

Solutions to Assignment #5

Total of 30 points.

2. (Exercise 2.6 in MORTON & MAYERS.) (10 points)

(i) Suppose b, c are real. Both roots of $z^2 + bz + c = 0$ are inside or on the unit circle if and only if $|c| \leq 1$ and $|b| \leq 1 + c$. *This region is shown shaded in figure 1 below.*

[INSERT FIG HERE]

FIGURE 1. Both roots of $z^2 + bz + c = 0$ are inside the unit circle if and only if (b, c) is a point in this closed triangle. Note the sign of the discriminant $b^2 - 4c$.

Proof. Recall $z_{\pm} = \frac{1}{2}(-b \pm \sqrt{b^2 - 4c})$ are the solutions of $z^2 + bz + c = 0$. Let \mathcal{P} be the proposition “both roots are in or on the unit circle.”

If $b^2 - 4c = 0$ then there is one root and $|z| = |b|/2$, so \mathcal{P} if and only if $|b| \leq 2$. But the portion of the curve $b^2 - 4c = 0$ which satisfies $|b| \leq 2$ is exactly that part inside the triangle.

Suppose $b^2 - 4c < 0$. Note this implies $c > 0$ in particular. There are two complex roots and

$$|z_{\pm}|^2 = \frac{b^2}{4} + \frac{4c - b^2}{4} = c.$$

Thus \mathcal{P} if and only if $c < 1$, in this case. The region where $c < 1$ and $b^2 - 4c < 0$ is part of the shaded triangle as shown in figure 1.

Suppose $b^2 - 4c > 0$. There are distinct real roots with magnitudes

$$|z_{\pm}| = \frac{|\sqrt{b^2 - 4c} \mp b|}{2}.$$

In the case $b \geq 0$, the root with larger magnitude is z_- . Thus, in this case, \mathcal{P} if and only if

$$\sqrt{b^2 - 4c} + b \leq 2 \quad \text{or} \quad b^2 - 4c \leq (2 - b)^2 \quad \text{or} \quad -4c \leq 4 - 4b \quad \text{or} \quad c \geq b - 1.$$

In case $b \leq 0$, the root with larger magnitude is z_+ . Thus \mathcal{P} if and only if

$$\sqrt{b^2 - 4c} - b \leq 2 \quad \text{or} \quad b^2 - 4c \leq (2 + b)^2 \quad \text{or} \quad -4c \leq 4 + 4b \quad \text{or} \quad c \geq -b - 1.$$

These conditions correspond to the lines which form the left and right sides of the triangle, as shown. \square

(ii) Substituting $U_j^n = \lambda^n e^{ik(j\Delta x)}$ into the scheme and then dividing by U_j^n gives:

$$\begin{aligned} \lambda - \lambda^{-1} &= \frac{1}{3}\mu \left\{ \lambda e^{ik\Delta x} - 2\lambda + \lambda e^{-ik\Delta x} + e^{ik\Delta x} - 2 + e^{-ik\Delta x} + \lambda^{-1} e^{ik\Delta x} - 2\lambda^{-1} + \lambda^{-1} e^{-ik\Delta x} \right\} \\ &= \frac{\mu}{3}\lambda \left(-4 \sin^2\left(\frac{1}{2}k\Delta x\right) \right) + \frac{\mu}{3} \left(-4 \sin^2\left(\frac{1}{2}k\Delta x\right) \right) + \frac{\mu}{3}\lambda^{-1} \left(-4 \sin^2\left(\frac{1}{2}k\Delta x\right) \right). \end{aligned}$$

We multiply this by 3λ and recognize it as a quadratic equation in λ . Let $S = \sin(\frac{1}{2}k\Delta x)$. Then

$$(3 + 4\mu S^2) \lambda^2 + (4\mu S^2) \lambda + (-3 + 4\mu S^2) = 0,$$

or

$$\lambda^2 + \frac{4\mu S^2}{3 + 4\mu S^2} \lambda + \frac{-3 + 4\mu S^2}{3 + 4\mu S^2} = 0$$

in “ $z^2 + bz + c = 0$ ” form; $b := 4\mu S^2/(3 + 4\mu S^2)$ and $c := (-3 + 4\mu S^2)/(3 + 4\mu S^2)$.

