Pre-allocate memory to make programs run faster

In Matlab, arrays are dynamic in size, meaning, whenever you need a larger array, you can just index based on this larger number and Matlab will automajically resize your original array to be the new bigger size. There is a cost with doing this, which is that your program will slow down, if you do it often enough. Thus, if you know (or have a good guess) at how big the array needs to be in the first place, you can "pre-allocate" it, meaning pre-size the array.

If the matrix size is not defined prior to populating it with data through a FOR loop, memory fragmentation problems may happen since MATLAB is not aware of the final matrix size upon the conclusion of the FOR loop. For example, look at the following FOR loop:

```
for i=1:10
```

```
      x(i)=i;

end
```

When this FOR loop is executed, MATLAB looks at the i=1, requests enough memory from the operating system to create a 1 x 1 matrix, and creates x(1)=1. When i=2, MATLAB requests more memory so a 1 x 2 matrix can be stored. If this additional memory is in the same continuous memory strip as when x(1)=1, MATLAB will simply add the additional number in the same memory strip. If the original memory strip is only big enough for a 1x1 matrix, MATLAB moves the x(1)=1 and places it into a memory spot that is large enough for the 1x2 matrix. Since the matrix is now 1x2, the original memory slot is useless to MATLAB for any matrix larger than 1x1. This memory is now fragmented, and this would cause significant problems with large FOR loops.

In order to work around this issue, you should pre-allocate memory by creating an initial matrix of zeros with the final size of the matrix being populated in the FOR loop. For example, if you create a large matrix by typing a = zeros(1000), MATLAB will reserve enough contiguous space in memory for the matrix 'a' with size 1000x1000. This way, instead of looking for a new block of contiguous free space in memory every time 'a' grows larger than the block that holds it, MATLAB will now only change the values in the pre-allocated memory space for the matrix 'a'. Following is an example on how to pre-allocate memory before entering a FOR loop:

```
x=zeros(30);
```

```
for i=1:30,

      for j=1:30

          x(i,j)=i+j;

      end

end
```

The above creates x which is a 30x30 matrix. This gives MATLAB a memory block large enough so it doesn't have to keep asking for fragmented memory. This method reduces the chance of receiving "Out of Memory" errors due to fragmentation. It also improves the performance of the program, as shown in the following code:

Running the following code where memory is NOT pre-allocated:

```
tic;
```

```
for i=1:1000,

      for j=1:1000,

          x(i,j)=i+j;

      end

end

toc
```

returns:

```
 Elapsed time is 12.175349 seconds.
```

On the other hand, pre-allocating the memory ahead of time

```
tic;
```

```
x=zeros(1000);

for i=1:1000,

      for j=1:1000,
```

```
          x(i,j)=i+j;

      end

end

toc
```

returns:

```
 Elapsed time is 1.761482 seconds.
```

For pre-allocating memory for a cell array, you may use the following command:

```
c = cell(m, n, p,...)
```

It creates an m-by-n-by-p-... cell array of empty matrices. Arguments m, n, p,... must be scalars.

It should be noted that preallocating memory does not make sense if you do not know the eventual size of the matrix you wish to create. This is because one of two cases are likely to occur. Either the preallocated memory will either be too large, resulting in wasted memory; or the allotted memory will be too small for the matrix you are trying to create, resulting in the need to allocate more memory and copy matrix elements to the new space. The latter case will cause you to be just as vulnerable to the memory fragmentation problems you are trying to avoid by preallocating memory.

If you know approximately how big an array will be when your program is done, you can ask Matlab to give you an "empty" array of that size to begin with. This will make your program much faster for large data sets. **It will have almost no affect for small data sets.**

While pre-allocating an array (or matrix) of numbers, you can use the "zeros" function. To pre-allocate an array (or matrix) of strings, you can use the "cells" function.

```
grade_list = zeros(1,100);       % approximately 100 students in class

names = cell(1,100);             % approximately 100 names for the students in class
```

Here is an example of a script showing the speed difference. Make sure you "clear" the array variable if you try the code more than once.

SLOW VERSION (NO-pre-alloaction)

```
% lets time the below code:

      clc, clear

      start_time = clock();



      % large amount of data, no pre-allocation  (we "create" the array as we go)

      large_data_size = 100000;



      for i=1:large_data_size

        random_number_array_1(i) = rand();

      end



      % look at how long that took

      fprintf('it took %d seconds\n', etime(clock(), start_time));
```

FAST VERSION (WITH pre-allocation)

```
% lets time the below code:

     clc, clear

      start_time = clock();



      % large amount of data, pre-allocation (we specifiy approximately how many valu
e)

      large_data_size = 100000;

      random_number_array_2 = zeros(1,large_data_size);
```

```
      for i=1:large_data_size

        random_number_array_2(i) = rand();

      end



      % look at how long that took

      fprintf('it took %d seconds\n', etime(clock(), start_time));
```

SMALL DATA SET VERSION

```
% lets time the below code:

      start_time = clock();



      % no pre-allocation, but with a small number of items

      large_data_size = 1000;    % although the variable is confusing



      for i=1:large_data_size

        random_number_array_3(i) = rand();

      end



      % look at how long that took

      fprintf('it took %d seconds\n', etime(clock(), start_time));
```

This article has been adapted from the following sources:

http://www.mathworks.com/matlabcentral/answers/99124-how-do-i-pre-allocate-memory-when-using-matlab

http://www.cs.utah.edu/~germain/PPS/Topics/Matlab/preallocating_space.html