Lecture 14: Computation Times

Computation Times

In MATLAB, we can use a very simple command to know how much time we are using for a block of program. Let’s see what they are and how to use them.

- Time Functions `timeit(f)`
  To measure the time required to run a function, use the `timeit` function. The `timeit` function calls the specified function multiple times, and returns the median of the measurements. It takes a handle to the function to be measured and returns the typical execution time, in seconds.

  For example, one can try to measure the time using for MATLAB to run singular value decomposition `svd` of a given matrix:

  ```matlab
  >> X = rand(200);
  f = @( ) svd(X);
  t1 = timeit(f)
  
  t1 =
  0.0050
  
  which means that my computer need 0.005 second to get the svd of this 200 × 200 matrix done. If one also want to know how fast it can be to get the three matrices of svd, that is, $U, S$ and $V$ matrices, the syntax is as following:

  ```matlab
  >> X = rand(200);
  f = @( ) svd(X);
  t2 = timeit(f,3)
  
  t2 =
  0.0129
  
  The number 3 in the second place of the bracket means the number of outputs, which are $U, S$ and $V$ matrices for svd case. Obvious, $[U,S,V]=svd(X)$ used longer time then the one have done `svd(X)`.

  You also can try to measure time required to run eigen-decomposition of a given matrix.

- Time Portions of Code `tic/toc`
  To estimate how long a portion of your program takes to run or to compare the speed of different implementations of portions of your program, use the stopwatch timer functions, `tic` and `toc`. Invoking `tic` starts the timer, and the next `toc` reads the elapsed time. For example, if we want to know how long your code takes to run, we can set `tic/toc` as following:
We are measuring the time required for calculating the sum of square of integers from 1 to 10000. Then we obtain

\[ t_1 = 8.8200 \times 10^{-4} \]
\[ n = 3.3338 \times 10^{11} \]

means that MATLAB use \( 8.8200 \times 10^{-4} \) second to calculate the answer \( n = 3.3338 \times 10^{11} \). We also can use one tic and several toc in the same program. For example, if we want to calculate the norm of the vector \( k \) and want measure time required first the sum then the square, we have

\[ t_1 = 0.0121 \]
\[ t_2 = 0.0239 \]
\[ n = 3.3338 \times 10^{11} \]
\[ \text{normn} = 5.7739 \times 10^{05} \]

\( t_1 \) means the using time to run form the beginning until the sum, and \( t_2 \) means the using time to run from the beginning until the square root.

Sometimes programs run too fast for tic and toc to provide useful data. If your code is faster than 1/10 second, consider measuring it running in a loop, and then average to find the time for a single run.

- The cputime Function cuptime
The `cputime` function measures the total CPU time and sums across all threads. This measurement is different from the wall-clock time that `timeit` or `tic/toc` return, and could be misleading. For example:

The CPU time for the pause function is typically small, but the wall-clock time accounts for the actual time that MATLAB execution is paused. Therefore, the wall-clock time might be longer. For example,

```matlab
tic;
initime = cputime;
time1 = clock;
pause(1.0); % Wait for a second;
fintime = cputime;
elapsed = toc;
time2 = clock;
fprintf('TIC TOC: %g
', elapsed);
fprintf('CPUTIME: %g
', fintime - initime);
fprintf('CLOCK: %g
', etime(time2, time1));
```

In this code we want the compute to wait a second and using three different time measurement command to see their different. Here goes the result:

- TIC TOC: 1.04015
- CPUTIME: 0.37
- CLOCK: 1.02821

Since `tic/toc` is using the clock time so it is very closed to the clock time but the computer time `cputime` is different.

All in all, the `timeit` function and the stopwatch timer functions, `tic/toc`, enable you to time how long your code takes to run. Use the `timeit` function for a rigorous measurement of function execution time. Use `tic` and `toc` to estimate time for smaller portions of code that are not complete functions.

It is recommended that you use `timeit` or `tic/toc` to measure the performance of your code. These functions return wall-clock time. Unlike `tic/toc`, the `timeit` function calls your code multiple times, and, therefore, considers first-time costs.

The result of time might vary under different circumstances. For example, if you use different computers to test the time using, the result will be different. For another, if your use different version of MATLAB, the using time for running the same command or program will be different as well.

**Using computation times command to compare different algorithms**

We can use `tic/toc` to check the running time for the same but using different ways to calculate it. For example, the sum of square of integers from 1 to 10000 as we mentioned before: We calculate the sum into three different ways, first one is using the function `sum()`:


$$\text{tic;}$$
$$n=0; \ k=1:10000; \ n=\text{sum}(k.^2);$$
$$t1=\text{toc},$$
$$n$$

$$t1 = 8.8200e-04$$
$$n = 3.3338e+11$$

Second one is using the command \texttt{dot()}:

$$\text{tic;}$$
$$n1=0; \ k=1:10000; \ n1=\text{dot}(k,k);$$
$$t2=\text{toc},$$
$$n1$$

$$t2 = 0.0010$$
$$n1 = 3.3338e+11$$

Third one I build up a \texttt{for} loop to calculate the sum:

$$\text{tic;}$$
$$n2=0; \ \text{for} \ k=1:10000, \ n2=n2+k^2; \ \text{end};$$
$$t3=\text{toc},$$
$$n2$$

$$t3 = 0.0044$$
$$n2 = 3.3338e+11$$

As you can see, if one use the functions which are build up by MATLAB, the time required will be faster than the one including the loop structures.
Exercises