This document contains instructions for using the MATLAB™ code used for quantifying filler dispersion in composites systems, as explained in the article[1] appearing in the scientific journal 'Polymer'

HS Khare, DL Burris

October 6, 2010

[1] **A Quantitative Method for Measuring Nanocomposite Dispersion** H.S. Khare, D.L. Burris*, Polymer 51 (2010) 719-729

I. Using the code:

The following is a step-by-step instruction on using the dispersion code.

Step 1

Upon execution, the code will prompt for a bitmap of the TEM image to be analyzed. Note that the format of this file should be **.bmp**. The image size (in pixels, or otherwise) or the aspect ratio is not important, however, **pixel values should be even** (for example, 231px x 422px will need to be cropped to 230px x 422px, etc.). The image *should* be in grayscale (with particles as black pixels, filler as white- or vice versa).

Step 2

In the next step, you are required to provide the physical width of the TEM image selected in step 1. The TEM micrograph width may be ascertained from the scale bar. This input is required for the code to work. Enter the width of the TEM image in nanometers at the following prompt:

```
Enter approximate width of TEM image in nanometers:
```

Step 3

A starting estimate of the characteristic box length (either free space length or agglomeration length) needs to be provided next. This 'starter' value is determined **visually** from the TEM image- looking at the TEM image, what is the approximate length scale of the filler-free space or the agglomeration space in nanometers? Enter this value at the following prompt:

```
Enter initial guess of characteristic square width in nanometers:
```

Remember, this value will always be approximate based on individual perception of the TEM image. The value you enter will only affect how long it takes for the solution to converge to the 'actual' value, and not the value itself. For suggestions on selecting the initial value, please refer to section II below.

Step 4

In order to generate histograms of particle distribution within the chosen box size, you next need to specify how many of the boxes are to be used for gathering data for the histograms. Do this at the following prompt:

```
Enter number of random squares to use for analysis:
```

A higher number of boxes ensures statistical robustness of your answer, but might take a longer time to complete due to lengthier computations. A smaller number of boxes ensures quicker computation, albeit a slightly inaccurate result, and can be used for non-rigorous analyses.

Typically, this number may vary between 100 and 1000 for an acceptable accuracy + speed combination. For suggestions on selecting the appropriate number of boxes, see section III

Step 5

After the automatic iteration to the characteristic box size is completed, the code prompts if you wish to proceed to the 'Manual Mode'. The manual mode is an optional extension of the code

that allows you to gather statistical information such as scatter in the value of the characteristic box size. If you wish you use the value generated by the automatic iteration, and not enter the manual mode, enter 'n' at the following prompt:

```
Automatic iteration completed. Continue to manual mode? (Y/N)
```

More information on the Manual Mode is given in Section IV

Results:

The final value after iteration from your input number of the characteristic box is displayed:

```
Value of required box width is (in nanometers):
```

Further, the area fraction of the TEM image occupied by fillers (identified as black pixels) in the matrix is also displayed:

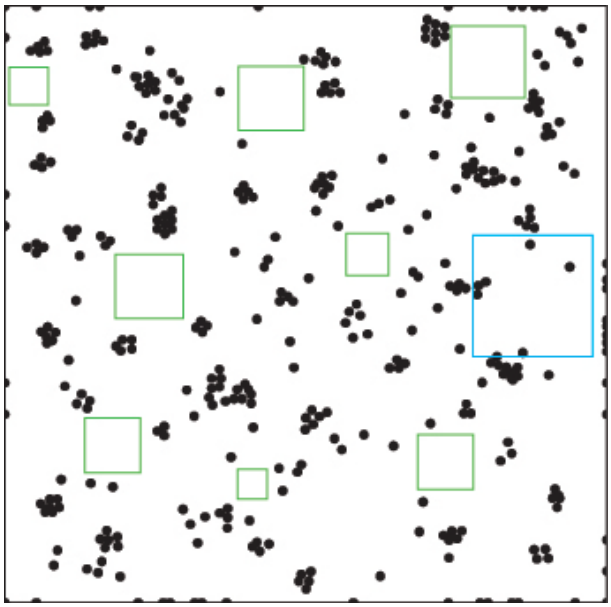
```
Particle area fraction (where particles are represented by black
pixels):
```

A histogram is generated showing the zero-mode corresponding to the box size displayed above.

