COL865: Special Topics in Computer Applications

Physics-Based Animation

11 — Continuum models and PDEs

Assignments and feedback

- ***Assignment 1*** submissions received, will grade soon
  - Feedback?
- ***Assignment 2*** on rigid bodies will be posted today

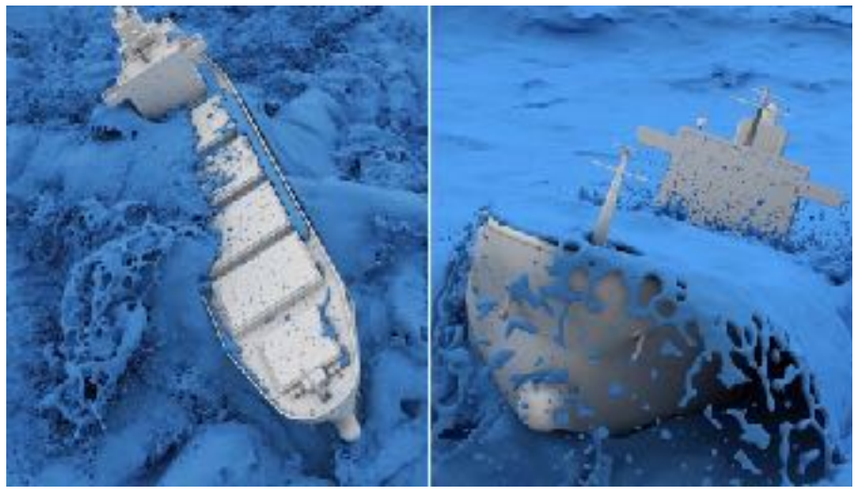
- ***General feedback*** about the course?
  - Fill in feedback form on Moodle (anonymous)

Paper discussions

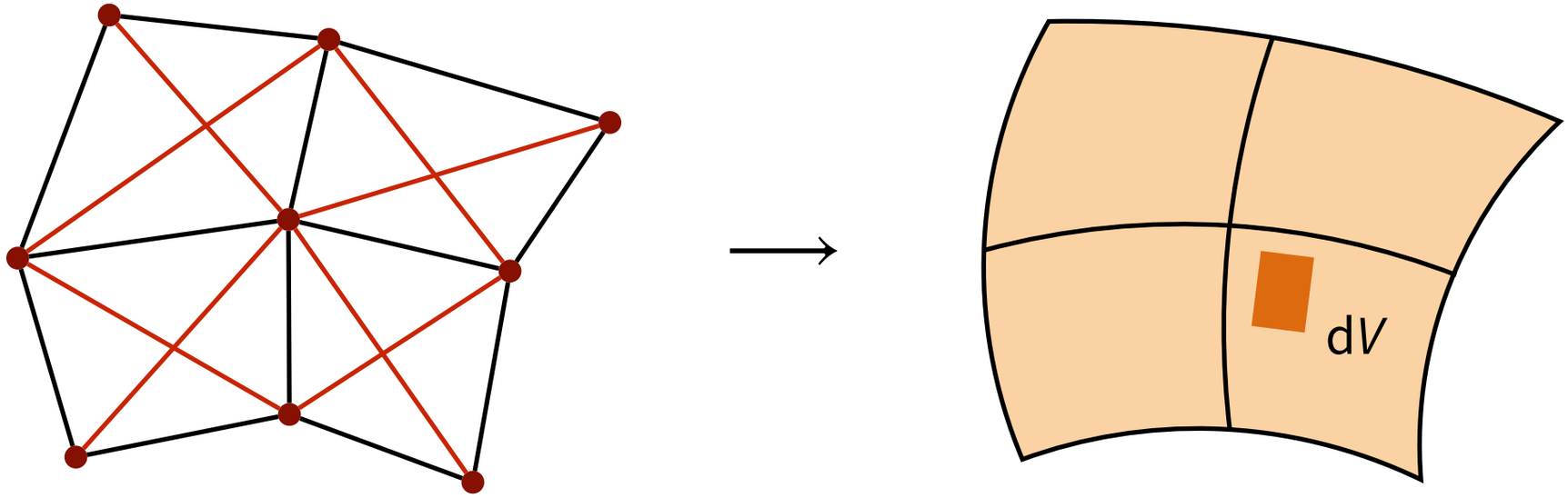
Wednesday, 12 Sep 2018

- ***Dwizotman***: Batty et al., “A Fast Variational Framework for Accurate Solid- Fluid Coupling”, 2007

- ***Amar***: Ihmsen et al., “Implicit Incompressible SPH”, 2014



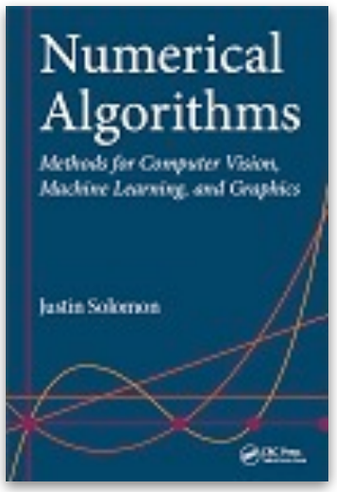
Discrete vs. continuous models



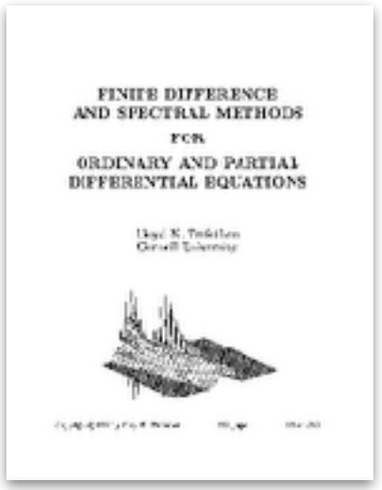
Assume materials are ***infinitely divisible***
(reasonable assumption above microstructure level)

- Mathematical representation?
- Equations of motion?
- Discretization?

