NUS

National University of Singapore

Sliding Control with Signed-Sigma and Smooth Control Law

Student : Widya Ageng Setya Tutuko

Table of Contents

List of Figures

ABSTRACT

In this project report, a sliding control scheme is going to be implemented in a system with two states. The quality of state values convergence towards steady state (x = 0) and the chattering amplitude at the control signal will be the main performance metrics to observe. Theoretical details in obtaining the form of control law that will diminish the values of a Lyapunov function will be given in Chapter 1. While simulation and discussion of results will be presented in Chapter 2. There are two simulations performed to simulate different control law with one using signed-sigma (sign of sliding surface value) and the other one with saturation limits on sliding surface value in order to avoid chattering which is significantly observed at the control signal.

I. Plant Model, Parameter and Sliding Control Algorithm

A. State-space Model of Plant and Sliding Surface

Given a system with following state-space model:

$$\dot{x}_1 = ax_1 + bu + d$$

$$\dot{x}_2 = x_1$$

Reconstruct the equations above in matrix form:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 \\ 0 \end{bmatrix} d \equiv \dot{x} = Ax + Bu + Dd$$

Since both state values dynamic is to be observed (are of interest), the output of the system can be equalized to both states, i.e.:

$$y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \equiv y = Cx$$

The sliding surface polynomial which is planned to be implemented in the above system is given below:

$$\sigma = c_1 x_1 + c_2 x_2 \equiv \sigma = \Gamma^T x$$

Where $\Gamma^T = [c_1 \quad c_2] > 0$. Parameters a, b, c_1, c_2, d are known *a priori* with $|d| \leq d_{max}$.

B. Sliding Control Algorithm

To assure the stability of the system when state values converge to 0 ($x \rightarrow 0$), a Lyapunov cost that is a function of sliding surface value should asymptotically reach 0 when $t \rightarrow \infty$ and has negative gradient with respect to t.

Define a Lyapunov function:

$$V(\sigma(x,t)) = \frac{\sigma(t)^2}{2}$$

Differentiate over t yields:

$$\dot{V} = \sigma\dot{\sigma} = \sigma\Gamma^T\dot{x} = \sigma\Gamma^T(Ax + Bu + Dd)$$

Let control signal be designed below:

$$u = \frac{-\Gamma^T Ax - \mu\text{sign}(\sigma) - \text{sign}(\sigma)\Gamma^T Dd_{max}}{\Gamma^T B}$$

Such that,

$$\dot{V} = \sigma\dot{\sigma} = \sigma\Gamma^T\dot{x} = \sigma\left(\Gamma^T Ax + \Gamma^T B\frac{-\Gamma^T Ax - \mu\text{sign}(\sigma) - \text{sign}(\sigma)\Gamma^T Dd_{max}}{\Gamma^T B} + \Gamma^T Dd\right)$$

$$\dot{V} = -\mu\sigma\text{sign}(\sigma) - \sigma\text{sign}(\sigma)\Gamma^T D(d_{max} - d); \mu > 0$$

Which guarantees that $\dot{V} < 0$ since $\sigma\text{sign}(\sigma) > 0$ for $\sigma \neq 0$ and $(d_{max} - d) > 0$ because $d_{max} = \max(d) > d$.

The result above shows that $V(\sigma(x,t))$ has negative gradient and is always a non-negative value since it is a square function, indicating that $V \to 0$ when $t \to \infty$. Consequently, $V \to 0$ means that $\sigma \to 0$ and according to sliding polynomial form, it also indicates that $x \to 0$ since $c_1, c_2 \neq 0$ and σ is a stable polynomial (having all the roots at LHP).

Thus, when state values are asymptotically zero, $x \to 0$, the Lyapunov cost value is asymptotically zero too $V \to 0$ at $t \to \infty$, concluding that system incorporating control law $u = \frac{-\Gamma^T Ax - \mu\text{sign}(\sigma) - \text{sign}(\sigma)\Gamma^T Dd_{max}}{\Gamma^T B}$ will ensure asymptotical stability when $x \to 0$.

C. Smooth Control Law

The control law that is introduced earlier generates a chatter in the control signal waveform. This is due to the fact the hysteresis effect from sign function, $\text{sign}(\sigma)$, in the control algorithm. To reduce the chattering, the control signal algorithm is modified as follow:

$$u = \frac{-\Gamma^T Ax - \mu sat(\sigma,\varepsilon) - sat(\sigma,\varepsilon)\Gamma^T Dd_{max}}{\Gamma^T B}$$

Where $sat(\sigma,\varepsilon) = \begin{cases} 1, for\ \sigma > \varepsilon \\ \frac{\sigma}{\varepsilon}, for\ -\varepsilon \leq \sigma \leq \varepsilon \\ -1, for\ \sigma < -\varepsilon \end{cases}$

With the saturation function being introduced in the control law, the hysteresis steps will not be too large and therefore the chattering effect will be reduced which will be shown during simulation in next chapter.

D. Stability Analysis and State Convergence

The sliding surface is given in the form:

$$\sigma = c_1 x_1 + c_2 x_2$$

Once the states reach sliding surface (i.e.: $\sigma = 0$), its convergence towards $\mathrm{x} \to 0$ is governed by coefficients c_1 and c_2. Let $\sigma = 0$, the solution to this surface value is solution to differential equation:

$$\sigma = \mathrm{c_1 \dot{x}_2 + c_2 x_2 = 0 \text{ since } \dot{x}_2 = x_1}$$

that has solution:

$$c_1 \frac{dx_2}{dt} = -c_2 x_2 \to c_1 \int_{x_2(0)}^{x_2(t)} \frac{dx_2}{x_2} = -c_2 \int_0^t dt \to c_1 ln\left(\frac{x_2(t)}{x_2(0)}\right) = -c_2 t$$

$$x_2(t) = \mathrm{x_2(0)exp}\left(-\frac{c_2}{c_1} t\right)$$

$$\dot{x_2} = x_1 \to x_1 = \frac{d}{dt} x_2(0) exp\left(-\frac{c_2}{c_1} t\right)$$

$$x_1(\mathrm{t}) = -\frac{c_2}{c_1} x_2(0) exp\left(-\frac{c_2}{c_1} t\right)$$

The polynomial ratio $\frac{c_2}{c_1}$ determines the decay rate of state values towards zero. Note that if c_1 and c_2 have different signs, the state values will grow and become unstable. Equivalently, having different signs mean that the root of polynomial $\sigma = c_1 x_1 + c_2 x_2 = c_1 \dot{x}_2 + c_2 x_2 = c_1 x_2 s + c_2 x_2 = 0 \to$ $s = -\frac{c_2}{c_1}$ lies on RHP, implying the sliding surface is unstable. Thus, if c_1 and c_2 have the same signs, it is guaranteed that surface polynomial σ is a stable surface and state values converge to zero.

II. Simulink Simulation and Result Discussion

A. Simulation of Signed-Sigma Control Law

To simulate and observe the dynamic of state values, the initial state values $x(0)^T = [x_1(0) \quad x_2(0)]$ must be defined. The simulated dynamic is based on the model given below:

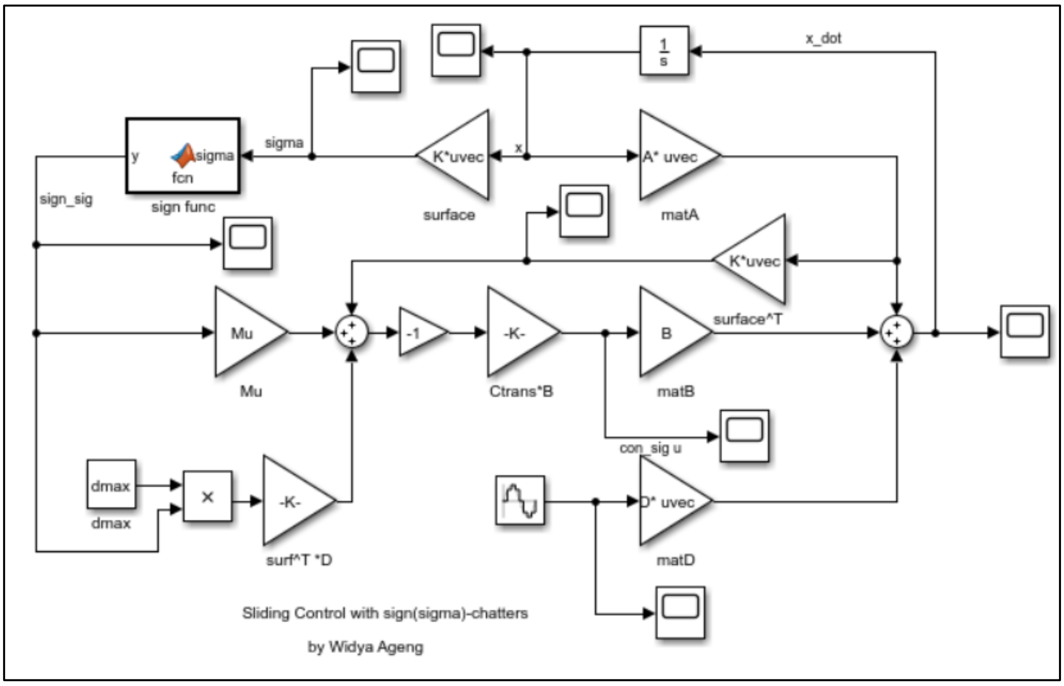


Figure 1 Simulink Modem of Sliding Control using signed Sigma

The embedded sign function contains sign assignment routine to determine the sign of σ, shown in detail below:

```
function y = fcn(sigma)

epsilon = 0.01;

if sigma>0
    out = 1;
else
    out = -1;
end

y = out;

