1. Derivation of Linear multistep (LMS) methods

$$U^{n+3} = U^{n+1} + \frac{k}{3}\left[7f(t_{n+2}, U^{n+2}) - 2f(t_{n+1}, U^{n+1}) + f(t_n, U^n)\right]$$

a) Derive this method using any technique you desire.

Consider

$$u(t_{n+3}) = u(t_{n+1}) + \int_{t_{n+1}}^{t_{n+3}} f(t, u(t))\,dt$$

$$u'(t) = f(t, u)$$

Consider a lagrange polynomial

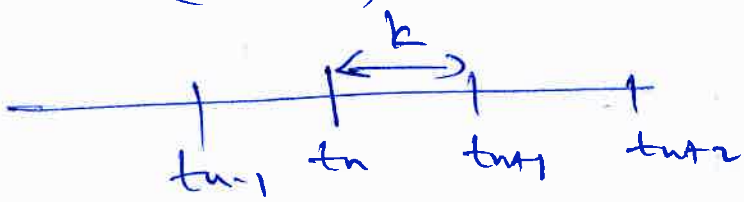
$$L_j(t) = \frac{(t-t_0)\cdots(t-t_{j-1})(t-t_{j+1})\cdots(t-t_n)}{(t_j-t_0)\cdots(t_j-t_{j-1})(t_j-t_{j+1})\cdots(t_j-t_n)}$$

$$\int_{t_{n+1}}^{t_{n+3}} f(t, u(t))\,dt = \sum_{j=0}^{n=2} \int_{t_{n+1}}^{t_{n+3}} L_j(t)\,dt\, f^{n+j}$$

$$\int_{t_{n+1}}^{t_{n+3}} f(t, u(t))\,dt = \int_{t_{n+1}}^{t_{n+3}} L_0(t)\,dt\, f^n + \int_{t_{n+1}}^{t_{n+3}} L_1(t)\,dt\, f^{n+1} + \int_{t_{n+1}}^{t_{n+3}} L_2(t)\,dt\, f^{n+2}$$

start with $\int_{t_{n+1}}^{t_{n+3}} L_0(t)\,dt\, f^n$

$$L_0(t) = \frac{(t-t_{n-1})(t-\cancel{t_{n+2}})}{(t_n-t_{n+1})(t_n-t_{n+2})}$$



, k is the time step

$$L_0(t) = \frac{(t-t_{n-1})(t-t_{n-2})}{2k^2}$$

~~let s = t + 1~~

$$\int_{t_{n+1}}^{t_{n+3}} \left(\frac{(t - t_{n-1})(t - t_{n-2})}{2k^2} \right) dt \, f^{n}$$

let $s = t - t_{n-1} \rightarrow ds = dt$

t	s
t_{n+1}	0
t_{n+3}	$2k$

and also

$t - t_{n-1} = s, \quad t - t_{n-2} = s - k$

so

$$\int_{t_{n+1}}^{t_{n+3}} \frac{(t - t_{n-1})(t - t_{n-2})}{2k^2} dt \, f^{n} = \int_0^{2k} \frac{s(s-k)}{2k^2} ds \, f^{n}$$

$$= \underline{\underline{\frac{k}{3} f^{n}}}$$

$$\int_{t_{n+1}}^{t_{n+3}} L_1(t) \, dt \, f^{n+1} = \int_{t_{n+1}}^{t_{n+3}} \frac{(t - t_n)(t - t_{n+2})}{-k^2} dt \, f^{n+1}$$

$$= \int_0^{2k} \frac{(s+k)(s-k)}{-k^2} ds \, f^{n+1} = \underline{\underline{\frac{-2k}{3} f^{n+1}}}$$

$$\int_{t_{n+1}}^{t_{n+3}} L_2(t) \, dt \, f^{n+2} = \int_{t_{n+1}}^{t_{n+3}} \frac{(t - t_n)(t - t_{n+1})}{2k^2} dt \, f^{n+2}$$

$$= \int_0^{2k} \frac{(s+k)s}{2k^2} ds \, f^{n+2} = \frac{7k}{3} f^{n+2}$$

So

$$\int_{t_{n+1}}^{t_{n+3}} f(t, u(t))\,dt = \frac{k}{3} f^{n} - \frac{2k}{3} f^{n+1} + \frac{7k}{3} f^{n+2}$$

therefore

$$U(t_{n+3}) = U(t_{n+1}) + \frac{k}{3} f^{n} - \frac{2k}{3} f^{n+1} + \frac{7k}{3} f^{n+2}$$

$$U^{n+3} = U^{n+1} + \frac{k}{3}\left(7 f^{n+2} - 2 f^{n+1} + f^{n}\right)$$

b) Draw the stencil for this method.

	U	f
t_{n+3}	$\square$	
t_{n+2}		$\beta_2 = 7/3$
t_{n+1}	$\alpha_1 = -1$	$\beta_1 = -2/3$
t_n		$\beta_0 = 1/3$

c) Determine if this method is zero-stable.

For a method to be zero-stable, $|w| \leq 1$, so

$$\rho(w) = w^3 - w = 0$$

$$w(w^2 - 1) = 0$$

$$w = 0, \quad w = \pm 1$$

Its clear that $|w| \leq 1$, hence the method is zero stable.

d) Determine if this method is consistent.

For a method to be consistent $\rho(1)=0$ and $\rho'(1) = \sigma(1)$.

So

$$\sigma(w) = \tfrac{7}{3}w^2 - \tfrac{2}{3}w + \tfrac{1}{3}$$

$$\sigma(1) = 2$$

$$\rho(w) = w^3 - w \Rightarrow \rho'(w) = 3w^2 - 1$$

$$\rho'(1) = 3 - 1 = 2$$

So ~~$\sigma(w) =$~~ $\sigma(1) = \rho'(1)$ and $\rho(1) = 0$, hence it is consistent.

e) Determine if the method converges.

from Lax theorem, if the method is both stable and consistent then it converges, therefore our method converges.

f) Determine the order of accuracy of this method.

The local truncation error is given by

$$T(t) = C_0 + C_1 u'(t) + \frac{C_2}{2} u''(t) + \frac{C_3}{3!} k^2 u'''(t) + \cdots$$

$$C_q = \frac{1}{q!}\left[\alpha_1 + 2\alpha_2 + \cdots + r\alpha_r - \frac{1}{(q-1)!}(\beta_1 + 2\beta_2 + \cdots + r^{q-1}\beta_r\right.$$

Since the method is consistent, then $C_0 = 0$ and $C_1 =$

So
$$C_2 = \tfrac{1}{2}(1-9) - \left(-\tfrac{2}{3} + \tfrac{14}{3}\right) = 0$$

$$C_3 = \tfrac{1}{6}(27-1) - \tfrac{1}{2}\left(-\tfrac{2}{3} + \tfrac{28}{3}\right) = 0$$

$$C_4 = \frac{1}{24}(-1+81) - \frac{1}{6}\left(-\tfrac{2}{3} + \tfrac{56}{3}\right) = \underline{\underline{\tfrac{1}{3}}}$$

