— Supporting Information —
Modelling Nanostructure in Graphene Oxide: Inhomogeneity and the Percolation Threshold

Robert C. Sinclair[†] and Peter V. Coveney[*,†,‡]

[†]*Centre for Computational Science - University College London, 20 Gordon Street, London, WC1H 0AJ, United Kingdom*

[‡]*Computational Science Laboratory, Institute for Informatics, Faculty of Science, University of Amsterdam, 1098XH, The Netherlands*

E-mail: robert.sinclair.15@ucl.ac.uk

In this supporting information, we provide additional discussions of the methods used in our work, including more details about the software which accompanies this article. This includes details about the extension of Yang *et al.*'s data set of graphene oxide reactivities in section I, an outline of the atomistic GO building program in section II, and more information about our percolation model in section III, IV, V and VI.

Methods for extending Yang's data set

Yang *et al.* used density functional theory to calculate the relative stability of different GO-MnO_4^- structures and thereby deduced the rates of reaction of various oxidation reactions. We refer the interested reader to their work.[1] We characterised the intermediate's structure as follows.

The MnO_4^- ion attaches to a pair of bonded carbons, the ion sitting on one face of a graphene(oxide) sheet. There are then 4 first neighbour carbon sites and 8 second neighbour carbon sites, the oxidation state of which significantly effect the stability of the intermediate structure. We record the number of first and second neighbour carbons with an alcohol group and epoxy group, and on which side of the flake they reside relative to the MnO_4^- ion. Given that approximately 16000 combinations of these parameters exist[1] and we only have a small set of 52 to train with, we must reduce the system's characteristics. We simplify the problem by assuming the direction of neighbouring functional groups is not important, and we only consider whether they are first or second neighbours. This assumption may become an issue when there are multiple neighbouring oxygen groups, but it has a good foundation based on Yang *et al.*'s work. They deduced that the reactivity is a function of broken adjacent π-bonds, steric availability, and hydrogen bond formation with the MnO_4^- ion, but it should be noted that they did not include directionality as an important factor. It is difficult to say if these factors are directly dependent on the relative positions of multiple functional groups; the most problematic case might be that steric availability may differ if two groups are adjacent or on opposite sides of the permangenate ion. However, reducing the feature set was necessary and we do not think it impacts the local structure significantly.

Now, each potential reactive site has 8 features: the number of alcohol or epoxy groups in first or second neighbour position and on the same or opposite side of the flake, relative to the MnO_4^- ion.

Given such a small training set, we found that some machine learning techniques did not perform well to predict the reactivity of different sites. The available data is far too sparse to train a neural network. However, a decision tree or random forest (RF) approach worked well (probably because the features were discrete. i.e. the number of first neighbour alcohol groups above the plane is an integer ranging from 0 to 4). We used the Scikit-learn software to generate our RF model:[2] the random forest had a maximum depth of 4 and the output is an average of 500 estimators.

It was also necessary, when training the RF, to use the logarithm of the reactivities. This is a method to regularise the data set and is necessary for target values such as these, which span many (31) orders of magnitude; otherwise the largest values would completely dominate the RF fitting process.

To validate our model the 52 known reactive sites we used various metrics to measure the model's success. The first was the coefficient of determination, R^2:

$$R^2(X) = 1 - \frac{\sum_i (y(X_i) - \overline{y(X)})^2}{\sum_i (y(X_i) - f(X_i))^2}. \tag{1}$$

For a data set X, the true rates are denoted $y(X)$, and the model predictions $f(X)$.

