AM 255: Final Exam

Douglas Lanman
dlanman@brown.edu
15 December 2006

Problem 1

Part 1: Consider the following initial value problems for the heat equation $u_t = u_{xx}$.

$$\begin{aligned} u_t &= u_{xx}, \quad -\infty < x < \infty, \, 0 \leq t \\ u(x,0) &= u_0(x) = \begin{cases} 1 & \text{if } |x| \leq 1/2 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

$$\begin{aligned} u_t &= u_{xx}, \quad -\infty < x < \infty, \, 0 \leq t \\ u(x,0) &= u_0(x) = \cos(\pi x), \quad -\infty < x < \infty \end{aligned} \tag{2}$$

In both cases we will examine periodic solutions on the interval $[-1, 1]$, evaluated at time $t = 1$. Begin by deriving exact solutions to these initial value problems.

We begin by observing that the initial conditions are periodic on the interval $[-1, 1]$, rather than the canonical interval $[-2\pi, 2\pi]$. Recall from [3] that the Fourier series of a periodic function $f(x)$ on $[-L/2, L/2]$ is given by the following pair of expressions.

$$f(x) = \sum_{\omega=-\infty}^{\infty} e^{i\,(2\pi\omega x/L)} \hat{f}(\omega)$$

$$\hat{f}(\omega) = \frac{1}{L} \int_{-L/2}^{L/2} e^{-i\,(2\pi\omega x/L)} f(x)\, dx$$

For this problem we can substitute $L = 2$ into the previous expressions to obtain the appropriate Fourier series representation.

$$f(x) = \sum_{\omega=-\infty}^{\infty} e^{i\pi\omega x} \hat{f}(\omega) \tag{3}$$

$$\hat{f}(\omega) = \frac{1}{2} \int_{-1}^{1} e^{-i\pi\omega x} f(x)\, dx \tag{4}$$

At this point we can proceed in the manner outlined on page 61 in [1] and assume a solution to the heat equation consisting of a single Fourier component.

$$u(x,t) = e^{i\pi\omega x} \hat{u}(\omega, t)$$

Substituting this expression into $u_t = u_{xx}$ yields the following ordinary differential equation

$$\hat{u}_t(\omega, t) = -(\pi\omega)^2 \hat{u}(\omega, t), \quad \hat{u}(\omega, 0) = \hat{u}_0(\omega),$$

which has the general solution

$$\hat{u}(\omega,t) = e^{-(\pi\omega)^2 t}\hat{u}_0(\omega) \quad \Rightarrow \quad u(x,t) = \sum_{\omega=-\infty}^{\infty} e^{i\pi\omega x} e^{-(\pi\omega)^2 t}\hat{u}_0(\omega). \tag{5}$$

Substituting into Equation 4, the Fourier coefficients of the initial condition in Equation 1 are given by

$$\hat{u}_0(\omega) = \frac{1}{2}\int_{-1/2}^{1/2} e^{-i\pi\omega x}\,dx = \begin{cases} \frac{\sin\left(\frac{\pi\omega}{2}\right)}{\pi\omega} & \text{if } \omega \neq 0 \\ \frac{1}{2} & \text{if } \omega = 0, \end{cases}$$

since $u_0(x)$ is non-zero only on the interval $[-1/2, 1/2]$. Applying this result to Equation 5 gives the exact solution for Equation 1 as an infinite series.

$$u(x,t) = \sum_{\omega=-\infty}^{\infty} e^{i\pi\omega x} e^{-(\pi\omega)^2 t}\hat{u}_0(\omega) = \frac{1}{2} + \sum_{\omega=1}^{\infty} e^{-(\pi\omega)^2 t}\left(\frac{\sin\left(\frac{\pi\omega}{2}\right)}{\frac{\pi\omega}{2}}\right)\left(\frac{e^{i\pi\omega x} + e^{-i\pi\omega x}}{2}\right)$$

$$\Rightarrow \boxed{u(x,t) = \frac{1}{2} + \sum_{\omega=1}^{\infty} e^{-(\pi\omega)^2 t}\left(\frac{\sin\left(\frac{\pi\omega}{2}\right)}{\frac{\pi\omega}{2}}\right)\cos(\pi\omega x)} \tag{6}$$

Similarly, the Fourier coefficients of the initial condition in Equation 2 can be obtained by direct inspection.

$$u_0(x) = \cos(\pi x) = \frac{e^{i\pi x} + e^{-i\pi x}}{2} = \sum_{\omega=-\infty}^{\infty} e^{i\pi\omega x}\hat{f}(\omega)$$

$$\Rightarrow \hat{u}_0(\omega) = \begin{cases} 1/2 & \text{if } \omega \in \{-1, 1\} \\ 0 & \text{otherwise} \end{cases}$$

Substituting these coefficients into Equation 5 gives the exact solution for Equation 2.

$$u(x,t) = \sum_{\omega=-\infty}^{\infty} e^{i\pi\omega x} e^{-(\pi\omega)^2 t}\hat{u}_0(\omega) = e^{-\pi^2 t}\left(\frac{e^{i\pi x} + e^{-i\pi x}}{2}\right)$$

$$\Rightarrow \boxed{u(x,t) = e^{-\pi^2 t}\cos(\pi x)} \tag{7}$$

This completes the derivation of the exact solutions. Note that Equations 6 and 7 will be used in the following parts to estimate the numerical order of accuracy of various finite difference schemes for the heat equation.

Part 2: Evaluate the following discrete difference approximation to Equations 1 and 2

$$v_j^{n+1} = v_j^n + \frac{k}{h^2}\left(v_{j+1}^n - 2v_j^n + v_{j-1}^n\right), \tag{8}$$

consisting of a second-order central difference in space and an explicit forward Euler scheme in time. Consider grid spacings given by $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}, \frac{2}{321}\}$ and time steps given by $\mu = \frac{k}{h^2} = 0.4$. Report L_2-errors and the numerical order of accuracy for all parameter sets.

Recall from page 63 in [1] that the scheme in Equation 8 is called the *Euler method*; in terms of the discrete difference operators, we can express this approximation in the following form.

$$v_j^{n+1} = (I + kD_+D_-)v_j^n \tag{9}$$

My implementation of this discrete difference approximation was completed using *Matlab* and is included as `EulerMethod.m`. Before presenting the results of my program, I will briefly outline the architecture of the source code. On lines 13-65 I select the initial condition, the values of $\{N, h, k\}$, and determine the resulting grid points $\{x, t\}$. (Note that on lines 55-57 I ensure that the last time is given by $t = 1$.) Lines 71-83 implement Equation 9. Note that I have implemented the D_+D_- operator as a stand-alone program `DpDm.m`. Lines 89-172 create the tables and plots shown in this write-up. Finally, note that the truncated infinite series approximating the exact solution to Equation 1 is defined on lines 175-182.

The approximation results for $k = 0.4h^2$ are tabulated below; Table 1.1 shows the results for the "boxcar" initial condition in Equation 1, whereas Table 1.2 shows the results for the cosine initial condition in Equation 2. Corresponding plots are shown in Figures 1 and 2.

h	N	L_2-error	order
2/21	20	3.275e-6	NA
2/41	40	8.822e-7	1.89
2/81	80	2.276e-7	1.95
2/161	160	5.771e-8	1.98
2/321	320	1.452e-8	1.99

Table 1.1: Results for Equation 1

h	N	L_2-error	order
2/21	20	5.145e-6	NA
2/41	40	1.386e-6	1.89
2/81	80	3.575e-7	1.95
2/161	160	9.065e-8	1.98
2/321	320	2.282e-8	1.99

Table 1.2: Results for Equation 2

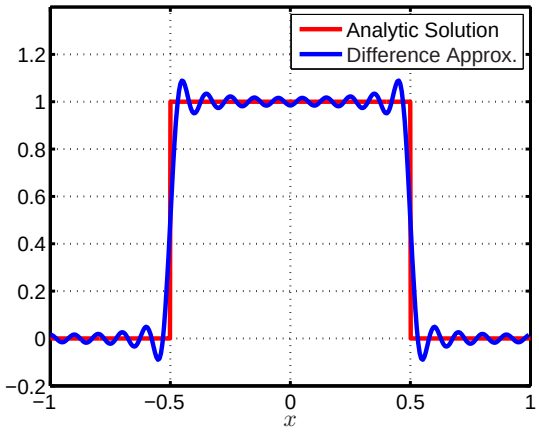
Note that the standard definition of the discrete L_2-norm was used to evaluate the total error as

$$L_2\text{-error}(N) \triangleq \sqrt{\sum_{j=0}^{N} |u(x_j, t^n) - v_j^n|^2 h}.$$

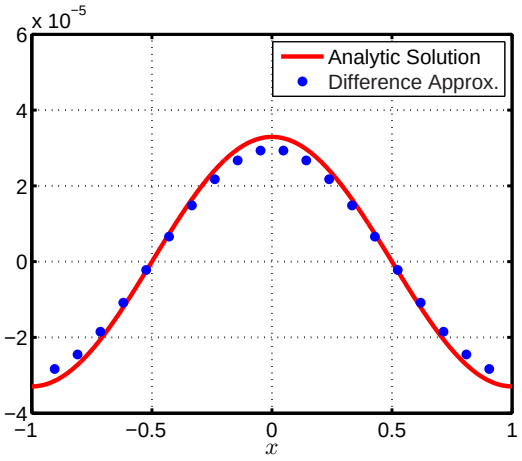
In addition, recall that the following definition of order of approximation was given in class.

$$\text{order} \triangleq \log_2\left(\frac{L_2\text{-error}(N)}{L_2\text{-error}(2N)}\right)$$

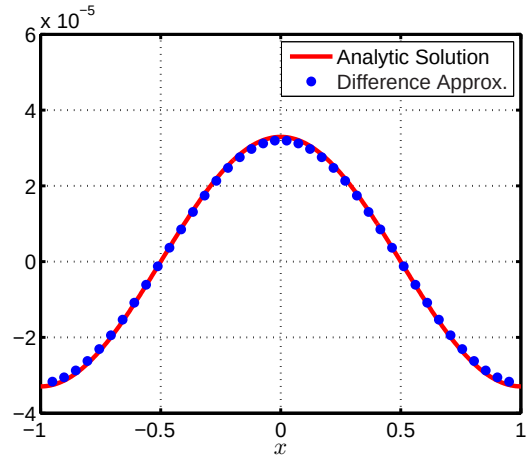
From these results, we find that the Euler method for the heat equation (as defined in Equation 9) is stable and has second-order numerical accuracy for both initial conditions. Recall from page 63 in [1] that the Euler method is stable for $k/h^2 \leq 1/2$. Since we used $k = 0.4h^2$, we find that these results are consistent with theoretical predictions. (Note that a discussion of special considerations for "boxcar" initial conditions is delayed until Part 3.)



