Homework Assignment
Numerical solution of partial differential equations, I (Course 221)
Handed out: Tuesday, September 30, 2003, Due: Tuesday, October 14, 2003

3 Diffusion Equation Numerical Experiments

3.1 Analytical solution on an infinite domain

Consider the IVP

$$\begin{aligned} q_t &= q_{xx} \\ q(x,0) &= \eta(x) = \cos(\pi x) + \sin(\pi x) \end{aligned} \tag{1}$$

for x real. Solve this problem using Fourier transform techniques.

3.1.1 Solution

The solution will maintain the initial condition's periodicity on $[-1,1]$. We can use a Fourier series solution

$$q(x,t) = \frac{a_0(t)}{2} + \sum_{k=1}^{\infty} \left[a_k(t)\cos(k\pi x) + b_k(t)\sin(k\pi x)\right] \tag{2}$$

The initial condition immediately leads to

$$q(x,t) = a_1(t)\cos(\pi x) + b_1(t)\sin(\pi x) \tag{3}$$

$$a_1(0) = b_1(0) = 1 \tag{4}$$

Substitution of the above in the PDE gives

$$a_1'(t)\cos(\pi x) + b_1'(t)\sin(\pi x) = -\pi^2\left[a_1(t)\cos(\pi x) + b_1(t)\sin(\pi x)\right] \tag{5}$$

which leads to

$$a_1' + \pi^2 a_1 = 0,\ b_1' + \pi^2 b_1 = 0 \tag{6}$$

with solutions

$$a_1(t) = b_1(t) = e^{-\pi^2 t} \tag{7}$$

so the solution to the problem is

$$q(x,t) = \left[\cos(\pi x) + \sin(\pi x)\right] e^{-\pi^2 t}\ . \tag{8}$$

The above procedure is most direct and involves the Fourier transformation from a periodic function to a Fourier series (countable number of terms), i.e.

$$a_k(t) = \int_{-1}^{1} q(y,t)\cos(k\pi y)\ dy,\ b_k(t) = \int_{-1}^{1} q(y,t)\sin(k\pi y)\ dy. \tag{9}$$

Remember the discussion from class that the infinite domain Fourier transform is obtained from the above by taking the interval of periodicity to infinity. We could also apply the infinite domain Fourier transform technique (though that would be a sign of not grasping the concept of different Fourier transforms arising for different intervals of periodicity). In this case we would use

$$\hat{q}(\xi,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} q(x,t)e^{-i\xi x}dx, \; q(x,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{q}(\xi,t)e^{i\xi x}d\xi \tag{10}$$

and obtained the PDE in Fourier space

$$\hat{q}_t = -\xi^2 \hat{q} \tag{11}$$

with the solution

$$\hat{q}(\xi,t) = \hat{\eta}(\xi)e^{-\xi^2 t}. \tag{12}$$

Here $\hat{\eta}(\xi)$ is the Fourier transform of the initial condition $\eta(x)$

$$\hat{\eta}(\xi) = \sqrt{\frac{\pi}{2}}\,[(1+i)\delta(\xi-\pi) + (1-i)\delta(\xi+\pi)] \tag{13}$$

The solution is

$$q(x,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{\eta}(\xi)e^{-\xi^2 t}e^{i\xi x}d\xi \tag{14}$$

$$= [\cos(\pi x) + \sin(\pi x)]\, e^{-\pi^2 t}. \tag{15}$$

3.2 Analytical solution on a finite domain

Consider the IVBP

$$\begin{gathered} q_t = q_{xx} \\ q(x,0) = \eta(x) = \cos(\pi x) + \sin(\pi x) \\ q(x=0,t) = g_0(t) = 1 \\ q(x=1,t) = g_1(t) = -1 \end{gathered} \tag{16}$$

This problem models the heating in time of a bar initially at temperature $\eta(x)$ which has its ends at constant fluxes $g_0(t) = g_1(t) = \pi$. Compute the analytical solution to this problem using separation of variables. Compare with the solution from the previous problem.

