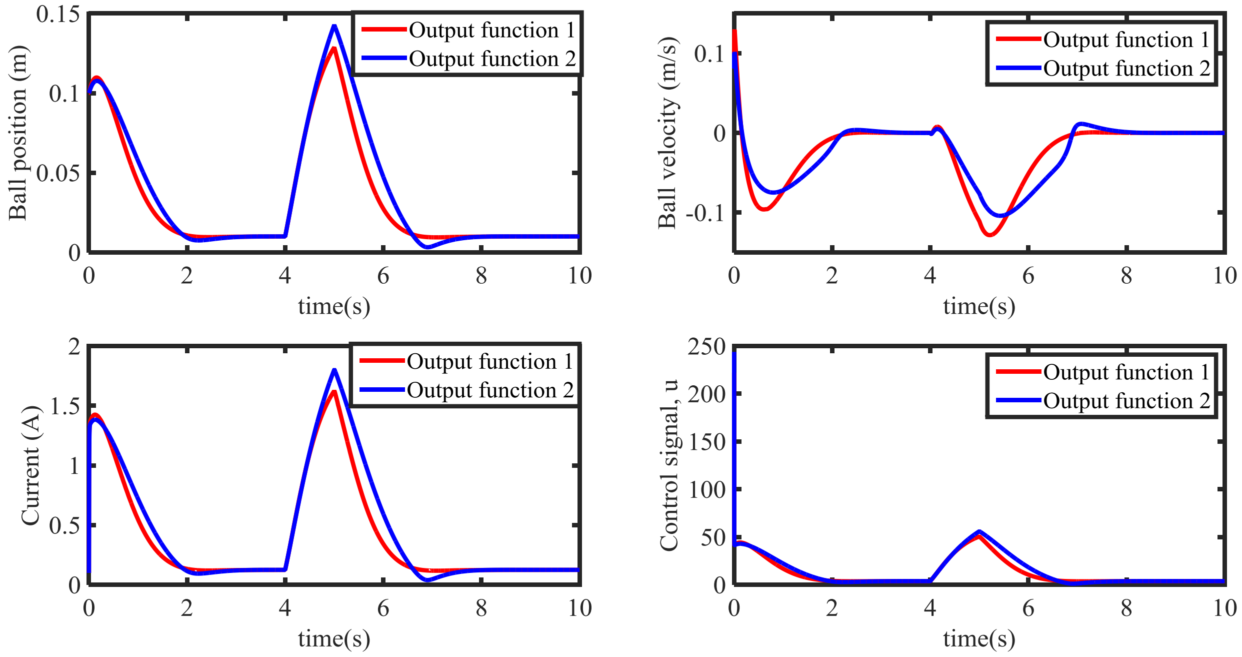

Fig. 12: Abnormal system operating condition under state homogeneity control
(a step disturbance signal of 0.16 amplitude and 1s duration)

