CS2035 - Assignment 2 - 2017

Out: February 3rd, 2017
In: Sunday, February 19th, 2017, at 11:55pm via Owl

Introduction

You are to write 4 MatLab functions and reuse 2 functions from your assignment 1 for this assignment to perform **vectorized** calculations and timing comparisons with the **serialized** calculations done on your assignment 1.

Assignment Tasks

All your functions should be coded in a single file called **ass2_2017.m**. The first half of this assignment uses mostly code written for **ass1_2017.m**.

1. First, code a 1^{st} function in **ass2_2017.m** called **ass2_2017** with a single input parameter **num_x** as on **ass1_2017.m**. There are no output parameters. **num_x** specifies the number of random numbers to be computed and stored in variables **x** (uniformly distributed random numbers) or **y** (for normally distributed random numbers). So far, you are only being asked you to do what you already did for assignment 1, so feel free to reuse that part of your code. You can use **num_x** with smaller values, say 100000, to reduce computational costs.

   Next, the **ass2_2017** function should compute the skewness and kurtosis values of **x** and **y** using the MatLab built-in functions, **skewness** and **kurtosis**, the **my_skewness** and **my_kurtosis** functions using **serialized** code that you wrote for assignment 1 and new **my_skewness2** and **my_kurtosis2** functions using **vectorized** code that you write for this assignment. These latter 2 functions do the same calculation as done by **my_skewness** and **my_kurtosis** but use **vectorized** code (does not use loops) instead of **serialized** code (uses loops). So, you have to write new 2^{nd} and 3^{rd} functions, **my_skewness2** and **my_kurtosis2**, and reuse your **my_skewness** and **my_kurtosis** functions from your assignment 1.

Perform timing on each calculation using **tic** and **toc**. For example, the following MatLab code:

```
start=tic;
k=kurtosis(x)
k_time=toc(start);
```

computes the execution times for the calculation of **kurtosis** on the random numbers in array **x**.

2. Code a 4^{th} function, **vectorized_mystery_calculation**, with header:

```
function result=vectorized_mystery_calculation(x,y).
```

This function computes:

$$result = \sum \sqrt[9]{\left(\frac{\sqrt[3]{2x(2-y)}}{\left(\sqrt[6]{\left(\sqrt[4]{(x^2y^3)} \right) + 5} \right)} \right)},$$

where x and y are 1D vectors (1D arrays) of random numbers we used above. To evaluate this function, use `".*"`, `"./"` and `".^"` to do element by element multiplication, Variable *result* requires 9^{th}, 6^{th} and 3^{rd} root calculations. To compute the n^{th} root of x use `x.^(1/n)`, for $n \geq 0$. And, of course, use **sum** to compute $\sum$. Call this function from **ass2_2017** twice, once with arguments **x** and **y** and a second time with arguments **y** and **x**. Your are supplied with `serialized_mystery_calculation.m` on the course webpage (by the assignment 2 handout) so you can check your vectorized code for correctness. Perform timing measurements for both the serialized and vectorized functions. Pretty print your results.

Serialization versus Vectorization

Consider the following code to compute the standard deviation of an array **x**. [This code is not necessarily the code you would have used on Assignment 1.] The serialized code uses a **for** loop and explicit array indexing and could be written as:

```
function value=serialized_std(x)
n=numel(x);
x_ave=mean(x);
value=0;
for i=1:n
    value=value+(x(i)-x_ave)^2;
end % i
value=sqrt(value/(n-1));
end
```

while vectorized code uses element by element arithmetic operators (`.*, ./` and `.^`) and could be written as:

```
function value=vectorized_std(x)
value=sqrt(sum((x-mean(x)).^2)/(numel(x)-1));
end
```

Function **serialized_std** explicitly references each element of **x**, computes the term **(x(i)-x_ave)** and squares it before summing. Note that, for efficiency considerations, **n** (used twice) and **mean(x)** (the value is used **n** times in the loop) have to be precomputed before the loop. Function **vectorized_std** does not use a loop and does not make explicit references to individual elements of **x**. As well, **mean(x)** and **numel(x)** are only used once and so do not need to be precomputed.

Finally, code in **ass2_example.m** might be:

```
function ass2_example()
x=rand(1000,1,'double');
fprintf('Serialized std: %f\n',serialized_std(x));
fprintf('Vectorized std: %f\n',vectorized_std(x));
end
```

which, when run, for one call to **rand** might print:

```
Serialized std: 0.289699
Vectorized std: 0.289699
```

This shows that the 2 functions produce the same answer.

Documentation

You must document your code (with appropriate comments). Write "beautiful" code, not just with comments but with appropriate code aligned nicely and use good, meaningful variable names. Don't use "magic constants", i.e. undocumented numbers. For example, instead of **x=rand(1000,1,'double')** use **x=rand(num_x,1,'double')**, where **num_x** might be 1000.0.

See the course webpage for examples of 2 well documented MatLab function examples, **commented_new_decimal2roman.m** and **commented_new_roman2decimal.m**. Have a block comment at the beginning of **ass2_2017.m** giving (at least) your full name (as it appears on your student card), your student number, your uwo email address and which assignment you are doing. For example:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  John Barron                               %
%  123456789, barron@uwo.ca                  %
%  CS2035, Assignment 2, 2017                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The source code **ass2_2017.m** should be submitted via OWL by the due date. We will run your code when grading your assignment. It is necessary to follow the assignment specs to optimize your grade.