MUS420 Lecture
Introduction to Physical Signal Models

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

June 27, 2020

Outline

- Signal Models
- Physical Signal Models
  - Formulations
  - Simple Examples
  - Preview of Topics

Review of Physical Model Formulations

Below are names of various kinds of physical model representations we have considered:

- Ordinary Differential Equations (ODE)
- Partial Differential Equations (PDE)
- Difference Equations (DE)
- Finite Difference Schemes (FDS)
- (Physical) State Space Models
- Transfer Functions (between physical signals)
- Modal Representations (Parallel Second-Order Filters)
- Equivalent Circuits
- Impedance Networks
- Wave Digital Filters (WDF)
- Digital Waveguide (DW) Networks

Formulation Summaries

- *ODEs and PDEs* are purely mathematical descriptions (being differential equations), but they can be readily "digitized" to obtain computational physical models
- *Difference equations* are simply digitized differential equations — digitizing ODEs and PDEs produce DEs
- A DE may also be called a *finite difference scheme*
- A discrete-time *state-space* model is a special formulation of a DE in which a vector of *state variables* is defined and propagated in systematically
- In the linear time-invariant (LTI) case, a discrete-time state-space model is a *vector first-order finite-difference scheme*
- LTI difference equations can be reduced to a collection of *transfer functions*, one for each pairing of input and output signals
- Alternatively, a single *transfer function matrix* can relate a vector of input-signal z transforms to a vector of output signal z transforms
- An LTI state-space model can be *diagonalized* to produce a so-called *modal representation*, yielding a

computational model consisting of a parallel bank of second-order digital filters

- *Impedance networks* and their associated *equivalent circuits* are at the foundations of electrical engineering, and analog circuits have been used extensively to model linear systems, etc.
- Impedance networks are also useful intermediate representations for computational physical models
- *Wave Digital Filters* (WDF) were developed for digitizing analog circuits element by element, preserving the "topology" of the original analog circuit (very useful when parameters are time varying)
- *Digital waveguide networks* can be viewed as highly efficient computational forms for propagating solutions to PDEs allowing wave propagation
- They can also be used to "compress" the computation associated with a sum of quasi harmonically tuned second-order resonators

ODEs

Ordinary Differential Equations (ODEs) typically result from Newton's laws of motion:

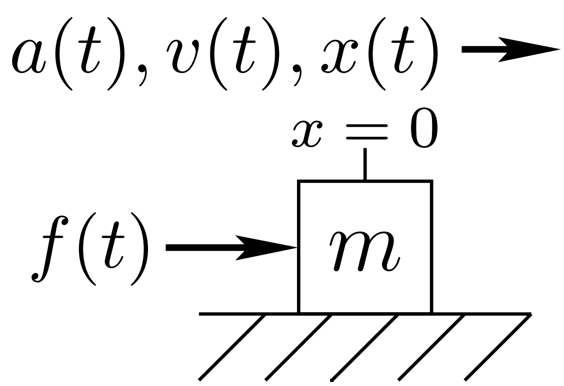
$$f(t) \; = \; m\,\ddot{x}(t) \quad \text{(Force = Mass times Acceleration)}$$

where

$$a(t) \;\stackrel{\Delta}{=}\; \ddot{x}(t) \;\stackrel{\Delta}{=}\; \frac{d^2x(t)}{dt^2}$$

Second-order ODE relating force $f(t)$ on mass m at time t to second time-derivative $\ddot{x}(t)$ of position $x(t)$

Physical Diagram:



Force $f(t)$ driving mass m along frictionless surface

Mass-Spring ODE

An *ideal spring* described by Hooke's law

$$f(t) \;=\; k\,x(t)$$

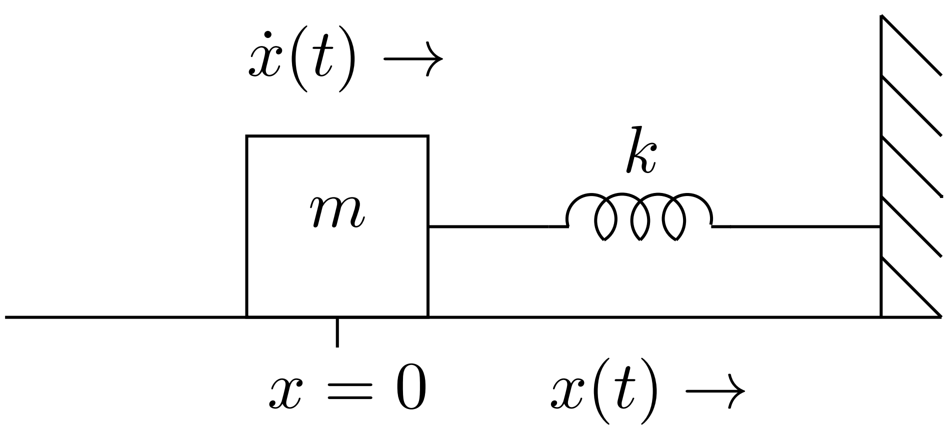
where k denotes the *spring constant*, $x(t)$ denotes the spring displacement from rest at time t, and $f(t)$ is the force required for displacement $x(t)$

If the force on a mass is due to a spring then, as discussed later, we may write the ODE as

$$k\,x(t) + m\,\ddot{x}(t) \;=\; 0$$

(Spring Force + Mass Inertial Force = 0)

Physical diagram:



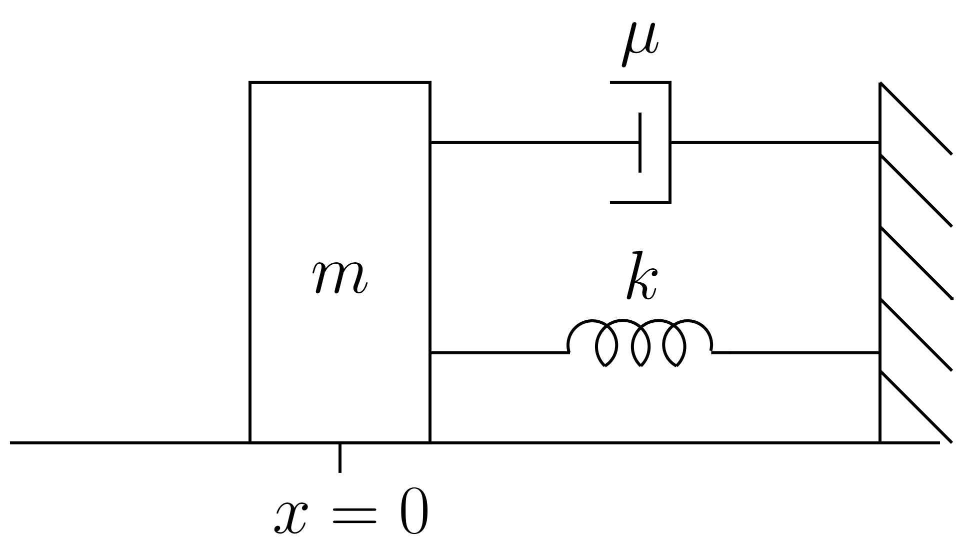
Mass-Spring-Dashpot ODE

If the mass is sliding with *friction*, then a simple ODE model is given by

$$k\,x(t) + \mu\,\dot{x}(t) + m\,\ddot{x}(t) \;=\; 0$$

(Spring + Friction + Inertial Forces = 0)

