MTL712: Computational Methods for Differential Equations

Harshit Goyal: 2021MT10143

August 2024

Contents

1 Description

1.1 Test Case 1

Find $u(x, 30)$ where

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

with the initial condition

$$u(x, 0) = -\sin(\pi x)$$

Exact solution

$$u(x, t) = -\sin(\pi(x - t))$$

1.2 Test Case 2

Find $u(x, 4)$ where

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

with the initial condition

$$g(x) = u(x, 0) = \begin{cases} 1 & |x| < \frac{1}{3} \\ 0 & \frac{1}{3} < |x| \leq 1 \end{cases}$$

Exact Solution

$$u(x, t) = g(x - t)$$

1.3 Test Case 3

Find $u(x, 4)$ and $u(x, 40)$ where

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

with the initial condition

$$g(x) = u(x, 0) = \begin{cases} 1 & |x| < \frac{1}{3} \\ 0 & \frac{1}{3} < |x| \leq 1 \end{cases}$$

Exact Solution

$$u(x, t) = g(x - t)$$

1.4 Test Case 4

Find $u(x, 0.6)$ where

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\frac{1}{2}u^2) = 0 \tag{1}$$

with the initial condition

$$u(x,0) = \begin{cases} 1 & |x| < \frac{1}{3} \\ 0 & \frac{1}{3} < |x| \leq 1 \end{cases}$$

Exact Solution

$$u(x,t) = \begin{cases} 0 & -1 \leq x < -\frac{1}{3} \\ \frac{x+\frac{1}{3}}{t} & -\frac{1}{3} \leq x < -\frac{1}{3} + t \\ 1 & -\frac{1}{3} + t \leq x < \frac{1}{3} + \frac{t}{2} \\ 0 & \frac{1}{3} + \frac{t}{2} < x \leq 1 \end{cases}$$

1.5 Test Case 5

Find $u(x, 0.3)$ where

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\frac{1}{2}u^2) = 0 \tag{2}$$

with the initial condition

$$u(x,0) = \begin{cases} 1 & \text{for } |x| < \frac{1}{3} \\ -1 & \text{for } \frac{1}{3} < |x| \leq 1 \end{cases}$$

Exact Solution

$$u(x,t) = \begin{cases} -1 & -1 \leq x < -\frac{1}{3} - t \\ \frac{x+\frac{1}{3}}{t} & -\frac{1}{3} - t \leq x < -\frac{1}{3} + t \\ 1 & -\frac{1}{3} + t \leq x < \frac{1}{3} \\ -1 & \frac{1}{3} < x \leq 1 \end{cases}$$

2 Strategy of solving

We have to solve the partial differential equation

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$$

with some initial condition,

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

to obtain $u(x, t) \quad \forall t > 0$.

1. In all the test-cases described in section 1, the wave is periodic, so we solve it in the interval $x \in [-1, 1]$ and hence, $u(-1, t) = u(1, t) \quad \forall t > 0$.

2. We consider k equally spaced points in $[-1, 1]$ (inclusive). Hence, $\Delta x = \frac{2}{k-1}$. Denote them as $x_1, x_2...x_k$ where $x_1 = -1$ and $x_k = 1$.

3. A fixed parameter λ is given in a problem and along the t axis, we take the spacing $\Delta t = \lambda \Delta x$. Denote the numerically obtained solution by $\hat{u}(x, t)$.

4. For a given t, to obtain $\hat{u}(x, t), x \in [-1, 1]$, we do total $m = \lfloor \frac{t}{\Delta t} \rfloor$ iterations by considering equally spaced values $t_1, t_2....t_{m+1}$ along the t-axis, where $t_1 = 0$ and t_{m+1} is close to t.

5. $\hat{u}(x_1, t_1), \hat{u}(x_2, t_1)...\hat{u}(x_k, t_1)$ are known to be $g(x_1), g(x_2)...g(x_k)$ respectively, as $t_1 = 0$.

6. For each $n = 1, 2...m$, $\hat{u}(x_1, t_{n+1}), \hat{u}(x_2, t_{n+1})...\hat{u}(x_k, t_{n+1})$ are iteratively obtained from $\hat{u}(x_1, t_n), \hat{u}(x_2, t_n)...\hat{u}(x_k, t_n)$ using the rules described in section 3. Note that $\hat{u}(x_j, t_n)$ is denoted as u_j^n for $j = 1, 2...k$ and $n = 1, 2...m + 1$ there.

7. Note that, for $j = 1$, $j + 1$ is 2 and $j - 1$ is $k - 1$. For $j = k$, $j + 1$ is 2 and $j - 1$ is $k - 1$. Clearly this implies, $u_1^n = u_k^n \quad \forall n = 1, 2...m$. This is because the wave is periodic, described in the earlier point.

3 Methods

All the below symbols and their meanings have been described in section 2.

