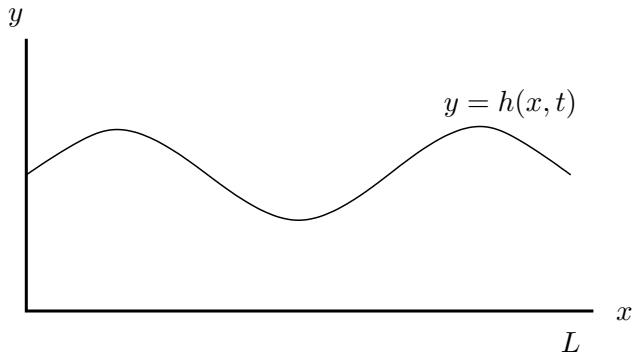
Scientific Programming
Wave Equation

1 The wave equation

The wave equation describes how waves propagate: light waves, sound waves, oscillating strings, wave in a pond, ... Suppose that the function $h(x,t)$ gives the the height of the wave at position x and time t. Then h satisfies the differential equation:

$$\frac{\partial^2 h}{\partial t^2} = c^2 \frac{\partial^2 h}{\partial x^2} \tag{1}$$

where c is the speed that the wave propagates.



Finite difference update rules Recall that the second derivative of a function can be approximated by the central difference

$$f''(x) \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$$

The same technique can be used to approximate the second derivatives:

$$\begin{aligned} \frac{\partial^2 h}{\partial t^2} &\approx \frac{h(x,t+\Delta t) - 2h(x,t) + h(x,t-\Delta t)}{\Delta t^2} \\ \frac{\partial^2 h}{\partial x^2} &\approx \frac{h(x+\Delta x,t) - 2h(x,t) + h(x-\Delta x,t)}{\Delta x^2} \end{aligned}$$

Using these approximations, an approximation for the wave equation can be found replacing the second derivatives by these approximation.

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

Note that one term is at time $t+\Delta t$, which we will refer to a future time. The rest involve time t (current time) or $t-\Delta t$ (past time). If we solve for this future time term we get:

$$h(x,t+\Delta t) = r^2(h(x+\Delta x,t) + h(x-\Delta x,t)) + 2(1-r^2)h(x,t) - h(x,t-\Delta t) \tag{2}$$

where $r = \frac{c\Delta t}{\Delta x}$. It turns out that as long as Δt is chosen small enough so that $r < 1/2$ the simulation will be stable. Otherwise, the waves will continue to grow larger and larger.

Matlab code for update The update rule involves past, current and future times. Suppose each is represented by an array of length n. Each spot in the array represents the height of the array at coordinates Δx units apart. Index n corresponds to $x = 0$ and index n to $x = L$.

```
for i=2:n-1
  future(i) = r^2*(current(i-1) + current(i+1)) + 2*(1-r^2)*current(i) - past(i)
end
```

This is just rewriting the mathematical update rule in Matlab. Notice that shifting x left or right Δx units correspond to adding or subtracting one from the index. Also, the for loop starts at 2 and ends at n−1. The reference to index i−1 would cause an error for i less than 2. Similarly, the reference to index i+1 is invalid for i greater than n−1. It is important to realize that this update rule cannot be used at the endpoints.

Using array operations the for loop above can be replaced by

```
future(2:n-1) = r^2*(current(1:n-2)+current(3:n)) + 2*(1-r^2)*current(2:n-1) - past(2:n-1)
```

The advantage of using these array operations is that Matlab does the calculation much faster.

2 Example: Plucked string

Consider a guitar string. Both ends are fixed at a height of 0. So the update rules for future(1) and future(n) can set the values to 0. This addresses the issues of the previous update rules for dealing with the boundary. Below is a complete Matlab program that simulates a plucked string.

```
1  dx = .01;          % Spacing of points on string
2  dt = .001;         % Size of time step
3
4  c = 5;             % Speed of wave propagation
5  L = 10;            % Length of string
6  stopTime = 30;     % Time to run the simulation
7
8  r = c*dt/dx;
9  n = L/dx + 1;
10
11 % Set current and past to the graph of a plucked string
12 current = .5-.5*cos(2*pi/L*[0:dx:L]);
13 past = current;
14
15 for t=0:dt:stopTime
16   % Calculate the future position of the string
17   future(1) = 0;
18   future(2:n-1) = r^2*(current(1:n-2)+current(3:n)) + 2*(1-r^2)*current(2:n-1) - past(2:n-1);
19   future(n) = 0;
20
21   % Set things up for the next time step
22   past = current;
23   current = future;
24
25   % Plot the graph after every 10th frame
26   if mod(t/dt, 10) == 0
27     plot([0:dx:L], current)
28     axis([0 L -2 2])
29     pause(.001)
30   end
31 end
```

