Amath 586: Homework 4

Due on May 12, 2015

Brian de Silva

Problem 1

Consider the following method for solving the heat equation $u_t = u_{xx}$:

$$U_i^{n+2} = U_i^n + \frac{2k}{h^2}(U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}).$$

(a) Determine the formal order of accuracy of this method (in both space and time) based on computing the local truncation error.

(b) Suppose we take $k = \alpha h^2$ for some fixed $\alpha > 0$ and refine the grid. Show that this method fails to be Lax-Richtmyer stable for any choice of α.

Do this in two ways:

- Consider the MOL interpretation and the stability region of the time-discretization being used.
- Use von Neumann analysis and solve a quadratic equation for $g(\xi)$.

(c) What if we take $k = \alpha h^3$ for some fixed $\alpha > 0$ and refine the grid. Would this method be stable?

Solution:

(a) Throughout this derivation of the local truncation error, let u denote $u(x, t+k)$, u_t $\frac{\partial}{\partial t}(u)|_{x,t+k}$, etc.. Rather than expanding each term about $u(x,t)$, we expand about $u(x,t+k)$ and interpret the method as the midpoint rule applied at $u(x,t+k)$. We have the following Taylor expansions:

$$\begin{aligned}
u(x,t+2k) &= u + ku_t + \frac{k^2}{2}u_{tt} + \frac{k^3}{6}u_{ttt} + O(k^4),\\
u(x,t) &= u - ku_t + \frac{k^2}{2}u_{tt} - \frac{k^3}{6}u_{ttt} + O(k^4),\\
u(x+h,t) &= u + hu_x + \frac{h^2}{2}u_{xx} + \frac{h^3}{6}u_{xxx} + \frac{h^4}{4!}u_{xxxx} + O(h^5),\\
u(x-h,t) &= u - hu_x + \frac{h^2}{2}u_{xx} - \frac{h^3}{6}u_{xxx} + \frac{h^4}{4!}u_{xxxx} + O(h^5).
\end{aligned}$$

Substituting these into the local truncation error expression

$$\tau(x,t+k) = \frac{u(x,t+2k) - u(x,t)}{2k} - \frac{1}{h^2}(u(x-h,t+k) - 2u(x,t+k) + u(x+h,t+k))$$

and making obvious cancellations, we obtain

$$\begin{aligned}
\tau(x,t+k) &= u_t + \frac{k^2}{6}u_{tt} + O(k^3) - (u_{xx} + \frac{h^2}{12}u_{xxxx} + O(h^3))\\
&= \left(\frac{k^2}{6} + \frac{h^2}{12}\right)u_{xxxx} + O(k^3 + h^3),
\end{aligned}$$

where we have used $u_t = u_{xx}$ to cancel terms and replace $u_{tt} with u_{xxxx}$. Hence the method is formally second order accurate in both h and k.

Problem 1 continued on next page...

(b) (i) **Method of Lines**

We interpret the above finite difference method as the midpoint method applied to the system of ODEs

$$U_i'(t) = \frac{1}{h^2}(U_{i-1}(t) - 2U_i(t) + U_{i+1}), \qquad i = 1, 2, \ldots, m,$$

which can be expressed more compactly as

$$U'(t) = AU(t) + g(t),$$

where

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & & 1 \\ & & & & 1 & -2 \end{bmatrix}$$

and $g(t)$ simply takes care of the boundary terms.

As was noted in class in order for this method to be stable we need each of the roots of $\pi(\zeta; k\lambda_p)$ to be less than one in magnitude, where λ_p is the p^{th} eigenvalue of A and $\pi(\zeta; z)$ is defined with respect to the midpoint method. For this to hold we need $k\lambda_p \in S$ for each p, where S is the stability region for the midpoint method. Or we at least need $k\lambda_p$ to approach S as k is refined. Recall that $S = z \in \mathbb{C} | z = ib,\ b \in \mathbb{R},\ |b| \leq 1$. A has eigenvalues $\lambda_p = \frac{2}{h^2}(cos(p\pi h) - 1)$ for $p = 1, 2, \ldots, m$, so if the method is to be stable it must satisfy $k\lambda_p = \frac{2k}{h^2}(cos(p\pi h) - 1) \in S$. The eigenvalue farthest from the origin is approximately $\frac{-4}{h^2}$, so $\frac{-4k}{h^2} \in S$ for stability. If we choose $k = \alpha h^2$ then this becomes $-4\alpha \notin S$. Hence the method is not stable for any choice of $\alpha > 0$.

(ii) **von Neumann Analysis**

Letting $\hat{U}^n$ be the Fourier transform of the grid function U^n we have

$$\hat{U}^{n+1} = g(\xi)\hat{U}^n.$$

If it can be shown that

$$|g(\xi)| \leq 1 + \alpha k$$

with α independent from ξ then it will follow that the method is stable. If we can show the opposite, that

$$|g(\xi)| > 1 + \alpha k$$

then the method is unstable. Assuming

$$U_j^n = e^{ijh\xi}$$

and

$$U_j^{n+1} = g(\xi)e^{ijh\xi},$$

we can express the finite difference scheme in terms of its Fourier transform and solve for $g(\xi)$

$$\begin{aligned} & g(\xi)^2 e^{ijh\xi} = e^{ijh\xi} + \frac{k}{h^2} g(\xi)(e^{i(j-1)h\xi} - 2e^{ijh\xi} + e^{i(j+1)h\xi}) \\ & \Rightarrow g(\xi)^2 = 1 + g(\xi)\frac{4k}{h^2}(cos(\xi h) - 1) \\ & \Rightarrow g(\xi)^2 - g(\xi)\frac{4k}{h^2}(cos(\xi h) - 1) - 1 = 0 \\ & \Rightarrow g(\xi) = \frac{2k}{h^2}(cos(\xi h) - 1) \pm \left(\frac{4k^2}{h^4}(cos(\xi h)^2 + 1\right)^{\frac{1}{2}}. \end{aligned}$$

Problem 1 continued on next page...

For $\xi = \frac{\pi}{h},\ cos(\xi h) = -1$ and so the lesser of the two possible solutions for g becomes

$$g(\xi) = \frac{-4k}{h^2} - \left(\frac{4k^2}{h^4} + 1\right)^{\frac{1}{2}}.$$

If we set $k = \alpha h^2$ then this simplifies to

$$g(\xi) = -4\alpha - \left(16\alpha^2 + 1\right)^{\frac{1}{2}} < -1$$

for any choice of $\alpha > 0$. Hence the method is unstable for any choice of $\alpha > 0$.

(c) Consider the condition for stability specified during the Method of Lines analysis. The eigenvalue farthest from the origin was $\lambda_p \approx \frac{-4}{h^2}$, so if we take $k = \alpha h^3$, $\lambda_p k \approx -4\alpha h$ which tends to $0 \in S$ as the grid is refined, i.e. it approaches S as $h \to 0$. As we saw in class, this allows us to conclude that the method is stable.