Note immediately that $0 \leq b \leq 1$ and that $|c| \leq 1$. But also

$$b - 1 = \frac{4\mu S^2 - (3 + 4\mu S^2)}{3 + 4\mu S^2} = \frac{-3}{3 + 4\mu S^2} \leq \frac{-3 + 4\mu S^2}{3 + 4\mu S^2} = c,$$

so (b, c) is inside the triangle and $|\lambda| \leq 1$. Since we have imposed no conditions on μ in order to get this result, we have *unconditional stability*.

(iii) Following a very similar procedure, we find the following equivalent equations:

$$\begin{aligned} \lambda - \lambda^{-1} &= \frac{1}{6}\mu \left\{ \lambda e^{ik\Delta x} - 2\lambda + \lambda e^{-ik\Delta x} + 4(e^{ik\Delta x} - 2 + e^{-ik\Delta x}) + \lambda^{-1} e^{ik\Delta x} - 2\lambda^{-1} + \lambda^{-1} e^{-ik\Delta x} \right\}, \\ (6 + 4\mu S^2) \lambda^2 + (16\mu S^2) \lambda + (-6 + 4\mu S^2) &= 0, \\ \lambda^2 + \frac{8\mu S^2}{3 + 2\mu S^2} \lambda + \frac{-3 + 2\mu S^2}{3 + 2\mu S^2} &= 0, \end{aligned}$$

where $S = \sin(\frac{1}{2}k\Delta x)$. Again, “ $z^2 + bz + c = 0$ ” with $b := 8\mu S^2/(3 + 2\mu S^2)$ and $c := (-3 + 2\mu S^2)/(3 + 2\mu S^2)$.

Here $b \geq 0$ and $|c| \leq 1$. To have both roots inside the unit circle we require $c \geq b - 1$, that is, “ $-3 + 2\mu S^2 \geq 8\mu S^2 - (3 + 2\mu S^2)$ ” or “ $2 \geq 6$,” which is not true(!). Therefore, by part (i), there is λ solving the above equations with $|\lambda| > 1$, and thus this method is unconditionally *unstable*.

A moral of this example is that two reasonable-looking and very similar schemes can be wildly different when used. (Think about poor L. F. Richardson.)

3. (8 points) Suppose $u(x, t)$ is the exact solution. As stated in the hint, but using one lower order than suggested in the hint, I apply Taylor's theorem to get an expression for

$$p(x + \epsilon) [u(x + \Delta, t) - u(x, t)].$$

Substituting into this expression successively $\epsilon = \Delta x/2$ and $\Delta = \Delta x$, and then $\epsilon = -\Delta x/2$ and $\Delta = -\Delta x$, I get:

$$\begin{aligned} & p(x + \Delta x/2) [u(x + \Delta x, t) - u(x, t)] - p(x - \Delta x/2) [u(x, t) - u(x - \Delta x, t)] \\ &= (p(x) + p'(\xi_1)\Delta x/2) \left[u_x(x, t)\Delta x + \frac{1}{2}u_{xx}(x, t)\Delta x^2 + \frac{1}{6}u_{xxx}(\nu_1, t)\Delta x^3 \right] \\ &\quad - (p(x) - p'(\xi_2)\Delta x/2) \left[u_x(x, t)\Delta x - \frac{1}{2}u_{xx}(x, t)\Delta x^2 + \frac{1}{6}u_{xxx}(\nu_2, t)\Delta x^3 \right] \\ &= p(x)u_{xx}(x, t)\Delta x^2 + \frac{p'(\xi_1) + p'(\xi_2)}{2}u_x(x, t)\Delta x^2 + O(\Delta x^3). \end{aligned}$$

Here $x < \xi_1, \nu_1 < x + \Delta x/2$ and $x - \Delta x/2 < \xi_2, \nu_2 < x$. The " $O(\Delta x^2)$ " term is complicated but the details do *not* matter.

