07_hyperbolic_PDEs_students

December 31, 2025

[1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

1 Hyperbolic Equations

Hyperbolic equations are well-known to us and arise in many models of physical systems as well as others with wave-like behavior. One of the most basic hyperbolic PDEs is the advection equation

$$u_t + au_x = 0$$

where the unknown $u(x,t)$ is some quantity being transported by a flow at speed a. This equation will form the basis of our discussion on this broad topic.

For the Cauchy problem (infinite spatial-domain) we only need an initial condition $u(x,0) = u_0(x)$ with which we can immediately write down the solution of this simple PDE as

$$u(x,t) = u_0(x - at).$$

This simplicity belies complexity however when it comes to numerical methods to solve this equation. Luckily this equation demonstrates many of the numerical difficulties with solving more complex hyperbolic PDEs while be exceedingly simple to solve.

If we have finite boundaries on the advection equation note that for consistency we only need one boundary, called the in-flow boundary, and none on the out-flow boundary. Which boundary is which is determined by the direction of the flow, i.e. the sign of a. Numerically this may cause problems depending on the discretization we use (if we need a point from the out-flow boundary for instance). Instead in practice we will get around this by numerically setting a boundary that does not conflict with the out going solution or by using a different one-sided approximation. We will come back to this issue a bit later but it is good to be aware of this as we discuss.

1.1 First Discretizations

The first approach may be to discretize this equation with a forward Euler method in time and second order, centered differences in space leading to the discretization

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

or

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

It turns out this method is not very useful in practice as will see when considering stability.

Another method that is a minor modification of the above is called the *Lax-Friedrichs method*

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

This method replaces the single evaluation U_j^n with a spatial average. Although this approach has better stability properties (it is Lax-Richtmyer stable) it still is not used often in practice.

Another common discretization of the advection equation is that found from Leapfrog (midpoint). Using this approach we find

$$\frac{U_j^{n+1} - U_j^{n-1}}{2\Delta t} + a\frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} = 0$$

or in update form

$$U_j^{n+1} = U_j^{n-1} - \frac{a\Delta t}{\Delta x}(U_{j+1}^n - U_{j-1}^n).$$

One final note before moving onto convergence analysis deals with the relation between Δt and Δx. For the Lax-Friedrichs method for instance we will find the relationship

$$\left|\frac{a\Delta t}{\Delta x}\right| \leq 1$$

which, apart from a, shows that Δt and Δx may change at the same order! This leads us to the general conclusion that hyperbolic PDEs are in fact less stiff in general that parabolic PDEs due to the derivatives involved. This is also why hyperbolic PDEs can effectively be solved using explicit time stepping discretizations rather than the implicit ones we found useful for parabolic PDEs.

1.2 Method of Lines Discretization

As a first pass at understanding the stability of our aforementioned discretizations let's formulate the advection equation in a method of lines approach and analyze the stability in terms of an initial value problem.

In order to consider all the methods before we will assume periodic boundary conditions (so that they do not have the in-flow/out-flow problem). These boundary conditions look like

$$u(0,t) = u(1,t), \quad \text{for } t \geq 0$$

where we have chosen $\Omega = [0,1]$ for simplicity. This setup also is similar to the original Cauchy problem.

We now introduce the unknown vector

$$U = \begin{bmatrix} U_1(t) \\ U_2(t) \\ \vdots \\ U_m(t) \\ U_{m+1}(t) \end{bmatrix}$$

where we have included an extra value at x_{m+1} with the idea that $U_0(t) = U_{m+1}(t)$. Using this discretization we have the system of ODEs

$$U_j'(t) = -\frac{a}{2\Delta x}\begin{cases}(U_2(t) - U_{m+1}(t)) & j = 1\\ (U_{j+1}(t) - U_{j-1}(t)) & 2 \le j \le m\\ (U_1(t) - U_m(t)) & j = m+1\end{cases}$$

This leads to the system $U'(t) = AU(t)$ with

$$A = -\frac{a}{2\Delta x}\begin{bmatrix} 0 & 1 & & & & -1\\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1\\ 1 & & & & -1 & 0\end{bmatrix}.$$

The values in the corners are indicative of many PDEs with periodic boundary conditions.

The matrix A turns out to be *skew-symmetric* ($A^T = -A$) which implies that its eigenvalues are purely imaginary. These eigenvalues are

$$\lambda_p = -\frac{ia}{\Delta x}\sin(2\pi p \Delta x) \quad \text{for } p = 1, 2, \dots, m+1$$

with corresponding eigenvectors

$$u_j^p = e^{2\pi i p j/\Delta x} \quad \text{for } j = 1, 2, \dots m+1.$$

Given that the eigenvalues all lie along the imaginary axis

$$\lambda \in \left[-\frac{ia}{\Delta x}, \frac{ia}{\Delta x}\right]$$

we conclude that with the spatial discretization above that we need a method with an absolute stability region that includes the imaginary axis

$$z \in \left[-\frac{ia\Delta t}{\Delta x}, \frac{ia\Delta t}{\Delta x}\right]$$

now also taking into account Δt. Possible methods then include that the midpoint method, and some of the Adams methods may be suitable while the BDF methods will not be.

1.2.1 Example: Forward Euler

The first approach we presented above used a forward Euler discretization in time. If we look at the absolute stability region we know

$$|1 + \Delta t\lambda| \le 1$$

which is a unit-circle centered at -1. The ratio $\Delta t/\Delta x$ will never lead to a stable method as this region only includes the point $z = 0$ in this scenario. If instead we let Δt go to zero faster than Δx we may be able to cause the λ_p from the matrix to shrink to the origin.

In terms of the discussion of Lax-Richtmyer stability we use the weaker bound here $||B|| \leq 1+\alpha\Delta t$ where $B = I + \Delta t A$. Using the fact that λ_p is purely imaginary and picking $\Delta t = \Delta x^2$ we know

$$\begin{aligned} |1 + \Delta t \lambda_p|^2 &\leq 1 + \left(\frac{a\Delta t}{\Delta x}\right)^2 \\ &\leq 1 + a^2\Delta x^2 = 1 + a^2\Delta t. \end{aligned}$$

We can then bound $||B||$ by

$$||B|| \leq 1 + \alpha\Delta t$$

with $\alpha = a^2$. If $n\Delta t \leq T$ we then know

$$\left\|(I + \Delta t A)^n\right\|_2 \leq (1 + a^2\Delta t)^{n/2} \leq e^{a^2 T/2}$$

and hence the method is Lax-Richtmyer stable.

1.2.2 Example: Lax-Friedrichs

It is useful to rewrite the Lax-Friedrichs method above

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

using the fact that

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

as

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

Note the similarity between the new, weighted average and the stencil for the second order, centered difference approximation to the second derivative.

This can be further rearranged to give us a direct discretization interpretation of the method

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

Checking the consistency of the approximation on the left leads to

$$\begin{aligned} &\frac{1}{\Delta t}\left[u + \Delta t u_t + \frac{\Delta t^2}{2}u_{tt} + \frac{\Delta t^3}{6}u_{ttt} + \mathcal{O}(\Delta t^4) - u\right] \\ &\quad + \frac{a}{2\Delta x}\left[u + \Delta x u_x + \frac{\Delta x^2}{2}u_{xx} + \frac{\Delta x^3}{6}u_{xxx} - u + \Delta x u_x - \frac{\Delta x^2}{2}u_{xx} + \frac{\Delta x^3}{6}u_{xxx}) + \mathcal{O}(\Delta x^4)\right] \\ &= u_t + a u_x + \frac{\Delta t}{2}u_{tt} + \frac{\Delta t^2}{6}u_{ttt} + a\frac{\Delta x^2}{6}u_{xxx} + \mathcal{O}(\Delta t^3) + \mathcal{O}(\Delta x^3) \end{aligned}$$

so we can conclude that with the left hand side only that we would be consistent but we have an extra term!

Expanding the right hand side in a Taylor series we see

$$\frac{\Delta x^2}{2\Delta t}\left(u_{xx} + \frac{\Delta x^2}{12}u_{xxxx} + \mathcal{O}(\Delta x^4)\right)$$

leading to the conclusion that this is a discretization to the advection-diffusion equation

$$u_t + au_x = \epsilon u_{xx}$$

where

$$\epsilon = \frac{\Delta x^2}{2\Delta t}.$$

We can rewrite this system as $U'(t) = A_\epsilon U(t)$ where

$$A_\epsilon = -\frac{a}{2\Delta t}\begin{bmatrix} 0 & 1 & & & & -1 \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ 1 & & & & -1 & 0 \end{bmatrix} + \frac{\epsilon}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{bmatrix}.$$

Note in particular that this is the same matrix A as with the forward Euler discretization when $\epsilon = 0$. Since the extra term in A_ϵ causes the matrix to no longer be skew-symmetric but rather symmetric we expect the eigenvalues to no longer be purely imaginary. This extra term can be viewed as a *regularization* or *relaxation* of the original problem via a diffusive term. It is also important to note that many practical discretizations of the advection equation often will be diffusive in nature either as a consequence of the discretization or as a built-in feature as we see here.

