*VisSim Tutorial Series*

# Biomedical Systems: Modeling and Simulation of Lung Mechanics and Ventilator Controls Design

***Mike Borrello,*** *Metran America, Inc.*

**VisSim Tutorial Series**

**Biomedical Systems: Modeling and Simulation of Lung Mechanics and Ventilator Controls Design**



Excerpted with permission from *Modeling and Visual Simulation in Industry*, A. Mulpur and P. Darnell, International Thomson Computer Press, Boston, MA, 1997.



Other books in the VisSim Tutorial Series include:

- *Fundamentals of Mathematical Modeling and Simulation*. Peter Darnell and Arun Mulpur, Visual Solutions, Inc.
- *Heating, Ventilation and Air Conditioning (HVAC) Controls: Variable Air Volume (VAV) Systems*. Nebil Ben-Aissa, Johnson Controls, Inc.
- *Introduction to 6-DOF Simulation of Air Vehicles*. Robert Josselson, ITT Aerospace Systems Group.
- *Simulation of Communication Systems*. Eugene Estinto, Eritek, Inc.
- *Simulation of Motion Control Systems*. William Erickson, Indramat-Rexroth.

# Table of Contents

# Introduction

This tutorial focuses on one representative application in the field of biomedical engineering: the modeling and simulation of lung mechanics for the purpose of designing feedback controls in mechanical ventilation. It contains the following information:

- How to derive a simulation model in VisSim
- How to apply measurement and control using real-time interface methods
- How to transfer the design to end product embedded controls

For those unfamiliar with programming with block diagrams, this tutorial offers a step-by-step example, starting with basic model development.

In *Model Development*, the general aspects of modeling lung mechanics are introduced. Components of the system are described and reduced to block models. The components are then assembled into a single mechanical ventilation model. Model variations are then considered and responses are compared with real-time, closed-loop control responses of a real system to select the simplest, but most suitable model for ventilator controls design.

The section *Using the Model in Simulation* describes how the VisSim lung/ventilator model is used in simulation to investigate properties and behavior of lung mechanics with feedback controls.

The section on *Programming VisSim for Embedded Applications* describes certain considerations when using VisSim for desktop hardware-in-the loop applications and real-time issues. Transferring the control design to embedded hardware is then discussed with tips on how to make this transfer as accurate and easy as possible.

The methodology presented in this tutorial is not only applicable to ventilation, but also to any product with embedded controls.

# Overview of a Ventilation System

When a patient's ability to breathe becomes unstable or compromised, a mechanical ventilator is often required to assist or control breathing. The ventilator provides precise control of the flow, pressure and composition of the gas for the patient. A ventilator typically employs a supply flow valve and an exhalation valve, which delivers gas into the patient's lungs and vents gas to atmosphere, respectively. By sequencing the operation of these valves, the act of breathing can be emulated. Gas reaches and is expelled by the patient through flexible tubular conduits known as the patient circuit. The exhalation and flow supply valves are connected to separate legs of the patient circuit, which join at a Y piece. The Y piece connects to a face mask, or under more invasive situations, a tube inserted into the patient's trachea. Pressure and/or flow sensors monitor the state of the gas and provide measurements for feedback control. Figure 1, shown below, illustrates the components of a typical ventilation system.

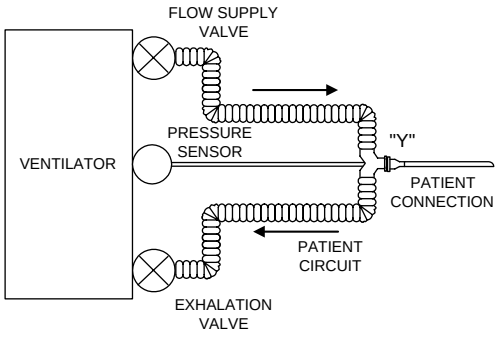


*Figure 1. Ventilation system*

The patient's condition determines the "mode" of ventilation prescribed to maintain life support. In most high-end ventilators, feedback control is used to support the operation of these modes and in a manner that is robust to a wide range of patients, patient circuits, and operating conditions.

Nonlinearities, time lags, and wide variability of parameters make the problem difficult to analyze by traditional control methods. Although these systems are not easily broken apart analytically, they can still be approximated mathematically and written into simulation for systematic study of behavior under various control techniques. This requires a fundamental understanding of the properties and behavior of the flow of gases and some familiarity with the anatomy and physiology of breathing.

There have been numerous repeated efforts to develop mathematical models of lung mechanics; however, for the study and application of feedback control in ventilation, these models lack key features that allow proper assessment: consideration of the patient circuit mechanics, and the nonlinear behavior of flow and pressure in various components. After several years of work on the problem, these essential features have been incorporated into a model that better serves its purpose. With the use of a visual simulation tool, such as VisSim, this model has been refined to the point where controls can now be totally designed in a simulation environment before transfer to the embedded software application. Furthermore, the real-time interface provided by VisSim allows linking mechanical components, such as valves and sensors so that the final control design can be completed in a PC Windows environment prior to working with the embedded processor. This helps isolate control design flaws from software bugs. Design and verification of the controls alone, before code issues are introduced, can greatly speed the time to market.

# Model Development

The ventilator/lung system consists of:

- The patient circuit
- Valves that artificially control flow into and out of the patient circuit
- An endotracheal tube that connects the patient circuit to the patient's upper airways
- The patient's lungs and connecting airways
- The patient's neuromuscular system that naturally drives ventilation

These components may, however, be broken down into smaller components that can be individually described by simple mathematical relationships. These relationships typically contain algebraic as well as dynamic components.

To control mechanical ventilation, a useful model must contain parameters that have a direct link to physical elements in the system. The model must behave very closely to the real system regardless of the parameters chosen.

Historically, strictly linear models were employed, which were useful when analytically deducing overall effects of various parameters on the behavior of the system. For simulation, however, the exclusion of nonlinearities also excludes certain characteristics observed in real systems, such as saturation effects and limit cycles, which are important considerations in ventilator controls.