3.2.1 Solution

From our qualitative knowledge of the behavior of solutions of the heat equation we suspect that the influence of the initial condition will not be felt at long times. At long times the dominant influence will come from the boundary conditions. The boundary conditions maintain a constant temperature difference between the ends of the rod. Thus the long time solution is expected to be a linear drop in temperature

$$\sigma(x) = 1 - 2x. \tag{17}$$

Note that $\sigma(x)$ satisfies the heat equation, i.e. $\sigma_t = \sigma_{xx} = 0$. We now seek the transient solution

$$\tau(x,t) = q(x,t) - \sigma(x) \ . \tag{18}$$

It satisfies the IBVP

$$\begin{aligned} \tau_t &= \tau_{xx} \\ \tau(x,0) &= \eta(x) - \sigma(x) = \cos(\pi x) + \sin(\pi x) - 1 + 2x \\ q(x=0,t) &= g_0(t) = 0 \\ q(x=1,t) &= g_1(t) = 0 \end{aligned} \tag{19}$$

Separation of variables $\tau(x,t) = X(x)T(t)$ immediately leads to the possible solutions

$$X(x) = A\cos(\lambda x) + B\sin(\lambda x) \tag{20}$$

$$T(t) = Ce^{-\lambda^2 t} \tag{21}$$

We now seek to satisfy boundary conditions.

$$X(x=0) = A = 0 \tag{22}$$

$$X(x=1) = B\sin(\lambda) = 0 \tag{23}$$

thus leading to

$$\lambda = k\pi \ . \tag{24}$$

The general solution to the problem is therefore of the form (superposition of all admitted eigenmodes)

$$\tau(x,t) = \sum_{k=1}^{\infty} a_k \sin(k\pi x) e^{-(k\pi)^2 t} \tag{25}$$

The coefficients a_k are determined from the initial condition

$$\tau(x,t=0) = \sum_{k=1}^{\infty} a_k \sin(k\pi x) = \cos(\pi x) + \sin(\pi x) - 1 + 2x \tag{26}$$

$$a_k = 2\int_0^1 \sin(k\pi x)\left[\cos(\pi x) + \sin(\pi x) - 1 + 2x\right]\, dx \tag{27}$$