In fact, dividing the above by Δx^2 , and including the u_t expression, we see that the truncation error satisfies

$$T(x, t) = u_t(x, t) + \frac{1}{2}u_{tt}(x, \tau)\Delta t - p(x)u_{xx}(x, t) - \frac{p'(\xi_1) + p'(\xi_2)}{2}u_x(x, t) + O(\Delta x)$$

where $t < \tau < t + \Delta t$. Now, the limit of the p' term does exist because $\lim_{\Delta x \rightarrow 0} \xi_1 = \lim_{\Delta x \rightarrow 0} \xi_2 = x$, and because we assume $p'(x)$ is continuous. That is,

$$\lim_{\Delta x \rightarrow 0} \frac{p'(\xi_1) + p'(\xi_2)}{2} = \frac{p'(x) + p'(x)}{2} = p'(x).$$

Therefore

$$\begin{aligned} \lim_{\Delta t \rightarrow 0, \Delta x \rightarrow 0} T(x, t) &= u_t(x, t) + 0 - p(x)u_{xx}(x, t) - \left[\lim_{\Delta x \rightarrow 0} \frac{p'(\xi_1) + p'(\xi_2)}{2} \right] u_x(x, t) + 0 \\ &= u_t(x, t) - p(x)u_{xx}(x, t) - p'(x)u_x(x, t) \\ &= u_t(x, t) - \frac{\partial}{\partial x} \left(p(x) \frac{\partial u}{\partial x}(x, t) \right) = 0, \end{aligned}$$

and *the method is consistent.*

Note: In my hint on the problem statement, and above, I unnecessarily suggested using Taylor's theorem out to the Δx^4 term for $u(x + \Delta x, t) - u(x, t)$. I should have stopped at the Δx^3 or even the Δx^2 term. All of these ways are correct. The moral is that if one wants the clearest formula for the truncation error then one needs to stop at just the right term, but for mere consistency many choices suffice.

4. (12 points) (a) The Crank-Nicolson method for this problem is balanced around the “★” point $(x_j, t_n + \Delta t/2)$:

$$(1) \quad \frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{1}{2} \left(\frac{(1 + 2x_j)}{\Delta x^2} (U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}) + 3U_j^{n+1} \right) + \frac{1}{2} \left(\frac{(1 + 2x_j)}{\Delta x^2} (U_{j-1}^n - 2U_j^n + U_{j+1}^n) + 3U_j^n \right) + f(x_j, t_n + \Delta t/2).$$

Note that the reaction term “ $3u$ ” in the PDE is approximated in a balanced way around the “★” point, too. Without a balanced approximation of that term, you lose the $O(\Delta t^2, \Delta x^2)$ truncation error of the Crank-Nicolson scheme. Here is an additional important point: Equation (1) applies for $j = 0, 1, 2, \dots, J - 1$, because of the left Neumann boundary condition $u_x = 0$ and its (2.114) implementation, which is the next equation:

$$(2) \quad \frac{U_1^n - U_{-1}^n}{2\Delta x} = 0.$$

Note this applies at all time steps (including the one you are denoting “ n ” and the next one “ $n + 1$ ”). Finally note $U_j^n = 0$.

Let

$$\mu_j = \frac{\Delta t}{2\Delta x^2} (1 + 2x_j).$$

Clearing denominators in (1) and (2), we get these equations for the unknowns $U_{-1}^{n+1}, U_0^{n+1}, U_1^{n+1}, \dots, U_{J-1}^{n+1}$:

$$\begin{aligned} -U_{-1}^{n+1} + U_1^{n+1} &= 0, \\ -\mu_j U_{j-1}^{n+1} + (1 + 2\mu_j - (3/2)\Delta t) U_j^{n+1} - \mu_j U_{j+1}^{n+1} &= b_j \quad (j = 0, 1, 2, \dots, J - 1), \end{aligned}$$

where

$$b_j = +\mu_j U_{j-1}^n + (1 - 2\mu_j + (3/2)\Delta t) U_j^n - \mu_j U_{j+1}^n + \Delta t f(x_j, t_n + \Delta t/2),$$

$j = 0, 1, 2, \dots, J - 1$, are the known “right-hand sides”. Typeset as a matrix equation, this is

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 0 & \dots & 0 \\ -\mu_0 & (1 + 2\mu_0 - \frac{3}{2}\Delta t) & -\mu_0 & 0 & 0 & & 0 \\ 0 & -\mu_1 & (1 + 2\mu_1 - \frac{3}{2}\Delta t) & -\mu_1 & 0 & & 0 \\ 0 & 0 & -\mu_2 & (1 + 2\mu_2 - \frac{3}{2}\Delta t) & -\mu_2 & & 0 \\ & \vdots & & & & \ddots & \\ 0 & 0 & \dots & & -\mu_{J-1} & (1 + 2\mu_{J-1} - \frac{3}{2}\Delta t) & \end{bmatrix} \begin{bmatrix} U_{-1}^{n+1} \\ U_0^{n+1} \\ U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{J-1}^{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{J-1} \end{bmatrix}.$$

Note that this matrix equation is *not* tridiagonal.

