Finite Difference Schemes for the 1D wave equation

S. Bilbao

The 1D Wave Equation

Some review from Acoustics last semester…

$$\partial_t^2 u = c^2 \partial_x^2 u$$

$u(x,t)$: dependent variable
$t \geq 0$: time (s)
c: wave speed (m/s)
∂_t, ∂_x: partial derivatives with respect to t and x

Different domains are of interest:

$x \in [0,\, L]$: finite $x \in \mathbb{R}$: infinite $x \in [0,\, \infty] = \mathbb{R}^+$: semi-infinite

Need to supply two initial conditions:

$$u(x,0) = u_0(x) \qquad \partial_t u(x,0) = v_0(x)$$

Comes up in two distinct settings in musical acoustics:

String vibration (u is string displacement)
Acoustic tubes (u is pressure)

Hypothesis: linear, lossless, non-stiff spatially homogeneous medium (strict!)

Dispersion Relation

To examine the dispersion characteristics of solutions to the wave equation, select test solution (ansatz) of the form

$$u(x,t) = \hat{u}e^{j\omega t + j\beta x} \qquad \omega\text{: angular frequency} \qquad \beta\text{: wavenumber}$$

Restrict attention to the infinite domain case $x \in \mathbb{R}$

(This is entirely equivalent to performing Fourier transforms in both space and time!)

Inserting in wave equation gives:

$$\left(\omega^2 - c^2\beta^2\right)\hat{u} = 0$$

Nontrivial solutions when

$$\omega = \pm c\beta$$

Phase velocity:

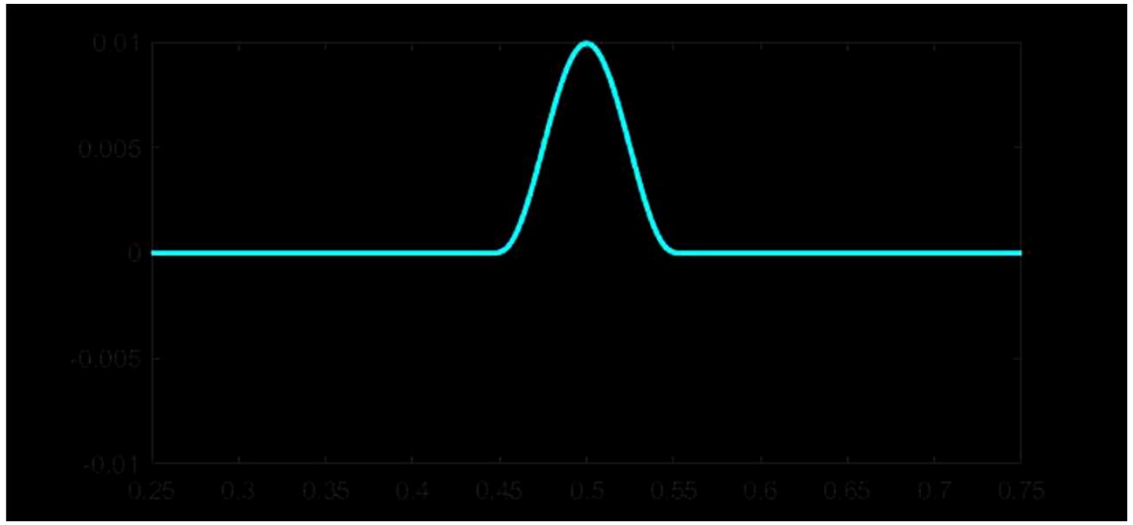
$$v_\phi = \frac{\omega}{\beta} = \pm c$$

Wave speed is constant for all frequencies

Traveling Waves

Because wave speeds are independent of frequency, we have a basic traveling wave solution:

$$u(x,t) = u_+(x - ct) + u_-(x + ct)$$



This is the basis for the “waveguide” approach to sound synthesis, pioneered by Julius Smith in the 1980s…we won’t cover waveguides in this course!

Energy

Restrict attention to the infinite domain case $x \in \mathbb{R}$

Inner product/norm defined as:

$$\langle f, g \rangle_{\mathbb{R}} = \int_{-\infty}^{\infty} f g dx \qquad \|f\|_{\mathbb{R}} = \langle f, f \rangle^{1/2}$$

Inner product of wave equation with $\partial_t u$

$$\partial_t^2 u - c^2 \partial_x^2 u = 0 \qquad \rightarrow \qquad \langle \partial_t u, \partial_t^2 u \rangle_{\mathbb{R}} - c^2 \langle \partial_t u, \partial_x^2 u \rangle_{\mathbb{R}} = 0$$

Integration by parts:

$$\langle \partial_t u, \partial_t^2 u \rangle_{\mathbb{R}} + c^2 \langle \partial_t \partial_x u, \partial_x u \rangle_{\mathbb{R}} = 0$$

Extract total derivative:

$$\frac{d}{dt} \left(\frac{1}{2} \|\partial_t u\|_{\mathbb{R}}^2 + \frac{c^2}{2} \|\partial_x u\|_{\mathbb{R}}^2 \right) = 0$$

Conservation law:

$$H(t) = \frac{1}{2} \|\partial_t u\|_{\mathbb{R}}^2 + \frac{c^2}{2} \|\partial_x u\|_{\mathbb{R}}^2 = \text{constant} \geq 0$$

NB: this is the physical energy to within a scaling constant

Boundary Conditions

Restrict attention to the semi-infinite domain case $x \in \mathbb{R}^+$

Allows an easy way to examine a single boundary in isolation…for realistic systems, we need two conditions of course (one at each end)!

