Experiments on Experiments, Determining Uncertainty for Complex Experiments using Matlab.

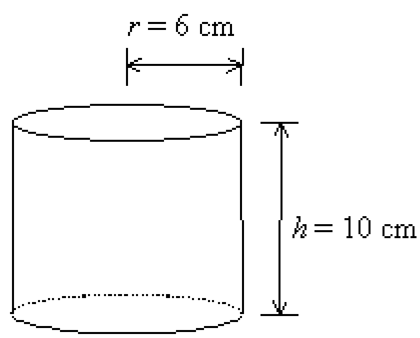
R. Fithen, PhD
Arkansas Tech University

Abstract

Often in engineering labs and experiments students expect result much like textbooks. They expect one answer to a complex problem even if the experiment were run hundreds of times. This transition to the real world through lab experiences can cause students to lose some faith in the fundamentals they gain in their course work. Usually these differences are explained through a few key lectures on systematic and random errors in experiments. However, once the professor begins his or her lecture on this topic the student experience a level of brain freeze due to the existence of numerous partial differential equations on the board. In this paper I describe a numerical approach to circumvent this brain freeze through the use of Matlab's random normal distribution function. Through a simple example of volume of a cylinder[1] we will show how the method works and comes to the same solution for the random errors. In addition, this Matlab based method can be used on incredibly complex experiments without requiring the use of any differential equations. It is worth noting that exchanging the use of differentials for programming a Matlab code is not always favorable to students in that it still requires work and understanding of the experiment on their part.

Random Errors

This section is a direct copy of Dr. Polik's web site[1]. I use his web site in class to show how random errors are propagated through the calculation of the volume of a cylinder.



In his example there is an assumption of an error in measurement of 0.1 cm in both the radius and height of the cylinder. The volume of the cylinder is calculated as:

$$\begin{aligned} V &= f(h,r) \\ &= h\pi r^2 \\ &= (10\ \text{cm})\,\pi\,(6\ \text{cm})^2 \\ &= 1131\ \text{cm}^3 \end{aligned}$$

The random error as a result of this 0.1 cm uncertainty of both the radius and height becomes:

$$\begin{aligned} s_V &= \sqrt{\left(\frac{\partial V}{\partial r}\right)_h^2 s_r^2 + \left(\frac{\partial V}{\partial h}\right)_r^2 s_h^2} \\ &= \sqrt{(h\pi 2r)^2 s_r^2 + (\pi r^2)^2 s_h^2} \\ &= \sqrt{[(10\text{ cm})\,\pi 2(6\text{ cm})]^2 (0.1\text{ cm})^2 + [\pi(6\text{ cm})^2]^2 (0.1\text{ cm})^2} \\ &= \sqrt{1{,}421\text{ cm}^6 + 128\text{ cm}^6} \\ &= 39\text{ cm}^3 \end{aligned}$$

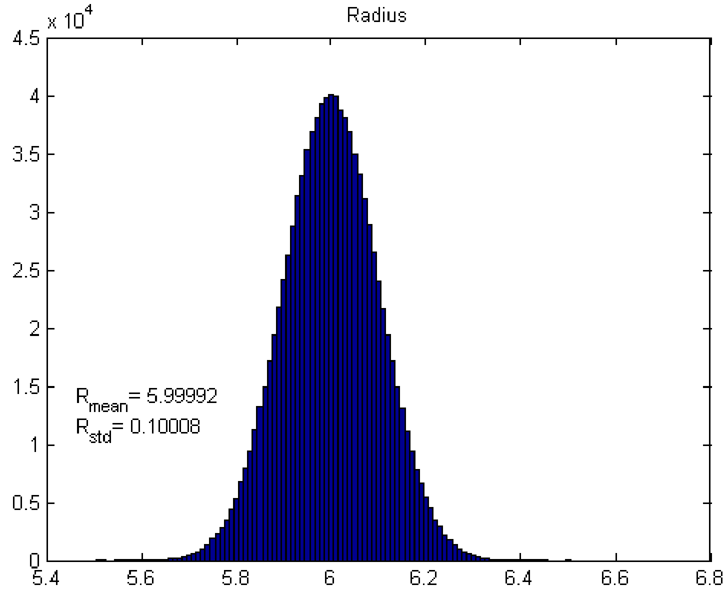
Therefore the final result of the volume becomes V=1131 ± 39 cm^3. Details of this calculation can be seen at the web site of Dr. Polik and the text by Holman[2].