II. Selecting size of initial box

At step 3, you are prompted to enter an 'initial guess of characteristic square width in nanometers'.

When using this code to calculate the size of the free space length, we are interested in finding the (largest) square size that, when thrown over the image at random locations a given number of times, would be most likely to have no particles contained within it. The code uses an initial user-input, and iterates to the actual value by generating these boxes, and placing them over across the TEM image. A good analogy is to compare it to throwing a fishing net at different points in a lake, with the hope of *not* catching any fish. Of course, you will need to keep reducing the size of your net, until your reach a 'critical' dimension where you're more probable to drop the net at different locations, and still not catch anything. The largest net for which this happens is what is analogous to the free space length. Looking at the TEM image then, we need a subjective measure of what *seems* like the free space length. The green boxes in the following image of a fictitious dispersion could all very well be this starting size:



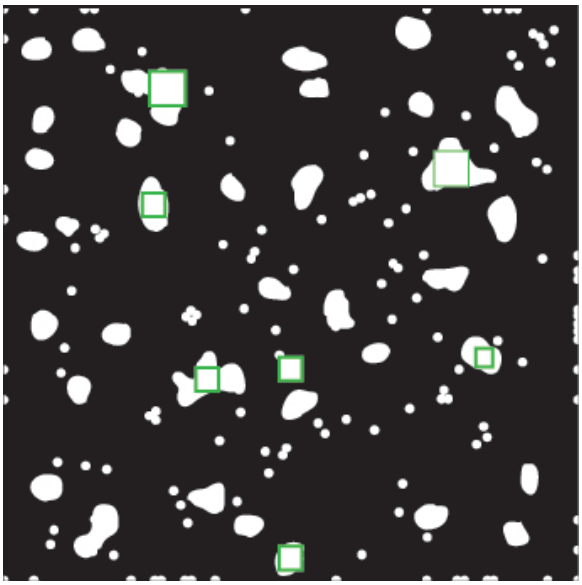
Note that a box size similar to the blue box would never be completely particle-free, regardless of where you position it. We seek to identify which among the green boxes is the truest representation of the free space length.

For this purpose, you may use a numeric value slightly larger than what seems like the free space length (green boxes in this case), and let the code iteratively reach the 'optimal' box size (work its way down). Alternatively, you may use a value which *seems* like the free space length (or even slightly smaller) and if you're too small, you can always start again with a marginally increased value (a good way to tell if you're too small is if the value the code outputs is the same as the value you entered for the box size). The disadvantage in using the former is wastage of computational time/effort. Since the code will be trying to work its way down from your starting value, depending how overestimated this starting value is, the code might take a prohibitively longer time simply an as artifact of the starting choice. For this purpose then, a size slightly

smaller than the perceived free space length is preferred because if it's too small, the code will "tell" us, and we can always restart with a slightly incremented starting value.

For the illustration shown above, then, if the width of the TEM image were 2 micrometers, the size of our initial guess (which should be definitely less than the size of the blue box), would be about a fourth of the image width (or maybe smaller still), or roughly 500 nanometers. We conclude for computational efficiency that it's a fourth (or a fifth, for that matter), and not a third, even though it is difficult to tell exactly. If your initial size is too small (say, the same size as one of the smaller green boxes), you can restart with a larger initial guess.

The same thought process applies when selecting an initial value to calculate the agglomeration size. Since now the image is inverted (that is, black pixels represent matrix and white pixels represent filler), we are still looking for that 'representative' box size, which is characteristic of the size of what are now 'white patches' of filler agglomerates. This is illustrated in the image below:



It is important to remember that we define an agglomerate as a cluster of fillers, where the filler-to-filler distance is less than the characteristic filler size[2].

[2] **A Quantitative Method for Measuring Nanocomposite Dispersion** H.S. Khare, D.L. Burris*, Polymer 51 (2010) 719-729

III. Selecting the number of boxes

In order for a given sized square to be indicative of the free space length or the agglomeration length, it is imperative that the given square is representative of the filler spatial statistics across the entire image. As a result, it is not difficult to realize that for a more accurate and statistically robust measurement of the characteristic box, a larger number of squares need to be overlaid at random locations on the image. While an increase in number of boxes overlaid (for any box size) increases accuracy (how close the number you're getting is to the actual free space length) of the eventual 'solution', increasing the number of boxes may increase computational effort.

