CONTROL AND AUTOMATION ENGINEERING DEPARTMENT

PROGRAMMING TECHNIQUE IN CONTROL

2025-2026 FALL SEMESTER

FINAL REPORT

GROUP 30

GROUP MEMBERS:

Afra Korkmaz	**110210225**
Batu Yeken	**040230515**
Yiğit Çakır	**040230608**
Deniz Kamalak	**040210607**
Fatih Talay	**040210617**

Question 1 (Modelling of the Lungs):

a. The Simscape Model

The model is constructed as in the figure below. The output data sent to the workspace are named according to the data they represent.
It should be noted that *out.tfminDout* represents the airflow variable that is computed via the first order system transfer function. Similar to this, *out.tfminout* is the tidal volume $V(t)$.

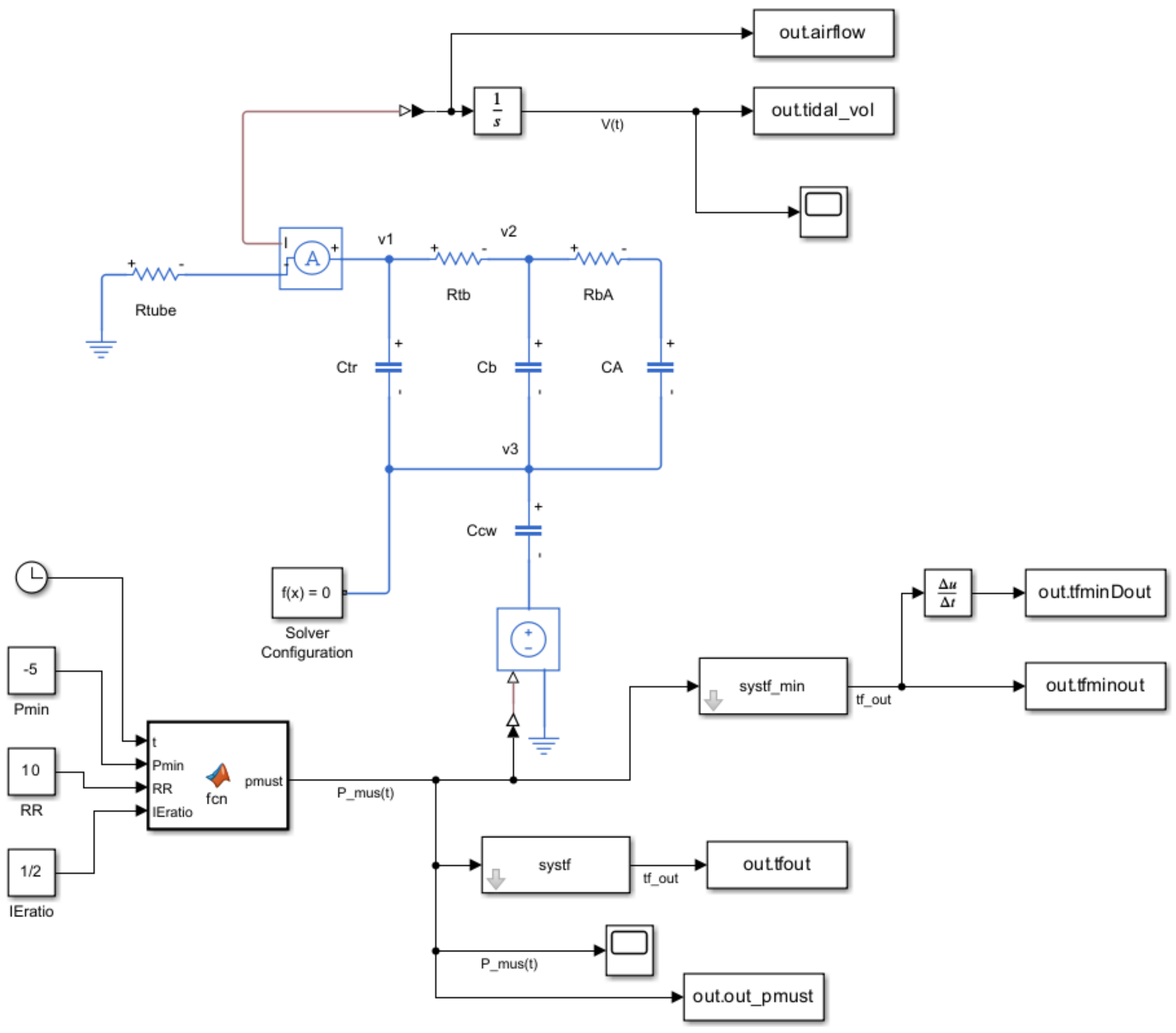


Figure 1: The Simscape model of the lungs.

b. Matlab Function Representing Spontaneous Breathing

Since this is a real time simulation, the input arguments of the function should contain the time variable t that is fed by a clock.

The necessary relations defining the function parameters are defined first.

Next, the variable t given in the mathematical model is converted to $mod(t, T)$ because on each inspiration-respiration cycle this time variable should be in the interval $[0, T]$

Lastly, the mathematical expressions are implemented separately inside if-else blocks.

```
function pmust = fcn(t, Pmin, RR, IEratio)

y = 0;

T = 60/RR;
TI = T*IEratio/(1+IEratio);
TE = T - TI;
tao = TE/10;

tmod = mod(t, T);

if tmod <= TI
    y = (Pmin)*(T*tmod - tmod^2)/(TI*TE);
elseif tmod > TI
    y = (Pmin)*(exp( -(tmod-TI)/tao ) - exp(-TE/tao) )/(1 - exp(-TE/tao));
end

pmust = y;
```

The signal produced by this MATLAB function is shown on the figure below:

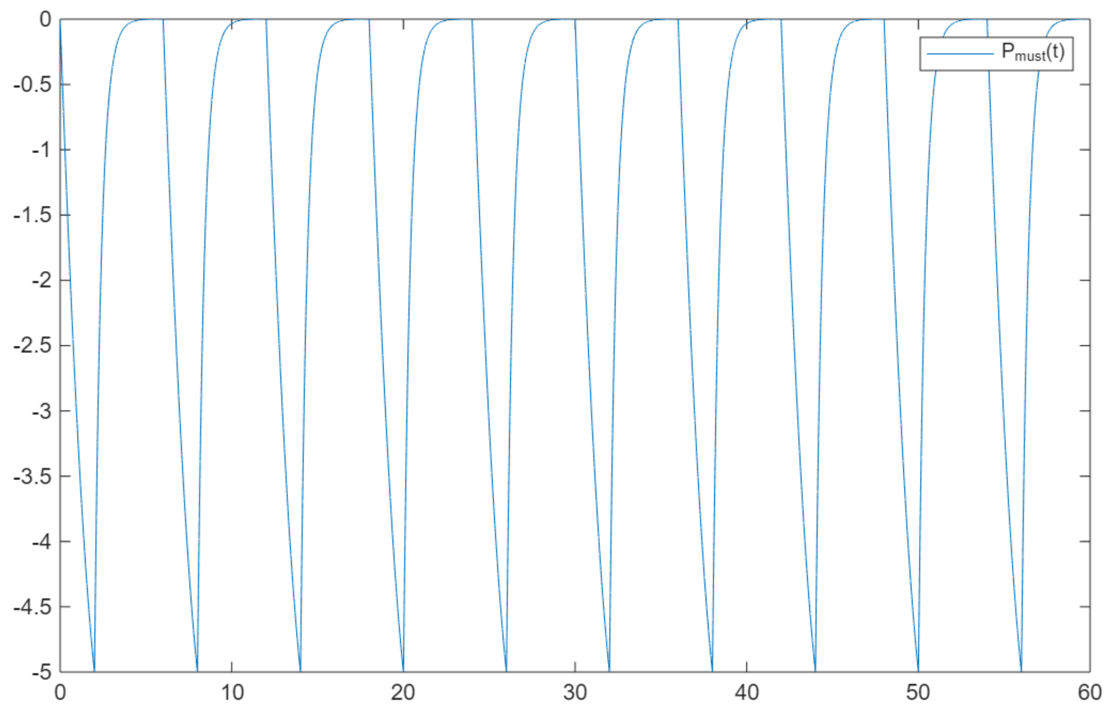


Figure 2: The $P_{must}(t)$ output.

c. Simulated $V'(t)$ and $V(t)$ results

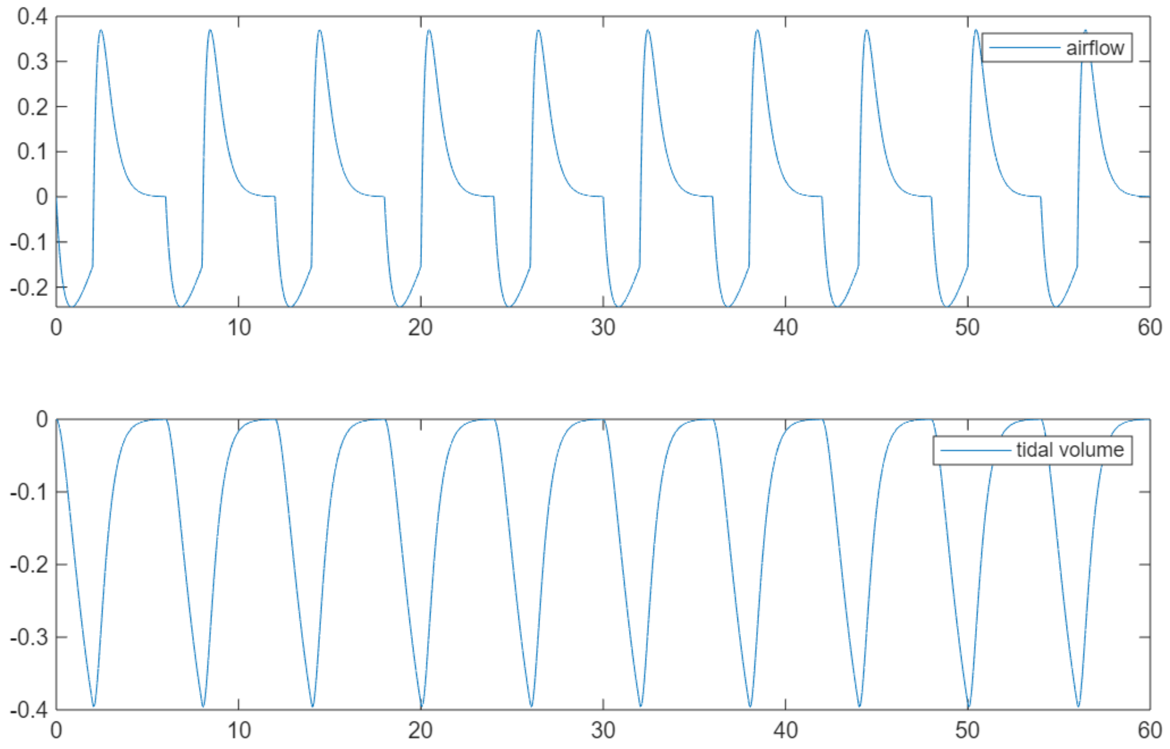


Figure 3: The airflow and tidal volume simulation outputs.

d. Obtaining the $\frac{V(s)}{P_{mus}(s)}$ transfer function

To obtain the transfer function, the model parameters are first defined below:

```
% Model parameters defined below

% Resistance values
Rtube = 4.0;
Rtb = 0.212;
RbA = 0.082;

% Capacitor values
Ctr = 0.0016;
Cb = 0.013;
CA = 0.15;
Ccw = 0.2;
```

In order to form KCL equations for the circuit *v1*, *v2* and *v3* nodes are defined on *Figure 1*

.Next, respectively, the relations between *v1* and $V'(t)$, $V'(s) - V(s)$ relation and the symbolic transfer function variable are defined.

```
% symbols to use in KCL equations
syms v1 v2 v3 s Pmus vdot vout tf_sym

v1_eq = v1/Rtube == vdot;
out_eq = vdot/s == vout;
tf_eq = vout/Pmus == tf_sym;
```

Then, KCL equations are constructed and solved for *v1, v2 and v3* nodes.

```
KCL1 = vdot - s*Ctr*(v3-v1) - (v2-v1)/Rtb == 0;
KCL2 = (v2-v1)/Rtb - s*Cb*(v3-v2) - (v3-v2)/(RbA + 1/(s*CA)) == 0;
KCL3 = s*Ctr*(v3-v1) + s*Cb*(v3-v2) + (v3-v2)/(RbA + 1/(s*CA)) - s*Ccw*(Pmus
    -v3) == 0;

KCL_all = [KCL1 KCL2 KCL3];

sol_kcl = solve([KCL_all out_eq v1_eq tf_eq], [tf_sym v1 v2 v3 Pmus vdot]);
```

The following code converts the symbolic transfer function (tf_sym) to a *tf* object that we can use. Also the denominator's highest order coefficient is normalized to 1.

```
[tfnum, tfden] = numden(sol_kcl.tf_sym);

tfnum = sym2poly(tfnum);
tfden = sym2poly(tfden);

% dividing the numerator and denominator to the highest order
% coefficient of the denominator.
tfnum = tfnum/tfden(1);
tfden = tfden/tfden(1);

systf = tf(tfnum, tfden);
```

As a result of these procedures, the resulting $\frac{V(s)}{P_{mus}(s)}$ transfer function is:

```
systf =

       0.25 s^2 + 1083 s + 7.587e05
  ---------------------------------------
  s^3 + 4488 s^2 + 3.256e06 s + 8.403e06
```

The figure below shows the comparison between the outputs of the analytically computed transfer function and the simulation results. The identical figures verify the obtained model.

