# CS7GV2: Mathematics of Light and Sound, M.Sc. in Computer Science.

## Lecture #4: Simulation

Fergal Shevlin, Ph.D.

School of Computer Science and Statistics,
Trinity College Dublin

October 7, 2022

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.

- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.

- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.
- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.
- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.

- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.

- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.

- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.

- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.

- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.

- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Analytical versus numerical methods

- For a quadratic polynomial $f(x) = a\,x^2 + b\,x + c$, the roots (zero-crossings) are found with the well-known formula,
$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2\,a}.$$

- In science and engineering it's more-often-than-not the case that problems do not have neat *closed-form* or *analytical* solutions except in very specific circumstances.

- What can we do about it? Approximation, iteration. For example, the "method of bisection" for root finding: guess where a root might be; keep halving the length of an interval around it such that $f(x)$ has different signs at the start and the end.

- Such solutions often described as *numerical methods* because they use numbers (and computers) versus *analytical methods* which use symbols (and thinking.)

# Wave Motion

- We've seen that wave motion is described by the second order PDE known as the wave equation,

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

- We've seen a *closed-form* solution for wave propagation,

$$u(x,t) = R\cos(kx - \omega t) + (1-R)\cos(kx + \omega t).$$

- This is perfect when there are no constraints. For example, light in a homogeneous medium, a wave on an infinitely long string (no end points,) or a sound in a huge volume of air.
- But the closed-form solution doesn't tell us, for example, how a string plucked in a particular way is going to move: `https://tinyurl.com/y4ncymx7`.

# Wave Motion

- We've seen that wave motion is described by the second order PDE known as the wave equation,

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

- We've seen a *closed-form* solution for wave propagation,

$$u(x,t) = R\cos(kx - \omega t) + (1-R)\cos(kx + \omega t).$$

- This is perfect when there are no constraints. For example, light in a homogeneous medium, a wave on an infinitely long string (no end points,) or a sound in a huge volume of air.

- But the closed-form solution doesn't tell us, for example, how a string plucked in a particular way is going to move: `https://tinyurl.com/y4ncymx7`.

# Wave Motion

- We've seen that wave motion is described by the second order PDE known as the wave equation,

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

- We've seen a *closed-form* solution for wave propagation,

$$u(x,t) = R\cos(kx - \omega t) + (1-R)\cos(kx + \omega t).$$

- This is perfect when there are no constraints. For example, light in a homogeneous medium, a wave on an infinitely long string (no end points,) or a sound in a huge volume of air.

- But the closed-form solution doesn't tell us, for example, how a string plucked in a particular way is going to move: `https://tinyurl.com/y4ncymx7`.

# Wave Motion

- We've seen that wave motion is described by the second order PDE known as the wave equation,

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

- We've seen a *closed-form* solution for wave propagation,

$$u(x,t) = R\cos(kx - \omega t) + (1-R)\cos(kx + \omega t).$$

- This is perfect when there are no constraints. For example, light in a homogeneous medium, a wave on an infinitely long string (no end points,) or a sound in a huge volume of air.
- But the closed-form solution doesn't tell us, for example, how a string plucked in a particular way is going to move: `https://tinyurl.com/y4ncymx7`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# Wave Simulation

- When there are specific constraints (also known as conditions,) there is usually no alternative but to *simulate* wave motion in an *iterative* way.
- Iterative means doing more-or-less the same sequence of calculations again and again.
- Usually the current iteration's calculations use results calculated in the previous iteration(s.)
- An iterative simulation can never be perfect. Error is inevitable, for example, because descretization is required.
- Error is typically cumulative so the results become less correct at each iteration.
- There are lots of nice interactive simulations of wave motions available, for example: `https://tinyurl.com/2xrsrz` and `https://tinyurl.com/mtwczmj`.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# EM Wave simulation

- Solve Maxwell's equations to find local wave characteristics at many discrete volumes of space at successive steps in time.
- The results for one discrete volume are used in the calculation of the characteristics of its neighbors.
- One of the most used techniques (e.g. in MEEP) is called *finite difference time domain* (FDTD.)
- Approaches like this in general are called *finite element methods* for the approximate solution of *boundary value problems* with *partial differential equations.*
- Advantages: can deal with complex geometries and different materials.
- Disadvantages: can be very computationally intensive which limits the spatial accuracy or the temporal duration, cf. weather forecasting.

