USF Neural Simulator

Revision History

Revision	Date	Author(s)	Description
1.0	13-Dec-2019	DS	Created from sim.tex. Initial release.
1.1	10-Jun-2020	DS	Add Afferent Signal Fiber Population.
1.2	26-aug-2020	DS	Add Hebbian Learning Syanpse Support.

Copyright (C) 2003,2007,2011,2019,2020 Bruce G. Lindsey, Kendall F. Morris

`Report bugs to <dshuman@usf.edu>`

Table of Contents

Chapter 1

Overview

The USF Neural Simulator software was developed in the laboratory of B. G. Lindsey at the University of South Florida by U.J. Balis, Kendall Morris and others using Fortran.[1] It started out as an implementation of Ronald J. MacGregor's SYSTM11 simulator as documented in his 1987 book *Neural and Brain Modeling*. This version required creating text files in an editor and running the simulator from a command line window. The simulation produced text files, some of which could be plotted by a plotting program.

A subsequent version was created from the Fortran code using the K&R C language, X-windows and Motif. This provided a Graphical User Interface (GUI). This version consisted of three programs, *newsned*, *sim*, and *snedskop* and several utility programs. The ability to visually create a model and display results directly was a major improvement. However, since the graphical model was not very interactive and low display resolutions of the time made adding new controls difficult, it was still tedious to create models. Editing a text file was often easier than using the GUI, so a lot of the workflow required a command line skill set and detailed knowledge about the internals of the text files.

In 2019, the source code was forked and a new simulator software package was created, collectively referred to as usfsim. Usfsim re-uses the computational software, but new GUI applications have been created using the Qt C++ cross-platform development framework. The major goals of this work was to create an interactive graphical editor with features common to many drawing programs, create an extensible body of code that provided a user interface that would be familiar to today's users, and to shield the user from the command line and file internals.

The programs have help menus and there are pop-up tooltips that explain what the controls do. While it is tempting to include annotated screenshots, experience shows these quickly get out of date and tend to create rather than prevent confusion. The best way to learn how to run the programs is to run them.

This document was derived from the previous user's manual. There are a few places in the

[1]Development of the program was supported by NIH grants NS19814 and NS46062 as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program.

text that follows that still assumes the user would be editing text files from a command line, but much of the original manual text that assumed this has been removed. The manual for the previous version, *sim.pdf*, is included in this package.

The package has three major applications. The *simbuild* program is used to create simulator models using a graphical editor and to launch and control a simulation. The *simrun* program is the original non-GUI simulation engine with some modifications. While it supports the previous version's command line interface, the use of this is deprecated. The *simviewer* program displays plots of the results of a simulation run. The user can interact with the plots.

The package also contains many of the utilities that were part of the *newsned* package. Most of these only run in Linux.

The usfsim software can read earlier models created by *simbuild*, but *newsnd* cannot read models created by *simbuild*.

Software development is done on Linux systems running the Debian Linux distribution using the GNU compiler collection, the Qt cross-platform framework, and the MXE cross-development environment. This produces software packages that run in Linux and in Windows.

The previous version of the simulator used text files to communicate information between the various programs. For performance reasons and to stage the simulator for future development in distributed environments, the usfsim programs communicate with each other using network interfaces. As part of the launching process each program establishes a connection with any other program it needs to communicate with. With the exception of the wave files, the files are still written, but the content of those files are sent directly to the receiving program.

There are two options for viewing the simulator results.

The first is to view the results in real-time as they are displayed by the *simviewer* program. The results can be sent directly to simviewer from simrun or saved to wave files for later viewing.

The second is an ".edt" or ".bdt" or ".smr" format file that lists the spike times of selected cells and the integrated activity of a population. The difference between ".bdt" files and ".edt" is ".bdt" files use a step time of 0.5 milliseconds and ".edt" files use a step time of 0.1 milliseconds. Custom step sizes are saved in the ".bdt" format. This is selected by clicking one the Edit Globals button and selecting the simulation step size. The ".edt" and ".bdt" files can displayed with the *scope* application, which is included in the Linux version of the usfsim package. The ".smr" file is a format that can be read with the Cambridge Electronic Design Limited product, Spike2, which runs on Windows. These files are usually viewed after the simulation completes.

NOTE: For the rest of this document, references to ".bdt" should assume it also means ".edt" unless otherwise stated.

Chapter 2

Installation

2.1 Installing on Windows

The Windows version contains a .zip file of the general form *usfsim_win_[version number].zip* and a text file *README_FOR_WINDOWS*. That text file contains installation instructions. Unzip the zip file into a directory of your choice and then create a desktop shortcut to *simbuild.exe* and optionally *simviewer.exe*. Be sure to set the "Start In" directory to be the directory where *simbuild.exe* is installed. The *simbuild.exe* program needs to know where to find *simrun.exe* and *simviewer.exe* and assumes they are in the same directory as it is..

For users with several working directories, an alternative to this is to add the location of the programs to the user's PATH environment variable. There are many web pages that explain how to do this.

2.2 Installing on Linux

The Linux version is a Debian package of the general form *usfsim_[version]_amd64.deb*. It can be installed using any Debian package installer. For example:

dpkg -i usfsim_[version]_amd64.deb

You need to have admin privileges.

The source for the package is available as a .tar.gz file.

Executables are installed in `/usr/local/bin`. Documentation is installed in `/usr/local/share/doc/usfsim`. Sample models and the Windows zip file are installed in `/usr/local/share/usfsim`.

After installation, you can invoke *simbuild* in a command line window by typing:

```
  simbuild
```

The user can also create shortcuts in Linux. The steps to do this depends on the desktop manager.

Simbuild will launch *simrun* and *simviewer* for you when starting a simulation. *Simviewer* can be run as a stand-alone application for viewing the output of previous simulations. While *simrun* can also be run as a stand-alone application, it is difficult to use. This usage is deprecated. See *sim.pdf* for more information about running *simrun* as a stand-alone application.

Chapter 3

Running the Simulator

One standard work-flow is to create one or more working directories and then run *simbuild* to create models using its graphical editor functions. Fiber and cell populations and axon/synapse connections are added to create circuits. Model parameters are specified for the populations and connectors. The simulation is started using the Open Launcher button. This opens a second window where the user can start and interact with the simulation. The *simviewer* program is also started. It will display simulation results.

If you use existing models created by *newsned*, it is suggested that you create a new working directory and copy the existing models to it from a *newsned* working directory. The *simbuild* program will upgrade the files to the new file version when you save them. Files created by *newsned* can be read by *simbuild*, the opposite is not the case. If you wish to continue to use *newsned*, you should keep the work environments separate.

The model geometry and parameters are saved in ".snd" files. As part of the launch operation, a ".sim" file is created that contains the information that *simrun* needs to perform the simulation computations. An ".ols" file is created that determines what *simviewer* displays and what is save to ".bdt" and ".smr" files. In the past, users would create ".sim" and ".ols" files by hand, or edit ones produced by *newsned*. The previous user interface had limited interactivity and it was often easier to simply edit the ".sim" file with a text editor to tweak a parameter. This option remains available and some of what follows is original text from *sim.pdf* that assumes the user is more likely to do things "by hand" than to use the interactive applications. New text in this document assumes the user will always use the GUI programs and leave the maintenance of the ".snd", ".sim", and ".ols" files to them.

3.1 The *simbuild* Network Editor

3.1.1 Overview of *simbuild*

The *simbuild* program provides two major functions, creating/editing models and running a simulation using a model.

The *simbuild* program has menus, a toolbar with common operations displayed as buttons with icons, a panel on the left side that displays interactive information, and a drawing panel on the right where the visual representation of the model is created and edited. A control called a “splitter”, which appears as a vertical thick dark yellow line, lets the user resize the areas.

Many of the selections the user makes, such as window location and size, are saved in a settings file. These are loaded when the programs are first launched.

3.1.2 Tutorial

Here is a short tutorial on how to create a simple model.

1. Create a working directory.
2. Run *simbuild*.
3. In the Options menu, select One Shot Add (this is to make it behave like *newsned*).
4. Click Add Fiber Population Button.
5. Click in an empty location in the drawing window. The fiber population will be added. It is displayed as a square.
6. Click Add Cell Population.
7. Click in an empty location in the drawing window. The cell population will be added. It is displayed as a circle.
8. Click Add Axon/Synapse.
9. Move the mouse to be over the square. A dotted square should appear around the solid square.
10. Click anywhere inside the dotted square.
11. Move mouse to a location between square and circle objects.
12. Click again. A line segment will appear.
13. Move to a new location, click again.
14. Another line segment will appear.
15. Move the mouse to be inside the circle. A dotted circle will appear.
16. Click again.
17. This completes adding the axon/synapse connector. (A connector can have up to 15 segments.)
18. Clicking on a single square, circle, or a connector will display the parameters for the object to the left of the drawing window.
19. Click on the File menu and select Save .snd file. Enter a file name and save it.

20. Click on the Open Launcher button.
21. The Simulator Launch Control window will appear on top of the *simbuild* window. Move it out of the way.
22. Select the fiber population. In the parameter area, click on Send to Launcher BDT table. This adds a row in the Launcher's BDT table. The behavior of this member of the fiber population will be saved to a ".bdt" file.
23. Select the cell population. In the parameter area, click on Send to Launcher BDT table.
24. Click on Send to Launcher Simview Table. This adds a row to the Launcher's simviewer table. The behavior of this member of the cell population will be plotted in the *simviewer* application.
25. Click on Launch Simulator.

3.1.3 Simbuild User Interface

3.1.3.1 File Menu

New
Start a new model.

Load .snd file
Load an existing model.

Save .snd file
Save the current existing model.

Save .sim file
This is an obsolete command that will be rarely used. A ".sim" file and a ".ols" file are the inputs to simrun. When launching a simulation, fixed file names, such as spawn0.sim and spawn0.ols, are used. These are over-written each time a simulation is launched. This item allows the user to specify the base name for the ".sim" and ".ols" files that will not be over-written. The user then has to create or modify a script0.txt file using these names. The script file will be the input to *simrun*.

Update Change Log
If a model has been loaded, selecting this item updates the change log file. Any changes will be added to the change log file.

Open Shell
A command line window can be started using the "Open Shell" selection. Some programs that are part of the simulator package are command line programs. This is a convenience item.

3.1.3.2 Find Menu

Find
Selecting this opens a list of all of the cell and fiber populations in the current model.

Selecting a row from this list scrolls the model to ensure that the population is visible and then displays the parameters for that population.

3.1.3.3 Options Menu

The options menu has items that control the behavior of the program. These include:

One Shot Add
If this is selected, exit Add mode after adding one population to the drawing. Otherwise, each click in the drawing area adds another population of the current type.

Do Not Warn About Unsaved Changes
If you intend to make a lot of changes to a model but do not intend to save them, select this to stop the program from nagging you about the changes.

Do Not Prompt For Change Log
Similarly, if you do not intended to update the change log, select this to silence the nagging.

Include .snd File Name As Part Of Change Log Name
If you have several models in a single directory, selecting this creates a unique change log file for each model. Otherwise, all changes go into a single file.

Launch Window Always On Top Of Simbuild
The Launch Window is implemented as a dialog box. The convention is that a dialog box is always in front of the parent window. The user can move it out of the way if they want to access the *simbuild* window. Selecting this detaches the Launch Window from the parent so it behaves as an independent window. If the user clicks on the *simbuild* window, it will be moved to the foreground. Clicking on the Open Launcher button will bring the Launch Window to the foreground.

3.1.3.4 Groups Menu

One or more populations can be selected and assigned to a group. The group can then be selected and, for example, moved as a unit. The intended use of this is to create subsets of populations that can be used as building blocks in other models. The group can be saved to a ".sndlib" file and then imported into a different model.

Assign To Group
Assign all of the select populations to a group.

Select Group
If a member of a group is already selected, selecting this item selects all of the members of the group. If no populations are selected, the group will be selected by first selecting this item, then clicking on any member of the group.

Export Group
The currently selected group will be exported to a .sndlib file.

Import Group
After the user selects an existing .sndlib file from an Open File dialog, it will be imported into the current model. The user will be prompted to click where in the drawing to draw the imported group.

Remove from group
All selected members of a group are removed from the group.

3.1.3.5 Help Menu

Help
Opens a dialog box with several pages of help.

View Manual
If Linux or Windows has been configured to use a specific program to open .pdf files (known as "setting file associations"), this program will be launched and what you are reading now, *usfsim.pdf* will be opened.

View Change Log
If Linux or Windows has been configured to use a specific program to open .txt files, this program will be launched and the current change log file will be opened.

View Release Notes
If Linux or Windows has been configured to use a specific program to open .pdf files, the current *ReleaseNotes.pdf* file will be opened. This contains the cumulative changes of each release of the simulator package to date.

View Older Manual
If Linux or Windows has been configured to use a specific program to open .pdf files, this program will be launched and the original newsned user's manual, *sim.pdf*, will be opened. It contains some information that has been removed from the updated user's manual, but which may be of historical interest.

3.1.3.6 Graphical Editing

Creating a model uses operations and conventions that are common to many drawing programs. Items can be selected, moved around, deleted, all using a point-and-click interface. In some cases, using the SHIFT or CTRL keys modify what mouse motion and buttons do. The *simbuild* Help system provides information that will be more up-to-date than this document, so the user should consult it. Many of the controls and menu items have popup tooltips that provide information about the control or menu item. Move the mouse to hover for a short time over a control or menu item and the tooltip popup will open.

Creating a model has two modes.

The first is Edit mode, which is used to make changes to the model. This is the default.

The second is Add mode, where new objects are added to the model.

EDIT MODE

When in edit mode, the mouse pointer shape will be an arrow.

Each population and connector has a “hot zone”, which is the object itself and a region outside the object. The region is indicated by a dotted circle, square, or rectangles. Clicking anywhere inside a zone selects the object

Left click inside a hot zone select an object.

Left click anywhere not over an object will deselect everything.

Left click outside any object and moving the mouse while holding the button down creates a resizable selection rectangle. Anything inside the rectangle will be selected.

CTRL + a series of Left clicks inside multiple objects’ hot zones selects a set of objects. This is more precise than using a selection rectangle. This is a toggle, so you can select a large set of objects, then CTRL + left click to deselect objects.

Clicking the left mouse button down while over a selected object and moving mouse moves all of the selected objects.

Since connector endpoints are behind the fibers and cells, you cannot grab the endpoints. Click on the Lines on Top button. This brings them into the foreground. If you hold down the CTRL key and scroll the mouse wheel, you can zoom in and out. You can grab the end points and move them around. You cannot move a connector from one population to another.

Line segments can be moved by selecting a line, then clicking in the middle of a segment. Clicking at the end of a segment selects that point. The end of the segment can be moved by holding down the left mouse button.

If you have a line with several segments and you click in the middle of one of the segments, the entire line is selected and you can move it as a unit. If you click on the connection point (think of this as the “elbow”) between two segments, you can move just the common point between the two segments. It is kind of hard to hit this region, which is where zooming in comes in handy. It is also possible if the joining angle is very acute, the hit region can be hard to click on, it is just a few screen pixels.

Mouse Wheel up and down scrolls horizontally.

SHIFT + Mouse Wheel up and down scrolls vertically.

SHIFT + left mouse click not over an object + mouse move scrolls the display.

