Computer Problem #2

Paul Dorman

March 13, 2008

Introduction

This problem involves solving a Couette Flow, laminar viscous flow between two plates moving relative to each other. The plates are taken to extend infinitely in the plane of motion, limiting all variations to exist normal to the plates. Consequently, the flow needs only to be solved at one x location, but for several points in the y direction to fully characterize it. This flow is governed by the PDE below:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2}$$

with initial and boundary conditions given by:

$$u\left(0,t\right) = 0, \quad u\left(L,t\right) = 1, \quad y\left(y,0\right) = \frac{y}{L} + sin\left(\pi \frac{y}{L}\right)$$

1 Exact Solution

The exact solution to this equation consists of a steady-state solution plus (in the general case of the PDE) an infinite series of transients, as shown below:

$$u_{exact}\left(y,t\right) = \frac{y}{L} + \sum_{n=1}^{\infty} a_n sin\left(n\pi \frac{y}{L}\right) e^{-\nu\left(\frac{n\pi}{L}\right)^2 t}$$

This amounts to a linear increase in velocity from 0 at the y = 0 boundary to 1 at the y = L boundary as the steady state, with decaying transients from the initial velocity condition. Given the particular initial condition for this problem, only the n = 1 transient remains, giving the following as the exact time-dependent solution:

$$u_{exact}\left(y,t\right) = \frac{y}{L} + sin\left(\pi \frac{y}{L}\right) e^{-\nu\left(\frac{\pi}{L}\right)^2 t}$$

This equation can be non-dimensionalized by dividing y and t by some functions to eliminate the L and ν terms from the equation. This non-dimensionalization can be done as follows:

$$y' = \frac{y}{L}, \quad \tfrac{\partial}{\partial y} = \tfrac{\partial}{\partial y'}\tfrac{\partial y'}{\partial y} = \tfrac{1}{L}\tfrac{\partial}{\partial y'}, \quad \frac{\partial^2}{\partial y^2} = \frac{1}{L^2}\frac{\partial^2}{\partial y'^2}$$

$$t' = \frac{t}{\tau}, \quad \tfrac{\partial}{\partial t} = \tfrac{\partial}{\partial t'}\tfrac{\partial t'}{\partial t} = \tfrac{1}{\tau}\tfrac{\partial}{\partial t'}, \quad \tau = \frac{L^2}{\nu}$$

Here τ has been specifically chosen to eliminate L and ν from the equation, as demonstrated below:

$$\frac{\partial u}{\partial t} = \frac{1}{\tau}\frac{\partial u}{\partial t'} = \frac{\nu}{L^2}\frac{\partial u}{\partial t'}$$

$$\nu \frac{\partial^2 u}{\partial y^2} = \frac{\nu}{L^2} \frac{\partial^2 u}{\partial y'^2}$$

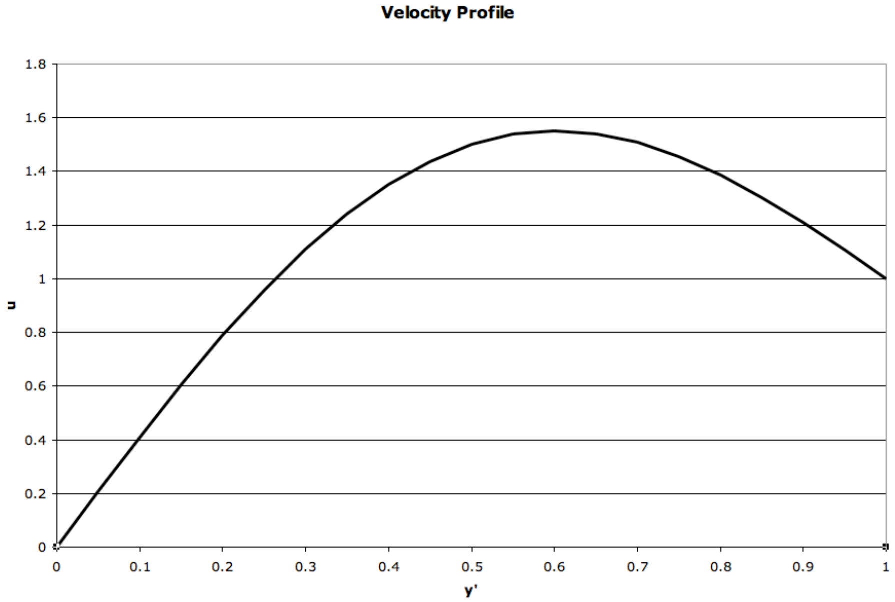
Therefore:

$$\frac{\partial u}{\partial t'} = \frac{\partial^2 u}{\partial y'^2}$$

The boundary and initial conditions for the non-dimensionalized equation are given by:

$$u\left(y' = 0, t\right) = 0, \quad u\left(y' = 1, t\right) = 1, \quad u\left(y', t' = 0\right) = y' + sin\left(\pi y'\right)$$

The initial condition is shown in the figure below:



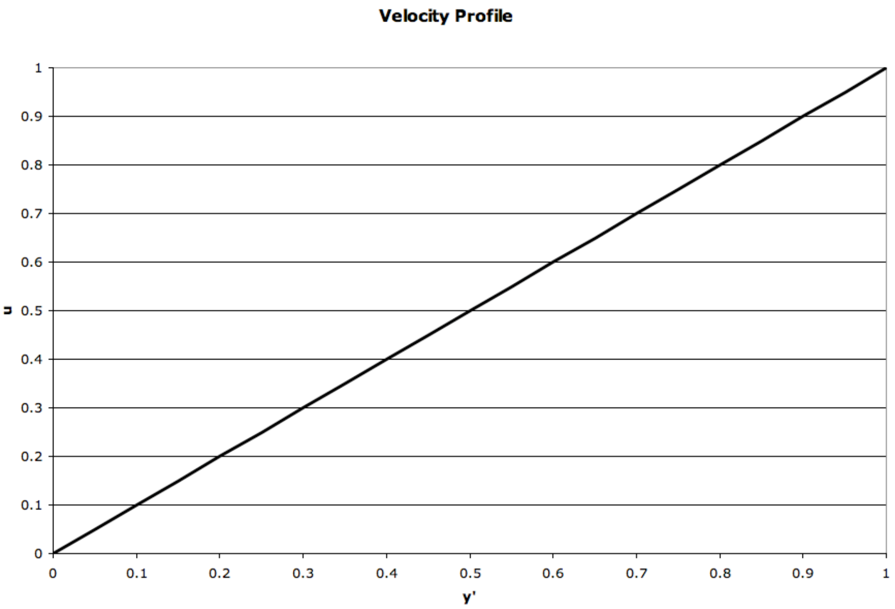
The exact solution to the non-dimensional formulation is given by:

$$u_{exact}\left(y', t'\right) = y' + sin\left(\pi y'\right) e^{-\pi^2 t}$$

The steady state solution to the non-dimensional formulation is given by:

$$u_{SS}\left(y'\right) = y'$$

The steady state solution is shown in the figure below:



2 Finite Difference Approximation

The finite difference scheme prescribed for this problem is an implicit-explicit scheme, given as *Combined Method A* in *Tannehill, Anderson, and Pletcher*. This scheme can vary from fully implicit to fully explicit, depending on a weighting factor θ. Although the general equation contains a ν term, that has been eliminated via non-dimensionalization, as shown in the previous section. The resultant finite difference approximation is given by the following expression:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t'} = \frac{1}{\Delta y'^2}\left[\theta\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + (1-\theta)\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)\right]$$

which can be rearranged into tri-diagonal form, as shown below:

$$(\Delta t'\theta)\,u_{j-1}^{n+1} - \left(\Delta y'^2 + 2\Delta t'\theta\right)u_j^{n+1} + (\Delta t'\theta)\,u_{j+1}^{n+1} = -\Delta t'\,(1-\theta)\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) - \Delta y'^2 u_j^n$$

where the coefficients of the u terms on the left and the entire RHS form the matrix entries for simultaneous solution to the following equation set:

$$b_j u_{j-1}^{n+1} + d_j u_j^{n+1} + a_j u_{j+1}^{n+1} = c_j$$

$$b_j = \Delta t'\theta$$

$$d_j = -\left(\Delta y'^2 + 2\Delta t'\theta\right)$$

$$a_j = \Delta t'\theta$$

$$c_j = -\Delta t'\,(1-\theta)\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) - \Delta y'^2 u_j^n$$

Putting these values into a tri-diagonal matrix allows for the simultaneous solution of all u_j at timestep *n+1* using the Thomas algorithm. Since u is fixed at locations $j = 1$ and $j = j_{max}$, the solution does not need to be computed there, and the matrix can be made smaller (and therefore faster to solve). However, at $j = 2$ and $j = j_{max} - 1$ special care must be taken as an entry is removed from the matrix diagonal band due to the matrix boundaries, and must be accounted for in the c value at that index, as shown below, where the values denoted with ' represent the new values:

$$c'_2 = c_2 - b_2 u_1 = c_2$$

$$c'_{j_{max}-1} = c_{j_{max}-1} - a_{j_{max}-1} u_{j_{max}} = c_{j_{max}-1} - a_{j_{max}-1}$$

This takes care of the boundary conditions from the PDE. The initial condition is taken care of by setting u for the first time step ($t' = 0$) at each j index to the discrete value of the initial condition (in its non-dimensional form) at that point:

$$u_j^1 = \frac{j-1}{j_{max}-1} + sin\left(\pi \frac{j-1}{j_{max}-1}\right)$$

3 Error Calculations

Three error calculations are performed for this problem: calculation of a root mean square (RMS) residual, RMS error relative to the exact time-dependent solution, and RMS error relative to the exact steady-state solution. The RMS residual gives an indication of how much the finite difference solution is changing from one iteration to the next. The loop termination condition is that this residual is less than 10^{-6}; i.e., the solution is changing by less than one part in a million at each iteration. Although this does not guarantee that the finite difference solution has matched the exact solution, it is changing so slowly at that point that it is impractical (and may be impossible) to run it until it converges on the exact solution. This residual is calculated as:

$$R = \sum_{j=2}^{j_{max}-1} \sqrt{\frac{\left(u_j^{n+1} - u_j^n\right)^2}{j_{max}-2}}$$

where the summation takes place between $j = 2$ and $j = j_{max} - 1$ because the value of u at *1* and j_{max} is held fixed.

The two RMS error terms give the root mean square error between the finite difference solution and the time-dependent and steady-state exact solutions. These expression are given below:

$$Error_{time-dependent} = \sum_{j=2}^{j_{max}-1} \sqrt{\frac{\left(u_j^{n+1} - u_{exact}\left(y', t'\right)\right)^2}{j_{max}-2}}$$

$$Error_{steady-state} = \sum_{j=2}^{j_{max}-1} \sqrt{\frac{\left(u_j^{n+1} - u_{SS}\left(y'\right)\right)^2}{j_{max}-2}}$$

where y' is given by $\frac{j-1}{j_{max}-1}$, and t' is calculated by starting from 0 and adding $\Delta t'$ at each iteration.

4 Stability Analysis

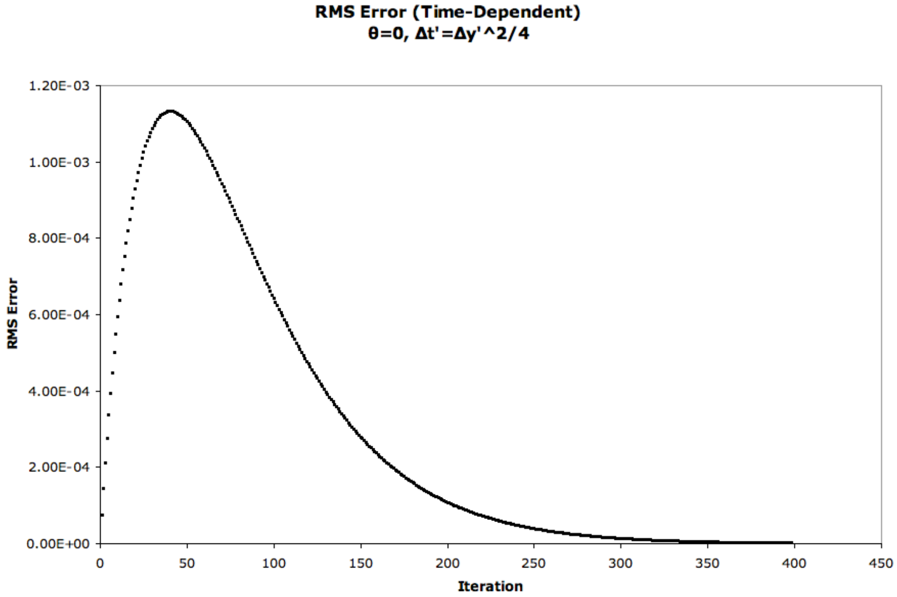
The finite difference approximation used here has varying regimes of stability. By von Neumann analysis, it can be found that the scheme is unconditionally stable when $\frac{1}{2} \leq \theta \leq 1$, and conditionally stable when $0 \leq \theta < \frac{1}{2}$. The condition for stability is that $0 \leq \frac{\Delta t'}{\Delta y'^2} \leq \frac{1}{2-4\theta}$; or, since $\Delta t'$ is always positive: $\Delta t' \leq \frac{\Delta y'^2}{2-4\theta}$. In all cases, $j_{max} = 11$ was used, giving $\Delta y' = 0.1$.