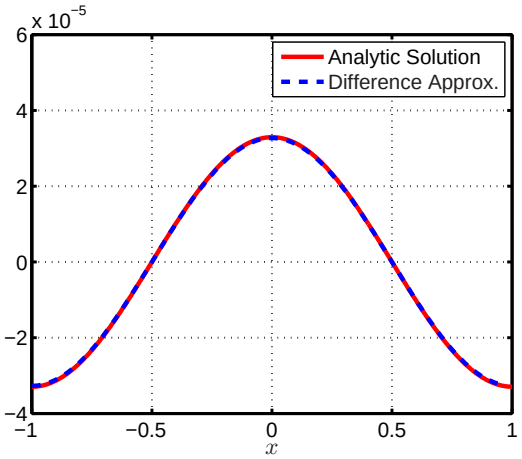
(a) initial condition



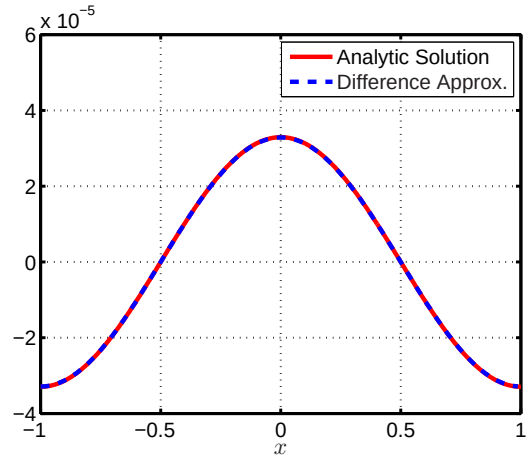
(b) $h = 2/21,\ k = 0.4h^2$



(c) $h = 2/41,\ k = 0.4h^2$

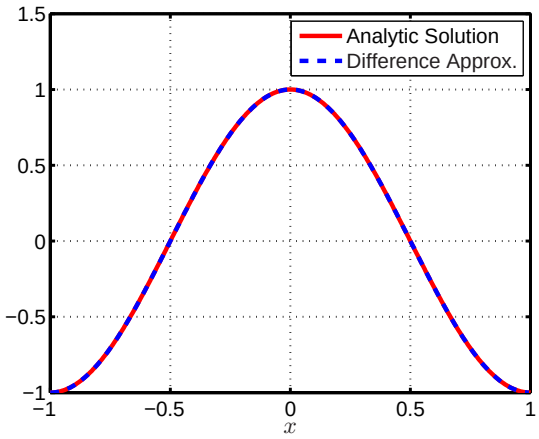


(d) $h = 2/81,\ k = 0.4h^2$

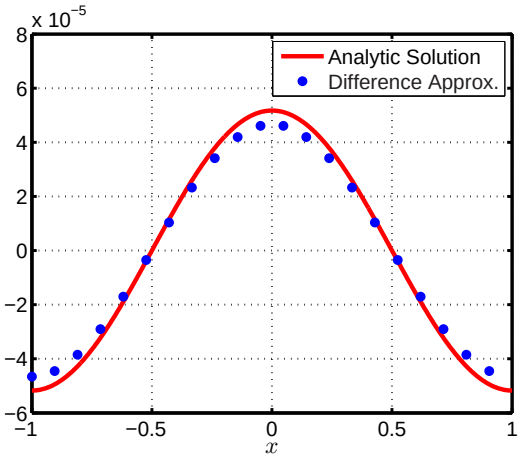


(e) $h = 2/161,\ k = 0.4h^2$

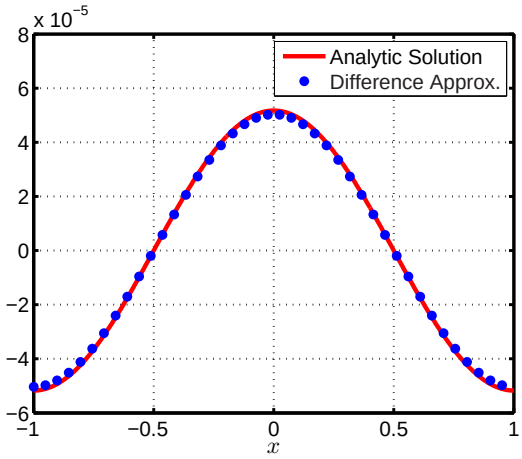
Figure 1: Approximation results corresponding to Table 1.1. (a) Comparison between exact initial condition and Fourier series approximation using 21 terms. (b)-(e) Comparison between Euler method and the analytic solution of Equation 1 at $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = 0.4h^2$. (Plots (b)-(e) show residuals after subtracting 1/2 from each solution.)



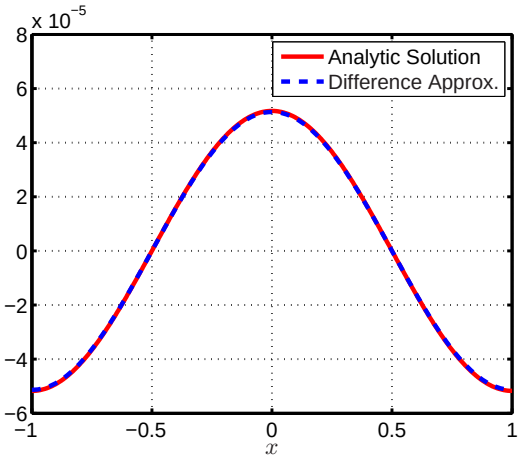
(a) initial condition



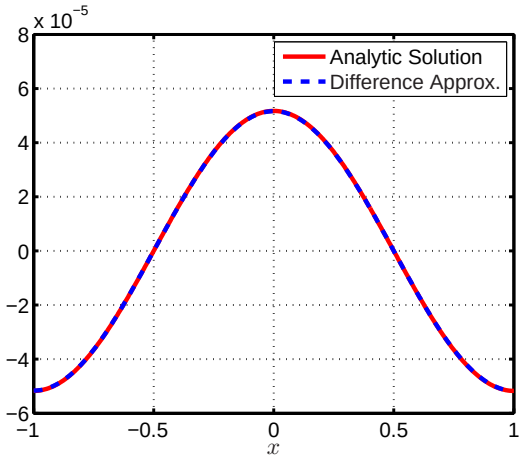
(b) $h = 2/21,\ k = 0.4h^2$



(c) $h = 2/41,\ k = 0.4h^2$



(d) $h = 2/81,\ k = 0.4h^2$



(e) $h = 2/161,\ k = 0.4h^2$

Figure 2: Approximation results corresponding to Table 1.2. (a) Plot of exact initial condition. (b)-(e) Comparison between the Euler method and the analytic solution of Equation 2 at time $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = 0.4h^2$.

Part 3: Evaluate the second-order Crank-Nicholson approximation to Equations 1 and 2

$$v_j^{n+1} = v_j^n + \frac{k}{2h^2}\left(v_{j+1}^{n+1} - 2v_j^{n+1} + v_{j-1}^{n+1} + v_{j+1}^n - 2v_j^n + v_{j-1}^n\right). \tag{10}$$

Consider grid spacings given by $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}, \frac{2}{321}\}$ and time steps given by $\mu = \frac{k}{h^2} = 10$ and $\lambda = \frac{k}{h} = 1$. Report L_2-errors and the numerical order of accuracy for all parameter sets.

Recall, from page 68 in [1], that the Crank-Nicholson scheme in Equation 10 can be written using the discrete difference operators as follows.

$$\left(I - \frac{k}{2}D_+D_-\right)v_j^{n+1} = \left(I + \frac{k}{2}D_+D_-\right)v_j^n \tag{11}$$

My implementation of this discrete difference approximation was completed using *Matlab* and is included as `CrankNicholson.m`. Before presenting the results of my program, I will briefly outline the architecture of the source code. On lines 13-70 I select the initial condition, the values of $\{N, h, k\}$, and determine the resulting grid points $\{x, t\}$. (Note that on lines 60-62 I ensure that the last time is given by $t = 1$.) Lines 73-103 implement Equation 11. Note that I have implemented the D_+D_- operator, in matrix form, on lines 79-87. Lines 106-192 create the tables and plots shown in this write-up. Finally, note that the truncated infinite series approximating the exact solution to Equation 1 is defined on lines 195-202.

The approximation results for $k = 10h^2$ are tabulated below; Table 1.3 shows the results for the "boxcar" initial condition in Equation 1, whereas Table 1.4 shows the results for the cosine initial condition in Equation 2. Corresponding plots are shown in Figures 3 and 4.

h	N	L_2-error	order
2/21	20	1.437e-2	NA
2/41	40	1.627e-6	13.11
2/81	80	6.504e-8	4.64
2/161	160	3.498e-8	0.89
2/321	320	9.980e-9	1.81
2/641	640	2.577e-9	1.95

Table 1.3: Equation 1 with $k = 10h^2$

h	N	L_2-error	order
2/21	20	2.495e-5	NA
2/41	40	1.335e-6	4.22
2/81	80	1.022e-7	3.71
2/161	160	5.495e-8	0.89
2/321	320	1.568e-8	1.81
2/641	640	4.048e-9	1.95

Table 1.4: Equation 2 with $k = 10h^2$

The approximation results for $k = h$ are tabulated below; Table 1.5 shows the results for the "boxcar" initial condition in Equation 1, whereas Table 1.6 shows the results for the cosine initial condition in Equation 2. Corresponding plots are shown in Figures 5 and 6.

h	N	L_2-error	order
2/21	20	3.088e-2	NA
2/41	40	5.185e-3	2.57
2/81	80	9.719e-6	9.06
2/161	160	3.627e-7	4.74
2/321	320	9.171e-8	1.98
2/641	640	2.305e-8	1.99

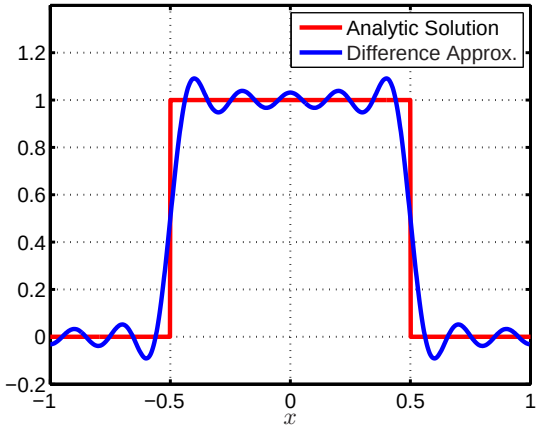
Table 1.5: Equation 1 with $k = h$

h	N	L_2-error	order
2/21	20	2.647e-5	NA
2/41	40	8.236e-6	1.68
2/81	80	2.217e-6	1.89
2/161	160	5.697e-7	1.96
2/321	320	1.441e-7	1.98
2/641	640	3.620e-8	1.99

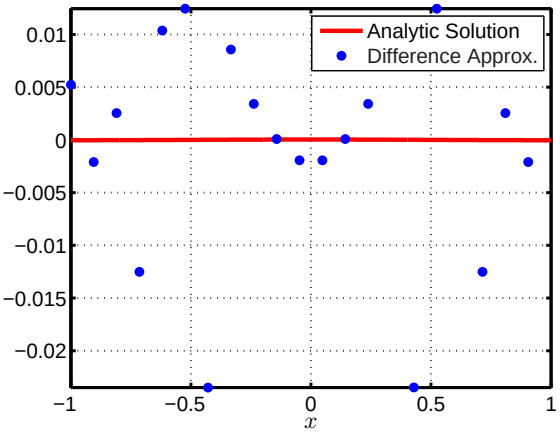
Table 1.6: Equation 2 with $k = h$

From these results, we find that the Crank-Nicholson approximation for the heat equation (as defined in Equation 11) is stable and has second-order numerical accuracy for both initial conditions. Recall from pages 68 and 69 in [1] that the Crank-Nicholson scheme is unconditionally stable. As a result, the experimental results confirm the theoretical stability analysis. Also recall that the Crank-Nicholson scheme was proven to be accurate to $\mathcal{O}(h^2+k^2)$ in Problem 3 of Problem Set 3. Once again, the numerical results confirm the theoretical prediction of second-order accuracy.