So

$$\tau(t) = \frac{C_4}{4!} k^3 u^{(iv)}(t) + \frac{C_5}{5!} k^4 u^{(v)}(t) + \cdots$$

$$\tau(t) = \frac{1}{72} k^3 u^{(iv)}(t) + \frac{C_5}{5!} k^4 u^{(v)}(t) + \cdots$$

therefore the order of Convergence, $\underline{\underline{p = 3}}$

2. Heat equation: Crank-Nicolson.

a) $u_t = \alpha u_{xx}, \quad 0 \le x \le 1, \quad t \geq 0$

$u(x,0) = f(x), \quad u(0,t) = g_0(t), \; u(1,t) = g_1(t)$

~~u_t~~ Using forward difference for u_t ~~and u_{xx}~~

$$u_t = \frac{u_j^{n+1} - u_j^n}{\Delta t}$$

also

$$u_{xx} = \frac{\nabla u_j^{n+1} + \nabla u_j^n}{2}$$

$$u_t = \alpha u_{xx}$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\alpha}{2}\left[\nabla u_j^{n+1} + \nabla_j u_j^n\right]$$

$$= \frac{\alpha}{2h^2}\left[u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1} + u_{j-1}^n - 2u_j^n + u_{j+1}^n\right]$$

$$U_j^{n+1} - U_j^n = \frac{\Delta t \alpha}{2h^2}\left[U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1} + U_{j-1}^n - 2U_j^n + U_{j+1}^n\right]$$

let $r = \frac{\Delta t \alpha}{2h^2}$

$$-rU_{j-1}^{n+1} + (1+2r)U_j^{n+1} - rU_{j+1}^{n+1} = rU_{j-1}^n + (1-2r)U_j^n + rU_{j+1}^n$$

$$\begin{bmatrix} 1+2r & -r & & & 0 \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ 0 & & -r & 1+2r & -r \\ & & & -r & 1+2r \end{bmatrix} U^{n+1} = \begin{bmatrix} d_1 \\ \vdots \\ \vdots \\ d_m \end{bmatrix}$$

where

$$d_1 = rU_0^n + (1-2r)U_1^n + rU_2^n + rU_0^{n+1}$$

$$d_m = rU_{m-1}^n + (1-2r)U_m^n + rU_{m+1}^n + rU_{m+1}^{n+1}$$

$$d_j = rU_{j-1}^n + (1-2r)U_j^n + rU_{j+1}^n, \quad j = 2, 3, \ldots, m-1$$

3. Heat equation: BDF2

The BDF2 is given by

$$u'(t) = f(t, u(t))$$

$$U^{n+2} = \frac{4}{3}U^{n+1} - \frac{1}{3}U^n + \frac{k^2}{3}f^{n+2}$$

$$U^{n+1} = \frac{4}{3}U^n - \frac{1}{3}U^{n-1} + \frac{2}{3}kf^{n+1}$$

$$\tfrac{3}{2}U^{n+1} - 2U^{n} + \tfrac{1}{2}U^{n-1} = kf^{n+1}$$

Given $U_t = \alpha U_{xx}$,

then

$$U_t = \frac{\tfrac{3}{2}U_j^{n+1} - 2U_j^{n} + \tfrac{1}{2}U_j^{n-1}}{k}$$

So

$$\frac{\tfrac{3}{2}U_j^{n+1} - 2U_j^{n} + \tfrac{1}{2}U_j^{n-1}}{k} = \alpha \nabla U_j^{n+1}$$

$$\left(1 - \frac{2\alpha k}{3}\nabla\right)U_j^{n+1} = \frac{4}{3}U_j^{n} - \frac{1}{3}U_j^{n-1}$$

$$U_j^{n+1} - \frac{2\alpha k}{3}\nabla U_j^{n+1} = \frac{4}{3}U_j^{n} - \tfrac{1}{3}U_j^{n-1}$$

$$U_j^{n+1} - \frac{2\alpha k}{3}\left[U_j^{n+1} - 2U_j + U_{j+1}^{n+1}\right] = \tfrac{4}{3}U_j^{n} - \tfrac{1}{3}U_j^{n-1}$$

take $r = \dfrac{2\alpha k}{3}$

$$-rU_{j-1}^{n+1} + (1+2r)U_j^{n+1} - rU_{j+1}^{n+1} = \frac{4}{3}U_j^{n} - \tfrac{1}{3}U_j^{n-2}$$

4. Linear Stability analysis

$$U_t + U_{xxx} = 0 \quad , \quad 0 \le x \le 1, \quad t \geq 0$$

IC: $u(x,0) = g(x)$, BC: periodic

$$\frac{u(x,t+k) - u(x,t)}{k} + \frac{-\frac{1}{2}u(x-2h,t) + u(x-h,t) - u(x+h,t) + \frac{1}{2}u(x+2h,t)}{h^3} =$$

$$U_j^{n+1} - U_j^n = \frac{k}{h^3}\left[+\frac{1}{2}U_{j-2}^n - U_{j-1}^n + U_{j+1}^n + -\frac{1}{2}U_{j+2}^n\right]$$

let $k/h^3 = r$

$$U_j^{n+1} - U_j^n = r\left[U_{j+1}^n - U_{j-1}^n + \frac{1}{2}\left(U_{j-2}^n - U_{j+2}^n\right)\right] \quad \text{—— (1)}$$

Using von Neuman stability analysis

$$U_j^n = \varepsilon^n e^{ikj\Delta x}$$

Therefore equation (1) becomes

$$\varepsilon^{n+1} e^{ikj\Delta x} - \varepsilon^n e^{ikj\Delta x} = r\left[\varepsilon^n e^{ik(j+1)\Delta x} - \varepsilon^n e^{ik(j-1)\Delta x} + \frac{1}{2}\left(\varepsilon^n e^{ik(j-2)\Delta x} - \varepsilon^n e^{ik(j+2)\Delta x}\right)\right]$$

$$\varepsilon - 1 = r\left[e^{ik\Delta x} - e^{-ik\Delta x} + \frac{1}{2}\left(e^{-2ik\Delta x} - e^{2ik\Delta x}\right)\right]$$

but
$$e^{ik\Delta x} - e^{-ik\Delta x} = 2i\sin(k\Delta x)$$
$$e^{-2ik\Delta x} - e^{2ik\Delta x} = -2i\sin(2k\Delta x)$$

So
$$\varepsilon - 1 = r\left(2i\sin k\Delta x - i\sin(2k\Delta x)\right)$$

$$\mathcal{E}-1 = ir\left(2\sin(k\Delta x) - \sin(2k\Delta x)\right)$$

$$\mathcal{E} = 1 + ir\left(2\sin(k\Delta x) - \sin(2k\Delta x)\right)$$

but $\sin(2k\Delta x) = 2\sin k\Delta x \cos k\Delta x$

$$\mathcal{E} = 1 + 2ir\sin(k\Delta x)\left(1-\cos(k\Delta x)\right)$$

$$\mathcal{E} = 1 - 2ir\sin(k\Delta x)\left(\cos(k\Delta x) - 1\right)$$

$$|\mathcal{E}| = \left|1 - 2ir\sin(k\Delta x)\left(\cos(k\Delta x) - 1\right)\right|$$

$$|\mathcal{E}| = 1 + 4r^2\sin^2(k\Delta x)\left(\cos(k\Delta x) - 1\right)^2$$

since $0 \leq \sin^2(k\Delta x) \leq 1$ and that

$4r^2\sin^2(k\Delta x)\left(\cos(k\Delta x) - 1\right)^2 > 0$

then

$|\mathcal{E}| = 1 +$ positive number > 1

$|\mathcal{E}| > 1$

So, since $|\mathcal{E}| > 1$, then the scheme is unconditionally unstable. thus should never be used, since it will never converge.