Recommended reading

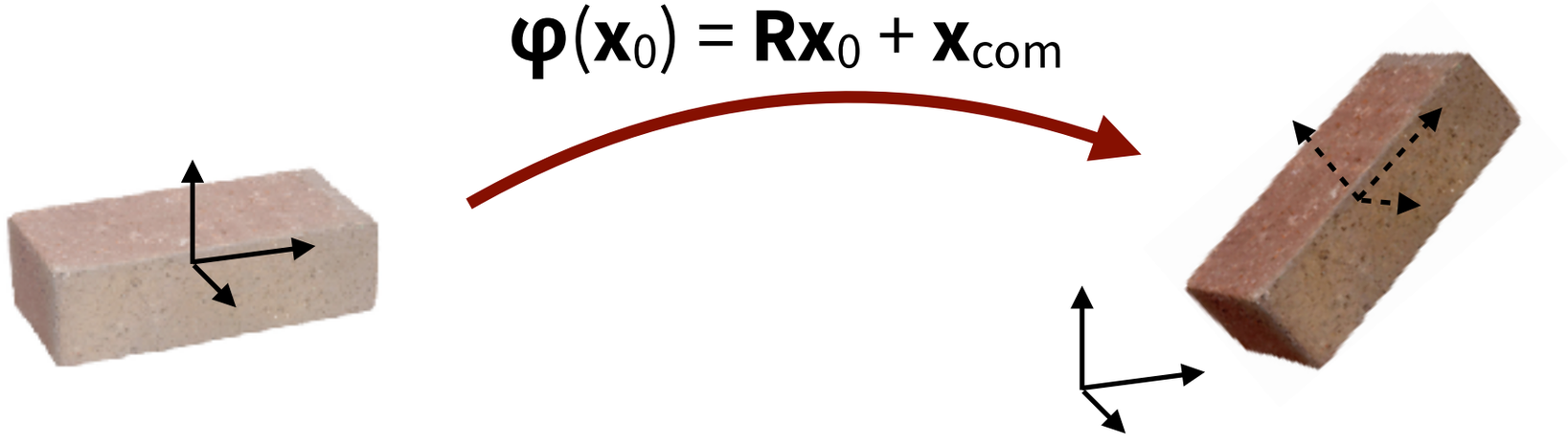


- **Basic intro to PDEs**:
  Solomon, *Numerical Algorithms*, 16.2–3



- **Optional (lots more theoretical detail)**:
  Trefethen, *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*, Ch. 3.1–3 and 4.1–4

Continuum modeling



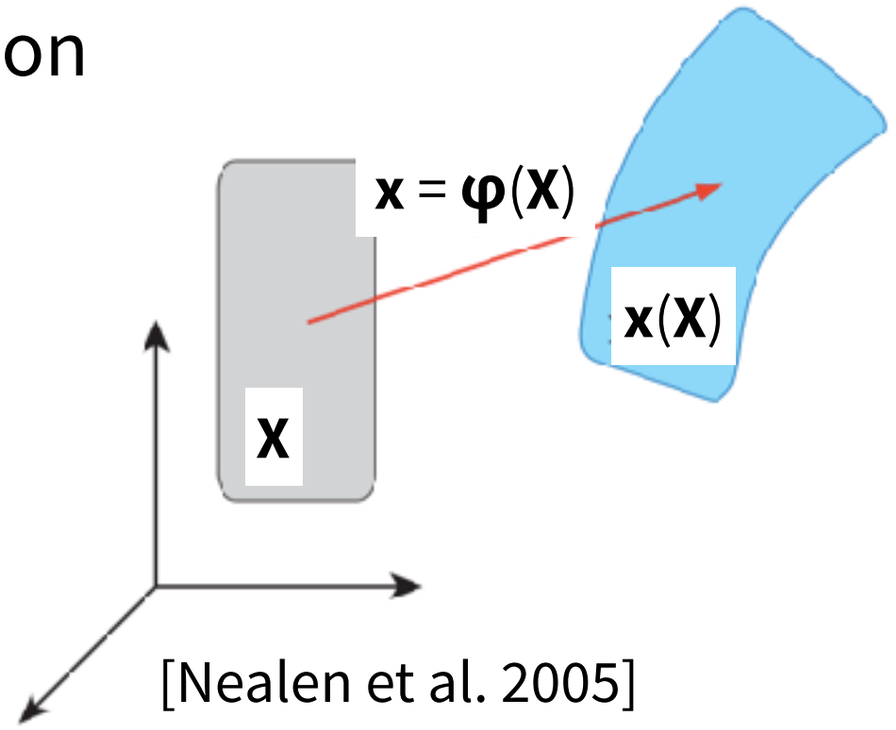
Rigid body:

φ : body → world is a rigid transformation

Deformable body:

φ is an arbitrary continuous function

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, t)$$
$$\mathbf{v} = \partial\boldsymbol{\varphi}(\mathbf{X}, t)/\partial t$$



[Nealen et al. 2005]

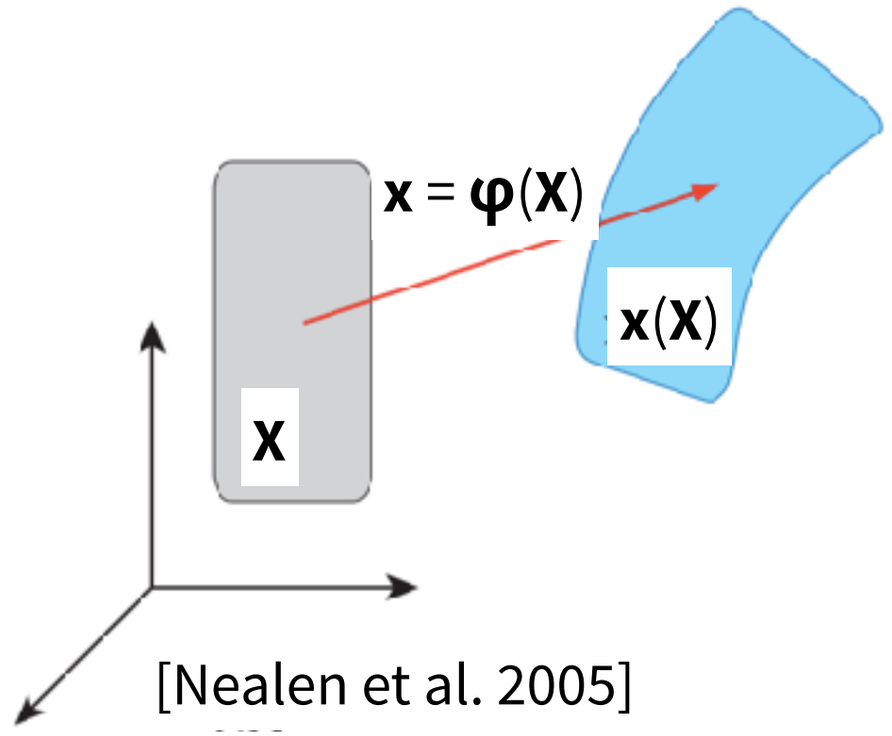
Continuum modeling

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, t)$$

Compare with mass-spring system:

