Super Twisting Implementation for Quadcopter control

Table des matières

Introduction: 1
System modeling: 1
Motor Modeling 1
Propeller Modeling: 2
System Dynamics 3
Nonlinear System Control: 4
Lyapunov stability criteria: 4
Sliding Mode Control 5
Super Twisting Sliding Mode Control 8
Deploy the algorithm to a Micro-controller Board 10

Introduction:

To test the algorithm, we will create an imaginary system based on real components (Technical specification from datasheets). And use the model of system to design, at first stage a classical sliding mode controller, and in a second stage, a Super-twisting-based sliding mode controller.

We will begin by creating a simulation for the system and the controller, then we will implement the controller on a microcontroller to control a quadcopter.

System modeling:

Motor Modeling

Let consider a DC Motor Model 130

Specs:

Rated Power: 0.462 W

Rated Voltage: 6 V

Rated speed: 4500 rpm

No-Load Speed: 9100 rpm

All the technical detail can be found at SuperTwisting-UEXProject/DatasheetMotor.pdf at main · AnasTaherGit/SuperTwisting-UEXProject (github.com)

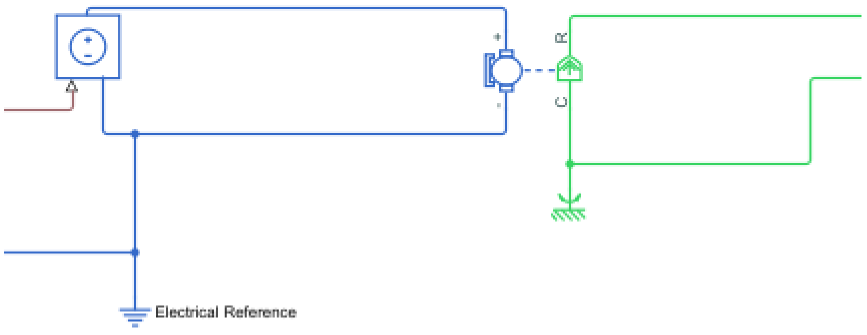


Figure 1: Simscape DC motor model

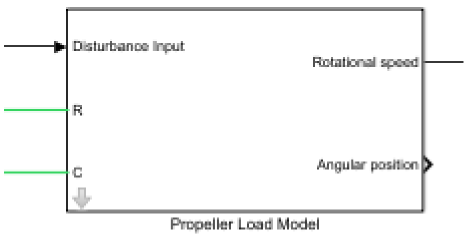
Propeller Modeling:

Propellers are considered loads of Category A, which mean that they satisfy the following equation:

$$T = k.\omega^2$$

For the propeller modeling, we will consider that the propeller satisfies the operating condition at Rated speed.

$$P = k.\omega^3 \rightarrow k = \frac{P}{w^3} = 4.414877e - 9 \quad W.s^3$$



So, the model of the propeller in Simscape, consists of a Controlled torque source. The rotational speed is squared and multiplied by the characteristic factor "k" of the propeller and injected as input for the Controlled torque source.

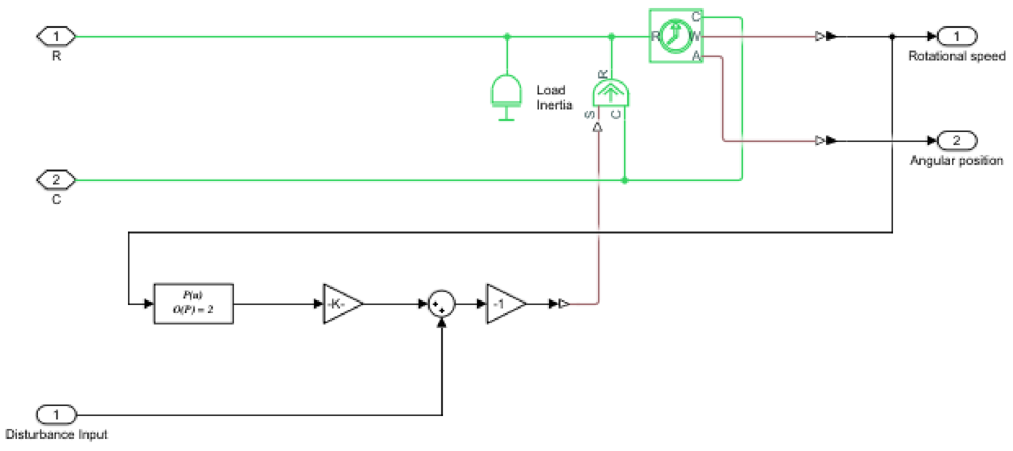


Figure 2: Propeller Model - Simulink

System Dynamics

According to Newton 2nd law of Rotation:

$$J\dot{\omega} = T_{motor} - T_{propeller}$$

And The model of the motor and the propeller:

$$\begin{cases} T_{motor} = -\dfrac{K_E}{rK_I}\omega + \dfrac{1}{rK_I}U \\ T_{propeller} = k.\omega^2 \end{cases}$$

