Week 1

The Signals and Systems Abstraction

The system transforms an input signal into an output signal

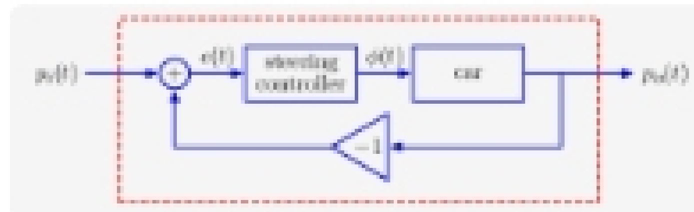
Both the input and output are *signals*

A signal is a mathematical function with an independent variable (i.e. time) and dependent variable

The *system* is described by the way that it transforms the input signal into the output signal

All useful models focus on the most relevant signals and ignore those of lesser significance

Check yourself 1: List at least four possible output signals for the car-steering problem



- The car's three-dimensional position
- The car's angular position
- The rotational speed of the wheels
- The temperature of the tires

Modularity, primitives, and composition

In a composite system the steering controller determines $\phi(t)$, which is input into the car. The car generates po(t), which is subtracted from pi(t) to get e(t) which is input to the steering controller

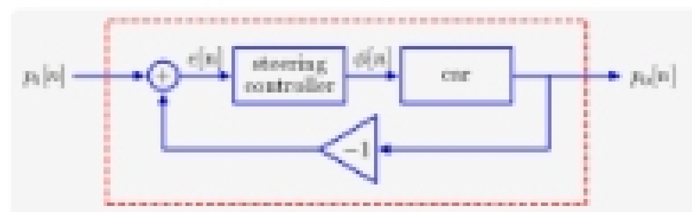
The triangular component is called a *gain* or *scale* of -1

Its output is equal to -1 times its input

The triangle symbol is used to indicate that we are multiplying all the values of the signal by a numerical constant, which is shown inside the triangle

The dashed-red box illustrates *modularity* of the signals and systems abstraction

Discrete-Time Signals and Systems



Signals whose independent variables are discrete (i.e. take only integer values)

Some signals are found in nature

For example, the primary structure of DNA is described by a sequence of base-pairs

Discrete-time signals are found in computers

For example, the difference between the desired position pi(t) and our actual position po(t) is an error signal e(T), which is a function of continuous time t

If the controller only observes this signal at regular sampling intervals T, then its input could be regarded as a sequence of values x[n] that is indexed by the integer

The relationship between the discrete-time sequence x[n] and the continuous signal x(t) is given by **x[n] = x(nT)**, where T is the length of one time step

Sampling converts a signal of continuous domain to one of discrete domain

For example, images are typically represented as arrays of pixels accessed by integer-valued rows and columns, rather than as continuous brightness fields, indexed by real-valued spatial coordinates

Linear, Time-Invariant Systems

State machines allow us to specify *any* discrete-time system whose output is computable from it history of previous inputs

The representation of systems as state machines allows us to execute a machine on any input we would like, in order to see what happens

Execution lets us examine the behavior of the system for any particular input for any particular finite amount of time, but does not let us characterize any general properties of the system of its long-term behavior

A small but powerful subclass of the whole class of state machines, called discrete-time *linear time-invariant (LTI) systems*, which will allow deeper forms of analysis

In LTI systems:

- Inputs and outputs are real numbers
- The state is some fixed number of previous inputs to the system as well as a fixed number of previous outputs of the system
- The output is a fixed, linear function of the current input and any of the elements in the state

We will restrict our attention to the case where the input is a single real number and the output is a single real number

LTI systems can be analyzed mathematically, in a way that lets us characterize some properties of their output signal for any possible input signal

Another important property of LTI systems is that they are compositional: the cascade, parallel, and feedback combinations of LTI systems are themselves LTI systems

Discrete-Time Signals

The PCAP system for discrete time signals

A *signal* is an infinite sequence of *sample* values at discrete time steps,

A capital X stands for the whole input signal and x[n] stands for the value of signal X at time step n

If there is a single system under discussion, use X for the inpiut signal to that system and Y for the output signal

We wil say that systems *transduce* input signals into output signals

Unit Sample Signal

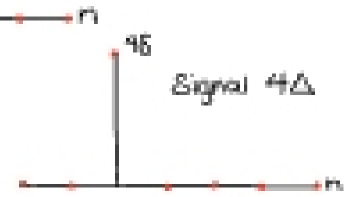
We will work with a single primitive, called the *unit sample signal*, Δ

It is defined on all positive and negative integer indices as follows $\delta[n] = \begin{cases} 1 & \text{if } n=0 \\ 0 & \text{otherwise} \end{cases}$

Scaling

Our first operation will be *scaling*, or *multiplication by a scalar*

The result of multiplying any signal X by a scalar c is a signal so that, if **Y = c · X, then y[n] = c · x[n]**



Delay

The next operation is the *delay* operation

The result of delaying a singal X by a new signal RX such that, **if $Y = RX$, then $y[n] = x[n-1]$**

Addition

Finally, we can *add* two signals together.

Addition of signals is accomplished component-wise, so that **if $Y = X1 + x2$, then $y[n] = x1[n] + x2[n]$**

Advancing (skipped)

Algebraic Properties of Operation and Signals

Adding is commutative and associative

Scaling is commutative

Scaling distributes over addition

Furthermore, R distributers over addition and scaling

These algebraic relationships mean that we can rewrite $3\Delta + 4R\Delta - 2R^2\Delta$ as $(3 + 4R - 2R^2)\Delta$

Feedforward Systems

A subclass of discrete-time LTI systems, which are exactly those that can be described as performing some combination of scaling, delay, and addition operations on the input signal

Representing Systems

We can represent systems using operator equations, difference equation, block diagrams, and Python state machines.

Operator equation

An operator equation is a description of how signal are related to one another, using the operations of scaling, delay, and addition on whole signals

Consider a system that has an input signal X, and whose output signal is X - XR

We can describe the system using the operator equation **$Y = X - XR$**

We cab rewrite this as $Y = (1 - R)X$

Feedforward systems can always be described using an operator equation of the form **$Y = \phi X$**, where ϕ is a polynomial in R

Difference Equation

An alternative representation of the relationship between signals is a *difference equation*

A difference equation describes a relationship that holds among samples (values at particular times) of signals

The operator equation **$Y = X - RX$** can be expressed as this equivalent equation **$y[n] = x[n] - x[n-1]$**

The operation delaying a signal can be seen here as referring to a sample of that signal at time step n-1

Difference equations are convenient for step-by-step analysis, letting us compute the value of an output signal at any time step, given the values of the input signal

Block Diagrams

Another way of describing a system is by drawing a block diagram, which is made up of components, connected by lines with arrow on them

These lines represent signals

All lines that are connected to one other (not going through a round, triangular, or circular component) represent the same signal

The components represent systems

There are primitive components corresponding to our operations on signals:

- Delay component is drawn as rectangles, labeled delay
  - If X is the signal on the line coming in to the delay, then the signal coming out is R
- Scale component is drawn as triangles, labeled with a positive or negative number c
  - If X is the signal on the line coming into the scale, then the signal coming out is c*X
- Adder component is drawn as circles labeled with +
  - if X1 and X2 are the signals on the lines that point to the adder the the signal coming out is x1 + x2

Combination of systems

To combine LTI systems, we will use the same cascade and parallel-add operations as we had for state machines

Cascade Multiplication

When we make a cascade combination of two systems, we let the output of one system be the input for another

So if the system M1 has an operator equation **$Y = \Phi 1X$** and system M2 has an operator equation **$Z=\Phi 2W$**, and then we compose M1 and M2 in a cascade by setting **$Y = W$**, then we have a new system, with input signal X and output signal Z and operator equation **$Z = (\Phi 1 \cdot \Phi 2)X$**

Parallel addition

When we make parallel addition combination of two systems, the output signal is the sum of the output signals that would have resulted from the individual systems

Because addition of polynomials is associative and commutative, then so is parallel addition of feedforward linear systems

Combining Cascade and Parallel Operations

Finally, the distributive law applies for cascade and parallel combinations, *for a system at rest*, in the same way that it applies for multiplication and addition of polynomials

Feedback Systems

Feedforward: the dependencies have all flowed from the input through to the output, with no dependence of an output on previous output values

We will expend our representation and analysis to handle the general class of LTI systems in which the output can depend on any finite number of previous input or output values

Accumulator Example

- Consider the difference equation **y[n] = x[n] + y[n-1]**
  - With the equation we run into the question: what is the value of y[n-1]?
- In our treatment of feedback systems, we will generally assume that they start "at rest", which means that all values of the inputs and outputs at steps less than zero are zero
  - In the example given, we see that we have a transient input with a persistent (infinitely many non-zero samples) output

General Form of LTI Systems

- In general, LTI systems can be described by the difference equations of the form

$$y[n] = c_0 y[n-1] + c_1 y[n-2] + \ldots + c_{k-1} y[n-k] + d_0 x[n] + d_1 x[n-1] + \ldots + d_j x[n-j]$$

  - The state of this system consists of the k previous output values and j previous input values