- index (i_1, i_2, i_3)
  → reference coordinate $\mathbf{X}$

- mass m_i → density ρ



[Nealen et al. 2005]

- spring vector $\mathbf{x}_i - \mathbf{x}_j$ → deformation gradient $d\boldsymbol{\varphi}/d\mathbf{X}$
  e.g. stretching ratio along $X_1 = \|\partial\boldsymbol{\varphi}/\partial X_1\|$

- Equation of motion:

$$m_i \, d^2\mathbf{x}_i/dt^2 = (\ldots \mathbf{x}_i - \mathbf{x}_j \ldots)$$
$$\rightarrow \; \rho \, \partial^2\boldsymbol{\varphi}/\partial t^2 = (\ldots \partial\boldsymbol{\varphi}/\partial\mathbf{X} \ldots)$$

Continuum models

State is usually a function $\mathbb{R}^n \to \mathbb{R}^m$

- ***Elastic solid***: reference space $\mathbb{R}^3 \to$ world space $\mathbb{R}^3$,
  cloth: reference space $\mathbb{R}^2 \to$ world space $\mathbb{R}^3$,
  fluid flow: position $\mathbb{R}^3 \to$ velocity $\mathbb{R}^3$,
  water waves: position $\mathbb{R}^2 \to$ height $\mathbb{R}^1$

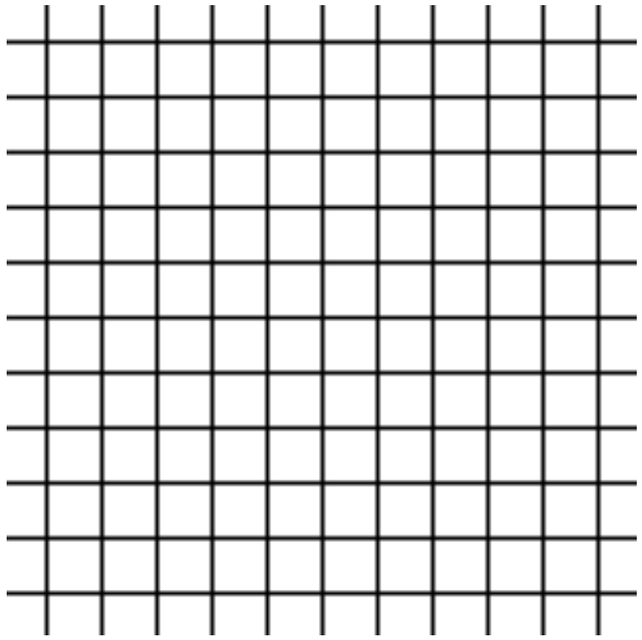
Force involves ***spatial derivatives*** of function,
Newton's second law becomes a partial differential equation

- How to store a function $\mathbb{R}^n \to \mathbb{R}^m$, compute spatial derivatives?
- How to solve a PDE involving such a function?

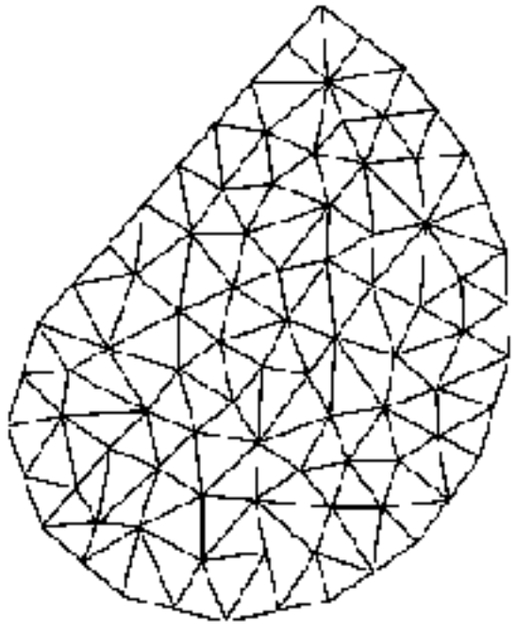
Spatial discretization

Spatial discretizations

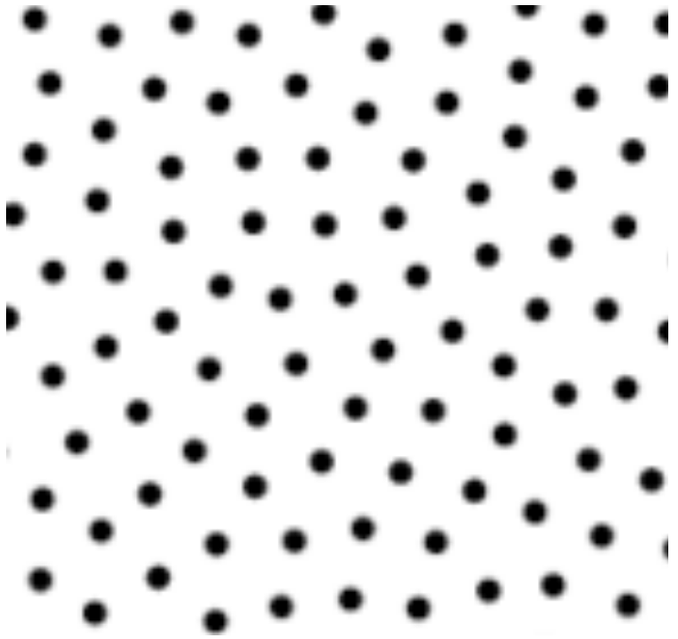
In 1D, not much choice. In *n*D, lots of options!