$where\ K_E\ is\ the\ voltage\ (speed) constant\ and\ K_I\ is\ the\ current\ (torque) constant.$

$r\ the\ internal\ resistance\ and\ U\ the\ input\ voltage.$

So, the state space model of the system become:

$$\dot{\omega} = -\frac{k}{J}\omega^2 - \frac{K_E}{JrK_I}\omega + \frac{1}{JrK_I}U = f(\omega, U)$$

Numeric Application:

$$\dot{\omega} = -0.08653\omega^2 - 39.902\omega + 6338.157\ U$$

We can observe that the model is nonlinear state space model. Therefore, the classical linear controller will not work for this type of system.

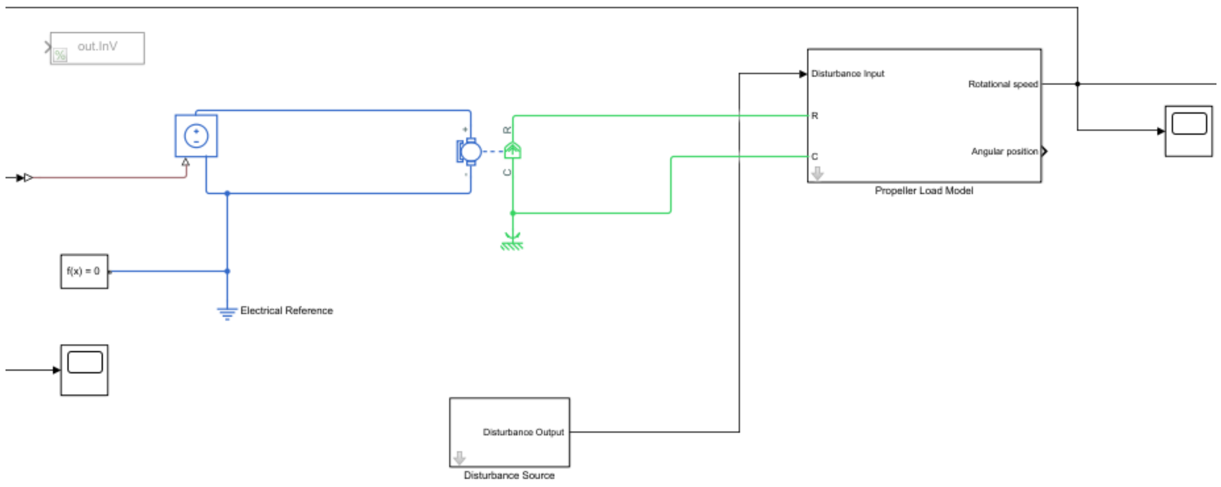


Figure 3:Motor-Propeller system model

Nonlinear System Control:

A nonlinear system is a system whom his dynamic behavior equation can be written on the form:

$$\dot{x} = f(x,u)$$

Where “x” is the space state vector, “u” is the control input, and f is a nonlinear function.

Lyapunov stability criteria:

Let us consider a dynamic system and its given model: $\dot{x} = f(x)$ and x=0 is the equilibrium point of the system.

The system is stable if it is moved from his equilibrium point to a given position, it will evolve to the equilibrium point and stay there.

We should point out that a given system can have multiple equilibrium points, so the system may be stable for some equilibrium points and may be not for others. Then the stability is not property of the system but rather is intrinsic property of the equilibrium point.

the Lyapunov stability criterion is a criterion which allows to perform a stability analysis of a given equilibrium point.

The criterion consists in the choice of a function called Lyapunov function or “Energy-like” function.

A function V should verify the following conditions to be considered as Lyapunov function:

1. V is C^1 $function$
2. V should be positive-definite (i.e., V>0 and V(0)=0)
3. $\dot{V} < 0$ and $\dot{V}(0) = 0$

Theorem:

If it exists a Lyapunov function for x=0, then x=0 (the equilibrium point) is stable.

Example:

Given a nonlinear system $\dot{x} = -x^3$ and x=0 his equilibrium point.

Let us choose a Lyapunov function to perform the analysis. The chosen function should verify the criteria above.

For example, let us consider the function $V(x) = \frac{1}{2}x^2$

1. This function is definitely $a\ C^1\ function$
2. V(x)>0 and V (0) =0
3. $\dot{V}(x) = x\dot{x} = -x^4 < 0\ and\ \dot{V}(0) = 0$

Then the Lyapunov function exist for this equilibrium point and according to the theorem above this equilibrium point is stable.

Sliding Mode Control

The Sliding mode controller (SMC) can be considered as the natural result of Lyapunov Stability Method. This algorithm consists of choosing a function S(x) called a “Sliding Surface”. The objective of the algorithm is to drive the system so that the sliding surface become zero and at the same time driving the state space vector to desired state.

