6.01: Introduction to EECS 1

software engineering
feedback and control
circuits
probability and planning

Dennis M. Freeman
Leslie P. Kaelbling
Tomas Lozano-Perez
Harold Abelson
Jacob White

6.01: Design Goals

6.01 is designed to be the entry-point to MIT's EECS Department.

- first of two required subjects: 6.01 and 6.02
- typically taken by first or second year students
- introductory: broad exposure, but not superficial
- teach core ideas in EECS within an authentic context
- practice – theory – practice

6.01: Intellectual Themes

The intellectual themes in 6.01 are recurring themes in EECS.

- Managing complexity
- Modeling and controlling physical systems
- Managing interaction of computing artifacts with physical systems
- Building systems that are robust to uncertainty

Intellectual themes are developed in context of a mobile robot.

Not a course about robots — robots provide versatile platform.

6.01: Pedagogical Structure

6.01 has a hands-on structure.

Weekly schedule:

- 1.5 hours of lecture
- 1.5 hours of "software lab"
- 3 hours of "design lab"
- online tutor

Grading:

- weekly nanoquizzes (15 minutes)
- weekly online tutor exercises
- 3 individualized oral interviews
- midterm and final exam

6.01: Development

6.01 was developed over two year time period.

6.01 is in its second year as a requirement.

Term	# of students
S06	24
F06	18
S07	39
F07	94
S08	234
F08	110
——	
S09	273 (pre-registration)

Topics

6.01 is organized in four modules.

- Software Engineering
- Feedback and Control
- Circuits
- Probability and Planning

Challenge: meaningfully cover 4 major topics in 1 introductory subject.

Approach: focus on a few important ideas to be pursued in depth.

Software Engineering: Fundamentals

Focus on a few important ideas.

- object-oriented programming
  - inheritance
  - evaluation and environments
- data structures: lists, dictionaries, arrays

Objects and Data Structures in the Real World

Laboratory exercises.

- Polynomial class
  - simple illustration of OOP
  - revisited in Signals and Systems (implementing SystemFunctions)
- state machines and terminating state machines
  - as robot "brains"

    `sensorInputs` → brain → `Actions`
  - combinators

    `CascadeSM`: make new SM by cascading two SM's
    `ParallelSM`: make new SM by running two SM's in parallel
    `RepeatTSM`: repeat a TSM for n steps
    `SequentialTSM`: do several TSM's one after another
    `StepwiseIf`: on each step, run either SM1 or SM2
    `StepwiseUntil`: step a SM until condition is true

High-level messages: Modularity and Abstraction

Signals and Systems: Fundamentals

Focus on a few important ideas (discrete time only!)

- Representations: diff eq's, block diagrams, system functions
- Responses: modes and poles
- Feedback and control: proportional/prop plus delay controllers

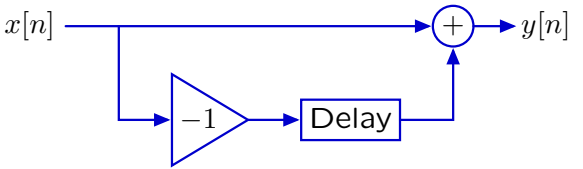
Signals and Systems: Representations

Abstractions and modularity in signals and systems.

Difference equations: mathematically compact.

$$y[n] = x[n] - x[n-1]$$

Block diagrams: illustrate signal flow paths.



Operator representation: analyze systems as polynomials.