Grids



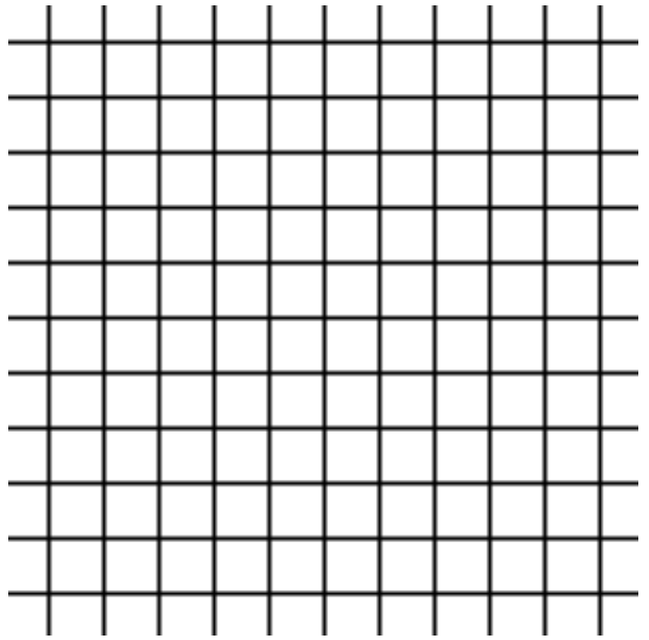
Meshes



Particles

Grids

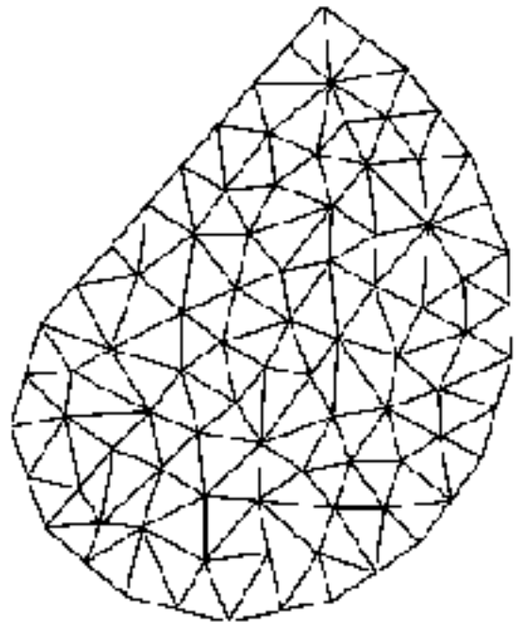
Values arranged in a regular array



- Easy indexing, efficient memory layout
- ***Reconstruction***: just apply 1D interpolation along each axis
- ***Differentiation***: 1D finite difference formulae
- Extra work needed to support non-axis-aligned boundaries
- Non-uniformity only along axes

Meshes

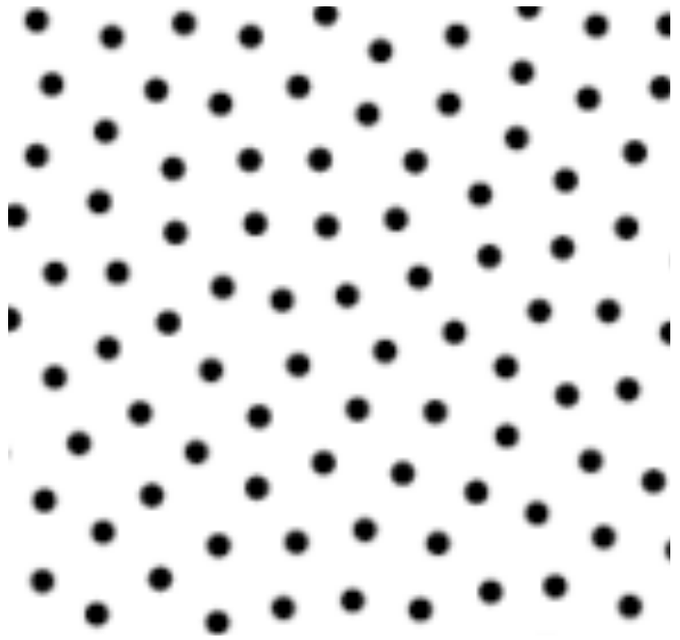
Divide space into elements (usually triangles, tetrahedra), put samples at vertices



- Requires more complex data structures
- ***Reconstruction***: linear / polynomial within each element
- ***Differentiation***: naively, gradient only defined in element interior.
  Sophisticated discretization via finite element method
- Can fit any shape with piecewise smooth boundary
- Element should be “well-shaped”, can be adaptive

Particles

Collection of unorganized point samples



- No connectivity needed, but requires neighbourhood search
- ***Reconstruction***: weighted average of nearby samples

$$\langle f \rangle(\mathbf{x}) = \frac{\sum_i f_i \, w(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_i w(\|\mathbf{x} - \mathbf{x}_i\|)}$$

- ***Differentiation***: differentiate reconstruction function

Various modifications used in practice: smoothed particle hydrodynamics, moving least squares

Numerical methods for PDEs

Numerical methods for PDEs

Spatial discretizations ≈ data structures

Numerical methods ≈ algorithms

Vast subject, many different techniques

- ***Finite differences*** (only on grids)
- ***Finite elements*** (usually on meshes and grids)
- ***Meshless methods*** (on particles)
- Many others less used in graphics: ***finite volumes***, ***spectral*** & ***pseudospectral*** methods