Numerical Simulation: 1D linear advection equation Solve

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

Domain

- **Spatial:** $x \in [0, 25.0]$
- **Temporal:** $t \in [0, 17.0]$

Initial Condition

At $t = 0$, the solution is given by:

$$u(x, 0) = \exp(-20.0(x - 2)^2) + \exp(-(x - 5)^2)$$

Boundary Conditions

The domain is **periodic**:

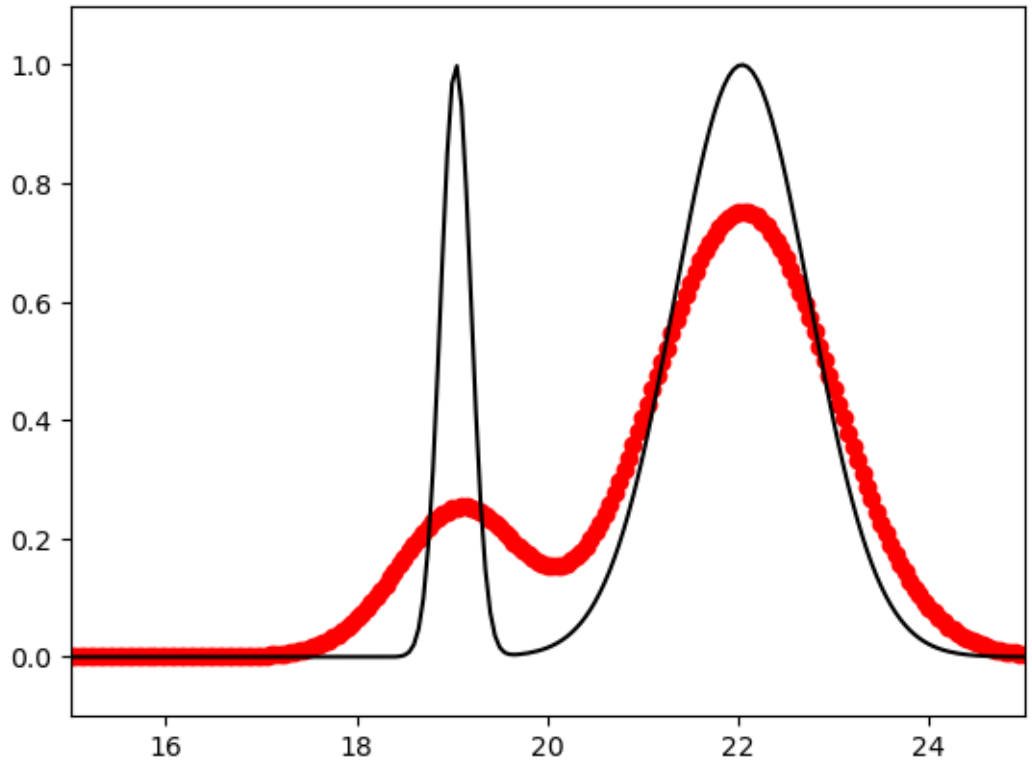
$$u(0, t) = u(25.0, t)$$

Exact Solution

The analytical solution to this problem is:

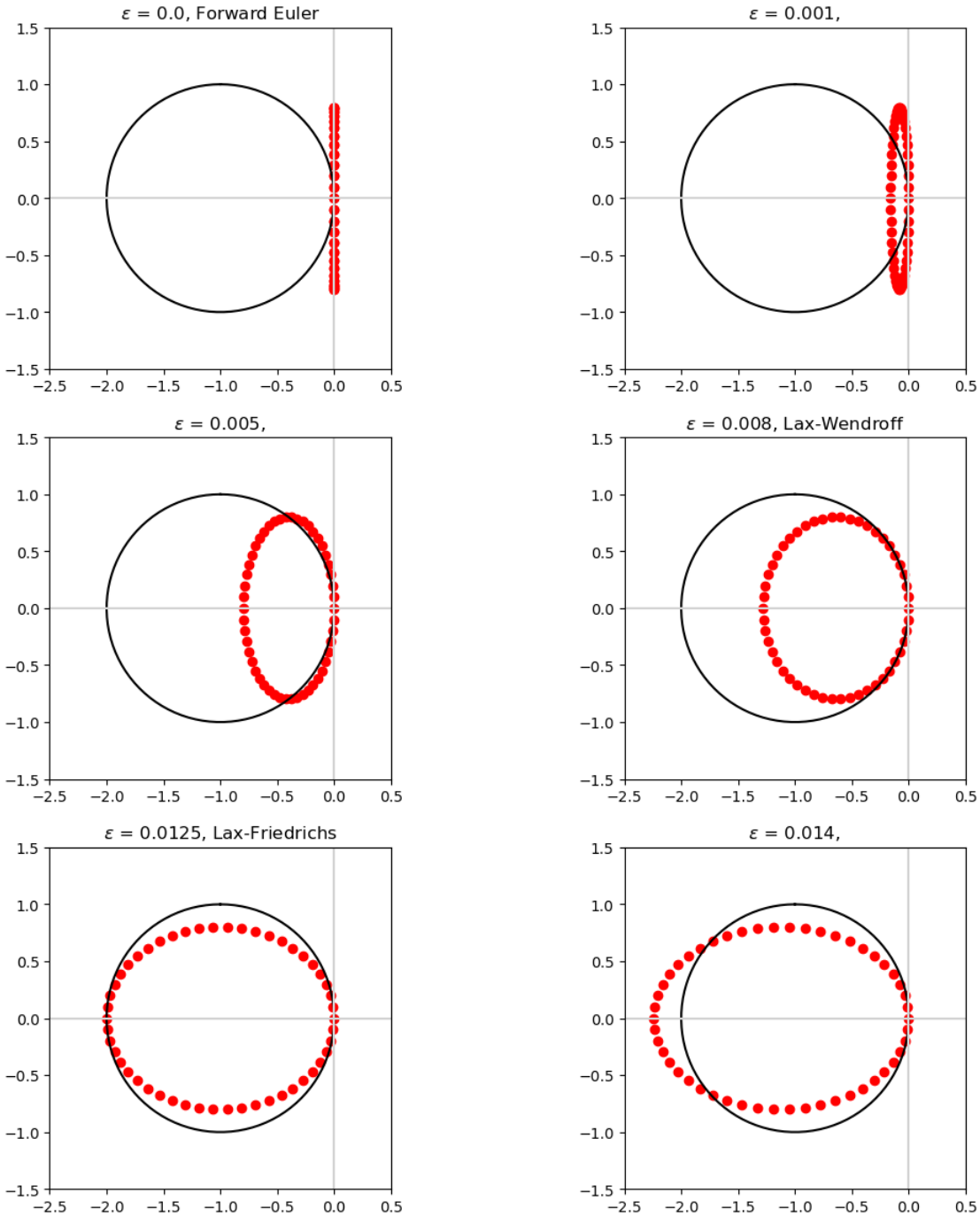
$$u(x, t) = \exp(-20.0((x - t) - 2)^2) + \exp(-((x - t) - 5)^2)$$

[ ]:
```
# Implement Lax-Friedrichs for this 1D linear advection equation
# You should get something like the following figure at t=17.0
```



[3]:
```
# Plot eigenvalues of the matrix A_\epsilon and plot relative
# absolute stability region

def construct_A(epsilon, a=1.0, delta_x=0.02):
    """Construct the matrix A from Leveque (10.15)
    """
    e = np.ones(int(1.0 / delta_x))
    A = np.diag(e[1:], 1) - np.diag(e[1:], -1)
    A[0, -1] = -1
    A[-1, 0] = 1

    B = np.diag(-2.0 * e, 0) + np.diag(e[1:], 1) + np.diag(e[1:], -1)
    B[0, -1] = 1
    B[-1, 0] = 1

    return -a / (2.0 * delta_x) * A + epsilon / delta_x**2 * B

delta_x = 0.02
delta_t = 0.8 * delta_x
```

```
a = 1.0
fig = plt.figure()
fig.set_figwidth(fig.get_figwidth() * 2)
fig.set_figheight(fig.get_figheight() * 4)
titles = ["Forward Euler", "", "", "Lax-Wendroff", "Lax-Friedrichs", ""]
for (i, epsilon) in enumerate((0.0, 0.001, 0.005, 0.008, 0.0125, 0.014)):
    axes = fig.add_subplot(4, 2, i + 1, aspect='equal')

    # Plot eigenvalues
    eigenvalues = np.linalg.eigvals(construct_A(epsilon, a, delta_x))
    axes.plot(delta_t * eigenvalues.real, delta_t * eigenvalues.imag, 'ro')

    # Plot offset circle
    theta = np.linspace(0.0, 2.0 * np.pi, 100)
    axes.plot(np.sin(theta) - 1.0, np.cos(theta), 'k')

    axes.set_xlim((-2.5, 0.5))
    axes.set_ylim((-1.5, 1.5))
    axes.set_title("$\epsilon$ = %s, %s" % (epsilon, titles[i]))

    axes.plot([-2.5, 0.5], [0.0, 0.0], color='lightgray')
    axes.plot([0.0, 0.0], [-1.5, 1.5], color='lightgray')

plt.show()
```



