UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 204 *Numerical methods*

Approximating solutions to the wave equation

Douglas Wilhelm Harder, LEL, M.Math.
dwharder@uwaterloo.ca
dwharder@gmail.com

Introduction

- In this topic, we will
  - Introduce the wave equation
  - Convert the wave equation to a finite-difference equation
  - Discuss the additional initial conditions required
  - Look at an implementation in MATLAB
  - Look at four examples

Wave equation

- The *wave equation* models the transfer of energy through waves

$$\frac{\partial^2}{\partial t^2} u(\mathbf{x},t) = c^2 \nabla^2 u(\mathbf{x},t)$$

  - The value c is the *wave speed,* which is equal to how quickly a wave can move through the medium

- If the heat transfer is restricted to one dimension, this simplifies to

$$\frac{\partial^2}{\partial t^2} u(\mathbf{x},t) = c^2 \frac{\partial^2}{\partial x^2} u(x,t)$$

  - This is the case for a guitar string or a Slinky® or an electromagnetic wave travelling down a wire

Wave equation

- In one dimension, this says:

$$\frac{\partial^2}{\partial t^2} u(x,t) = c^2 \frac{\partial^2}{\partial x^2} u(x,t)$$

  - The acceleration is proportional to the concavity of the wave in space
  - If the concavity is locally zero (the wave is flat), there is no acceleration

Finite-difference approximation

- In one dimension, we can substitute our two approximations:

$$\frac{u(x,t-\Delta t)-2u(x,t)+u(x,t+\Delta t)}{(\Delta t)^2}=c^2\,\frac{u(x-h,t)-2u(x,t)+u(x+h,t)}{h^2}$$

- We can rewrite this as follows:

$$\begin{aligned}u(x,t+\Delta t)&=\boxed{2u(x,t)-u(x,t-\Delta t)}+(c\Delta t)^2\,\frac{u(x-h,t)-2u(x,t)+u(x+h,t)}{h^2}\\&=\boxed{u(x,t)+\Delta t\,\frac{u(x,t)-u(x,t-\Delta t)}{\Delta t}}+(c\Delta t)^2\,\frac{u(x-h,t)-2u(x,t)+u(x+h,t)}{h^2}\end{aligned}$$

  – Compare this with Taylor series:

$$f(t+\Delta t)=f(t)+f^{(1)}(t)(\Delta t)+\frac{1}{2}f^{(2)}(t)(\Delta t)^2$$

Finite-difference approximation

- Suppose we have a string:
  - We could pluck that string and let go
  - We have two boundary conditions: the end-points of the plucked string are fixed
- We have a second derivative with respect to time
  - This requires not only an initial condition,
    but also an initial velocity
  - Often, the initial rate-of-change will be zero:
    - We are plucking the string and just about to let go

Finite-difference approximation

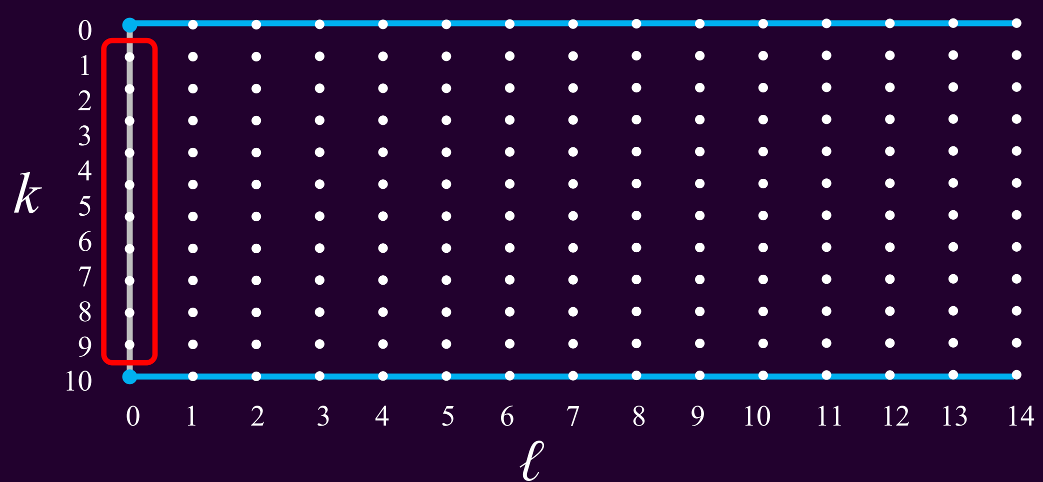
- We don't know what $u(x, t)$ is, so we will approximate it
  - First, divide the interval $[a, b]$ into n_x sub-intervals, each of width h
  - Thus, $x_k = a + kh$ so $x_0 = a$ and $x_{n_x} = b$