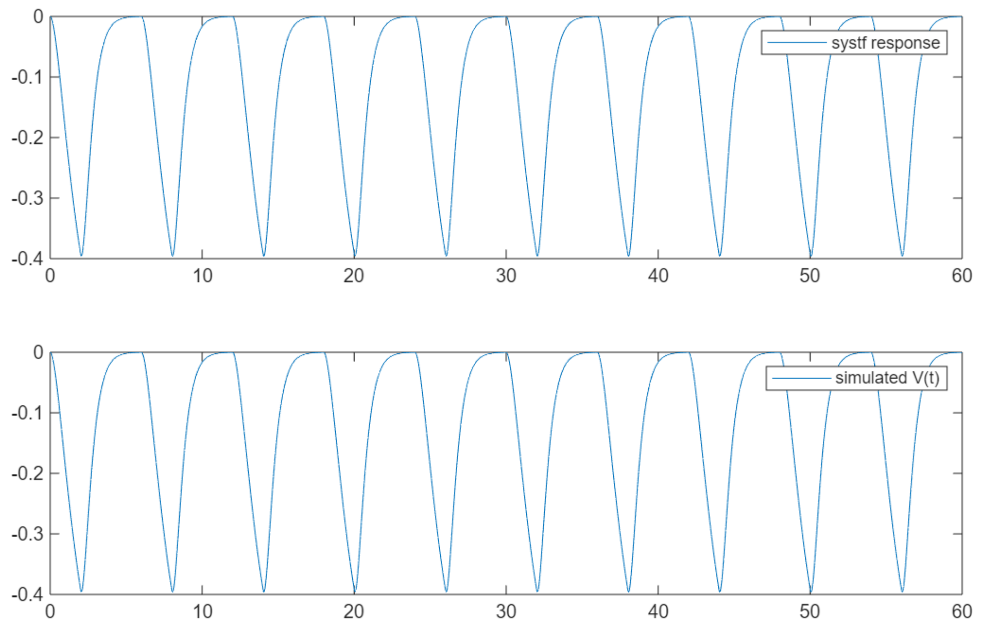


Figure 4: systf and simulated $V(t)$ results.

e. Calculation of the reduced order system model

In order to observe the zeros, poles and the gain of the system, the *zp2tf* function is constructed to extract zero, pole and k values.

```
[z, p, k] = tf2zp(tfnum, tfden)
```

The output is:

```
z =
   1.0e+03 *

   -3.4509
   -0.8794

p =
   1.0e+03 *

   -3.5786
   -0.9066
   -0.0026

k =
    0.2500
```

The results show that the zero-pole pairs are:

$z1 = -3450 \quad p1 = -3578$ and $z2 = -879 \quad p2 = -906$ and $p3 = -2.6$

It is clear that there are 2 pole-zero pairs that are approximately 350 and 1350 times further from the imaginary axis than the closest pole at $s = -2.6$.

The next piece of code reduces the third degree transfer function to a degree of one. Since the cancelling pole-zero pairs are numerically not close enough(but their angle and gain contribution is still minimal), the tolerance parameter should be high enough to achieve cancelation.

```
tol = sqrt(eps)*10000000;
systf_min = minreal(systf, tol)
```

The resulting first order transfer function is:

```
systf_min =

    0.25
  --------
  s + 2.59
```

The required R and C values can easily be obtained by:

$$\frac{1}{R} = 0.25 \qquad \frac{1}{RC} = 2.59 \tag{1}$$

R and C are found as:

$$R = 4 \quad C = 0.0965 \tag{2}$$

f. Comparison of the Reduced Order Model and the Simulation

The figure below shows the outputs of the reduced order system and the simulation output:

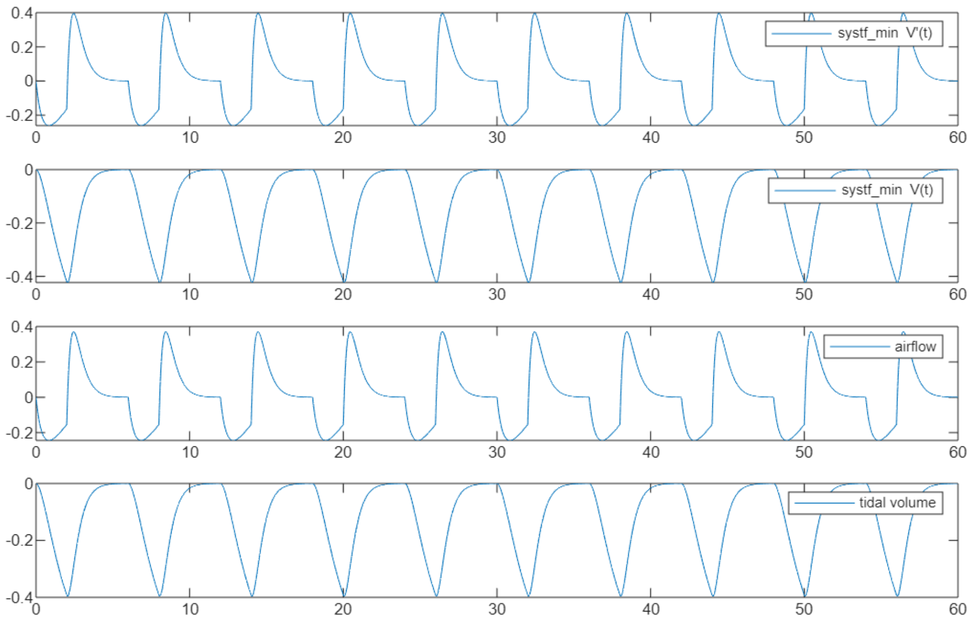


Figure 5: Comparison between the first order TF and the simulation.

A slight difference between the peak values of the responses are observed, especially on the airflow plots (1 and 3). To examine this discrepancy, the DC gain values of the reduced and non-reduced transfer functions should be calculated.

$$\text{systf:} \qquad G(0) = \frac{0.25(0)^2 + 1083(0) + 7.587\times10^5}{(0)^3 + 4488(0)^2 + 3.256\times10^6(0) + 8.403\times10^6} = \frac{758700}{8403000} \approx 0.0903$$

$$\text{systf\_min:} \qquad G_{min}(0) = \frac{0.25}{0+2.59} = \frac{0.25}{2.59} \approx 0.0965$$

The result show us that the zero-pole cancellation results in a change in DC gain. This can be considered as the reason behind the acceptable difference on these plots.