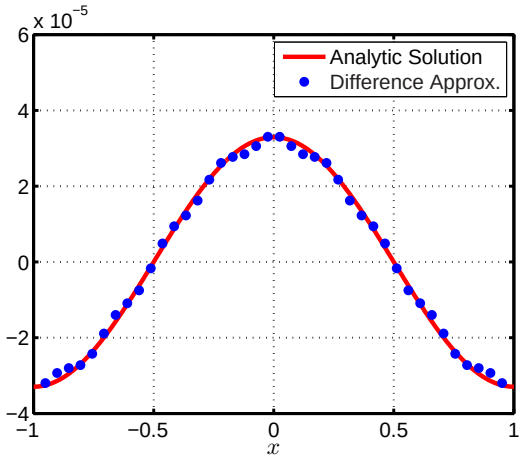
To conclude my analysis I would like to mention a particular detail of my implementation which was required to obtain second-order accuracy for the "boxcar" initial condition in Equation 1. Specifically, I found that this initial condition could not be represented directly as a discontinuous "boxcar" function. Instead, I used a truncated Fourier series to synthesize the initial condition (as shown in Figures 1(a), 3(a) and 1(d)). Experimentally, I found that allowing between 11 to 21 terms in the Fourier series expansion led to second-order accuracy in Parts 2 and 3. As discussed on page 62 in [1], the heat equation is a parabolic equation and, as a result, each Fourier component is damped with time; this damping is particularly strong for high frequencies. As a result, the higher-order terms in the Fourier series do not significantly affect the solution at time $t = 1$ and can be removed from the initial condition. As I found experimentally, this improved the stability of the solution and also led to second-order accuracy. While outside the scope of this course, I believe that this effect can be explained by the Nyquist-Shannon sampling theorem [2]. Specifically, unless we truncate the higher-order terms, the "boxcar" function will be undersampled for small grid-spacings (e.g., for $N < 320$ in Tables 1.3-1.6). In conclusion, we find that the Crank-Nicholson scheme has second-order numerical accuracy if the initial conditions are handled with care; otherwise, we only find first-order accuracy experimentally.



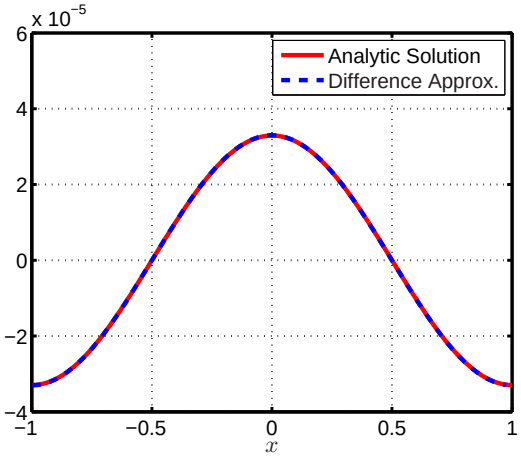
(a) initial condition



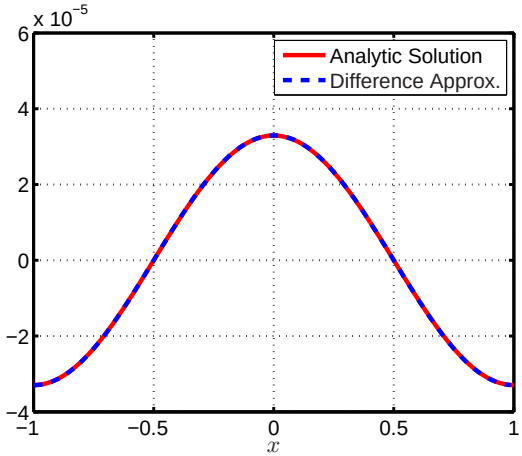
(b) $h = 2/21,\ k = 10h^2$



(c) $h = 2/41,\ k = 10h^2$

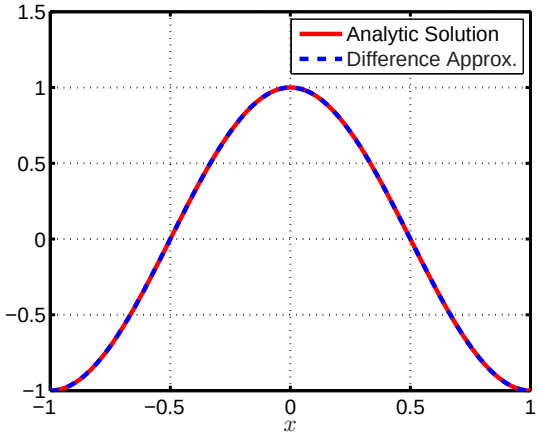


(d) $h = 2/81,\ k = 10h^2$

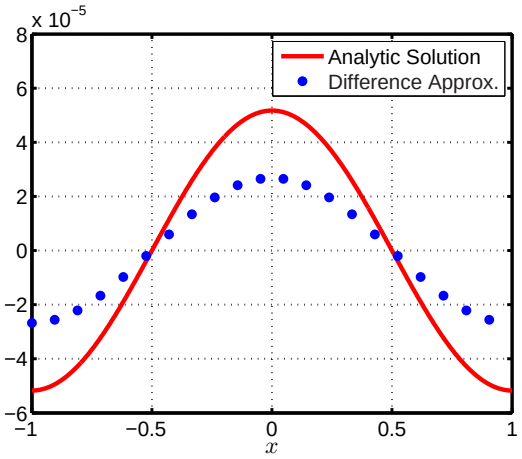


(e) $h = 2/161,\ k = 10h^2$

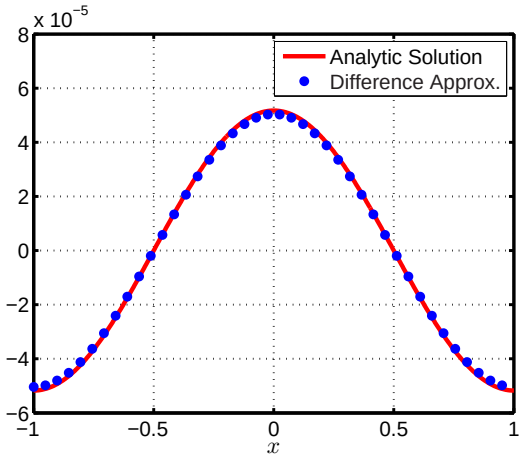
Figure 3: Approximation results corresponding to Table 1.3. (a) Comparison between exact initial condition and the Fourier series approximation using 11 terms. (b)-(e) Comparison between the Crank-Nicholson approximation and the analytic solution of Equation 1 at time $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = 10h^2$. (Note that plots (b)-(e) show residuals after subtracting the mean value of 1/2 from the solutions.)



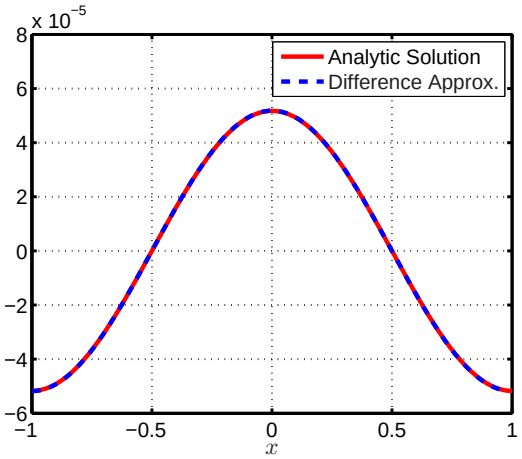
(a) initial condition



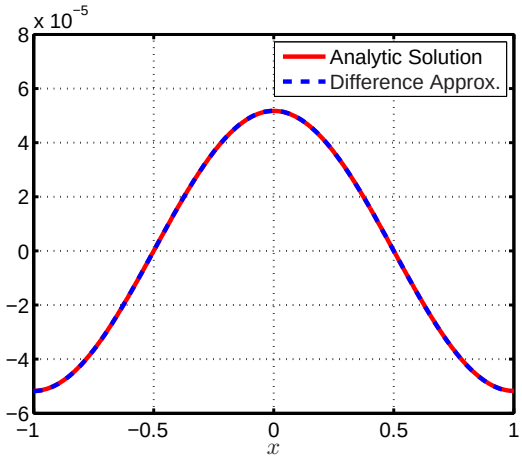
(b) $h = 2/21,\ k = 10h^2$



(c) $h = 2/41,\ k = 10h^2$

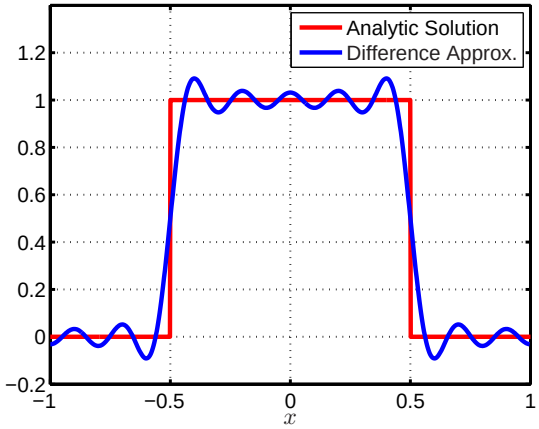


(d) $h = 2/81,\ k = 10h^2$

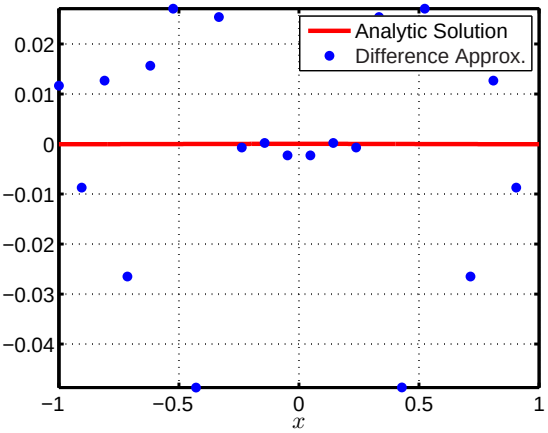


(e) $h = 2/161,\ k = 10h^2$

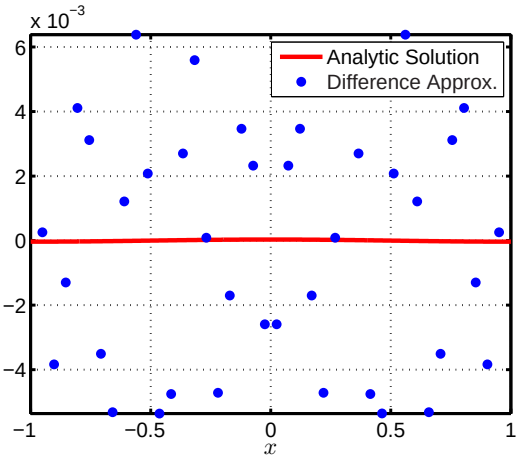
Figure 4: Approximation results corresponding to Table 1.4. (a) Plot of exact initial condition. (b)-(e) Comparison between Crank-Nicholson and the analytic solution of Equation 2 at time $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = 10h^2$.