- Next, we cannot approximate the solution at each point in time, so we will break time into steps
  - Define $t_\ell = t_0 + \ell\Delta t$

- We will try to approximate $u(x_k, t_\ell)$
  - As before, $u(x_k, t_\ell) \approx u_{k,\ell}$

Finite-difference approximation

- As with the heat equation, we require an initial state of the string or other medium being oscillated as well as boundary conditions
  - For example, $u_0(x) \approx \sin(x)$



Finite-difference approximation

- So now what?

$$u(x,t+\Delta t) = 2u(x,t) - u(x,t-\Delta t) + (c\Delta t)^2 \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2}$$

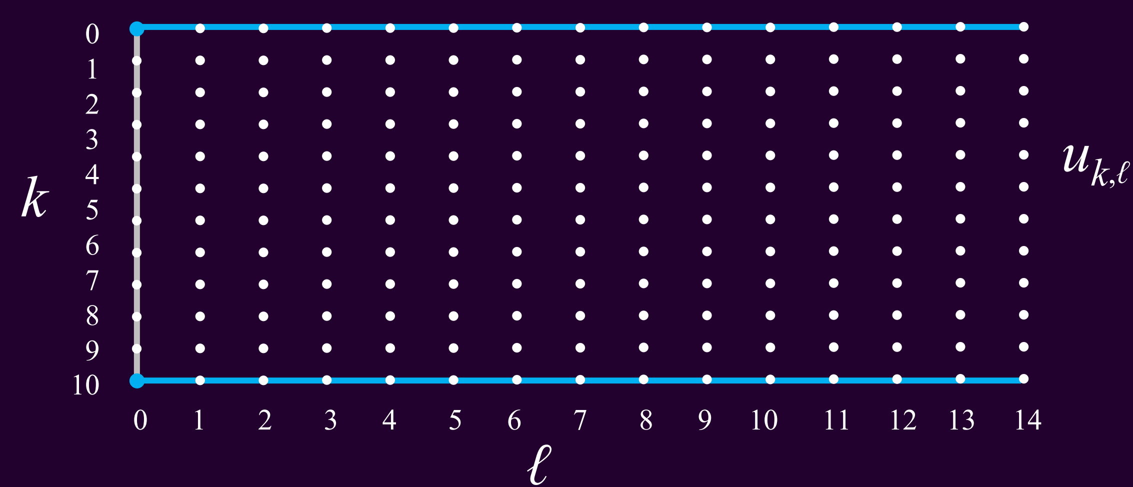
$$u(x_k,t_\ell+\Delta t) = 2u(x_k,t_\ell) - u(x_k,t_\ell-\Delta t) + (c\Delta t)^2 \frac{u(x_k-h,t_\ell) - 2u(x_k,t_\ell) + u(x_k+h,t_\ell)}{h^2}$$

$$u(x_k,t_{\ell+1}) = 2u(x_k,t_\ell) - u(x_k,t_{\ell-1}) + (c\Delta t)^2 \frac{u(x_{k-1},t_\ell) - 2u(x_k,t_\ell) + u(x_{k+1},t_\ell)}{h^2}$$

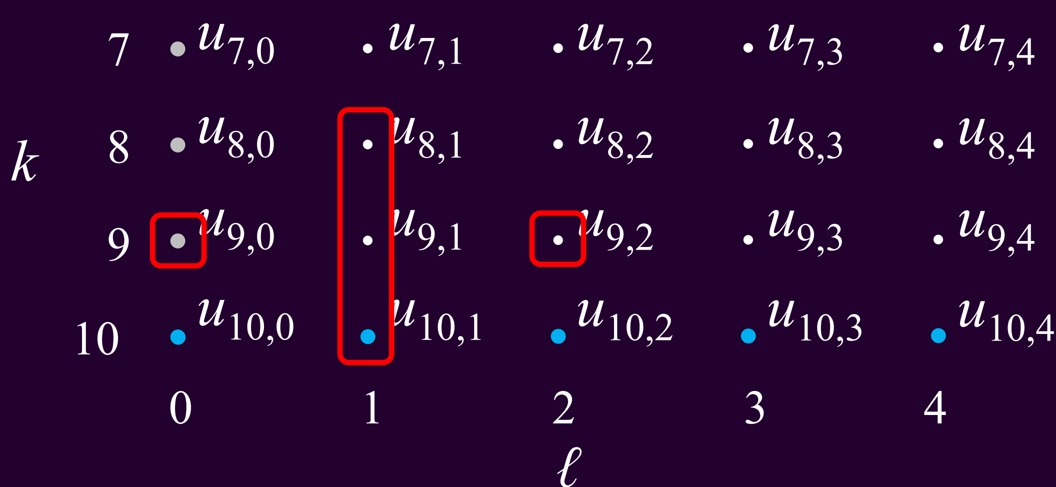
$$u_{k,\ell+1} = 2u_{k,\ell} - u_{k,\ell-1} + (c\Delta t)^2 \frac{u_{k-1,\ell} - 2u_{k,\ell} + u_{k+1,\ell}}{h^2}$$