Matlab Approach

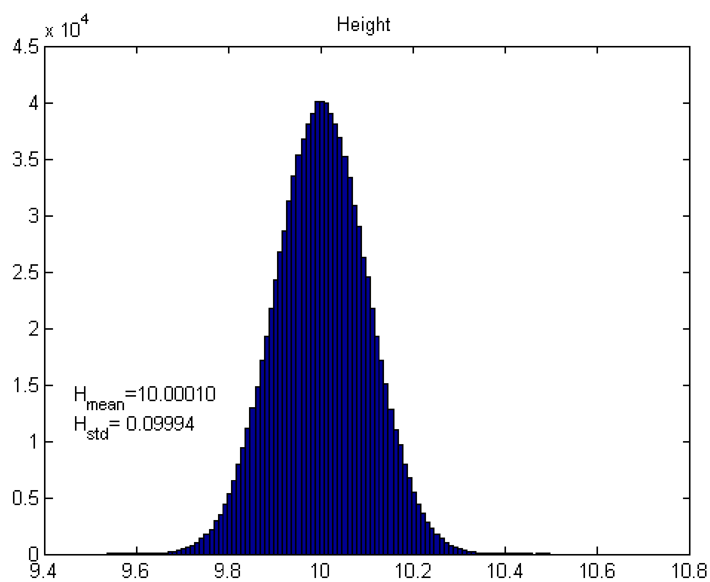
Use of Matlab for this calculation is incredibly simple. A listing of the Matlab code is shown below, and a description of the code follows.

```
% Written by Dr. Fithen
% Arkansas Tech University
N=1000000;
% Radius Setup
x = 5.5:0.01:6.5; %setup x axis on histogram
r = 6+.1* randn(N,1); %generate random normal distribution
figure(1)
hist(r,x) % create histogram
title('Radius')
mr=mean(r);
Sr=std(r);
S=sprintf(' R_{mean}=%8.5f\n R_{std}=%8.5f',mr,Sr);
text(5.45,N/80,S,'HorizontalAlignment','left') % print results on plot
% Height Setup
x = 9.5:0.01:10.5; %setup x axis on histogram
h = 10+.1* randn(N,1); %generate random normal distribution
figure(2)
hist(h,x) % create histogram
title('Height')
mh=mean(h);
Sh=std(h);
S=sprintf(' H_{mean}=%8.5f\n H_{std}=%8.5f',mh,Sh);
text(9.45,N/80,S,'HorizontalAlignment','left') % print results on plot
% Results
Volume=pi*h.*r.^2; % Calculate results for ALL data points
L=min(Volume);
H=max(Volume);
Bin=(H-L)/100;
x=L:Bin:H;
figure(3)
hist(Volume,x)
Title('Volume')
mVolume=mean(Volume);
SVolume=std(Volume);
S=sprintf(' Volume_{mean}=%8.4f\n Volume_{std}=%8.4f',mVolume,SVolume);
text(L,N/40,S,'HorizontalAlignment','left')
```

