MUS420 Lecture
Discrete-Time Lumped Models

Stefan Bilbao and Julius O. Smith III (jos@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

February 5, 2019

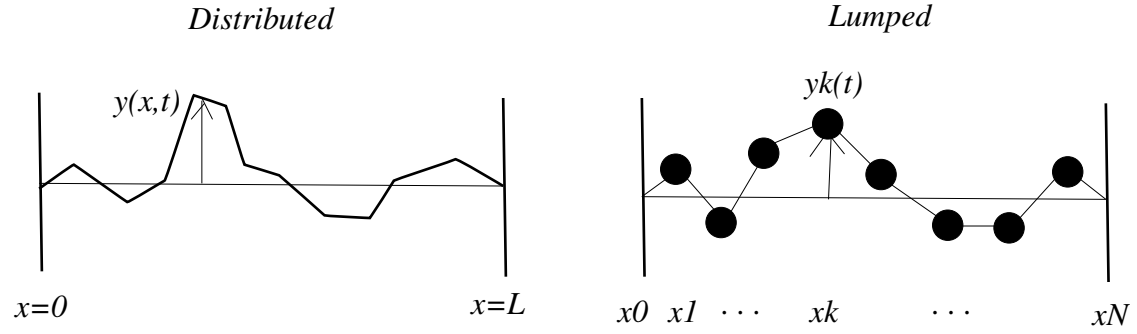
Outline:

- Lumped vs. distributed systems
- Digitizing Differential Equations
- Accuracy
- Stability (Von Neumann Analysis)
- Examples

Lumped vs. Distributed Systems

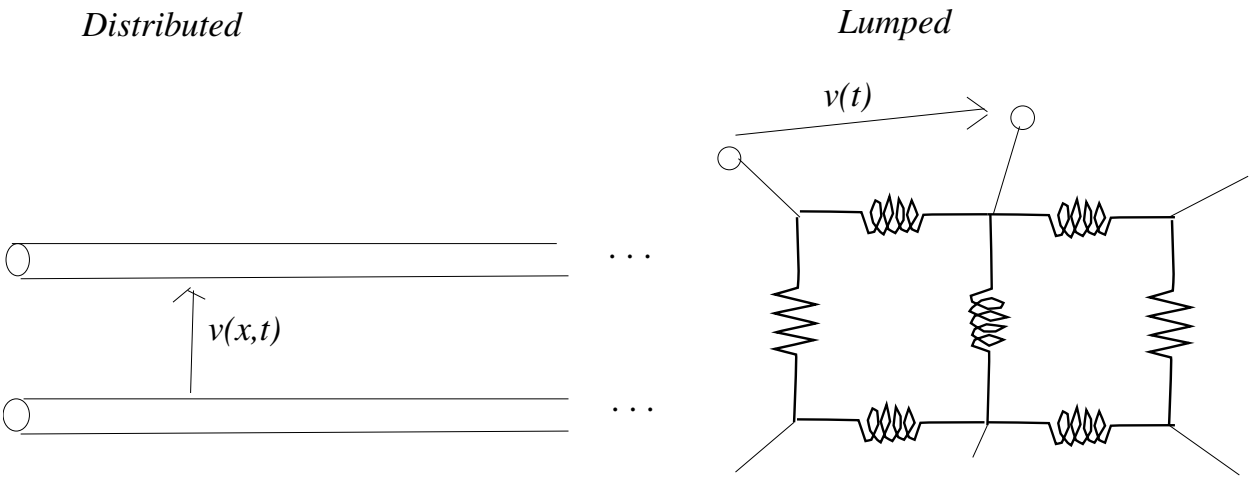
- A lumped system is one in which the dependent variables of interest are a function of time alone. In general, this will mean solving a set of ordinary differential equations (ODEs)
- A *distributed* system is one in which all dependent variables are functions of time *and* one or more spatial variables. In this case, we will be solving partial differential equations (PDEs)

For example, consider the following two systems:



- The first system is a *distributed* system, consisting of an infinitely thin string, supported at both ends; the dependent variable, the vertical position of the string $y(x,t)$ is indexed continuously in both space and time.

