An Introduction to LabView Virtual Instruments[1]

Biology 390 – Physiology

Introduction: This section of the lab will be your introduction to a very important trend in scientific instrumentation -- the use of computers both to analyze and also to *gather* data from scientific experiments. You are probably familiar to some degree with the use of the computer to analyze data (and the other half of this lab deals with the use of a particular type of program, a spreadsheet, to analyze data) so instead we will dwell on how computers are part of data gathering systems.

The ABCs of Data Collection by Computers:

Basically, to gather data we need the following:

(i) some sort of **collector/transducer** to collect and convert a biological signal into an electrical signal (if it isn't already one);

(ii) a **signal conditioner** -- a device that filters out extraneous signals (sometimes called noise and distortion) and, if needed amplifies the signal of interest. Another way of putting things is that the conditioner improves the **signal to noise ratio**, that is, it increases the conspicuousness of the signal;

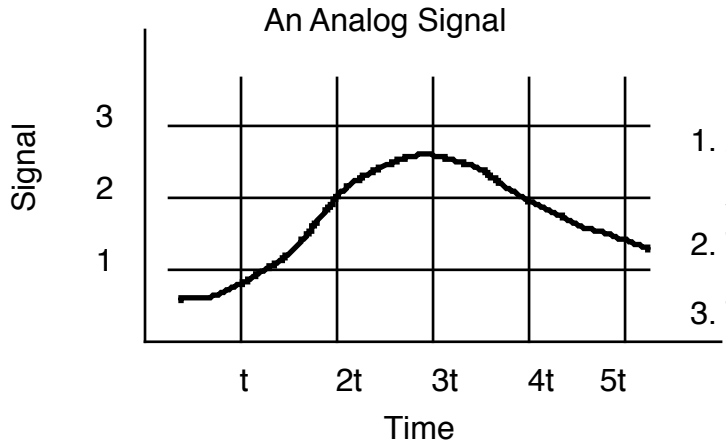
(iii) a **display and storage device**.

It is not necessary that all of these elements always be present. Examples of transducer/collectors are force transducers (often used to convert mechanical action such as a muscle contraction to an electrical signal that is proportional to the mechanical distortion); signal conditions are filters and preamplifiers and display/storage devices include paper and pencil, chart recorders, oscilloscopes, and computers.

Often the signal from a collector/transducer is what we refer to as an **ANALOG** signal. That is, it is a potentially **continuous variable** instead of one that can only have a series of discrete values. The electrical signal from the heart is an example of an analog signal. It is a continuously, smoothly varying voltage.

However, modern digital computers cannot work with analog signals; they can only deal with values that can be represented as exact numbers. Thus, if we are dealing with an analog phenomenon such as the electrical activity of the heart, we must "digitize" it -- that is, accurately represent it as a series of discrete voltages at regular time intervals. A device called an **ANALOG to DIGITAL CONVERTER** (i.e., **A/D converter** (read as **"A to D converter"**)) does this. Essentially, it produces a list of points where for each x is time and y is a voltage that corresponds the magnitude of the event of interest. These can be stored in the computer's memory and then manipulated to make a display that resembles the original signal:

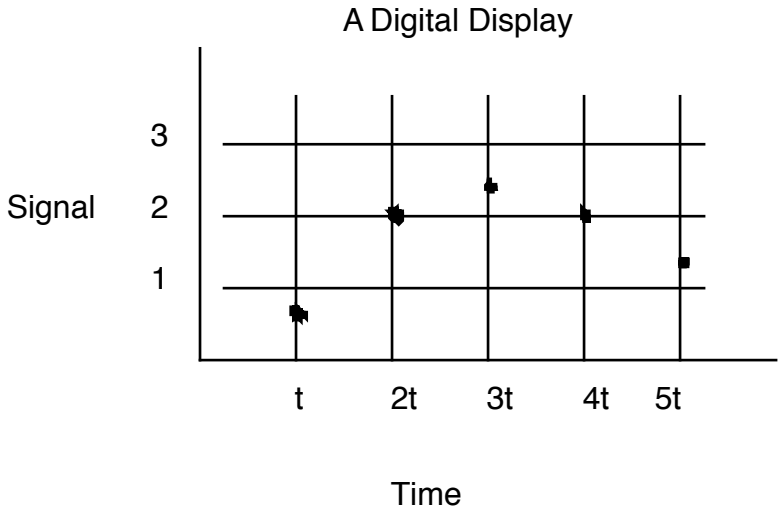
Please see next page



The Conversion to Digital Values

1. At each time Interval **t** (shown as a vertical line), the instantaneous voltage is read.
2. The values between these interval are, therefore, ignored.
3. Thus, we would have these values:

time (x)	voltage (y)
t	0.8
2t	2.0
3t	2.6
4t	2.0
5t	1.4



Notice that the individual points can either be graphed as such or the computer can connect them using any of a number of mathematical techniques -- from as simple as connect the dots to rather complicated averaging curves. Note also that the accuracy of the digital representation of the actual event is largely determined by the **sampling rate** -- obviously high rates of sampling fill in more of the graph and leave less to guessed by the computer program!

Virtual Instruments:

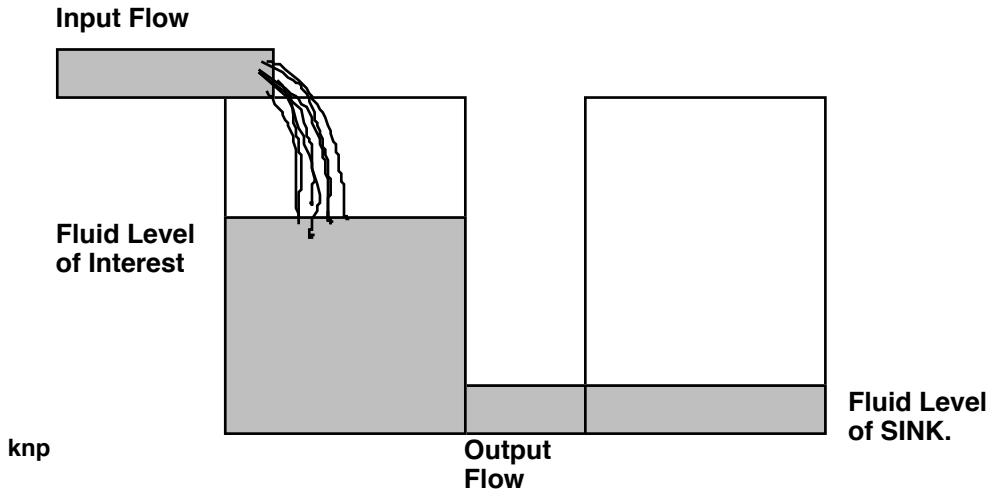
We have just seen how a computer can receive data in a continuous (or discrete) format from device and store or even manipulate and display it. It has recently become possible to design computer programs that gather data and display the data in a format that is similar to that of dedicated instruments such as oscilloscopes. As most of you may remember from Introductory Biology Laboratory, graphically-oriented computers can be used to produce displays that not only look like the **front panels** (**interfaces**) of traditional, dedicated instruments. Furthermore, these displays can be made to include controls that look like knobs etc. that can be turned using a mouse and used to control the way the data are collected, manipulated, and displayed. We refer to such displays that mimic real instruments as **VIRTUAL INSTRUMENTS (VIs)**. We will use virtual instruments throughout this course and you will in fact make a very simple one today as a way of understanding the basics of how they work.

To make virtual instruments, we will use a program called LabView. This a programming language, but it is probably different from any you may know about of have used. It is an example of an **iconic language** -- that is, instead of writing a program using lines of computer statements like "input v$;" this program uses symbols that are wired together. Furthermore, the symbols themselves represent small (or even large) programs themselves -- they do rather complicated things. When a single term or icon stands for a small program these elements are called **objects** and larger programs are created by stringing these objects together, the programming language is called **object-oriented**. LabView 2 is an example of an **object-oriented iconic language.** This type of program is becoming increasingly common and there are many examples of its use medicine. For instance, many of the programs used in medical imaging are built from iconic object oriented programs.