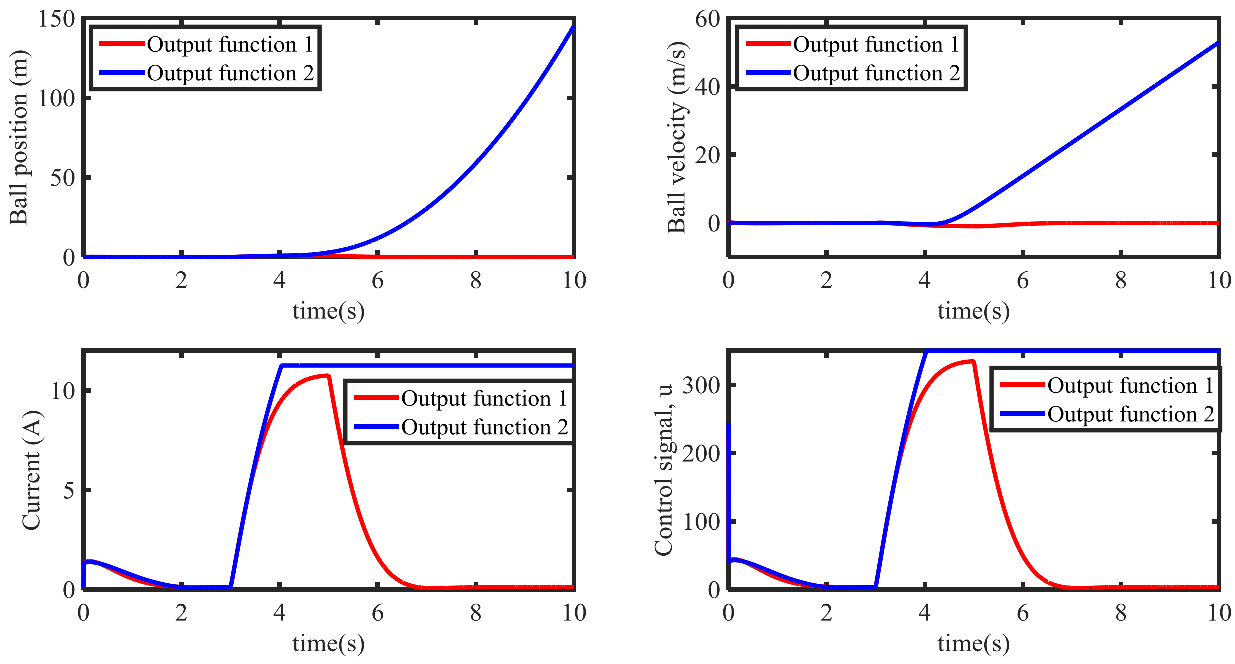

Fig. 13: Abnormal system operating condition under state homogeneity control (a unit step disturbance signal of 2s duration)

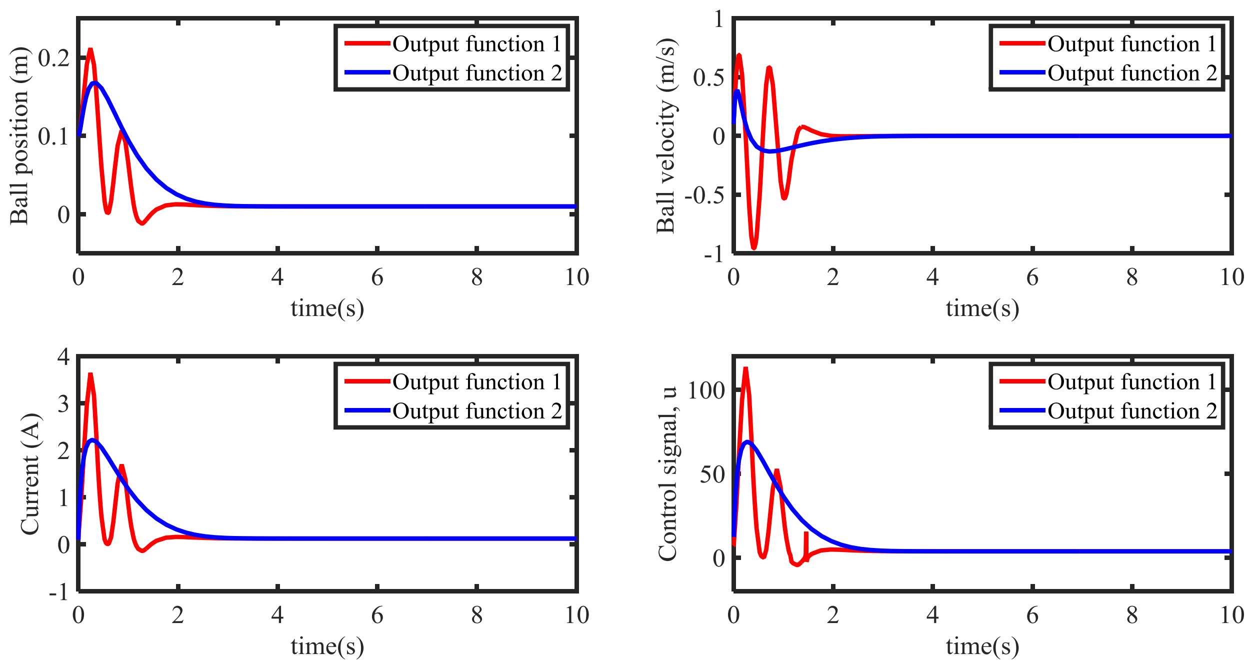

Fig. 14: Normal system operating condition under modified state homogeneity control