4.1 $\theta = 0$

For the $\theta = 0$ case, which is fully explicit, the scheme should be stable when $\Delta t' \leq \frac{\Delta y'^2}{2}$. To verify this, tests were performed with $\Delta t'$ equal to $\frac{\Delta y'^2}{4} = 0.0025$, $\frac{\Delta y'^2}{2} = 0.005$, and $\Delta y'^2 = 0.01$. These analytic limits are borne out in practice, as is demonstrated below.

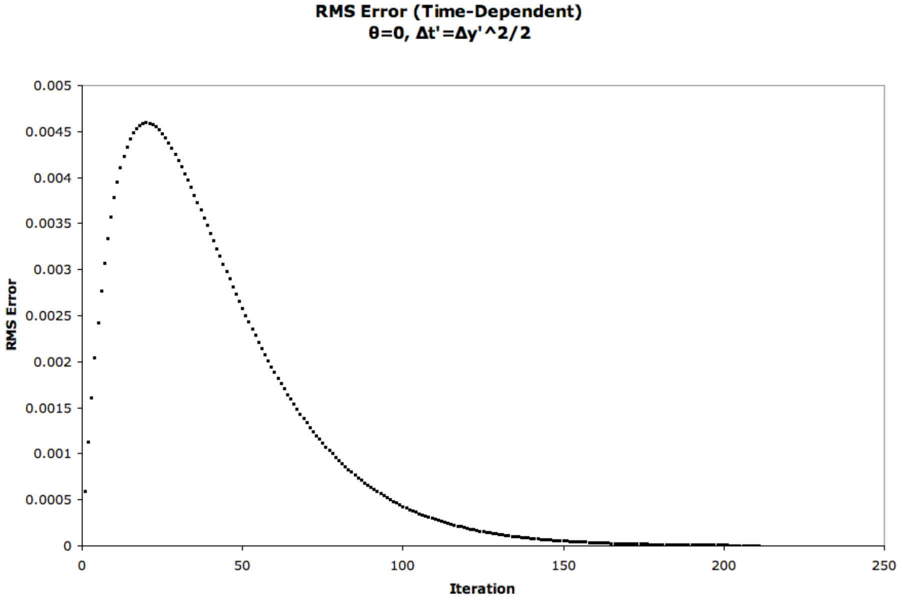
4.1.1 $\Delta t' = \frac{\Delta y'^2}{4}$

For this value of $\Delta t'$, the numerical solution was stable and convered to an RMS residual of less than 10^{-6} in 398 iterations, with a steady-state RMS error of 3.89×10^{-5}. As can be seen in the figure below, the time-dependent RMS error initially rose, but then began to converge quickly.



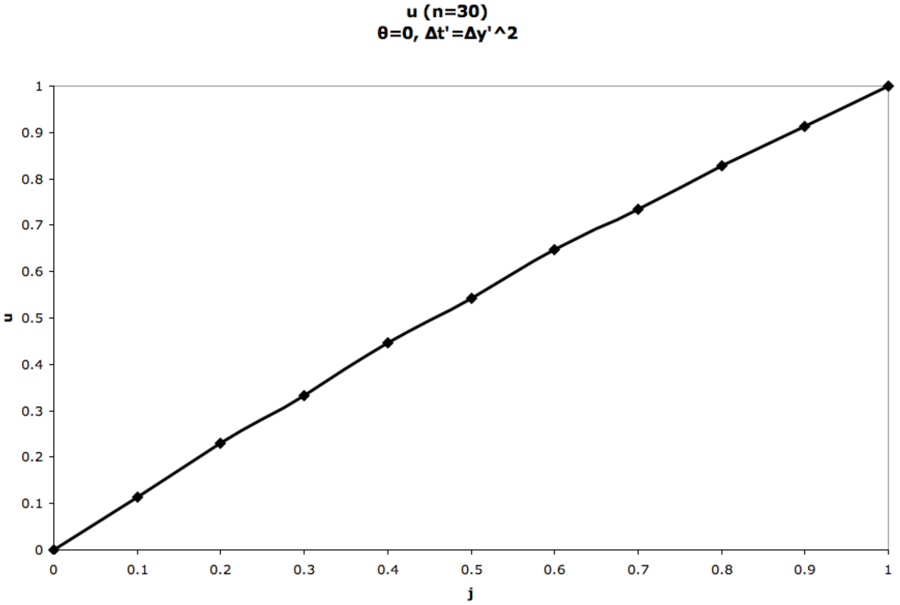
4.1.2 $\Delta t' = \frac{\Delta y'^2}{2}$

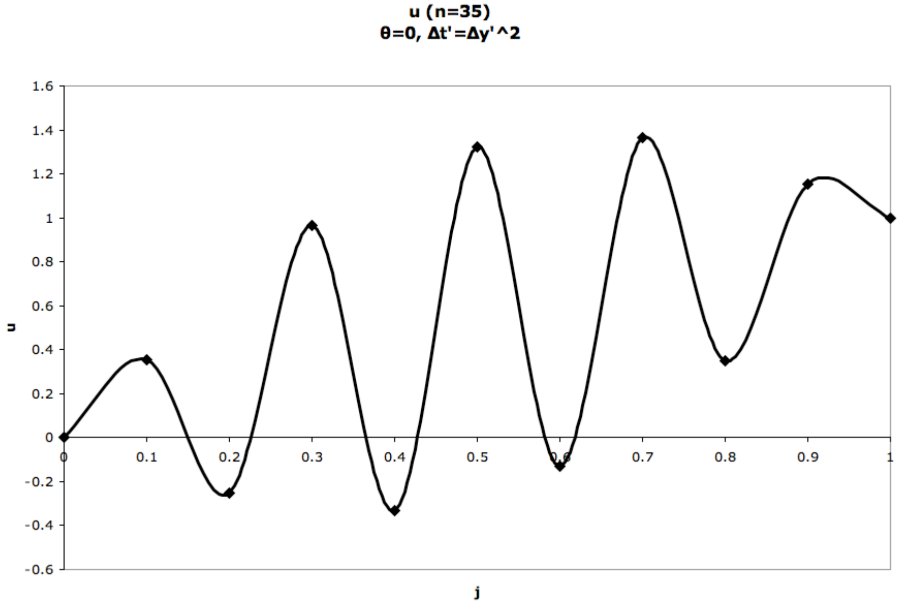
In this case, the solution was also stable and actually converged more quickly than with the smaller value of $\Delta t'$, in terms of numbers of iterations. This should be expected, as the step size was twice as big. However, it took 211 iterations to reach the RMS residual tolerance, which is slightly more than half the number if iterations it took with half the step size, most likely because the scheme was only marginally stable. The RMS error followed the same trend as before, although the error climbed higher before returning to 0, as seen in the figure below.