```

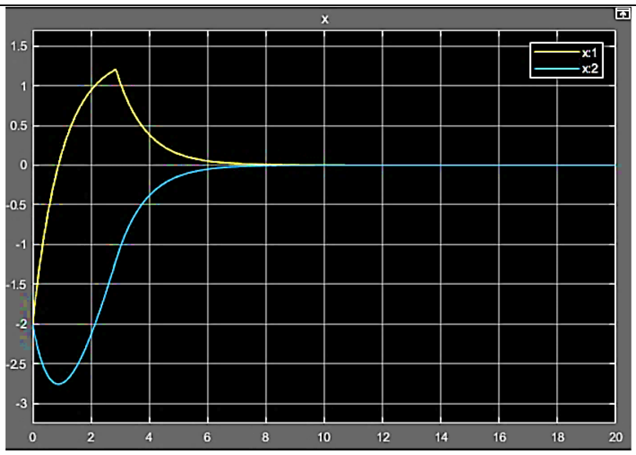
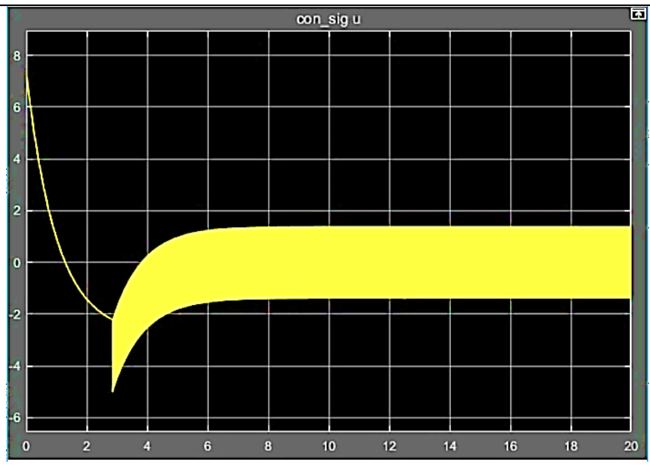
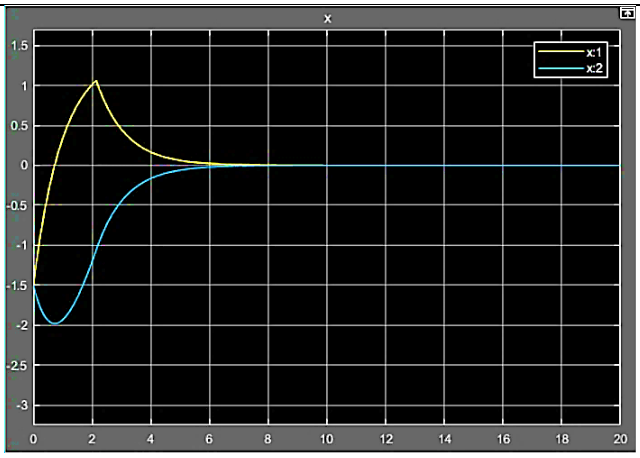
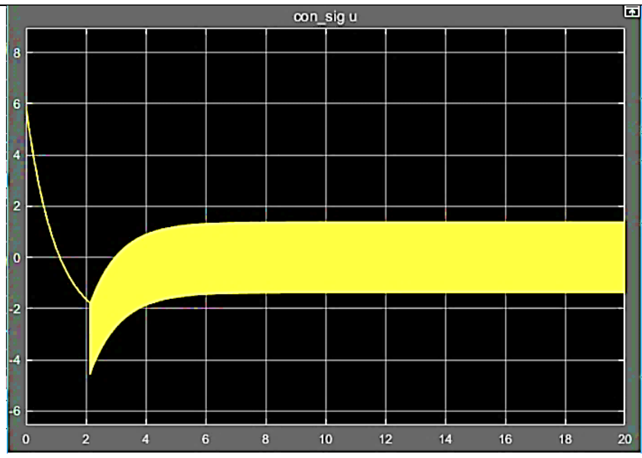
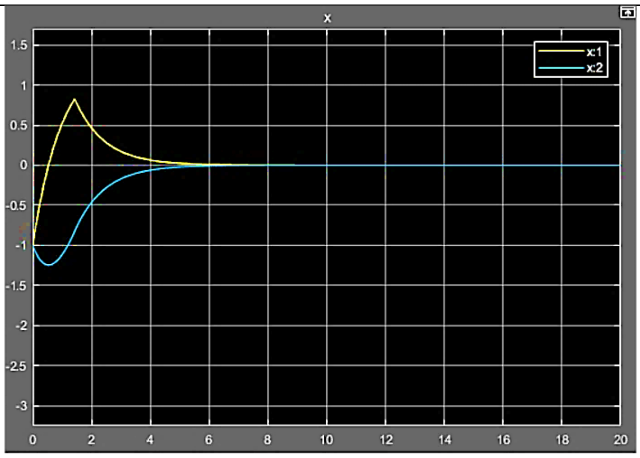
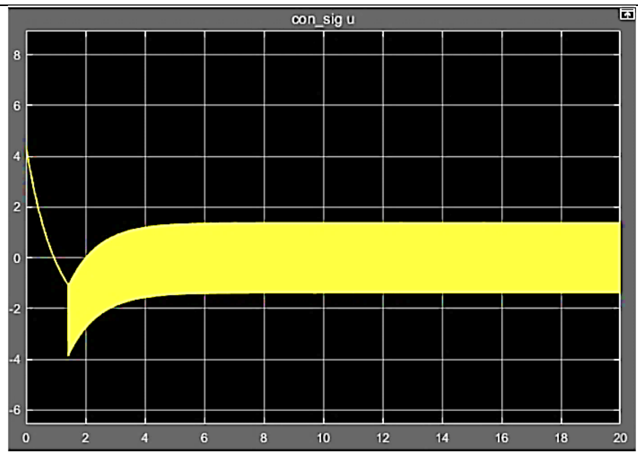
Figure 2 Sign function in embedded Simulink block