A couple of things to point out in the Matlab code. First, in lines 22-23 the current string heights are copied to the past and that future to the current. This sets things up for the next time step. In the next time step the current heights are what were the future and the past is what was the current. Also, in lines 26-31 a plot of the current string is displayed. The if statement ensures that it is only printed out every 10 steps. This value can be changed and will affect the speed of the animation. The **pause** command inserts a small time delay to ensure the graph is displayed to the screen.

3 Example: Jump rope

Instead of fixing the ends of a rope, we can have them follow some function. For example, a two meter long jump rope with the left end going up and down using the function $\sin(5t)$ and the right end using the function $\sin(12t)$. Other than the starting conditions and how the left and right endpoints are handled, the code is essentially identical.

```
1  dx = .01;         % Spacing of points on rope
2  dt = .001;        % Size of time step
3
4  c = 1;            % Speed of wave propagation
5  L = 2;            % Length of rope
6  stopTime = 30;    % Time to run the simulation
7
8  r = c*dt/dx;
9  n = L/dx + 1;
10
11 current = zeros(1,n);
12 past = zeros(1,n);
13
14 for t=0:dt:stopTime
15   % Calculate the future position of the rope
16   future(1) = sin(2*t);
17   future(2:n-1) = r^2*(current(1:n-2)+current(3:n)) + 2*(1-r^2)*current(2:n-1) - past(2:n-1);
18   future(n) = sin(3*t);
19
20   % Set things up for the next time step
21   past = current;
22   current = future;
23
24   % Plot the graph after every 10th frame
25   if mod(t/dt, 10) == 0
26     plot([0:dx:L], current)
27     axis([0 L -8 8])
28     pause(.001)
29   end
30 end
```

4 Example: Reflected wave

In the previous two examples we specifically identified what was happening at the boundaries. This avoided the issue that equation 2 cannot be used at the boundary. We can also deal with this issue by having other types of constraints on the boundary. For example, have the wave reflect perfectly off of the boundary. It turns out a reflected wave is perpendicular to the boundary at all times. This adds the restrictions that

$$\begin{aligned} \frac{\partial h}{\partial x}\Big|_{x=0} &= 0 \\ \frac{\partial h}{\partial x}\Big|_{x=L} &= 0 \end{aligned}$$

Combining these two equations with equation 1 when $x = 0$ gives use the following:

$$\begin{aligned} \frac{\partial^2 h}{\partial t^2} &= c^2 \frac{\partial^2 h}{\partial x^2} \\ \frac{\partial h}{\partial x}\Big|_{x=0} &= 0 \end{aligned}$$

Replacing the derivatives with their central difference approximations and substituting in $x = 0$ yields the following:

$$\begin{aligned} \frac{h(0, t+\Delta t) - 2h(0,t) + h(0, t-\Delta t)}{\Delta t^2} &= c^2 \frac{h(\Delta x, t) - 2h(0,t) + h(-\Delta x, t)}{\Delta x^2} \\ \frac{h(\Delta x, t) - h(-\Delta x, t)}{2\Delta x} &= 0 \end{aligned}$$

The term $h(-\Delta x, t)$ does not make sense since the x coordinate is negative. However, we can solve for this in the second equation yielding that $h(-\Delta x, t) = h(\Delta x, t)$. Plugging this into the first and solving for $h(0, t + \Delta t)$ yields

$$h(0, t+\Delta t) = 2r^2 h(\Delta x, t) + (1 - r^2)h(0,t) - h(0, t - \Delta t)$$

