Bangladesh University of Engineering & Technology

Project Report

Course No: EEE 318

Course Name: Control Systems I Laboratory

Project Name:

Reproducing and Analysis of the control system described in the publication

"Variable-gain control for respiratory systems" [1]

<u>Group Detail (Names & IDs)</u>:

SM Haider Ali Shuvo	(1606134)
Shakil Ahmed	(1606140)
Ratul Kundu	(1606147)
Prasun Datta	(1606148)
Raisa Bentay Hossain	(1606151)

Section: C1

Department: EEE

[1] By Hunnekens, Bram, Sjors Kamps, and Nathan Van De Wouw on IEEE Transactions on Control Systems Technology (2018).

Table of Contents

Task 1: Determine the transfer function of the respiratory system shown in Fig. 3.

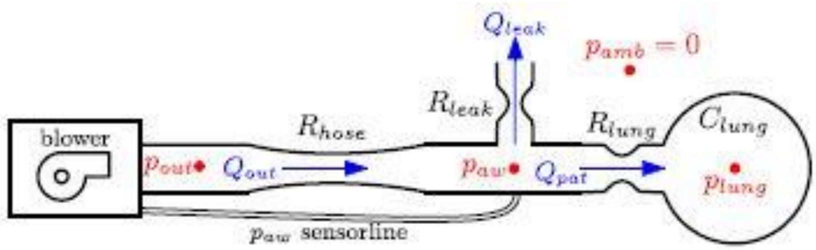


Fig. 3. Schematic of a respiratory system showing the different pressures (red), flows (blue), and resistances and compliance (black).

The transfer function of the system in state space representation:

$H(s)=C_h\,(sI-A_h)^{-1}\,B_h + D_h$

The parameters of the equations are:

$$A_h = -\frac{\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}}{R_{\text{lung}}C_{\text{lung}}\left(\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}\right)}$$

$$B_h = \frac{\frac{1}{R_{\text{hose}}}}{R_{\text{lung}}C_{\text{lung}}\left(\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}\right)}$$

$$C_h = \left[\frac{\frac{1}{R_{\text{lung}}}}{\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}} \; -\frac{\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}}{R_{\text{lung}}\left(\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}\right)}\right]^T$$

$$D_h = \left[\frac{\frac{1}{R_{\text{hose}}}}{\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}} \; \frac{\frac{1}{R_{\text{hose}}}}{R_{\text{lung}}\left(\frac{1}{R_{\text{lung}}}+\frac{1}{R_{\text{hose}}}+\frac{1}{R_{\text{leak}}}\right)}\right]^T$$

$[P_{aw}\ Q_{pat}]^T = C_h\, P_{lung} + D_h\, P_{out}$

So, C_{h1}, D_{h1} are associated with P_{aw} and C_{h2}, D_{h2} are associated with Q_{pat}.

Let,

$H(s)=[H_1(s)\ H_2(s)]^T$
$C_h= [C_{h1}\ C_{h2}]^T$
$D_h= [D_{h1}\ D_{h2}]^T$

Therefore $H_1(s)=\frac{Paw(s)}{Pout(s)} = \frac{Ch1}{(s-Ah)} * B_h+D_{h1}$

$H_2(s)= \frac{PQ\ pat(s)}{Pout(s)} =\frac{Ch2}{(s-Ah)} * B_h+D_{h2}$

The parameters for above equations are:

TABLE I
PARAMETERS SETTINGS USED FOR SIMULATIONS

Variable	Value	Unit
R_{lung}	$\frac{5}{1000}$	$\frac{mbar}{\frac{mL}{s}}$
C_{lung}	20	$\frac{mL}{mbar}$
R_{leak}	$\frac{60}{1000}$	$\frac{mbar}{\frac{mL}{s}}$
R_{hose}	$\frac{4.5}{1000}$	$\frac{mbar}{\frac{mL}{s}}$
ω_n	$2\pi 30$	$\frac{rad}{s}$

$R_{lung}=0.005$; $C_{lung}=20$; $R_{leak}=0.06$; $R_{hose}=0.0045$

$$\frac{1}{Rlung} + \frac{1}{Rleak} + \frac{1}{Rhose} = 438.89$$

$$\frac{1}{Rhose} + \frac{1}{Rleak} = 238.89$$

$$\frac{1}{Rlung} = 200$$

$$\frac{1}{Rhose} = 222.22$$

Therefore, the parameters of the equations are:

$A_h = \frac{-238.89}{43.89} = -5.443$
$B_h = \frac{222.22}{(438.89*0.005*20)} = 5.063$

$C_{h1} = \frac{200}{438.89} = 0.456$
$C_{h2} = - \frac{238.89}{0.005*438.89} = -108.861$

$D_{h1} = \frac{222.2}{438.89} = 0.5063$

$D_{h2} = \frac{222.2}{0.005*438.89} = 101.2646$

$$H_1(s) = C_{h1}\frac{B_h}{s-A_h} + D_{h1}$$

$$= \frac{(0.456*5.063)+0.5063*(s+5.443)}{s+5.443} = \frac{0.5063s+5.0647}{s+5.443}$$

$$H_2(s) = C_{h2}\frac{B_h}{s-A_h} + D_{h2}$$

$$= \frac{(-108.861*5.063)+101.2646*(s+5.443)}{s+5.443} = \frac{101.2646s}{s+5.443}$$

Task 2: Determine the overall transfer function of the closed loop control system shown in Fig. 4.

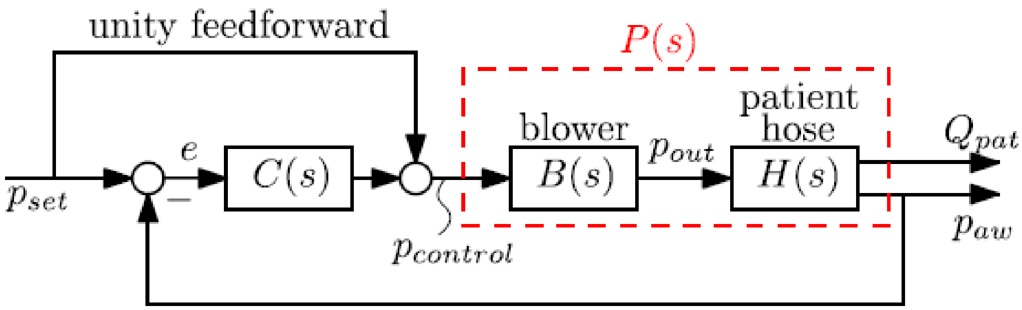


Fig. 4. Closed-loop control scheme with a linear controller $C(s)$.

Here $H_1(s)$ is the transfer function for P_{aw} & $H_2(s)$ is the transfer function for Q_{pat}

So, for P_{aw} , $P(s)=B(s)\ H_1(s)$

for Q_{pat} , $P(s)=B(s)\ H_2(s)$

From above schematic we can write 3 equations for P_{aw} :

$P_{set}-P_{aw}=e$

$e*C(s) +P_{set}=P_{control}$

$P_{control} *P(s)=P_{aw}$

Therefore, $P_{aw}= (e*C(s) +P_{set})*P(s)$

$=e*C(s)*P(s)+P_{set}*P(s)$

$= (P_{set}-P_{aw})*C(s)*P(s)+P_{set}*P(s)$

$=P_{set}*C(s)*P(s)-P_{aw}*C(s)*P(s)+P_{set}*P(s)$

$=>P_{aw}+P_{aw}*C(s)*P(s) = P_{set}*P(s)*(1+C(s))$

$=>P_{aw}(1+C(s)*P(s)) =P_{set}*P(s)*(1+C(s))$

$=>P_{aw}/P_{set}=(P(s)*(1+C(s)))/(1+C(s)*P(s))$

So,

Overall transfer function for $P_{aw}=\frac{Paw}{Pset}= \frac{B(s)*H1(s)*(1+C(s))}{1+(C(s)*B(s)*H1(s))}$

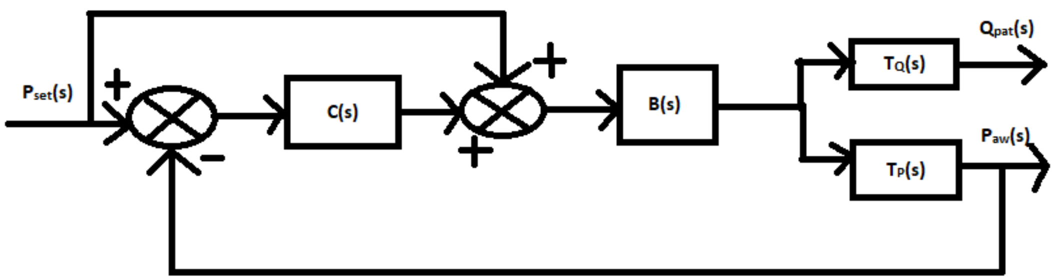
Overall transfer function for $Q_{\ pat}=\frac{Qpat}{Pset}= (P_{aw}(s)/P_{set}(s)) \times (Q_{pat}(s)/P_{aw}(s))$

$$=\frac{B(s)*H1(s)*(1+C(s))}{1+(C(s)*B(s)*H1(s))} * \frac{H2(s)}{H1(s))}$$

$$=\frac{B(s)*H2(s)*(1+C(s))}{1+(C(s)*B(s)*H1(s))}$$

Task-3: Sketch the root locus of the control system shown in Fig. 4 for 0<ki< of the integral controller C(s).

The Overall closed loop system can be drawn as below in a simplified way –



Where , $C(s) = \frac{k_i}{s}$, $B(s) = \frac{2*pi*30}{s^2+2*2*pi*30*s+(2*pi*30)^2}$.

To find out $T_p(s)$ and $T_Q(s)$ we convert the State domain equation derived in previous section to Laplace Domain.

State domain Representation of Hose System is –

$$\frac{dp_{lung}}{dt} = A_h P_{lung} + B_h P_{out}$$

$$\begin{pmatrix} P_{aw} \\ Q_{pat} \end{pmatrix} = C_h P_{lung} + D_h P_{out}$$

Where , P_{out} is the output of Blower .

Where the Constants A_h, B_h, C_h, D_h are as below –

$$A_h = -\frac{\frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}}}{R_{\text{lung}} C_{\text{lung}} \left(\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}} \right)}$$

$$B_h = \frac{\frac{1}{R_{\text{hose}}}}{R_{\text{lung}} C_{\text{lung}} \left(\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}} \right)}$$

$$C_h = \left[\frac{\frac{1}{R_{\text{lung}}}}{\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}}} - \frac{\frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}}}{R_{\text{lung}} \left(\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}} \right)} \right]^T$$

$$D_h = \left[\frac{\frac{1}{R_{\text{hose}}}}{\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}}} \quad \frac{\frac{1}{R_{\text{hose}}}}{R_{\text{lung}} \left(\frac{1}{R_{\text{lung}}} + \frac{1}{R_{\text{hose}}} + \frac{1}{R_{\text{leak}}} \right)} \right]^T$$

We express C_h, D_h Matrices as the following to simplify things –

$$C_h = [C_{h1} \; C_{h2}]^T$$

$$And \; D_h = [D_{h1} \; D_{h2}]^T$$

Then , For the System generating P_{aw} output , We can write down Simplified state space Equation –

$$\frac{dp_{lung}}{dt} = A_h P_{lung} + B_h P_{out}$$

$$P_{aw} = C_{h1} P_{lung} + D_{h1} P_{out}$$

Now if we convert the Single input Single output state domain representation into Laplace domain then –

$$T_p(s) = \frac{P_{aw}(s)}{P_{out}(s)}$$

$$= C_{h1}(sI - A_h)^{-1} + D_{h1} \text{ , } Here\; I\; is\; 1x1\; Identity\; Matrix$$

$$= C_{h1}\frac{B_h}{s - A_h} + D_{h1}$$

$$= \frac{C_{h1}B_h + D_{h1}(s - A_h)}{s - A_h}$$

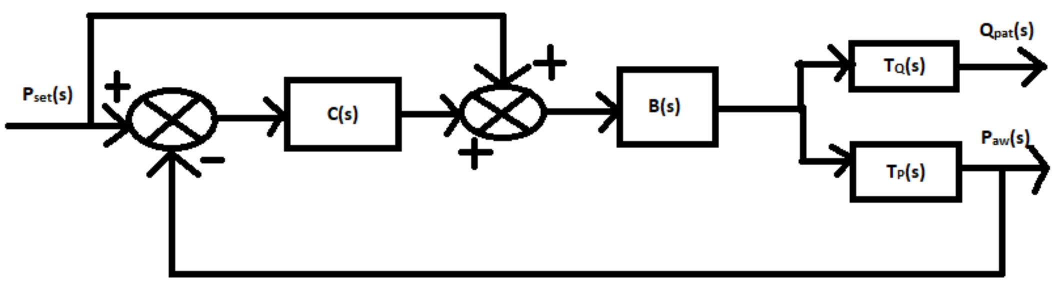
$$= \frac{D_{h1}(s - \left(A_h - \frac{C_{h1}B_h}{D_h}\right))}{s - A_h}$$

Similarly ,

$$T_Q(s) = \frac{Q_{pat}(s)}{P_{out}(s)} = \frac{D_{h1}(s - \left(A_h - \frac{C_{h2}B_h}{D_h}\right))}{s - A_h}$$

We Can see both T_P and T_Q systems are first order systems with a single zero and single pole .

Now , For the Pressure system , If we use some Laplace Domain Algebra we can easily find out our target $\frac{P_{aw}}{P_{set}}$.



$$P_{aw}(s) = [(P_{set}(s) - P_{aw}(s)) * C(s) + R(s)] * T_p(s) * B(s)$$

$$=> \; P_{aw}(s) + P_{aw}(s) * C(s) * T_p(s) * B(s)$$
$$= P_{set}(s) * C(s) * T_P(s) * B(s) + P_{set}(s) * B(s) * T_p(s)$$

$$=> \frac{P_{aw}(s)}{P_{set}(s)} = \frac{C(s) * B(s) * T_p(s)}{1 + C(s) * B(s) * T_p(s)} + \frac{B(s) * T_p(s)}{1 + C(s) * B(s) * T_p(s)}$$

$$=> \frac{P_{aw}(s)}{P_{set}(s)} = \frac{k_i * \frac{1}{s} * B(s) * T_p(s)}{1 + k_i * \frac{1}{s} * B(s) * T_p(s)} + \frac{B(s) * T_p(s)}{1 + k_i * \frac{1}{s} * B(s) * T_p(s)}$$

We can see Denominator of Overall Transfer function is of the form $1 + k_i * \frac{1}{s} * B(s) * T_p(s)$.

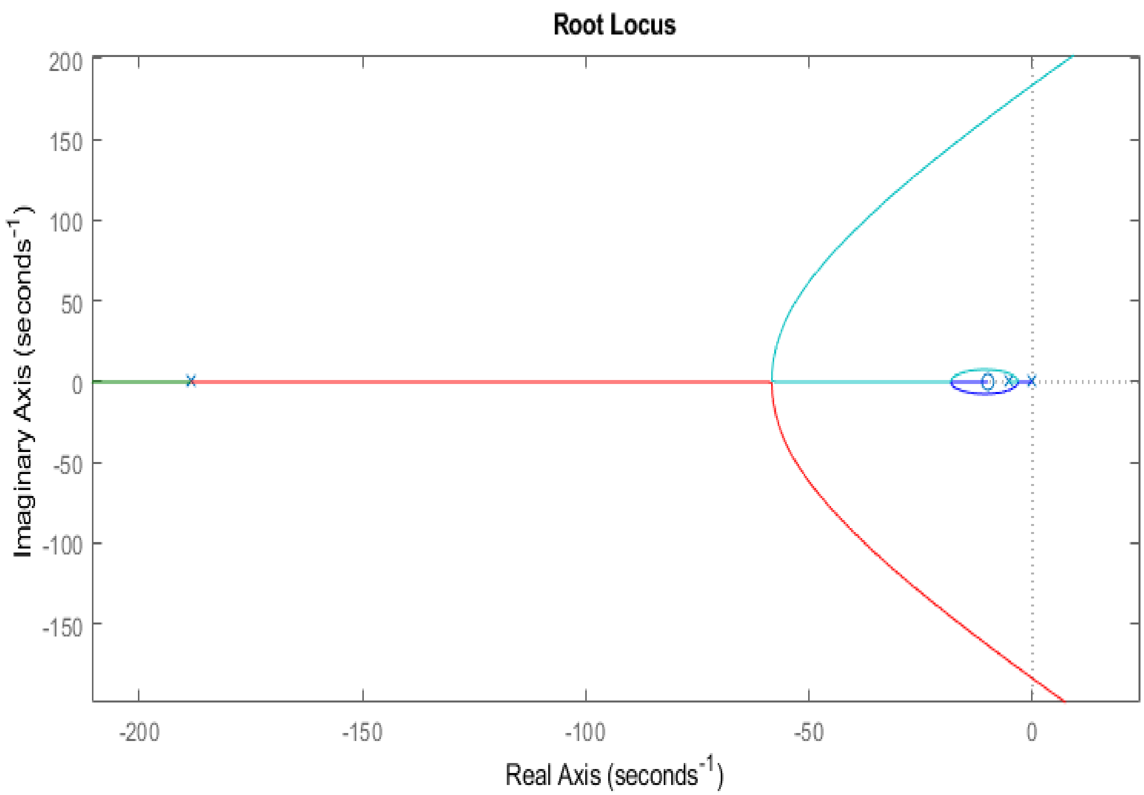
So , Our effective Open Loop Transfer Function is $\frac{1}{s} * B(s) * T_p(s)$.

Inside the rlocus functions argument We put this as our Open loop Transfer function and we get Root Locus as we vary $0<k_i<\infty$.

Code –

```
%% Parameters
Rlung = 5/1000 ; Clung = 20 ; Rleak = 60/1000 ; Rhose = 4.5/1000 ; wn =
2*pi*30 ;

Ah = -(1/Rhose + 1/Rleak) / (Rlung*Clung*(1/Rlung + 1/Rhose + 1/Rleak));
Bh = (1/Rhose) / (Rlung*Clung*(1/Rlung + 1/Rhose + 1/Rleak));
Ch1 = (1/Rlung)/((1/Rlung + 1/Rhose + 1/Rleak));
Dh1 = (1/Rhose)/((1/Rlung + 1/Rhose + 1/Rleak));
Ch2 = -(1/Rhose + 1/Rleak) / (Rlung*(1/Rlung + 1/Rhose + 1/Rleak)) ;
Dh2 = (1/Rhose) / (Rlung*(1/Rlung + 1/Rhose + 1/Rleak));
%%
s = tf('s');
B = wn^2 / (s^2+2*wn*s+wn^2);
T = (Dh1*(s-(Ah-(Ch1*Bh)/Dh1)) / (s-Ah))*B;
controller = 1/s ;
rlocus(T*controller , 0:.01:5000);
axis([-250,100,-300,300]);
```

Root Locus –



TASK 4 (PART 1) Reproduce the results shown in Fig. 7 for the combined feedback and feedforward control system. Discuss the necessity of both feedback and feedforward control.

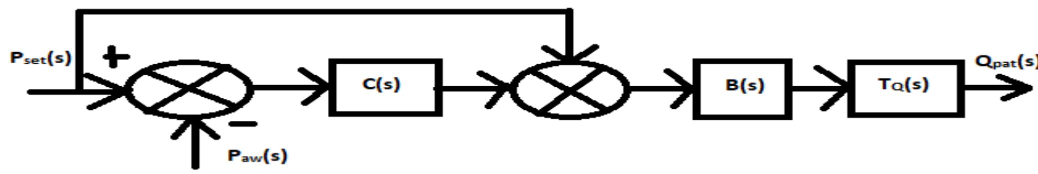
As we have shown in last section ,

$$\frac{P_{aw}(s)}{P_{set}(s)} = \frac{k_i * \frac{1}{s} * B(s) * T_p(s)}{1 + k_i * \frac{1}{s} * B(s) * T_p(s)} + \frac{B(s) * T_p(s)}{1 + k_i * \frac{1}{s} * B(s) * T_p(s)}$$

We Simulate The system both using Matlab Code and Simulink .In Matlab Code , We can write down the Pressure System as the following –

sys = feedback(ki*controller*T,1) + feedback(ki*controller*T,1)/(ki*controller) ;
Where we define , controller = $\frac{1}{s}$, T = $T_P(s) * B(s)$.

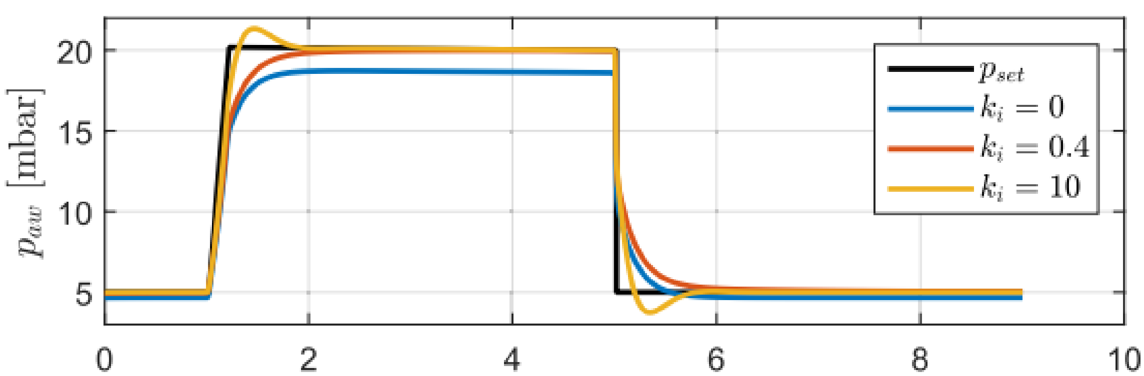
To Get Q_{pat} using Matlab Code , We realize the Air Flow system can be effectively thought of an open loop system with the following configuration –



So , We first simulate to get $P_{aw}(t)$ and Then using $P_{aw}(t)$ and $P_{set}(t)$we can get $Q_{pat}(t)$ using the following System as Transfer function –

$$Q_{pat}(s)$$
$$= [1 + C(s)] * B(s) * T_Q(s) * \boldsymbol{P_{set}(s)} - C(s) * B(s) * T_Q(s) * \boldsymbol{P_{aw}(s)}$$

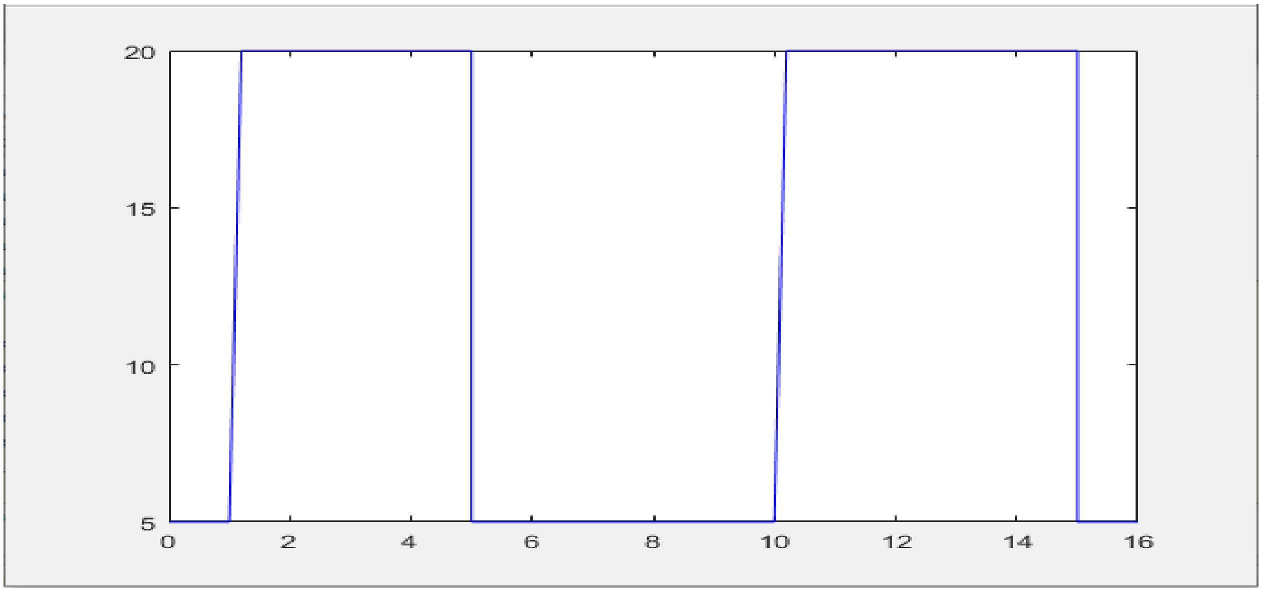
Now , Exact shape of $P_{set}(t)$ is not mentioned in the paper . We make some engineering approximation here .



From information present in the above image , We assume a rise time from 5 to 20 as ~200ms and Fall time as a discontinuous sudden file . Also , we assume time at each state (5 and 20 mbar) being ~5second .

Finally , We take 2 cycles of P_{set} and simulate for it since in the first cycle there might be some transient effect .

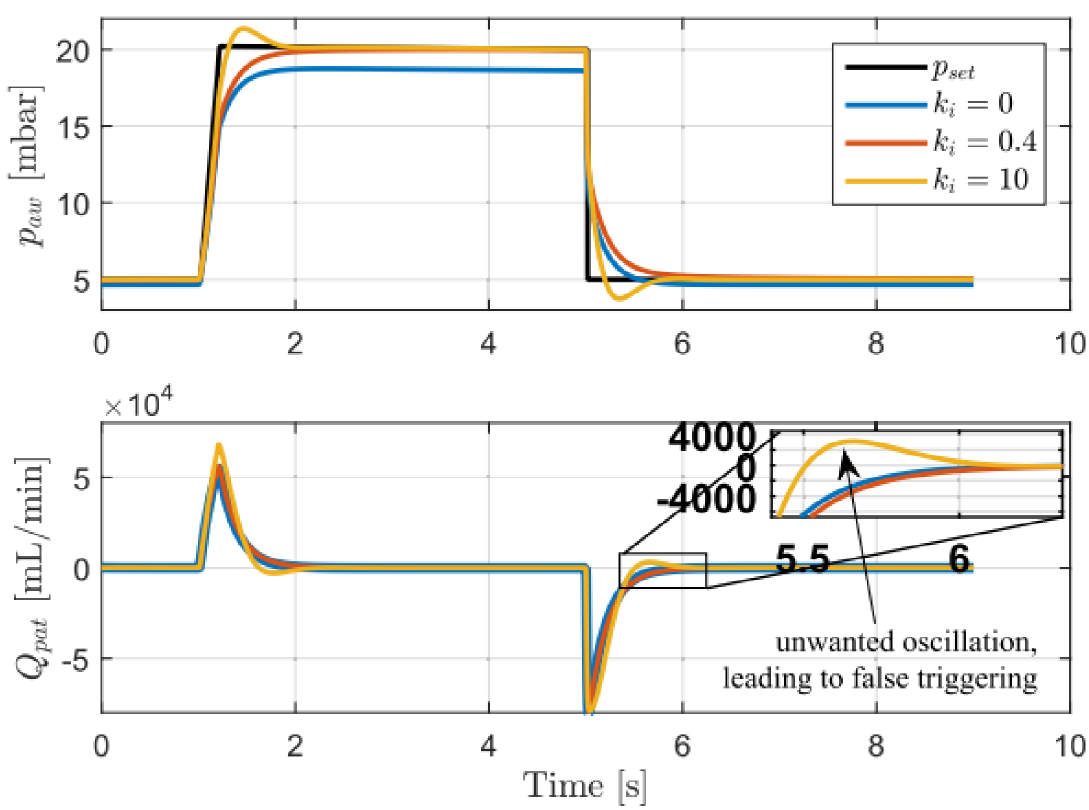
Our constructed P_{set} is as follows –



Our code to achieve P_{aw} and Q_{pat} for one value of K_i is shown below to give a clear picture of what's said in preceeding pages.