Physical diagram:



We will use such ODEs to model mass, spring, and dashpot elements, and their equivalent circuits

PDEs

- A *partial* differential equation (PDE) extends ODEs by adding one or more independent variables (usually spatial variables)

- For example, the wave equation for the ideal vibrating string adds one spatial dimension x (along the axis of the string) and may be written as

  $$K\,y''(x,t) = \epsilon\,\ddot{y}(t)$$

  (Restoring Force = Inertial Force)

  where $y(x,t)$ denotes the *transverse* displacement of the string at position x along the string and time t, and y' is the *string slope:*

  $$y'(x,t) \;\stackrel{\Delta}{=}\; \frac{\partial}{\partial x} y(x,t)$$

  (*partial derivative* of y with respect to x)

- The physical parameters in this case are string tension K and string mass-density ϵ

- As we'll see, this PDE is the starting point for highly practical *digital waveguide models* and *finite difference schemes*

Difference Equations (Finite Difference Schemes)

- There are many methods for converting ODEs and PDEs to difference equations
- For example, we'll use a very simple, order-preserving method which *replaces each derivative with a finite difference:*

$$\dot{x}(t) \triangleq \frac{d}{dt}x(t) \triangleq \lim_{\delta \to 0} \frac{x(t) - x(t-\delta)}{\delta}$$
$$\approx \frac{x(nT) - x[(n-1)T]}{T}$$

  for sufficiently small T (the sampling interval)

- This is formally known as the *backward difference* for approximating differentiation
- We'll look a few others as well

Difference Equation for a Force-Driven Mass

- Newton's $f = ma$ can be written

$$f(t) = m\,\dot{v}(t)$$

- The backward-difference substitution gives

$$f(nT) \;=\; m\frac{v(nT) - v[(n-1)T]}{T}, \quad n = 0, 1, 2, \ldots$$

- Solving for $v(nT)$ yields a *difference equation* (finite difference scheme):

$$v(nT) \;=\; v[(n-1)T] + \frac{T}{m}f(nT), \quad n = 0, 1, 2, \ldots$$

  with $v(-T) \stackrel{\Delta}{=} 0$, or, in a lighter notation:

$$v_n \;=\; v_{n-1} + \frac{T}{m}f_n$$

  with $v_{-1} \stackrel{\Delta}{=} 0$

- Note that a *delay-free loop* appears if $f(nT)$ depends on $v(nT)$ (*e.g.*, due to friction)
- In such a case, the difference equation is not *computable* in this form
- We can address this by using a *forward-difference* in place of a backward difference

Replacing Backward-Difference by Forward-Difference

- Alternate finite difference scheme:

$$\dot{x}(t) \; = \; \lim_{\delta \to 0} \frac{x(t+\delta) - x(t)}{\delta} \; \approx \; \frac{x[(n+1)T] - x(nT)}{T}$$

- As $T \to 0$, the forward and backward difference operators approach the same limit, since $x(t)$ is assumed continuous

- The forward difference gives an *explicit finite difference scheme* even if the driving force depends on current velocity:

$$v_{n+1} \; = \; v_n + \frac{T}{m} f_n, \quad n = 0, 1, 2, \ldots$$

  with $v_0 \stackrel{\Delta}{=} 0$.

Explicit and Implicit Finite Difference Schemes

Explicit:

$$y_{n+1} = x_n + \frac{1}{2} y_n$$

Implicit:

$$y_{n+1} = x_n + \frac{1}{2} y_{n+1}$$

- A finite difference scheme is said to be *explicit* when it can be computed forward in time using quantities from previous time steps
- We will associate explicit finite difference schemes with *causal digital filters*
- In *implicit* finite-difference schemes, the output of the time-update (y_{n+1} above) depends on itself, so a causal recursive computation is not specified
- Implicit schemes are generally solved using
  - iterative methods (such as Newton's method) in nonlinear cases, and
  - matrix-inverse methods for linear problems
- Implicit schemes are typically used *offline* (not in real time)

Semi-Implicit Finite Difference Schemes

- *Implicit* schemes can often be converted to *explicit* schemes (*e.g.*, for real-time usage) by limiting the number of iterations used to solve the implicit scheme
- These are called *semi-implicit finite-difference schemes*
- Iterative convergence is generally improved by working at a very high sampling rate, and by initializing each iteration to the solution for the previous sample
- See the 2009 CCRMA/EE thesis by David Yeh[1] for semi-implicit schemes for real-time computational modeling of nonlinear analog guitar effects (such as overdrive distortion)

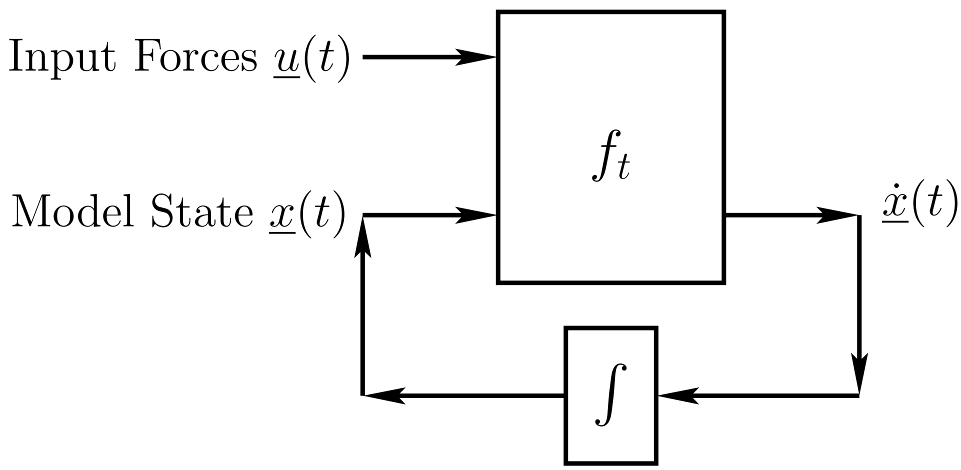
[1] `http://ccrma.stanford.edu/~dtyeh`

Explicit Finite Difference Schemes as Digital Filters

- In this course, we will be concerned almost exclusively with *explicit* finite-difference schemes, *i.e.*, *digital filter models* of one sort or another
- In other words, we concentrate mainly on the "physical modeling power" of ordinary digital filters and delay lines, together with memoryless nonlinearities (table look-ups and/or low-order polynomials)

State Space Models

Equations of motion for any physical system may be conveniently formulated in terms of its *state* $\underline{x}(t)$:



$$\dot{\underline{x}}(t) = f_t[\underline{x}(t), \underline{u}(t)]$$

where

$\underline{x}(t)$ = *state* of the system at time t
$\underline{u}(t)$ = vector of *external inputs* (typically driving forces)
f_t = general function mapping the current state $\underline{x}(t)$ and inputs $\underline{u}(t)$ to the state time-derivative $\dot{\underline{x}}(t)$