So, we actually want to have a stable state at $S(x_{desired}) = 0$

To achieve it we must define a control signal that can verify the Lyapunov stability criterion for $S(x_{desired}) = 0.$

Let’s define first Lyapunov function.

$$V(S) = \frac{1}{2}S^2$$

The chosen Lyapunov function is a function of Class 1. And V(S)>0 and V (0)=0

So, the remaining condition is to have $\dot{V}(S) < 0\ and\ \dot{V}(0) = 0.$

To achieve stability, we must define the control input u so we can have $\dot{V} < 0$

$$\dot{V} = S\dot{S}$$

Let us consider a system defined by: $\dot{x} = f(x) + U$

And its sliding surface is $S = x - x_{desired}$

$$\dot{V} = S(\dot{x}) = S(f(x) + U)$$

So, we need to choose U in a way that we can have $\dot{V} < 0$

For example, $U = -f(x) - \eta\, sign(S)$

$$\dot{V} = -\eta . sign(S) . S = -\eta |S| < 0$$

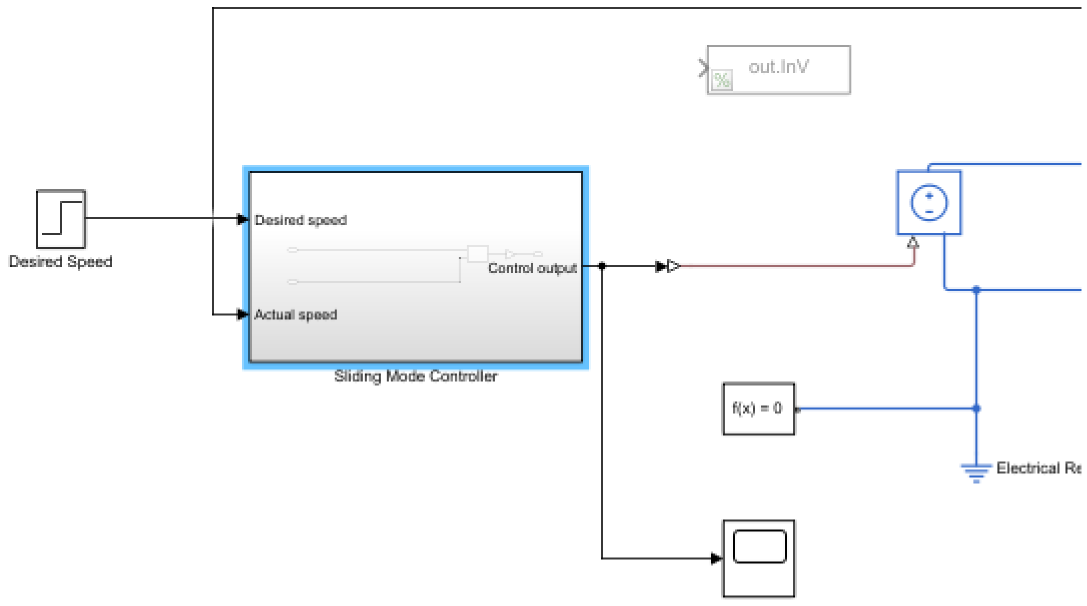
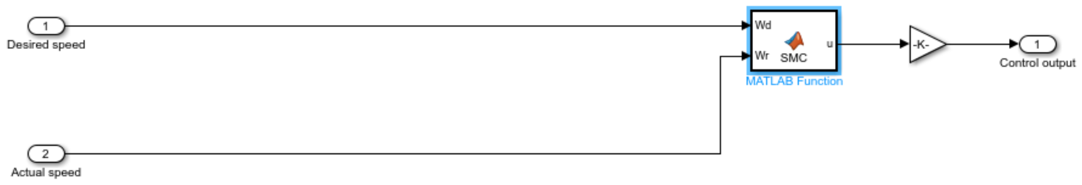
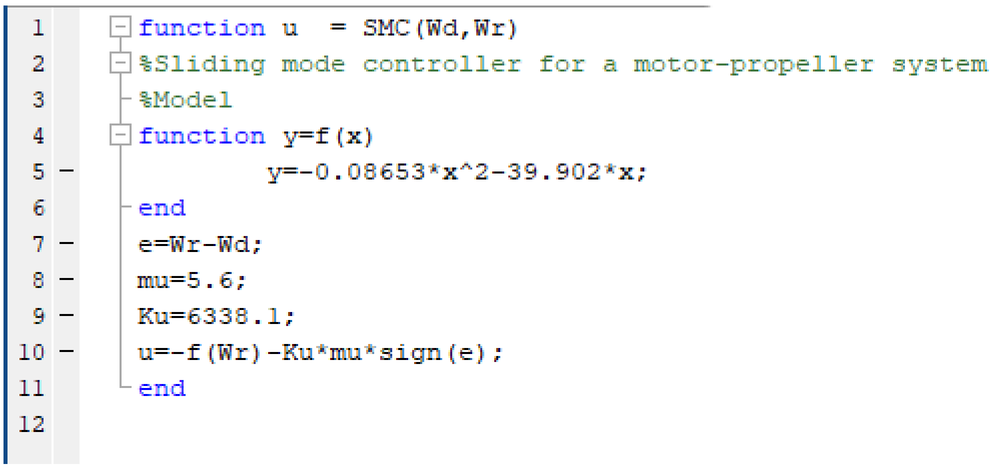