CTRL + Mouse Wheel up and down zooms in and out centered on the mouse position.

ADD MODE

When in add mode, the mouse pointer shape will be a +.

To add objects, click on one of the Add selector buttons in the toolbar at the top of the window.

Click Add Fiber Population Mode, then left mouse click in the drawing area adds a Fiber population.

Click Add Cell Population Mode, then left mouse click in the drawing area adds a Cell population.

Add Mode can be exited by clicking on the currently selected button to toggle its state, pressing spacebar, or press ESC. If the Options→One Shot Add item is selected, Add mode will be exited after adding one object. Otherwise, the next click in the drawing area will add another instance of the population type.

Add Axon/Synapse creates a series of line segments connecting populations. The source (start) population can be either Fiber or Cell. Only cell populations can be the target (end) of a segment

Add Axon/Synapse mode + first mouse click inside a population hot zone begins the connector.

Add Axon/Synapse mode + next series of mouse clicks not inside a population hot zone adds a segment.

Add Axon/Synapse after first click within a population hot zone terminates the connector.

A connector can have up to 15 segments.

SETTING PARAMETERS

When a fiber, cell, or axon/synapse object is selected, the parameter page for that object is shown in the left panel. Different types have different sets of parameters.

The newsned GUI program was created at a time where monitors were low resolution compared with the current technology. It seems that it may have been assumed that monitors were 640 x 480. With this limited screen space, it was very difficult to add new controls to the user interface. Over time, additions were made by using specific values to indicate a type of cell. For example, in newsned the user indicated a cell population was a Hybrid IF ("burster") by setting the Potassium Conductance Time to 0. This changed the labels on the input fields, but they remained the same controls. This made using newsned very confusing, which is why it was often easier to edit a ".snd" or .sim file by hand.

The simbuild program provides a list of radio buttons for fibers and cells to explicitly select the type of population. Parameter fields appropriate for each type are provided. As detailed in the Lung Model section (Motor Populations, page 40), the lung model objects are identified by set of naming conventions that must be part of the comment field. When you select a lung population type, *Simbuild* will try to enforce the rules, but if using more than one object of the same type, it is up to the user to use the number conventions and to use the correct order when creating the associated equations (see Recruitment Equations, page 40).

3.1.3.7 Edit Globals

Clicking on this button displays a list of global options. This is grab-bag collection of unrelated parameters. These are documented in Global Parameters, page 34.

In addition, it allows you to set the '`Phrenic recruitment equation`' and the '`Lumbar recruitment equation`'. See Recruitment Equations, page 40.

3.1.3.8 Edit Synapse Types

Use the Edit Synapse Types button on the toolbar to add new synapse types. These appear in the table beneath the Synapse Parameters page. Use the Add Synapse to create a standard type. Use Add Learn to add a Hebbian learning type, documented in Hebb (1949),

The parameters are documented in Synapse Type Parameters, page 31.

Each standard synapse can optionally have a presynaptic and postsynaptic modifier. These can be thought of as children of a normal synaptic type. These change the behavior of the normal synapse types. The presynaptic type modifies conductance changes before they reach the normal synapse. The postsynaptic type modifies the membrane conductance due to the normal synapse.

A strength of 1 in a presynaptic or postsynaptic synapse has no effect on the normal synapse. A strength of less than 1 multiplies the normal synapse conductance, and if the strength is greater than 1, the amount by which it is greater than 1 is added to the normal synapse conductance.

A presynaptic or postsynaptic connection to a target cell simultaneously affects all the normal connections of the parent normal type to that target cell. This means you must add both the normal synaptic type and one or both of the modifying types (that is, draw two or three lines.)

The Learn synapse models Hebbian learning. This is based on the LRSYS30 program as documented in MacGregor (1987). Standard synapses use a fixed strength. If a learning synapse's target cell fires a short time after receiving a firing event from a sending fiber or cell, the synapse strength is slightly increased up to a maximum value. With the right choice of parameters, the learning can also be inhibitory. That is, the synapse strength is decreased down to a minimum value.

3.1.4 Simulator Launcher Control Window

The Simulator Launcher Control Window is a child window of simbuild. It is used to set up, launch, and manage a simulation. Information is sent from *simbuild* to the Launcher Window. Use the Open Launcher button on the toolbar to display the window. It has to be open before sending it information.

3.1.4.1 Create ".bdt" and ".smr" files

The table in the left side of the Launcher Control window contains a list of cell and fiber population members that can optionally be saved to a ".bdt" file and optionally to an ".smr" file. To add a row to this table, select a cell or fiber population in the model and click on the Send to Launcher BDT Table button on the parameter page. By default, a random population member is selected. The user can edit the member number. The member numbers should be unique for each population. Some of the down-stream software assumes there are not duplicates. Checkboxes determine what will be generated.

3.1.4.2 Create Analog Entries

Analog entries are set up in the middle panel. If '`Create analog entries`' checkbox is selected, the values in this panel determine the analog data that will be included in the ".bdt" and ".smr" files. The ID field of the analog entries is

$$AnalogID * 4096 + 2048 + analogvalue.$$

The value of the '`Analog ID`' does not matter unless you will be merging the ".bdt" file with others, except that the value has to be greater than 0 and less than 25. If you are merging you probably want to use different '`Analog ID`' values.

The analog value is the overall firing rate for a specified '`Cell Population`'.

If '`Create analog entries`' is selected, a power spectrum of the analog signal will be displayed at the end of the simulation. see Section 3.3.2: Power Spectrum, page 37.

The '`Rate`' entry determines how often analog data will be written to the ".bdt" file. An analog entry will be made every $\frac{1000}{\texttt{rate}}$ milliseconds of sample time. Only the integer part of '`Rate`' is read, and the result of the division rounded down to an integer. If you enter a number greater than 1000, no analog data will be written.

The analog value is updated each time it is written, as follows:

$$\texttt{analog} = \texttt{analog} * \texttt{e}^{\frac{-\left(\frac{1000}{\texttt{Rate}}\right)}{\texttt{Integration Time}}} + (\texttt{spikes} * \texttt{Scaling Factor})$$

The analog value is an integer. The quantity '`spikes`' is the number of action potentials that have occurred in the specified population since the last time the analog data was written. The '`Integration Time`' and '`Scaling Factor`' should be chosen so that the analog value does not exceed 2047. Note that the '`Integration Time`' is in milliseconds.

To generate a power spectrum, the '`Analog ID`' must be set to 1, the '`Cell Population #`' should be set to the number of the phrenic population, '`Rate`' should be set to 1000, '`Integration Time K`' should be set to 1, and '`Scaling Factor`' should be set to 1. see Section 3.3.2: Power Spectrum, page 37.

3.1.4.3 Displaying Simulation Results

The table in the right side of the Launcher Control window contains a list of items that will be plotted in a /progsimviewer window. There are two ways to add a row to this table. The first is to select a cell or fiber population in the *simbuild* window and click on the Send to Launcher Simview Table. By default, a random population member is selected. The user can edit the member number. The member numbers can be unique or can be the same.

The *newsned* program did not plot fiber events. This ability has been added to *simbuild*. The events are plotted as vertical lines, except for the afferent fiber type. The plots for this are the events, the original signal, or both.

The plot results can be sent directly to *simviewer* or can be written to wave files. Using the direct connection is faster but not permanent. If you save the plot results to wave files, you can use *simviewer* as a stand-alone program to display them. Note that the unique part of each wave file name is four digits, 0000 to 9999. When a wave file with 9999 as part of the file name is written, *simrun* wraps around the name counter so the next file name will contain 0000. This means older files will be over-written with newer files. This means the entire output of a long simulation run will not be available for later viewing.

The rows can represent different types of items to plot. There are cases where not all of the cells in a row are used. There are cases where a cell must have a value, but units of the value will vary. The text in the headers of the two right-most columns indicates the type of a value for the selected cell. If there is no header, that cell does not require a value.

The type of plot is selected from a drop-down list in the Plot Variable column. If the cell type is not a Hybrid IF type, the choices are:

Membrane Potential
Potassium Conductance
Threshold
Instantaneous Pop Activity
Use Bin Width and Scale ==>

Hybrid IF types get a different list, like so:

Membrane Potential
Hybrid IF H Value for Burster Populations
Threshold
Instantaneous Pop Activity
Use Bin Width and Scale ==>

If the variable being plotted is “Use Bin Width and Scale ==>”, the value being plotted is not specific to any one cell in the population. The bin width and scale values are entered in the cells to the right of the Plot Variable cell. The vertical size of the plot in *simviewer* will be reduced by the indicated factor. The displayed spikes/second will still be correct.

If the fiber population is an afferent signal source, this list is used:

Events
Signal
Events and Signal

If the lung model is used, since there may be more than one population that is used in the lung model and since the lung plots use all of the populations in calculations, there is no obvious graphical item to use to add rows to the table. Use the Add Lung Plots button to add a row. The drop-down list will contain the type of lung model information that will be plotted. The Population and Member columns do not need values. See The Lung Model, page 40 section for more information.

The rows can be sorted by clicking on a header.

The bdt table entries have the convention that cell populations are entered first, then fiber populations. The order of these will determine the order of the rows when viewer ".bdt" and ".smr" files.

The order in the plot table determines the order they are displayed in *simviewer*. These can also be sorted by clicking on a header.

You can manually sort both tables by left clicking on the left-most cell, hold the button down, and drag and drop the row to a new location. If you place fibers before cells in the bdt table, the order will be different when viewing the ".bdt" and ".smr" files.

3.1.4.4 Launch Slots

It is possible to run multiple invocations of the simulator in parallel. Each invocation is given its own "launch number". The sim file for each invocation is named "spawnN.sim". The ".bdt" and ".smr" files are similarly named. If Save Plot Data To Files is selected, the wave files are named "waveNN.nnnn", where "N" represents the launch number (two digits in the wave file name). Your choices about what data to output are provided to the simulator in a file named "scriptN.txt". The "nnnn" in the wave file name represents a sequence number. No more than 100 time steps are written to each wave file, and then the sequence number is incremented and a new file is started. The sequence number starts with 0000, and the simulation will end after it gets to 9999.

The launch number that will be used is indicated in a control in the upper left of the window, where there are up and down arrow buttons to change it. The value can also be entered directly from the keyboard.

After starting the first simulation in the usual way, click the Copy Values To Next Launch Slot button in order to copy the fiber and cell lists on that panel to the next launch number. Then click the up arrow next to the launch # in the upper left to increment the launch number. Then make any changes you wish to the model for the next simulation, and click Launch Simulator. Then use Copy Values To Next Launch Slot and repeat, up to 20 times. Launching a new simulation starts a new instance of *simrun* and *simviewer*.

This feature was added at a time when computer and monitor resources were limited. Simulation run-times could be hours or even days and running multiple instances of the entire simulator package might not have been possible. Managing them on a single low-res monitor screen could be a challenge. The typical use was to start a set of simulations and come back a few days later to check on the results.

In addition, there are couple of caveats to launches. While each launch gets its own set of parameters, the lung model requires information from the ".snd" file. There may be cases where changes to the .snd file, such as altering the network, population names, and other changes, may have unexpected consequences. Also, changes to parameters are not automatically saved to the current ".snd" file. If persistence and repeatability is desired, it may be useful to launch several instances of *simbuild* and have unique ".snd" files.

3.1.4.5 Simulator Launcher Control

A simulation is started by clicking on the Launch Simulator button. This will start the *simrun* program and if plot results are generated, the *simviewer* program will also start. Each launch number starts a new instance if *simrun* and *simviewer*.

The simulation can be paused and resumed. If you want it to pause after a specific amount of time, the Auto Pause control can be used to select when to pause.

The files that are associated with a launch number are written to the directory you were in when you started *simbuild*.

It is possible to update some parameters *simrun* is using during a simulation run. First, pause the simulation, make changes to the parameters, and then click on the Mid-Run Update button. The new model will be sent to *simrun* that is running the currently displayed launch number, and the simulation continues with the new parameter values.

There are limitations to what can be changed for a mid-run update. The *simrun* program looks for changes in cell and fiber population parameters or and axon/synapse parameters. Changes to the network, such as adding an axon/connector, changing its type, changing a cell type, or changing the number of cells in a population or number of terminals, will not be seen. Changes to the reporting options will also not be seen.

Note this is different from using launches. In the launch case, a new copy of *simrun* is started, which will build the network using the current network configuration. Of course, you could use both features to start a several sets of launches and then do a mid-run update to all of them. Keep track of the all of that could be a challenge.

The Close Window button hides the Launcher Window. This does not stop any simulations in progress. The simulations will continue even after you quit *simbuild*.

Selecting the Create Condi File checkbox causes *simrun* to write a file named "condi_[launch number].csv" before starting the simulation. The file contains a cell-by-cell list of all the connections in the network being simulated. The file is in comma-separated-value format for import to a spreadsheet. Each line of the file has the source population, source cell, target population, target cell, number of terminals, and synapse type for a connection. It also causes a file “condi_mean_sdev_[launch number].csv” which contains some statics about the simulation.

Selecting the Baby Lung checkbox tells *simrun* to use a different set of parameters for the lung model. There is not much documentation in the code or documents that explain what this is.

3.2 Model Parameters

The model geometry and parameters can be specified by using *simbuild*.

In earlier versions, users edited the “.snd” and “.sim” files. There are some non-obvious relationships between the entries in these files and it is easy to create a broken file. It it impossible to select values for some entries without the source code.

The labels used for the parameters in the “.snd” file are different in *simbuild* than they are in the “.sim” files. In the following sections, each parameter is described and the labels used in *simbuild* and in the “.sim” files, and the variable names used in MacGregor (1987), and the symbols used in Breen et al. (2003), are given in the following format:

SIMBUILD:	label used in *simbuild*
SIM:	label used in the “.sim” file
BREEN:	label used in Breen et al. (2003)
VAR:	variable name used in MacGregor (1987)
UNIT:	unit of measure for the parameter
TYP:	typical values for the parameter

If a particular entry does not appear for a particular parameter, that parameter is not used in that place.

Several of the parameter descriptions refer to variables that decay exponentially toward a value, with a specified time constant. That means that the variable is governed by an equation of the form:

$$\mathtt{V(t)} = \mathtt{A} + (\mathtt{V_0} - \mathtt{A}) * \mathtt{e}^{\frac{-\mathtt{t}}{\mathtt{t0}}}$$

where `t0` is the time constant and `A` is the value toward which the variable is decaying. ($\mathtt{V_0}$ is the value of the variable at `t = 0`).

3.2.1 MacGregor Cell Parameters

SIMBUILD:	Accommodation Time
SIM:	DCTH
VAR:	TTH
UNIT:	milliseconds
TYP:	20-25

The firing threshold moves up and down in proportion to membrane potential changes, but it doesn't move immediately to the new value. Instead, it approaches it exponentially. This parameter is the time constant of the exponential decay toward the new value, except for SIM, where it is $\mathtt{e}^{\frac{-\mathtt{step}}{\mathtt{TTH}}}$.

SIMBUILD:	Potassium Conductance Time
SIM:	TGK
VAR:	TGK
UNIT:	milliseconds
TYP:	3-10

In the absence of spikes, the potassium conductance decays exponentially toward zero with the time constant specified by this parameter. This controls the refractory period. During a spike, the potassium conductance decays exponentially toward a specified value (see [B], page 20), with this same time constant.