Proceeding as before:

$$\partial_t^2 u - c^2 \partial_x^2 u = 0 \qquad \rightarrow \qquad \langle \partial_t u, \partial_t^2 u \rangle_{\mathbb{R}^+} - c^2 \langle \partial_t u, \partial_x^2 u \rangle_{\mathbb{R}^+} = 0$$

Integration by parts:

boundary term B

$$\langle \partial_t u, \partial_t^2 u \rangle_{\mathbb{R}^+} + c^2 \langle \partial_t \partial_x u, \partial_x u \rangle_{\mathbb{R}^+} + \boxed{c^2 \partial_t u(0,t) \partial_x u(0,t)} = 0$$

Extract total derivative:

$$\frac{d}{dt} H + B = 0$$

$$H(t) = \tfrac{1}{2} \|\partial_t u\|_{\mathbb{R}^+}^2 + \tfrac{c^2}{2} \|\partial_x u\|_{\mathbb{R}^+}^2 = \text{constant} \geq 0 \qquad B(t) = c^2 \partial_t u(0,t) \partial_x u(0,t)$$

Boundary Conditions

Energy conservation when boundary term vanishes:

$$u(0,t) = 0\text{: Dirichlet (fixed)} \qquad \partial_x u(0,t) = 0\text{: Neumann (free)}$$

Fixed is most important for strings…but there are many other choices!

边界处反相

Dirichlet

边界处翻倍，后变为原相

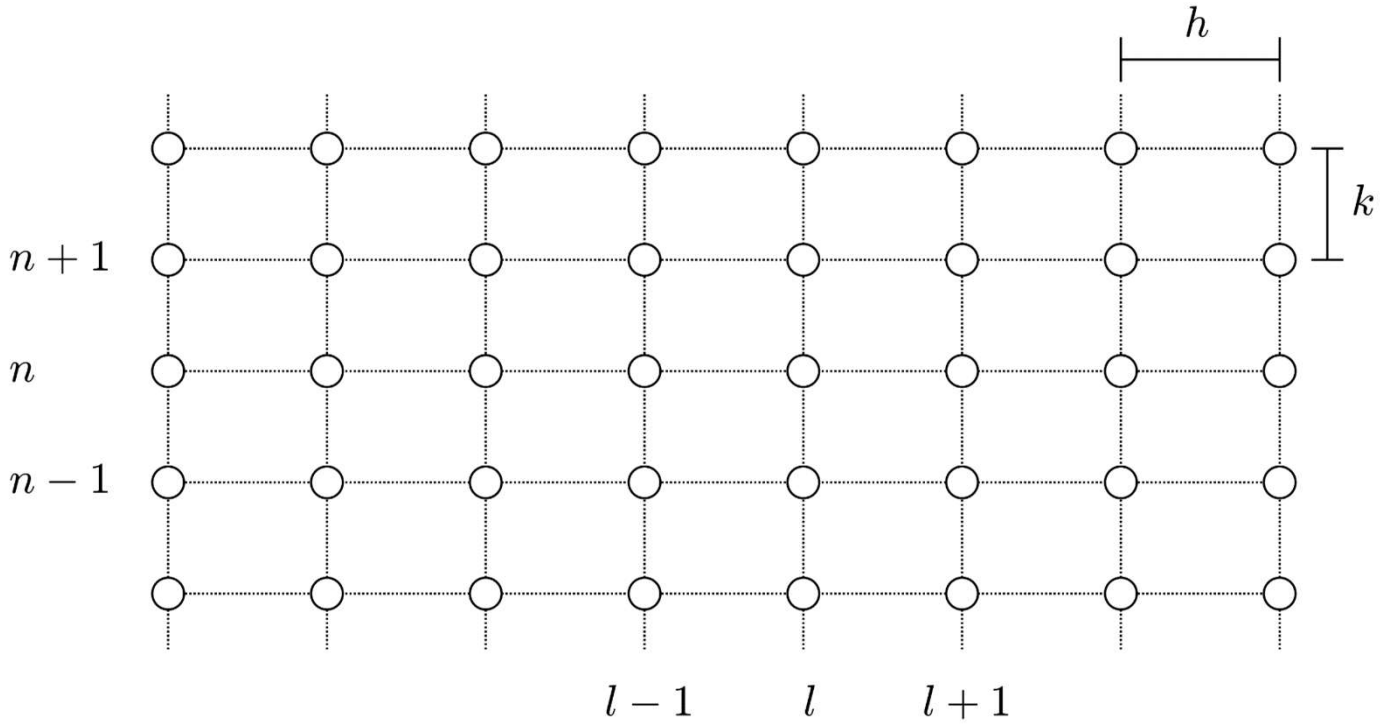
Neumann

Have looked at a semi-infinite interval---but this analysis extends to the case of a finite interval (BCs can be determined at both ends at once)