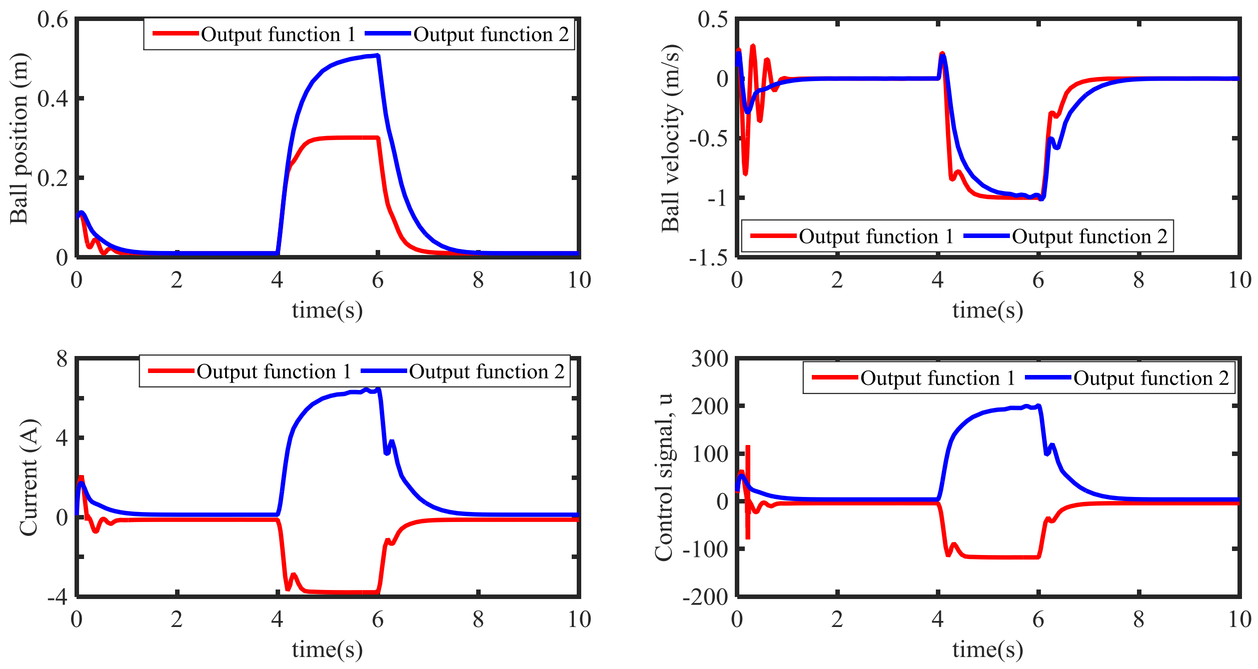

Fig. 15: Abnormal system operating condition under modified state homogeneity control (a unit step disturbance signal of 2s duration)

V. CONCLUSION

This paper has treated nonlinear dynamic simulation of a magnetic levitation system. Various Simulink models as well as MATLAB files are provided to aid the study. With the dynamics of the system shown to be unstable, three nonlinear control laws, based on state feedback linearization control, are constructed to improve the dynamic performance of the system. The key point is the transformation of the system model into completely controllable normal forms corresponding to system output functions chosen a priori and tested to meet necessary transformation conditions. The system performance is investigated under the control action of the control laws during both normal and abnormal operating conditions, thereby facilitating the assessment of the relative ability of each controller to stabilize and maintain acceptable dynamics of the system. The study will substantially benefit budding researchers, especially postgraduate and senior undergraduate students, who are pursuing carreer in nonlinear control.

Acknowledgements

This work is based on the research supported in part by the National Research Foundation of South Africa (Unique Grant No. 107541), Department of Higher Education and Training Research Department Grant (DHET-RDG), and Centre for Energy and Electric Power, Tshwane University of Technology (TUT), South Africa.

APPENDIX

Complete Derivation of a Magnetic Levitation System

The schematic of a magnetic levitation system shown in Fig. 1 has two parts: a mechanical part, which consists of the ball and a pair of light sensors for monitoring the motion of the ball along the vertical direction, and an electromagnetic part, which is mainly a magnet around which a current-carrying coil is wound.

The dynamic equations associated with these two parts can be obtained using the Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) - \frac{\partial L}{\partial y} = 0; \qquad \text{(A1)}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} + \frac{\partial P}{\partial \dot{q}} = v, \qquad \text{(A2)}$$

where L is termed the Lagrangian (which is the difference between the kinetic energy K and potential energy V of the entire system, i.e., K - V), y is the ball's position, q is the charge, P is the half-rate power function (due to the coil resistance R of the electromagnet), and v is the voltage applied across the electromagnet.

Noting that

$$K = \frac{1}{2}m\dot{y}^2 + \frac{1}{2}L_{ind}(y)\dot{q}^2;$$

$$V = mg(-y) = -mgy;$$

$$P = \frac{1}{2}R\dot{q}^2;$$

equations (A1) and (A2) become

$$m\ddot{y} + \frac{1}{2}\frac{L_0 y_0}{y^2} - mg = 0; \qquad \text{(A3)}$$

$$-\frac{L_0 y_0}{y^2}\dot{y}\dot{q} + L_{ind}(y)\frac{d\dot{q}}{dt} + R\dot{q} = v. \qquad (A4)$$

Note also that inductance $L_{ind}(y)$ varies according to the position of the ball, and is given by [10]

$$L_{ind}(y) = L_c + \frac{L_0 y_0}{y},$$

where Lc is a fixed inductance of the coil of the electromagnet, and Lo is the inductance given with respect to reference position yo.