Finite-difference approximation

- Let's zoom in:

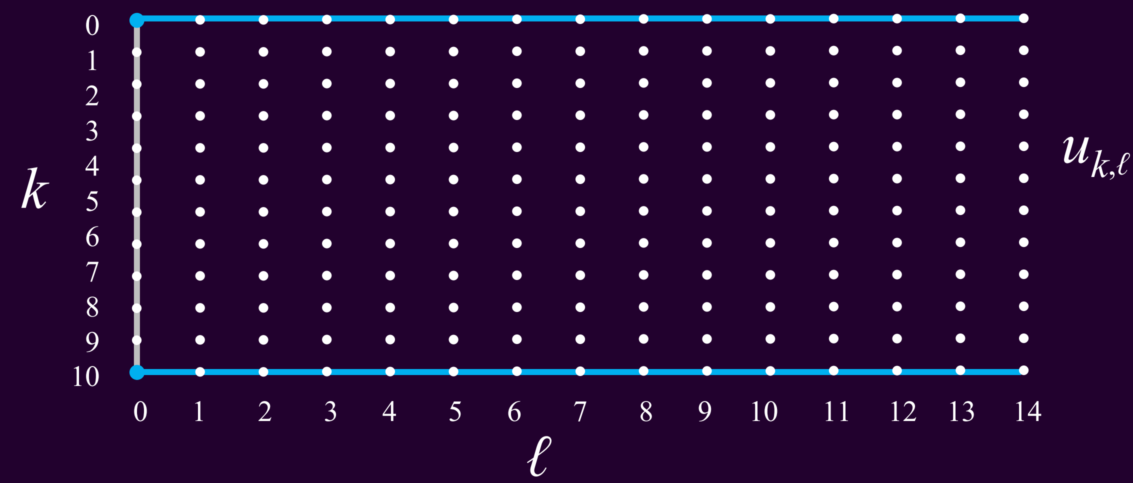


$$u_{k,\ell+1} = 2u_{k,\ell} - u_{k,\ell-1} + (c\Delta t)^2 \frac{u_{k-1,\ell} - 2u_{k,\ell} + u_{k+1,\ell}}{h^2}$$

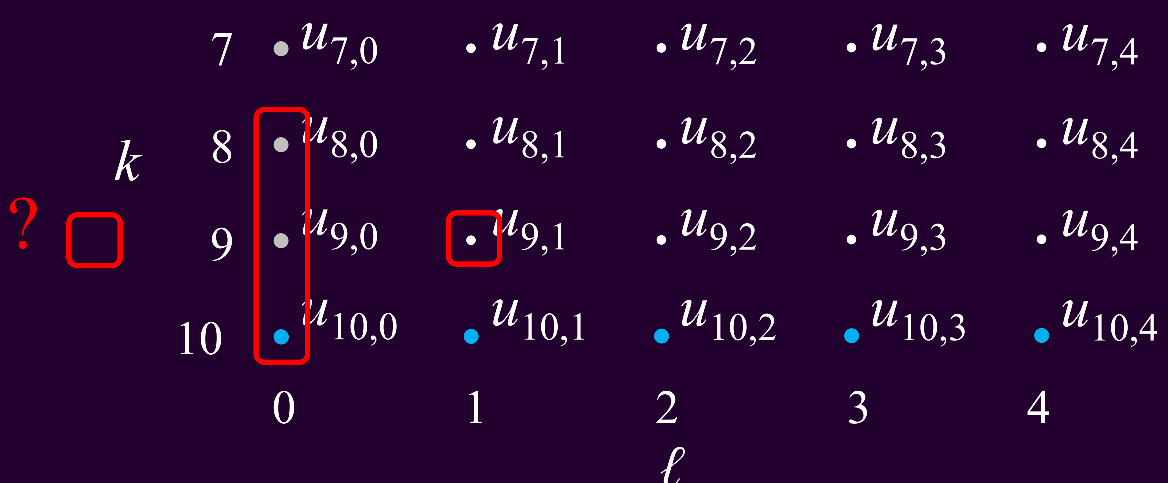


Finite-difference approximation

- Let’s zoom in:



$$u_{k,\ell+1} = 2u_{k,\ell} - u_{k,\ell-1} + (c\Delta t)^2 \frac{u_{k-1,\ell} - 2u_{k,\ell} + u_{k+1,\ell}}{h^2}$$



Finite-difference approximation

- So now what?