Grids

Restrict attention to the infinite domain case: $x \in \mathbb{R}$

k: time step $\qquad$ h: grid spacing



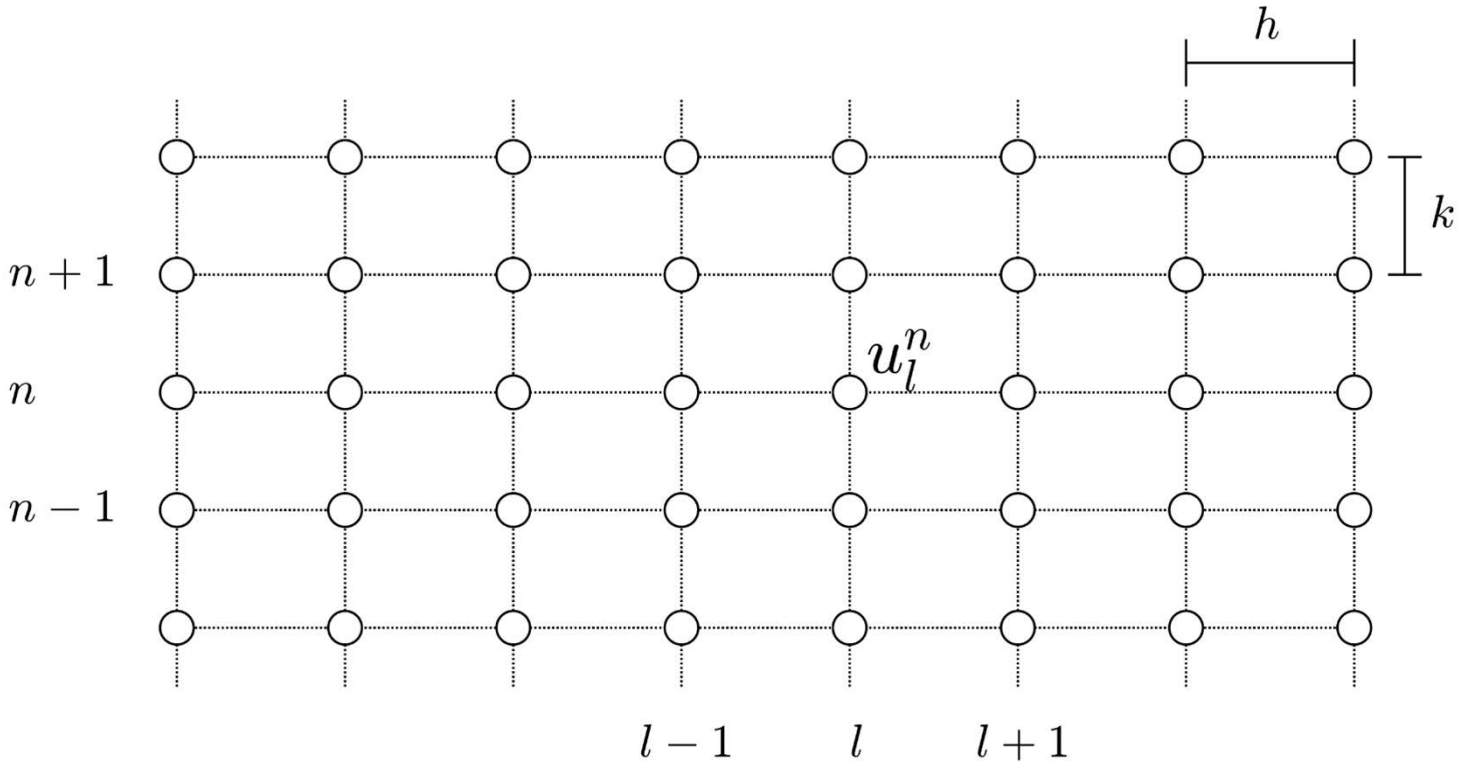
Grid points correspond to locations and times as:

$$x = lh \qquad\qquad t = nk$$

Grid Functions

Define a grid function approximating u:

$$u_l^n \approxeq u(x = lh, t = nk)$$



Shift and Difference Operators

Define basic temporal and spatial shift operators as

$$e_{t+}u_l^n = u_l^{n+1} \qquad e_{t-}u_l^n = u_l^{n-1} \qquad e_{x+}u_l^n = u_{l+1}^n \qquad e_{x-}u_l^n = u_{l-1}^n$$

Forward, backward and centered time difference operators (approximating a time derivative) are defined as

$$\delta_{t+} = \frac{1}{k}\left(e_{t+} - 1\right) \qquad \delta_{t-} = \frac{1}{k}\left(1 - e_{t-}\right) \qquad \delta_{t\cdot} = \frac{1}{2k}\left(e_{t+} - e_{t-}\right)$$