- The function f_t may be time-varying, in general
- This potentially nonlinear time-varying model is extremely general (but causal)
- Even the *human brain* can be modeled in this form

State-Space History

1. Classic *phase-space* in physics (Gibbs 1901)
   System state = point in *position-momentum space*
2. Digital computer (1950s)
3. Finite State Machines (Mealy and Moore, 1960s)
4. Finite Automata
5. State-Space Models of Linear Systems
6. Reference:
   Linear system theory: The state space approach
   L.A. Zadeh and C.A. Desoer
   Krieger, 1979

Key Property of State Vector

The key property of the state vector $\underline{x}(t)$ in the state space formulation is that it *completely determines the system at time* t

- Future states depend only on the current state $\underline{x}(t)$ and on any inputs $\underline{u}(t)$ at time t and beyond
- All past states and the entire input history are "summarized" by the current state $\underline{x}(t)$
- State $\underline{x}(t)$ includes all "memory" of the system

Force-Driven Mass Example

Consider $f = ma$ for the force-driven mass:

- Since the mass m is constant, we can use *momentum* $p(t) = m\,v(t)$ in place of velocity (more fundamental, since momentum is *conserved*)
- $x(t_0)$ and $p(t_0)$ (or $v(t_0)$) define the *state* of the mass m at time t_0
- In the absence of external forces $f(t)$, all future states are *predictable* from the state at time t_0:

$$\begin{aligned} p(t) &= p(t_0) \quad \text{(conservation of momentum)} \\ x(t) &= x(t_0) + \frac{1}{m}\int_{t_0}^{t} p(\tau)\,d\tau, \quad t \geq t_0 \end{aligned}$$

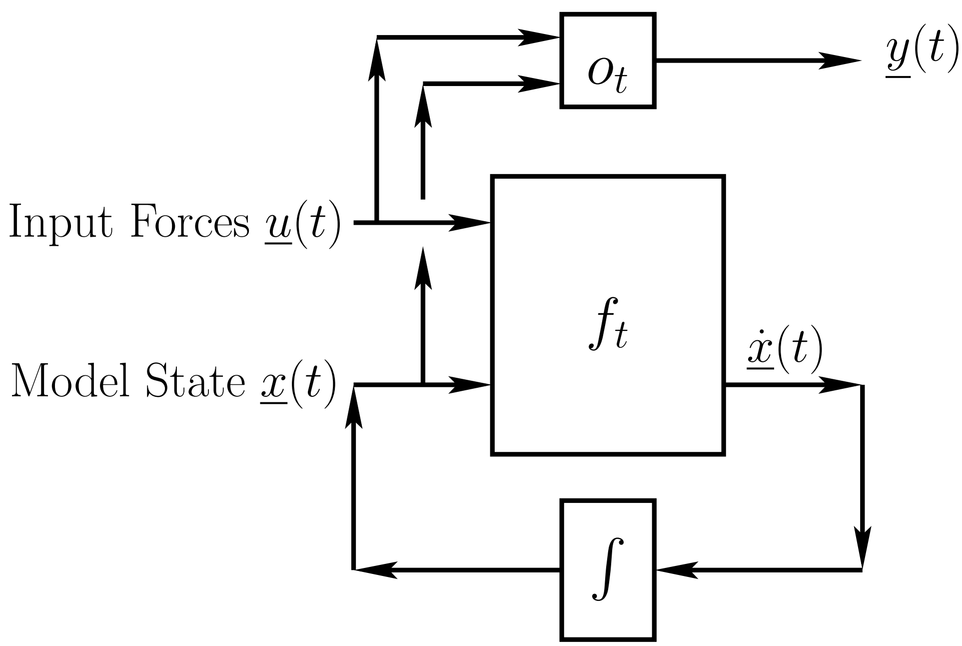
- External forces $f(t)$ *drive the state* to arbitrary points in state space:

$$\begin{aligned} p(t) &= p(t_0) + \int_{t_0}^{t} f(\tau)\,d\tau, \quad t \geq t_0, \quad p(t) \stackrel{\Delta}{=} m\,v(t) \\ x(t) &= x(t_0) + \frac{1}{m}\int_{t_0}^{t} p(\tau)\,d\tau, \quad t \geq t_0 \end{aligned}$$

Forming Outputs

Any system *output* is some function of the state, and possibly the input (directly):

$$\underline{y}(t) \stackrel{\Delta}{=} o_t[\underline{x}(t), \underline{u}(t)]$$



Usually the output is a *linear combination* of state variables and possibly the current input:

$$\underline{y}(t) \stackrel{\Delta}{=} \mathbf{C}\underline{x}(t) + \mathbf{D}\underline{u}(t)$$

where $\mathbf{C}$ and $\mathbf{D}$ are constant matrices of linear-combination coefficients

Numerical Integration

Recall the general state-space model in continuous time:

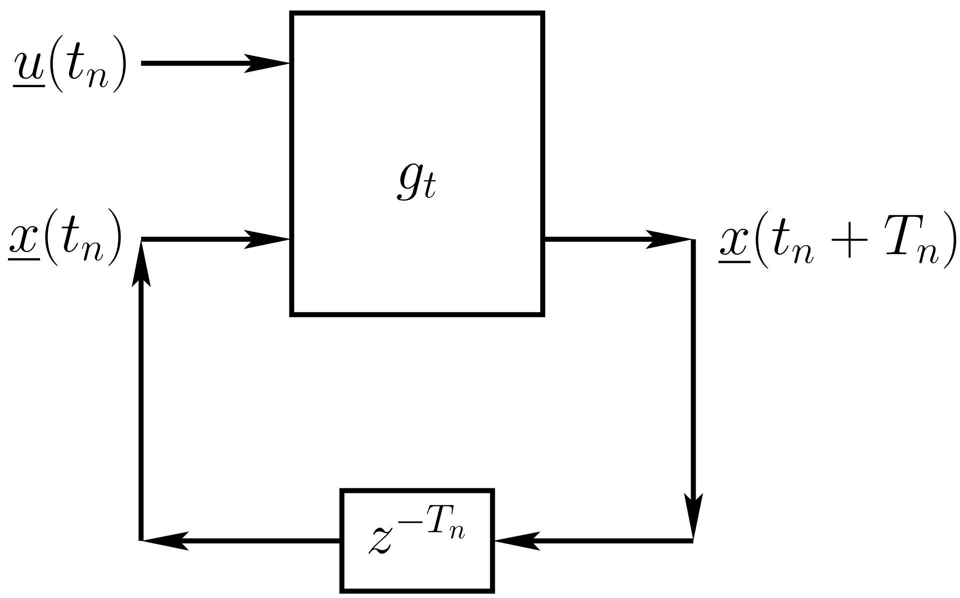
$$\dot{\underline{x}}(t) = f_t[\underline{x}(t), \underline{u}(t)]$$