1.2.3 Example: Leapfrog

For the leapfrog (midpoint) method

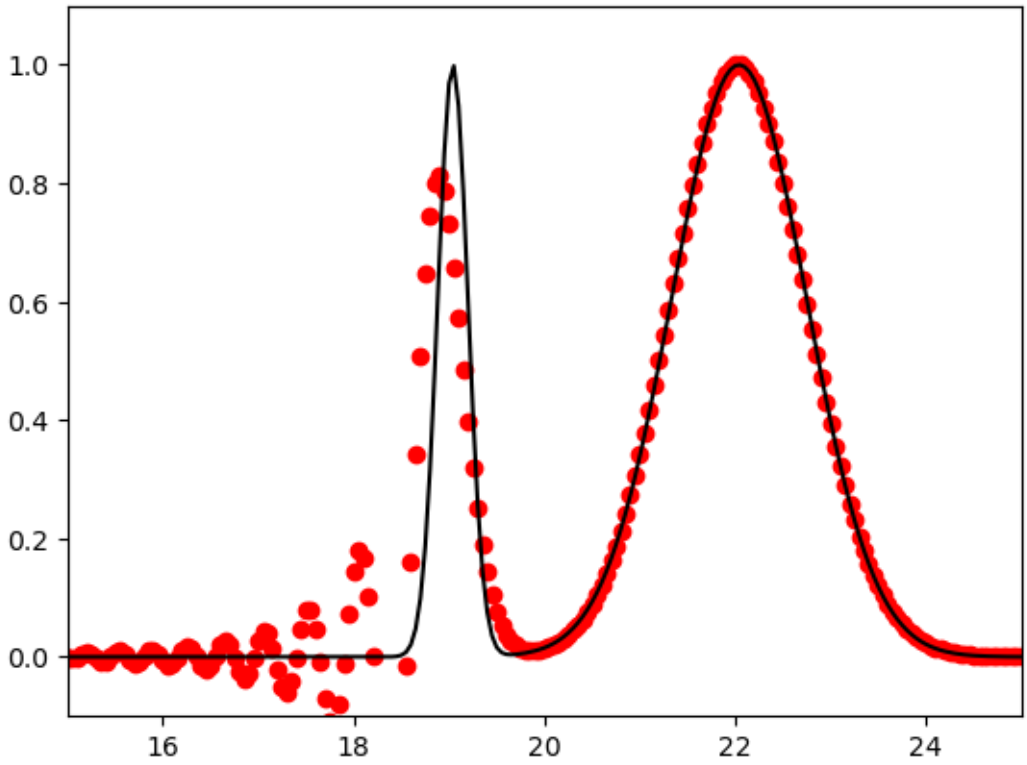
$$U_j^{n+1} = U_j^{n-1} - \frac{a\Delta t}{\Delta x}(U_{j+1}^n - U_{j-1}^n)$$

we know that its stability region is the interval along the imaginary axis $z \in [-i, i]$. This implies via method of lines that if

$$\left|\frac{a\Delta t}{\Delta x}\right| \leq 1$$

that this method will be stable. The one caveat to this is that $z = \Delta t \lambda_p$ will always lie on the boundary of this region (everything does) which means if we see a slight perturbation to the eigenvalues we may be in trouble. There is no growth but also no decay of any eigenmode of the approximation, also known as being *nondissipative*. This is not a bad behavior per se, the advection equation is nondissipative itself, but the other problem is that the modes move at different speeds rather than at the one dictated by a leading to dispersive error. This can also be difficult if the problem is something more complex than the advection equation (inhomogeneous or nonlinear problems for instance).

[ ]:
```
# Implement Leapfrog for the above 1D linear advection equation
# You should get something like the following figure at t=17.0
```



1.3 The Lax-Wendroff Method

So far the only method that we have used that was second order in space **and** time was the leapfrog discretization but that we saw that we had issues with its nondissipative behavior while also being multi-step method. Let's explore another approach that is still a one-step method. Consider the following options:

1. Make the method implicit (trapezoidal rule)
2. Use a Runge-Kutta method (RK 2)
3. Use a Taylor series method

1. Make the method implicit (trapezoidal rule) - We have already mentioned that we should not need to use an implicit method since the advection equation (and many hyperbolic PDEs) are not stiff.
2. Use a Runge-Kutta method (RK 2) - This becomes complex near boundaries and may again require extra storage as with leapfrog
3. Use a Taylor series method - This may require multiple evaluations of the spatial discretization but leads to what's called the *Lax-Wendroff method*.

Take a bit of time and see if you can derive the Lax-Wendroff method by expanding in the time derivative and using a second-order spatial discretization via a method-of-lines approach.

Expanding in time using a Taylor series leads to the system of ODEs

$$U^{n+1} = U^n + \Delta t A U^n + \frac{\Delta t^2}{2} A^2 U^n + \mathcal{O}(\Delta t^3)$$

where A is the matrix derived from our original second order, centered difference approximation.

Writing this out gives us the update formula

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

Note that the stencil has now become wider in space due to the A^2 term and may pose issues with boundaries.

Instead we could observe that the last term looks like an approximation to u_{xx} and instead of using the wider stencil, use the more familiar stencil. Instead of doing the above that matches the method of lines approach we can instead use Taylor expansions on the original PDE and simply keep more terms (similar to Taylor expansion methods discussed earlier) we can derive a simpler method.

The relevant expansion is

$$u(x, t + \Delta t) = u(x, t) + \Delta t u_t(x, t) + \frac{\Delta t^2}{2} u_{tt}(x, t) + \frac{\Delta t^3}{6} u_{ttt}(x, t) + \mathcal{O}(\Delta t^4)$$

and assuming the required smoothness we can replace the t derivates to x derivatives and the original equation we find

$$u(x, t + \Delta t) = u(x, t) - a\Delta t u_x(x, t) + \frac{a^2\Delta t^2}{2} u_{xx}(x, t) + \frac{a^3\Delta t^3}{6} u_{xxx}(x, t) + \mathcal{O}(\Delta t^4)$$

We see that we have a method that has the following approximation

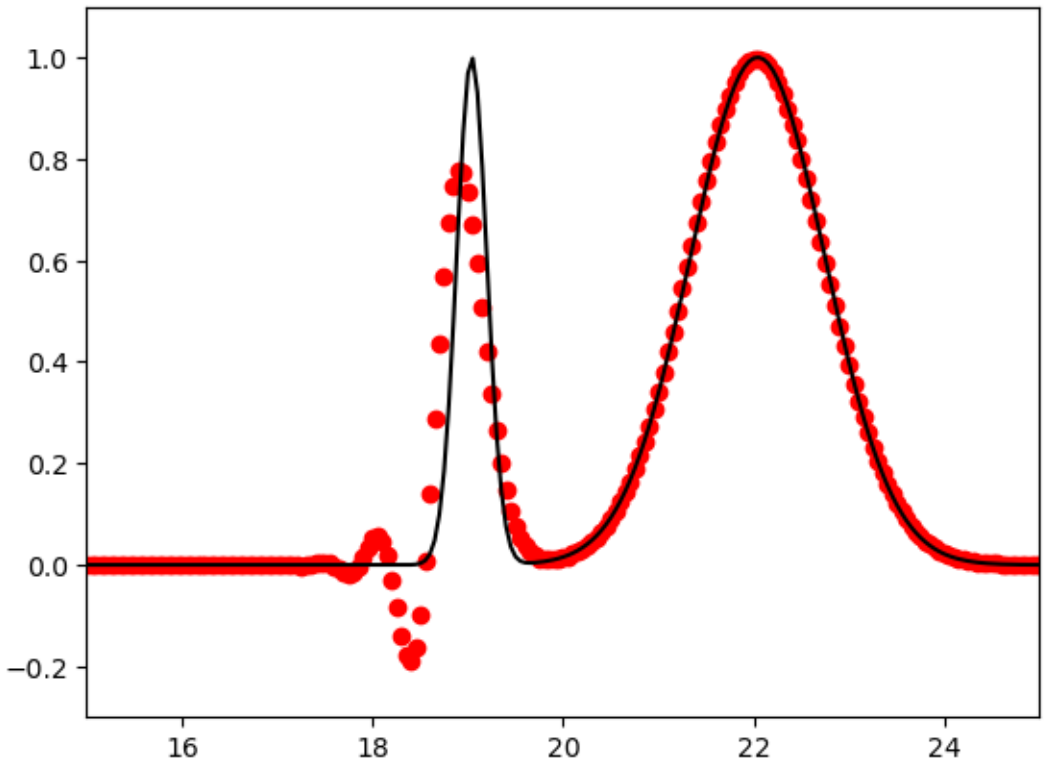
$$\begin{aligned} U_j^{n+1} &= U_j^n - a\Delta t D_0 U_j^n + \frac{a^2\Delta t^2}{2} D_2 U_j^n + \frac{a^3\Delta t^3}{6} u_{xxx}(x, t) + \mathcal{O}(\Delta t^4) \\ &= U_j^n - \frac{a\Delta t}{2\Delta x}(U_{j+1}^n - U_{j-1}^n) + \frac{a^2\Delta t^2}{2\Delta x^2}(U_{j+1}^n - 2U_j^n + U_{j-1}^n) + \frac{a^3\Delta t^3}{6} u_{xxx}(x, t) + \mathcal{O}(\Delta t^4) \end{aligned}$$

leading to the method

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

From here we see that we indeed can use a small stencil and that the dominant error is dispersive in nature (the third order x-derivative).