which gives

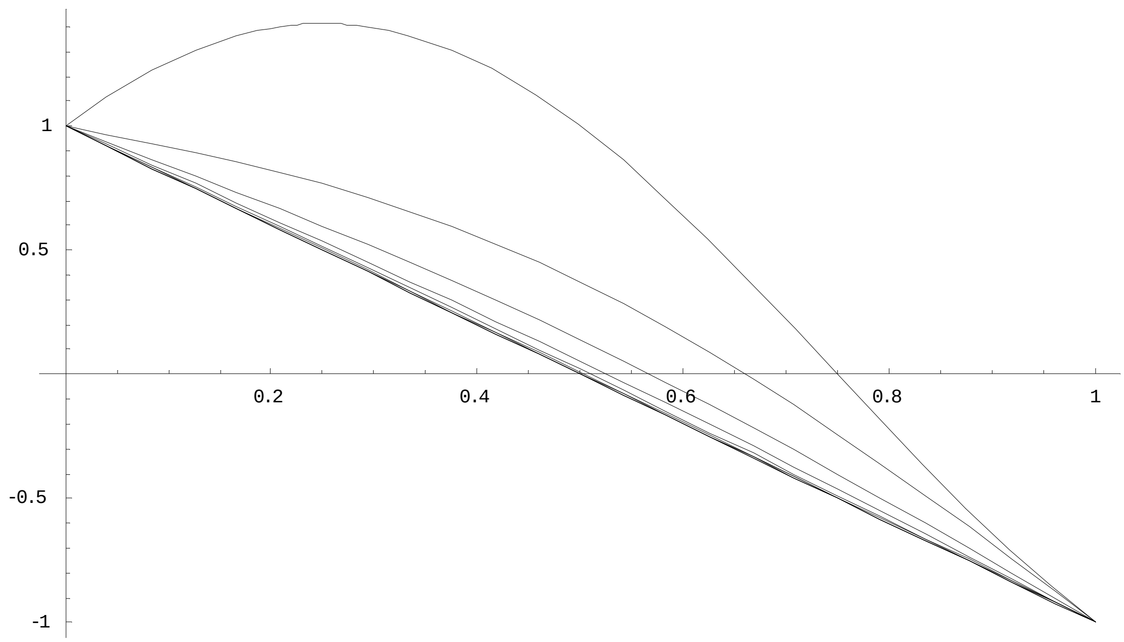
$$a_1 = 1 \tag{28}$$

$$a_k = \frac{4}{(k\pi)(k^2-1)} \text{ for even } k \tag{29}$$

$$a_k = 0 \text{ for odd } k > 1 \tag{30}$$

The overall solution is

$$q(x,t) = \sin(\pi x)e^{-\pi^2 t} + \sum_{p=1}^{\infty} \frac{2\sin(2p\pi x)}{(p\pi)(4p^2-1)} e^{-(2p\pi)^2 t} \tag{31}$$



and is plotted below for $t = 0, 0.1, \dots, 1$. The transient rapidly dies away. Note: this problem is not directly realted to numerical methods; nonetheless the analytical techniques used here should be familiar to all numerical methods practitioners since they are needed to construct reference solutions in which one evaluates the accuracy of a numerical method).

3.3 Experimenting with FTCS convergence

Now solve the above finite-domain problem using the FTCS algorithm to find the temperature distribution for $t \in [0, 0.5]$. Study the convergence of the method by:

1. Taking succesively smaller step sizes (k, h) in the FTCS discretization. Try three sequences:

   (a) $h = 2^{-p}$, $k = (p+1)h^2/(2p)$, $p = 2, 3, 4, 5, 6$;

   (b) $h = 2^{-p}$, $k = h^2/2$, $p = 2, 3, 4, 5, 6$;

   (c) $h = 2^{-p}$, $k = (p-1)h^2/(2p)$, $p = 2, 3, 4, 5, 6$;

2. Computing the relative error between the numerical approximation and the analytical solution

$$\varepsilon(t) = \frac{\|U_{num}(t) - u_{exact}(t)\|}{\|u_{exact}(t)\|} \,. \tag{32}$$

Use the $1-$norm

$$\|U_{num}\| = \sum_{j=1}^{M} |U_j| \tag{33}$$

in the above computation of the relative error ($h = 1/(m+1)$).

Plot the relative errors at $t_n = n/10$, $n = 1, 2, \ldots, 5$ as a function of h in log-log coordinates. Does the observed convergence behavior verify the theoretical predictions for the FTCS method? Comment on the difference in results for the three sequences of (k, h) specified above.

3.3.1 Solution

Discretization using FTCS is straightforward

$$Q_j^{n+1} = Q_j^n + \frac{k}{h^2}\left(Q_{j+1}^n - 2Q_j^n + Q_{j-1}^n\right), \; j = 1, 2, \ldots, M-1 \tag{34}$$

