LabVIEW 1: Loops, Graphs, and Mathscript

Reading :

Hands-On Introduction to LabVIEW by J. Essick

Reading	pages
Chapter 1	All
Chapter 2	All
Chapter 3	first few pages

Main focus :

- Learn how to use and program time-efficiently the *Front Panel* and *Block Diagram.*
- Know how to use *While Loop, For Loop, Waveform Chart*, and *Waveform Graph.*
- Know how to put an input and output terminal on a *mathscript* node and how to program a simple arithmetic operation, such as output = intput/number.

Most relevant VI's : None

Due Date: Friday, 11AM, Nov. 7th.

Hand in a printout of the *Block Diagram* and *Front Panel*, where the *Front Panel* graphs should display relevant data. Make sure to include your name and date and most importantly sufficiently detailed comments. The comments on the *Block Diagram* should concern the details of how the program works, and why you chose to use certain LabVIEW language programming structures. The comments on the *Front Panel* can be used to talk about the meaning of the results.

Assignments:

1. Write a VI (LabVIEW program) that functions as a stopwatch with precision of 0.01 seconds.

2. [Chapter 1, Page 29] Write a VI that performs the following task N times in a row: Execute the **Wait(ms)** icon with **milliseconds to wait** equal to 1, then store the resultant value of **millisecond timer value**. After this sequence of operations, make your program plot the resultant element array of **millisecond timer value**s. Run your program with N large (e.g. $N > 500$). Does the resulting plot meet your expectations? You may consider plotting the difference between this and your expectation. To make the time start with zero consider taking the an initial reading of the millisecond timer value and subtract that from all N array values. Try moving the mouse, typing on the keyboard, or switching from the front panel to block diagram while your program is running. Anything unusual happen on the plot that your program produces?

Intro to Problem 3 - Random Numbers: One way to think of a sequence of random numbers ($X = x_1, x_2, \ldots$) is as a particular realization of a process where numbers are drawn one after the other according to a probability distribution which, in turn, can be characterized by its probability density function $p(x)$. For example, a uniform probability distribution simply means that the probability density function is a constant:

$$p(x) = c = const. \qquad \forall x \in [a, b] \in \mathbb{R} \tag{1}$$

and $p(x) = 0$ for all other arguments, i.e. $x \notin [a, b]$. A probability distribution has to be properly normalized, therefore

$$1 = \int_a^b p(x)\,dx = c\,(b-a) \quad \rightarrow \quad p(x) = \frac{1}{b-a} \tag{2}$$

One way to characterize probability distributions is in terms of moments. For us only the first two of them are of interest for now. That is, we will be looking at the mean and variance. The mean μ (also called the expected value $E(X)$) is given by the first moment about 0.

$$\mu = E(X) = \int_a^b (x-0)\,p(x)\,dx. \tag{3}$$

For the unifrom distribution this means

$$\mu = E(X) = \frac{1}{b-a}\int_a^b x\,dx = \frac{b^2-a^2}{2(b-a)} = \frac{b+a}{2} \tag{4}$$

If, as usual, $a = 0$ and $b = 1$, meaning that we draw uniformly distributed numbers between 0 and 1, then the mean value is 0.5. This should not come as a surprise. A maybe less obvious question is what the variance or standard deviation of a sequence of random numbers should be that is drawn from a uniform distribution.

The variance σ^2 is defined as

$$\sigma^2 = E([X-\mu]^2) = \int_a^b (x-\mu)^2 p(x)\,dx. \tag{5}$$

You should convince yourself that the following is true

$$\sigma^2 = E([X-\mu]^2) = E([X-E(X)]^2) = E(X^2) - E(X)^2 \tag{6}$$

Thus, it is possible to determine the expected variance based on a calculation of the second moment around 0

$$E(X^2) = \int_a^b x^2 p(x)\,dx. \tag{7}$$

The standard deviation is then simply the square root of the variance

$$\sigma = \sqrt{\sigma^2}. \tag{8}$$

3. (a) Write a program that generates an array of random double-precision floating point numbers that have a uniform distribution in the range 0 to 1. Place on the front panel a control called **Number of Samples** that will allow the user to enter a value for the size of the array. Then calculate and display the measured mean and standard deviation on the a front-panel alongside the *theoretically expected values* for the mean and standard deviation. Make sure to change the front-panel indicator's **Digits of precision** to be at least 4. How large does the number of samples have to be to come consistently close to the expected values (*i.e.* within 2%)?

You will find the following two icons helpful **Random Number (0-1)** in **Functions⇒Programming ⇒Numeric** and **Standard Deviation and Variance** in **Functions⇒Mathematics⇒ Probability&Statistics**.

(b) Save your previous program and then create another program that also looks at the mean of random numbers (you might want to use **Save As** with the **Substitute For Original** option). In this program we take a different view of what we did in (a) - we view it as characterizing a measurement process. We might interpret what we did in (a) as taking a single measurement of a quantity that is equal to 0.5 and corrupted by uniform noise, *i.e.* noise with some uncommon characteristics. To characterize the uncertainty of a single measurement process we needed to take many samples in order to be able to look at the statistical distribution and determine the standard deviation. Now lets characterize what happens if you are a diligent student and decide to perform the experiment N times, and to take the result to be the mean of those N trials. To get some statistical information let us perform this experiment, *i.e.* taking the mean of N samples, 1000 times in a row and calculate the standard deviation [of the means] and for fun also the mean [of the means]. To be clear – do not do this by hand, but automate the process of running the experiment 1000 times by using an appropriate loop structure to generate an array of 1000 numbers from which to determine μ and σ. Again, display the measured standard deviation (and mean) on the a front-panel.

Note that the case N=1 in (b) should give you the same answer that you obtained in (a) with N=1000, your uncertainty as characterized by the standard deviation is big. What happens for N>1? Can you guess the dependence of the standard deviation on the number of samples N (which is fixed for the whole run of 1000 experiments)? Test your prediction by displaying the theoretically expected values for the standard deviation alongside the measured one. One option is to use the mathscript node to calculate the expected standard deviation.

For those of you who like graphics, you can visualize the distribution by utilizing the **Histogram** function in **Functions⇒Mathematics⇒Probability&Statistics**.