It is observed that, usually, the same computational effort is required for a larger number of smaller sized squares (say, an order of magnitude smaller than the image size), as is for a smaller number of large-sized squares (say, comparable or of the same order of magnitude as the image size). The relative importance of computational speed and accuracy of result is the choice of the user.

For images where a larger starting box size is necessitated by nature of composite microstructure, it is advisable to limit the number of squares in the initial stages. Thereafter, depending on the convergence speed, the user may subsequently increase the number of squares to obtain a more accurate solution.

Always bear in mind, however, that smaller sampling will always give you numbers which will be off substantially from the actual characteristic box. If the code is run multiple times on the same image (say, in manual mode), with the same parameters, changing only the number of boxes, a large scatter in your free space value will be seen for a smaller 'number of boxes' value. This is largely mitigated at higher 'number of boxes' values.

Ideally, a user may initially wish to experiment with box numbers between 100 and 1000, to obtain the required accuracy coupled with desired convergence speed. As mentioned earlier, if the square size is large relative to the image size, one may start with box numbers as few as 50, and work their way upwards till computational time becomes prohibitive. Once an accurate-enough value for characteristic box is obtained (reduced scatter between successive runs), 'fine tuning' to get the exact value may be done in the manual mode.

IV. Manual Mode

After the code iterates to the value of the characteristic box size, an option is provided by which the user can enter a 'manual' mode. Using the value provided by the automatic mode, the user can, without having to reselect the image, input only the new characteristic size and number of boxes he or she wishes to simulate, and obtain results for those parameters.

The manual mode only provides the particle distribution for the given box size and box numbers- it does not attempt to iterate on its own to an 'optimal' value. This feature is helpful for those users who wish to 'fine tune' the value provided by the automatic iteration, or those who wish to gather information on how changing an individual parameter changes the output (for instance, how changing the number of boxes for the same box size changes the shape of the histogram).

'Fine tuning' may be performed by using as a reference the final value outputted by the automatic iteration (provided care was taken to use an adequate sampling size to reduce scatter). Using this value, the user may manually increment or decrement in the box size and observe the generated histogram to ascertain the value of the characteristic box size. Further, since no step-wise iteration is performed, a single 'manual-mode' measurement is significantly faster than the automatic mode, which means that a higher sampling size (number of boxes) may be used to truly iron out the real characteristic box size.

As an illustration, one may use 200 boxes for a given rough starting box size, work up to 500 boxes till computation time becomes prohibitive. Then, using the value the code gives you at 500 boxes, and incrementing/decrementing that value with a much higher number of boxes (say, even 3000, etc.) one can find the size of the characteristic box to a resolution of a single nanometer (this, of course, would be a function of statistical error derived from your sampling size).

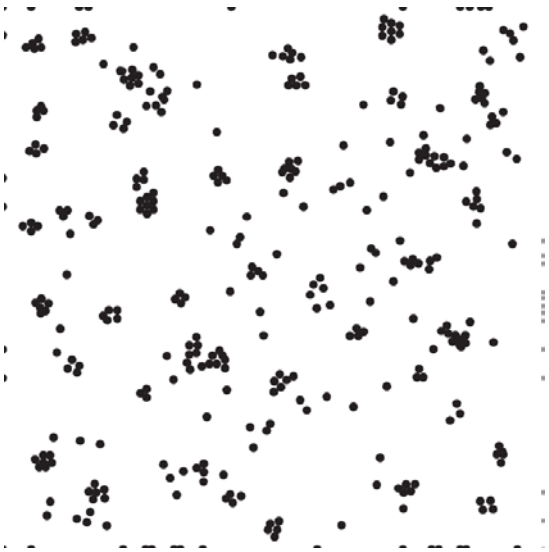
It may be mentioned here that the only data that 'carries over' from the automatic mode to the manual mode is the image pixel matrix, and the image size in nanometers. Aside from these, values you obtained from the automatic mode (box size, histogram, etc.) do not influence your results in the manual mode. The auto mode can be thought of as merely something to give a reference point for starting the manual tuning.