# Initial and Boundary Conditions

- To simulate a specific solution for $u(x,t)$ described by the wave equation,
$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} \quad x \in [0, L],\ t \in [0, T],$$
for a string of length $L$ over a time period $T$, we need:
- two *initial conditions* at time $t = 0$,
$$u(x,0) = I(x), \quad x \in [0, L]$$
$$\frac{\partial}{\partial t} u(x,0) = 0, \quad x \in [0, L]$$
where $I(x)$ specifies the initial shape of the string,
- and two *boundary conditions* at distances $x = 0$ and $x = L$,
$$u(0,t) = 0, \quad t \in [0, T]$$
$$u(L,t) = 0, \quad t \in [0, T]$$

# Initial and Boundary Conditions

- To simulate a specific solution for $u(x,t)$ described by the wave equation,

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} \quad x \in [0, L], \; t \in [0, T],$$

  for a string of length $L$ over a time period $T$, we need:

- two *initial conditions* at time $t = 0$,

$$\begin{aligned} u(x,0) &= I(x), \quad x \in [0, L] \\ \frac{\partial}{\partial t} u(x,0) &= 0, \quad x \in [0, L] \end{aligned}$$

  where $I(x)$ specifies the initial shape of the string,

- and two *boundary conditions* at distances $x = 0$ and $x = L$,

$$\begin{aligned} u(0,t) &= 0, \quad t \in [0, T] \\ u(L,t) &= 0, \quad t \in [0, T] \end{aligned}$$

# Initial and Boundary Conditions

- To simulate a specific solution for $u(x,t)$ described by the wave equation,

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} \quad x \in [0, L],\ t \in [0, T],$$

  for a string of length $L$ over a time period $T$, we need:
- two *initial conditions* at time $t = 0$,

$$u(x,0) = I(x), \quad x \in [0, L]$$
$$\frac{\partial}{\partial t} u(x,0) = 0, \quad x \in [0, L]$$

  where $I(x)$ specifies the initial shape of the string,
- and two *boundary conditions* at distances $x = 0$ and $x = L$,

$$u(0,t) = 0, \quad t \in [0, T]$$
$$u(L,t) = 0, \quad t \in [0, T]$$

# Discretization of domain

- Computer operations take a finite amount of time to complete so there can't be infinitely many time steps in the simulation.

  The time period $[0, T]$ has to be descretized, e.g. into intervals of equal duration $\Delta t$,

  $$t_i = i\,\Delta t, \quad i = 0, \ldots N_t \text{ (where } N_t = T/\Delta t.)$$

- Computer memory is finite so there can't be infinitely many distances in the simulation.

  The length $[0, L]$ have to be descretized, e.g. into intervals of equal distance $\Delta x$,

  $$x_j = j\,\Delta x, \quad j = 0, \ldots N_x \text{ (where } N_x = L/\Delta x.)$$

# Discretization of domain

- Computer operations take a finite amount of time to complete so there can't be infinitely many time steps in the simulation.

  The time period $[0, T]$ has to be descretized, e.g. into intervals of equal duration $\Delta t$,

  $$t_i = i\,\Delta t, \quad i = 0, \ldots N_t \text{ (where } N_t = T/\Delta t.)$$

- Computer memory is finite so there can't be infinitely many distances in the simulation.

  The length $[0, L]$ have to be descretized, e.g. into intervals of equal distance $\Delta x$,

  $$x_j = j\,\Delta x, \quad j = 0, \ldots N_x \text{ (where } N_x = L/\Delta x.)$$

# Discretization of domain

- Computer operations take a finite amount of time to complete so there can't be infinitely many time steps in the simulation.