SIMBUILD:	Membrane Time Constant
SIM:	TMEM
VAR:	TMEM
UNIT:	milliseconds
TYP:	5-11

In the absence of potassium or synaptic conductances, the membrane potential will decay exponentially toward its resting potential (which is taken as 0 in this model) with the time constant specified by this parameter.

SIMBUILD:	Resting Threshold
SIM:	Th0
VAR:	THO
UNIT:	millivolts
TYP:	10-20

When the membrane potential has been at its resting value so that the firing threshold is not affected by accommodation, the firing threshold will have the value specified by this parameter, relative to the resting membrane potential.

SIMBUILD:	SD of Threshold if NZ
SIM:	[unused]
VAR:	[unused]
UNIT: n/a	
TYP:	

If non-zero, the value of Th0 will be distributed around Th0 with a normal distribution with a standard deviation Th0_sd.

SIMBUILD:	K conductance change with AP
SIM:	B
VAR:	B
UNIT:	dimensionless
TYP:	25-27

During an action potential (which has a duration of one time step), the potassium conductance will decay exponentially toward this value. This value is in units of the resting membrane conductance. In other words, a value of 2 means twice the resting membrane conductance.

SIMBUILD:	Accommodation Parameter
SIM:	MGC
VAR:	C
UNIT:	dimensionless
TYP:	.04-.85

As the membrane potential rises above its resting value (taken to be 0 in this model), the firing threshold will eventually rise in proportion from its resting value. This parameter is the proportionality constant. For example, if this parameter is .5, then when the membrane potential rises 2 mV above its resting potential, the firing threshold will rise exponentially toward 1 mV above its resting value, with a specified time constant (see TTH, page 19).

SIMBUILD:	Population Size
SIM:	cell_count
VAR:	N
UNIT:	count
TYP:	1-1000

All the cells in a population have the same values of the other parameters specified in this section, but each cell in the population can have a different membrane potential and firing pattern because it can have a different firing pattern at its synapses than the other cells in the population, so each cell is simulated separately. This parameter specifies how may cells in the population are simulated.

SIMBUILD:	DC Injected Current
SIM:	GE0
VAR:	SC
UNIT:	millivolts
TYP:	0-17

An injected current will raise the membrane potential by an amount that is inversely proportional to the membrane conductance. Instead of being specified directly as a current, the "DC Injected Current" is specified in terms of the effect it has on the membrane potential. This parameter is specified in millivolts, and it is interpreted as the current that is required to raise the membrane potential by the specified number of millivolts when the membrane conductance has its resting value. The effect on the membrane potential at other membrane conductances will be inversely proportional to the conductance.

SIMBUILD:	Cell Population Comment
SIM:	[unused]
VAR:	[unused]
UNIT:	n/a
TYP:	n/a

This is a label for a cell population on *simbuild's* graphical display. It does not appear in the ".sim" file. The comment is used in *simbuild's* Find function, so a unique informative comment for each population is suggested.

If the cell population type is one used by the Lung Model, (Phrenic, Lumbar, etc.), *simbuild* will prepend the required prefix to the comment and include the ".snd" file name in the ".sim" file. The *simrun* program will read the ".snd" file and parse the comment to determine the Lung Model type of this population.

SIMBUILD:	[implicit]
SIM:	targetpop_count
VAR:	NTGR
UNIT:	count
TYP:	n/a

This parameter is the number of connections between populations whose source is this cell population. This is not explicitly entered in *simbuild*, because *simbuild* determines it from the network drawing. A single "connection between populations" represents a connection from each cell of the source population to some number of cells of the target population. There can be multiple "connections between populations" from the same source population to the same target population, and each of those connections is counted separately for this parameter.

SIMBUILD:	Noise Amplitude
SIM:	noise_amp
UNIT:	nanosiemens
TYP:	0 - .3

Each cell has an internal noise generator that acts like two synapses, one with an equilibrium potential of 70 mV above resting and the other with -70mV. Each acts like it has an incoming firing probability of .05 per time step, and a synapse time constant of 1.5ms. This parameter is the conductance that gets added to the synapse conductance on each (virtual) spike.

3.2.2 Hybrid IF Cell Parameters

If cell population type is set to "Burster", the cell model changes to Hybrid IF. The Hybrid IF model is derived from the model described in

> Barbara J. Breen, William C. Gerken, Robert J. Butera, Jr., Hybrid integrate-and-fire model of a bursting neuron, *Neural Computation, v.15* n.12, p.2843-2862, December 2003.

A set of parameter controls specific to this cell type is displayed in *simbuild*.

The TYP values shown below are from Breen et al. (2003). If we have used a different value, it is shown in parentheses.

There is no description of these parameters unless the description differs from the parameter used in Breen et al. (2003).

If a parameter that appears on the *simbuild* Cell Population Page for a Hybrid IF population does not appear in this list, it is documented under the MacGregor Parameters (see [MacGregor Parameters], page 19).

SIMBUILD:	Time Constant for h
SIM:	taubar_h
BREEN:	$\bar{\tau}_h$
UNIT:	milliseconds
TYP:	10,000 (2,000)

SIMBUILD:	NaP conductance
SIM:	g_NaP_h
BREEN:	$g_{\mathrm{NaP-h}}$
UNIT:	nanosiemens
TYP:	2.8 (3)

SIMBUILD:	Half-voltage for h
SIM:	theta_h
BREEN:	Θ_h
UNIT:	millivolts
TYP:	-48 (-51)

SIMBUILD:	Slope for h
SIM:	sigma_h
BREEN:	σ_h
UNIT:	millivolts
TYP:	6 (5)

SIMBUILD:	Half-voltage for activation
SIM:	theta_m
BREEN:	$\Theta_{m-\text{NaPh}}$
UNIT:	millivolts
TYP:	-40 (-43)

SIMBUILD:	Slope for activation
SIM:	sigma_m
BREEN:	$\sigma_{m-\text{NaPh}}$
UNIT:	millivolts
TYP:	-6

SIMBUILD:	Reset Voltage @h=0
SIM:	Vreset
BREEN:	$V_{\text{Reset}}(0)$
UNIT:	millivolts
TYP:	-47.359 (-42)

SIMBUILD:	Threshold Voltage
SIM:	Vthresh
BREEN:	$V_{\text{Thresh}}(h)$
UNIT:	millivolts
TYP:	(-37)

In Breen et al. (2003), this parameter is a function of h. In the simulator, it is a constant.

SIMBUILD:	Delta_h @ h=0
SIM:	delta_h
BREEN:	$\Delta h(0)$
UNIT:	dimensionless
TYP:	-0.00078 (0)

SIMBUILD:	Applied Current (Iapp)
SIM:	GE0
BREEN:	I_{app}
UNIT:	picoamps
TYP:	13-25 (0)

3.2.3 Pulmonary Stretch Receptor Parameters

If cell population type is set to "PSR", the PSR model is used for the population. The PSR model is intended for modeling Pulmonary Stretch Receptors. The steady-state outgoing firing rate from a PSR cell is the same as the steady-state incoming firing rate, but there is an exponential delay between changes in the incoming firing rate and changes in the outgoing firing rate. A set of parameter controls specific to this cell type is displayed in *simbuild*. Only the firing rate of the incoming connections matters, not the strength or the equilibrium potential or the time constant. The firing rate is converted in the PSR model to a firing probability per time step.

SIMBUILD:	Rise Time
SIM:	DCTH
UNIT:	milliseconds
TYP:	500

If the incoming firing probability, `IFP`, is higher than the outgoing firing probability, `OFP`, the outgoing firing probability is updated on each time step as follows:

$$\texttt{OFP} = \texttt{IFP} + (\texttt{OFP} - \texttt{IFP}) * \texttt{e}^{\frac{-\texttt{step}}{\texttt{Rise Time}}}$$

So the `OFP` will get most of the way up to the `IFP` in `Rise Time` milliseconds.

SIMBUILD:	Fall Time
SIM:	DCG
UNIT:	milliseconds
TYP:	500

If the incoming firing probability, `IFP` is lower than the outgoing firing probability, `OFP`, the outgoing firing probability is updated on each time step as follows:

$$\texttt{OFP} = \texttt{IFP} + (\texttt{OFP} - \texttt{IFP}) * \texttt{e}^{\frac{-\texttt{step}}{\texttt{Fall Time}}}$$

So the `OFP` will get most of the way down to the `IFP` in `Fall Time` milliseconds.

SIMBUILD:	Output Threshold
SIM:	Thr
UNIT:	dimensionless
TYP:	0

The probability that there will be a spike on a particular time step is calculated as $OFP - OutputThreshold$ (OFP = outgoing firing probability). If the result is negative, there are no output spikes.

3.2.4 Normal Fiber Population Parameters

SIMBUILD:	Probability of Firing
SIM:	probability
VAR:	P
UNIT:	dimensionless (probability)
TYP:	.01-.05

Each fiber in the population generates an action potential at each time step with the probability specified by this parameter. Therefore, if the time step in seconds is T, the average firing rate of each fiber will be P/T.

SIMBUILD:	Time to begin firing
SIM:	start
VAR:	INSTR
UNIT:	milliseconds
TYP:	

The probability of firing for all the fibers in the population is 0 until the time specified by this parameter, at which point the "Probability of Firing" parameter takes over. This parameter is in milliseconds from the start of the simulation.

SIMBUILD:	Time to end firing
SIM:	stop
VAR:	INSTP
UNIT:	milliseconds
TYP:	

The probability of firing for all the fibers in the population is 0 after the time specified by this parameter. The "Probability of Firing" parameter has no effect after this time. This parameter is in milliseconds from the start of the simulation. If the value is -1, the time end firing is the same as the length of the simulation.

SIMBUILD:	Random Number Seed
SIM:	infsed
VAR:	INFSED
UNIT:	
TYP:	

The firing patterns of the fibers in the population is determined by a pseudo-random number generator, and this parameter determines the sequence of numbers generated by the pseudo-random number generator. The firing pattern will always be the same for the same value of this parameter (but different for different fibers in the population). Different fiber populations should have different values of this parameter in order to generate different firing patterns.

SIMBUILD:	Fibers in population
SIM:	fiber_count
VAR:	N
UNIT:	count
TYP:	

All the fibers in a population have the same values of the other parameters specified in this section, but each fiber in the population will have a different firing pattern because it uses different numbers from the pseudo-random number generator, so each fiber is simulated separately. This parameter specifies how many fibers in the population are simulated.

SIMBUILD:	Fiber Comment
SIM:	[not used]
VAR:	[not used]
UNIT:	
TYP:	

This is not really a simulation parameter, but instead is just a label for a fiber population on *simbuild*'s graphical display. It does not appear in the ".sim" file. The comment is used in *simbuild's* Find function, so a unique informative comment for each population is suggested. It is listed here because it appears in *simbuild* on the fiber population parameters screen.

SIMBUILD:	[not used]
SIM:	targetpop_count
VAR:	NTGR
UNIT:	
TYP:	

This parameter is the number of connections between populations whose source is this fiber population. This is not explicitly entered in *simbuild*, because *simbuild* determines it from the network drawing. A single "connection between populations" represents a connection from each fiber of the source population to some number of cells of the target population. There can be multiple "connections between populations" from the same source population to the same target population, and each of those connections is counted separately for this parameter.

3.2.5 Electric Stimulation Fiber Population Parameters

This type of fiber population simulates an electrical stimulation from an external source. See Normal Fiber Parameters, page 25 for parameters that are common. There is always one fiber in this type of population. There are two types, deterministic and fuzzy.

3.2.5.1 Deterministic Electric Stimulation

This type fires with a probability of one at the specified frequency. The random number seed is unused.

SIMBUILD:	Frequency
SIM:	frequency
VAR:	n/a
UNIT:	Hz
TYP:	1

This is the frequency of firing in Hz. Typical parameters for the axon/synapse for this population is a value of one for the minimum and maximum conductance time (that is, one simulation step) and a synapse strength of one.

3.2.5.2 Fuzzy Electric Stimulation

This type randomly fires at a time that is in a range centered on the frequency times.

SIMBUILD:	Fuzzy Frequency Width
SIM:	fuzzy_range
VAR:	
UNIT:	milliseconds
TYP:	

This specifies the width of the interval in which a firing event centered on the frequency time will occur. The offset from the center is random. The random number seed is used to seed the random number generator.

3.2.6 External Afferent Signal Fiber Population Parameters

This type of fiber population uses a file as an external signal source. At this time, this is a CED Spike2 file. A typical example would be a blood pressure recording from an experiment, but any type of signal can be used. See Normal Fiber Parameters, page 25 for parameters that are common. A range of expected signal values is mapped to a corresponding range of firing probabilities. Values occurring within a range use the start and stop probability values. If they are the same, all values within the range have the same probability. If they differ, linear interpolation is used to determine the firing probability for each value in the range. Values outside of all of the range have a probability of zero. The range must be in monotonic increasing order. *Simbuild* will sort the list when changes are Applied.

It is intended to eventually provide the means to connect to different kinds of sources, such as another simulator that can provide values in real time.

SIMBUILD:	Afferent File
SIM:	afferent_file_name
VAR:	
UNIT:	n/a
TYP:	

This is the name of the .smr or .smrx file containing the signal information. This should have a single ADC channel that starts at time zero.

SIMBUILD:	Signal Offset
SIM:	offset
VAR:	
UNIT:	n/a
TYP:	

The zero value of Y axis in *simviewer* is near the bottom of plot rows. Signals that start with values larger or smaller than this results in plots that do not use the maximum range of the row. This value is used to translate the signal up or down in the plot. For example, if the smallest value in a signal file was 100, an offset of -100 would move it down in the plot.

SIMBUILD:	Afferent Value
SIM:	aff_val
VAR:	
UNIT:	n/a
TYP:	

SIMBUILD:	Firing Probability
SIM:	aff_prob
VAR:	
UNIT:	n/a
TYP:	

These values/probability pairs are entered into a table in *simbuild.* At least 2 entries are required, up to a maximum of 32. The user may think of these as points in the plane. For example:

(0,0.1) (100,0.3) (140,0.3) (200,0,1)

Any signal value between 0 and 100 would have an interpolated firing probability between 0.1 and 0.3. Values between 100 and 140 would have a firing probability of 0.3, and so on.

SIMBUILD: implied	
SIM:	num_aff
VAR:	
UNIT:	n/a
TYP:	

The value/probability arrays are of a fixed size. This is the number of entries in the arrays.

3.2.7 Axon/Synapse Parameters

SIMBUILD:	Conduction Time
SIM:	NCT
VAR:	NCT
UNIT:	time steps
TYP:	0-4

When an action potential fires in a cell of a source population, the effect will be felt at the target cell after some number of time steps. This parameter specifies the maximum number of time steps for this connection between populations. Each individual cell-to-cell or fiber-to-cell connection within this connection between populations will have its own conduction time, randomly chosen between 1 and the value of this parameter, inclusive, but always the same for a particular cell-to-cell or fiber-to-cell connection.