V. Illustrations

In this section, a walk-through is provided for measuring the free space length and agglomeration length of two dispersions: the first is a hypothetical dispersion where fillers cluster together at a few sites, creating small agglomerates and creating small filler-free voids in the matrix. The second is an actual TEM image of an alumina-epoxy nanocomposites system.

Clustered Dispersion

In the illustrative, fictitious nanocomposite system shown on the right, we notice that spherical filler particles cluster at various sites forming small agglomerates, while leaving volumes lacking in fillers.



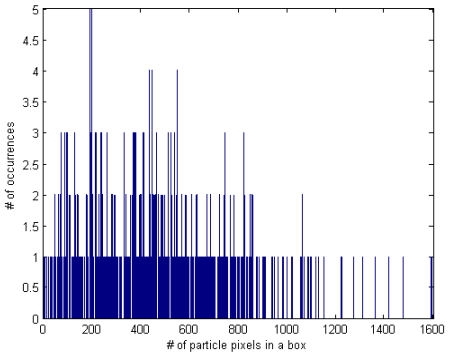
Our aim in this example is to measure the free space length and the agglomeration length. It is assumed that the image shown on the right was obtained from a TEM micrograph by manually isolating fillers from matrix (in a graphics package).

First, let us establish some parameters. For the sake of computation, let us designate the width of the image to be 2µm. This gives us an individual filler diameter of 30nm (assumed spherical) at a filler loading of 6%. In an actual TEM micrograph (as we shall see in the next example), the size of the TEM image and size of the particle (although not required in our analysis), are often known. The steps involved in evaluating the free space length (L_f) and agglomeration length (L_a) are as listed below:

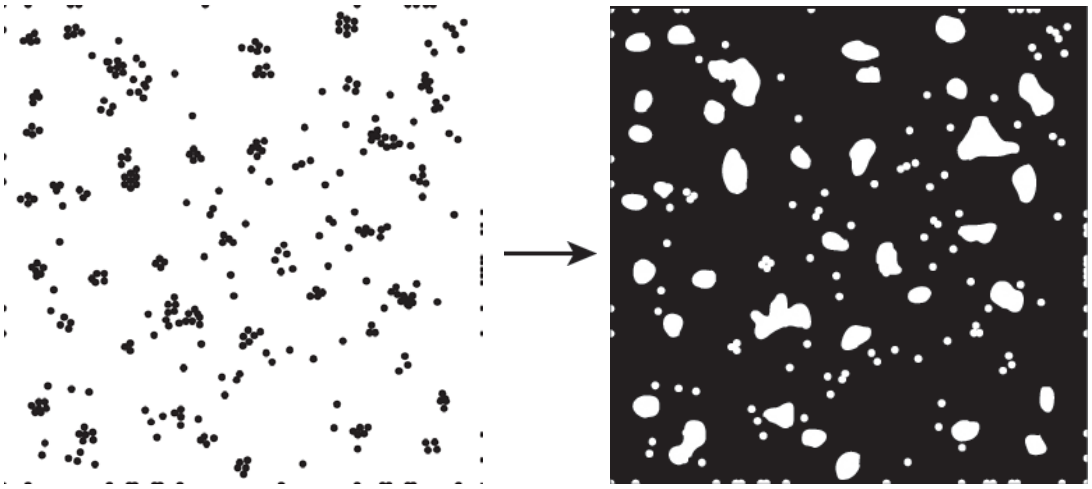
1. Executing the .m file, at the prompt, we select the file Clustered.bmp
2. At the following prompt: `Enter approximate width of TEM image in nanometers:` we enter the size of the image, 2µm as 2000nm
3. The following prompt asks us to `Enter initial guess of characteristic square width in nanometers:` Observing that the TEM image width is 2000nm, and as per the discussion is section II, the pockets of filler-free space seem to be about a fourth of the image width, we enter 400nm as our initial guess (underestimate the one-fourth value).
4. Next, we are asked to `Enter number of random squares to use for analysis:` Since we are unsure of how computationally intensive our selection of 400nm is going to be, we choose to use 500 squares (which is about halfway between the recommended 100 and 1000)
5. After allowing the computation to finish (in this case, the simulation ran for a little under 3 minutes, while this may vary from one run to another even with same parameters), we obtain the following results:

```
Value of required box width is (in nanometers): 245
Particle area fraction (where particles are represented by black
pixels): 0.0661
```