$$u_0^{(1)}(x_k) \approx \frac{u_{k,1} - u_{k,-1}}{2\Delta t}$$

$$2u_0^{(1)}(x_k)\Delta t - u_{k,1} \approx -u_{k,-1}$$

$$u_{k,1} = 2u_0(x_k) - u_{k,-1} + (c\Delta t)^2 \frac{u_{k-1,0} - 2u_{k,0} + u_{k+1,0}}{h^2}$$

$$u_{k,1} = 2u_0(x_k) + 2u_0^{(1)}(x_k)\Delta t - u_{k,1} + (c\Delta t)^2 \frac{u_{k-1,0} - 2u_{k,0} + u_{k+1,0}}{h^2}$$

$$2u_{k,1} = 2u_0(x_k) + 2u_0^{(1)}(x_k)\Delta t + (c\Delta t)^2 \frac{u_{k-1,0} - 2u_{k,0} + u_{k+1,0}}{h^2}$$

$$u_{k,1} = u_0(x_k) + u_0^{(1)}(x_k)\Delta t + \frac{1}{2}(c\Delta t)^2 \frac{u_{k-1,0} - 2u_{k,0} + u_{k+1,0}}{h^2}$$

$$y(t+\Delta t) = y(t) + y^{(1)}(t)\Delta t + \frac{1}{2}y^{(1)}(t)(\Delta t)^2$$

Restrictions

- There is one restriction to this algorithm:

$$\frac{\Delta t c}{h} < 1$$

  - A reasonable strategy: given c and h, suppose we want to approximate the solution from t_0 to t_f
    - We want $n_t \Delta t = t_f - t_0$ so $\Delta t = \frac{t_f - t_0}{n_t}$
    - Thus, let's ensure $\frac{t_f - t_0}{n_t} \frac{c}{h} \le \frac{1}{2}$
    - That is, $\frac{1}{n_t} \le \frac{h}{2c\left(t_f - t_0\right)}$

$$n_t \ge \frac{2c\left(t_f - t_0\right)}{h} \qquad n_t = \left\lceil \frac{2c\left(t_f - t_0\right)}{h} \right\rceil$$

Implementation

```
function [xs, ts, Us] = wave( c, x_rng, t_rng, u_init, du_init,
                                                    u_bndry, u_dirichlet, nx )
  h = (x_rng(2) - x_rng(1))/nx;

  nt = ceil( 2.0*c*(t_rng(2) - t_rng(1))/h );
  dt = (t_rng(2) - t_rng(1))/nt;

  xs = linspace( x_rng(1), x_rng(2), nx + 1 )';
  ts = linspace( t_rng(1), t_rng(2), nt + 1 );

  Us = zeros( nx + 1, nt + 1 );
```

Implementation

```
Us(2:nx, 1) = u_init( xs(2:nx) );

dirichlet = u_dirichlet( ts(1) );
boundary  =     u_bndry( ts(1) );

if dirichlet(1)
  Us(1, 1) = boundary(1);
else
  Us(1, 1) = -2.0/3.0*boundary(1)*h + 4.0/3.0*Us(2, 1) ...
                                    - 1.0/3.0*Us(3, 1);
end

if dirichlet(2)
  Us(nx+1, 1) = boundary(2);
else
  Us(nx+1, 1) = 2.0/3.0*boundary(2)*h + 4.0/3.0*Us(nx, 1) ...
                                      - 1.0/3.0*Us(nx-1, 1);
end
```

Implementation

```
Us(2:nx, 2) = Us(2:nx, 1) + du_init( xs(2:nx) )*dt ...
                          + 0.5*(c*dt)^2*diff( Us(:, 1), 2 )/h^2;

dirichlet = u_dirichlet( ts(2) );
boundary  =     u_bndry( ts(2) );

if dirichlet(1)
  Us(1, 2) = boundary(1);
else
  Us(1, 2) = -2.0/3.0*boundary(1)*h + 4.0/3.0*Us(2, 2) ...
                                    - 1.0/3.0*Us(3, 2);
end

if dirichlet(2)
  Us(nx+1, 2) = boundary(2);
else
  Us(nx+1, 2) = 2.0/3.0*boundary(2)*h + 4.0/3.0*Us(nx, 2) ...
                                      - 1.0/3.0*Us(nx-1, 2);
end
```

