Hi Chris,

here are promised notes that hopefully clarify what my code does and how you can help to make these calculations more efficient.

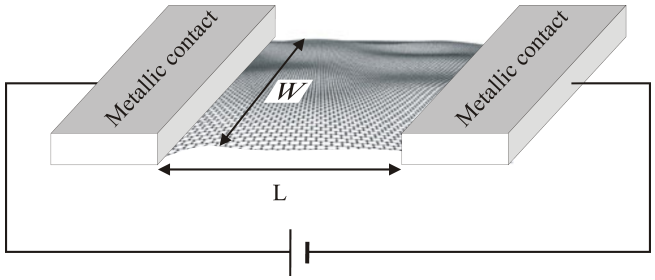

Figure 1: The setup we are studying: a graphene sheet is attached to metallic leads, a voltage U is applied and the current I is measured. The ratio $G = I/U$ is called conductivity and it's this quantity we want to calculate.

As you can see in above picture, the graphene sheet, we are looking at, is not flat but is rippled in a random way. The consequences of this ubiquitous rippling for electronic transport are not yet fully understood. To find the conductivity for such a randomly rippled graphene sheet, we compute the conductivity for many different concrete configurations of rippled graphene and then average over these results. As the calculation for a single configuration already takes quite long, it would take ages to do that for very many samples. But this task is very much suited to be optimized by making it parallel (`http://en.wikipedia.org/wiki/Parallel_computing`) and this is the part where your skills are instrumental.

The people from the parallel computing community call the procedure you'll use "trivial paralellelism":

"[Trivial parallelism] does not involve parallelising the code at all, rather the sequential code is run on a number of processors in parallel, with each processor having its own set of input parameters. In order to do this the different instances of the sequential code must be independent of each other. This implies that one version of the simulation cannot depend on the results of another simulation which is being run at the same time. Each processor is working independently and there is no communication between processors. [...] In summary, trivial parallelism can offer a simple method of obtaining results by running multiple versions of the same simulation with different input parameters in parallel."[1]

In simple words: instead of calculating the conductivities of the different samples one after the other on the same machine, you make sure that many processors do that simultaneously; after that you collect their results and compute the average. To do so you don't really need the details of my code; all you need to know is summarized in Fig 2.

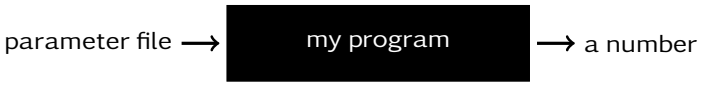

Figure 2: Look at the program as if it was just a black box that takes a parameter file as an input and gives a number as an output. You would use the program by a command like `./conductivity paramFile.txt`. The program would write the calculated conductivity simply in the stdout so you can pipe it wherever you need it.

The code I sent you last week is quite similar to the real code in the sense that it also writes a number to stdout (it doesn't need to have a parameterfile as an input, though). You can use this fake code to test your script after you implemented the parallelism: compile and run the test code on several machines (maybe several hundered times), collect the computed numbers and find their average. For that test code it's known that the average will tend to $1/\pi$.

[1] `http://www.epcc.ed.ac.uk/library/documentation/training/`