Problem 2

Consider the PDE

$$u_t = \kappa u_{xx} - \gamma u, \tag{1}$$

which models diffusion combined with decay provided $\kappa > 0$ and $\gamma > 0$. Consider methods of the form

$$U_j^{n+1} = U_j^n + \frac{k\kappa}{2h^2}[U_{j-1}^n - 2U_j^n + U_{j+1}^n + U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}] - k\gamma[(1-\theta)U_j^n + \theta U_j^{n+1}] \tag{2}$$

where θ is a parameter. In particular, if $\theta = 1/2$ then the decay term is modeled with the same centered-in-time approach as the diffusion term and the method can be obtained by applying the Trapezoidal method to the MOL formulation of the PDE. If $\theta = 0$ then the decay term is handled explicitly. For more general reaction-diffusion equations it may be advantageous to handle the reaction terms explicitly since these terms are generally nonlinear, so making them implicit would require solving nonlinear systems in each time step (whereas handling the diffusion term implicitly only gives a linear system to solve in each time step).

(a) By computing the local truncation error, show that this method is $\mathcal{O}(k^p + h^2)$ accurate, where $p = 2$ if $\theta = 1/2$ and $p = 1$ otherwise.

(b) Using von Neumann analysis, show that this method is unconditionally stable if $\theta \geq 1/2$.

(c) Show that if $\theta = 0$ then the method is stable provided $k \leq 2/\gamma$, independent of h.

Solution:

(a) In this derivation of the local truncation error let u denote $u(x,t)$, u_t denote $\frac{\partial}{\partial t}(u)$, etc.. We Taylor expand about $u(x,t)$:

$$\begin{aligned}
u(x,t+k) &= u + ku_t + \frac{k^2}{2}u_{tt} + \frac{k^3}{4!}u_{ttt} + O(k^4) \\
u(x-h,t) &= u - hu_x + \frac{h^2}{2}u_{xx} - \frac{h^3}{6}u_{xxx} + \frac{h^4}{4!}u_{xxxx} + O(h^5) \\
u(x+h,t) &= u + hu_x + \frac{h^2}{2}u_{xx} + \frac{h^3}{6}u_{xxx} + \frac{h^4}{4!}u_{xxxx} + O(h^5) \\
u(x+h,t+k) &= u + hu_x + ku_t + \frac{1}{2}(h^2u_{xx} + k^2u_{tt} + 2hku_{xt}) + \frac{1}{6}(h^3u_{xxx} + k^3u_{ttt} + 3hk^2u_{xtt} + 3h^2ku_{xxt}) + \dots \\
u(x-h,t+k) &= u - hu_x + ku_t + \frac{1}{2}(h^2u_{xx} + k^2u_{tt} + 2hku_{xt}) + \frac{1}{6}(-h^3u_{xxx} + k^3u_{ttt} - 3hk^2u_{xtt} + 3h^2ku_{xxt}) + \dots .
\end{aligned}$$

Substituting these into the finite difference equation and making obvious cancellations yields

$$\begin{aligned}
\tau(x,t) &= (u_t - \kappa u_{xx} + \gamma u) + \frac{k}{2}u_{tt} + \frac{k^2}{6}u_{ttt} - \frac{\kappa h^2}{12}u_{xxxx} - \frac{k\kappa}{2}u_{xxt} + \gamma\theta k u_t \\
&\quad + \frac{\theta\gamma k^2}{2}u_{tt} + O(h^4 + k^3) \\
&= \frac{k}{2}\frac{\partial}{\partial t}(u_t - \kappa u_x x + 2\gamma\theta u) + \frac{k^2}{6}u_{ttt} - \frac{\kappa h^2}{12}u_{xxxx} + O(h^4 + k^3).
\end{aligned}$$

If $\theta = \frac{1}{2}$ we can use the PDE to cancel the $O(k)$ term in the local truncation error, giving overall $O(k^2 + h^2)$ accuracy. Otherwise this term persists and the method is only $O(k + h^2)$ accurate.

Problem 2 continued on next page...

(b) We proceed in a similar manner as before, we substitute $e^{ijh\xi}$ for U_j^n, $g(\xi)e^{ijh\xi}$ for U_j^{n+1}, etc. into the finite difference scheme to determine $g(\xi)$

$$g(\xi)e^{ijh\xi} = e^{ijh\xi} + e^{ijh\xi}\frac{2k\kappa}{2h^2}\left(cos(\xi h) - 1 + g(\xi)(cos(\xi h) - 1)\right) - k\gamma e^{ijh\xi}(1 - \theta + \theta g(\xi))$$
$$\Rightarrow g(\xi) = \frac{h^2(1 - k\gamma + k\gamma\theta) + k\kappa(cos(\xi h) - 1)}{h^2(1 + k\gamma\theta) + k\kappa(1 - cos(\xi h))}.$$

Letting $C = k\kappa(1 - cos(\xi h))$, we see that $C \geq 0$ for any ξ and g can be written

$$g(\xi) = 1 - \frac{h^2 k\gamma + 2C}{h^2(1 + k\gamma\theta) + C}.$$

It is easy to see that this quantity will always be less than 1, so the only way the method could be unstable is if $g(\xi) < -1$. For this to occur we would need to have

$$\begin{aligned} &\frac{h^2 k\gamma + 2C}{h^2(1 + k\gamma\theta) + C} \geq 2 \\ \Rightarrow \quad & h^2 k\gamma + 2C \geq 2(h^2(1 + k\gamma\theta) + C) \\ \Rightarrow \quad & k\gamma \geq 2(1 + k\gamma\theta) \\ \Rightarrow \quad & k\gamma(1 - 2\theta) \geq 2. \end{aligned}$$

If $\theta \geq \frac{1}{2}$ then $1 - 2\theta \leq 0 \;\Rightarrow\; k\gamma(1 - 2\theta) \leq \; 0 \; \leq 2 \quad \forall k > 0$. Hence the method is unconditionally stable for $\theta \geq \frac{1}{2}$.

(c) If $\theta = 0$ then for $g(\xi)$ as above, $|g(\xi)| \leq 1$ if $k\gamma(1 - 2\theta) = k\gamma \; \leq 2 \; \Rightarrow k \; \leq \; \frac{2}{\gamma}$. Note that this restriction holds independent of h.

Problem 3

(a) The m-file `heat_CN.m` from the book repository solves the heat equation $u_t = \kappa u_{xx}$ (with $\kappa = 0.02$) using the Crank-Nicolson method.

Run this code, and by changing the number of grid points, confirm that it is second-order accurate. (Observe how the error at some fixed time such as $T = 1$ behaves as k and h go to zero with a fixed relation between k and h, such as $k = 4h$.)

Note that in order for the time step to evenly define the time interval the way this code is set up, you need to specify values such as $m = 19, 39, 79$. (The number of interior grid points.)

Produce a log-log plot of the error versus h.

(b) Modify this code to produce a new version that implements the TR-BDF2 method on the same problem. Test it to confirm that it is also second order accurate. Explain how you determined the proper boundary conditions in each stage of this Runge-Kutta method.