An example PDE: The wave equation

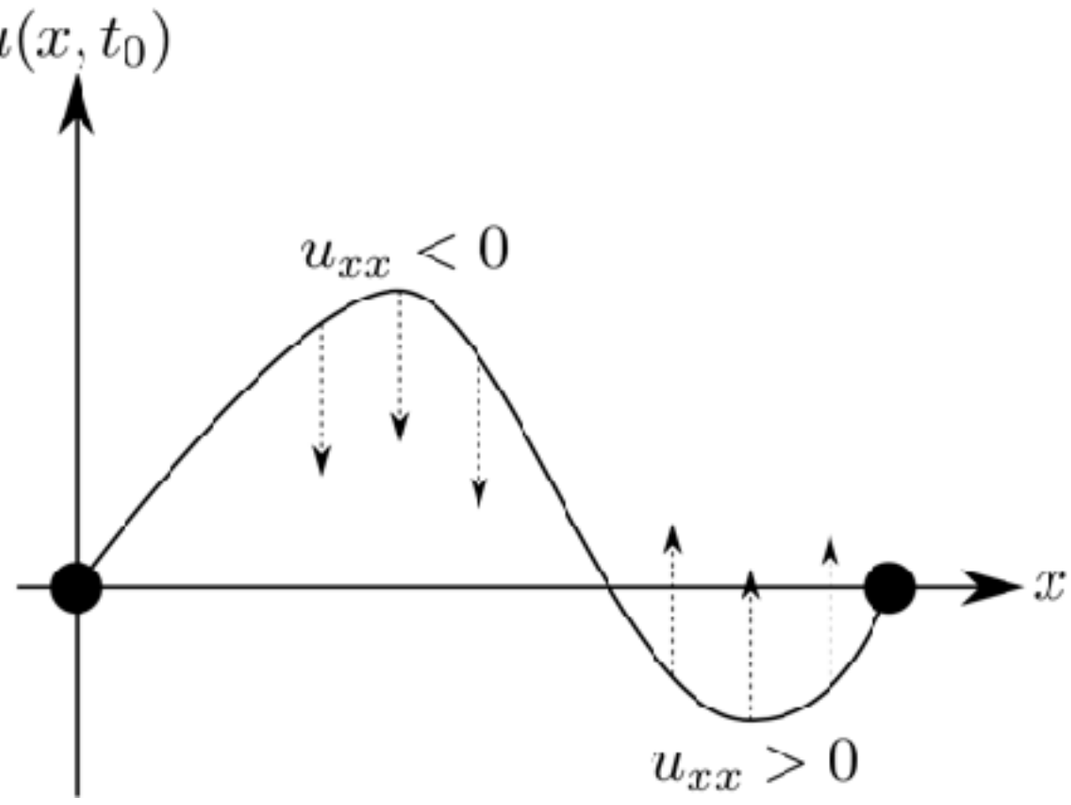
Describes propagation of waves in a medium (sound waves, waves in stretched string/membrane, surface waves in water, …)

In 1D (string): $u_{tt} = c\, u_{xx}$

On 2D surface: $u_{tt} = c\, \nabla^2 u$

Like with ODE, need to specify initial conditions:

$$u(x, t_0) = u_0(x)$$
$$u_t(x, t_0) = \dot{u}_0(x)$$



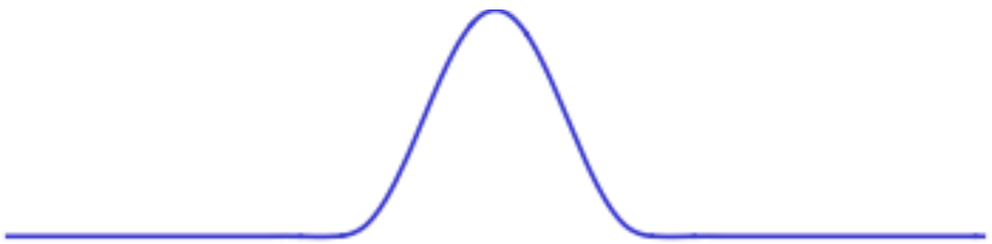
[Solomon]

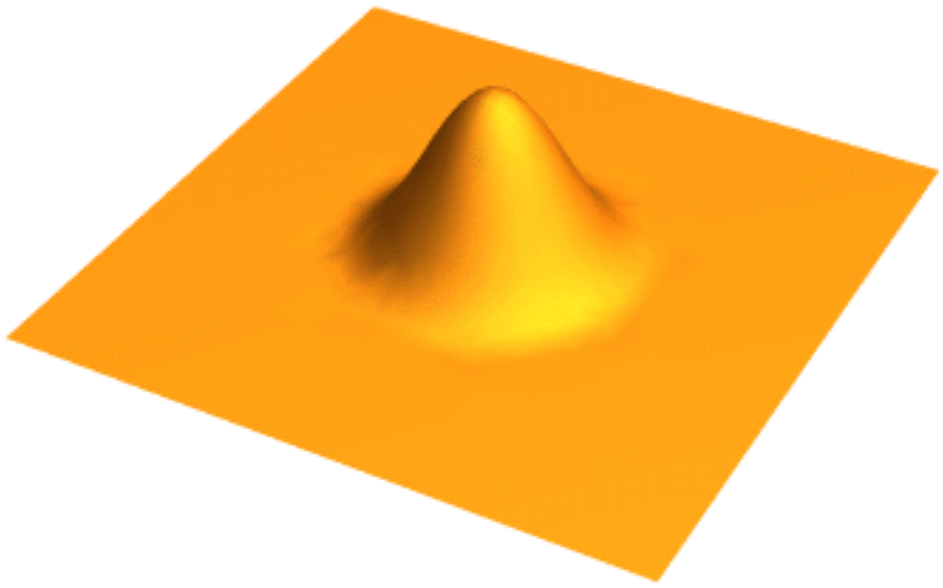
The wave equation

$$u_{tt} = c\,\nabla^2 u$$

$$u(x, t_0) = \ldots$$
$$u_t(x, t_0) = \ldots$$

On an unbounded domain, this is enough:

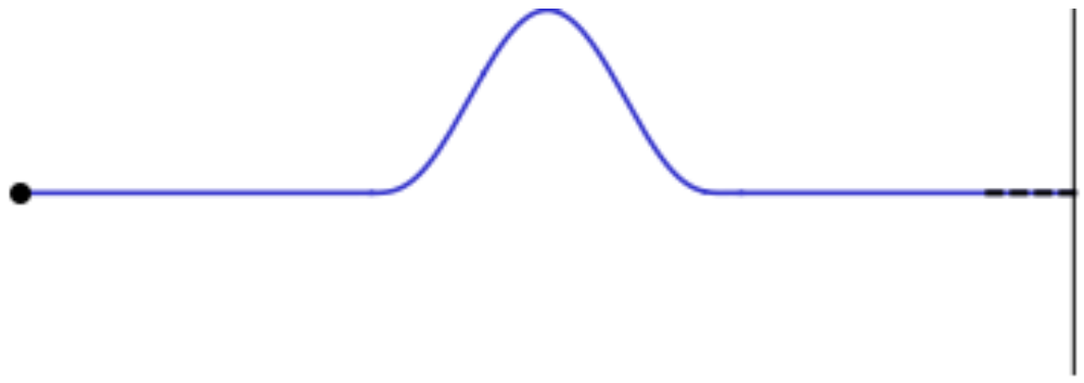




Boundary conditions

Usually domain Ω is bounded: what happens at the boundary?