Question 2 (Mechanical Ventilation Device Design)

a. Open-Loop Transfer Function and Stability Analysis

The respiratory system is represented by a single compartment model. The open-loop transfer function, including a first-order smoothing filter, is given by:

$$G_{open-loop}(s) = \frac{V(s)}{P_{vent}(s)} = \frac{1/R}{s + \frac{1}{RC}} \cdot \frac{1}{\tau_f s + 1}$$

For a PI controller $F(s) = k_I/s$, the closed-loop characteristic polynomial is derived as:

$$(C \cdot R \cdot \tau_f)s^3 + (C \cdot R + \tau_f)s^2 + s + C \cdot k_I = 0$$

To determine the stabilizing range for k_I, the Routh-Hurwitz table is constructed:

$$\begin{array}{c|cc} s^3 & CR\tau_f & 1 \\ s^2 & CR + \tau_f & Ck_I \\ s^1 & \frac{(CR+\tau_f) - C^2 R\tau_f k_I}{CR+\tau_f} & 0 \\ s^0 & Ck_I & \end{array}$$

For the system to be stable, the first column must contain only positive values, resulting in:

- $k_I > 0$
- $k_I < \frac{CR+\tau_f}{C^2 R\tau_f}$

The MATLAB code used to verify the characteristic polynomial is:

```
syms R C Tao_f K K_i w s;
assume(R > 0 & C > 0 & Tao_f > 0);

% System Transfer Function
G_f = 1 / (Tao_f*s + 1);
G = (1/R) / (s + 1/(R*C));

% Getting Characteristic polynomial
F_pi = K_i/s;
L = F_pi * G_f * G;
T_pi = simplify(L / (1 + L));
[~, Pcs] = numden(T_pi);
coefficent = coeffs(Pcs, s, "All");
disp("Characteristic polynomial for Close Loop of PI Controller");
```

b. P-Type Controller Design for Minimum Settling Time

Using a P-type controller $F(s) = k$, the characteristic equation becomes:

$$(C \cdot R \cdot \tau_f)s^2 + (\tau_f + C \cdot R)s + (C \cdot k + 1) = 0$$

To achieve the minimum settling time without overshoot, the system must be critically damped. Solving for k using the discriminant $\Delta = 0$:

$$k = \frac{(\tau_f + C \cdot R)^2 - 4 \cdot C \cdot R \cdot \tau_f}{4 \cdot C^2 \cdot R \cdot \tau_f}$$

The symbolic solution was obtained with the following code:

```
F_p = K;
L_2 = F_p * G_f * G;
T_p = simplify(L_2 / (1 + L_2));
[~, Pcs_2] = numden(T_p);
coefficent_2 = coeffs(Pcs_2, s, "All");
Delta = coefficent_2(2)^2 - 4 * coefficent_2(1) * coefficent_2(3);
sol = solve(Delta==0, K);
disp("Solutions for K when Delta = 0:");
disp(sol);
```

As lung compliance C increases, the real part of the poles σ decreases, leading to an increase in settling time $T_s \approx 4/\sigma$.

c. PCV Mode Implementation

The PCV mode generates a periodic square wave pressure signal. The Simulink model structure is shown in Figure 1.

```
function P_vent = fcn(t, VRR, VIE, Pc)
% Inspiration and expiration timing logic
T = 60/ VRR ;
TI = T * VIE /(1+ VIE ) ;
P_sq = Pc * (mod(t, T) <= TI);
P_vent = P_sq;
```

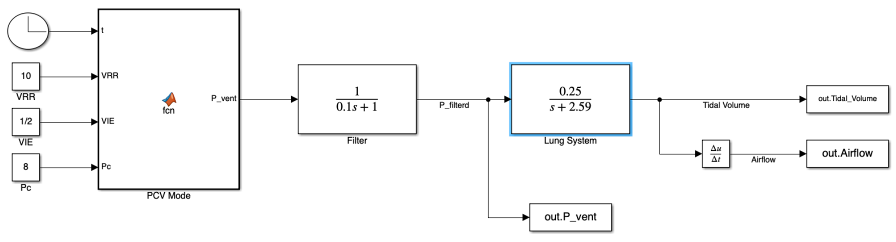


Figure 1: Simulink model structure for the PCV Mode simulation.

d. Simulation Results for Different Control Pressures

The system was simulated for $P_c = 8$ cmH$_2$O and $P_c = 4$ cmH$_2$O. The waveforms demonstrate that tidal volume and airflow scales almost linearly with applied pressure.

```
% Script to plot PCV Mode simulation results
figure('Name', 'PCV Mode Waveforms', 'Color', 'w');

% Plot Filtered Pressure
subplot(3,1,1);
plot(out.P_vent.Time, out.P_vent.Data);
title('Filtered Airway Pressure');
ylabel('Pressure [cmH2O]');
grid on;

% Plot Tidal Volume
subplot(3,1,2);
plot(out.Tidal_Volume.Time, out.Tidal_Volume.Data);
title('Tidal Volume (V)');
ylabel('Volume [L]');
grid on;

% Plot Airflow
subplot(3,1,3);
plot(out.Airflow.Time, out.Airflow.Data);
title('Airflow (V'')');
ylabel('Flow [L/s]');
xlabel('Time [s]');
grid on;
```