(c) Modify the code to produce a new m-file or Python code `heat_FE` that implements the forward Euler explicit method on the same problem. Test it to confirm that it is $\mathcal{O}(h^2)$ accurate as $h \to 0$ provided when $k = 24h^2$ is used, which is within the stability limit for $\kappa = 0.02$. Note how many more time steps are required than with Crank-Nicolson or TR-BDF2, especially on finer grids.

(d) Test `heat_FE` with $k = 26h^2$, for which it should be unstable. Note that the instability does not become apparent until about time 4.5 for the parameter values $\kappa = 0.02,\ m = 39,\ \beta = 150$. Explain why the instability takes several hundred time steps to appear, and why it appears as a sawtooth oscillation.

Hint: What wave numbers ξ are growing exponentially for these parameter values? What is the initial magnitude of the most unstable eigenmode in the given initial data? The expression (E.30) for the Fourier transform of a Gaussian may be useful.

Solution:
Note: All trials are run from $t = 0$ to final time $t = 1$ unless explicitly stated otherwise.

(a) To test `heat_CN.m` I observed how the error behaved at $T = 1$ as k and h were refined for $k = 4h$.

As is demonstrated in Figure 1 the Crank-Nicolson method exhibits second-order accuracy when applied to the heat equation with $\kappa = 0.02$. The numerically approximated order of accuracy was 2.06.

(b) Next, to test the TR-BDF2 method I created, I followed the same procedure as in (a) with all parameters the same as for that test.

Figure 2 illustrates that the method is achieving second order accuracy on the heat equation as well. The boundary conditions for the first step of the two-stage Runge-Kutta method were the same as in the Crank-Nicolson method, except with the boundary conditions evaluated at time $t + \frac{k}{2}$ rather than at t since we are only taking a half-step at the first stage. In the second stage the boundary conditions were determined from the following equation

$$U_j^{n+1} = \frac{1}{3}(4U_j^* - U_j^n + \frac{k\kappa}{h^2}(U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1})).$$

Letting $j = 1$ and $j = m$ we find that the first and last components (first and mth) of the right-hand-side vector need to have added to them the values at the left and right boundaries (multiplied by $\frac{k\kappa}{3h^2}$) at time $t + k$, respectively.The numerically approximated order of accuracy was 2.00. See the m-file `heat_TRBDF2.m`.

Problem 3 continued on next page...

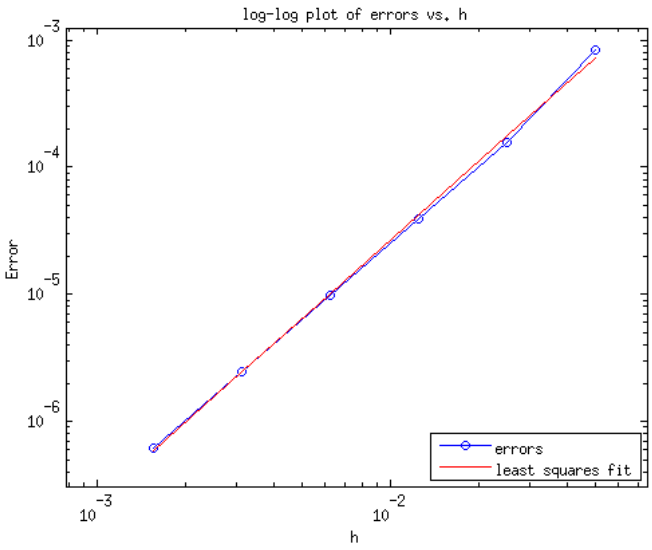


Figure 1: log-log plot of error using Crank-Nicolson method

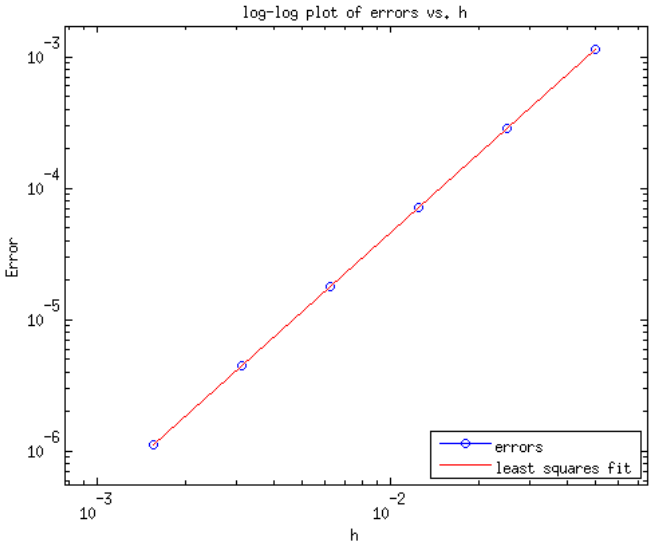


Figure 2: log-log plot of error using TR-BDF2 method

(c) After modifying `heat_CN.m` to use the Forward Euler method to advance the numerical solutions forward in time I tested its accuracy in a similar manner as before, but setting $k = 24h^2$.

Figure 3 demonstrates that the method achieved second order accuracy. The numerically approximated order of accuracy was 2.00. See the m-file `heat_FE.m`.

(d) For this problem I used final time $t = 5$ instead of $t = 1$. Setting $k = 26h^2$ causes the method of lines approach using Forward Euler to discretize the time derivative in the heat equation to be unstable.

In Figure 4 we see how the instability progresses as more time steps are taken when 41 grid points are used. Notice how the instability does not become noticeable until after roughly 260-280 time steps have been taken. The instability gets much worse if time is progressed further (see the attached Matlab notebook).