Similarly, forward and backward spatial difference operators (approximating a space derivative) are defined as

$$\delta_{x+} = \frac{1}{h}\left(e_{x+} - 1\right) \qquad \delta_{x-} = \frac{1}{h}\left(1 - e_{x-}\right)$$

Approximations to second derivatives may be defined, in operator form, as

$$\delta_{xx} = \delta_{x+}\delta_{x-} \qquad \delta_{tt} = \delta_{t+}\delta_{t-}$$

Applied to a grid function:

$$\delta_{tt}u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) \qquad \delta_{xx}u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right)$$

Scheme for the 1D Wave Equation

Return now to the wave equation. Replacing derivatives with difference operators gives, in operator form:

$$\partial_t^2 u = c^2 \partial_x^2 u \qquad \rightarrow \qquad \delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n$$

Expanding out the operator form gives a two step recursion:

$$u_l^{n+1} = 2u_l^n - u_l^{n-1} + \lambda^2 \left(u_{l+1}^n - 2u_l^n + u_{l-1}^n \right)$$

A very important dimensionless parameter introduced here is the Courant number:

$$\lambda = ck/h$$

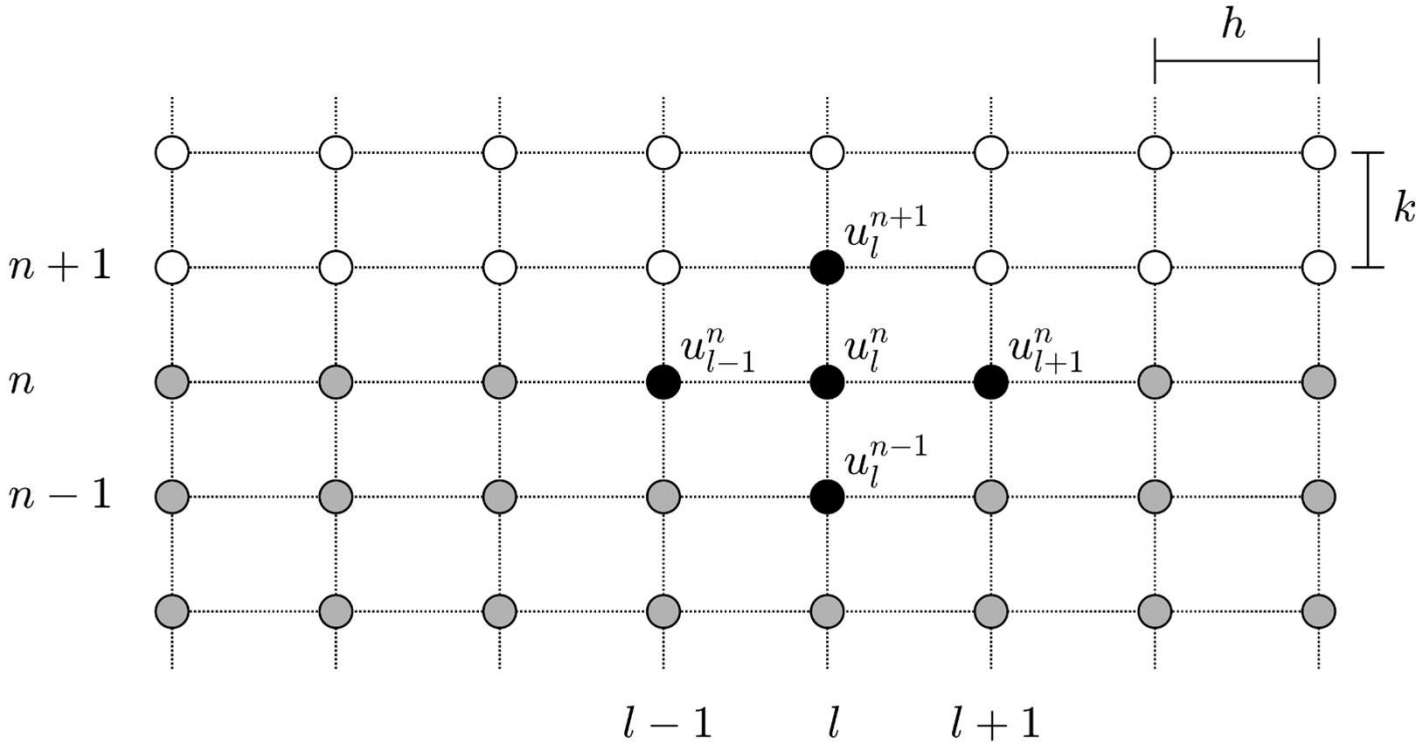
Initialisation: scheme needs to be set with two initial sets of values, at time steps n=0 and n=1 (as it requires two steps of “lookback”). Can set these as

$$u_l^0 = u_{l,0} \qquad u_l^1 = u_{l,0} + k v_{l,0}$$

in terms of given initial displacement and velocity distributions $u_{l,0}, v_{l,0}$

Scheme for the 1D Wave Equation

Pictorially:

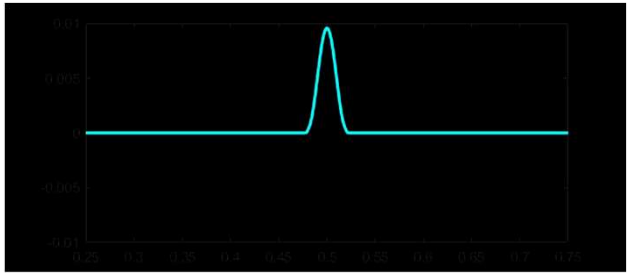


Update depends on neigbouring values at the previous time step, and one value two time steps previously.

