Finite Difference Schemes Synthesis in Faust

The fds library

February 2021

Riccardo Russo

AALBORG UNIVERSITY
DENMARK

Agenda

Finite Difference Schemes theory

- Introduction to FDSs.
- 1D wave equation.
- Damped wave equation.

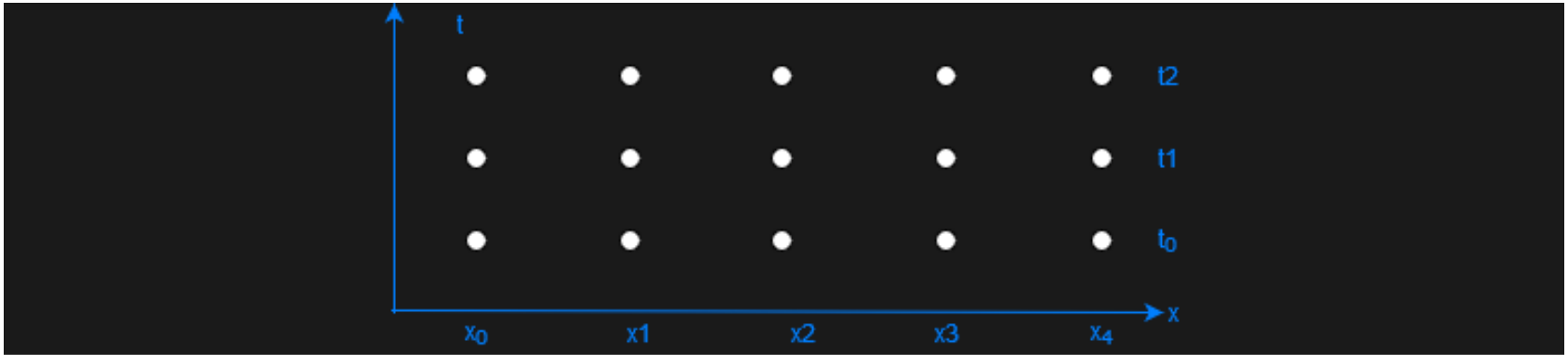
Hands on

- The first Faust FDS.
- The fds library - CA approach.
- Building physical models with the library.
- Interacting with the models.

Finite Difference Schemes
Introduction

What is a Finite Difference Scheme?

- Discrete model based on Finite Difference Time Domain methods (FDTD), largely used in engineering for systems simulation.
- It consists in numerically approximating a continuous system (PDE) with a discrete space-time grid.



Finite Difference Schemes

FDS in Physical Modeling

Characteristics

- Very accurate.
- Few assumptions on the system solution.
- Generally applicable and flexible (i.e. nonlinear systems).
- Computationally expensive

Applications: The NESS project[1]

Trombone: Guitar:

[1] University of Edinburgh, NESS: Next Generation Sound Synthesis, url: www.ness.music.ed.ac.uk/

Finite Difference Schemes
Basics

Musical acoustic systems are defined by partial differential equations (PDEs), accompanied by regions of definitions, boundary conditions and forcing terms. The state of these systems depends on the time instant and the spatial coordinates, and can be defined as $u(\mathbf{x}, t)$.

Steps for defining a FDS

- Formulation of a mathematical model.
- Definition of the space and time sampling steps.
- Discretization of the differential operators to obtain an update equation.
- Implementation of the boundary conditions.

Notation

Stefan Bilbao in his book Numerical Sound Synthesis uses the following notation for partial derivatives:

$$\frac{\partial u(x,t)}{\partial t} := u_t \qquad \text{First order derivative}$$

$$\frac{\partial^2 u(x,t)}{\partial^2 t} := u_{tt} \qquad \text{Second order derivative}$$

$$\frac{\partial^3 u(x,t)}{\partial t \partial^2 x} := u_{txx} \qquad \text{Mixed derivative}$$

The 1D Wave Equation

Mathematical Model

The simplest possible equation in acoustics is the 1D wave equation:

$$u_{tt}(x,t) = c^2 u_{xx}(x,t)$$

- Second-order PDE in time and space, with solution $u(x,t)$.
- u is the state of the system (the displacement of a string, the pressure value inside a tube at position x and time t, etc.).
- c is the speed of sound in the medium (i.e. in the material of which the system is made: air, metal, etc.).
- t is defined over the positive real domain, while x is usually bounded over a finite length L.
- The fundamental frequency is given by $f_0 = c/2L$

The Sampling Grid

Definition of the Space and Time Sampling Steps

Grid Definition

In order to simulate the mathematical model we need to start by discretizing time and space.

- Definition of the sampling steps k, h such that $t = nk$, $x = lh$ with $n = 0, 1, 2, ...$ and $l \in \mathbb{Z}$.
- We obtain the discrete u_l^n which approximates $u(x, t)$, made of $N = L/h$ data points.