b) $U_t + U_{xxx} = 0$

$$U_t = -U_{xxx}$$

$$U_t = \frac{U_j^{n+1} - U_j^n}{k} = \frac{U_j^{n+1} - U_j^n}{\Delta t}$$

$$U_{xx} = \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}$$

$$U_{xx} = \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}$$

$$U_{xxx} = \frac{\partial}{\partial x}(U_{xx}) = \frac{1}{\Delta x^2}\left(\frac{\partial}{\partial x}U_{j+1}^n - 2\frac{\partial}{\partial x}U_j^n + \frac{\partial}{\partial x}U_{j-1}^n\right)$$

$$U_{xxx} = \frac{1}{\Delta x^2}\left(\frac{U_{j+2}^n - U_{j+1}^n}{\Delta x} - 2\left(\frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}\right) + \left(\frac{-U_{j-2}^n + U_{j-1}^n}{\Delta x}\right)\right)$$

$$U_{xxx} = \frac{1}{\Delta x^3}\left(U_{j+2}^n - U_{j+1}^n - U_{j+1}^n + U_{j-1}^n - U_{j-2}^n + U_{j-1}^n\right)$$

$$\cancel{= \frac{1}{\Delta x^3}\left(U_{j+2}^n - 2U_{j+1}^n + U_{j-2}^n\right)}$$

$$U_{xxx} = \frac{1}{\Delta x^3}\left(U_{j+2}^n - 2U_{j+1}^n + 2U_{j-1}^n - U_{j-2}^n\right)$$

Therefore $U_t = -U_{xxx}$

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = -\left(\frac{1}{\Delta x^3}\left(U_{j+2}^n - 2\left(U_{j+1}^n - U_{j-1}^n\right) - U_{j-2}^n\right)\right)$$

$$U_j^{n+1} - U_j^n = -\frac{\Delta t}{\Delta x^3}\left(U_{j+2}^n - 2\left(U_{j+1}^n - U_{j-1}^n\right) - U_{j-2}^n\right)$$

let $\frac{\Delta t}{\Delta x^3} = r$,

Using Von Neumann stability, let $U_j^n = \varepsilon^n e^{ikj\Delta x}$

$$\varepsilon^{n+1}e^{ikj\Delta x} - \varepsilon^n e^{ikj\Delta x} = -r\left(\varepsilon^n e^{ik(j+2)\Delta x} - 2\left(\varepsilon^n e^{ik(j+1)\Delta x} - \varepsilon^n e^{ik(j-1)\Delta x}\right) - \varepsilon^n e^{ik(j-2)\Delta x}\right)$$

$$\mathcal{E}-1 = -r\left(e^{2ik\Delta x} - 2\left(e^{ik\Delta x} - e^{-ik\Delta x}\right) - e^{-2ik\Delta x}\right)$$

$$\mathcal{E} = 1 - r\left(2i\sin(2k\Delta x) - 2\left(2i\sin(k\Delta x)\right)\right)$$

$$\mathcal{E} = 1 - 2ir\left(\sin 2k\Delta x - 2\sin k\Delta x\right)$$

$$\mathcal{E} = 1 - 2ir\left(2\cos k\Delta x \sin k\Delta x - 2\sin k\Delta x\right)$$

$$\mathcal{E} = 1 - 2ir\left(\cos k\Delta x - 1\right)\left(2\sin k\Delta x\right)$$

$$\mathcal{E} = 1 + 2ir\left(1 - \cos k\Delta x\right)\left(2\sin k\Delta x\right)$$

$$|\mathcal{E}| = \left|1 + 2ir\left(1-\cos k\Delta x\right)\left(2\sin k\Delta x\right)\right|$$

$$|\mathcal{E}| = 1 - 4r^2\left(1-\cos k\Delta x\right)^2\left(4\sin^2 k\Delta x\right)$$

Since $0 \leq \sin^2 k\Delta x \leq 1$ and that

$$4r^2\left(1-\cos k\Delta x\right)^2\left(4\sin^2 k\Delta x\right) > 0$$

$$|\mathcal{E}| = 1 - \text{positive value} < 1$$

$$|\mathcal{E}| < 1$$

Hence the Scheme is Conditionally Stable therefor

I propose this as an alternative Scheme.

$$U_t + U_{xxx} = 0$$

$$U_j^{n+1} - U_j^n = -r\left(U_{j+2}^n - 2\left(U_{j+1}^n - U_{j-1}^n\right) - U_{j-2}^n\right)$$

where $r = \dfrac{\Delta t}{\Delta x^3}$

$$u_j^{n+1} - u_j^n + r\left(u_{j+2}^n - 2\left(u_{j+1}^n - u_{j-1}^n\right) - u_{j-2}^n\right) = 0$$

$$u(x, t+k) - u(x,t) + r\Big[u(x+2h,t) - 2\big(u(x+h,t) - u(x-h,t)\big) - u(x-2h,t)\Big] = 0$$

$$\frac{u(x,t+k) - u(x,t)}{\Delta t} + \frac{u(x+2h,t) - 2\big(u(x+h,t) - u(x-h,t)\big) - u(x-2h,t)}{\Delta x^3} = 0$$

$$0 \leq x \leq 1, \quad t \geq 0$$

IC: $u(x,0) = g(x)$, BC: periodic

5. Wave equation

a) Show that the two-way wave equation

$$u_{tt} = c^2 u_{xx}, \quad 0 \le x < 2\pi, \; t > 0, \; u(x,0) = f(x),$$

$$u_t(x,0) = g(x),$$

Can be transformed into

$$u_t + v_x = 0$$

$$v_t + c^2 u_x = 0$$

let $u_t = -v_x$

$$u_{tt} = -\frac{\partial}{\partial t}(v_x) = c^2 u_{xx} = \frac{\partial}{\partial x}(c^2 u_x)$$

$$\cancel{u_{tt}} = \frac{\partial}{\partial t}(\cancel{v_t}), \; \cancel{so} \quad \& \quad u_{tt} = \frac{\partial}{\partial t}(u_t)$$

also $\cancel{\frac{\partial}{\partial x}(u_t)} = -\cancel{\frac{\partial}{\partial x}(v_x)}$

$$-\frac{\partial}{\partial x}(v_t) = \frac{\partial}{\partial x}(c^2 u_x)$$

$$-v_t = c^2 u_x \Rightarrow v_t = -c^2 u_x$$

hence

$$u_t + v_x = 0$$

$$v_t + c^2 u_x = 0$$

Given $q_t + A q_x = 0 \Rightarrow q_t = -A q_x$

from $u_t + v_x = 0$ and $v_t + c^2 u_x = 0$

$$A = \begin{pmatrix} 0 & 1 \\ c^2 & 0 \end{pmatrix}$$

$$q_t = F(q_x),$$

let $d_1 = q_j^n$

$$d_2 = q_j^n + \frac{k}{2}F(d_1) = q_j^n - \frac{k}{2}A(q_x)_j^n$$

$$d_3 = q_j^n + \frac{k}{2}F(d_2) = q_j^n + \frac{k}{2}F\left(q_j^n - \frac{k}{2}A(q_x)_j^n\right)$$