4.1.3 $\Delta t' = \Delta y'^2$

With $\Delta t'$ above the conditional stability limit, the scheme displays the expected unstable behavior. Interestingly, it starts out well, approaching the solution, as seen in the figure below from iteration number 30, but then it develops oscillations, seen in the figure from iteration number 35, which due to the instability increase until the numerical limits of the machine are reached.



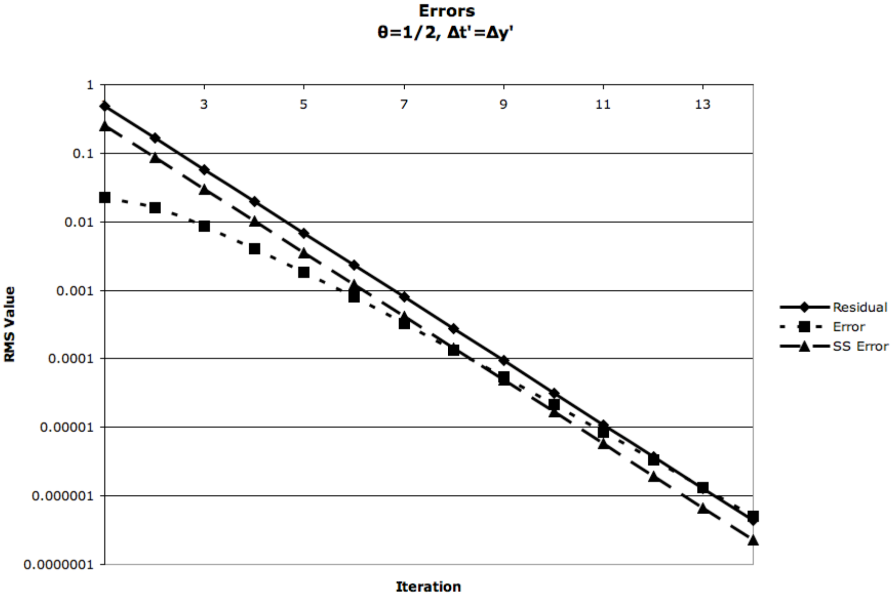


4.2 $\theta = \frac{1}{2}$

Because this combined implicit-explicit scheme is nominally unconditionally stable, two relatively large values of $\Delta t'$ were chosen to test: $\Delta t' = \Delta y' = 0.1$ and $\Delta t' = 10\Delta y' = 1$. As is demonstrated in the following sections, this scheme was indeed found to be uncondtionally stable.

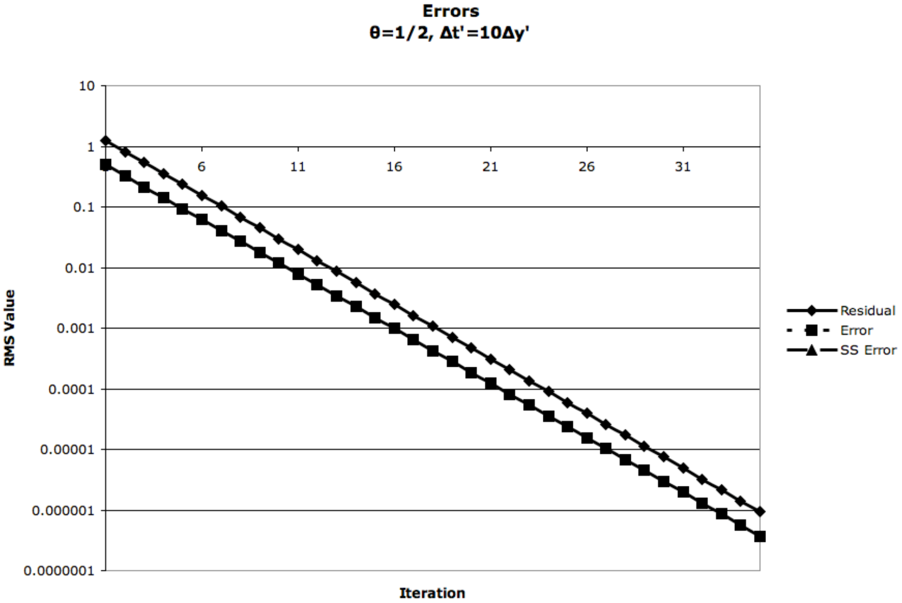
4.2.1 $\Delta t' = \Delta y'$

This value of $\Delta t'$ allowed the scheme to converge in 14 iterations, with a final steady-state RMS error of 2.31×10^{-7}. Unlike with the fully explicit scheme, the RMS residual and errors decreased monotonically toward 0, as seen in the figure below.



4.2.2 $\Delta t' = 10\Delta y'$

For this case with the larger value of $\Delta t'$, the solution took more iterations to converge, but had done so by the 35th iteration with a steady-state RMS error of 3.74×10^{-7}. Additionally, the steady-state and time-dependent errors lay right on top of each other for this case because the time step was larger so the time-dependent solution had essentially become the steady-state solution.

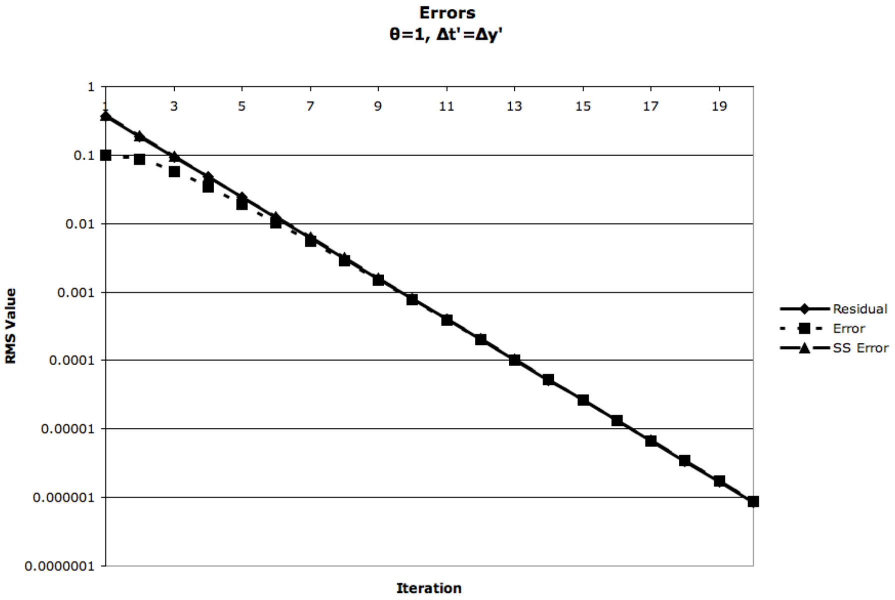


4.3 $\theta = 1$

The last scheme to be tested for stability is the fully implicit scheme. Again, this scheme is nominally unconditionally stable, so the same values of $\Delta t'$ of $\Delta y' = 0.1$ and $\Delta y' = 1$ were chosen. And again, the stability was in practice as it was determined it should be in theory.