Using LabView: To launch LabView either click on the icon for LabView or click on the icon for the particular VI that you will be using. The instructions below will familiarize you with the basic operation of LabView by taking you through the operation of a virtual instrument that does not gather data from lab equipment but instead is set up as a mathematical model. The particular model we will use is related to our first class -- it is an example of the behavior of an unregulated system in response to a perturbation.

About the Model: this is a very useful model and you will see a number of different versions that become increasingly complex as the semester progresses. In fact, very soon you will be asked to at least come up with the mathematical equivalent of this and more complex models.

This particular model is the flow tank at the end of the first set of class notes. In it water flows into a tank at the top and out at the bottom. You are interested in the **effects of perturbation of the system on the height of the fluid in the tank** and on the flow rates:



The **flow in** will be taken as **a constant that you set** at the start of the experiment. The other **three factors that you can control are**:
(i) initial height (proportional to volume) of the fluid in the system; that is the initial state of the variable you are primarily interested in. In this system
(ii) the height of the fluid in the tank into which the fluid flows from the tank of interest. We will call this tank the **sink**. Its volume is regulated as a constant (we do not need to know how).

Notice that the difference between these two variables sets up a **driving force** (pressure) to move fluid from the tank of interest out into the tank that acts as a sink.

! As you probably remember from physics, pressure in a system such as that above is due to the product of the density of the fluid (ρ) times the height (**h**) times the acceleration due to gravity, G. Any difference in pressure between the two tanks will therefore be due to a difference in **h** since density and G are assumed to be the same for both tanks.

Thus, you can vary the initial driving force by setting the relative values of h for the two tanks.

Now there is one other factor that is important. It is a number that tells how easily fluid gets from one tank to the other. This type of variable is generally referred to as a resistance or **conductance (symbolized as G** -- resistance and conductance are the inverse of each other). The greater the conductance, the easier fluid can flow out of the tank into the sink. Mathematically:

Flow =conductance * (Pressure difference, i.e., pressure gradient)

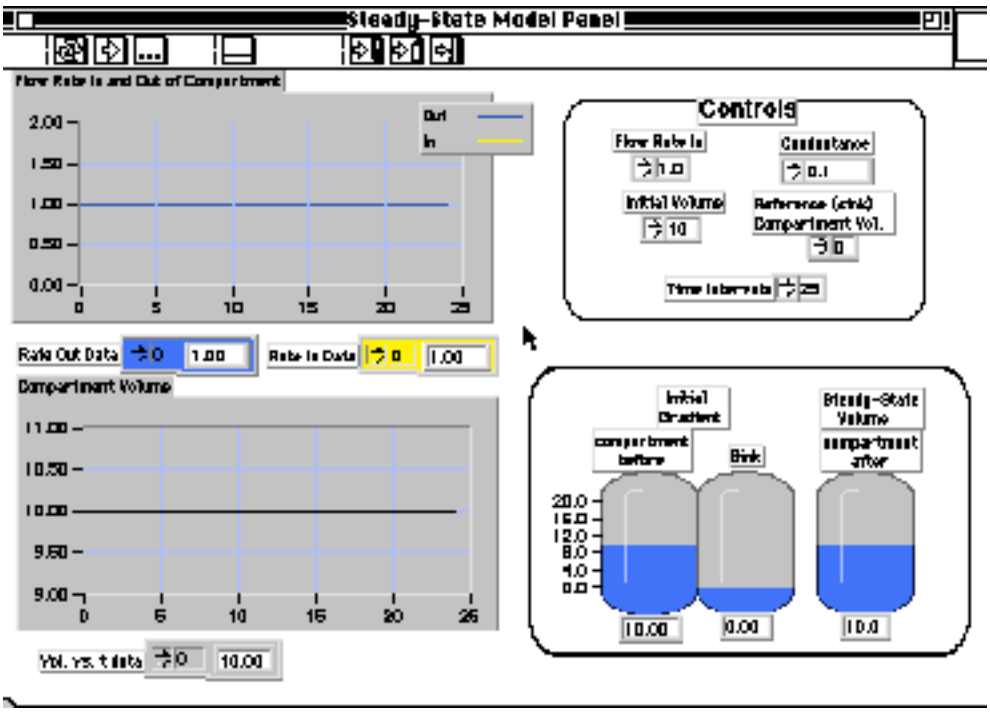
You will also be able to set the value of G as a constant.
In summary, you will be able to control the initial values of (i) tank height; (ii) sink height; (iii) G, (iv) flow into the tank of interest.