Scheme for the 1D Wave Equation

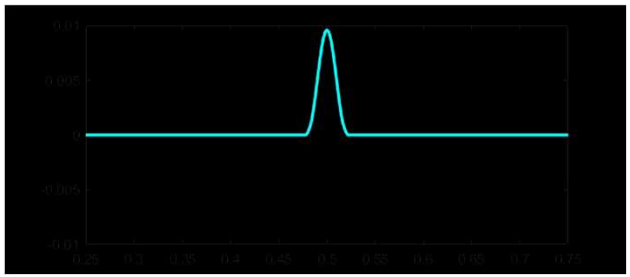
Stability and Dispersion

The properties of this algorithm depend strongly on the choice of the Courant number.



$\lambda = 1$: No dispersion

$\lambda = 0.5$: Anomalous dispersion



$\lambda = 1.01$: Explosive growth

von Neumann analysis

For the scheme for the wave equation,

$$\partial_t^2 u = c^2 \partial_x^2 u \qquad \rightarrow \qquad \delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n$$

just as in the continuous case, can examine a test solution (ansatz) of the form:

$$u_l^n = \hat{u} e^{j\omega nk + j\beta lh} \qquad \omega: \text{ angular frequency} \qquad \beta: \text{ wavenumber}$$

This is essentially a wave of discrete angular frequency and wavenumber:

$$\omega \in [-\pi/k, \, \pi/k]: \text{ angular frequency} \qquad \beta \in [-\pi/h, \, \pi/h]: \text{ wavenumber}$$

Difference operators behave as multiplicative factors:

$$\delta_{tt} u_l^n = -\tfrac{4}{k^2} \sin^2 (\omega k/2) \, u_l^n \qquad \delta_{xx} u_l^n = -\tfrac{4}{h^2} \sin^2 (\beta h/2) \, u_l^n$$

von Neumann analysis

Frequency domain techniques (called von Neumann analysis) allow us to examine this behaviour. For the ansatz: $u_l^n = \hat{u} e^{j\omega nk + j\beta lh}$

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n \qquad \rightarrow \qquad \left(\frac{4}{k^2} \sin^2 (\omega k/2) - \frac{4c^2}{h^2} \sin^2 (\beta h/2) \right) \hat{u} = 0$$

Nontrivial solutions when:

$$\sin^2 (\omega k/2) - \lambda^2 \sin^2 (\beta h/2) = 0$$

Need solutions for real frequency only (otherwise, we have complex frequency, and thus exponential growth!).

$$\sin (\omega k/2) = \pm \lambda \sin (\beta h/2)$$

Leads to famous Courant-Friedrichs-Lewy (CFL) condition for numerical stability:

$$\lambda \leq 1 \qquad \rightarrow \qquad h \geq ck$$

Numerical dispersion relation

$$\sin^2(\omega k/2) - \lambda^2 \sin^2(\beta h/2) = 0$$

This is the numerical dispersion relation for the scheme, relating frequency and wavenumber---now in the discrete setting!

Provided the CFL condition is satisfied, we can write

$$\omega = \pm \tfrac{2}{k} \sin^{-1}(\lambda \sin(\beta h/2)) \neq c\beta$$

In general, wave propagation is dispersive…and progressively so for higher frequencies or wavenumbers

But, when the Courant number is 1, the exact dispersion relation is recovered, and the scheme computes as exact solution! This is exactly what digital waveguides are doing

In general, want to choose Courant number as close to 1 as possible! But not larger, as the scheme becomes unstable.

Perspectives

Frequency domain analysis is a very powerful tool, but is limited:

Applies easily only to linear and shift invariant systems (like the wave equation)

Does not handle boundary conditions easily!