SIMBUILD:	Number of terminals
SIM:	NT
VAR:	NT
UNIT:	count
TYP:	10-100

Each cell or fiber in the source population makes the same number of connections in the target population, and this parameter specifies that number. The particular target cells that a particular source fiber or cell is connected to are chosen at random from among the target population, and they remain the same for the duration of the simulation. Multiple source cells or fibers can be connected to the same target cell, and more than one of the terminals from single source fiber or cell can be on the same target cell.

SIMBUILD:	Synapse Strength
SIM:	STR
VAR:	STR
UNIT:	dimensionless - ratio to resting conductance
TYP:	.0025-1.4

When a synapse fires on a cell in the target population (after the conduction time), the membrane conductance of the target cell increases instantaneously by the amount specified by this parameter. The value is specified in units of the resting membrane conductance, so 2 means twice the resting membrane conductance.

SIMBUILD:	Random Number Seed
SIM:	INSED
VAR:	INSED
UNIT:	
TYP:	

The particular target cells that are chosen for each source cell or fiber, and the associated conduction time, are determined by the value of this parameter. The same choices are always made for the same value of this parameter. This parameter should be set to a different value for each connection in order to get different connection patterns.

SIMBUILD:	Synapse type
SIM:	TYPE
VAR:	TYPE
UNIT:	index or name
TYP:	1-6 or name

All the synapses associated with a particular connection between populations are of the type specified by this parameter. In the ".sim" file, this parameter is an index into the list of synapse types. The properties of a synapse type are specified by other parameters (see Synapse Parameters). In *simbuild*, this parameter is specified by using a name associated with the synapse index number (see the synapse type parameters "Synapse Name" and "Synapse Number").

SIMBUILD:	[implicit]
SIM:	IRCP
VAR:	IRCP
UNIT:	index
TYP:	1-10

In the ".sim" file, for each source population, this parameter specifies the identity of the target cell population for each connection. It is not entered explicitly in *simbuild*, because *simbuild* determines it from the network drawing.

3.2.8 Synapse Type Parameters

All of the synapse types have Equilibrium Potential and Time Constant parameters. The Hebbian Learing type has several additional ones.

SIMBUILD:	Synapse Eq. Potential
SIM:	EQ
VAR:	EQ
UNIT:	millivolts
TYP:	-25 - 115

Each of the per-synapse conductances and the potassium conductance and the resting membrane conductance has an equilibrium potential associated with it. The membrane potential decays exponentially toward the weighted average of these equilibrium potentials, each weighted by its conductance. This parameter specifies the equilibrium potential for synapses of this synaptic type, relative to the resting membrane potential.

SIMBUILD:	Synapse Time Constant
SIM:	DCS
VAR:	T
UNIT:	milliseconds
TYP:	0.1 - 2.0

When a synapse fires on a target cell, the membrane conductance of the target cell increases instantaneously by the synaptic strength (see the STR Axon/Synapse parameter). This additional conductance decays exponentially toward 0 with the time constant specified by this parameter (except $\mathtt{DCS} = \mathtt{e}^{\frac{-\mathtt{step}}{\mathtt{T}}}$).

SIMBUILD:	Learn Window
SIM:	lrnWindow
VAR:	MLAG
UNIT:	ticks
TYP:	4

Learning synapses maintain a history list that records the time a sending node fired. MacGregor MacGregor (1987) refers to this as the arrival time. This parameter determines how many ticks an event will be considered significant.

This is a per terminal list and the actual value will be a random number 2-N for each terminal. If a receiving cell fires, it will consult the history list. If there are events in the list, the synapse strength will be modified, controlled by parameters listed next.

At the end of each tick, the arrival time is decremented. When *arrival time* ≤ 1, the event is removed from the list. A sender and a receiver firing at the same time is not considered a significant event.

SIMBUILD:	Strength Multiplier
SIM:	lrnStrDelta
VAR:	DLSTR
UNIT:	
TYP:	.05-.25

The synapse strength is modified using this value. If this value is positive, the strength is increased up to a maxium value. If it is negative, the strength is decreased down to a minium value.

The initial strength value is set on the Synapse Parameter page. Using *strength* as the starting and subsequent strength value, strength_max as the maximum possible value, using fabs as the floating point absolute value function, the equation for the new strength value is:

$$strength = strength + lrnStrDelta * (fabs(strength\_max - strength))$$

Note that *strength_max* can be considered the minimum for lrnStrDelta values less than zero.

If $strength = strength\_max$, the modification value becomes zero, so there will no longer be changes to strength.

fish

SIMBUILD:	Max/Min Strength
SIM:	lrnStrMax
VAR:	STRMX
UNIT:	
TYP:	1-2

This is the minimum, in the case of negative lrnStrDelta values, or maximum, in the case of positive lrnStrDelta values.

Some parameter combinations may not be useful. Starting with a very small initial strength and using a delta multiplier that is negative and a larger mimimum value may not have the any effect.

SIMBUILD:	Synapse Number
SIM:	syntype
VAR:	[varies]
UNIT:	index
TYP:	1-6

In *simbuild*, the synapse type parameters for each synapse type are associated with a synapse type index number, which is referenced by the Axon/Synapse parameter TYPE. This parameter specifies that index number. In the ".sim" file, "syntype" starts with 0, so "syntype" is the *simbuild* synapse type minus 1.

SIMBUILD:	Synapse Name
SIM:	[not used]
VAR:	[not used]
UNIT:	
TYP:	

This is not a simulation parameter, but just a label specified on *simbuild*'S "Synapse Types Page" for a synapse type, for use on *simbuild*'s "Synapse Parameters Page", instead of using the synapse type index number. It does not appear in the ".sim" file.

3.2.9 Global Parameters

SIMBUILD:	Length of simulation
SIM:	step_count
VAR:	LTSTOP
UNIT:	seconds
TYP:	60.00

The duration of the simulation in simulated time, in seconds. The size of a time step is specified by the Simulation step size in milliseconds, page 34.

SIMBUILD:	Potassium Equilibrium Potential
SIM:	EK
VAR:	EK
UNIT:	millivolts
TYP:	-10

Each of the per-synapse conductances and the potassium conductance and the resting membrane conductance has an equilibrium potential associated with it. The membrane potential decays exponentially toward the weighted average of these equilibrium potentials, each weighted by its conductance. This parameter specifies the equilibrium potential for the potassium conductance, relative to the resting membrane potential. This potassium equilibrium potential is the same for all cells in all populations in the model.

SIMBUILD:	Step size
SIM:	step
VAR:	STEP
UNIT:	milliseconds
TYP:	.1, .5, other

The value of the membrane potentials and the firing state are calculated at the interval specified by this parameter. Two explicit values, 0.5 for ".bdt" files and 0.1 for ".edt" files are provided. The user can also enter a custom value. This is also taken to be the duration of an action potential (see the cell population parameter "Change in potassium conductance with action potential".)

SIMBUILD:	Global Comment
SIM:	[not used]
VAR:	[not used]
UNIT:	
TYP:	

This is not really a simulation parameter, but just a label for the model on *simbuild*'s graphical display. It does not appear in the ".sim" file. It is listed here because it appears in *simbuild* on the "Globals Page".

SIMBUILD:	[implicit]
SIM:	[not used]
VAR:	NTPOPS
UNIT:	
TYP:	

The number of cell populations plus the number of fiber populations. Specified in ".sim" file. Determined by *simbuild* from the network drawing.

SIMBUILD:	[implicit]
SIM:	fiberpop_count
VAR:	NFPOPS
UNIT:	
TYP:	

Specified in ".sim" file. Determined by *simbuild* from the network drawing.

SIMBUILD:	[implicit]
SIM:	cellpop_count
VAR:	NCPOPS
UNIT:	
TYP:	

Specified in ".sim" file. Determined by *simbuild* from the network drawing.

SIMBUILD:	[implicit]
SIM:	[not used]
VAR:	MCTP1
UNIT:	
TYP:	

Maximum of the Axon/Synapse "Conduction Time" parameters, plus 1. Specified in ".sim" file. Determined by *simbuild* from the Axon/Synapse "Conduction Time" parameters.

SIMBUILD:	[implicit]
SIM:	syntype_count
VAR:	SNTP
UNIT:	
TYP:	

Specified in ".sim" file. Determined by *simbuild* from the synapse type definitions.

SIMBUILD:	[implicit]
SIM:	[not used]
VAR:	NTGMX
UNIT:	
TYP:	

Maximum of the Cell Population " Number of targets of cell populations" parameters. Specified in ".sim" file. Determined by SIMBUILD from the network drawing..

SIMBUILD:	[implicit]
SIM:	[not used]
VAR:	NCLS
UNIT:	
TYP:	

Maximum of the Cell Population "Population Size" parameters. Specified in ".sim" file. Determined by *simbuild* from the Cell Population "Population Size" parameters.

3.3 The *simrun* Simulation Engine

3.3.1 OLS Generation

At the beginning of every run, the simulator will write a file named "sample_cells_[launch number].ols" to the current directory, overwriting any existing file by that name. It contains a subset of the cells in the model such that one of each endpoint of every connection between populations in the model is represented. The file can be renamed and used as a ".ols" input file to *simrun*. It only contains information on cells to be written to the ".bdt" or ".smr" file.

In addition, the simulator will write a file named "sample_cells_rosetta_[launch number].txt", which documents the mapping between the cell numbers as seen in the ".bdt" file and the population and cell numbers as seen in *simbuild*. It also documents the same information listed by connection, together with the synapse type.

The cells included in "sample_cells.ols" are chosen using a pseudo-random number generator. If the file "sample_cell_seed" is found in the working directory, the seed for the random number generator it taken from it. Otherwise, the current time in seconds is used as a seed, and is written to "sample_cell_seed".

3.3.2 Power Spectrum

NOTE: This section only applies to Linux installations.

The simulator can create a power spectrum from a simulated phrenic signal.

If the option '`Create analog entries`' is checked on the '`Simulator Launcher Window`' (see [Simulator Launcher Control Window], page 12), at the end of a run, the simulator will generate a power spectrum from the analog signal written to the "spawnN.bdt" or "spawnN.edt" file during the run. The parameters for the analog signal should be set properly on the '`Launch Window`' as explained in Simulator Launcher Control Window, page 12.

The plot data for the spectrum is written to a file named "power_spectrum_plot_data_N", in the current directory, overwriting any existing file by that name. The "N" in the file name "power_spectrum_plot_data_N" is the spawn number (see [Simviewer Waveform Display], page 38). It will use a program named "gnuplot" if it exists. The program "gnuplot" is not part of this package, but it is available free of charge on most Linux systems. available.

The simulator assumes that the analog data in the ".bdt" file is an integrated phrenic signal, and it processes it to determine the beginning and end of each burst. The spectrum is the average spectrum of the first 512 milliseconds of each burst. The timing marks for the beginning and end of each burst are written as channels 97 and 98 to a file named "ieN.edt" (the extension is always "edt"), and then that file and spawnN.bdt are combined to create mergeN.edt.

3.3.3 Simulator Options

Support for running simrun from the command line has been removed from this manual since it is expected almost no one would do this. See this section in sim.pdf if you want to run this from the command line.

3.4 Simviewer Waveform Display

The *simviewer* program can be used in two modes.

When running a simulation, *simbuild* will start an instance for each Launch number. The default is to use the network connection between *simrun* and *simviewer* to display the results in real time.

If the Save Plot Data To Files option was selected in the Simulator Launcher Control Window, the plot data was saved to wave files. The file name format is wave.[launch number].[sequence number]. The first file for launch number 0 is wave.00.0000, the second one is wave.00.0001, and so on.

The *simviewer* program is invoked from the command line like so:

```
simviewer [-l launch number] [-p port number] [--file | --socket]
```

The -p and –socket options should not be used from the command line. If they are missing and -l or –file is omitted, the program defaults to a launch number of 0 and will try to read wave files.

Most of the controls do obvious things. Since new controls are often added, running the program and exploring the options is the best way to learn how to use it. There is pop-up tooltip help for many of the controls.

The sliders are used to select values. The boxes where values are displayed are not editable. To select specific values, the up and down arrow keys increase or decrease the value by one.

The Save PDF option saves all of the rows, but only includes what is visible in the horizontal axis of plotting window. The user should manipulate the display to include only what what they want to be sent to the .pdf file.

The label displayed with each row starts with the population number and the Cell Population Comment.

If the row is displaying population activity, the next item in the label is the time interval over which the firing rate is calculated (similar to the bin size of a CTH). And the final item is the firing rate in spikes per second at the voltage marker line, which changes as the voltage marker lines are moved.

If the row is displaying some value other than population activity, the Cell Population Comment is followed by the name of the variable being displayed, which will be one of the following:

Vm:	membrane potential
gK:	potassium conductance
Thr:	firing threshold
h:	hybrid IF inactivation variable

followed by the value of the variable at the voltage marker line and the units in which it is measured. The value of the variable at the voltage marker line changes as the voltage marker lines are moved.

If “h” is being displayed, there are no units because “h” is dimensionless. If “gK” is being displayed, there are no units because it is a ratio to resting membrane conductance. If “Vm” or “Thr” are being displayed, the units are “mV abs” for a hybrid IF cell because the voltage is absolute (the actual voltage across the membrane), and “mV rel” for the MacGregor model because it is relative to the resting potential.

3.5 The Lung Model

3.5.1 Interface to the Network Model

3.5.1.1 Motor Populations

The simulator includes a model of the lungs. It takes as input the mean firing rate of the phrenic motor, the lumbar (abdominal muscle) motor, and the inspiratory and expiratory laryngeal motor neuron populations, and produces as output the lung volume, alveolar pressure, and tracheal flow.

To use the lung model, you must label the above-mentioned populations appropriately (in *simbuild*), as follows:

Population	Name must contain the word
Phrenic	phrenic
Lumbar	lumbar
Inspiratory Laryngeal	ILM or PCA
Expiratory Laryngeal	ELM or TA

Any of the letters in the word may be upper or lower case.

Also, the word ‘`pre`’ must not appear in the name. This is also case insensitive.

You may have more than one phrenic and more than one lumbar population. The word “phrenic” or “lumbar” in the additional populations names must be followed immediately by a small positive integer (e.g. phrenic1).

3.5.1.2 Recruitment Equations

The activation of the diaphragm by the phrenic and the abdominal muscles by the lumbar are determined by equations provided by the user. The equations are entered in the appropriate fields of the ‘`Edit Globals`’ page, which is displayed by clicking on the Edit Globals button. The phrenic recruitment equation (actually an expression) should be in terms of variables named `P0`, `P1`, etc., – one for each of the phrenic populations. Each of these variables represents the mean instantaneous firing rate of the corresponding population. `P0` is the variable for the unnumbered-numbered phrenic population. If no equation is entered, it defaults to $\frac{\texttt{P0}}{\texttt{100}}$. The value of the expression is the muscle activation, where 0 is no activation and 1 is maximum activation. If the value is greater than 1, it acts like 1, and if it is less than 0, it acts like 0. The lumbar recruitment equation is similar, except that the variables are `L0` etc., and the default equation is $\frac{\texttt{L0}}{\texttt{20}}$.

If you have two phrenic populations, and the firing rate of individual phrenic neurons maxes out at 100 spikes/second, an appropriate phrenic recruitment equation might be $\frac{\texttt{.3*P0+.7*P1}}{\texttt{100}}$. This approximates the plot in Fig. 1 of Mantilla and Sieck (2011). A max rate of 100 $\frac{\texttt{spikes}}{\texttt{second}}$ for phrenic is suggested by Nail et al. (1972).