$$d_3 = \left(1 - \frac{kA}{2}\right)q_j^n + \frac{k^2A^2}{4}(q_x)_j^n$$

$$d_4 = q_j^n + \frac{k}{2}F(d_3)$$

$$d_4 = \left(1 - kA + \frac{k^2A^2}{2}\right)q_j^n - \frac{k^3A^3}{4}(q_x)_j^n$$

from

$$q_j^{n+1} = q_j^n + \frac{k}{6}\left(F(d_1) + 2F(d_2) + 2F(d_3) + F(d_4)\right)$$

then

$$F(d_1) = F(q_j^n) = -Aq_j^n$$

$$F(d_2) = F\left(q_j^n - \frac{k}{2}A(q_x)_j^n\right) = -A\left(q_j^n - \frac{kA}{2}(q_x)_j^n\right)$$

$$F(d_3) = -\left(A - \frac{kA^2}{2}\right)q_j^n - \frac{k^2A^2}{4}(q_x)_j^n$$

$$F(d_4) = -\left(A - kA^2 + \frac{k^2A^3}{2}\right)q_j^n - \frac{k^3A^4}{4}(q_x)_j^n$$

from

$$q_j^{n+1} = q_j^n + \frac{k}{6}\left(F(d_1) + 2F(d_2) + 2F(d_3) + F(d_4)\right)$$

$$q_j^{n+1} = q_j^n + \frac{k}{6}\Big[-Aq_j^n + 2\Big(-Aq_j^n + \frac{kA^2}{2}(q_x)_j^n\Big) + 2\Big(-\Big(A - \frac{kA^2}{2}\Big)q_j^n$$

$$- \frac{k^2A^2}{4}(q_x)_j^n\Big) + -\Big(A - kA^2 + \frac{k^2A^3}{2}\Big)q_j^n - \frac{k^3A^4}{4}(q_x)_j^n$$

$$q_j^{n+1} = q_j^n + \frac{k}{6}\Big[-3Aq_j^n + kA^2(q_x)_j^n - 2Aq_j^n + kA^2q_j^n - \frac{k^2A^2}{2}(q_x)$$

$$-Aq_j^n + kA^2q_j^n - \frac{k^2A^3}{2}q_j^n - \frac{k^3A^4}{4}(q_x)_j^n\Big]$$

$$q_j^{n+1} = q_j^n + \frac{k}{6}\Big[-6Aq_j^n - kA^2(q_x)_j^n + 2kA^2q_j^n - \frac{k^2A^2}{2}(q_x)_j^n$$

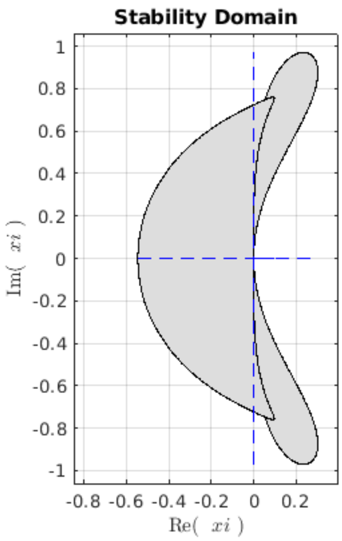
$$- \frac{k^2A^3}{2}q_j^n - \frac{k^3A^4}{4}(q_x)_j^n\Big]$$

$$q_j^{n+1} = q_j^n + \frac{k}{6}\Big[6A + 3kA^2 + k^2A^3 + \frac{k^3A^4}{4}\Big]q_j^n$$

```
clear all;
close all;

%Setting up variables for plotting purposes.
LW =  'LineWidth' ;
lw  =  1;
clr  =  [221  221  221]/255;
xlbl  =  'Re( $\ xi$ )';
ylbl  =  'Im( $\ xi$ )';
intrptr  =  'Interpreter';
ltx  =  'Latex';

%Stability domain for the method.
% Define  the  unit  circle  in  the  complex  plane
N =  1000;
th = linspace(0,2*pi,N);
w = exp(1i*th);

%solution of the characteristic equation in terms of $\xi$
f=@(w) 3*w.*((w.^2)-1)./(7*(w.^2)-2*w+1);
g=@(w) 12*(w.^3 - w.^2)./(23*w.^2 - 16*w + 5);

%Evaluate $f$ at the points on the unit circle and then plot the results:
xi = f(w);

plot(xi, 'k-', LW,lw), hold on
fill (real(xi), imag(xi), clr)
plot([min(real(xi)) max(real(xi))],[0 0],'b--',LW,lw)
plot([0 0], [min(imag(xi)) max(imag(xi))],'b--',LW,lw)
xlabel(xlbl,intrptr, ltx), ylabel(ylbl, intrptr,ltx)
xlim([min(real(xi))-0.3 max(real(xi))+0.3])
ylim([min(imag(xi))-0.3 max(imag(xi))+0.3])

%AB3
xii = g(w);
plot(xii, 'k-', LW,lw), hold on
fill (real(xii), imag(xii), clr)
plot([min(real(xii)) max(real(xii))],[0 0],'b--',LW,lw)
plot([0 0], [min(imag(xii)) max(imag(xii))],'b--',LW,lw)
xlabel(xlbl,intrptr, ltx), ylabel(ylbl, intrptr,ltx)
xlim([min(real(xii))-0.3 max(real(xii))+0.3])
ylim([min(imag(xii))-0.3 max(imag(xii))+0.3])

title('Stability Domain')
grid on

daspect([1 1 1]), hold off

%check for the root condition at a point inside and outside the apperent
%domain.

%compare
xii = 0.2 + 0.8*1i; %inside
xio = 0.2 - 0.4*1i; %outside
coeffii = [1 -7/3*xii (-1+2/3*xii) -xii/3];
coeffio = [1 -7/3*xio (-1+2/3*xio) -xio/3];

ep1=abs(roots(coeffii))
ep2=abs(roots(coeffio))

%for AB3
xiiA = 0.2 + 0.2*1i; %inside
xioA = -0.4 - 0.6*1i; %outside
coeffiiA = [12 (-1-23)*xiiA (16*xiiA) -5*xiiA];
coeffioA = [12 (-1-23)*xioA (16*xioA) -5*xioA];
ep1A=abs(roots(coeffiiA))
ep2A=abs(roots(coeffioA))

%intersection between the two domains
xis = 0.09534 + 0.7597*1i;
xio = 0.2 - 0.4*1i; %outside
coeffiis = [1 -7/3*xis (-1+2/3*xis) -xis/3];
coeffio = [1 -7/3*xio (-1+2/3*xio) -xio/3];

ep1s=abs(roots(coeffiis))
ep2s=abs(roots(coeffio))

fprintf('Compare and Contrast\n')
fprintf('Most of the region of the stability domain for AB3 lines in the negative real part of x and both in the negative and \n positive imagina

fprintf('Would you ever want to use this method?\n');

fprintf('I would never want to use this method because checking for root condition at the point\n inside to and outside to the apparent domain, a
```

```
ep1 =

    1.1710
    1.1915
    0.1970


ep2 =

    1.2252
```

```
    0.8707
    0.1397


ep1A =

    0.7233
    0.4436
    0.3673


ep2A =

    1.8750
    0.4071
    0.3936


ep1s =

    1.3172
    1.0454
    0.1853


ep2s =

    1.2252
    0.8707
    0.1397

Compare and Contrast
Most of the region of the stability domain for AB3 lines in the negative real part of x and both in the negative and
 positive imaginary part of x, while for the other LMS method, the satbility domain lies in the positive real part of x and also both in the posi
 imaginery part of x. However these two have a region in common.
Would you ever want to use this method?
I would never want to use this method because checking for root condition at the point
 inside to and outside to the apparent domain, at least one root has a modulus greater
 than one, hence the method  is unconditionally unstable for all episilon, inside and
 outside the domain.
```