3.1 Lax-Friedrichs Method

3.1.1 Iteration Rule

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\lambda}{2}(f(u_{i+1}^n) - f(u_{i-1}^n))$$

3.1.2 MATLAB Code Implementation

```
function u_next = compute_u_next(u_minus, u_plus, lambda, f)
 u_next = 0.5 * (u_plus + u_minus) - 0.5 * lambda * (f(u_plus) - f(u_minus));
end

function u_next = compute_friedrichs(initial_solutions, t_final, f, lambda)
    u_curr = initial_solutions;
    u_next = zeros(size(u_curr));
    k = length(u_curr);
    delta_x = 2 / (k - 1);
    delta_t = lambda * delta_x;
    m = floor(t_final / delta_t);
    for t = 1: m
        for i = 2: k - 1
            u_next(i) = compute_u_next(u_curr(i - 1), u_curr(i + 1), lambda, f);
        end
        u_next(1) = compute_u_next(u_curr(k - 1), u_curr(2), lambda, f);
        u_next(k) = u_next(1);
        u_curr = u_next;
    end
end
```

Listing 1: Code for Lax-Friedrichs method

3.2 Lax-Wendroff Method

3.2.1 Iteration Rule

$$u_i^{n+1} = u_i^n - \frac{\lambda}{2}(f(u_{i+1}^n) - f(u_{i-1}^n)) + \frac{\lambda^2}{2}[a_{i+\frac{1}{2}}^n(f(u_{i+1}^n) - f(u_i^n)) - a_{i-\frac{1}{2}}^n(f(u_i^n) - f(u_{i-1}^n))]$$

where

$$a_{i+\frac{1}{2}}^n = \begin{cases} \frac{f(u_{i+1}^n) - f(u_i^n)}{u_{i+1}^n - u_i^n} & \text{for } u_i^n \neq u_{i+1}^n, \\ a(u_i^n) & \text{for } u_i^n = u_{i+1}^n. \end{cases}$$

$$a_{i-\frac{1}{2}}^n = \begin{cases} \frac{f(u_{i-1}^n) - f(u_i^n)}{u_{i-1}^n - u_i^n} & \text{for } u_i^n \neq u_{i-1}^n, \\ a(u_i^n) & \text{for } u_i^n = u_{i-1}^n. \end{cases}$$

$$a(x) = \frac{d}{dx}f(x)$$

To check for $u_i^n = u_{i-1}^n$, we test $|u_i^n - u_{i-1}^n| < 10^{-6}$ due to floating point errors.

3.2.2 MATLAB Code Implementation

```
function derivative = fdash(u1, u2, f, a)
    if abs(u1 - u2) > 1e-6
        derivative = (f(u2) - f(u1)) / (u2 - u1);
    else
        derivative = a((u1 + u2) / 2);
    end
end
function u_next = compute_u_next(u_minus, u, u_plus, lambda, f, a)
    a_plus  = fdash(u, u_plus, f, a);
    a_minus = fdash(u_minus, u, f, a);
    u_next = u - 0.5 * lambda * (f(u_plus) - f(u_minus)) ...
           + 0.5 * lambda^2 * (a_plus * (f(u_plus) - f(u)) - a_minus * (f(u) - f(u_minus)));
end
function u_next = compute_wendroff(initial_solutions, t_final, f, a, lambda)
    u_curr = initial_solutions;
    u_next = zeros(size(u_curr));
    k = length(u_curr);
    delta_x = 2 / (k - 1);
    delta_t = lambda * delta_x;
    m = floor(t_final / delta_t);
    for t = 1: m
        for i = 2: k - 1
            u_next(i) = compute_u_next(u_curr(i - 1), ...
            u_curr(i), u_curr(i + 1), lambda, f, a);
        end
        u_next(1) = compute_u_next(u_curr(k - 1), u_curr(1), u_curr(2), lambda, f, a);
        u_next(k) = u_next(1);
        u_curr = u_next;
    end
end
```

