Statistics 322, Statistical Computing, Home Work 2

Handed Out: Aug. 27, 1996 Due: Sept. 5, 1996

These are some simple exercises which are intended to help you learn to think in terms of matrix operations, instead of do - loops, and to help you learn about "debugging".

1. Generate a 400x400 matrix of Uniform(0,1) random variables. Calculate the sum of all its entries by:

   a. having a double loop, which runs through all the entries, and adds them, one at a time to the total.

Important Note: a "double loop", means an "outer" loop (say which runs through rows), together with an "inner" loop (which *for each row*, runs through all the columns).

   b. having a single loop, which runs through all of the entries, and adds them up using the command `sum` (either figure this out by experimenting, or read the manual).

   c. having no loops, but using the `sum command` twice (be careful here about the form of the result of `sum`, you want a scalar at the end!).

   For each of these methods, keep track of the time required. Turn in only a table, which gives the times needed by each of the three methods.

2. Generate a 60x60 matrix of Uniform(0,1) random variables. This time calculate the matrix product of this matrix with itself:

   a. having a triple loop, which first runs through the rows and columns of the result, and then for each entry, computes the dot product by yet another loop.

   b. having a double loop, which runs through the entries of the result above, but calculates the dot product as a MATLAB matrix product.

   c. simply doing the one line MATLAB matrix product of the whole thing.

   Again turn in only the times required for each of the three methods.

3. The frequentist interpretation of a confidence interval is that if one could repeat the experiment a large number of times, the proportion of times that the CI brackets the true value should be the stated probability. It is easy to use MATLAB to actually do this type of thing (e.g. for a class project when you are teaching. Write a program which efficiently:

a. Generates 10,000 data sets of 100 Uniform(0,1) variables each.

b. For each data set, calculates endpoints of the usual asymptotic Normal theory Confidence Intervals for the mean (use the standard 1.96 as the table value).

c. Counts the number of data sets for which the CI covers the theoretical expected value of $\frac{1}{2}$.

d. Returns the number of times $\frac{1}{2}$ was outside of the CI, and the "empirical coverage probability".

This time turn in both the output, and also a listing of the program used to generate it. Grading will be both on the efficiency, and also readability of your program.

Suggestion: Looping through all 10,000 data sets is a little slow. However, your computer may not have enough memory to handle the 100x10,000 matrix that would be needed to solve this problem with no looping. Hence consider a compromise.

Hint: The commands `SUM` and `STD` are useful. In addition counting is greatly facilitated by the "element by element relational operators" .< and .>, etc. These give matrices of 0's and 1's that can be easily combined and added up.

4. To gain experience with "debugging" MATLAB scripts, copy the program s322eg3.m to your directory, and modify it until it runs correctly. Turn in a print of your final version of the program.

Some tips on this are:

i. Instead of trying read the program line by line to understand what it is doing, try inserting temporary "write" statements at many points, to reveal the value of important variables as the program runs.

ii. If you can not figure out where a program crashes, put in write statements in many places, and see what the last one is that writes.

iii. Write statements are generally useful to reassure you that a program is running. However, too many of them can slow down your program too much (screen writing can become <u>most</u> of what the program does). A good rule of thumb is to have a program which writes to the screen every few seconds.

As always in this course, feel free to discuss, and compare ideas with other class members.