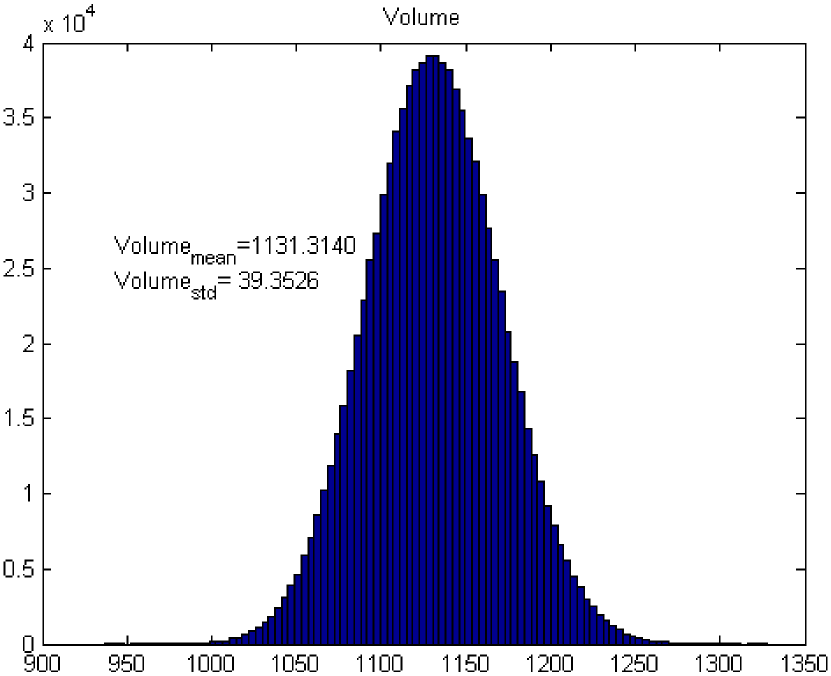
To begin with we must specify how many numerical trials we wish to run (`N=1000000;`). In this example we specify one million. The higher this number the better our results will fit a normally distributed results and therefore become closer to the results of the results of Dr. Polik. The line of Matlab code (`r = 6+.1* randn(N,1); %generate random normal distribution`), will generate normally distributed set of radiuses. The mean of this distribution should be 6 and the standard deviation should be 0.1, just like in Dr. Polik's example (see below);



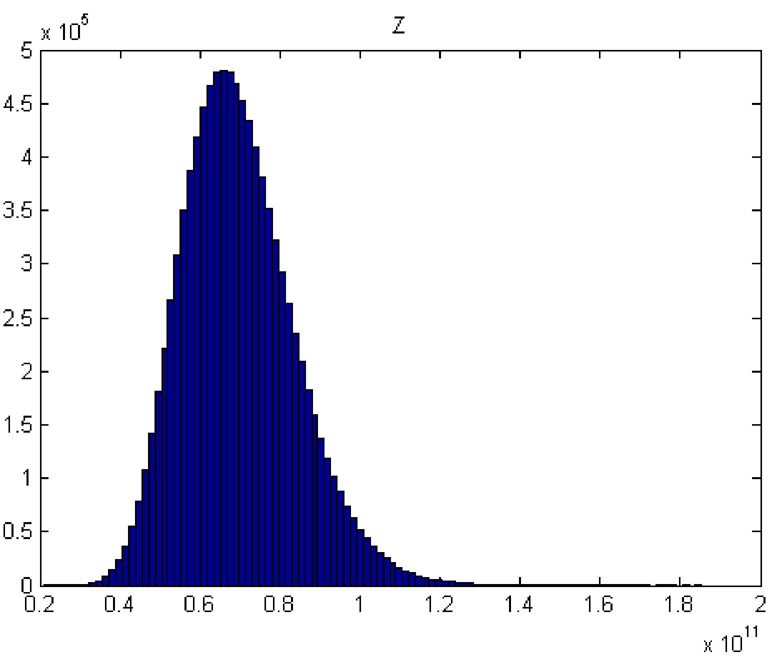
In fact we accomplish this as shown by the figure above. The mean is around 6 and the standard deviation is around 0.1 and the distribution appears to be normal. Likewise the height plot is similar (see below). In this plot the mean is around 10 and the standard deviation is around 0.1. It is worth noting that by increasing the number of numerical experiments (the `N=1000000;` variable in the Matlab code) the closer our results become, I.E. (R=6 ± 0.1 cm) & (H=10 ± 0.1 cm).



Of course the main result is the Volume of the cylinder (shown below).



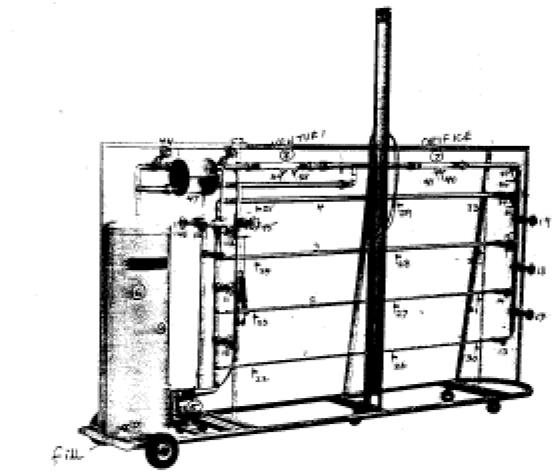
Notice the final results (V=1131.314 ± 39.3526 cm^3) almost the exact solution obtained by Dr. Polik. Of particular interest is the distribution, although it seems very much like a normal distribution, it is not. It may not be apparent from a visual inspection of the plot, but it is very apparent if out equation changes. Let's take a case where we are calculating some other parameter, say (`Z=pi*h.*r.^12`). Even though this variable is fictitious, it does show the point; multiple normally distributed inputs do not guarantee a normally distributed result.