Listing 2: Code for Lax-Wendroff method

4 Output Plots

4.1 Test Case 1

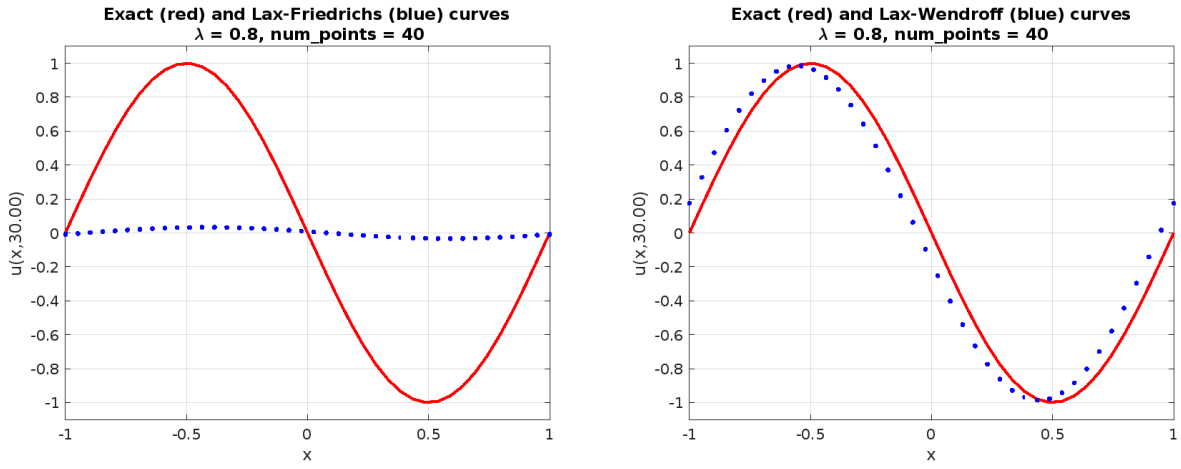

Figure 1: Test Case 1

4.2 Test Case 2

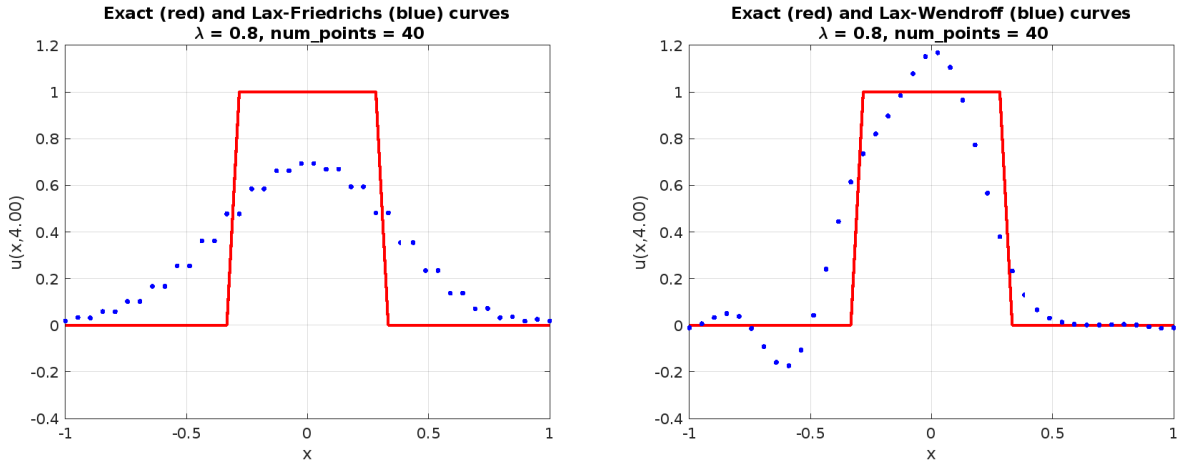

Figure 2: Test Case 2

4.3 Test Case 3

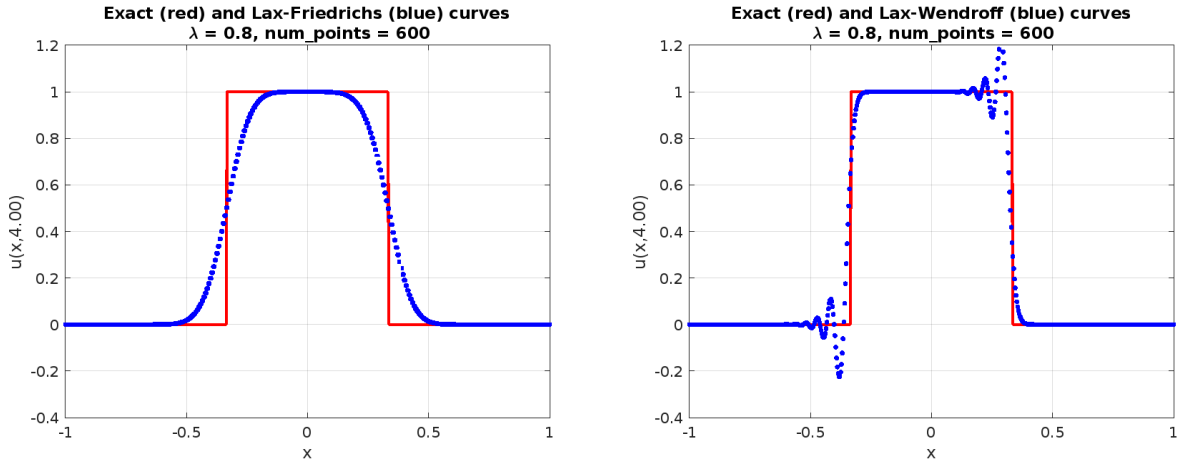

Figure 3: Test Case 3a

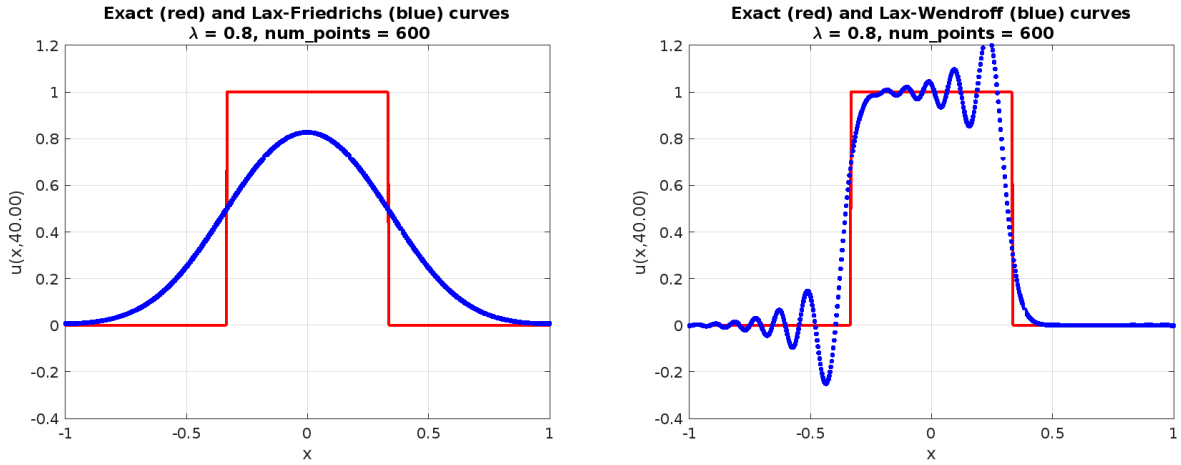

Figure 4: Test Case 3b

4.4 Test Case 4