In the clinical environment, the patient's anatomy, valves, circuits, and other components of the ventilation system may not be well defined. Disease or injury can add a further degree of complexity to the problem. For this reason, among other obvious reasons, the design and evaluation of ventilator controls is most often done using mechanical devices that emulate the elastance and resistance of a lung. These devices include fixed isothermal compartments, as well as adjustable spring loaded bellows. Developing a simulation model can help bridge the gap between mechanical devices and actual clinical application. Complex features are much easier to implement in simulation.

As with any physical system, it can be described mathematically by examining individual components. In the ventilator/lung system, each physical component is looked at in order to describe how parameters of each component interact with flows and pressures in the system.

## *Patient Circuit*

The patient circuit is where control of pressure and flow takes place since this is typically where pressure is monitored. The patient circuit is the conduit that provides the transfer of flow from the ventilator to the patient and from the patient to the atmosphere,

Much like a spring's elastic properties allow it to store energy by deflecting its coils, a closed compartment can store energy by introducing gas volume. In a spring, a set deflection stores a fixed force. In the closed compartment, an introduced volume creates a set pressure. Neglecting temperature effects, the closed compartment has a near linear relationship between introduced gas volume and pressure. The change in pressure with respect to change in volume is constant and known as the elastance of the fixed container. If the walls of the container remain rigid and fixed, this relationship holds true. If, however, the walls are flexible, and deflect as volume is added, a nonlinear relationship occurs between flow and pressure. The patient circuit can be modeled using the properties of either rigid or semi-flexible compartments.

Most analytical studies that model mechanical ventilation have entirely neglected the effects of patient circuit dynamics. Consideration of the patient circuit elastance is essential to provide a realistic response. Resistance of the circuit should also be considered when the circuit is highly restrictive and the controlled pressure is at the patient Y. This is almost always the situation when neonatal circuits are used.

The patient circuit typically consists of long lengths of tubing for both the inspiratory and expiratory leg. This can impose significant process delays with respect to both sensing and actuation of pressure and flow. The delay imposed is governed by the speed of sound: about 1 ms for every foot of circuit.

The patient circuit compresses or expands gas based on flow being introduced by the flow supply valve or vented by the exhalation valve, respectively. For the most part, circuit elastance is primarily the elastance of the gas within the circuit compartment. Thus, pressure is linearly proportional to volume introduced to the fixed geometry. For some patient circuits, which may not have very rigid walls, the length or diameter of the circuit may change slightly with increasing pressure, causing a nonlinear relationship of pressure to volume (nonlinear elastance).

For a rigid walled patient circuit, disregarding circuit resistance, the model can be written as

$$P_C = E_C \cdot \int (Q_p - Q_L)dt$$

Figure 2 shows the block diagram representation of the above equation.

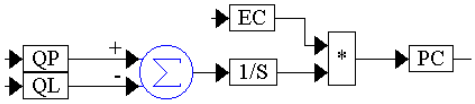


*Figure 2: Patient circuit block diagram*

## Endotracheal Tube

The endotracheal (ET) tube, which links flow between the patient circuit Y piece and the patient's upper airways, gives the system its strongest nonlinear characteristic relating the delta-pressure and flow parameters. Although nearly exact relationships can be described by writing the compressible gas equations for continuity, state and energy, this description can become quite complex and cumbersome. For nearly all ET tube sizes, a simple parabolic relationship for flow and pressure provides an accurate description within 1 or 2%.

ET tubes are flexible, and depending on curvature during insertion may have more or less resistance, but these effects are minimal with respect to the fundamental characteristics. Tube diameter sizes range from 2.0 to 3.5 mm (neonate) to about 4.0 to 5.0 mm (pediatric) to 5.5 to 10.0 mm (adult). Both the diameter and length of the ET tube effect the resistance to flow. However, the range of possible diameters has more of an effect on the parabolic flow constant than length. The smaller the diameter, the larger the size of the parabolic constant.

The parabolic constant directly effects the time constant of the lung (albeit nonlinear time constant). The possible range of parabolic constants represents a span of three orders of magnitude. This creates time constants that vary with the same degree of severity. For a simple closed-loop pressure circuit with a fixed gain, uniform performance over such wide range of load conditions cannot be expected. This is the crux of the problem in ventilator controls. The control must either be more than simple or at least very clever.

Considering the nonlinear relationship, and that flow through the ET tube is bi-directional, the equation for the ET tube becomes

$$P_C - P_L = K_L \cdot Q_{L^2} \cdot sgn(Q_L)$$

Figure 3 shows how this equation is represented in block diagram form.

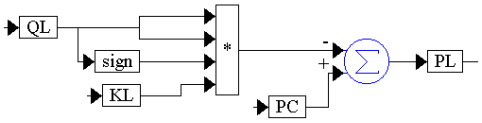


*Figure 3. Endotracheal tube block diagram*

Studies using VisSim reveal that some of the slow limit cycles observed in ventilation controls may not be due to valve seat properties, but rather to a closed-loop near resonant condition and the nonlinear resistance of the ET tube. Without this nonlinearity present, stable oscillations dissipate energy and eventually settle towards a point of equilibrium. Unstable oscillations expand towards

infinity. If the nonlinear resistance is introduced, the oscillations expand but reach a bounded amplitude. These sustained oscillations are known as limit cycles. In simulation, limit cycles persist in situations where the dynamics of the valve mechanisms have been totally removed.

## Flow Valves

Control of the flow of gas into and out of the lungs is typically done by separate flow supply valves and exhalation valves, respectively. These valves are actuated by motors or linear actuators.

For a flow supply valve, upstream pressure on the valve is high relative to downstream pressure and is fairly constant. In this situation, the flow source is of low impedance, and the delivery of flow is not so sensitive to downstream loading. The model for the valve can usually be considered a function independent of the pressure differential.

Assuming a look-up table is used in the final design to linearize gain, or that flow feedback is used, the flow to command relationship can be treated as a simple constant. The dynamics of the actuating device and/or flow feedback loop should also be included. This is most often a DC motor, stepper motor, or linear voice coil actuator feedback loop for valve actuation. The equation describing a flow supply valve is written as

$$Q_v = K_v \cdot v \cdot H_i(s)$$

Figure 4 shows the block diagram of the equation.