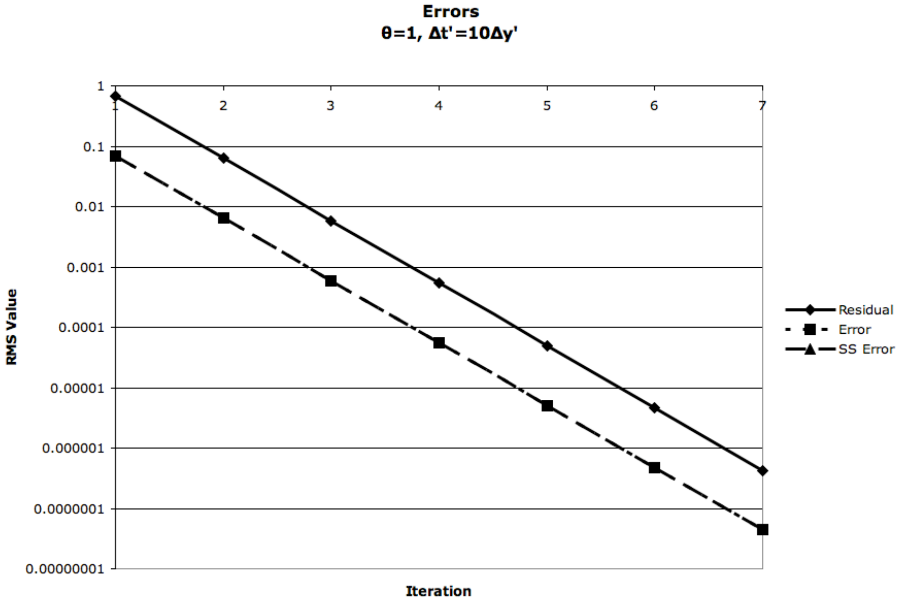
4.3.1 $\Delta t' = \Delta y'$

This case converged in 20 iterations with a final steady-state RMS error of 8.79×10^{-7}. This is a few more iterations than for the same $\Delta t'$ with $\theta = 0.5$, suggesting that the fully implicit case is not as computationally efficient. The errors are shown in the figure below. In this case, the residual and the steady-state error lay atop each other.



4.3.2 $\Delta t' = 10\Delta y'$

For this case, the solution converged in only seven iterations to a steady-state RMS error of 4.38×10^{-8}. This was unexpected as previously a larger $\Delta t'$ led to a greater number of iterations required for convergence. Additionally, the final steady state error was an order of magnitude less than at convergence for the other setups. Once again, the time-dependent and steady-state errors lie atop each other, as at these time scales the two solutions are essentially identical.



5 Accuracy

For the cases $\theta = 0$ and $\theta = 1$, the accuracy of the finite difference expression should be $O\left(\Delta t', \Delta y'^2\right)$. For the $\theta = \frac{1}{2}$ case, the accuracy of the finite difference expression should be $O\left(\Delta t'^2, \Delta y'^2\right)$. These can be determined by finding the truncation error of the finite difference expression and locating the lowest order terms. To find the actual accuracy of the code, the RMS time-dependent error from the first iteration was looked at, since that represented a single timestep from the initial condition at which it was numerically identical to the exact solution. Grids of varying fineness were used to examine the $\Delta y'$ accuracy, and various timesteps were used to examine the $\Delta t'$ accuracy. To determine if an accuracy was linear, it was plotted with Excel and the R^2 value for a linear fit taken, with the requirement that it be near 1. To determine if it was quadratic, it was again plotted with Excel, this time with a 2nd order polynomial, and the magnitude of the quadratic term was compared with the magnitude of the linear term, with the requirement that it be significantly larger. In general, the order of accuracy of the code was found to correspond very well with the predictions, although the $\Delta y'$ accuracy for the $\theta = 1$ case fit less well.

5.1 $\theta = 0$

For the explicit grid, values for $\Delta t'$ of $\frac{\Delta y'^2}{2}$, $\frac{\Delta y'^2}{3}$, and $\frac{\Delta y'^2}{4}$ were used, for both the fine ($\Delta y' = 0.02$) grid. In this case, the RMS errors were as shown in the table below, which correspond to a linear dependence with an R^2 value of 0.9893.

$\Delta t'$	RMS Error
0.0002000	9.27×10^{-7}
0.0001333	3.09×10^{-7}
0.0001000	1.16×10^{-7}

To find the $\Delta y'$ accuracy, a constant value of $\Delta t' = 0.0002$ was used, with $\Delta y'$ as 0.02, , 0.0333, 0.05, 0.0667, and 0.10. This resulted in the expected quadratic error fit (quadratic term 19 times the linear dependence), with an R^2 value of 0.9938, as shown in the table below. There is an outlier at $\Delta y' = 0.0333$, which may be the result of cancellations in the truncation error resulting in a better overall accuracy.

$\Delta y'$	RMS Error
0.0200	9.27×10^{-7}
0.0333	1.03×10^{-7}
0.0500	1.53×10^{-6}
0.0667	3.85×10^{-6}
0.1000	1.06×10^{-5}

5.2 $\theta = \frac{1}{2}$

For the implicit-exlicit scheme, values for $\Delta t'$ of $\frac{\Delta y'}{8}$, $\frac{\Delta y'}{4}$, $\frac{\Delta y'}{2}$, $\Delta y'$, and $2\Delta y'$ were used for a $\Delta y'$ of 0.02 to determine the $\Delta t'$ accuracy. The results are that the RMS error term has a quadratic dependence 71 times the linear dependence on $\Delta t'$, with an R^2 value of 0.9995, with the data given in the table below.