This approach actually gives rise to an understanding of how distributions of experiments can be altered, whereas, taking the "random error differentiation" approach of Dr. Polik and Holman do not lend themselves to an understanding of these altering outcome distribution effects.

Fluids Lab Example

As an example of the use of Matlab to determine uncertainty we investigate a standard pipe flow experiment as shown below.



TECHNOVATE FLUID CIRCUIT SYSTEM, MODEL 9009

This experiment exist in the labs of most Mechanical Engineering program around the country. The main purpose behind this experiment in determining the pressure drop as a function of the flow rate. After going through a non dimensional analysis of this system (Buckingham Pi Theorem) we obtain,

$$\frac{\Delta P}{L} = fun(Re, f, \varepsilon)$$

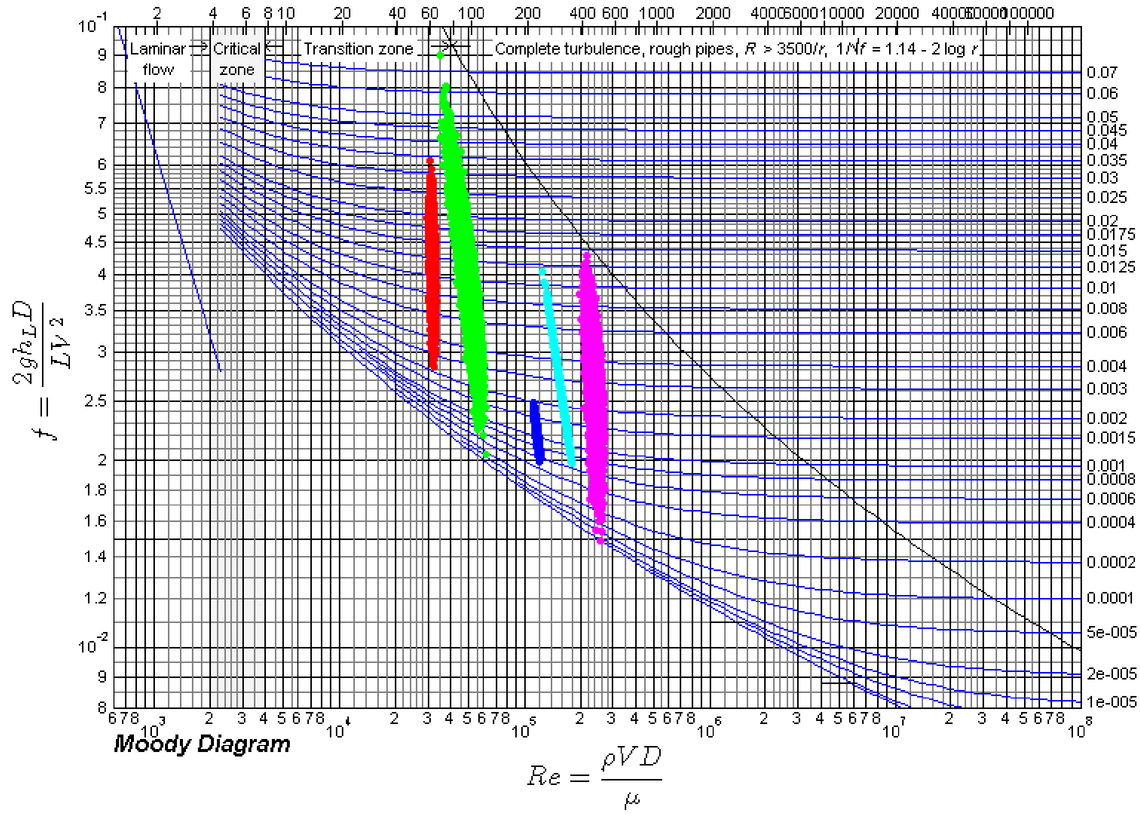
Where Re is the Reynolds number, f is the friction fact and ε is the relative roughness. Decades ago this experiment resulted in the creation of the Moody diagram. The goal of this experiment in the fluids lab is to perform a set of experiments and place these points on this very same Moody diagram. However, very often the student's results end up with points all over the Moody diagram. This leads students to conclude the experiment is ill posed or the theory is ill posed. This is where the reality of uncertainty comes into play. To begin with we use a Matlab code developed and shared by Tom Davis[3] to construct our Moody diagram. For the sake of brevity I will not include this section of Matlab Code. However, there are two other Matlab codes which are included. The main driver code is shown below. There are 5 experiments shown in this file. The variable `DhP` is the pressure drop in inches of water across the pipes. The uncertainty of this pressure drop is represented in the variable `DhPUN` . Likewise the pressure drop and uncertainty across the venturi meter are represented by the variables `DhV` and `DhVUN` respectively.

```
function []=PipeDriver()
% Pipe Driver code
% Written by Dr. Fithen
N=50000;
% Pressure drop in inches of water in the Pipe
DhP=[  1.2, 2.7, 8.5, 18.0 , 43.0]; %inches
DhPUN=[ .1,  .2,  .1,   .1,   4.2];
% Pressure drop in inches of water across the Venturi Meter
DhV=[   3.7, 9.1 , 50.0 , 90.0 ,210.0 ]; %inches
DhVUN=[  .1, 1.2,   1.1,   7.1,  15.0];
% Colors are...
%Red Green Blue Cyan Magenta Yellow Black White
Colors=['r','g','b','c','m','y','k','w',];
L=length(DhP);
% Call Tom Davis's code to create the Moody Diagram
%NOTE: I changed a few items on the plot, X & Y axis titles, etc
moody('IMPERIAL','A4')
hold on
for i=1:L
    PlotPipeFluidsExperimentUncertian(Colors(i),N,DhP(i),DhPUN(i),DhV(i),DhVUN(i));
end
end
```