You are certainly aware that in most cases the result of changing these values will be that the height of the fluid in the tank of interest will change. However, eventually a steady-state will be achieved. The calculation of the process of reaching steady-state of height with respect to time is very simple. For each time interval the flow out of the tank is calculated based on the pressure difference (height difference in this example) and conductance. This value is then combined with the present volume and the flow into the tank to give the new tank volume. Then the process is repeated. You will have control over how many times this calculation is done (but in most cases the default setting of 25 will be more than enough to achieve a steady-state).

? Questions before you start:
Will a steady-state represent regulation?
Which of the factors (tank heights, g) can most easily be viewed as the value of some physiological variable within an organism? Which represents the environmental value? Must it always be seen this way?
Which variable(s) represent(s) processes wherein regulation might be invoked to achieved a homeostatic response?

Using the LabView Model:

1. Launch the icon named "**Steady-State Model**".
You will see a front panel to a modeling VI that will look roughly like this:



The panel is divided into four areas: two graphs that show the volume or flow rates (in and out) with respect to time, a fluid tank model showing the gradient at the start (left) and the final volume of the tank of interest (right) and a set of controls.

About the Controls: The values of each of the controls are under some constraints (for instance, volumes may not be greater than 20 or less than 0). However, **you may change any of the value by one of two different operations**:

(a) using the "clicker" arrows to increase or decrease the displayed values.

(i) First be sure that the cursor you see on the monitor looks like a hand with a pointing index finger. If it doesn't (most likely because it looks like an opened hand) select the pull-down menu **Tools** and you will see the tool pallet:

You can move this pallet into permanent view by holding the mouse key down while you are on the top of the pallet box and then dragging the pallet to wherever you wish.

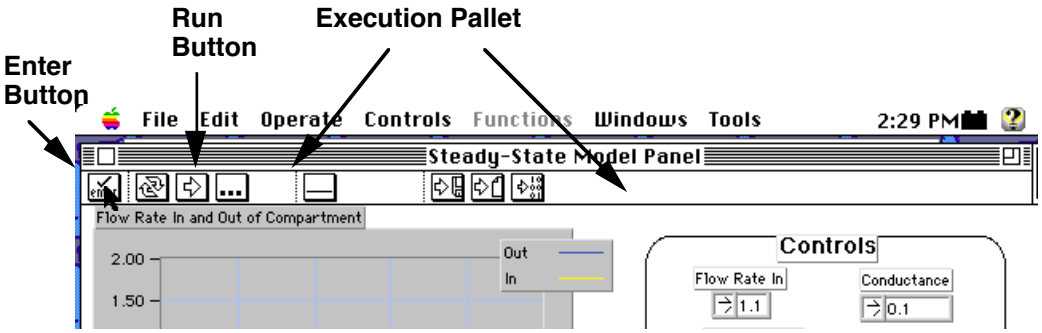
(ii) Now select the appropriate tool (the "pointing finger" -- no joke intended) and then go to the control whose value you wish to change.

(iii) drag the mouse (button down) over the indicator until it changes color

(iv) type in the value you wish followed and then either:

(a) hit the **enter** key (on the keyboard) **or**

(b) click the **enter button** on the **execution pallet** (upper left):



Running the Program: Once you have decided what values you wish to use, you can run the program by simply **pressing the forward arrow icon** (**Run Button**) on the execution pallet:

the program will then execute and you can inspect your results and decide what to do next.