Published with MATLAB® R2020a

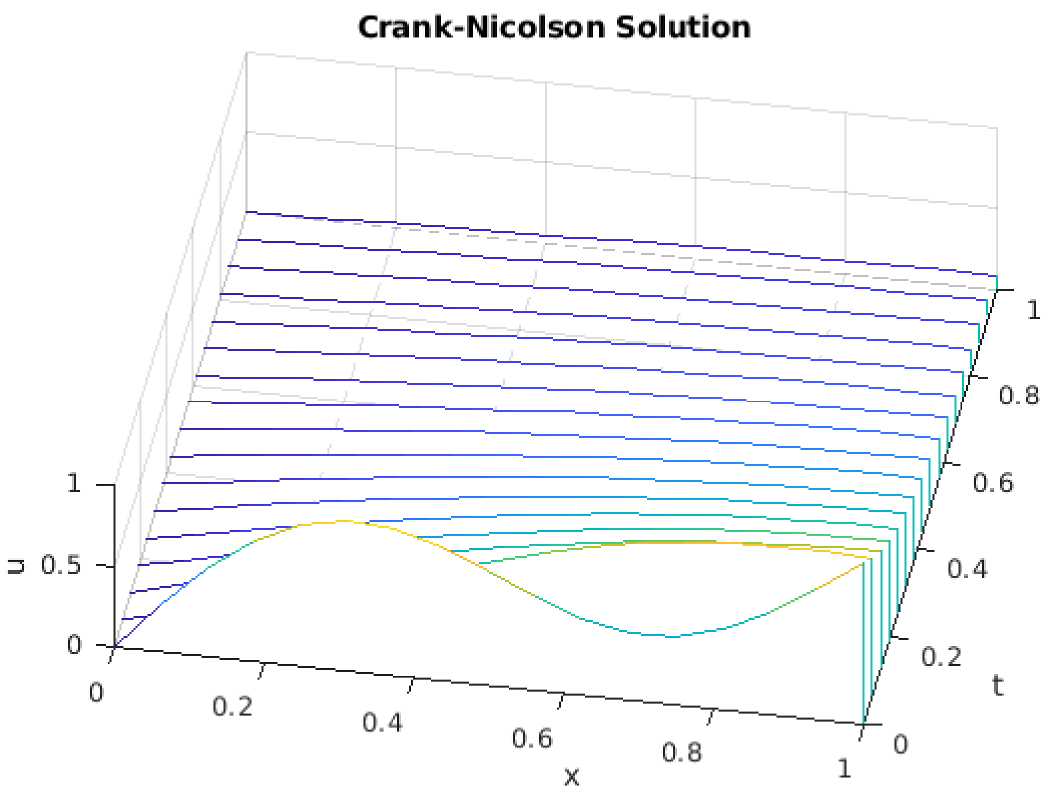
```
% This function Numerically solves the 1-D heat equation using the
% Crank-Nicolson scheme
% input
% f : function representing intial condition
% g0 and g1 : function represents boundary conditions
% tspan : time span over
% N : Number of time steps
% m : Number of points for the uniform spatial discretisation.
% Return
% u : Approximate solution at each time step (matrix)
% t : Vector containing all time steps
% x : Vector containing the spatial discretization.

function [u,t,x] = cnhteq(f,g0,g1,tspan,alp,N,m)

h = 1/(m+1);
k = tspan/N;
r = (alp*k)/(2*h^2);
U0 = zeros(1,m+2);
U = zeros(N+1,m+2);

%at to
t= zeros(N,1); t(1) = 0;
x= zeros(m+2,1);

for i = 1:m+2
x(i) = (i-1)*h;
U0(i) = f(x(i));
end

U(1,:) = U0;

%BC
gl = zeros(m,1); %left
gr = zeros(m,1); %Right

a1 = (1+2*r);
a2 = (1-2*r);
b = r;

% Using sparse library to transform A
A = sparse(toeplitz([a1 -b zeros(1, m-2)]));
B = sparse(toeplitz([a2 b zeros(1, m-2)]));


for i = 1:N
    t(i+1) = i*k;
    %BC
    gl(1) = g0(t(i+1));
    gr(1) = g0(t(i));
    gl(m) = g1(t(i+1));
    gr(m) = g1(t(i));
    %interior
    e = U0(2:m+1)';
    d = B*e + r*(gl + gr);
    %solving the linear system
    U1 = A\d;
    %Treating bounary values
    Un = [gl(1), U1', gl(m)];
    U(i+1,:) = Un;
    U0 = Un;

end
```

```
u = U;
end
```

```
Not enough input arguments.

Error in cnhteq (line 17)
h = 1/(m+1);
```