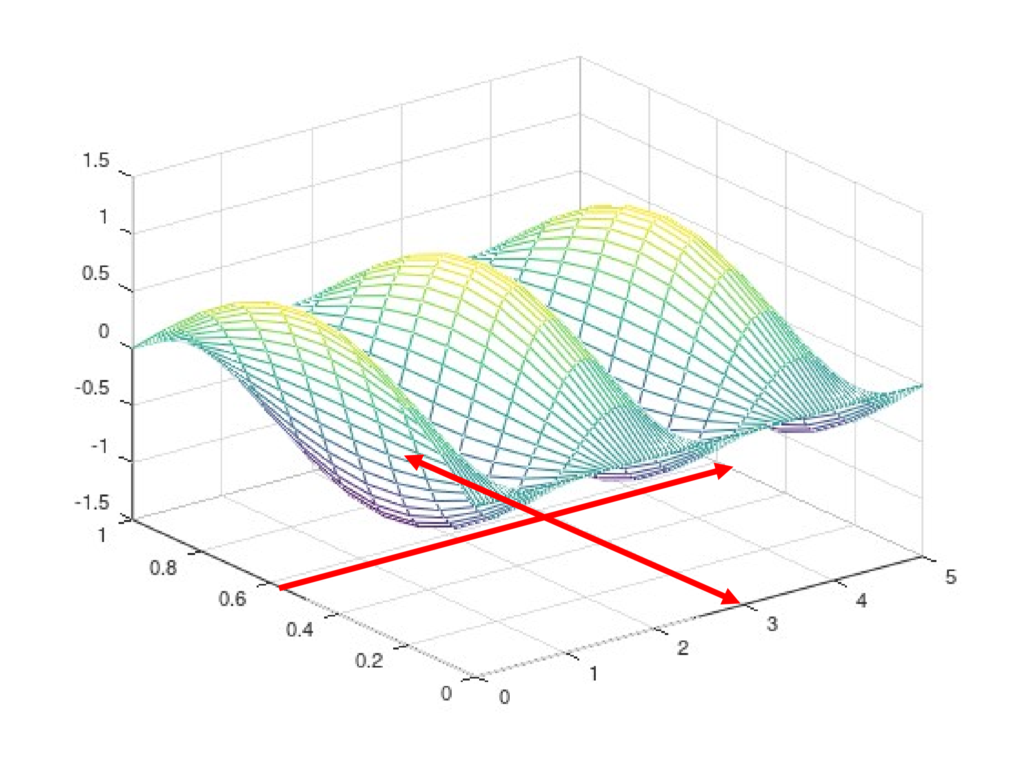
Implementation

```
for ell = 2:nt
  Us(2:nx, ell + 1) = 2*Us(2:nx, ell) - Us(2:nx, ell-1)
                                    + (c*dt)^2*diff( Us(:, ell), 2 )/h^2;

  dirichlet = u_dirichlet( ts(ell + 1) );
  boundary  =     u_bndry( ts(ell + 1) );

  if dirichlet(1)
    Us(1, ell+1) = boundary(1);
  else
    Us(1, ell+1) = -2.0/3.0*boundary(1)*h + 4.0/3.0*Us(2, ell+1) ...
                                        - 1.0/3.0*Us(3, ell+1);
  end

  if dirichlet(2)
    Us(nx+1, ell+1) = boundary(2);
  else
    Us(nx+1, ell+1) = 2.0/3.0*boundary(2)*h + 4.0/3.0*Us(nx, ell+1) ...
                                          - 1.0/3.0*Us(nx-1, ell+1);
  end
 end
end
```

Example 1

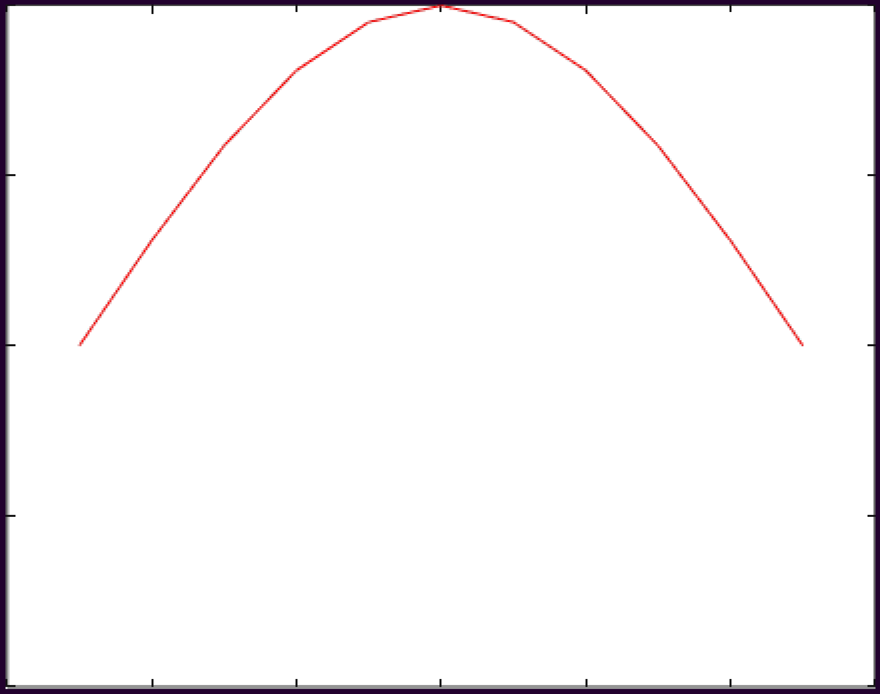
- Consider this example:

```
>>  u1_in = @(x)( sin( pi*x ) );
>> du1_in = @(x)( zeros( size( x ) ) );
>>  u1_bn = @(t)( [0.0, 0.0]' );
>>  u1_di = @(t)( [true, true]' );
>> [x1s, t1s, U1s] = wave( 0.3, [0, 1], [0, 0.5], u1_in, du1_in, u1_bn, u1_di, 10 );
>> mesh( t1s, x1s, U1s );
```