$\Delta t'$	RMS Error
0.0025	4.78×10^{-6}
0.0050	4.23×10^{-6}
0.0100	3.09×10^{-5}
0.0200	3.39×10^{-4}
0.0400	2.46×10^{-3}

For the determination of the $\Delta y'$ accuracy, $\Delta t'$ was held fixed at 0.0025, and the finite difference expression was calculated for $\Delta y'$ values of 0.01, 0.02, 0.0333, 0.04, and 0.05. The accuracy was found to be of order $\Delta y'^2$, with the error term in the quadratic being 730 times the linear term and an R^2 value of 1, from the data shown in the table below.

$\Delta y'$	RMS Error
0.0100	5.39×10^{-7}
0.0200	4.78×10^{-6}
0.0333	1.49×10^{-5}
0.0400	2.20×10^{-5}
0.0500	3.50×10^{-5}

5.3 $\theta = 1$

For the fully implicit scheme, to determine the $\Delta t'$ order of accuracy, $\Delta y'$ was held fixed at 0.02, while $\Delta t'$ was set to 0.01, 0.02, 0.03, 0.04, and 0.05. The result was a linear order accuracy, with an R^2 value of 0.993, for the data given in the table below.

$\Delta t'$	RMS Error
0.01	0.00298
0.02	0.01023
0.03	0.01992
0.04	0.03086
0.05	0.04225

For the determination of the order of $\Delta y'$ accuracy, $\Delta t'$ was held fixed at 0.05, while $\Delta y'$ was set to 0.01, 0.02, 0.0333, 0.04, and 0.05. For this test, the quadratic term came out to be only 7.3 times the linear term, with an R^2 value of 1, which does not match as well with the theory as for the other cases. In fact, using a linear fit gave an R^2 value of 0.9965, which indicates it was nearly linear. The data are given in the table below.

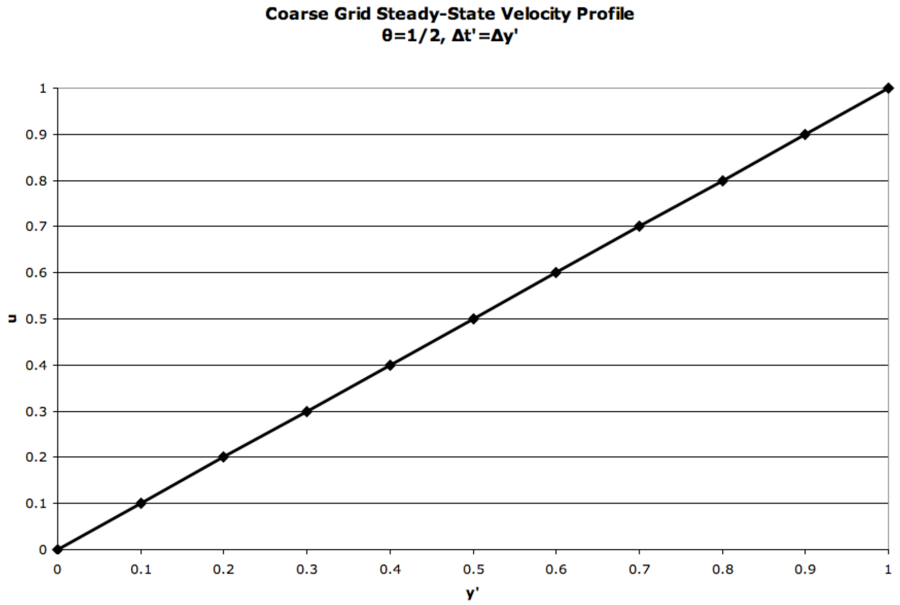
$\Delta y'$	RMS Error
0.0100	0.0420
0.0200	0.0423
0.0333	0.0426
0.0400	0.0428
0.0500	0.0432

6 Steady-State Solutions

For the steady-state solutions using a coarse grid of $j_{max} = 11$ grid points and a fine grid of $j_{max} = 51$ grid points, $\theta = \frac{1}{2}$ and $\Delta t' = \Delta y'$ were chosen as the input parameters.

6.1 Coarse Grid

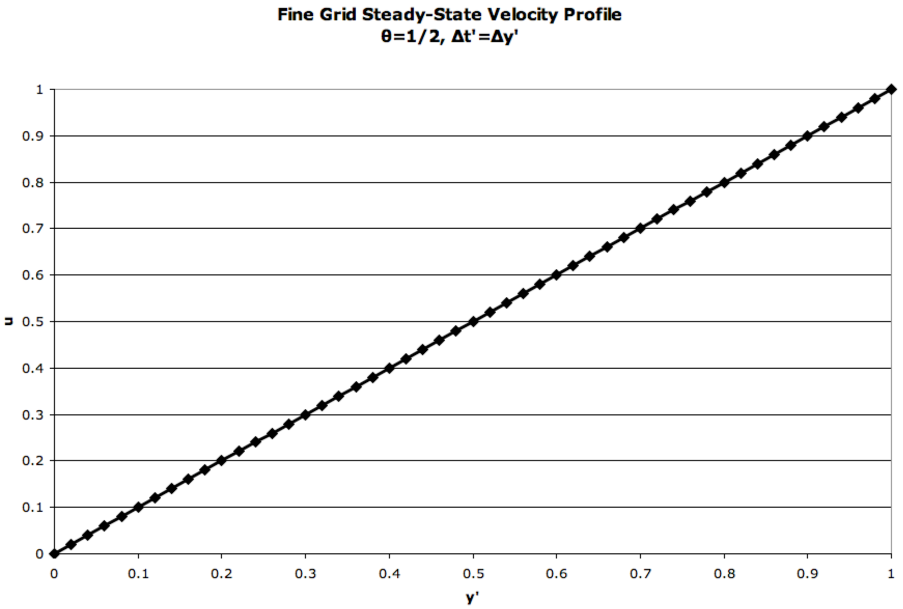
Using the input parameters described above with $\Delta t' = \Delta y' = 0.1$, the coarse grid converged in 14 iterations with a final steady-state RMS error of 2.31×10^{-7}.



y’	u
0.0	0.00000000
0.1	0.10000010
0.2	0.20000018
0.3	0.30000025
0.4	0.40000029
0.5	0.50000031
0.6	0.60000029
0.7	0.70000025
0.8	0.80000018
0.9	0.90000010
1.0	1.00000000