Published with MATLAB® R2020a

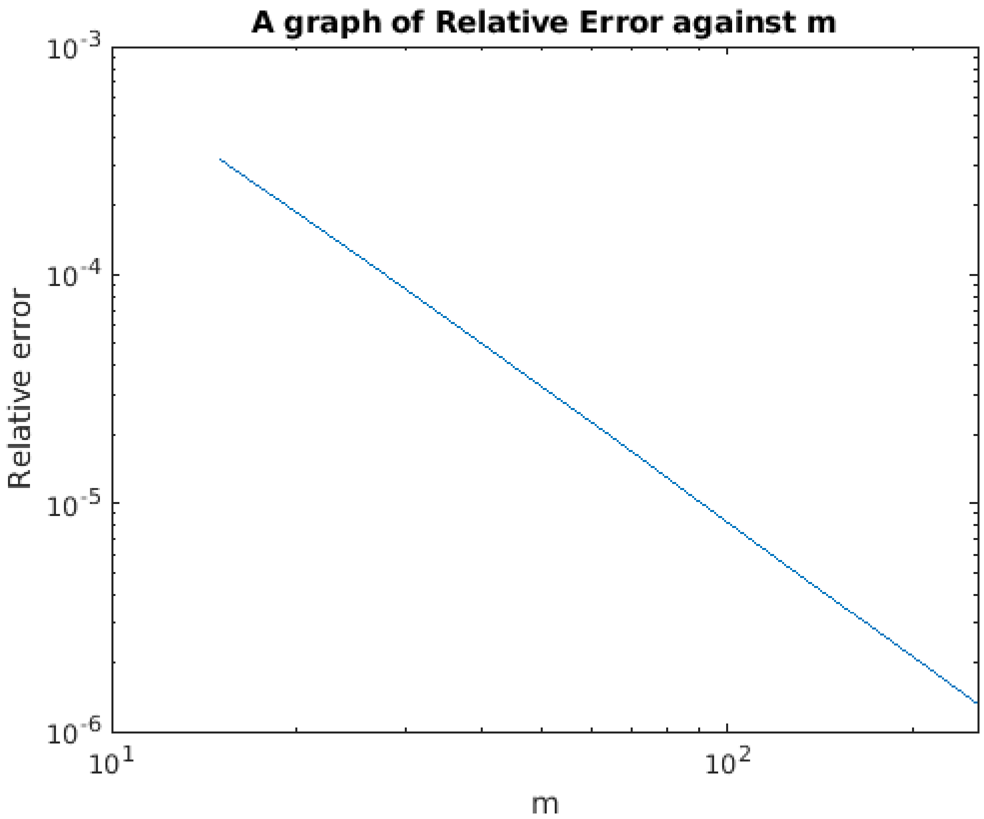
```
% Uses a waterfall  plot to plot the Crank-Nicolson solution

clear all;
close all;

N = 16;
m = 15;
alp = 1;
tspan = 1;

f = @(x) sin(pi*x/2) + 0.5*sin(2*pi*x);

g0 = @(t) 0;
g1 = @(t) exp((-pi^2*t)/4);

%Numerical solution
[u,t,x] = cnhteq(f,g0,g1,tspan,alp,N,m);

waterfall(x,t,u), view(10,70)
axis([0 1 0 1 0 1] ), xlabel x, ylabel t, zlabel u
title('Crank-Nicolson Solution');
```



Published with MATLAB® R2020a

```
clear all;
close all;

f = @(x) sin(pi*x/2) + 0.5*sin(2*pi*x);

g0 = @(t) 0;
g1 = @(t) exp((-pi^2*t)/4);

%Exact solution
uexact = @(x,t) exp(-(pi^2)*t/4).*sin((pi*x)/2) + 0.5*exp(-(4*pi^2)*t).*sin(2*pi*x);

alp = 1;
tspan = 1;
n = 4:8;
m = 2.^n - 1;
N = 2.^n;

Re_Err = zeros(5,1);

for i =1:5

    %Numerical solution
    [u,t,x] = cnhteq(f,g0,g1,tspan,alp,N(i),m(i));
    u = u(end,:);
    %exact solution
    uex = uexact(x,1)';
    Re_Err(i) = RelNorm(u,uex);

end

loglog(m,Re_Err);
title('A graph of Relative Error against m');
xlabel('m');
ylabel('Relative error');

p = polyfit(log(m),log(Re_Err),1);
fprintf('The order of accuracy is  %f \n', p(1));

function L2 = RelNorm(U,Uexact)
error = (U - Uexact).^2;
L2 = sqrt(sum(error)/sum(Uexact.^2));
end
```

```
The order of accuracy is  -1.948416
```



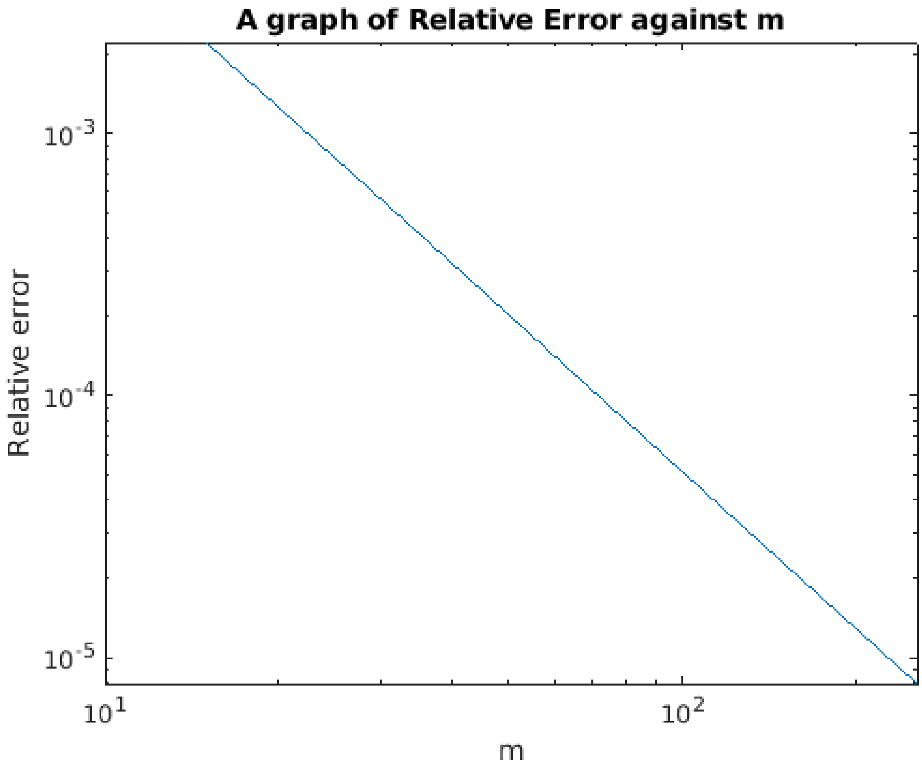
Published with MATLAB® R2020a

```matlab
% The program uses BDF2 for the time integratorl instead of the trapezoidal
% Rule.

% f : function representing intial condition
% g0 and g1 : function represents boundary conditions
% tspan : time span over
% N : Number of time steps
% m : Number of points for the uniform spatial discretisation.
% Return
% u : Approximate solution at each time step (matrix)
% t : Vector containing all time steps
% x : Vector containing the spatial discretization.% This function Numerically solves the 1-D heat equation using the
% Crank-Nicolson scheme
% input


function [u,t,x] = BDF2(f,g0,g1,tspan,alp,N,m)

h = 1/(m+1);
k = tspan/N;
r = (2*alp*k)/(3*h^2);

%space and time steps
x = 0:h:1;
t = 0:k:tspan;

%Crank-Nicolson
u1 = cnhteq(f,g0,g1,tspan,alp,N,m);

U = zeros(m+2,m+2);
gl = zeros(m,1);

U0 = f(x);
U(1,:) = U0;
U1 = u1(2,:);
U(2,:) = U1;

%Matrix Coefficients
a= (1+2*r);
b = r;

% Using the sparse library to transform A
A = sparse(toeplitz([a -b zeros(1, m-2)]));

for n = 3: N+1

    gl(1) = g0(t(n));
    gl(m) = g1(t(n));

    Un1 = U0(2:m+1)';
    Un2= U1(2:m+1)';

    % The Right Hand Side
    B = (4/3)*Un2 - (1/3)*Un1 + r*gl;

    % Solving the linear sysytem for the interior points

    Un = A\B;

    % Treating the boundary conditions
    Un = [gl(1), Un', gl(m)];
    U(n,:) = Un;

    %update solution
    U0 = U1;
    U1 = Un;

end
u = U;
end
```

```
Not enough input arguments.

Error in BDF2 (line 19)
h = 1/(m+1);
```

```
clear all;
close all;

f = @(x) sin(pi*x/2) + 0.5*sin(2*pi*x);

g0 = @(t) 0;
g1 = @(t) exp((-pi^2*t)/4);

%Exact solution
uexact = @(x,t) exp(-(pi^2)*t/4).*sin((pi*x)/2) + 0.5*exp(-(4*pi^2)*t).*sin(2*pi*x);

alp = 1;
tspan = 1;
n = 4:8;
m = 2.^n - 1;
N = 2.^n;

Re_Err = zeros(5,1);

for i =1:5

    %Numerical solution
    [u,t,x] = BDF2(f,g0,g1,tspan,alp,N(i),m(i));
    u = u(end,:);
    %exact solution
    uex = uexact(x,1);
    Re_Err(i) = RelNorm(u,uex);

end

loglog(m,Re_Err);
title('A graph of Relative Error against m');
xlabel('m');
ylabel('Relative error');

p = polyfit(log(m),log(Re_Err),1);
fprintf('The order of accuracy is  %f \n', p(1));
fprintf('The BDF2 is more accurate since it converges faster than the Trapezoidal Time integrator\n');

function L2 = RelNorm(U,Uexact)
error = (U - Uexact).^2;
L2 = sqrt(sum(error)/sum(Uexact.^2));
end
```

```
The order of accuracy is  -1.990486
The BDF2 is more accurate since it converges faster than the Trapezoidal Time integrator
```