[ ]:
```
# Implement Lax-Wendroff for the above 1D linear advection equation
# You should get something like the following figure at t=17.0
```



1.3.1 Stability

Analyzing the Lax-Wendroff method follows from the approach we took to the Lax-Friedrichs method with the consideration of the general method using the A_ϵ matrix. Here though instead of $\epsilon = \Delta x^2/2\Delta t$ we use $\epsilon = a^2\Delta t/2$. The eigenvalues of this A_ϵ are

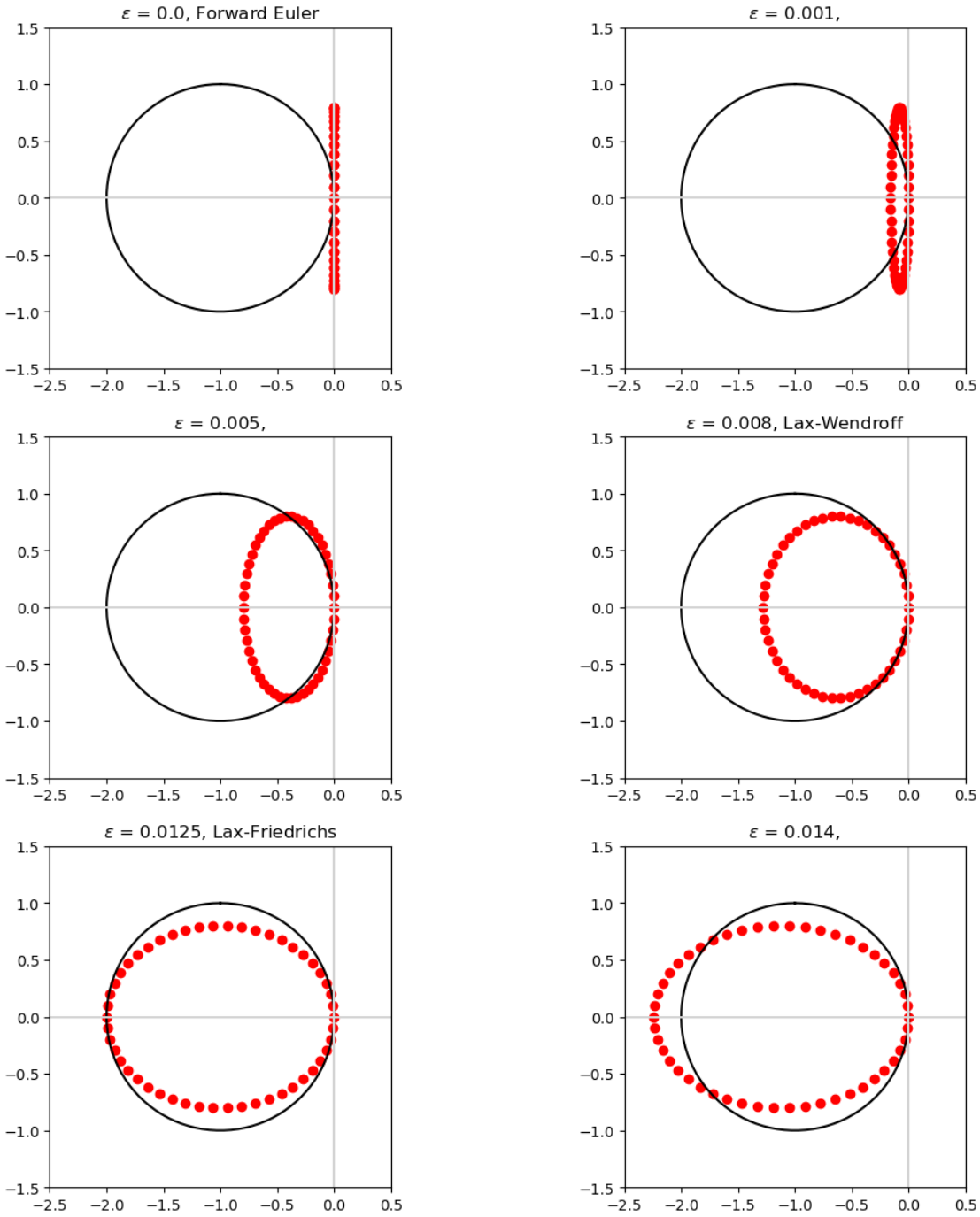
$$\Delta t\lambda_p = -i\frac{a\Delta t}{\Delta x}\sin(p\pi\Delta x) + \left(\frac{a\Delta t}{\Delta x}\right)^2(\cos(p\pi\Delta x) - 1).$$

From the numerical example above we see that these lie inside of Euler's method and has the same stability restriction of Lax-Friedrichs. This is great since we expect Lax-Wendroff to be second order accurate in time and space while Lax-Friedrichs was only first order.

Another observation of note is that the eigenvalues for the Lax-Wendroff method seem to look "optimally" close to the stability region boundary. This is due the second order nature of the Lax-Wendroff method and has the nice property that it uses the minimal amount of damping needed to remain stable.

[6]:
```
# Plot eigenvalues of the matrix A_\epsilon and plot relative
# absolute stability region

def construct_A(epsilon, a=1.0, delta_x=0.02):
    """Construct the matrix A from Leveque (10.15)
    """
    e = np.ones(int(1.0 / delta_x))
    A = np.diag(e[1:], 1) - np.diag(e[1:], -1)
    A[0, -1] = -1
    A[-1, 0] = 1

    B = np.diag(-2.0 * e, 0) + np.diag(e[1:], 1) + np.diag(e[1:], -1)
    B[0, -1] = 1
    B[-1, 0] = 1

    return -a / (2.0 * delta_x) * A + epsilon / delta_x**2 * B

delta_x = 0.02
delta_t = 0.8 * delta_x
a = 1.0
fig = plt.figure()
fig.set_figwidth(fig.get_figwidth() * 2)
fig.set_figheight(fig.get_figheight() * 4)
titles = ["Forward Euler", "", "", "Lax-Wendroff", "Lax-Friedrichs", ""]
for (i, epsilon) in enumerate((0.0, 0.001, 0.005, 0.008, 0.0125, 0.014)):
    axes = fig.add_subplot(4, 2, i + 1, aspect='equal')

    # Plot eigenvalues
    eigenvalues = np.linalg.eigvals(construct_A(epsilon, a, delta_x))
    axes.plot(delta_t * eigenvalues.real, delta_t * eigenvalues.imag, 'ro')

    # Plot offset circle
    theta = np.linspace(0.0, 2.0 * np.pi, 100)
    axes.plot(np.sin(theta) - 1.0, np.cos(theta), 'k')

    axes.set_xlim((-2.5, 0.5))
    axes.set_ylim((-1.5, 1.5))
    axes.set_title("$\epsilon$ = %s, %s" % (epsilon, titles[i]))

    axes.plot([-2.5, 0.5], [0.0, 0.0], color='lightgray')
    axes.plot([0.0, 0.0], [-1.5, 1.5], color='lightgray')
```

```
plt.show()
```



1.4 Upwind Methods

One aspect of the simple advection equation we are considering that has yet to be exploited is the asymmetry in the equation due to a. If $a > 0$ then waves propagate to the right and if $a < 0$

they propagate to the left. This suggests that perhaps a one-sided difference may be sufficient to approximate the solution rather than the centered approximations we have up until now considered.

Consider the one-sided differences

$$u_x(x_j,t) \approx \frac{1}{\Delta x}(U_j - U_{j-1}) \quad \text{and} \quad u_x(x_j,t) \approx \frac{1}{\Delta x}(U_{j+1} - U_j)$$

that are both first-order accurate approximations to the first derivative. Using these in conjunction with a forward Euler time stepping scheme leads to

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n) \quad \text{or} \quad U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_{j+1} - U_j)$$

which are then first order schemes in both time and space.

So how do we exploit the asymmetry mentioned due to a and our one sided differences?

The true solution of the advection equation at the point $(x_j, t + \Delta t)$ is

$$u(x_j, t + \Delta t) = u(x_j - a\Delta t, t)$$

representing the idea that the value of the solution at $u(x_j, t)$ is "flowing" to the new point (following the characteristics). In the case where $a > 0$ the solution is being dictated by a point to the left of x_j, specifically at $x_j - a\Delta t$. If $a < 0$ it then is determined by a point to the right of x_j.

