Luke Buschmann

AMS 213: Numerical Solutions

Project 2: Final

PDE Solver: Advection Diffusion Equation

6/8/2014

Question 1 : Stability of the Crank-Nicholson algorithm for the 1D diffusion equation.

Using a Von Neumann analysis on the 1D diffusion equation, we have the discretized function below:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$$

Which we get an amplification factor $G(k)$

$$G(k) = \frac{1 - \frac{D\Delta t}{\Delta x^2}(1 - cos(k\Delta x))}{1 + \frac{D\Delta t}{\Delta x^2}(1 - cos(k\Delta x))}$$

where $|G(k)| < 1$ for stability. Since this condition is true for all values of Δt, the Crank-Nicholson algorithm is *unconditionally stable* for the 1D diffusion equation.

Question 2 : 2D diffusion equation

I implemented both the Crank-Nicholson and Forward Time Centered Space (FTCS) routines for solving the 2D diffusion equation.

The crank-nicholson solver operates by first calling an initialization function and then by calling the solver at each timestep. The solver performs a cholesky back substitution at each timestep to obtain a new mesh u. The FTCS routine is fast as it only performs scalar

multiplications on a matrix to obtain an updated mesh u. The user configures both routines from the driver routine (projectfinal.f90).

The driver routine first create vectors x and y populated with points along the mesh. In this problem, our mesh is the range $x = [0,1],\ y = [0,1]$ with the number of mesh points specified by the user (bigI and bigJ). The user sets values for initial time (t0), final time (tf), timestep (dt), and diffusion coefficient (D). The user enters the function for the initial mesh u at time zero and enters dirichlet boundary conditions for each side of the square mesh.

When the user executes the main driver routine, it will prompt him/her for a choice of which routine to execute (FTCS or Crank-Nicholson). The user must be cautious of the timestep used in the FTCS routine as it may diverge if the timestep if too large. The Crank-Nicholson is unconditionally stable (shown in Q1).

The timestep used in the examples shown is *dt=0.00001*. From t0=0 to tf=1, the mesh converges in less than 0.1 seconds with a D value of 1.

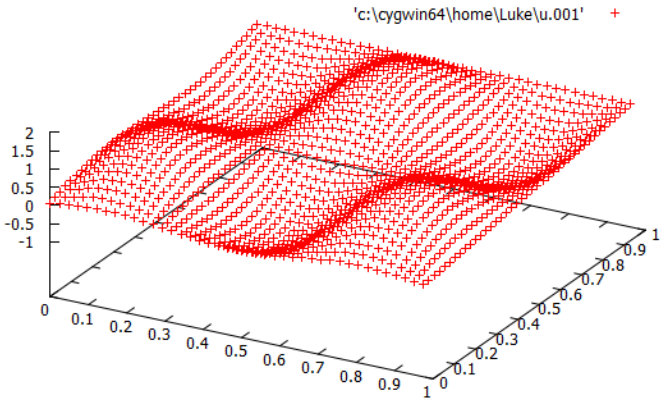
```
! Subroutine CRANK_NICHOLSON_2D_INIT is the initialization routine for the
!   Crank-Nicholson solver for the 2D diffusion equation. It creates the A matrix
!   and performs a cholesky decomposition for use in the solve routine.
!
! ** Input parameters **
! x : input x vector (evenly spaced)
! y : input y vector (evenly spaced)
! bigI : input mesh I value
! bigJ : input mesh J value
! dt : timestep
! D : diffusion coefficient
! fg1,fg2,fg3,fg4 : dirichlet boundary conditions
! f3a,f3b : advection boundary functions
! A : out matrix 'U' cholesky decomposed matrix of LHS (n+1 timestep)
SUBROUTINE CRANK_NICHOLSON_2D_INIT(x,y,bigI,bigJ,dt,D,fg1,fg2,fg3,fg4,A)
```

```
! Subroutine CRANK_NICHOLSON_2D_SOLVER is a Crank-Nicholson solver
!       for the 2D diffusion equation
!
! ** Input parameters **
! x : input x vector (evenly spaced)
! y : input y vector (evenly spaced)
! bigI : input mesh I value
! bigJ : input mesh J value
! dt : timestep
! D : diffusion coefficient
! fg1,fg2,fg3,fg4 : dirichlet boundary conditions
! f3a,f3b : advection boundary functions
! A : 'U' cholesky decomposed matrix of LHS (n+1 timestep)
! u : output solution mesh matrix
SUBROUTINE CRANK_NICHOLSON_2D_SOLVER(x,y,bigI,bigJ,dt,D,fg1,fg2,fg3,fg4,A,u)
```