Discrete Energy Identities

Remaining with the case of the wave equation defined over an infinite domain, consider the following spatial inner product/norm definition:

$$\langle f, g \rangle_{\mathbb{Z}} = \sum_{l=-\infty}^{\infty} h f_l g_l \qquad\qquad \|f\|_{\mathbb{Z}} = \langle f, f \rangle_{\mathbb{Z}}^{1/2}$$

Essentially a Riemann approximation to the continuous IP/norm definitions…

The following identities follow (just as in the case of the SHO):

$$\langle \partial_t f, f \rangle_{\mathbb{R}} = \frac{d}{dt} \left(\frac{1}{2} \|f\|_{\mathbb{R}}^2 \right) \quad \rightarrow \quad \langle \delta_{t\cdot} f, f \rangle_{\mathbb{Z}} = \delta_{t+} \left(\frac{1}{2} \langle f, e_{t-} f \rangle_{\mathbb{Z}} \right) \qquad \text{(A1)}$$

$$\langle \partial_t f, \partial_t^2 f \rangle_{\mathbb{R}} = \frac{d}{dt} \left(\frac{1}{2} \|\partial_t f\|_{\mathbb{R}}^2 \right) \quad \rightarrow \quad \langle \delta_{t\cdot} f, \delta_{tt} f \rangle_{\mathbb{Z}} = \delta_{t+} \left(\frac{1}{2} \|\delta_{t-} f\|_{\mathbb{Z}}^2 \right) \qquad \text{(A2)}$$

Integration by parts now becomes summation by parts:

$$\langle \partial_x f, g \rangle_{\mathbb{R}} = -\langle f, \partial_x g \rangle_{\mathbb{R}} \quad \rightarrow \quad \langle \delta_{x+} f, g \rangle_{\mathbb{Z}} = -\langle f, \delta_{x-} g \rangle_{\mathbb{Z}} \qquad \text{(A3)}$$

Discrete Energy Conservation

Returning to the scheme for the wave equation,

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n$$

Take an inner product with $\delta_{t\cdot} u_l^n$ to get:

$$\langle \delta_{t\cdot} u, \delta_{tt} u \rangle_{\mathbb{Z}} - c^2 \langle \delta_{t\cdot} u, \delta_{xx} u \rangle_{\mathbb{Z}} = 0$$

Recalling that $\delta_{xx} = \delta_{x+} \delta_{x-}$, and using summation by parts (A3),

$$\langle \delta_{t\cdot} u, \delta_{tt} u \rangle_{\mathbb{Z}} + c^2 \langle \delta_{t\cdot} \delta_{x+} u, \delta_{x+} u \rangle_{\mathbb{Z}} = 0$$

Now, using identities (A1) and (A2),

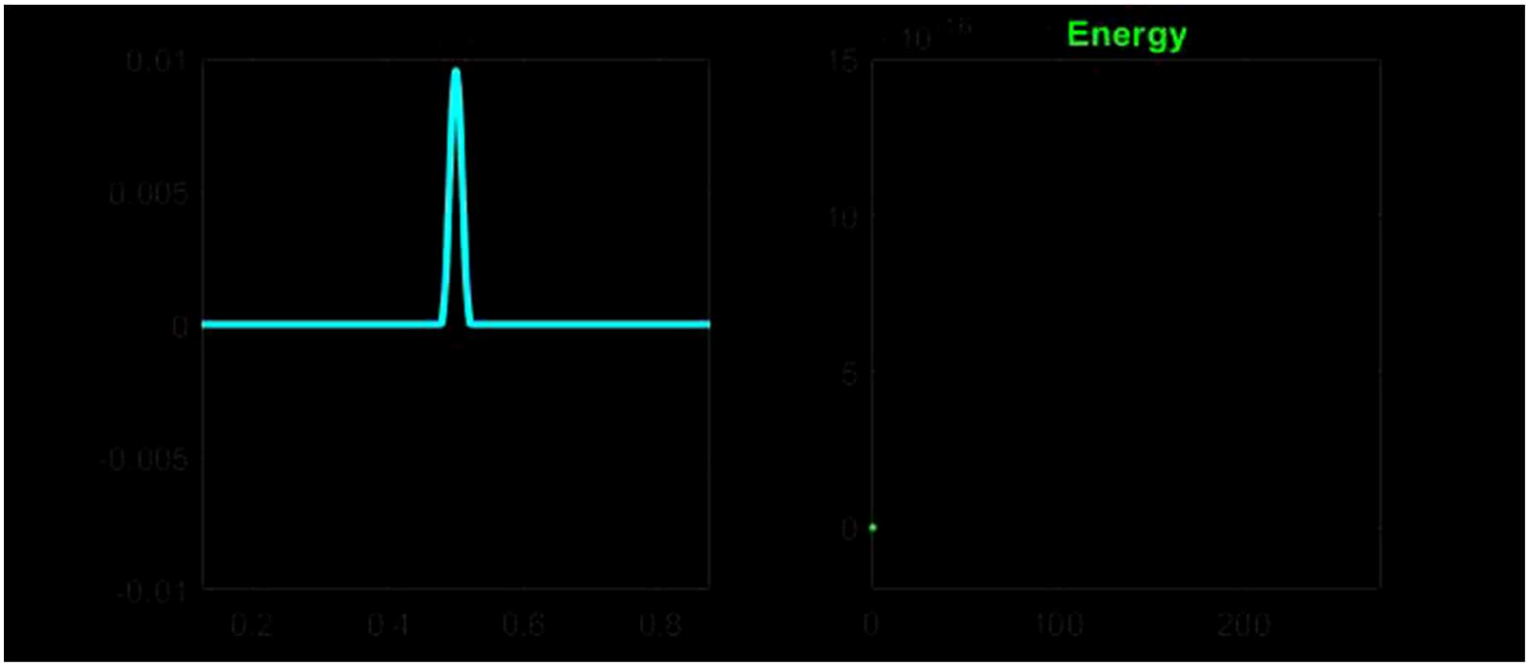
$$\delta_{t+} \left(\tfrac{1}{2} \| \delta_{t-} u \|_{\mathbb{Z}}^2 + \tfrac{c^2}{2} \langle \delta_{x+} u, e_{t-} \delta_{x+} u \rangle_{\mathbb{Z}} \right) = 0$$