The laryngeal motor neurons don’t have recruitment equations. The firing rate for maximum activation for both ILM and ELM can be entered by clicking on the Edit Globals button to display the Globals page.

3.5.1.3 Pulmonary Stretch Receptors

The volume output of the lung model can be use to control the injected current of any cell population, thereby creating a pulmonary stretch receptor. To do this, simply enter an expression involving the variable `V` (representing lung volume) in the Injected Expresison control the Cell Parameters Panel. The simulator will evaluate the expression at every step using the lung volume at that step to determine the injected current at that step. If no cell population has such an expression, the lung model will not be used, which will speed up the simulation, and the lung model outputs will not be available for plotting with *simviewer*.

For example, if you want an injected current that is proportional to lung volume, with 0 at RV, an appropriate expression might be '`.5*RV`'. Or if you want an injected current that is 0 at TLC and rises with decreasing lung volume, you might use '`.5*(RV-TLC)`'.

subsubsectionVariable Plots

The volume, pressure, and flow and other variables can be plotted with *simviewer* by selecting the appropriate item in the drop-down list in the Plot Variable cell in the Launcher Window.

State Variable	Units
lung volume	%VC, relative to RV
tracheal flow	%VC per second expiration positive (up)
alveolar pressure	cmH_2O
Phr_d: diaphragm activation	dimensionless ratio to max
u: abdominal muscle activation	dimensionless ratio to max
lma: net laryngeal muscle activation	dimensionless ratio to max
Vdi: diaphragm volume	liters
Vab: abdominal volume	liters
Vdi_t: derivative of Vdi	liters/sec
Vab_t: derivative of Vab	liters/sec
Pdi: transdiaphragmatic pressure	cmH_2O
Pab: abdominal pressure	cmH_2O
PL: transpulmonary pressure	cmH_2O
Phr_d: diaphragm activation limited, $0 \rightarrow 1$	dimensionless ratio to max
u: abdominal muscle activation limited, $0 \rightarrow 1$	dimensionless ratio to max
lma: net laryngeal muscle activation limited, $-1 \rightarrow 1$	dimensionless ratio to max

The offset and scaling of the plot are entered in the two right-most cells of the table. The Offset specifies the value at the baseline in units of the variable and the Scaling specifies the number of units of the variable per pixel (at 1X gain). Since the user may wish to specify a fractional value for the offset or scale, and since these fields can only take integers, the user must multiply the desired value by 10,000 and round to the nearest integer before entering the value. The text in the header of the columns indicates this.

3.6 Utilities

The utilities are command line programs than can be run stand-alone or from scripts. These are only available for the Linux verison.

3.6.1 renumber

The “renumber” program renumbers “.ols” files. The “.ols” file generated by *simbuild* contains the list of cells, fibers and populations that are to be written by the simulator to the “spawnN.bdt” file and to the “wave.NN.*” files. It is a text file and can be edited by hand, but the lines are numbered and the numbers must be in order. This makes it awkward to edit by hand, because if the lines are rearranged or deleted, every line number might have to be updated.

The “renumber” utility will do this renumbering for you. After editing the “.ols” file without regard to the line numbers, type

```
renumber whatever.ols
```

and the file ‘`whatever.ols`’ will be renumbered. A backup file is created with the name ‘`whatever.ols.orig`’, before the renumbering is done. Any previous backup file will be overwritten.

An interpreter for the “perl” programming language must be installed on the system at /usr/bin/perl for this utility to work.

3.6.2 pickwave

The “pickwave” program picks a waveform out of a specified set of wave.* files, and prints the values as text to standard output (the screen). If you type

```
pickwave --help
```

at the command line, it will type

```
Usage: pickwave [OPTION...] [FILE...]
picks a waveform out of a specified set of wave.* files

  -n, --spawn=N           take input from wave.* files with spawn number N
  -s, --spikes            pick spike data instead of waveform data
  -w, --wave=N            output data for wave number N
  -?, --help              Give this help list
      --usage             Give a short usage message
  -V, --version           Print program version

Mandatory or optional arguments to long options are also mandatory or
optional for any corresponding short options.

The wave number for the -n option is the position in the wave.* file.
The first position is 1, not 0.
```

The program is run from the command line, and may be useful for importing the data into other programs such as Matlab.

3.6.3 pickedt

The "pickedt" program picks a waveform out of a specified ".bdt" file, and prints the values as text to standard output (the screen). If you type

```
pickedt --help
```

at the command line, it will type

```
Usage: pickedt [OPTION...] [FILE...]
picks a waveform out of specified .bdt or .edt file(s)

  -a, --analog_id=N         output data for analog id N
  -c, --ids                 output all ids on stderr
  -o, --offset=N            add N to each analog value
  -s, --spike_id=N          output data for spike id N
  -t, --starttime           first line of output is start time in units of
                            the sample period
  -?, --help                Give this help list
      --usage               Give a short usage message
  -V, --version             Print program version

Mandatory or optional arguments to long options are also mandatory or
optional for any corresponding short options.

The output is text, one value per line, on standard output
```

This program is used by the power_spectrum.sh utility. It can also be run from the command line, and may be useful for importing the data into other programs such as Matlab.

3.6.4 spectrum

The “spectrum” program computes the average power spectrum of phrenic bursts. Type

```
spectrum --help
```

at the command line, and it will type

```
Usage: spectrum [OPTION...] [FILE...]
computes the average power spectrum of phrenic bursts

  -d, --detrend           remove the best-fit line from the phrenic burst
                          data
  -f, --filter            bandpass filter the input .3 to 200 Hz before
                          processing (default is no filter)
  -m, --spawnnum=N        the I and E pulses are written to ieN.edt
  -n, --fftsz=N           power spectrum is of first N samples of each
                          phrenic burst (overrides -p)
  -p, --period=N          power spectrum is of first N seconds of each
                          phrenic burst (default .5)
  -r, --rectify           subtract the mean and then take the
                          absolute value of the input data
  -s, --stepsize=N        set stepsize of input data to N ms (default .5)
  -t, --threshold=N       controls the placement of the I pulse - larger
                          = later (default .025)
  -w, --window            apply a Hann window to the phrenic burst data
                          after any detrend
  -?, --help              Give this help list
      --usage             Give a short usage message
  -V, --version           Print program version

Mandatory or optional arguments to long options are also mandatory or
optional for any corresponding short options.

The input is binary C floats representing phrenic population activity.
The output is lines of text on standard output, two values per line
The first value is frequency in Hz, the second is power

If the -f option is used, the filtered input is written to a file named
"filtered" in the current directory

FFT optimization information is read from and written to a file named
"wisdom" in the current directory.
```

This program is used by power_spectrum.sh. It can also be run from the command line.

3.6.5 txt2flt

The “txt2flt” program converts text numbers to binary numbers.

If you type

```
txt2flt --help
```

at the command line, it will type

```
Usage: txt2flt [OPTION...] [FILE...]
converts text numbers to binary numbers

  -d, --double              output binary double precision C floating point
                            values
  -f, --float               output binary single precision C floating point
                            values (default)
  -i, --int                 output binary C int (integer) values
  -s, --short               output binary C short integer values
  -?, --help                Give this help list
      --usage               Give a short usage message
  -V, --version             Print program version

The output is binary numbers on standard output.
```

This program is used by the power_spectrum.sh utility. It can also be run from the command line.

3.6.6 merge

The “merge” program combines two “.bdt” or “.edt” files into a single file. The input files can be of different types. The types are determined by the contents of the files, not by the names. The type of the output is “.edt” if either input is, otherwise it is “.bdt”.

It is invoked as follows:

```
merge input1.[eb]dt input2.[eb]dt > output.[eb]dt
```

The output file will be sent to standard output (the screen), so you would normally want to redirect it to a file, as shown above.

The “merge” program can also be used to merge more than two input files, as in this example:

```
merge input1.bdt input2.edt | merge input2.bdt > output.edt
```

This program is used by power_spectrum.sh, but can also be run from the command line.

3.6.7 power_spectrum.sh

The “power_spectrum.sh” program generates a power spectrum from analog channel 1 of a “.bdt” file named spawnN.bdt. It takes two arguments. The first is the value of N in spawnN.bdt. The second is the step size of the analog data in milliseconds.

The plot data for the spectrum is written to a file named "power_spectrum_plot_data_N", in the current directory, overwriting any existing file by that name. If there is a program named "gnuplot" on the system, power_spectrum.sh will use that to open the plot data. The program "gnuplot" is not part of this package, but it is available free of charge on most Linux systems.

The "power_spectrum.sh" program assumes that the analog data in the ".bdt" file is an integrated phrenic signal, and it processes it to determine the beginning and end of each burst. The spectrum is the average spectrum of the first 512 ms of each burst. The timing marks for the beginning and end of each burst are written as channels 97 and 98 to a file named "ieN.edt", and then that file and spawnN.bdt are combined to create mergeN.edt.

The "power_spectrum.sh" program is used by the simulator to automatically generate a power spectrum at the end of a run, but it can also be run from the command line.

3.6.8 edt2spike2, edt2spike2.exe

This is a command line utility that reads ".bdt" and ".edt" files and creates a ".smr" file that the CED Spike2 program can read. There are both Linux and Windows versions of this utility. The base name of the input file will be used to create the name of the output file.

This functionality is already in the *simrun* program. This should be used to create a ".smr" file from a ".bdt" file produced by other programs.

```
edt2spike2 -h
```

at the command line, it will display

```
Program to convert .bdt or .edt files to Spike2 .smr files.

Usage: edt2spike2 -n <filename.edt | filename.bdt>
For example:

edt2spike2 -n 2014-06-24_001.edt
or
edt2spike2 -n c:\path\to\2014-06-24_001.edt
```

Chapter 4

Lung Model Internals

4.1 The Top Level

At the top level, there are just two equations in the model (the variable names are the same as those used in the source code found in the file named lung.c):

```
-VL_t * Rrs - sigma_L + (fa + Fdi) * sigma_di + Pica - sigma_rc = 0

sigma_ab + VL_t * Rrs + sigma_L - sigma_di = 0
```

VL_t is $\frac{\mathtt{VL}}{\mathtt{dt}}$, the rate of change of lung volume in liters/sec, and is therefore the tracheal flow, with positive values during inspiration.

Rrs is airway resistance in cmH_2O/(lters/sec).

sigma_L is the recoil pressure of the lung in cmH_2O due to its elastance, positive values acting to contract the lung.

fa is the fraction of the rib cage exposed to abdominal pressure. The remainder is exposed to pleural pressure. This is a dimensionless value between 0 and 1.

Fdi is the fraction of the diaphragm pressure that acts to expand the rib cage by way of insertional forces from the diaphragm acting on the lower ribs. This is a dimensionless value equal to .15, from Loring and Mead (1982) where it is $\mathtt{c} \cdot \frac{\mathtt{Adi}}{\mathtt{Arc}} = \mathtt{0.25} \cdot \mathtt{0.6}$. (Their Fdi is defined differently.)

sigma_di is the recoil pressure of the diaphragm in cmH_2O due to muscle tension, elastance, and resistance, positive values acting to contract the diaphragm.

Pica is the pressure in cmH_2O generated by the intercostal and accessory muscles, positive values acting to expand the rib cage.

sigma_rc is the recoil pressure of the rib cage due to its elastance and resistance, positive values acting to contract the rib cage.

sigma_ab is the recoil pressure of the abdominal wall as seen at the level of the diaphragm in cmH_2O due to muscle tension, elastance, and resistance in the abdominal wall, positive values acting to contract the abdominal wall.

Each of these values is a function of the four motor outputs of the network model (see next section), and the diaphragm and abdominal wall volumes (`Vdi` and `Vab`) and their time derivatives (`Vdi_t` and `Vab_t`). At each time step, the motor outputs are known from the network model, the volumes are known from the end of the last time step, and the two equations are solved for the two derivatives to calculate the volumes at the next time step.

The volumes are in liters and their derivatives are in liters/sec.

4.1.1 The Motor Outputs

The phrenic motor output from the network model at each time step is calculated by first counting the total spikes from all the cells in each phrenic population, and dividing by the duration of a time step in seconds and the number of cells in each population, to get an instantaneous firing rate in spikes/sec/cell for each phrenic population. These numbers are combined by the phrenic recruitment equation (see see Section 3.5.1.2: Recruitment Equations, page 40) to generate the raw diaphragm muscle activation, with 0 indicating no activation and 1 indicating maximum activation. This raw activation is integrated by means of a simulated RC filter with a time constant of 60 ms, and then limited to a range of 0 to 1 to generate the diaphragm activation '`Phr_d`'.

The lumbar motor output is handled the same way, except that the lumbar populations and recruitment equation are used to generate the abdominal muscle activation '`u`'.

The Inspiratory Laryngeal Motor output from the network model at each time step is calculated by first counting the total spikes from all the cells in the ILM population, and dividing by the duration of a time step in seconds and the number of cells in the population, to get an instantaneous firing rate in spikes/sec/cell.

The Expiratory Laryngeal Motor output is handled the same way, except with the ELM population.

The ELM firing rate is then subtracted from the ILM firing rate, and the difference is divided by a maximum firing rate (see Section 3.5.1.2, last paragraph), and integrated by means of a simulated RC filter with a time constant of 35 ms, and then limited to a range of -1 to 1, to generate the net laryngeal muscle activation, '`lma`', where 1 indicates a maximally open airway, -1 indicated fully closed, and 0 indicates resting diameter.

4.2 The Pressure Equations

To derive the top level model equations, we start with pressure balance equations for the rib cage, lungs, diaphragm, and abdominal wall, and the pressure drop across the larynx. In this model, we ignore inertial forces, which are forces necessary to accelerate masses, in effect treating the components of the respiratory system as if they were massless. For evidence that (at least some) inertial forces are negligible, see Mead (1956). For a precedent for ignoring inertial forces in a lung model, see Younes and Riddle (1981). In the absence of inertial forces, the forces on an object must sum to zero (Newton, 1687).

All the pressure variables below are in units of cmH_2O.

4.2.1 Pressure balance on the rib cage

```
(1 - fa) * Ppl + fa * Pab + Pica + Fdi * sigma_di = sigma_rc
```

The inner surface of the rib cage above the diaphragm sees the pleural pressure, `Ppl`. The rest of the inner surface of the rib cage (the zone of apposition) sees the abdominal pressure, Pab. The variable '`fa`' represents the fraction of the rib cage that sees `Pab`, so the total pressure from those sources is `(1 - fa) * Ppl + fa * Pab`. Positive values of these pressures tend to expand the rib cage.

The action of the intercostal and accessory muscles also tends to expand or contract the rib cage. This effect is represented by the variable `Pica`, which is an equivalent pressure defined so that positive values tend to expand.

When the diaphragm contracts, the force at the diaphragm insertions tends to lift the rib cage, and through the pivoting action of the ribs at the spine, tends to expand it as well. The equivalent pressure generated by this effect is taken to be proportional to the pressure generated by the diaphragm, `sigma_di`, with `Fdi` being the proportionality constant.

These pressures that tend to expand the rib cage have to be balanced by the recoil pressure of the rib cage generated by its elastance. This pressure, `sigma_rc`, is positive when it tends to contract the rib cage.