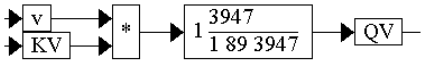


*Figure 4. Flow supply valve block diagram*

Unlike the flow supply valve, the exhalation valve upstream (circuit) pressure is not much higher than downstream (atmospheric) pressure. In this case, the flow source is of relatively high impedance and is effected by the load. For the exhalation valve, the model can be written using a parabolic relationship between differential pressure and flow with a “variable” coefficient that represents the flow control area. Assuming atmospheric pressure is constant, the expression for the exhalation valve can be written as a function of circuit pressure. Of course, like the inspiratory valve, the dynamics of the actuating device should also be considered. The equation describing an exhalation valve is written as

$$Q_E = A_o(v) \cdot \sqrt{P_c} \cdot H_e(s)$$

Figure 5 shows the block diagram representation of the equation.

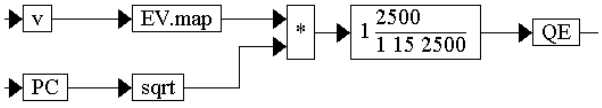


*Figure 5. Exhalation valve block diagram*

Because of the substantial difference in the flow-pressure relationship in the two valves, two separate control loops are required. During exhalation, pressure is measured in the circuit, and depending on the particular method of ventilation, is controlled by releasing gas to atmosphere through the exhalation valve while the inspiratory flow valve is either off or fixed to a constant value of flow. During inspiration, the exhalation valve is shut off and the inspiratory flow valve is

controlled in feedback with the pressure measurement. More advanced methods of ventilation may require that both control systems be active simultaneously.

## Patient Resistive Airways

In addition to the resistance created by the ET tube, patient airway resistance must be considered in the actual clinical application. This resistance can be further grouped into the “upper” airway resistance imposed by the trachea and upper branches, and the “lower” airway resistance imposed by the minute and numerous branches that lead to the alveoli. All these passages are subject to change both during the course of a breath and over longer periods of time. Other than the typical nonlinear relationship of the pressure drop to flow through conduits, additional time varying resistance changes may occur in the airways due to mucous, blockage, injury, or deformation of the anatomical structures. These effects are often unpredictable, unrepeatable, and difficult to define. They can, however, be approximated in simulation by perturbing the selected parameters with bounded variations created by noise sources passing through filters representing a specific power spectral density (colored noise) or by designing a random event generator for discrete or sudden events. Such modeling can prove useful in determining, for example, the reaction of the control system to coughing or Cheyne-Stokes breathing.

## The Lung as an Elastic Compartment

The lung is a compartment with flexible walls that significantly expand as volume is introduced. Thus, as described in the section on patient circuit, the lung follows a nonlinear relationship as described above. This nonlinear relationship of pressure to volume typically follows an “S” curve, as shown in Figure 6.

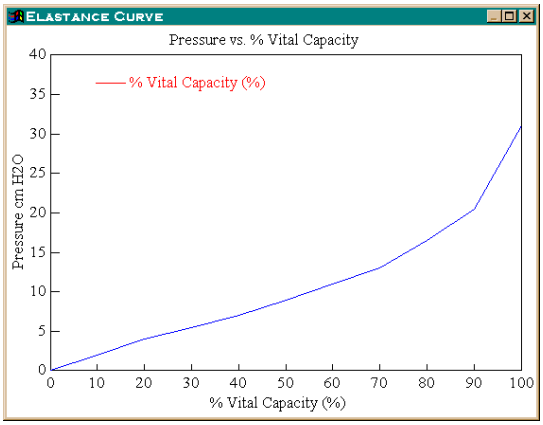


*Figure 6. Pressure volume relationship in human lung*

The lungs are enclosed in a cavity between the parietal and visceral pleura. The diaphragm and its actuating muscles cause the intrathoracic pressure within this space to drop below atmospheric pressure. This causes the lungs to expand and air to be drawn down into the trachea and numerous spaces beyond. The degree of elasticity of the lung with respect to the strength of the diaphragmatic muscles effects the ability of the patient to draw in an adequate volume of gas. Elasticity of the lung can be effected by various diseases and /or injuries . A stiffer lung (larger elasticity) makes it more difficult for the patient to draw in gas.

In the clinical test lab, the “S” curve relationship that exists in a real lung is often difficult to emulate using linearly elastic test lungs. Simulation provides a way to overcome this problem.

For a linearly compliant lung, the equation is written as

$$P_L = E_L \cdot \int Q_L dt + P_M$$

The corresponding block diagram for this equation is shown in Figure 7.

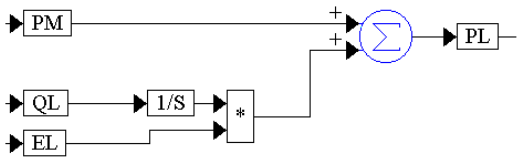


*Figure 7. Linearly elastic lung equation block diagram*

To simulate a nonlinear elastic lung, the data shown in Figure 6 can be curve fit to obtain coefficients for a polynomial, or a sufficient number of data points can be taken as a table and used by VisSim's `map` block. The `map` block linearly interpolates between data points to provide a continuous relationship in the simulation. The block diagram for the nonlinear elastic lung is shown in Figure 8.

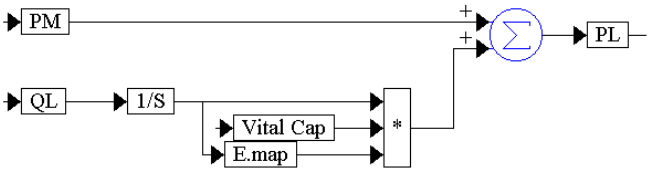


*Figure 8. Nonlinear elastic lung block diagram*

## Respiratory Drive

Contraction and relaxation of the patient's diaphragmatic and thoracic muscles provide a cyclic pressure differential from within the lung to atmosphere, causing flow to enter and exit the lung. The driving signal for the muscle groups is a complex function of chemical and physical relationships in the patient. The controlling signal for the respiratory drive constitutes an additional, complex feedback loop that may interact with ventilator pressure and flow control systems, possibly leading to an overall undesirable response. In some cases, this interaction can cause the patient to fight with the ventilator.