Figure 4: Sliding Mode Controller

The sliding mode controller is implemented via a user-defined function.



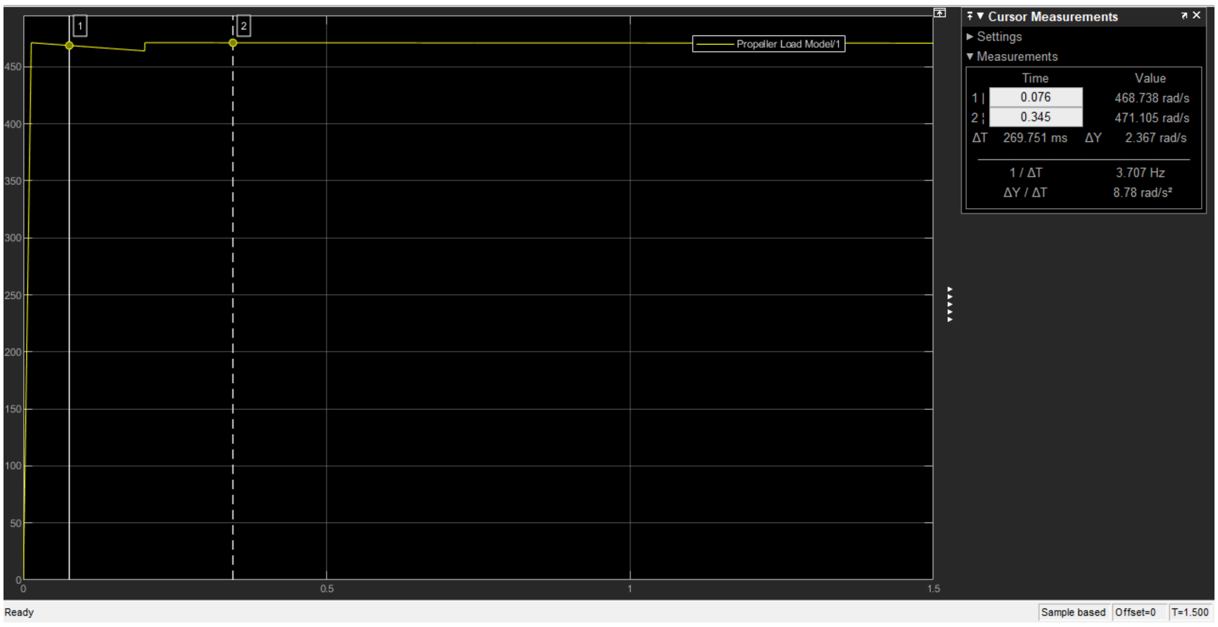


```
function u  = SMC(Wd,Wr)
%Sliding mode controller for a motor-propeller system
%Model
function y=f(x)
        y=-0.08653*x^2-39.902*x;
end
e=Wr-Wd;
mu=5.6;
Ku=6338.1;
u=-f(Wr)-Ku*mu*sign(e);
end
```

➔ Result of Simulation

The simulation scenario is to set a desired speed at 471.15 rd/s and from 0.2 s we introduce a sinusoidal disturbance torque signal.