- The second system, a series of "beads" connected by massless string segments, constrained to move vertically, can be thought of as a *lumped* system, perhaps an approximation to the continuous string.
- For electrical systems, consider the difference between a lumped RLC network and a transmission line



- The importance of *lumped approximations* to distributed systems will become obvious later, especially for waveguide-based physical modeling, because it enables one to cut computational costs by solving ODEs at a few points, rather than a full PDE (generally much more costly)

Discretization

Problem: Given

- Integro-differential equations (DEs)
- Boundary conditions

Find:

1. Numerical solution for system motion (classical problem), or
2. Real-time *computational model*:
   - Solves DEs with a computational structure
   - Input and control signals effectively "change the boundary conditions"

This course is concerned primarily with the second case, although the first case also arises when verifying acoustic theory or a particular computational model.

Classical Solutions

- Numerical Integration via Finite Differences
- Finite Element Method (FEM)
- Boundary Element Method (BEM)
- Ray Tracing
- Many others, depending on problem
- Literature vast

Computational Physical Models Suitable as Real-Time Synthesis Models

- Digital filters from Finite Differences
- Wave Digital Filters (WDF)
- Digital Waveguide Mesh (DWM) for

All are special cases of *finite difference schemes*.
All are *digital filters* of one type or another.

Historical examples:

- Kelly-Lochbaum vocal-tract model
- Linear Prediction voice models
- Ladder and Lattice digital filter structures
- Formant filter-bank speech model
  ("modal representation")
- Digital waveguide models
  (winds, strings, voice, membranes, ...)

Signal Processing Approach

- Every linear differential equation (with constant coefficients) gives rise to a *linear, time-invariant (LTI)* "network" or "medium"
- LTI media are characterized by their *frequency response* $H(e^{j\omega})$ which breaks down into
  - Attenuation versus frequency:
    $$G(\omega) = \left|H(e^{j\omega})\right|$$
  - Time-delay versus frequency:
    $$P(\omega) = -\angle H(e^{j\omega})/\omega$$

  for each input-output pair
- **Typical Procedure:** Design an *optimal digital filter* to give desired frequency response
- In multi-input, multi-output cases, a *matrix frequency response* applies
- Often the filter structure can be chosen to maintain a *physical interpretation* of all state variables

Finite Differences: Some basics

Suppose we begin with the simplest possible differential equation, that relating current and voltage in an inductor:

$$v(t) = L\frac{di}{dt}$$

We assume that one of the quantities, say i is provided by some source. We need to discretize this continuous time equation. First sample i at regular intervals nT (in the simplest case), to obtain a sequence i_n, and assume that we will be obtaining a voltage sequence v_n from it. There are many ways of approximating the differentiation operator;

- set $\frac{di}{dt} \simeq (i_n - i_{n-1})/T$. This yields the equation:

  $$v_n = (L/T)(i_n - i_{n-1})$$

  This is probably the simplest way of discretizing a derivative, and is called a *backwards difference*.

- a slightly more sophisticated discretization involves rewriting the equation for the inductor in the following way:

  $$i(t) = \frac{1}{L}\left(\int_0^t v(t')dt'\right) + i_0$$

and thus

$$\begin{aligned} i(nT) &= \frac{1}{L}\left(\int_0^{nT} v(t')dt'\right) + i_0 \\ &= \frac{1}{L}\left(\int_0^{(n-1)T} v(t')dt'\right) + i_0 + \frac{1}{L}\int_{(n-1)T}^{nT} v(t')dt' \\ &= i\left((n-1)T\right) + \frac{1}{L}\int_{(n-1)T}^{nT} v(t')dt' \\ &\simeq i\left((n-1)T\right) + \frac{T}{2L}\left(v\left((n-1)T\right) + v(nT)\right) \end{aligned}$$

yielding the scheme:

$$v_n = -v_{n-1} + \frac{2L}{T}(i_n - i_{n-1})$$

- This is called the *trapezoid rule* of numerical integration (or differentiation), and it can be seen as the basis for the WDF approach to filtering and numerical integration.
- There are many other ways of performing this discretization; in general we can imagine writing a general scheme:

$$\sum_{k=0}^{\infty} a_k v_{n-k} = \sum_{k=0}^{\infty} b_k i_{n-k}$$

which in some sense behaves like the original continuous time equation.

Accuracy

Suppose we take the backward-difference approximation $v_n = (L/T)(i_n - i_{n-1})$, and expand i_{n-1} in Taylor series about i_n. This yields:

$$\begin{aligned} v_n &= (L/T)\left(i_n - \left(i_n - T\left.\frac{di}{dt}\right|_{nT} + O(T^2)\right)\right) \\ &= \left. L\frac{di}{dt}\right|_{nT} + O(T) \end{aligned}$$

So the difference scheme approximates the continuous time equation to an accuracy that depends on T, the step size. Thus we expect that the discretization will do a better job as T gets small.