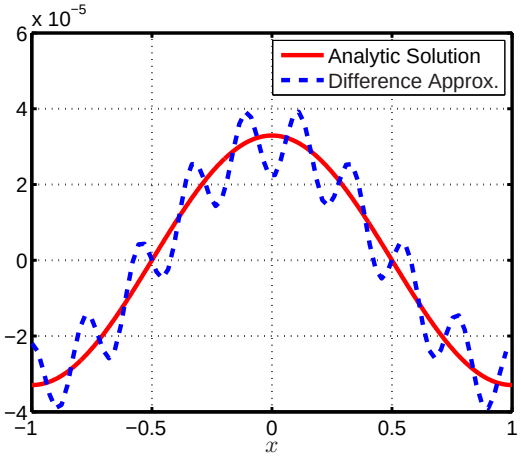
(a) initial condition



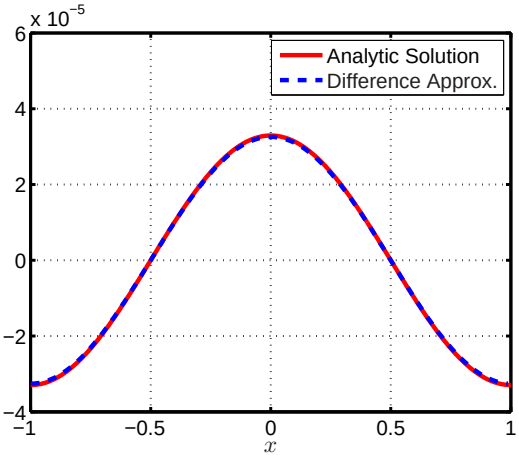
(b) $h = 2/21,\ k = h$



(c) $h = 2/41,\ k = h$

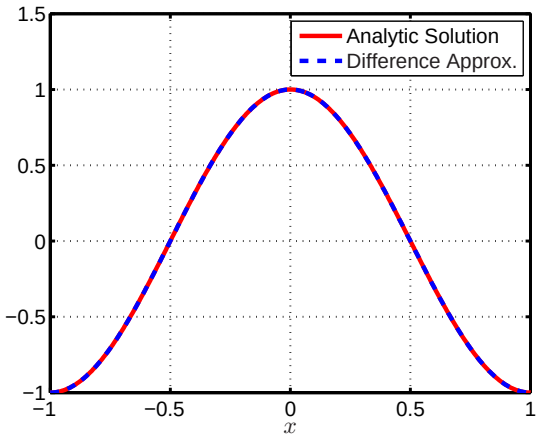


(d) $h = 2/81,\ k = h$

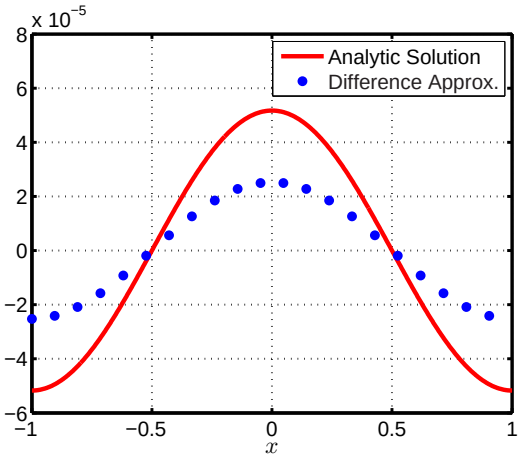


(e) $h = 2/161,\ k = h$

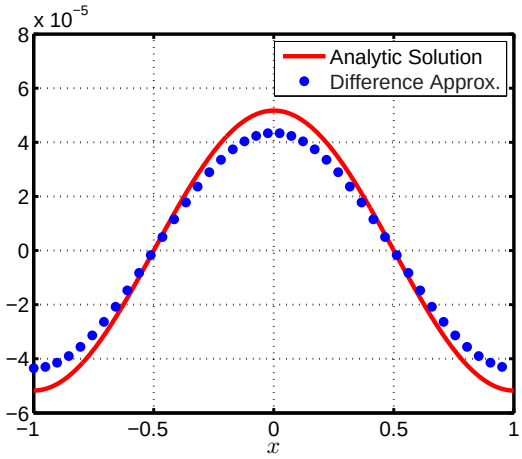
Figure 5: Approximation results corresponding to Table 1.5. (a) Comparison between exact initial condition and the Fourier series approximation using 11 terms. (b)-(e) Comparison between the Crank-Nicholson approximation and the analytic solution of Equation 1 at time $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = h$. (Note that plots (b)-(e) show residuals after subtracting the mean value of 1/2 from the solutions.)



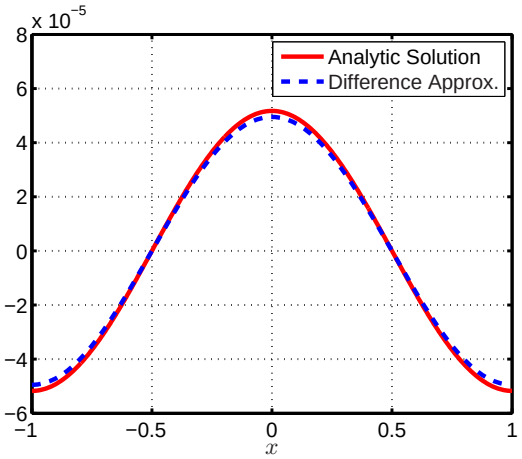
(a) initial condition



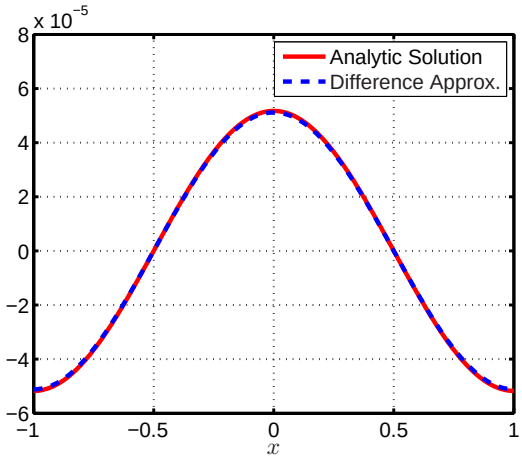
(b) $h = 2/21,\ k = h$



(c) $h = 2/41,\ k = h$



(d) $h = 2/81,\ k = h$



(e) $h = 2/161,\ k = h$

Figure 6: Approximation results corresponding to Table 1.6. (a) Plot of exact initial condition. (b)-(e) Comparison between Crank-Nicholson and the analytic solution of Equation 2 at time $t = 1$, for $h = \{\frac{2}{21}, \frac{2}{41}, \frac{2}{81}, \frac{2}{161}\}$ and $k = h$.

Problem 2

Find the constants $\{a, b, c, d\}$ (which may depend on $\lambda = k/h$) in the explicit scheme

$$v_j^{n+1} = av_{j-1}^n + bv_j^n + cv_{j+1}^n + dv_{j+2}^n \tag{12}$$

to obtain a third-order accurate method for approximating the first-order hyperbolic wave equation $u_t = u_x$. Analyze the stability and dissipativity of your scheme.

Let's begin our analysis by converting Equation 12 to the following "normalized" form (by subtracting v_j^n from both sides and dividing by k).

$$\frac{v_j^{n+1} - v_j^n}{k} = \left(\frac{a}{k}\right) v_{j-1}^n + \left(\frac{b-1}{k}\right) v_j^n + \left(\frac{c}{k}\right) v_{j+1}^n + \left(\frac{d}{k}\right) v_{j+2}^n \tag{13}$$

In order to derive a third-order accurate method for the first-order hyperbolic wave equation, we will assume that u is a smooth function. From pages 59 and 60 in [1], recall the following Taylor series expansion for the left-hand side of Equation 13 (where we have applied the PDE to relate temporal to spatial derivatives).

$$\begin{aligned}\frac{u(x,t+k) - u(x,t)}{k} &= u_t(x,t) + \frac{k}{2}u_{tt}(x,t) + \frac{k^2}{6}u_{ttt}(x,t) + \mathcal{O}(k^3) \\ &= u_x(x,t) + \frac{k}{2}u_{xx}(x,t) + \frac{k^2}{6}u_{xxx}(x,t) + \mathcal{O}(k^3)\end{aligned} \tag{14}$$

In addition, we recall the following Taylor series expansions for the right-hand terms.

$$u(x-h,t) = u(x,t) - hu_x(x,t) + \frac{h^2}{2}u_{xx}(x,t) - \frac{h^3}{6}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{15}$$

$$u(x+h,t) = u(x,t) + hu_x(x,t) + \frac{h^2}{2}u_{xx}(x,t) + \frac{h^3}{6}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{16}$$

$$u(x+2h,t) = u(x,t) + 2hu_x(x,t) + 2h^2u_{xx}(x,t) + \frac{4h^3}{3}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{17}$$

In order to obtain a third-order accurate scheme we observe that the terms in u_x, u_{xx}, and u_{xxx} must cancel between the left-hand and right-hand sides. Substituting Equations 24-17 into Equation 13 gives the following constraints on the coefficients $\{a, b, c, d\}$ (by equating these terms across sides).

$$\begin{aligned} a + b + c + d &= 1 \\ -a + c + 2d &= \lambda \\ a + c + 4d &= \lambda^2 \\ -a + c + 8d &= \lambda^3 \end{aligned}$$

These constraints define a linear system of equations which can be solved to obtain $\{a, b, c, d\}$. Expressed as a matrix-vector product, the constraints have the following form.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 4 \\ -1 & 0 & 1 & 8 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 \\ \lambda \\ \lambda^2 \\ \lambda^3 \end{pmatrix}$$

Inverting this expression yields the following solution for the third-order accurate "upwind" method for approximating the first-order hyperbolic wave equation $u_t = u_x$. (Note that, by construction, the truncation error is given by $\tau_j^n = \mathcal{O}(h^3 + k^3)$ and the resulting scheme is accurate to order (3,3).)

$$\boxed{\begin{aligned} v_j^{n+1} &= a v_{j-1}^n + b v_j^n + c v_{j+1}^n + d v_{j+2}^n \\ a &= -\tfrac{1}{3}\lambda + \tfrac{1}{2}\lambda^2 - \tfrac{1}{6}\lambda^3 \\ b &= 1 - \tfrac{1}{2}\lambda - \lambda^2 + \tfrac{1}{2}\lambda^3 \\ c &= \lambda + \tfrac{1}{2}\lambda^2 - \tfrac{1}{2}\lambda^3 \\ d &= -\tfrac{1}{6}\lambda + \tfrac{1}{6}\lambda^3 \end{aligned}}$$