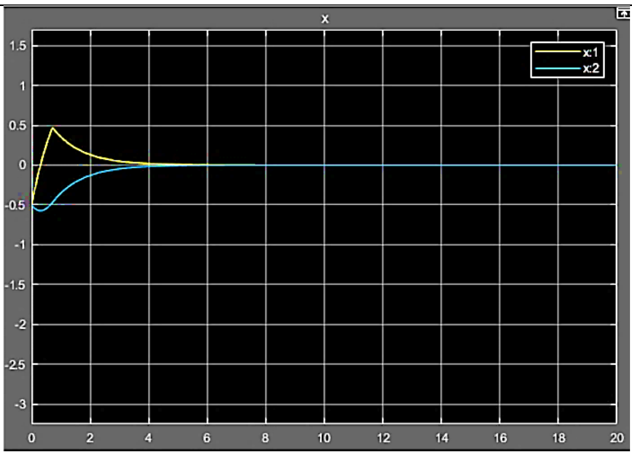
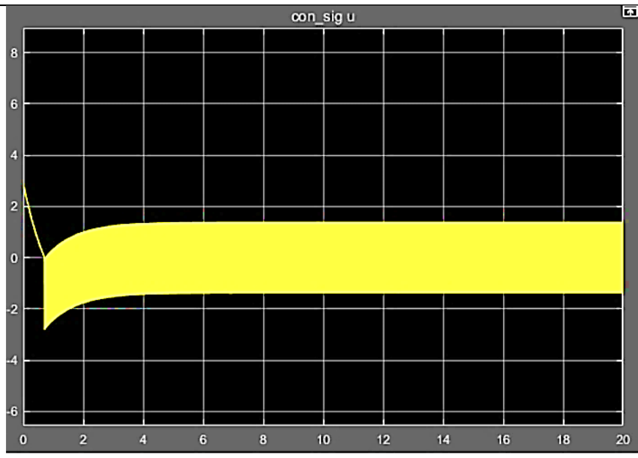
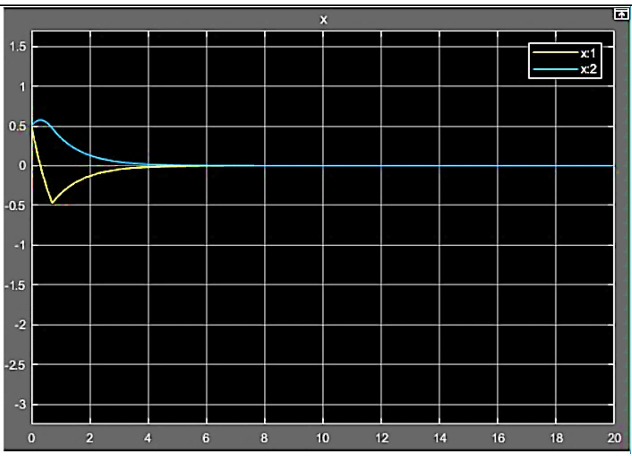
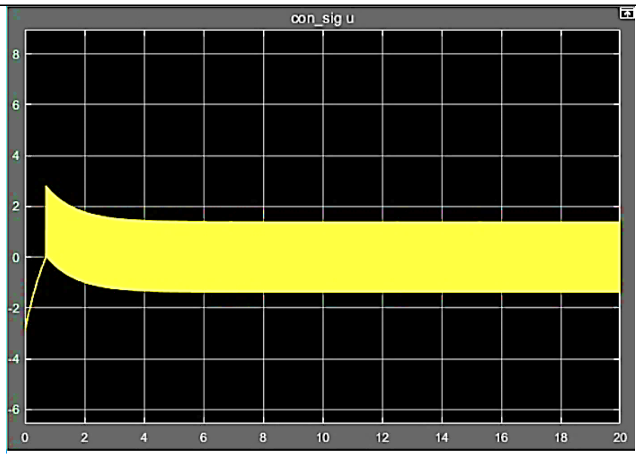
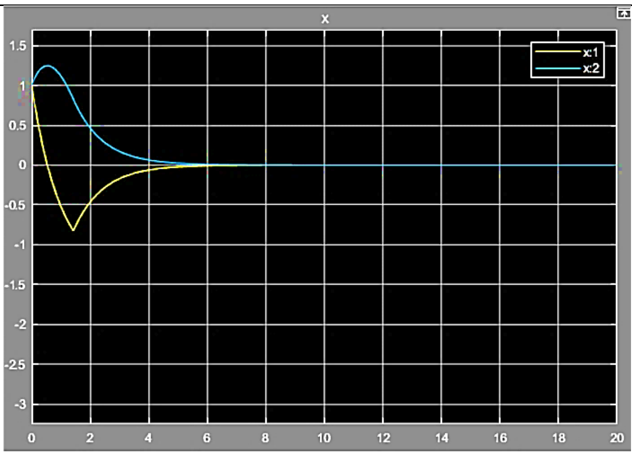
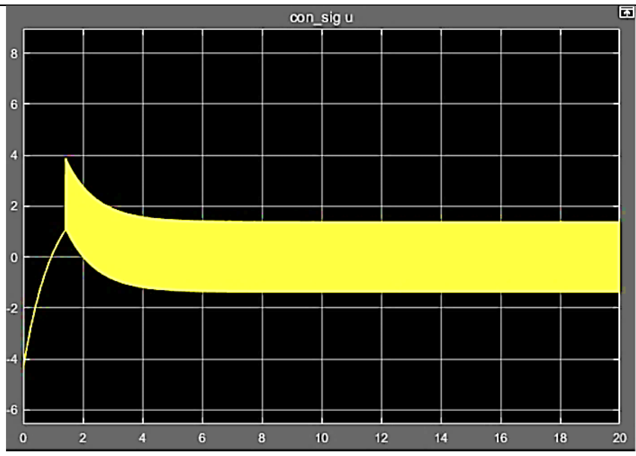
B. Simulation Results of Signed-Sigma Sliding Control

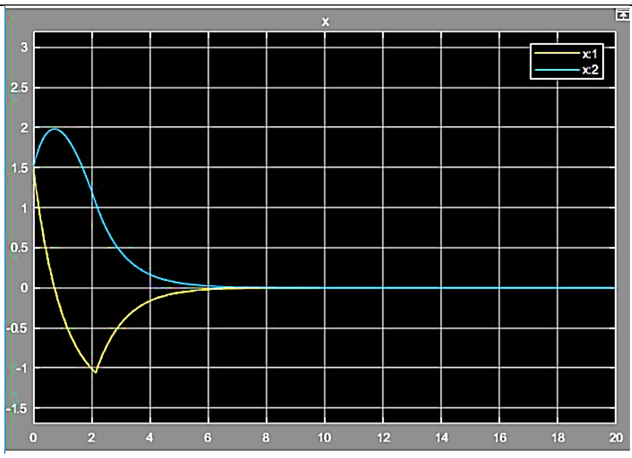
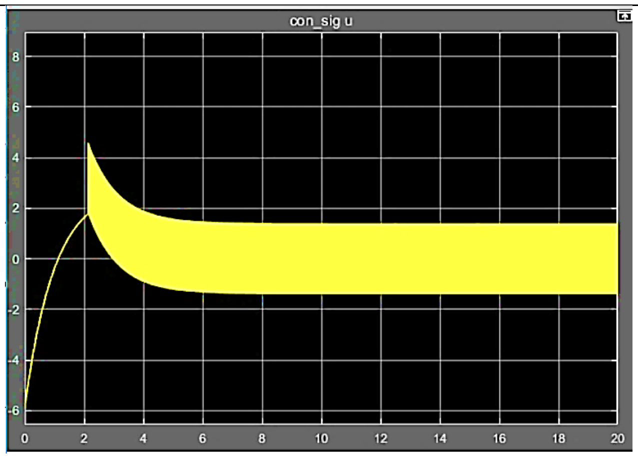
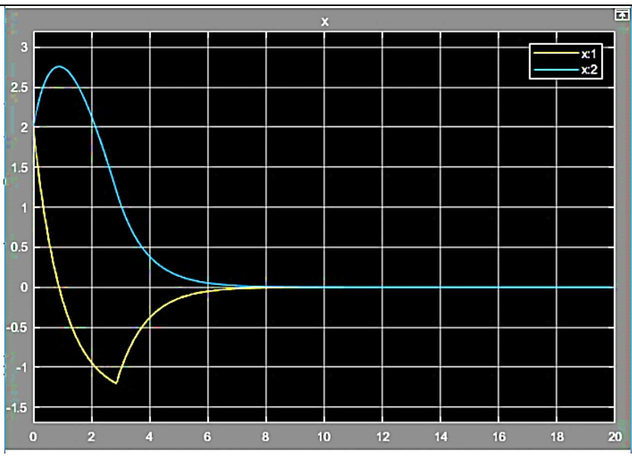
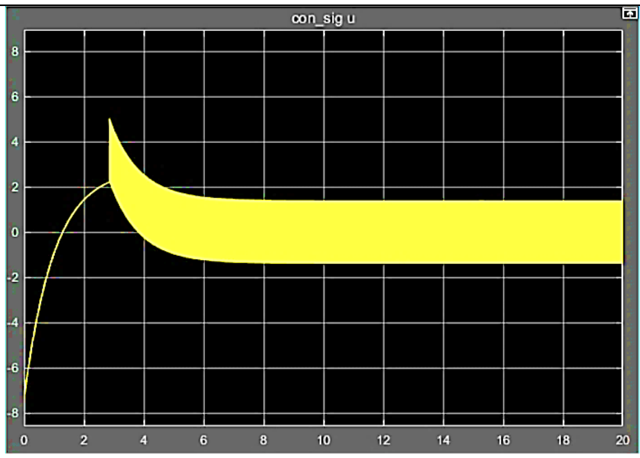
Various initial conditions given below are tested:

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} -2 & -1.5 & -1 & -0.5 & 0.5 & 1 & 1.5 & 2 \\ -2 & -1.5 & -1 & -0.5 & 0.5 & 1 & 1.5 & 2 \end{bmatrix}$$

The control signal and state convergence are tabulated as follows:

<table>
<tr><th>Initial States</th><th>State Convergence to zero</th><th>Control Signal/Effort</th></tr>
<tr><td>$\begin{bmatrix} -2 \\ -2 \end{bmatrix}$</td><td>
</td><td>
</td></tr>
<tr><td>$\begin{bmatrix} -1.5 \\ -1.5 \end{bmatrix}$</td><td>
</td><td>
</td></tr>
<tr><td>$\begin{bmatrix} -1 \\ -1 \end{bmatrix}$</td><td>
</td><td>
</td></tr>
</table>

$\begin{bmatrix}-0.5\\-0.5\end{bmatrix}$		
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$		
$\begin{bmatrix}1\\1\end{bmatrix}$		

$\begin{bmatrix}1.5\\1.5\end{bmatrix}$		
$\begin{bmatrix}2\\2\end{bmatrix}$		

Note that the transient time t_σ for this case is given below:

$$\dot{V} = \sigma\dot{\sigma} = -\mu\sigma\text{sign}(\sigma) - \sigma\text{sign}(\sigma)\Gamma^T D(d_{max} - d); \mu > 0$$

$$\dot{\sigma} = -\mu\text{sign}(\sigma) - \text{sign}(\sigma)\Gamma^T D(d_{max} - d)$$

Integrating both sides,

$$-\sigma_o = -\text{sign}(\sigma)(\mu + \Gamma^T D d_{max})t_\sigma + \text{sign}(\sigma)\frac{\Gamma^T D d_{max}}{628}\cos(628t)$$