We thus have a discrete conserved quantity in our scheme

$$\mathfrak{h} = \tfrac{1}{2} \| \delta_{t-} u \|_{\mathbb{Z}}^2 + \tfrac{c^2}{2} \langle \delta_{x+} u, e_{t-} \delta_{x+} u \rangle_{\mathbb{Z}} = \text{constant}$$

Energy Conservation to Machine Accuracy

Discrete energy conserved to machine accuracy (about 1e-16)



This is an excellent debugging tool! You can actually compute the numerical energy in the loop, and if you see variations beyond machine accuracy, you know there is an error womewhere…

Stability

We now have a conserved energy, just like the continuous system…

$$\mathfrak{h} = \frac{1}{2}\|\delta_{t-} u\|_{\mathbb{Z}}^2 + \frac{c^2}{2}\langle \delta_{x+} u, e_{t-}\delta_{x+} u\rangle_{\mathbb{Z}} = \text{constant}$$

non-negative not necessarily non-negative

But, we have not said anything about numerical stability…how can we do this?

Need to ensure non-negativity of the energy function! This is sometimes known as Lyapunov theory…

Can prove (need to go to NSS for this!) that the energy is non-negative under the CFL condition:

$$\lambda \leq 1 \qquad \rightarrow \qquad h \geq ck$$

Same as the result from frequency domain analysis…but now much more general!

Finite Domains

Now, let's turn to a finite domain (what you will actually code). $x \in [0,\ L]$

First, choose a sample rate F_s from which the time step follows as $k = 1/F_s$

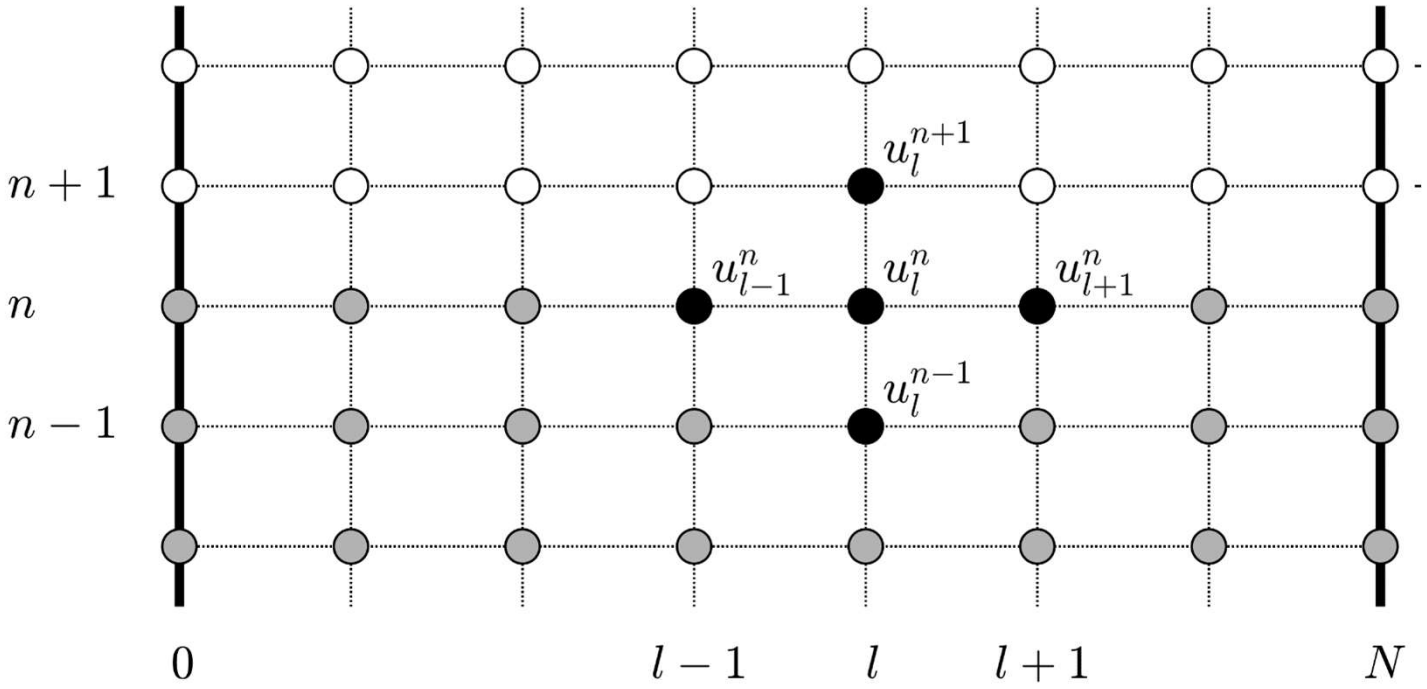
Now, we would like to select the grid spacing h so that the CFL condition is satisfied as close to equality as possible, and also that the domain length L is divided into N equal segments of length h. Thus:

$$N = \mathrm{floor}(L/ck) \qquad \mathrm{and\,then\,set} \qquad h = L/N$$

In this way, the Courant number will be nearly equal to 1 (but below it!).