with boundary conditions $Q_0^n = 1, Q_M^n = -1$. A typical result and convergence plot is shown below for each sequence of time step sizes. Here's the Matlab code (for sequence (c)):

```
% Heat equation
clc; clf; clear all;
for p=2:6
    h=2^(-p);
    hp(p-1)=h;
    k=(p-1)*h^2/(2*p);
    M=2^p;
    x=(0:M)*h;
    t=0.; tfinal=0.1; sigma=k/h^2;
    q0=cos(pi*x)+sin(pi*x);
    q1=q0; qex=q0;
    % Time loop
    nt=0; Nsteps=floor(tfinal/k)+2;
    while t<tfinal
       % Compute using FTCS
       for i=2:M
         q1(i) = q0(i) + sigma*(q0(i+1)-2*q0(i)+q0(i-1));
       end
       % Iterate
       t=t+k; nt=nt+1;
       q0=q1;
       % Compute exact solution
       qex = sin(pi*x)*exp(-pi^2*t) + 1-2*x;
       for l=1:10
           ak = 2/(l*pi)/(4*l^2-1);
           fact = ak*exp(-(2*l*pi)^2*t);
```

```
            qex = qex + fact*sin(2*l*pi*x);
        end
        % Compare
        figure(1); plot(x,q1,x,qex,'o');
        xlabel('x'); ylabel('q');
        title(strcat('t=',num2str(t)));
        axis([0 1 -1 1.5]);
        pause(0.01);
    end
    err(p-1)=norm(q1-qex); eps(p-1)=err(p-1)/norm(q1);
    disp('Hit a key for next p');
    pause
end
figure(2);
loglog(hp,eps,'ok');
xlabel('log(h)'); ylabel('log(eps)')
afit=polyfit(log(hp),log(eps),1);
txt='Relative error convergence rate=';
txt=strcat(txt,num2str(afit(1)))
title(txt);
```

Sequence (a) Method is unstable.

Sequence (b) Method is stable.

Sequence (c) Method is stable.

3.4 Probing Crank-Nicolson convergence

Now solve the IVBP from problem 3 using the Crank-Nicolson scheme. Use a tridiagonal solver inside of your Crank-Nicolson time step. Study the convergence of the Crank-Nicolson method by intelligently reducing the step sizes (k, h). Again, plot your results as log-log graphs of the relative error versus h at different times. Do you obtain the predicted theoretical behavior? How do the FTCS and the Crank-Nicolson approaches compare from the viewpoint of practicality?

3.4.1 Solution

The Crank-Nicolson method is unconditionally stable and $O(k^2, h^2)$. We should use $k = O(h)$ when investigating the convergence of Crank-Nicolson as opposed to $k = O(h^2)$ used for FTCS. For instance we could try the sequences $k = (p + l)h/p$ with $l = -1, 0, 1$ or something similar. We can rewirte the Crank-Nicolson update as

$$-Q_{j+1}^{n+1} + (2+\sigma)Q_j^{n+1} - Q_{j-1}^{n+1} = Q_{j+1}^n - (2-\sigma)Q_j^n + Q_{j-1}^n = b_j \qquad (35)$$