In order to find the necessary conditions for stability, we begin by making the ansatz

$$v_j^n = \frac{1}{\sqrt{2\pi}} e^{i\omega x_j} \hat{v}^n(\omega),$$

where the solution is composed of a single Fourier component and $x_j = jh$. Substituting this expression into Equation 12 and canceling common terms on both sides, we obtain the following form for the amplification factor $\hat{Q}$.

$$\hat{v}^{n+1}(\omega) = \hat{Q}\hat{v}^n(\omega), \quad \hat{Q} = a e^{-i\omega h} + b + c e^{i\omega h} + d e^{2i\omega h} \tag{18}$$

Recall from page 44 in [1] that we consider a method *stable* if

$$\sup_{0 \le t_n \le T, \omega, k, h} |\hat{Q}^n| \le K(T),$$

as $h, k \to 0$. As was done in the textbook, we can choose k and h such that

$$|\hat{Q}| \le 1 \Rightarrow |\hat{Q}|^2 \le 1.$$

Substituting for the symbol $\hat{Q}$ from Equation 18, we derive the following expression (where $\xi = \omega h$).

$$\begin{aligned} |\hat{Q}|^2 &= \left(a e^{-i\omega h} + b + c e^{i\omega h} + d e^{2i\omega h}\right)\left(a e^{i\omega h} + b + c e^{-i\omega h} + d e^{-2i\omega h}\right) \\ &= (a^2 + b^2 + c^2 + d^2) + 2(ab + bc + cd)\cos(\xi) + 2(ac + bd)\cos(2\xi) + 2ad\cos(3\xi) \end{aligned}$$

At this point we recall the following trigonometric identities from [4].

$$\sin^2\left(\frac{\xi}{2}\right) = \frac{1 - \cos(\xi)}{2} \qquad \cos(3\xi) = 4\cos^3(\xi) - 3\cos(\xi)$$

Applying these identities allows us to simplify the previous expression for $|\hat{Q}|^2$. (For a full explanation, please see that attached *Mathematica* notebook. In addition, note that we have also substituted $a + b + c + d = 1$ to obtain the first term in the expression.)

$$|\hat{Q}|^2 = 1 - 4(ac + bd)\sin^2(\xi) - 4(ab + bc + cd + ad)\sin^2\left(\frac{\xi}{2}\right) - 8ad\cos(\xi)\sin^2\left(\frac{\xi}{2}\right)$$

At this point we can substitute for $\{a, b, c, d\}$ to obtain a closed-form expression for $|\hat{Q}|^2$ in terms of λ. Once again, this involves a fair amount of algebraic manipulation; we quote the final expression here (although a complete derivation is presented in the attached *Mathematica* notebook).

$$|\hat{Q}|^2 = 1 - \alpha(\lambda)\sin^4\left(\frac{\xi}{2}\right) - \beta(\lambda)\cos(\xi)\sin^4\left(\frac{\xi}{2}\right) \tag{19}$$

$$\alpha(\lambda) \triangleq \frac{8\,\lambda}{3} + \frac{4\,\lambda^2}{9} - \frac{16\,\lambda^3}{3} + \frac{4\,\lambda^4}{9} + \frac{8\,\lambda^5}{3} - \frac{8\,\lambda^6}{9}$$
$$\beta(\lambda) \triangleq \frac{-16\,\lambda^2}{9} + \frac{8\,\lambda^3}{3} + \frac{8\,\lambda^4}{9} - \frac{8\,\lambda^5}{3} + \frac{8\,\lambda^6}{9}$$

To derive the stability criterion, we note that the following inequality must hold in order for $|\hat{Q}|^2 \leq 1$.

$$\alpha(\lambda)\sin^4\left(\frac{\xi}{2}\right) + \beta(\lambda)\cos(\xi)\sin^4\left(\frac{\xi}{2}\right) > 0$$

Dividing by the common term in $\sin^4\left(\frac{\xi}{2}\right)$ gives the following inequality.

$$\alpha(\lambda) + \beta(\lambda)\cos(\xi) > 0$$

Since $-1 \leq \cos(\xi) \leq 1$, we note that $\alpha(\lambda) \pm \beta(\lambda) > 0$. This provides the following two stability constraints on λ.

$$\alpha(\lambda) + \beta(\lambda) = \frac{8\lambda}{3} - \frac{4\lambda^2}{3} - \frac{8\lambda^3}{3} + \frac{4\lambda^4}{3} > 0$$
$$\alpha(\lambda) - \beta(\lambda) = \frac{8\lambda}{3} + \frac{20\lambda^2}{9} - 8\lambda^3 - \frac{4\lambda^4}{9} + \frac{16\lambda^5}{3} - \frac{16\lambda^6}{9} > 0$$

Solving these inequalities for λ gives the following stable region for the proposed scheme. (Once again, see the attached *Mathematica* notebook for a complete derivation of the polynomial roots and stability region.)

$$\boxed{0 < \lambda < 1} \tag{20}$$

We conclude our analysis by analyzing the dissipativity of the proposed scheme. Recall from Definition 5.2.1 on page 179 in [1] that an approximation scheme (with symbol $\hat{Q}$) is dissipative of over $2r$ if all the eigenvalues z_ν of $\hat{Q}$ satisfy

$$|z_\nu| \leq (1 - \delta|\xi|^{2r})e^{\alpha_S k}, \qquad |\xi| \leq \pi,$$

where $\delta > 0$ is a positive constant independent of ξ or h. First, we observe that the scheme satisfies the fourth-order dissipativity condition near $\xi = 0$. For small ξ we recall that $|\sin(\xi)| \approx |\xi|$ and $|\cos(\xi)| \approx 1$. As a result, Equation 19 gives the following result.

$$|\hat{Q}|^2 = |z|^2 \approx 1 - \left(\frac{\alpha(\lambda) + \beta(\lambda)}{16}\right)|\xi|^4, \text{ for } |\xi| \text{ near } 0$$

Note that, by the stability condition, $\alpha(\lambda)+\beta(\lambda) > 0$. As a result, Definition 5.2.1 is satisfied and **we conclude that the proposed scheme is dissipative of order 4** for $0 < \lambda < 1$.

Problem 3

Part 1: Discuss the accuracy and stability of the Lax-Wendroff scheme for $u_t = Au_x$, assuming 2π periodicity, for

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Recall from pages 227 and 228 in [1] that the Lax-Wendroff scheme is based on the following Taylor series expansion.

$$u(t_{n+1}) = u(t_n) + ku_t(t_n) + \frac{k^2}{2}u_{tt}(t_n) + \mathcal{O}(k^3)$$

Note that the PDE can be applied to convert temporal derivatives to spatial derivatives as follows.

$$u_{tt} = A(u_x)_t = A(u_t)_x = A(Au_x)_x = A^2u_{xx}$$

Apply this result to the previous Taylor series gives the following expression.

$$u(t_{n+1}) = u(t_n) + kAu_x(t_n) + \frac{k^2}{2}A^2u_{xx}(t_n) + \mathcal{O}(k^3)$$

Using finite differences to approximate terms up to second-order in k yields the following general form for the Lax-Wendroff scheme for $u_t = Au_x$. (Note that this expression is identical to that derived in Problem 1 of Problem Set 8, as well as in class on 12/5/06.)

$$v_j^{n+1} = \left(I + kAD_0 + \frac{k^2}{2}A^2D_+D_-\right)v_j^n \tag{21}$$

At this point we can substitute for A and A^2 to obtain the explicit form of the Lax-Wendroff scheme for this part.

$$v_j^{n+1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} v_j^n + k\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} D_0v_j^n + \frac{k^2}{2}\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} D_+D_-v_j^n \tag{22}$$

Before proceeding to analyze the accuracy of this scheme, we briefly observe that this problem is ill-posed. By Theorem 4.3.1 on page 116 in [1], the initial value problem for $u_t = Au_x$ will be well-posed if, and only if, the eigenvalues of A are real and distinct and there is a complete system of eigenvectors. By inspection, the eigenvalues of A are $\lambda = \{-i, i\}$ and the associated eigenvectors are $\{(i \quad 1)^T, (-i \quad 1)^T\}$. Regardless, we can continue to apply the standard accuracy and stability analysis procedures to the approximation scheme in Equation 22.

To analyze the accuracy of the proposed scheme, we begin by writing Equation 21 in the "normalized" form.

$$\frac{v_j^{n+1} - v_j^n}{k} = AD_0v_j^n + \frac{k}{2}A^2D_+D_-v_j^n$$

Next, we assume that u is a smooth function and substitute into the previous expression to obtain the truncation error τ_j^n evaluated at (x_j, t_n).

$$\tau_j^n = \frac{u_j^{n+1} - u_j^n}{k} - AD_0u_j^n - \frac{k}{2}A^2D_+D_-u_j^n \tag{23}$$

At this point we recognize that the first term has the following Taylor series expansion (where we have applied the PDE to relate temporal to spatial derivatives).

$$\begin{aligned}\frac{u(x,t+k)-u(x,t)}{k} &= u_t(x,t)+\frac{k}{2}u_{tt}(x,t)+\frac{k^2}{6}u_{ttt}(x,t)+\mathcal{O}(k^3)\\ &= Au_x(x,t)+\frac{kA^2}{2}u_{xx}(x,t)+\frac{k^2A^3}{6}u_{xxx}(x,t)+\mathcal{O}(k^3)\end{aligned} \tag{24}$$

In addition, we recall the following Taylor series expansions for the remaining terms (from page 59 in [1]).

$$D_0u(x,t) = u_x(x,t)+\frac{h^2}{3!}u_{xxx}(x,t)+\frac{h^4}{5!}u_{xxxxx}(x,t)+\mathcal{O}(h^6) \tag{25}$$

$$D_+D_-u(x,t) = u_{xx}(x,t)+\frac{2h^2}{4!}u_{xxxx}(x,t)+\frac{2h^4}{6!}u_{xxxxxx}(x,t)+\mathcal{O}(h^6) \tag{26}$$

Substituting Equations 24-26 into Equation 23 gives the following expression for the truncation error. (Note that this result was verified using the attached *Mathematica* notebook.)

$$\begin{aligned}\tau_j^n = &\left\{Au_x(x,t)+\frac{kA^2}{2}u_{xx}(x,t)+\frac{k^2A^3}{6}u_{xxx}(x,t)+\mathcal{O}(k^3)\right\}-\\ &A\left\{u_x(x,t)+\frac{h^2}{3!}u_{xxx}(x,t)+\frac{h^4}{5!}u_{xxxxx}(x,t)+\mathcal{O}(h^6)\right\}-\\ &\frac{k}{2}A^2\left\{u_{xx}(x,t)+\frac{2h^2}{4!}u_{xxxx}(x,t)+\frac{2h^4}{6!}u_{xxxxxx}(x,t)+\mathcal{O}(h^6)\right\}\end{aligned}$$

Since the terms in u_x and u_{xx} cancel, we are left with the following expression for the truncation error. (Also note that, since $A \neq A^3$, no higher-order accuracy can be obtained by optimizing the choice of k and h.)

$$\tau_j^n = \mathcal{O}(h^2+k^2)$$

Recall from page 165 in [1] that we generally assume a relationship between k and h, where $k=\lambda h^p$ (with p the order of the differential operator in space and a positive constant λ). Since this is a first-order PDE, we assume $k=\lambda h$. As a result, we conclude that the proposed scheme is **accurate to order (2,2)** according to the expression for the truncation error. (In general, however, one must be careful when defining the initial condition since the system is ill-posed. Theoretically, this remains a second-order scheme based on the truncation error.)