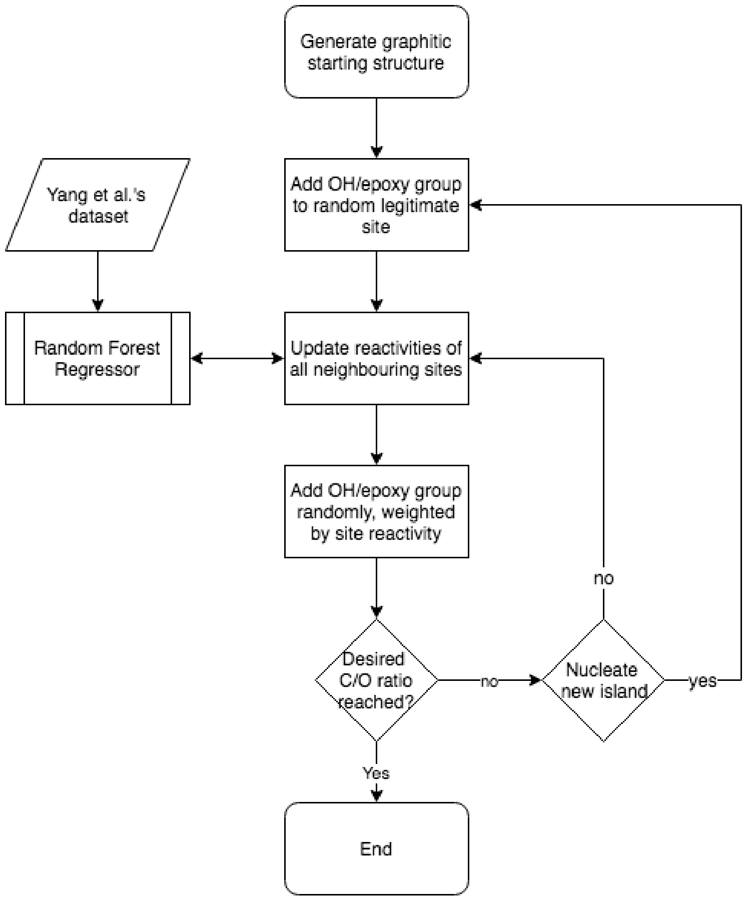
For this validation our data was split into a training, X_{train}, and test set, X_{test}, with a split of 39:13 (75% : 25%). The model was fitted to the training set and then assessed with the test set. This was repeated on several randomly generated training/test sets to cross-validate our scores.

Our RF model achieves $R^2(X_{\text{test}}) = 0.29$, which shows some level of correlation but is far from the ideal value of 1. Note that $R^2(X_{\text{train}}) = 0.72$, because we have reduced the features of the system, the model will never predict exact values for all of the training set because some of the information has been lost.

It is more important that this model correctly predicts the most reactive sites than the least, so the sum of all residuals is likely not to be the best measure of the model's suitability. The second method we use to evaluate the model is how well it predicts the most reactive structure from a subset. Taking a subset of size n, $S_n(X)$, of the set X, we can rank our model by how often it correctly predicts the most reactive sites and the average ranking of the site it predicts to be the most reactive.

For a subset $S_5(X_{\text{test}})$, the RF model predicts the most reactive site correctly 53% of the time. For a larger subset (reusing some of the training data points to increase the size) $S_{20}(X)$ it predicts correctly 30% of the time. By these measures the RF model performs far better than other methods we tried such as linear regression or multi-layer perceptron neural networks.

Atomistic graphene oxide builder program



Generate graphitic starting structure
Yang et al.'s dataset
Add OH/epoxy group to random legitimate site
Random Forest Regressor
Update reactivities of all neighbouring sites
Add OH/epoxy group randomly, weighted by site reactivity
no
Desired C/O ratio reached?
no
Nucleate new island
yes
Yes
End

Figure 1: Simple flowchart representing the general method taken to oxidise a graphitic structure using a random forest regressor to calculate site reactivites.

An outline of the way we oxidise a graphitic structure, using the program given in Ref.,[3] is shown in figure 1. The RF regressor described in the previous section is used to calculate the reactivities of graphene C-C bonds to permanganate oxidation.

As is described by Yang *et a*,[1] we assume that whether an alcohol or epoxy group forms after a permanganate ion binds to a C-C bond is random and equally likely.

The function that decides whether to nucleate a new island takes two arguments: a user specified ‘nucleation frequency’ (this is analogous to k_n, described in the main text); and the total reactivity of the system. We use the total reactivity of the system to predict the time elapsed between adding OH or epoxy groups. New islands are added according to a Poisson distribution, with mean equal to the ‘nucleation frequency’ multiplied by the time between adding functional groups.

Percolation model sensitivity

The percolation model has three input parameters: χ, which is discussed at length in the main text; δt, the timestep used to increment the islands’ radii and how often the algorithm attempts to add new islands; and the number of Monte Carlo points used to estimate the coverage area once the percolation threshold has been reached.

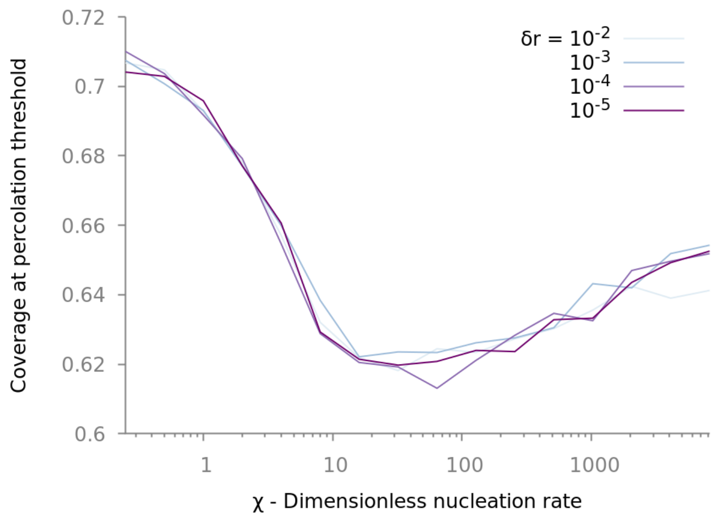


Figure 2: The fractional coverage of a sheet in oxidised islands against different values of χ, for diminishing discritisation. These are mean values of 1000 runs at each point; the mean is essentially unchanged for different precisions but the error in these values is correlated, as shown figure 3.

Recall that $\delta r = \delta t k_{rx} \sqrt{A}$. We can show that the qualitative outcome of the simulations, taken as the average coverage at the percolation threshold for different values of χ, is independent of the timestep used; see the plot in figure 2. However, the error in the result due to the change in coverage between timesteps is pronounced; see figure 3.

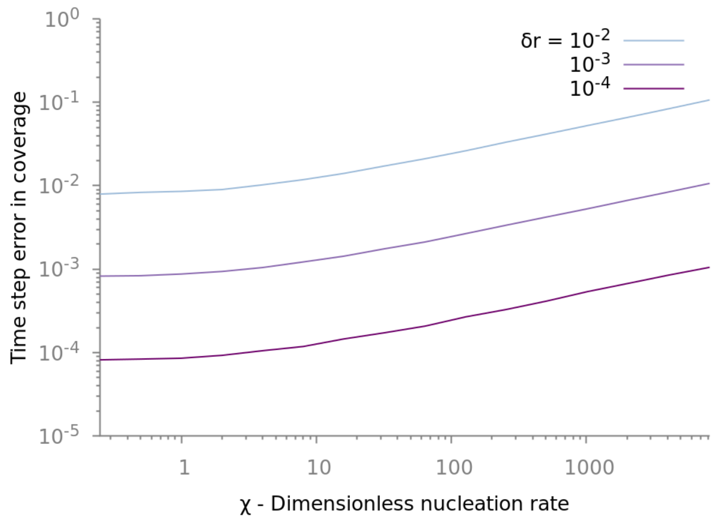


Figure 3: Average uncertainty in the coverage over 1000 runs for different precisions vs. the nucleation rate χ. The percolation threshold is reached at time c, therefore $\phi(t-\delta t) < \phi(c) < \phi(t)$. The uncertainty in the result is reported as $[\phi(t) - \phi(t-\delta t)]/2$.

The error reported in our method of calculating the coverage for a given system state is straightforward to explain. See figure 4, in which the errors shown are relative to the best estimate we had, namely that using 10^6 points.

Calculating the C/O ratio of an oxidised domain

For the continuum model, we approximated oxygenated regions to a circle with even density whereas in reality they are more irregularly shaped and have unoxidised islands that must be accounted for. Calculating where the effective boundary of this region lies (to approximate it as a circle) is not a trivial task. Knowledge of the region's effective area is necessary to compute the C/O ratio of an oxidised island.

Using the random forest algorithm described in the previous section, a sheet was nucleated only once and 10,000 oxygen atoms were added to analyse the resulting oxidised region. The

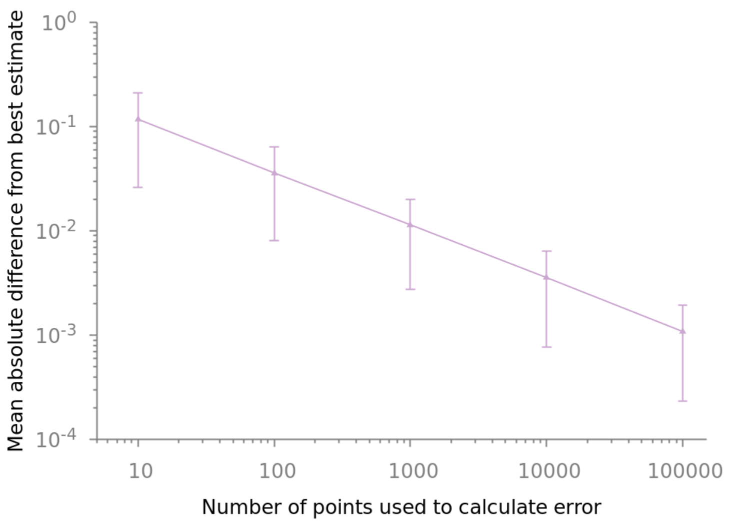


Figure 4: Average difference in the estimated coverage of the same set of 10000 systems, compared to an estimate using 10^6 points.

values discussed are taken from an ensemble of 15 such simulations.

We plot the O/C ratio as a radial distribution of oxygenated sites from the centre of mass of the oxidised region in figure 5 (the O/C ratio is used here so the value does not tend to infinity at large radii). The radial distribution forms a plateau near the centre of the island, which can be taken as the O/C ratio of a fully oxygenated region. The average O/C ratio between r =0-5 nm is 0.53. This cut-off is somewhat arbitrary but is necessary so that we can derive an effective area of the island. We can then say that the radius at which one is equally likely to find oxidised and unoxidised region has a O/C ratio of 0.26: reading from figure 5, this is at $r = 12.0 \pm 0.7$ nm.

However, this constitutes a lower bound to the effective radius of an oxidised island. We are concerned with when these islands overlap so we want to know about the island's boundary. The offshoots that grow from the edges constitute the long tail in figure 5 and are more important than the density at the centre.

One way to measure the area of the oxidised region is to find the area of the convex hull that encloses all the oxygenated sites. The convex hull can be thought of as the shape enclosed by an elastic band stretched around all oxygenated sites. The effective radius of this

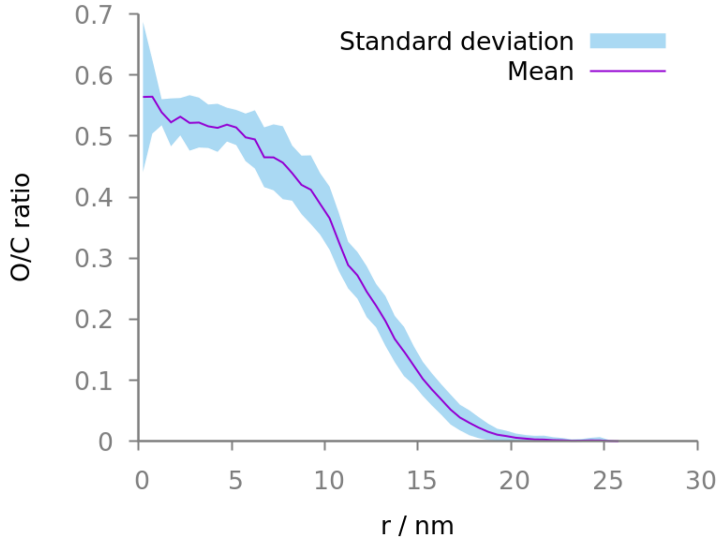


Figure 5: Oxygen density as measured from the centre of an oxidised island. The island has a higher oxygen density near the centre. We use an O/C ratio (instead of the usual C/O) because it does not rise to infinity at large radial distance. It is not immediately obvious what the effective radius of this island is.

island would be the radius that gives a circle with the same area as this convex hull. Over the same 15 simulations, the radius calculated by this method is 151 ± 6 nm. This method captures the area around the irregular boundary. Some examples are shown in figure 6.

In reality, the effective radius of these islands probably lies between the two values calculated. Any value in between can be arrived at by constructing the concave hull of the island with a given shape parameter. A concave hull is a polygon which contains all the oxygenated sites but has less area than a convex hull; several algorithms exist which can be used to construct such a polygon. Here, we constructed a Delaunay triangulation using the coordinates of all the oxygen sites, then removed the triangles with edges over length α.[4] We find the outer edges of the resulting mesh and use this boundary to compute the effective area of the island. Examples are shown in figure 7. The code used for this is given at the end of the Supporting Information. By sweeping through different values of α, the relationship in figure 8 is found.

The carbon to oxygen ratio is then calculated by dividing the number of carbon atoms that fall within the effective area of the island by the number of oxygen atoms added (10,000).

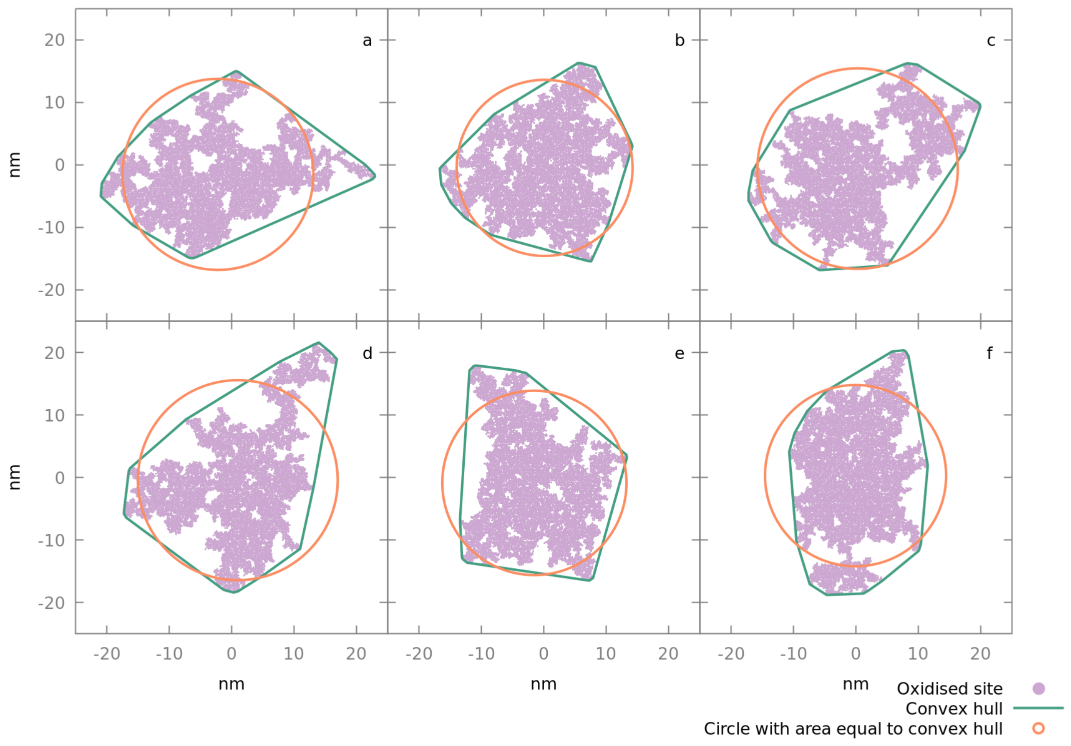


Figure 6: This figure shows 6 examples, (a)-(f) of oxidised domains. We are trying to calculate the effective radius of these islands to map to the continuum model described in the main text. To do this we draw the convex hull of the island (green) and select a radius which gives the same area, which is also shown in each case (orange).

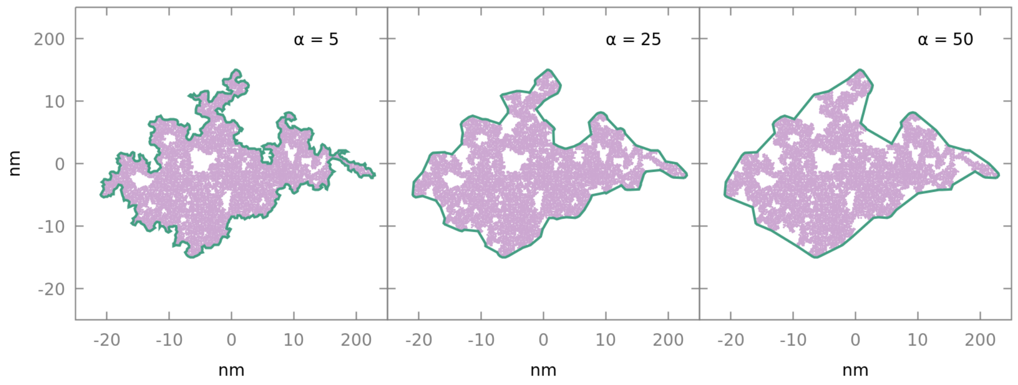


Figure 7: The convex hull drawn in figure 6a probably overestimates an island's effective radius. Here we show the same island but draw the concave hull with different α values. The exact algorithm to produce these boundaries is given in the text.

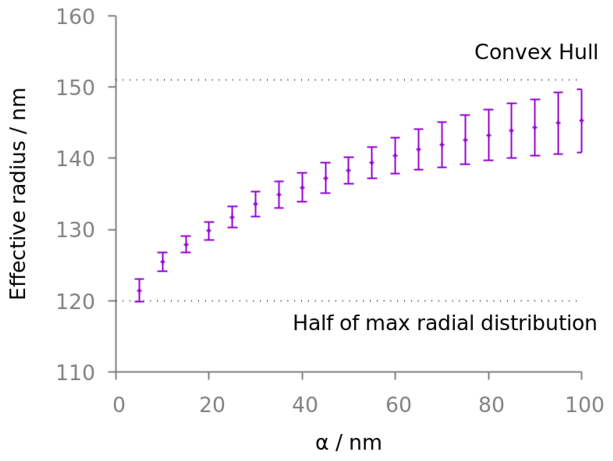


Figure 8: The effective radius of an oxidised domain with 10,000 oxygen atoms calculated by different means. The convex hull method provides a maximum, and the half maximum of the radial distribution of oxygen atoms is a minimum. The real value is expected to lie between these values.

The density of carbon atoms on a graphene sheet is 38.46 carbons per nm^2. Using the first method described (half the radial distribution plateau), the C/O ratio within a propagating oxidised island is 1.94 ± 0.20. Using the area of the convex hull, C/O $= 2.76 \pm 0.22$. In the main text we use a ratio of 2.76 which is the most conservative estimate for calculating the percolation threshold of graphene oxide. The true answer is likely to be smaller.

Percolation Threshold

Figure 4 in the main text shows error bars as the 95% confidence interval in the average. We show here the same data but with the median and percentile which give a different view of the data, in figure 9.

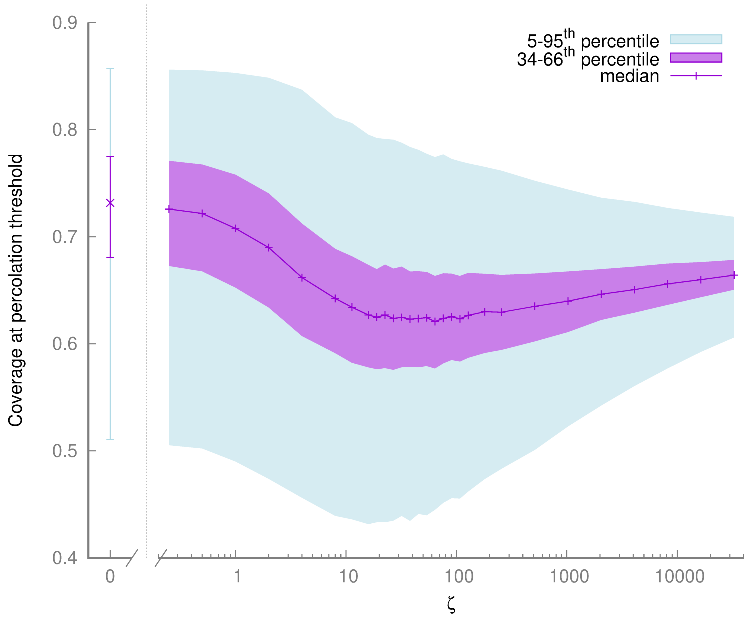


Figure 9: The fractional coverage of a graphene sheet at the percolation threshold for different values of χ. We show different percentiles for the distribution of coverages calculated to indicate the shape of these distributions.

Recalling that $\chi = Ak_n/k_{rx}$, a real sample of graphene flakes will have surface areas that cross several orders of magnitude. This is inevitable by any of today's current methods of synthesising graphene. Therefore, a sample of graphene will not have a unique value of χ that can describe it. It is clear, however, that there is minimum value that ϕ_c takes and within one standard deviation ϕ_c will allways be greater than 0.58. Similarly, we can say that, when $\phi_c > 0.43$, at least 95% of the sheets will be below the percolation threshold, whatever the flake size distribution. In the main text we use a coverage of 0.43 to define the conservative estimate of graphene oxide that lies below the percolation threshold.

Concave hull code fragment

Below is a snippet of python code used to calculate the concave hull of an graphene-oxide island, modified from the code found in Ref.[4]

```python
def alpha_shape(points, alpha):

    def add_edge(edges, edge_points, coords, i, j):
            if (i, j) in edges or (j, i) in edges:
                return
            edges.add( (i, j) )
            edge_points.append(coords[ [i, j] ])

    tri = Delaunay(points)
    edges = set()
    edge_points = []
    # loop over triangles:
    # ia, ib, ic = indices of corner points of a triangle
    for ia, ib, ic in tri.vertices:
        pa = coords[ia]
        pb = coords[ib]
        pc = coords[ic]
        # Lengths of sides of triangle
        a = math.sqrt((pa[0]-pb[0])**2 + (pa[1]-pb[1])**2)
        b = math.sqrt((pb[0]-pc[0])**2 + (pb[1]-pc[1])**2)
        c = math.sqrt((pc[0]-pa[0])**2 + (pc[1]-pa[1])**2)

        if a > alpha or b > alpha or c > alpha:
            pass
        else:
            add_edge(edges, edge_points, points, ia, ib)
            add_edge(edges, edge_points, points, ib, ic)
            add_edge(edges, edge_points, points, ic, ia)
    m = geometry.MultiLineString(edge_points)
    triangles = list(polygonize(m))
    concave_hull = cascaded_union(triangles), edge_points
    return concave_hull
```

References

(1) Yang, J.; Shi, G.; Tu, Y.; Fang, H. High Correlation Between Oxidation Loci on Graphene Oxide. *Angewandte Chemie International Edition* **2014**, *53*, 10190–10194.

(2) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python . *J. Mach. Learn. Res* **2011**, *12*, 2825–2830.

(3) Sinclair, R. C. make-graphitics. 2019; `10.5281/zenodo.2642230`.

(4) Dwyer, K. Drawing Boundaries In Python - http://blog.thehumangeo.com/2014/05/12/drawing-boundaries-in-python/ Accessed: 2019-01-9.