# Model Construction

For a "lumped parameter" model, several configurations can be considered by either including or excluding certain elements or components. For example, the patient circuit elastance can be included or the simple linear R-C model can be used. A simple ET tube resistance or the nonlinear characteristic can be used. To begin, consider the following electro-acoustic analogy of the lung-patient circuit shown in Figure 9.

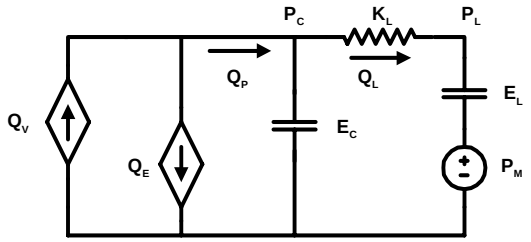


*Figure 9. System circuit analogy*

| | |
|---|---|
| $Q_V$ | Flow from inspiratory valve |
| $Q_E$ | Flow venting exhalation valve |
| $Q_C$ | Net flow |
| $P_C$ | Patient circuit pressure |
| $K_L$ | Parabolic flow constant of ET tube |
| $E_C$ | Circuit elastance |
| $P_L$ | Pressure in lung |
| $E_L$ | Lung elastance |
| $Q_L$ | Flow into lung |
| $P_M$ | Respiratory drive pressure |

In this circuit analogy, $E_C$ can be set to zero to ignore patient circuit elastance. $K_L$ can be considered a constant for a linear ET tube resistance or as a function of $Q_L$ for the nonlinear model. The analogy provides a structure to visualize the behavior of flow delivery to the lung and the effects on pressure. Once a simulation is constructed, the behavior can be quantified precisely.

$Q_P$, the net difference of flow delivered and flow vented ($Q_V$ minus $Q_E$), is split into $Q_L$, the flow which enters the ET tube, and the remaining flow which compresses the gas within the patient circuit elastance. The compression of this remaining flow causes the circuit pressure, $P_C$. Note that if $Q_L$ is zero, an equilibrium in pressure exists between the circuit and lung elastance. If $K_L$ is relatively small, this equilibrium adjusts itself much faster than when $K_L$ is larger. If $E_C$ is not considered, the rise in circuit pressure is instantaneous.

## Constructing the VisSim Block Diagram

Using the component descriptions, and the electro-acoustic analogy, a block diagram can be constructed in VisSim for simulation. If a nonlinear model is chosen, the block diagram shown in Figure 10 is the result.

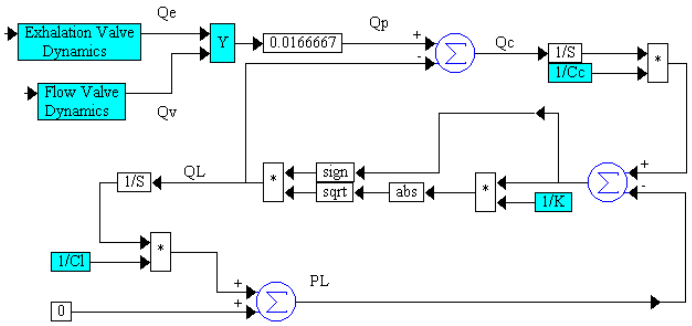


*Figure 10. VisSim simulation model of ventilator/lung system*

To characterize the nonlinear resistance, actual ET tubes ranging from 2.5 mm to 9 mm are connected to a motor-driven flow source. A flow transducer is connected in line, as well as an upstream pressure transducer. VisSim is then used to automatically ramp the flow and acquire flow and pressure data for each ET tube size. The data is used to determine the parabolic flow constant for each ET tube. The parabolic flow constants are plotted against their respective ET tube diameters, and the data is processed through a regression analysis to determine a best fit polynomial. From this polynomial, a VisSim block is constructed that calculates an appropriate parabolic constant given the ET tube diameter.

## Model Simplification

One of the goals of the model is to keep it as simple as possible, without compromising realistic behavior. One method for deciding model complexity is to construct several models, each including or excluding various components, and compare the response of these models with the response of the real physical system.

For the ventilator/lung system model, four lung models can be constructed:

- A simple linear RC model
- A nonlinear RC model
- A linear RC model including circuit elastance
- A nonlinear RC model including circuit elastance

A very simple PI feedback controller is constructed, duplicated, and connected to each model. A fifth feedback controller is duplicated and connected to real-time pressure measurement feedback and real-time output. The fifth controller controls actual valve flow to a mechanical test lung. The controls are de-tuned to purposely overshoot and excite transients. ET tube sizes and mechanical lung elastance are varied with matching simulated lung and circuit parameters to determine which model best compares to the real-world response. Figure 11 illustrates a sample comparison run for one of the various lung conditions.

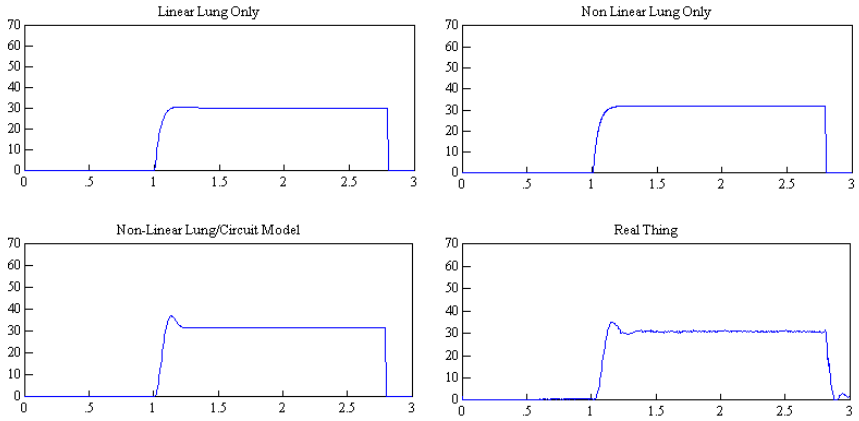


*Figure 11. Model comparison*

By varying parameters of resistance and compliance, and observing response, the following conclusions can be made:

- RC models without circuit elastance, whether linear or nonlinear, do not adequately model response in most cases. Circuit elastance is necessary to obtain the type of dynamics observed in real feedback systems.
- For systems that experience minor changes in flow and pressure, there is little difference between the linear and nonlinear models except when resistance is very high. In general, for systems that experience major changes in flow and volume, it is important to include the nonlinear behavior of the resistance and elastance. Generally, dynamic behavior is more sensitive to the resistive components than elastic components.