At this point we need to obtain the amplification factor $\hat{Q}$ by substituting the typical ansatz

$$v_j^n = \frac{1}{\sqrt{2\pi}}\,e^{i\omega x_j}\hat{v}^n(\omega)$$

into Equation 21. Recall the following Fourier transforms were derived in Problems 1 and 5 in Problem Set 4.

$$\begin{aligned}D_0e^{i\omega x_j} &= \frac{i}{h}\sin(\xi)e^{i\omega x_j}\\ D_+D_-e^{i\omega x_j} &= -\frac{4}{h^2}\sin^2\left(\frac{\xi}{2}\right)e^{i\omega x_j}\end{aligned}$$

Substituting these identities and following our derivation from Problem 1 in Problem Set 8, we obtain the following form for the symbol $\hat{Q}$.

$$\hat{Q} = I + i\lambda A \sin(\xi) - 2\lambda^2 A^2 \sin^2\left(\frac{\xi}{2}\right) \tag{27}$$

Substituting for A obtains the following expression for the amplification factor.

$$\hat{Q} = \begin{pmatrix} 1 + 2\lambda^2 \sin^2\left(\frac{\xi}{2}\right) & i\lambda \sin(\xi) \\ -i\lambda \sin(\xi) & 1 + 2\lambda^2 \sin^2\left(\frac{\xi}{2}\right) \end{pmatrix}$$

We note that there are two eigenvalues z_ν of $\hat{Q}$ given by the following expressions.

$$z_1 = 1 - \lambda \sin(\xi) + 2\lambda^2 \sin^2\left(\frac{\xi}{2}\right) \qquad z_2 = 1 + \lambda \sin(\xi) + 2\lambda^2 \sin^2\left(\frac{\xi}{2}\right)$$

Recall that Theorem 5.2.2 on page 173 in [1] defines the von Neumann stability condition for approximations with constant coefficients; specifically, a necessary condition for stability is that the eigenvalues z_ν of $\hat{Q}$ satisfy

$$|z_\nu| \leq e^{\alpha_S k}, \qquad |\xi| \leq \pi,$$

for all $h \leq h_0$. For $|\xi| \leq \pi$, we note that $\sin(\xi) \leq 1$ and $\sin^2\left(\frac{\xi}{2}\right) \leq 1$. As a result, the eigenvalues are bounded from above by $|z_\nu| \leq 1 + \lambda + 2\lambda^2$, for $\lambda > 0$, and the von Neumann condition is not satisfied for all λ. Furthermore, we recall from Corollary 5.2.1 on page 175 in [1] that the von Neumann condition is sufficient if $\hat{Q}$ is Hermitian. Since $\hat{Q}$ is Hermitian, **we conclude that the proposed Lax-Wendroff scheme is unconditionally unstable**.

Part 2: Discuss the accuracy and stability of the Lax-Wendroff scheme for $u_t = Au_x$, assuming 2π periodicity, for

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Before proceeding to analyze the accuracy and stability of this scheme, we briefly observe that this problem is also ill-posed. By Theorem 4.3.1 on page 116 in [1], the initial value problem for $u_t = Au_x$ will be well-posed if, and only if, the eigenvalues of A are real and distinct and there is a complete system of eigenvectors. By inspection, the eigenvalues of A are $\lambda = \{0, 0\}$ and the associated eigenvectors are $\{(1 \quad 0)^T, (0 \quad 0)^T\}$. (Note that by Definition 4.3.1 on page 119 in [1] this problem is weakly-hyperbolic, since the eigenvalues are real but not distinct.) Regardless, we can continue to apply the standard accuracy and stability analysis procedures to the approximation scheme in Equation 21.

To begin our analysis, we note that all the powers of A are zero for $n \geq 2$.

$$A^n = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \text{ for } n \geq 2$$

Substituting for A and A^2 in Equation 21 gives the following explicit form of the Lax-Wendroff scheme for this part.

$$v_j^{n+1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} v_j^n + k \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} D_0 v_j^n \tag{28}$$

Note that, since all higher powers of A are zero, the Lax-Wendroff scheme reduces to the forward Euler scheme. (Also recall from page 167 of [1] that the one-dimensional forward Euler scheme is only first-order accurate and unconditionally unstable. As we proceed we'll look for similar results to manifest themselves in this example.) As before, we can write Equation 21 in the "normalized" form (with $A^2 = 0$).

$$\frac{v_j^{n+1} - v_j^n}{k} = AD_0 v_j^n$$

Next, we assume that u is a smooth function and substitute into the previous expression to obtain the truncation error τ_j^n evaluated at (x_j, t_n).

$$\tau_j^n = \frac{u_j^{n+1} - u_j^n}{k} - AD_0 u_j^n \tag{29}$$

Substituting Equations 24 and 25 into Equation 29 gives the following expression for the truncation error. (Note that this result was verified using the attached *Mathematica* notebook. Also note that the terms in A^n have been eliminated for $n \geq 2$.)

$$\begin{aligned}\tau_j^n &= Au_x(x,t) - A\left\{u_x(x,t) + \frac{h^2}{3!}u_{xxx}(x,t) + \frac{h^4}{5!}u_{xxxxx}(x,t) + \mathcal{O}(h^6)\right\} \\ &= -\frac{h^2}{3!}Au_{xxx}(x,t) - \frac{h^4}{5!}Au_{xxxxx}(x,t) + \mathcal{O}(h^6)\end{aligned}$$

Since the term in u_x cancels, we are left with the following expression for the truncation error.

$$\tau_j^n = \mathcal{O}(h^2)$$

Note that, since all terms in k have been eliminated, the truncation error is independent of k. In conclusion, we report that the the proposed scheme is **second-order accurate** according to the expression for the truncation error. (As before, one must be careful when defining the initial condition since the system is ill-posed. Theoretically, this remains a second-order scheme based on the truncation error.)

To assess the stability of this scheme we proceed as in the previous part. First, we substitute for A in Equation 27 to obtain the following expression for the symbol $\hat{Q}$.

$$\hat{Q} = \begin{pmatrix} 1 & i\lambda\sin(\xi) \\ 0 & 1 \end{pmatrix}$$

We note that there is a single repeated eigenvalue z of $\hat{Q}$ given by $z = 1$ (since $\hat{Q}$ is upper triangular). As a result, the von Neumann condition is satisfied; note, however, that this condition is not necessarily sufficient since $\hat{Q}$ is not a normal matrix. Recall from Theorem 5.2.1 on page 173 in [1] that the approximation scheme is stable if, and only if,

$$|\hat{Q}^n(\xi)| \leq K_S e^{\alpha_S t_n},$$

for all $h = 2\pi/(N+1) \leq h_0$ and all $|\omega| \leq N/2$. In this case, the powers of $\hat{Q}^n$ can be obtained directly.

$$\hat{Q}^n = \begin{pmatrix} 1 & in\lambda\sin(\xi) \\ 0 & 1 \end{pmatrix}$$

Note that the norm of $\hat{Q}^n(\xi)$ is of order n, which cannot be bounded by $K_S e^{\alpha_S t_n}$. We conclude that, by Theorem 5.2.1, **the proposed Lax-Wendroff scheme is unconditionally unstable** for this problem.

Problem 4

Consider the following scheme for the hyperbolic system $u_t + Au_x = 0$.

$$v_j^* = v_j^n - \frac{kA}{6h}\left(-v_{j+2}^n + 8v_{j+1}^n - 7v_j^n\right) \tag{30}$$

$$v_j^{n+1} = \frac{1}{2}\left(v_j^n + v_j^* - \frac{kA}{6h}\left(7v_j^* - 8v_{j-1}^* + v_{j-2}^*\right)\right) \tag{31}$$

What is the accuracy of this scheme?

In order to estimate the truncation error, we begin by substituting Equation 30 for the first instance of v_j^* in Equation 31.

$$v_j^{n+1} = v_j^n - \frac{kA}{12h}\left(-v_{j+2}^n + 8v_{j+1}^n - 7v_j^n\right) - \frac{kA}{12h}\left(7v_j^* - 8v_{j-1}^* + v_{j-2}^*\right)$$

Next, we "normalize" this expression by subtracting v_j^n from both sides and dividing by k.

$$\frac{v_j^{n+1} - v_j^n}{k} = -\frac{A}{12h}\left(-v_{j+2}^n + 8v_{j+1}^n - 7v_j^n\right) - \frac{A}{12h}\left(7v_j^* - 8v_{j-1}^* + v_{j-2}^*\right) \tag{32}$$

Note that the left-hand side is the forward difference approximation to u_t, whereas the right-hand side approximates $-Au_x$. To proceed, we recognize the following expressions for v_{j-1}^* and v_{j-2}^*.

$$v_{j-1}^* = v_{j-1}^n - \frac{kA}{6h}\left(-v_{j+1}^n + 8v_j^n - 7v_{j-1}^n\right) \tag{33}$$

$$v_{j-2}^* = v_{j-2}^n - \frac{kA}{6h}\left(-v_j^n + 8v_{j-1}^n - 7v_{j-2}^n\right) \tag{34}$$

Substituting Equations 30, 33 and 34 into Equation 32 gives the following form for the difference scheme.

$$\begin{aligned}\frac{v_j^{n+1} - v_j^n}{k} = &\left(-\frac{A}{12h} - \frac{7kA^2}{72h^2}\right)v_{j-2}^n + \left(\frac{2A}{3h} + \frac{8kA^2}{9h^2}\right)v_{j-1}^n + \left(-\frac{19kA^2}{12h^2}\right)v_j^n + \\ &\left(-\frac{2A}{3h} + \frac{8kA^2}{9h^2}\right)v_{j+1}^n + \left(\frac{A}{12h} - \frac{7kA^2}{72h^2}\right)v_{j+2}^n\end{aligned}$$

Following the approach on pages 59 and 60 in [1], the truncation error τ_j^n evaluated at (x_j, t_n) is obtained by assuming u is a smooth function and substituting into the previous expression.

$$\begin{aligned}\tau_j^n = &\frac{u_j^{n+1} - u_j^n}{k} - \left(-\frac{A}{12h} - \frac{7kA^2}{72h^2}\right)u_{j-2}^n - \left(\frac{2A}{3h} + \frac{8kA^2}{9h^2}\right)u_{j-1}^n - \\ &\left(-\frac{19kA^2}{12h^2}\right)u_j^n - \left(-\frac{2A}{3h} + \frac{8kA^2}{9h^2}\right)u_{j+1}^n - \left(\frac{A}{12h} - \frac{7kA^2}{72h^2}\right)u_{j+2}^n\end{aligned} \tag{35}$$

At this point we recognize that the first term has the following Taylor series expansion (where we have applied the PDE to relate temporal to spatial derivatives).

