Writing Faster MATLAB® Code

Some tricks and tips on the efficient usage of MATLAB were provided in the August 2013 issue of IEEE Control Systems Magazine [1]. This column provides some tips on writing faster MATLAB code, which is important in applications to larger scale problems. Some codes can be sped up by more than three orders-of-magnitude by following these tips.

Tip #1: Define vector and matrix dimensions before using them. A feature of MATLAB is that its ability to redefine the numbers of elements in an array on the fly, without having to define the dimensions beforehand. For example, a valid MATLAB code for creating a vector of numbers from the standard normal distribution is (to be compared on a fair basis, all variables were cleared from memory before running each snippet of code)

```
>> for i = 1:1000
>>     A(i) = randn;
>> end
```

This feature reduces the code length but has a high computational cost, as it is much more efficient to define the vector or matrix definitions beforehand. For example, consider two snippets of MATLAB code, for defining a matrix whose elements come from a standard normal distribution, that are exactly the same except that the second code snippet defines the matrix dimensions beforehand (all costs are reported for a 1.6 GHz Quad Core Intel i7 processor):

```
>> tic
>> for i = 1:1000
>>     for k = 1:1000
>>         A(i,k) = randn;
>>     end
>> end
>> toc

Elapsed time is 2.422478 seconds.

>> tic
>> A = zeros(1000);
>> for i = 1:1000
>>     for k = 1:1000
>>         A(i,k) = randn;
>>     end
>> end
>> toc

Elapsed time is 0.066432 seconds.
```

The latter MATLAB code snippet was more than 35 times faster. Redefining dimensions takes time; the large difference in runtime is because the former code snippet redefines the matrix dimensions many times whereas the latter code snippet defines the matrix dimensions only once.

The speedup achieved by predefining the dimensions depends on the cost of the other calculations in the program. The above example used the `randn` command so that some calculations occurred in each loop. Consider the runtimes for the aforementioned two code snippets with each random number call replaced with the number 1:

```
>> tic
>> for i = 1:1000
>>     for k = 1:1000
>>         A(i,k) = 1;
>>     end
>> end
>> toc

Elapsed time is 2.366193 seconds.

>> tic
>> A = zeros(1000);
>> for i = 1:1000
>>     for k = 1:1000
>>         A(i,k) = 1;
>>     end
>> end
>> toc

Elapsed time is 0.017636 seconds.
```

Predefining the dimensions in the latter code snippet resulted in a speedup of more than 100 times.

MATLAB is an *interpreted language*, which means that it avoids explicit program compilation. When a code is run, each statement of the MATLAB code is executed one statement at a time. If the matrix dimensions are not defined beforehand, then MATLAB needs to redefine the matrix dimensions on-the-fly, which results in longer runtimes.

Many MATLAB codes can be sped up by orders of magnitude by applying some simple tips.

Tip #2: Use built-in MATLAB commands where possible. Using built-in MATLAB commands leads to code that is both shorter and faster. For example, consider an alternative code snippet for generating a matrix whose elements come from a standard normal distribution:

```
>> tic
>> A = randn(1000);
>> toc

Elapsed time is 0.021536 seconds.
```

This code snippet is about three times faster than repeatedly calling the `randn` command in a loop (0.066432 s). This snippet uses a built-in MATLAB command both defines the matrix dimensions and assigns its elements in precompiled code. The `randn` command does not consist of individual lines of MATLAB code and so can be optimized to have shorter runtimes. This tip is consistent with a previous tip [1] to avoid loops as much as possible when using MATLAB, due to its low efficiency in handling loops.

The efficiency improvement of using built-in MATLAB commands depends on the number of elements and the amount of computation in the definition of each element. For example, consider the code that defines a 1000 × 1000 matrix of ones:

```
>> tic
>> A = ones(1000);
>> toc

Elapsed time is 0.003583 seconds.
```

This code snippet is about five times faster than defining each element of the matrix *A* within two nested loops (0.017636 s).

Orders of Magnitude Speedups. Combining Tips #1 and #2 can produce truly astounding reductions in runtimes. For example, running the MATLAB code snippet

```
>> for i = 1:3000
>>     for k = 1:3000
>>         A(i,k) = 1;
>>     end
>> end
```

took 64.804833 seconds compared to 0.027642 seconds running the code snippet

```
>> A = ones(3000);
```

For this example, the runtime was reduced by more than a factor of 2000.

References

[1] Kam K. Leang, "MATLAB tricks and tips," *IEEE Control Systems*, vol. 33, no. 4, pp. 39–40, August 2013.

– Mark C. Molaro and Richard D. Braatz

10.34 Numerical Methods Applied to Chemical Engineering

Fall 2015