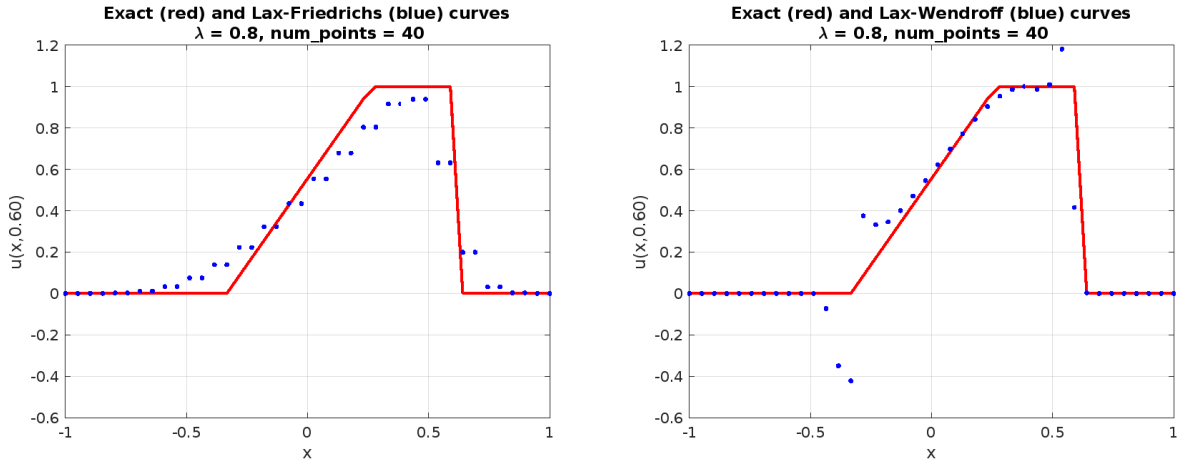

Figure 5: Test Case 4

4.5 Test Case 5

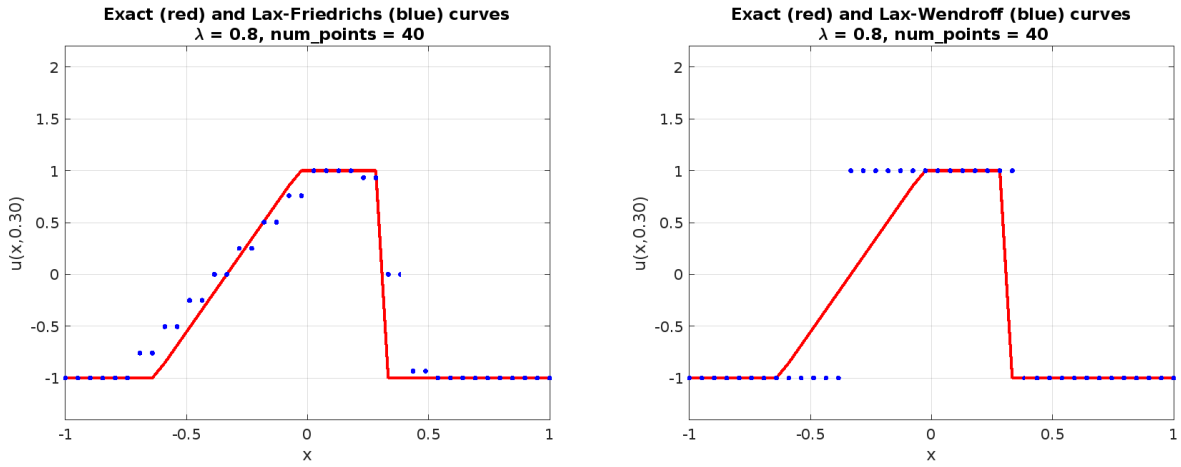

Figure 6: Test Case 5

5 Interpretation

5.1 General Observations

1. It's clear from comparing plots in Figure 2 and Figure 3 that the numerical solution goes more closer to the actual solution as we increase number of points taken along x-axis, keeping λ same.