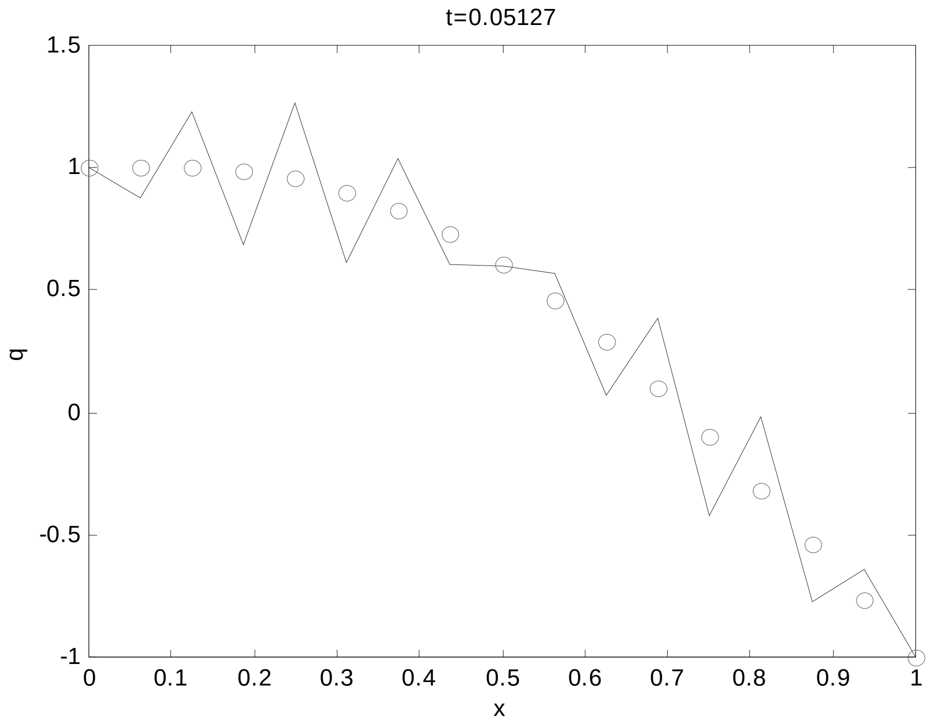


Figure 1: Typical result showing incipient instability.

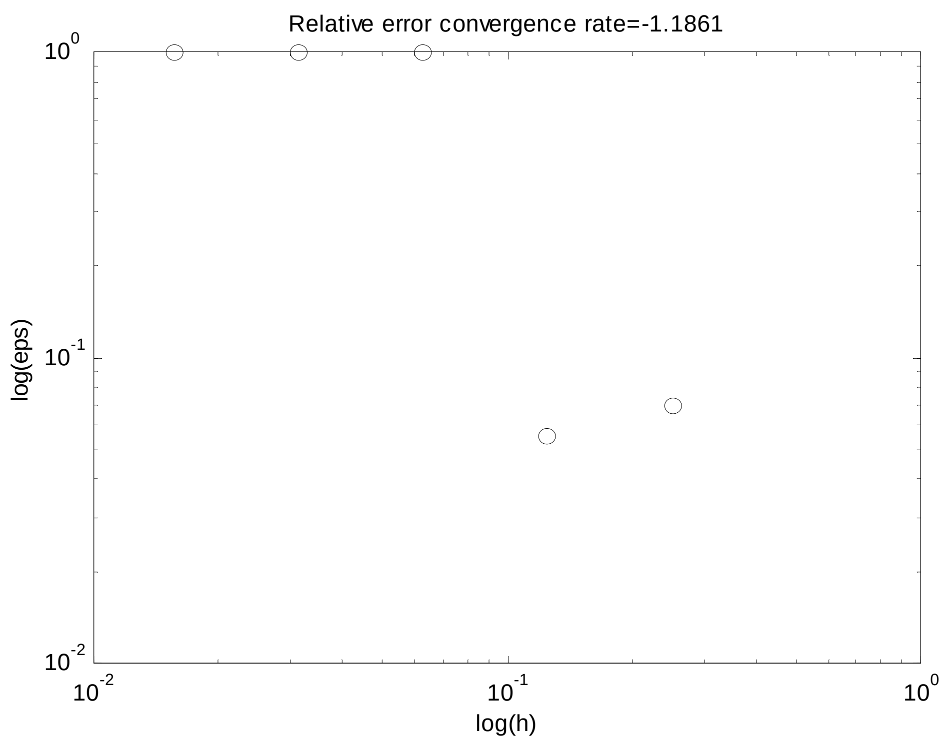


Figure 2: Error plot showing method divergence when time steps exceeding the stsbility limit are used.