  The time period $[0, T]$ has to be descretized, e.g. into intervals of equal duration $\Delta t$,

  $$t_i = i\,\Delta t, \quad i = 0, \ldots N_t \text{ (where } N_t = T/\Delta t.)$$

- Computer memory is finite so there can't be infinitely many distances in the simulation.

  The length $[0, L]$ have to be descretized, e.g. into intervals of equal distance $\Delta x$,

  $$x_j = j\,\Delta x, \quad j = 0, \ldots N_x \text{ (where } N_x = L/\Delta x.)$$

# Discretization of domain

- Computer operations take a finite amount of time to complete so there can't be infinitely many time steps in the simulation.

  The time period $[0, T]$ has to be descretized, e.g. into intervals of equal duration $\Delta t$,

  $$t_i = i\,\Delta t, \quad i = 0, \dots N_t \text{ (where } N_t = T/\Delta t.)$$

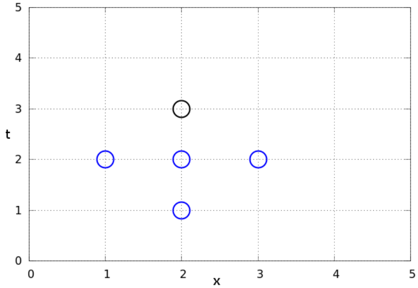
- Computer memory is finite so there can't be infinitely many distances in the simulation.

  The length $[0, L]$ have to be descretized, e.g. into intervals of equal distance $\Delta x$,

  $$x_j = j\,\Delta x, \quad j = 0, \dots N_x \text{ (where } N_x = L/\Delta x.)$$

# Solution mesh

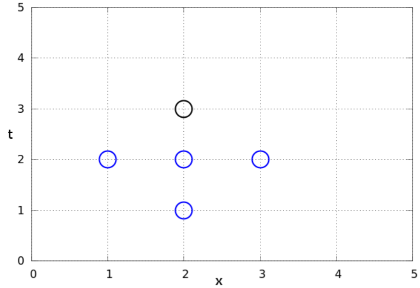
- The discrete points in space and time can be visualized as a two-dimensional *mesh* (or net.)




- The solution for wave height $u(x_j, t_i)$ at each mesh point is found using already-calculated solutions at neighbouring mesh points ...
- ... except for certain exterior mesh points whose values have been specified through the initial conditions, i.e. $I(x)$.

# Solution mesh

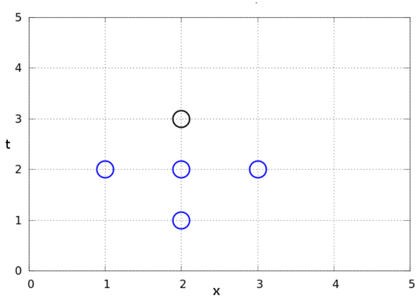
- The discrete points in space and time can be visualized as a two-dimensional *mesh* (or net.)




- The solution for wave height $u(x_j, t_i)$ at each mesh point is found using already-calculated solutions at neighbouring mesh points . . .
- . . . except for certain exterior mesh points whose values have been specified through the initial conditions, i.e. $I(x)$.

# Solution mesh

- The discrete points in space and time can be visualized as a two-dimensional *mesh* (or net.)




- The solution for wave height $u(x_j, t_i)$ at each mesh point is found using already-calculated solutions at neighbouring mesh points ...
- ...except for certain exterior mesh points whose values have been specified through the initial conditions, i.e. $I(x)$.

# Discretization of equations

- *Wave equation.* Use the symmetric second difference approximation of the second derivative,

$$\frac{u(x_j, t_{i+1}) - 2\,u(x_j, t_i) + u(x_j, t_{i-1})}{\Delta t^2} \approx c^2 \frac{u(x_{j+1}, t_i) - 2\,u(x_j, t_i) + u(x_{j-1}, t_i)}{\Delta x^2}.$$

Alternative notation can be used to make the parameters more obvious,

$$\frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta t^2} \approx c^2 \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}, \tag{1}$$

- *Initial condition.* Use the centered first difference approximation of the first derivative,

$$\frac{\partial}{\partial t} u(x_j, t_i) \approx \frac{u_j^{i+1} - u_j^{i-1}}{2\Delta t} \tag{2}$$

Note division by $2\Delta t$ because the difference is between values of $u(x, t)$ separated by two time intervals.