Alternatively, you can press **cmd-r** (command and r at the same time -- the command key has the ⌘ on it) or go to the **Operate** pull-down menu and select **Run VI**.

Note: Sometimes you will want to get all of the control settings back to their initial values (the ones they had before you started working with the VI). To do this go to the pull-down menu item called **Operate** and select "Set all to default values":

(please see next page)

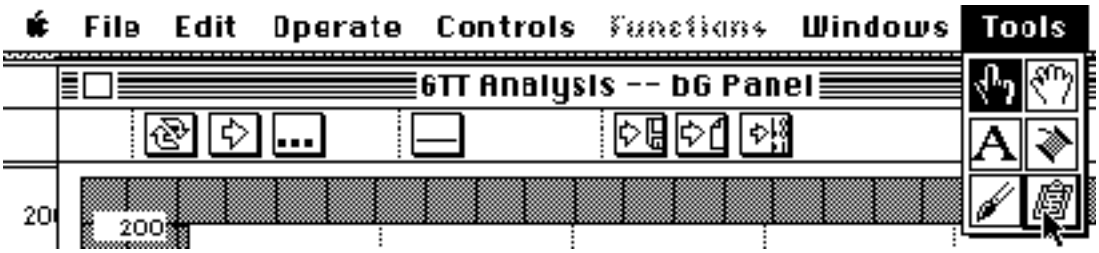
IMPORTANT: **Please** ***never make your current values default***.

To Make Comparisons of the Results of Different Runs: It is often useful to be able to compare, graphically, the results of different runs. To do this, there is a second VI that you must use. Type **cmd-o** (for open) and then use the dialog box to find the VI in the **models** menu that is called **S-S**

Comparisons. Double click on it to launch it. Once it is up, **go to the "Windows" menu item and move back to the Steady-State Simulation.**
To transfer data for comparison purposes, do the following:

Note: the pictures in the example below uses a different VI than S-S comparisons but the principle is the same.

(i) Get the clipboard from the TOOLS menu



(ii) the mouse pointer will become a cool looking little clipboard if you did this correctly.
(iii) go to the array that you wish to analyze, in our exercise this will probably be the volume data array.
(iv) while the clipboard is positioned over that array, press the OPTION KEY. You should see an arrow that points towards the clipboard appear. **If you see this arrow, appear, Click the mouse**. In a moment the clipboard will flash at you -- that means the data has been transferred. If this doesn't happen, try again.
(v) Go to the WINDOWS MENU and select the S-S Comparisons panel program**:**

You will now see another program containing a graph and three arrays where data can be entered from a clipboard.

(v) TO ENTER DATA

(a) simply move the clipboard tool over one of the arrays.

(b) If the clipboard contains data, an arrow pointing off the clipboard will appear as soon as it is over an array.

(c) **CLICK THE MOUSE TO COPY THE DATA TO THE ARRAY**. Note: you do not need to option click to do this.

(d) The clipboard icon will flash and the data in the first element of the array will appear

! Arrays are lists of data (here volumes for each instant dt in time), the lists you will see are called one dimensional arrays. The elements start with number 0 up to n-1 where n is the total number of array elements. To see the values, click on the arrows on the first number of the array (the index, n) and the other display will give the value for that position in the array.

(vi) To view the data graphically, simply run the program by pressing the forward arrow.
(vii) Use the **WINDOWS MENU** to get back to the **Steady-State model** simulation.

(viii) Repeat this process to enter arrays from other simulation runs. You can over-write any data by simply entering data over what is already in the array.

Now work with the model and see if it you understand how steady-states are achieved.