The term $\text{sign}(\sigma)\frac{\Gamma^T D d_{max}}{628}\cos(628t)$ is small compared to earlier term since $d_{max} = 0.9$ and $\Gamma^T D = 1$. For $\sigma_o > 0$, it is noted that:

$$\sigma_o = (\mu + \Gamma^T D d_{max})t_\sigma$$

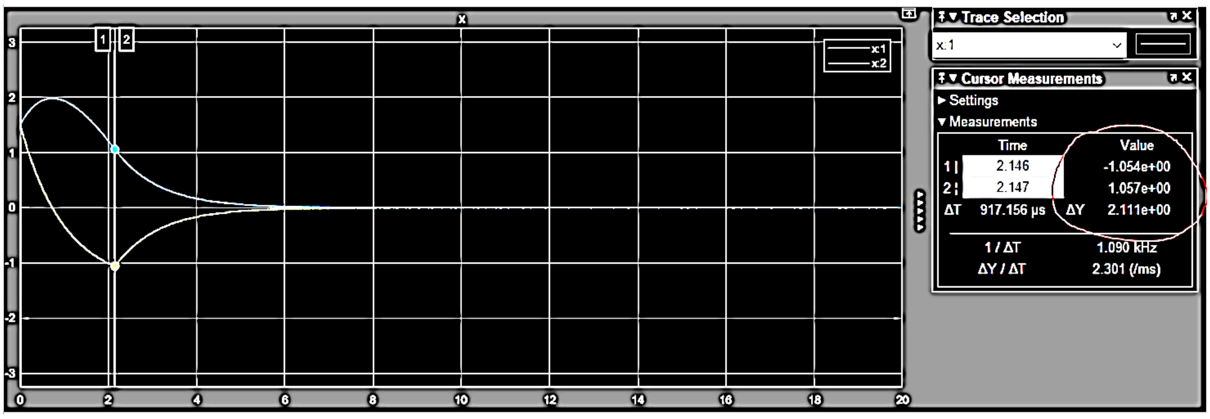
While for $\sigma_o > 0$,

$$-\sigma_o = (\mu + \Gamma^T D d_{max})t_\sigma$$

This leads to the approximated formula of transient time:

$$t_\sigma = \frac{|\sigma_o|}{\mu + \Gamma^T D d_{max}} = \frac{|\sigma_o|}{1.4} \text{ for } \mu = 0.5; \Gamma^T D = 1; d_{max} = 0.9$$

Seeing it other way, sliding surface is achieved when x_1 mirrors x_2 with respect to 0 line since $c_1 = c_2 = 1$. For example, the case of initial value $[1.5 \;\; 1.5]^T$ shown below:



At $t_\sigma = \frac{3}{1.4}$ =2.14 secs, $x_1 = -1.054$ and $x_2 = 1.057$, implies sliding surface is almost reached since x_1 mirrors x_2.

Additionally, shown in the portion of control signals, the chattering takes effect when σ is close to zero. The swing amplitude of chattering is closely the same as the swinging of disturbance that is modelled with sinusoidal value with amplitude 0.9. To reduce of relay chattering effect, we can introduce smooth control law as elaborated in Chapter 1 that makes use of saturation limit for σ under some limit saturation ε.

On the other hand, phase portraits of state convergence are given in the following figure.

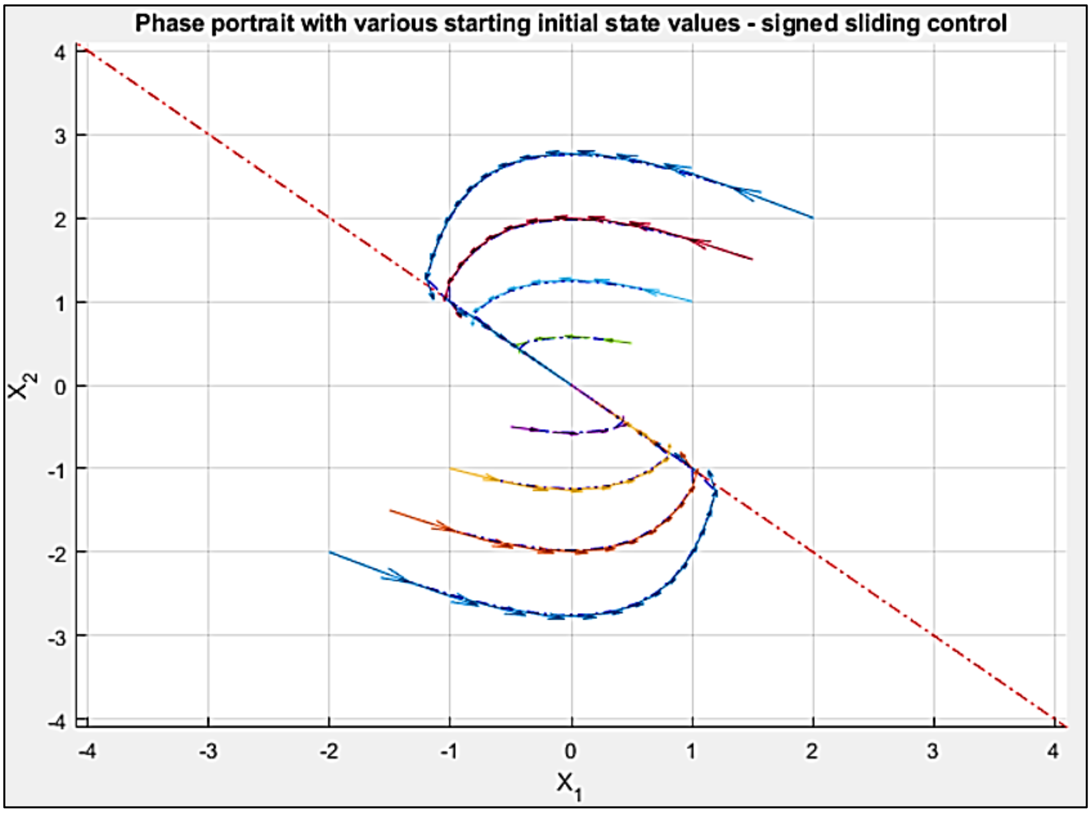


Figure 3 Phase portraits of State Convergence with various initial state values

Once the phase trajectory reaches the sliding surface indicated by diagonally dashed line in the figure above, the movement of phase trajectory will be continued along the sliding surface line and towards the origin (0,0) as time increases. This is in alignment with theoretical conclusion explained in Chapter 1 where $\sigma \rightarrow 0$ as $t \rightarrow \infty$, implying that $x \rightarrow 0$ since polynomial σ is of a stable polynomial type.