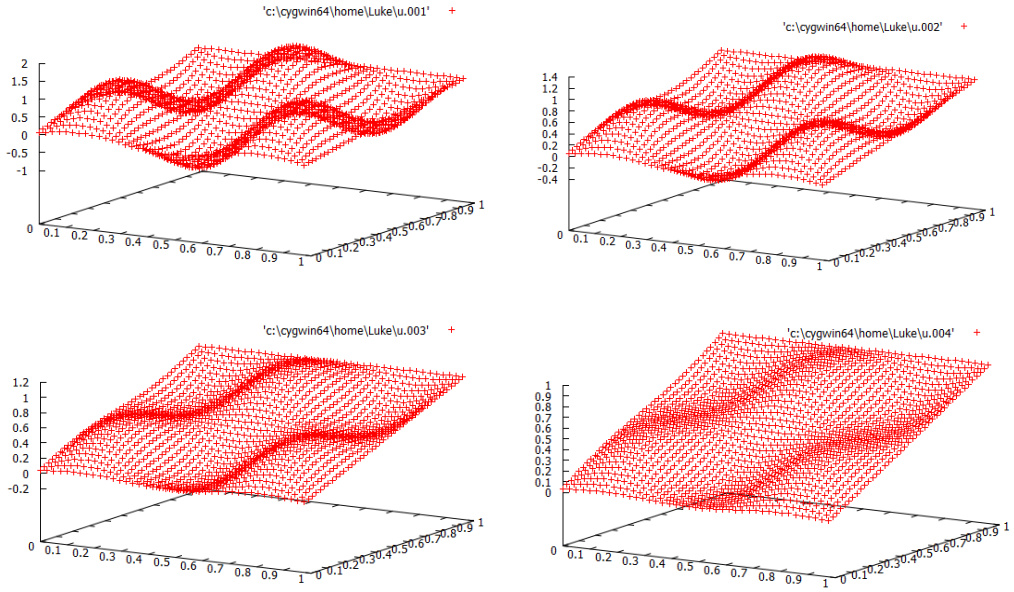
```
! Subroutine  FTCS_2D_SOLVER is a FTCS solver for the 2D diffusion equation
!
! ** Input parameters **
! x : input x vector (evenly spaced)
! y : input y vector (evenly spaced)
! bigI : input mesh I value
! bigJ : input mesh J value
! dt : timestep
! D : diffusion coefficient
! fg1,fg2,fg3,fg4 : dirichlet boundary conditions
! u : output solution mesh matrix
SUBROUTINE FTCS_2D_SOLVER(x,y,bigI,bigJ,dt,D,fg1,fg2,fg3,fg4,u)
```

The routines output 10 files by default (nOutputFiles=10), named u.001 through u.010. The output times for each mesh are printed on the screen as well as in accompanying files t.001 through t.010. The u files are in a three-column format easily plotted using gnuplot's splot.

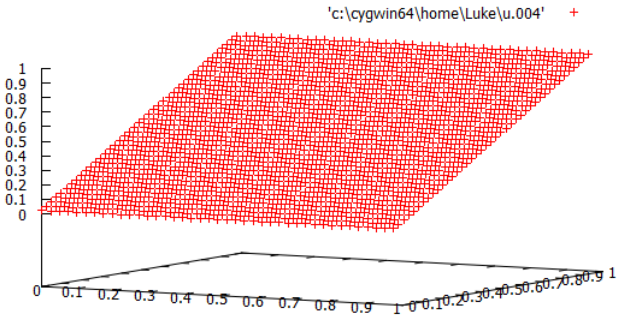
For the conditions specified in the project description and Question 2, the initial mesh is shown below.



