# Preallocation is not an option

Posted on June 25, 2012 by Jerome

As any activity, programming does require that you follow some rules of good practice. As you should clean up your room regularly, you should keep your memory tidy and neat. In this post, I introduce you to the joy of memory preallocation. To me, Memory preallocation is just like this : Something that you should ALWAYS do (in fact I am better at preallocation than cleaning up my home...). As usual, I end this post with some tricks that should be interesting for advanced programmers as well.
But what is memory preallocation all about?

Contrary to C and most compiled languages, Matlab takes care of memory allocation for you. So when you start a program, Matlab finds out for all of your variables how much memory to reserve for your data. It does this based on the local information it has, like data types or the required size for all elements. For instance when you do :

```
1    x=1:100;

2    y=exp(x);
```

Line 1, Matlab knows that x is a matrix made of 100 double elements. So it allocates the necessary size at this exact moment. On the second line, y is of the same size so Matlab also allocates a Matrix before it fills it with the exponential of x.

An immediate consequence of this mechanism is that EVERY TIME you ask Matlab to store some new dataset, Matlab will allocate some memory. So, for instance, in :

```
1    x=1:100;

2    x(101)=1;
```

Here,x is first made of 100 hundreds elements, so when you fill the 101 element, you basically ask Matlab to reallocate some memory so that it can store 101 elements in x.

Why can this be a big deal?

Because Matlab requires contiguous memory for all the data in a matrix. As a result, if the memory space number 101 closed to the current memory spot for x is taken, then Matlab find another place in memory for the entire x and copy it over to the new spot with room for 101 elements.

If x was first made of 1 million elements instead of 100, the memory size for x is significant and these big spots are not so easy to find in memory. It's like trying to park a big van in downtown San Francisco... It can take some time.

This becomes an even bigger issue if you start using for loops. Like :

```
1    for i=1:10000000
2       y(i)=exp(i);
3    end
```

Here, at every step of the for loop, you ask Matlab to add an element to y. So, in fact, you ask Matlab to GROW y in memory at each step.
At the beginning, this is easy, because y is small. But as you progress in the loop, y becomes bigger and bigger, so Matlab has to find bigger and bigger spots in memory a total of 100 million time (in this example). This a complete memory nightmare.

The solution is quite simple : **Preallocate**.
The idea of preallocation is that Matlab does NOT know in advance the size of your matrices but YOU do. So between human and computer beings, we should communicate.
Most people are used to pre-fill their matrix with zeros, like so :

```
1    y=zeros(10000000,1);
2    for i=1:10000000
3       y(i)=exp(i);
4    end
```

On my computer, the first version takes 4 seconds versus 0.2 s for the second version. A decent 20 times boost.

Yair from UndocumentedMatlab recently introduced a very nice trick to boost this even further. It turns out that most of the time spent in the function zeros is wasted filling the spots with tons of zeros. You can bypass all of this by just asking to fill the last element, like so :

```
1       y(10000000)=0;

2       for i=1:10000000

3          y(i)=exp(i);

4       end
```

Here, the code runs in only 0.19 s. So we get an extra 10 ms time boost by not filling with zeros. This is not very significant but it might be in some particular cases where you have to preallocate multiple times very big matrices for some unknown reasons.

SOURCE

http://www.matlabtips.com/preallocation-is-not-an-option/