Guide to Student Assessment for the "Parallel Numerical Simulation of Boltzmann Transport in Single-Walled Carbon Nanotubes" learning module

There are several student exercises presented in the text. Most of them are rather open-ended so that students can explore the code and learn from hands-on experience running the provided codes. A few of the questions require derivation. Here are a few comments to guide in assessing the student performance in answering the questions, listed by chapter at the end of which they appear.

Boltzmann Transport Simulation of Single-Walled Carbon Nanotubes:

1. The students should plot current vs. time for several values of the relaxation time constant τ to show that increasing the relaxation time slows the response down and leads to a longer time being required to reach steady state.
2. Increasing the timestep above the value allowed by the CFL conditions leads to numerical instability (there are two, one related to velocity and Δk, and the other related to electric field and the real space mesh size Δx).
3. When the size of the mesh in momentum space (Δk) decreases below the value allowed by the CFL condition ($|v(k)|* \Delta t/ \Delta k<1$), the solution becomes unstable, as can be seen from the rapid oscillations in the solution.
4. Initially, the stability criterion for the spatial discretization is much easier to satisfy than the Δk in the previous question, so the instability might not show up. However, increasing the applied voltage makes the CFL condition $|F(x)|* \Delta t/ \Delta x<1$ more restrictive on the mesh size Δx so the instability becomes more apparent.
5. Based on the answers to the previous two questions, it is much easier to see the instability from decreasing the momentum space mesh size Δk then the real space mesh size Δx. This is because the velocity $v(k)$ is typically higher than the applied field $F(x)$. However, increasing the applied voltage $V(x)$ can change the CFL condition for the real space mesh Δx, and by increasing the voltage (or, equivalently, decreasing the length of the nanotube), we can make the $|F(x)|* \Delta t/ \Delta x<1$ condition more restrictive.

Note: the discussion is deliberately left more qualitative since the actual velocities $v(k)$ are not constant and the condition can vary from one value of momentum "k" to another.

Parallel Programming With The Message Passing Interface

1. The value of the number of processors “p” which leads to $E_p=1$ is p=1. The efficiency is highest for a single processor due to there being no difference between parallel and serial implementation, so no additional overhead due to the serial fraction to diminish efficiency. This does not imply, however, that the fastest implementation is the serial one, just that any parallel implementation will be less than “p” times faster than the serial one due to there being some portion of the code which is not perfectly parallel.
2. This question requires that we take the limit as “p” gets large to see that all the terms except for the 1/s term drop out, leaving the limit that no matter how many processors we throw at a problem, we are always limited by the serial fraction.
3. Plug in $S_p=1/16$ into the expression for Amdahl’s law to show that the serial fraction “s” is approximately 1/32. From this we can conclude that there is not much use utilizing more than 1/s processors if we cannot eliminate the serial fraction s of work that cannot be parallelized.
4. Here we need to show that the speed up is $S_p = p*N_x*N_k / (N_x*N_k + 2*N_k + 2)$ by dividing the serial running time, given by $T_s=N_x*N_k$ by the parallel running time, given by the sum of total computation and communication times, $T_p=T_{calc}+T_{comm}= N_x*N_k / p + 2*N_k + 2$.
5. Similar to 4.), but here we have a different parallel $T_p=T_{calc}+T_{comm}= N_x*N_k / p + 2 N\_x/\sqrt{p} + 2*N_k/\sqrt{p} + 2$, which leads to the speedup being given by $S_p = p*N_x*N_k / (N_x* N\_k + 2 N\_x/\sqrt{p} + 2*N_k/\sqrt{p} + 2)$.
6. As can be seen from the solutions to the previous two problems, the one-dimensional decomposition is more efficient for smaller values p, while the larger values of p favor the two-dimensional (and more general) decomposition. The figures should show that the 2d decomposition from figure 5 scales better with increasing number of processors.