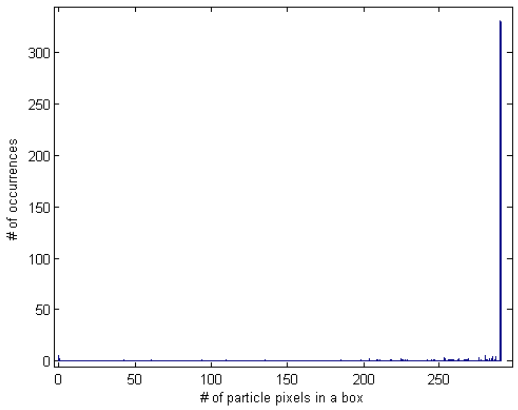
Further, the adjoining histogram is generated. Clearly then (and as was discussed in section II), our starting guess of 400nm was a large overestimate of the actual 245nm. In hindsight, we *might* have saved some computation time by selecting an initial starting value of, say, 250nm, or 300nm. While not illustrated in this example, the value of 245nm may be further tuned in manual mode.



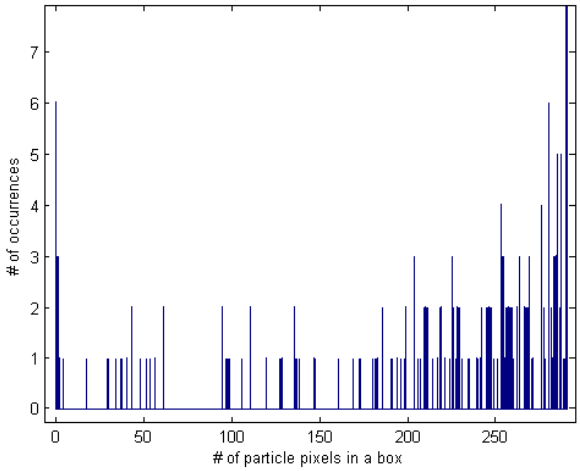
6. Next, to evaluate L_a, in the TEM image, we identify filler agglomerates in a manner discussed in the paper, and again using a graphics software (such as MSPaint, or Adobe Photoshop/Illustrator), we generate the following image from the previous:



7. Using this new image, we go through the steps 1 through 5, this time selecting our initial estimate (of agglomerate size, here) as about 5% of the image, or 100nm. The number of boxes is kept at 500. The computation takes only a few seconds, and the agglomerate size converges to 48nm. The following histogram is generated

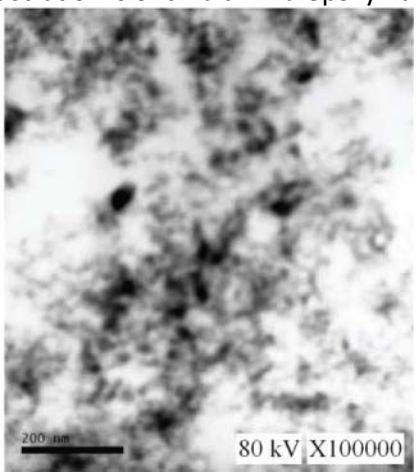


It may be noticed that the actual mode occurs at 280, and not zero. This is merely an artifact of the fact that black pixels occupy a majority of the image area. However, in computation of the agglomeration size (48nm in our case), this artifact is suppressed such that the value we obtain is the true characteristic agglomeration size. Zooming in toward the bottom of the histogram, we see that the 'second mode' is still at zero, indicative of our measurement:



Actual alumina-epoxy dispersion

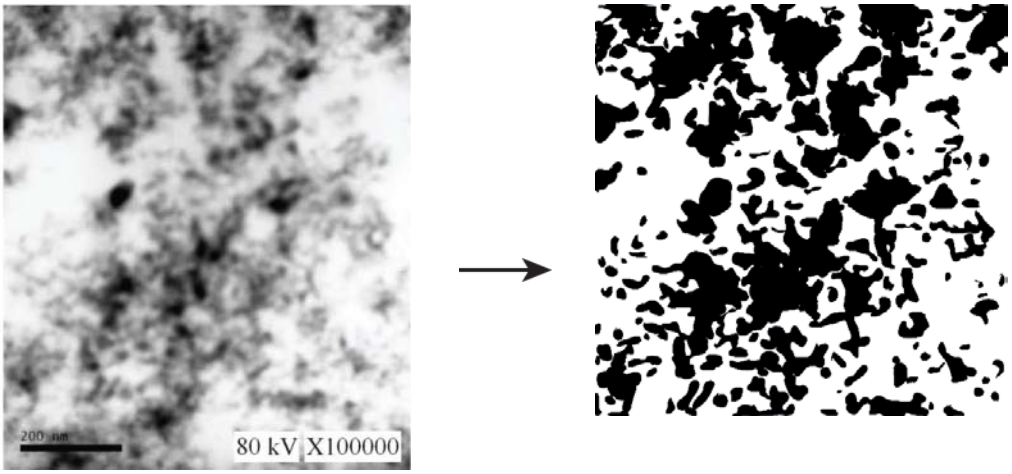
The TEM image used in this illustration is of an alumina-epoxy nanocomposites system[3]:



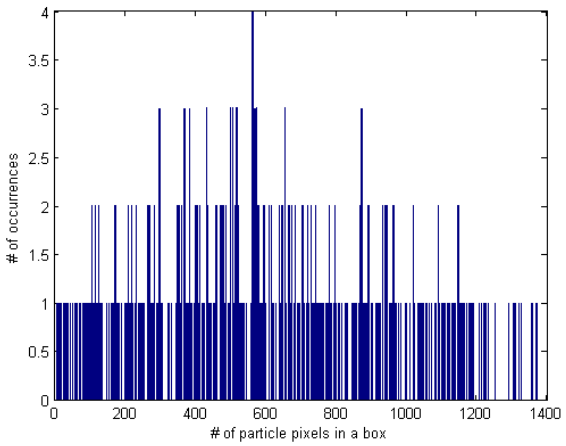
The following outlines the steps involved in calculating L_f and L_a from this image.

1. Before starting the code, we need to convert the given TEM image into a usable bitmap by identifying fillers from the matrix. Using, in our case, Adobe Illustrator, we identify filler locations and create the following bitmap:

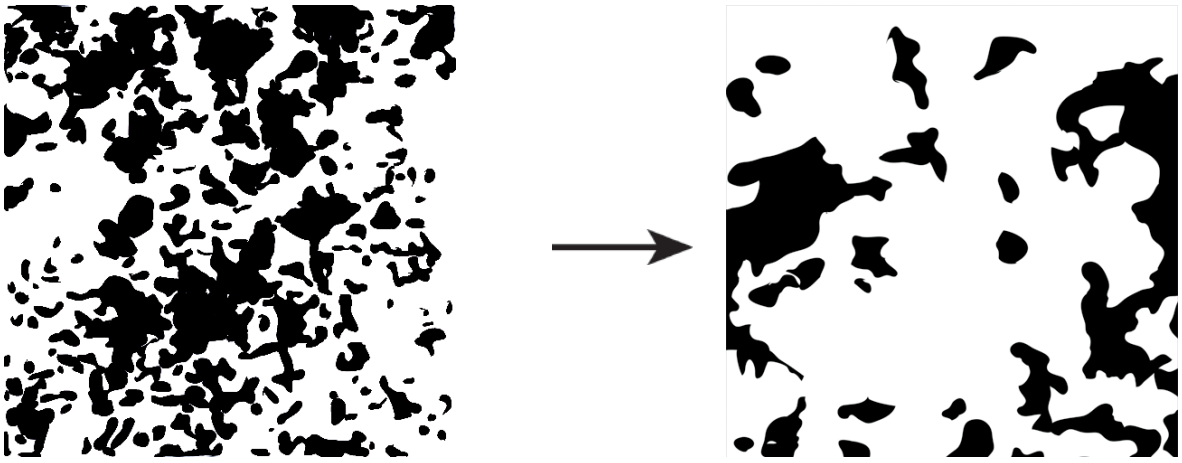
[3] Chen CH, Jian JY, Yen FS. Composites Part A – Applied Science and Manufacturing 2009;40(4):463-8