Now, the above will work; it corresponds to the first program I wrote and it produced the same result as what follows. We can be a bit simpler and more efficient by recognizing that equation (2) is actually telling us exactly that $U_{-1}^{n+1} = U_1^{n+1}$, and that $U_{-1}^n = U_1^n$ as noted, and thus we can eliminate the artificial $j = -1$ point entirely from consideration. In particular, the above matrix system is equivalent to the smaller system

$$\begin{bmatrix} (1 + 2\mu_0 - \frac{3}{2}\Delta t) & -2\mu_0 & 0 & 0 & 0 & 0 \\ -\mu_1 & (1 + 2\mu_1 - \frac{3}{2}\Delta t) & -\mu_1 & 0 & 0 & 0 \\ 0 & -\mu_2 & (1 + 2\mu_2 - \frac{3}{2}\Delta t) & -\mu_2 & 0 & 0 \\ & \vdots & & \ddots & & \\ 0 & \dots\dots & & & -\mu_{J-1} & (1 + 2\mu_{J-1} - \frac{3}{2}\Delta t) \end{bmatrix} \begin{bmatrix} U_0^{n+1} \\ U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{J-1}^{n+1} \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{J-1} \end{bmatrix}.$$

Note the (1,2) location in the new system. This new system *is* tridiagonal.

I wrote the program a5prob4.m below.

```
function U = a5prob4(J,tf,f,u0);
% A5PROB4 solves the heat equation
% u_t = (1+2x) u_xx + 3 u + f(x,t)
% on the interval 0 < x < 1 with boundary conditions
% u_x(0,t)=0 and u(0,t)=0 and initial condition u(x,0) = u0(x).
% Note f and u0 are function handles; see below.
% Example I: to solve A#5 problem 4a
% >> tf = 1.0, myf = @(x,t) (0), myu0 = @(x) 1-x
% >> U20 = a5prob4(20,tf,myf,myu0);
% >> U200 = a5prob4(200,tf,myf,myu0);
% Example II: to solve A#5 problem 4c, see script a5prob4c;

% ELB 3/9/07

dx = 1/J; x = 0:dx:1;
nu = 1/2; dt = nu * dx;
N = ceil(tf/dt); dt = tf/N; % assure N dt = tf
disp(['J = ' num2str(J) ', N = ' num2str(N) ', dt = ' num2str(dt) ...
      ', mu = dt / dx^2 = ' num2str(dt/(dx*dx)) '])

% build A in linear problem to solve at each time step: A U^{n+1} = b
muj = (dt / (2 * dx * dx)) * (1 + 2 * x);
A = sparse(J,J);
A(1,1:2) = [1 + 2 * muj(1) - (3/2) * dt, -2 * muj(1)]; % insert row 1
for j=2:J
    A(j,j-1) = - muj(j);
    A(j,j) = 1 + 2 * muj(j) - (3/2) * dt;
    if j < J % last row is exception
        A(j,j+1) = - muj(j);
    end
end
end
% full(A), figure, spy(A) % uncomment to see entries and pattern in A

% fill in initial condition
U = zeros(size(x)); % note U(J+2) = 0, which is right boundary condition
for k = 1:J+1, U(k) = u0(x(k)); end
figure, plot(x,U,'g--'), hold on

% run
for n = 0:N-1
    tstar = (n + (1/2)) * dt;
    b = zeros(J,1); % build b = RHS for this time step
    b(1) = (1-2*muj(1)+(3/2)*dt) * U(1) + ...
           2 * muj(1) * U(2) + dt * f(x(1),tstar);
```

```

for j = 2:J
    b(j) = muj(j) * U(j-1) + (1-2*muj(j)+(3/2)*dt) * U(j) + ...
        muj(j) * U(j+1) + dt * f(x(j),tstar);
end
% size(A), size(b), size(U), return % uncomment to check sizes
U(1:J) = (A \ b)'; % do time step
% plot(x,U) % uncomment to see every step
end

plot(x,U,'r'), hold off
xlabel x, ylabel u, title('solid red is u(x,tf); dashed green is u(x,0)')