2. See Figure 19 and Figure 20, for Test Cases 1, 2 and 3. For $\lambda = 1.1$, u_j^n's blow up and are are very far from their actual values,(orders of magnitude different in some cases) hence no/ very few points within the plot. Due to the larger step-size in t.

3. From the plots in subsection 6.1, we can see, $\hat{u}(x,t)$ is obtained, best from a certain value of λ, at very low values of λ, the approximation is bad (despite smaller step size in t, as the number of points along x is fixed).

4. In subsection 6.1 we can see for Test Cases 1, 2 and 3, for both Lax-Friedrichs and Lax-Wendroff Method give solutions, almost identical to the exact solution.

5.2 Test Case 1

This has a completely smooth exact solution with no sonic points. Neither of the two methods show dispersion as there is only one frequency in the exact solution.

Lax-Friedrichs Method: The amplitude is very small compared to exact solution. Slight error in phase but right shape.

Lax-Wendroff Method: There is lagging phase error. Amplitude and shape look good.

5.3 Test Case 2

The two jump discontinuities in the solution correspond to contact discontinuities. There progressive contact smearing (smoothing of sharp corners i.e. high frequency components) and dispersion (different frequency components travel at different speed).

Lax-Friedrichs Method: Odd-Even oscillations (Even indexed points behave differently from odd indexed points). There is smearing at the discontinuities of the exact solution.

Lax-Wendroff Method: Overshoots and undershoots the exact solution. Very slight phase error. Less smearing.

5.4 Test Case 3

It illustrates how dissipation, dispersion, and other numerical artifacts accumulate with large times for discontinuous solutions.

Lax-Friedrichs Method: The odd-even plateaus almost disappears. In Figure 3 and Figure 4, the exact solution is the same but the latter has more error (clearly more smearing) because of larger t value.

Lax-Wendroff Method: From Figure 2 to Figure 4 ,increasing the number of grid points creates large ringing oscillations to the left of the jump discontinuities. From Figure 3 to Figure 4 increasing the final time also increases the ringing oscillations.

5.5 Test Case 4

The jump from 0 to one at $x = -\frac{1}{3}$ creates an expansion fan, while the jump from one to zero at $x = \frac{1}{3}$ creates a shock. The unique sonic point for Burgers' equation is $u^* = 0$.

Lax-Friedrichs Method: There are strange odd-even two-point plateaus

Lax-Wendroff Method: The solution overshoots by near the shock and undershoots (left) and overshoots (right) near the tail of the expansion.

5.6 Test Case 5

The jump from 0 to one at $x = -\frac{1}{3}$ creates an expansion fan, while the jump from one to zero at $x = \frac{1}{3}$ creates a shock. The unique sonic point for Burgers' equation is $u^* = 0$.

Lax-Friedrichs Method: There are strange odd-even two-point plateaus

Lax-Wendroff Method: The steady shock is perfectly captured. However, the expansion fan is captured as an expansion shock.