C. Simulation of Saturated Sigma Sliding Control (Smooth Control Law)

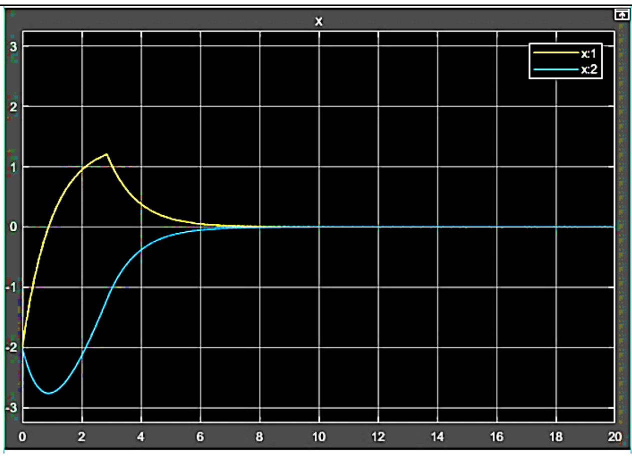
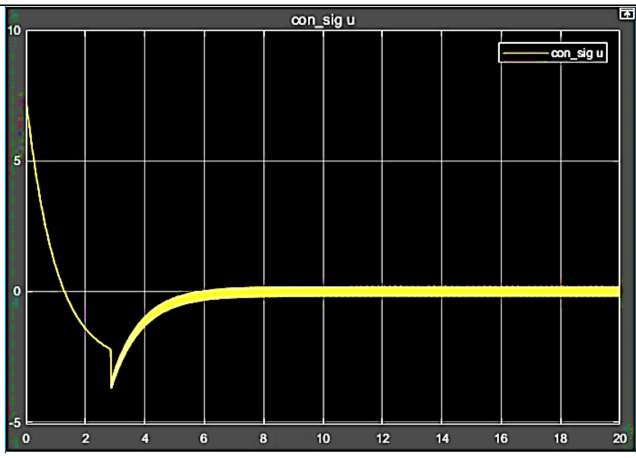
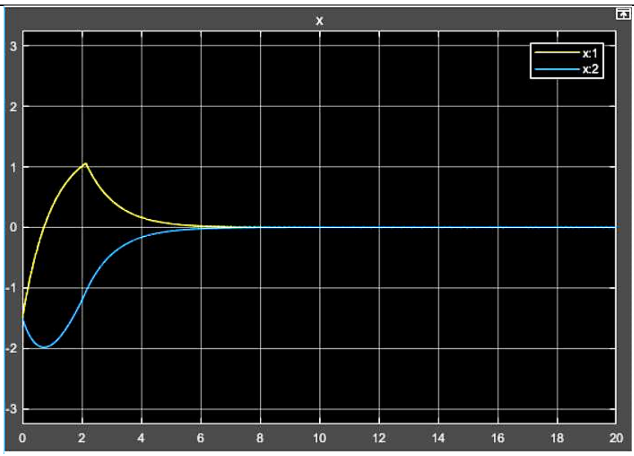
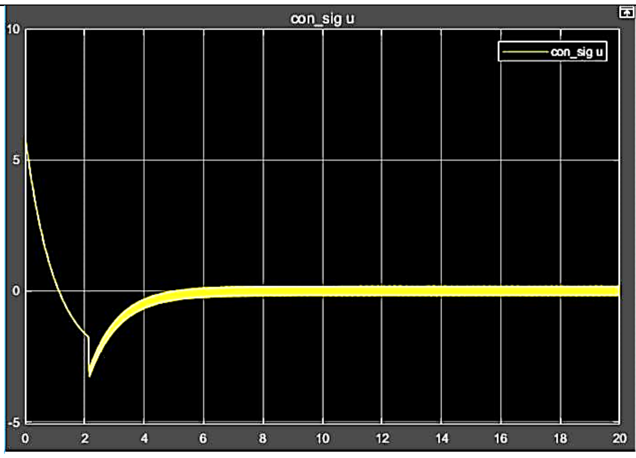
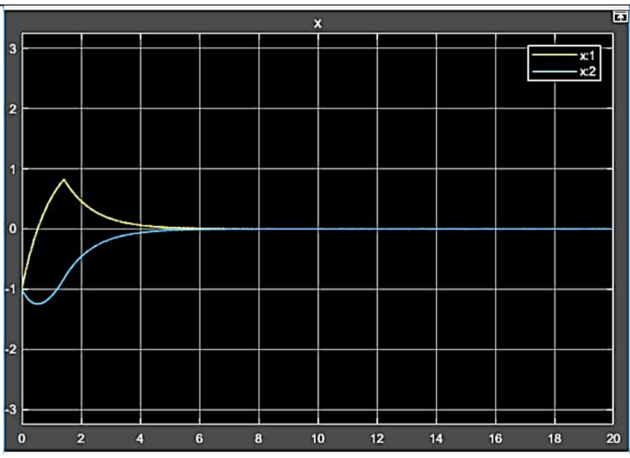
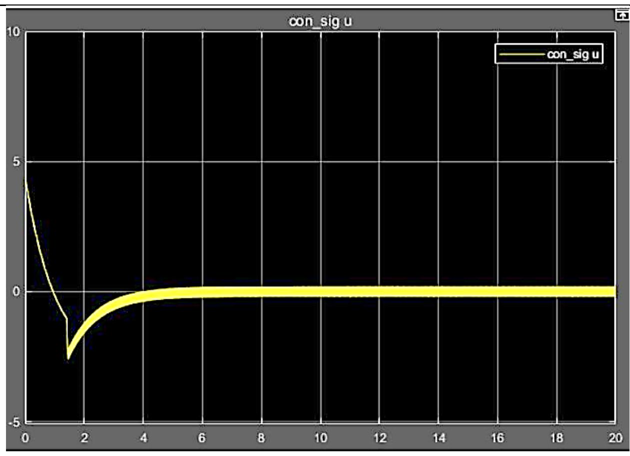
The formulation of control signal is slightly different by involving saturation term instead of sign-term. Mathematically, the control law is expressed as follow:

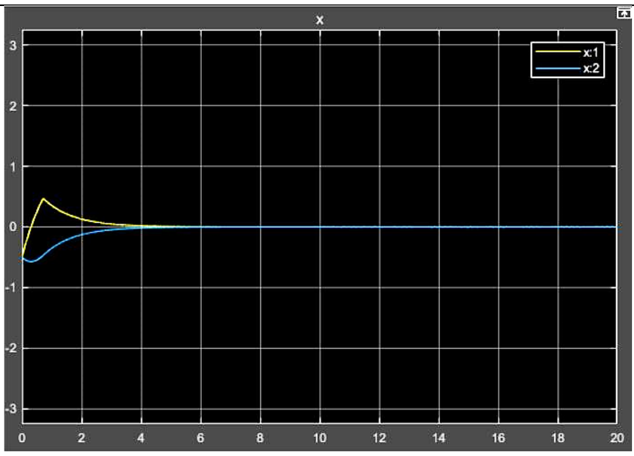
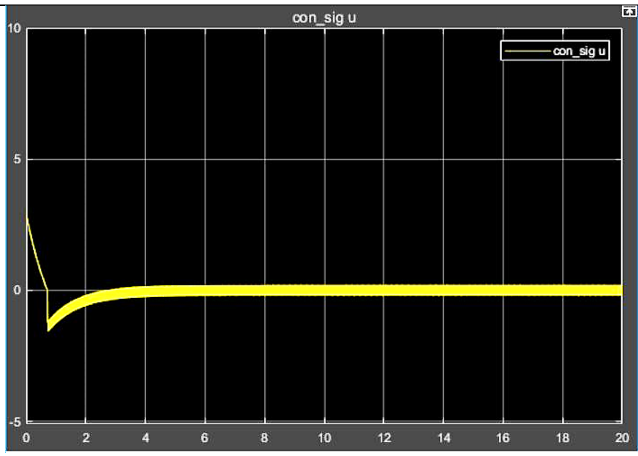
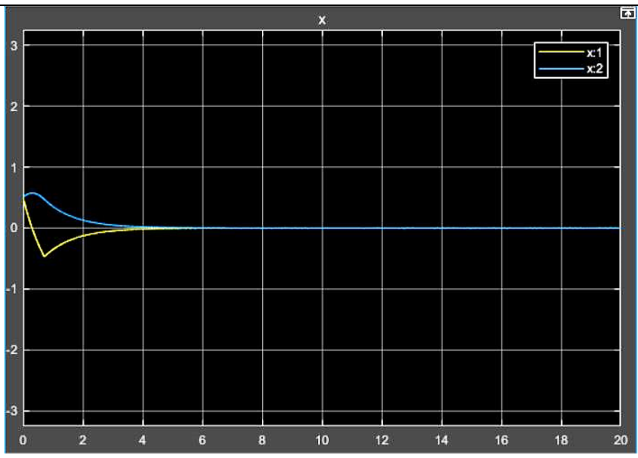
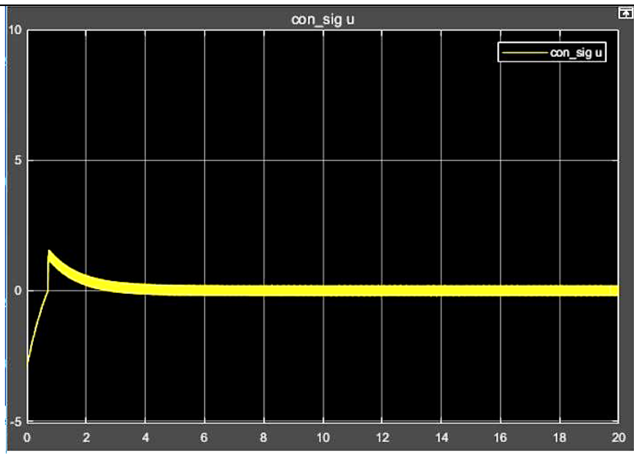
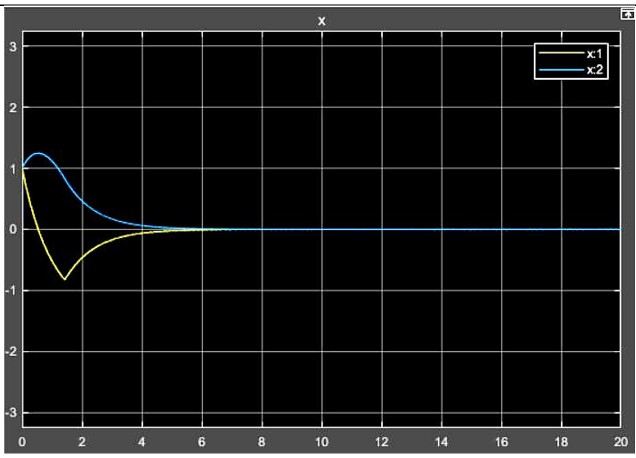
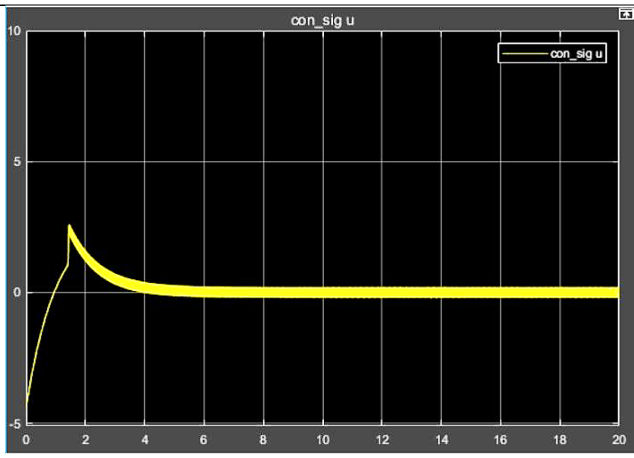
$$u = \frac{-\Gamma^T Ax - \mu sat(\sigma, \varepsilon) - sat(\sigma, \varepsilon)\Gamma^T Dd_{max}}{\Gamma^T B}$$