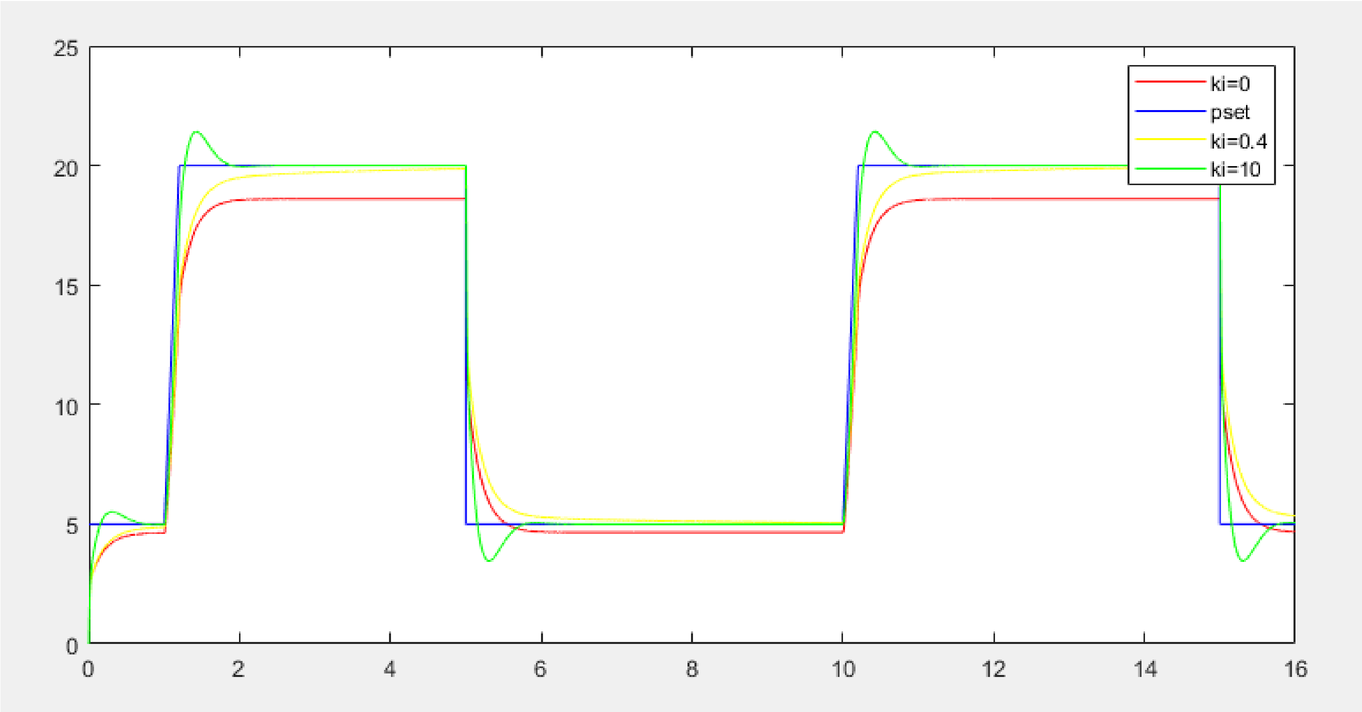
```
%% Code to Get P_aw
ki = 0.4 ;
sys = feedback(ki*controller*T,1) +
feedback(ki*controller*T,1)/(ki*controller) ;
[y t x] = lsim(sys , pset , t);
plot(t,y,'y');hold on;

%% Code to Get Flow Response
T_f = (Dh2*(s-(Ah-(Ch2*Bh)/Dh2)) / (s-Ah))*B;
ki = .4 ; eps=1e-8 ;
sys = feedback(ki*controller*T,1) +
feedback(ki*controller*T,1)/(ki*controller) ;
[y t x] = lsim(sys , pset , t);
sys_f_1 = (1+controller*ki)*T_f ;
sys_f_2 = -controller*ki*T_f;
[y_f1 t x] = lsim(sys_f_1 , pset , t);
[y_f2 t x] = lsim(sys_f_2 , y , t);
plot(t,(y_f1+y_f2)*60 , 'r'); hold on;
```

Our **<u>Expected output</u>** is as below –

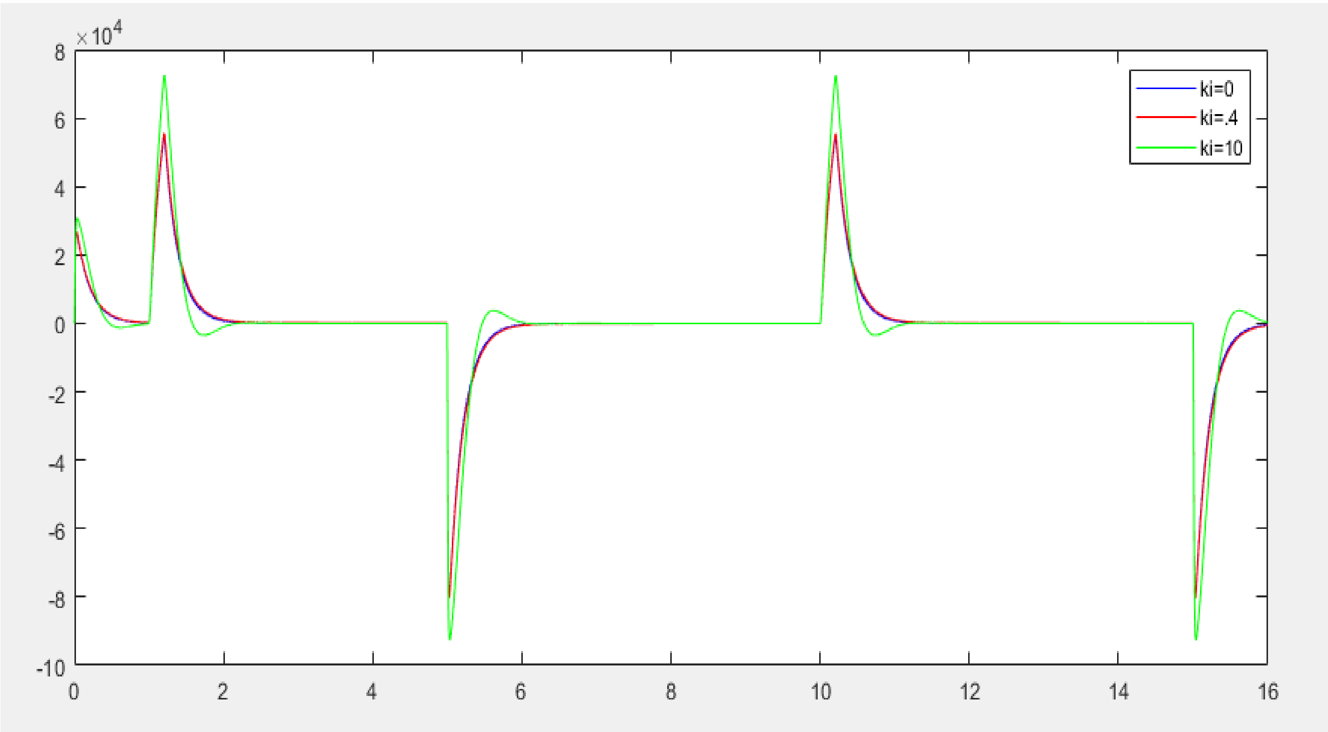


Our Achieved output is shown below –

1) $P_{aw}(t)$ for $k_i = 0{,}0\ .4{,}10$

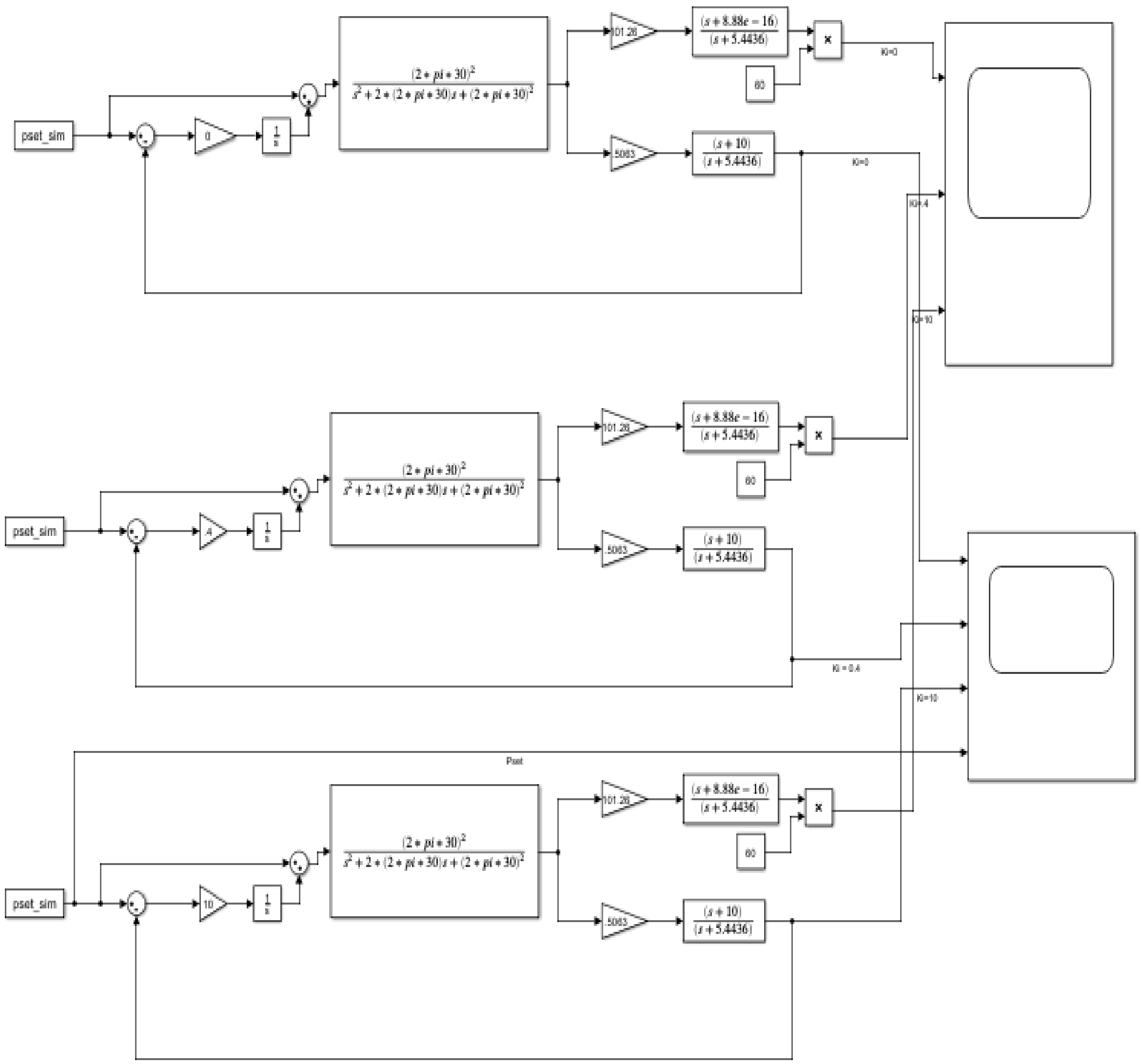


2) $Q_{pat}(t)$ for $k_i = 0{,}0\ .4{,}10$

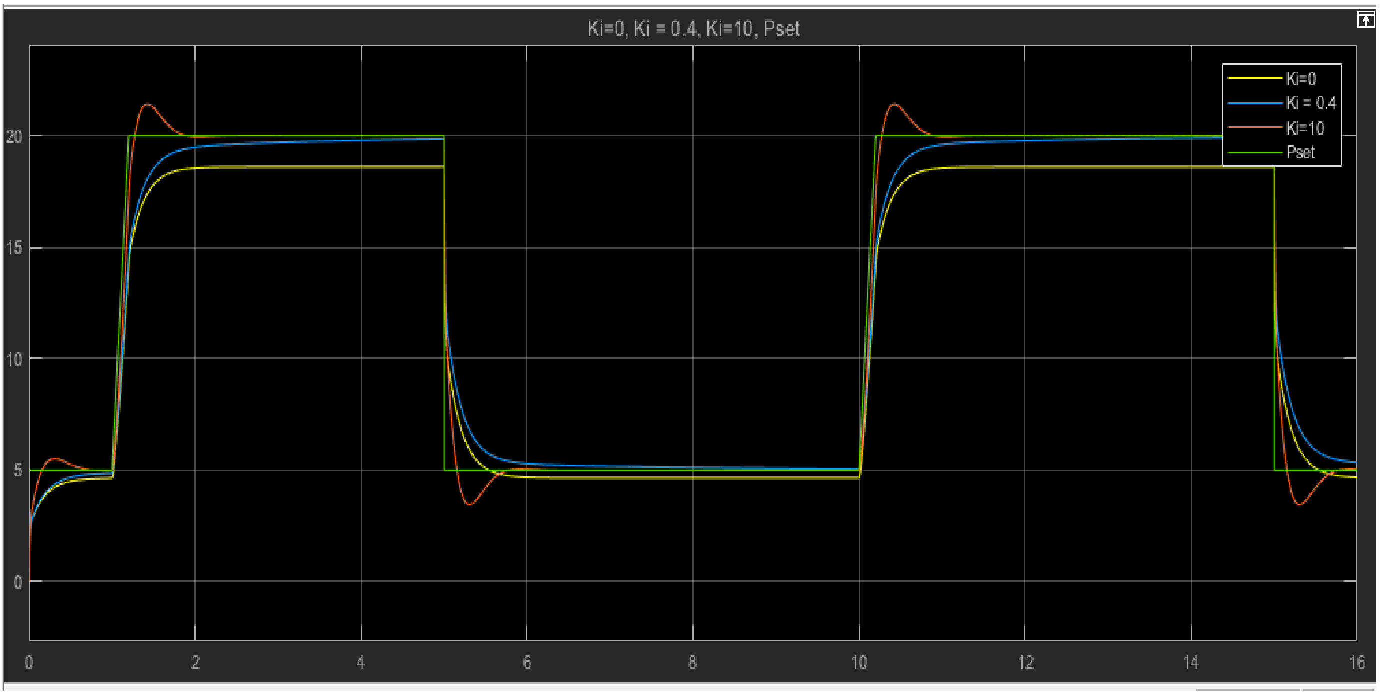


We also simulate the system in Simulink for different k_i which gives identical output as shown below .

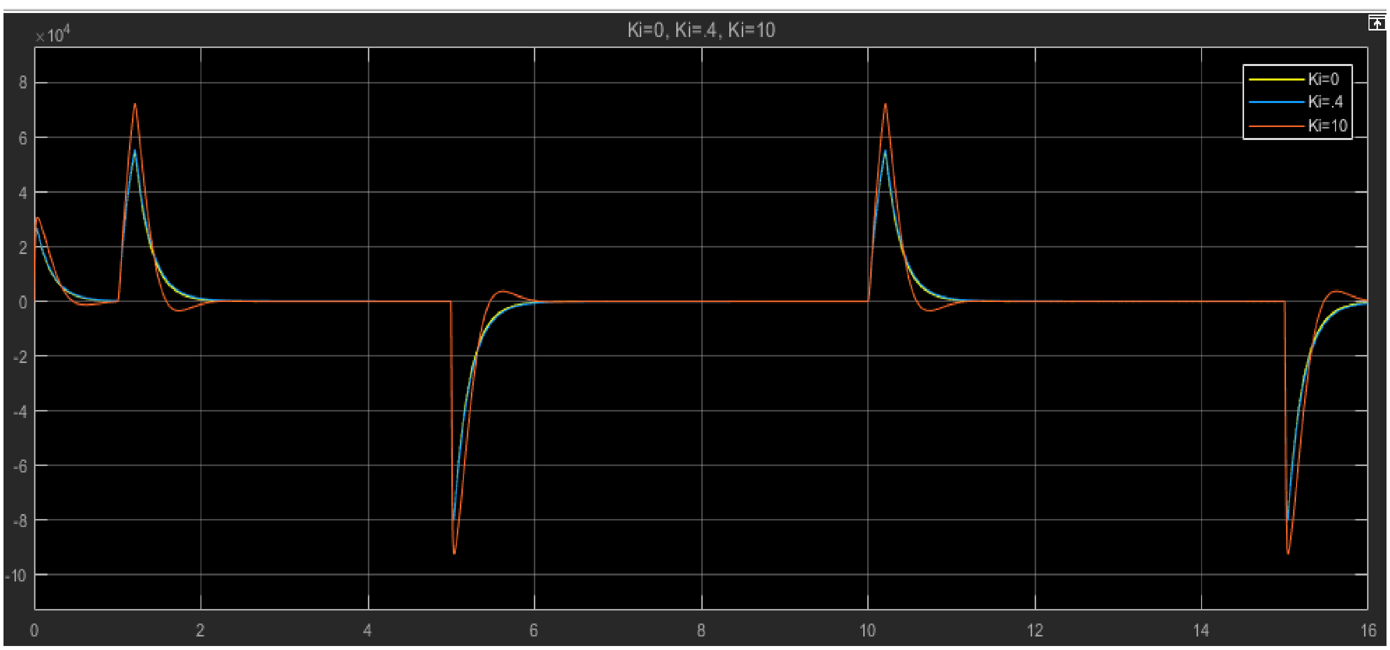
Simulink System –



Produced Result –



And

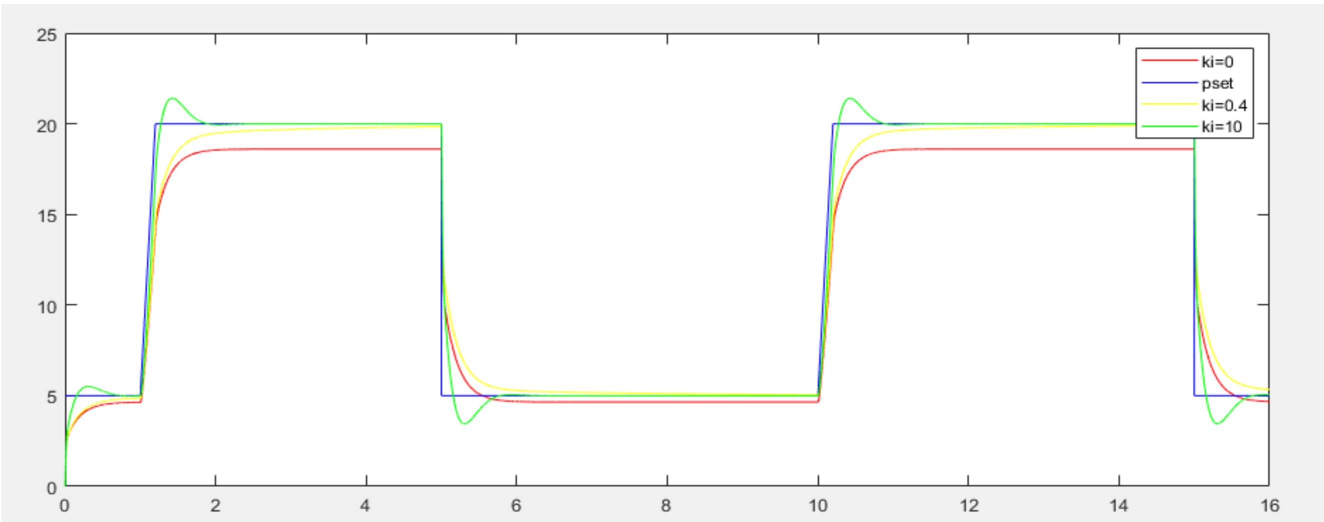


We get similar and expected outputs in both Matlab Code and Simulink .

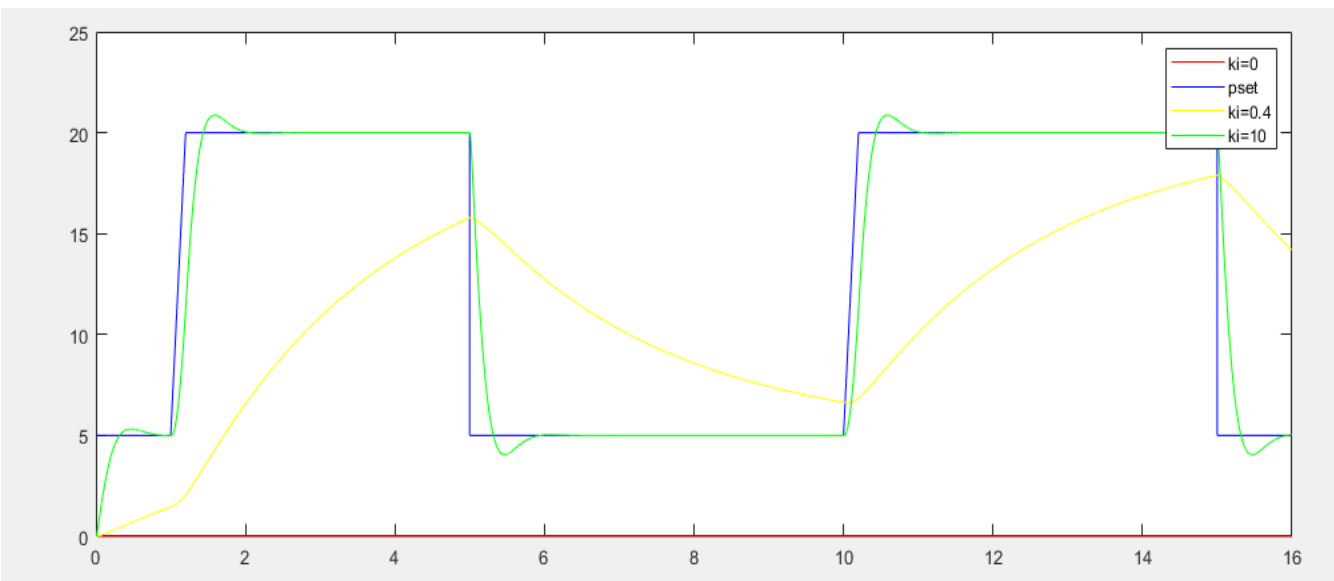
TASK 4 (PART 2): Necessity of Both Feedback and Feedforward Control

Feedback control is part of any modern control system , Without feedback its impossible to make any real system follow a certain output . But Why Feedforward ? To Investigate , We take the Experimentalist's approach . We simply first remove the feedforward path and then compare systems response for cases with feedforward and without feedforward .

$P_{aw}(t)$ for System with feedforward –



$P_{aw}(t)$ for System without feedforward –



From the both graphs, We can make the following observations :

Feedforward path helps in speeding up the response . For example , without the feedforward path , for ki=0.4 case the output P_{aw} couldn't reach steady state at all .

For **ki=10 ,** Rise time is nearly **~180 seconds with feedforward** and **Without feedforward path it is ~300 seconds.**

Feedback path is important to improve the performance of any control system. Feedback path is used to identify the variance between actual and expected outcome of a system.

Feedback path has an effect on overall gain of the system. It can also affect the sensitivity of the system to parameter variation. On top of that a negative feedback path can significantly decrease the effect of noise on the system

On the other hand we saw that the feed forward path helps to improve the rise time and speeds up the system. It helps to monitor the primary response of the system to the input without any influence of load.

TASK 5 : COMPARISM BETWEEN IDEAL INTEGRATOR, PI AND LAG CONTROLLER

In order to find which linear controller will be more preferable, we can analysis steady state response and transient response using each controller and compare between them.

Let us first choose the define each controller.

Ideal integrator controller: $\frac{10}{s}$; PI controller : $\frac{10*(s+0.1)}{s}$; Lag controller : $\frac{10*(s+0.1)}{(s+0.01)}$.

Transient State Analysis:

We have used Simulink to do the transient state analysis.

Simulink Circuit:

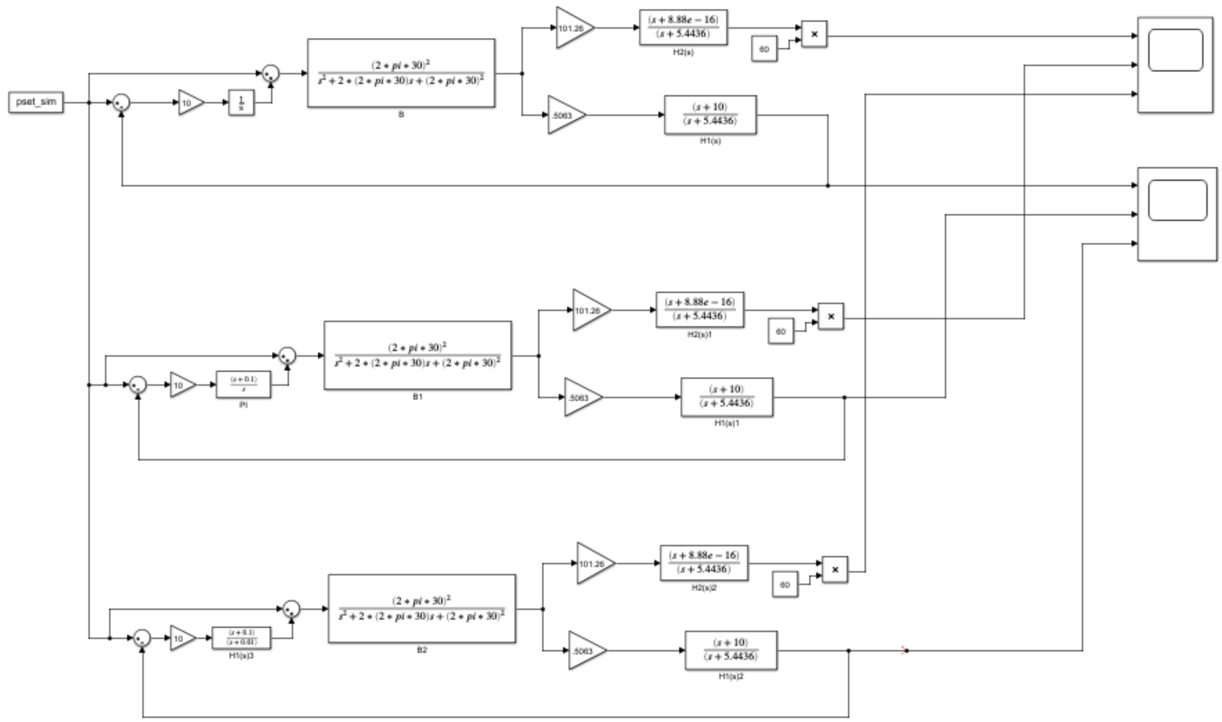


Figure: Schematic circuit for different controller

$P_{aw}(t)$ of the System:

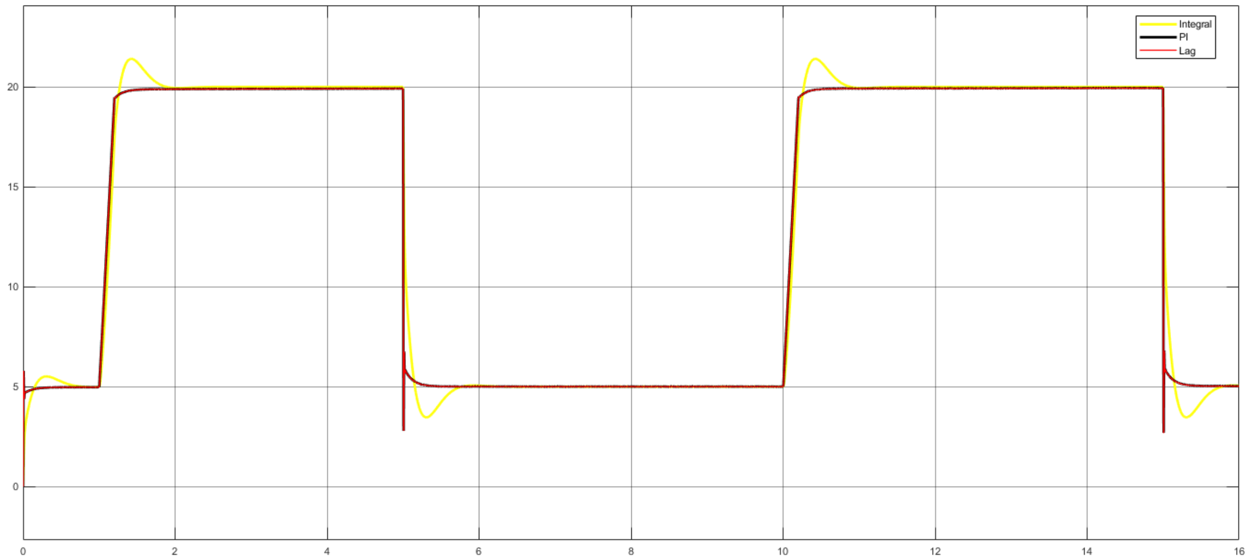


Figure: Paw(t) for different linear Controller

Rise time and percent overshoot (%OS) for each controller:

Trace Selection

Integral

Bilevel Measurements

► Settings

▼ Transitions

High	2.002e+01
Low	5.031e+00
Amplitude	1.499e+01
+ Edges	2
+ Rise Time	178.862 ms
+ Slew Rate	67.028 (/s)
- Edges	2
- Fall Time	102.873 ms
- Slew Rate	-116.539 (/s)

▼ Overshoots / Undershoots

+ Preshoot	0.273 %
+ Overshoot	9.283 %
+ Undershoot	1.346 %
+ Settling Time	--
- Preshoot	-0.113 %
- Overshoot	1.688 %
- Undershoot	10.459 %
- Settling Time	--

► Cycles

Trace Selection

PI

Bilevel Measurements

► Settings

▼ Transitions

High	1.986e+01
Low	5.091e+00
Amplitude	1.477e+01
+ Edges	2
+ Rise Time	163.249 ms
+ Slew Rate	72.398 (/s)
- Edges	2
- Fall Time	3.496 ms
- Slew Rate	-3.380 (/ms)

▼ Overshoots / Undershoots

+ Preshoot	0.795 %
+ Overshoot	0.267 %
+ Undershoot	1.834 %
+ Settling Time	--
- Preshoot	0.462 %
- Overshoot	11.430 %
- Undershoot	15.923 %
- Settling Time	--

► Cycles

Trace Selection

Lag

Bilevel Measurements

► Settings

▼ Transitions

High	1.986e+01
Low	5.089e+00
Amplitude	1.477e+01
+ Edges	2
+ Rise Time	163.199 ms
+ Slew Rate	72.393 (/s)
- Edges	2
- Fall Time	3.494 ms
- Slew Rate	-3.381 (/ms)

▼ Overshoots / Undershoots

+ Preshoot	0.802 %
+ Overshoot	0.289 %
+ Undershoot	1.896 %
+ Settling Time	--
- Preshoot	0.465 %
- Overshoot	11.402 %
- Undershoot	15.961 %
- Settling Time	--

► Cycles

We can compare the rise time and overshoot of the controllers to reach the conclusion that, both the PI and Lag controller are better than pure integral controller.