6 Other Plots

6.1 Plots for $\lambda = 0.7, 0.8, 0.9, 1.0$

6.1.1 Lax-Friedrichs method

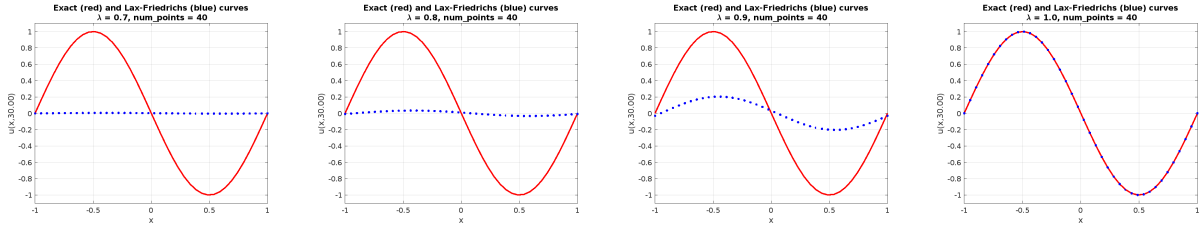

Figure 7: Test Case 1

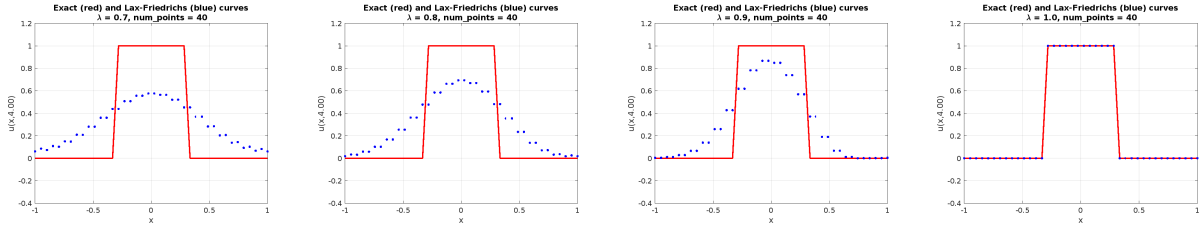

Figure 8: Test Case 2

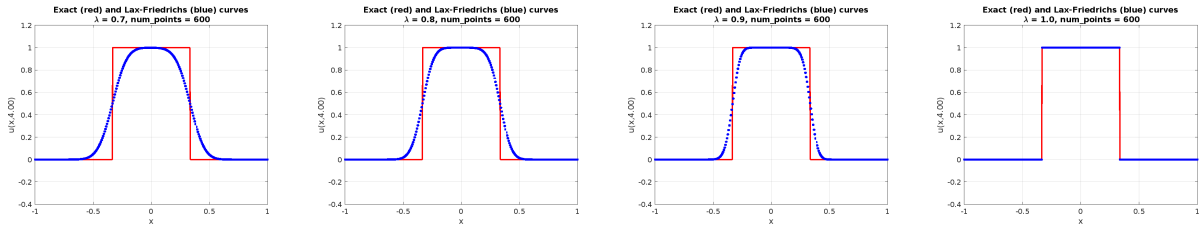

Figure 9: Test Case 3a

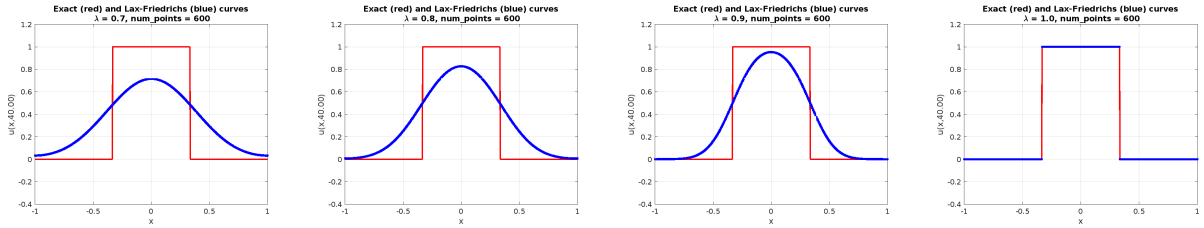

Figure 10: Test Case 3b

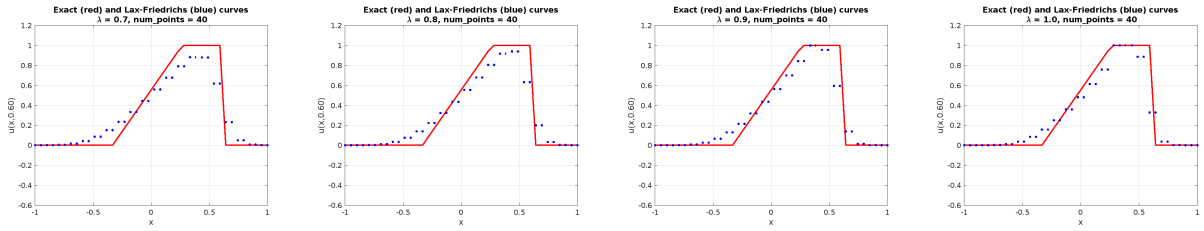

Figure 11: Test Case 4