Performing the same analysis for the trapezoid rule yields:

$$v_n = \left. L\,\frac{di}{dt}\right|_{nT} + O(T^2)$$

So we say that the trapezoid rule is *second-order* accurate in T.

- In general, the *more accurate* a difference scheme, the more information from neighboring grid points it will require.

Frequency Domain Interpretation

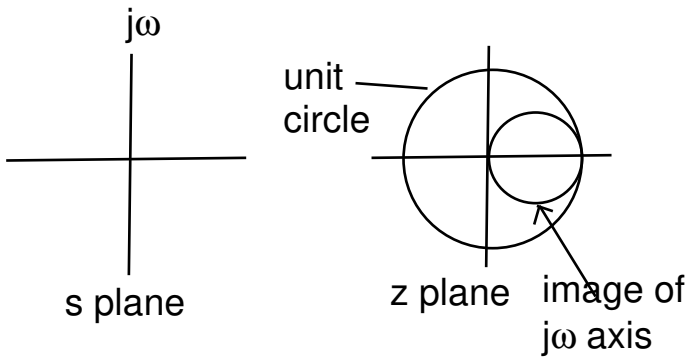
The equation for the inductor, assuming zero initial conditions, transforms to

$$V(s) = LsI(s)$$

where s is the complex frequency variable. Taking z transforms of the sequences v and i in the backward-difference scheme yields:

$$V(z^{-1}) = L\frac{1 - z^{-1}}{T}I(z^{-1})$$

Thus we can think of our discretized scheme as one obtained under the mapping $s \rightarrow \frac{1-z^{-1}}{T}$. So here we are mapping from the s plane to the z plane. The following figure illustrates where real continuous time frequencies (the $j\omega$ axis) are mapped:



- dc ($s = 0$) mapped to dc ($z = 1$)

- infinite frequency mapped to ($z = 0$)

We can write the scheme for the trapezoid rule as follows:

$$V(z^{-1}) = \frac{2L}{T}\frac{1 - z^{-1}}{1 + z^{-1}} I(z^{-1})$$

- the mapping $s \to \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$ maps all real frequencies to real frequencies uniquely. In partcular dc$\to$dc, and infinite frequency maps to $z = -1$.

  One way of examining the frequency mapping more closely is by looking at $\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$ on the unit circle, i.e., where $z = e^{j\omega T}$. This yields:

$$\frac{2}{T}\frac{1 - e^{-j\omega T}}{1 + e^{-j\omega T}} = \frac{2j}{T}\tan(\omega T/2)$$

- notice in particular the behaviour of the mapping near dc ($\omega = 0$):

$$\frac{2j}{T}\tan(\omega T/2) \simeq j\omega + O(T^3)$$

  where, since $\tan(\theta)$ is odd, there are no even-order terms in its series expansion. Thus, the trapezoid rule is a second-order accurate approximation to a derivative, in the limit of small T (i.e., near dc).

More General Differential Equations

A more general linear constant coefficient differential equation can be written as:

$$\sum_{k=0}^{N} a_k \frac{d^k v}{dt^k} = \sum_{k=0}^{M} b_k \frac{d^k i}{dt^k}$$

or, in the frequency domain, assuming zero initial conditions,

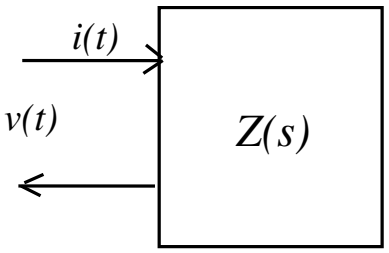
$$\sum_{k=0}^{N} a_k s^k V(s) = \sum_{k=0}^{M} b_k s^k I(s)$$

We can define a transfer-function relationship as follows:

$$Z(s) \stackrel{\Delta}{=} \frac{V(s)}{I(s)} = \frac{b_0 + b_1 s + b_2 s^2 + \cdots + b_M s^M}{1 + a_1 s + a_2 s^2 + \cdots + a_N s^N}$$

where we have normalized $a_0 \neq 0$ to 1. Note that $Z(s)$ is a rational function of s of order $\max(N, M)$.