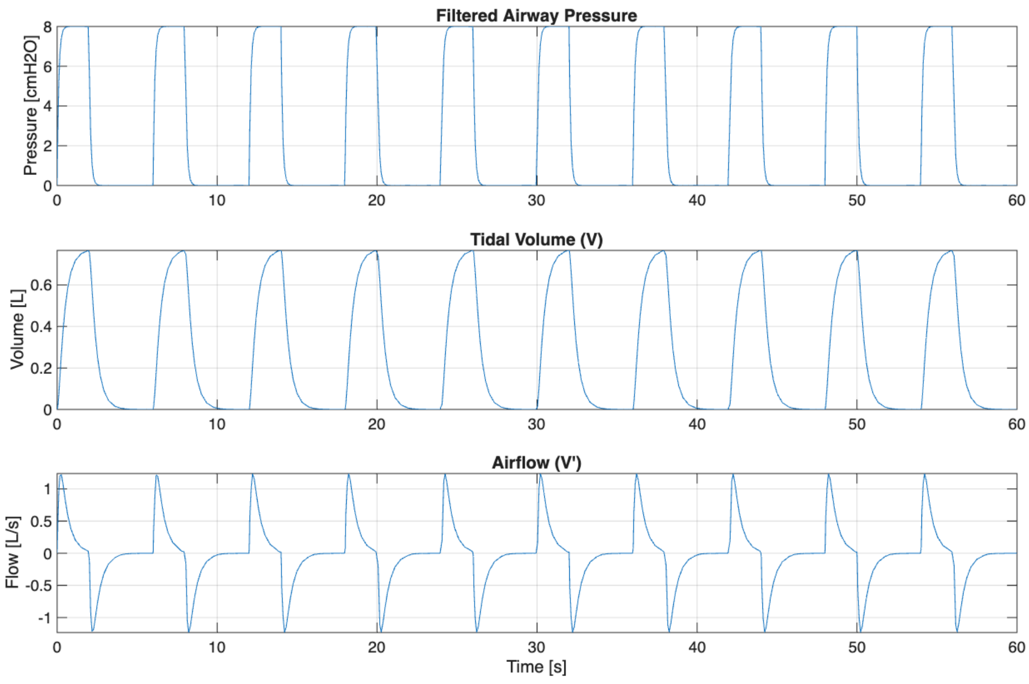


Figure 2: Simulation waveforms for $P_c = 8$ cmH$_2$O.

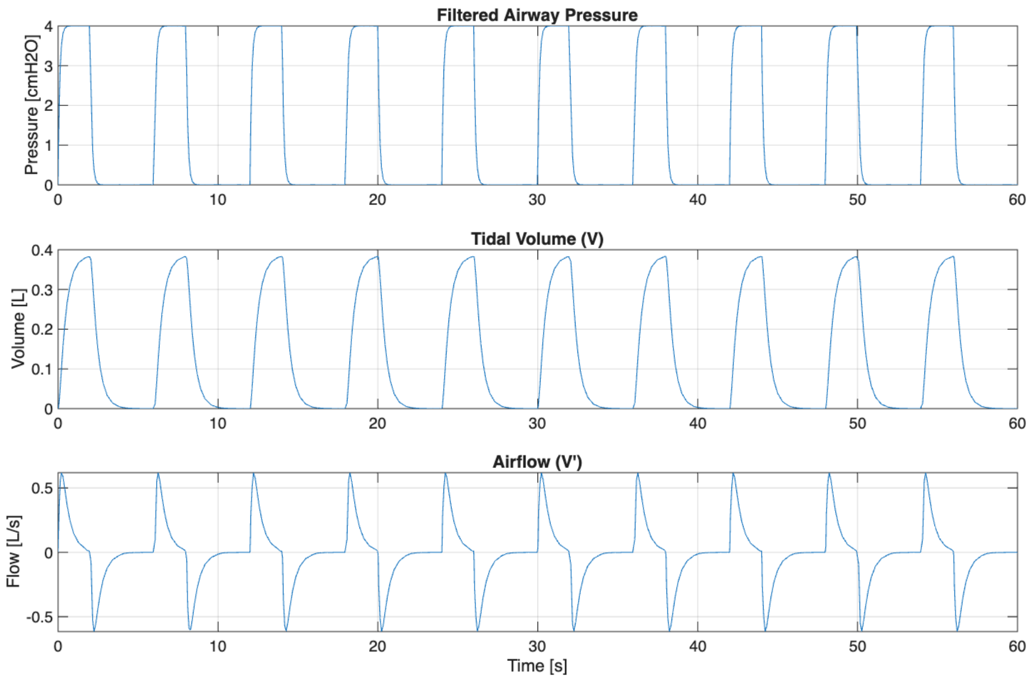


Figure 3: Simulation waveforms for $P_c = 4$ cmH$_2$O.

e. Effect of Increased Airway Resistance

When airway resistance R is increased by 100%, the lung filling process slows significantly. As shown in Figure 4, the tidal volume achieved within the inspiratory time is reduced due to the increased time constant.

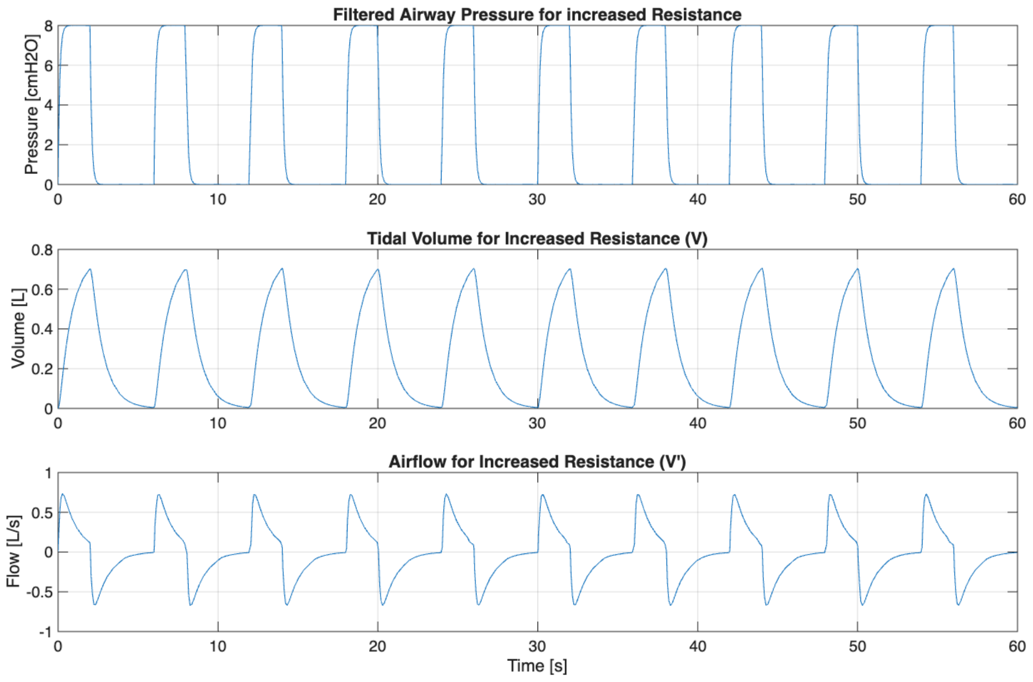


Figure 4: Waveforms demonstrating the impact of doubled airway resistance.

f. VCV Implementation: Volume Reference

In this section, a controller will be implemented for a specific volume reference. The reference that we'll be using is going to be a square wave with an amplitude of V_{ref}, inspiration/expiration ratio of VIE_{ratio} and a respiratory rate of VRR. The frequency of the square wave can be obtained by using VRR and VIE_{ratio}. The Simulink diagram for generating such a reference is shown in Figure 5. 60 seconds divided by the respiratory rate will give us the period of the square wave in terms of seconds. We can obtain the pulse width of the square wave as $100*VIE/(1+VIE)$ (just simple math) and implement this as a MATLAB Function block in Simulink.