$Y = (1 - \mathcal{R})\, X$; where the $\mathcal{R}$ operator represents a unit delay

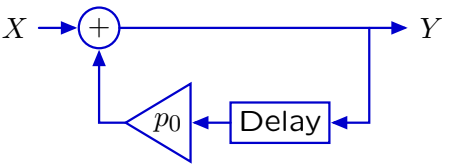
System Functions

$$\frac{Y}{X} = (1 - \mathcal{R}) = 1 - \frac{1}{z} = \frac{z-1}{z} \; ; \quad \text{where} \quad z = \frac{1}{\mathcal{R}}$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

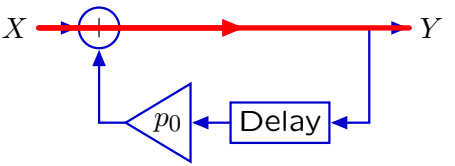


$$\frac{Y}{X} = \frac{1}{1 - p_0 \mathcal{R}}$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

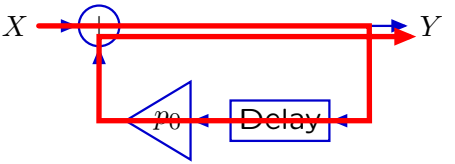


$$\frac{Y}{X} = \frac{1}{1 - p_0 \mathcal{R}} = 1$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

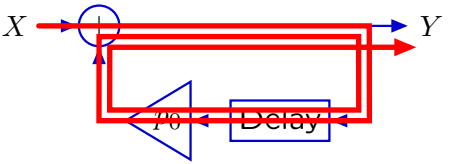


$$\frac{Y}{X} = \frac{1}{1 - p_0\mathcal{R}} = 1 + p_0\mathcal{R}$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

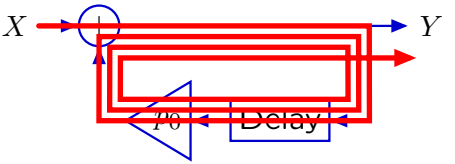


$$\frac{Y}{X} = \frac{1}{1 - p_0\mathcal{R}} = 1 + p_0\mathcal{R} + p_0^2\mathcal{R}^2$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

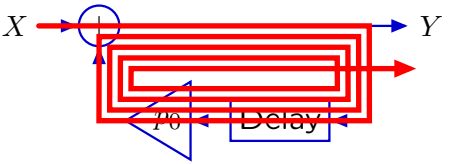


$$\frac{Y}{X} = \frac{1}{1 - p_0\mathcal{R}} = 1 + p_0\mathcal{R} + p_0^2\mathcal{R}^2 + p_0^3\mathcal{R}^3$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent ... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

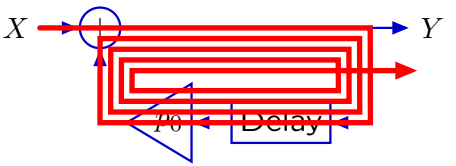


$$\frac{Y}{X} = \frac{1}{1 - p_0\mathcal{R}} = 1 + p_0\mathcal{R} + p_0^2\mathcal{R}^2 + p_0^3\mathcal{R}^3 + p_0^4\mathcal{R}^4 + \cdots$$

Signals and Systems: Modes and Poles

The response of a cyclic system can be persistent
... even when its input is transient.

Example: Find $y[n]$ when $x[n] = \delta[n]$.

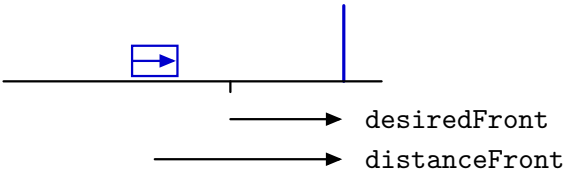


$$\frac{Y}{X} = \frac{1}{1 - p_0\mathcal{R}} = 1 + p_0\mathcal{R} + p_0^2\mathcal{R}^2 + p_0^3\mathcal{R}^3 + p_0^4\mathcal{R}^4 + \cdots$$

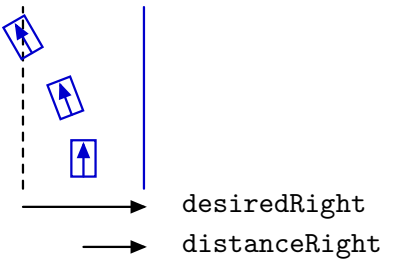
The response is a geometric sequence with base p_0 (**a pole**).