By making these comparisons in VisSim, it can be seen that over the range of possible parameter variations the patient circuit should be included along with the nonlinear ET tube model. Thus, the circuit analogy shown earlier in Figure 3 represents a fairly good descriptive model.

When considering infant applications, the resistance of the patient circuit becomes a significant factor, especially if pressure, measured at the flow or exhalation valve is to be used in feedback. Nonlinear resistive elements similar to the ET tube resistance, but representing circuit resistance, should be inserted in series with each flow source. The parabolic flow constants for each leg of the circuit should be determined by running a characteristic pressure vs. flow test. The resulting data can be used to estimate the constants.

# Using the Model in Simulation

Now that a block diagram model of the ventilator /lung system has been developed and verified with an actual system, it can be applied towards the design of feedback controls with the goal that these feedback controls will be applied in the final product as software (code) running in a real-time embedded processor.

Before beginning to design any feedback controls, the block diagram model can be used to explore properties that would otherwise be difficult or impossible to investigate in the real lung/ventilator system. This type of exploration can provide insight and a deeper understanding of the system. Certain fundamental behaviors are revealed that guide the designer towards the selection of particular control methods.

## *Investigating More Complex Properties of the Lung/Ventilator Model*

With a fairly accurate and consistent model in simulation, complex properties can begin to be explored such as the effects of nonlinearities under feedback control.

Without feedback controls, the passage of flow through the patient circuit to the lungs is a stable process free of resonant dynamics. With feedback, however, there is a trade-off between fast accurate control and stability. The trade-off would be trivial for a motion control system with a fixed load. Assuming the use of linear controls in ventilation, however, the wide variation of ET tube sizes, lung elastance, and patient circuits will make the system slow and sluggish at one extreme and highly oscillatory at another.

The parameter of circuit elastance is crucial to the structure of the model reinforces the idea that the system is dominantly a two compartment or second order system. When the ET tube becomes less resistive, the system breaks down into what appears like a single compartment and the closed-loop dynamics behave extraordinarily differently.

In an actual ventilator system, dynamic behavior is often subdued by saturation limits of the control elements, or breath phase cycling criteria. Valves either supply flow or vent gas to atmosphere, but usually not both. In simulation, hypothetical situations without limit can be set up. For example, a valve that can both vent flow as well as supply it. Such extensions to the real world in simulation can provide a better understanding of what is actually occurring. The plots in Figure 12 show a step response for a low resistance, stiff lung, the first using a realistic flow valve, and the second using a hypothetical valve that sinks flow as well as sourcing it.

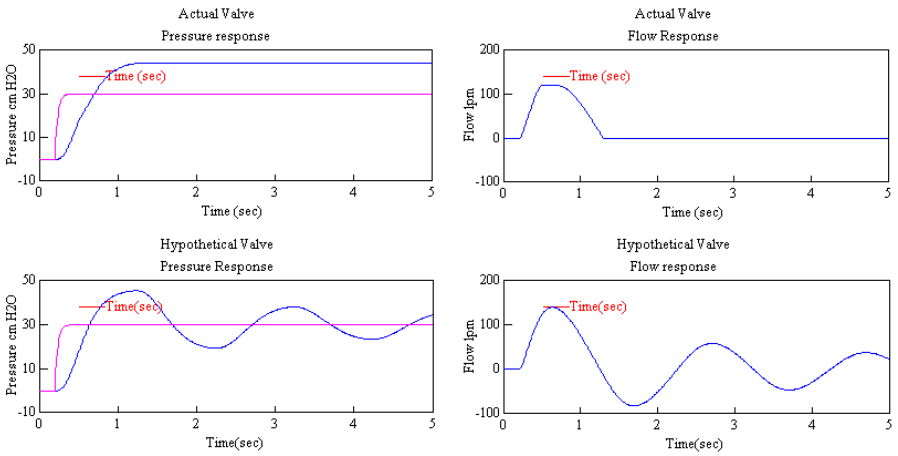


*Figure 12. Comparison of closed-loop pressure response between system with and without hypothetical flow valve*

Under feedback control, the real valve shuts down rapidly and the pressure locks up at a level above the intended trajectory. In attempting to correct the design, the gain can be increased, which for this particular lung, does reduce the steady state error. The problem with increasing the gain is that this control will harshly oscillate if a resistive ET tube is introduced. The hypothetical valve, which allows gas to be vented as well as supplied, shows that the dynamics are oscillatory at a low frequency. This indicates that a more robust control structure should be employed.

In controlling set pressure values with the exhalation valve under constant bias flow, one of the problems often encountered is low frequency, persistent oscillations. These oscillations can be uncomfortable for the patient, as well as interfere with breath triggering systems in the ventilator. Historically, these oscillations have been attributed to the dynamics of the exhalation valve;

however, studies in VisSim have shown that they could be limit cycle oscillations. These limit cycles are caused by the two compartment system connected by a nonlinear resistance and subjected to pressure feedback. The nonlinear resistance (ET tube) is the primary cause of the limit cycles. An experiment was done in VisSim where the exhalation valve dynamics were removed (made infinitely fast). With a hypothetical linear resistive ET tube, induced oscillations would either die out or expand depending on the controller gain. Limit cycles are not possible using a linear resistive (simulated) ET tube; the oscillations either expand or die out. Limit cycle magnitude is larger with the lower resistive tubes, the same as observed in a real ventilator. Figure 13 shows a phase plot (pressure vs. flow) of a limit cycle observed in the study.