This suggests that we may want to use the method

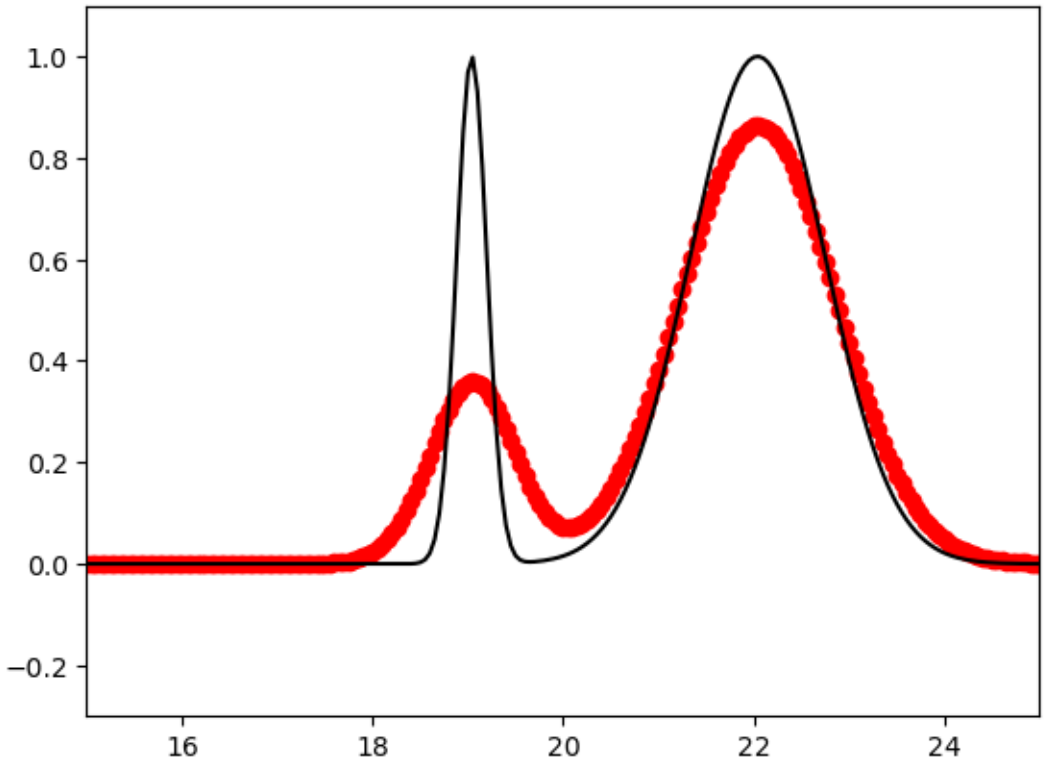
$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n)$$

if $a > 0$ and

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_{j+1} - U_j)$$

if $a < 0$. These methods are called *upwind methods* because they use points "upwind" of the current point to find the solution.

[ ]:
```
# Implement the Upwind method for the 1D linear advection equation
# You should get something like the following figure at t=17.0
```



1.4.1 Stability

Let's now examine the stability of these upwind methods. We can rewrite the upwind method for $a > 0$ with

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

leading to an effective $\epsilon = a\Delta x/2$.

We know from our method of lines analysis that a method like this will be stable if

$$\left|\frac{a\Delta t}{\Delta x}\right| \leq 1$$

and

$$-2 < -\frac{2\epsilon\Delta t}{\Delta x^2} < 0.$$

For Lax-Friedrichs ϵ was indepdent of a and Lax-Wendroff ϵ has a^2 so $\epsilon > 0$ and therefore this second condition would be satisfied with appropriate constraints on Δt and Δx on the lower bound. For upwind methods this is no longer true as the sign of a can cause $\epsilon < 0$. In fact this leads to a condition that tells us what form of the upwind method we need to use depending on the sign of a.

For the method analyzed above we have

$$-2 < -\frac{2\epsilon\Delta t}{\Delta x^2} < 0 \quad \Rightarrow 0 \leq \frac{a\Delta t}{\Delta x} \leq 1$$

which we know will satisfy the upper bound if $a > 0$.

For the method looking to the right we have the condition

$$-2 < -\frac{2\epsilon\Delta t}{\Delta x^2} < 0 \quad \Rightarrow -1 \leq \frac{a\Delta t}{\Delta x} \leq 0.$$

We can also again plot the eigenvalues of the matrix and see what happens in each case. Relating these values of ϵ again back to those for Lax-Friedrichs and Lax-Wendroff we have

$$\epsilon_{LW} = \frac{a^2\Delta t}{2} = \frac{a\Delta x\nu}{2} \quad \epsilon_{UP} = \frac{a\Delta x}{2} \quad \epsilon_{LF} = \frac{\Delta x^2}{2\Delta t} = \frac{a\Delta x}{2\nu}$$

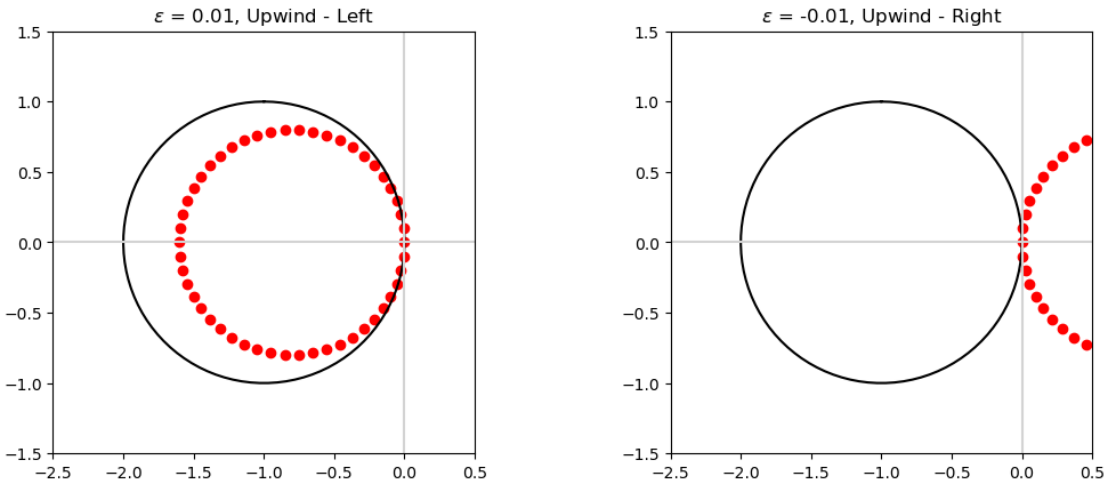
where $\nu = a\Delta t/\Delta x$ (called the Courant number). Note that

$$\epsilon_{LW} = \nu\epsilon_{UP} \quad \text{and} \quad \epsilon_U P = \nu\epsilon_{LF}$$

and if $0 < \nu < 1$ then we know that $\epsilon_{LW} < \epsilon_{UP} < \epsilon_{LF}$. In other words we see again that Lax-Wendroff and Lax-Friedrichs form the bounds on values of ϵ that remain stable and the upwind method sits in between the two extremes.

[8]:
```
# Plot eigenvalues of the matrix A_\epsilon and plot relative
# absolute stability region

def construct_A(epsilon, a=1.0, delta_x=0.02):
    """Construct the matrix A from Leveque (10.15)
    """
    e = np.ones(int(1.0 / delta_x))
    A = np.diag(e[1:], 1) - np.diag(e[1:], -1)
    A[0, -1] = -1
    A[-1, 0] = 1

    B = np.diag(-2.0 * e, 0) + np.diag(e[1:], 1) + np.diag(e[1:], -1)
    B[0, -1] = 1
    B[-1, 0] = 1

    return -a / (2.0 * delta_x) * A + epsilon / delta_x**2 * B

delta_x = 0.02
delta_t = 0.8 * delta_x
a = 1.0
fig = plt.figure()
fig.set_figwidth(fig.get_figwidth() * 2)
titles = ["Upwind - Left", "Upwind - Right"]
for (i, epsilon) in enumerate((a * delta_x / 2.0, -a * delta_x / 2.0)):
    axes = fig.add_subplot(1, 2, i + 1, aspect='equal')

    # Plot eigenvalues
    eigenvalues = np.linalg.eigvals(construct_A(epsilon, a, delta_x))
    axes.plot(delta_t * eigenvalues.real, delta_t * eigenvalues.imag, 'ro')
```

```
    # Plot offset circle
    theta = np.linspace(0.0, 2.0 * np.pi, 100)
    axes.plot(np.sin(theta) - 1.0, np.cos(theta), 'k')

    axes.set_xlim((-2.5, 0.5))
    axes.set_ylim((-1.5, 1.5))
    axes.set_title("$\epsilon$ = %s, %s" % (epsilon, titles[i]))

    axes.plot([-2.5, 0.5], [0.0, 0.0], color='lightgray')
    axes.plot([0.0, 0.0], [-1.5, 1.5], color='lightgray')

plt.show()
```