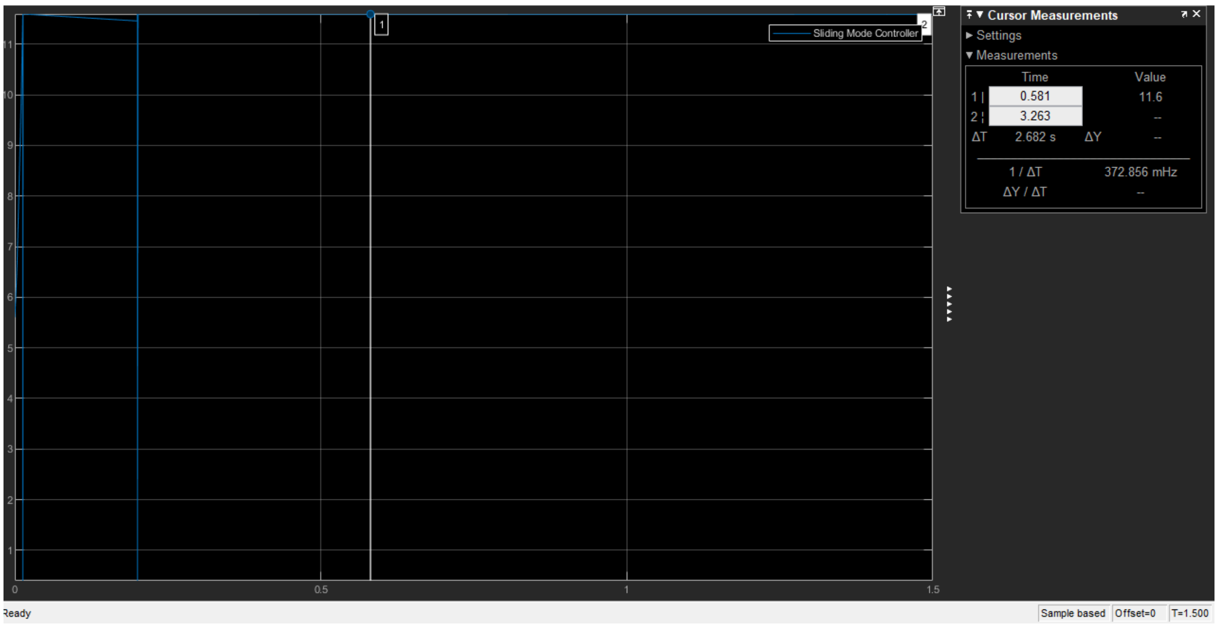
Here are the results:



Cursor Measurements
▸ Settings
▾ Measurements

	Time	Value
1	0.076	468.738 rad/s
2	0.345	471.105 rad/s
ΔT	269.751 ms	ΔY 2.367 rad/s
1 / ΔT		3.707 Hz
ΔY / ΔT		8.78 rad/s²

Figure 5: Rotational speed under sinusoidal disturbance



Cursor Measurements
▸ Settings
▾ Measurements

	Time	Value
1	0.581	11.6
2	3.263	--
ΔT	2.682 s	ΔY --
1 / ΔT		372.856 mHz
ΔY / ΔT		--

Figure 6: Control signal under sinusoidal disturbance

Super Twisting Sliding Mode Control

Even if the classical Sliding Mode Control is good and robust algorithms, it can present some unwanted behavior, such as chattering. Chattering is the phenomena where high frequency oscillation occurs when the system reaches the sliding surface. Preventing this phenomenon requires infinitely fast switching components. Which is impossible to realize since all physical system have inertia.

The control input for Super-Twisting control technique is:

$$u = -f(\omega) - \mu . |S|^{1/2} . sign(S) - \beta . \int_0^t sign(S) d\tau$$

As we can observe, this control technique is no longer discontinuous. Indeed, $sign(S) . |S|^{1/2}$ and $\int_0^t sign(S) d\tau$ are both continuous.

Let us implement this new technique on our simulation.

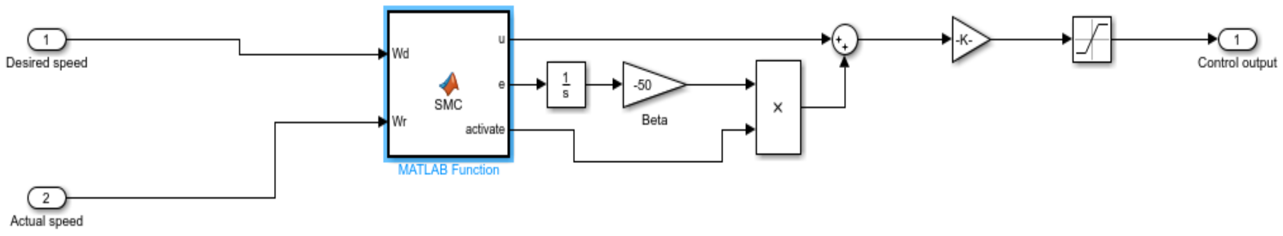
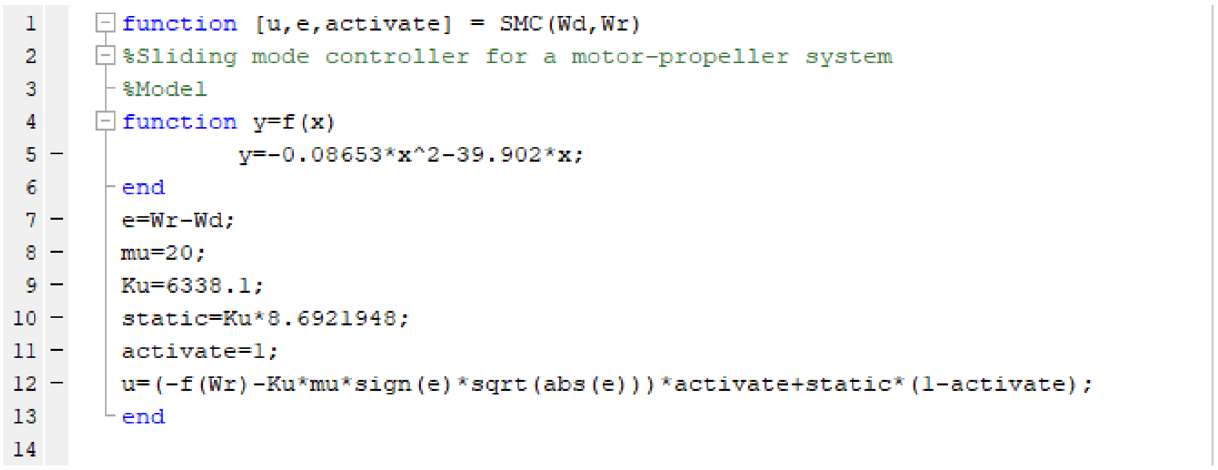