If $i(t)$ and $v(t)$ are measured at the *same point*, then $Z(s)$ is a *driving point impedance*, as depicted below:



If the circuit (or mechanical system) is *physically passive*, then $Z(s)$ must be *positive real*

Distributed Example: 1-D wave equation, solution by FDA approach

Suppose we want to simulate one direction in an *acoustic space* in which the air is described by the second-order wave equation

$$\boxed{\frac{\partial^2 u}{\partial^2 t} = c^2 \frac{\partial^2 u}{\partial^2 x}}$$

where $u(x,t)$ is particle velocity of the air relative to equilibrium.

- This is the familiar 1-D wave equation, with wave speed given by

  $$c = \sqrt{\frac{\gamma P_0}{\rho_0}}$$

  where
  - $\gamma = 1.4$ for air ("adiabatic gas constant"),
  - P_0 is ambient pressure, and
  - ρ_0 is mass density.
- The same equation holds also for pressure $p(x,t)$ and density $\rho(x,t)$, all with the same wave speed c.

Let's "digitize" this wave equation to create a *finite difference scheme* (FDS).

Second-Order Finite Difference Scheme

The simplest, and traditional way of discretizing the 1-D wave equation is by replacing the second derivatives by second order differences:

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_{x=k\Delta,t=nT} \simeq \frac{u_k^{n-1} - 2u_k^n + u_k^{n+1}}{T^2}$$
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=k\Delta,t=nT} \simeq \frac{u_{k-1}^n - 2u_k^n + u_{k+1}^n}{\Delta^2}$$

where u_k^n is defined as $u(k\Delta, nT)$. Here we have sampled the time-space plane in a uniform grid, with a timestep of T and a space step of Δ. The u_k^n are the grid variables here. Now, through substitution, the wave equation becomes:

$$u_k^{n-1} - 2u_k^n + u_k^{n+1} = \frac{c^2T^2}{\Delta^2}(u_{k-1}^n - 2u_k^n + u_{k+1}^n)$$

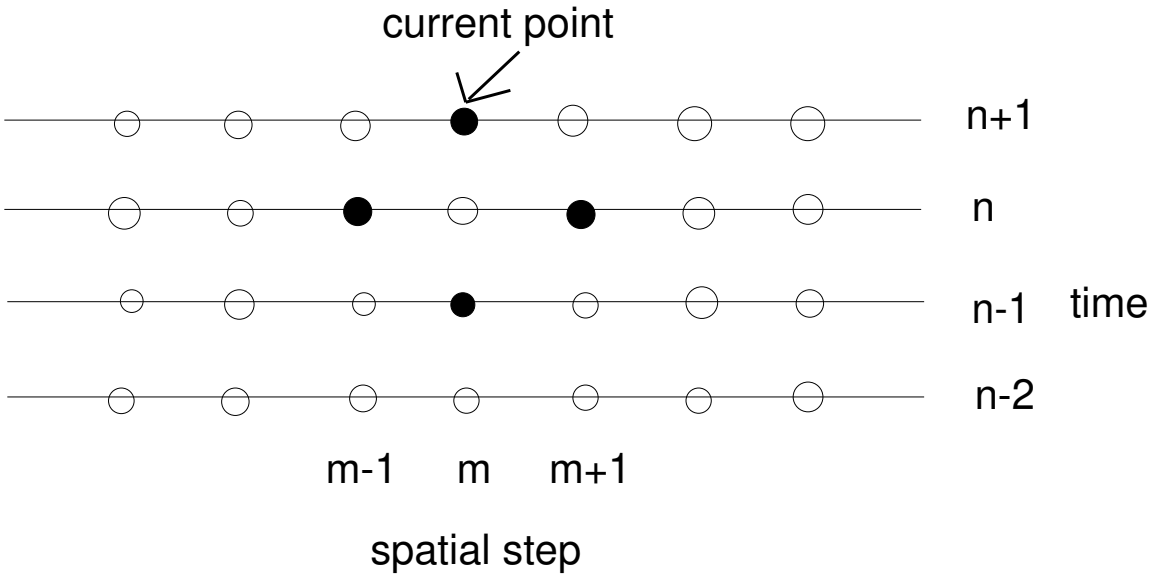
- Note that if we choose $T/\Delta = 1/c$, the equation reduces further to:

$$u_k^{n+1} = u_{k-1}^n + u_{k+1}^n - u_k^{n-1}$$

Let's examine this recursion on the time/space grid, assuming for the moment no boundary conditions:

Time-Space Grid of Second-Order FDS

$$u_k^{n+1} = u_{k-1}^n + u_{k+1}^n - u_k^{n-1}$$



- Grid variable at "current" point depends on value at *two* previous time steps (a *second order* scheme in time). We thus need to specify initial data for all m at times $n = 0$ and $n = 1$.
- Grid variable at "current" point depends on values at adjacent locations on the string (at previous time).
- Difference scheme is *explicit* (thus *parallelizable*); that is, each grid variable at time $n + 1$ depends only on grid variables at previous time instants. This is a very desirable property.