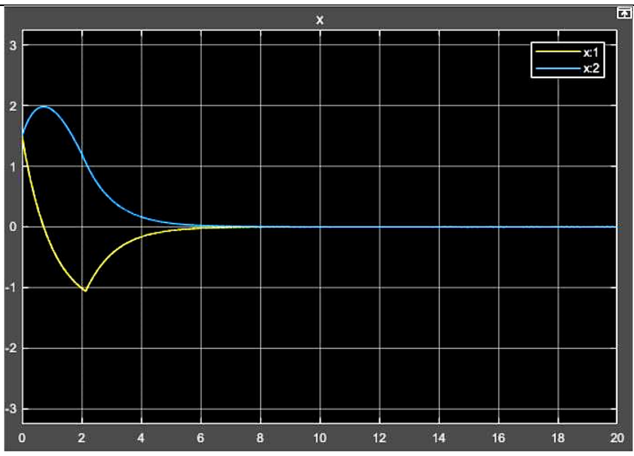
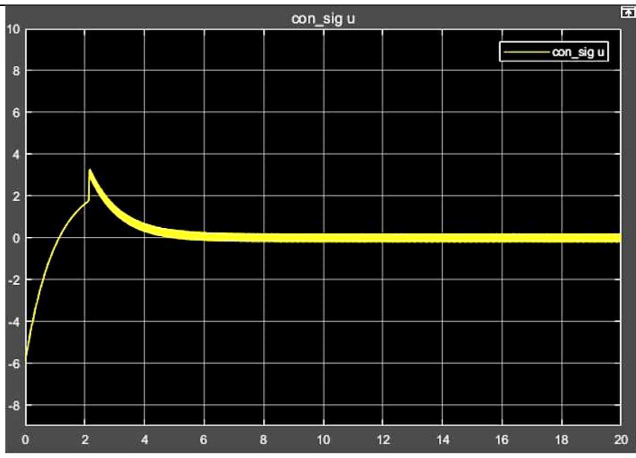
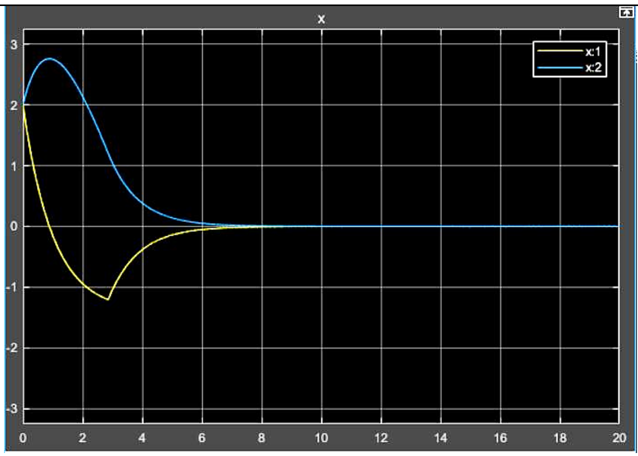
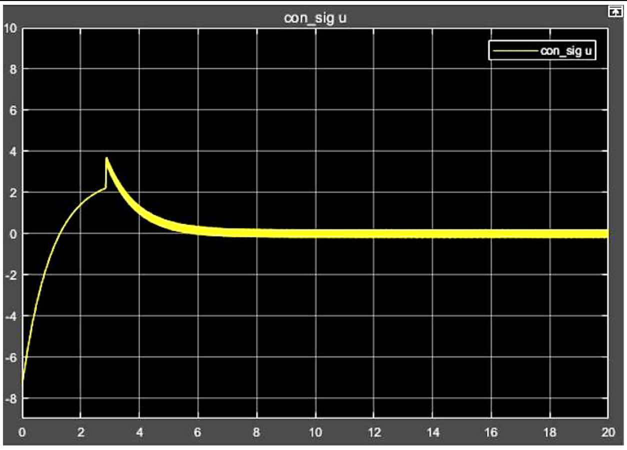
as what has been derived in earlier chapter.

The saturation function ensures that when σ is close to zero, it is then amplified by a factor of $1/\varepsilon$ to ensure that zero-crossing is steep enough to introduce less chattering control signal. Therefore, smooth control law enables the control signal to suffer less chattering compared to control signal with sign function in its formulation. To test, the previous set of initial condition are set again as initial state values in this simulation.

The result is tabulated as follows:

<table>
<tr><th>Initial States</th><th>State Convergence to zero</th><th>Control Signal/Effort</th></tr>
<tr><td>$\begin{bmatrix}-2\\-2\end{bmatrix}$</td><td>
</td><td>
</td></tr>
<tr><td>$\begin{bmatrix}-1.5\\-1.5\end{bmatrix}$</td><td>
</td><td>
</td></tr>
<tr><td>$\begin{bmatrix}-1\\-1\end{bmatrix}$</td><td>
</td><td>
</td></tr>
</table>

$\begin{bmatrix}-0.5\\-0.5\end{bmatrix}$		
$\begin{bmatrix}0.5\\0.5\end{bmatrix}$		
$\begin{bmatrix}1\\1\end{bmatrix}$		

$\begin{bmatrix}1.5\\1.5\end{bmatrix}$		
$\begin{bmatrix}2\\2\end{bmatrix}$		

The visually obvious difference between saturated-sigma (smooth control law) and signed-sigma control law is that the control signals have less chattering when saturated sigma control law is implemented. While the impact of saturated sigma function is less noticeable in state value convergence. The transient time of state values to reach sliding surface is the same as obtained in signed-sigma control law, since the ε window is small such that the value of:

$$-\sigma_o = -\mathrm{sat}(\sigma_o)(\mu + \Gamma^{T} D d_{max}) t_\sigma$$

is almost equivalent to:

$$-\sigma_o = -\mathrm{sign}(\sigma)(\mu + \Gamma^{T} D d_{max}) t_\sigma$$

From phase portrait point of view, the saturation function will actually smoothen the movement of phase trajectory along the sliding surface. Figures below provide comparison of phase trajectories using signed-sigma sliding control and saturated-sigma sliding control.

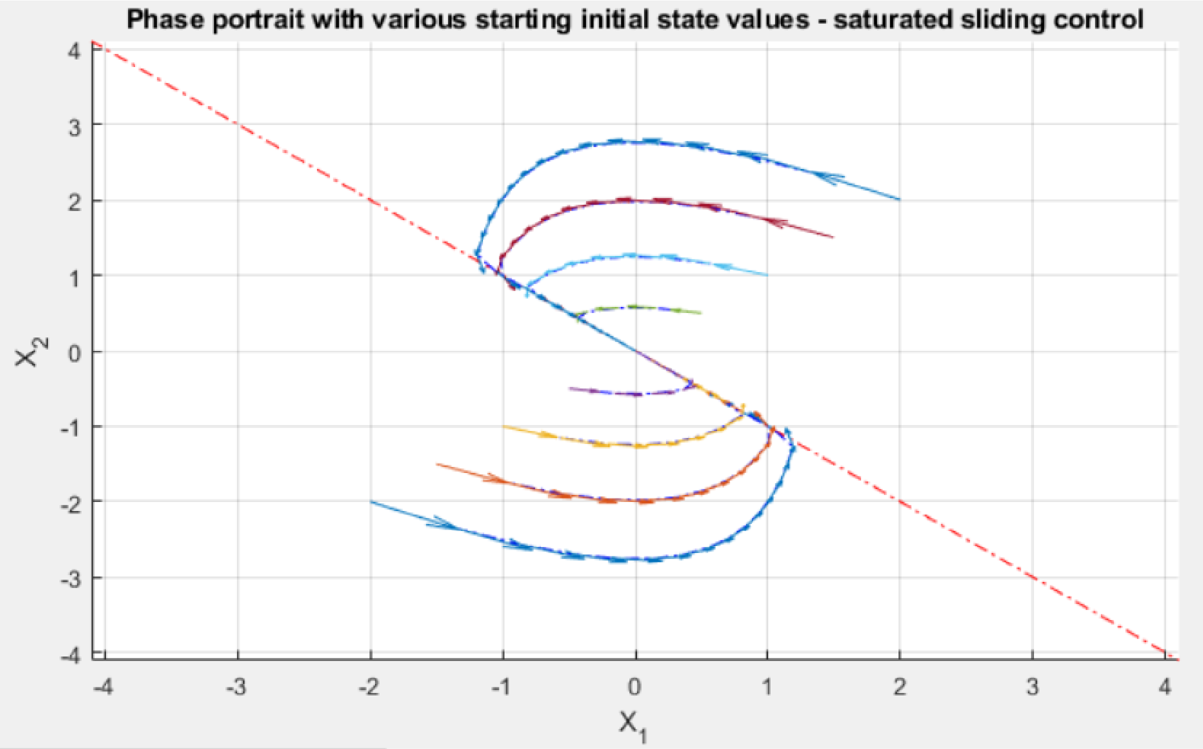


Figure 4 Phase portrait with different initial state values for saturated-sigma sliding control