# Discretization of equations

- *Wave equation.* Use the symmetric second difference approximation of the second derivative,

$$\frac{u(x_j, t_{i+1}) - 2\,u(x_j, t_i) + u(x_j, t_{i-1})}{\Delta t^2} \approx$$
$$c^2 \frac{u(x_{j+1}, t_i) - 2\,u(x_j, t_i) + u(x_{j-1}, t_i)}{\Delta x^2}.$$

  Alternative notation can be used to make the parameters more obvious,

$$\frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta t^2} \approx c^2 \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}, \tag{1}$$

- *Initial condition.* Use the centered first difference approximation of the first derivative,

$$\frac{\partial}{\partial t} u(x_j, t_i) \approx \frac{u_j^{i+1} - u_j^{i-1}}{2\Delta t} \tag{2}$$

  Note division by $2\Delta t$ because the difference is between values of $u(x, t)$ separated by two time intervals.

## Discretization of equations

- *Wave equation.* Use the symmetric second difference approximation of the second derivative,

$$\frac{u(x_j, t_{i+1}) - 2\,u(x_j, t_i) + u(x_j, t_{i-1})}{\Delta t^2} \approx$$
$$c^2 \frac{u(x_{j+1}, t_i) - 2\,u(x_j, t_i) + u(x_{j-1}, t_i)}{\Delta x^2}.$$

  Alternative notation can be used to make the parameters more obvious,

$$\frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta t^2} \approx c^2 \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}, \tag{1}$$

- *Initial condition.* Use the centered first difference approximation of the first derivative,

$$\frac{\partial}{\partial t} u(x_j, t_i) \approx \frac{u_j^{i+1} - u_j^{i-1}}{2\Delta t} \tag{2}$$

  Note division by $2\Delta t$ because the difference is between values of $u(x,t)$ separated by two time intervals.

# Initial Conditions

- Using approximation (2), initial condition $\frac{\partial}{\partial t}u(x_j, 0) = 0$ means,

$$u_j^{i-1} = u_j^{i+1}, \quad j = 0, \ldots, N_x. \quad i = 0.$$

- The intial condition of shape is simply,

$$u_j^0 = I(x_j), \quad j = 0, \ldots, N_x.$$

# Initial Conditions

- Using approximation (2), initial condition $\frac{\partial}{\partial t}u(x_j, 0) = 0$ means,

$$u_j^{i-1} = u_j^{i+1}, \quad j = 0, \ldots, N_x. \quad i = 0.$$

- The intial condition of shape is simply,

$$u_j^0 = I(x_j), \quad j = 0, \ldots, N_x.$$

# Formulae

- $u_j^{i+1} = -u_j^{i-1} + 2u_j^i + C^2 \left( u_{j+1}^i - 2u_j^i + u_{j-1}^i \right)$
- $C = c\frac{\Delta t}{\Delta x}$.
- $u_j^1 = u_j^0 - \frac{1}{2} C^2 \left( u_{j+1}^i - 2u_j^i + u_{j-1}^i \right)$

# Iterative Simulation Algorithm

1. Initialize $u_j^0 = I(x_j)$ for $j = 0, \ldots N_x$.
2. Compute $u_j^1$ and set $u_j^1 = 0$ for the boundary points $i = 0$ and $i = N_x$, for $i = 1, \ldots N - 1$
3. For each time level $i = 1, \ldots N_t - 1$
    3.1 find $u_j^{i+1}$ for $j = 1, \ldots N_x - 1$.
    3.2 set $u_j^{i+1} = 0$ for the boundary points $j = 0, j = N_x$.