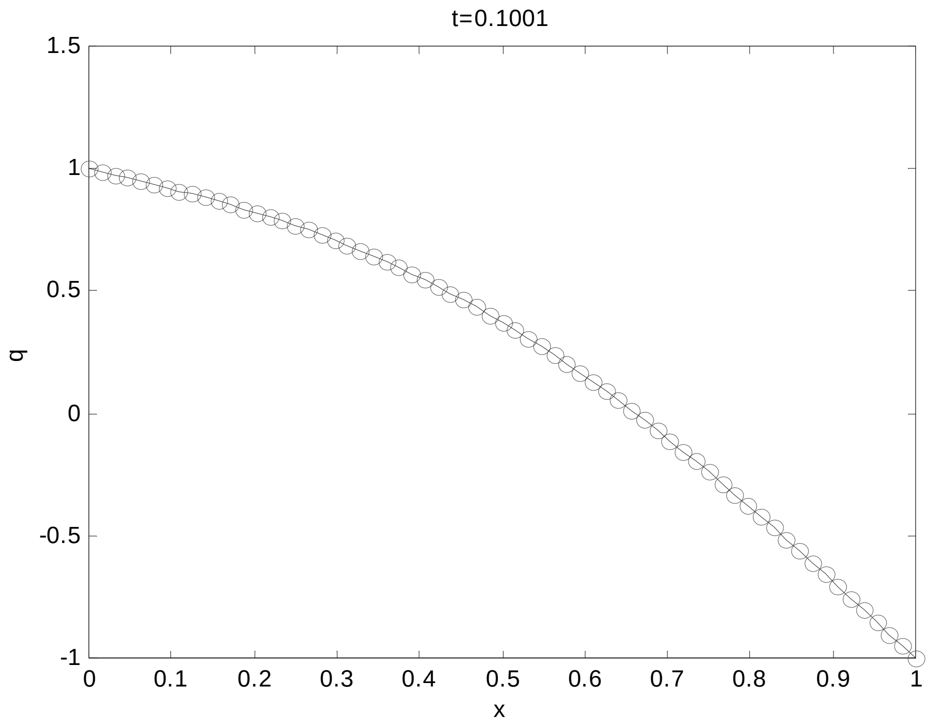


Figure 3: Typical numerical result using FTCS (line=numerical, circles=exact solution)

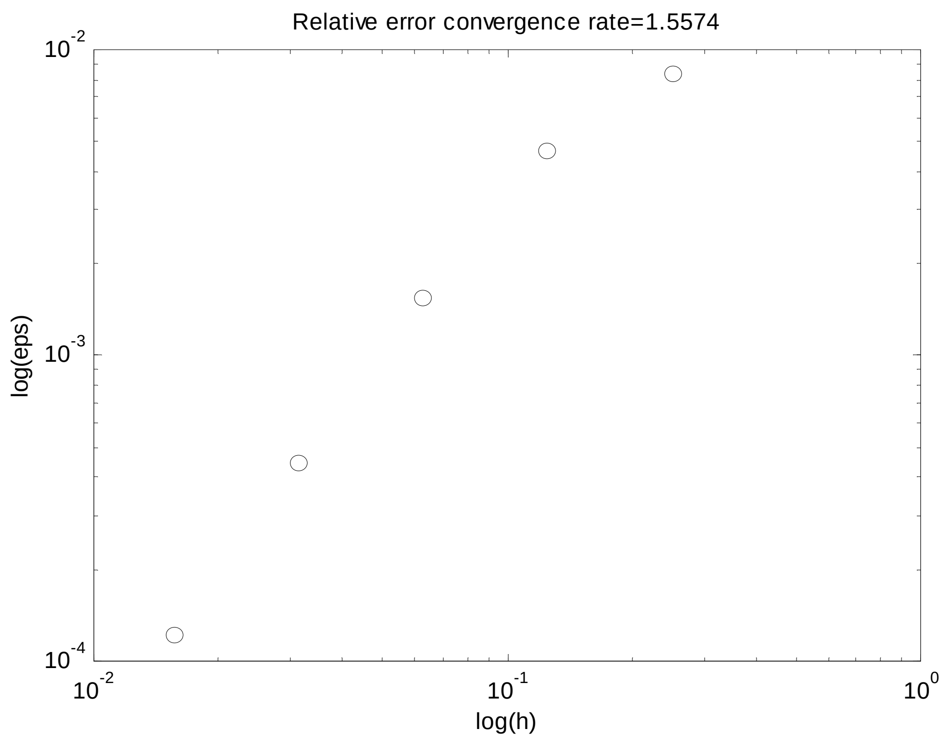


Figure 4: Convergence plot at $t = 0.1$. Though the method approaches $O(h^2)$ convergence it isn't observed yet for this range of step sizes.