$$\begin{aligned}\frac{u(x,t+k) - u(x,t)}{k} &= u_t(x,t) + \frac{k}{2}u_{tt}(x,t) + \frac{k^2}{6}u_{ttt}(x,t) + \mathcal{O}(k^3) \\ &= -Au_x(x,t) + \frac{kA^2}{2}u_{xx}(x,t) - \frac{k^2A^3}{6}u_{xxx}(x,t) + \mathcal{O}(k^3)\end{aligned} \tag{36}$$

In addition, we recall the following Taylor series expansions for the remaining terms.

$$u(x-2h,t) = u(x,t) - 2hu_x(x,t) + 2h^2u_{xx}(x,t) - \frac{4h^3}{3}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{37}$$

$$u(x-h,t) = u(x,t) - hu_x(x,t) + \frac{h^2}{2}u_{xx}(x,t) - \frac{h^3}{6}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{38}$$

$$u(x+h,t) = u(x,t) + hu_x(x,t) + \frac{h^2}{2}u_{xx}(x,t) + \frac{h^3}{6}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{39}$$

$$u(x+2h,t) = u(x,t) + 2hu_x(x,t) + 2h^2u_{xx}(x,t) + \frac{4h^3}{3}u_{xxx}(x,t) + \mathcal{O}(h^4) \tag{40}$$

Substituting Equations 36-40 into Equation 35 gives the following expression for the truncation error. (Note that this result was verified using the attached *Mathematica* notebook.)

$$\tau_j^n = \left\{-Au_x(x_j,t_n) + \frac{kA^2}{2}u_{xx}(x_j,t_n) - \frac{k^2A^3}{6}u_{xxx}(x_j,t_n) + \mathcal{O}(k^3)\right\} - \left\{-Au_x(x_j,t_n) + \frac{kA^2}{2}u_{xx}(x_j,t_n) - \frac{kh^2A^2}{18}u_{xxxx}(x_j,t_n) + \frac{h^4A}{30}u_{xxxxx}(x_j,t_n) + \mathcal{O}(kh^4)\right\}$$

Since the terms in u_x and u_{xx} cancel, we are left with the following expression for the truncation error.

$$\tau_j^n = \mathcal{O}(h^4 + kh^2 + k^2)$$

Recall from page 165 in [1] that we generally assume a relationship between k and h, where $k = \lambda h^p$ (with p the order of the differential operator in space and a positive constant λ). Since this is a first-order PDE, we assume $k = \lambda h$. As a result, we conclude that the proposed scheme is **second-order accurate** according to the expression for the truncation error (with $k = \lambda h$).

References

[1] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time Dependent Problems and Difference Methods*. John Wiley & Sons, 1995.

[2] S. K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, New York, NY, 2006.

[3] Eric W. Weisstein. Fourier series. `http://mathworld.wolfram.com/FourierSeries.html`.