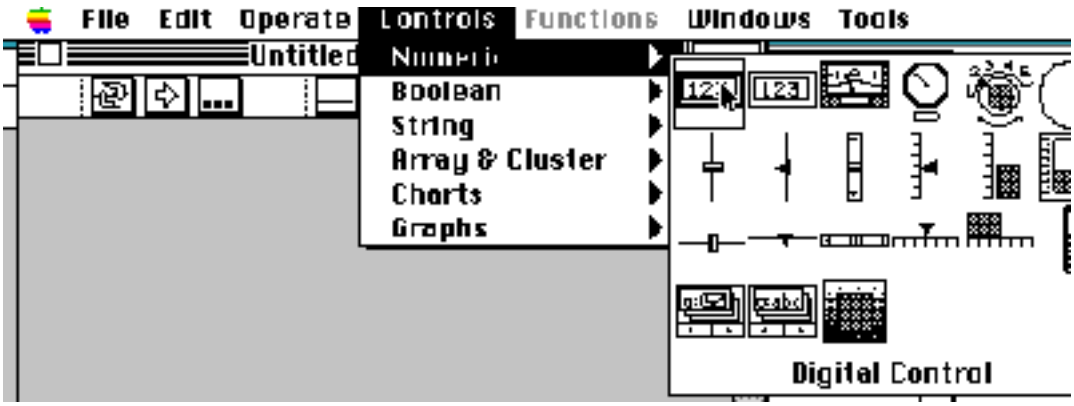
General comments about using models: Carefully think about what to do to fully investigate the situations where this model might give different general types of results. Do not be superficial.

Programming in LabView: We are going to make (together) a simple LabView Program as a way of you learning how LabView and other data gathering programs work. You will then write a somewhat more complicated one. Both are calculators -- something that LabView can do but not something that we would normally program it to do.

First you will need to start up LabView by clicking on its icon, if it is not already running. You will see an untitled front panel. If you are already running LabView and you need a new front panel, type cmd-n (for new).

Now you need to put some controls and indicators on the panel. We are going to make an adding machine that also takes the square root of whatever it adds. You will need two inputs and two outputs on the front panel (the inputs for the numbers to be added and the output for the result and for the square root of that result). To get these, go to the item on the LabView menu called **Controls** and pull it down. Then slide over the menu item called **Numeric** and then slide to the first item there. You will see its name (**Digital Control**) appear at the bottom of the box:

(please see next page)



Slide over to the object that looks like it just to the right -- this is an **indicator**. Thus, one of these objects can serve as an input (control) the other an output (indicator).

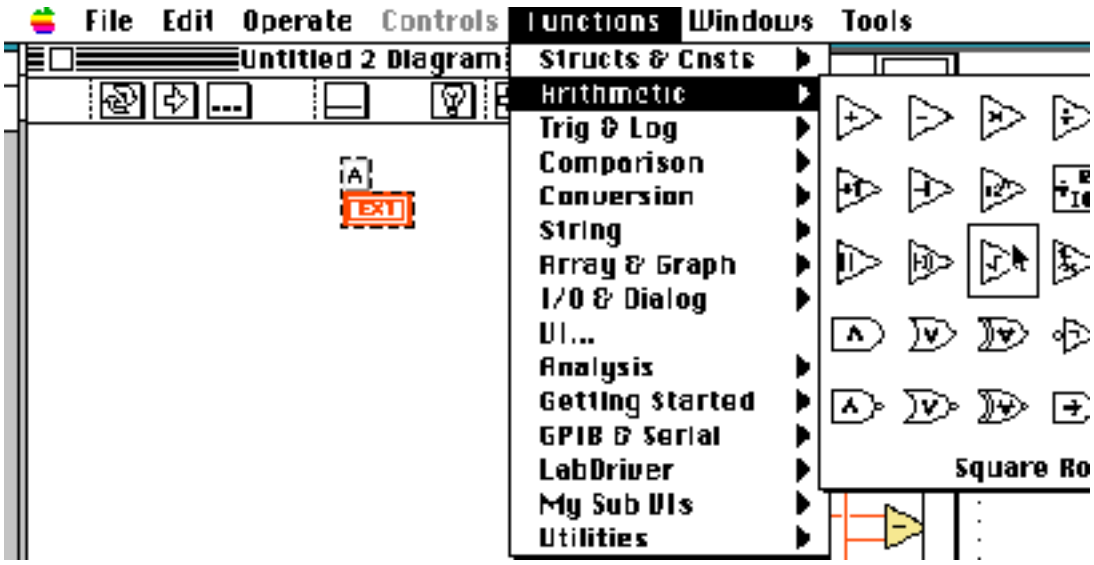
Select the digital control. When you release the mouse button it will appear on the front panel. Give it a name like **A**. To do this, be sure you have the pointer tool (from the tool menu or by hitting the "tab" key) then type in A. If you loose the box to type in the name (label box) , **click using the pointer tool on the indicator while you hold the option key down (this is called**