for $j = 1, 2, \ldots, M-1$ with $Q_0^n = 1$ and $Q_M^n = -1$ at all time levels n and $\sigma = 2h^2/k$. Note that the tridiagonal system above is diagonally dominat and hence will exhibit good numerical properties. We can either directly code the tridiagonal solve or use built-in Matlab facilities for handling sparse matrices. The Matlab code below shows the use of sparse matrices. For $k = h$ we get the results shown below. Crank-Nicolson is much more efficient in terms of computation time by comparison to FTCS because of the larger permitted step sizes.

```
% Heat equation
clc; clf; clear all;
for p=2:8
    h=2^(-p);
    hp(p-1)=h;
    k=h;
    M=2^p;
    x=(0:M)*h;
    t=0.; tfinal=0.1; sigma=k/h^2;
    q0=cos(pi*x)+sin(pi*x);
    q1=q0; qex=q0;
    % Time loop
    nt=0;
    % Form Crank-Nicolson sparse matrix
    e = ones(M-1,1); sigma=2*h^2/k;
    A = spdiags([-e (2+sigma)*e -e], -1:1, M-1, M-1);
    while t<tfinal
       % Compute using Crank-Nicolson
       % 1. form rhs of linear system
       for i=2:M
          b(i) = q0(i+1) - (2-sigma)*q0(i) + q0(i-1);
       end
       % 2. Impose b.c.'s
       b(2)=b(2)+q1(1); b(M)=b(M)+q1(M+1);
       % 3. Solve tridiag system
       q1(2:M) = (A \ b(2:M)')';
       % Iterate
       t=t+k; nt=nt+1;
       q0=q1;
       % Compute exact solution
       qex = sin(pi*x)*exp(-pi^2*t) + 1-2*x;
       for l=1:10
           ak = 2/(l*pi)/(4*l^2-1);
           fact = ak*exp(-(2*l*pi)^2*t);
           qex = qex + fact*sin(2*l*pi*x);
       end
       % Compare
```

```
        figure(1); plot(x,q1,x,qex,'o');
        xlabel('x'); ylabel('q');
        title(strcat('t=',num2str(t)));
        axis([0 1 -1 1.5]);
        pause(0.01);
    end
    err(p-1)=norm(q1-qex); eps(p-1)=err(p-1)/norm(q1);
    disp('Hit a key for next p');
    pause
end
figure(2);
loglog(hp,eps,'ok');
xlabel('log(h)'); ylabel('log(eps)')
afit=polyfit(log(hp),log(eps),1);
txt='Relative error convergence rate=';
txt=strcat(txt,num2str(afit(1)))
title(txt);
```

3.5 Bonus: 2D ADI and boundary conditions

Consider the 2D problem

$$\begin{aligned} q_t &= q_{xx} + q_{yy} \\ q(x,y,0) &= \sin(\pi xy) \end{aligned} \tag{36}$$

with zero conditions on $q(x,y,t)$ on the boundary of the unit square $[0,1] \times [0,1]$. How do you generate boundary conditions for the intermediate stage value Q^* when applying the ADI (alternating direction implicit) algorithm to this problem?