4.2.2 Pressure balance on the lung

```
Palv - Ppl = sigma_L
```

The alveolar pressure, `Palv`, is the air pressure inside the lung, with positive values tending to expand. The pressure outside the lung is the pleural pressure, `Ppl`, and positive values of this tend to contract the lung. The difference in these pressures, `Palv-Ppl`, positive values of which tend to expand the lung, must be balanced by the recoil pressure of the lung, `sigma_L`, generated by its elastance, positive values of which tend to contract the lung.

4.2.3 Pressure balance on the diaphragm

```
Pab - Ppl = sigma_di
```

The abdominal pressure, Pab, is the pressure inside the abdomen, with positive values tending to expand the diaphragm. The pressure on the other side of the diaphragm is the pleural pressure, Ppl, and positive values of this tend to contract the diaphragm. The difference in these pressures, Pab-Ppl, positive values of which tend to expand the diaphragm, must be balanced by the recoil pressure of the diaphragm, sigma_di, generated by its elastance and muscle contraction, positive values of which tend to contract the diaphragm.

4.2.4 Pressure balance on the abdominal wall

`Pab = sigma_ab`

The abdominal pressure, `Pab`, is the pressure inside the abdomen, with positive values tending to expand the abdominal wall. This must be balanced by the recoil pressure of the abdominal wall, `sigma_ab`, generated by its elastance and muscle contraction, positive values of which tend to contract the abdominal wall.

4.2.5 Pressure drop across the larynx

`Palv = -VL_t * Rrs`

The alveolar pressure, `Palv`, is the air pressure inside the lung. This must be balanced by the pressure drop across the larynx, which, in the hydraulic analog of Ohm's law, is proportional to the flow out of the lungs, with `Rrs` being the proportionality constant. `VL_t` is the time derivative of lung volume, so positive values represent flow into the lungs, so `-VL_t` is the flow out of the lings.

4.3 Deriving the Model Equations from the Pressure Equations

By combining the pressure equations above, we can eliminate the variables `Pab`, `Ppl`, and `Palv`, leaving two equations.

To derive the first model equation from the pressure equations, start with the pressure balance on the rib cage:

`(1 - fa) * Ppl + fa * Pab + Pica + Fdi * sigma_di = sigma_rc`

Rearrange by collecting `Pab` and `Ppl` together, which both multiply `fa`:

`Ppl + (Pab - Ppl) * fa + Pica + Fdi * sigma_di - sigma_rc = 0`

Using the diaphragm pressure balance equation, replace `(Pab - Ppl)` with `sigma_di`:

`Ppl + sigma_di * fa + Pica + Fdi * sigma_di - sigma_rc = 0`

Using the lung pressure balance equation, replace `Ppl` with `Palv-sigma_L`:

`Palv - sigma_L + sigma_di * fa + Pica + Fdi * sigma_di - sigma_rc = 0`

Using the equation for the pressure drop across the larynx, replace `Palv` with `-VL_t*Rrs`:

`-VL_t * Rrs - sigma_L + sigma_di * fa + Pica + Fdi * sigma_di - sigma_rc = 0`

Rearrange by collecting `fa` and `Fdi`, which both multiply `sigma_di`:

`-VL_t * Rrs - sigma_L + (fa + Fdi) * sigma_di + Pica - sigma_rc = 0`

To derive the second model equation from the pressure equations, start with the pressure balance on the diaphragm:

`Pab - Ppl = sigma_di`

Using the abdominal wall pressure balance equation, replace `Pab` by `sigma_ab`:

`sigma_ab - Ppl = sigma_di`

Using the lung pressure balance equation, replace `Ppl` with `(Palv-sigma_L)`:

`sigma_ab - (Palv - sigma_L) = sigma_di`

Using the equation for the pressure drop across the larynx, replace `Palv` with `-VL_t*Rrs`:

`sigma_ab - (-VL_t * Rrs - sigma_L) = sigma_di`

Rearrange:

`sigma_ab + VL_t * Rrs + sigma_L - sigma_di = 0`

So the two model equations are:

`-VL_t * Rrs - sigma_L + (fa + Fdi) * sigma_di + Pica - sigma_rc = 0`

`sigma_ab + VL_t * Rrs + sigma_L - sigma_di = 0`

Each variable in these equations is a function of the diaphragm and abdominal wall volumes and their derivatives, and the muscle activations, represented by these variables:

`Vdi`:	diaphragm volume
`Vab`:	abdominal wall volume
`Vdi_t`:	time derivative of the diaphragm volume
`Vab_t`:	time derivative of the abdominal wall volume
`Phr_d`:	phrenic activation of the diaphragm
`u`:	lumbar activation of the abdominal muscles
`lma`:	net laryngeal muscle activation

In the following sections, we will show the function of these variables for each variable in the model equations. But first, we will show the volume equations, which are used in more than one of the variable equations and are fundamental to the model.

4.4 The Volume Equations

4.4.1 Abdominal Volume

`Vdi + C1 * Vrc + Vab = Vsum`

The diaphragm volume, `Vdi`, is the volume above the level of the diaphragm insertions on the rib cage and below the dome of the diaphragm. The rib cage volume, `Vrc`, is defined in the next section. The abdominal wall volume, `Vab`, is the volume between the abdominal wall and a frontal plane that coincides with the contracted position of the abdominal wall. The equation says that a properly weighted sum of these three volumes is a constant, `Vsum`, which is a computed parameter.

It has frequently been assumed that the abdominal contents are incompressible, and that the abdominal cavity has only two movable walls, the diaphragm and the abdominal wall, and that therefore a displacement in one must be met by an equal displacement of the other (Mead and Loring (1982), Grassino et al. (1978), Grimby et al. (1976), Macklem et al. (1978), Lichtenstein et al. (1992)); in other words, that `Vdi + Vab = Vsum` (`Vsum` constant). In particular, Lichtenstein et al. (1992), from which this model was adapted, includes the equation $\texttt{[Vab - Vab}_0\texttt{] + [Vdi - Vdi}_0\texttt{] = 0}$.

If you assume, as we do and as Lichtenstein et al. (1992) does, that the static contractile pressure generated by the diaphragm at a given muscle activation depends only on the diaphragm volume `Vdi`, then `Vdi + Vab = Vsum` implies that the abdominal volume and the diaphragm activation determine the static pressure generated by the diaphragm. But this is inconsistent with the data in Grassino et al. (1978) Figure 4, which shows that the pressure depends also on rib cage volume. This can be accounted for by the fact that a change in volume at the abdominal wall can be reflected in a change in rib cage volume as well as diaphragm volume. Mead and Loring (1982) point out the importance of including the rib cage as one of the movable walls of the abdominal container.

We account for the rib cage as a movable wall of the abdominal container by adding a term, `C1 * Vrc`, to the equation. This allows changes in abdominal volume to be reflected in rib cage volume even if diaphragm volume is constant, though not necessarily 1-for-1, depending on the value of `C1`.

We can determine the value of `C1` by fitting our model to the data. From Grassino et al. (1978) Figure 4, we can read 33 data points over 4 subjects giving diaphragm pressure as a function of rib cage and abdominal volume at 30% diaphragm activation. Using our model equations, we can calculate the diaphragm pressure from the volumes, given a value of `C1`. We can then find the value of `C1` that gives us the best fit of the calculated pressure values to the measured values, and we get a value of 0.369 ± 0.035.

4.4.2 Rib Cage Volume

```
Vrc = VL + Vdi + Vc
```

The rib cage volume, `Vrc`, is the sum of the lung volume, `VL`, the diaphragm volume `Vdi`, and a constant volume, `Vc`. The constant volume represents the mediastinal volume (Vms) plus the pulmonary blood volume and lung tissue volume (Vbt). Cluzel et al. (2000) measured Vms using MRI in 5 human subjects and got an average of 586 ml at TLC and 725 ml at RV. We use the harmonic mean of these, 648, to minimize the percentage error of assuming a constant value. Cluzel et al. (2000) estimates Vbt from the literature at 1100 ml. We use a value of 1108 ml to give the best fit of this rib cage volume equation to measured values from Cluzel et al. (2000), giving a `Vc` value of 1756 ml.

4.5 Tracheal Flow

$$\texttt{VL\_t} = \frac{\texttt{(-(1 + C1) * Vdi\_t - Vab\_t)}}{\texttt{C1}}$$

The tracheal flow is defined internally as the time derivative of the lung volume, `VL_t`, so positive flow is inspiration (but this is inverted for display). The equation is derived by using the rib cage volume equation to substitute `VL + Vdi + Vc` for `Vrc` in the abdominal volume equation, solving for `VL`, and taking the time derivative.

4.6 Airway Resistance

`Rrs = k1 + k2 * |VL_t| + .72 + .44 * |VL_t|`

For the airway resistance we use the widely adopted Rohrer's equation (Hey and Price (1982), Rohrer (1915)):

$$Pressure = K_1 \cdot Flow + K_2 \cdot Flow^2$$

Dividing through by flow gives us:

$$Pressure/Flow = Resistance = K_1 + K_2 \cdot Flow$$

The equation is applied twice, once for laryngeal resistance (`k1 + k2 * |VL_t|`) and once for the resistance of the oropharynx and lower airway (`.72 + .44 * |VL_t|`).

The values .72 and .44 come from Renotte et al. (1998). Their Equation 8 implies that K_2 for the oropharynx is 0 for a constant oropharyngeal volume, and their Table 2 gives .44 for the lower airway, for a total of .44. Assuming K_1 for the larynx is negligible during quiet breathing (see below), Table 2 gives .34 for the lower airway and .38 for the oropharynx, for a total of .72.

The values of k_1 and k_2 for the larynx depend on the diameter of the glottis. We use the following equations:

`k1 = .153 / (d^2 * B^2)`

`k2 = .167 * ((1 - B^2) / B^4 - (1 - B^2))`

`d = min (max (10.9 * (1 + lma), 0), D)`

`D = 18`

`B = d / D`

`D` is the diameter of a human trachea, and the value is based on numbers in Breatnach et al. (1984) and Kamel et al. (2009).

The variable `d` is the diameter of the glottis, or more precisely, the diameter of a circle with the same area as the opening of the glottis. The resting diameter (when `lma` = 0) is taken to be 10.9 mm, based on numbers in D'Urzo et al. (1988), Brancatisano et al. (1983), and Baier et al. (1977), and is assumed to change in proportion to `lma`, based on the results in Tully et al. (1990) and Tully et al. (1991).

The coefficient `k2` is assumed to be proportional to `((1 - B^2) / B^4 - (1 - B^2))`, as in the equation for flow through an orifice in Table II of Simpson (1968). Renotte et al. (1998) Table 2 gives two different values for `k2` for the upper airway, one for inspiration and one for expiration, which we assume is due to changes in `d`. Assuming equal excursions from the resting diameter, we solved for the coefficient that would give us the reported values of `k2`, and got .167. This results in a resting value for `k2` of 0.681.

Using Tully et al. (1990) Table 2 to get the ratio of the mean resting value of `k1` to the mean resting value of `k2`, we applied that to 0.681 to get a resting value of `k1` of .0035 for the larynx, which is indeed small relative to K_1 for the rest of the airway.

For the `k1` equation, we use the Darcy-Weisbach equation for laminar flow, which says that the pressure drop is proportional to $flow/d^4$ (Kreith et al., 2004). This makes it proportional to the `(d^2 * B^2)` used in the equation, because `D` is constant, making `B^2` proportional to `d^2`. Plugging in the resting value of `k1` determined above, and the resting value of `d`, we can solve for the coefficient, which gives us .153.

4.7 Recoil Pressure of the Lung

`sigma_L = (VL - VL0) / CL`

This equation assumes a linear relationship between lung volume and recoil pressure.

`CL = .201` is lung compliance, and the value is the average of the inspiratory compliance of 28 men in Table 2 of Permutt and Martin (1960).

`VL0` is the lung volume at zero recoil pressure, which we take to be equal to the Residual Volume (RV) after a maximal exhalation based on the results in Permutt and Martin (1960) and Harris (2005), and which we calculate as follows:

`VL0 = VLFRC - (VrcKM_FRC + VabKM_FRC) * VC`

`VLFRC = 2.290` is the lung volume in liters at FRC (the end of a passive expiration) from Cluzel et al. (2000), Table 1. The value is the mean spirometric supine lung volume of 5 men.

`VrcKM_FRC = .1282` is the rib cage contribution to lung volume at FRC as a fraction of VC relative to RV from Konno and Mead (1967) Figure 14B. The value is an average over 6 supine human subjects.

`VabKM_FRC = .1282` is the abdominal contribution to lung volume at FRC as a fraction of VC relative to RV from Konno and Mead (1967) Figure 14B. The value is an average over 6 supine human subjects.

`VC = 5.37` is the mean Vital Capacity in liters from Roca et al. (1998) for 6479 men.

4.8 Abdominal Fraction of Rib Cage

```
fa = (Vdi - VdiTLC) / (Vdi - VdiTLC + VL) / (1 + C1) + .15
```

`fa` is the fraction of the rib cage exposed to abdominal pressure.

At TLC (Total Lung Capacity), none of the diaphragm is apposed to the rib cage (Mead and Loring, 1982), so we assume that at all lung volumes, a portion of the diaphgram volume equal to the diaphragm volume at TLC (`VdiTLC`) does not contribute to the zone of apposition. The remainder (`Vdi - VdiTLC`) is divided by the remainder plus the lung volume (`Vdi - VdiTLC + VL`) to give an estimate of the fraction of the rib cage surface above the diaphragm insertions that is exposed to abdominal pressure.

The "obligatory ring" below the diaphragm insertions is always exposed to abdominal pressure. Mead and Loring (1982) estimate this as 15% of the rib cage surface. But their rib cage volume is larger than ours, because they include parts below the diaphragm insertions. They measure rib cage volume by the method of Konno and Mead (1967), which means that a rib cage volume change is equal to a lung volume change with the abdominal volume held constant. From our volume equations, this means that their rib cage volume is larger than ours by a factor of `(1 + C1)`. So we divide the previously calculated fraction of the smaller rib cage by `(1 + C1)` to turn it into a fraction of the larger rib cage before adding it to the `.15`.

4.9 Recoil Pressure of the Diaphragm

```
sigma_di = Phr_d * Pdimax * Ffl * Ffv
           + max (Vdi - VdiFRC, 0) * Kdi_psv
           * (Vdi - VdiFRC)^2 + Rdi * Vdi_t
```

`Phr_d * Pdimax * Ffl * Ffv` corresponds to the Hill muscle model equation A.6 from Ratnovsky et al. (2003), except that `Pdimax` is a pressure rather than a force, and `Ffl` and `Ffv` are functions of volume and its derivative (flow) respectively rather than length and velocity. We can substitute pressure for force, because by Laplace's Law (Laplace, 1808), the pressure is proportional to the force when the curvature is constant, and Braun et al. (1982) found that the curvature of the human diaphragm dome does not change significantly with volume. Also, Kim et al. (1976) found a constant ratio between diaphragm force and pressure in the dog. Substituting volume for length (with an offset) is supported by the numbers in Cluzel et al. (2000), where the relationship between diaphram pressure and length is not clearly different from linear when measured at RV, FRV, and TLC. To the extent that the action of the diaphragm resembles that of a piston (Kim et al., 1976), this is to be expected. Younes and Riddle (1981) provide a precedent for a Hill style model in terms of pressure and flow for the respiratory system.