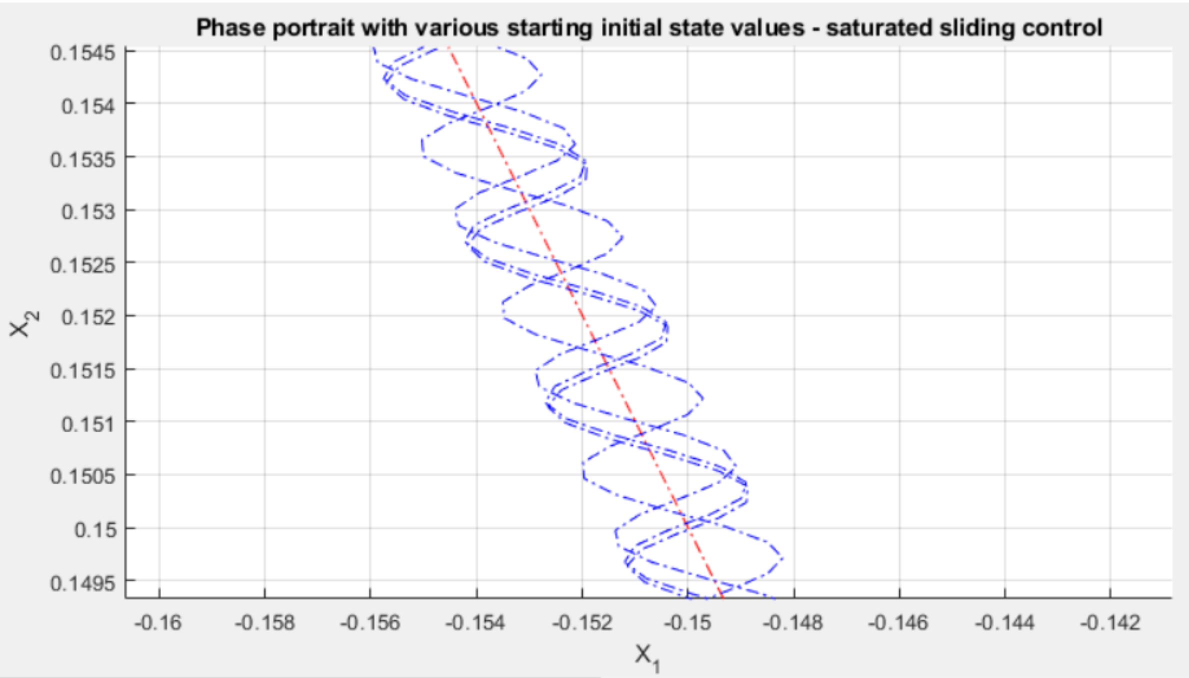


Figure 5 Phase movement trajectory along sliding surface with saturated sliding control