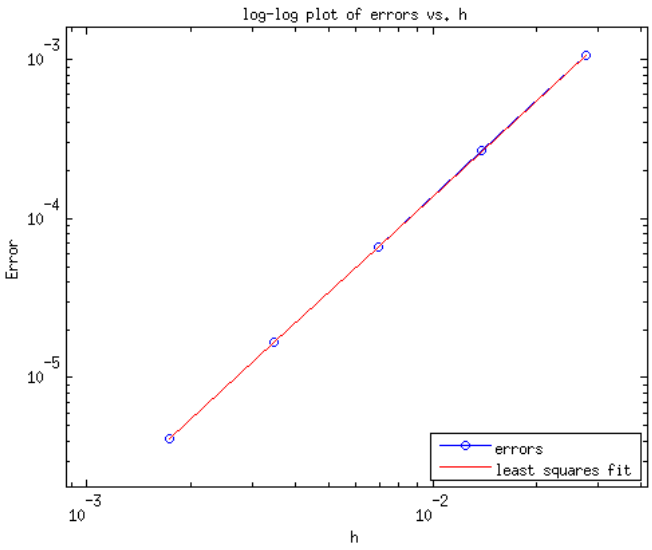


Figure 3: log-log plot of error using Forward Euler method

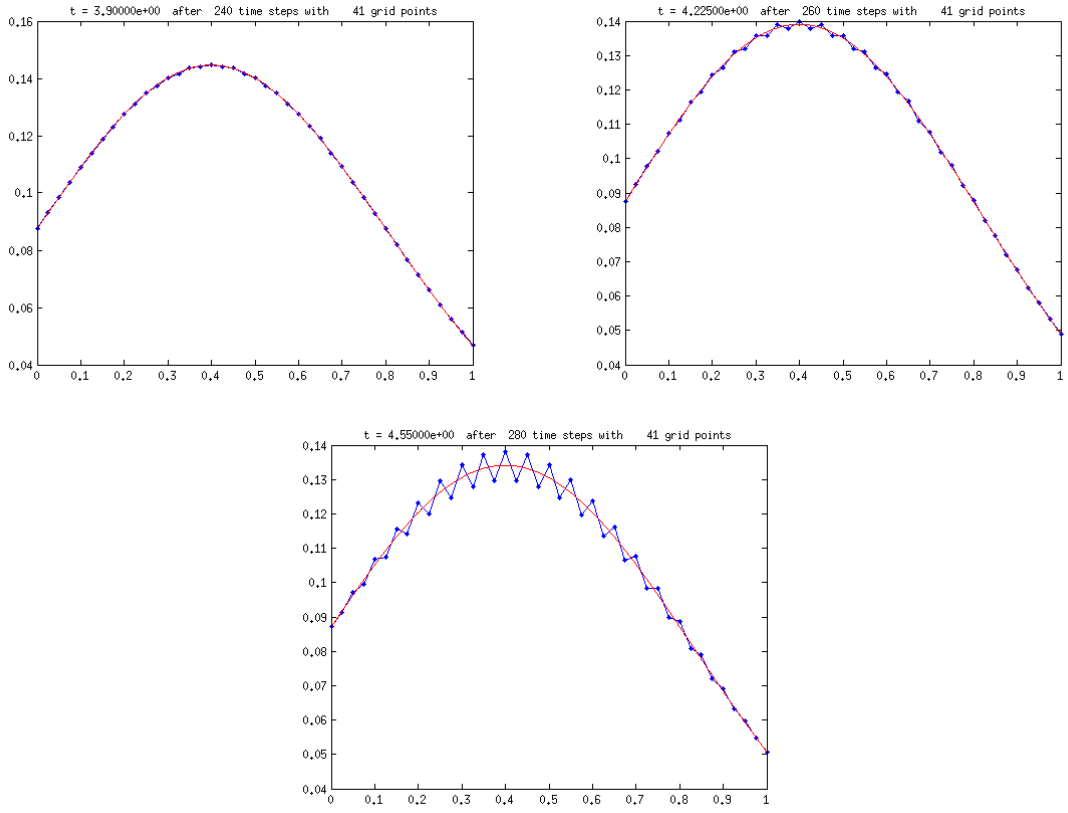


Figure 4: Evolution of instability in Forward Euler Method of Lines (left: 240 time steps, middle: 260, right: 280

Recall from the earlier von Neumann analysis that if we consider how the method works on a single wavenumber by setting $U_j^n = e^{ijh\xi}$ and $U_j^{n+1} = g(\xi)e^{ijh\xi}$, then the method will be unstable if $|g(\xi)| > 1$.

Substituting the expressions above into the method and solving for $g(\xi)$ yields

$$g(\xi) = 1 + \frac{2k\kappa}{h^2}(cos(\xi h) - 1).$$

Observe that if we let $k = 26h^2$ and $\xi = \frac{\pi}{h}$ (the largest wavenumber representable on our grid), then $g(\xi) = -1.08$. Thus for the wavenumber $\xi = \frac{\pi}{h}$, after n time steps the method will give $\hat{U}_j^n = (-1.08)^n \hat{U}_j^0$, where $\hat{U}_j^n$ is the Fourier transform of U_j^n. Using Parseval's identity, the initial magnitude of this unstable eigenmode is simply the magnitude of the Fourier transform of the initial condition, evaluated at $\xi = \frac{\pi}{h}$. Following the hint and using the fact that $u(x,0) = u_0(x) = e^{-\beta(x-0.4)^2}$, we have

$$|\hat{u}_0(\xi)| = \frac{1}{\sqrt{2\beta}} e^{\frac{-\xi^2}{4\beta}}.$$

For $\beta = 150$, $\xi = \frac{\pi}{h} = 40\pi$, $|\hat{u}_(\xi)| \approx 2.14 \times 10^{-13}$. Since $(1.08)^n \cdot \approx 2.14 \times 10^{-13}$ does not reach $O(10^{-3}$ until some time after $n = 200$ the exponential growth in this eigenmode is not noticed until t becomes fairly large. The reason the instability manifests itself as a sawtooth oscillation is because a sawtooth oscillation is exactly the way the eigenmode corresponding to $\xi = \frac{\pi}{h}$ is represented on our chosen grid. It oscillates at a rate exactly matching our grid spacing and so the grid points only capture its points of highest and lowest amplitude, one after another.

Problem 4

(a) Modify `heat_CN.m` (or `heat_CN.py`) to solve the heat equation for $-1 \leq x \leq 1$ with step function initial data

$$u(x,0) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{if } x \geq 0. \end{cases} \tag{3}$$

With appropriate Dirichlet boundary conditions, the exact solution is

$$u(x,t) = \frac{1}{2} \operatorname{erfc}\left(x/\sqrt{4\kappa t}\right) \tag{4}$$

for $t > 0$, where erfc is the *complementary error function*

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-z^2}\, dz.$$

Note that `erfc` is a built-in Matlab function and in Python you could use `scipy.special.erfc`.

(i) Use $\kappa = 0.02$ and test this method using $m = 39$ and $k = 10h$ for $-1 \leq x \leq 1$. Note that there is an initial rapid transient decay of the high wave numbers that is not captured well with this size time step.

(ii) How small do you need to take the time step to get reasonable results? For a suitably small time step, explain why you get much better results by using $m = 38$ than $m = 39$. What is the observed order of accuracy as $k \to 0$ when $k = \alpha h$ with α suitably small and m even?

Hint: You might also see what you get if you redefine the initial data so that $u(0,0) = 1/2$ with $u(x,0)$ the same everywhere else, rather than $u(0,0) = 0$.

(b) Modify your TR-BDF2 solver from Problem 3 to solve the heat equation for $-1 \leq x \leq 1$ with step function initial data as above. Test this routine using $k = 10h$ and estimate the order of accuracy as $k \to 0$ with m even. Why does the TR-BDF2 method work better than Crank-Nicolson?

Solution:

(a) The modified version of `heat_CN.m` is stored in `heat_CN_alt_BCs.m`.

(i) First we test the method using the specified parameters.