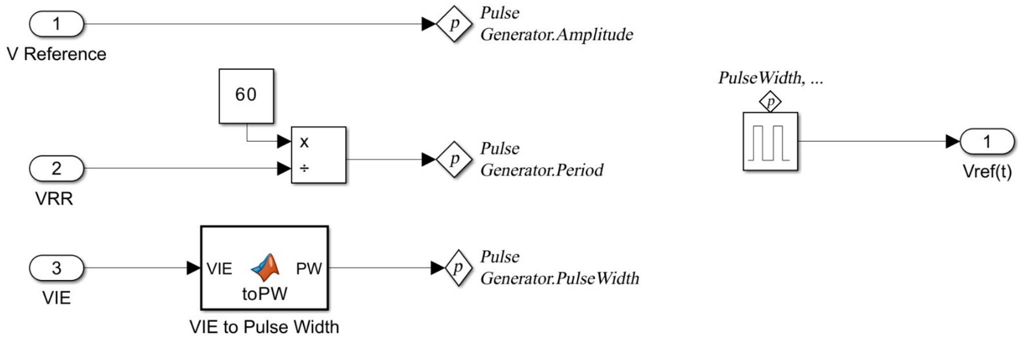
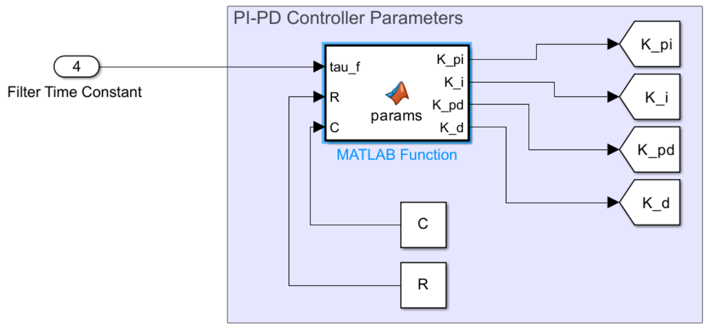


Figure 5: Simulink block diagram for generating the volume reference signal.

g. Calculating PI-PD Controller Parameters

The reference signal is generated using the arbitrary values. MATLAB Function Block takes τ_f, R, C as input to generate PI-PD parameters according to the formulas given in the problem (Figure 6).



$K_{pi} = \frac{16R}{25\tau_f}$	$K_i = \frac{64R}{25\tau_f^2}$
$K_{pd} = \frac{32R}{5\tau_f} - \frac{1}{C}$	$K_d = \frac{23R}{5} - \frac{\tau_f}{C}$

Figure 6: Simulink diagram to set PI-PD parameters. PI-PD parameter values defined in the problem.

R and C values are defined in the MATLAB workspace as 4 and 0.0965. These values are calculated in the previous sections. Time constant τ_f is an input of the subsystem *VCV Mode*.

h. Closed-Loop System Transfer Function

Close-loop block diagram of the system is shown in Figure 7. There are two closed-loop systems nested inside in the figure. One of them is from $X(s)$ to $V(s)$ and the other is from $V_{Ref}(s)$ to $V(s)$.

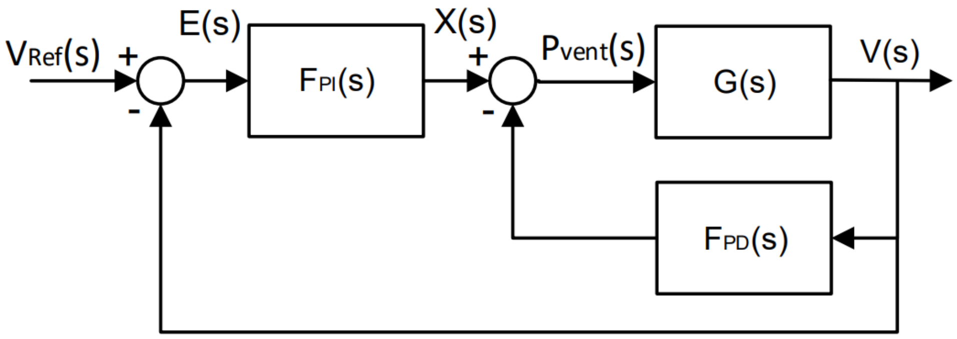


Figure 7: Closed-loop system block diagram.

The transfer function from $X(s)$ to $V(s)$,

$$G_p(s) = \frac{V(s)}{X(s)} = \frac{G(s)}{1 + G(s)F_{PD}(s)} \tag{1}$$

and the transfer function from $V_{Ref}(s)$ to $V(s)$,

$$T(s) = \frac{V(s)}{V_{Ref}(s)} = \frac{F_{PI}(s)G_p(s)}{1 + F_{PI}(s)G_p(s)} \tag{2}$$

Replace $G_p(s)$ in equation (2) above. Therefore, the closed-loop transfer function $T(s)$ can be obtained as,

$$T(s) = \frac{F_{PI}(s)G(s)}{1 + [F_{PI}(s) + F_{PD}(s)]G(s)} \tag{3}$$

Where,

$$G(s) = \frac{1/R}{s + 1/RC} \tag{4}$$

$$F_{PI}(s) = K_{PI} + K_I/s \tag{5}$$

$$F_{PD}(s) = K_{PD} + K_D s \tag{6}$$

If the parameters in Figure 6 are substituted in equation (3),

$$T(s) = \frac{1.198s + 47.93}{s^2 + 13.18s + 47.93} \tag{7}$$

$T(s)$ is a second-order closed-loop transfer function of the system. Pole-zero map of $T(s)$ is shown in Figure 8. Using MATLAB pzmap() function will also give us the overshoot and the damping ratio for the poles of the system. According to the plot, for the poles where $\tau_f = 0.1$, the damping ratio $\zeta = 0.952$ and the percent overshoot $OS\% = 0.0057$. The settling time will be,

$$T_s = \frac{4}{\zeta \omega_n} = \frac{4}{6.59} = 0.606 \tag{8}$$

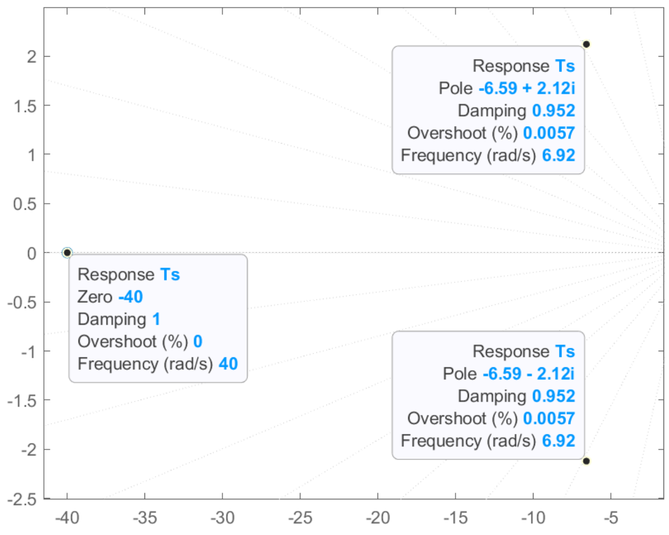


Figure 8: Pole-zero map of $T(s)$. There are two poles at $-6.59 \pm j2.12$ and a zero at -40.