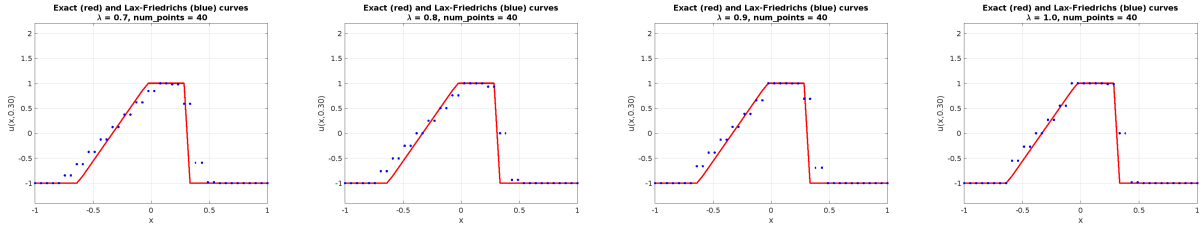

Figure 12: Test Case 5

6.1.2 Lax-Wendroff method

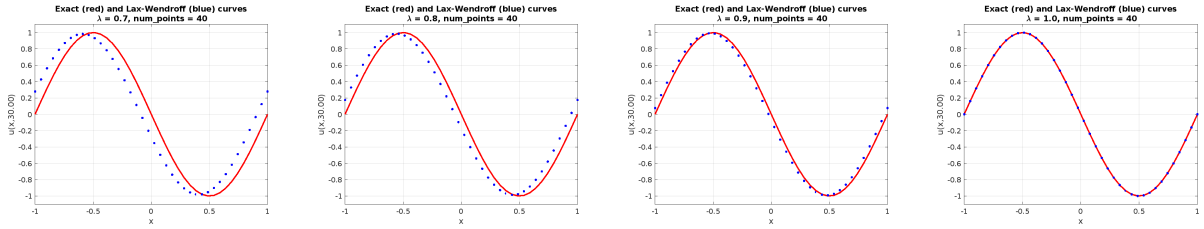

Figure 13: Test Case 1

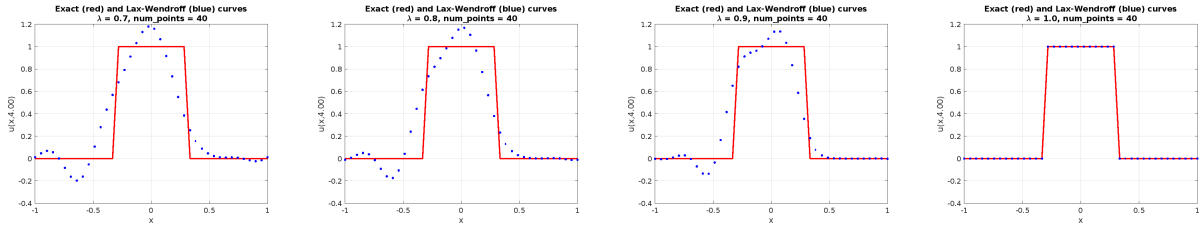

Figure 14: Test Case 2

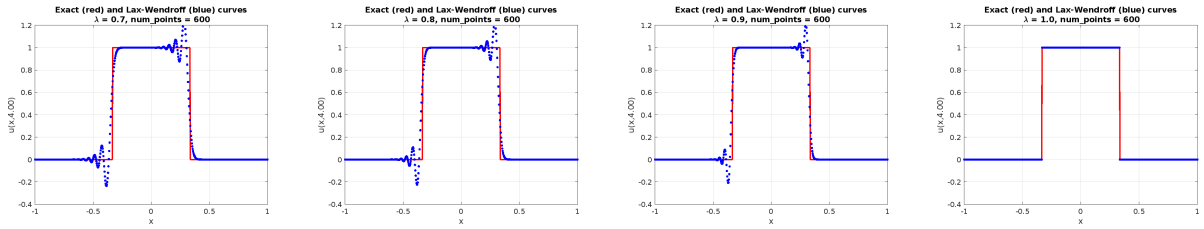

Figure 15: Test Case 3a

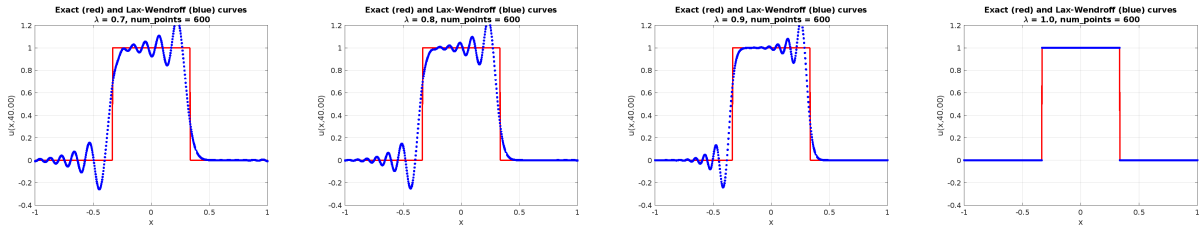

Figure 16: Test Case 3b

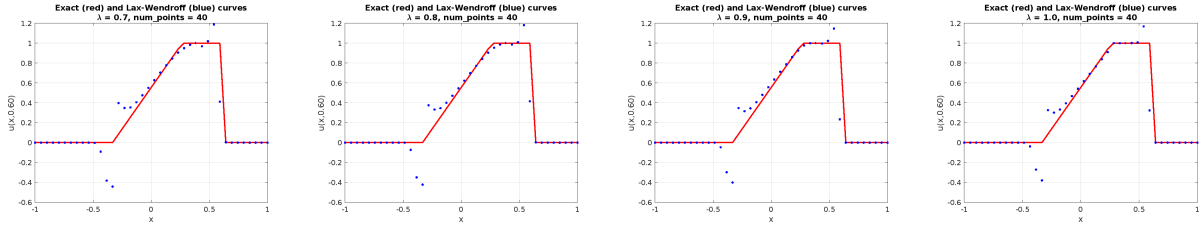

Figure 17: Test Case 4

