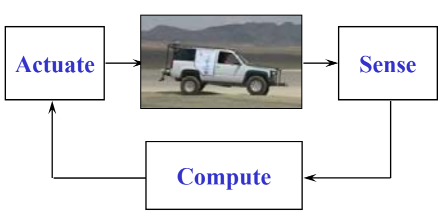
CDS 101: Lecture 2.1
System Modeling

Richard M. Murray
6 October 2003

Goals:
- Define what a model is and its use in answering questions about a system
- Introduce the concepts of state, dynamics, inputs and outputs
- Provide examples of common modeling techniques: differential equations, difference equations, finite state automata

Reading:
- Åström and Murray, *Analysis and Design of Feedback Systems,* Ch 2
- Advanced: Lewis, *A Mathematical Approach to Classical Control*, Ch 1

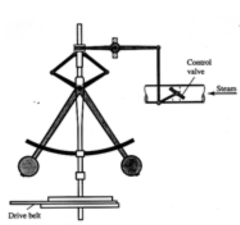
Review from last week

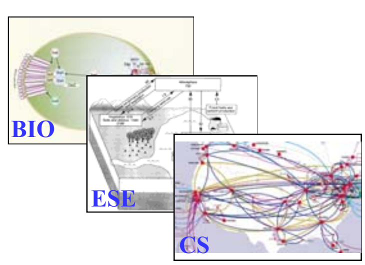


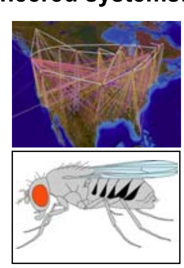
Control =
Sensing + Computation + Actuation

Feedback Principles
- Robustness to Uncertainty
- Design of Dynamics

Many examples of feedback and control in natural & engineered systems:







Model-Based Analysis of Feedback Systems

Analysis and design based on *models*

- A model provides a *prediction* of how the system will behave
- Feedback can give counter-intuitive behavior; models help sort out what is going on
- For control design, models don't have to be exact: *feedback* provides robustness

Control-oriented models: *inputs* and *outputs*

The model you use depends on the questions you want to answer

- A single system may have many models
- Time and spatial scale must be chosen to suit the questions you want to answer
- Formulate questions *before* building a model

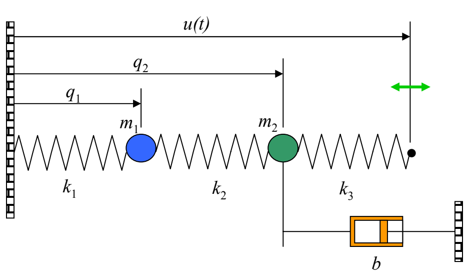
Weather Forecasting

- Question 1: how much will it rain tomorrow?
- Question 2: will it rain in the next 5-10 days?
- Question 3: will we have a drought next summer?

Different questions ⇒ different models

6 Oct 03 R. M. Murray, Caltech CDS 3

Example #1: Spring Mass System



Applications

- Flexible structures (many apps)
- Suspension systems (eg, "Bob")
- Molecular and quantum dynamics

Questions we want to answer

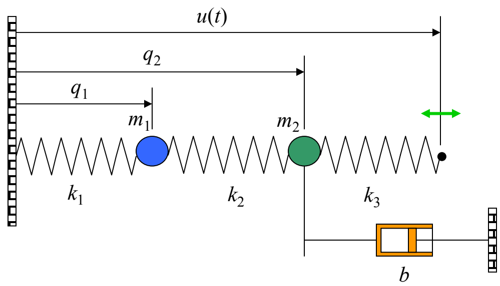
- How much do masses move as a function of the forcing frequency?
- What happens if I change the values of the masses?
- Will Bob fly into the air if I take that hill at 25 mph?

Modeling assumptions

- Mass, spring, and damper constants are fixed and known
- Springs satisfy Hooke's law
- Damper is (linear) viscous force, proportional to velocity

6 Oct 03 R. M. Murray, Caltech CDS 4

Modeling a Spring Mass System



Model: rigid body physics (Ph 1)

- Sum of forces = mass $*$ acceleration
- Hooke's law: $F = k(x - x_{rest})$
- Viscous friction: $F = b\, v$

$$\begin{aligned} m_1 \ddot{q}_1 &= k_2(q_2 - q_1) - k_1 q_1 \\ m_2 \ddot{q}_2 &= k_3(u - q_2) - k_2(q_2 - q_1) - b\dot{q}_2 \end{aligned}$$

Converting models to state space form

- Construct a *vector* of the variables that are required to specify the evolution of the system
- Write dynamics as a *system* of first order differential equations:

$$\frac{dx}{dt} = f(x,u) \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^p$$

$$y = h(x) \qquad y \in \mathbb{R}^q$$

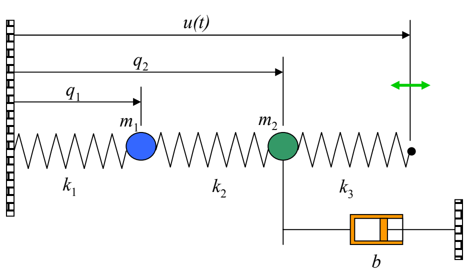
$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \frac{k_2}{m}(q_2 - q_1) - \frac{k_1}{m} q_1 \\ \frac{k_3}{m}(u - q_2) - \frac{k_2}{m}(q_2 - q_1) - \frac{b}{m}\dot{q}_2 \end{bmatrix}$$

$$y = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

"State space form"

6 Oct 03 R. M. Murray, Caltech CDS 5

Frequency Response for a Mass Spring System



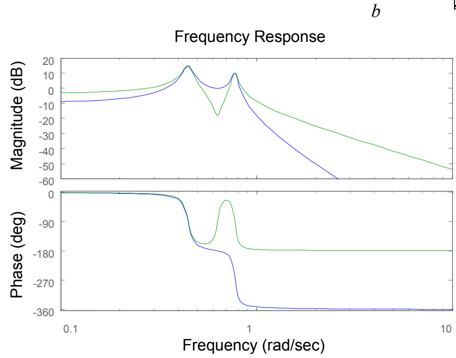
Steady state frequency response

- Force the system with a sinusoid
- Plot the "steady state" response, after transients have died out
- Plot relative magnitude and phase of output versus input (more later)

Matlab simulation (see handout)

```
function dydt = f(t, y, ...)
u = 0.00315*cos(omega*t);
dydt = [
  y(3);
  y(4);
  -(k1+k2)/m1*y(1) + k2/m1*y(2);
  k2/m2*y(1) - (k2+k3)/m2*y(2)
      - b/m2*y(4) + k3/m2*u ];

t,y] = ode45(dydt,tspan,y0,[], k1,
k2, k3, m1, m2, b, omega);
```



6 Oct 03 R. M. Murray, Caltech CDS 6

Modeling Terminology

State **captures effects of the past**

- independent physical quantities that determines future evolution (absent external excitation)

Inputs **describe external excitation**

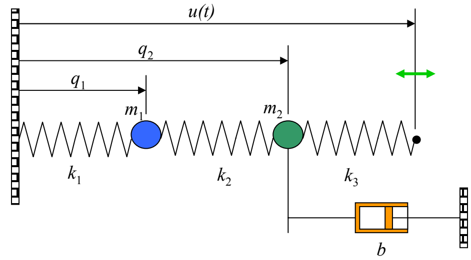
- Inputs are *extrinsic* to the system dynamics (externally specified)

Dynamics **describes state evolution**

- update rule for system state
- function of current state and any external inputs

Outputs **describe measured quantities**

- Outputs are function of state and inputs ⇒ not independent variables
- Outputs are often *subset* of state



Example: spring mass system

- State: position and velocities of each mass: $q_1, q_2, \dot{q}_1, \dot{q}_2$
- Input: position of spring at right end of chain: $u(t)$
- Dynamics: basic mechanics
- Output: measured positions of the masses: q_1, q_2

Modeling Properties

Choice of state is not unique

- There may be *many* choices of variables that can act as the state
- Trivial example: different choices of units (scaling factor)
- Less trivial example: sums and differences of the mass positions (HW 2.4)

Choice of inputs and outputs depends on point of view

- Inputs: what factors are *external* to the model that you are building
  - Inputs in one model might be outputs of another model (eg, the output of a cruise controller provides the input to the vehicle model)
- Outputs: what physical variables (often states) can you *measure*
  - Choice of outputs depends on what you can sense and what parts of the component model interact with other component models

Can also have different *types* of models

- Ordinary differential equations for rigid body mechanics
- Finite state machines for manufacturing, Internet, information flow
- Partial differential equations for fluid flow, solid mechanics, etc

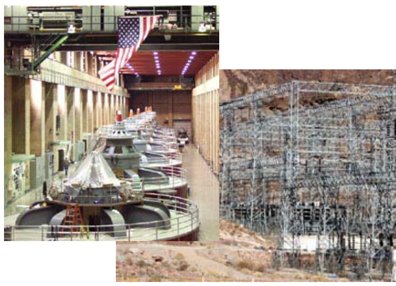
Differential Equations

Differential equations model continuous evolution of state variables

- Describe the rate of change of the state variables
- Both state and time are continuous variables

$$\frac{dx}{dt} = f(x,u)$$
$$y = h(x)$$

Example: electrical power grid



Swing equations

$$\ddot{\delta}_1 + D_1\dot{\delta}_1 = \omega_0\left(P_1 - B\sin(\delta_1 - \delta_2) + G\cos(\delta_1 - \delta_2)\right)$$
$$\ddot{\delta}_2 + D_1\dot{\delta}_2 = \omega_0\left(P_2 - B\sin(\delta_1 - \delta_2) + G\cos(\delta_1 - \delta_2)\right)$$

- Describe how generator rotor angles (δ_i) interact through the transmission line (G, B)
- Stability of these equations determines how loads on the grid are accommodated

State:	rotor angles, velocities ($\delta_i, \dot{\delta}_i$)
Inputs:	power loading on the grid (P_i)
Outputs:	voltage levels and frequency (based on rotor speed)

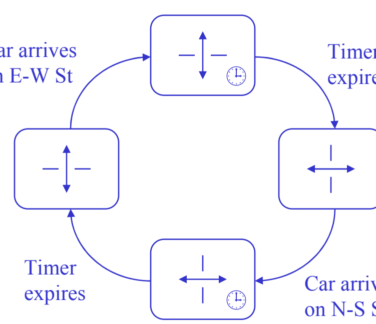
6 Oct 03 R. M. Murray, Caltech CDS 9

Finite State Machines

Finite state machines model discrete transitions between finite # of states

- Represent each configuration of system as a state
- Model transition between states using a graph
- Inputs force transition between states

Example: Traffic light logic



State:	current pattern of lights that are on + internal timers
Inputs:	presence of car at intersections
Outputs:	current pattern of lights that are on

6 Oct 03 R. M. Murray, Caltech CDS 10

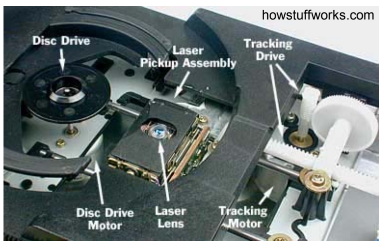
Difference Equations

Difference eqs model discrete transitions between continuous variables

- "Discrete time" description (clocked transitions)
- New state is function of current state + inputs
- State is represented as a *continuous* variable

$$x_{k+1} = f(x_k, u_k)$$
$$y_{k+1} = h(x_{k+1})$$

Example: CD read/write head *controller* (implemented on DSP)



Controller operation (every 1/44,100 sec)

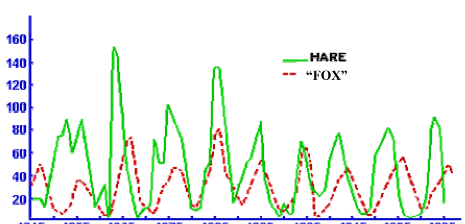
- Get analog signal from read head
- Determine the data (1/0) plus estimate the location of the track center
- Update estimate of "wobble"
- Compute where to position disk head for next read (limited by motor torque)

State: estimated center, wobble
Inputs: read head signal
Outputs: commanded motion

Performance specification

- Keep disk head on track center
- Reject disturbances due to disk shape, shaking and bumps, etc

Example #2: Predator Prey



http://www.math.duke.edu/education/ccp/materials/diffeq/predprey/contents.html

Questions we want to answer

- Given the current population of rabbits and foxes, what will it be next year?
- If we hunt down lots of foxes in a given year, what will the effect on the rabbit and fox population be?
- How do long term changes in the amount of rabbit food available affect the populations?

Modeling assumptions

- The predator species is totally dependent on the prey species as its only food supply.
- The prey species has an external food supply and no threat to its growth other than the specific predator.

Example #2: Predator Prey (2/2)

Discrete Lotka-Volterra model

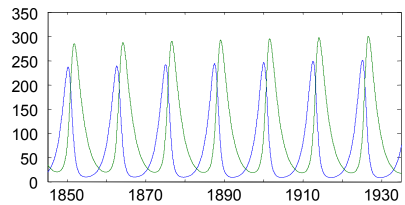
- State
  - R_k # of rabbits in period k
  - F_k # of foxes in period k
- Inputs (optional)
  - u_k amount of rabbit food
- Outputs: # of rabbits and foxes
- Dynamics: Lotka-Volterra eqs

$$R_{k+1} = R_k + b_r(u)R_k - aF_kR_k$$
$$F_{k+1} = F_k - d_fF_k + aF_kR_k$$

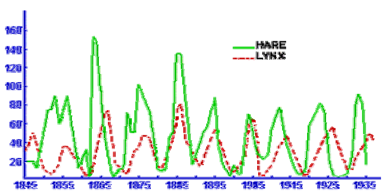
- Parameters/functions
  - $b_r(u)$ rabbit birth rate (per year) (depends on food supply)
  - d_r fox death rate (per year)
  - a interaction term

Matlab simulation (see handout)

- Discrete time model, "simulated" through repeated addition



Comparison with data



6 Oct 03 R. M. Murray, Caltech CDS 13

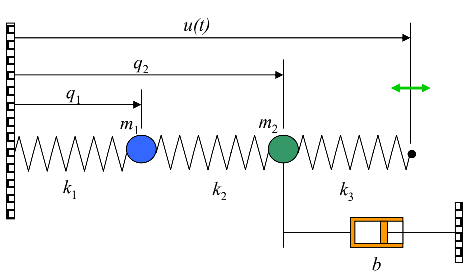
Summary: System Modeling

Model = state, inputs, outputs, dynamics


$$\frac{dx}{dt} = f(x,u)$$
$$y = h(x)$$

$$x_{k+1} = f(x_k, u_k)$$
$$y_{k+1} = h(x_{k+1})$$

***Principle*: Choice of model depends on the questions you want to answer**




```
function dydt = f(t,y, k1, k2,
k3,   m1, m2, b, omega)
u = 0.00315*cos(omega*t);
dydt = [
  y(3);
  y(4);
  -(k1+k2)/m1*y(1) +
      k2/m1*y(2);
  k2/m2*y(1) - (k2+k3)/m2*y(2)
      - b/m2*y(4) + k3/m2*u ];
```

6 Oct 03 R. M. Murray, Caltech CDS 14

```
% L2_1_modeling.m - Lecture 2.1 MATLAB calculations
% RMM, 6 Oct 03

%
% Spring mass system
%

% Spring mass system parameters
m = 250; m1=m; m2=m;                    % masses (all equal)
k = 50; k1=k; k2=k; k3=k;               % spring constants
b = 10;                                          % damping
A = 0.00315; omega = 0.75;              % forcing function

% Call ode45 routine (MATLAB 6 format; help ode45 for details)
tspan=[0 500];                                   % time range for
simulation
y0 = [0; 0; 0; 0];                      % initial conditions
[t,y] = ode45(@springmass, tspan, y0, [], k1, k2, k3, m1, m2, b, A, omega);

% Plot the input and outputs over entire period
figure(1); plot(t, A*cos(omega*t), t, y(:,1), t, y(:,2));

% Now plot the data for the final 10% (assuming this is long enough...)
endlen = round(length(t)/10);           % last 10% of data record
range = [length(t)-endlen:length(t)]';   % create vector of indices (note ')
tend = t(range);
figure(2); plot(tend, A*cos(omega*tend), tend, y(range,1), tend, y(range,2));

% Compute the relative phase and amplitude of the signals
%
% We make use of the fact that we have a sinusoid in steady state,
% as well as its derivative.  This allows us to compute the magnitude
% of the sinusoid using simple trigonometry ( sin^2 + cos^2 = 1).

u = A*cos(omega*tend); udot = -A*omega*sin(omega*tend);
ampu = mean( sqrt((u .* u) + (udot/omega .* udot/omega)) );
fprintf(1, 'Amplitude = %0.5e cm', ampu*100);

%
% Predator prey system
%

% Set up the initial state
R(1) = 20; F(1) = 35;

% For simplicity, keep track of the year as well
year(1) = 1845;

% Set up parameters
br = 0.7; df = 0.5; a = 0.007;
nperiods = 208;
duration=90;

% Iterate the model
for k = 1:duration*nperiods
  b = br;                                        % constant food supply
% b = br*(1+0.5*sin(2*pi*k/(4*nperiods)));       % varying food supply
(try it!)
  R(k+1) = R(k) + (b*R(k) - a*F(k)*R(k))/nperiods;
  F(k+1) = F(k) + (a*F(k)*R(k) - df*F(k))/nperiods;
  year(k+1) = year(k) + 1/nperiods;
end;

% Plot the populations of rabbits and foxes versus time
figure(3); plot(year, R, year, F);
```

```
% springmass.m - ODE45 function for a spring mass system
% RMM, 6 Oct 03
%
% This file contains the differential equation that describes
% the mass spring system used as an example in CDS 101.  It
% allows individual mass and spring values, plus sinusoidal
% forcing.
%
% The state is stored in the vector y.  The values for y are
%
%    y(1) = q1, position of first mass
%    y(2) = q2, position of second mass
%    y(3) = q1dot, velocity of first mass
%    y(4) = q2dot, velocity of second mass

function dydt = springmass(t, y, k1, k2, k3, m1, m2, b, A, omega)

% compute the input to drive the system
u = A*cos(omega*t);

% compute the time derivative of the state vector
dydt = [
  y(3);
  y(4);
  -(k1+k2)/m1*y(1) + k2/m1*y(2);
  k2/m2*y(1) - (k2+k3)/m2*y(2) - b/m2*y(4) + k3/m2*u
];
```