i. Testing the Controller Design in Simulink

Implementation of the system is shown in Figure 9. PI and PD controllers are implemented through the *Varying PID Controller* block. For PD controller, filter coefficient (N) is set to a large number to reduce the effect of the filter to achieve an ideal PD controller. A saturation block is put along the way of the input of G(s) to limit the controller output between [0,30]. The block diagram which sets the controller parameters are shown in Figure 6. The parameters in Table 1 are used for the simulation. Results are shown in Figure 10. $P_{vent}(t)$ plot is show in Figure 11.

$V_{Ref} = 400\,\text{mL}$	$\tau_f = 0.1\,\text{s}$
$VRR = 10\,\text{breaths/min}$	$VIE_{ratio} = 1:2$

Table 1: VCV parameters

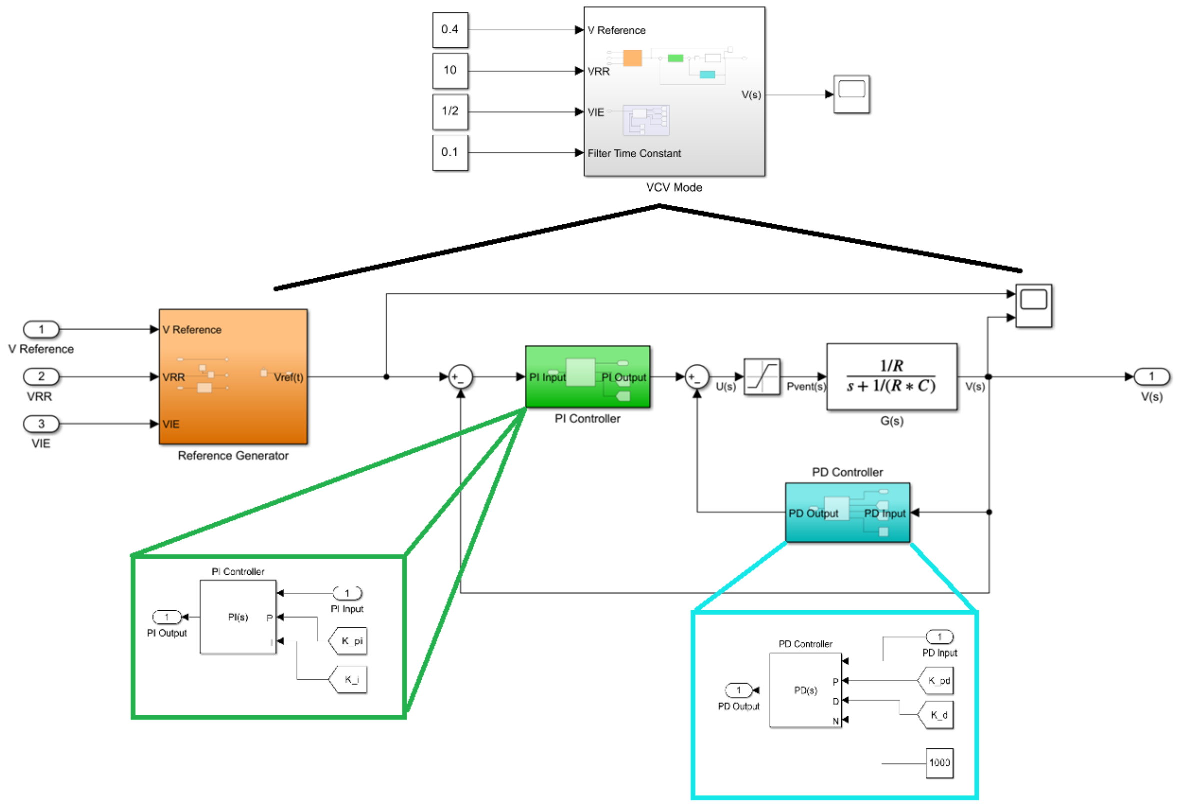


Figure 9: Simulink block diagram of the VCV system.

If we compare the reference volume and the output in Figure 10, we can say that the controller is pretty successful in controlling the tidal volume. It does not overshoot much and settles fast. Also we can see that system is constantly working without any problem to reach the desired tidal volume.

j. Chronic Obstructive Pulmonary Disease (COPD)

When simulating the same system on a person who has COPD, lung compliance is multiplied by 1.5 and the other parameters are kept the same. The results are shown in Figure 11. If we look at the $V(t)$ graph, we can clearly say that we achieve the desired tidal volume after more time passed when we compare it to the person who does not have the disease. That is what we should expect since the lung compliance is analogous to the capacitance in a RLC circuit. Also, $P_{went}(t)$ graph shows that the patient with this disease needs more pressure applied to his lungs to reach the same desired tidal volume.

