Matlab and C array conversion

Peter R.T. Munro

August 15, 2003

This theory could be developed for a multidimensional array however I will restrict it to arrays of tow and threee dimensions. Matlab constructs its arrays by laying them out row-wise in memory. This is depicted below in Figure

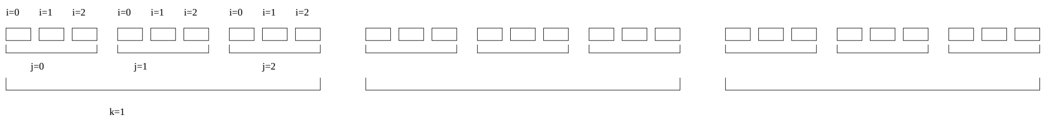

Figure 1: Layout of a matlab array in memory

This depicts an $3 \times 3 \times 3$ array that would be index according to:

```
>> array(i,j,k)
```

in matlab. Then the array could be pictured as being composed of rows indexed by i, columns indexed by j and layers indexed by k. The address of a particular element `array(i,j,k)` in memory is thus given by:

```
(mArray +k*nrows*ncols+ j*nrows + i)
```

where `mArray` is the `(double *)` pointer given within the `mexFunction`. Thus the array can be indexed this way. It is also possible to cast the memory in such a way that it can be indexed using the usual array notation `array[k][j][i]` where one should note that the indices are in reverse order.

In order to do this we must declare a variable of type `(double ***)`. It is structed so that if `array` is our pointer, then `array[k]` points to a layer (see diagram). Then `array[k][j]` points to a column within a layer. The the final index is `i` which is used to traverse along the row. The code for performing this is included below.

```
/*Casts a 3-dimensional array such that it may be indexed according to the
  usual array indexing scheme array[k,j,i].

  array is a point to a matlab 3 dimensional array
  nrows the number of rows in the array
  ncols the number of columns in the array
  nlayers the number of layers, each of dimension nrows*ncols

*/

double ***castMatlab3DArray(double *array, int nrows, int ncols, int nlayers){
  double ***p;
  int i,j,k;

  p = (double ***)malloc((unsigned) (nlayers*sizeof(double **)));
  for(k =0; k<nlayers;k++)
    p[k] = (double **)malloc((unsigned) (ncols*sizeof(double *)));

  for(k =0; k<nlayers;k++)
    for(j =0; j<ncols;j++)
      p[k][j] = (array + k*nrows*ncols+ j*nrows);
  return p;

}

/*Frees the axilliary memory used by the castMatlab3DArray
 */
void freeCastMatlab3DArray(double ***castArray, int nlayers){
  for(int k =0; k<nlayers;k++)
    free(castArray[k]);
  free(castArray);
}
```