1.4.2 Beam-Warming Method

You may have been disconcerted that we have reverted back to a first order method again but do not fear, the *Beam-Warming method* is a second order accurate method with the same type of one-sided properties. If we go back to the expansion we found when deriving the Lax-Wendroff method

$$u(x,t+\Delta t) = u(x,t) - a\Delta t u_x(x,t) + \frac{a^2\Delta t^2}{2}u_{xx}(x,t) + \frac{a^3\Delta t^3}{6}u_{xxx}(x,t) + \mathcal{O}(\Delta t^4)$$

and approximate the derivatives using one-sided differences rather than centered we get for $a > 0$

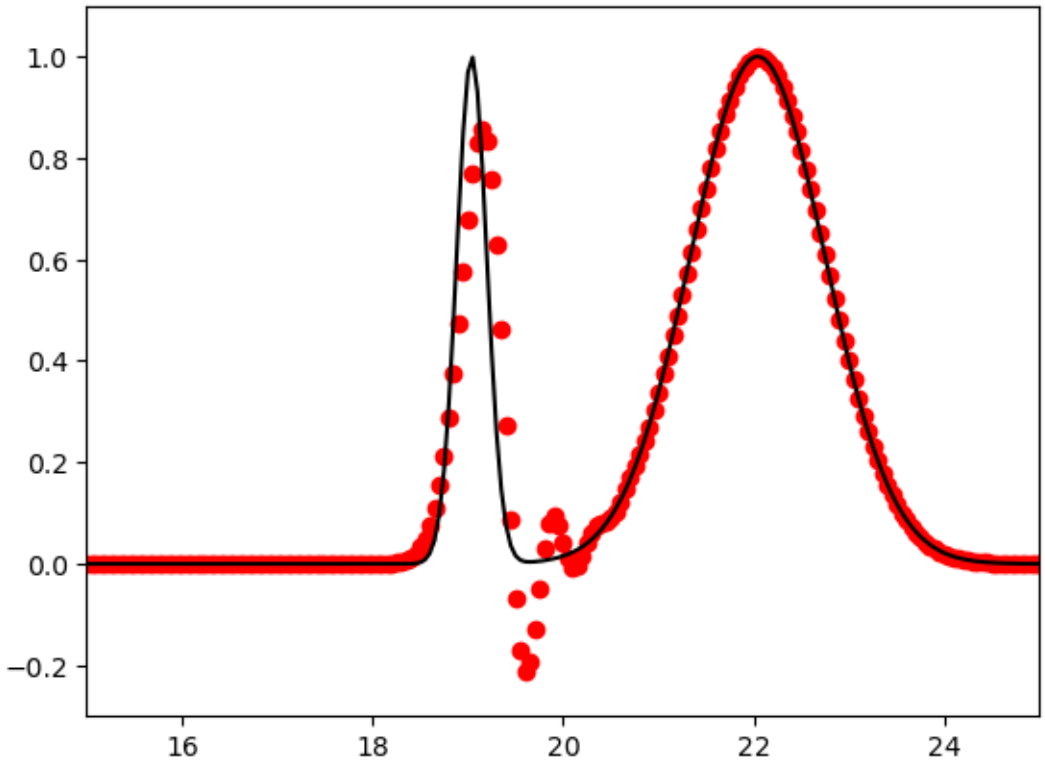
$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{2\Delta x}(3U_j^n - 4U_{j-1}^n + U_{j-2}^n) + \frac{a^2\Delta t^2}{2\Delta x^2}(U_j^n - 2U_{j-1}^n + U_{j-2}^n)$$

and for $a < 0$

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{2\Delta x}(-3U_j^n + 4U_{j+1}^n - U_{j+2}^n) + \frac{a^2\Delta t^2}{2\Delta x^2}(U_j^n - 2U_{j+1}^n + U_{j+2}^n).$$

These methods are stable if $0 \le \nu \le 2$ and $-2 \le \nu \le 0$ respectively.

```
[ ]: # Implement the Beam-Warming method for the above 1D linear advection equation
     # You should get something like the following figure at t=17.0
```



1.5 Von Neumann Analysis

We now try to derive similar constraints as we saw with the method of lines discretization but using von Neumann analysis instead.

Recall that von Neumann analysis proceeded by replacing U_j^n with $g(\xi)^n e^{i\xi j\Delta x}$ and finding an expression for the amplification factor $g(\xi)$. If the amplification factor satisfies

$$|g(\xi)| \leq 1$$

then we know the method is stable.

Below we will also introduce the notation

$$\nu \equiv \frac{a\Delta t}{\Delta x}$$

as it will be useful later on.

1.5.1 Example - Upwind

The upwind method for $a > 0$ is

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n).$$

Derive the amplification factor and the resulting stability condition.

Plugging in the plane-wave solution we have

$$\begin{aligned} g(\xi) &= 1 - \frac{a\Delta t}{\Delta x}(1 - e^{-i\xi\Delta x}) \\ &= (1-\nu) + \nu e^{-i\xi\Delta x}. \end{aligned}$$

This amplification factor varies with the wave number ξ such that it traces out a circle of radius ν centered at $1-\nu$. Note that if $\nu = 1$ the circle is centered at the origin and has radius 1, i.e. the unit ball. In other words they are on the edge of being stable. The other extreme with $\nu = 0$ is not quite so interesting (since this is a trivial case with $a = 0$). The conclusion then is that for stability the upwind method must have

$$0 \leq \nu \leq 1.$$

For the case $a < 0$ we find the analogous case with $-1 \leq \nu \leq 0$.

1.5.2 Example - Lax-Wendroff

Derive the amplification factor for the Lax-Wendroff method

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

$$\begin{aligned} g(\xi) &= 1 - \frac{\nu}{2}\left(e^{i\xi\Delta x} - e^{-i\xi\Delta x}\right) + \frac{\nu^2}{2}\left(e^{i\xi\Delta x} - 2 + e^{-i\xi\Delta x}\right) \\ &= 1 - i\nu\sin(\xi\Delta x) + \nu^2(\cos(\xi\Delta x) - 1) \\ &= 1 - i\nu 2\sin(\xi\Delta x/2)\cos(\xi\Delta x/2) - \nu^2 2\sin(\xi\Delta x/2) \end{aligned}$$

(the last line is written so that we can easily take the modulus).

$$\begin{aligned} |g(\xi)|^2 &= (1 - 2\nu^2\sin^2(\xi\Delta x/2))^2 + 4\nu^2\sin^2(\xi\Delta x/2)\cos^2(\xi\Delta x/2) \\ &= 1 + 4\nu^2\sin^2(\xi\Delta x/2)(\cos^2(\xi\Delta x/2) - 1) + 4\nu^4\sin^4(\xi\Delta x/2) \\ &= 1 - 4\nu^2(1-\nu^2)\sin^4(\xi\Delta x/2). \end{aligned}$$

We know $0 \leq \sin^4(\xi\Delta x/2) \leq 1$ for any ξ so we need $|\nu| \leq 1$ for the method to be stable.

1.5.3 Example - Lax-Friedrichs

Derive the amplification factor for Lax-Friedrichs

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

$$\begin{aligned} g(\xi) &= \frac{1}{2}(e^{i\xi\Delta x} + e^{-i\xi\Delta x}) - \frac{\nu}{2}(e^{i\xi\Delta x} - e^{-i\xi\Delta x}) \\ &= \cos(\xi\Delta x) - i\nu\sin(\xi\Delta x) \end{aligned}$$

leading to

$$|g(\xi)|^2 = \cos^2(\xi\Delta x) + \nu^2\sin^2(\xi\Delta x)$$

and therefore the condition that $|\nu| \leq 1$.

1.5.4 Example - Leapfrog

A bit more tricky but doable, derive the amplification factor and stability criteria for the leapfrog method

$$U_j^{n+1} = U_j^{n-1} - \frac{a\Delta t}{\Delta x}(U_{j+1}^n - U_{j-1}^n)$$

$$\begin{aligned} g(\xi)^2 &= 1 - \nu g(\xi)(e^{i\xi\Delta x} - e^{-i\xi\Delta x}) \\ &= 1 - 2\nu i \sin(\xi\Delta x) g(\xi). \end{aligned}$$

This then implies two branches for $g(\xi)$:

$$g_{1,2}(\xi) = -\nu i \sin(\xi\Delta x) \pm \sqrt{1 - \nu^2 \sin^2(\xi\Delta x)}$$

and therefore a modulus of

$$|g_{1,2}(\xi)|^2 = \nu^2 \sin^2(\xi\Delta x) + (1 - \nu^2 \sin^2(\xi\Delta x)) \leq 1$$

if $|\nu| \leq 1$ again.

1.6 Characteristic Tracing

The common way to solve hyperbolic PDEs analytically is by using the method of characteristics but up until now we really have not tried to use this theory to construct a numerical method. Thinking about the value of the solution at a point $(x_j, t + \Delta t)$ we know for $a > 0$ that we look backwards to the point $(x_j - a\Delta t, t)$ where the solution there informs the solution at the point of interest.

This works great until we start thinking about a discretized grid where we only know the solution at time t at a discrete set of points.