This gives us an update rule to use on the left hand side when the wave is reflected. A similar argument can be used when $x = L$ to yield

$$h(L, t+\Delta t) = 2r^2 h(L - \Delta x, t) + (1 - r^2)h(L,t) - h(L, t - \Delta t)$$

where $r = \frac{c\Delta t}{\Delta x}$. Below is the implementation with reflection on both sides with a "pebble" dropped in a 10 meter wide pond.

```
1  dx = .01;          % Spacing of points on the pond
2  dt = .001;         % Size of time step
3
4  c = 1;             % Speed of wave propagation
5  L = 10;            % Width of the pond
6  stopTime = 30;     % Time to run the simulation
7
8  r = c*dt/dx;
9  n = L/dx + 1;
10
11 current = zeros(1,n);
12 % Put a depression in the middle of the pond
13 current(3/dx:4/dx) = -.5+.5*cos(linspace(0,2*pi,4/dx-3/dx+1));
14 past = current;
15
16 for t=0:dt:stopTime
17   % Calculate the future position of ponds surface
```

```
18    future(1) = 2*r^2*current(2) + 2*(1-r^2)*current(1) - past(1);
19    future(2:n-1) = r^2*(current(1:n-2)+current(3:n)) + 2*(1-r^2)*current(2:n-1) - past(2:n-1);
20    future(n) = 2*r^2*current(n-1) + 2*(1-r^2)*current(n) - past(n);
21
22    % Set things up for the next time step
23    past = current;
24    current = future;
25
26    % Plot the graph after every 10th frame
27    if mod(t/dt, 10) == 0
28      plot([0:dx:L], current)
29      axis([0 L -1 1])
30      pause(.001)
31    end
32  end
```

5 Example: Wave continuing on forever

In the previous example, we have perfect reflection of a wave. What if we want a wave to continue on forever? Obviously we cannot deal with an infinitely long wave; however, we can have a boundary condition that will not affect the wave whatsoever. The derivation of this is somewhat complicated, so we will not include it. But if we want to have the right end not affect the wave at all then we want

$$h(L, t + \Delta t) = \frac{2h(L,t) + (r-1)h(L, t-\Delta t) + 2r^2(h(L-\Delta x, t) - h(L,t))}{1+r}$$

where $r = \frac{c\Delta t}{\Delta X}$. This update rule can be used to have a "transparent" boundary.

Assuming that the left hand side has the value $sin(t)$ and right side is transparent we get the following Matlab implementation: the right hand side is transparent

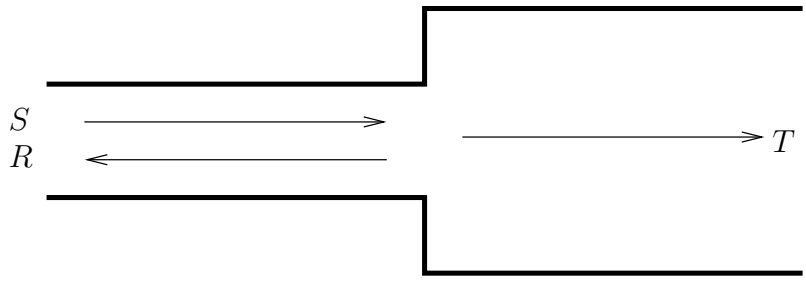
```
1   dx = .01;          % Spacing of points along the x-axis
2   dt = .001;         % Size of time step
3
4   c = 1;             % Speed of wave propagation
5   L = 10;            % Length of region considered
6   stopTime = 30;     % Time to run the simulation
7
8   r = c*dt/dx;
9   n = L/dx + 1;
10
11  current = zeros(1,n);
12  past = current;
13
14  for t=0:dt:stopTime
15    % Calculate the future position of the wave
16    future(1) = sin(t);
17    future(2:n-1) = r^2*(current(1:n-2)+current(3:n)) + 2*(1-r^2)*current(2:n-1) - past(2:n-1);
18    future(n) = (2*current(n) + (r-1)*past(n) + 2*r^2*(current(n-1)-current(n)))/(1+r);
19
20    % Set things up for the next time step
21    past = current;
22    current = future;
23
24    % Plot the graph after every 10th frame
25    if mod(t/dt, 10) == 0
```

```
26        plot([0:dx:L], current)
27        axis([0 L -2 2])
28        pause(.001)
29      end
30    end
```

Notice that when this is run, the wave moves from left to right and seems to continue past the right boundary.