- ***Dirichlet*** BCs: $u(x, t) = \ldots$ for all $x \in \partial\Omega$
- ***Neumann*** BCs: $u_x(x, t) = \ldots$ for all $x \in \partial\Omega$
  - In nD, only fix ***normal*** derivative $\mathbf{n} \cdot \nabla u$
- ***Mixed*** BCs: Dirichlet & Neumann on subsets of $\partial\Omega$

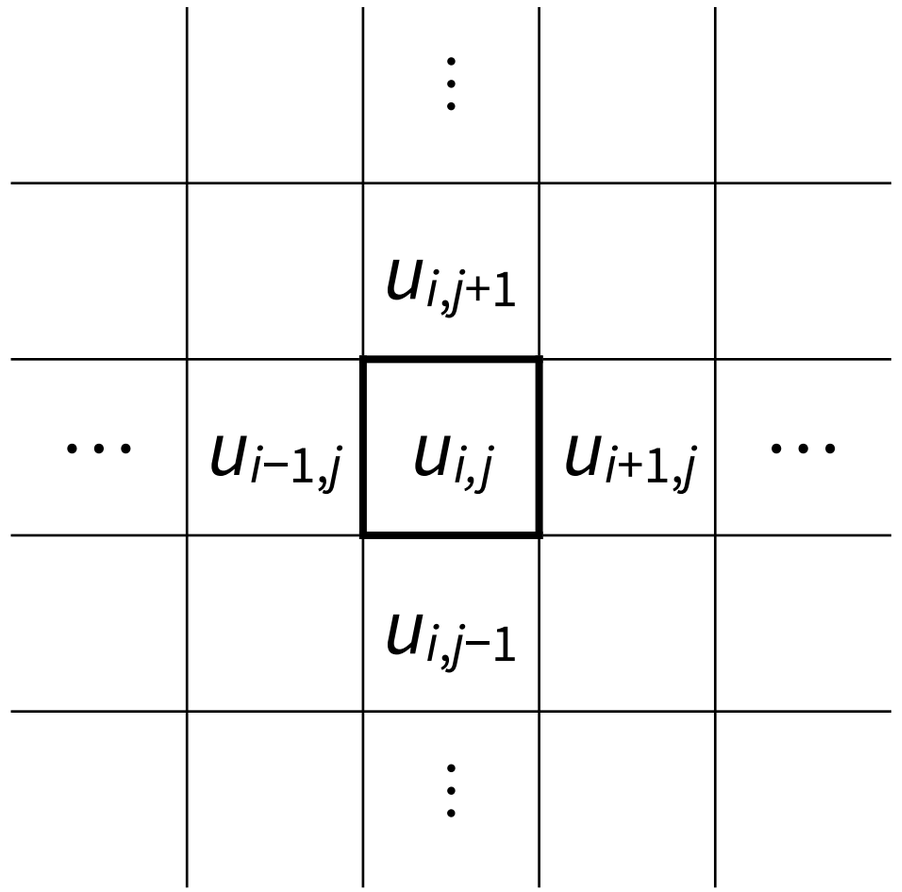


- ***Robin*** BCs: $u(x, t) + a\, u_x(x, t) = \ldots$
- ***Absorbing*** BCs: act as if infinite domain (hard!)

Finite differences for the wave equation

Discretize space using a grid:

$$u(x, y, t) \approx u_{i,j}(t)$$

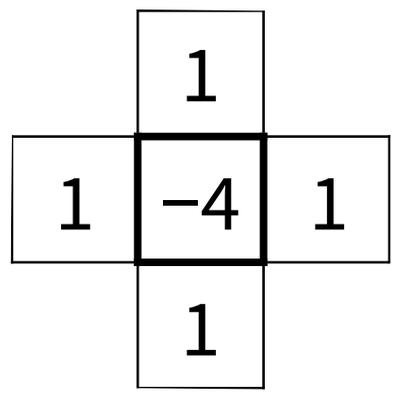


Discretize spatial derivatives using 1D finite difference formulas:

$$\partial^2 u/\partial t^2 = c\,\nabla^2 u$$
$$= c\,(\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2)$$

$$\partial^2 u/\partial x^2 \approx (u_{i-1,j} - 2\,u_{i,j} + u_{i+1,j})/\Delta x^2$$
$$\partial^2 u/\partial y^2 \approx (u_{i,j-1} - 2\,u_{i,j} + u_{i,j+1})/\Delta x^2$$

$$\Rightarrow\ \nabla^2 u \approx (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4\,u_{i,j})/\Delta x^2$$