Poles in the real world: laboratory exercises.

wallFinder: move to desired distance from a wall.



wallFollower: move along wall maintaining desired distance from it.



Feedback and Control: Software Tools

Use software tools to facilitate analysis and design.

Example: SystemFunction class (builds on previous use of Polynomial class).

```
h1 = SystemFunction(Polynomial([1,0,0]),Polynomial([1,2,1]))
h2 = SystemFunction(Polynomial([1,0]),Polynomial([1]))
h3 = CascadeSF(h1,h2)
h4 = FeedbackSF(h3)
h4.poles()
```

Used to plot root loci and to find gains for higher order controllers for wallFinder and wallFollower.

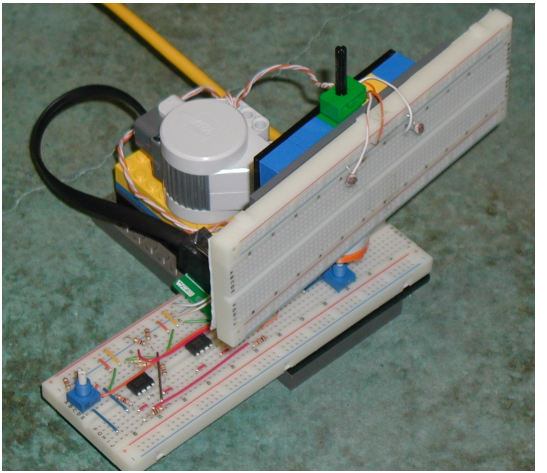
Circuits: Fundamentals

Focus on a few important ideas.

- KVL, KCL, and constitutive equations
- Series and parallel connections
- Voltage and current dividers
- Thevenin equivalents
- Op-amps

Circuits: Tangible Examples

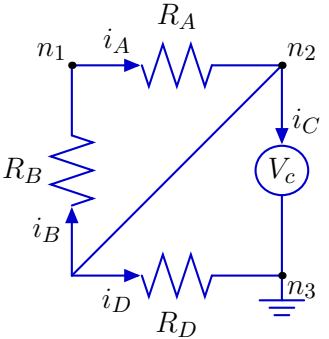
Laboratory exercises.



- motor servo controller (rotating neck for robot "head")
- phototransistor (robot "eyes")
- light tracking system

Circuits: Software Tools

Solving linear constraints.




```
c = Circuit([Resistor(100,'v1','v2'),     # insert components
             Resistor(200,'v2','v1'),
             Resistor(100,'v2','v3'),
             VSrc(10,'v2','v3')])
ckt = c.makeEquationSet('v3')             # 'v3' is ground
ckt.solve()                               # solve
```

Probability and Planning: Fundamentals

Focus on a few important ideas.

- discrete probability and state estimation
  - Bayes' rule
- Search and dynamic programming
  - bread-first
  - depth-first
  - pruning

Probability and Planning: Applications

Laboratory exercises.

- Mapping: drive robot around unknown space and make map.
- Localization: give robot map and ask it to find where it is.
- Planning: plot a route to a goal in a maze

Grand finale: locate light(s) in a maze

- locate position in known maze
- plan a path to search the maze
- look for light with "eyes" (phototransducer)
- track light by turning head ("neck" servo)

Challenges

Our students enter 6.01 with wide range of programming experience. Challenge: how to engage students with such different skill levels?

Strategy

- Start subject with two weeks of programming, focusing on object-oriented techniques.
- Introduce new concepts in programming throughout term, to support other learning objectives (signals and systems, circuits, probability and planning).

Rationale

- Minimizes programming hurdle at beginning of subject
- Students with little programming background see a gentler learning curve
- Students with substantial programming background see programming in new contexts

6.01: Introduction to EECS 1

software engineering
feedback and control
circuits
probability and planning

More information
http://web.mit.edu/6.01

Contact: Denny Freeman
freeman@mit.edu