Figure 7:Super twisting Sliding mode controller

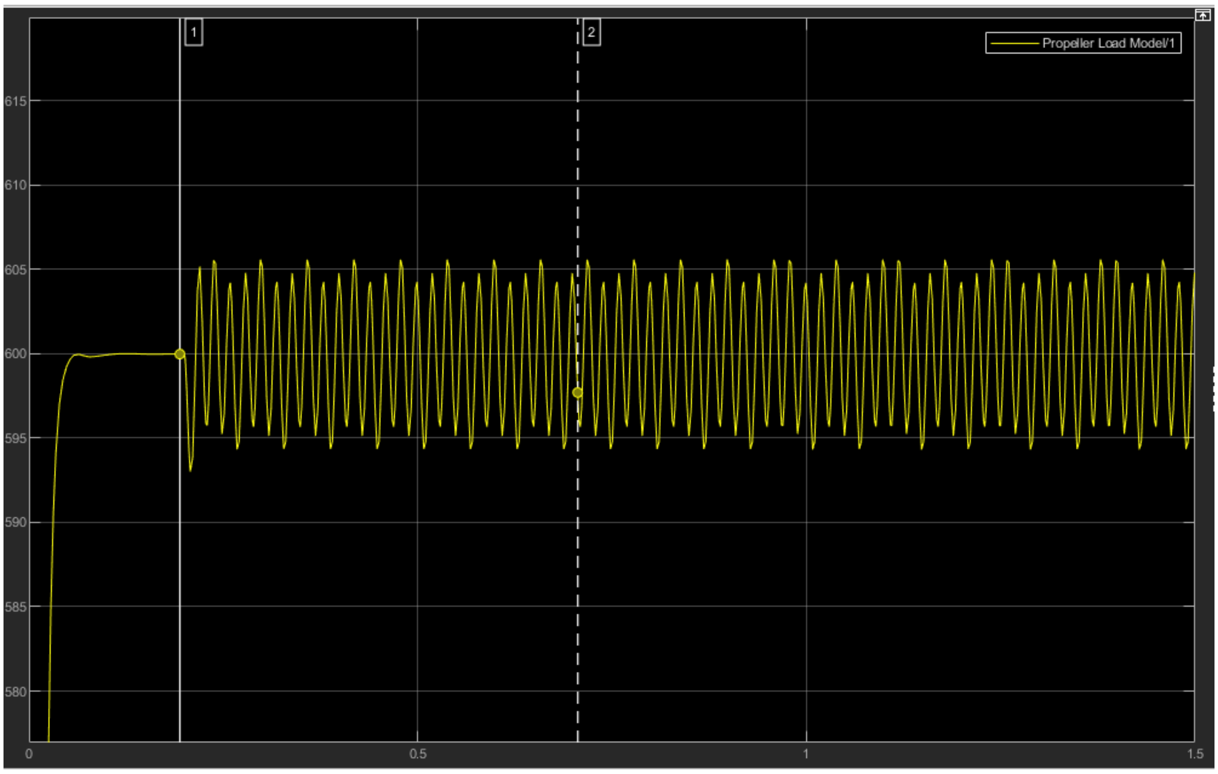


```
function [u,e,activate] = SMC(Wd,Wr)
%Sliding mode controller for a motor-propeller system
%Model
function y=f(x)
        y=-0.08653*x^2-39.902*x;
end
e=Wr-Wd;
mu=20;
Ku=6338.1;
static=Ku*8.6921948;
activate=1;
u=(-f(Wr)-Ku*mu*sign(e)*sqrt(abs(e)))*activate+static*(1-activate);
end
```

➔ **Simulation Results**

The simulation scenario is to set a desired speed at 600 rd/s and from 0.2 s we introduce a sinusoidal disturbance torque signal.

Let us see first the behavior of the system without the controller (under constant input).