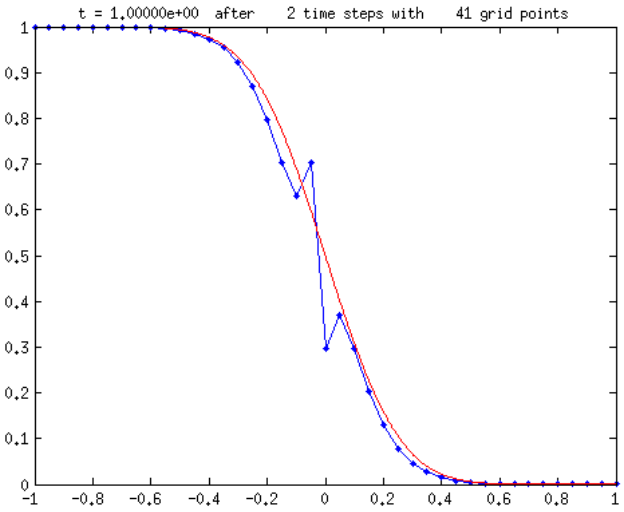


Figure 5: Crank-Nicolson Method with $k = 10h$

Figure 5 shows the numerical solution generated by the Crank-Nicolson method when $k = 10h$. Notice the oscillation which occurs at $x = 0$. The maximum error at $t = 1$ is 0.204.

(ii) After multiple numerical tests using even and odd numbers of grid points, $\alpha = 2$ seems to be an appropriate restriction to get reasonable results. That is for $\alpha = 2$ the error in the approximation at $t = 1$ behaves predictably as k is refined. If α is taken to be much larger than 2, 2.5, say, then as the grid is refined, eventually the error stops decreasing and starts leveling off instead.

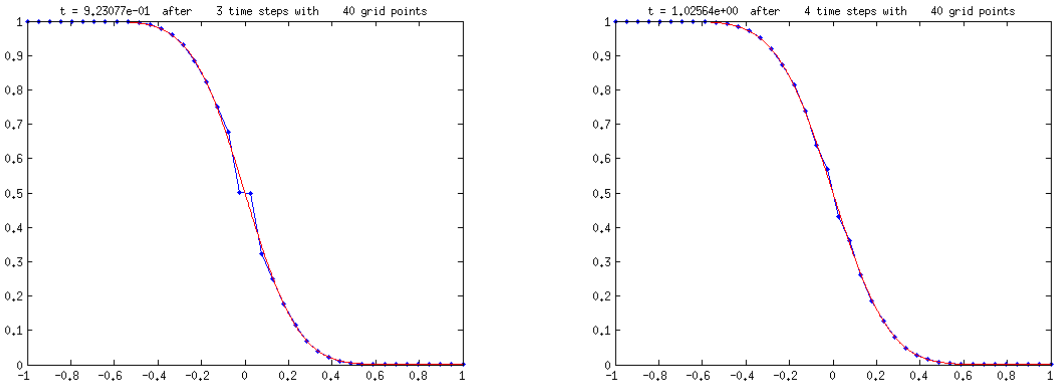


Figure 6: Crank-Nicolson method with left:$k = 6h$, right:$k = 5h$

Figure 6 shows an example of how the solution seems able to resolve the transient for $\alpha = 5$, but not $\alpha = 6$. However, if the grid is further refined, we find that $\alpha = 5$ is too lax a condition as mentioned above, and the error starts to worsen.

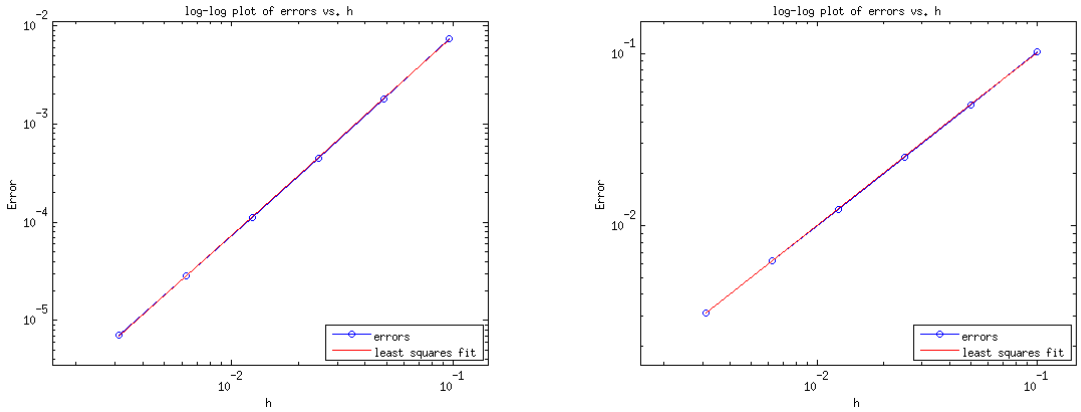


Figure 7: Crank-Nicolson method with (left:even, right:odd) number of grid points

In Figure 7 we have the maximum error at $t = 1$ using even and odd numbers of grid points. The computed orders of accuracy were 2.03 using an even number and 1.01 using an odd number. The reason for the disparity in accuracy between the two cases is that when an odd number of grid points are used, one of the points lies directly at 0. This causes the initial condition to be less accurately represented on the odd grid than the even one. If we follow the hint and set $u(0, 0) = \frac{1}{2}$ then the method attains second order accuracy on both the even and odd grids. This tells us that the difference in performance must be due to the way the initial condition is resolved.

(b) The TR-BDF2 solver allows us to take a larger time step than we could using the Crank-Nicolson method, resulting in faster overall performance. As noted in part (a)(ii) of the problem, it is essential that we choose an even number of grid points to better resolve the initial condition and get second order convergence.

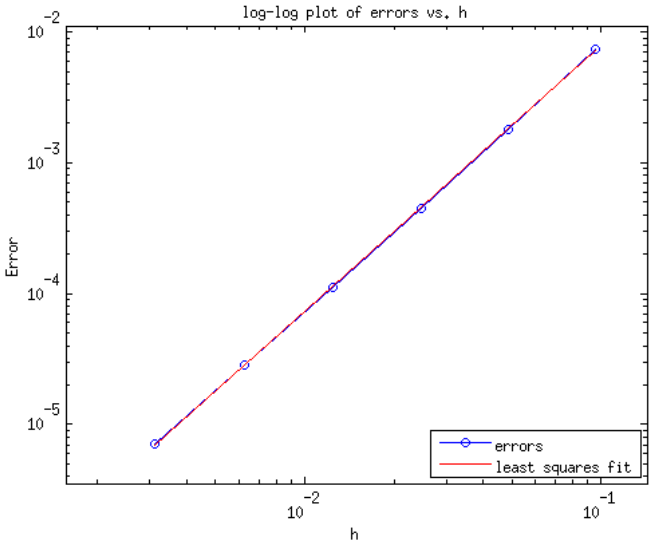


Figure 8: TR-BDF2 Method error with $k = 10h$

In Figure 8 we show the behavior of the maximum error in the TR-BDF2 method as k is decreased. This method works better than Crank-Nicolson because it is both L-stable and A-stable, whereas the implicit method relied upon in the Crank-Nicolson method, the Trapezoid rule, is merely A-stable.

