PHYC 500: Introduction to LabView

M.P. Hasselbeck, University of New Mexico

Exercise 4 (v 1.2)

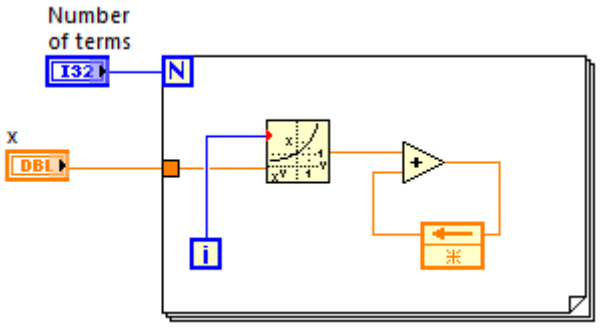
For Loop

The For Loop is similar to the While Loop except that it will not execute indefinitely; it stops after a specified number of iterations. To understand the operation of a For Loop, a VI will be written to evaluate a truncated geometric series:

$$f(x) = \frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$$

for $|x| < 1$. Open a blank VI, go to the Block Diagram, and select a For Loop from the Structures palette. The Run button is immediately broken because the number of loop iterations has not been specified. Right-click on the **[N]** terminal on the upper-left corner of the structure and create an integer control; this will specify the number of terms in the series. There are a variety of clever approaches to calculate each term. An easy way is to select Mathematics: Elementary & Special Functions: Exponential: Power of X (with label **x^y**) and place it in the For Loop. Right click on the **x** input terminal and create a control. Drag this DBL icon outside the For Loop and re-wire it. The exponent is implemented by wiring input **y** to the iteration terminal **[i]** at the lower-left corner of the For Loop. The first value of the loop counter is always 0, which creates the desired first term $x^0 = 1$. It is important to remember that the iteration terminal always starts counting at 0 for both While and For Loops.

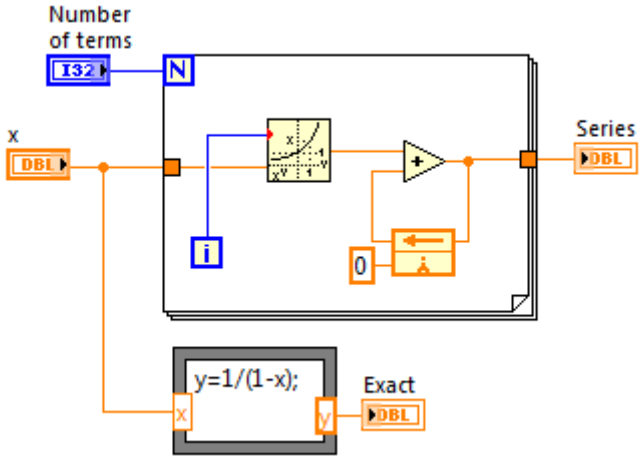
Next we need a mechanism to add each term in the series. One way to do this is with a Feedback Node. Wire the output of the **x^y** function to one terminal of a Sum operation. Loop the Sum output back into the other input; a Feedback Node should automatically appear as shown below:



This operation can also be placed in the Block Diagram directly via Structures: Feedback Node. It's important to initialize the summing operation. For the Taylor Series it will be zero. Right-click on the asterisk (*) on the Feedback Node and create a constant. It should show the default value of zero.

Note: The Feedback Node will retain its last value in volatile memory. When the VI is re-started, it will initialize the Feedback Node to this stored value *unless* the * terminal has been set to a desired starting value.

Create a floating-point numerical indicator for the Taylor Series and place it outside the For Loop. When you attempt to wire the sum operation to this indicator, however, a broken wire will appear. The reason is that the default output of this For Loop tunnel is a 1-D array corresponding to the incremental construction of the series, i.e. each term in the updated sum. We are only interested in the final result on the last iteration. To prevent the For Loop from producing an array output, right-click on the output tunnel and select Tunnel Mode: Last Value. This disables indexing at the tunnel. You can now wire directly to the single-value numeric indicator as seen in the diagram:



Also shown is an exact evaluation of *f(x)* implemented with a formula node; inputs and outputs are created by right-clicking on the edge of the structure. You can alternatively use a combination of numeric operations for the calculation. The two results can be compared to determine how accurate the truncated Taylor Series approximation is. Verify that more terms are necessary to maintain accuracy as $|x|$ approaches 1.

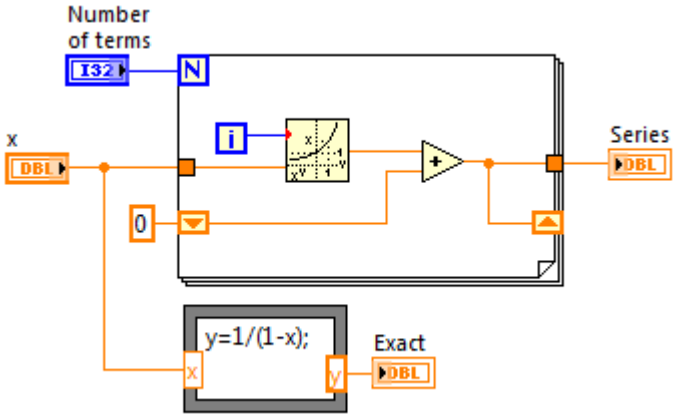
Comparison of While and For Loops

i) Both begin counting at index 0.
ii) While Loop must execute at least once; For Loop will *not* execute if N=0.
iii) While Loop has indexing of output data disabled by default; only the *last* value of the loop operation passes through the output tunnel.

iv) For Loop has indexing of output data enabled by default; values from *every* iteration are passed through the output tunnel, generating an array.
v) The default indexing behavior can be changed by right-clicking on the output tunnel and selecting Tunnel Mode.

Shift-Register

The Shift-Register is a more general implementation of a Feedback Node. Here, a direct replacement can be made. Right-click on the Feedback Node and select "Replace with Shift Register". The Shift Register appears as a pair of down/up arrows on the left/right edges of the For Loop as shown in the following Block Diagram.



As with the Feedback Node, the output of iteration number $i-1$ becomes the input for iteration i. The Shift Register is initialized for iteration $i = 0$ by placing the constant 0 on its left terminal. In this example, this initializes the sum to 0. If the Shift Register is not initialized, it will start with the value it has its in memory, i.e. the value of the last iteration of the VI.

Save your working VI.

The essential idea of a Feedback Node and Shift Register is to make values from previous iterations of a loop available to the current iteration. You can access data from the $i-2$ iteration by clicking on the Shift Register and selecting "Add Element". If you do this again, data from the $i-3$ iteration is available and so on. The Shift-Register and Feedback Node work in both the For Loop and While Loop. Their operation is essentially identical and deciding which one to use is a matter of personal preference. Most LabView programmers use Shift-Registers because it more flexible and the code is easier to read.

Make an Interactive VI

Use the concepts of While Loop, Event Structure, and Case Structure developed in the previous exercises to make the Taylor Series VI continuously interactive. Place the Block Diagram code inside an Event Structure and surround this with a While Loop. Evaluate the loop when a value change occurs at the controls x or **Number of Terms**. Also, check for the condition $|x| < 1$ and warn the user if this is violated.

Tip: If needed, the For Loop can be configured to stop before the specified number of iterations is reached. Right-click on it boundary and select the Conditional Terminal. Its function is the same as in a While Loop. Note that the different appearance of the count terminal on the upper-left corner (see below).