The table below summaries the comparism.

Controller	Rise Time (ms)	Percent Overshoot(%OS)
Ideal Integrator	178.826	9.283%
PI Controller	163.249	0.267%
Lag Controller	163.199	0.289%

Q_{pat}(t) of the System:

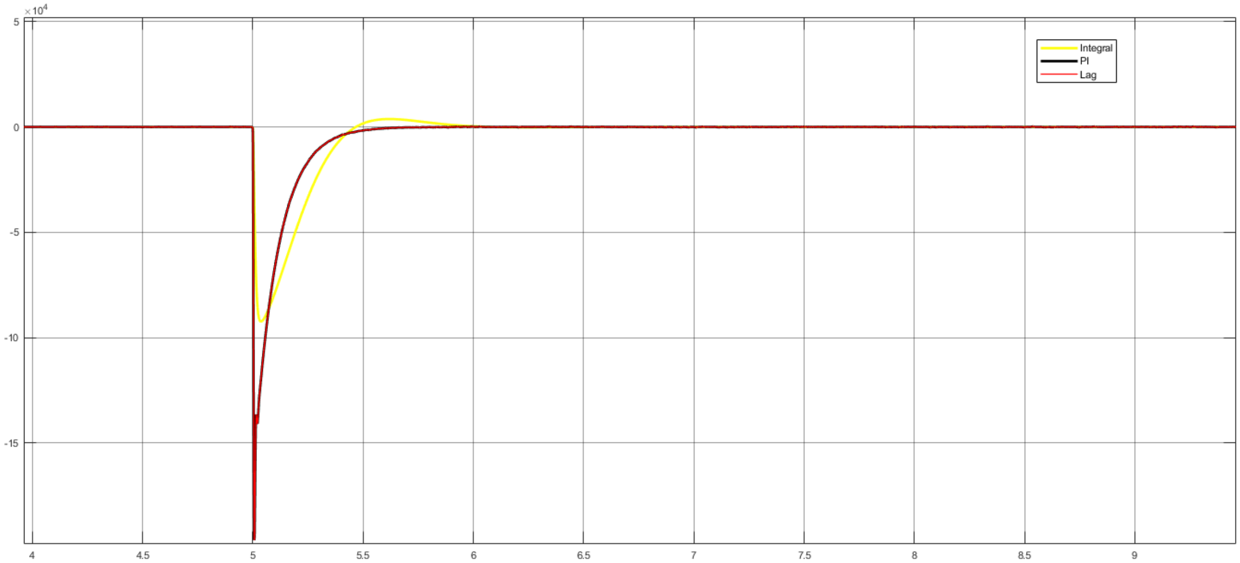


Figure: Qpat(t) of the system for different controller

The response of ideal integrator shows a negative overshoot during expiration, which is not expected. So the **PI** and **Lag** controller is more expectable linear controller.

Steady State Analysis:

For steady state analysis, we can find the steady state error constants of the system and compare between them.

It will be easier to calculate the steady state error constant if we can reduce the overall system in a single negative feedback system. From TASK 2 we have found,

Overall transfer function for P_{aw}=$\frac{Paw}{Pset}$= $\frac{\mathrm{B(s)*H1(s)*(1+C(s))}}{\mathrm{1+(C(s)*B(s)*H1(s))}}$

$$= \frac{1.797e04*s+1.798e05}{s^3+382.4*s^2+1.934e05+C(s)*(1.797e04*s+1.798e05)} * (1+C(s))$$

Overall transfer function for Q $_{pat}$=$\frac{\text{Qpat}}{Pset}$ = $\frac{\text{B(s)*H2(s)*(1+C(s))}}{\text{1+(C(s)*B(s)*H1(s))}}$

$= \frac{3.595e06*s}{s^3+382.4*s^2+1.934e05+C(s)*(1.797e04*s+1.798e05)} * (1 + C(s)).$

(Calculations are done using MATLAB).

Now we can find the forward path gain of the reduced system from the relation,

T(s) = $\frac{G(s)}{1+G(s)}$. Or, G(s) = $\frac{T(s)}{1-T(s)}$.

So, G_{aw}(s) = $\frac{Paw(s)}{1-Paw(s)}$ and Gpat(s) = $\frac{Qpat(s)}{1-Qpat(s)}$.

1. <u>Ideal Integrator:</u> For ideal integral controller, C(s) = $\frac{10}{s}$. So,

Paw= $\frac{1.797e04*s+1.798e05}{s^3+382.4*s^2+1.934e05+\frac{10}{s}*(1.797e04*s+1.798e05)} * (1 + \frac{10}{s}) = \frac{17970\ s^3 + 359500\ s^2 + 1.798e06\ s}{s^5 + 382.4\ s^4 + 373100\ s^2 + 1.798e06\ s}$

G_{aw}(s) = $\frac{Paw(s)}{1-Paw(s)}$

$= \frac{17970\ s^8 + 7.231e06\ s^7 + 1.393e08\ s^6 + 7.392e09\ s^5 + 1.664e11\ s^4 + 1.317e12\ s^3 + 3.233e12\ s^2}{s^10 + 764.8\ s^9 + 1.283e05\ s^8 - 6.485e06\ s^7 + 1.497e08\ s^6 - 6.017e09\ s^5 - 2.724e10\ s^4 + 2.445e10\ s^3}$

This is a type 1 system. $Kv1 = \lim_{s\to 0} s * G(s) = \frac{3.233e12}{2.445e10} = 132.229$.

2. <u>PI controller:</u> For PI controller, C(s) = $\frac{10*(s+0.1)}{s}$. So,

Paw= $\frac{1.797e04*s+1.798e05}{s^3+382.4*s^2+1.934e05+\frac{10*(s+0.1)}{s}*(1.797e04*s+1.798e05)} * (1 + \frac{10*(s+0.1)}{s})$

$= \frac{197670\ s^3 + 1.996e06\ s^2 + 179800\ s}{s^5 + 382.4\ s^4 + 179700\ s^3 + 2.009e06\ s^2 + 179800\ s}$

G_{aw}(s) = $\frac{Paw(s)}{1-Paw(s)}$

$= \frac{197670\ s^8 + 7.758e07\ s^7 + 3.628e10\ s^6 + 7.559e11\ s^5 + 4.078e12\ s^4 + 7.201e11\ s^3 + 3.233e10\ s^2}{s^10 + 764.8\ s^9 + 3.08e05\ s^8 + 6.387e07\ s^7 - 2.455e09\ s^6 - 3.36e10\ s^5 + 2.41e10\ s^4 + 2.445e09\ s^3}$

This is a type 1 system. $Kv2 = \lim_{s\to 0} s * Gaw(s) = \frac{3.233e10}{2.445e09} = 13.2229$.

3. <u>Lag controller</u>: For lag controller, C(s) = $\frac{10*(s+0.1)}{s+0.01}$. So,

Paw= $\frac{1.797e04*s+1.798e05}{s^3+382.4*s^2+1.934e05+\frac{10*(s+0.1)}{s+0.01}*(1.797e04*s+1.798e05)} * (1 + \frac{10*(s+0.1)}{s+0.01})$

=

$$\frac{197670\ s^3 + 1.996e06\ s^2 + 179800\ s197670\ s^3 + 1.998e06\ s^2 + 2.016e05\ s + 1816}{s^5 + 382.4\ s^4 + 179700\ s^3 + 2.009e06\ s^2 + 179800\ ss^5 + 382.4\ s^4 + 1.797e05\ s^3 + 2.011e06\ s^2 + 2.018e05\ s + 1817}$$

$G_{aw}(s) = \frac{Paw(s)}{1-Paw(s)}$

=

$$\frac{197670\ s^8 + 7.759e07\ s^7 + 3.629e10\ s^6 + 7.567e11\ s^5 + 4.094e12\ s^4 + 8.093e11\ s^3 + 4.796e10\ s^2 + 7.328e08\ s + 3.3e06}{s^{10} + 764.8\ s^9 + 3.08e05\ s^8 + 6.388e07\ s^7 - 2.454e09\ s^6 - 3.367e10\ s^5 + 2.305e10\ s^4 + 3.183e09\ s^3 + 8.133e07\ s^2 + 7.655e05\ s + 2472}$$

This is a type 0 system. Kp3 = $\lim_{s\to 0} Gaw(s)$ = $\frac{3.3e06}{2472}$ = 1334.95.

The following table summarize the steady state response for different system.

Controller	Kp	Kv
Ideal Integrator $(\frac{10}{s})$	*Infinity*	132.29
PI Controller $(\frac{10*(s+0.1)}{s})$	*Infinity*	13.229
Lag Controller $(\frac{10*(s+0.1)}{s+0.01})$	1334.95	0

Judging from these values we can say that, in case of ideal integrator, errors for both the step input and ramp input is less than the other two controller. So the ideal integrator will give a better steady state response. However, the difference between the integrator and PI controller is not that much, and the error for Pi controller can be reduced by adjusting the position of zero.

Conclusion:

From the transient analysis we found that PI and Lag controller is better than ideal integral controller. But ideal integrator gives us a better steady state response. In both cases the PI controller is more convenient than the other two controller. Thus it will be a better choice.

<u>TASK 6: Design your preferred linear controller in order to meet the specifications stated in page 166 between column 1 and 2.</u>

We are asked to design a Linear Controller that can fulfill the following requirements –

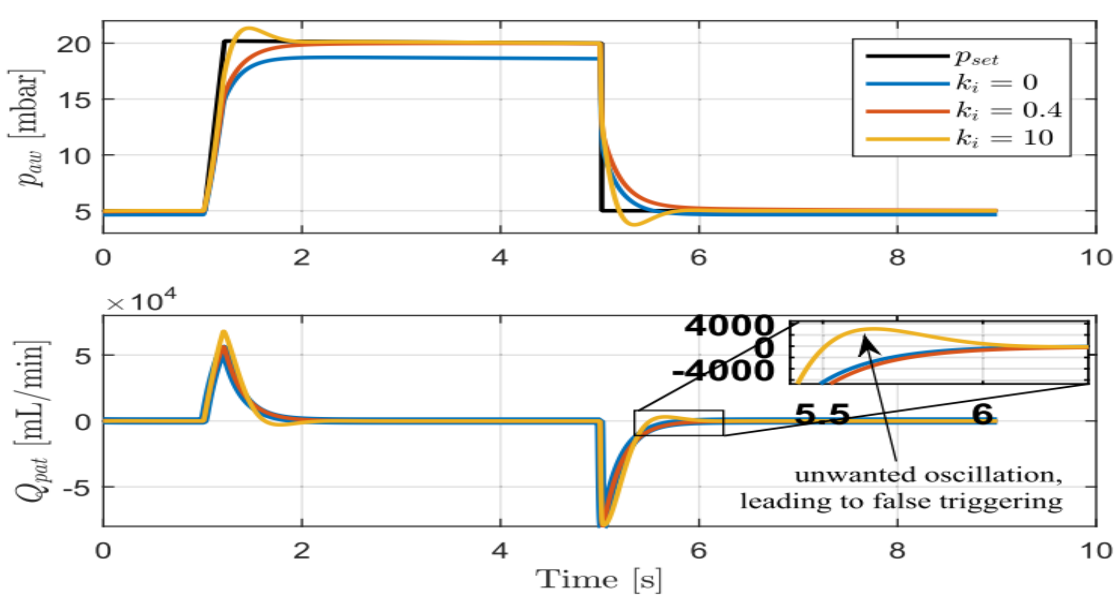
1)Rise time from 10% to 90% of a pressure set point should be approximately 200ms.

2)Pressure at the end of an inspiration , the so called plateau pressure should be within pressure band of 2mbar of the pressure set point.

3)Overshoot in the flow during an expiration should be below the trigerring value 2 L/min.

Now , How to design the linear controller ? We will first have to convert the requirements into step response requirement to be able to design a controller using Root Locus .

The Overshoot requirement is for Q_{pat}.Lets look at the following graph .



We have crossed the Overshoot limit in Q_{pat} for $K_i = 10$ in which case there is a significant overshoot in P_{aw} response too.

Furthermore , Q_{pat}, P_{aw} *and* P_{lung} are related by the following equations

$$Q_{pat} = \frac{P_{aw} - P_{lung}}{R_{lung}}$$

$$and \frac{dP_{lung}}{dt} = \frac{1}{C_{lung}} Q_{pat}$$

We can roughly say if we can control Overshoot in P_{aw}then we can control the overshoot in Q_{pat} too .

For Ki=10 , The peak was ~21.4 ➔ $\frac{1.4}{20-5} = 9.33\%$ Overshoot Which was too much for us.

We will design for 1% overshoot i.e. $\zeta = .8261$

Now , Next requirement is a faster rise time i.e. rise time < 200 ms.

Its difficult to design for Rise time . But we know Rise time is smaller than Peak time . So, We will take conservative approach . We will design for Peak time $T_p < \frac{200}{15}\ ms$.

We normalize by 15 since 200 ms rise time is expected for 15 step increase.

Now ,

$$\frac{\pi}{\omega_d} < \frac{200 * 10^{-3}}{15}$$

$$=> \omega_d > 235.619$$

Since we want to operate on $\zeta = .8261$ line , We want

$$\sigma_d = \frac{235.619}{\tan(\cos^{-1}.8261)} = 345.307$$

So , Our desired operating point is -345+j235 .

Poles of our pressure open loop system are -188.5 , -188.5 and -5.44.

Zeros of our pressure open loop system are at -10 .

Angle contribution at desired operating point = -247.69.

Angle contribution needed from zero = +67.69

$$\tan(67.69) = \frac{235.619}{z - 345.407}$$

$$==> z = 442.1$$

So , Our PD controller will be (s+442.1).

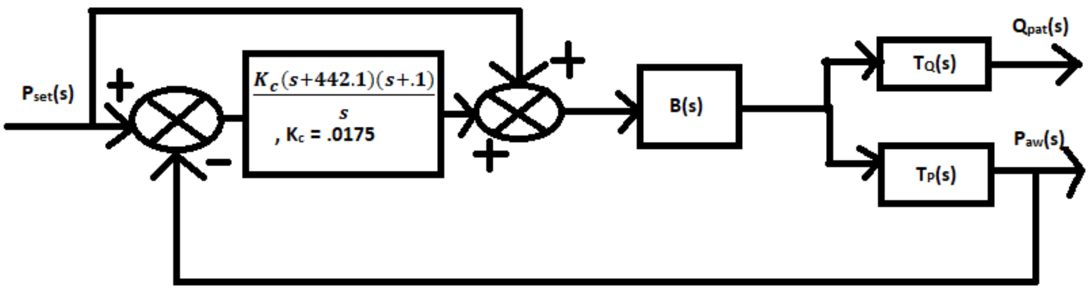
Since we want low steady state error , we will also add a PI controller of transfer function $\frac{s+.1}{s}$.

So , Our Controller is $\boldsymbol{\frac{K_c(s+442.1)(s+.1)}{s}}$.

We find K_c,Gain needed to operate on desired point using rlocfind and we find $K_c \cong .0175$.

We keep the feedforward path since we know it will help in decreasing rise time .

So overall our desired linear system is –

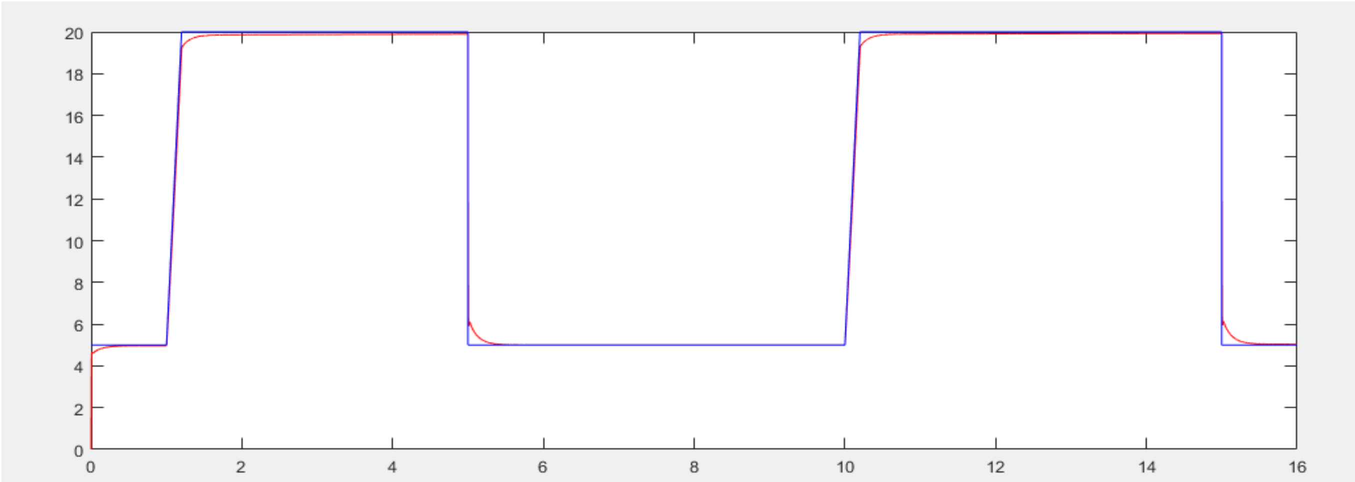


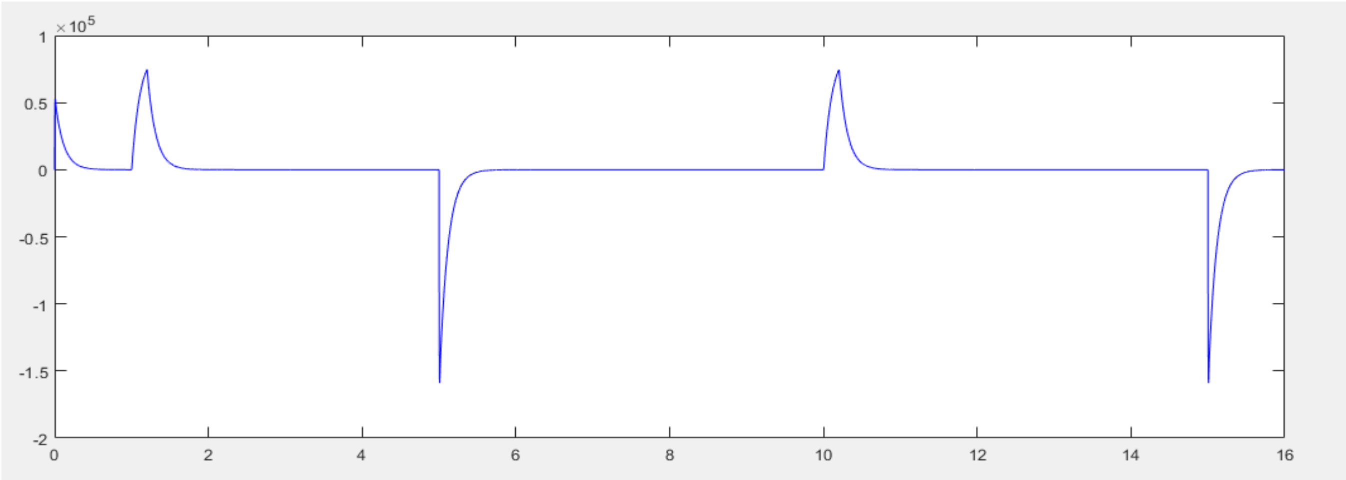
We simulate the system both in Matlab Code and in Simulink .

Matlab Code is attached below to give clear illustration of our work.

```
%%
s = tf('s');
B = wn^2 / (s^2+2*wn*s+wn^2);
T = (Dh1*(s-(Ah-(Ch1*Bh)/Dh1)) / (s-Ah))*B;
controller = ((s+.1)*(s+442))/s ;
rlocus(T*controller);
sgrid(.8261,0);
[k poles] = rlocfind(T*controller);
%% k= .0175 can be a default good choice if you are unable to adjust
sys = feedback(k*controller*T,1) + feedback(k*controller*T,1)/(k*controller)
;
[y t x] = lsim(sys , pset , t);
figure ; plot(t,y,'r',t,pset,'b');hold on;
%%
T_f = (Dh2*(s-(Ah-(Ch2*Bh)/Dh2)) / (s-Ah))*B;
%%
sys = feedback(k*controller*T,1) + feedback(k*controller*T,1)/(k*controller)
;
[y t x] = lsim(sys , pset , t);
sys_f_1 = (1+controller*k)*T_f ;
sys_f_2 = -controller*k*T_f;
[y_f1 t x] = lsim(sys_f_1 , pset , t);
[y_f2 t x] = lsim(sys_f_2 , y , t);
figure;
plot(t,(y_f1+y_f2)*60 , 'b'); hold on;
```

We Get following pressure and air flow curves .





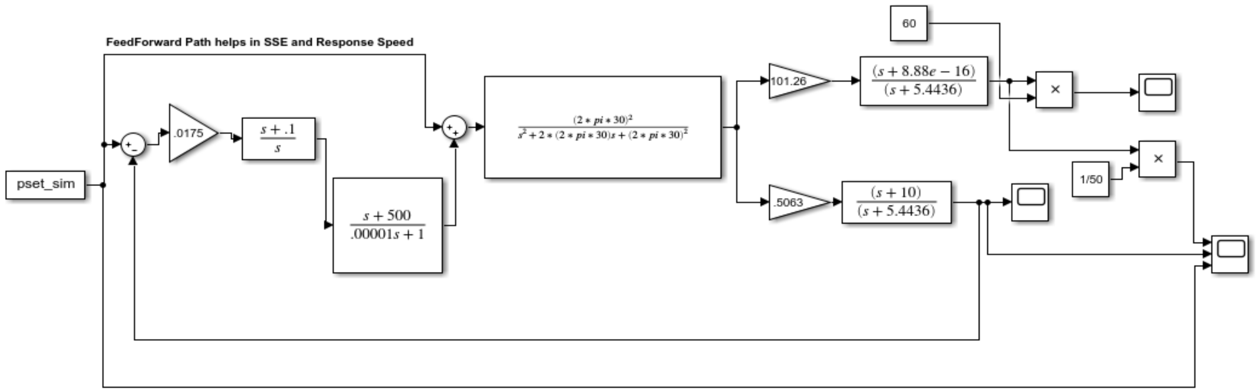
Using our designed controller –

1)Rise time ~= 160 ms

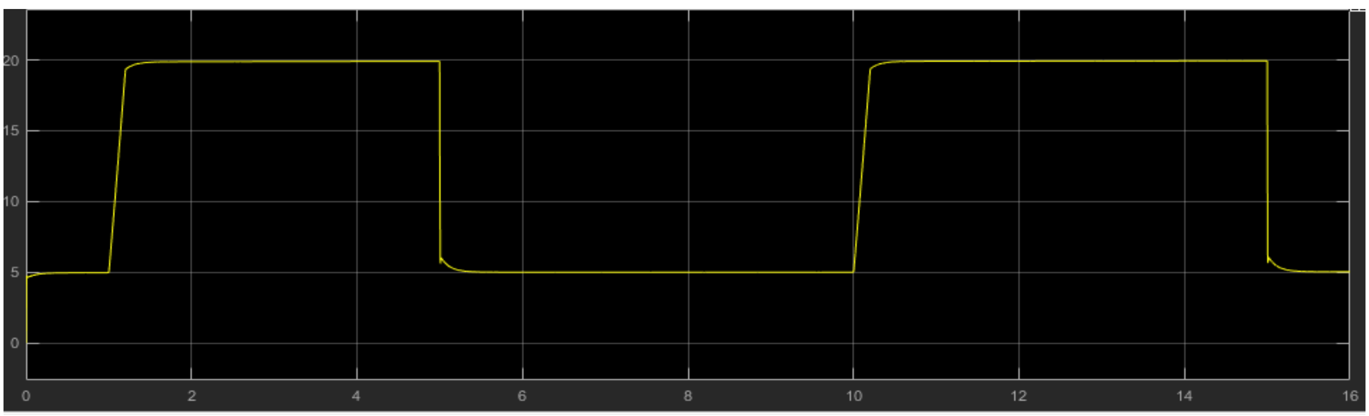
2) Plateau pressure is 19.87 mbar . (Only 0.13 mbar below pressure set point).

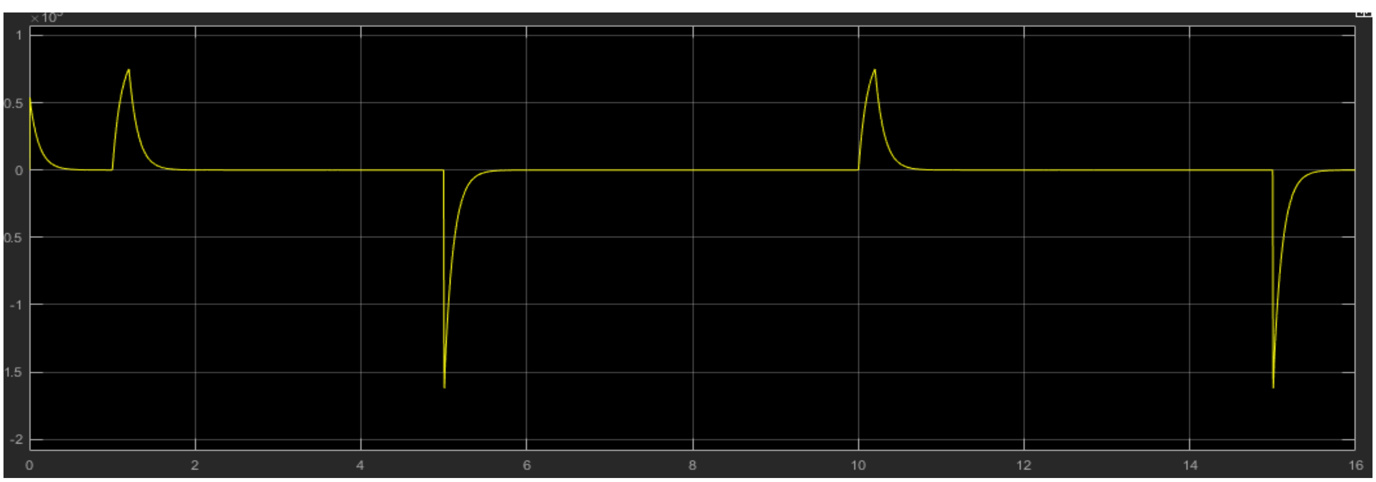
3)Very negligible overshoot in air flow curve during expiration .