```

As noted in the comments to this program, one provides as input J , t_f , and the “handles” of two functions which are the nonhomogeneity $f(x, t)$ and the initial value $u_0(x) = u(x, 0)$. These function handles can be generated—under version 7.0 or later of MATLAB, anyway—by defining “anonymous functions”, which really are functions, as follows.

```

>> myu0 = @(x) 1-x
>> myu0(0.2), myu0(1)
ans =
    0.8000
ans =
    0

```

Similarly one defines `myf` for part (a) as the zero function, and then one calls `a5prob4` with $J = 20$ and $J = 200$:

```

>> tf = 1.0, myf = @(x,t) (0), myu0 = @(x) 1-x
>> U20 = a5prob4(20,tf,myf,myu0);
[J = 20, N = 40, dt = 0.025, mu = dt / dx^2 = 10]
>> U200 = a5prob4(200,tf,myf,myu0);
[J = 200, N = 400, dt = 0.0025, mu = dt / dx^2 = 100]
>> max(abs(U200(1:10:201)-U20))
ans =
    0.00373587805031

```

Pictures are produced which show visual agreement between the $J = 20$ and $J = 200$ cases. The last command above actually measures the agreement between the $J = 20$ and $J = 200$ cases by comparing their values at the corresponding grid points; they agree to 2 digits at least. But this certainly does not mean that either answer is correct to 2 digits!

Note that the values of $\mu = \Delta t / \Delta x^2$ in use here are far too big for an explicit method to be stable.

(b) Now, the next question is how close our numerical approximation is to the exact solution. I do not know the exact solution to the problem in part (a). So I create a closely-related PDE and an exact solution to it. That is, I *choose* $u(x, t) = t \cos(\pi x/2)$ to be the solution of

$$u_t = (1 + 2x)u_{xx} + 3u + f(x, t)$$

by computing the $f(x, t)$ which makes it so:

$$\begin{aligned} f(x, t) &= u_t - (1 + 2x)u_{xx} - 3u = \frac{\partial}{\partial t} \left(t \cos \left(\frac{\pi x}{2} \right) \right) - (1 + 2x) \frac{\partial^2}{\partial x^2} \left(t \cos \left(\frac{\pi x}{2} \right) \right) - 3t \cos \left(\frac{\pi x}{2} \right) \\ &= \cos \left(\frac{\pi x}{2} \right) - (1 + 2x)t \left(-\frac{\pi^2}{2^2} \cos \left(\frac{\pi x}{2} \right) \right) - 3t \cos \left(\frac{\pi x}{2} \right) \end{aligned}$$

which gives the formula for $f(x, t)$ stated in the problem, and which shows that this $u(x, t)$ solves the PDE. Note $u(x, 0) = 0$ and that the exact solution satisfies the boundary conditions $u_x(0, t) = 0$ and $u(0, t) = 0$ by design.

(c) The $f(x, t)$ manufactured in part (b) can be written as an anonymous function or as its own m-file. I chose the latter:

```
function f = manf(x,t);
% MANF Compute manufactured nonhomogeneity for problem 4c on A#5
CC = pi^2 / 2^2;
f = cos(pi*x/2) * ( 1 + t * (CC * (1+2*x) - 3) );
```

Now I wrote a script

```
% A5PROB4C Script to use A5PROB4 to solve problem 4c on A#5

tf = 1.0;
manu0 = @(x) 0; manu = @(x,t) t * cos(pi*x/2);
J = [5 10 20 40 80 160 320 640 1280]; % expanded refinement path
err = zeros(size(J));
for m = 1:length(J)
    U = a5prob4(J(m),tf,@manf,manu0);
    x = linspace(0,1,J(m)+1);
    err(m) = max(abs(U - manu(x,tf)));
end

dx=1./J;
c=polyfit(log(dx),log(err),1) % fit line to data
disp(['convergence at rate 0(dx^{', num2str(c(1),7) ', }')])

loglog(dx,err,'o',dx,exp(c(1)*log(dx)+c(2)), '--')
xlabel('\Delta x'), ylabel('error = max |U_j^N - u(x_j,t_f)|')
axis tight, set(gca,'XTick',[1e-3 5e-3 1e-2 5e-2 1e-1])
```

which uses the same function `a5prob4` above, but on the manufactured problem to which we have an exact solution. It goes down the refinement path, a little bit further than specified in the problem, and measures the error as suggested.

The output of this script looks like this:

```
>> a5prob4c
[J = 5, N = 10, dt = 0.1, mu = dt / dx^2 = 2.5]
[J = 10, N = 20, dt = 0.05, mu = dt / dx^2 = 5]
[J = 20, N = 40, dt = 0.025, mu = dt / dx^2 = 10]
[J = 40, N = 80, dt = 0.0125, mu = dt / dx^2 = 20]
[J = 80, N = 160, dt = 0.00625, mu = dt / dx^2 = 40]
[J = 160, N = 320, dt = 0.003125, mu = dt / dx^2 = 80]
```

```

[J = 320, N = 640, dt = 0.0015625, mu = dt / dx^2 = 160]
[J = 640, N = 1280, dt = 0.00078125, mu = dt / dx^2 = 320]
[J = 1280, N = 2560, dt = 0.00039063, mu = dt / dx^2 = 640]
c =
  2.0010  -1.1728
convergence at rate O(dx^{2.001})

```

The last $J = 1280$ calculation takes a noticeable few seconds, but then it is solving 2500 matrix problems of size 1280×1280 , and so would you!

The result is a lot of figures showing nearly identical solutions for the different J values, and then a final figure which shows the error decay as a function of Δx , as in figure 2. My only complaint about this figure is that it is too good to publish; it looks very suspicious!

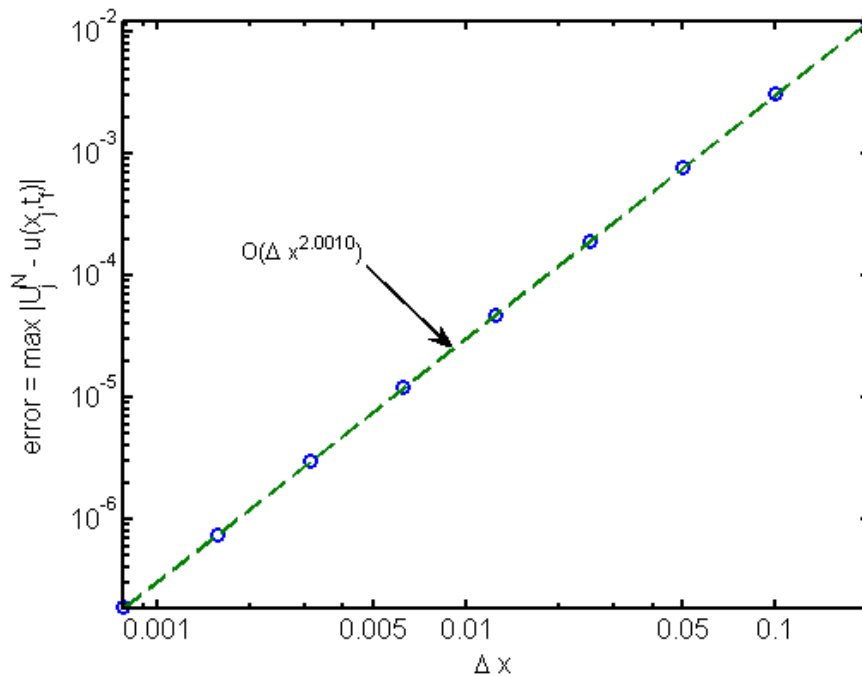


FIGURE 2. We win! The error decays as the best possible power of Δx , that is, the power which appears in the truncation error $T = O(\Delta t^2, \Delta x^2)$.

We have *not* shown, by the computations in part (c), a proof of convergence at the optimal rate. In practice, we have shown something much more useful than that! We have demonstrated that our actual implementation of the numerical method gets all the accuracy it claims to be able to get, at least on a problem close to the one we originally posed. That is, if our goal is a numerical solution of the problem in part (a), then we can probably trust it a lot after doing part (c) *with the same code* `a5prob4`.