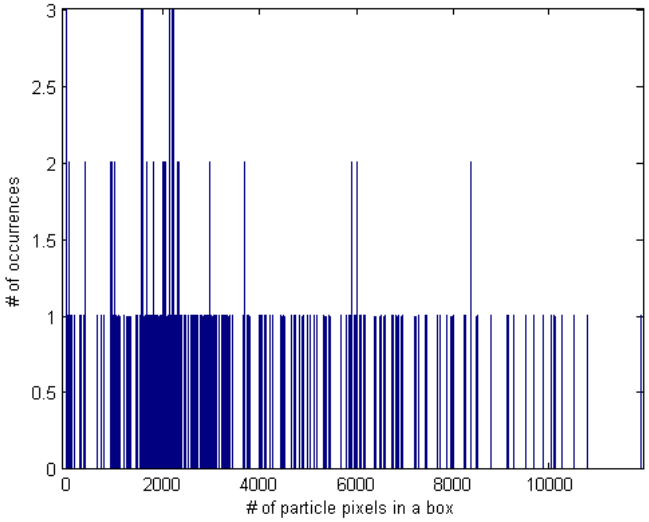
2. Next, as we did in the earlier example, we execute the .m file, and at the prompt, we select the file TEM.bmp
3. Proceeding as before, with our image size now approximately 800nm (adjudged from the scale bar at the lower left of the TEM micrograph), we note that the free space length now *seems* to be a little over a tenth, or, say, 100nm (slight overestimation). We choose to simulate 500 boxes again.
4. The simulation yields a free space length of 59nm in around a minute's time. The following histogram is also generated.



5. In order to evaluate the agglomeration length L_a, we invert the previous image whilst again identifying filler agglomerates to obtain the following image (remember, agglomerates will now be depicted in white after consolidation and color inversion):



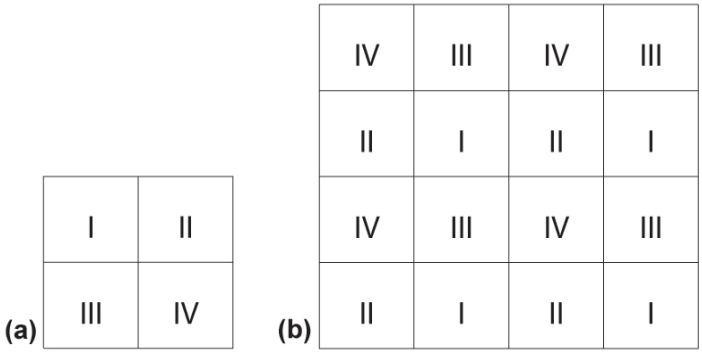
6. Repeating the steps performed earlier, we now select our starting size as a fourth of the image size, or about 200nm (500 boxes). The code converges to a value of 187nm in a little less than 3 minutes and generates the following histogram.



VI. Code algorithm:

The general algorithm of the code is as outlined:

1. A bitmap (.bmp) black/white TEM image (any pixel size, any aspect ratio) is imported to MATLAB™, and stored as matrix 'A', it's elements being the grayscale values associated with each of the pixels in the image.
2. With a threshold grayscale value of 150 to distinguish particles from the matrix, particle area fraction ('af') is evaluated.
3. A 'data' matrix is next generated, which has periodic boundary conditions. In other words, the data matrix is composed of the 'A' pixel matrix at the core, adjoined by portions of the 'A' matrix such that a periodic boundary is obtained at the edge of the 'data' matrix. This is illustrated in figure A1, where I, II, III, IV indicate the quadrants of the original 'A' matrix.
4. An initial value of box size is inputted from the user, together with the number of random boxes that the user wishes to analyze the image with and the size of the TEM image in nanometers.
5. The code converges on the characteristic box size, thereafter performing incremental 'jump ups' and 'trim downs' to converge to the actual characteristic box size. If the distribution is bimodal, the 'false' mode is suppressed in order to obtain the correct value of 'l'.
6. Results are displayed, together with a histogram of particle occurrences for the given 'Nsquare' boxes.
7. A Manual mode is provided at the end of the code, which in essence has the similar algorithm as the computation above, except for the automatic box-size adjustments (jump up/trim down) which cause the code to iterate to the characteristic value. Instead, the manual mode just provides the histogram for the value of box size and sampling size selected by the user.



(a) The image pixel-matrix 'A', with its four quadrants indicated by numerals. (b) The 'data' matrix, with 'A' at its core, and periodic boundaries along the edges. Note that if 'A' is of order 500x500 (say), the 'data' matrix is of order 1000x1000.