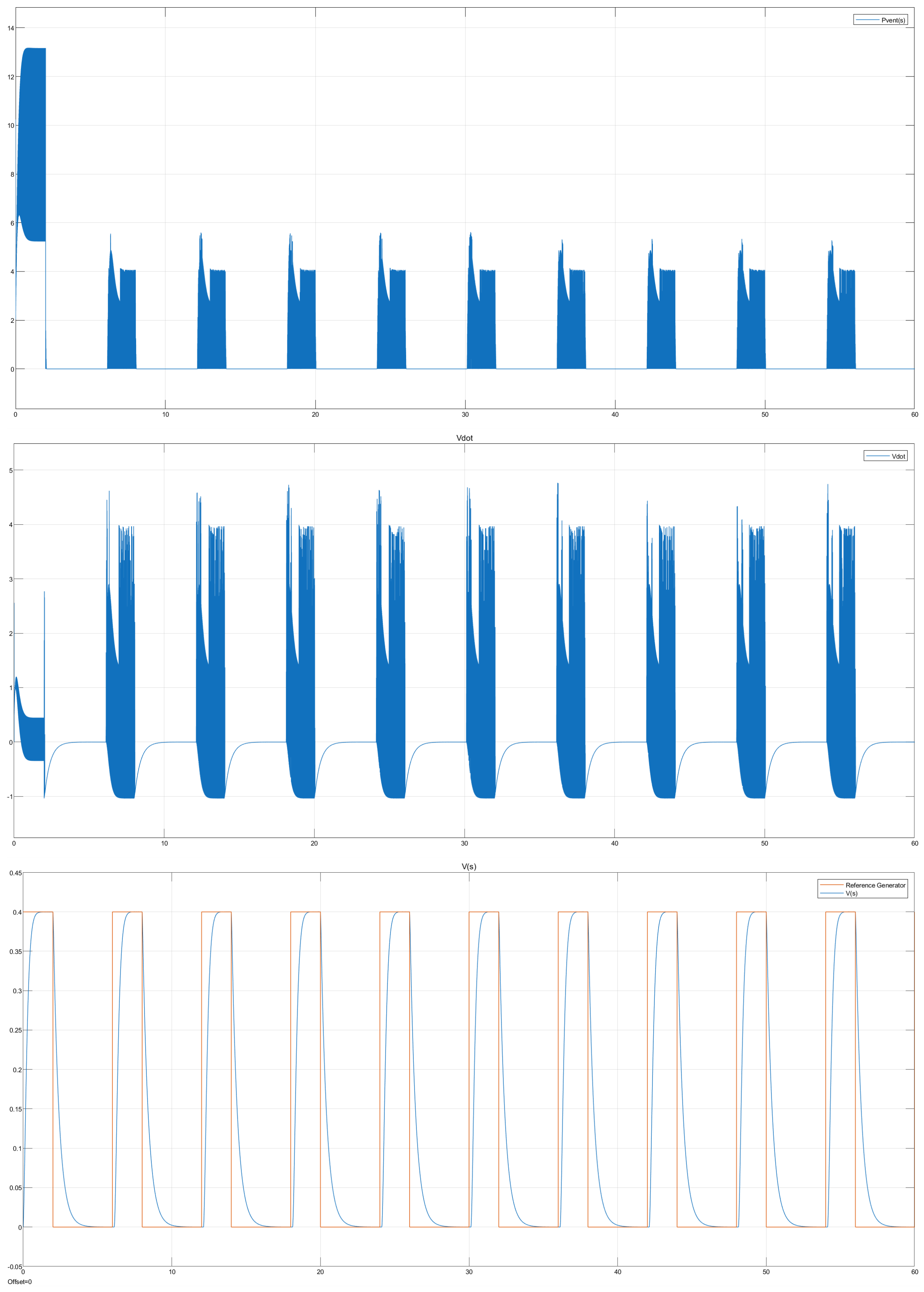


Figure 10: Simulating for 60 seconds. $P_{vent}(t)$, $\dot{V}(t)$ and V(t).

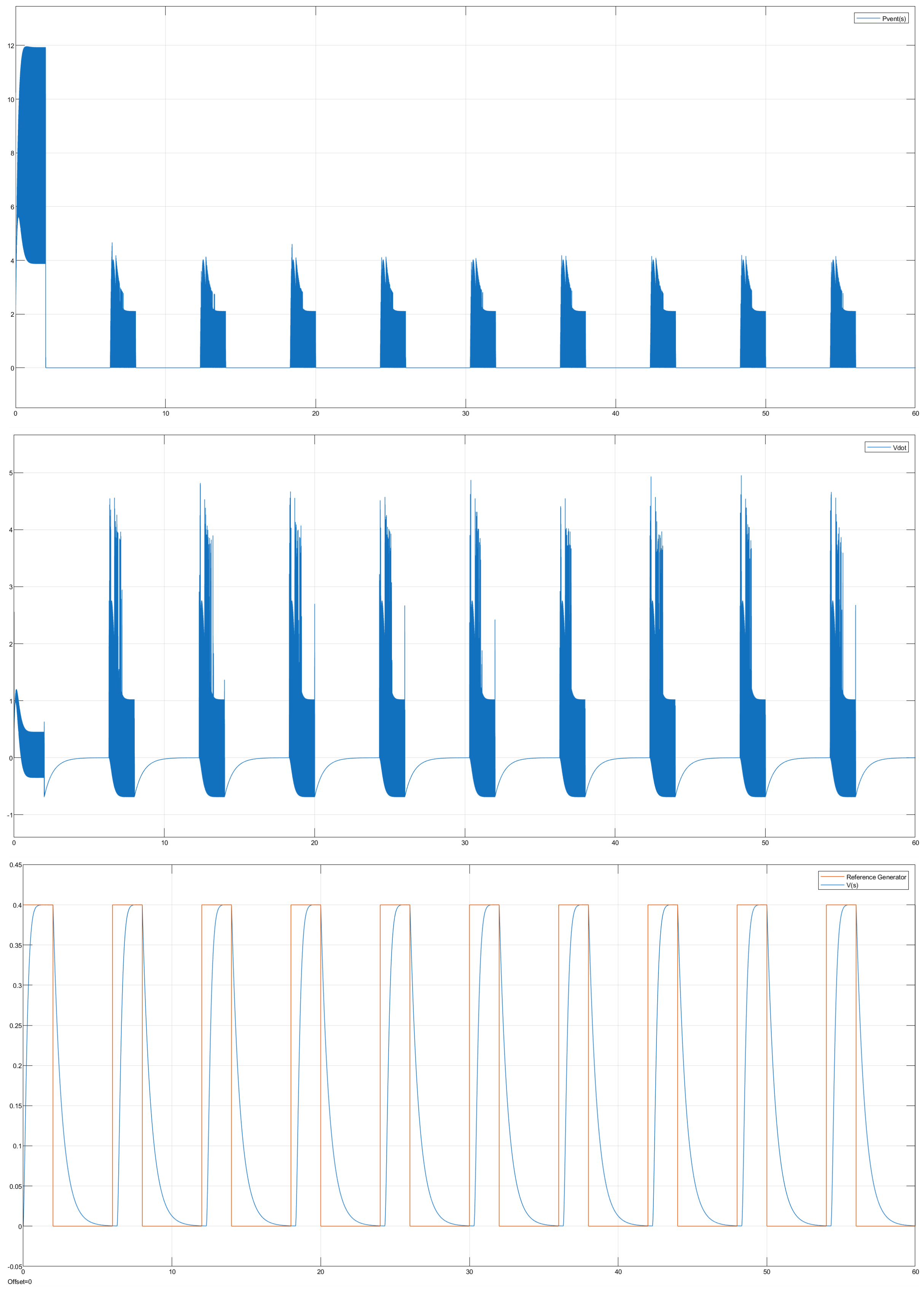


Figure 11: Simulating for 60 seconds on a person with COPD. $P_{vent}(t)$, $\dot{V}(t)$ and V(t).