Note that the TR-BDF2 method also uses the Trapezoid method in one of its stages, but it is L-stable nevertheless. The rapid initial transient in the solution leads to a loss in stability in the Crank-Nicolson method, but not TR-BDF2, since rapid initial transients are known to be troublesome for the trapezoidal method by itself. The numerically approximated order of accuracy is 2.59. The code for the modified method can be found in `TRBDF2_alt.m`.

Appendix: Code

Amath 586 homework 4

Table of Contents

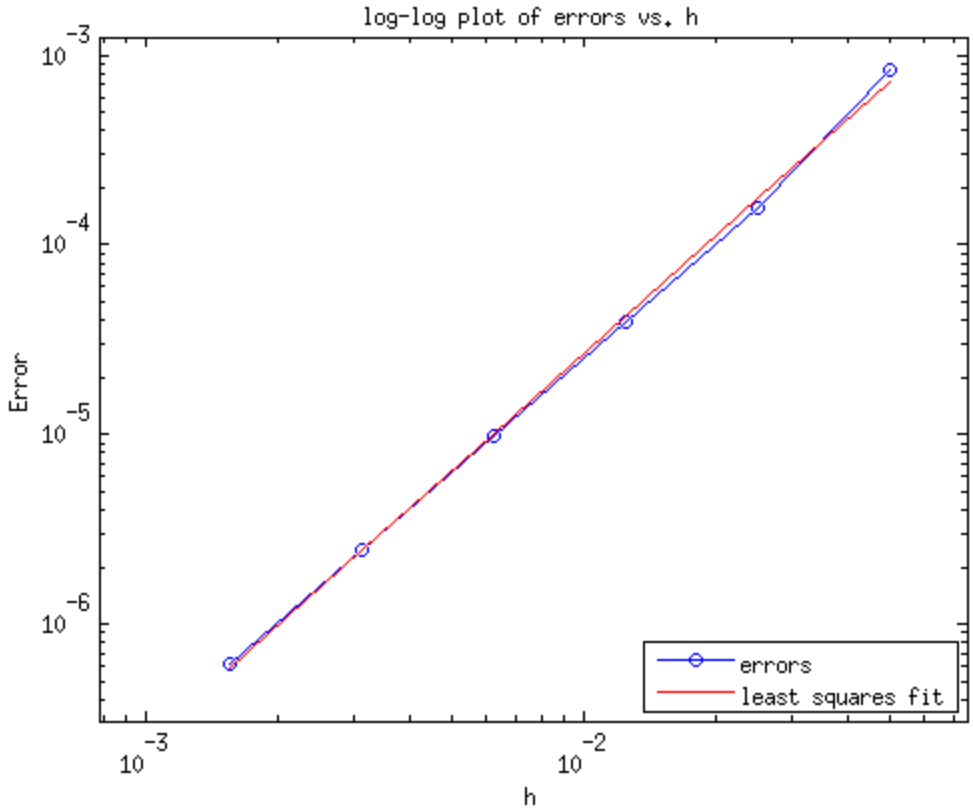
Brian de Silva

Problem 3 - The Heat Equation

Part (a) - Crank Nicolson

```
% Choose a number of trials to run
numTrials = 6;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% Approximate PDE using increasing number of grid points (20, 40, 80, ...)
% Note: we choose k=4h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_CN(20*2^(j-1) - 1);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);

% Computed order of accuracy: 2.06
```

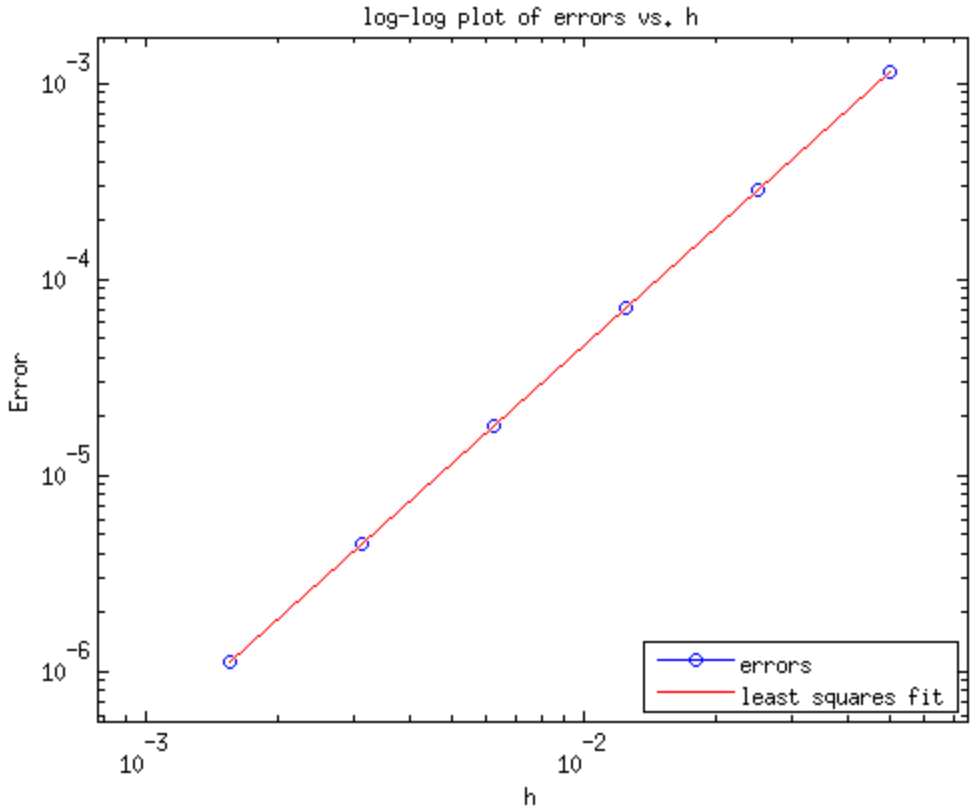
*Least squares fit gives E(h) = 0.344312 * h^2.05585*



Part (b) - TR-BDF2

```
% Choose a number of trials to run
numTrials = 6;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% Approximate PDE using increasing number of grid points (20,40,80,...)
% Note: we choose k=4h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_TRBDF2(20*2^(j-1) - 1);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);
% Computed order of accuracy: 2.00
```