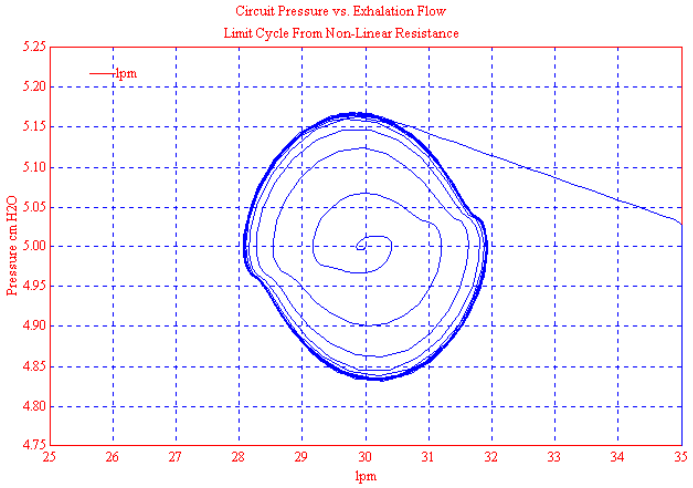


*Figure 13. Limit cycles caused by nonlinear resistance*

# Programming VisSim for Embedded Applications

## *Hardware-in-the-Loop Simulation*

Desktop engineering now includes the realm of real-time hardware-in-the-loop simulations. The ability to design and test controls before designing hardware and software speeds up product development, reduces development costs, and ultimately provides a better product to the customer. In practice, it is as easy as installing a compatible data acquisition card in a PC and wiring it to the system sensors and actuators.

Very often, development of products that require control and an embedded processor are carried out using the processor itself as the development platform. This method of development requires the controls to be written up front, in the language of the processor, which may be assembly or, at best, a high level language, such as C. Even with a high level language, the control development requires a control engineer proficient in software, a software engineer who understands control design, or the two working together with mutual understanding.

Of even more concern is the difficulty of debugging the design. Sometimes it is hard to narrow down the cause of an inappropriate system response. *Is the problem a control design bug or software bug?* Iterations can be numerous as well as lengthy before a final design is reached. Separating control design development from software design development reduces this uncertainty and allows each engineer to focus on issues within their discipline. Hardware-in-the-loop simulations allow this separation, and together with VisSim, the PC serves as the means for doing so at relatively low cost. VisSim's GUI environment allows rapid development and testing.

More iterations in a shorter time frame are possible, and greater continuity in the thought process is achieved. The control designer can focus on the control design, not writing, compiling, and debugging code.

VisSim/Real-TimePRO is an interactive real-time data acquisition and control add-on option to VisSim. It provides drivers for numerous boards from popular manufacturers for various applications: analog input, analog output, thermocouple interface, digital I/O, and motor controller/ resolver interfaces.

Before beginning, it is a good idea to review some considerations when building real-time feedback controls in either the PC or embedded environment.

## Aliasing and Filtering Measurements

For applications involving real-time feedback control with hardware-in-the-loop, it is important to consider problems that arise as a result of sampling and sampling signals that contain noise at frequencies that exceed half the sampling rate. Many questions about aliasing that are not considered in this tutorial are answered in *Digital Signal Processing* by Oppenheim and Schafer.

When sampling flow and pressure for feedback control, or any other sampled measurement for that matter, it is imperative to consider the effects of aliasing and provide adequate filtering to prevent such problems. High frequency noise that is allowed to be sampled by the A/D converter can appear as an aliased (lower) frequency in the sample band. The control loop sees the aliased signal as an actual perturbation on the measurement and attempts to correct for it by actuating the valve. Response to the aliased signal propagates noise into the control loop. Depending on the amplitude and frequency of the noise, it can create disturbances anywhere from low level jitter to significant oscillations.

Quite often engineers not familiar with the mechanism of aliasing, “filter” the input measurements by processing the converted signal through a software filter; however, aliasing has already occurred at the A/D, and sneaks right past the software filter in the passband. Digital filtering is not only ineffective in removing effects of aliasing, but also adds additional and unnecessary phase lag to the control loop. The only way to effectively prevent aliasing is to use a hardware filter before the A/D converter.

The application of anti-aliasing filters in the hardware-in the-loop simulation should carry over to and be incorporated as part of the final embedded design.

## Verifying Real Time in VisSim

Another important issue is whether all the calculations are completed within the allotted sample period. At present, VisSim provides no warning except a message at end of simulation whether the computations performed exceeded real-time capability of the host processor. If the Auto-Restart (continuous) and Real-Time options are activated, it is recommended to see if the rate that has been specified in VisSim’s Simulation Setup menu is being achieved by first disabling Auto-Restart then performing a single run. Although the end of simulation message does provide numbers for real-time and simulation time, these measurements are not very accurate. It is better to measure real-time against simulation time directly by comparing them on a plot.

To run a real-time test, enclose a `plot` block within a compound block. Activate the XY plot and Discrete plot options, and select a number of points to exceed the total steps for the simulation by 1. Plot the output of a `realTime` block on the vertical axis and on the horizontal axis, plot the output of a `ramp` block multiplied by 1,000. Activate the Fixed-Axis option in the `plot` block and set the coordinate span for the simulation range multiplied by 1,000. Run the simulation with Auto Restart deactivated. The plot should appear as a staircase line from 0 to the range end in milliseconds without overrun, underrun, or any breaks or discontinuities. Figure 14 shows how the output should appear.

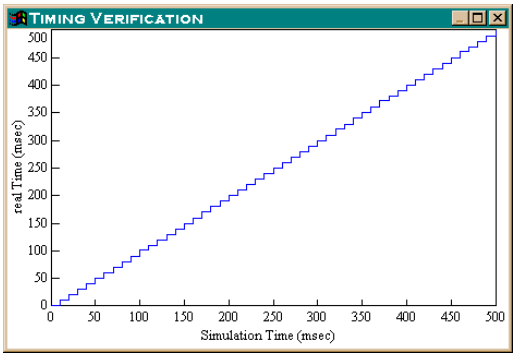


*Figure 14. Checking for proper system timing in VisSim*

By checking timing this way, a very fine resolution can be achieved, and intermittent or discrete events (such as, background processing or TSRs), which may be occurring and effecting real-time, can be detected. While adding to a design, it is best to check periodically to verify system resources have not been exceeded. VisSim release 3.0 is expected to include a monitor on the status bar that will alert the user in the event of timing problems, should they occur. The method described above can detect these problems, as well as help diagnose their cause.