Stability Condition

The sampling steps are not independent and bounded by a stability condition. For the 1D wave equation we can define $\lambda = ck/h$ and we have:

$$\lambda \leq 1 \Rightarrow h \geq ck$$

Which is the Courant-Fredrichs-Levy (CFL) condition.

The Difference Operators

Once defined the sampling steps we can go on and discretize the partial derivative operators:

$$u_t \approx \begin{cases} \delta_{t+} u_l^n = \frac{u_l^{n+1} - u_l^n}{k} & \text{Forward time operator} \\ \delta_{t-} u_l^n = \frac{u_l^n - u_l^{n-1}}{k} & \text{Backward time operator} \\ \delta_{t\cdot} u_l^n = \frac{u_l^{n+1} - u_l^{n-1}}{2k} & \text{Centred time operator} \end{cases}$$

$$u_x \approx \begin{cases} \delta_{x+} u_l^n = \frac{u_{l+1}^n - u_l^n}{h} & \text{Forward space operator} \\ \delta_{x-} u_l^n = \frac{u_l^n - u_{l-1}^n}{h} & \text{Backward space operator} \\ \delta_{x\cdot} u_l^n = \frac{u_{l+1}^n - u_{l-1}^n}{2h} & \text{Centred space operator} \end{cases}$$

$$\delta_{tt} u_l^n := \delta_{t+}\delta_{t-} u_l^n = \frac{u_l^{n+1} - 2u_l^n + u_l^{n-1}}{k^2} \quad \text{Second order time operator}$$

$$\delta_{xx} u_l^n := \delta_{x+}\delta_{x-} u_l^n = \frac{u_{l+1}^n - 2u_l^n + u_{l-1}^n}{h^2} \quad \text{Second order space operator}$$

The 1D Wave Equation
Discretization

We have:

$$u_{tt}(x,t) = c^2 u_{xx}(x,t)$$

If we apply the difference operators:

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

Expanding:

$$\frac{u_l^{n+1} - 2u_l^n + u_l^{n-1}}{k^2} = c^2 \frac{u_{l+1}^n - 2u_l^n + u_{l-1}^n}{h^2}$$

Now u_l^{n+1} is our unknown, solving we obtain:

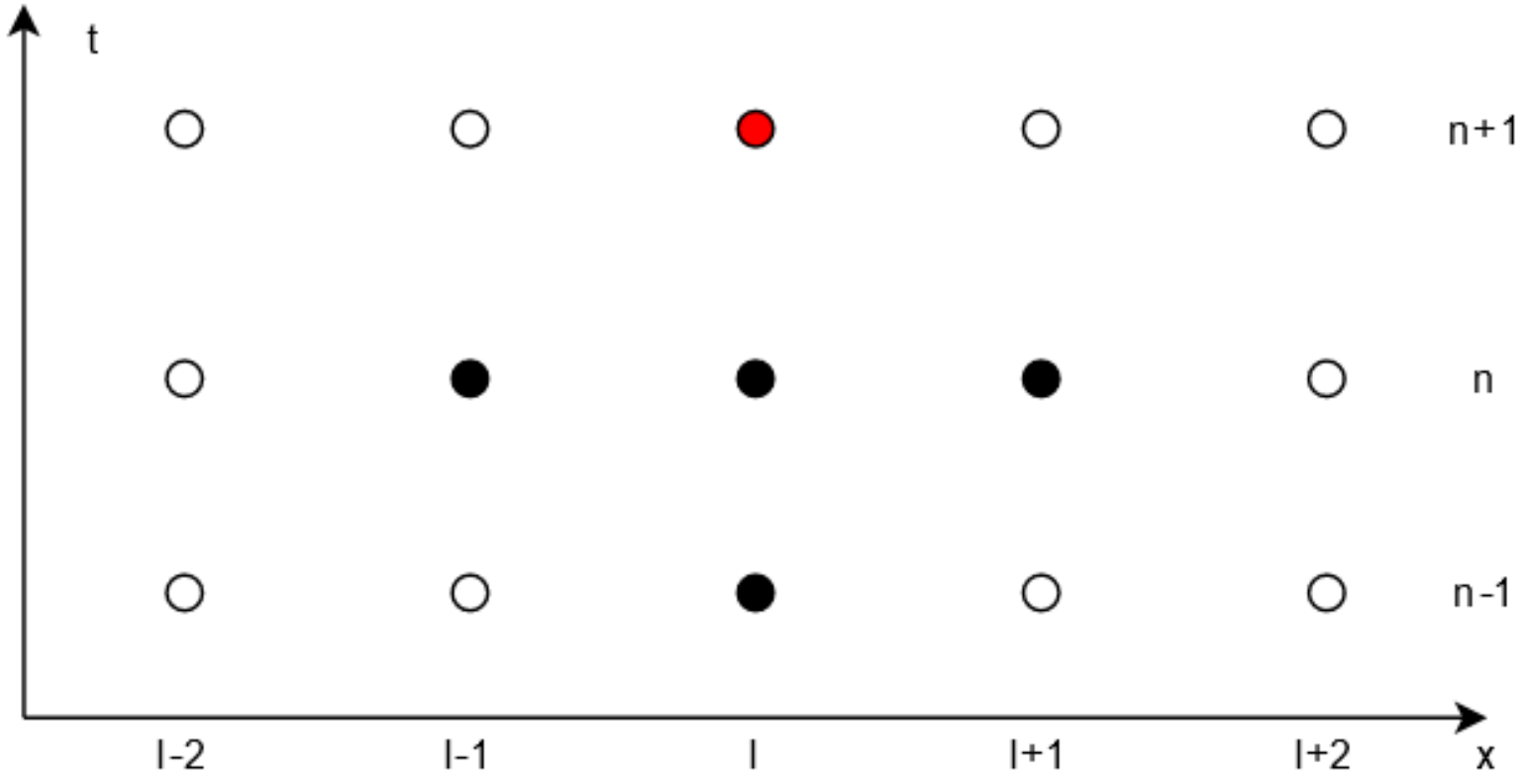
$$u_l^{n+1} = 2\left(1 - \lambda^2\right) u_l^n - u_l^{n-1} + \lambda^2 \left(u_{l+1}^n + u_{l-1}^n\right)$$