Once the code determines how many data points are available, `L=length(DhP),` we loop through all these data points and call a routine call `PlotPipeFluidsExperimentUncertian`. This code, shown below, contains all the available data for geometry and constants for our experiment. The final two lines calculate the friction factor and Reynolds number. The final line plots the data on the already existing Moody diagram.

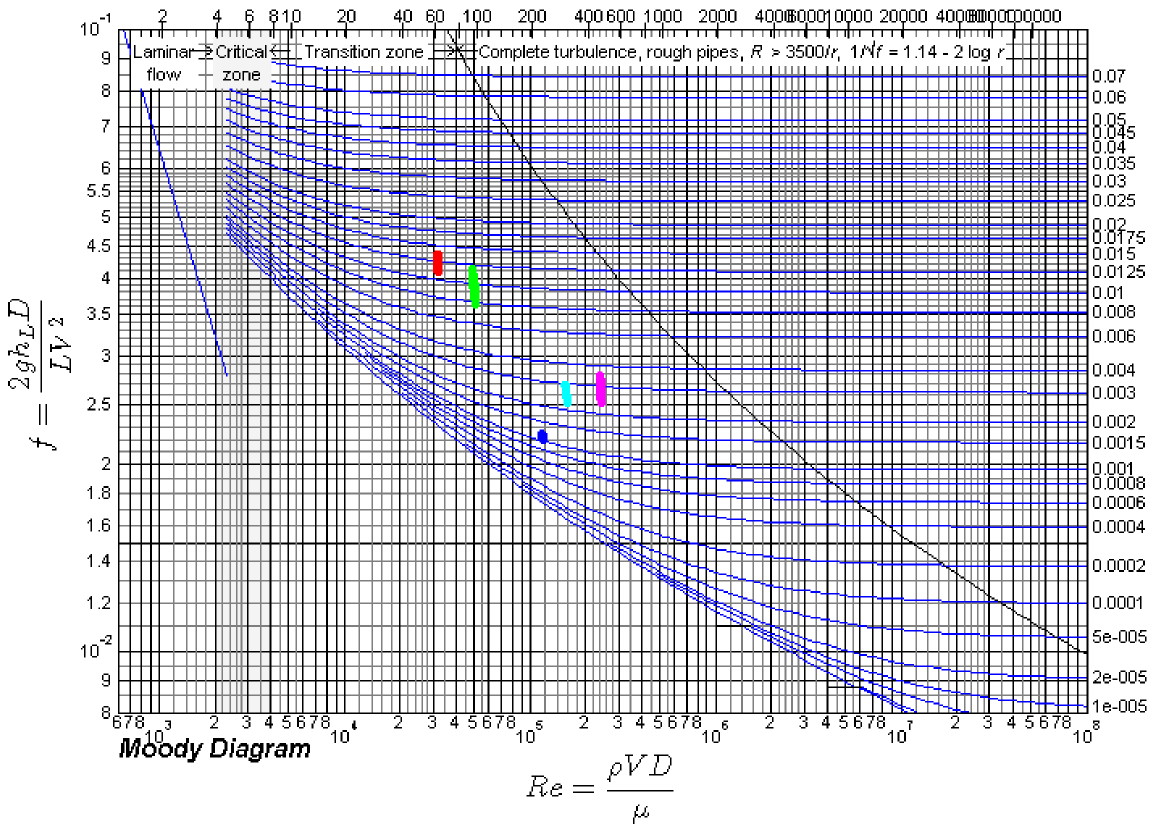
```
function []=PlotPipeFluidsExperimentUncertian(Color,N,DhP,DhPUN,DhV,DhVUN)
% Written By Dr. Fithen
DhP=DhP+DhPUN*randn(N,1);
DhV=DhV+DhVUN*randn(N,1);
DhV=DhV/12; %feet
DhP=DhP/12; %feet
%-----------------
% GEometry Must be changed
DiaPipe=[1]*ones(N,1); %inches
DiaPipe=DiaPipe/12; %feet
LPipe=[.6]*ones(N,1); %feet
Gamma=62.4; %lb/ft^3
rho=1.94; % slugs/ft^3
mu=2.34e-5 %UNITS?
g=32.174 %ft/s^2
%
% Assuming no correction on Venturi meter!!!!
%
Dsmall=1; %inches
Dsmall=Dsmall/12; %feet
Dlarge=2; %inches
Dlarge=Dlarge/12; %feet
Alarge=pi*(Dlarge/2)^2; %ft^2
Asmall=pi*(Dsmall/2)^2; %ft^2
Q=sqrt(2*DhV*Gamma/(rho*(1/Asmall^2-1/Alarge^2))); %CFS?????
VPipe=Q./(pi*DiaPipe.^2/4);
f=2*g*DiaPipe.*DhP./LPipe./VPipe.^2;
Re=rho*VPipe.*DiaPipe/mu;
%
% Plot data on the Moody Diagram
%
plot(Re,f,'or','LineWidth',.5,'MarkerEdgeColor',Color,...
            'MarkerFaceColor',Color,'MarkerSize',3)
end
```

It is worth noting that all calculations for the friction factor and Reynolds number are done on and element by element basis. Hence, the last line of the code will plot all the data at once. After executing this code we obtain the plot below.