An approximate discrete-time numerical solution is

$$\underline{x}(t_n + T_n) \;=\; \underline{x}(t_n) + T_n\, f_{t_n}[\underline{x}(t_n), \underline{u}(t_n)]$$

for $n = 0, 1, 2, \ldots$. (*Forward Euler*)

Let $g_{t_n}[\underline{x}(t_n), \underline{u}(t_n)] \stackrel{\Delta}{=} \underline{x}(t_n) + T_n\, f_{t_n}[\underline{x}(t_n), \underline{u}(t_n)]$:



- This is a simple example of *numerical integration* for solving the ODE
- ODE can be nonlinear and/or time-varying
- The sampling interval T_n may be fixed or adaptive

State Definition

We need a *state variable* for the amplitude of each *physical degree of freedom*

Examples:

- Ideal Mass:

  $$\text{Energy} = \frac{1}{2}mv^2 \;\Rightarrow\; \text{state variable} = v(t)$$

  Note that in 3D we get three state variables (v_x, v_y, v_z)

- Ideal Spring:

  $$\text{Energy} = \frac{1}{2}kx^2 \;\Rightarrow\; \text{state variable} = x(t)$$

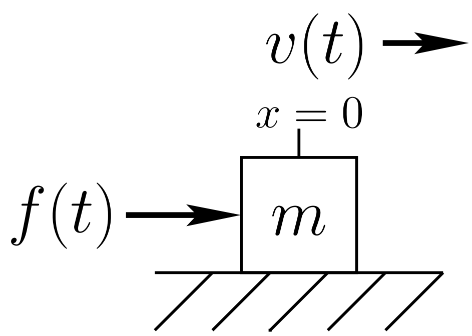
- Inductor: Analogous to mass, so *current*

- Capacitor: Analogous to spring, so *charge* (or voltage = charge/capacitance)

- Resistors and dashpots need no state variables assigned—they are *stateless* (no "memory")

State-Space Model of a Force-Driven Mass

For the simple example of a mass m driven by external force f along the x axis:



- There is only one energy-storage element (the mass), and it stores energy in the form of *kinetic energy*
- Therefore, we should choose the state variable to be velocity $v = \dot{x}$ (or momentum $p = mv = m\dot{x}$)
- Newton's $f = ma$ readily gives the state-space formulation:

$$\boxed{\dot{v} \; = \; \frac{1}{m}f}$$

or

$$\boxed{\dot{p} \; = \; f}$$

- This is a first-order system (no vector needed)

Force-Driven Mass Reconsidered

Why not include *position* $x(t)$ as well as velocity $v(t)$ in the state-space model for the force-driven mass?

$$\begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f(t)$$

We might expect this because we know from before that the complete physical state of a mass consists of its velocity v *and* position x!

Force-Driven Mass Reconsidered and Dismissed

- *Position x does not affect stored energy*

$$E_m = \frac{1}{2} m\, v^2$$

- Velocity $v(t)$ is the only *energy-storing degree of freedom*
- Only velocity $v(t)$ is needed as a state variable
- The initial position $x(0)$ can be kept “on the side” to enable computation of the complete state in position-momentum space:

$$x(t) \;=\; x(0) + \int_0^t v(\tau)\, d\tau$$

- In other words, the position can be derived from the velocity history without knowing the force history
- Note that the external force $f(t)$ can only drive $\dot{v}(t)$. It cannot drive $\dot{x}(t)$ directly:

$$\left[\begin{array}{c} \dot{x}(t) \\ \dot{v}(t) \end{array}\right] \;=\; \left[\begin{array}{cc} 0 & 1 \\ 0 & 0 \end{array}\right] \left[\begin{array}{c} x(t) \\ v(t) \end{array}\right] + \left[\begin{array}{c} 0 \\ 1/m \end{array}\right] f(t)$$

State Variable Summary

- State variable = *physical amplitude* for some *energy-storing degree of freedom*
- **Mechanical Systems:**
  State variable for each
  - *ideal spring* (linear or rotational)
  - *point mass* (or moment of inertia)

  times the number of dimensions in which it can move
- **RLC Electric Circuits:**
  State variable for each *capacitor* and *inductor*
- **In Discrete-Time:**
  State variable for each *unit-sample delay*
- **Continuous- or Discrete-Time:**
  Dimensionality of state-space = *order* of the system (LTI systems)

Modal Representation

The *parallel second-order filter bank* can be computed from the general transfer function (a ratio of polynomials in z) by means of the *Partial Fraction Expansion* (PFE):

$$H(z) \;\stackrel{\Delta}{=}\; \frac{B(z)}{A(z)} \;=\; \sum_{i=1}^{n} \frac{r_i}{1 - p_i z^{-1}}$$

where

$$\begin{aligned} B(z) \;&=\; b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M} \\ A(z) \;&=\; 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}, \quad M < N \end{aligned}$$

- The PFE expands any (strictly proper) transfer function as a parallel bank of (complex) *first-order* resonators
- When the polynomial coefficients b_i and a_i are real, complex poles p_i and residues r_i occur in conjugate pairs, and these can be combined to form second-order sections:

$$\begin{aligned} H_i(z) \;&=\; \frac{r_i}{1 - p_i z^{-1}} + \frac{\overline{r_i}}{1 - \overline{p_i} z^{-1}} \;=\; \frac{r_i - r_i \overline{p_i} z^{-1} + \overline{r_i} - \overline{r_i} p_i z^{-1}}{(1 - p_i z^{-1})(1 - \overline{p_i} z^{-1})} \\ &=\; \frac{2\text{re}\left\{r_i\right\} - 2\text{re}\left\{r_i \overline{p_i}\right\} z^{-1}}{1 - 2\text{re}\left\{p_i\right\} z^{-1} + \left|p_i\right|^2 z^{-2}} \;=\; 2G_i \frac{\cos(\phi_i) - \cos(\phi_i - \theta_i) z^{-1}}{1 - 2R_i \cos(\theta_i) z^{-1} + R_i^2 z^{-2}} \end{aligned}$$

where $p_i \stackrel{\Delta}{=} R_i e^{j\theta_i}$ and $r_i \stackrel{\Delta}{=} G_i e^{j\phi_i}$

Modal Representation, Cont'd

$$H(z) \stackrel{\Delta}{=} \frac{B(z)}{A(z)} = \sum_{i=1}^{n} \frac{r_i}{1 - p_i z^{-1}}$$

- Every transfer function $H(z)$ with real coefficients can be realized as a parallel bank of real first- and/or second-order digital filter sections, as well as a parallel FIR branch when $M \geq N$
- *Modal Synthesis* employs a "source-filter" synthesis model consisting of some driving signal into a parallel filter bank in which each filter section implements the transfer function of some *resonant mode* in the physical system
- Each section (mode) is typically second-order, but fourth-order sections are sometimes used as well (Chant, piano partials)
- In modal synthesis of vibrating strings, each second-order filter implements one "ringing partial overtone" in response to an excitation such as a finger-pluck or piano-hammer-strike

State Space to Modal Synthesis

- The partial fraction expansion works well to create a modal-synthesis system from a transfer function
- It is straightforward to share poles across inputs or outputs:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{A} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \left\{ \frac{1}{A} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right\}$$