Which is an explicit recursion relation to be solved numerically.

The 1D Wave Equation
Stencil

Space and time dependency for the 1D update equation:

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



The 1D Wave Equation
Boundary Conditions

To numerically solve the recursion relation we need to provide initial and boundary conditions:

Initial Conditions

State and velocity of the system at $t = 0$ for each data point, therefore $u_l^0 = u0$ nd $\delta_x u_l^0 = v0$. For simplicity we can set both at zero, so the system is at rest.

Boundary Conditions

Conditions at the endpoints of the domain 0 and L:

$u_0^n = 0$ Dirichlet

String $\Rightarrow$ fixed end
Tube $\Rightarrow$ open end

$\delta_x u_0^n = 0$ Neumann

String $\Rightarrow$ free end
Tube $\Rightarrow$ closed end

The 1D Wave Equation

Neumann Condition

Now we want to try to implement a Neumann condition at $l = 0$. We have:

$$\delta_x . u_0^n = 0 \rightarrow u_{-1}^n = u_1^n$$

Applying this to the update equation:

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

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

Which is the update equation to be used only for the left boundary.

The Damped Wave Equation

Every instrument in the real world experiences damping, which causes the outcoming sound to fade out. In math this is described by a term which depends on the velocity.

$$u_{tt}(x,t) = c^2 u_{xx}(x,t) - 2\sigma_0 u_t(x,t)$$

σ_0 is called the damping coefficient, it's a number which depends on the physical characteristics of the object.

The Damped Wave Equation

Update Equation

This slide contains an error in the coefficients C1 and C2, which is present also in the Faust code in the video. Check the code for the damped wave equation in this folder to get the correct version.

We can discretize the damped wave equation as we did before:

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - 2\sigma_0 \delta_{x\cdot} u_l^n$$

By expanding the operators we obtain:

$$u_l^{n+1} = \frac{2(1-\lambda^2)}{C1} u_l^n + \frac{C2}{C1} u_l^{n-1} + \frac{\lambda^2}{C1}(u_{l+1}^n + u_{l-1}^n)$$

With:

C1 = sigma0*k + 1

C2 = sigma0*k-1

$$C1 = \frac{2\sigma_0 k^2}{h} + 1 \qquad C2 = \frac{2\sigma_0 k^2}{h} - 1$$

This update equation has the same space-time dependency and stability condition of the one we saw before.

The 2D Wave Equation

The wave equation in 2D is:

$$u_{tt}(x, y, t) = c^2(u_{xx}(x, y, t) + u_{yy}(x, y, t))$$

First we need to introduce another spatial index m, which is the result of the discretization of the second space dimension y. This comes with the definition of another sampling step, therefore we have:

$$x = lh_x \qquad\qquad y = mh_y$$

For simplicity we can assume that the system is isotropic, i.e. it has the same physical characteristics in every spatial direction. In this case we can set:

$$h_x = h_y := h$$

And the stability condition is given by:

$$\lambda \leq \frac{1}{\sqrt{2}} \Rightarrow h \geq ck\sqrt{2}$$

The 2D Wave Equation

Update Equation

A discretized version of the 2D wave equation can be written as:

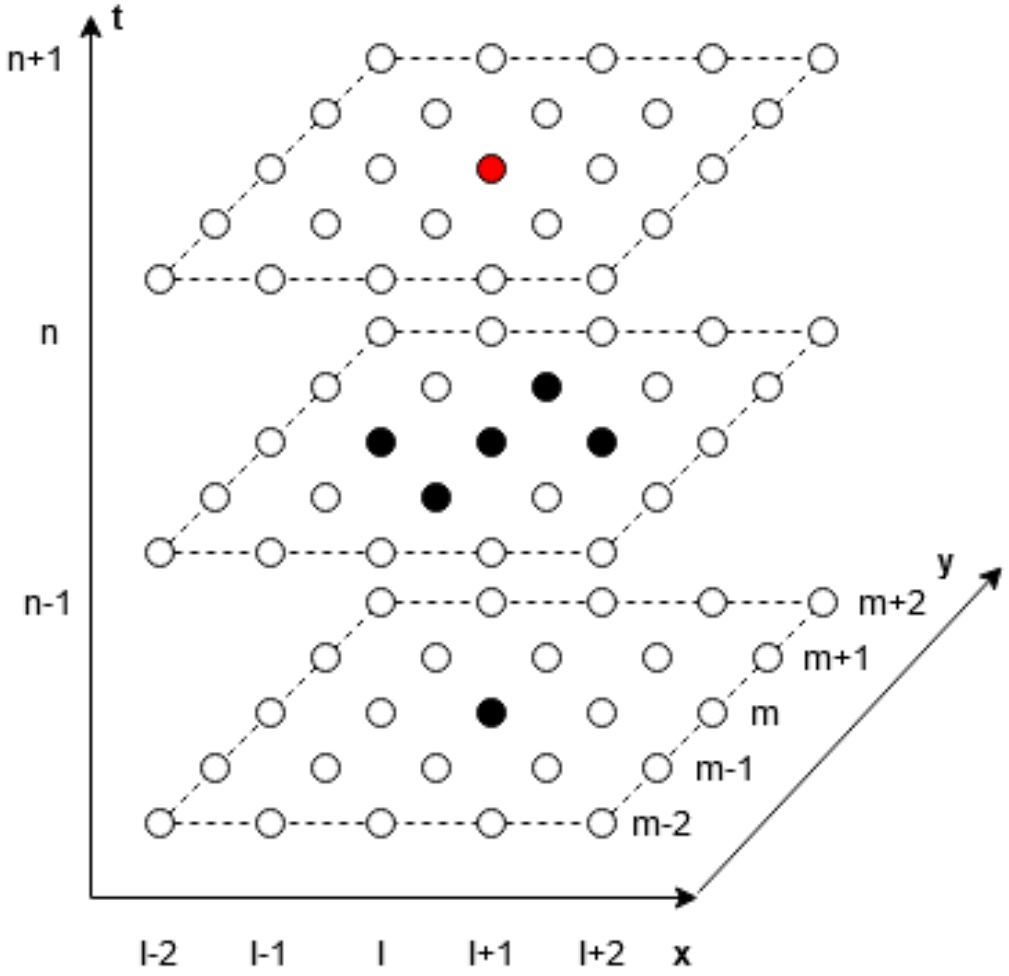
$$\delta_{tt} u^n_{l,m} = c^2(\delta_{xx} u^n_{l,m} + \delta_{yy} u^n_{l,m})$$

By expanding the operators and solving for the unknown we obtain the explicit recursion relation:

$$u^{n+1}_{l,m} = 2(1-\lambda^2)u^n_{l,m} - u^{n-1}_{l,m} + \lambda^2(u^n_{l+1,m} + u^n_{l-1,m} + u^n_{l,m+1} + u^n_{l,m-1})$$

The 2D Wave Equation
Stencil

Space-time dependency for the 2-D wave equation:



Implementation

How do we actually implement a FDS algorithm?

- In imperative programming languages we can use arrays:
  - We allocate 3 arrays for the states $n-1$, n, $n+1$, each one with a number of data points equal to the total number of spatial grid points given by $N = L/h$.
  - Then we cycle between them updating the states at audio rate.
  - We can obtain sound by reading the current state of a single data point in one of the arrays.

```
for n=1:dur
    for l=2:N-1
        uNext(l) = 2*(1-lambda^2)*u(l) - uPrev(l) + (lambda^2)*(u(l+1) + u(l-1));
    end
    out(n)= uNext(outPos);
    uPrev = u;
    u = uNext;
end
```

- In Faust we don't have arrays, we need to find another way.

Thank you for your attention!

rrusso19@student.aau.dk

AALBORG UNIVERSITY
DENMARK