GLY 5828, Assignment 3:

1. Derive the heat (diffusion) equation.

2. Derive a finite difference expression for the heat equation.

3. Use D = 10^{-5} cm^2s^{-1} and simulate diffusion in a 12-cm long, 1-D domain. Use C(0,t) = 1 at the lhs boundary. Because we wish to compare with the analytical solution $C = erfc\left[\frac{x}{\sqrt{4Dt}}\right]$ (which assumes an infinite domain with C(∞,t) = 0), we must limit our investigation to a similar numerical solution and the right boundary ideally won't affect the solution (i.e., put it far from any 'action'). Solve for the concentration profiles at 10,000, 100,000, and 200,000 seconds using a spreadsheet solution of the fully explicit numerical algorithm, Mathematica, and the analytical solution. Plot the analytical solution as a solid line and plot the numerical solution as open symbols on the same chart.

The "inconsistent boundary/initial condition" "warning" you may be getting in Mathematica appears to be critical in that it prevents the program from obtaining the solution.

It arises because the left BC C(0, t) = 1 is inconsistent with the IC C(x, 0) = 0. It needs to be addressed by the methods outlined in the help (i.e., replacement of one of the conditions by an approximate condition involving an Exp function).

If you choose the left BC, then you want something that is zero at t = 0, but then immediately switches to 1 at t > 0 like C[0, t] == 1 - Exp[-1000*t].

4. Do 3 again for D = 5 * 10^{-5} cm^2s^{-1}.

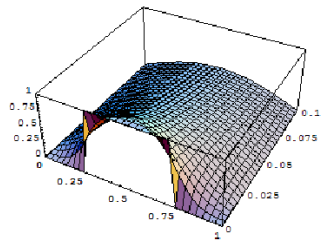
5. Here is another analytical solution for diffusion:

$$C = \frac{C_0}{2}\left[erf\,\frac{h-x}{\sqrt{4Dt}} + erf\,\frac{h+x}{\sqrt{4Dt}}\right]$$

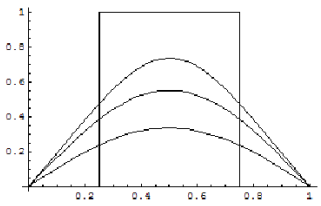
The h is the half-width of an initial source centered at x = 0 cm. The error function (erf) can be found in Excel but it has problems with the negative arguments that arise: use MATLAB. The boundary conditions are at infinity. Use an explicit spreadsheet and Mathematica to numerically solve for the concentration profiles at 100, 2000, and 11000 seconds in a domain from -500 to +500 cm with D = 1/6 cm^2 s^{-1} for h =10 cm and C_0 = 1. That is, the initial condition is C(x<-10,0) = 0,
C(-10<x<10,0) = 1, and C(x>10,0) = 0. Compare the results to the numerical solution.

Here is some Mathematica code that should help get you started.

```
In[1]:= Remove["Global`*"]
        solution = NDSolve[{D[f[x, t], t] == D[f[x, t], x, x], f[x, 0] == UnitStep[x - 0.25] - UnitStep[x - 0.75], f[0, t] == 0, f[1, t] == 0}, f, {x, 0, 1}, {t, 0, .1}]

        Remove::rmnsm : There are no symbols matching "Global`*". More...

        NDSolve::mxsst : Using maximum number of grid points 10000 allowed by the MaxPoints or MinStepSize options for independent variable x. More...

Out[2]= {{f → InterpolatingFunction[{{0., 1.}, {0., 0.1}}, <>]}}

In[3]:= Plot3D[Evaluate[f[x, t] /. First[solution]], {x, 0, 1}, {t, 0, 0.1}, PlotRange → {0, 1}, PlotPoints → 30]
```




```
Out[3]= - SurfaceGraphics -

In[4]:= t = {0, .025, .050, .1}
        Plot[Evaluate[f[x, t] /. First[solution]], {x, 0, 1}]

Out[4]= {0, 0.025, 0.05, 0.1}
```




```
Out[5]= - Graphics -
```

Think carefully about the initial condition and how it is being specified by the UnitStep functions. You might want to try a plot like the following (where you fill in the arguments x1 and x2) to make sure everything is correct:

Plot[UnitStep[x1] - UnitStep[x2], {x, -100, 100}]

Also, you might want to manipulate the value in the "PlotPoints -> " option in the 3D plot (if you make it) in order to see more detail of the solution. Be warned though; it will hang your computer if you go too high.

Finally, I found it critical to use the PlotRange -> {0, 1} option in the 3-D plot.

6. Address in your write-up the quality of the numerical solution relative to the analytical solution. Are there systematic deviations? Are there changes in the quality of the match with time or diffusion coefficient? If so, why?