We also simulated the system in Simulink to see if it works on real life .



Corresponding system pressure and air flow curves –





We see that the requirements are fulfilled from Simulink response curves.

Task 7 :Reproduce the results shown in Fig. 14 for linear and variable gain controllers. What are the pros and cons of nonlinear control over linear control?

Simulink Block Diagram:

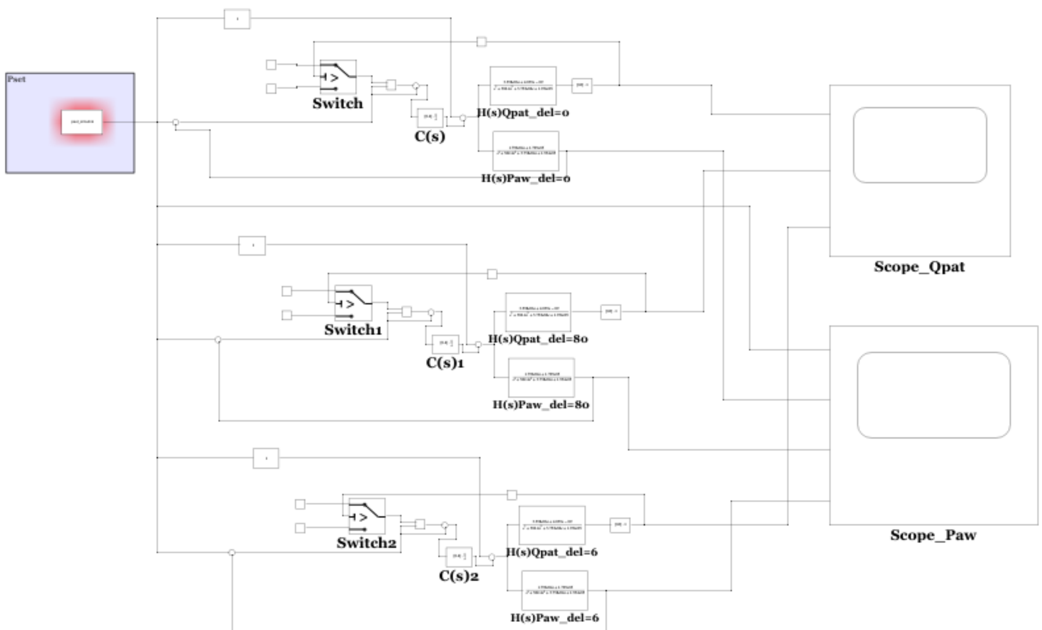


Fig : Simulink Diagram

Desired Outcome:

- Outcome for Paw(airway pressure)

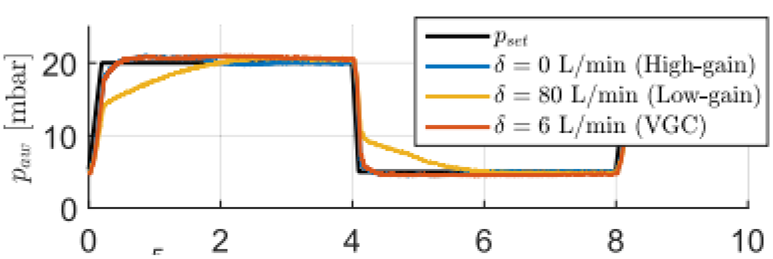


Fig : Paw for different delta/threshold value

- Outcome for Qpat(patient flow response)

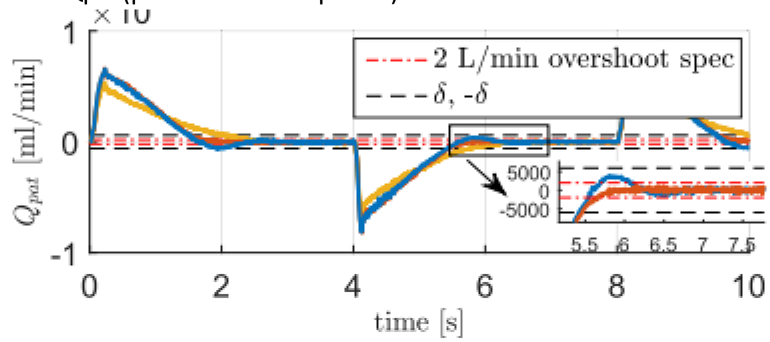


Fig. 14. Experimental time-domain response of the linear controllers and a variable-gain controller with $\delta = 6$ L/min. Note that $\delta = 0$ represents the linear high-gain controller and $\delta = 80$ L/min represents the linear low-gain controller.

Fig : Qpat for different delta/threshold value

Reproduced Outcome:

- Reproduced outcome for Paw:

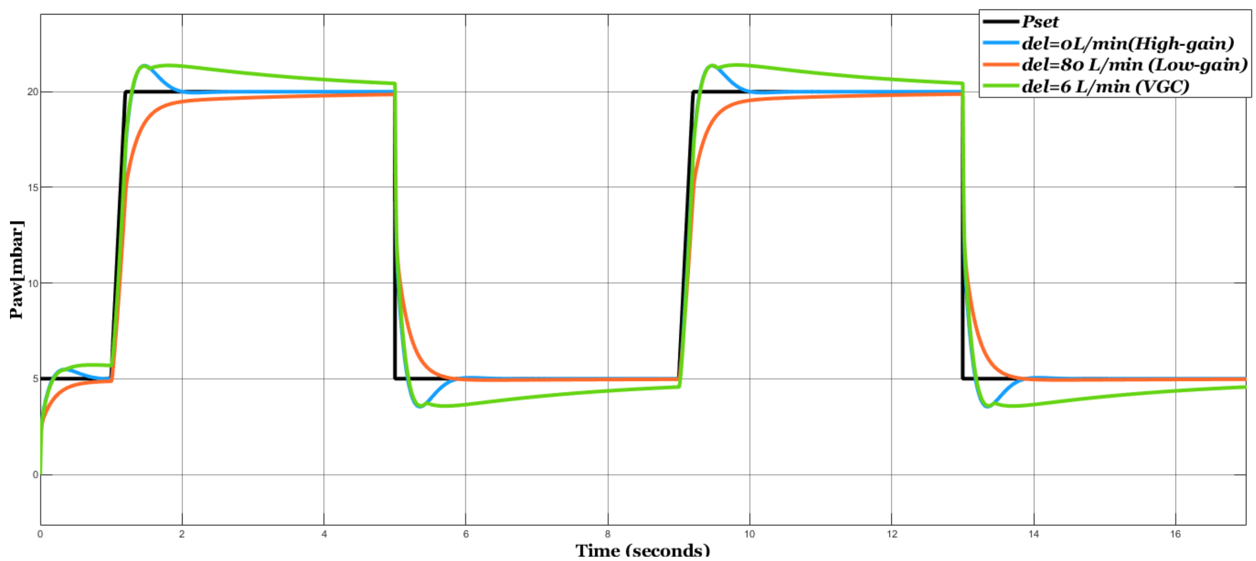


Fig : Paw for different delta/threshold values

➢ Reproduced outcome for Qpat:

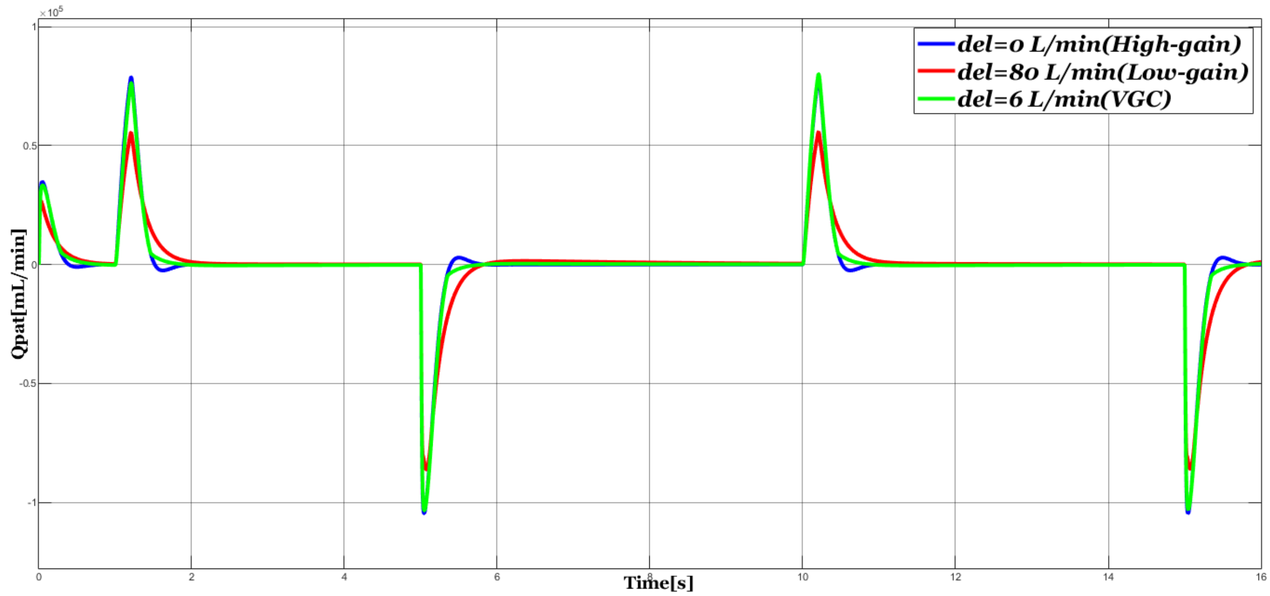


Fig : Qpat for different delta/threshold values

Pros of Nonlinear Controller over Linear Controller:

- The tradeoff criteria between fast pressure buildup and overshoot in the flow response is perfectly met by the nonlinear controller.
- The switching length delta is so robust that it remains stable for different lung characteristics.
- This variable gain controller limits the amount of patient-flow overshoot by switching to the low-gain controller which in not expected from linear controller

Cons of Nonlinear Controller over Linear Controller:

- The nonlinear controller is more optimized for the Qpat response since it basically switches its value based on some threshold Qpat values preset in its block.
- Indeed there maybe some undesirable response in Paw
- From the paper's perspective this may not be a con. But if we vary lung ans system parameters this con has some significant effect.

Task 8 : **Discuss the performance of both linear and nonlinear control systems in presence of uncertainties such as different lung parameters, pressure drop etc.**

❖ Performance of Linear Controller:

Effect of varying Clung

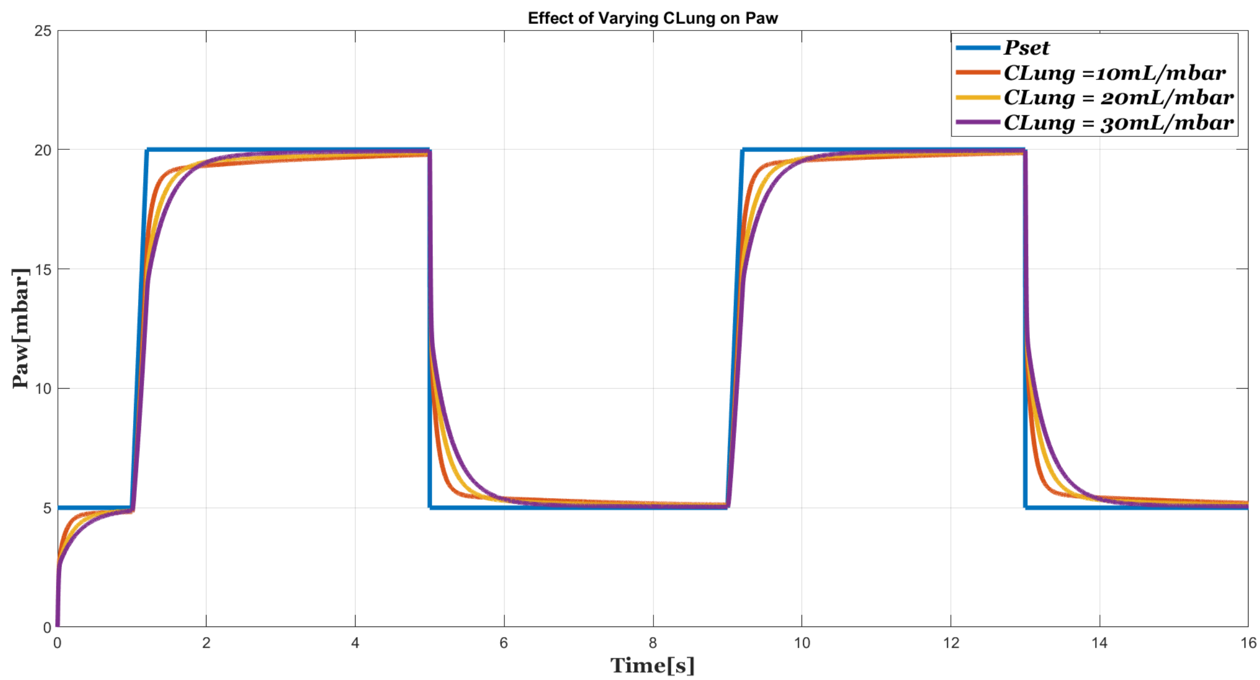


Fig : Paw for different Clung

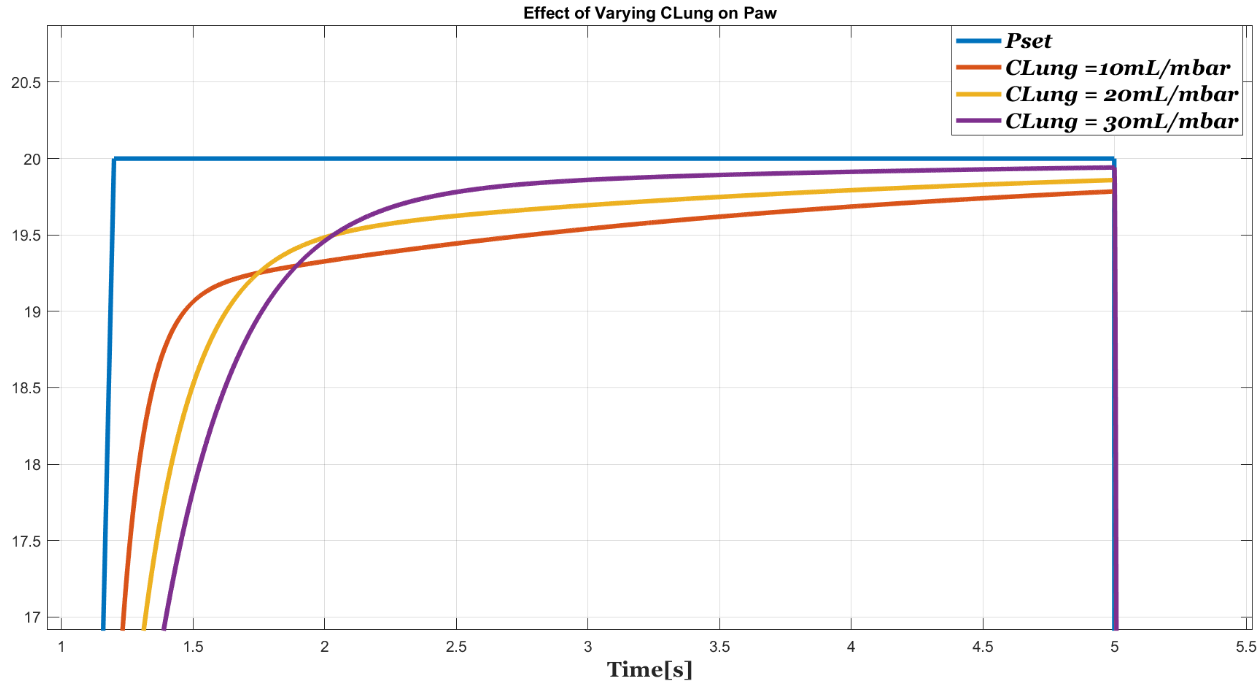


Fig : Observation of Rise Time & Paw(max) for different Clung

Comment :

- ✓ Rise time increases with the increase of Clung.
- ✓ Paw(max) increases with the increase of Clung.

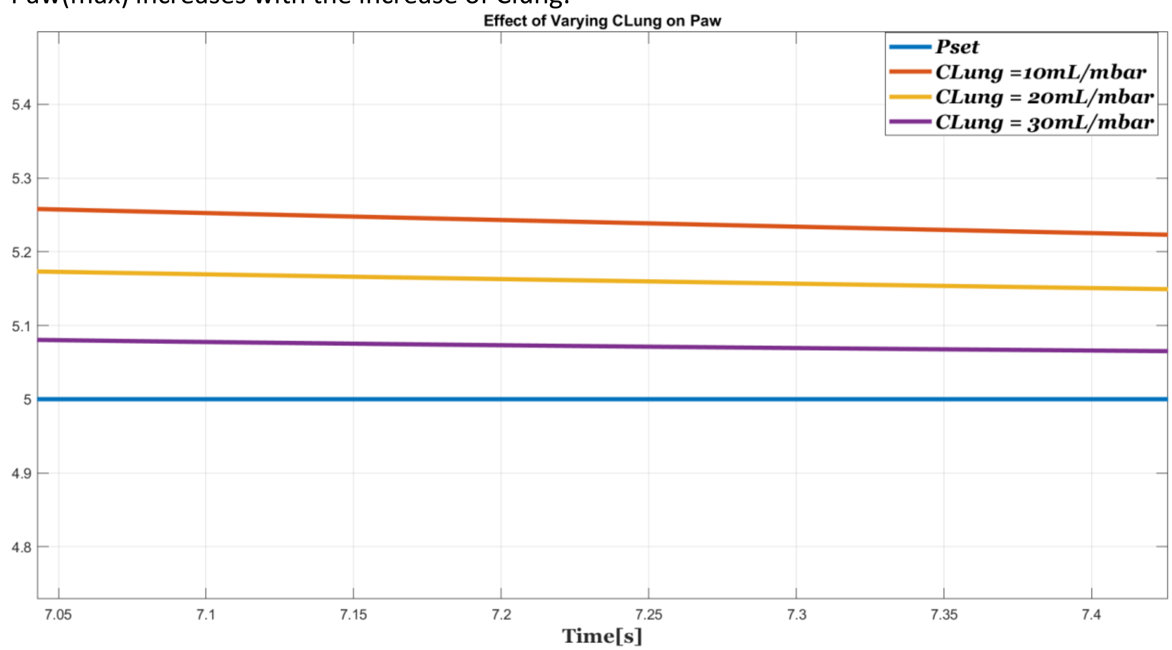


Fig : Observation of Steady State Error(SSE) for different Clung

Comment :

- ✓ SSE decreases with the increase of Clung.

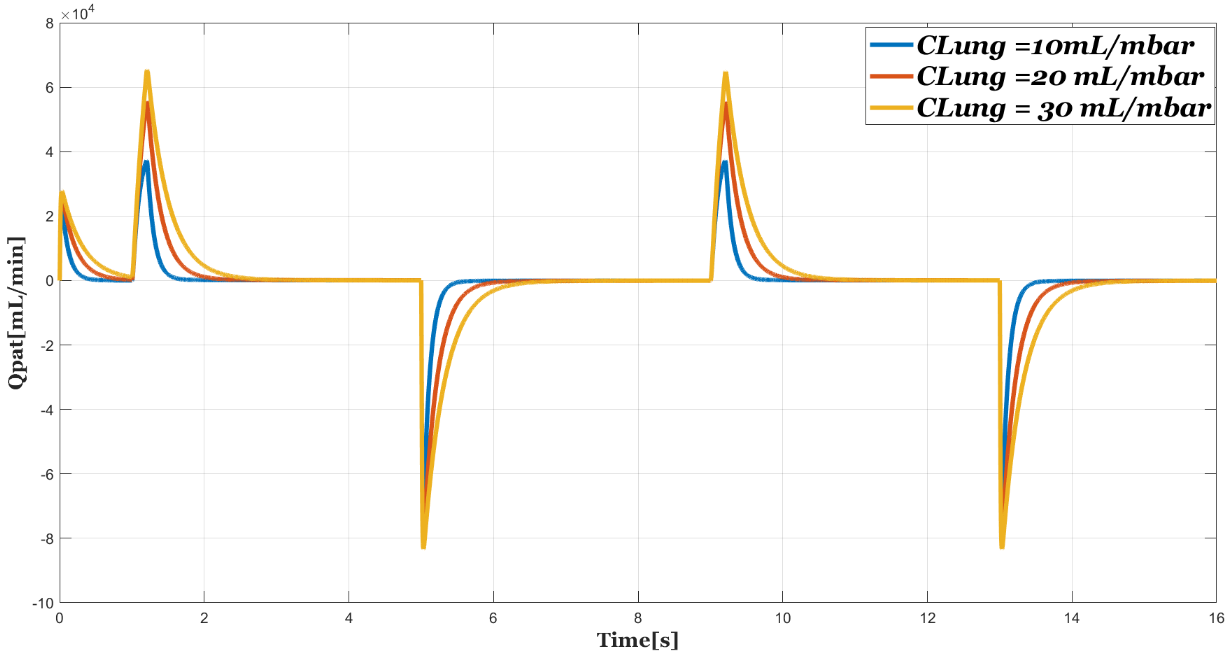


Fig : Qpat for different Clung.

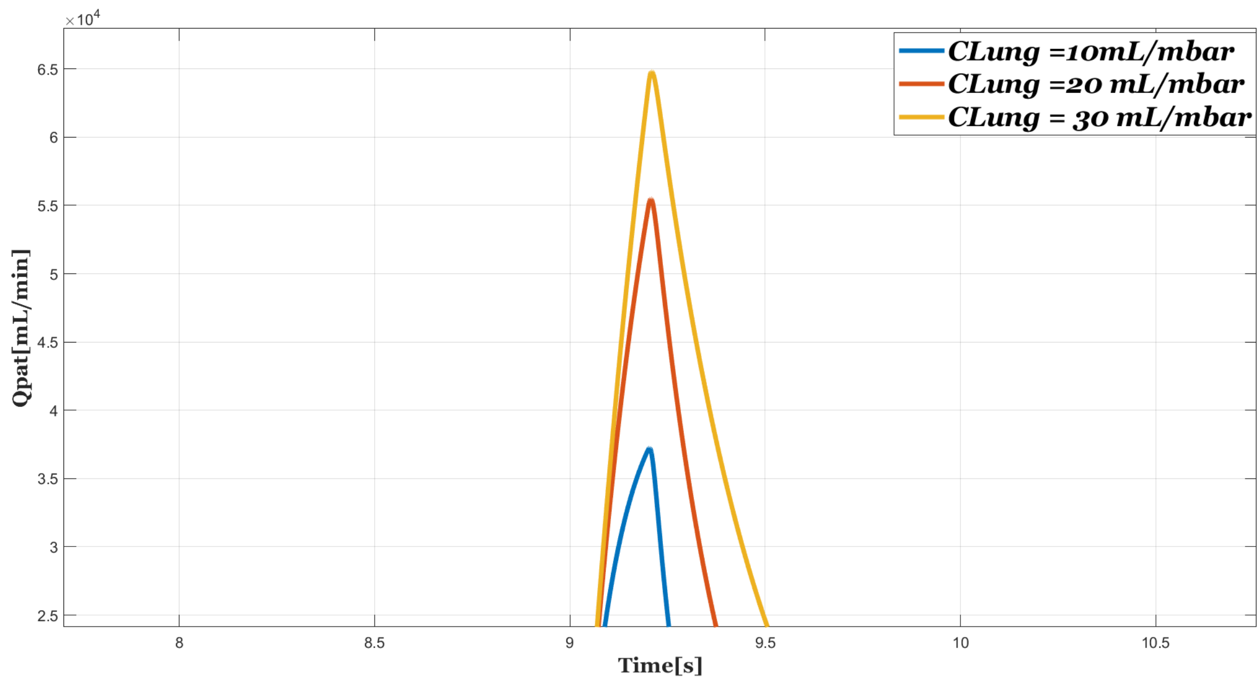


Fig: Observation of Qpat(max) for different Clung

Comment :

✓ Qpat(max) increases with the increase of Clung.