option clicking and is an important operation in LabView). A dialog box will appear where show label is one item; select it.
Now add another input label it **B**, then two outputs labeled **sum** and **sqrt.**
Using the **hand tool** you can now move the various controls and indicators around to some arrangement that you like.

We now have to add the actual program. This is done on the back panel. To make it appear, type **cmd-f** (hold the command and f key down at the same time). When it appears you will see the terminals for the controls and indicators with their labels showing. Move them using the hand tool if you wish.

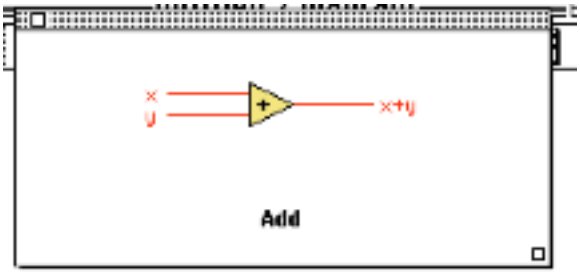
Now, for operators, go to the **Functions** menu item and slide down to **arithmetic**. Select either the addition or square root operator. Then get the other one:

(please see next page)



You now must move data from the controls to the operators to the indicators. This is done by wiring -- the wires indicate data flow. However, wiring must be done carefully -- not surprisingly, the correct wires must go to the correct terminals. To see the terminals on any item in your program, you must first access a **help window**. This is done by pressing **cmd-h**. Next, you must have the wiring tool, which you select from the **Tool** menu -- it looks like a spool of wire; when selected the cursor will also look like a wire spool:

If you run this tool (**without clicking**) over any terminal (for instance the addition operator) you will see a wiring diagram of that operator, complete with its terminals appear:



Try this also for the square root operator (there are only two terminals -- in and output) and the indicator terminals (nothing happens -- the entire surface is a terminal).

Wiring: Using the wiring tool, simply click on the item you wish to wire from and then move the mouse to the item you wish to wire to (**do not hold the mouse key down)**. When you get to the correct terminal, **double click to stop the wire**.

Wiring Notes:

Turns: you may wish to make an orderly route to your wiring. To do this, any time you wish to make a turn, simply click as you wire

Goofs: double click to stop and then select the wire by clicking on it and hit the **delete key**. Or you can type **cmd-w**.

Broken Wires: Sometimes you will finish wiring and you will get a dotted wire. Remove as above. There are many reasons this can happen but one of the most common is that you have wired something incorrectly. One example would be to wire one of your output terminals to the input side of an operator such as the square root icon. Try it. LabView recognizes that the indicator cannot act as a data source unless it is wired to something else that is actually the data source; thus it will not allow the indicator to be wired to the sqrt input side as if it were a source.

Go ahead and complete your diagram. Then go back to the front panel either by clicking on it or by hitting **cmd-f** (this lets you toggle back and forth between the front and back panels).

Try your calculator out.

In-class exercise: Write a program that will solve an equation of the form $ax^2 + bx + c = 0$ for x. Obviously, this simply involves solving the quadratic equation.

Hints: **1.** Your **front panel** will need **inputs** for a, b, and c (label each) and two **outputs** (x1 and x2 -- recall the solution to the quadratic equation).
2. Be organized about your back panel. Solve items such as the square root term first, then obtain the two solutions for the numerator before finally calculating the two overall answers.
3. You will find a VI for exponentiation under the TRIG and LOG submenu, look for one that exponentiates x to the y. Be sure to correctly identify the terminals for the exponent and base inputs.

We will try one additional example of programming in future weeks when you learn how to program an actual VI that reads and scales a voltage from a pH meter.