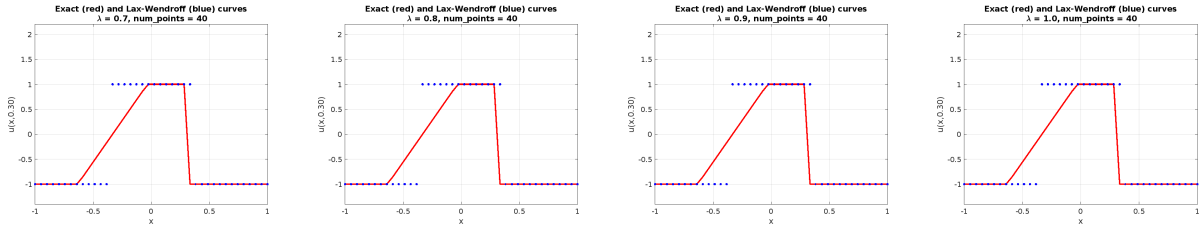

Figure 18: Test Case 5

6.2 Plots for $\lambda = 1.1$

6.2.1 Lax-Friedrichs method

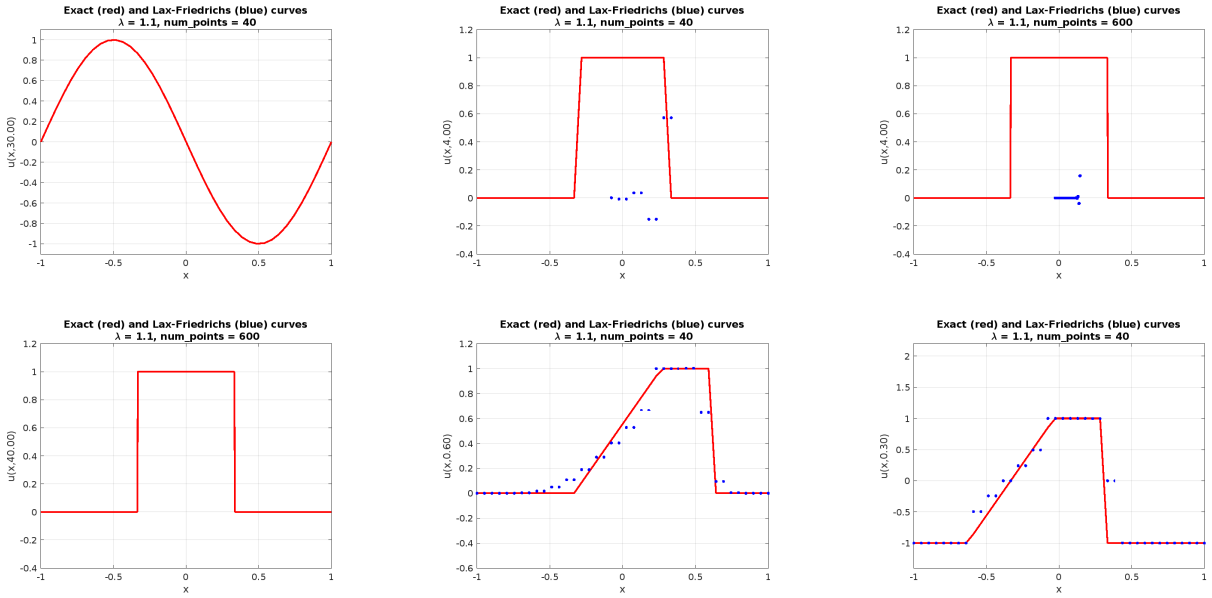

Figure 19: Plots for Test Cases 1, 2, 3a, 3b, 4, 5 respectively with $\lambda = 1.1$ (Lax-Friedrichs)

6.2.2 Lax-Wendroff method

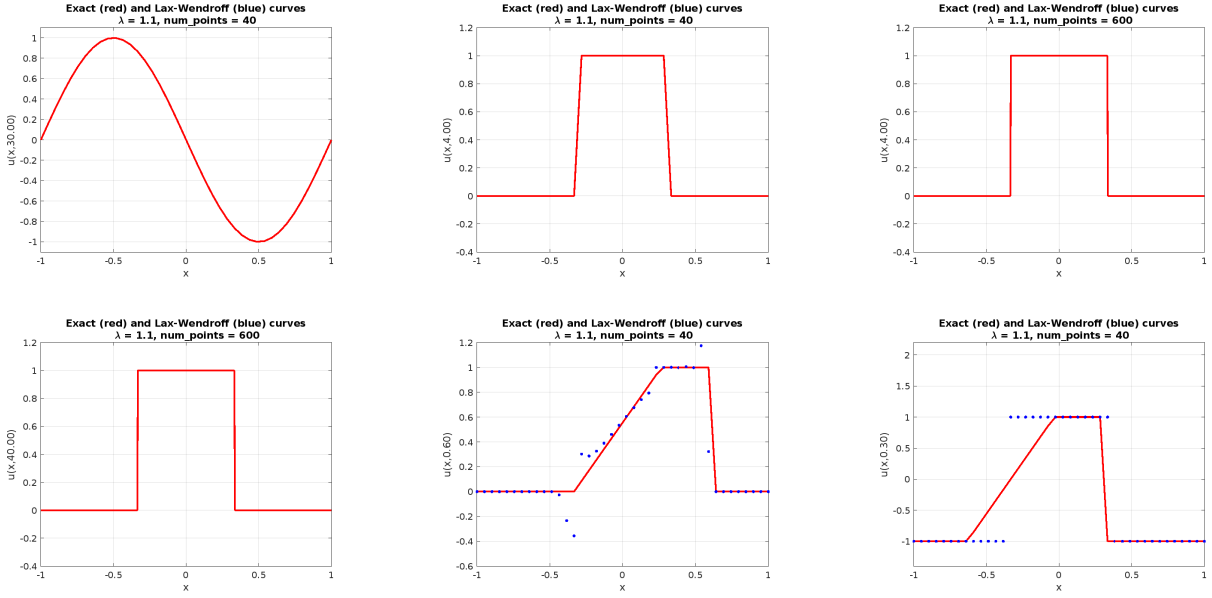

Figure 20: Plots for Test Cases 1, 2, 3a, 3b, 4, 5 respectively with $\lambda = 1.1$ (Lax-Wendroff)