If the characteristics that intersects with $(x_j, t + \Delta t)$ also intersects with a point at time t then we are ok. Usually this will not be the case unless we specifically choose Δx and Δt such that this is true. It turns out of course that this happens if

$$\frac{a\Delta t}{\Delta x} = 1,$$

exactly the upper bound of our stability results.

Similarly if $\nu < 1$ then we know that the characteristic will not hit the grid points exactly. Note also that due to the constraint that $|\nu| \leq 1$ that we know that the characteristic cannot pass x_{j-1}.

We could also instead interpolate between the two values that the characteristic splits and find the value in question. Show that doing this using linear interpolation leads to the upwind method. For the linear interpolation we know that the intersection is at $x_p = x_j - a\Delta t$. The linear interpolant is

$$\begin{aligned} P_1(x) &= \frac{x - x_{j-1}}{x_j - x_{j-1}} U_j^n + \frac{x - x_j}{x_{j-1} - x_j} U_{j-1}^n \\ &= \frac{x - x_{j-1}}{\Delta x} U_j^n - \frac{x - x_j}{\Delta x} U_{j-1}^n \end{aligned}$$

so that the value is

$$\begin{aligned} U_j^{n+1} = P_1(x_j - a\Delta t) &= \frac{x_j - a\Delta t - x_{j-1}}{\Delta x} U_j^n - \frac{x_j - a\Delta t - x_j}{\Delta x} U_{j-1}^n \\ &= \frac{\Delta x - a\Delta t}{\Delta x} U_j^n + \frac{a\Delta t}{\Delta x} U_{j-1}^n \\ &= U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n). \end{aligned}$$

Using a similar technique we can also find the Beam-Warming method with quadratic interpolation:

$$\begin{aligned} P_2(x) &= \frac{(x - x_{j-1})(x - x_{j-2})}{(x_j - x_{j-1})(x_j - x_{j-2})} U_j^n + \frac{(x - x_j)(x - x_{j-2})}{(x_{j-1} - x_j)(x_{j-1} - x_{j-2})} U_{j-1}^n + \frac{(x - x_j)(x - x_{j-1})}{(x_{j-2} - x_j)(x_{j-2} - x_{j-1})} U_{j-2}^n \\ &= \frac{(x - x_{j-1})(x - x_{j-2})}{2\Delta x^2} U_j^n - \frac{(x - x_j)(x - x_{j-2})}{\Delta x^2} U_{j-1}^n + \frac{(x - x_j)(x - x_{j-1})}{2\Delta x^2} U_{j-2}^n \\ &= \frac{1}{\Delta x^2}\left[\frac{1}{2} U_j^n (x - x_{j-1})(x - x_{j-2}) - U_{j-1}^n (x - x_j)(x - x_{j-2}) + \frac{1}{2} U_{j-2}^n (x - x_j)(x - x_{j-1})\right] \end{aligned}$$

and finally

$$\begin{aligned} U_j^{n+1} = P_2(x_j - a\Delta t) &= \frac{1}{\Delta x^2}\left[\frac{1}{2} U_j^n (\Delta x - a\Delta t)(2\Delta x - a\Delta t) - U_{j-1}^n (-a\Delta t)(2\Delta x - a\Delta t) + \frac{1}{2} U_{j-2}^n (-a\Delta t)(\Delta x - a\Delta \right. \\ &= \frac{1}{\Delta x^2}\left[\frac{1}{2} U_j^n (2\Delta x^2 - 3a\Delta t\Delta x + a\Delta t^2) - U_{j-1}^n (-2a\Delta t\Delta x + a^2\Delta t^2) + \frac{1}{2} U_{j-2}^n (-a\Delta t\Delta x + a^2 \right. \\ &= U_j^n - \frac{a\Delta t}{2\Delta x}(3U_j^n - 4U_{j-1}^n + U_{j-2}^n) + \frac{a\Delta t^2}{2\Delta x^2}(U_j^n - 2U_{j-1}^n + U_{j-2}^n) \end{aligned}$$

1.7 The Courant-Friedrichs-Lewy (CFL) condition

One interesting result of our characteristic analysis was that the stability criteria caused the characteristic intersection with t_n to be within the interval $[x_{j-1}, x_j]$ when $a > 0$. This is indicative of a more general principle for stability for numerics for PDEs, due to Courant, Friedrichs, and Lewy and often called the CFL condition. The stability condition that we have been observing time and time again

$$\nu = \left|\frac{a\Delta t}{\Delta x}\right| \leq 1$$

turns out to be a necessary condition for methods developed to solve the advection equation. The value ν is often called the *Courant number* due to this.

1.7.1 Domain of Dependence

To make the more general statement about the CFL condition and the Courant number we need to talk about what the *domain of dependence* is for a given PDE. We already know what this is for the advection equation. We know the solution at (X, T) is $u(X, T) = u_0(X - aT)$. The domain of dependence then is

$$\mathcal{D}(X, T) = \{X - aT\}.$$

Another way to think about this is to consider what points could we change that would effect the solution at (X, T). In the case of the advection equation it is one point. More generally for other PDEs we might expect the domain of dependence to be larger than a single point but rather a set of points (as is the case for systems of advection equations) or an entire interval. The heat equation

is one such equation and has domain of dependence $\mathcal{D}(X,T) = (-\infty, \infty)$. In other words all points in the domain effect all other points at any future time. This type of equation is also said to have infinite *propagation speed* and is the case for any parabolic PDE and constitutes another way to classify more complex PDEs.

One could possibly reject this idea for the heat equation after all the Green's function for a particular point decays exponentially fast away from a point but unfortunately is still not fast enough. This is also the source of the conclusion and physical break down of the diffusion model, material (or heat) will travel infinitely fast.

1.7.2 Numerical Domain of Dependence

A numerical method also has a domain of dependence determined by the stencil used. For instance the Lax-Friedrichs method has the solution U_j^n dependent on the points U_{j+1}^{n-1}, U_j^{n-1}, and U_{j-1}^{n-1}. This is generally true for the three-point methods we developed earlier including the Lax-Wendroff method.
We can also trace backwards further in time to see which points U_{j+1}^{n-1}, U_j^{n-1}, and U_{j-1}^{n-1} depend on to see a growing cone of dependence.

As the grid is refined in both time and space respecting the stability criteria (the CFL condition) we then might expect that the numerical domain of dependence might converge to the true one. This is actually not true for the three-point stencils but in fact a weaker condition does hold, that the numerical domain of dependence should contain the PDE's domain of dependence. If we say continue to refine our grid with the ratio between $\Delta t/\Delta x = r$ then the domain of dependence for the point (X,T) will fill in the interval $[X - T/r, X + T/r]$. Since we want the computed solution $U(X,T)$ to converge to the true solution $u_0(X - aT)$ we need to require

$$X - T/r \leq X - aT \leq X + T/r.$$

This basically implies that $u_0(X - aT)$ lies in the numerical cone of dependence. This also implies that $|a| \leq 1/r$ and therefore $|a\Delta t/\Delta x| \leq 1$ again giving us the familiar stability criteria. This then leads us to the general statement of the CFL condition. The CFL condition can then be summed up in the following necessary condition: $>$ A numerical method is convergent only if its numerical domain of dependence contains the domain of dependence determined from the original PDE as $\Delta t \to 0$ and $\Delta x \to 0$. ### Example - Upwind Methods

Numerical domain depends on the sign of a but has a 2 point stencil. Note that if we pick the wrong direction for the upwinding that as Δt and Δx go to 0 the point $X - aT$ will never lie in the cone of dependence.

1.7.3 Example - Heat Equation

We have mentioned already that the true domain of dependence for the heat equation is the entire domain. How does that work for the heat equation then, especially with an implicit method? This would imply that any 3-point stencil (which was what we had been using) in fact violates the CFL condition. This is indeed true if we fix the ratio of $\Delta t/\Delta x$ but in fact we had a stricter requirement for the relationship, that $\Delta t/\Delta x^2 \leq 1/2$. This expands the domain of dependence as $\Delta t \to 0$ fast enough that it will cover the entire domain.

For implicit methods, such as Crank-Nicholson, the CFL condition is satisfied for any time step Δt due to the coupling of every point to every other point.

1.8 Modified Equations

Another powerful tool for analyzing numerical methods is the use of modified equations. This approach is similar to what we used for deriving local truncation error and reveals more about how we might expect a given numerical method to perform and what the error might appear as. The basic idea is to find a new PDE that may be solved **exactly** by the numerical method. In other words if we had a PDE $v_t = f(v, v_x, v_{xx}, \cdots)$ then our approximate solution given some Δt and Δx would satisfy $U_j^n = v(x_j, t_n)$. The question can also be posed "is there a PDE that U_j^n better captures?". We can answer this question via Taylor series expansions.