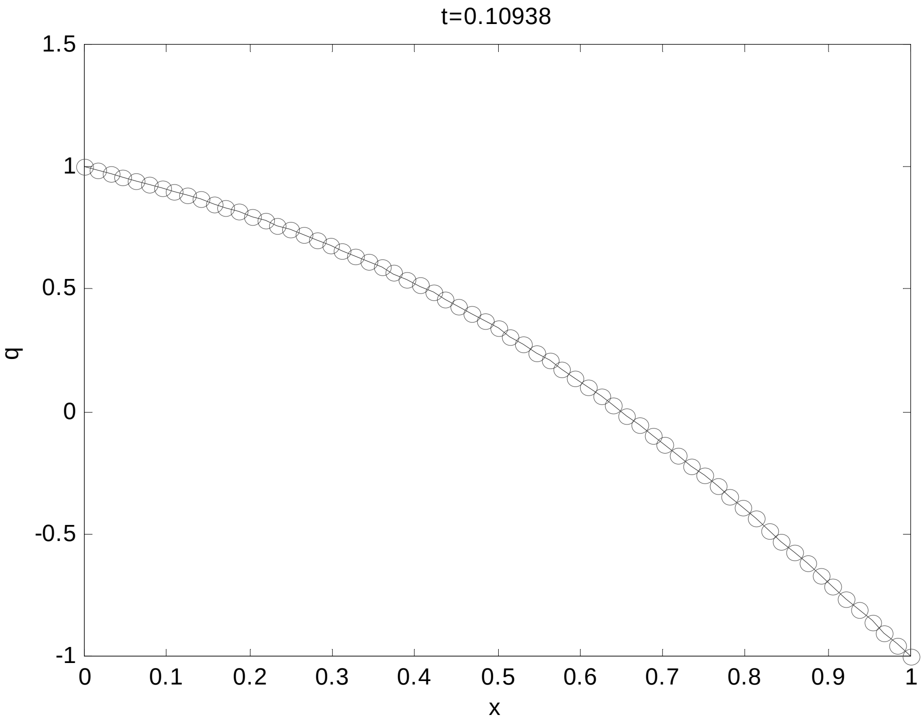


Figure 5: Typical result from Crank-Nicolson computation at $t = 0.1$ using $k = h$.

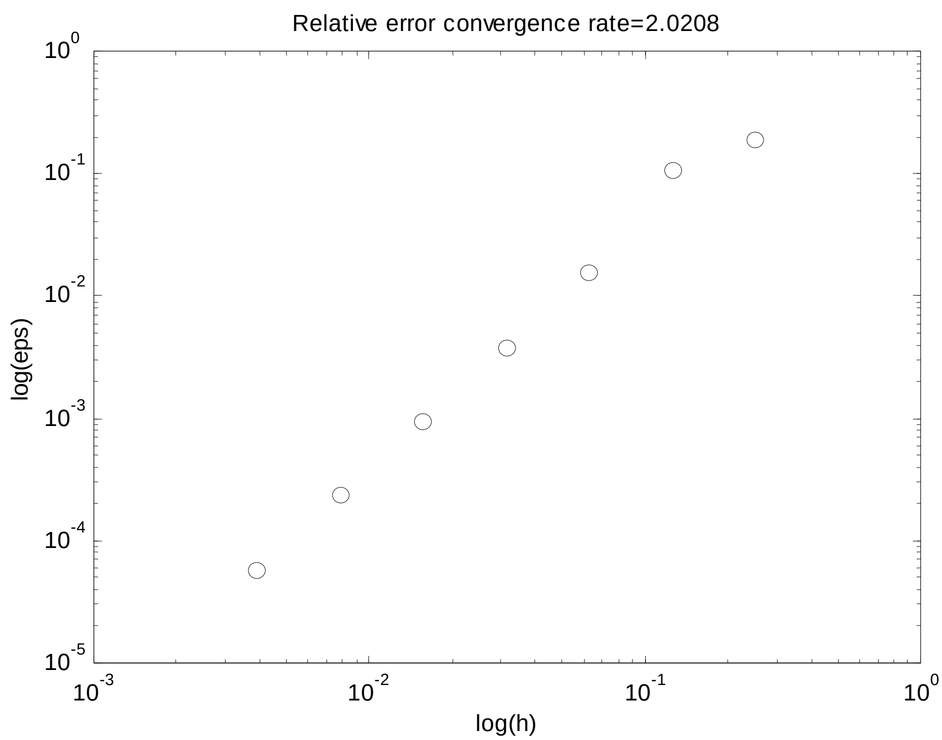


Figure 6: Convergence plot for Crank-Nicolson at $t = 0.1$ showing we do obtain the predicted $O(h^2)$ convergence behvaior. Note that it was possible to extend the range of stepsizes to $h = 2^{-8}$ since the method uses much less computational time.