Problem 1. *Write programs to solve*

$$u_t = \Delta u \text{ on } \Omega = (0,1) \times (0,1)$$

$$u = 0 \text{ on } \delta\Omega$$

$$u(x,y,0) = exp(-100((x-0.3)^2 + (y-0.4)^2))$$

to time $t = 1$ using Forward Euler and Crank-Nicolson. For Crank-Nicolson use a fixed time step of $\Delta t = 0.01$, and for Forward Euler use a time step just below the stability requirement.

(a) Time your codes for different grid sizes and compare the time to solve using Forward Euler and Crank-Nicolson.

I altered my Crank-Nicolson code from last assignment to solve the 2D diffusion equation with Dirichlet BC's, implementing the 2D discrete Laplacian instead of the 1D, and changing the setup and initial conditions appropriately. I used a fixed time step of $\Delta t = 0.01$ and varied grid spacings from $h = \Delta x = \Delta y = 2^{-1}, 2^{-2}, \ldots, 2^{-7}$ to solve the diffusion equation.

I coded Forward Euler for the 2D diffusion, and solved it on the same grid spacings h. The time step requirement for FE stability for the 2D diffusion equation is

$$\Delta t \leq \frac{h^2}{4D},$$

and since $D = 1$ in this problem, I used a time step of $\Delta t = \frac{h^2}{5}$. The runtime comparison of the two methods is given in Table 1. FE is faster for coarse grid spacings due to the overhead in solving the linear system in C-N, but C-N easily outperforms FE for finer grid spacings from 2^{-4} and beyond.

Table 1: Runtime Comparison for C-N, FE for 2-D Diffusion

grid spacing h	Crank-Nicolson Run time (seconds)	Forward Euler run time (seconds)
2^{-1}	0.144207	0.010568
2^{-2}	0.140511	0.027865
2^{-3}	0.110168	0.10511
2^{-4}	0.158117	0.440522
2^{-5}	0.35293	2.15241
2^{-6}	2.88994	11.1677
2^{-7}	14.1016	101.184
2^{-8}	95.351819	———

(b) In theory, how should the (run)time scale as the grid is refined for each algorithm? How did the (run)time scale with the grid size in practice?

For CN, we used the same number of time steps for each solve, so that doesn't factor into the runtime scaling. Each timestep, however, required a linear solve with an $N^2 \times N^2$ sparse matrix, which takes $\emptyset(N^2)$ work. Each $N = 1/h = 2^i$, so the work at each time step should have been 2^{2i}. The next grid spacing level would have $N = 2^{i+1}$ and thus 2^{2i+2}

work, for a successive ratio of $2^2 = 4$. Thus the time to solve should scale by a factor 4. In practice for CN, this scaling was never achieved, as shown in Table 2.

Table 2: C-N Runtime Scaling

grid spacing h	Crank-Nicolson Run time (seconds)	Ratio
2^{-1}	0.144207	———
2^{-2}	0.140511	0.974
2^{-3}	0.110168	0.78
2^{-4}	0.158117	1.44
2^{-5}	0.35293	2.23
2^{-6}	2.88994	8.19
2^{-7}	14.1016	4.88
2^{-8}	95.351819	6.76

For FE, the time step size was $\Delta t = h^2/5$, for grid spacings $h = 2^{-1}, \ldots, 2^{-7}$. Thus at each grid level $h = 2^{-i}$, there were $5 \cdot 2^{2i}$ time steps total. Each timestep requires multiplication by a $2^{2i} \times 2^{2i}$ matrix, so 2^{2i} work. At the next level, there are then a factor of 2^2 more time steps and a factor of 2^2 more work involved at each step, so the runtime should scale by a factor of 16.

In practice for FE, the runtime scaling factor increases as the runtime increases, possibly the factor of 16 is achieved asymptotically. The runtime scaling factors are shown in Table 3.

Table 3: FE Runtime Scaling

grid spacing h	Forward Euler run time (seconds)	Ratio
2^{-1}	0.010568	———
2^{-2}	0.027865	2.64
2^{-3}	0.10511	3.77
2^{-4}	0.440522	4.19
2^{-5}	2.15241	4.89
2^{-6}	11.1677	5.19
2^{-7}	101.184	9.06

(c) For this problem we could use an FFT-based Poisson solver which will perform the direct solve in $\emptyset(N \log(N))$, where N is the total number of grid points. We could also use multigrid and perform the solve in $\emptyset(N)$ time. How should the time scale as the grid is refined for Crank-Nicolson if we used an $\emptyset(N)$ solver?