Rearranging equations (A3) and (A4) yields

$$\ddot{y} = g - \frac{K}{m}\frac{i^2}{y^2}; \qquad (A5)$$

$$\frac{d\dot{q}}{dt} = \frac{v}{L_{ind}(y)} - \left(\frac{R}{L_{ind}(y)} - \frac{2K}{L_{ind}(y)}\frac{\dot{y}}{y^2}\right)\dot{q}. \qquad (A6)$$

$K = \frac{1}{2}L_0 y_0$ is defined as the magnetic force constant.

Equations (A5) and (A6) can be rewritten in state-space form given in equation (2) if the following state variables are assumed: $x_1 = y$; $x_2 = \dot{y}$; and $x_3 = \dot{q}$.

Often $L_c >> L_o$, and therefore, $L_{ind}(y)$ is replaced with L_c. Specific parameter values [16] for the model are R = 31.1Ω; Lc = 0.109H; g = 9.8067m/s^2; K = 0.0006590Nm^2/A^2; m = 0.01058kg.

REFERENCES

[1] P. Albertos and I. Mareels, Feedback and Control for Everyone. Springer-Verlag, Berlin Heidelberg, 2010.

[2] A. C. Antoulas, Approximation of large-scale dynamical systems. SIAM, Advances in Design and Control, 2005.

[3] W. Jia-Jun, "Simulation studies of inverted pendulum based on PID controllers", Simulation Modelling, Practice, and Theory, vol. 19, pp. 440–449, 2011.

[4] K. H. Ang, G. Chong, and Y. Li, "PID Control System Analysis, Design, and Technology", IEEE Trans. on Control System Technology, vol. 13, no. 4, pp. 559-576, 2005.

[5] H. Ying, Fuzzy Control and Modeling: Analytical Foundations and Applications. IEEE Press, 2000.

[6] M. Aycin, and R. Benekohal, Stability and performance of car-following models in congested traffic, Journal of Transportation Engineering, vol. 127, pp. 2–12, 2001.

[7] C. Unsal and P. Kachroo, "Sliding mode measurement feedback control for antilock braking systems", IEEE Transactions on Control Systems Technology, vol. 7(2), pp. 271-281, 1999.

[8] Feedback Instrument Limited, Magnetic levitation system—getting started, http://www.fdb.com, 18th September, 2012.

[9] Quanser Inc., Magnetic levitation workstation, http://www.quanser.com, 2013.

[10] A. A. Awelewa, I. A. Samuel, A. Abdulkareem, and I. O. Samuel, "An Undergraduate Control Tutorial on Root Locus-Based Magnetic Levitation System Stabilization", International Journal of Engineering & Computer Science IJECS-IJENS, vol. 13, no. 1, 2013.

[11] M. B. Naumivic, B. R. Veselic, "Magnetic levitation system in control engineering education", Automatic Control and Robotics, vol. 7, no. 1, pp. 151-160, 2008.

[12] G. Cho, Y. Kato and D. Spilman, "Sliding mode and classical controllers in magnetic levitation systems," IEEE Control System Magazine, vol. 13, pp. 42-48, 1993.

[13] J. Phuah, J. Lu, T. Yahagi, "Chattering free sliding mode control in magnetic levitation system", IEEJ Trans., EIS, vol. 125, no. 4, pp. 600-606, 2005.

[14] T. Agus, Y. Muhammad, L. Jianming, and Y. Takashi, "Implementation of a Fuzzy PID Controller Using Neural Network on the Magnetic Levitation System", 2006 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS2006) Yonago Convention Center, Tottori, Japan.

[15] MATLAB Online Documentation, Release 2015a, The MathWorks, Inc., United States.

[16] B. Shahian, and M. Hassul, Control System Design Using MATLAB. Englewood Cliffs: Prentice Hall, 1993.

[17] J. E. Slotine and W. Li, Applied Nonlinear Control. Prentice Hall, Eaglewood Cliffs, New Jessey, 1991.

[18] H. K. Khalil, Nonlinear Control (Global ed.). Pearson Education Ltd., England, 2015.

[19] A. Isidori, Nonlinear Control Systems: An Introduction. Springer Verlag, New York, 1995.

[20] C. K. Benjamin and G. Farid, Automatic Control Systems. John Wiley & Sons (Asia) Pte. Ltd., Singapore, 2003.

[21] K. Ogata, Modern Control Engineering. Prentice-Hall of India, New Delhi, 2010.

[22] S. Bhat and D. Bernstein, "Geometric Homogeneity with Applications to Finite Time Stability", Math. Control, Signals Systems, vol. 17, pp. 101-127, 2005.