6.2 Fine Grid

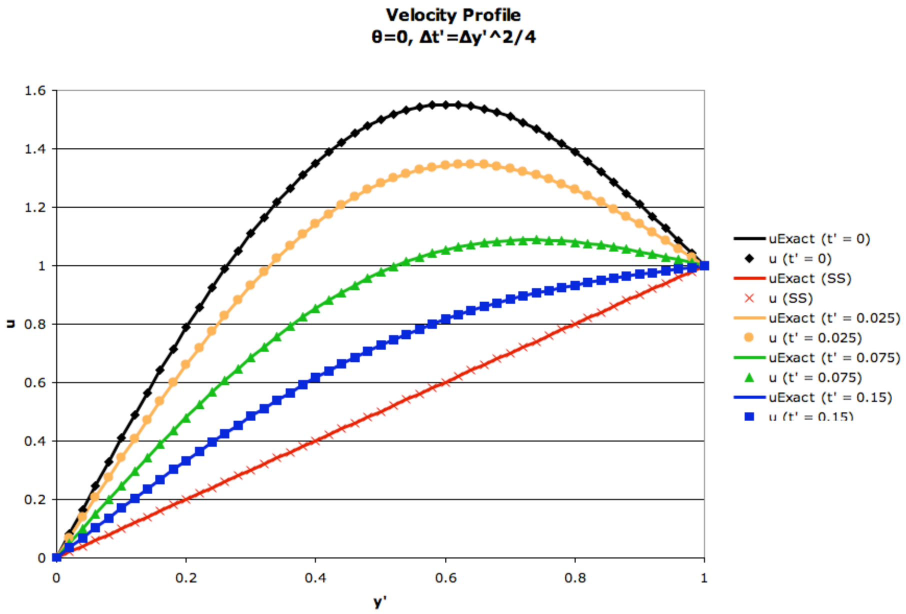
The fine grid converged in 61 iterations to a steady-state RMS error of 4.07×10^{-6}, with $\Delta t' = \Delta y' = 0.02$.

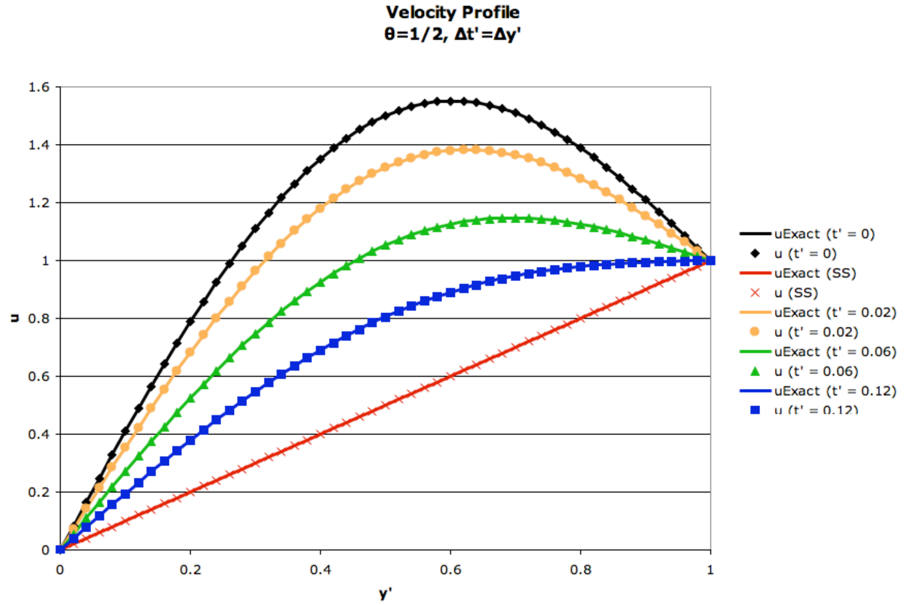


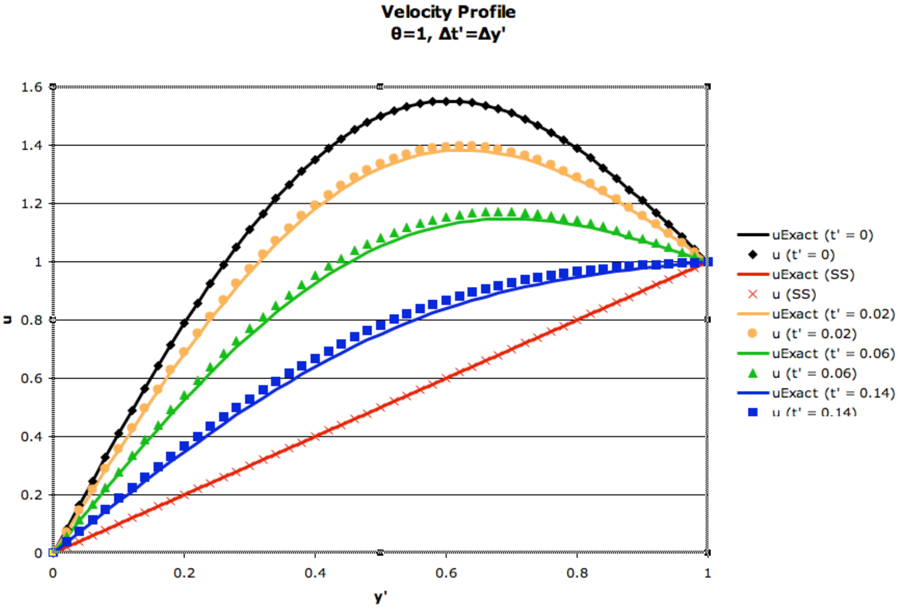
y’	u	y’	u	y’	u	y’	u	y’	u
0.00	0.00000000	0.20	0.20000335	0.40	0.40000541	0.60	0.60000541	0.80	0.80000335
0.02	0.02000036	0.22	0.22000363	0.42	0.42000551	0.62	0.62000529	0.82	0.82000305
0.04	0.04000071	0.24	0.24000390	0.44	0.44000559	0.64	0.64000515	0.84	0.84000274
0.06	0.06000107	0.26	0.26000415	0.46	0.46000565	0.66	0.66000499	0.86	0.86000242
0.08	0.08000142	0.28	0.28000439	0.48	0.48000568	0.68	0.68000481	0.88	0.88000210
0.10	0.10000176	0.30	0.30000461	0.50	0.50000569	0.70	0.70000461	0.90	0.90000176
0.12	0.12000210	0.32	0.32000481	0.52	0.52000568	0.72	0.72000439	0.92	0.92000142
0.14	0.14000242	0.34	0.34000499	0.54	0.54000565	0.74	0.74000415	0.94	0.94000107
0.16	0.16000274	0.36	0.36000515	0.56	0.56000559	0.76	0.76000390	0.96	0.96000071
0.18	0.18000305	0.38	0.38000529	0.58	0.59000551	0.78	0.78000363	0.98	0.98000036
								1.00	1.00000000

7 Time Velocity Profiles

To compare the time velocity profiles, three values of the control parameter θ were used: 0, $\frac{1}{2}$, and 1. To ensure the stability of the solution, $\Delta t' = \frac{\Delta y'^2}{4}$ was used for the first case, and $\Delta t' = \Delta y'$ was used for the latter two cases. For all cases, $j_{max} = 51$ was used, giving a corresponding value of $\Delta y' = 0.02$. As can be seen in the figures below, the finite difference approximation was very close to the exact solution for the $\theta = 0$ and $\theta = \frac{1}{2}$ cases, while it was slightly off between the initial condition and steady state for the $\theta = 1$ case.