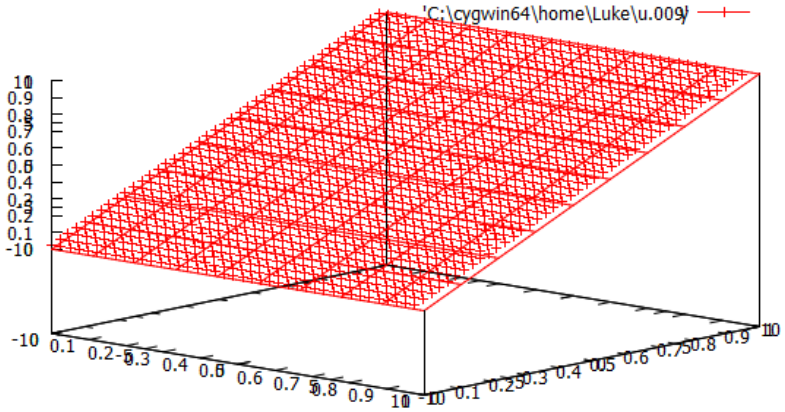
The diffusion equation smoothes out this mesh as throughout timesteps.



And eventually converges to the following mesh:



Below is the converged solution plotted with the analytical solution $u(x,y) = y$ using multiplot.



Question 3 : Full problem

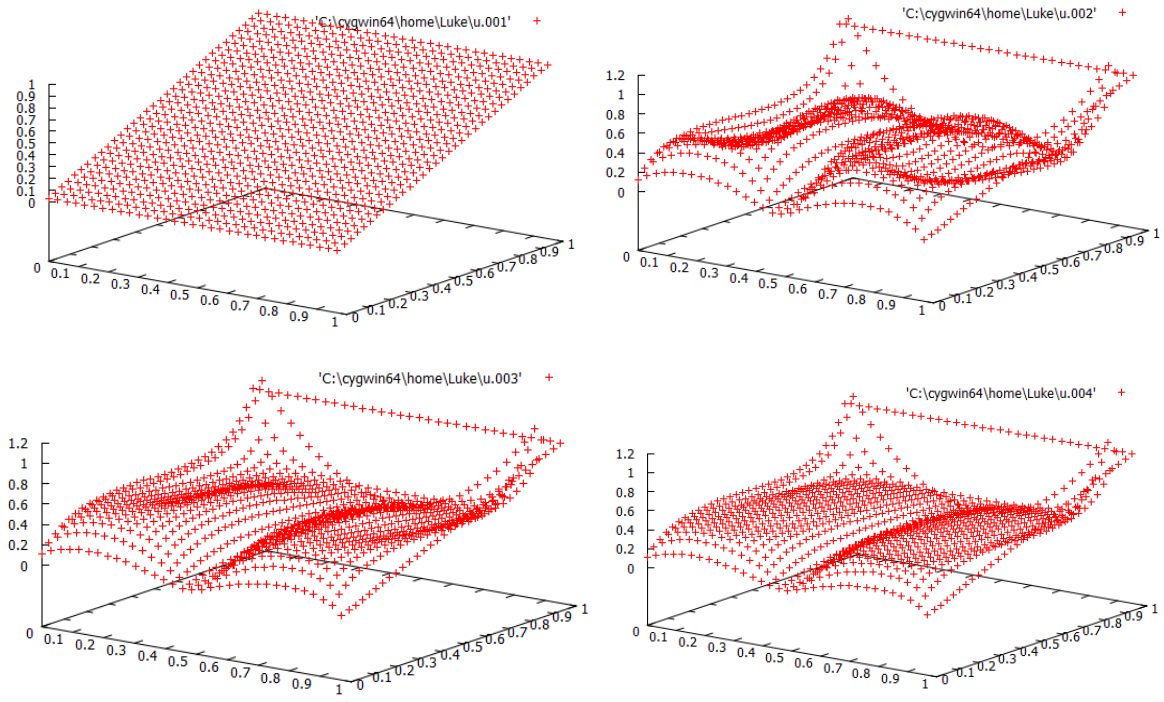
The same driving routine is used for questions 2 and 3. For question 3, an additional subroutine called combines the Crank-Nicholson routine for the diffusion equation with the FTCS scheme for the advection equation. The Crank-Nicholson initialization routine remains unchanged (creating an identical LHS A matrix). The CN and FTCS parts combine to create an updated RHS vector B at each timestep. The combined discretized function is shown below.