Finite Domains

Grid now looks like:



NB: grid contains N+1 points (i.e., it includes the endpoints). We will look at boundary conditions shortly…

Numerical Boundary Conditions: Dirichlet

Consider now the scheme for the wave equation, now restricted to a finite domain:

$$u_l^{n+1} = 2u_l^n - u_l^{n-1} + \lambda^2 \left(u_{l+1}^n - 2u_l^n + u_{l-1}^n \right) \qquad l = 0, \ldots, N$$

The Dirichlet or fixed condition is the simplest to implement (at both ends):

$$u_0^n = u_N^n = 0 \qquad \text{Dirichlet}$$

In practice, this means that we simply do not need to include the points at l=0 or l=N in our calculation, so we have an update operating over N-1 points, instead of N+1:

$$u_l^{n+1} = 2u_l^n - u_l^{n-1} + \lambda^2 \left(u_{l+1}^n - 2u_l^n + u_{l-1}^n \right) \qquad l = 1, \ldots, N-1$$

Numerical Boundary Conditions: Neumann

The Neumann condition is harder---we will not use it in this course, but it comes up in the setting of room acoustics! We want to enforce the equivalents of:

$$\partial_x u(0,t) = \partial_x u(L,t) = 0$$

Examine the scheme at (for example) the left end grid point l=0:

$$u_0^{n+1} = 2u_0^n - u_0^{n-1} + \lambda^2 \left(u_1^n - 2u_0^n + u_{-1}^n\right)$$

This seems to require access to the grid point at l=-1, which is not part of the domain! We can set this using an approximation to the Neumann condition

$$u_1^n = u_{-1}^n \qquad u_{N-1}^n = u_{N+1}^n \qquad \text{Neumann}$$

This gives a complete update. At l=0, e.g.,:

$$u_0^{n+1} = 2u_0^n - u_0^{n-1} + \lambda^2 \left(2u_1^n - 2u_0^n\right)$$

Now, we are operating over the full N+1 points at every time step!:

Vector-matrix update form

It is very convenient to represent such updates over grids in terms of vectors and matrices. This has the advantage of reducing the complexity of your code (making it more readable) and also allowing for the use of fast sparse matrix operations. This is the way to go in Matlab, for sure.

First, at a given time step, consolidate all the values of the grid function into a vector (N-1 values for Dirichlet or N+1 for Neumann), as

$$\mathbf{u}^n = [u_1, \ldots, u_{N-1}]^T \qquad \text{Dirichlet} \qquad \mathbf{u}^n = [u_0, \ldots, u_N]^T \qquad \text{Neumann}$$

Now, note that the linear operator δ_{xx} can be represented as a matrix $\mathbf{D}_{xx}$

$$\mathbf{D}_{xx} = \frac{1}{h^2} \underbrace{\begin{bmatrix} -2 & 1 & \cdot & \cdot & \cdot \\ 1 & -2 & 1 & \cdot & \cdot \\ \cdot & \ddots & \ddots & \ddots & \cdot \\ \cdot & \cdot & 1 & -2 & 1 \\ \cdot & \cdot & \cdot & 1 & -2 \end{bmatrix}}_{\text{Dirichlet}(N-1)\times(N-1)} \qquad \mathbf{D}_{xx} = \frac{1}{h^2} \underbrace{\begin{bmatrix} -2 & 2 & \cdot & \cdot & \cdot \\ 1 & -2 & 1 & \cdot & \cdot \\ \cdot & \ddots & \ddots & \ddots & \cdot \\ \cdot & \cdot & 1 & -2 & 1 \\ \cdot & \cdot & \cdot & 2 & -2 \end{bmatrix}}_{\text{Neumann}(N+1)\times(N+1)}$$

Boundary conditions are now “built in” to the matrices directly…which are sparse! Try learning about the function spdiags in Matlab…

Vector-matrix update form

A total update is now of the form:

$$\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n - \mathbf{u}^{n-1} \qquad \mathbf{B} = 2\mathbf{I} + c^2k^2\mathbf{D}_{xx}$$

Here is a little piece of pseudocode now for the run-time loop:

```
initialise u1, u2;

for n=1:last_time_step

      u = B*u1-u2;  % perform scheme update

      u2 = u1;      % shift values
      u1 = u;

end
```

There's more to it, of course, but this basic form persists even for extremely complex musical instrument and room acoustics simulation designs…