CDS 101/110a: Lecture 1.2
System Modeling

Richard M. Murray
1 October 2008

Goals:
- Define a “model” and its use in answering questions about a system
- Introduce the concepts of state, dynamics, inputs and outputs
- Review modeling using ordinary differential equations (ODEs)

Reading:
- Åström and Murray, *Feedback Systems,* Sections 2.1–2.3, 3.1 [40 min]
- Advanced: Lewis, *A Mathematical Approach to Classical Control*, Chapter 1

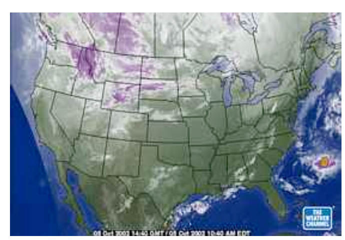
Model-Based Analysis of Feedback Systems

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

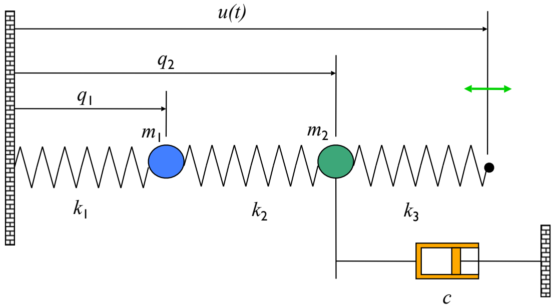
Weather Forecasting



- **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

Example #1: Spring Mass System



Applications

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

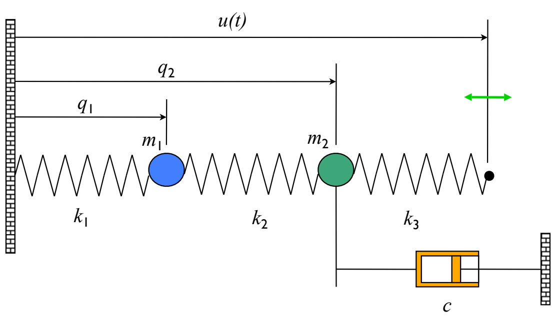
Questions we want to answer

- 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 speed bump 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

Modeling a Spring Mass System



Model: rigid body physics (Ph 1)

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

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

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:

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

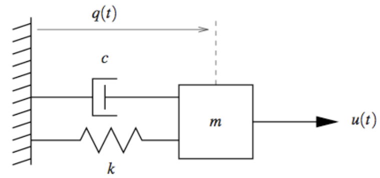
$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \frac{k_2}{m}(q_2 - q_1) - \frac{k_1}{m} q_1 \\ \frac{k_3}{m}(u - q_2) - \frac{k_2}{m}(q_2 - q_1) - \frac{c}{m}\dot{q} \end{bmatrix}$$

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

"State space form"

Review: Second Order Differential Equations

$$m\ddot{q} + c\dot{q} + kq = u$$



Particular response: zero initial conditions

- $q(0) = 0,\ \dot{q}(0) = 0$
- Response to constant (step) input, *u*(*t*) = *F*

$$q(t) = \frac{F}{m\omega_0^2}\left(1 - e^{-\zeta\omega_0 t}\cos\omega_d t + \frac{\zeta}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_0 t}\sin\omega_d t\right)$$

- Response to sinusoidal input, *u*(*t*) = *A* sin ω *t*

$$q(t) = MA\sin(\omega t + \theta) - MA\sin\theta, \qquad Me^{i\theta} = \frac{\omega_0^2}{\omega_0^2 - \omega^2 + 2i\zeta\omega_0\omega}$$

- Form of the solution: sinusoid at same frequency, with shift in mag & phase
- Solving by hand is a mess; we will learn much better ways later

Complete solution: homogeneous + particular

- Warning: be careful to make sure the initial conditions are satisfied

More General Forms of Differential Equations

State space form

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

$$y = h(x,u) \qquad y = Cx + Du \qquad y \in \mathbb{R}^q$$

General form — Linear system

- x = state; *n*th order
- u = input; will usually set p = 1
- y = output; will usually set q = 1

Higher order, linear ODE

$$\frac{d^nq}{dt^n} + a_1\frac{d^{n-1}q}{dt^{n-1}} + \cdots + a_n q = u$$

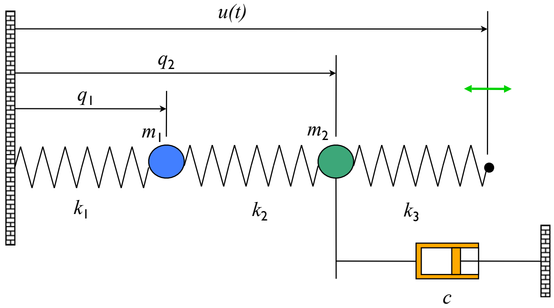
$$y = b_1\frac{d^{n-1}q}{dt^{n-1}} + \cdots + b_{n-1}\dot{q} + b_n q$$

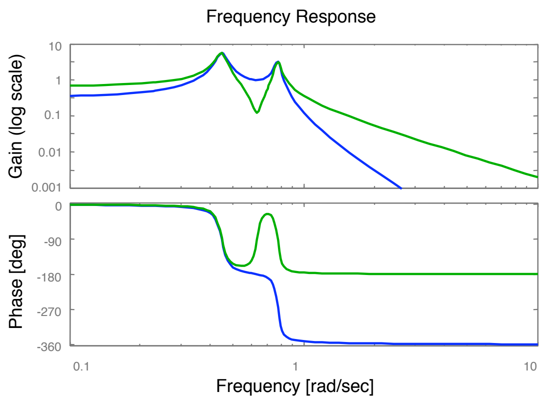
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d^{n-1}q/dt^{n-1} \\ d^{n-2}q/dt^{n-2} \\ \vdots \\ dq/dt \\ q \end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} x$$

Simulation of 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)
         - c/m2*y(4) + k3/m2*u ];

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


Modeling Terminology

***State* captures effects of the past**

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

***Inputs* describe external excitation**

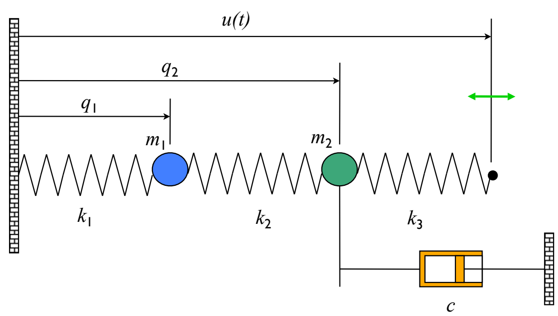
- Inputs are *extrinsic* to the system dynamics (externally specified)

***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

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

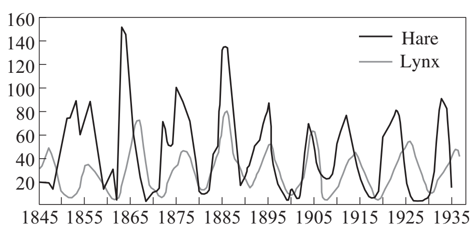
Difference Equations

Difference equations 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

$$x[k+1] = f(x[k], u[k])$$
$$y[k] = h(x[k])$$

Example: predator prey dynamics



Questions we want to answer

- Given the current population of hares and lynxes, what will it be next year?
- If we hunt down lots of lynx in a given year, how will the populations be affected?
- How do long term changes in the amount of food available affect the populations?

Modeling assumptions

- Track population annual (discrete time)
- 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 Modeling

Discrete Lotka-Volterra model

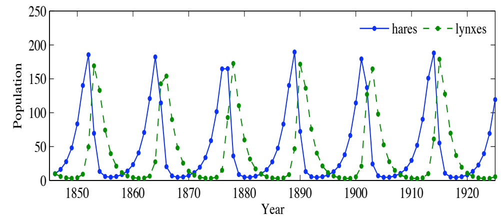
- State
  - $H[k]$ # of rabbits in period k
  - $L[k]$ # of foxes in period k
- Inputs (optional)
  - $u[k]$ amount of rabbit food
- Outputs: # of rabbits and foxes
- Dynamics: Lotka-Volterra eqs

$$H[k+1] = H[k] + b_r(u)H[k] - aL[k]H[k],$$
$$L[k+1] = L[k] + cL[k]H[k] - d_f L[k],$$

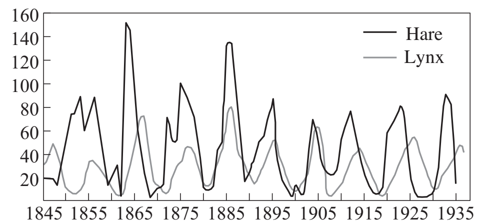
- Parameters/functions
  - $b_r(u)$ hare birth rate (per period); depends on food supply
  - d_f lynx mortality rate (per period)
  - a, c interaction terms

MATLAB simulation (see handout)

- Discrete time model, "simulated" through repeated addition



Comparison with data



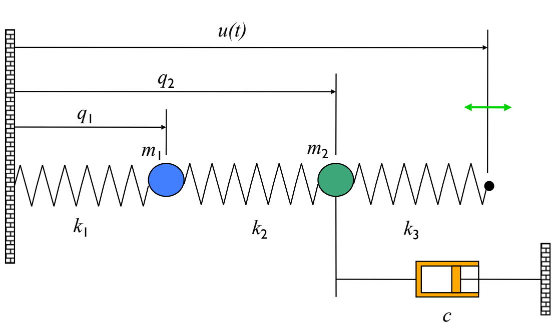
Summary: System Modeling

Model = state, inputs, outputs, dynamics

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

$$x[k+1] = f(x[k], u[k])$$
$$y[k] = h(x[k])$$

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




```
function dydt = f(t,y, k1, k2,
k3,   m1, m2, c, 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 ];
```

```
% L1_2_modeling.m - Lecture 1.2 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); clf
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); clf
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
clear H L year
H(1) = 10; L(1) = 10;

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

% Set up parameters (note that c = a in the model below)
br = 0.6; df = 0.7; a = 0.014;
nperiods = 365;                         % simulate each day
duration = 90;                          % number of years for simulation

% 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!)
  H(k+1) = H(k) + (b*H(k) - a*L(k)*H(k))/nperiods;
  L(k+1) = L(k) + (a*L(k)*H(k) - df*L(k))/nperiods;
  year(k+1) = year(k) + 1/nperiods;

  if (mod(k, nperiods) == 1)
    % Store the annual population
    Ha((k-1)/nperiods + 1) = H(k);
    La((k-1)/nperiods + 1) = L(k);
  end;
```



```
end;

% Store the final population
Ha(duration) = H(duration*nperiods+1);
La(duration) = L(duration*nperiods+1);

% Plot the populations of rabbits and foxes versus time
figure(3); clf;
plot(1845 + [1:duration], Ha, '.-', 1845 + [1:duration], La, '.--');

% Adjust the parameters of the plot
axis([1845 1925 0 250]);
xlabel('Year');
ylabel('Population');

% Now reset the parameters to look like we want
lgh = legend(gca, 'hares', 'lynxes', 'Location', 'NorthEast', ...
  'Orientation', 'Horizontal');
legend(lgh, 'boxoff');


--------------------------------------------------------



% 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
];
```