`Phr_d` is phrenic activation of the diaphragm (see discussion of `Phr_d` in Section 4.1.1).

`Pdimax` is static diaphragm recoil pressure at optimum length and maximum activation in cmH_2O. This parameter is calculated to make `sigma_di` at active TLC equal to the sum of abdominal and lung recoil pressure at active TLC. These pressures depend on the corresponding volumes, which are calculated as described in Volume Parameters.

`max (Vdi - VdiFRC, 0) * Kdi_psv * (Vdi - VdiFRC)^2` is the passive transdiaphragmatic pressure as a function of diaphragm volume. This is taken to be zero at resting diaphragm volume `VdiFRC` (Agostoni and Rahn, 1960), and zero below and quadratic above (Reid et al., 1987). [NOTE — the Grassino data may give a better estimate of where the quadratic starts.] The parameter `Kdi_psv` is calculated to make the passive transdiaphragmatic pressure at `VdiRV` (`PdiRV`) equal to 20 cmH_2O. The value 20 was chosen to be roughly consistent with the results in Siafakas et al. (1979), Grassino et al. (1978), and Agostoni et al. (1966). [NOTE - PdiRV needs a better justification.]

`Rdi * Vdi_t` represents the passive resistance of the diaphragm. The `Ffv` factor in the first term of the `sigma_di` equation is a resistance effect too, but the resistance it represents goes to zero when the diaphragm is not activated, which is not realistic — it would result in any pressure difference causing an infinite flow. So we add a small passive resistance, `Rdi` = 1 cmH_2O/(L/s). The Ffv resistance is, by contrast, about 300 cmH_2O/(L/s) at full activation. [NOTE — need a better justification for the value of `Rdi`.] `Vdi_t` is the time derivative of the diaphragm volume.

4.9.1 `Ffl`

```
Ffl = exp (-0.5 * (((1 - Ldi_min) / Vdi0 * Vdi + Ldi_min - 1.05) / 0.19)^2)
```

`Ffl` is the static pressure-volume relationship of the diaphragm, corresponding to equation A.7 from Ratnovsky et al. (2003), with the relative length replaced by a linear function of volume, as described above. Equation A.7 builds in the assumption that the peak tetanic length is 5% greater than the resting length.

`Vdi0` is the volume of the diaphragm at the resting length, in liters. This is taken to be `VdiRV`, the diaphragm volume at RV (see Volume Parameters), based on the observation in Smith and Bellemare (1987) that the "highest Pdi twitch amplitude was recorded at RV". [NOTE — the optimum twitch length is not necessarily the same as the resting length.]

4.9.1.1 Ldi_min

```
Ldi_min = (VdiTLC - .65 * VdiRV) / (VdiTLC - VdiRV / 1.05);
```

`Ldi_min` is the length of the diaphragm at zero diaphragm volume, in units of the resting length. This parameter is calculated by assuming that the length of the diaphragm at TLC is 65% of that at RV, and that the peak tetanic tension is at RV (both from Smith and Bellemare (1987)). [NOTE — Smith said peak twitch was at RV, which may not be the same as peak tetanic, and for `Vdi0` we assumed that resting length was at RV, which is not the same as peak tetanic.]

`VdiRV` is described in Volume Parameters.

`VdiTLC` is described in Volume Parameters.

`Vdi` is the current volume of the diaphragm, in liters.

4.9.2 Ffv

```
Ffv = 0.1433 / (0.1074 + exp (-1.409 * sinh (3.2 * Vdi_t / Vdi_t_max
                                    + 1.59443531272566456619)));
```

`Ffv` is the pressure-flow relationship of the diaphragm, corresponding to equation A.8 from Ratnovsky et al. (2003) with the velocity replaced by flow, as discussed above. Also, a typo has been corrected – the corrected version is as found in Artemiadis and Kyriakopoulos (2005), Rosen et al. (1999), and Hatze (1981). Also, the value 1.6 has been replaced with the exact value that will make `Ffv` equal to 1 when the flow is zero. [NOTE — adjusting one of the other constants may make more sense - 1.6 is 3.2/2 in Hatze.]

`Vdi_t` is the current rate of change of diaphragm volume.

`Vdi_t_max` = 2.449 is the maximum rate of change of diaphragm volume. This parameter is derived from Fig. 6 in Goldman et al. (1978), which gives transdiaphragmatic pressure as a function of flow at several levels of diaphragm activation up to 45%. Because they held the rib cage still, the flow represents the rate of change of diaphragm volume. Fitting the curves to a Hill-style relation between pressure and flow (Younes and Riddle, 1981), we can find a maximum flow (where the pressure goes to zero) for each level of diaphragm activation. Chow and Darling (1999) Fig 4 implies that the maximum flow increases somewhat linearly to 80% activation and then levels off. That assumption together with the results for the maximum flow at lower activations allows us to compute a maximum flow at 100% activation, which we use for `Vdi_t_max`. [NOTE — we are using the fact that `Vdi_t_max` changes with activation, but in our model, it doesn't.]

4.10 Pressure of the Intercostal and Accessory Muscles

```
Pica = (if (Vdi < VdiFRC) (Vdi - VdiFRC) / (VdiTLC - VdiFRC) * Pdirc
        else 0) * sigma_di
       + u * (Pica_ab_RV
              + (Vrc - VrcRV) / (VrcTLC - VrcRV) * (Pica_ab_TLC - Pica_ab_RV))
```

`Pica` is the effective pressure in cmH_2O generated by the intercostal and accessory muscles, positive values acting to expand the rib cage.

The first half of this expression is the pressure due to the action of the inspiratory intercostals, which are assumed to be inactive when the diaphragm volume is above `VdiFRC` (low lung volumes). Below `VdiFRC`, the pressure exerted by the inspiratory intercostals is assumed to be proportional to the pressure exerted by the diaphragm (`sigma_di`), and the proportionality constant itself is assumed to scale linearly from 0 at `VdiFRC` to its maximum value of `Pdirc` at `VdiTLC`.

The parameter `Pdirc` is calculated as the ratio of `Pica` at TLC to `sigma_di` at TLC.

We calculate `Pica` and `sigma_di` at TLC by solving the model equations for them while assuming TLC conditions: maximal phrenic, zero abdominal, previously calculated values of diaphragm and abdominal volumes (VdiTLC and VabTLC), and zero derivatives. This gives us the intercostal pressure necessary to reach TLC.

The second half of this expression is the pressure due to the action of the expiratory intercostals, which is assumed to be proportional to the abdominal muscle activation (`u`), and the proportionality constant itself is assumed to scale linearly with the rib cage volume, changing from `Pica_ab_RV` to `Pica_ab_TLC`.

`Pica_ab_TLC` = -135 is the pressure due the expiratory intercostals at TLC and maximal abdominal activation. This parameter's value is calculated by taking the mean male maximal mouth pressure at TLC from Ratnovsky et al. (2008) Table 1 and subtracting it from the rib cage recoil pressure at TLC. [NOTE — this neglects the lung recoil pressure, which is less than 1 SD of this value.]

`Pica_ab_RV` is the pressure due the expiratory intercostals at RV and maximal abdominal activation. We calculate this parameter by solving the model equations for Pica while assuming RV conditions: zero phrenic, maximal abdominal activation, previously calculated values of diaphragm and abdominal volumes (VdiTLC and VabTLC), and zero derivatives. This gives us the intercostal pressure necessary to reach RV.

`Vdi` is the current volume of the diaphragm, in liters.

`Vrc` is the current volume of the rib cage, in liters.

`VdiFRC`, `VdiTLC`, `VrcRV`, and `VrcTLC` are parameters calculated as described in Volume Parameters.

4.11 Recoil Pressure of the Rib Cage

```
Prc = log ((Vrc_max - Vrc) / (Vrc - Vrc_min)) / Prc_div + Prc_add
```

The volume of the rib cage is assumed to be a sigmoid function of the recoil pressure of the rib cage, `Prc`. With increasing pressure the volume asymptotically approaches a maximum, and with decreasing pressure it asymptotically approaches a minimum. A generalized logistic function is used for the sigmoid, `Vrc` as a function of `Prc`:

```
Vrc = Vrc_min + (Vrc_max - Vrc_min) / (1 + exp (Prc_div * (Prc - Prc_add)))
```

This equation is then solved for `Prc` to give `Prc` as a function of `Vrc`, which is the equation above for `Prc`.

4.11.1 Vrc_max

```
Vrc_max = VrcTLC + .05 * (VrcTLC - VrcRV)
```

`Vrc_max` is the upper asymptote. This parameter is calculated to be 5% of the range of `Vrc` above VrcTLC.

`VrcRV`, and `VrcTLC` are parameters calculated as described in Volume Parameters.

4.11.2 Vrc_min

```
Vrc_min = VrcRV  - .99 * (VrcTLC - VrcRV)
```

`Vrc_max` is the lower asymptote. This parameter is calculated to be 99% of the range of `Vrc` below RV, so that it is not approached during the normal range of breathing. [NOTE — putting it 5% below resulted in simulation failure. This bears further investigation]

`VrcRV`, and `VrcTLC` are parameters calculated as described in Volume Parameters.

4.11.3 Vrc_div

```
Prc_div = -4 * Crc / (Vrc_max - Vrc_min) / (1 + C1)
```

`Prc_div` is a parameter that controls the maximum slope of the sigmoid, which is the compliance of the rib cage. It is calculated to make the compliance equal to `Crc / (1 + C1)`. The factor of `(1 + C1)` appears because `Crc` is for a rib cage volume defined differently than `Vrc` (see Section 4.8).

`Crc = .110` is the compliance of the rib cage in liters per cmH_2O, an average of values from Table 1 of Gilroy et al. (1985) for three sitting subject.

4.11.4 Vrc_add

```
Prc_add = log ((Vrc0 - Vrc_min) / (Vrc_max - Vrc0)) / Prc_div
```

`Prc_div` is a parameter that controls the location of the sigmoid along the `Prc` axis. It is calculated to make the volume equal to Vrc0 when the pressure is 0.

`Vrc0` is the volume of the rib cage at zero recoil pressure. It is set equal to `VrcFRV` (see Volume Parameters).

4.12 Recoil Pressure of the Abdominal Wall

```
sigma_ab = u * FCEmax * Ffl * Ffv * k * (1/rt + 1/rs)
           + (Vab - Vab0) / Cab + Rab * Vab_t
```

The abdominal wall is modeled as a surface with a circular segment cross-section in each transverse plane, all with the same radius, and a circular segment cross-section in each sagittal plane, all with another radius. The volume of the abdominal wall, `Vab`, is bounded by this surface and by a frontal plane. The values for the sagittal and transverse radius were derived from the results in Song et al. (2006), who measured the radii during insufflation for laparoscopic surgery in humans. Exponential curves were fit to the data points in Fig 3, and the resulting relationship between the fitted sagittal and transverse radii were found to be well approximated by a linear function:

```
rs = 8.00479 * rt - 1.10158
```

The length of the longest transverse chord in the bounding frontal plane was found which gave the volume change stated in the paper, $1.27 \times 10^{-}3$ m^3. The resulting chord length in meters was

```
ct = 0.320496
```

The volume of the abdominal wall is calculated by numerical integration at different radii, to generate a relationship between the transverse radius and the volume:

```
rt = radius (Vab)
```

`u * FCEmax * Ffl * Ffv` is the Hill muscle model equation A.6 from Ratnovsky et al. (2003).

`u` is activation of the diaphragm by the lumbar motor neurons (see Section 4.1.1).

`FCEmax = 33` is the maximal force capacity of the contractile element, in Newtons, for 1.5 cm^2 cross-section of canine external oblique muscle, from Ratnovsky et al. (2003) Table 1.

Using the Law of Laplace (Laplace, 1808), we can convert the tension in the abdominal wall to a pressure. [NOTE — an unduloid would meet the assumptions of the Law of Laplace more exactly.]

`k = .67981067` is a constant to convert from a force in Newtons for 1.5 cm^2 of muscle to a surface tension in meter-cmH2O.

`(1/rt + 1/rs)` converts the surface tension to a pressure, using the Law of Laplace.

`(Vab - Vab0) / Cab` is the passive recoil pressure of the abdominal wall.

`Vab0` is the volume of the abdominal wall at which the recoil pressure is 0. This is taken to be equal to VabFRC (see Volume Parameters).), since we are assuming a supine position.

`Cab = .108` is the compliance of the abdominal wall in liters/cmH2O. Estenne et al. (1985) reported a mean value of 0.088±0.027 in approximately 10 supine men aged 24-39 for diaphragm-abdomen compliance assuming that supine lune compliance is the same as sitting. Cala et al. (1993) gives a value of .196. [NOTE — this needs more work.]

`Rab * Vab_t` represents the passive resistance of the abdominal wall. The `Ffv` factor in the first term of the `sigma_ab` equation is a resistance effect too, but the resistance it represents goes to zero when the abdominal wall muscles are not activated, which is not realistic — it would result in any pressure difference causing an infinite flow. So we add a small passive resistance, `Rab` $= 1$ cmH_2O/(L/s). [NOTE — need a better justification for the value of `Rab`.] `Vab_t` is the time derivative of the abdominal wall volume.

4.12.1 Ffl

```
Ffl = exp (-0.5 * ((LCE / LCE0 - 1.05) / 0.19)^2)
```

`Ffl` is the static pressure-volume relationship of the abdominal wall, equation A.7 from Ratnovsky et al. (2003),

`LCE0 = 19.1` is the resting length of the contractile element, in cm. The value is from Gaumann et al. (1999), an average value for the transversus abdominis of 15 male and 11 female cadavers aged 62-87.

4.12.1.1 LCE

```
LCE = 2 * rt * asin (ct / (2 * rt)) * 100 / 2
```

`LCE` is the length of the contracting element. It is calculated by this equation from `ct`, and from `rs`, `rt`, which are calculated from `Vab` as described above.

4.12.2 Ffv

```
Ffv = 0.1433 / (0.1074 +  exp (-1.409 * sinh (3.2 * LCE_t / VCEmax
                                      + 1.5944353127256645661 9)))
```

`Ffv` is the force-velocity relationship of the abdominal wall muscles, corresponding to equation A.8 from Ratnovsky et al. (2003) A typo has been corrected – the corrected version is as found in Artemiadis and Kyriakopoulos (2005), Rosen et al. (1999), and Hatze (1981). Also, the value 1.6 has been replaced with the exact value that will make `Ffv` equal to 1 when the flow is zero. [NOTE — adjusting one of the other constants may make more sense - 1.6 is 3.2/2 in Hatze.]

`LCE_t` is the velocity of the contractile element, calculated from the time derivative of `Vab`, from the relationships described above.

`VCEmax = 34.7` is the maximal contractile velocity of the contractile element, in cm/s, for canine external oblique muscle, from Ratnovsky et al. (2003) Table 1.

4.13 Volume Parameters

The volumes of the rib cage, lung, diaphragm, and abdomen at RV, FRC, and TLC are calculated as parameters to the model. The names of the resulting parameters are

```
VrcRV  VdiRV  VLRV  VabRV
VrcFRC VdiFRC VLFRC VabFRC
VrcTLC VdiTLC VLTLC VabTLC
```

`VrcFRC`, `VdiFRC`, and `VLFRC` are taken from Cluzel et al. (2000), the average of 5 supine healthy males age 27-33.