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = \frac{D}{2}\left(\frac{u_{i+1,j}^{n+1} - 2u_{ij}^{n+1} + u_{i-1,j}^{n+1} + u_{i+1,j}^{n} - 2u_{ij}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{ij+1}^{n+1} - 2u_{ij}^{n+1} + u_{ij-1}^{n+1} + u_{ij+1}^{n} - 2u_{ij}^{n} + u_{ij-1}^{n}}{\Delta y^2}\right) - \frac{a(x,y)}{2\Delta x^2}\left(u_{i+1,j}^{n} - u_{i-1,j}^{n}\right) - \frac{b(x,y)}{2\Delta y^2}\left(u_{i,j+1}^{n} - u_{i,j-1}^{n}\right)$$

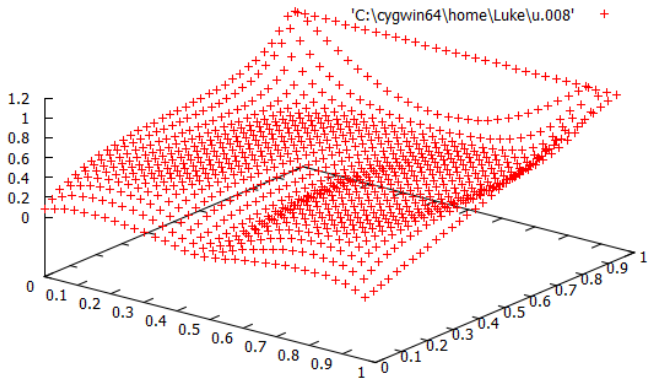
The subroutine is called at each timestep automatically when the user enters the option to select the routine upon running the driver routine. The routine parameters are shown below.

```
! Subroutine CN_FTCS_FULL_SOLVER is a Crank-Nicolson -
!        FTCS solver for the advection-diffusion equation
!
! ** Input parameters **
! x : input x vector (evenly spaced)
! y : input y vector (evenly spaced)
! bigI : input mesh I value
! bigJ : input mesh J value
! dt : timestep
! D : diffusion coefficient
! fg1,fg2,fg3,fg4 : dirichlet boundary conditions
! f3a,f3b : advection boundary functions
! A : 'U' cholesky decomposed matrix of LHS (n+1 timestep)
! u : output solution mesh matrix
SUBROUTINE CN_FTCS_FULL_SOLVER(x,y,bigI,bigJ,dt,D,fg1,fg2,fg3,fg4,f3a,f3b,A,u)
```

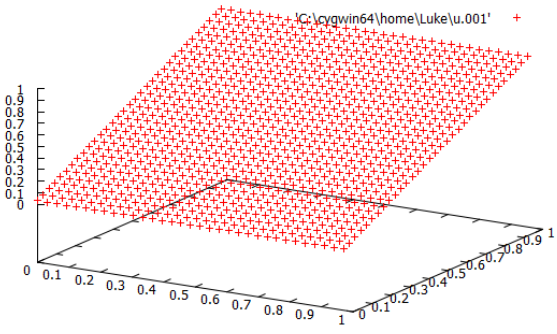
On the specified initial mesh u0, we see the system evolve through timesteps shown below. (D=1.0)