Published with MATLAB® R2020a

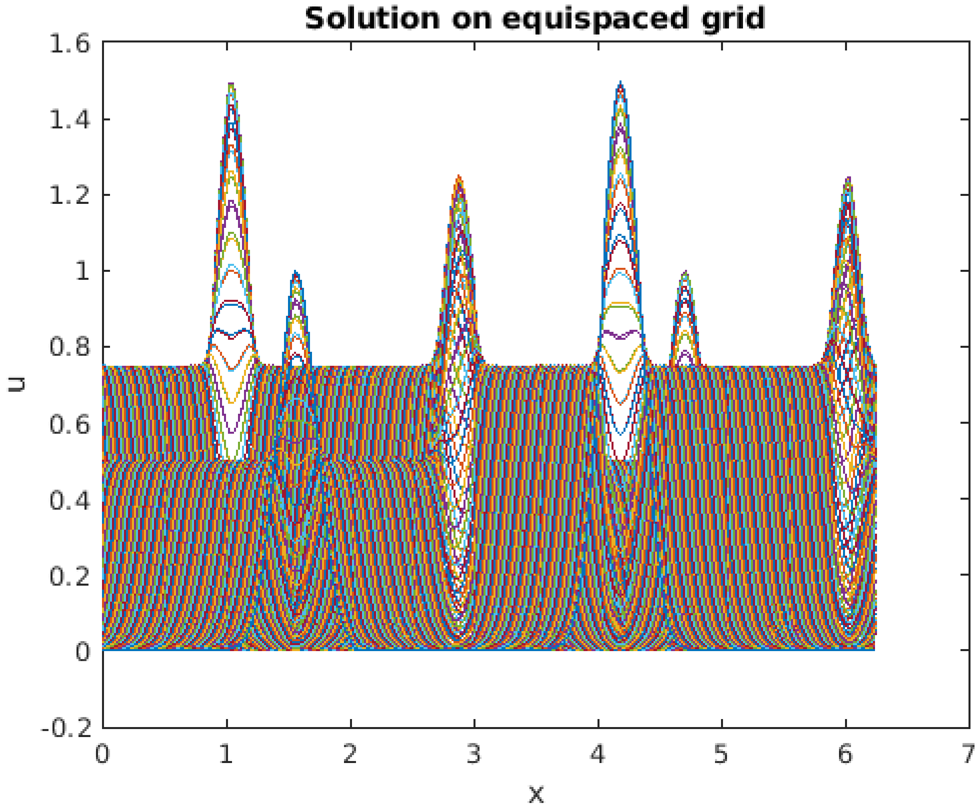
```
% The function numerically solves (2) using fourth-order centered finite differences
% in space and the standard fourth order Runge-Kutta (RK4) method in time.

function [u,v,x,t] = wave(f,g,m,c)

Nt = 400;
h = 2*pi/(m+1);
k = 2*pi/Nt;

x = 0:h:2*pi-h;
j = 0:m; t = j*k;

D1 = (1/(12*h))*(circulant([0,8,-1,zeros(1,m-4),1,-8],1));
A = [zeros(m+1) -D1; -(c^2)*D1 zeros(m+1)];
A = sparse(A);
N = size(A,1);

u0 = f(x);
v0 = g(x);

w0 = [u0,v0];

U = zeros(Nt,m+1); U(1,:) = u0;
V = zeros(Nt,m+1); V(1,:) = v0;

for i = 1:Nt

   wn = w0' + (k/6)*((6*A)+(3*k*A^2)+(k^2*A^3)+(k^3*A^4)/4)*w0';
   u0 = wn(1:N/2)';
   v0 = wn((N/2)+1:end)';
   U(i,:) = u0; V(i,:) = v0;
   w0 = [u0,v0];

end
u=U;
v=V;
end
```

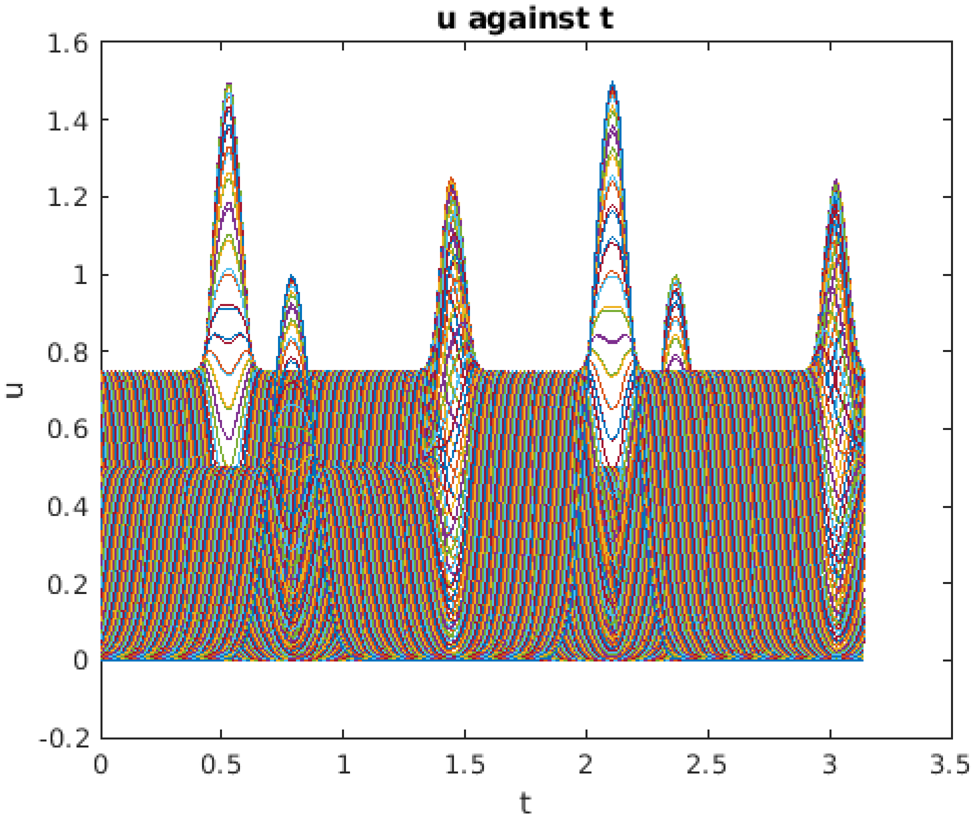
```
Not enough input arguments.

Error in wave (line 7)
h = 2*pi/(m+1);
```