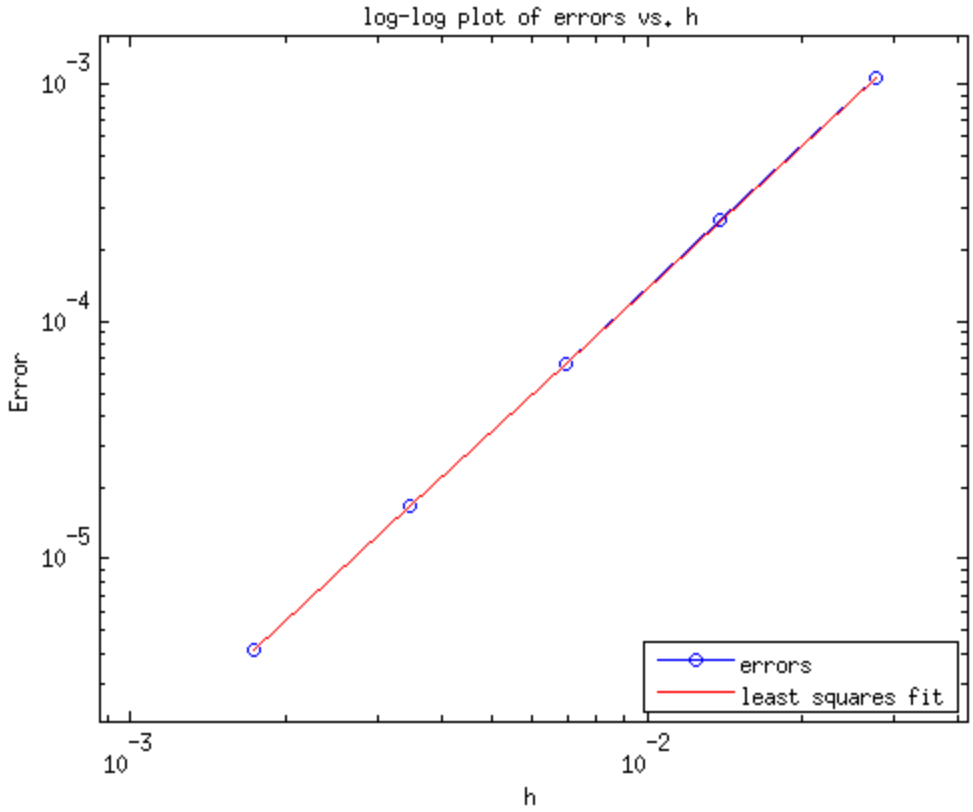
*Least squares fit gives E(h) = 0.455545 * h^1.99841*



Part (c) - Forward Euler

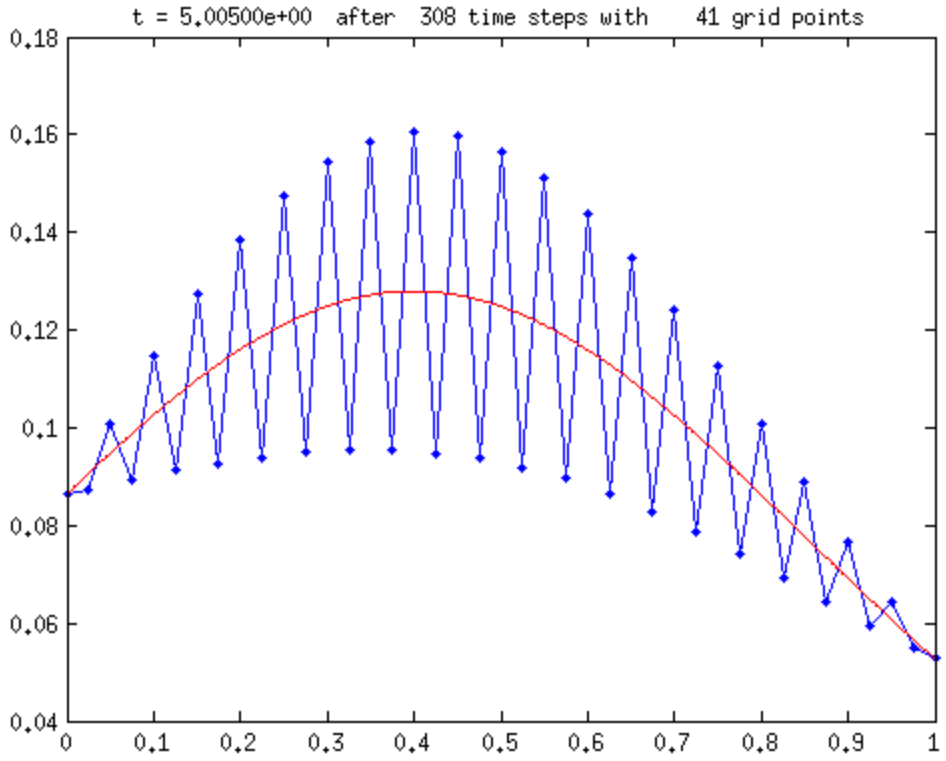
```
% Choose a number of trials to run
numTrials = 5;
testno = 1;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% Approximate PDE using increasing number of grid points (36,72,144, ...)
% Note: we choose k=24h^2 for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_FE(36*2^(j-1) - 1,testno);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);
% Computed order of accuracy: 2.00
```

*Least squares fit gives E(h) = 1.35781 * h^1.99749*



Part (d) - Forward Euler Instability

```
testno = 2;

% Test heat_FE with k = 26h^2 (taking time out to 5)
[~,~,~] = heat_FE(39,testno);
```



Problem 4

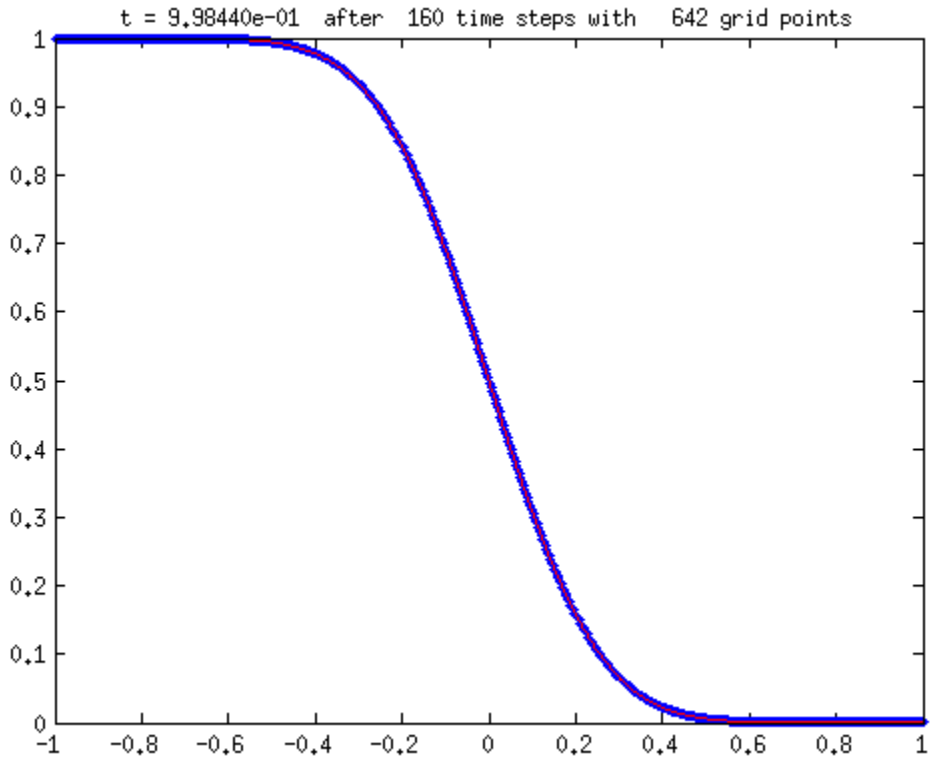
```
% Part (a) (i)
% % Note: we choose k=10h for this test.
alpha = 10;
[~,~,err] = heat_CN_alt_BCs(39,alpha);
disp(err)

% Part (a) (ii)

alpha = 2;
% Choose a number of trials to run
numTrials = 6;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% Approximate PDE using increasing odd numbers of grid points (19,39,79, ...)
% Note: we choose k=2h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_CN_alt_BCs(20*(2^(j-1))-1,alpha);
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);
```