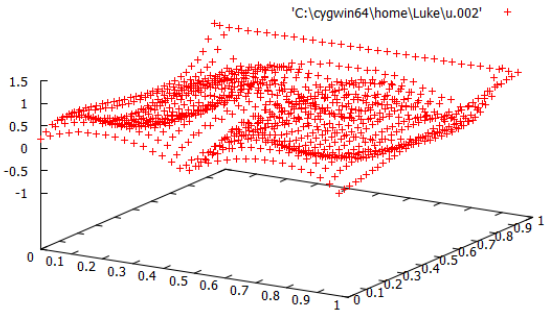


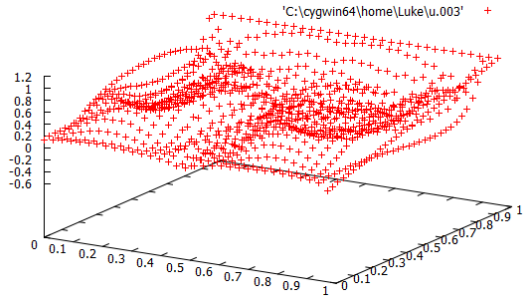
Converging to a final mesh:

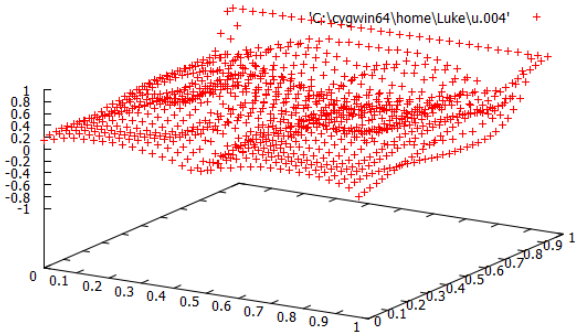


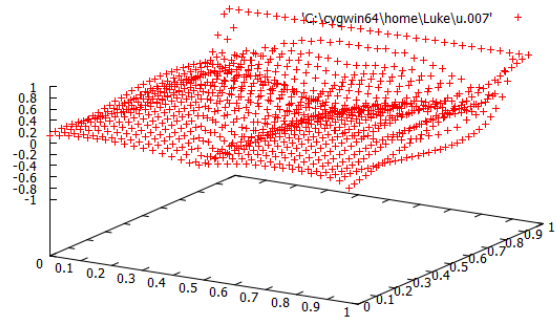
Lowering the value of diffusion coefficient D, we see a slightly different evolution of the mesh (D=0.1)

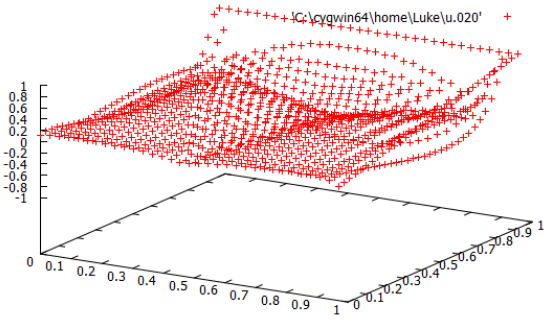


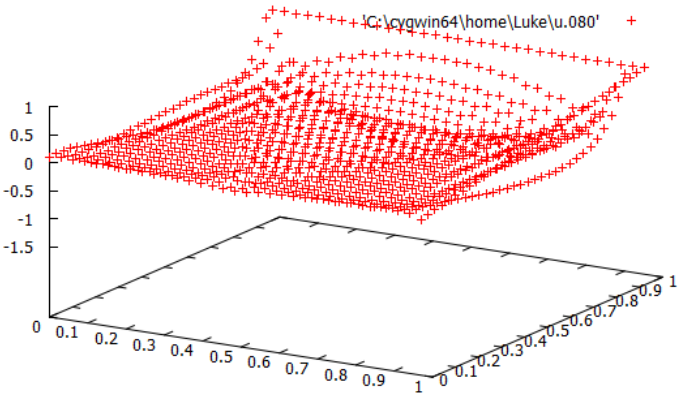












Along with the diffusion equation smoothing our function u, we have the advection equation transporting u via the specified velocity fields. The plot below shows the velocity fields in action.