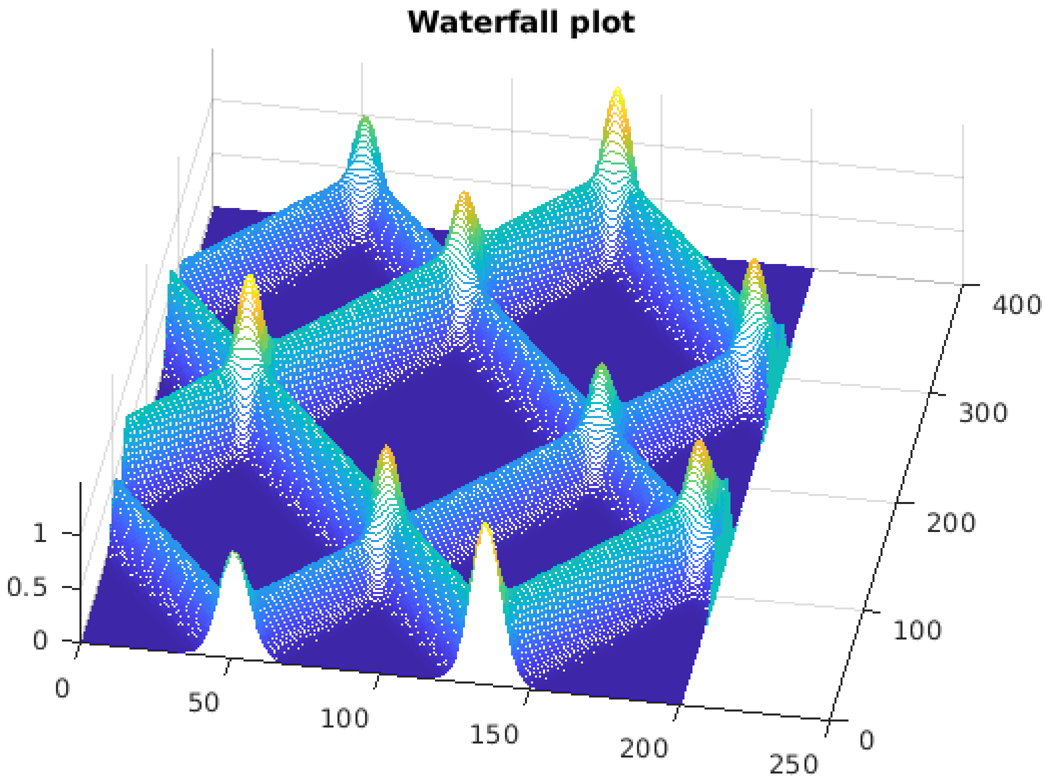
Published with MATLAB® R2020a

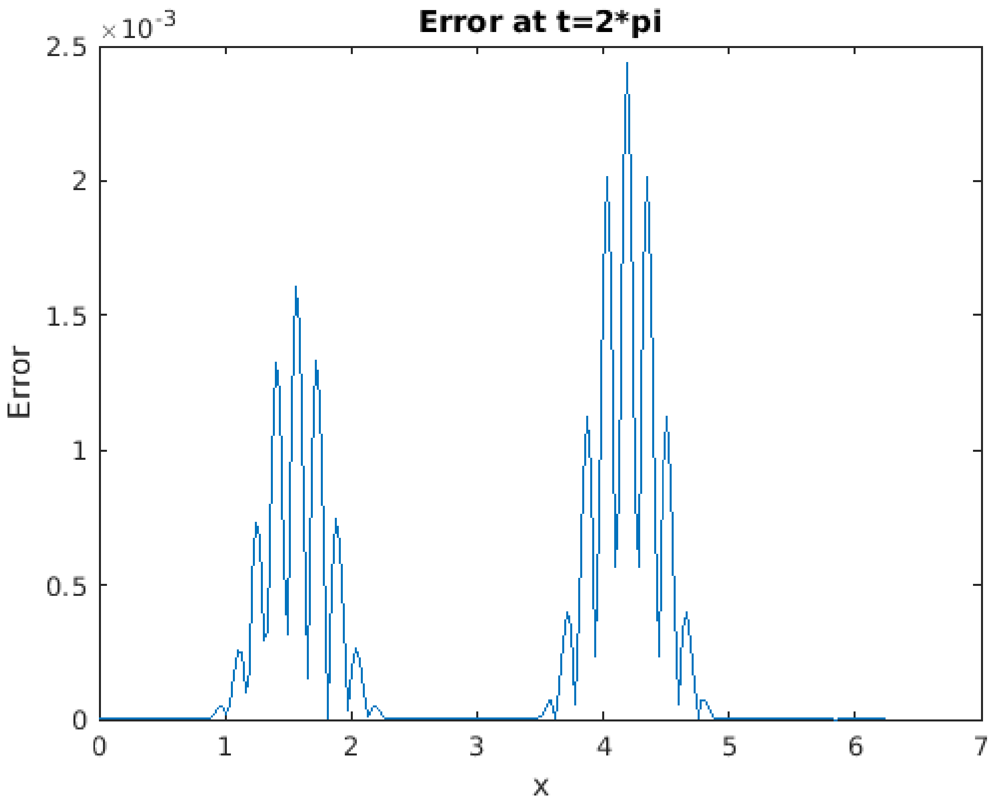
```
close all
clear all

f = @(x) exp(-20*(x-pi/2).^2) + (3/2)*exp(-20*(x-4*pi/3).^2);
g = @(x) 0*x;

m = 200; c = 1;

[u,v,x,t] = wave(f,g,m,c);

figure(1);
plot(x,u);
title('Solution on equispaced grid');
xlabel('x'); ylabel('u');

figure(2);
plot(t,u);
title('u against t');
xlabel('t'); ylabel('u');

figure(3)
waterfall(u), view(10,70)
%axis([0 1 0 1 0 1] ), xlabel x, ylabel t, zlabel u
title('Waterfall plot');



%init condition
un = u(end,:);
uexact = f(x);

%error
error = abs(un - uexact);
errorn = max(error)

figure(4);
plot(x,error);
title('Error at t=2*pi');
xlabel('x'); ylabel('Error');
```

```
errorn =

    0.0024
```


Solution on equispaced grid
u
x
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0 1 2 3 4 5 6 7


u against t
u
t
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0 0.5 1 1.5 2 2.5 3 3.5





Published with MATLAB® R2020a