6 Example: Sound wave going from a small tube into a large one

Our final example demonstrates what happens when a sound traveling down a small tube enters a larger one. As the pressure wave hits the larger tube, some gets reflected and some gets transmitted. The transmitted sound is of lower amplitude since it is spread out over a wider area. This approximation only works well for narrow tubes and low frequencies. In larger tubes you need to keep track of sound reflecting off of the sides of the tubes, which we will ignore.



We will use three separate waves: S the source (or input wave), R the wave that is reflected back and T, the transmitted wave. Assume that the smaller tube has area A_1, the larger A_2 and both tubes have length L. Each of the three waves are governed by the wave equation. And each will have a transparent end. The start of the source will be a 1 kHz signal. And the starting point of the reflected and transmitted wave can be determined by the following formulas

$$R(0,t+\Delta t) = \frac{(1-\frac{A_2}{A_1})S(L,t+\Delta t) - S(L,t-\Delta t) + R(0,t-\Delta t) + \frac{A_2}{A_1}T(0,t-\Delta t))}{1+\frac{A_2}{A_1}}$$

$$T(0,t+\Delta t) = \frac{2S(L,t+\Delta t) - S(L,t-\Delta t) + R(0,t-\Delta t) + \frac{A_2}{A_1}T(0,t-\Delta t)}{1+\frac{A_2}{A_1}}$$

This formulas are derived using acoustically laws that we will not go into.

The following Matlab script implements these equations. Note that each of the three waveforms has a past, current and future version.

```
1  dx = .01;          % Spacing of points along the tube
2  dt = .00001;       % Size of time step
3
4  c = 340;           % Speed of sound
5  L = 1;             % Length of each tube
6  stopTime = .1;     % Time to run the simulation
7
8  A1 = 5;
9  A2 = 15;
10
11 r = c*dt/dx;
```

```
12 n = L/dx + 1;
13
14 currentS = zeros(1,n);
15 pastS = zeros(1,n);
16 currentR = zeros(1,n);
17 pastR = zeros(1,n);
18 currentT = zeros(1,n);
19 pastT = zeros(1,n);
20
21 for t=0:dt:stopTime
22   % Update the interiors and end of each using the wave equation
23   futureS(2:n-1) = r^2*(currentS(1:n-2)+currentS(3:n)) + 2*(1-r^2)*currentS(2:n-1) - pastS(2:n-
24   futureS(n) = (2*currentS(n) + (r-1)*pastS(n) + 2*r^2*(currentS(n-1)-currentS(n)))/(1+r);
25
26   futureR(2:n-1) = r^2*(currentR(1:n-2)+currentR(3:n)) + 2*(1-r^2)*currentR(2:n-1) - pastR(2:n-
27   futureR(n) = (2*currentR(n) + (r-1)*pastR(n) + 2*r^2*(currentR(n-1)-currentR(n)))/(1+r);
28
29   futureT(2:n-1) = r^2*(currentT(1:n-2)+currentT(3:n)) + 2*(1-r^2)*currentT(2:n-1) - pastT(2:n-
30   futureT(n) = (2*currentT(n) + (r-1)*pastT(n) + 2*r^2*(currentT(n-1)-currentT(n)))/(1+r);
31
32   % Set the values for the start of each wave
33   futureS(1) = sin(2*pi*1000*t);
34
35   futureR(1) = ( (1-A2/A1)*futureS(n) - pastS(n) + pastR(1) + A2/A1*pastT(1) )/(1+A2/A1);
36   futureT(1) = ( 2*futureS(n) - pastS(n) + pastR(1) + A2/A1*pastT(1) )/(1+A2/A1);
37
38   % Set things up for the next time step
39   pastS = currentS;
40   currentS = futureS;
41   pastR = currentR;
42   currentR = futureR;
43   pastT = currentT;
44   currentT = futureT;
45
46   % Plot the graph after every 100th frame
47   if mod(t/dt, 100) == 0
48     plot([0:dx:L], currentS, [L:-dx:0], currentR, [L:dx:2*L], currentT)
49     axis([0 2*L -2 2])
50     legend('Source signal', 'Reflected signal', 'Transmitted signal')
51     pause(.001)
52   end
53 end
```

In this program c is the speed of sound (about 340 m/s) and the values for dt and dx are chosen small enough to represent a 1 kilohertz signal well.