To obtain as much bandwidth as possible in the PC, disable screen savers and terminate network operations such as local backup or sharing operations that can cause glitches in the real-time operation. If the chosen step size is overrun, a larger step size can be chosen if possible, or the design can be edited to remove blocks that tend to eat up excessive time. Such blocks include `displays`, `stripCharts` and `lights`. Plots that specify a large number of points also eat up time, so it is best to keep the number of points at a minimum. Fix the bounds on `plot` blocks where possible. When auto-scaling occurs, significant time is lost, and this can cause the control to suddenly jump. If Auto Restart and Retain State are activated in the Simulation Properties menu, continuous and smooth operation can be achieved, but the retrace of a `plot` block can sometimes cause small “hiccups” at the start of each cycle as the plot is re-drawn. One way to minimize these effects by the `plot` blocks is to enclose the plot in a compound block to hide it from the top level. Data can be recorded and viewed after the simulation is stopped.

**Note:** In VisSim 2.0, `slider` blocks suspend the simulation if operated during a run. In VisSim 3.0, real-time `sliders` prevent suspension.

Other ways to speed up the simulation include:

- Use 32-bit VisSim
- Use a PC with a faster clock rate
- Use a faster graphics card and/or add more video memory
- Convert all or some of your application blocks to DLLs
- Use VisSim/DSP options

## Step Rate Selection Considerations

The step rate must be selected to accommodate the bandwidth of the closed-loop design. Theoretically, the sampling rate should be at least twice the rate of the highest frequency in the desired closed-loop system. This is only practical, however, for applications where an interpolation (reconstruction) filter can be employed. For feedback applications, it's best to choose a rate at least 5 to 10 times faster. Consider also that the rolloff of the anti-aliasing filter must be such that sufficient attenuation is achieved at half the sampling frequency. If the sampling rate is increased, it opens up the measurement band to more ambient noise. The chosen step rate must be fast enough to provide information at the rate required, but not too fast to admit excessive measurement noise. For an extensive treatment of sample rate selection, refer to *Digital Control of Dynamic Systems* by Franklin and Powell.

When running applications in VisSim with the Run in Real-Time option selected, the speed of the system is paced by a real-time clock. This option should be selected when designing real-time controls. If the selected step rate is 1 ms or slower, VisSim uses the Windows system clock to pace the system. If a faster step rate is selected, VisSim reads real-time clocks provided by the data acquisition hardware to pace the system. After the clock is selected, a real-time test should be run, as described in the previous section, to test the uniformity of the clock. Step rates slower than 1 ms should be selected in 1 ms increments. If a step rate of 1.3 ms was selected, the real-time simulation would run, but possibly with erratic timing.

## Using transferFunction Blocks

The implementation of differential equations can be done by either constructing them with integrator blocks or by direct use of the `transferFunction` block. In VisSim, `transferFunction` block output is calculated using a matrix series expansion to calculate the state transition matrix. If integrators are used to realize the `transferFunction`, integration of the states is done using the integration algorithm selected in the Simulate/Simulation Setup dialog box. This causes the `transferFunction` method to be less sensitive to the chosen step rate. If the goal is to eventually transfer the design to embedded controls, it is best to start with integrators.

## Choice of Integration Method

If the control design is to be transferred to an embedded system, then it is best to avoid using the integration methods supplied by VisSim and instead write the controls as difference equations using `unitDelay` blocks. For control applications, the simplest, stable integration method to use is backward rectangular (Euler) integration.

$$y(n) = Tx(n) + y(n-1)$$

This method assures that any poles in the left half s-plane are mapped within the unit circle in the z-plane. Although VisSim supplies two Euler integration methods (Euler and backward Euler), neither is satisfactory for control applications. The Euler integration method is actually forward Euler, which could map poles outside the unit circle leading to instabilities. The backward Euler (stiff) method uses a matrix-relinearization of the data and is too slow for practical real-time applications.

For a quick start, however, the forward Euler method can be used to get a rough idea of the response. This should be followed up by replacing the continuous integrators with discrete integrators that implement straight backward rectangular integration.

# *Transferring the Control Design*

Once the control design is proven to meet its performance specifications by hardware-in-the loop simulation, the next step is to port the design into the embedded system. Development trends are strongly leaning towards a system that will provide automatic generation of code for any given platform, but there are still many issues that must be addressed before this can be flawlessly achieved.

## Selecting an Embedded Processor

As mentioned before, control design is often done directly in the embedded environment at the same time code is written. In this situation, selection of an appropriate processor is done beforehand, based on a best guess estimate of how much speed will be required by the algorithms. If misjudged, this could result in a more-expensive-than-necessary processor, or worse, a processor unable to handle the processing load. Guessing can lead to loss in engineering time, as well as substantial development tool investments. It is much more efficient and cost effective to design the controls, test their performance, and then based on more exacting needs of the algorithm, choose the appropriate processor.

To select the embedded processor, a common measurement between the process run in the PC environment and the one run in the embedded processor environment must be found. Factors that exist in the PC environment, such as system and application processing overhead, as well as higher number precision, must be taken into account.

Overhead processing in VisSim accounts for approximately 5% of the time slot, but can be accurately measured by comparing real-time with simulation time measurements. This measurement is easily made in VisSim by plotting the output of the `realTime` block against a `ramp` block. Note that this overhead includes the measurement operation itself, and to maintain this constant, the measurement calculations should be included with the simulation calculations.

Manufacturers of processors sometimes publish benchmark algorithms and their results. These same algorithms can be run in VisSim, and a performance ratio can be calculated to relate the two operating environments. Of course the benchmark should contain similar operations that will be run in the final control algorithm.

## Modular Programming

Good software practices universally recognize the importance of modularity to provide organization for ease of understanding, maintainability, and the ability to reuse existing processes or algorithms. VisSim compound blocks and `embed` blocks provide design modularity but also add to it a new dimension.

### Compound blocks

Compound blocks allow the design to be created using a bottom up or top down approach. For example, a problem can be approached by either building each compound block and then connecting them to form the complete system, or constructing the overall system connections with less detail, corralling components together as sub-modules and then adding detail within the compound blocks. Both approaches are valid; the choice depends on the application.

By using compound blocks, processes can be grouped and organized to improve understanding and reduce the likelihood of mistakes. Compound blocks provide a hierarchical structure to the design. Top level components show major connectivity; successive layers reveal local complexity. Hierarchical modeling is easy to build, easy to understand, and easy to maintain.