`VLFRC = 2.290` is a spirometric lung volume from Cluzel et al. (2000) Table 1.

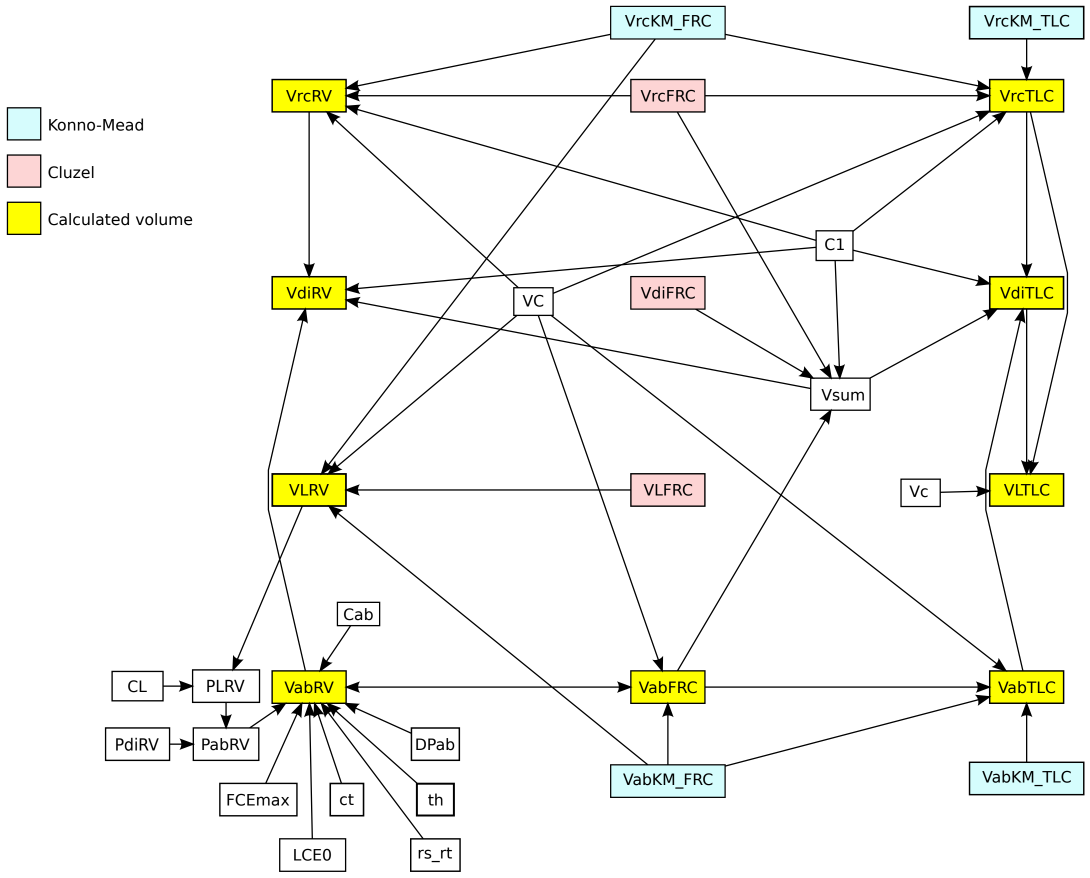
`VrcFRC = 7.013` and `VdiFRC = 2.967` are supine MRI volumes from Cluzel et al. (2000), adjusted to fit with Cluzel's spirometric lung volumes with minimimum implied percentage error and assuming a constant Vms (mediastinal volume) of .648, which minimize the maximum implied percentage error of Cluzel's Vms volumes.

Konno and Mead (1967) Fig 14B gives us the rib cage and abdominal volumes in %VC relative to FRC, average of 6 supine human subjects.

For `VC`, we use 5.37, the mean Vital Capacity in liters from Roca et al. (1998) for 6479 men.

Starting from the Cluzel and Konno-Mead volume numbers, and using the volume equations (Section 4.4), we calculate `VrcRV`, `VLRV`, and `VrcTLC`. Then bringing in the recoil pressure equations, we can calculate the rest of the volume parameters. The following diagram shows the relationships:

➤


VrcKM_FRC
VrcKM_TLC
Konno-Mead
Cluzel
Calculated volume
VrcRV
VrcFRC
VrcTLC
C1
VdiRV
VC
VdiFRC
VdiTLC
Vsum
VLRV
VLFRC
Vc
VLTLC
Cab
CL
PLRV
VabRV
VabFRC
VabTLC
PdiRV
PabRV
DPab
VabKM_FRC
VabKM_TLC
FCEmax
ct
th
LCE0
rs_rt

Bibliography

Agostoni, E., Gurtner, G., Torri, G., and Rahn, H. (1966). Respiratory mechanics during submersion and negative-pressure breathing. *Journal of Applied Physiology*, 21(1):251–258.

Agostoni, E. and Rahn, H. (1960). Abdominal and thoracic pressures at different lung volumes. *Journal of Applied Physiology*, 15(6):1087–1092.

Artemiadis, P. and Kyriakopoulos, K. (2005). Teleoperation of a robot manipulator using emg signals and a position tracker. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1003 – 1008.

Baier, H., Wanner, A., Zarzecki, S., and Sackner, M. A. (1977). Relationships among glottis opening, respiratory flow, and upper airway resistance in humans. *Journal of applied physiology: respiratory, environmental and exercise physiology*, 43(4):603–11. Research Support, U.S. Gov't, P.H.S.,.

Brancatisano, T., Collett, P. W., and Engel, L. A. (1983). Respiratory movements of the vocal cords. *Journal of Applied Physiology*, 54(5):1269–1276.

Braun, N. M., Arora, N. S., and Rochester, D. F. (1982). Force-length relationship of the normal human diaphragm. *Journal of Applied Physiology*, 53(2):405–412.

Breatnach, E., Abbott, G., and Fraser, R. (1984). Dimensions of the normal human trachea. *American Journal of Roentgenology*, 142(5):903–906.

Breen, B. J., Gerken, W. C., and Butera Jr, R. J. (2003). Hybrid integrate-and-fire model of a bursting neuron. *Neural Computation*, 17(12):2843–2862.

Cala, S. J., Edyvean, J., and Engel, L. A. (1993). Abdominal compliance, parasternal activation, and chest wall motion. *Journal of Applied Physiology*, 74(3):1398–1405.

Chow, J. W. and Darling, W. G. (1999). The maximum shortening velocity of muscle should be scaled with activation. *Journal of Applied Physiology*, 86(3):1025–1031.

Cluzel, P., Similowski, T., Chartrand-Lefebvre, C., Zelter, M., Derenne, J.-P., and Grenier, P. A. (2000). Diaphragm and chest wall: Assessment of the inspiratory pump with mr imaging—preliminary observations. *Radiology*, 215(2):574–583.

D'Urzo, A. D., Rubinstein, I., Lawson, V. G., Vassal, K. P., Rebuck, A. S., Slutsky, A. S., and Hoffstein, V. (1988). Comparison of glottic areas measured by acoustic reflections vs. computerized tomography. *Journal of Applied Physiology*, 64(1):367–370.

Estenne, M., Yernault, J. C., and De Troyer, A. (1985). Rib cage and diaphragm-abdomen compliance in humans: effects of age and posture. *Journal of applied physiology (Bethesda, Md. : 1985)*, 59(6):1842–1848.

Gaumann, A., Hoeckel, M., and Konerding, M. (1999). The anatomic basis of the transversus and rectus abdominis musculoperitoneal (tramp) composite flap. *Hernia*, 3:39–41. 10.1007/BF01576742.

Gilroy, R. J., Lavietes, M. H., Loring, S. H., Mangura, B. T., and Mead, J. (1985). Respiratory mechanical effects of abdominal distension. *Journal of Applied Physiology*, 58(6):1997–2003.

Goldman, M. D., Grassino, A., Mead, J., and Sears, T. A. (1978). Mechanics of the human diaphragm during voluntary contraction: dynamics. *Journal of Applied Physiology*, 44(6):840–848.

Grassino, A., Goldman, M. D., Mead, J., and Sears, T. A. (1978). Mechanics of the human diaphragm during voluntary contraction: statics. *Journal of Applied Physiology*, 44(6):829–839.

Grimby, G., Goldman, M., and Mead, J. (1976). Respiratory muscle action inferred from rib cage and abdominal v-p partitioning. *Journal of Applied Physiology*, 41(5):739–751.

Harris, R. S. (2005). Pressure-volume curves of the respiratory system. *Respiratory care*, 50(1):78–98; discussion 98–9.

Hatze, H. (1981). *Myocybernetic Control Models of Skeletal Muscle*. University of South Africa.

Hebb, D. (1949). *The Organization of Behavior*. Wiley.

Hey, E. N. and Price, J. F. (1982). Nasal conductance and effective airway diameter. *The Journal of Physiology*, 330(1):429–437.

Kamel, K. S., Lau, G., and Stringer, M. D. (2009). In vivo and in vitro morphometry of the human trachea. *Clinical Anatomy*, 22(5):571–579.

Kim, M. J., Druz, W. S., Danon, J., Machnach, W., and Sharp, J. T. (1976). Mechanics of the canine diaphragm. *Journal of Applied Physiology*, 41(3):369–382.

Konno, K. and Mead, J. (1967). Measurement of the separate volume changes of rib cage and abdomen during breathing. *Journal of applied physiology*, 22(3):407–422.

Kreith, F., Berger, S., Churchill, S., Tullis, J. P., Tullis, B., White, F., Kumar, A., Todd, J., Chen, J., Irvine, T., Capobianchi, M., Kennedy, F., Booser, E. R., Wilcock, D., Boehm, R., Reitz, R., Kim, J., McDonald, A., Sherif, S., and Bhushan, B. (2004). Fluid mechanics. In Kreith, F., editor, *The CRC Handbook of Mechanical Engineering, Second Edition*, Handbook Series for Mechanical Engineering. CRC Press.

Laplace, P. (1808). *Traité De Mécanique Céleste: Supplément au dixième livre du traité de méchanique céleste sur l'action capillaire*. Number v. 6 in Traité De Mécanique Céleste: Supplément au dixième livre du traité de méchanique céleste sur l'action capillaire. Duprat.

Lichtenstein, O., Ben-Haim, S. A., Saidel, G. M., and Dinnar, U. (1992). Role of the diaphragm in chest wall mechanics. *Journal of Applied Physiology*, 72(2):568–574.

Loring, S. H. and Mead, J. (1982). Action of the diaphragm on the rib cage inferred from a force-balance analysis. *Journal of Applied Physiology*, 53(3):756–760.

MacGregor, R. J. (1987). *Neural and Brain Modeling*. Academic Press.

Macklem, P. T., Gross, D., Grassino, G. A., and Roussos, C. (1978). Partitioning of inspiratory pressure swings between diaphragm and intercostal/accessory muscles. *Journal of Applied Physiology*, 44(2):200–208.

Mantilla, C. B. and Sieck, G. C. (2011). Phrenic motor unit recruitment during ventilatory and non-ventilatory behaviors. *Respiratory Physiology & Neurobiology*, 179(1):57 – 63. ¡ce:title¿Recruitment of Respiratory Motoneurons¡/ce:title¿.

Mead, J. (1956). Measurement of inertia of the lungs at increased ambient pressure. *Journal of Applied Physiology*, 9(2):208–212.

Mead, J. and Loring, S. H. (1982). Analysis of volume displacement and length changes of the diaphragm during breathing. *Journal of Applied Physiology*, 53(3):750–755.

Nail, B. S., Sterling, G. M., and Widdicombe, J. G. (1972). Patterns of spontaneous and reflexly-induced activity in phrenic and intercostal motoneurons. *Experimental Brain Research*, 15:318–332. 10.1007/BF00235915.

Newton, I. (1687). *PhilosophiæNaturalis Principia Mathematica*.

Permutt, S. and Martin, H. B. (1960). Static pressure-volume characteristics of lungs in normal males. *Journal of Applied Physiology*, 15(5):819–825.

Ratnovsky, A., Elad, D., and Halpern, P. (2008). Mechanics of respiratory muscles. *Respiratory Physiology & Neurobiology*, 163(1–3):82 – 89. ¡ce:title¿Respiratory Biomechanics¡/ce:title¿.

Ratnovsky, A., Zaretsky, U., Shiner, R. J., and Elad, D. (2003). Integrated approach for in vivo evaluation of respiratory muscles mechanics. *Journal of Biomechanics*, 36(12):1771 – 1784.

Reid, M. B., Feldman, H. A., and Miller, M. J. (1987). Isometric contractile properties of diaphragm strips from alcoholic rats. *Journal of Applied Physiology*, 63(3):1156–1164.

Renotte, C., Remy, M., and Saucez, P. (1998). Dynamic model of airway pressure drop. *Medical and Biological Engineering and Computing*, 36:101–106. 10.1007/BF02522865.

Roca, J., Burgos, F., Sunyer, J., Saez, M., Chinn, S., Anto, J., Rodriguez-Roisin, R., Quanjer, P., Nowak, D., and Burney, P. (1998). References values for forced spirometry. group of the european community respiratory health survey. *European Respiratory Journal*, 11(6):1354–1362.

Rohrer, F. (1915). Der stromungswiderstand in den menschlichen atemwegen und der einfluss der unregelmassigen verzweigung des bronchialsystems auf den atmungsverlauf in verschiedenen lungenbezirken. *Archiv fur die Gesamte Physiologie des Menschen und der Tiere*, 162:225–299.

Rosen, J., Fuchs, M. B., and Arcan, M. (1999). Performances of hill-type and neural network muscle models–toward a myosignal-based exoskeleton. *Computers and Biomedical Research*, 32(5):415 – 439.

Siafakas, N. M., Morris, A. J., and Green, M. (1979). Thoracoabdominal mechanics during relaxed and forced vital capacity. *Journal of Applied Physiology*, 47(1):38–42.

Simpson, L. L. (1968). Sizing piping for process plants. *Chemical Engineering"*, 75(13):192–214.

Smith, J. and Bellemare, F. (1987). Effect of lung volume on in vivo contraction characteristics of human diaphragm. *Journal of Applied Physiology*, 62(5):1893–1900.

Song, C., Alijani, A., Frank, T., Hanna, G. B., and Cuschieri, A. (2006). Mechanical properties of the human abdominal wall measured in vivo during insufflation for laparoscopic surgery. *Surgical endoscopy*, 20(6):987–990.

Tully, A., Brancatisano, A., Loring, S. H., and Engel, L. A. (1990). Relationship between thyroarytenoid activity and laryngeal resistance. *Journal of Applied Physiology*, 68(5):1988–1996.

Tully, A., Brancatisano, A., Loring, S. H., and Engel, L. A. (1991). Influence of posterior cricoarytenoid muscle activity on pressure-flow relationship of the larynx. *Journal of Applied Physiology*, 70(5):2252–2258.

Younes, M. and Riddle, W. (1981). A model for the relation between respiratory neural and mechanical outputs. i. theory. *Journal of Applied Physiology*, 51(4):963–978.

Index