We observe that the speed oscillates under the disturbance effect and reach an amplitude of 5 rd/s of deviation.

Now let us evaluate the performance of the controller under the same condition.

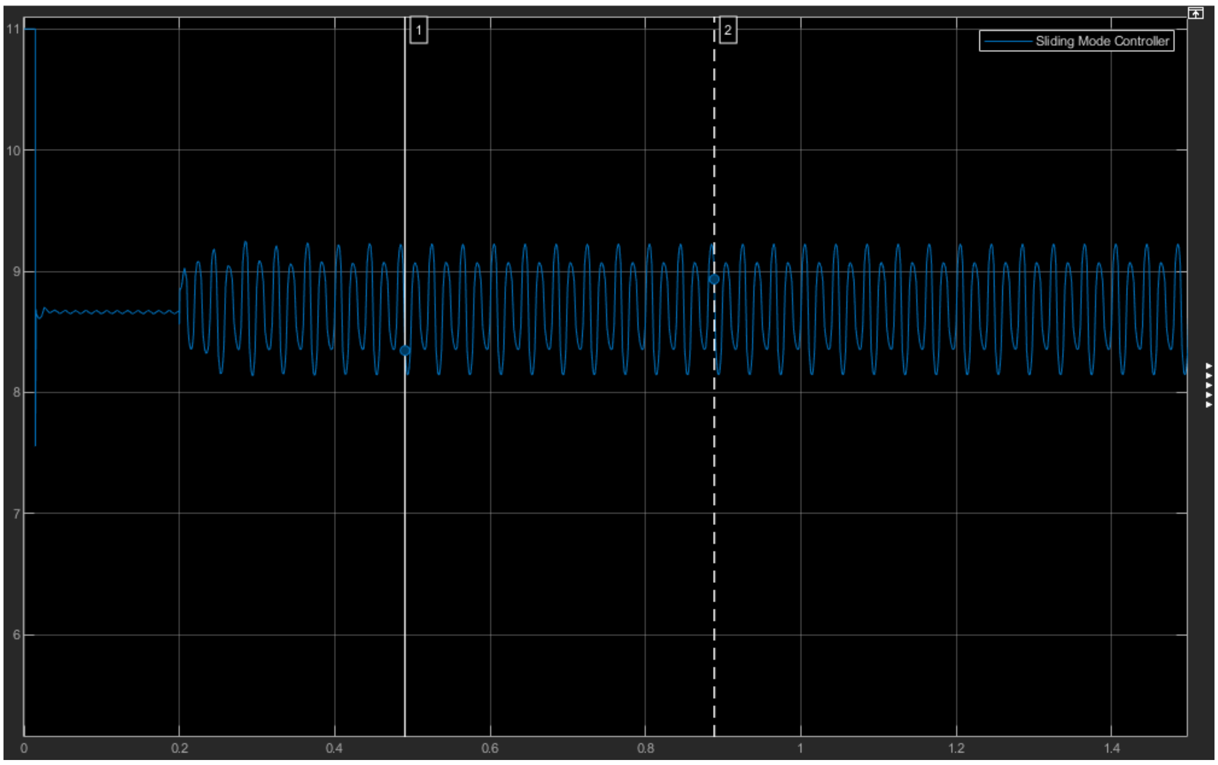


Figure 8:Super Twisting Control Signal

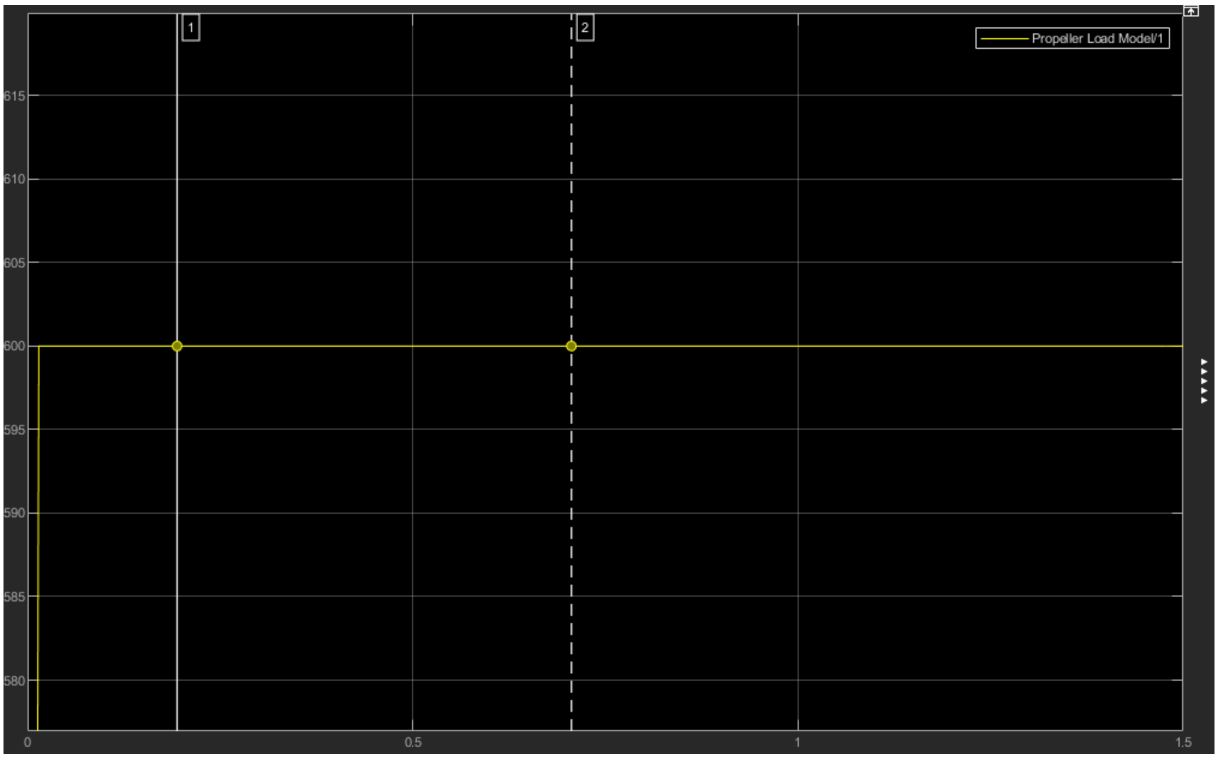
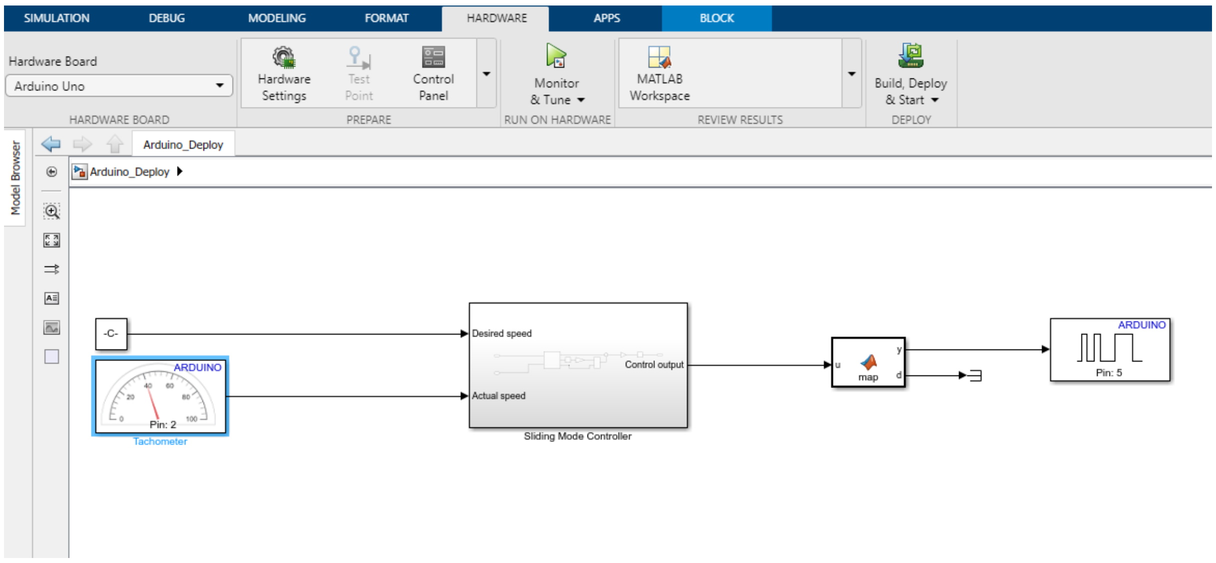


Figure 9:System Behavior

As we can see, the super twisting controller made an excellent job controlling the unknown disturbance. The oscillations are totally attenuated.

Deploy the algorithm to a Micro-controller Board

To do that, we use the Arduino Support Package for Simulink. It will enable us to compile our model-based Algorithm and deploy it to an Arduino board.


SIMULATION
DEBUG
MODELING
FORMAT
HARDWARE
APPS
BLOCK
Hardware Board
Arduino Uno
HARDWARE BOARD
Hardware Settings
Test Point
Control Panel
PREPARE
Monitor & Tune
RUN ON HARDWARE
MATLAB Workspace
REVIEW RESULTS
Build, Deploy & Start
DEPLOY
Model Browser
Arduino_Deploy
Arduino_Deploy
-C-
ARDUINO
Pin: 2
Tachometer
Desired speed
Actual speed
Control output
Sliding Mode Controller
u
map
y
d
ARDUINO
Pin: 5