18.336 Homework 3 hints :: Spring 2010

- Question 2. Getting the desired outcome for this system of equations can be tricky. In my implementation, I needed N (the number of points per dimension) to be at least 100 to get the correct behavior for the solution. But above $N = 200$ or 300 the solver gets really slow. In your convergence experiment, it is fine if you consider values of N in that neighborhood. They can be closer to each other than powers of 2.

  For me the steady state was reached at time $T \simeq 100$, so be prepared to run long simulations. As usual, make sure that the numerical method reaches exactly T, so make sure Δt divides T, otherwise you may be comparing solutions at different times.

  I recommend starting implementing the (faster) FD code with $N = 128$ so you kow what behavior to expect. Visualize the solution with imagesc, and do a plot every 10 frames or so. Remember that the BC are periodic for the FD code as well.

  For the spectral code I recommend first making sure that you get the Laplacian right by trying it out on a very smooth function like a low-wavenumber sine wave, and comparing it with the 5-point Laplacian. Be mindful what to multiply the Fourier transform with $(-|k|^2, -4\pi^2|k|^2?)$ as we are dealing with the square $[0, 1]^2$ in (x, y), not $[0, 2\pi]^2$.

  If you want to play a bit with the system of equations, random initial conditions can be fun to try: take them small enough that the code does not blow up, and large enough that the solution does not tend to a constant. You can play with the parameters as well. In particular, α just rescales the problem – it is the characteristic length scale of the solution.

- Question 3. A "tensor" Chebyshev grid is the natural 2D version of the Chebyshev grid, see p.67 of Trefethen's book. For the Chebyshev differentiation matrices, you have the choice of either using the explicit matrices in Chapter 6 of Trefethen, or the faster FFT method of Chapter 8. In both cases, code is available to form these matrices (see cheb.m and chebfft.m), so you can use that. Some code is also available for the wave equation (Programs 19 and 20) if you need inspiration.

  Implementing Dirichlet BC with Chebyshev polynomials is easy – see Chapter 7 in Trefethen. Implementing Neumann boundary conditions in a little more tricky, but the book has the solution. As in the case of finite differences, at each time step the problem is to update the value of the unknown at the boundary from the knowledge that the normal derivative is zero. This time, the normal derivative is computed spectrally using Cheb poly again. See the lines that concern BC in Program 37. The line that defines BC contains a small matrix inversion.

  Program 37 also contains a nice trick for Chebyshev differentiation in 2D, using only the 1D differentiation matrices. See how D2x multiples vv on the right, and D2y multiplies vv on the left in the line vvnew = ... in Program 37. In your code you should never have to form N^2-by-N^2 differentiation matrices.