The first and most important observation deals with the scatter on both the friction factor and the Reynolds number. We must remember what we are doing here before any generalizations can be made. We are taking one data point along with an uncertainty associated with this data point and creating a number of fake data points based on these 2 data points. This set of fake data points is normally distributed. Suppose our real data point is at the lower end of the uncertainty. If this is the case our cloud of fake data points will be shifted appropriately. This means the position of the cloud may not be completely accurate. However, the size of the cloud is of extreme importance. The uncertainty in the input data generates the size of the cloud. This can be seen in the next plot by decreasing the uncertainty in the data by one order of magnitude. Notice the size of the cloud significantly decreased in size.

The major item students walk away with from this exercise is experiment design. By observing the effects of uncertainty on the cloud size on the plot, students can adjust the uncertainty (select different measurement equipment) in a manner to reduce the resulting uncertainty (cloud size on the resulting plot).



Bibliography

1. William F. Polik, Professor of Chemistry, Hope College, http://www.chem.hope.edu/~polik/Chem345-2000/errorpropagation.htm
2. Jack Holman, Experimental Methods for Engineers, McGraw Hill, 7th edition 2007.
3. Tom Davis, "Moody Diagram", http://www.mathworks.com/matlabcentral/fileexchange/7747-moody-diagram

Biographical Information

Robert Fithen
Associate Professor
Arkansas Tech University
rfithen@atu.edu