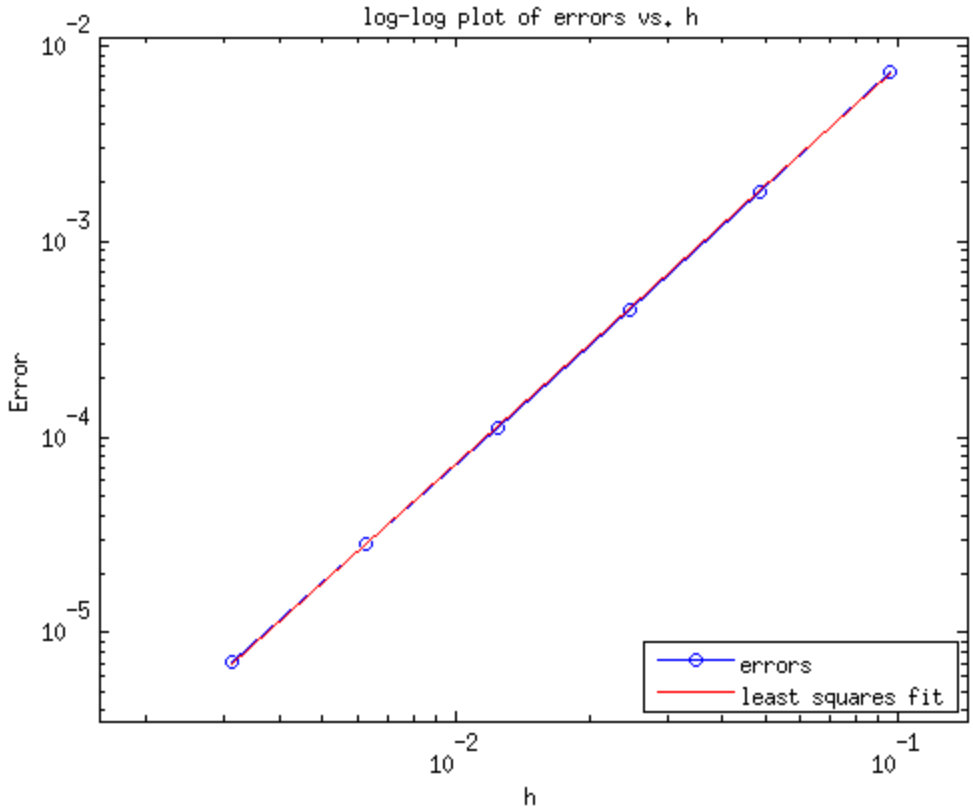
```
% input('Hit <return> to continue  ');

% Approximate PDE using increasing even numbers of grid points (20,40,80, ...)
% Note: we choose k=2h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_CN_alt_BCs(20*(2^(j-1)),alpha);
end

% Create loglog plot of error as grid is refined
figure()
error_loglog(hVec,errorVec);

% Computed orders of accuracy:
% odd: 1.01,   even: 2.03
```

```
    0.2037
```

```
Least squares fit gives E(h) = 1.03148 * h^1.00678
```

```
Least squares fit gives E(h) = 0.845831 * h^2.03017
```

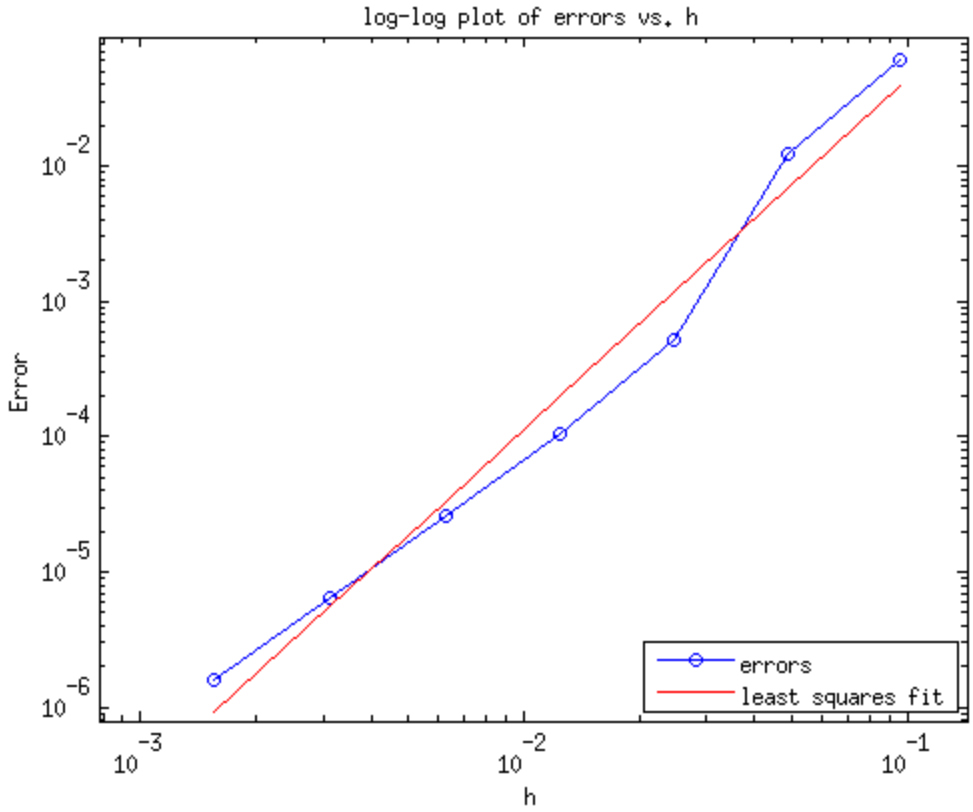




Part (b)

```
% Choose a number of trials to run
numTrials = 7;

hVec = zeros(numTrials,1);
errorVec = zeros(numTrials,1);

% Approximate PDE using increasing even numbers of grid points (32,40,48, ...)
% Note: we choose k=10h for these trials.
for j=1:numTrials
    [hVec(j),~,errorVec(j)] = heat_TRBDF2_alt(20*(2^(j-1)));
end

% Create loglog plot of error as grid is refined
error_loglog(hVec,errorVec);

% Computed order of accuracy: 2.03
```

*Least squares fit gives E(h) = 16.9844 * h^2.58825*



Published with MATLAB® 7.14