Unlike programming in C, using VisSim allows a designer to begin by piecing blocks together to create an entire process, and then grouping the elements into organized sub-processes with

compound blocks. Using this approach sometimes reveals system structures that perhaps would not have been so apparent had the simulation been written in a script language. The visuality of VisSim provides this distinct advantage over script languages.

### Embedded blocks

It is often the case that a sub-model is used repeatedly in many diagrams. Using VisSim's `embed` block, a copy of the sub-model can be embedded in any diagram. The copy has all the functionality of the original sub-model; however it cannot be edited. Only the original sub-model can be edited, and those changes are automatically propagated to all the diagrams that use the embedded sub-model. Use of `embed` blocks provide greater control and ease over the revision process.

## Tracking Product Development

To track product development, a revision history can be attached to any block diagram created in VisSim. This is particularly important for large projects that span multiple engineering groups. Author names and comments can be included, as well as longer, more descriptive diagram names. Statistics about the block diagram, such as its size and last date of modification are also included.

## Working with Variables

Although it is possible in VisSim to build complete systems using only wire connections between blocks, `variable` blocks should be used to simplify and clarify connections where possible. In VisSim, a variable name can be up to 22 characters in length, making the use of meaningful variable names easy.

VisSim contains three types of variables:

- Built-in variables designated by a "$" prefix
- Local variables designated by a ":" prefix
- Global variables (all other names)

For discrete control designs that require one or more clocks, it is best to synchronize all the clocks from the basic built-in variable `$timeStep`. Clocks can be built from the `pulse generator` block, with a second input that allows external input of the pulse interval. The first input is pulse delay which is normally set to zero.

Local variables are only passed at the level at which they are defined, meaning they are not seen outside compound blocks in which they are defined, nor within inner compound blocks. When designing functional blocks, it is best to use local variables to maintain modularity. Global variables are passed throughout the entire diagram, recognized at any compound block level.

Cut and paste operations of blocks that contain variables with input definitions can cause multiple references. These are handled by VisSim by redirecting the variable name to an unused memory location. Thus if the variable `Valve Position` were to be copied as a multiple reference, VisSim would rename it as `Valve Position@236` for example. It is therefore important to review the variable list especially after cut and paste operations. This can be done easily using the variable find feature and searching the list for "@" symbols in the variable names. If local variables are used in all functional blocks, redirection should not occur.

## Discretizing the Design

VisSim uses floating point numbers and all block operations are done in floating point precision. Most embedded applications using inexpensive processors require the design to be converted and run with integer processing. Beyond the fact that signals and constants will likely have to be scaled to fit the processor word size, problems such as overflow and truncation error, and quantization noise may degrade performance from the initial floating point design.

If these problems are an issue, the system should be rescaled in VisSim, all filters and integrators should be built as discrete equivalents using delays, and quantized signals should be used where appropriate. By doing so, degradation can be assessed, corrections made before the design is transferred to software engineering.

In VisSim, quantization blocks can be used to simulate a quantized signal and can provide a measure of how quantization effects accuracy and performance. VisSim 3.0 provides the capability to select number types in either floating point or fixed length integer.

In most embedded applications with appreciable bandwidth and controller complexity, floating point is prohibitive unless the processor handles floating point in hardware and is very fast. High volume, consumer and commercial products usually cannot use such a processor because of cost limitations. The mandate for a more inexpensive processor can dictate the requirement to implement the design in integer. Thus once a design is proven in floating point, it must be discretized unless originally implemented in integer. In VisSim this means replacing `transferFunction` and `integrator` blocks with equivalent functions using `unitDelay` blocks, and transcendentals with approximations as needed. Upcoming versions of VisSim promise to provide data type specification which will extend its capabilities in the design verification process before writing code.

Although `transferFunction` blocks in VisSim can be designed as discrete as well as continuous, if the design is to be transferred by means of written documents or specifications, it is better to implement directly with `unitDelay` blocks. The `transferFunction` blocks are specified by coefficients in the Z-domain, but designing with `unitDelays` is directly reducible to difference equations. For most programmers it is easier to interpret diagrams written with `unitDelay` blocks; z-transforms typically take some explaining.

Once the design is discretized, it must be communicated to the programmer for coding. This can be done in a number of ways. Flow charts can easily be written from well organized VisSim diagrams. Or step-by-step lists of equations and pseudo-instructions can be provided, but with little instruction it is relatively easy for programmers to read directly from the block diagrams. Using the block diagrams themselves eliminates having to produce an intermediate document and reduces the likelihood of mistakes.

## The C-Code Generator for Real-Time Control Applications

VisSim/C-Code was originally provided to allow users to write their own routines in C, create a corresponding dynamic link library (DLL), and use these routines as custom blocks in the VisSim environment. This provided an increase in speed of operation and a means to integrate previously developed software routines with VisSim. Recent developments in VisSim/C-Code are now providing new applications including automatic DLL source generation and direct portability to the TMS320C32 DSP. These capabilities open up new real-time applications for VisSim.

# References

Borrello, M. A. 1991. Modeling and simulation of pressure regulated control systems for ventilation of the lung. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 13, no. 5:2178-79.

Borrello, M. A. and D. W. Guillaume. 1991. Simulating gas flow through the exhalation leg of a respirator's patient circuit. *Journal of Biomedical Engineering*. 13:77-82.

Campbell, D. and J. Brown. 1963. The electrical analog of the lung. *British Journal of Anaesthesiology*. 35:684-93.

Franklin, C. F. and J. D. Powell. 1981. *Digital Control of Dynamic Systems.* Massachusetts: Addison Wesley Publishing.

Jacquez, J. A. 1979. *Respiratory Physiology.* Hemisphere Publishing Corporation, *et al*.

Oppenheim, A.V. and R.W. Schafer. 1975.*Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall.

Otis, A. B., C. B. McKerrow, R. A. Bartlett, J. Mead, M. B. McIlroy, N. J. Selverstone, E.P. Radford, Jr. 1956. Mechanical factors in the distribution of pulmonary ventilation. *Journal of Applied Physiology*. 8:427-42.