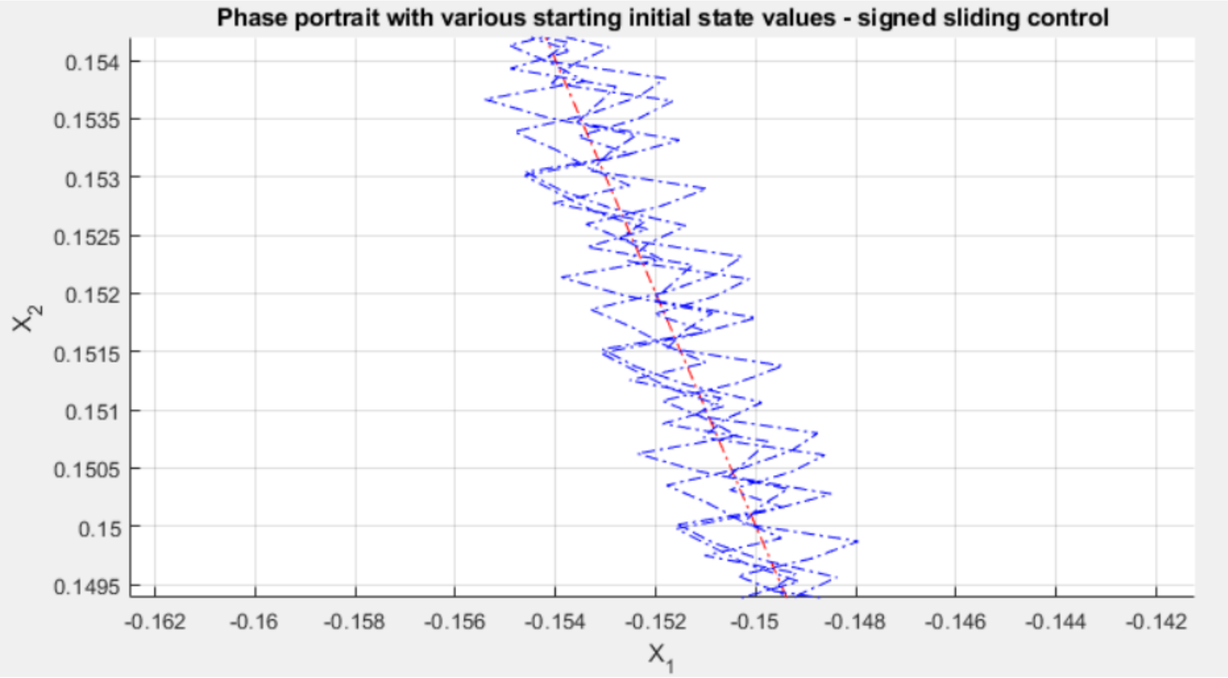


Figure 6 The phase trajectory along sliding surface with signed sliding control

III. Appendix: Code and Model

A. Sliding Control Code

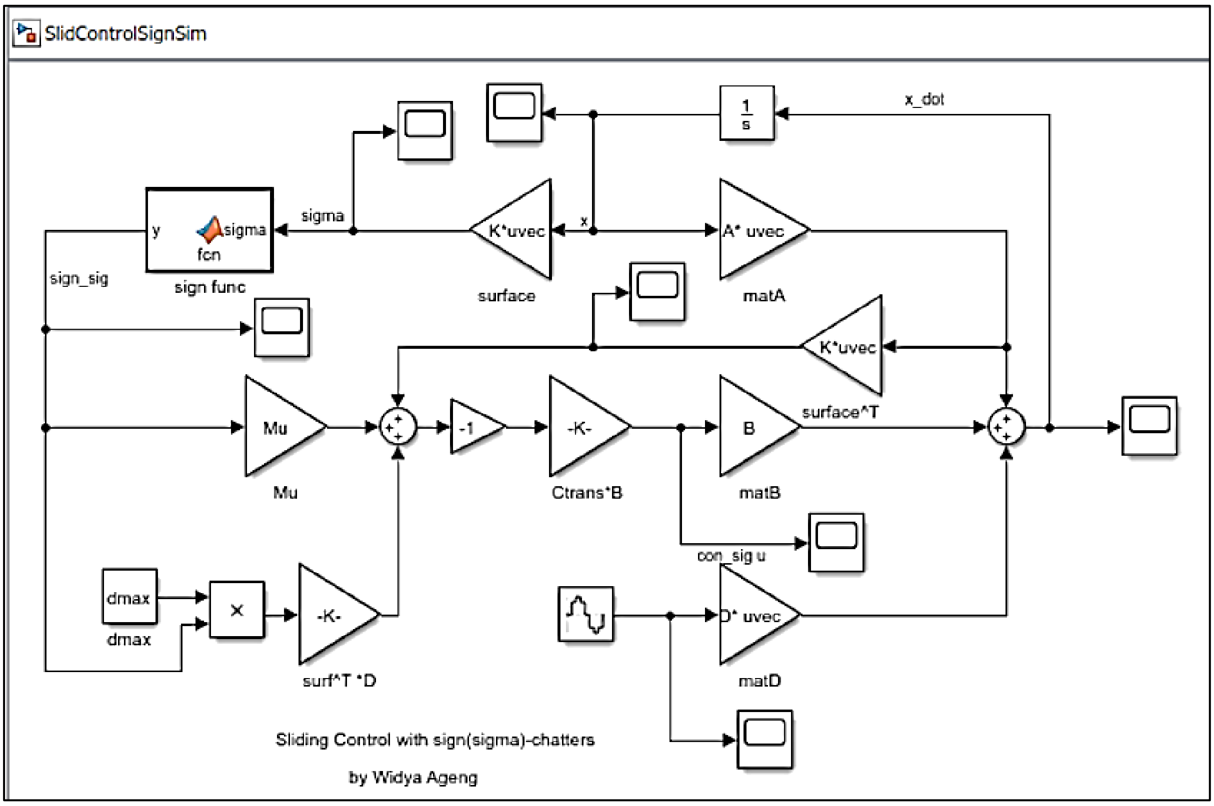
```
%%-------------------------
%----- SLIDING CONTROL------
%%-------------------------
clear;clc;
%% State space of system
a = 2;
b = 1;
c1 = 1;
c2 = 1;
Mu = 0.5;
t = (0:0.01:10)';
dmax = 0.9;
d = dmax*sin(628*t);

% initial state values
init_cond = [1 1]';

A = [a 0;1 0];
B = [b;0];
D = [1;0];
Cy = [1 0;0 1];

%% Sliding surface polynomial coefficients
surf_C = [c1;c2];

% surface stability check
if (-1)*surf_C(2)/surf_C(1) < 0
    disp('Stable');
else
    disp('Roots of surface polynomial is on closed right plane,
unstable/marginally stable');
end

%% Plot of Phase Portrait and surface line with sliding control (signed
function)
surfline = [(-1)*(-10:0.1:10)' (-10:0.1:10)'];
figure;
hold on;
plot(surfline(:,1),surfline(:,2),'-.r');
time_inc = 1e-3;

% pools of initial state values whose convergence towards zero to be
% investigated and plotted in phase portrait plane
x1_init = [-2 -1.5 -1 -0.5 0.5 1 1.5 2];
x2_init = x1_init;

for i=1:length(x1_init)
    x = [x1_init(i);x2_init(i)];
    xdot = x;
    surf_val = surf_C'*x;
    x_traj = x';
    t = 0;

    % looping to terminate the dynamics after reaching steady state
    % condition/reaching x-->0 close to zero, ie : [x1 x2] ~ [0 0]
```

```
    while sum(abs(x))>1e-3
        t = t + time_inc;
        u_con = (-1)*(surf_C'*A*x +
Mu*sign(surf_val)+sign(surf_val)*surf_C'*D*dmax);
        u_con = u_con/(surf_C'*B);

        %state space and difference equation
        xdot = A*x + B*u_con + D*dmax*sin(628*t);
        x = x + xdot*time_inc;

        %evolution/movement of state values
        x_traj = [x_traj;x'];

        %movement of sliding polynomial values, ie: p1x1+p2x2
        surf_val = surf_C'*x;
        sumtrack = sum(abs(x)) % tracking convergence of state values
towards zero/below acceptable small value (epsilon)
    end

    space = 200; %trajectory spacing to not dense the phase portrait arrows

    %getting direction of convergence
    p1 = x_traj(:,1);
    p2 = x_traj(:,2);
    pquiv1 = p1(1:space:end);
    pquiv2 = p2(1:space:end);
    dpquiv1 = gradient(pquiv1);
    dpquiv2 = gradient(pquiv2);


    plot(pquiv1,pquiv2,'-.b');

quiver(pquiv1,pquiv2,dpquiv1,dpquiv2,'AutoScaleFactor',1.3,'MaxHeadSize',
1);
end
hold off;
grid;
axis([2*min(x1_init)-0.1 2*max(x1_init)+0.1 2*min(x1_init)-0.1
2*max(x1_init)+0.1]);
xlabel('X_1');
ylabel('X_2');
title('Phase portrait with various starting initial state values - signed
sliding control');

%% Plot of Phase Portrait and surface line with smooth sliding control
(saturated function)
surfline = [(-1)*(-10:0.1:10)' (-10:0.1:10)'];
figure;
hold on;
plot(surfline(:,1),surfline(:,2),'-.r');
time_inc = 1e-3;
epsilon = 0.01;

% pools of initial state values whose convergence towards zero to be
% investigated and plotted in phase portrait plane
x1_init = [-2 -1.5 -1 -0.5 0.5 1 1.5 2];
x2_init = x1_init;

for i=1:length(x1_init)
```

```
    x = [x1_init(i);x2_init(i)];
    xdot = x;
    surf_val = surf_C'*x;
    x_traj = x';
    t = 0;

    % looping to terminate the dynamics after reaching steady state
    % condition/reaching x-->0 close to zero, ie : [x1 x2] ~ [0 0]
    while sum(abs(x))>1e-3
        t = t + time_inc;
        u_con = (-1)*(surf_C'*A*x +
Mu*sat(surf_C,x,epsilon)+sat(surf_C,x,epsilon)*surf_C'*D*dmax);
        u_con = u_con/(surf_C'*B);

        %state space and difference equation
        xdot = A*x + B*u_con + D*dmax*sin(628*t);
        x = x + xdot*time_inc;

        %evolution/movement of state values
        x_traj = [x_traj;x'];

        %movement of sliding polynomial values, ie: p1x1+p2x2
        surf_val = surf_C'*x;
        sumtrack = sum(abs(x)) % tracking convergence of state values
towards zero/below acceptable small value (epsilon)
    end

    space = 200; %trajectory spacing to not dense the phase portrait arrows

    %getting direction of convergence
    p1 = x_traj(:,1);
    p2 = x_traj(:,2);
    pquiv1 = p1(1:space:end);
    pquiv2 = p2(1:space:end);
    dpquiv1 = gradient(pquiv1);
    dpquiv2 = gradient(pquiv2);

    plot(pquiv1,pquiv2,'-.b');

quiver(pquiv1,pquiv2,dpquiv1,dpquiv2,'AutoScaleFactor',1.3,'MaxHeadSize',
1);
end
hold off;
grid;
axis([2*min(x1_init)-0.1 2*max(x1_init)+0.1 2*min(x1_init)-0.1
2*max(x1_init)+0.1]);
xlabel('X_1');
ylabel('X_2');
title('Phase portrait with various starting initial state values -
saturated sliding control');

%% modularized function for surface saturation
function satval = sat(surfcoeff,state,epsilon)
    if surfcoeff'*state>epsilon
        satval = 1;
    elseif and((surfcoeff'*state>=(-
1)*epsilon),(surfcoeff'*state<=epsilon))
        satval = surfcoeff'*state/epsilon;
    else
```

```
        satval = -1;
    end
end
```

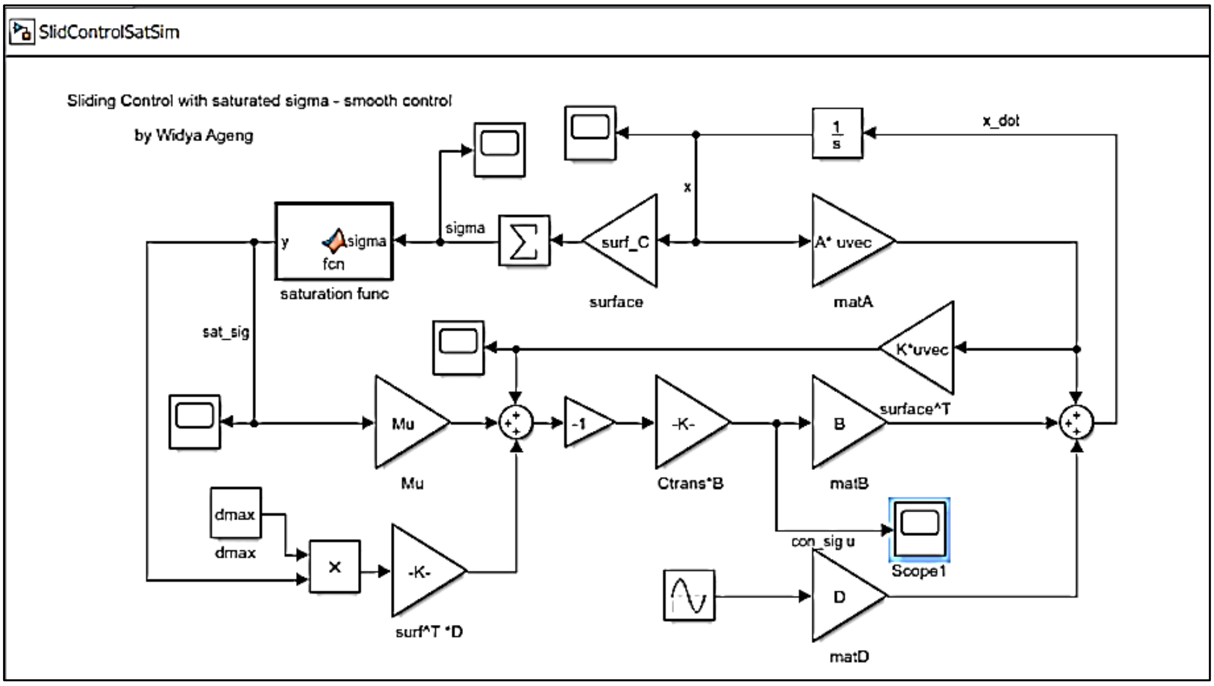
B. Simulink Model of Sliding Control with Signed-Sigma Control Law



Embedded function in 'sign func':

```
function y = fcn(sigma)

epsilon = 0.01;

if sigma>0
    out = 1;
else
    out = -1;
end

y = out;
```

C. Simulink Model of Sliding Control with Saturated-Sigma Control Law



Embedded function in 'saturation func':

```
function y = fcn(sigma)

epsilon = 0.01;

if sigma>epsilon
    out = 1;
elseif sigma<(-1)*epsilon
    out = -1;
else
    out = sigma/epsilon;
end

y = out;
```