A Peek at Stability of Finite Difference Schemes

Let's look again at the difference scheme we derived for the 1-D wave eq, with the special time/space step $c = T/\Delta$:

$$u_k^{n+1} = u_{k-1}^n + u_{k+1}^n - u_k^{n-1}$$

The velocity sample $u(k, n)$ is a two-dimensional sequence with a time index and a spatial coordinate index.

Suppose we now take the DTFT with respect to the *spatial* index k:

$$\sum_{k=-\infty}^{\infty} u_k^{n+1} e^{-j\omega k\Delta} = \sum_{k=-\infty}^{\infty} (u_{k-1}^n + u_{k+1}^n - u_k^{n-1}) e^{-j\omega k\Delta}$$

or

$$U^{n+1}(\omega) = (e^{-j\omega\Delta} + e^{j\omega\Delta}) U^n(\omega) - U^{n-1}(\omega)$$

where here $U^n(\omega)$ is the *spatial spectrum* of the solution at time n, and ω is the *spatial frequency* variable. We can also write this in vector form as:

$$\begin{bmatrix} U^{n+1}(\omega) \\ U^n(\omega) \end{bmatrix} = \begin{bmatrix} 2\cos\omega\Delta & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} U^n(\omega) \\ U^{n-1}(\omega) \end{bmatrix}$$

Note that the state of the system is completely determined by $U^n(\cdot)$ and $U^{n-1}(\cdot)$.

Von Neumann Analysis

The matrix

$$A \stackrel{\Delta}{=} \begin{bmatrix} 2\cos\omega\Delta & -1 \\ 1 & 0 \end{bmatrix}$$

can be called the *state transition matrix* corresponding to the *state-space description* determined by the choice of *state vector*

$$x(n) \stackrel{\Delta}{=} \begin{bmatrix} U^n(\omega) \\ U^{n-1}(\omega) \end{bmatrix}$$

and the state update can be written more simply in vector form as $x(n+1) = Ax(n)$. Note that the state-space description is indexed by frequency ω, regarded as fixed.

- From linear systems theory, we know that such a system will be *asymptotically* stable if the eigenvalues λ of the matrix A are both less than 1 in magnitude.
- It is easy to show that the eigenvalues of A are $\lambda_+ = e^{j\omega\Delta}$ and $\lambda_- = e^{-j\omega\Delta}$. Thus, $|\lambda_\pm(\omega)| = 1, \forall\omega$.
- While we are not guaranteed asymptotic stability, $|\lambda(\omega)| = 1$ does imply that, in some sense, our solution is *not* getting *larger* with time at any spatial frequency. This can be defined as *marginal stability*.

- Note that we should *expect* the eigenvalues to have unit modulus, because the wave equation we started with corresponds to a *lossless* medium (an ideal gas). The original PDEs were derived without any loss mechanisms.
- A lossless discrete-time simulation can be highly desirable, particularly as a modeling starting point.
- This kind of "Von Neumann analysis" can be applied to any constant-coefficient FDS which is linear in its spatial directions.

Problems with FDS

- **Convergence:** Since the approximations to the second derivatives we used were second order accurate (in T and Δ), the scheme as a whole is accurate as $O(T^2, \Delta^2)$.
- Making an FDS more *accurate* (i.e., converge faster) generally requires a recursion involving more grid variables.
- An FDS for a higher order PDE also generally involves more grid variables.
- From a signal processing point of view, a more accurate simulation of an LTI medium is obtained by increasing the *order* of the filter.
- Note that an optimal filter design yields FDS coefficients which may be translated back to differential equation coefficients (which may or may not have physical meaning).
- Stability becomes more difficult to ensure in general (need to check eigenvalue magnitudes). The addition of boundary conditions makes this even more difficult.

- A good finite difference scheme may not be *explicit*, and hence may require matrix inversions (generally sparse).

For example, the dependence diagram below represents an *implicit* scheme: We cannot calculate the grid variables at the current timestep as weighted sums of grid variables at previous instants.