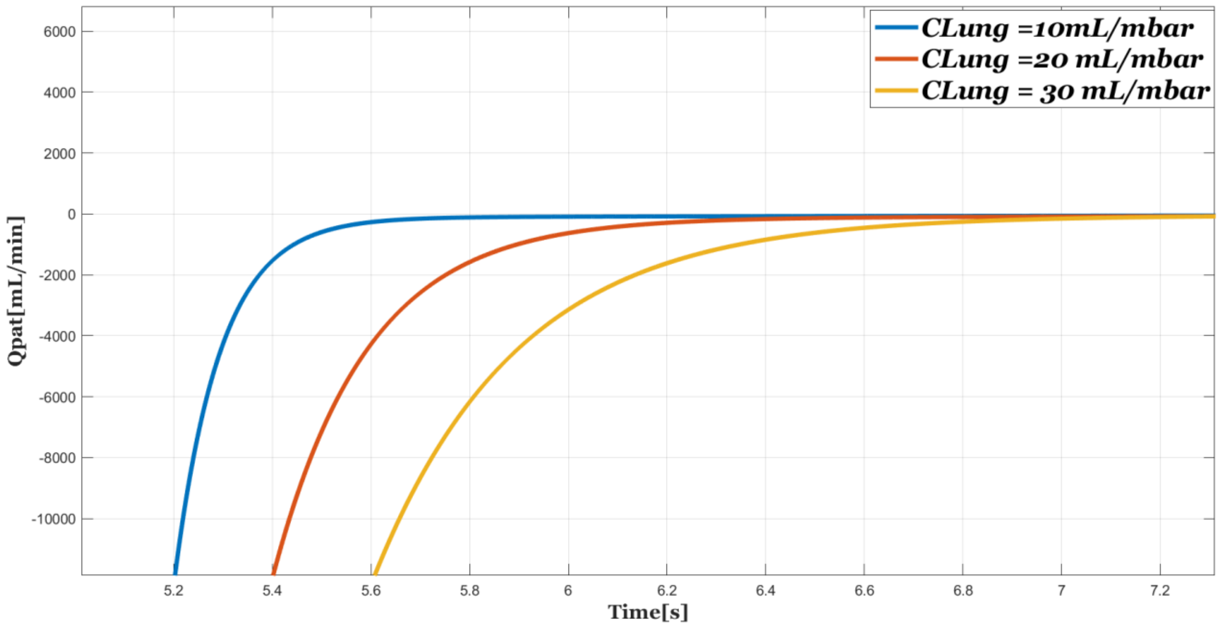


Fig : Observation of settling time for different Clung

Effect of varying RLung

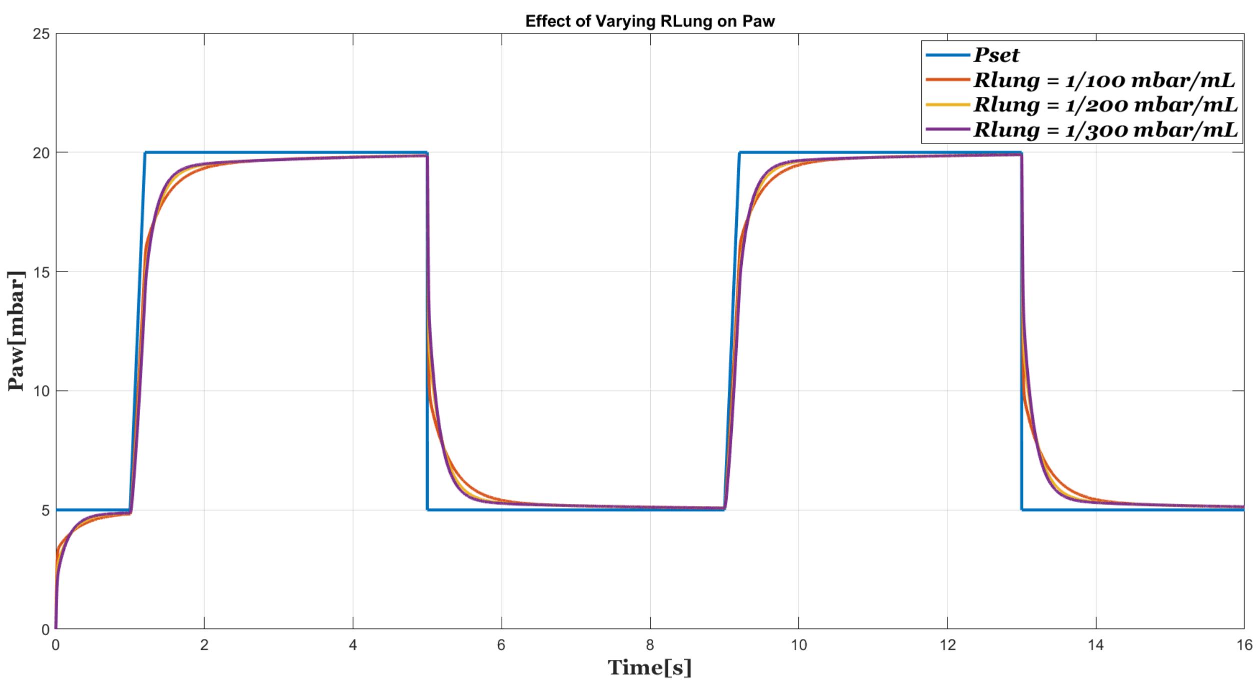


Fig : Paw for different RLung

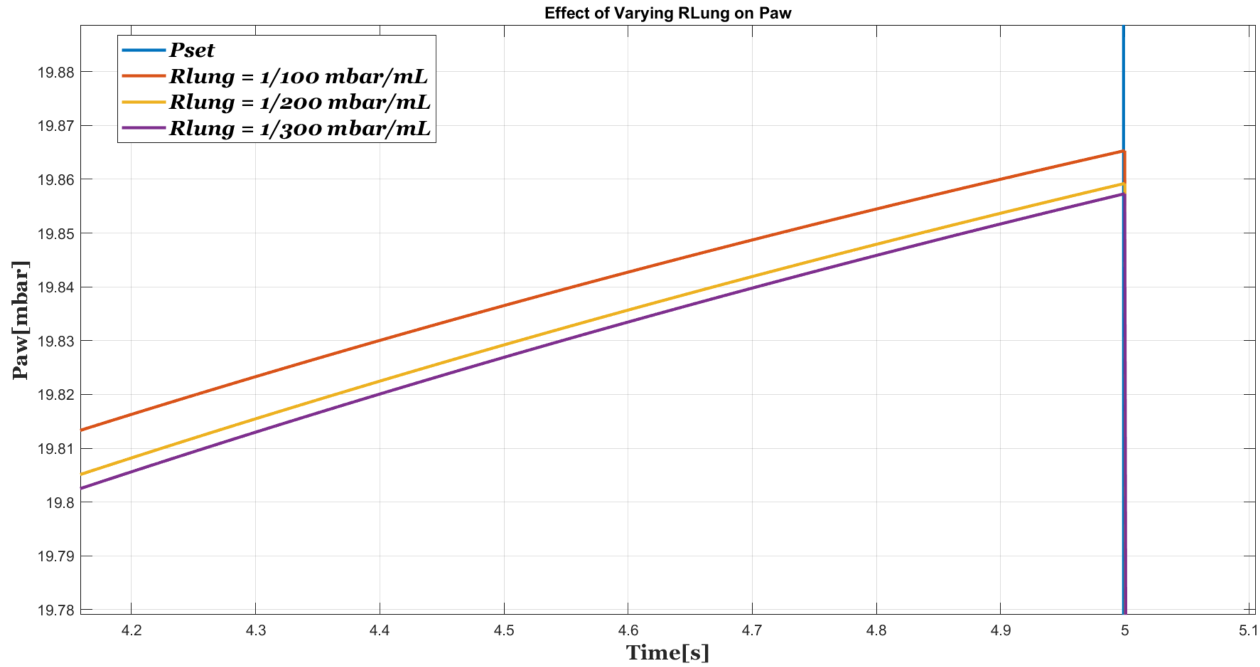


Fig : Observation of Paw(max) for different RLung

Comment:

- ✓ With the increase of GLung (decrease of Rlung) Paw(max) decreases.

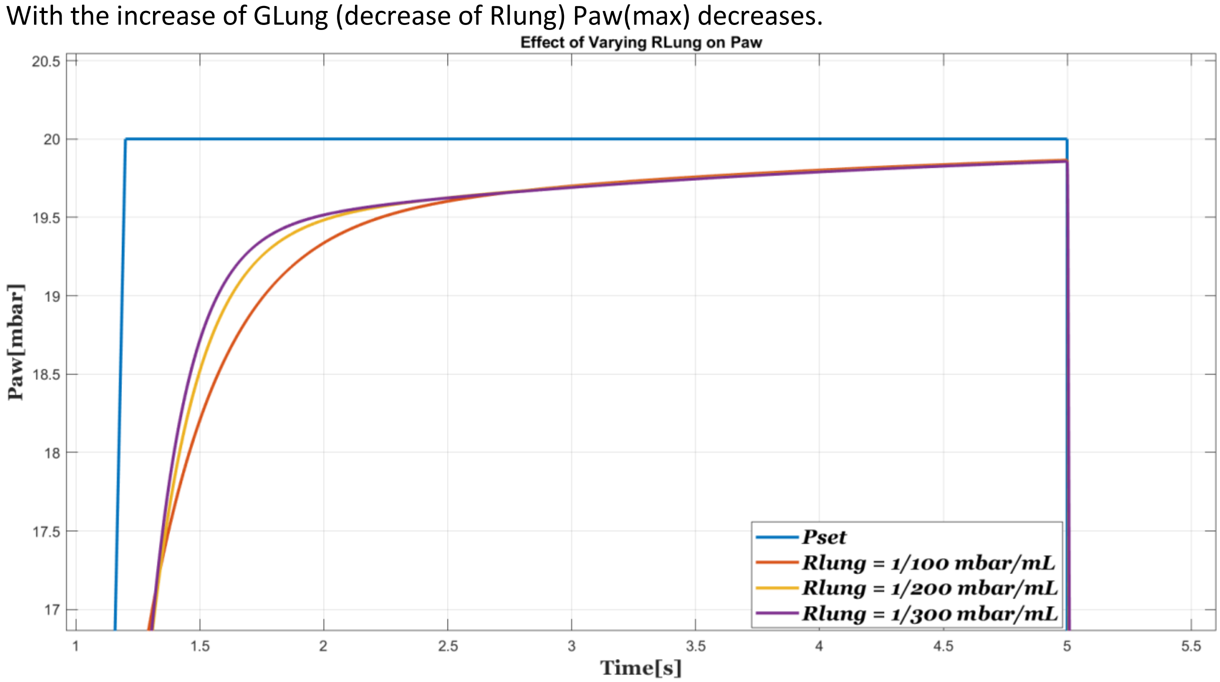


Fig: Observation of rise time for different RLung

Comment:

- ✓ With the increase of GLung (decrease of Rlung) rise time decreases.

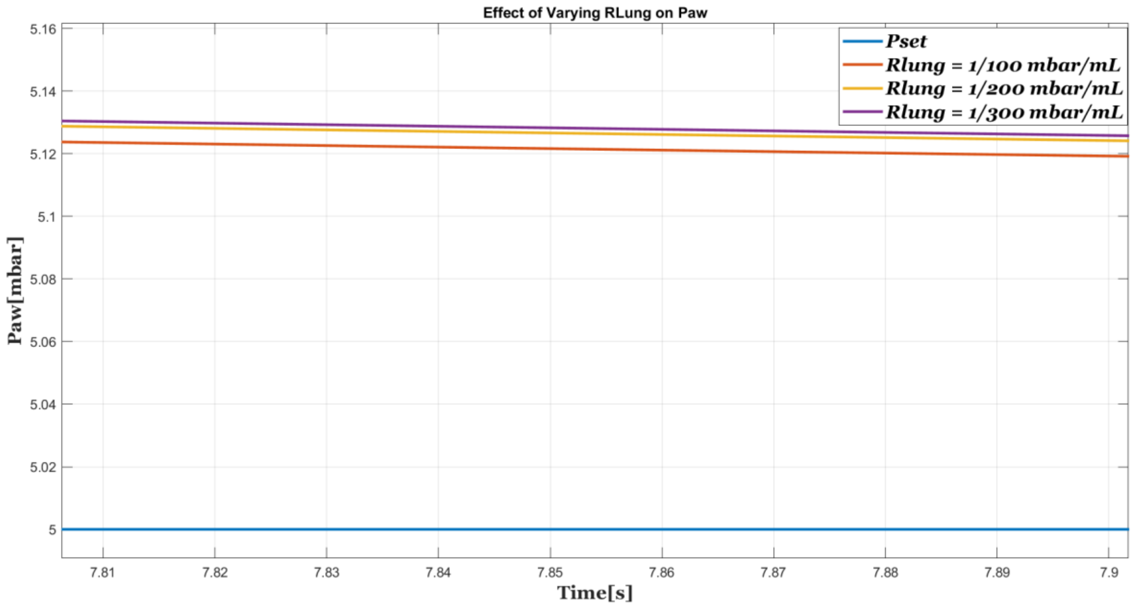


Fig: Observation of SSE for different RLung

Comment:

- ✓ With the increase of GLung (decrease of Rlung) SSE increases.

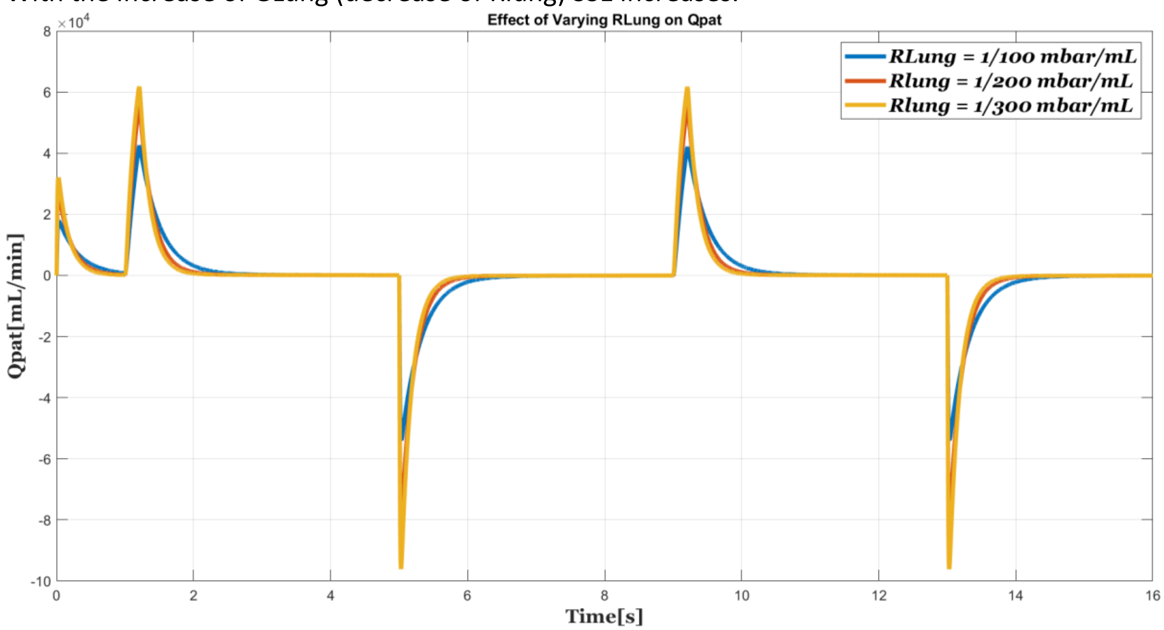


Fig: Qpat for different RLung

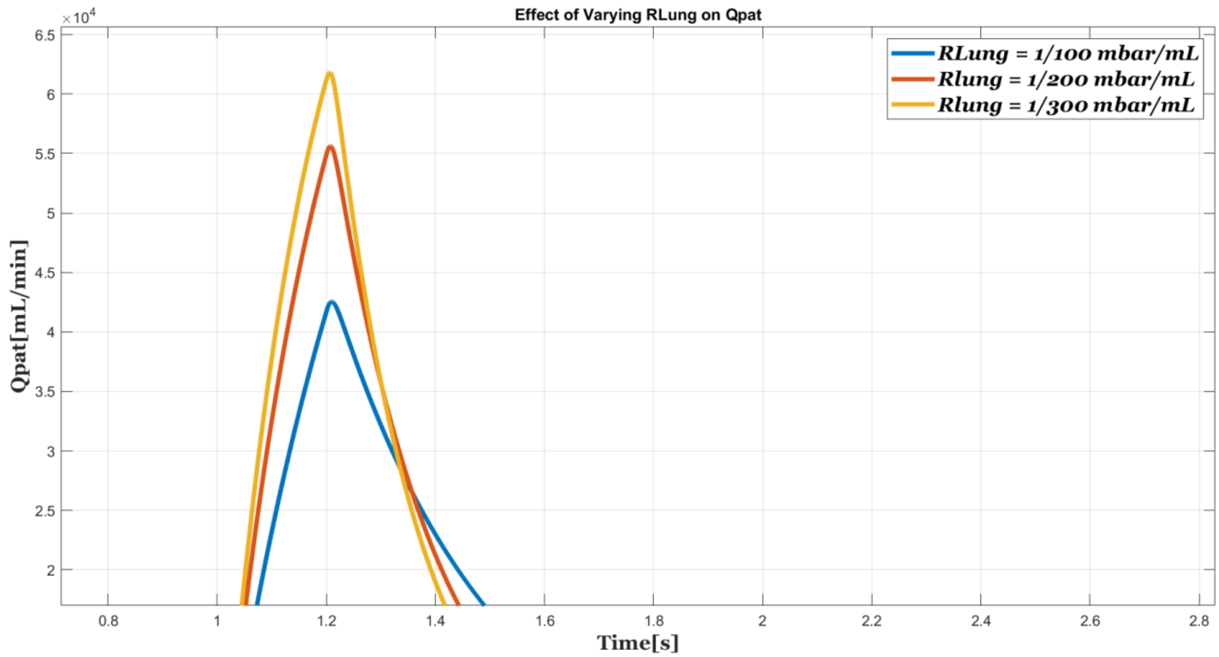


Fig: Observation of Qpat(max) for different RLung

Comment:

✓ With the increase of GLung (decrease of Rlung) Qpat increases.

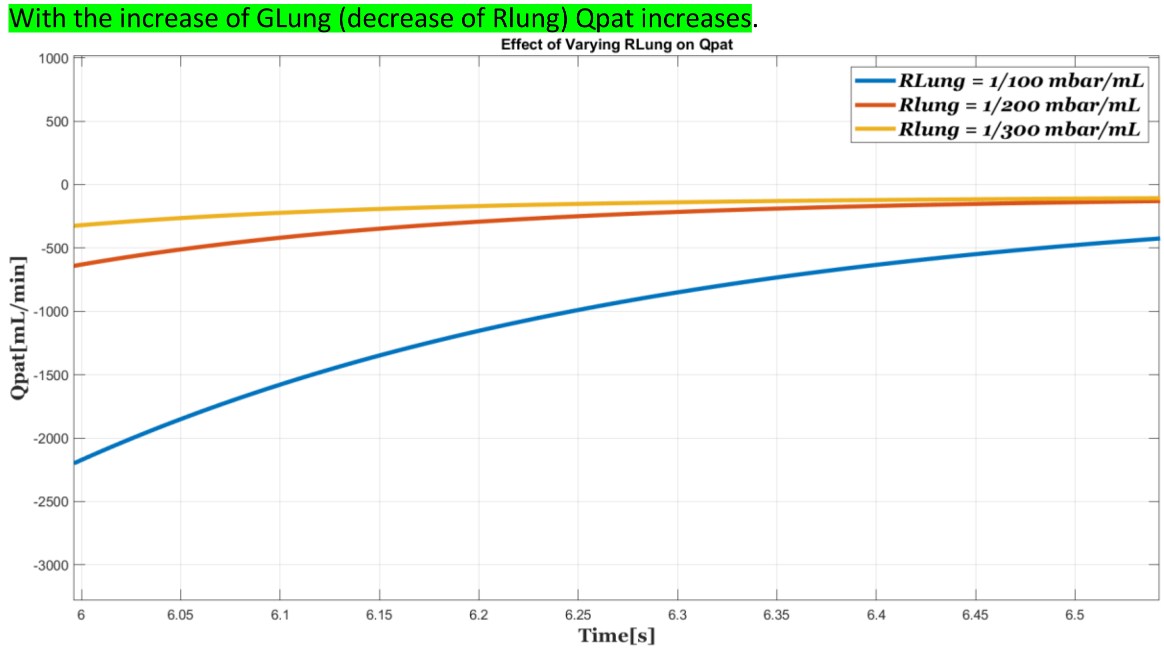


Fig: Observation of settling time for different RLung

Comment:

✓ With the increase of GLung (decrease of Rlung) settling time decreases.

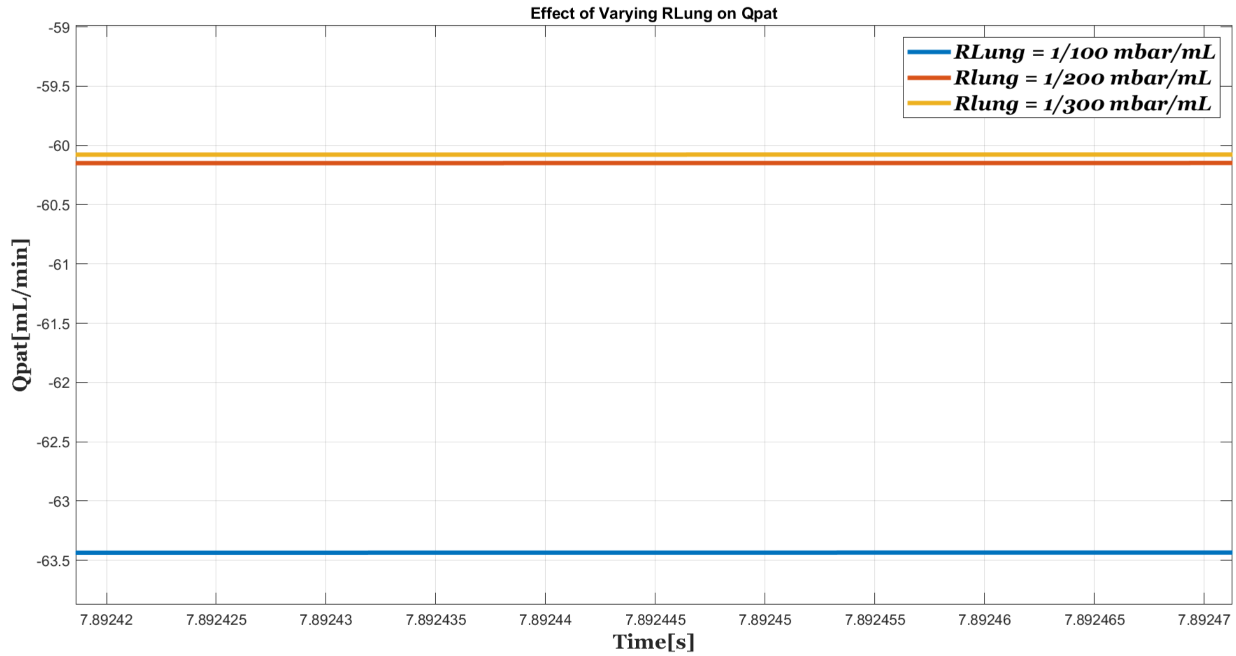


Fig: Observation of SSE for different RLung

Comment:

- ✓ With the increase of GLung (decrease of Rlung) SSE decreases.

Effect of varying RLeak

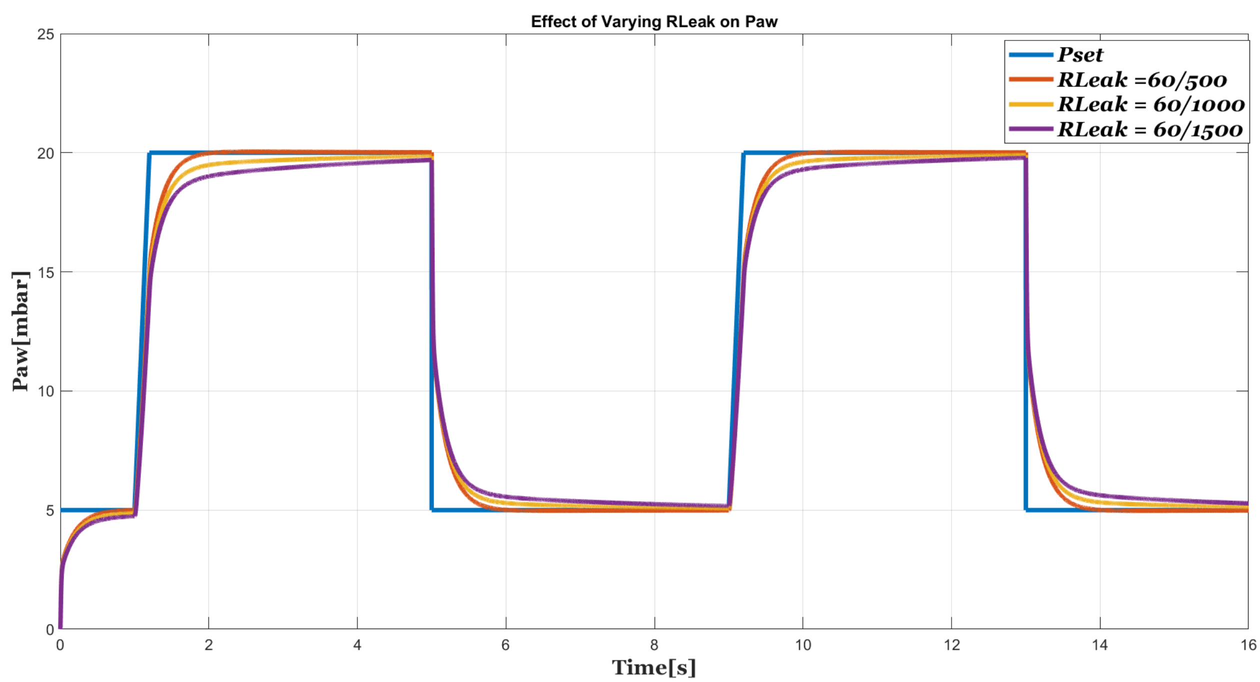


Fig : Paw for different Rleak

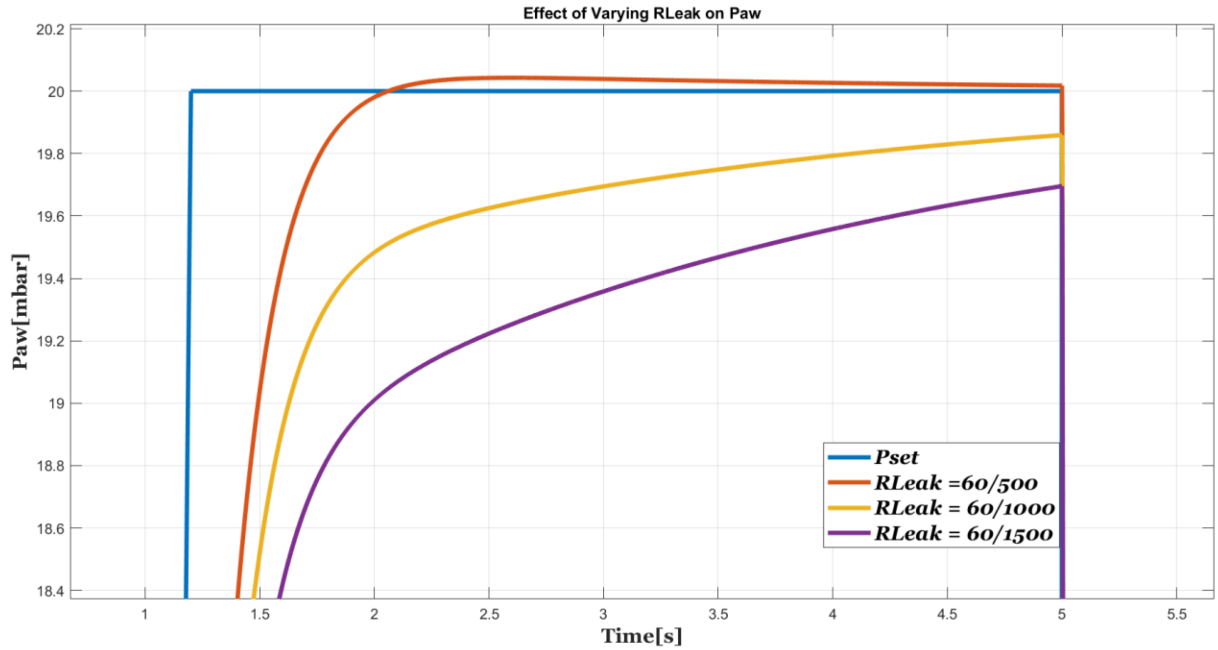


Fig: Observation of rise time & Paw(max) for different RLeak

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) peak value decreases.
- ✓ With the increase of GLeak(decrease of RLeak) rise time decreases.

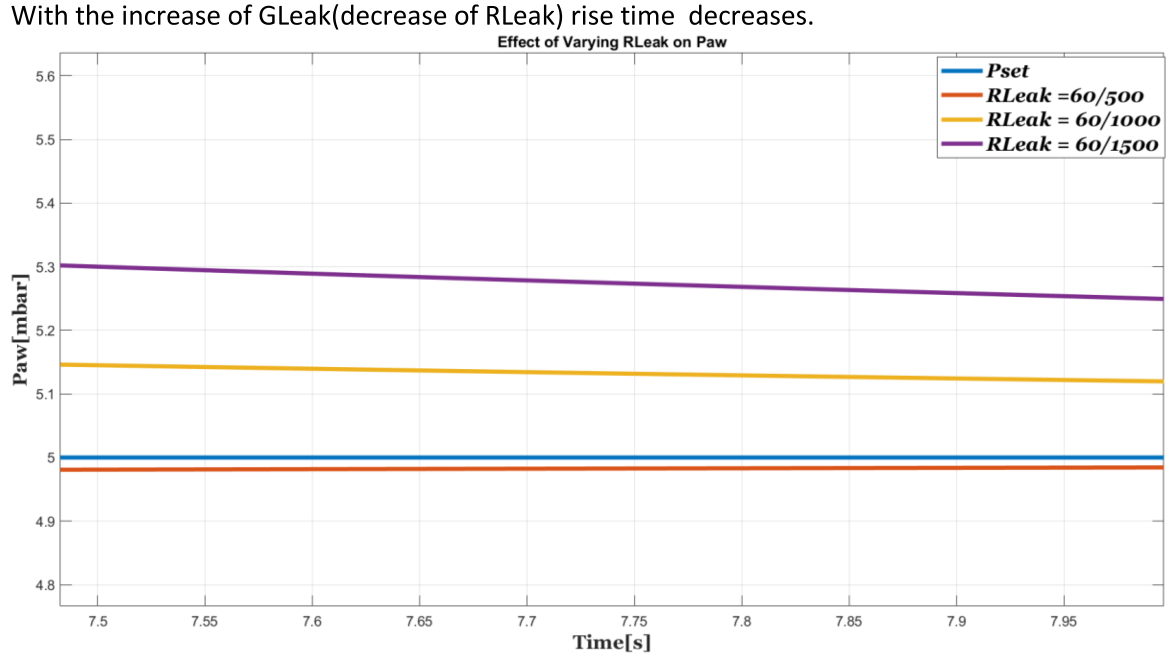


Fig: Observation of SSE for different RLeak

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) SSE increases.