Five-point stencil for 2D Laplacian

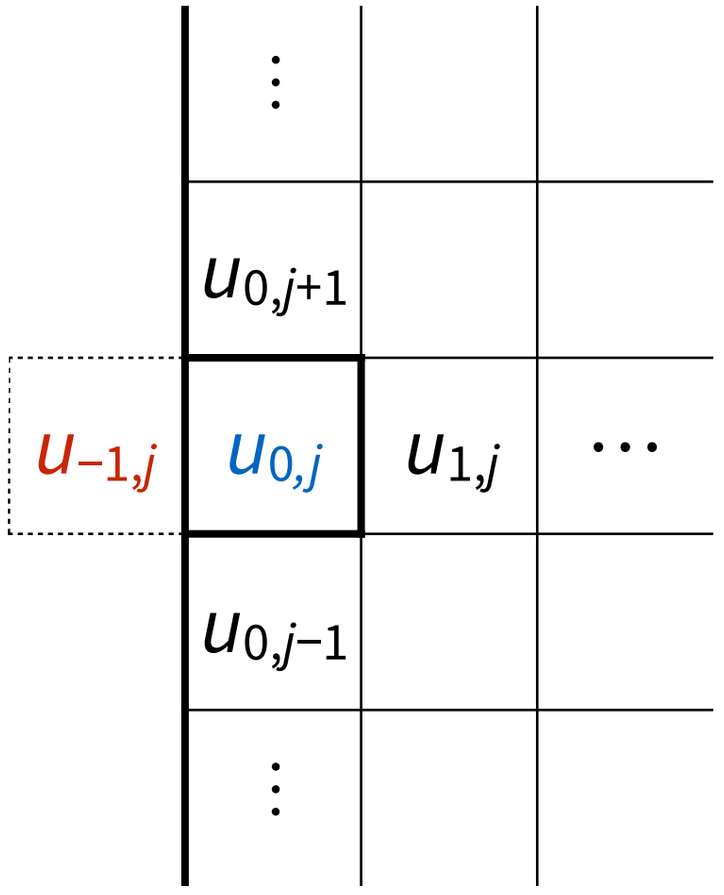
Boundary conditions

Dirichlet: $u = f$ on $\partial\Omega$

- Just fix value of $u_{i,j} = f$ on boundary

Neumann: $\mathbf{n} \cdot \nabla u = g$ on $\partial\Omega$

- First-order: Fix value of $u_{0,j}$ so that $(u_{1,j} - u_{0,j})/\Delta x = g$
- Second-order: "***ghost nodes***"
  - Compute fictitious value of $u_{-1,j}$ so that $(u_{1,j} - u_{-1,j})/(2\,\Delta x) = g$
  - Plug into finite difference Laplacian



Finite differences

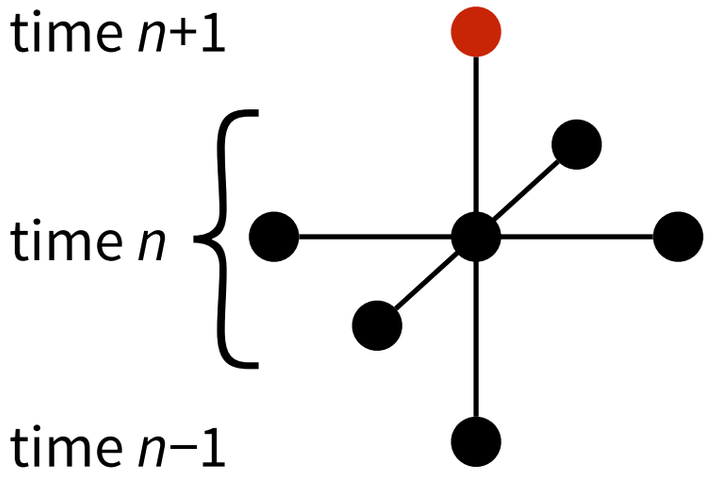
After spatial discretization, PDE becomes system of ODEs:

$$\partial^2 u / \partial t^2 = c \, \nabla^2 u$$

$$\longrightarrow \; \mathrm{d}^2 u_{i,j} / \mathrm{d}t^2 = (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4\, u_{i,j}) / \Delta x^2$$

Solve with any time integration scheme

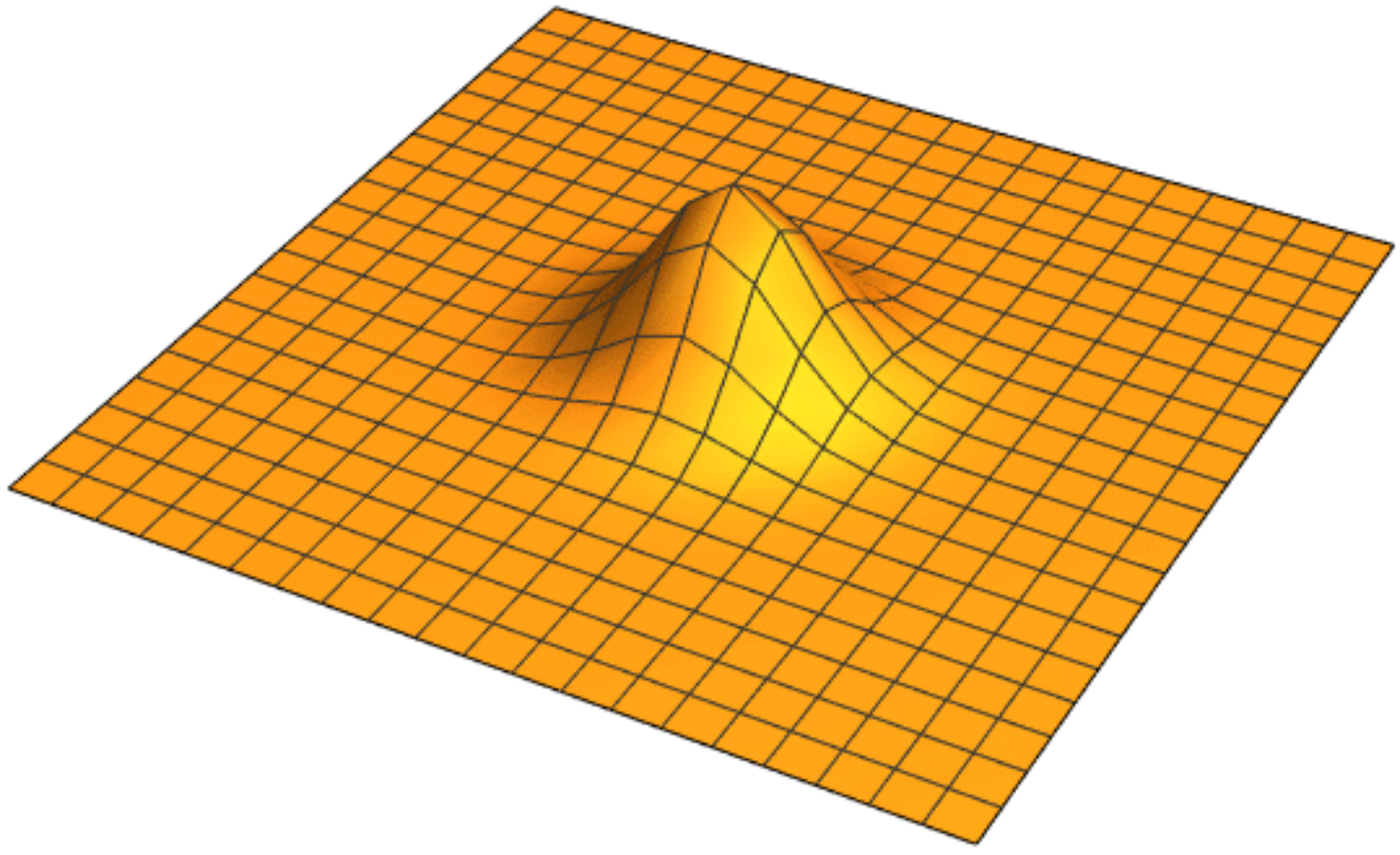
e.g. 3-point difference in time:



$$\frac{1}{\Delta t^2}(u_{i,j}^{n-1} + u_{i,j}^{n+1} - 2u_{i,j}^{n})$$

$$= \frac{1}{\Delta x^2}(u_{i-1,j}^{n} + u_{i+1,j}^{n} + u_{i,j-1}^{n} + u_{i,j+1}^{n} - 4u_{i,j}^{n})$$

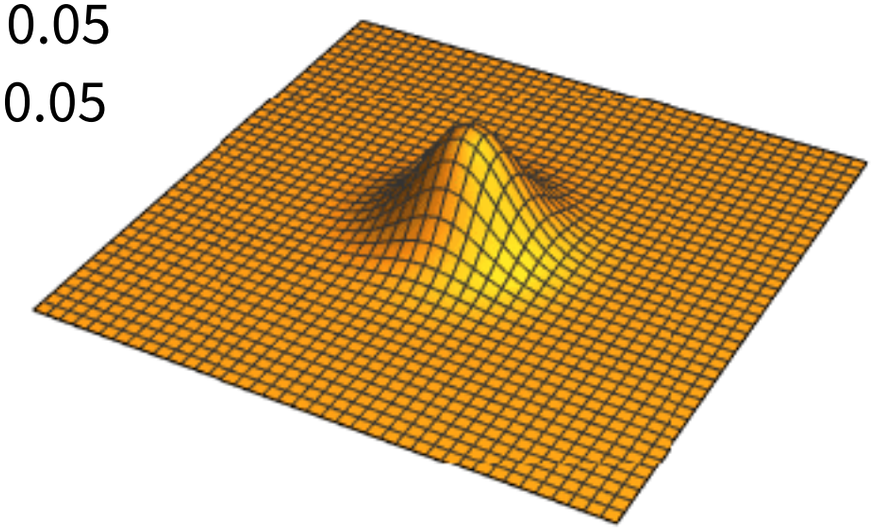
Finite difference solution



$c = 1, \Delta x = 0.1, \Delta t = 0.05$

Stability

$\Delta x = 0.05$
$\Delta t = 0.05$



Can't always reduce Δx without changing Δt

Stability via ***von Neumann analysis*** (Trefethen Ch. 3.5, 4.4): amplification factor of ***all*** Fourier modes should be ≤ 1

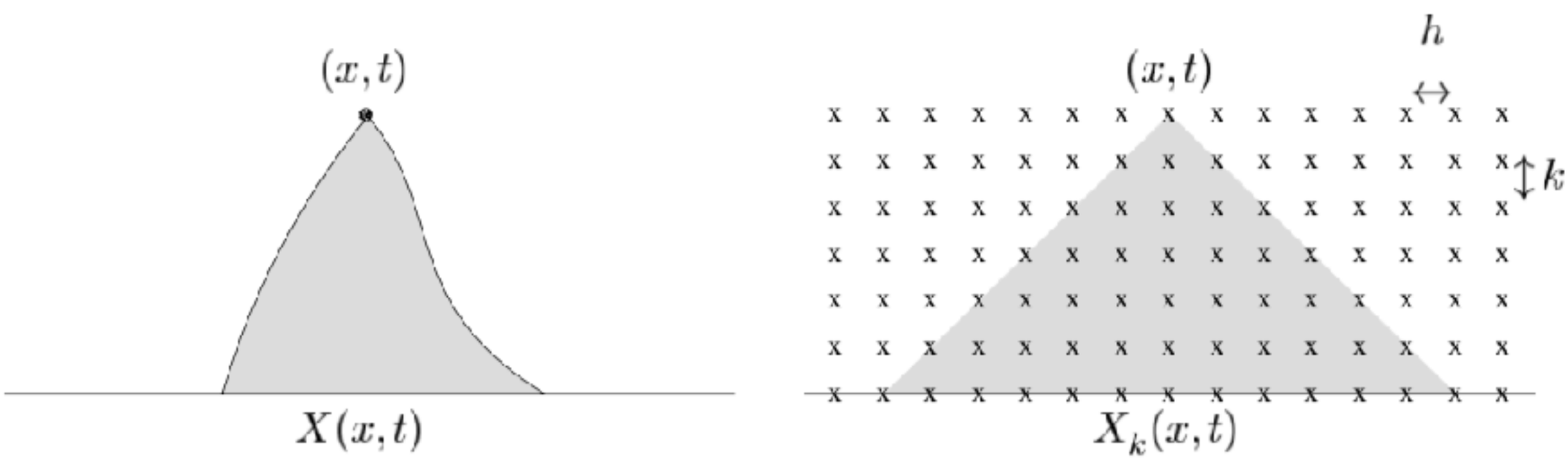
- For wave equation in n dimensions, $\Delta t \leq \Delta x/(n^{1/2}\, c)$

The Courant-Friedrichs-Lewy (CFL) condition

True solution at $u(x, t)$ depends on some subset of initial data u_0

- e.g. waves propagate at finite speed c

Numerical method cannot converge unless it "takes all the necessary data into account" (Trefethen Ch. 4.3)



[Trefethen]

Compare with wave eq. stability: $\Delta t \leq \Delta x/(n^{1/2}\, c)$

Next class

The Navier-Stokes equations for fluids

- Reading: Stam, “Stable Fluids”, 1999

[Zhang et al. 2015]