[4] Eric W. Weisstein. Half-angle formulas. `http://mathworld.wolfram.com/Half-AngleFormulas.html`.

```
 1 function EulerMethod
 2
 3 % AM 255, Final Exam, Problem 1: Part 2
 4 %    Solves heat equation initial value problems using
 5 %    the Euler method. Results are displayed graphically
 6 %    and tabulated for inclusion in the write-up.
 7 %
 8 % Douglas Lanman, Brown University, Dec. 2006
 9
10 % Reset Matlab environment.
11 clear; clc;
12
13 % Set Euler method parameters.
14 initialCondition = 2;  % (1 = "boxcar", 2 = cosine)
15
16
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 % Part I: Specify discrete grid parameters.
19
20 % Specify the initial condition and exact solution.
21 switch initialCondition
22    % Case 1: First initial condition (i.e., "boxcar").
23    % Note: Must take care when approximating "boxcar".
24    %       See write-up for explaination.
25    case 1
26       IC = @(x) u(x,0,20);    % use 10 terms to approximate initial condition
27       ES = @(x,t) u(x,t,100); % use 100 terms to approximate exact solution
28    % Case 2: Second initial condition (i.e., cosine).
29    otherwise
30       IC = @(x) cos(pi*x);
31       ES = @(x,t) exp(-pi^2*t)*cos(pi*x);
32 end
33
34 % Define space grid interval(s) for evaluation.
35 h = 2./[21 41 81 161 321]; % space steps
36 N = (2./h)-1;              % #gridpoints s.t. N+2 on [-1,1]
37
38 % Select the final time for evaluation.
39 % Note: Initial time is assumed to be zero.
40 tf = 1.0;
41
42 % Select time step.
43 k = 0.4*(h.^2);
44
45 % Set discrete positions/time-steps for evaluation.
46 % Note: All time steps will be equal, except the
47 %       last; it will be adjusted so that the final
48 %       time will be exactly 'tf'.
49 x = cell(1,length(N));
50 t = cell(1,length(N));
```

```
for i = 1:length(N)
   x{i} = -1:h(i):1;
   x{i} = x{i}(1:end-1);
   t{i} = (0:k(i):tf);
   if t{i}(end) ~= tf
      t{i}(end+1) = tf;
   end
end

% Initialize the numerical solution(s).
v = cell(1,length(N));
for i = 1:length(N)
   v{i} = zeros(length(t{i}),N(i)+1);
   v{i}(1,:) = IC(x{i}); % boundary values
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part II: Solve IVP using the Euler method.

% Update solution sequentially (beginning with I.C.).
% Note: DpDm.m implements the second-order difference operator.
%       Modify amplication factor for the last time step.
for i = 1:length(N)
   for n = 1:(length(t{i})-1)
      if n ~= (length(t{i})-1)
         v{i}(n+1,:) = v{i}(n,:) + k(i)*DpDm(v{i}(n,:),h(i));
      else
         kf = diff(t{i}(end-1:end));
         v{i}(n+1,:) = v{i}(n,:) + kf*DpDm(v{i}(n,:),h(i));
      end
   end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part III: Plot/tabulate modeling results.

% Evaluate the exact solution.
xe = linspace(-1,1,1000);
fe = ES(xe,tf);

% Determine the L2-error and the approximation order.
L2_error = zeros(1,length(N));
order    = zeros(1,length(N));
for i = 1:length(N)
   L2_error(i) = sqrt(sum((abs(ES(x{i},t{i}(end))-v{i}(end,:)).^2)*h(i)));
   if i > 1
      order(i) = log2(L2_error(i-1)/L2_error(i));
   end
```

```
101 end
102
103 % Tabulate results.
104 disp(' N     L2-error     order');
105 disp('--------------------------');
106 for i = 1:length(N)
107    if i > 1
108       fprintf('%3d   %.5g   %+2.2f\n',N(i),L2_error(i),order(i));
109    else
110       fprintf('%3d   %.5g\n',N(i),L2_error(i));
111    end
112 end
113
114 % Compare approximation to exact solution.
115 for i = 1:length(N)
116    figure(i); clf;
117    if initialCondition ~= 1
118       plot(xe,fe,'r-','LineWidth',3);
119    else
120       plot(xe,fe-0.5,'r-','LineWidth',3);
121    end
122    hold on;
123       if i > 2
124          if initialCondition ~= 1
125             plot(x{i},v{i}(end,:),'--','LineWidth',3);
126          else
127             plot(x{i},v{i}(end,:)-0.5,'--','LineWidth',3);
128          end
129       else
130          if initialCondition ~= 1
131             plot(x{i},v{i}(end,:),'.','MarkerSize',20);
132          else
133             plot(x{i},v{i}(end,:)-0.5,'.','MarkerSize',20);
134          end
135       end
136    hold off;
137    set(gca,'LineWidth',2,'FontSize',14,'FontWeight','normal');
138    xlabel('$x$','FontName','Times','Interpreter','Latex','FontSize',16);
139    %title('Difference Approximation vs. Analytic Solution');
140    legend('Analytic Solution','Difference Approx.');
141    grid on; xlim([-1 1]);
142    if initialCondition ~= 1
143       ylim([-6e-5 8e-5]);
144    else
145       ylim([-4e-5 6e-5]);
146    end
147 end
148
149 % Compare initial condition to truncated Fourier series.
150 figure(length(N)+1); clf;
```

```
151 if initialCondition == 1
152    plot(xe,abs(xe)<=.5,'r-','LineWidth',3);
153 else
154    plot(xe,ES(xe,0),'r-','LineWidth',3);
155 end
156 hold on;
157    if initialCondition == 1
158       plot(x{end},v{end}(1,:),'-','LineWidth',3);
159    else
160       plot(x{end},v{end}(1,:),'--','LineWidth',3);
161    end
162 hold off;
163 set(gca,'LineWidth',2,'FontSize',14,'FontWeight','normal');
164 xlabel('$x$','FontName','Times','Interpreter','Latex','FontSize',16);
165 %title('Difference Approximation vs. Analytic Solution');
166 legend('Analytic Solution','Difference Approx.');
167 grid on; xlim([-1 1]);
168 if initialCondition == 1
169    ylim([-0.2 1.4]);
170 else
171    ylim([-1 1.5]);
172 end
173
174
175 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
176 % Define exact solution to "boxcar" initial condition.
177 % Note: K defines number of terms in series approximation.
178 function f = u(x,t,k)
179 f = (1/2)*ones(size(x));
180 for w = 1:k
181    f = f + exp(-(pi*w)^2*t)*(sin(pi*w/2)/(pi*w/2))*cos(pi*w*x);
182 end
```

```
 1 function b = DpDm(a,h)
 2
 3 % DpDm Sequential backward/forward difference operator.
 4 %    DpDm(A,H) evaluates the sequential backward/forward
 5 %    difference of the array A with grid-spacing H, as
 6 %    defined in:
 7 %
 8 %    "Time Dependent Problems and Difference Methods",
 9 %    B. Gustafsson, H.-O. Kreiss, and J. Oliger, 1995.
10 %
11 % Douglas Lanman, Brown University, Dec. 2006
12
13 % Determine the length of the input array.
14 N = length(a);
15
16 % Shift array indices (modulo the array length).
17 fj = mod((1:N)-2,N)+1; % shift forward
18 bj = mod((1:N),N)+1;   % shift backward
19
20 % Evaluate the sequential backward/forward difference.
21 b = (a(bj)-2*a+a(fj))/(h^2);
```

```
 1 function CrankNicholson
 2
 3 % AM 255, Final Exam, Problem 1: Part 2
 4 %    Solves heat equation initial value problems using
 5 %    the Crank-Nicholson method. Results are displayed
 6 %    graphically and tabulated for the write-up.
 7 %
 8 % Douglas Lanman, Brown University, Dec. 2006
 9
10 % Reset Matlab environment.
11 clear; clc;
12
13 % Set Euler method parameters.
14 initialCondition = 1;  % (1 = "boxcar", 2 = cosine)
15 kMode            = 2;  % (1 = {k = 10*h^2}, 2 = {k = h})
16
17
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19 % Part I: Specify discrete grid parameters.
20
21 % Specify the initial condition and exact solution.
22 switch initialCondition
23    % Case 1: First initial condition (i.e., "boxcar").
24    % Note: Must take care when approximating "boxcar".
25    %       See write-up for explaination.
26    case 1
27       IC = @(x) u(x,0,10); % use 2 terms to approximate initial condition
28       ES = @(x,t) u(x,t,100); % use 100 terms to approximate exact solution
29    % Case 2: Second initial condition (i.e., cosine).
30    otherwise
31       IC = @(x) cos(pi*x);
32       ES = @(x,t) exp(-pi^2*t)*cos(pi*x);
33 end
34
35 % Define space grid interval(s) for evaluation.
36 h = 2./[21 41 81 161 321 641]; % space steps
37 N = (2./h)-1;                  % #gridpoints s.t. N+2 on [-1,1]
38
39 % Select the final time for evaluation.
40 % Note: Initial time is assumed to be zero.
41 tf = 1.0;
42
43 % Select time step.
44 if kMode == 1
45    k = 10*(h.^2);
46 else
47    k = h;
48 end
49
50 % Set discrete positions/time-steps for evaluation.
```

```
% Note: All time steps will be equal, except the
%       last; it will be adjusted so that the final
%       time will be exactly 'tf'.
x = cell(1,length(N));
t = cell(1,length(N));
for i = 1:length(N)
   x{i} = -1:h(i):1;
   x{i} = x{i}(1:end-1);
   t{i} = (0:k(i):tf);
   if t{i}(end) ~= tf
      t{i}(end+1) = tf;
   end
end

% Initialize the numerical solution(s).
v = cell(1,length(N));
for i = 1:length(N)
   v{i} = zeros(length(t{i}),N(i)+1);
   v{i}(1,:) = IC(x{i}); % boundary values
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part II: Solve IVP using Crank-Nicholson.

% Update solution sequentially (beginning with I.C.).
for i = 1:length(N)

   % Store forward/backward difference operators.
   I  = eye(N(i)+1);
   Dp = (1/h(i))*(circshift(I,[0 1])-I);
   Dm = (1/h(i))*(I-circshift(I,[0 -1]));

   % Evaluate amplification factor.
   A = I - (k(i)/2)*Dp*Dm;
   B = I + (k(i)/2)*Dp*Dm;
   Q = B/A;

   % Calculate Crank-Nicholson solution.
   % Note: Modify amplication factor for the last time step.
   for n = 1:(length(t{i})-1)
      if n ~= (length(t{i})-1)
         v{i}(n+1,:) = (Q*v{i}(n,:)')';
      else
         kf = diff(t{i}(end-1:end));
         A  = I - (kf/2)*Dp*Dm;
         B  = I + (kf/2)*Dp*Dm;
         Q  = B/A;
         v{i}(n+1,:) = (Q*v{i}(n,:)')';
      end
```

```
101    end
102
103 end % End of Crank-Nicholson solution.
104
105
106 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
107 % Part III: Plot/tabulate modeling results.
108
109 % Evaluate the exact solution.
110 xe = linspace(-1,1,1000);
111 fe = ES(xe,tf);
112
113 % Determine the L2-error and the approximation order.
114 L2_error = zeros(1,length(N));
115 order    = zeros(1,length(N));
116 for i = 1:length(N)
117    L2_error(i) = sqrt(sum((abs(ES(x{i},t{i}(end))-v{i}(end,:)).^2)*h(i)));
118    if i > 1
119       order(i) = log2(L2_error(i-1)/L2_error(i));
120    end
121 end
122
123 % Tabulate results.
124 disp(' N     L2-error     order');
125 disp('--------------------------');
126 for i = 1:length(N)
127    if i > 1
128       fprintf('%3d   %.5g   %+2.2f\n',N(i),L2_error(i),order(i));
129    else
130       fprintf('%3d   %.5g\n',N(i),L2_error(i));
131    end
132 end
133
134 % Compare approximation to exact solution.
135 for i = 1:length(N)
136    figure(i); clf;
137    if initialCondition ~= 1
138       plot(xe,fe,'r-','LineWidth',3);
139    else
140       plot(xe,fe-0.5,'r-','LineWidth',3);
141    end
142    hold on;
143       if i > 2
144          if initialCondition ~= 1
145             plot(x{i},v{i}(end,:),'--','LineWidth',3);
146          else
147             plot(x{i},v{i}(end,:)-0.5,'--','LineWidth',3);
148          end
149       else
150          if initialCondition ~= 1
```

```
151             plot(x{i},v{i}(end,:),'.','MarkerSize',20);
152          else
153             plot(x{i},v{i}(end,:)-0.5,'.','MarkerSize',20);
154          end
155       end
156    hold off;
157    set(gca,'LineWidth',2,'FontSize',14,'FontWeight','normal');
158    xlabel('$x$','FontName','Times','Interpreter','Latex','FontSize',16);
159    %title('Difference Approximation vs. Analytic Solution');
160    legend('Analytic Solution','Difference Approx.');
161    grid on; xlim([-1 1]);
162    if initialCondition ~= 1
163       ylim([-6e-5 8e-5]);
164    else
165       ylim([-4e-5 6e-5]);
166    end
167 end
168
169 % Compare initial condition to truncated Fourier series.
170 figure(length(N)+1); clf;
171 if initialCondition == 1
172    plot(xe,abs(xe)<=.5,'r-','LineWidth',3);
173 else
174    plot(xe,ES(xe,0),'r-','LineWidth',3);
175 end
176 hold on;
177    if initialCondition == 1
178       plot(x{end},v{end}(1,:),'-','LineWidth',3);
179    else
180       plot(x{end},v{end}(1,:),'--','LineWidth',3);
181    end
182 hold off;
183 set(gca,'LineWidth',2,'FontSize',14,'FontWeight','normal');
184 xlabel('$x$','FontName','Times','Interpreter','Latex','FontSize',16);
185 %title('Difference Approximation vs. Analytic Solution');
186 legend('Analytic Solution','Difference Approx.');
187 grid on; xlim([-1 1]);
188 if initialCondition == 1
189    ylim([-0.2 1.4]);
190 else
191    ylim([-1 1.5]);
192 end
193
194
195 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
196 % Define exact solution to "boxcar" initial condition.
197 % Note: K defines number of terms in series approximation.
198 function f = u(x,t,k)
199 f = (1/2)*ones(size(x));
200 for w = 1:k
```

```
201     f = f + exp(-(pi*w)^2*t)*(sin(pi*w/2)/(pi*w/2))*cos(pi*w*x);
202 end
```

```
function upwind

% AM 255, Final Exam, Problem 2
%    Solves the first-order hyperbolic wave equation using
%    the third-order upwind method. Results are displayed
%    graphically and tabulated for inclusion in the write-up.
%
% Douglas Lanman, Brown University, Dec. 2006

% Reset Matlab environment.
clear; clc;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part I: Specify discrete grid parameters.

% Specify the initial condition and exact solution.
IC = @(x) sin(x);
ES = @(x,t) sin(x+t);

% Define space/time grid interval(s) for evaluation.
N = [10 20 40 80 160 320 640]; % #gridpoints s.t. N+2 on [0,2*pi]
h = 2*pi./(N+1);               % resulting space steps

% Select time step.
%k = 0.5*h;
k = 1.9*h;

% Select the final time for evaluation.
% Note: Initial time is assumed to be zero.
tf = 4*pi;

% Set discrete positions/time-steps for evaluation.
% Note: All time steps will be equal, except the
%       last; it will be adjusted so that the final
%       time will be exactly 'tf'.
x = cell(1,length(N));
t = cell(1,length(N));
for i = 1:length(N)
   x{i} = h(i)*(0:N(i));
   t{i} = (0:k(i):tf);
   if t{i}(end) ~= tf
      t{i}(end+1) = tf;
   end
end

% Initialize the numerical solution(s).
v = cell(1,length(N));
for i = 1:length(N)
   v{i} = zeros(length(t{i}),N(i)+1);
```

```
51     v{i}(1,:) = IC(x{i}); % boundary values
52 end
53 
54 
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56 % Part II: Solve IVP using the updwind method.
57 
58 % Evaluate upwind coefficients.
59 lambda = k./h;
60 a = -(1/3)*lambda + (1/2)*lambda.^2 - (1/6)*lambda.^3;
61 b = 1 - (1/2)*lambda - lambda.^2 + (1/2)*lambda.^3;
62 c = lambda + (1/2)*lambda.^2 - (1/2)*lambda.^3;
63 d = -(1/6)*lambda + (1/6)*lambda.^3;
64 
65 % Update solution sequentially (beginning with I.C.).
66 % Note: Modify amplication factor for the last time step.
67 for i = 1:length(N)
68    for n = 1:(length(t{i})-1)
69       if n ~= (length(t{i})-1)
70          v{i}(n+1,:) = a(i)*circshift(v{i}(n,:),[1 1]) + ...
71                        b(i)*circshift(v{i}(n,:),[1 0]) + ...
72                        c(i)*circshift(v{i}(n,:),[1 -1]) + ...
73                        d(i)*circshift(v{i}(n,:),[1 -2]);
74       else
75          kf = diff(t{i}(end-1:end));
76          lf = kf/h(i);
77          af = -(1/3)*lf + (1/2)*lf.^2 - (1/6)*lf.^3;
78          bf = 1 - (1/2)*lf - lf.^2 + (1/2)*lf.^3;
79          cf = lf + (1/2)*lf.^2 - (1/2)*lf.^3;
80          df = -(1/6)*lf + (1/6)*lf.^3;
81          v{i}(n+1,:) = af*circshift(v{i}(n,:),[1 1]) + ...
82                        bf*circshift(v{i}(n,:),[1 0]) + ...
83                        cf*circshift(v{i}(n,:),[1 -1]) + ...
84                        df*circshift(v{i}(n,:),[1 -2]);
85       end
86    end
87 end
88 
89 
90 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91 % Part III: Plot/tabulate modeling results.
92 
93 % Evaluate the exact solution.
94 xe = linspace(0,2*pi,1000);
95 fe = ES(xe,tf);
96 
97 % Determine the L2-error and the approximation order.
98 L2_error = zeros(1,length(N));
99 order    = zeros(1,length(N));
100 for i = 1:length(N)
```

```
101     L2_error(i) = sqrt(sum((abs(ES(x{i},t{i}(end))-v{i}(end,:)).^2)*h(i)));
102     if i > 1
103        order(i) = log2(L2_error(i-1)/L2_error(i));
104     end
105  end
106
107  % Tabulate results.
108  disp(' N     L2-error     order');
109  disp('--------------------------');
110  for i = 1:length(N)
111     if i > 1
112        fprintf('%3d   %.5g   %+2.2f\n',N(i),L2_error(i),order(i));
113     else
114        fprintf('%3d   %.5g\n',N(i),L2_error(i));
115     end
116  end
117
118  % Compare approximation to exact solution.
119  for i = 1:length(N)
120     figure(i); clf;
121     plot(xe,fe,'r-','LineWidth',3);
122     hold on;
123        if i > 2
124           plot(x{i},v{i}(end,:),'--','LineWidth',3);
125        else
126           plot(x{i},v{i}(end,:),'.','MarkerSize',20);
127        end
128     hold off;
129     set(gca,'LineWidth',2,'FontSize',14,'FontWeight','normal');
130     xlabel('$x$','FontName','Times','Interpreter','Latex','FontSize',16);
131     %title('Difference Approximation vs. Analytic Solution');
132     legend('Analytic Solution','Difference Approx.');
133     grid on; xlim([0 2*pi]);
134  end
```