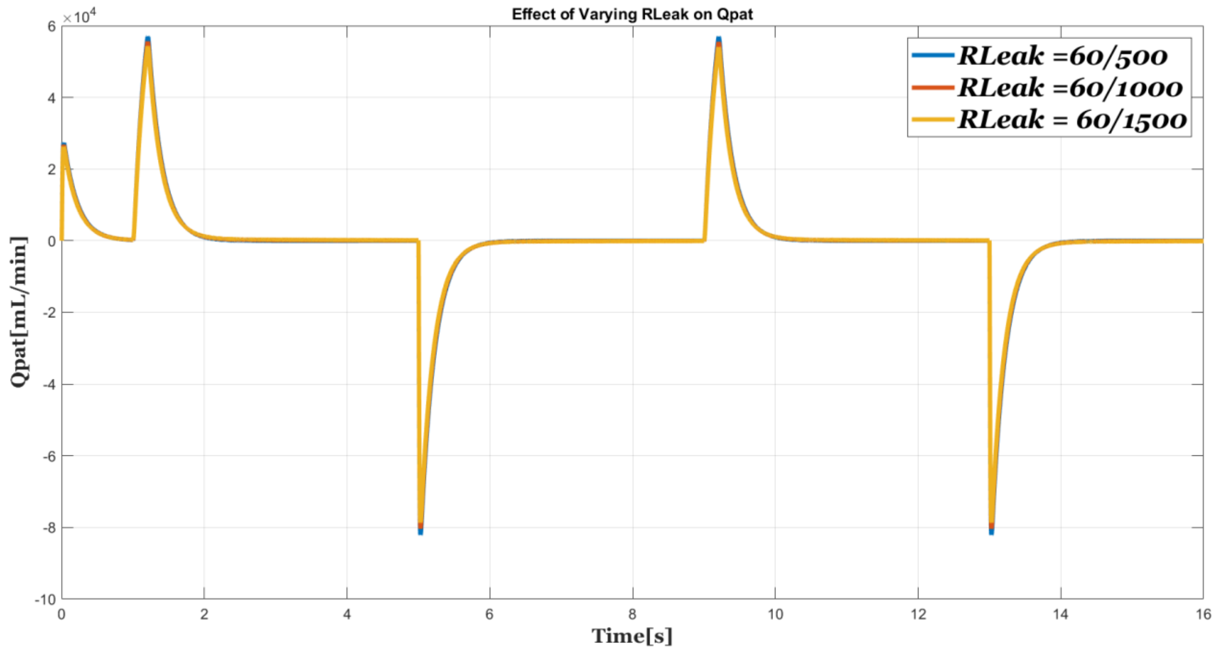


Fig: Qpat for different Rleak

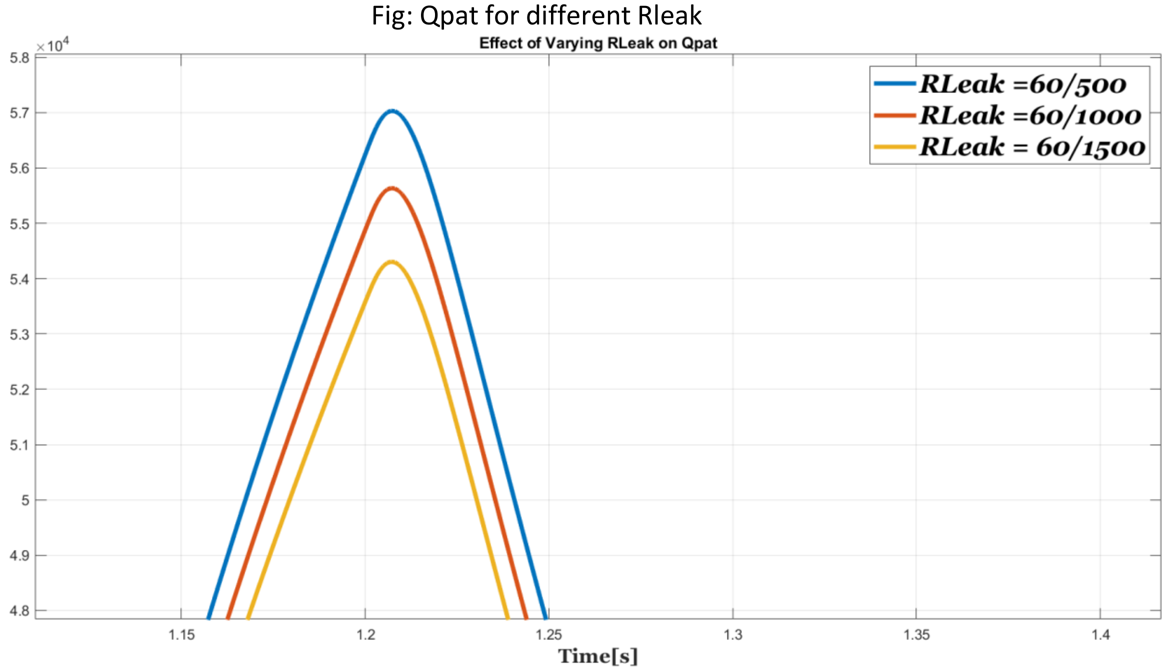


Fig: Observation of Qpat(max) for different RLeak

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) Qpat(max) decreases.

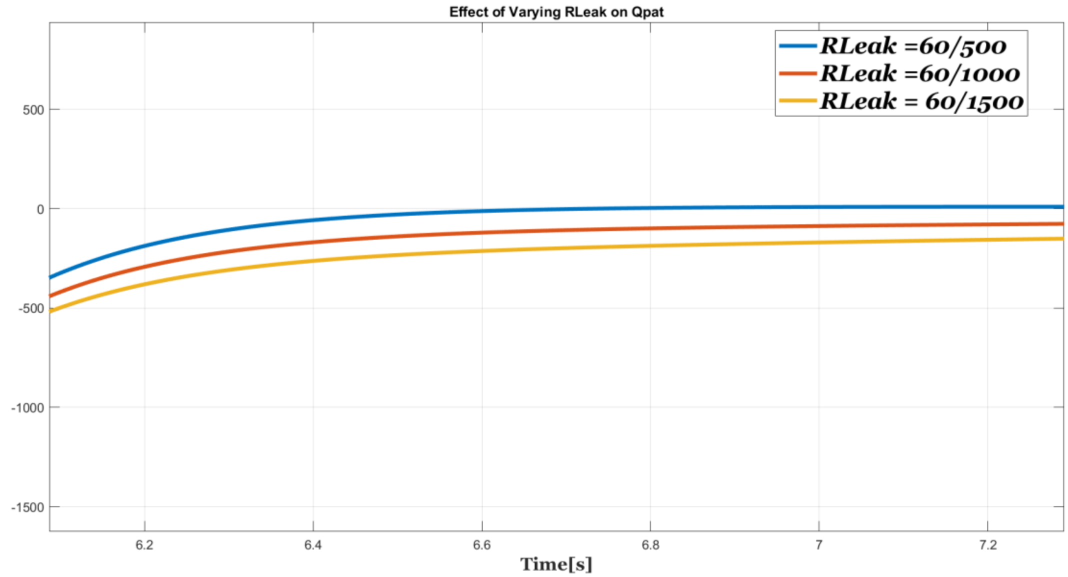


Fig: Observation of settling time for different RLeak

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) settling time decreases.

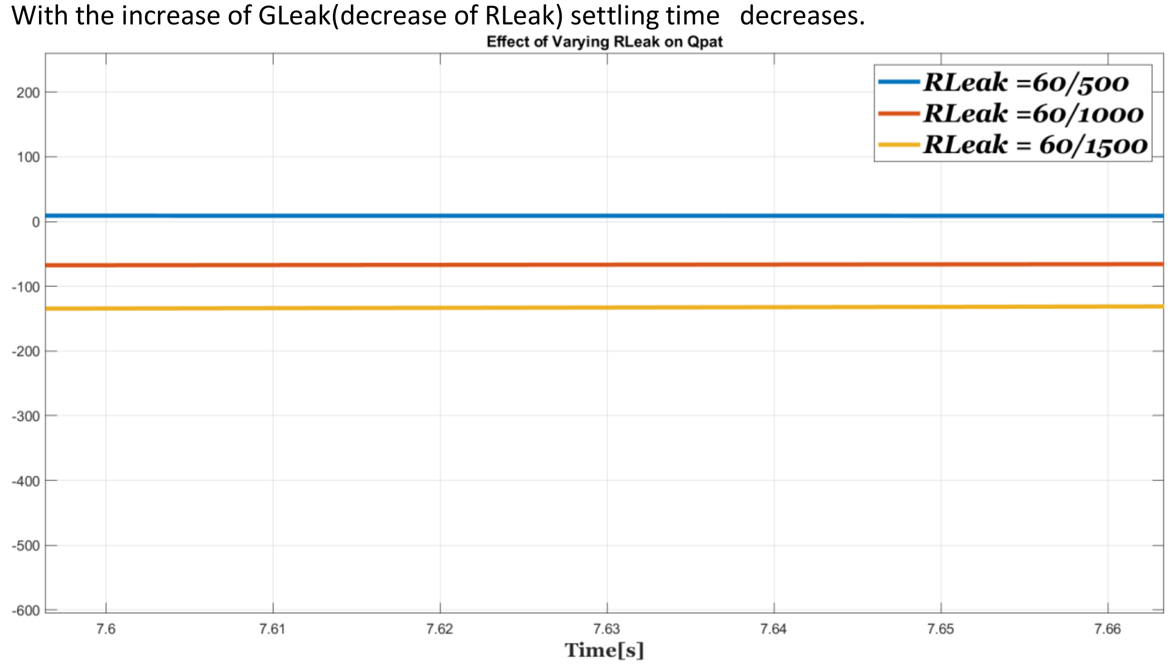


Fig: Observation of SSE for different RLeak

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) SSE increases.

Effect of varying RHose

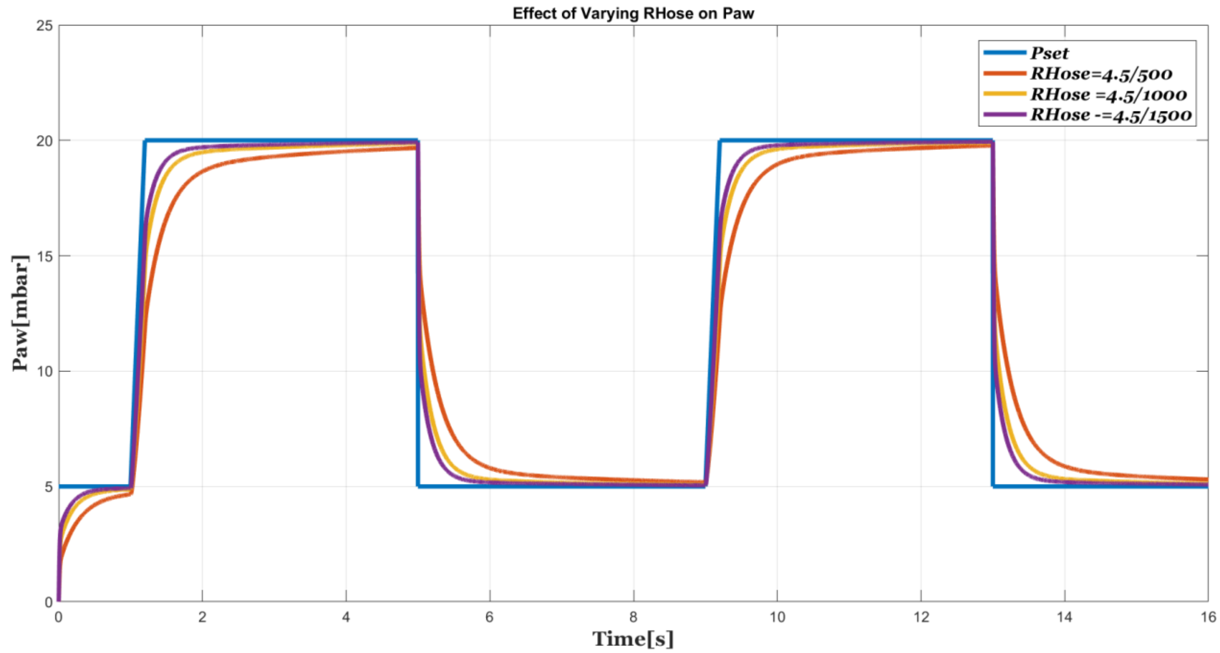


Fig : Paw for different RHose

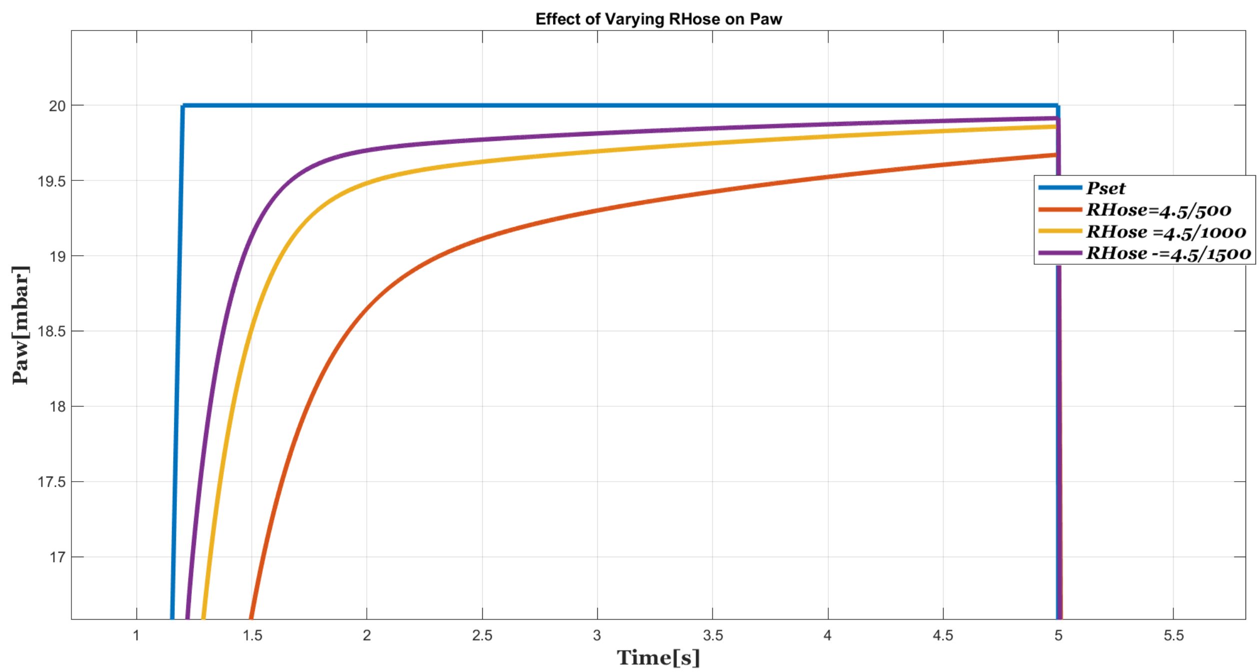


Fig: Observation of rise time & Paw(max) for different RHose

Comment:

- ✓ With the increase of GHose(decrease of RHose) rise time decreases & Paw(max) increases.

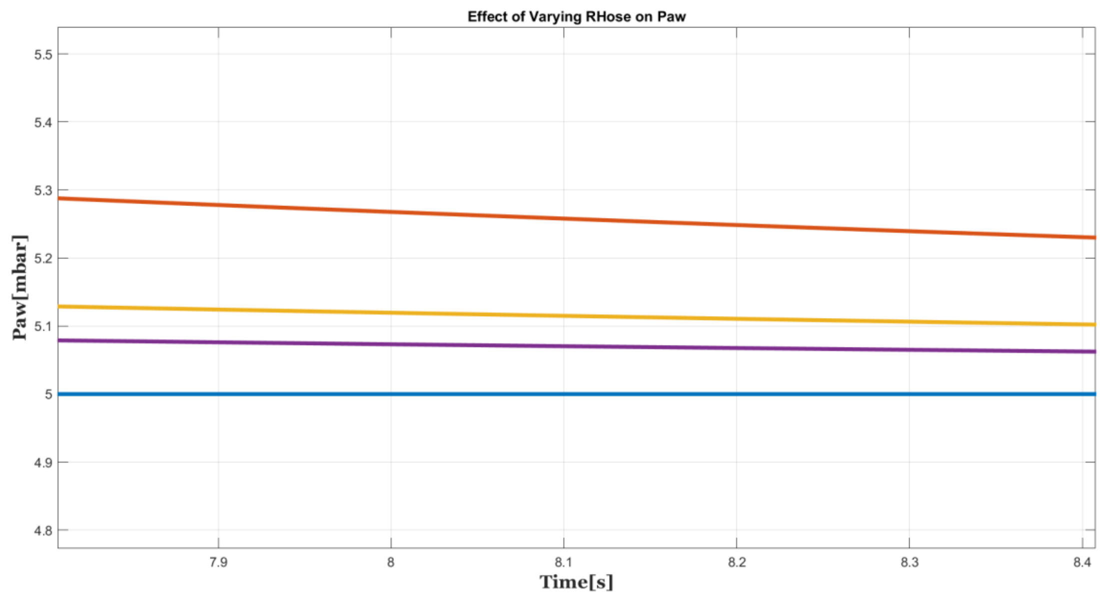


Fig: Observation of SSE for different RHose

Comment:

- ✓ With the increase of GLeak(decrease of RLeak) SSE decreases.

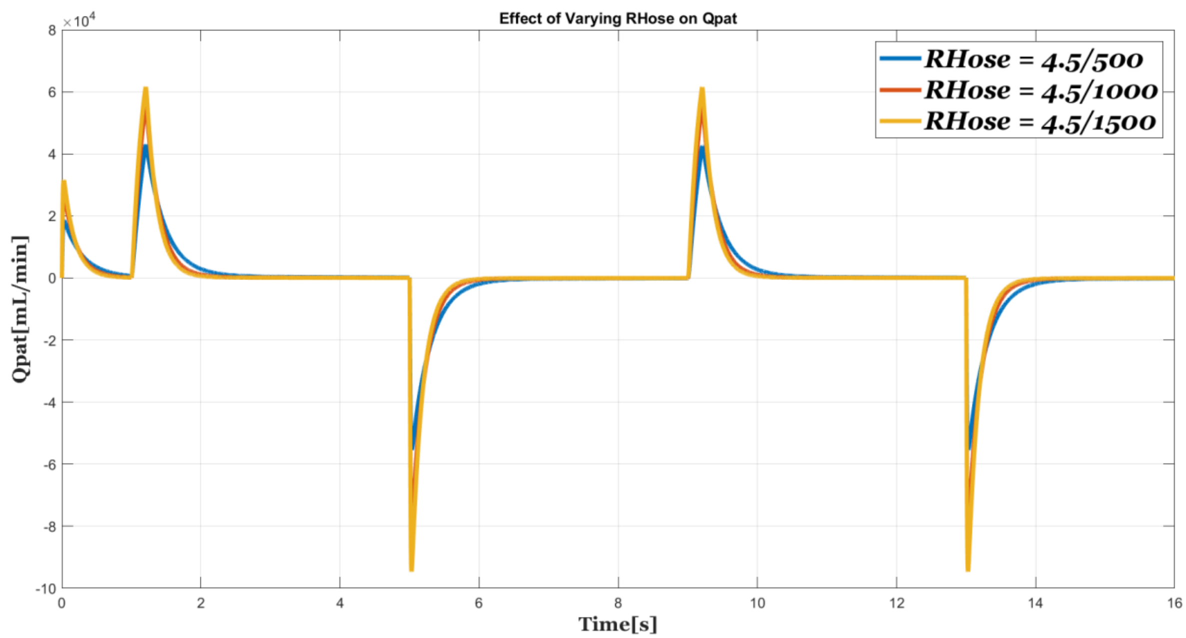


Fig : Qpat for different RHose.

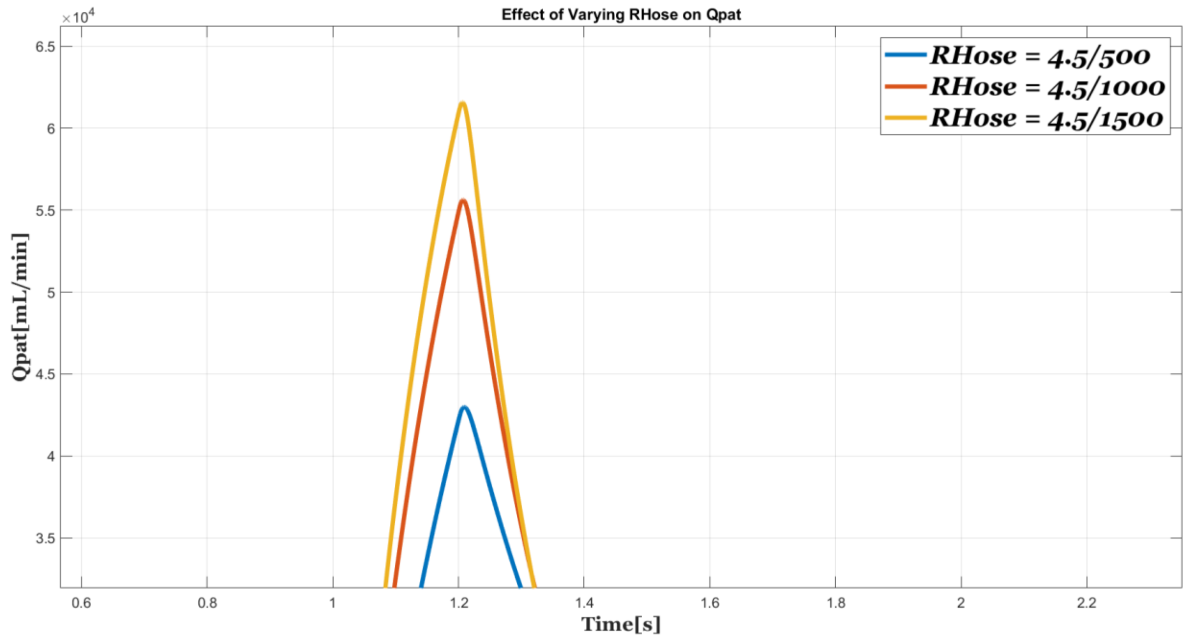


Fig: Observation of Qpat(max) for different RHose

Comment:

- ✓ With the increase of GHose(decrease of RHose) Qpat(max) increases.

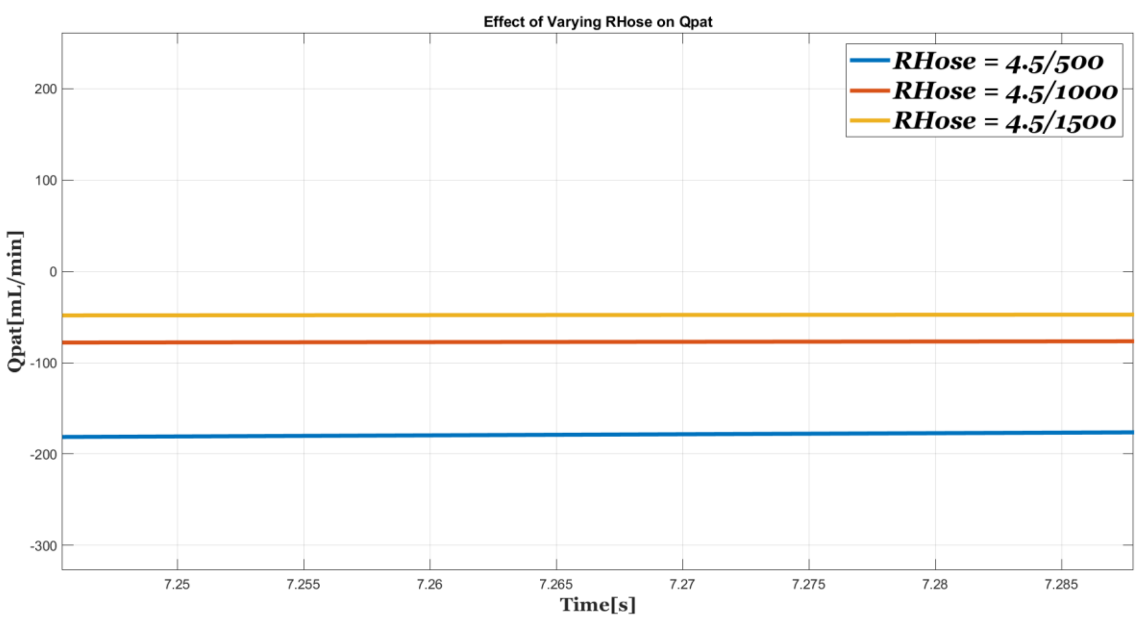


Fig: Observation of SSE for different RHose

Comment:

- ✓ With the increase of GHose(decrease of RHose) SSE decreases

Table Showing Effect on Paw for Varying Parameters(Linear Controller)

Parameters	Paw(max)	Rise Time	Steady State Error
CLung	Proportional	Proportional	Inverse
GLung	Inverse	Inverse	Prportional
GLeak	Inverse	Inverse	Proportional
GHose	Proportional	Inverse	Inverse

Table Showing Effect on Qpat for Varying Parameters(Linear Controller)

Parameters	Qpat(max)	Settling Time	Steady State Error	Overshoot
CLung	Proportional	Proportional	Not any specific relation	Not observed
GLung	Proportional	Inverse	Inverse	Not observed
GLeak	Inverse	Inverse	Proportional	Not observed
GHose	Proportional	Inverse	Inverse	Not observed

❖ Performance of Non-Linear Controller:

Effect of Varying CLung

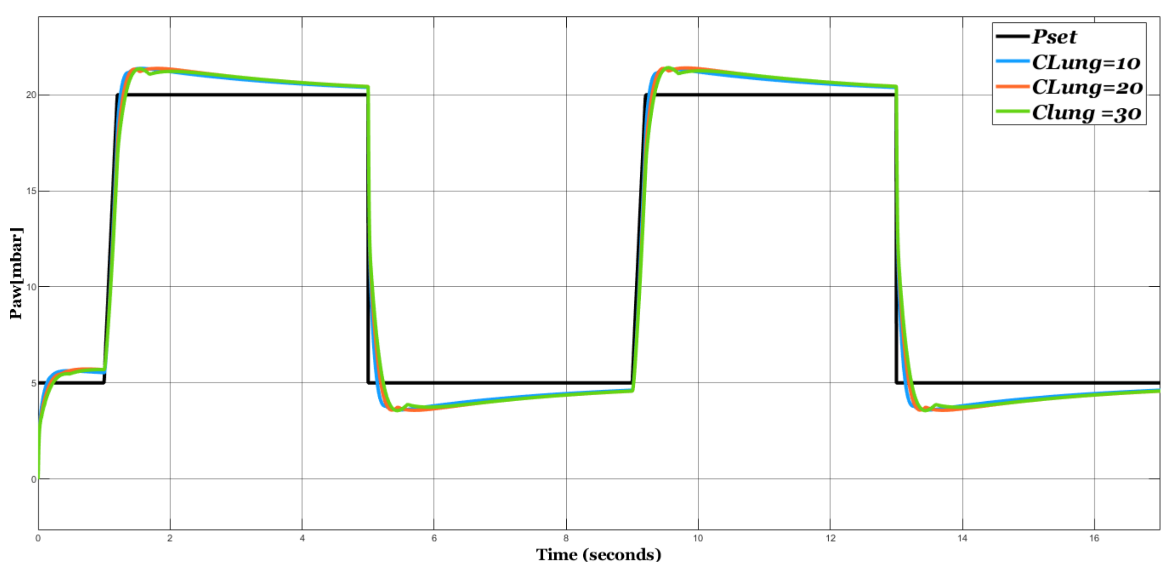


Fig : Paw for different CLung

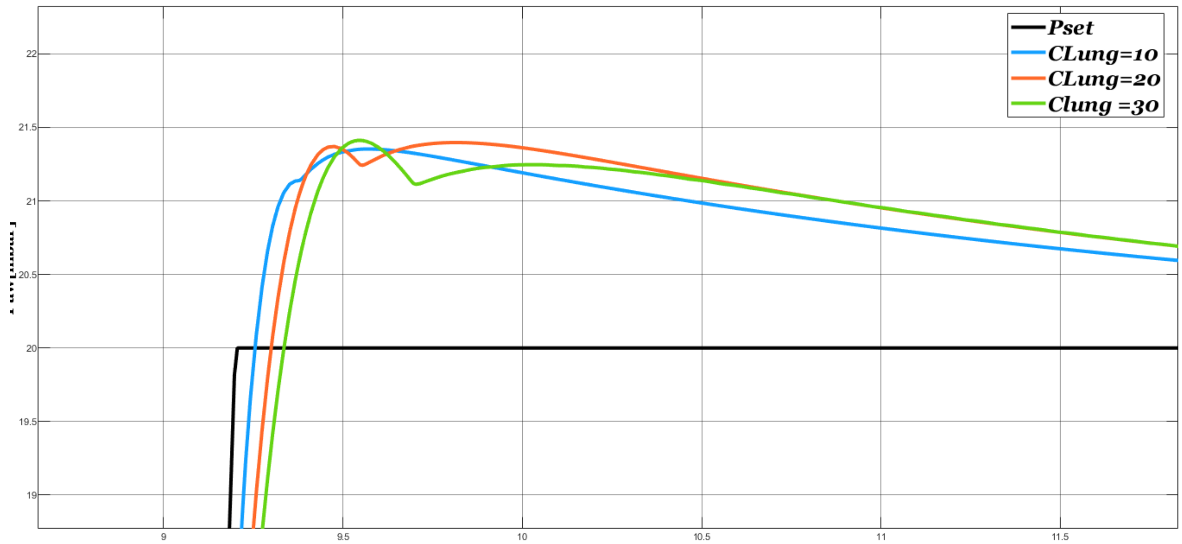


Fig: Observation of Paw(max) for different CLung

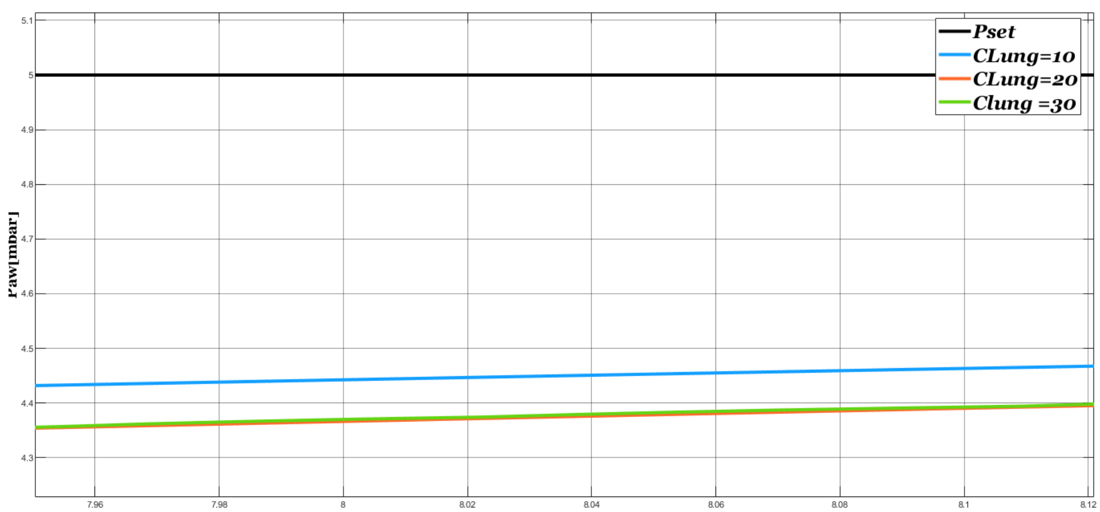


Fig: Observation of SSE for different CLung

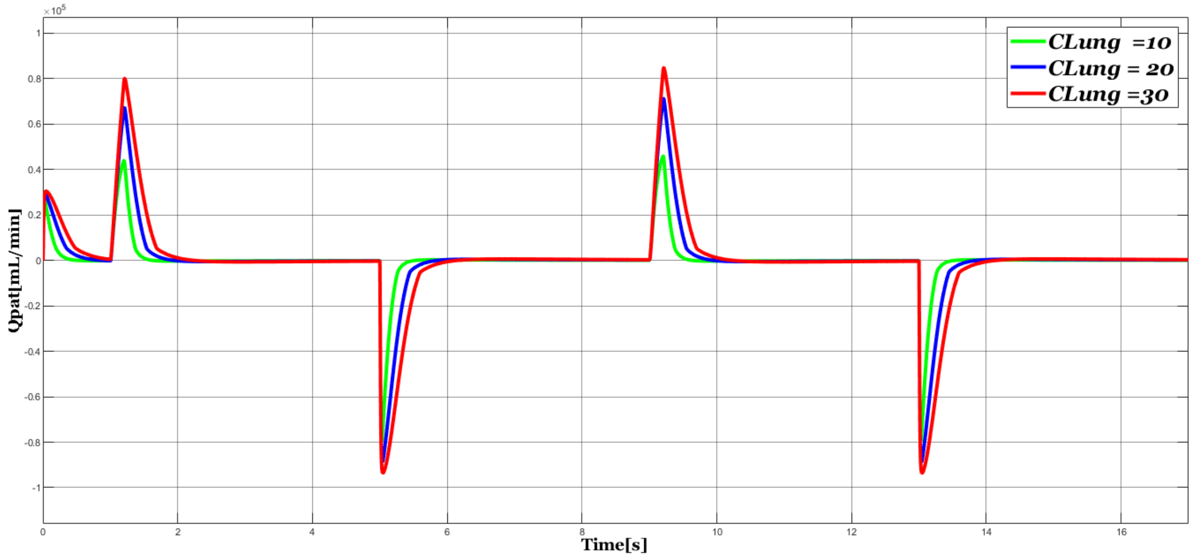


Fig : Qpat for different CLung

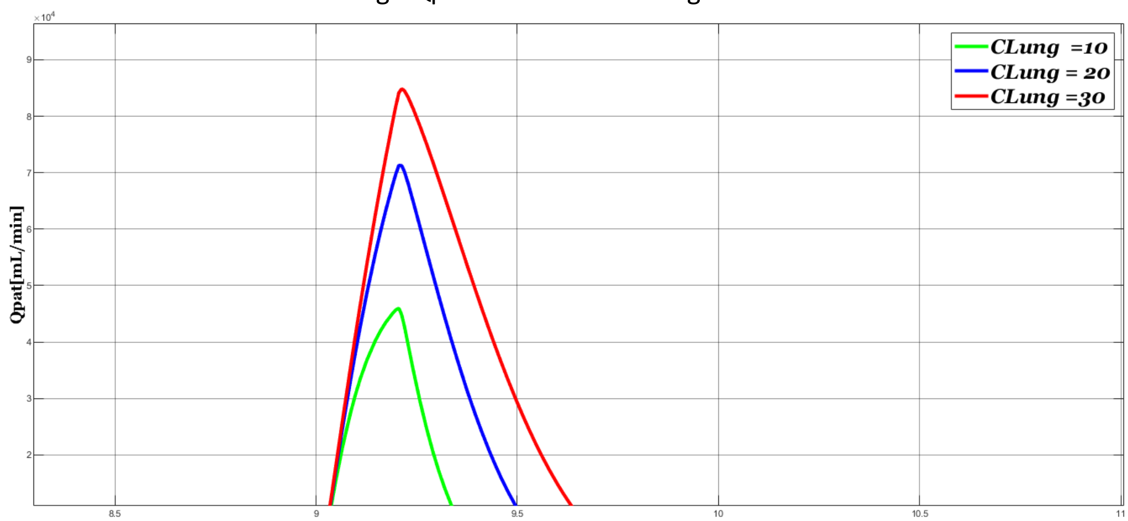


Fig: Observation of Qpat(max) for different CLung

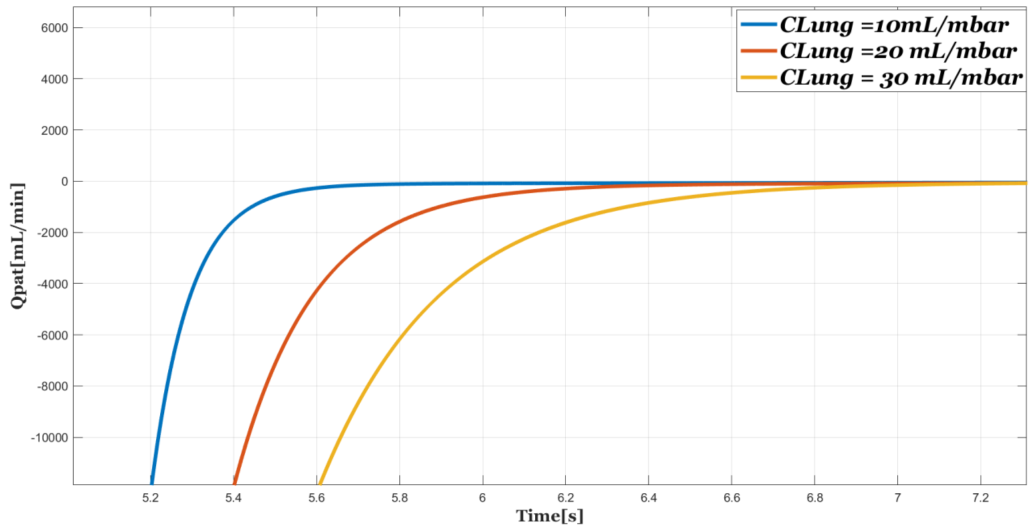


Fig: Observation of settling time for different CLung

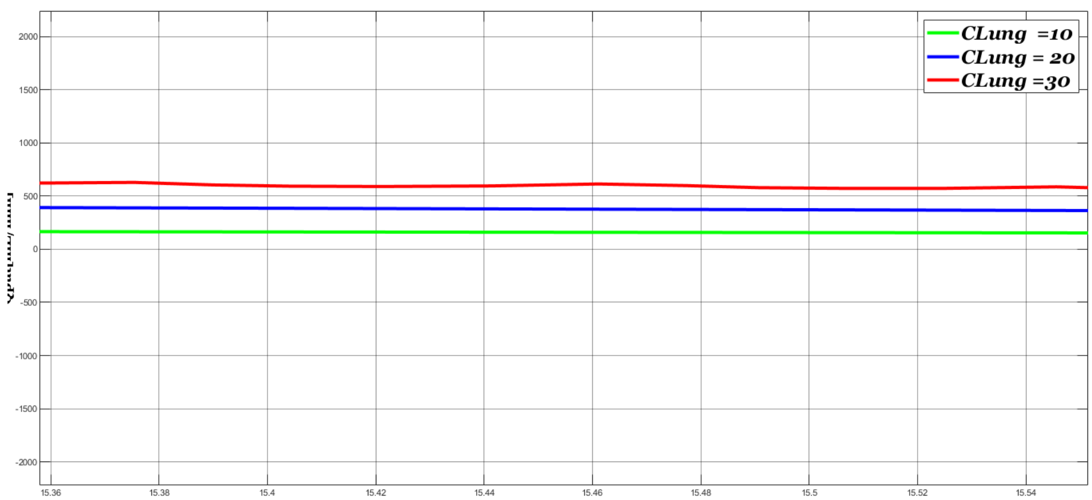


Fig: Observation of SSE for different CLung

Effect of Varying RLung

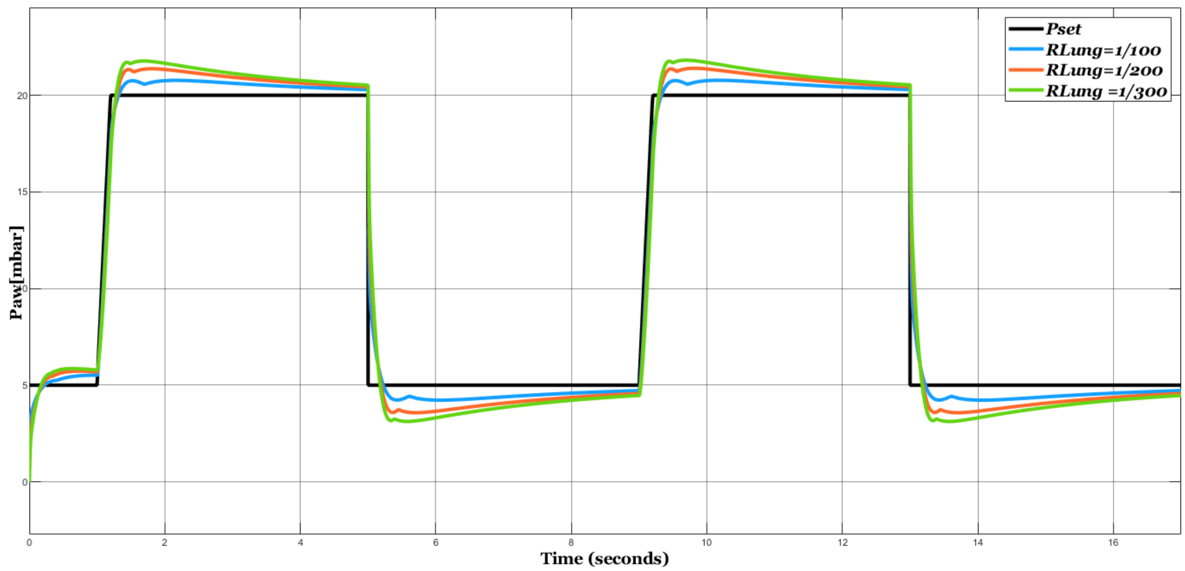


Fig : Paw for different RLung

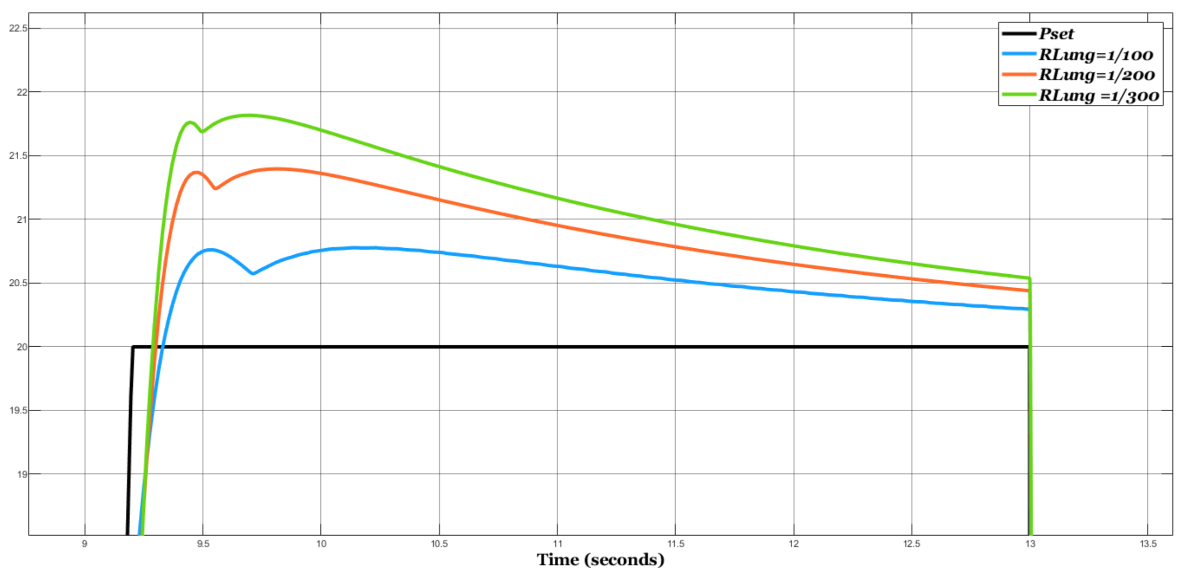


Fig: Observation of Peak Value   for different RLung

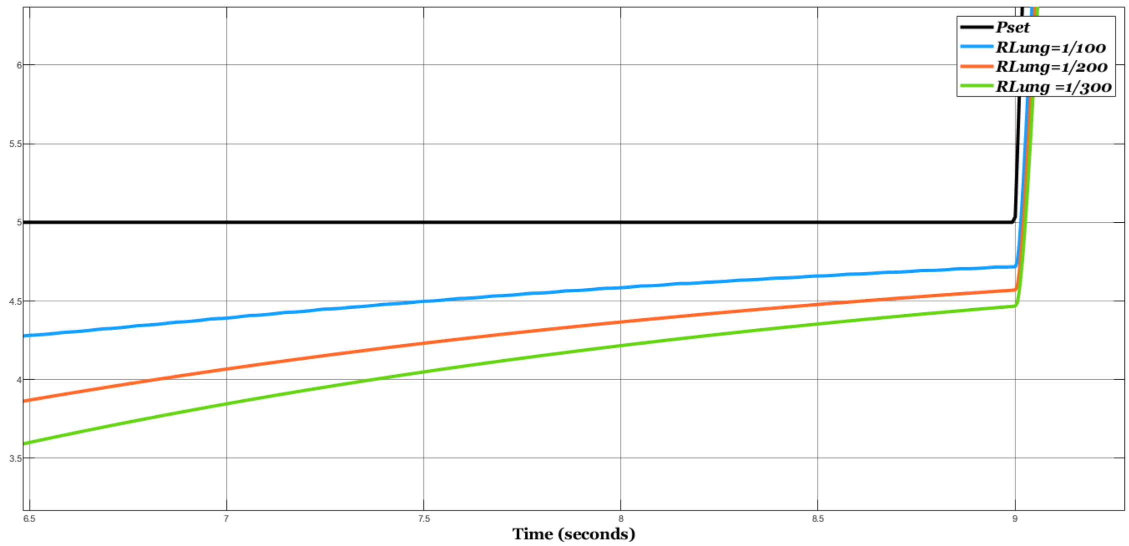


Fig: Observation of SSE for different RLung

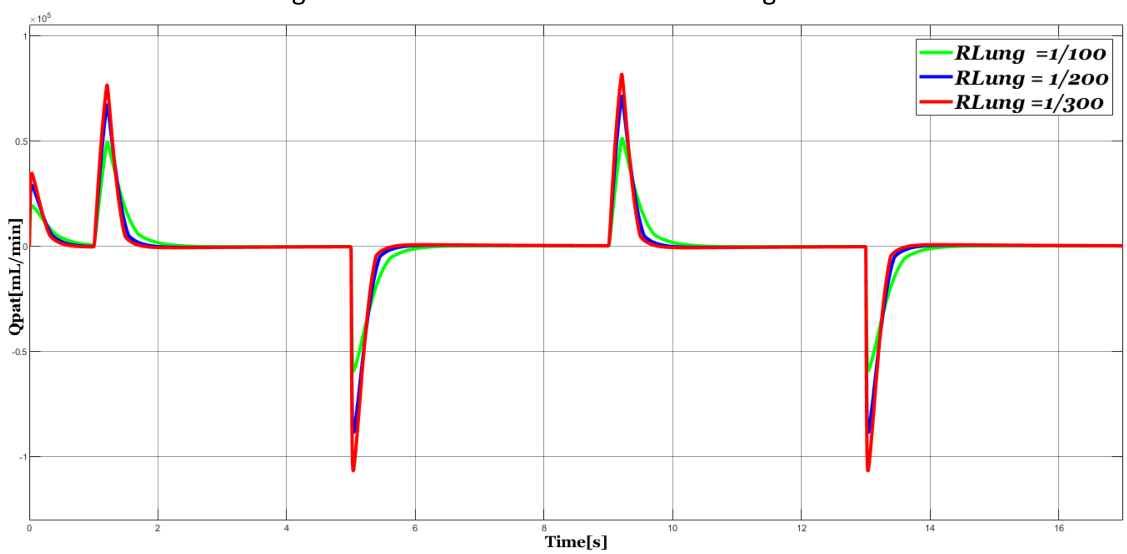


Fig: Effect of Varying RLung

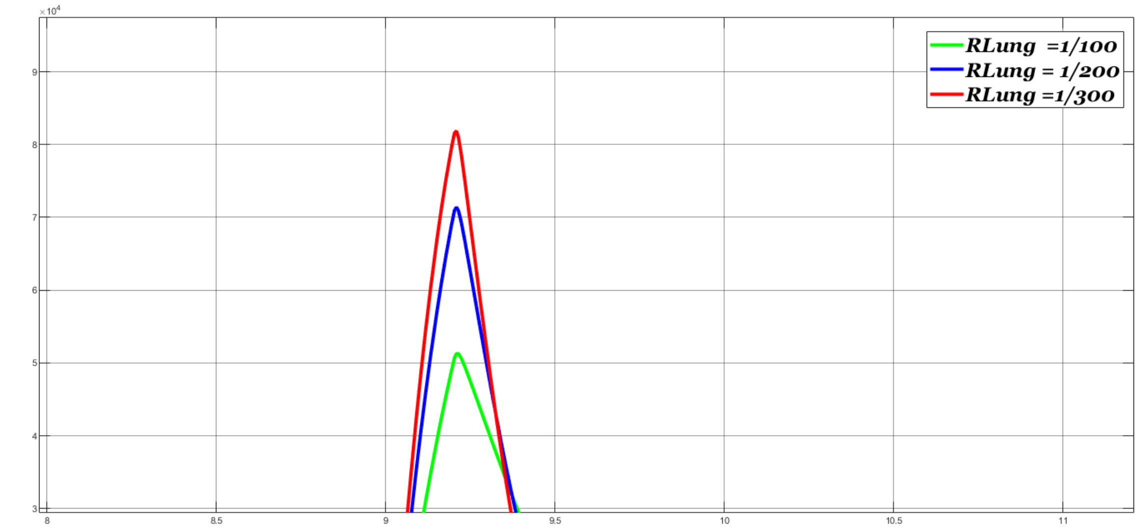


Fig: Observation of Qpat(max) for different RLung

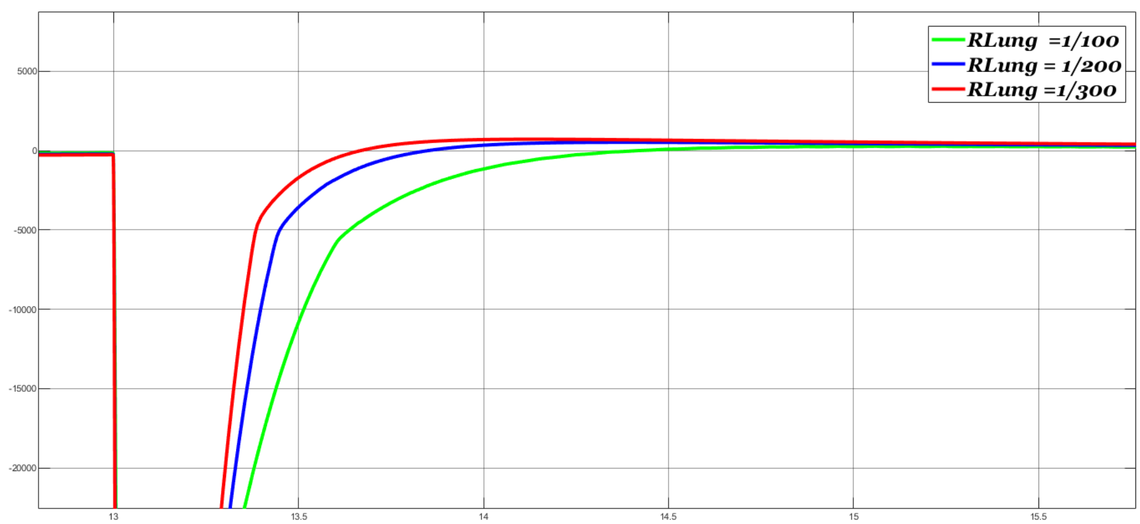


Fig: Observation of settling time for different RLung

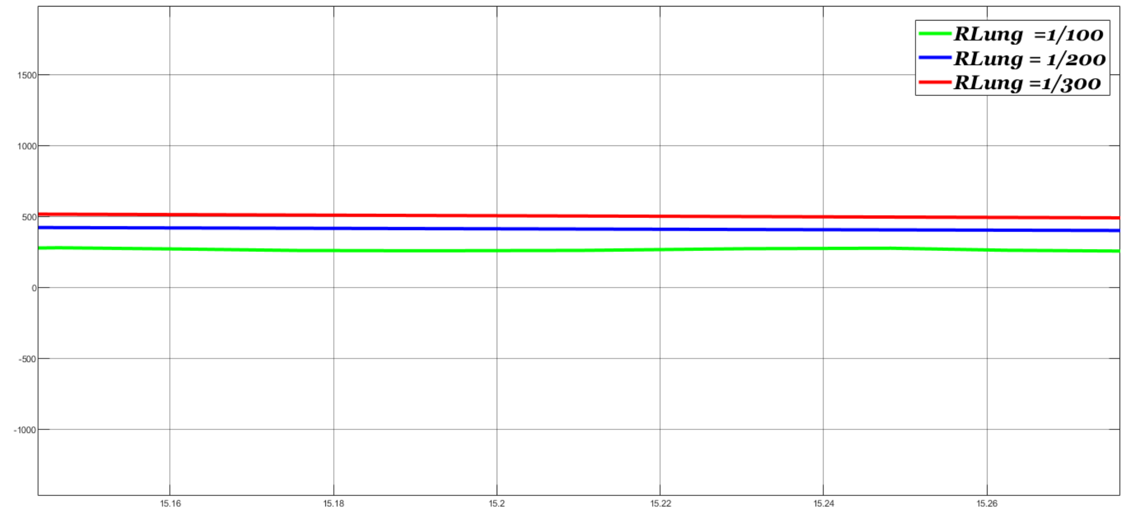


Fig: Observation of SSE for different RLung

Effect of Varying RLeak

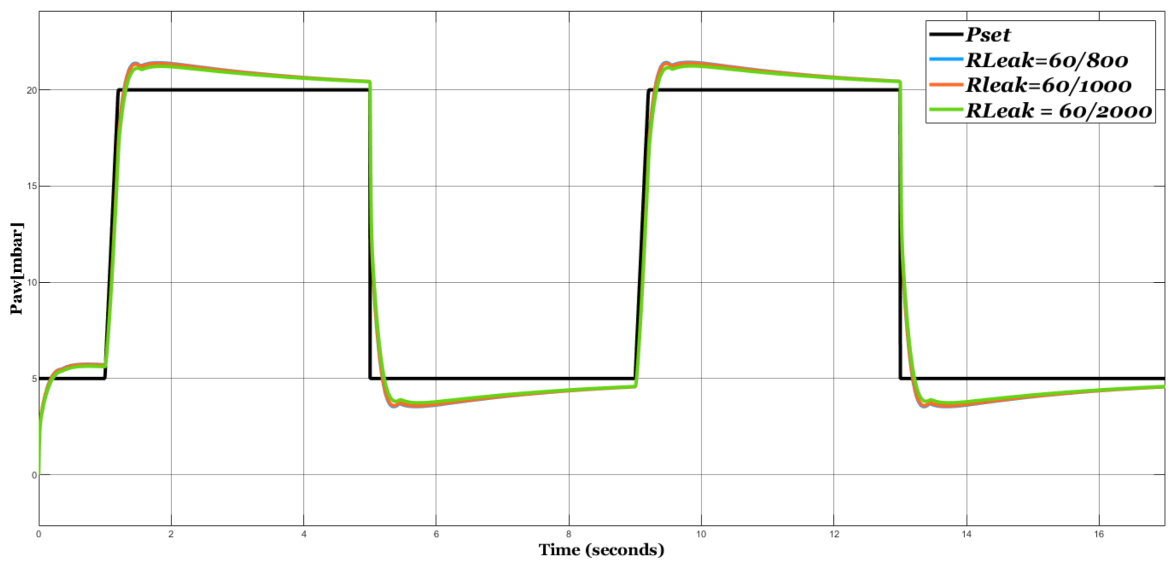


Fig: Paw for different RLeak

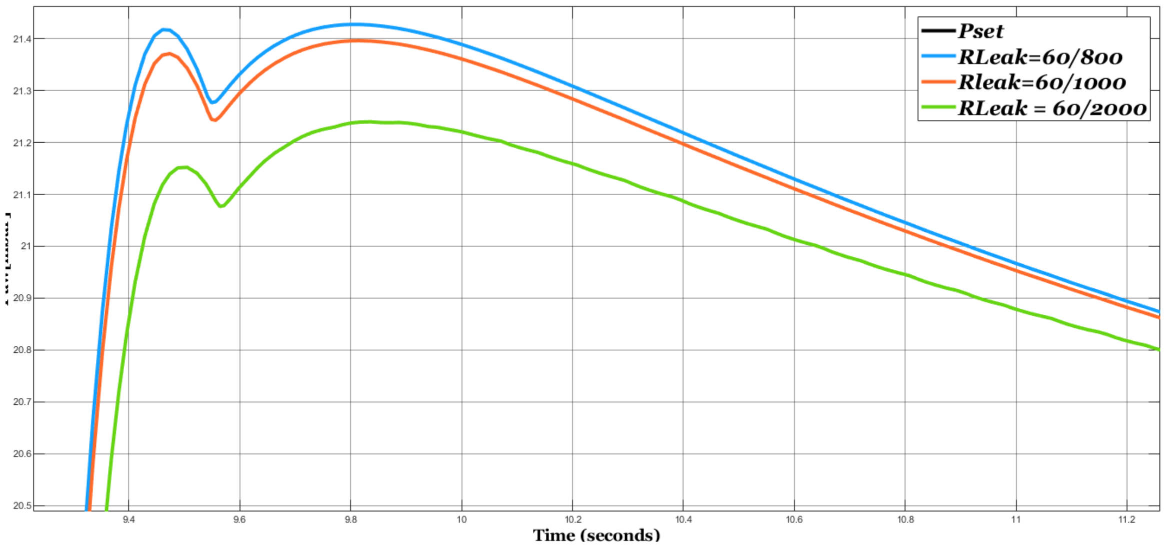


Fig: Observation of Paw(max) for different RLeak

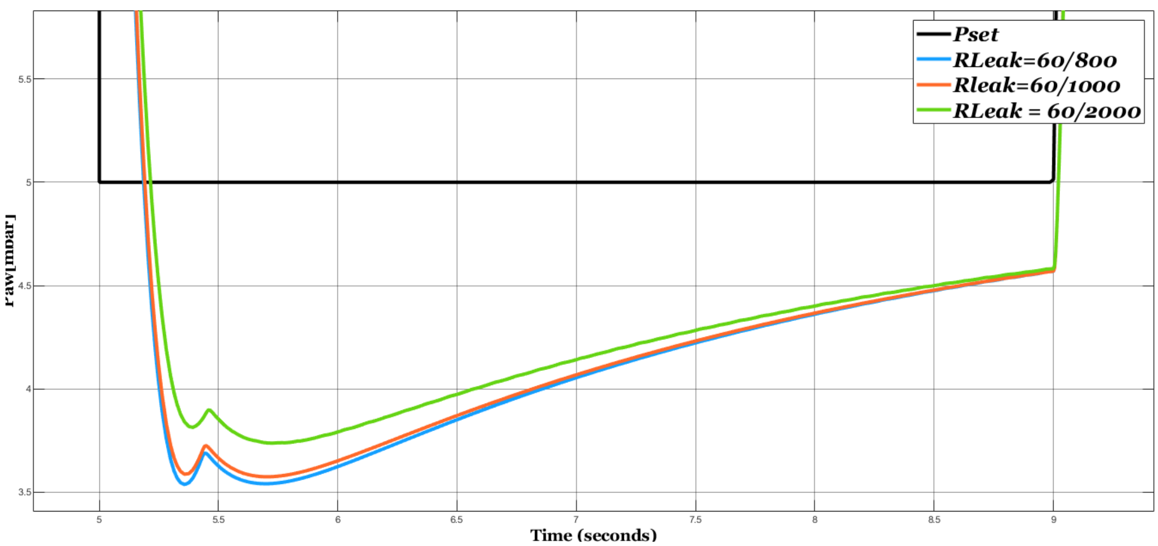


Fig: Observation of SSE for different Rleak

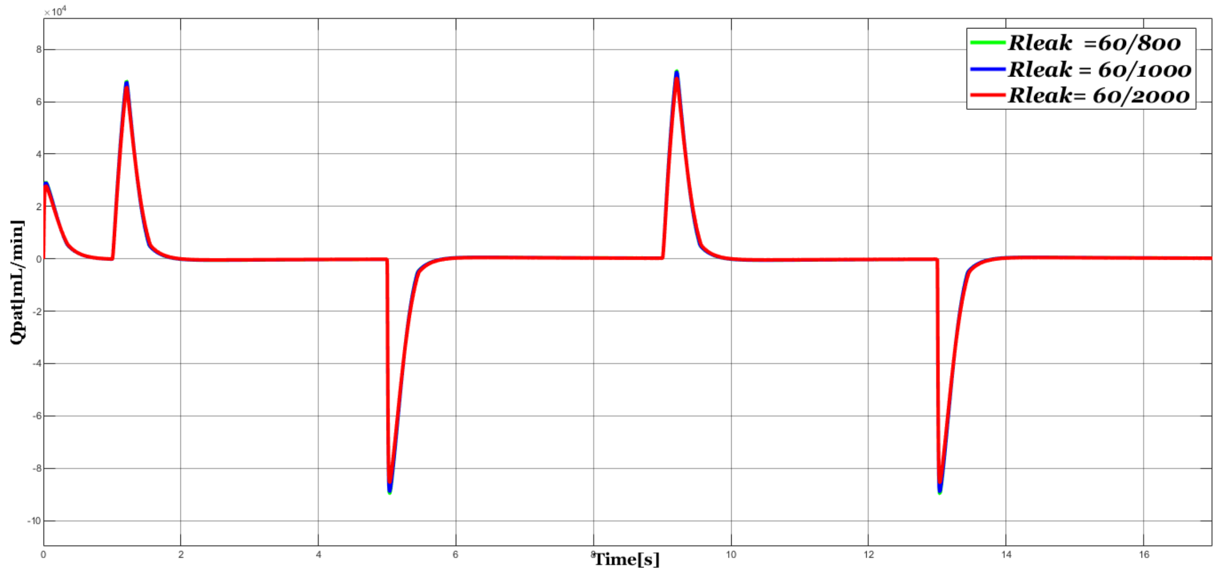