1.8.1 Example - Upwind Method

The upwind method is

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n).$$

assuming $a > 0$. Assume that we have a function $v(x,t)$ and an associated PDE (which we do not know yet) that the upwind method solves exactly. First replace the discrete solution U with the continuous function $v(x,t)$ so that we have

$$v(x, t+\Delta t) = v(x,t) - \frac{a\Delta t}{\Delta x}(v(x,t) - v(x-\Delta x, t)).$$

Using Taylor series we know

$$\left(v + v_t\Delta t + \frac{\Delta t^2}{2}v_{tt} + \frac{\Delta t^3}{6}v_{ttt} + \cdots\right) - v + \frac{a\Delta t}{\Delta x}\left(v - v + \Delta x v_x + \frac{\Delta x^2}{2}v_{xx} - \frac{\Delta x^3}{6}v_{xxx} + \cdots\right) = 0$$

$$v_t + \frac{\Delta t}{2}v_{tt} + \frac{\Delta t^2}{6}v_{ttt} + \cdots + a\left(v_x + \frac{\Delta x}{2}v_{xx} - \frac{\Delta x^2}{6}v_{xxx} + \cdots\right) = 0.$$

Reorganizing the terms in the equation we have

$$v_t + av_x = \frac{1}{2}(a\Delta x v_{xx} - \Delta t v_{tt}) + \frac{1}{6}(a\Delta x^2 v_{xxx} - \Delta t^2 v_{ttt}) + \cdots$$

This is the PDE that v satisfies. We can see here that if Δt and Δx go to zero we can expect that we will recover the original equation the method was meant to solve. The dominant terms on the right hand side though give us a glimpse of the behavior of the solution v if Δt and Δx are non-zero. For instance if we consider the $\mathcal{O}(\Delta x, \Delta t)$ terms we have the equation

$$v_t + av_x = \frac{1}{2}(a\Delta x v_{xx} - \Delta t v_{tt}),$$

an equation that also includes something that looks like the second order wave equation. We can rewrite this even more explicitly by differentiating both sides with respect to t

$$v_{tt} = -av_{xt} + \frac{1}{2}(a\Delta x v_{xxt} - \Delta t v_{ttt})$$

and with respect to x

$$v_{tx} = -av_{xx} + \frac{1}{2}(a\Delta x v_{xxx} - \Delta t v_{ttx})$$

which combined leads to

$$v_{tt} = a^2 v_{xx} + \mathcal{O}(\Delta t).$$

Inserting this back into the original expression on the right hand side we can get rid of the second order derivative in time to find

$$v_t + av_x = \frac{1}{2}a\Delta x\left(1 - \frac{a\Delta t}{\Delta x}\right)v_{xx} + \mathcal{O}(\Delta x^2, \Delta t^2)$$

which is an advection-diffusion equation similar to what we saw before except now explicitly formulated in the continuous case. We can also say then that the upwind discretization gives a solution to the above advection-diffusion equation to second-order accuracy. So what can we take away from this?
- This leading order behavior leads us to believe that the error will be diffusive in nature. - If $a\Delta t = \Delta x$, i.e. the Courant number $\nu = 1$, then the exact solution will be recovered. - The coefficient in front of the diffusion operator is $\frac{1}{2}(a\Delta x - a^2\Delta t)$ which is positive if $0 < a\Delta t/\Delta x < 1$, another way to see the stability criteria.

1.8.2 Example - Lax-Wendroff

Following the same procedure we can derive the leading order terms (up to $\mathcal{O}(\Delta t^2, \Delta x^2)$) for Lax-Wendroff to find

$$v_t + av_x = -\frac{1}{6}a\Delta x^2\left(1 - \left(\frac{a\Delta t}{\Delta x}\right)^2\right)v_{xxx}.$$

We can observe a few things from this modified equation - The Lax-Wendroff approximates an advection-dispersion equation to third order. - The dominant error will be dispersive (the third derivative does this) although this error will be smaller than the diffusive error from the up-wind method above.

An Aside - Dispersion Consider the PDE

$$u_t = u_{xxx}$$

as a Cauchy problem. If we Fourier transform the equation we arrive at the ODE

$$\hat{u}_t(\xi, t) = -i\xi^3\hat{u}(\xi, t)$$

which has the solution

$$\hat{u}(\xi, t) = \hat{u}_0(\xi)e^{-i\xi^3 t}.$$

Note that this looks like the general solution to an advection problem in that waves will maintain their amplitude, however each Fourier mode now propagates at its own speed dependent on its wave number. We can see this by taking the inverse Fourier transform to find

$$u(x, t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\hat{u}_0(\xi)e^{i\xi(x-\xi^2 t)}d\xi.$$

Examining the integrand we can see that the ξ wave number travels at the speed ξ^2. In contrast the similar path with the advection equation leads to

$$u(x, t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\hat{u}_0(\xi)e^{i\xi(x-at)}d\xi$$

where we clearly see all wave numbers ξ traveling at the speed a. This is the essential difference between advection and dispersion, the components of the solution spread out due to their different effective speeds. We can extend this to more general equations of the form

$$u_t + au_x + bu_{xxx} = 0$$

where we can write the solution

$$u(x,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{u}_0(\xi) e^{i\xi(x-(a-b\xi^2)t)} d\xi.$$

Here we see the speed of the components travel at $a - b\xi^2$ so the relative values of a and b will determine which effect will be more dominant. Back to the Lax-Wendroff method's modified equations we can write down the group velocity as

$$c_g = a - \frac{1}{2} a\Delta x^2 \left(1 - \left(\frac{a\Delta t}{\Delta x}\right)^2\right) \xi^2.$$

For this particular method $c_g < a$ for all ξ and hence the dispersion error trails the waves as seen in the numerical example. We can also retain more terms in the modified equation, if we did this to fourth order we would find

$$v_t + av_x + \frac{1}{6} a\Delta x^2 \left(1 - \left(\frac{a\Delta t}{\Delta x}\right)^2\right) v_{xxx} + \epsilon v_{xxxx} = 0$$

where $\epsilon = \mathcal{O}(\Delta x^3 + \Delta t^3)$. We now see that past the dispersive error we will find hyper-diffusion as the leading error. Dispersion and talking about wave numbers ξ also brings up another important consideration. If we were interested in highly oscillatory waves relative to the grid, i.e. when $\xi\Delta x \gg 0$, we may run into problems representing them on a given grid. For $\xi\Delta x$ sufficiently small this is not a problem and the modified equation gives a reasonable estimate as to the dispersion and therefore the group velocity. If our expected solution contains waves with $\xi\Delta x \gg 0$ then higher order terms may be needed to correctly represent the solution. Usually we therefore rely on plugging in the ansatz

$$u(x,t) = e^{i(\xi x_j - \omega(\xi) t_n)}.$$

This clearly has a relation to von Neumann analysis where we have replaced $g(\xi)$ with $e^{-i\omega(\xi)\Delta t}$.

1.8.3 Example: Beam-Warming

As a contrast to the Lax-Wendroff error behavior consider the modified equation for the Beam-Warming method which is

$$v_t + av_x = \frac{1}{6} a\Delta x^2 \left(2 - \frac{3a\Delta t}{\Delta x} + \left(\frac{a\Delta t}{\Delta x}\right)^2\right) v_{xxx}.$$

We saw with the numerical example that the dispersion error proceeded the wave and we now can see why as in this case $c_g > a$.

1.8.4 Example: Leapfrog

The modified equation for leapfrog leads to some interesting conclusions as we have some fortunate cancellations. Writing the leapfrog method as

$$\frac{v(x,t+\Delta t)-v(x,t-\Delta t)}{2\Delta t}+a\frac{v(x+\Delta x,t)-v(x-\Delta x,t)}{2\Delta x}=0$$

we can observe that the modified equations take the form

$$v_t+av_x+\frac{1}{6}a\Delta x^2\left(1-\left(\frac{a\Delta t}{\Delta x}\right)^2\right)v_{xxx}=\epsilon_1 v_{xxxxx}+\epsilon_2 v_{xxxxxxx}+\cdots.$$

It turns out that all even order derivative terms drop out leaving us only with dispersive error. In fact up to fourth order the leapfrog discretization solves an advection-dispersion equation. We can also see now again why leapfrog should be called non-dissipative as there are no error terms that have even derivatives, i.e. diffusion is not present. As a further exercise we can also compute the exact dispersion relation of the numerical method (the dispersion relation relates the wave number ξ to the phase speed, usually denoted $\omega(\xi)$). Plugging in the familiar ansatz similar to von Neumann analysis $e^{i(\xi x_j-\omega t_n)}$ we have

$$e^{-i\omega\Delta t}=e^{i\omega\Delta t}-\frac{a\Delta t}{\Delta x}\left(e^{i\xi\Delta x}-e^{-i\xi\Delta x}\right)$$

leading to

$$\sin(\omega\Delta t)=\frac{a\Delta t}{\Delta x}\sin(\xi\Delta x).$$

We can also compute the group velocity c_g from this since

$$c_g=\frac{\mathrm{d}\omega}{\mathrm{d}\xi}=\frac{a\cos(\xi\Delta x)}{\cos(\omega\Delta t)}=\pm\frac{a\cos(\xi\Delta x)}{\sqrt{1-\nu^2\sin^2(\xi\Delta x)}}.$$

Note again what happens if $\nu=1$.