Example 1

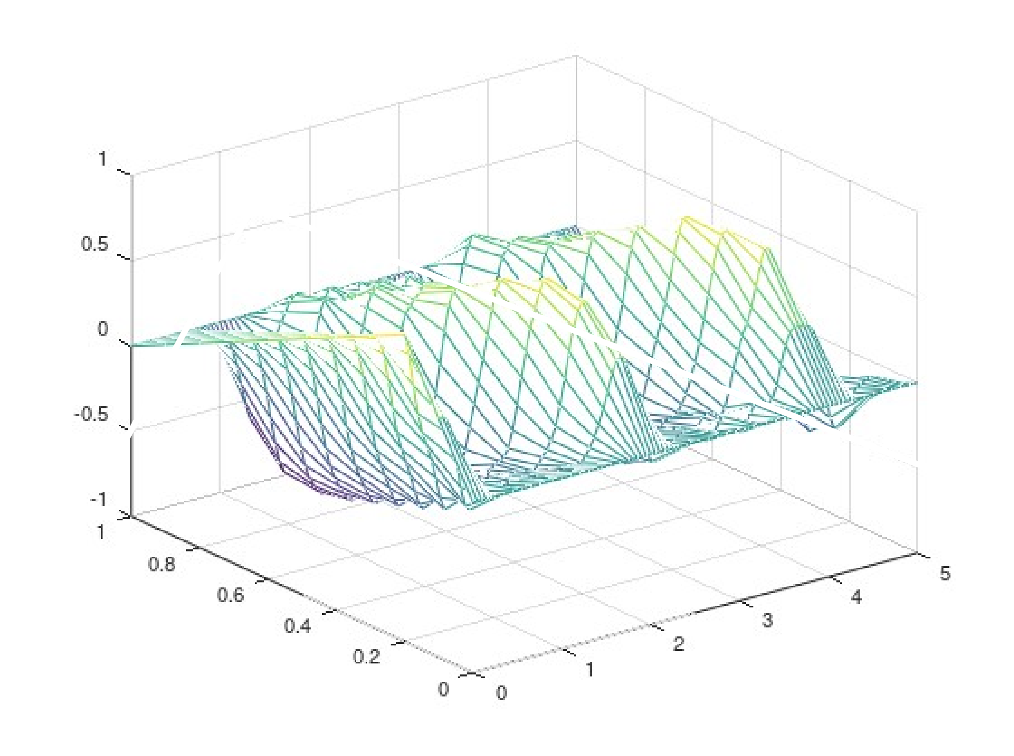
- Recalling that $n_x = 10$, we see how the wave oscillates over time



Example 2

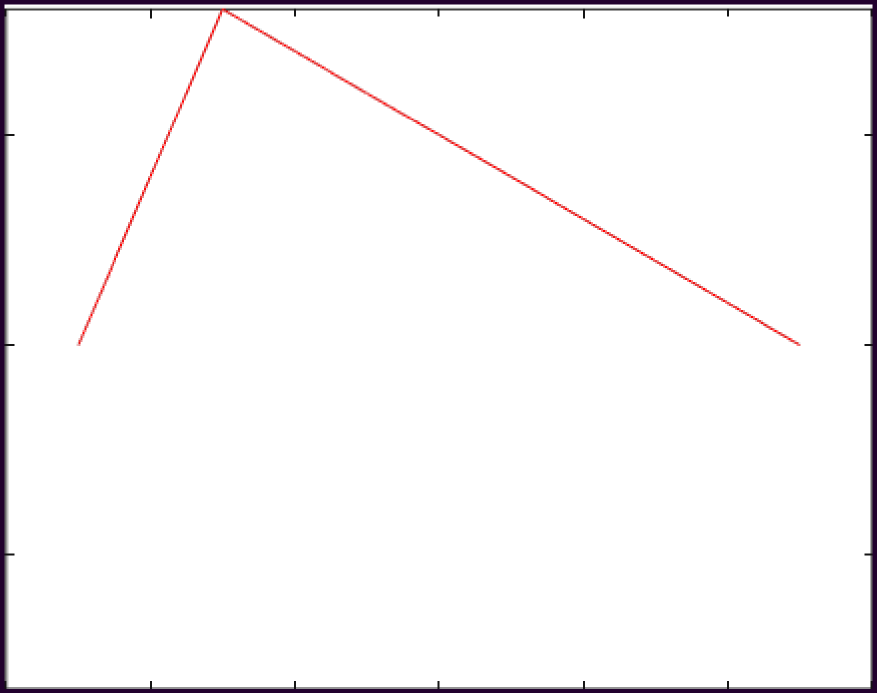
- Consider this example:

```
>>  u2_in = @(x)( 4.0*(x <= 0.2).*x + (x > 0.2).*(1.0 - x) );
>> du2_in = @(x)( zeros( size( x ) ) );
>>  u2_bn = @(t)( [0.0, 0.0]' );
>>  u2_di = @(t)( [true, true]' );
>> [x2s, t2s, U2s] = wave( 0.3, [0, 1], [0, 0.5], u1_in, du2_in, u2_bn, u2_di, 10 );
>> mesh( t2s, x2s, U2s );
```