Fig: Qpat for various Rleak

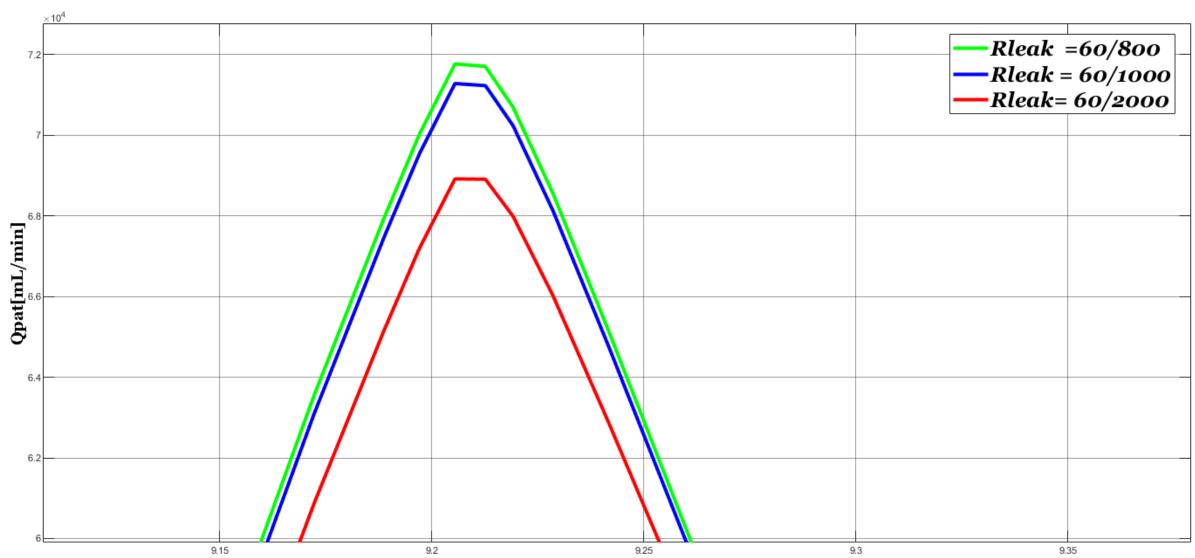


Fig: Observation of Paw(max) for different RLeak

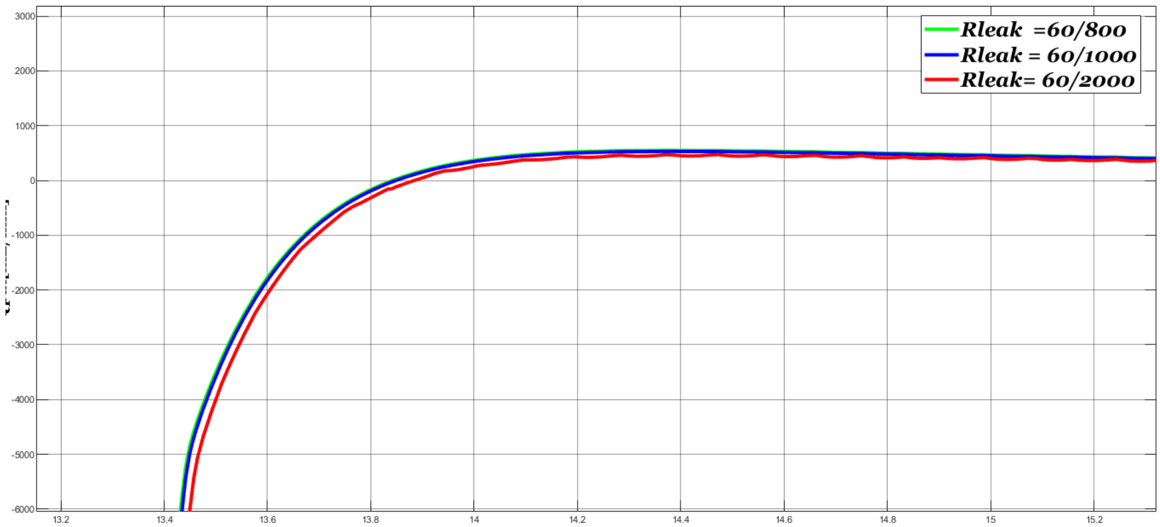


Fig: Observation of settling time for different Rleak

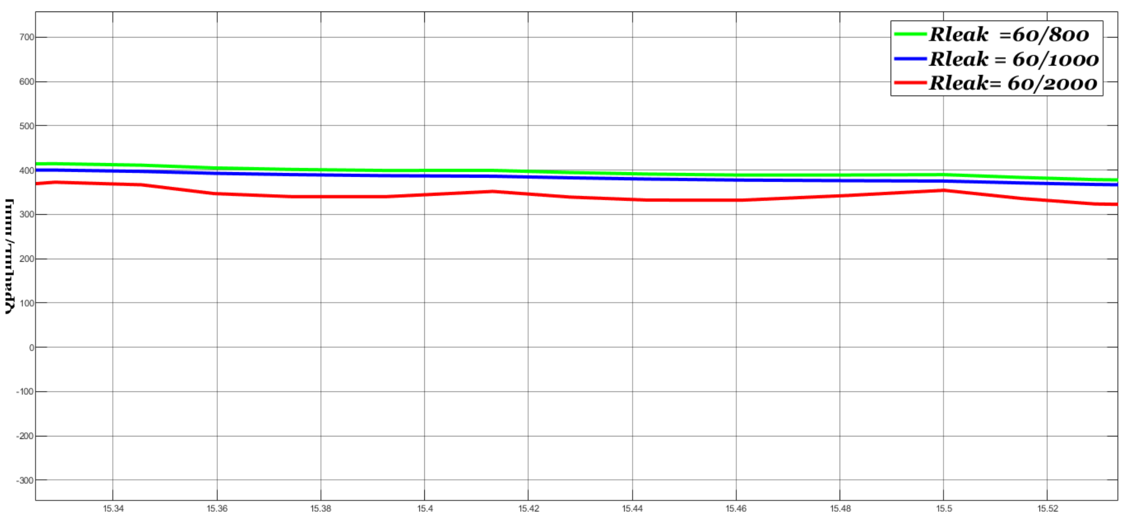


Fig: Observation of SSE for different Rleak

Effect of Varying RHose

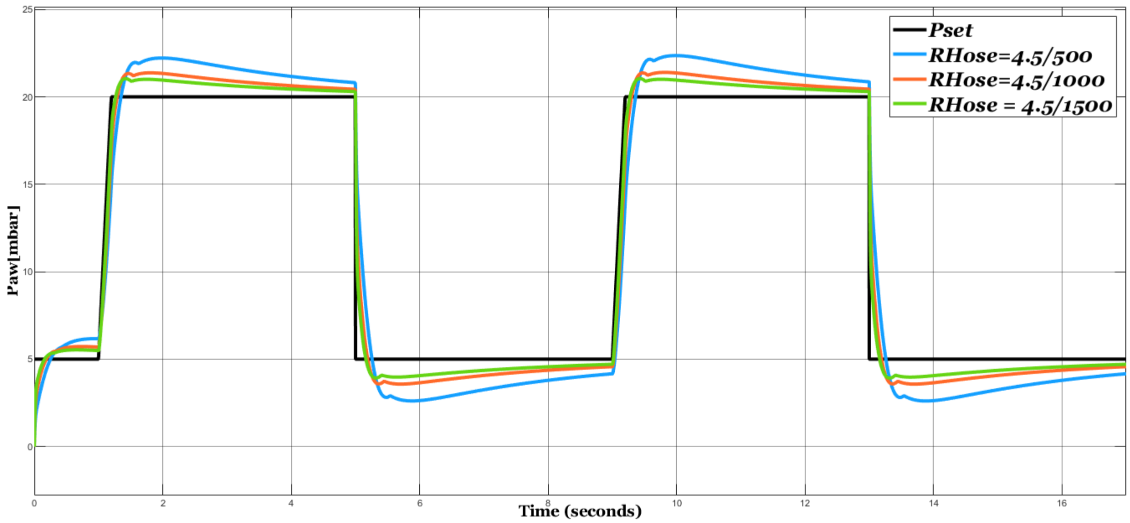


Fig: Paw for different RHose

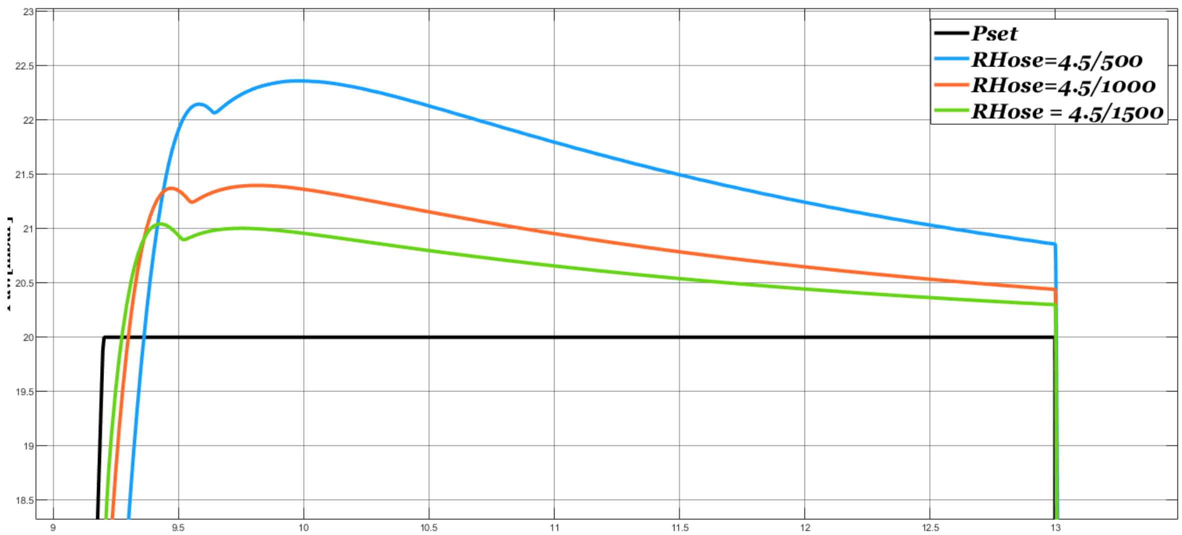


Fig: Observation of Paw(max) for different RHose

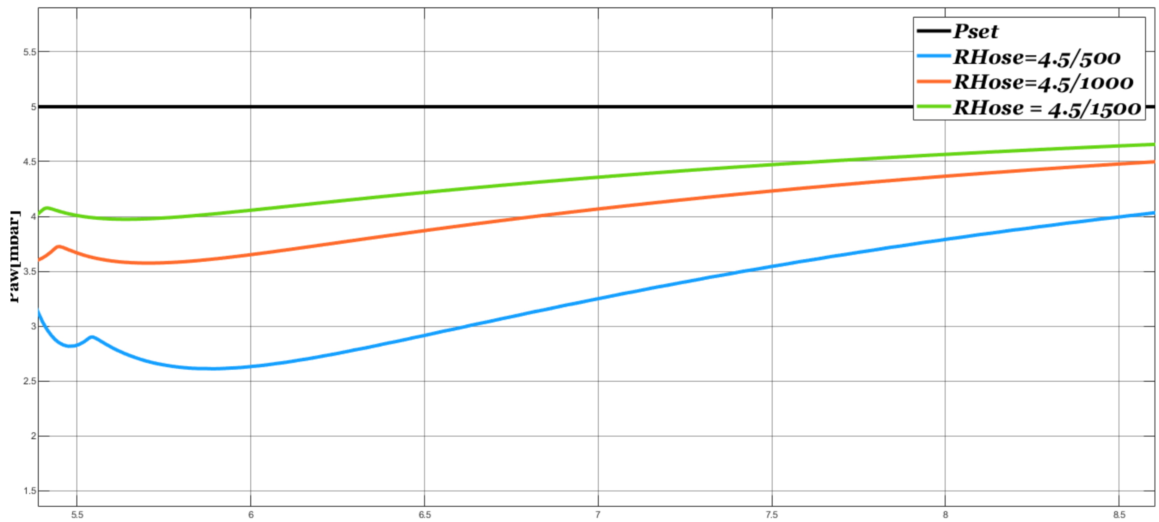


Fig: Observation of SSE for different RHose

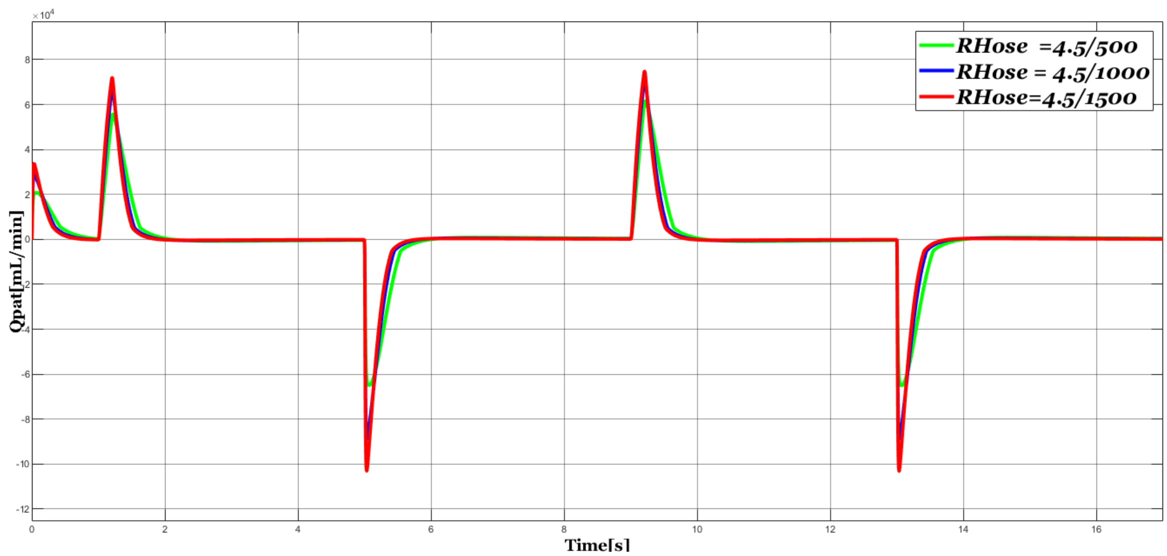


Fig: Qpat for different RHose

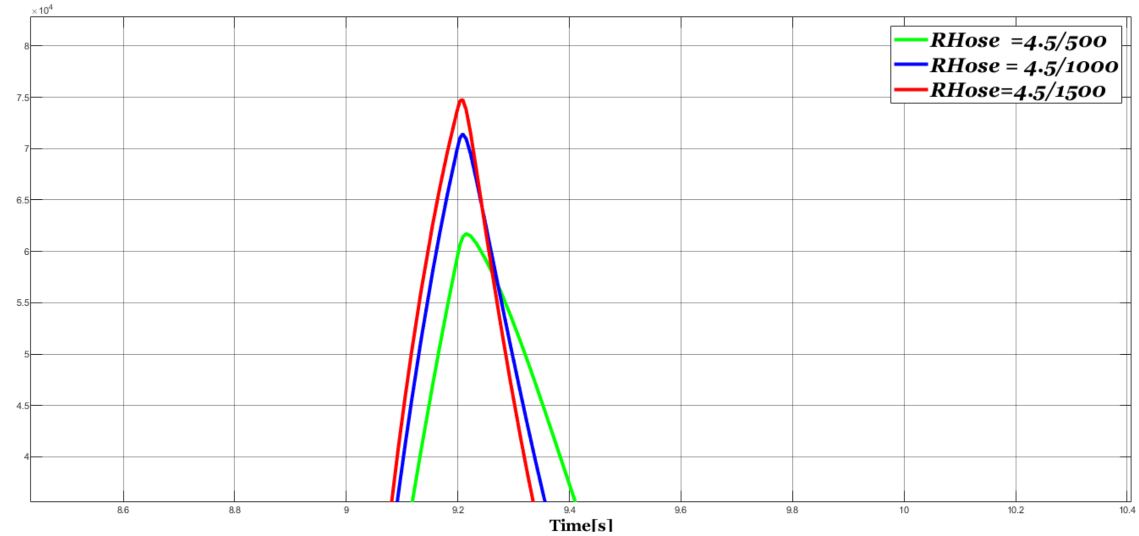


Fig: Observation of Qpat(max) for different RHose

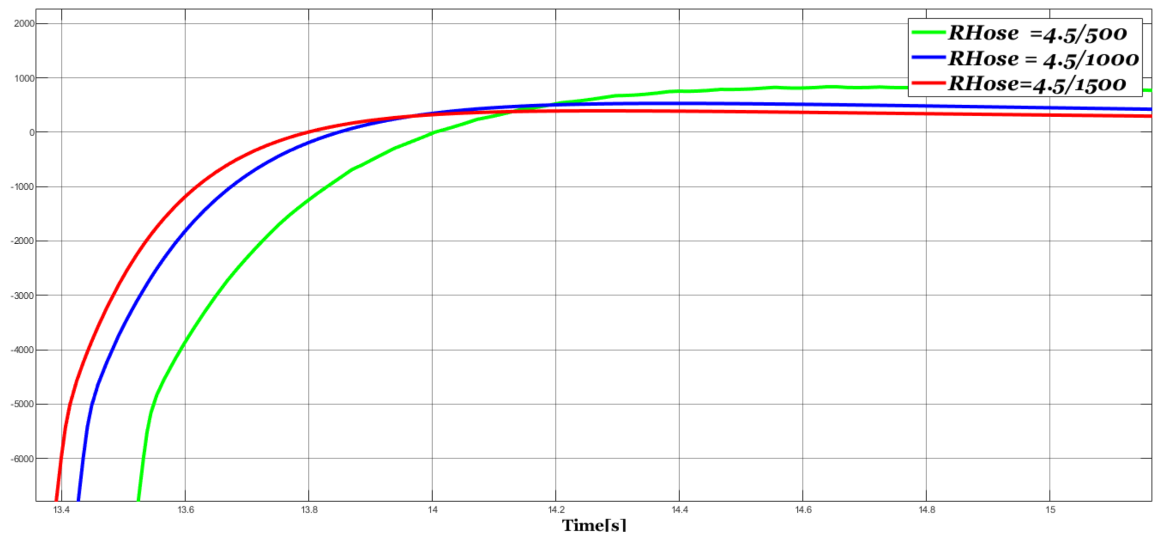


Fig: Observation of settling time for different RHose

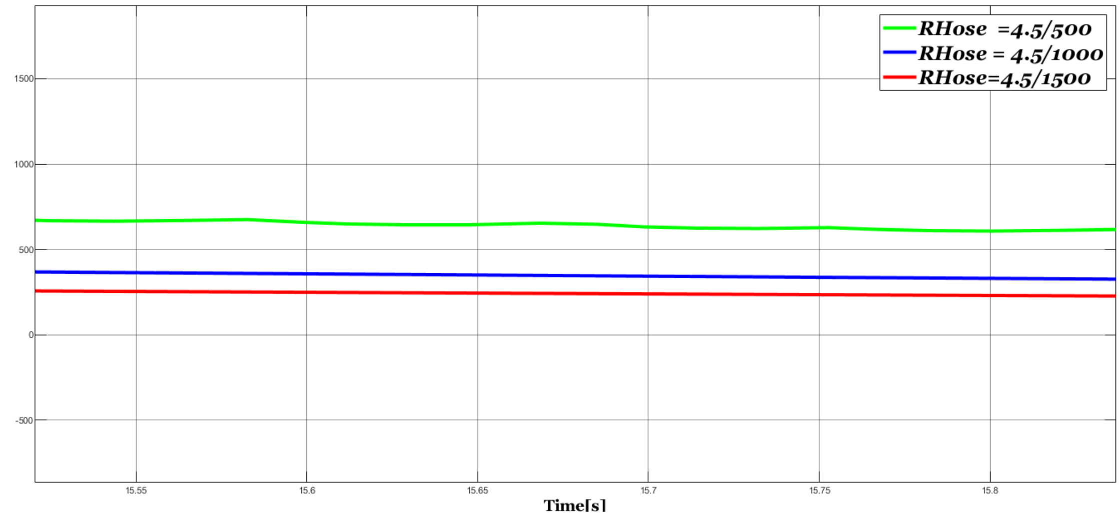


Fig: Observation of SSE for different RHose

Table Showing Effect on Paw for Varying Parameters (Non-Linear Controller)

Parameters	Paw(max)	Rise Time	Steady State Error
CLung	Proportional	Proportional	Proportional
GLung	Proportional	Same Value	Proportional
GLeak	Inverse	Same Value	Inverse
GHose	Inverse	Proportional	Inverse

Table Showing Effect on Qpat for Varying Parameters(Non-Linear Controller)

Parameters	Qpat(max)	Settling Time	Steady State Error	Overshoot
CLung	Proportional	Proportional	Inverse	Not observed
GLung	Proportional	Inverse	Proportional	Not observed
GLeak	Inverse	Same	Inverse	Not observed
GHose	Proportional	Inverse	Inverse	Not observed