Velocity Profile
θ=0, Δt'=Δy'^2/4
u
y'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
0.2
0.4
0.6
0.8
1
uExact (t' = 0)
u (t' = 0)
uExact (SS)
u (SS)
uExact (t' = 0.025)
u (t' = 0.025)
uExact (t' = 0.075)
u (t' = 0.075)
uExact (t' = 0.15)
u (t' = 0.15)


Velocity Profile
θ=1/2, Δt'=Δy'
u
y'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
0.2
0.4
0.6
0.8
1
uExact (t' = 0)
u (t' = 0)
uExact (SS)
u (SS)
uExact (t' = 0.02)
u (t' = 0.02)
uExact (t' = 0.06)
u (t' = 0.06)
uExact (t' = 0.12)
u (t' = 0.12)



8 Code

The code is written in ANSI C, compiled with GCC 4.0.1 on Mac OS X 10.5. The grid spacing parameter j_{max} is given as *JMAX* on line 11. The variables *theta, maxItr,* and *tol* give the finite difference expression control parameter, maximum number of iterations, and RMS residual tolerance for convergence, respectively. The $\Delta t'$ parameter is given as *dt* on line 40. Upon running, the code will generate the file *Residual.csv*, which contains the RMS residual, RMS time-dependent error, and RMS steady-state error; and several *grid-N.csv*, which contain the solution at iteration number *N*.

```
#include <stdio.h>
#include <math.h>

void initialConditions();            // sets up the initial state of u
void iteration();                    // performs a single time-step iteration
void dumpGrid( int i );              // prints the current state of u to a CSV file
double uExact( int j, double t );
    // returns the exact solution at grid-point j and non-d time t
double uSS( int j );                 // returns the steady state solution at grid-point j
void clean();                        // empties the directory of generated *.csv files

#define JMAX 51

double u[JMAX];

double residual;
double error;
double errorSS;

double theta = 1;
int maxItr = 10000;
double tol = 1E-6;

double dy = 1.0/(JMAX-1);
double t = 0.0;
double dt;
```

```
int main (int argc, const char * argv[]) {
        int i;
        FILE *rPlot;

        clean();

        rPlot = fopen( "Residual.csv", "w" );
        fprintf( rPlot, "Iteration,Residual,Error,SS Error\n" );
    initialConditions();

        dumpGrid(0);

        dt = dy;

        for( i = 1; i <= maxItr; i++ ){
                t += dt;
                iteration();
                dumpGrid(i);
                fprintf( rPlot, "%3d,%g,%g,%g\n", i, residual, error, errorSS );
                if( residual <= tol )
                        break;
        }

        fclose( rPlot );

        return 0;
}

// sets up the grid with non-dimensional initial condition: u = y' + sin(pi*y')
void initialConditions(){
        int j;

        for( j = 0; j < JMAX; j++ ){
                double yPrime = j/(JMAX-1.0);
                u[j] = yPrime + sin(M_PI*yPrime);
        }
}

// performs the Thomas algorithm over the entire grid
void iteration(){
        int j;
        double b[JMAX-2], d[JMAX-2], a[JMAX-2], c[JMAX-2];

        residual = 0.0;
        error = 0.0;
        errorSS = 0.0;

        // set up coefficients
        // j indices offset from u array by 1
        for( j = 0; j < JMAX-2; j++ ){
                b[j] = dt*theta;
                d[j] = -(2.0*dt*theta + dy*dy);
                a[j] = dt*theta;
                c[j] = -dt*(1.0-theta)*(u[j+2]-2.0*u[j+1]+u[j]) - dy*dy*u[j+1];
        }

        // correct for diagonal band truncation
        c[JMAX-3] = c[JMAX-3] - a[JMAX-3];

        // Thomas algorithm
        for( j = 1; j < JMAX-2; j++ ){
                d[j] = d[j] - b[j]/d[j-1]*a[j-1];
                c[j] = c[j] - b[j]/d[j-1]*c[j-1];
        }

        // j indices offset from u array by 1
        c[JMAX-3] = c[JMAX-3]/d[JMAX-3];
        residual += pow( c[JMAX-3] - u[JMAX-2], 2 );
        error += pow( c[JMAX-3] - uExact( JMAX-2, t ), 2 );
        errorSS += pow( c[JMAX-3] - uSS( JMAX-2 ), 2 );
        u[JMAX-2] = c[JMAX-3];
        for( j = JMAX-4; j >= 0; j-- ){
                double newU = (c[j] - a[j]*u[j+2])/d[j];

                residual += pow( newU - u[j+1], 2 );
                error += pow( newU - uExact( j+1, t ), 2 );
                errorSS += pow( newU - uSS( j+1 ), 2 );

                u[j+1] = newU;
        }
        residual = sqrt( residual/(JMAX-2) );
        error = sqrt( error/(JMAX-2) );
        errorSS = sqrt( errorSS/(JMAX-2) );
}

// print the grid out to grid-n.csv
void dumpGrid( int n ){
        FILE *file;
        int j;
        char fileName[32];

        sprintf( fileName, "grid-%d.csv", n );

        file = fopen( fileName, "w" );

        fprintf( file, "y',u,uExact\n" );

        for( j = 0; j < JMAX; j++ ){
                fprintf( file, "%f,%16.8f,%16.8f\n", j/(JMAX-1.0), u[j], uExact( j, t ) );
        }

        fclose( file );
}

// exact solution at j, t'
double uExact( int j, double t ){
        double y = j/(JMAX-1.0);
        return y + sin(M_PI*y)*exp(-M_PI*M_PI*t);
}

// steady state solution at j
double uSS( int j ){
        return j/(JMAX-1.0);
}

// remove all generated *.csv files
void clean(){
        char fileName[32];
        int n;

        for( n = 0; n < maxItr; n++ ){
                sprintf( fileName, "grid-%d.csv", n );
                remove( fileName );
        }

        remove( "Residual.csv" );
}
```