Example 2

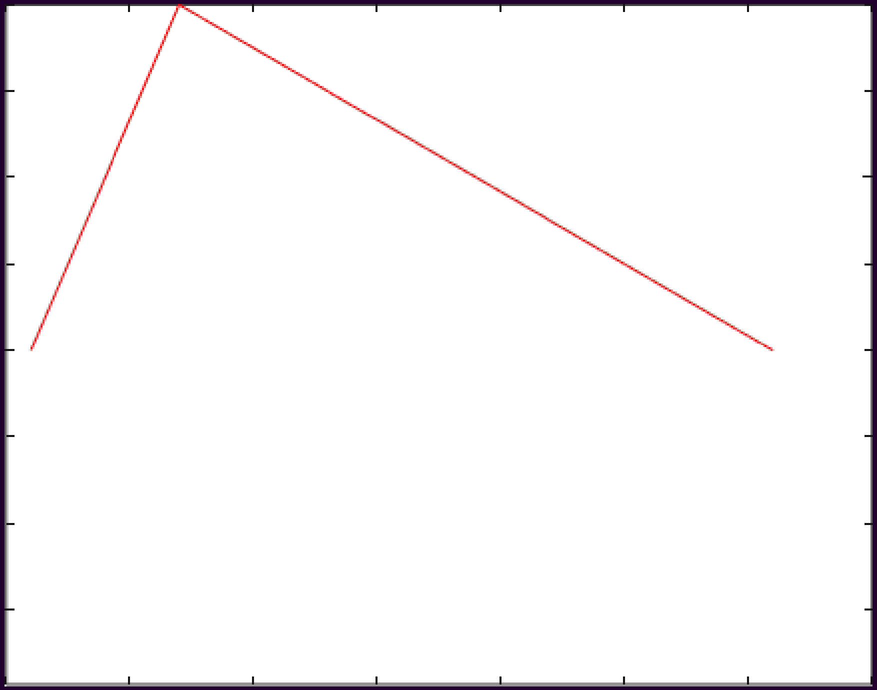
- The wave appears to reflect through the center, not vertically



Example 2

- Try it again with 30 sub-intervals:

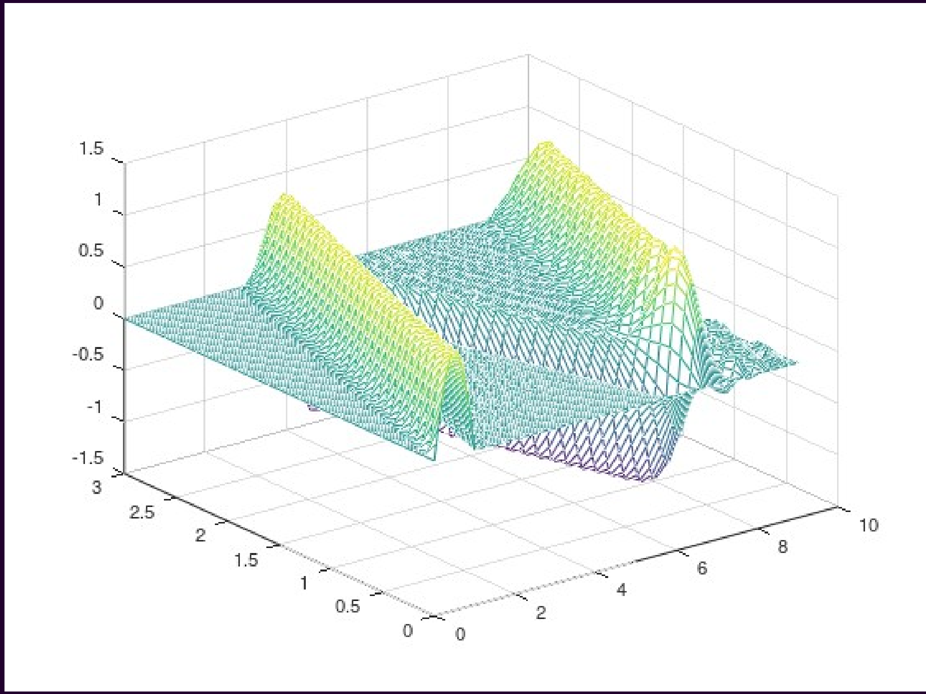
```
>> [x2s, t2s, U2s] = wave( 0.3, [0, 1], [0, 0.5], u1_in, du2_in, u2_bn, u2_di, 30 );
```



