Only one lab this week.

Due date/time Saturday, February 25, 9 pm.

[HW6-1] Write a code to solve the Laplace equation inside a rectangular box $0 < x < L_x$ and $0 < y < L_y$ with boundary conditions $V = 0$ on the bottom ($0 < x < L_x; y = 0$); $V = 0$ the left side ($x = 0; 0 < y < L_y$); $V = 0$ on the right side ($x = L_x; 0 < y < L_y$); and $V = V_0$ on the top ($0 < x < L_x; y = L_y$).

[HW6-2] Run your code for $L_x = 3, L_y = 2, V_0 = 8$ and box size $dx = dy = 0.1$. Make a plot of the potential along the horizontal line $y = 1$ bisecting the box. In your plot, put the analytic solution (see posted notes and discussion in class) and also your numerical solution for different numbers of iterations: $N_{\text{its}} = 100, 200, 500, 1000, 2000$. How long does your code take to run with 1000 iterations, and how does that compare to a simple estimate based on a 3 GHz processor?

[HW6-3] Do the same as in [HW6-2] but with $dx = dy = 0.01$. (I found that now I needed $10^5$ iterations to get good agreement with the analytic solution!) Can you give an argument for why the code needs more iterations? How long does your code take to run with $10^5$ iterations, and how does that compare to a simple estimate based on a 3 GHz processor? Since $N_{\text{its}} \sim 10^3$ was pretty good for $dx = dy = 0.1$, but you need $N_{\text{its}} \sim 10^5$ for $dx = dy = 0.01$, it is reasonable to surmise that $N_{\text{its}} \sim (1/dx)^2$. If that were true, how long would your code take to run for $dx = dy = 0.001$?

[HW6-4] Modify your code so that it has $V = 8$ also along the right side of the box, in addition to the top. Run your code for $dx = dy = 0.01$.

[HW6-5] Extra Credit: Can you figure out the analytic solution?
Hint: Use the superposition principle.

[HW6-6] Extra Credit: Write a Laplace code which updates the potential ‘on the fly’, i.e. without using a separate array for the new potential. We discussed this a bit in class, e.g. how the value of $V$ will break various geometric symmetries and depend on the order with which you visit the boxes. Violating symmetries seems very bad to a physicist. Does your code converge to the right answer? (Is symmetry ‘restored’ after many iterations?) Does your code converge faster or slower than with a separate array?