- Diagonalizing a state-space model (described below) effectively shares the poles across all of the inputs *and* outputs
- If the original state-space model is a physical model, then the diagonalized system gives a parallel filter bank that is excited from the inputs and observed at the outputs in a physically correct way

State-Space Model Diagonalization Procedure

Linear State-Space Model:

$$\begin{aligned} \underline{y}(n) &= C\underline{x}(n) + D\underline{u}(n) \\ \underline{x}(n+1) &= A\underline{x}(n) + B\underline{u}(n) \end{aligned}$$

To diagonalize this model:

- Find the eigenvectors of A by solving

  $$A\underline{e}_i = \lambda_i \underline{e}_i$$

  for $\underline{e}_i$, $i = 1, 2$, where λ_i is simply the ith pole (eigenvalue of A)

- The N eigenvectors $\underline{e}_i$ are collected into a *similarity transformation matrix:*

  $$E = \left[\, \underline{e}_1 \;\; \underline{e}_2 \;\; \cdots \;\; \underline{e}_N \,\right]$$

  If there are coupled repeated poles, the corresponding missing eigenvectors can be replaced by generalized eigenvectors

- A *generalized eigenvector* $\mathbf{p}$ of matrix A corresponding to eigenvalue λ having multiplicity k is a nonzero solution of $(A - \lambda I)^k \mathbf{p} = 0$

- In matlab: `[Evects,Evals] = eig(A)` — see the state-space appendix in the MUS320 filter book[2] for example matlab
- The E matrix is then used to diagonalize the system by means of a simple *change of coordinates:*

$$\underline{x}(n) \stackrel{\Delta}{=} E\,\underline{\tilde{x}}(n)$$

The new diagonalized system is then

$$\begin{aligned} \underline{\tilde{x}}(n+1) &= \tilde{A}\,\underline{\tilde{x}}(n) + \tilde{B}\,\underline{u}(n) \\ \underline{y}(n) &= \tilde{C}\,\underline{\tilde{x}}(n) + \tilde{D}\,\underline{u}(n), \end{aligned} \tag{1}$$

where

$$\begin{aligned} \tilde{A} &= E^{-1}AE \\ \tilde{B} &= E^{-1}B \\ \tilde{C} &= CE \\ \tilde{D} &= D. \end{aligned} \tag{2}$$

- The transformed system describes the same system relative to new state-variable coordinates $\underline{\tilde{x}}(n)$
- For example, it can be checked that the transfer-function matrix is unchanged

[2] https://ccrma.stanford.edu/~jos/filters/State_Space_Filters.html

Efficiency of Diagonalized State-Space Models

- A general Nth-order state-space model requires approximately N^2 multiply-adds to update for each time step
- After diagonalization by a similarity transform, complexity drops from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$
- All efficient digital filter realizations are $\mathcal{O}(N)$
- Thus, a diagonalized state-space model (modal representation) is a strong contender for applications that can benefit from independent control of resonant modes
- Another advantage is that frequency-dependent characteristics of hearing can be brought to bear
  - Low-frequency modes can be modeled more accurately than high-frequency modes
  - High-frequency modes can be converted into more efficient digital waveguide loops by retuning them to the nearest harmonic mode series

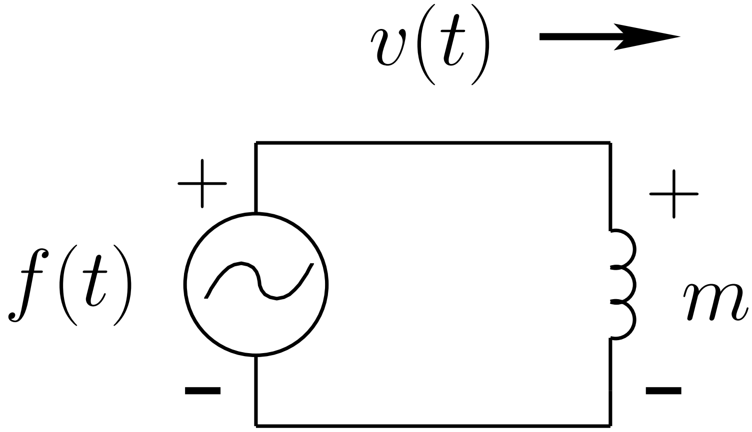
Equivalent Circuits

- “Circuits” and “Ports” from classical circuit/network theory are very useful for *partitioning* complex systems into self-contained sections having well-defined (small) interfaces
- For example, in a “voltage transfer” connection, a low-output-impedance stage drives a high-input-impedance stage
- The large impedance ratio allows us to neglect “loading effects”
- Circuit sections (stages) can be modeled separately

Analog Equivalent Circuits

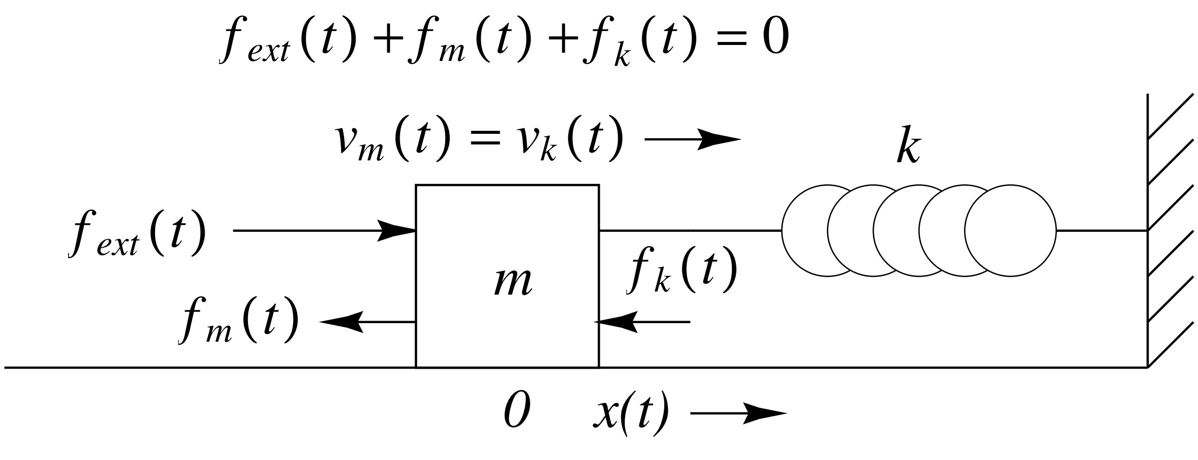
- The name "analog circuit" comes from the following:
  - Electrical capacitors (denoted C) are analogous to physical springs
  - Inductors (L) are analogous to physical masses
  - Resistors (R) are analogous to "dashpots"
- Rs Ls and Cs are called *lumped elements* (as opposed to distributed-parameter devices such as capacitance and inductance per unit length in a transmission line
- Lumped elements are described by ODEs while distributed-parameter systems are described by PDEs
- RLC analog circuits can be constructed as *equivalent circuits* for lumped dashpot-mass-spring systems
- These equivalent circuits can be digitized by *finite difference* or *wave digital* methods
- PDEs describing *distributed*-parameter systems can be digitized by finite difference methods as well, or, when wave propagation is the dominant effect, *digital waveguide* methods

Equivalent Circuit for a Force-Driven Mass



- Mass m is an *inductor* $L = m$ Henrys
- Driving force $f(t)$ is a *voltage source*
- Mass velocity $v(t)$ is the *loop current*

Mass-Spring-Wall System

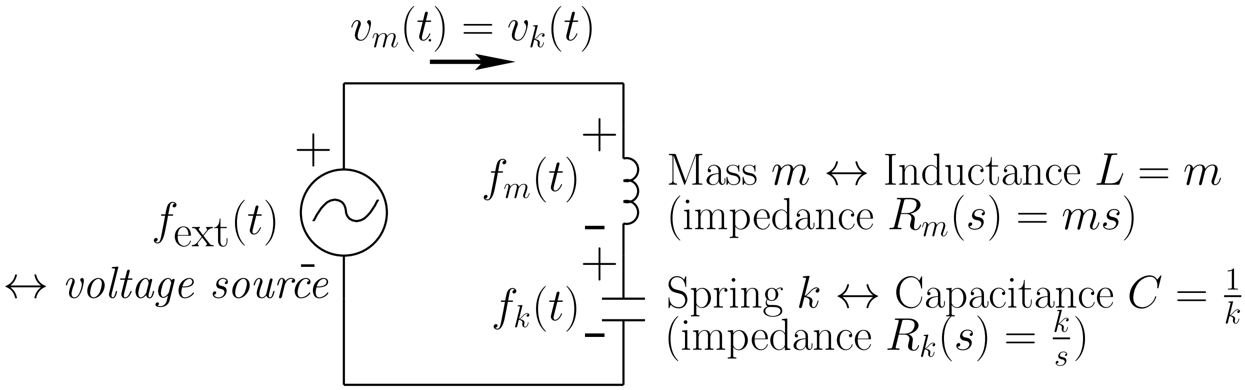


- Driving force $f_{\text{ext}}(t)$ is to the right on the mass
- Driving force + mass inertial force + spring force = 0
- Mass velocity = spring velocity
- This is a *series* combination of the spring and mass

If two physical elements are connected so that they share a *common velocity*, then they are said to be formally connected *in series*

Equivalent Circuit for Mass-Spring-Wall

The "series" nature of the connection becomes more clear when the *equivalent circuit* is considered:



- The driving force is applied to the mass such that a positive force results in a positive mass displacement and positive spring displacement (compression)
- The common mass and spring velocity appear as a single current running through the inductor and capacitor that model the mass and spring, respectively

Impedance Networks

The concept of impedance is central in classical electrical engineering. The simplest case is *Ohm's Law* for a resistor R:

$$V(t) = R\,I(t)$$

where

$V(t)$ denotes the voltage across the resistor at time t

$I(t)$ is the current through the resistor

Impedance is the resistance R

For the corresponding *mechanical* element, the *dashpot*, Ohm's law becomes

$$f(t) = \mu\, v(t)$$

- $f(t)$ is the force across the dashpot at time t
- $v(t)$ is its *compression velocity*
- Dashpot impedance value μ is a *mechanical resistance*
- Dashpots and resistors are always *real, positive impedances*

Complex Impedances

- Models of damping in practical physical systems are rarely completely independent of frequency, like the ideal dashpot
- Thanks to the *Laplace transform* (or *Fourier transform*), the concept of impedance easily extends to masses and springs as well
- We need only allow impedances to be *frequency-dependent*
- For example, the Laplace transform of Newton's $f = ma$ yields, by the *differentiation theorem*,

$$F(s) \;=\; m\,X(s) \;=\; m\,sV(s) \;=\; m\,s^2A(s)$$

  where

  - $F(s) = \mathcal{L}_s\{f\} =$ Laplace transform of $f(t)$ (initial conditions assumed zero)
  - *Impedance of a point-mass* is

$$R_m(s) \;\stackrel{\Delta}{=}\; \frac{F(s)}{V(s)} \;=\; ms$$

  - Specializing the Laplace transform to the Fourier transform by setting $s = j\omega$ gives

$$R_m(j\omega) \;=\; jm\omega$$

– Impedance of a *spring* with spring-constant k is

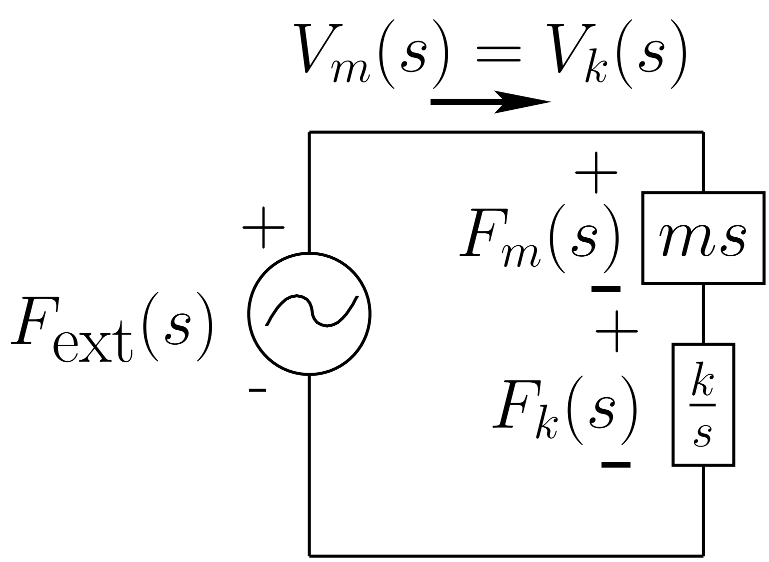
$$R_k(s) = \frac{k}{s}$$

$$R_k(j\omega) = \frac{k}{j\omega}$$

Important Benefit of Frequency-Domain Impedance

Every interconnection of masses, springs, and dashpots (every RLC equivalent circuit) can be analyzed as a simple *resistor network*

Impedance Diagram for Force-Driven Series Mass-Spring



Impedance diagram for the force-driven, series arrangement of mass and spring

Viewing the circuit as a (frequency-dependent) resistor network, it is easy to write down, say, the Laplace transform of the force across the spring using the *voltage divider* formula:

$$F_k(s) \; = \; F_{\text{ext}}(s)\frac{R_k(s)}{R_m(s)+R_k(s)} \; = \; F_{\text{ext}}(s)\frac{k/m}{s^2+k/m}$$

We will discuss further equivalent-circuit and impedance-network models such as these, as well as ways to digitize them into digital-filter form

Wave Digital Filters

The idea of wave digital filters is to digitize RLC circuits (and certain more general systems) as follows:

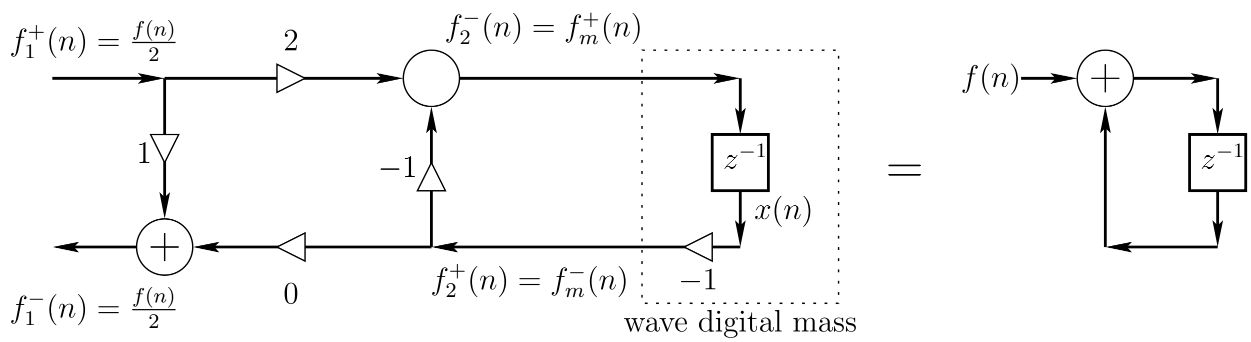
1. Determine the ODEs describing the system (PDEs also workable)
2. Express all physical quantities (such as force and velocity) in terms of *traveling-wave components*
3. The traveling wave components are called *wave variables*
4. For example, the force $f(n)$ on a mass is decomposed as $f(n) = f^+(n) + f^-(n)$, where $f^+(n)$ is regarded as a traveling wave propagating *toward* the mass, while $f^-(n)$ is seen as the traveling component propagating *away from* the mass
5. A "traveling wave" view of force mediation (at the speed of light) is actually much closer to underlying physical reality than any instantaneous model
6. Second, digitize the resulting traveling-wave system using the *bilinear transform*

   The bilinear transform is equivalent in the time domain to the *trapezoidal rule for numerical integration*

7. Connect N elementary units together by means of N-*port scattering junctions*
8. There are two basic types of scattering junction, one for parallel, and one for series connection
9. The theory of scattering junctions is introduced in the *digital waveguide* context

A more detailed introduction to WDFs is provided in an appendix of the text

Wave digital model (mass driven by external force $f(n)$):



- We will not make much use of WDFs in this course, preferring instead more prosaic finite-difference models for simplicity
- However, closely related concepts are used extensively in the digital waveguide modeling context

Lumped Elements versus Distributed Parameters

- Masses, springs, dashpots, inductors, capacitors, and resistors are examples of so-called *lumped* elements
- Perhaps the simplest *distributed* element is the continuous ideal delay line
- Because it carries a *continuum* of independent amplitudes, the order (number of state variables) is *infinity* for a continuous delay line of any length!
- However, we typically work with *sampled, bandlimited* systems $\Rightarrow$ delay lines have a *finite number of state variables* (one for each delay element)
- Networks of lumped elements yield finite-order state-space models
- Even one distributed element jumps the order to infinity (until it is bandlimited and sampled)

Digital Waveguides

- Digital waveguide models are built out of digital delay-lines and filters (and nonlinear elements), and they can be understood as propagating and filtering sampled traveling-wave solutions to the wave equation (PDE), such as for air, strings, rods, and the like
- Strings, woodwinds, and brasses comprise three of the four sections of an orchestra (all but percussion)
- Digital waveguides have also been extended to propagation in 2D, 3D, and beyond
- They are not finite-difference models, but paradoxically they are equivalent under certain conditions)
- A summary of historical aspects appears in an appendix of the text

Digital Waveguide Models

We may begin with the PDE for the *ideal 1D wave equation:*

$$y'' = c^2\ddot{y}$$

where

$$\begin{aligned} c &= \text{traveling-wave propagation speed} \\ y(t,x) &= \text{displacement at time } t \text{ and position } x \end{aligned}$$

- For example, y can be the transverse displacement of an ideal stretched string or the longitudinal displacement (or pressure, velocity, etc.) in an air column
- The independent variables are time t and the distance x along the string or air-column axis
- The partial-derivative notation is more completely written out as

$$\begin{aligned} \ddot{y} &\triangleq \frac{\partial^2}{\partial t^2}y(t,x) \\ y'' &\triangleq \frac{\partial^2}{\partial x^2}y(t,x). \end{aligned}$$

- Recall that the ideal wave equation derives directly from Newton's laws $f = ma$

- In the case of vibrating strings, the wave equation is derived from first principles to be

$$Ky'' = \epsilon \ddot{y}$$

(Restoring Force $=$ Mass Density times Acceleration), where

$$\begin{aligned} K &\stackrel{\Delta}{=} \text{string tension} \\ \epsilon &\stackrel{\Delta}{=} \text{linear mass density.} \end{aligned}$$

- The left-hand side of the wave equation (the restoring force as tension times "curvature"), was first derived by Brook Taylor of "Taylor series" fame
- Thus, it turns out that the propagation speed c can be written in terms of the string tension K and mass density ϵ as

$$c = \sqrt{\frac{K}{\epsilon}}$$

- As has been known since d'Alembert, the 1D wave equation is obeyed by arbitrary *traveling waves* at speed c:

$$y(t,x) = y_r(t - x/c) + y_l(t + x/c)$$

(Just plug $y_r(t - x/c)$ or $y_l(t + x/c)$ or any linear combination of them into the wave equation to verify this)

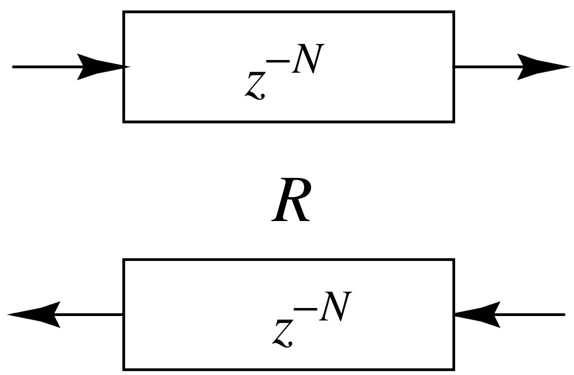
- Next, the traveling-waves are *sampled:*

$$\begin{aligned} y(nT, mX) &= y_r(nT - mX/c) + y_l(nT + mX/c) \quad (X \triangleq cT) \\ &= y_r(nT - mT) + y_l(nT + mT) \\ &\triangleq y^+(n-m) + y^-(n+m) \end{aligned}$$

  where T denotes the time sampling interval in seconds, $X = cT$ denotes the spatial sampling interval in meters, and y^+ and y^- are defined for notational convenience

- An ideal string (or air column) can thus be simulated using a *bidirectional delay line* for the case of an N-sample section of string or air column

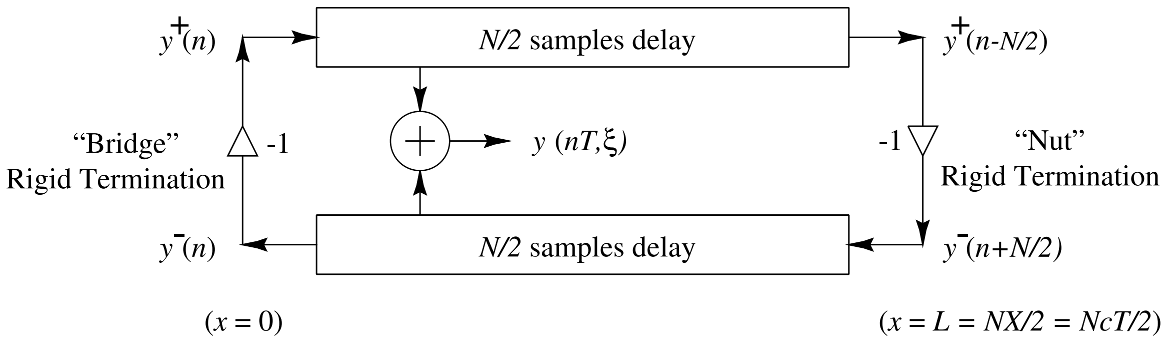
Digital Waveguide Definition



A *digital waveguide* is defined as a *bidirectional delay line* at some wave impedance R

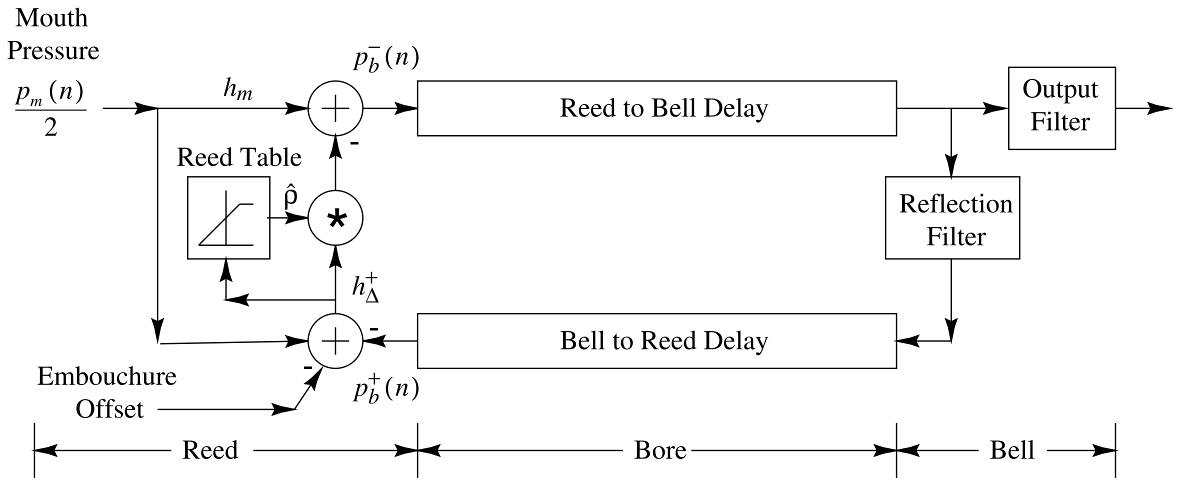
- A digital waveguide simulates (exactly) sampled traveling waves in ideal strings and acoustic tubes
- The "R" label denotes its *wave impedance*, which is needed to connect digital waveguides to each other and to other kinds of computational physical models (such as finite difference schemes)
- While propagation speed on an ideal string is $c = \sqrt{K/\epsilon}$, we will derive that the wave impedance is $R = \sqrt{K\epsilon}$.

Digital waveguide model of a rigidly terminated ideal string

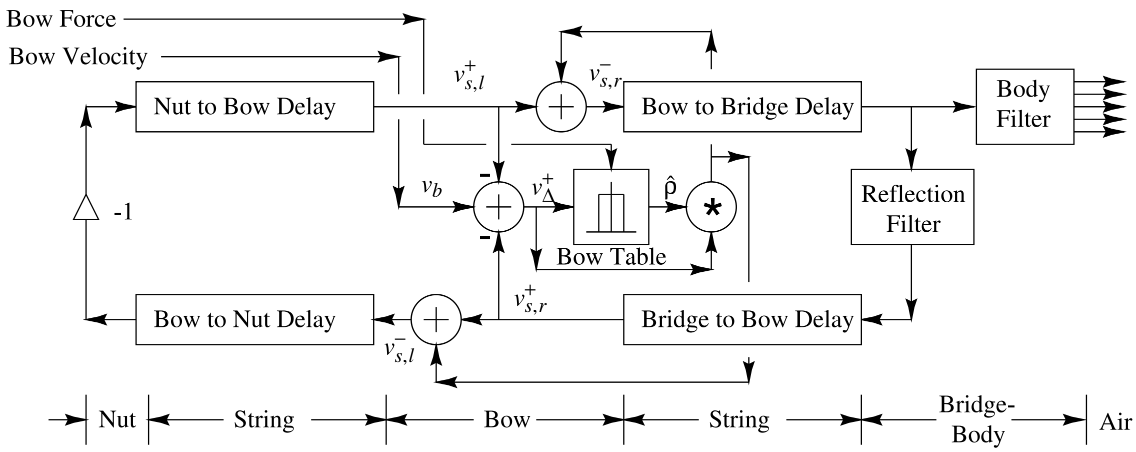


- One polarization-plane of transverse vibration
- Traveling-wave components $\stackrel{\Delta}{=}$ *displacement* samples
- Diagram for velocity and acceleration waves identical (all have inverting reflection at each rigid termination) (slope and force waves reflect with no sign inversion)
- Output signal $y(nT, \xi)$ formed by summing traveling-wave components at the desired "virtual pickup" location (position $x = \xi$)

Digital waveguide model of a single-reed, cylindrical-bore woodwind, such as a clarinet



Digital waveguide model for a bowed-string instrument, such as a violin



Summary of Models Considered

1. Ordinary Differential Equations (ODE)
2. Partial Differential Equations (PDE)
3. Difference Equations (DE)
4. Finite Difference Schemes (FDS)
5. (Physical) State Space Models
6. Transfer Functions (between physical signals)
7. Modal Representations (Parallel 2nd-Order Filters)
8. Equivalent Circuits
9. Impedance Networks
10. Wave Digital Filters (WDF)
11. Digital Waveguide (DW) Networks

General Modeling Procedure

While each situation tends to have special opportunities, the following procedure generally works well:

1. Formulate a *state-space model*
2. If it is nonlinear, use *numerical time-integration*:
   - Explicit (causal finite difference scheme)
   - Implicit (iteratively solved each time step)
   - Semi-Implicit (truncated iterations of Implicit)
3. In the linear case, *diagonalize* the state-space model to obtain the *modal representation*
   - Implement isolated modes as second-order filters ("biquads")
   - Implement *quasi-harmonic* mode series as *digital waveguides*

It is usually good to partition the system into separate modules when possible

For example, strings, horns, and woodwind bores have quasi-harmonic modes and can be modeled as digital waveguides from the outset