Example 3

- Consider this example:

```
>>  u3_in = @(x)( zeros( size( x ) ) );
>> du3_in = @(x)( zeros( size( x ) ) );
>>  u3_bn = @(t)( [(t <= 1)*sin(pi*t), 0]' );
>>  u3_di = @(t)( [true, true]' );
>> [x3s, t3s, U3s] = wave( 1.0, [0, 3], [0, 9], u3_in, du3_in, u3_bn, u3_di, 30 );
>> mesh( t3s, x3s, U3s );
```



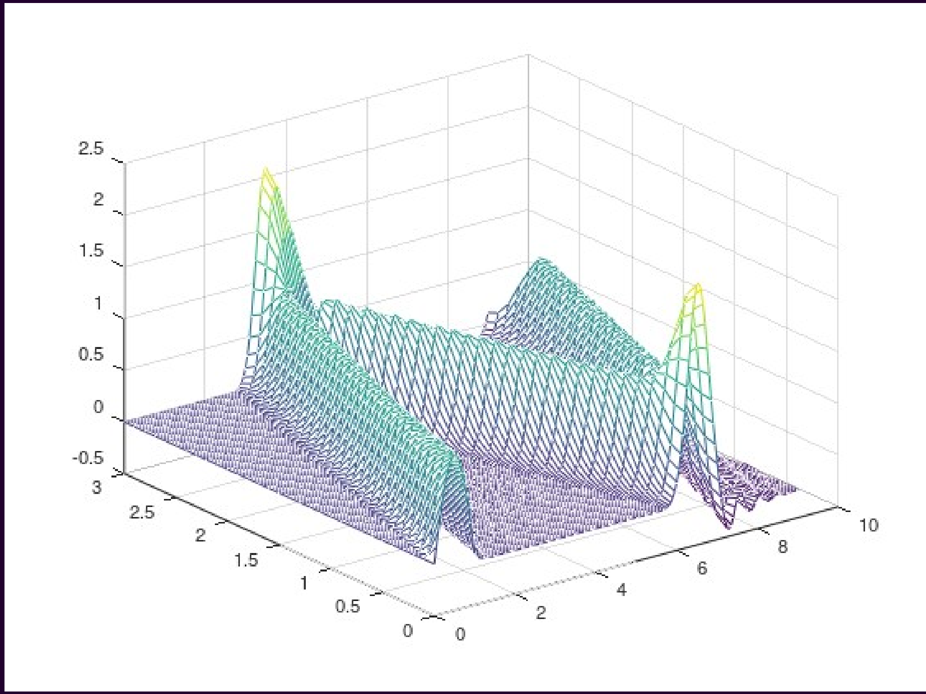
Example 3

- The Slinky® travels back and forth between the fixed end points

Example 4

- Consider this example:

```
>>  u4_in = @(x)( zeros( size( x ) ) );
>> du4_in = @(x)( zeros( size( x ) ) );
>>  u4_bn = @(t)( [(t <= 1)*sin(pi*t), 0]' );
>>  u4_di = @(t)( [t <= 1, false]' );
>> [x4s, t4s, U4s] = wave( 1.0, [0, 3], [0, 9], u4_in, du4_in, u4_bn, u4_di, 30 );
>> mesh( t4s, x4s, U4s );
```



Example 4

- Note the water wave bounces back and forth between the two boundaries—the edges of a pool do not fix the water height

Summary

- Following this topic, you now
  - Understand how to approximate the wave equation with a finite-difference equation
  - Understand we require one more initial condition:
    the initial speed
  - Are aware of how to implement such a solution in MATLAB
  - Have seen four examples including Dirichlet (fixed) and Neumann (fixed derivative) boundary conditions

References

[1] https://en.wikipedia.org/wiki/Wave_equation

Acknowledgments

None so far.

Colophon

These slides were prepared using the Cambria typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`. Mathematical equations are prepared in MathType by Design Science, Inc.

Examples may be formulated and checked using Maple by Maplesoft, Inc.

The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer