# **Computational Methods** (PHYS 2030)

Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

Schedule: Lecture: MWF 11:30-12:30 (CLH M)

Website: http://www.yorku.ca/cberge/2030W2018.html

York University
Winter 2018
Lecture 32

# PHYS 2030: Course philosophy

"What I cannot create, I do not understand."

"The purpose of computing is insight, not numbers."

- R.W. Hamming (Numerical methods for scientists and engineers, 1962)


Barbie
i can be...
the remix!
now with less sexism!
A Computer Engineer
public class HelloWorld {
public static void main
(String[] args) {
System.out.println
("Hello, World");
}
remixed by
Casey Fiesler
www.caseyfiesler.com
This non-commercial
transformative work
constitutes a fair use of
any copyrighted material,
as provided for in Section
107 of the U.S. Copyright Act.


"Your robot puppy is so sweet," says Skipper. "Can I play your game?"

"I'm only creating the design ideas," Barbie says, laughing. "I'll need Steven's and Brian's help to turn it into a real game!"

"Hey, your game has **robot** puppies!" says Skipper, looking over her shoulder at the art. "Robots are **sweet**! Can I play it?"

"Well, it's not quite done yet," Barbie says, smiling. "Really good games are made by a team of people. I'm doing some of the coding now, but Stephen and Brian are helping, too. There are lots of pieces to making a game, like art and music and storyline. Brian drew that puppy. You're a good artist, Skipper. Maybe you could be a graphic designer when you grow up."

Skipper grins. "I love art, but I really love science, too. Physics is my favorite class. I think I want to be a physicist."

http://www.slate.com/articles/technology/future_tense/2014/11/barbie_is_a_computer_engineer_the_terrible_book_gets_remixed.html

# PHYS 2030 Overview: Course themes

Note: This is the list shown on Lec.1

1. **Programming/computing skills**
    - o knowledge of syntax & capabilities
    - o comfortability
    - o debugging (pay attention to error messages)
    - o style: modular versus comprehensive programming
    - o finding online resources (incl. scientific journals)
2. **Numerical methods**
    - o solving differential equations —> harmonic oscillator
    - o eigenvalue problems —> Principle component analysis
    - o Monte Carlo
    - o regression
    - o complex numbers —> fractals
    - o Fourier transforms
    - o bifurcation analysis (period doubling cascade and chaos)
3. **Data acquisition/analysis**
    - o measuring data
    - o visualizing data
    - o analyzing data & signal processing
    - o statistics —> Anscombe's data
    - o compressed sensing

# 2030 W18 Topics

- Harmonic oscillator, complex #s
- Numerical integration, differentiation
- ODEs: slope fields, phase plane
- ODEs: numerical methods (e.g., Euler, RK4, ode45)
- ODEs: systems of eqns. and 2nd order eqns.
- ODEs: linear vs nonlinear
- ODEs: linear methods
- Regression: notion of least-squares
- Regression: data extraction via deplot.m
- Regression: linear vs nonlinear vs non-parametric
- Nonlinear systems: fractals

Note: A key idea here was the notion of a 'series expansion' representation (e.g., Taylor)

Note: A key idea here was the notion of a 'series expansion' representation (e.g., Taylor)

Note: A key idea here was the notion of a 'series expansion' representation (e.g., Taylor)

→ You get the idea....

2030 W18 Topics (cont.)

- Nonlinear systems: bifurcations, period doubling, & chaos
- Basic elements of digital signal processing (e.g., sampling)
- Fourier analysis: basics
- Fourier analysis: DFT and signal processing
- Convolutions (incl. image processing)
- Monte carlo methods
- DAQ: (more) basics on digital signal processing
- DAQ: data analysis, visualization

‘Golden rules’ of (2030) computing

1. **The computer only does what you tell it to do**

2. Think ahead when writing code: **try to minimize future work**
   (e.g., even remotely archaic code now will certainly equal headache later on)

3. Stay organized. Find a **style** that works for you

4. Comment. Comment. **Comment**. [clearly and efficiently]

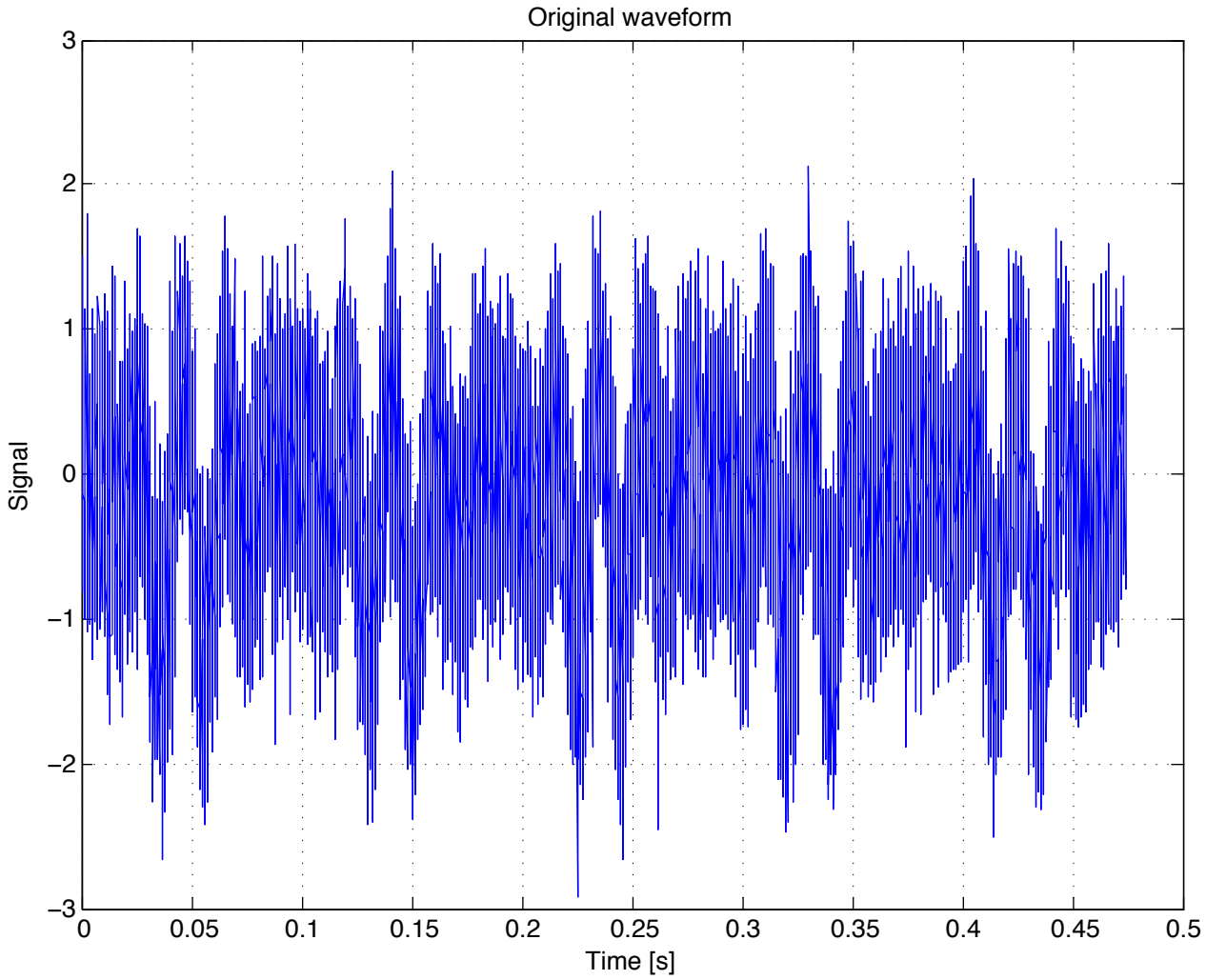
Other tips:

- 'Preserve all 'versions' of workable code.'
  [Caveat: Don't modify someone else's code! Copy and make your own version.]

- When using external routines (i.e., code you didn't write yourself), keep the spectre of the 'blackbox' in mind at all time.

Ex.

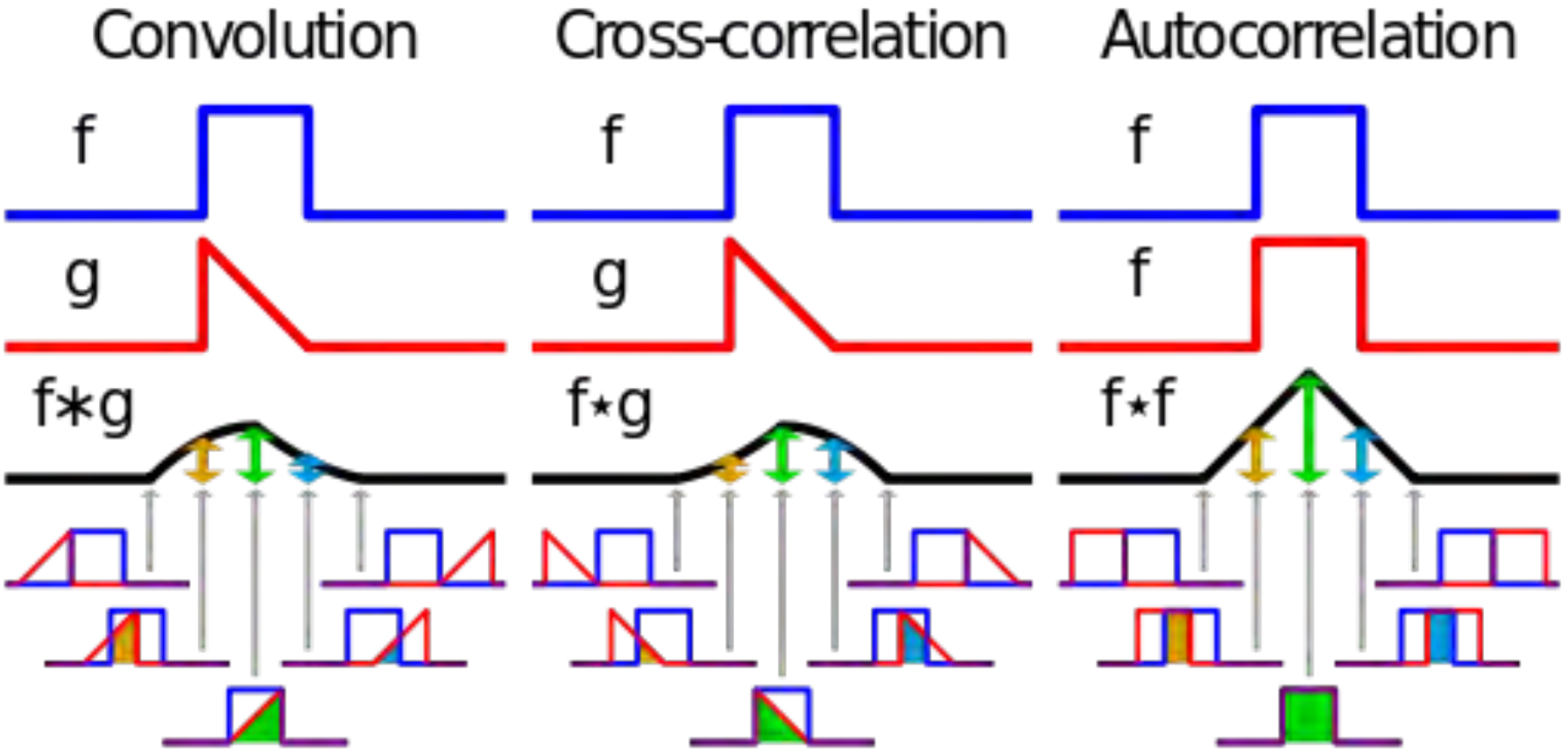
- Let's say you have a noisy signal that you think repeats itself. How would you quantitatively characterize such?




- Visibly, is there a periodicity present?
- How would you quantitatively assess such?
- Fourier transform?

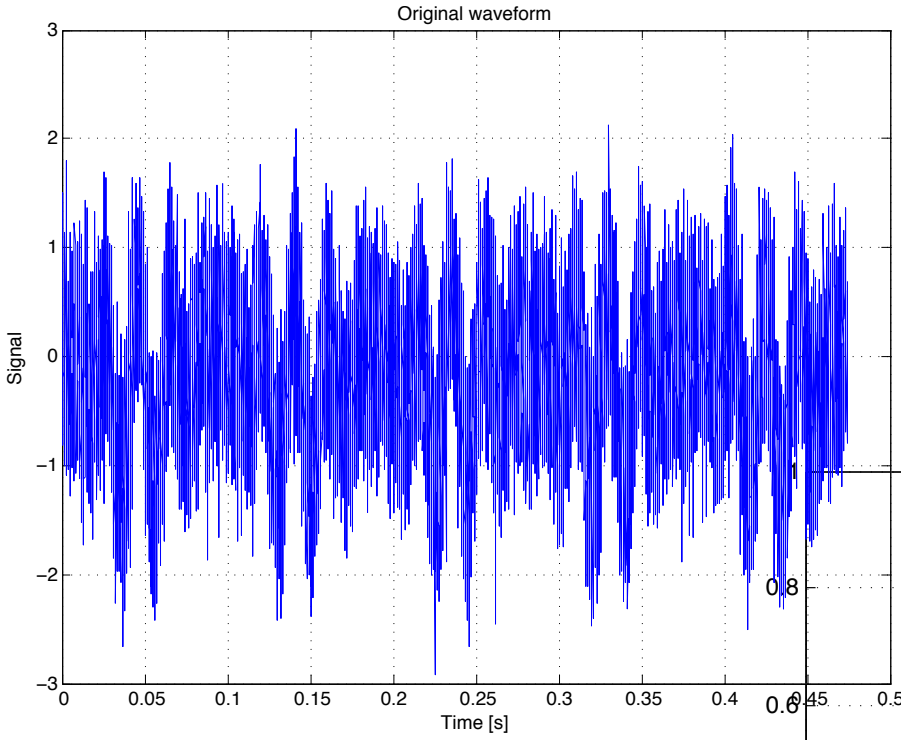
→ Autocorrelation

Ex. Autocorrelation

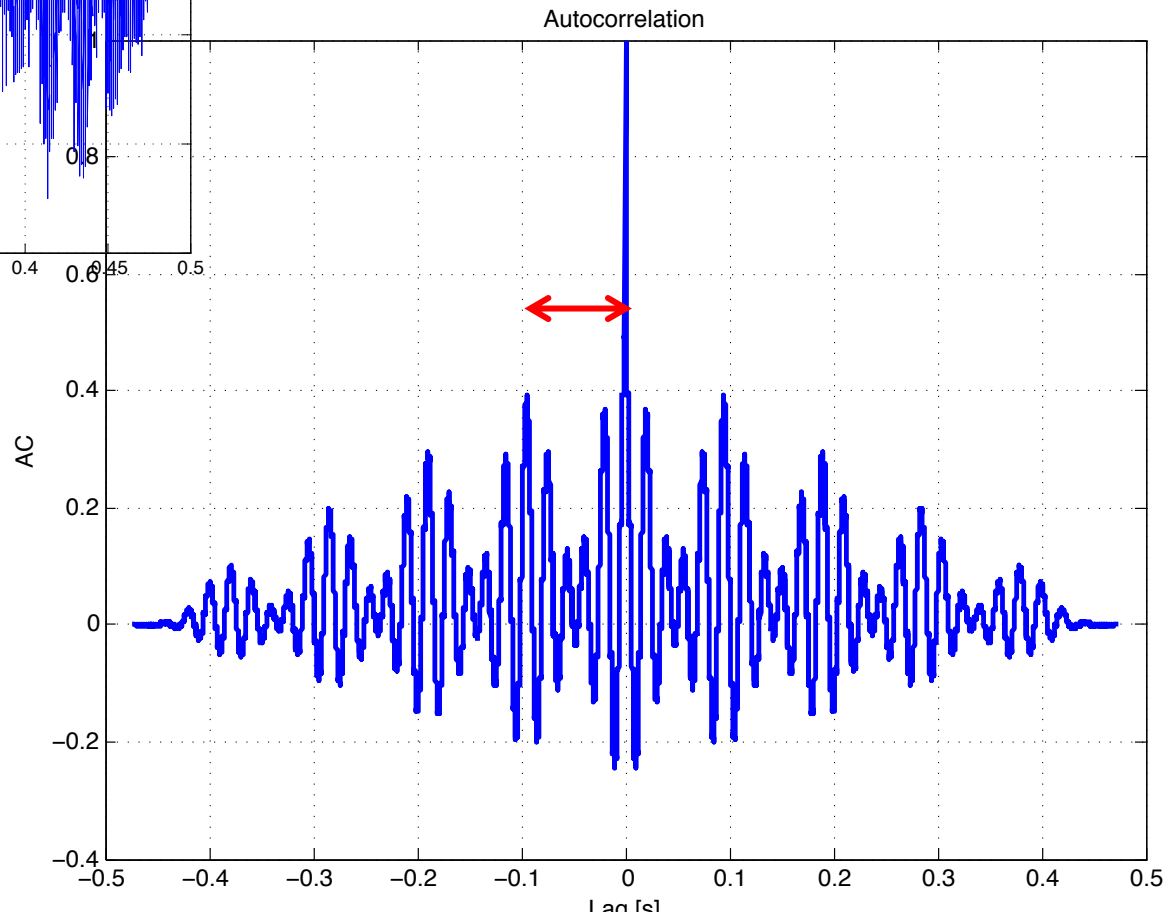



Wikipedia (cross-correlation)

```
% ### EXautocorr4.m ###     12.02.14
% Demonstrates how to use autocorrelation to ascertain 'how long does it
% take for a noisy signal to repeat itself?'
clear
% ---
maxN= 8192;     % largest possible buffer length
SR= 44100;      % sample rate [Hz]
scaleN= 0.5;    % scale factor for noise
R= 5;          % number of repeats
% ------------------------------------------------
N= ceil(rand(1)*maxN);
x= linspace(0,N,N); % determine range
t= x/SR;        % convert to a time
polyS= ceil(10*rand(1));   % randomly determine polynomial order
Pv= 2*rand(polyS,1)-1;  % polynomial coefficients
freqR= 100*(1+scaleN*randn(1));     % randomize up a bit the sin freq. (but not from run to run)
dt= 1/SR;  % spacing of time steps
nPts= N*R;      % total # of points
freq= SR*(0:nPts/2)./nPts;    % create a freq. array (for FFT bin labeling)
% ---
% kludge way to build up signal; tamps down at edges (then adds noise)
y= []; time= [];
for nn=1:R
    temp= hanning(N)'.*(polyval(Pv,t)+sin(freqR*2*pi*t))+scaleN*randn(N,1)';
    y= [y temp]; % pseudo-random 'data'

end
% ---
% autocorrelation (via Matlab's cross-correlation routine or a home-built function;
% the two yhield the same answer, but Matlab's function is faster)
if (1==1),  hh= xcorr(y,y);
else hh= correlate1(y,y); end
thh= linspace(-N*R/SR,N*R/SR,numel(hh));
hh= hh/max(hh(:));  % normalize to unity
% +++
% plot time waveform
figure(1); clf; plot(linspace(0,numel(y)/SR,numel(y)),y);  hold on; grid on;
xlabel('Time [s]'); ylabel('Signal'); title('Original waveform');
% +++
% plot autocorrelation
figure(2); clf; plot(thh,hh); hold on; grid on; xlabel('Lag [s]'); ylabel('AC'); title('Autocorrelation');
% ---
% now we need some means to extracts the spacing between the relevant maxima...
```




```
maxN= 8192;      % largest possible buffer length
SR= 44100;       % sample rate [Hz]
scaleN= 0.5;     % scale factor for noise
R= 5;            % number of repeats
```



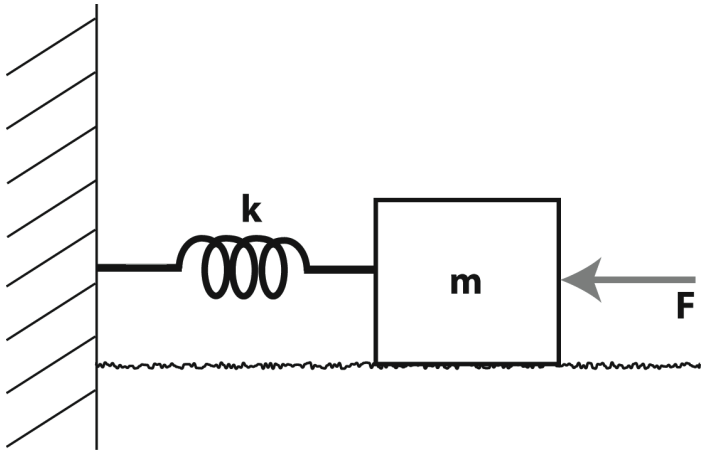

- Now what?

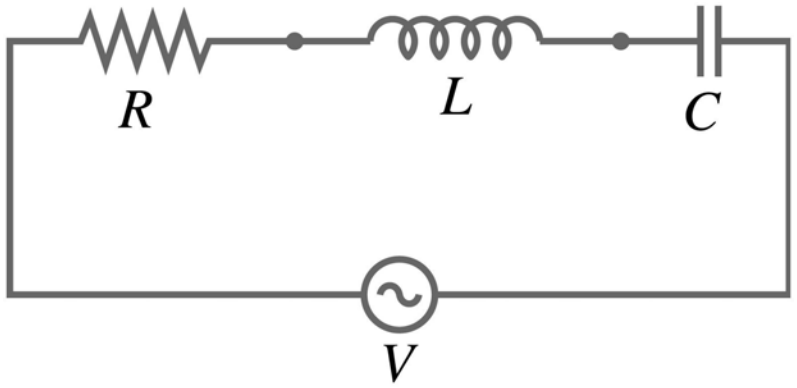
→ Spectral methods to extract the relevant periodicity

Note: There is a deep connection between the autocorrelation and the Fourier transform

Ex. RLC circuit = Damped, Driven Harmonic Oscillator

| Mechanical | | Electrical | |
|---|---|---|---|
| $F$ (force) | ←→ | $V$ (potential) | state variables |
| $v$ (velocity) | ←→ | $I$ (current) | state variables |
| $x$ (position) | ←→ | $q$ (charge) | state variables |
| $m$ (mass) | ←→ | $L$ (inductance) | system properties |
| $b$ (damping) | ←→ | $R$ (resistance) | system properties |
| $k$ (spring) | ←→ | $1/C$ (capacitance) | system properties |



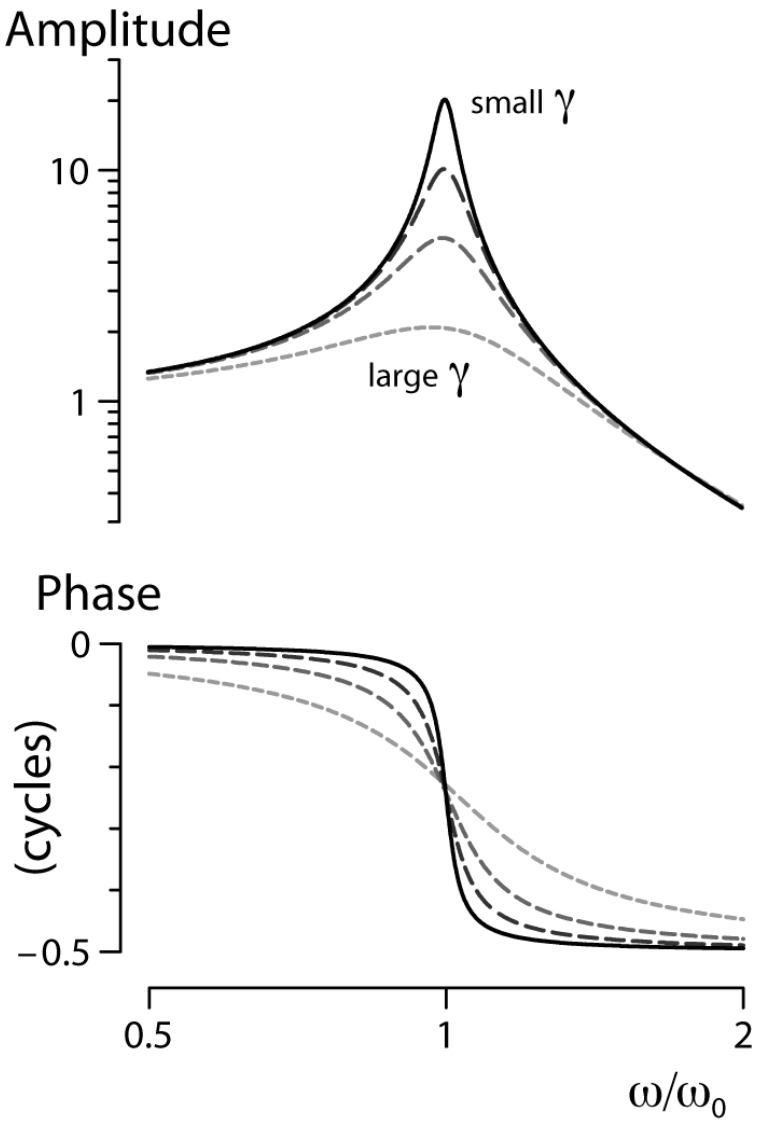
# Theoretical considerations

More realistic case: Damped, driven

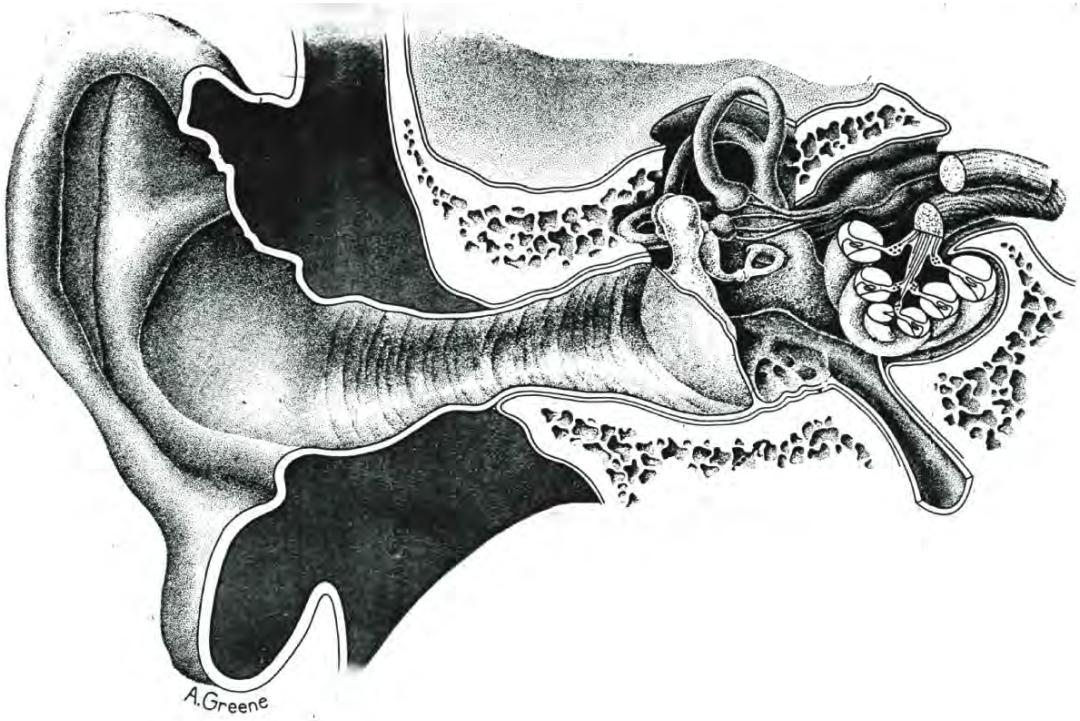
$$A(\omega) = \frac{F_o/m}{[(\omega_o^2 - \omega^2)^2 + (\gamma\omega)^2]^{1/2}}$$

$$\delta(\omega) = \arctan\left(\frac{\gamma\omega}{\omega^2 - \omega_o^2}\right)$$

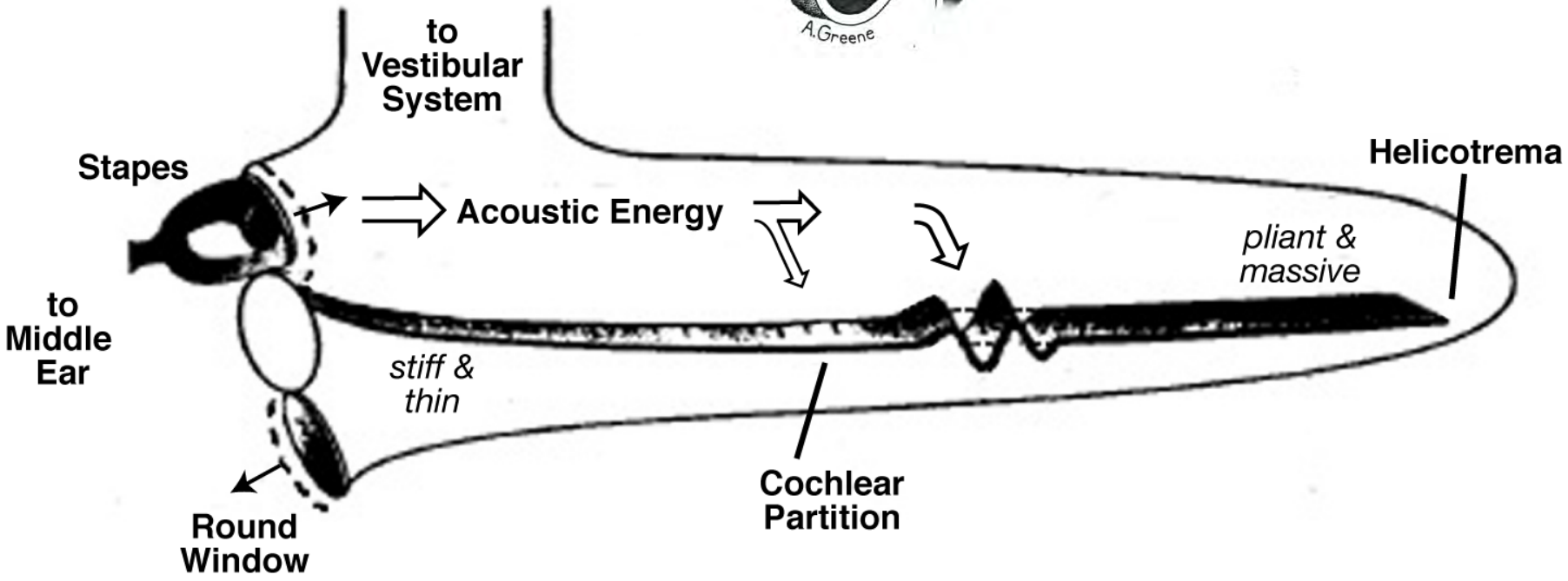
⇒ Second-order oscillator behaves as as *band-pass filter*



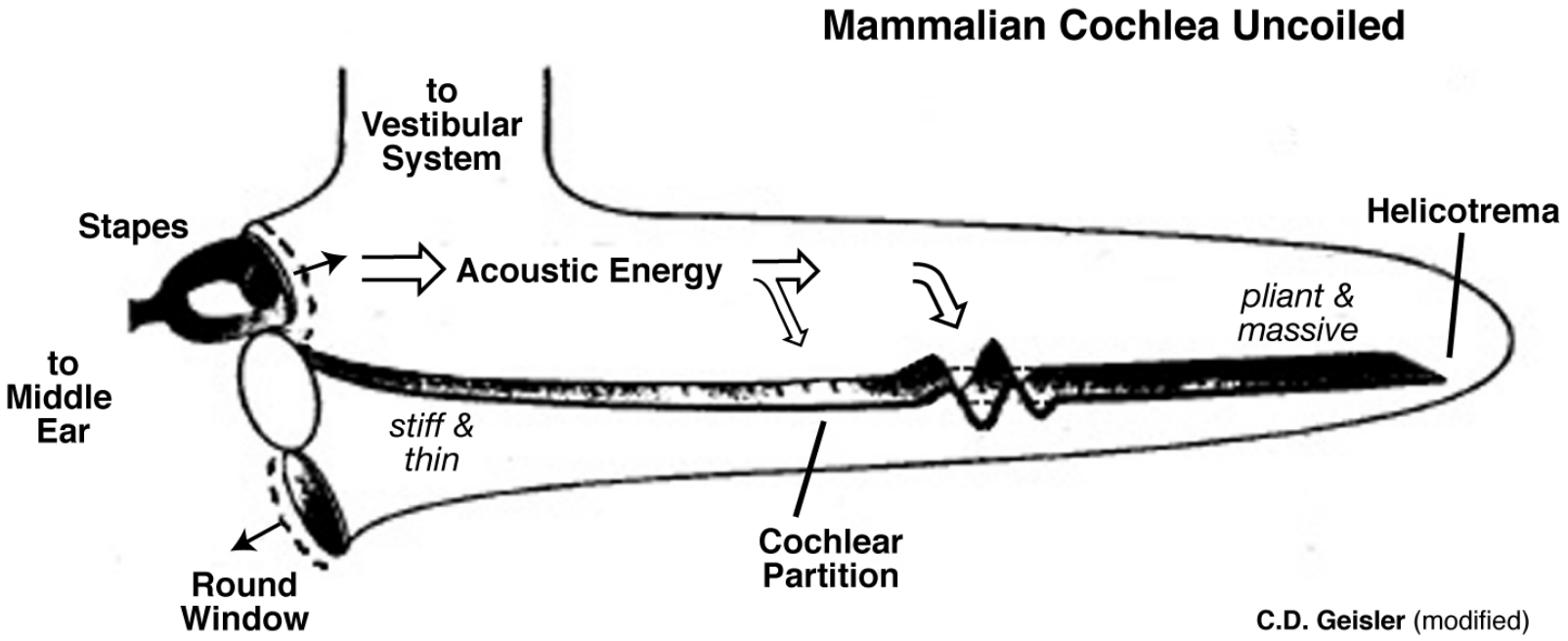
(inner) Ear is a long fluid-filled tube



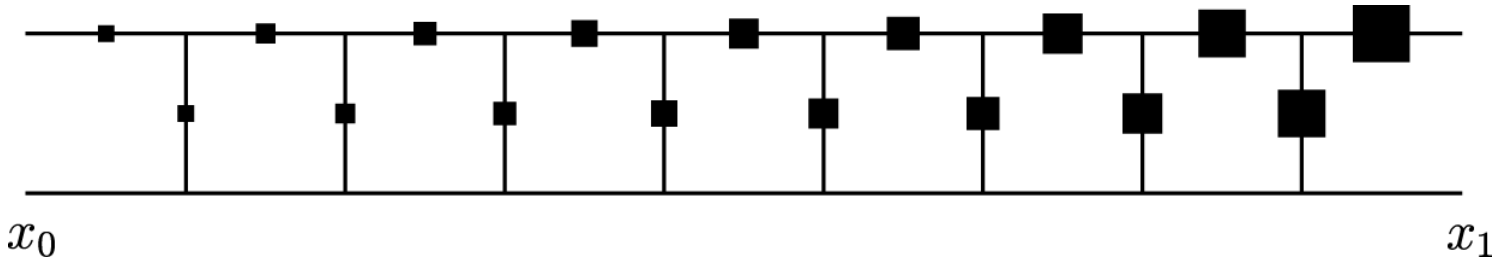

Traveling waves




Geisler (1996)




Model: Non-uniform transmission line

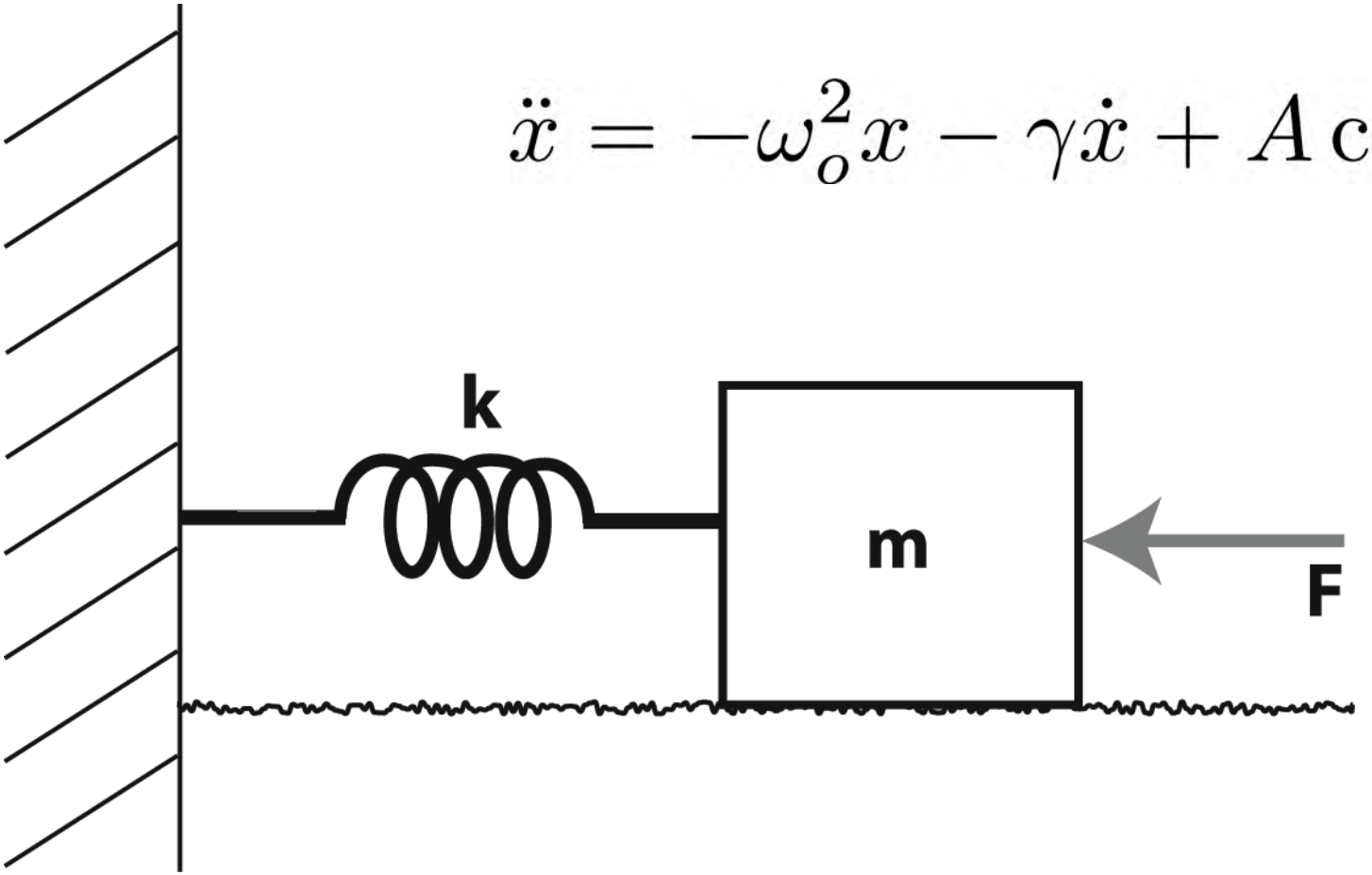


Key Motivation:
Many (biomechanical) things oscillate...

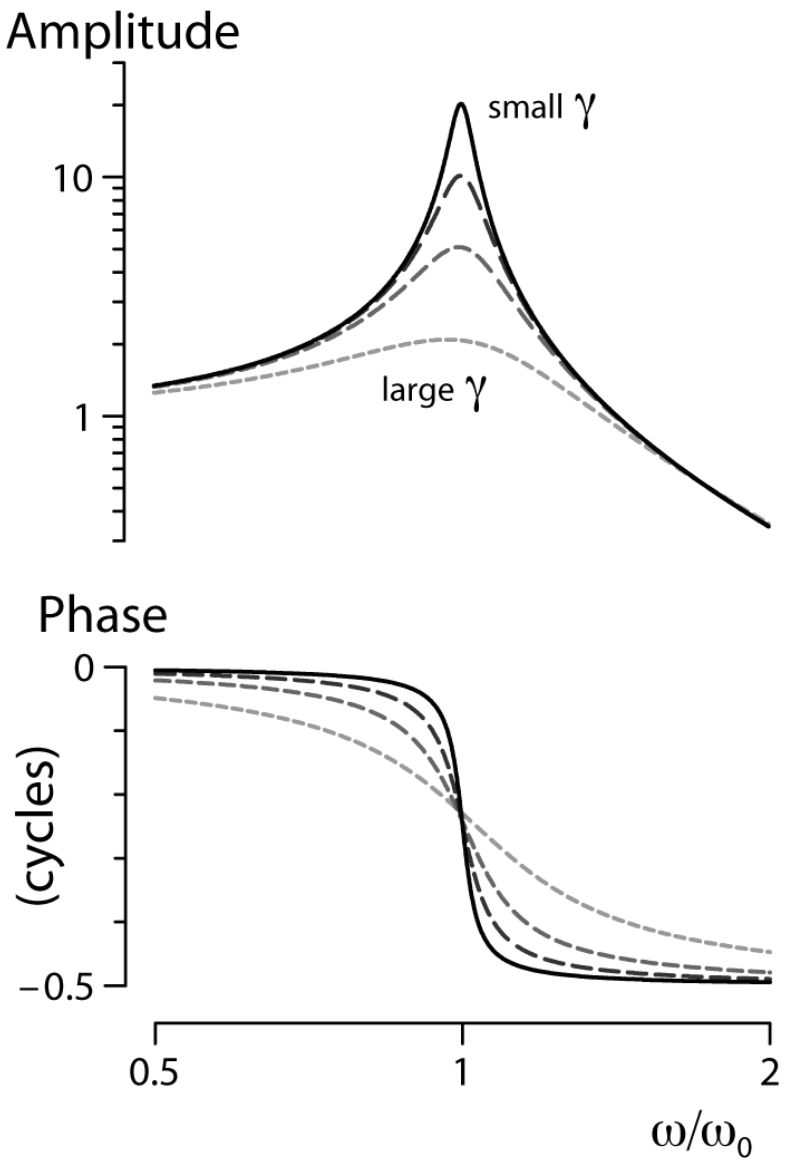
Wikipedia

Foundation: Harmonic Oscillator

$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$



# Foundation: Resonance



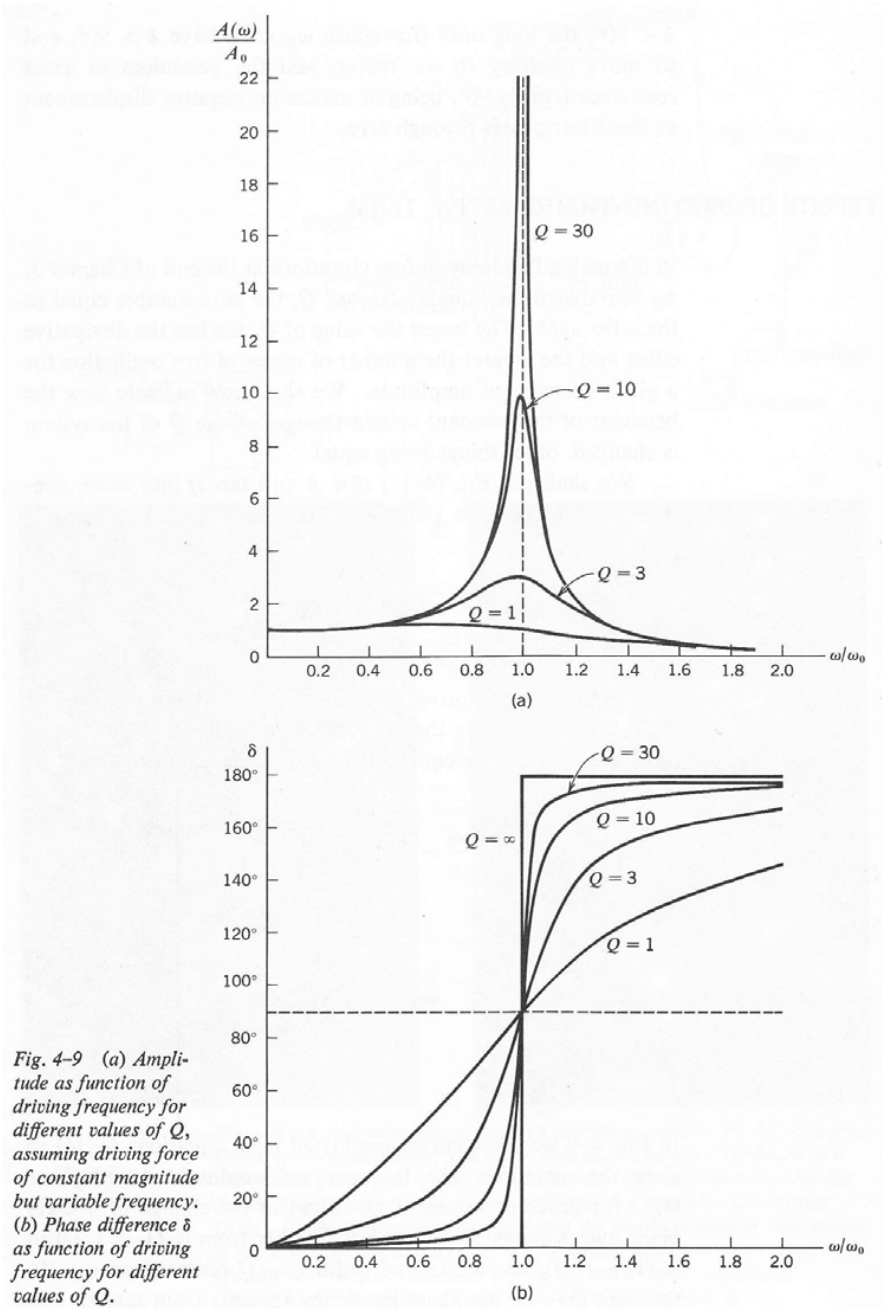




*Fig. 4–9 (a) Amplitude as function of driving frequency for different values of Q, assuming driving force of constant magnitude but variable frequency. (b) Phase difference δ as function of driving frequency for different values of Q.*

French (1971)

# Practical example: MEMS (**M**icro**e**lectro**m**echanical **s**ystems)

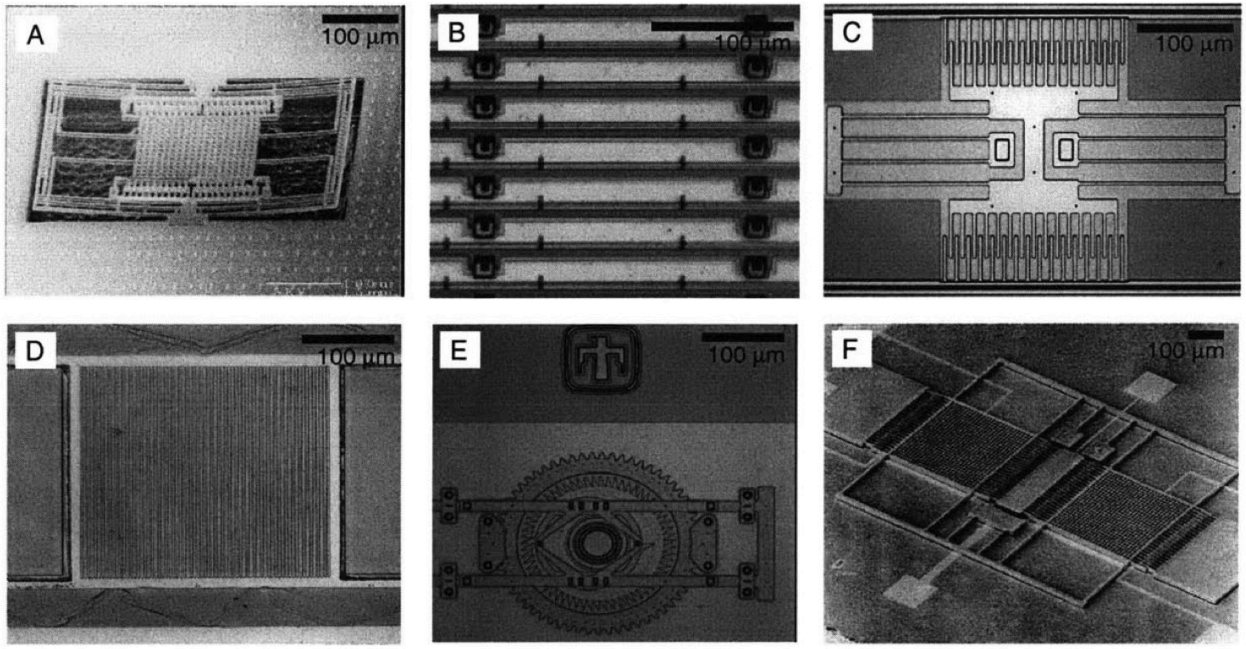



Figure 1-1: Microelecromechanical systems encompassing a wide variety of applications and a broad spectrum of fabrication processes: (A) SEM image of CMOS MEMS (Fedder et al., 1996) multiple degree-of-freedom microresonator fabricated at Carnegie Mellon University, (B) Optical micrograph of surface micromachined polysilicon diffraction gratings fabricated at the State University of New York at Albany, (C) Optical micrograph of surface micromachined lateral resonator fabricated using Cronos MUMPs, (D) Optical micrograph of platinum diffraction gratings fabricated at the Massashusetts Institute of Technology, (E) Optical micrograph of indexing motor fabricated using Sandia SUMMiT4 fabrication process, and (F) SEM image of Draper Laboratory tuning fork gyroscope (SEM picture courtesy Charles Stark Draper Laboratory, Cambridge MA).

$\Rightarrow$ Resonant behavior

http://dspace.mit.edu/handle/1721.1/87312#files-area

Practical example: MEMS (**M**icro**e**lectro**m**echanical **s**ystems)

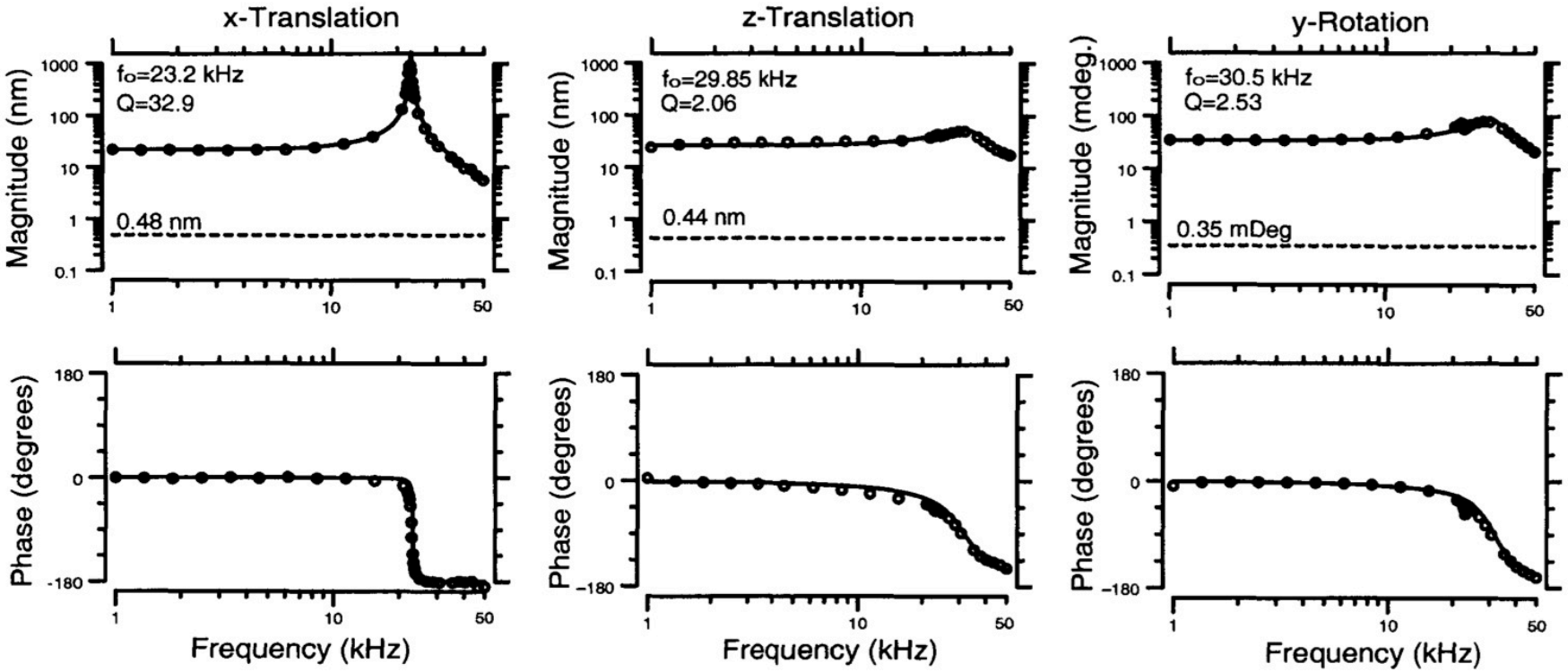
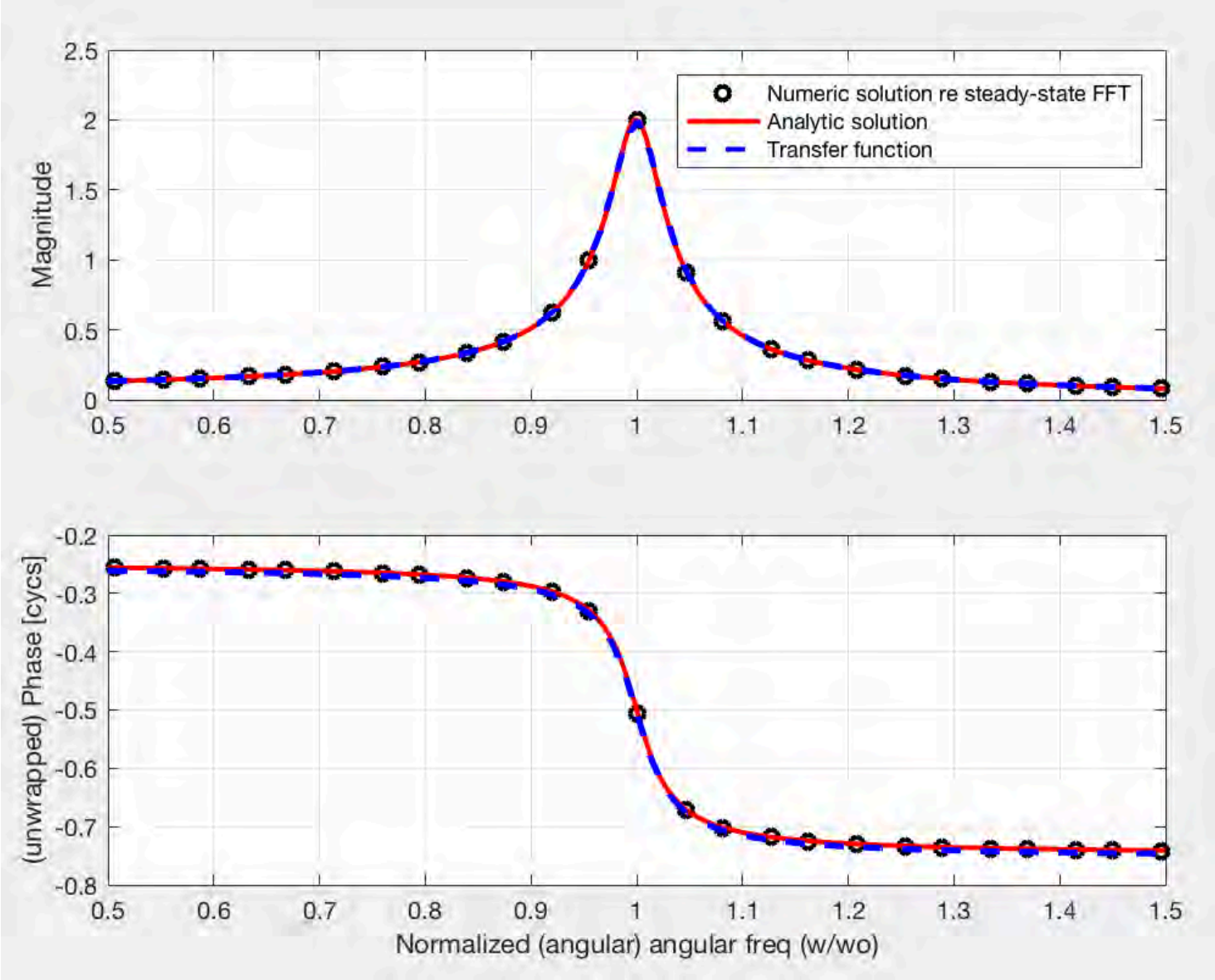



Figure 4-5: Magnitude and phase of frequency response for translation along $x$ (left panel), translation along $z$ (center panel), and rotation about $y$ (right panel)

http://dspace.mit.edu/handle/1721.1/87312#files-area

EXhoResonance.m

# Focal Point (for today)



```
% ### EXhoResonance.m ###     2017.02.04 CB

% Code to solve the damped (sinusoidally-) driven harmonic oscillator (DDHO)
% for a variety of driving freqs. so to buildup the "resonance curve" via
% computation of the mag/phase of the Fourier transform of the steady-state
% response. Furthermore, the analytic solution for the DDHO as well as the
% transfer function are shown to be equivalent (Fig.1)

% Damped driven Harmonic Oscillator (DDHO)
% d^2xdt^2 = -((P.wo)^2)*x - P.gamma*dx/dt + (P.A)*sin(P.w*t)

% Reqs:
% EXhoResonanceFunc.m (re ode45), rfft.m

clear;
% ================================================================
% ---
% Oscillator params. and ICs
P.p0 = 0.0; % Initial position {0}
P.v0 = 0.0; % Initial velocity {0}
P.wo= 10;            % resonant (angular) freq {10}
P.gamma= 0.5;      % damping coefficient {0.5}
% ---
% Sinusoidal driving term params.
P.A = 10; % Driving force amplitude {10}
P.wDrive= [5 15];    % start and end angular drive freqs. {[5 15]}
P.wDriveN= 25;          % # of drive freqs. to run {25}
% ---
P.tmax = 200; % Maximum time to solve [s; arb] {200}
P.SR= 150;     % sample rate for time step [Hz; arb] {150}
P.Npoints= 8192;    % Number of points in time series for FFT, must be 2^n {8192}
% ---
P.plotN= 1;     % boolean re plotting the waveform and spectra for one driving freq.
{1}
P.plotNnum= round(P.wDriveN/2);     % driving freq. index to plot {round(P.wDriveN/2)}
% ---
P.solveType= 1;   % 0-ode45, 1-hard-coded RK4 {1}
P.stepF= 0;     % boolean re using a fixed step-size for ode45 {0}
% ================================================================
```

```
% ================================================================
% ---
dt= 1/P.SR;  % spacing of time steps
init0 = [P.p0 P.v0]'; % Column vector of initial conditions.
tspan = [0:dt:P.tmax]; % time interval for entire computation
tW=[0:1/P.SR:(P.Npoints-1)/P.SR];  % (shorter/later) time interval for FFT window
L = length(tspan); TW = L-(P.Npoints-1);  % create offset point extracting FFT window
% ---
% create relevant freq. arrays (e.g., for FFT bin labeling)
freq= [0:P.Npoints/2];  % Note: these values are not angular (i.e., [freq]= 1/s, not
rads/s)
freq= P.SR*freq./P.Npoints;
df = P.SR/P.Npoints;            % freq. spacing between bins
wDT= linspace(P.wDrive(1),P.wDrive(2),500); % create ang. freq. array for plotting
analytic solution
% +++++++++++++++
% various relevant derived quantities (used post- main for loop)
Q= P.wo/P.gamma;    % "quality factor" (Note: tau=1/P.gamma=Q/P.wo, where tau is time
const. of build-up)
lambdaP= 0.5*(-P.gamma+ sqrt(P.gamma^2-4*P.wo^2));  % Eigenvalues, for x=0 (undriven)
lambdaM= 0.5*(-P.gamma- sqrt(P.gamma^2-4*P.wo^2));
% Note - Can also get eigenvalues via command: eig([0 1;-P.wo^2 -P.gamma])
Z= P.gamma+ i*(wDT- P.wo^2./wDT);   % impedance (see notes above; assumes mass is
unity)
Y= 1./Z;    % admittance (reciprocal of impedance)
% ---
% grabbing driving freqs. from freq array
%%indx= find(freq>=P.fDrive(1) & freq<=P.fDrive(2));    % find relevant indicies
indx= find(freq>=P.wDrive(1)/(2*pi) & freq<=P.wDrive(2)/(2*pi));    % find relevant
indicies
indxB= round(linspace(indx(1),indx(end),P.wDriveN));    % one means to get the desired
subset
freqD= 2*pi*freq(indxB); % array of driving angular freqs
```

EXhoResonance.m (cont)

```
for mm=1:numel(freqD)
    P.w= freqD(mm);     % extract driving freq.
     % *** Solve in one of two ways ***
    if P.solveType==0
        % use Matlab's ode45
        % ---
        % tell it to actually use the specified step-size
        if(P.stepF==1), options = odeset('MaxStep',1/P.SR);  else options=[];   end
        [t,y] = ode45(@EXhoResonanceFunc,tspan,init0,options,P);
    else
        % use 4th order Runge-Kutta code
        xPoints(1) = P.p0;  vPoints(1) = P.v0;   % initialize ICs into dummy arrays
        x= P.p0; v= P.v0;   % kludge
        dt= 1/P.SR;     % time step
        for nn=1:(length(tspan)-1)
            % ---
            t = tspan(nn);   % Current time.
            % step1
            xk1= v;
            vk1= -((P.wo)^2)*x - P.gamma*v + (P.A)*sin(P.w*t);
            % step 2
            xk2 = v + (dt/2)*vk1;
            vk2= -((P.wo)^2)*(x + (dt/2)*xk1) - P.gamma*(v + (dt/2)*vk1)...
                + (P.A)*sin(P.w*(t+(dt/2)));
            % step 3
            xk3 = v + (dt/2)*vk2;
            vk3= -((P.wo)^2)*(x + (dt/2)*xk2) - P.gamma*(v + (dt/2)*vk2)...
                + (P.A)*sin(P.w*(t+(dt/2)));
            % step 4
            xk4 = v + dt*vk3;
            vk4= -((P.wo)^2)*(x + (dt)*xk3) - P.gamma*(v + dt*vk3)...
                + (P.A)*sin(P.w*(t+(dt/2)));
            % apply RK4 weighting
            x = x + (dt/6)*(xk1 + 2*xk2 + 2*xk3 + xk4);
            v = v + (dt/6)*(vk1 + 2*vk2 + 2*vk3 + vk4);
            % store away position and velocity
            xPoints(nn+1) = x; vPoints(nn+1) = v;
        end
        y(:,1)= xPoints'; y(:,2)= vPoints';   % repackage output

    end
```

```
    % ---
    ySPEC= y(TW:TW+P.Npoints-1,1);  % steady-state portion of waveform for FFT
    sigSPEC= rfft(ySPEC);
    % ---
    wDrive(mm)= 2*pi*freq(indxB(mm));   % store away driving freqs.
    mag(mm)= abs(sigSPEC(indxB(mm)));   % store away SS mag.
    % ---
    % need to correct the phase re the duration of the window allowed for settling into steady-state
    tPhase= angle(sigSPEC(indxB(mm)));  % extract the phase
    tPhase= angle(exp(i*(tPhase- wDrive(mm)*tspan(TW))));  % correct phase re onset
    phase(mm)= tPhase;
    % ---
    % visualize relevant bits for one of the drive freqs.
    if mm==P.plotNnum
        % ---
        % integrated waveform and segment extracted for spectral analysis
        figure(2); clf;
        h1= plot(tspan,y(:,1)); hold on; grid on;
        xlabel('Time'); ylabel('Position');
        title('Time Waveform of integrated solution to damped driven HO equation')
        L = length(tspan); TW = L-(P.Npoints-1);  % create offset point
        ySPEC= y(TW:TW+P.Npoints-1,1);  % steady-state portion of waveform for FFT
        h2= plot(tspan(TW:TW+P.Npoints-1),ySPEC,'r.','MarkerSize',3);
        legend([h1 h2],'Entire waveform','Steady-state portion (used for FFT)')
        % ---
        % phase space for waveform (entire and steady-state)
        figure(3); clf;
        hPS1= plot(y(:,1),y(:,2)); hold on; grid on;
        hPS2= plot(y(TW:TW+P.Npoints-1,1),y(TW:TW+P.Npoints-1,2),'r.-');
        xlabel('Position'); ylabel('Velocity'); title('Phase plane');
        legend([hPS1 hPS2],'Entire waveform','Steady-state portion (used for FFT)')
        % ---
        % plot spectra of steady-state waveform
        figure(4); clf;
        hS1= plot(2*pi*freq,db(sigSPEC)); hold on; grid on;
        xlabel('Freq [rads/s]'); ylabel('Spectral amplitude [dB]');
        hS2= plot(2*pi*freq(indxB(mm)),db(mag(mm)),'rs');   % indicate extracted freq.
        legend([hS1 hS2],'Steady-state spectra','Driving freq.');
    end
    % ---
    disp([num2str(100*mm/numel(freqD)),'% done']);
end
```

```
% ++++++++++++++
% [Fig.1] ** Mags/phases extracted from the numeric steady-state responses **
figure(1); clf;
subplot(211); hh1= plot(wDrive/P.wo,mag,'ko','MarkerSize',6,'LineWidth',2); hold on; grid on;
ylabel('Magnitude');
subplot(212); hh2= plot(wDrive/P.wo,unwrap(phase)/(2*pi),'ko','MarkerSize',6,'LineWidth',2); hold on; grid on;
xlabel('Normalized (angular) angular freq (w/wo)'); ylabel('(unwrapped) Phase [cycs]');

% ++++++++++++++
% [Fig.1]** Analytic solution ** (see French, 1971; as noted above, these expressions are
% equivalent to using Fourier transforms, which implicitly assume sinusoidal steady-state, to
% solving the main ODE)
magT= P.A./sqrt((P.wo^2-wDT.^2).^2 + ((P.gamma*wDT).^2));  % mag (theory)
phaseT= atan((P.gamma*wDT)./(-P.wo^2+wDT.^2));              % phase (theory; note sign change in denom. re
convention)
phaseT= phaseT+ phase(1)+ abs(phaseT(1));    % (kludge) correct for (arb?) phase offset in numeric solution
figure(1);
subplot(211); hh3= plot(wDT/P.wo,magT,'r-','LineWidth',2);
subplot(212); hh4= plot(wDT/P.wo,unwrap(2*phaseT)/(4*pi),'r-','LineWidth',2); % kludge to get unwrapping
working

% ++++++++++++++
% [Fig.1] ** "Transfer function" ** re linear systems theory (i.e., the Fourier transform of the
% impulse response of the DHO)
init0 = [0 10]'; % set ICs such that there is an "impulse" at t=0
P.w= 0;     % make sure to "turn off" drive
options= []; [t,yI] = ode45(@EXhoResonanceFunc,tspan,init0,options,P);
specI= rfft(yI(1:P.Npoints));
magI= abs(specI);
magI= magI* (max(mag)/max(magI));       % scale impulse mag. re max. value of driven case
phaseI= angle(specI);
phaseI= phaseI+ phase(1);    % (kludge) correct for (arb?) phase offset in numeric solution
figure(1);
subplot(211); hh5= plot(2*pi*freq/P.wo,magI,'b--','LineWidth',2); xlim([wDT(1) wDT(end)]/P.wo);
subplot(212); hh6= plot(2*pi*freq/P.wo,unwrap(2*phaseI)/(4*pi),'b--','LineWidth',2); % kludge to get unwrapping
working
xlim([wDT(1) wDT(end)]/P.wo);
```

```
% ++++++++++++++
% [Fig.1] Make a legend to put it all together (re Fig.1)
figure(1); subplot(211); legend([hh1 hh3 hh5],'Numeric solution re steady-state FFT',...
    'Analytic solution','Transfer function');

% ++++++++++++++
% [Fig.5] Plot the impulse response and comparison to admittance
figure(5); clf;
subplot(221); plot(tW,yI(1:P.Npoints)); hold on; grid on; xlabel('Time [s]'); ylabel('x');
title('Impulse response (no drive; P.w=0, P.p0=0, P.v0=10'); xlim([0 tW(round(numel(tW)/3))]);
subplot(222); hI2= plot(freq,db(specI),'LineWidth',2); grid on; hold on; ylabel('Amplitude [dB]');
title('Transfer function (mag. of FFT of IR)'); xlim(P.wDrive/(2*pi));
subplot(224); hI3= plot(freq,angle(specI)/(2*pi),'LineWidth',2);  grid on; hold on;
xlabel('Frequency [Hz]'); ylabel('Phase [cycles]');
title('Transfer function (phase of FFT of IR)'); xlim(P.wDrive/(2*pi));
subplot(223);
hZa= plot(wDT,abs(Y),'k-'); grid on; hold on;  hZb= plot(wDT,abs(Z),'r.');
grid on; hold on; ylabel('Amplitude'); xlabel('Ang. requency [rad/s]'); legend([hZa
hZb],'admittance','impedance');
% ---
% for reference, also include (scaled) admittance to indicate (near?) equivalence
offset= max(db(Y))- max(db(specI)); % scaling (in dB) to match up
%magY= fliplr(db(Y)- offset);  % kludge
magY= (db(Y)- offset);
subplot(222); hI2b= plot(wDT/(2*pi),magY,'r--');
legend([hI2 hI2b],'Transf. func.','(scaled) Admittance','Location','SouthWest');
angleY= angle(Y)/(2*pi) - angle(Y(1))/(2*pi);   % there will be a slight vert. ofseet re
angle(specI)/(2*pi)
subplot(224); hI3b= plot(wDT/(2*pi),angleY,'r--');


% ++++++++++++++
% display some relevant #s to screen
disp(['Quality factor (P.wo/P.gamma)= ',num2str(Q)]);
disp(['Eigenvalues (for x=0, undriven case): ',num2str(lambdaP),' and ',num2str(lambdaM)]);
```

```
% Notes
% o To solve this numerically, need to turn 2nd order ODE into series of 1st order ODEs:
%   dx/dt= y
%   dy/dt= -P.wo^2*x - P.gamma*y + (P.A)*sin(P.w*t)
% o For autonomous case (i.e., no drive), can rewrite in matrix form such that
% A= [0 1;-P.wo^2 -P.gamma]; straight-forward to find associated eigenvalues (see below)
% o via P.solveType, user can solve either via ode45 or a hard-coded RK4
% (both should yield the same solution!); Note that (surprisingly) ode45 actually seems
% slower than the RK4 (the slowest is ode45 w/ the fixed step-size), possibly due to
% the passing of the large-ish structure P; also note that the default ode45 routine
% (i.e., adaptive step-size) introduces harmonic disortions in the spectra
% due to its nonlinear nature
% o For the analytic solution (below manifest as magT and phaseT), the
% expression used below, as derived in French (1971) for the steady-state,
% is   exactly the same as if one simply put in the Fourier transform and
% solved for the resulting magnitude and phase [confirmed on the back of an
% envelope; let x(t)= X(w)exp(i*w*t) and plug back in, solving for X(w); note then
% that magT=abs(X) and phaseT= angle(X)]
% o There are a few minor kludges below [e.g., vertical adjustment of the
% analytic solution so to match the (arbitrary?) ref. phase of the numeric
% solution)
% o Impedance (Z) for DDHO is (by definition) the complex ratio of the driving
% force and the (steady-state) velocity (see 4080W2016L10REF.pdf). Real part of Z (resistance)
% describes energy loss while imaginary part (reactance) describes energy storage
% o Comparison of the mags. for the transfer function and admittance
% (Fig.5, top right) are a bit kludgy (unsure why fliplr was needed) and
% off (worser overlap as you move away from wo)
```

```
function dy = EXhoResonanceFunc(t,y,P)
% Damped driven HO
% d^2xdt^2 = -((P.wo)^2)*x - P.gamma*dx/dt + (P.A)*sin(P.w*t)
% Note: y(1) = x, y(2)= dx/dt
dy = zeros(2,1);    % A column vector to be returned

dy(1) = y(2);
dy(2) = -((P.wo)^2)*y(1)- P.gamma*y(2)+ (P.A)*sin(P.w*t);
```

```
% RFFT: scaled real FFT, X=rfft(x)
% Returns the positive-frequency half of the transform X=FFT(x).
% The transform X is normalized so that if {x} is a sine wave of
% unit amplitude and frequency n*df, then X[n]=1.
% Usage:    X=rfft(x)
% If x is N points long, NF=N/2+1 complex points are returned.
% See also IRFFT, FAST, FSST, FFT, IFFT,

function X=rfft(x)
  [m,n]=size(x);
  if (m==1 | n==1)
    % original...
    N=length(x)/2+1;
    xc=fft(x);
    X=xc(1:fix(N));
  else
    % do it column-wise...
    N=m/2+1;
    xc=fft(x);
    X=xc(1:fix(N),:);
  end

  X = X / (length(x)/2);
  return
```

# Basic concepts this code demonstrates....

## Physics

- ODEs (e.g., Newton's 2nd, Hooke's Law)
- Resonance
- Notion of "steady-state"

## Mathematics

- Fourier transforms
- Complex #s
- Eigenvalues
- Phase space

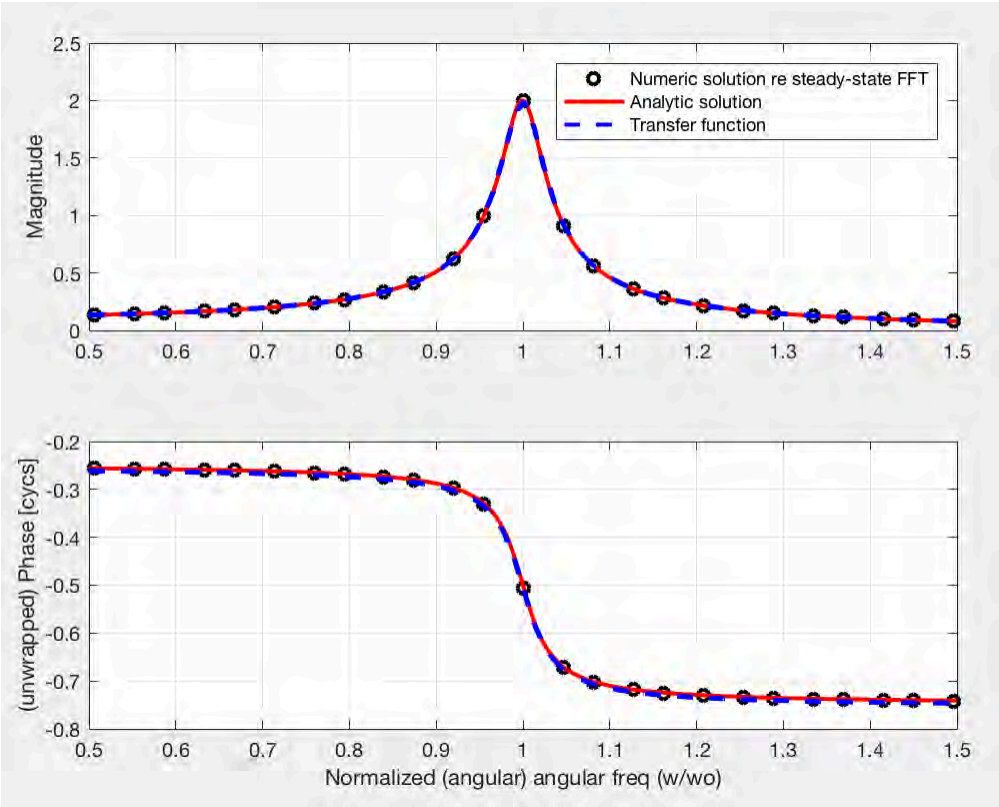
## Engineering

- Linear systems theory
- Convolutions
- Impedance/Admittance
- Impulse response
- Transfer functions

## Numerical

- Discrete Fourier transforms (FFT)
- Numerically solving ODEs (e.g., Euler, RK4, adaptive step-size and associated problems)
- Matlab syntax

$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$




Three basic approaches:
(all arriving at the same answer)

1. Numerically solve the ODEs and extract the relevant magnitudes and phases (via an FFT)

2. Analytic solution (via Fourier transforms)

3. Impulse response (and associated *transfer function*)

EXhoResonance.m

## Approach 1 – Numeric + FFT

$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$

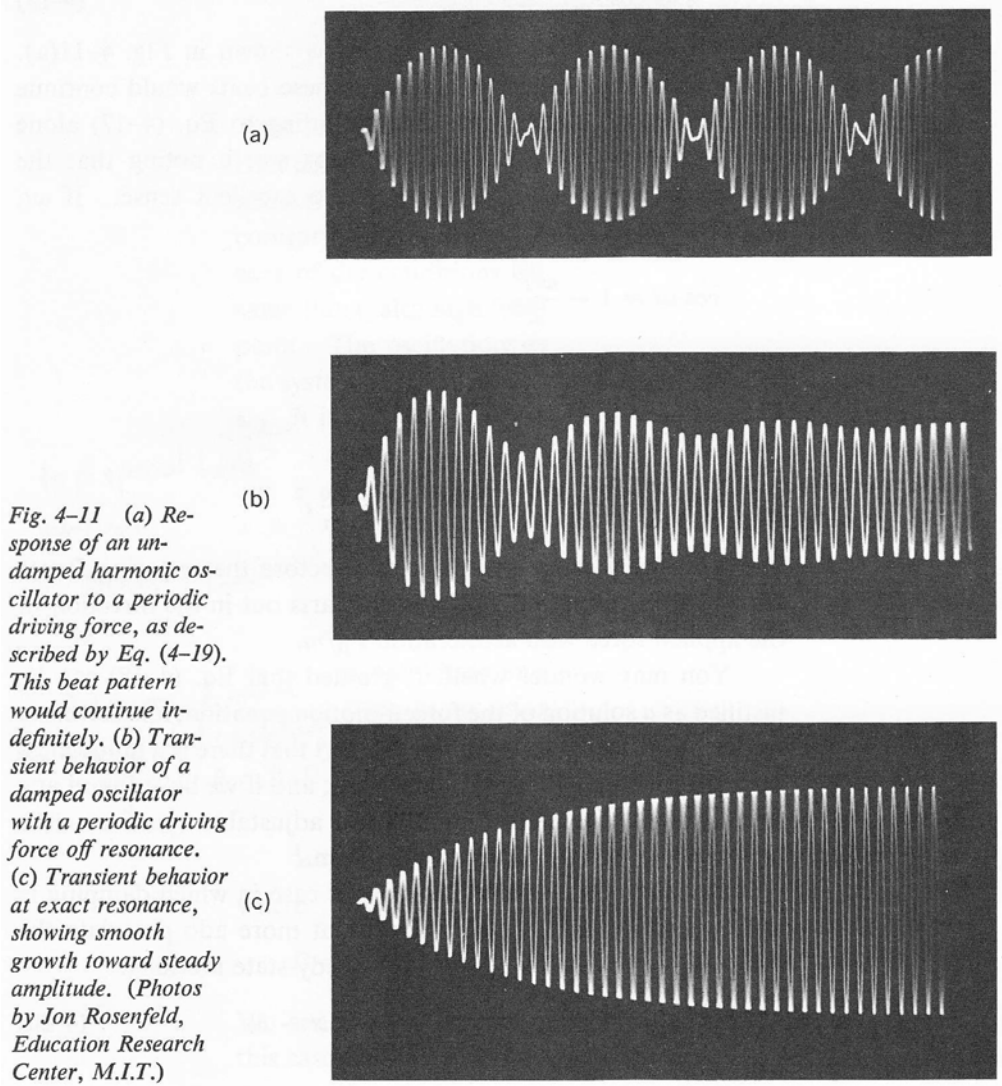


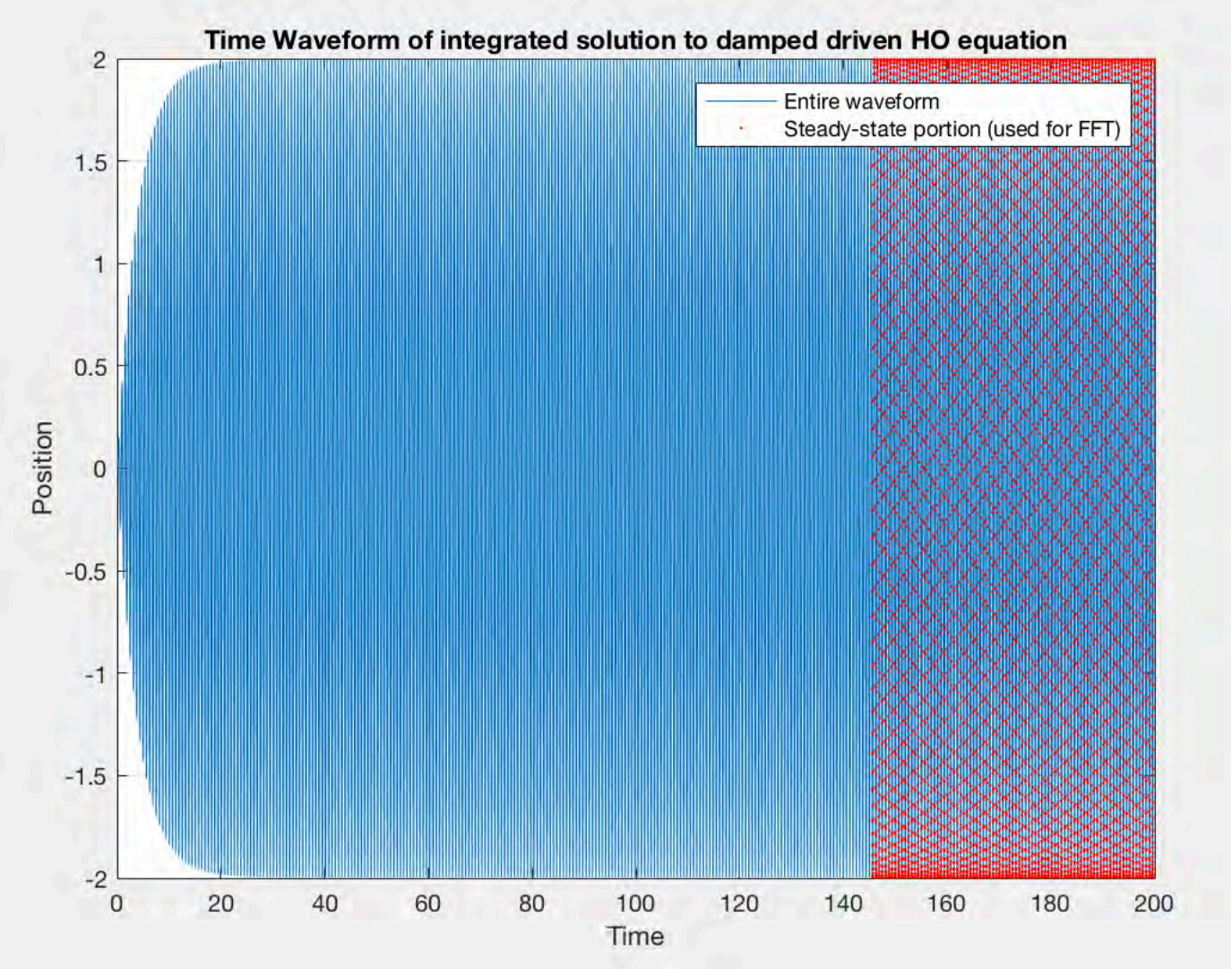

*Fig. 4–11 (a) Response of an undamped harmonic oscillator to a periodic driving force, as described by Eq. (4–19). This beat pattern would continue indefinitely. (b) Transient behavior of a damped oscillator with a periodic driving force off resonance. (c) Transient behavior at exact resonance, showing smooth growth toward steady amplitude. (Photos by Jon Rosenfeld, Education Research Center, M.I.T.)*

French (1971)

EXhoResonance.m

## Approach 1 – Numeric + FFT

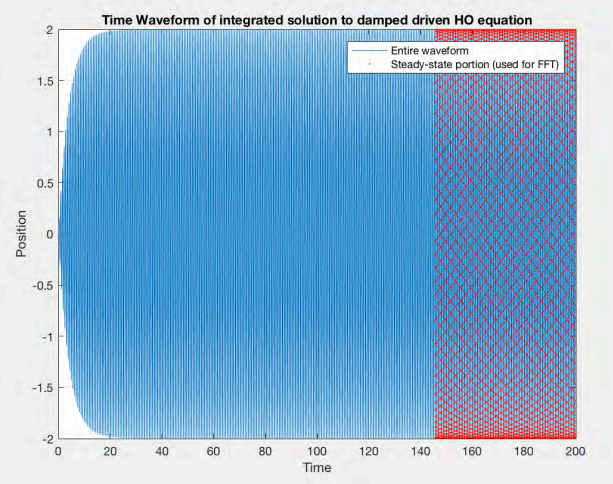
$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$



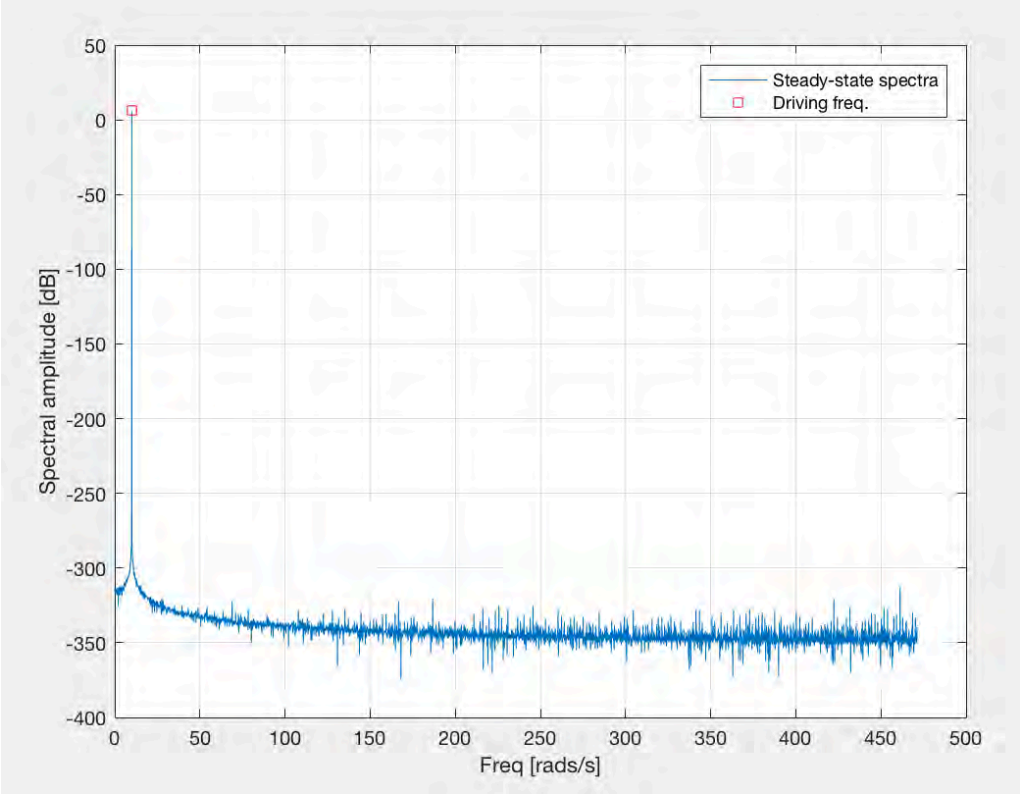
EXhoResonance.m

# Approach 1 – Numeric + FFT

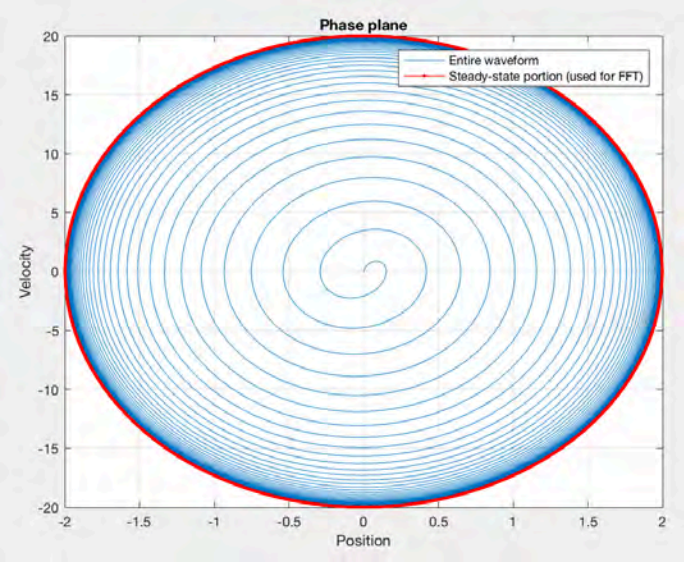
$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$

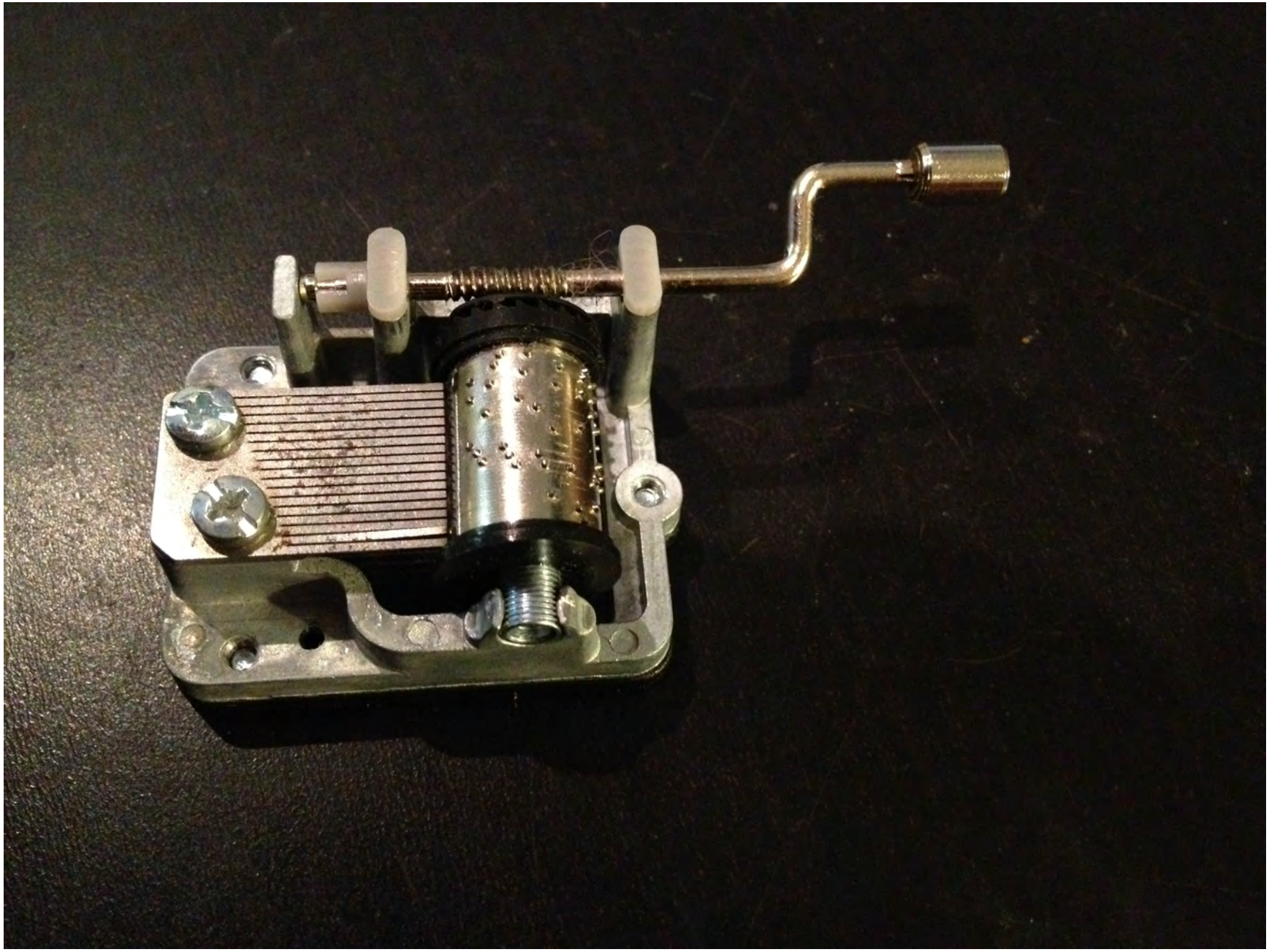



Compute the FFT to extract the magnitude (and phase; not shown)

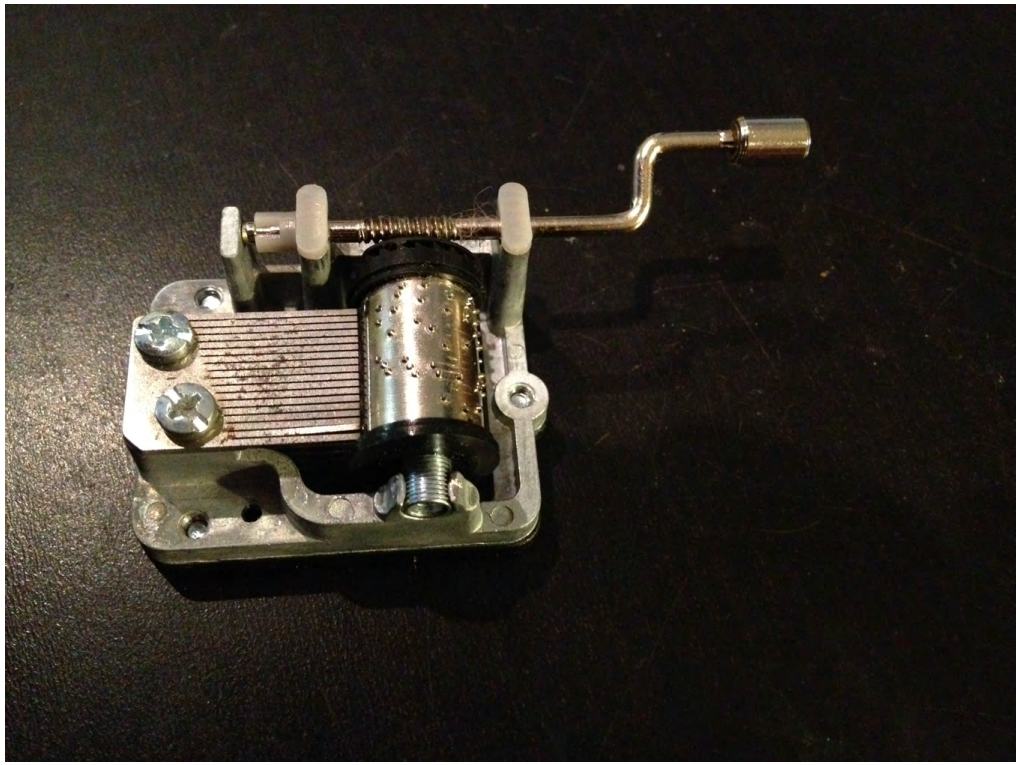



Can also plot in phase space...





# Aside: Impedance Matching

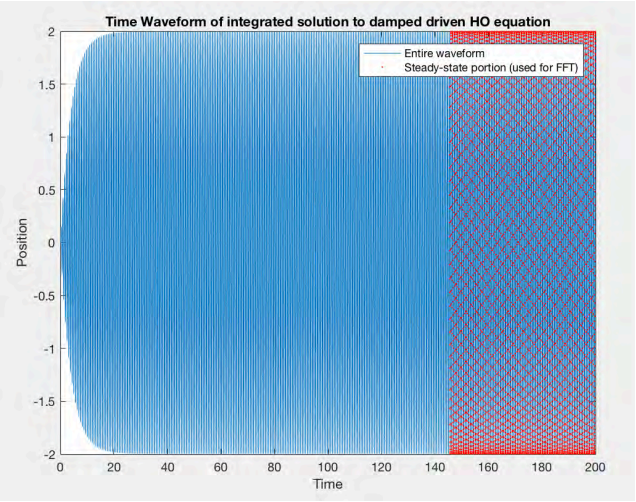


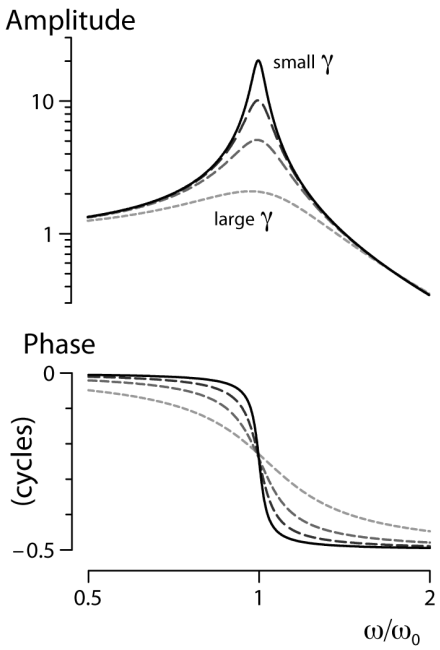
→ Notion of optimal “coupling” so to maximize flow of power....


https://en.wikipedia.org/wiki/Transformer

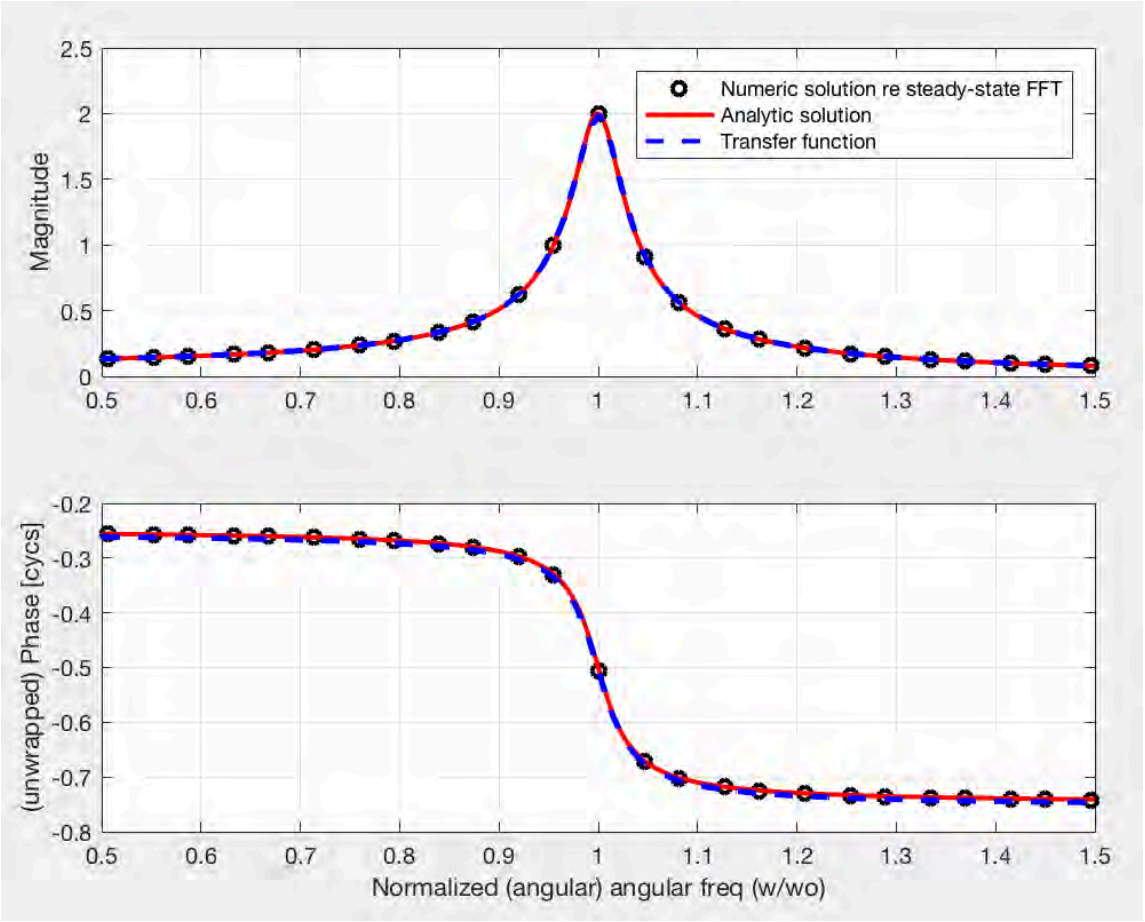
EXhoResonance.m

# Approach 1 – Numeric + FFT




→ End up w/ the black circles....



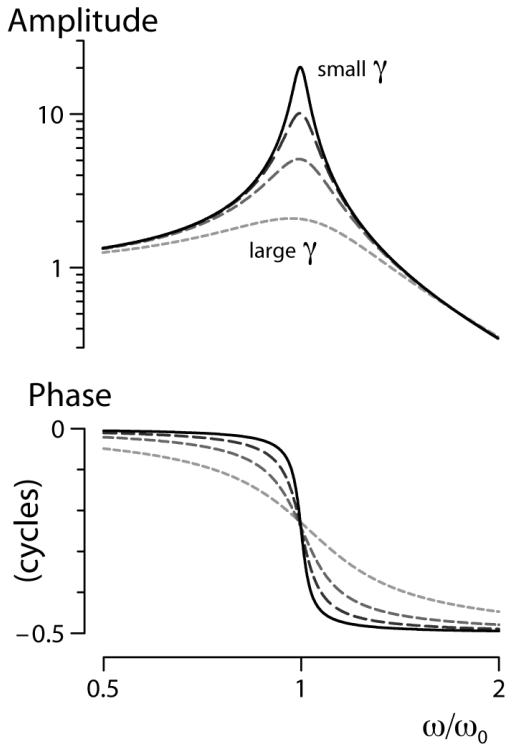
# Approach 2 – Analytic solution

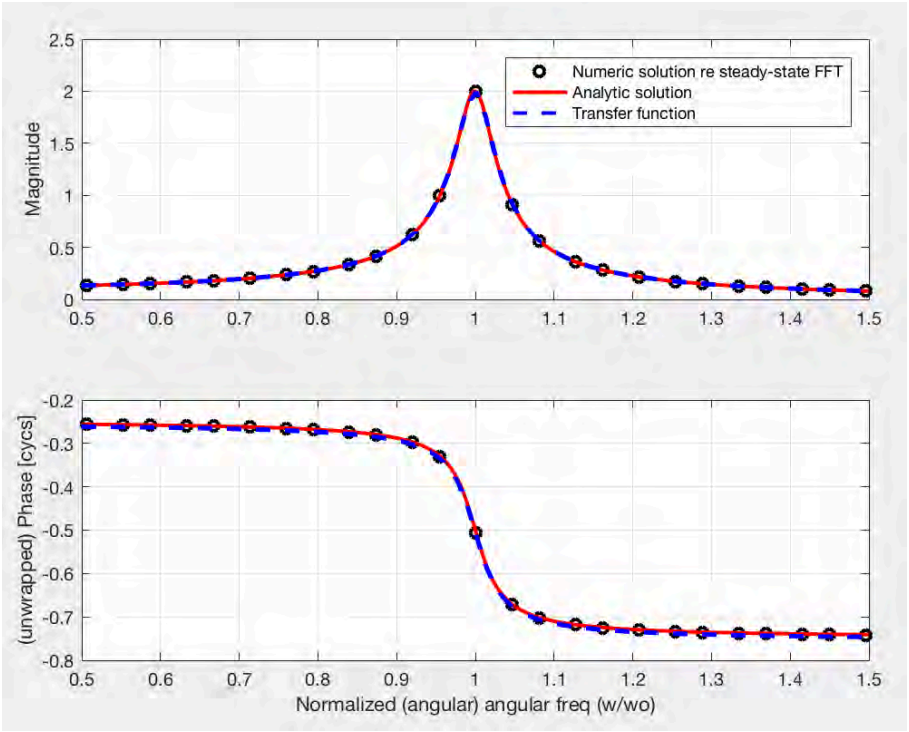
$$A(\omega) = \frac{F_o/m}{[(\omega_o^2 - \omega^2)^2 + (\gamma\omega)^2]^{1/2}}$$

$$\delta(\omega) = \arctan\left(\frac{\gamma\omega}{\omega^2 - \omega_o^2}\right)$$

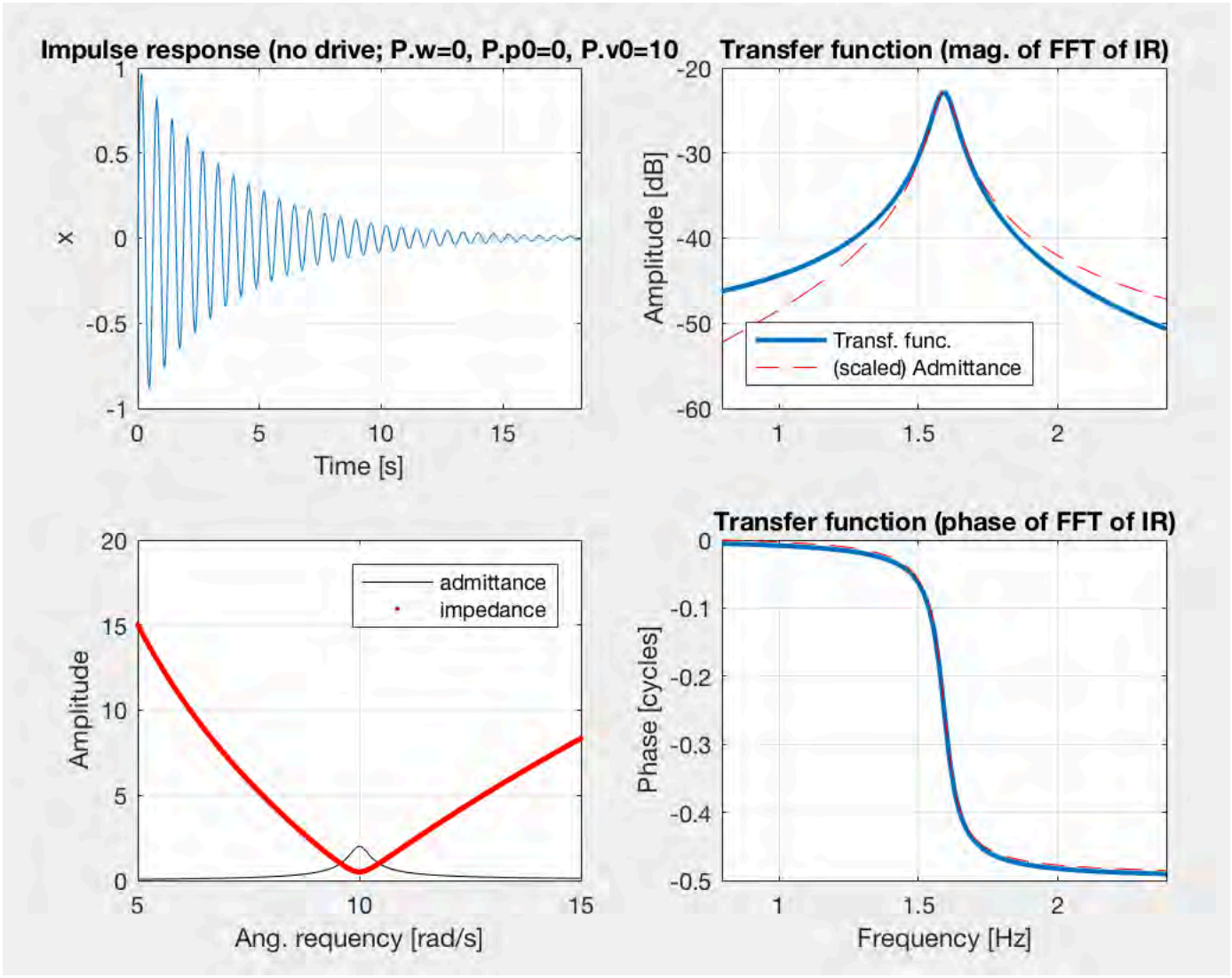
Can arrive here in a variety of ways
(including Fourier transforms; see notes at end)

→ End up w/ red line



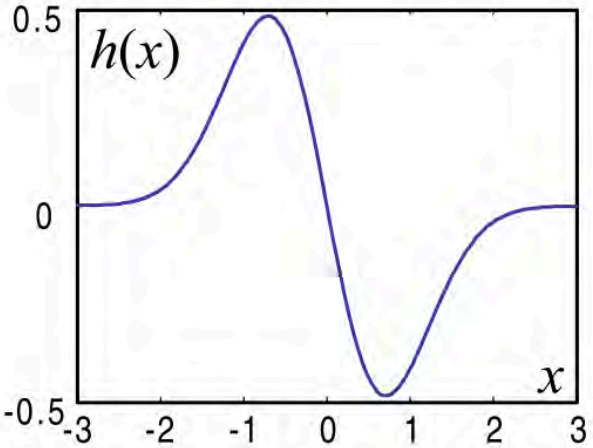



# Approach 3 – Transfer Function



# Impulse Response

"Impulse response"
(this fully characterizes the "system")




Input: Incoming "signal"
(arbitrary)

Output: "filtered" signal
(combination of the system and the input)

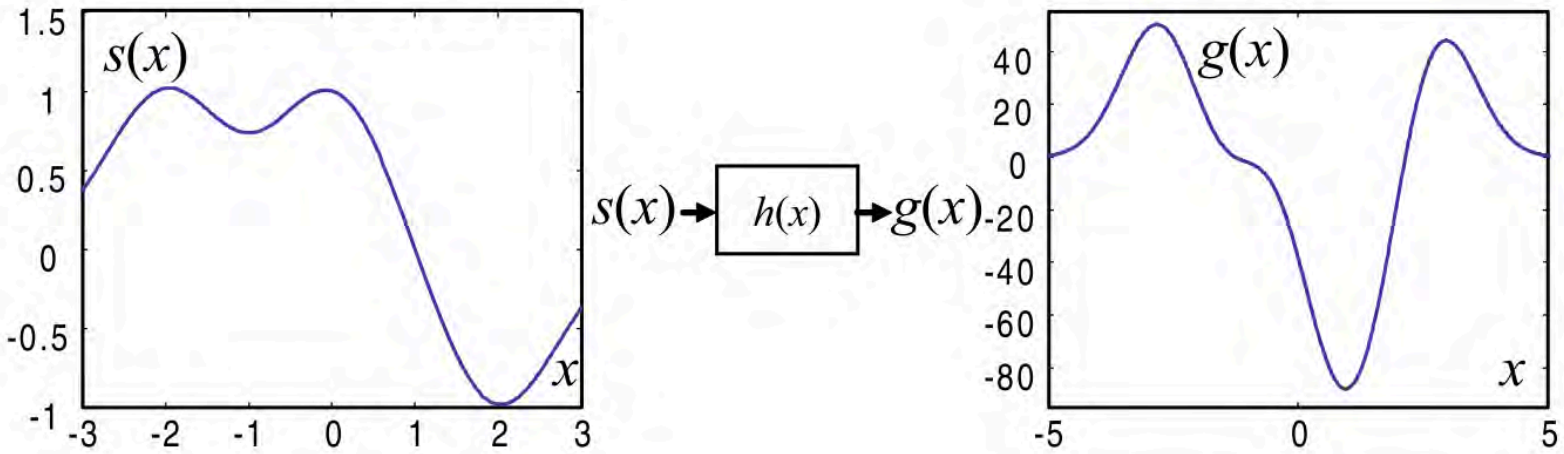
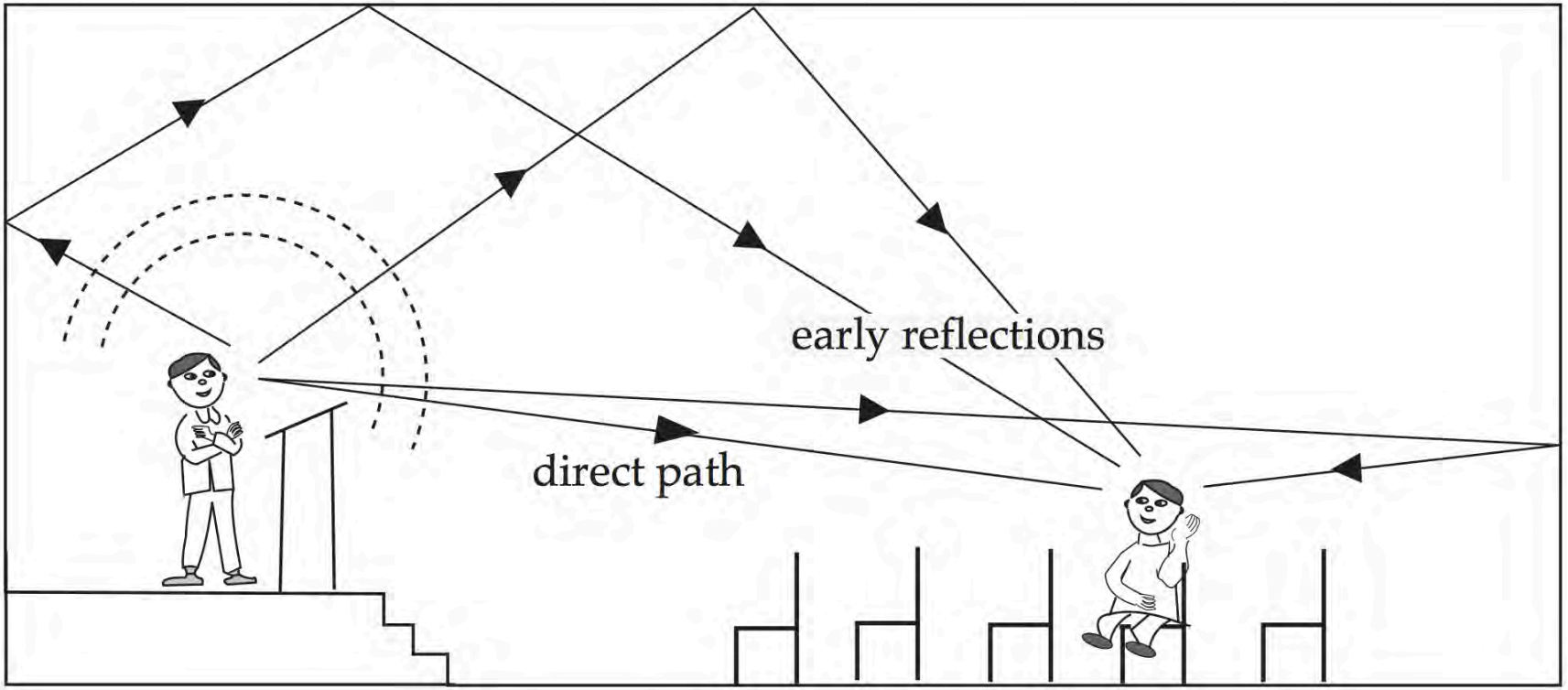



**Fig. 4.11.** Transmission of a signal. The transmitted signal is given by the convolution of the signal $s(x)$ with the system's impulse response $h(x)$

Buzug (2008)

# Ex. Acoustic Impulse Response




$$g(x) = \mathcal{L}\{s(x)\}$$

Room response ($g$) "filters" an input sound ($s$)

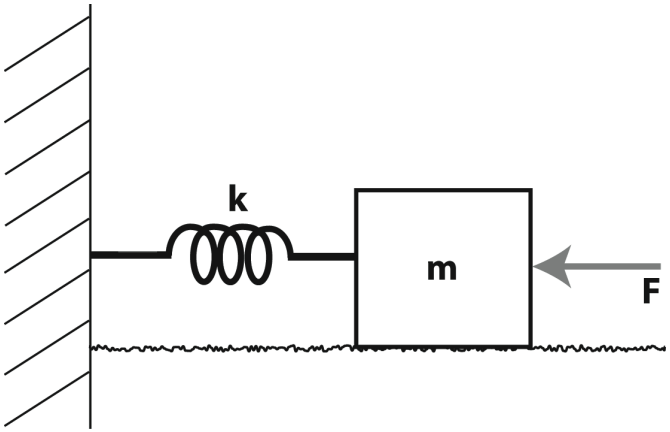
$$g(x) = s(x) * h(x) = \int_{-\infty}^{+\infty} s(\xi)h(x-\xi)\,\mathrm{d}\xi$$

Room response ($g$) is just "convolution" between $s$ and room's impulse response ($h$)

→ All the relevant bits of the room's acoustics are contained in $h$ (which we can easily measure!)

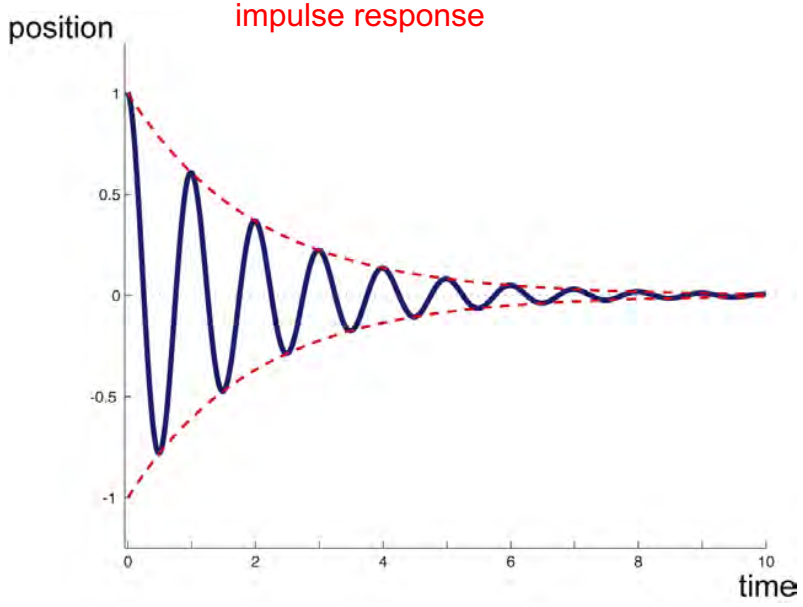
Pulkki & Karjalainen (2015)

# Transfer functions



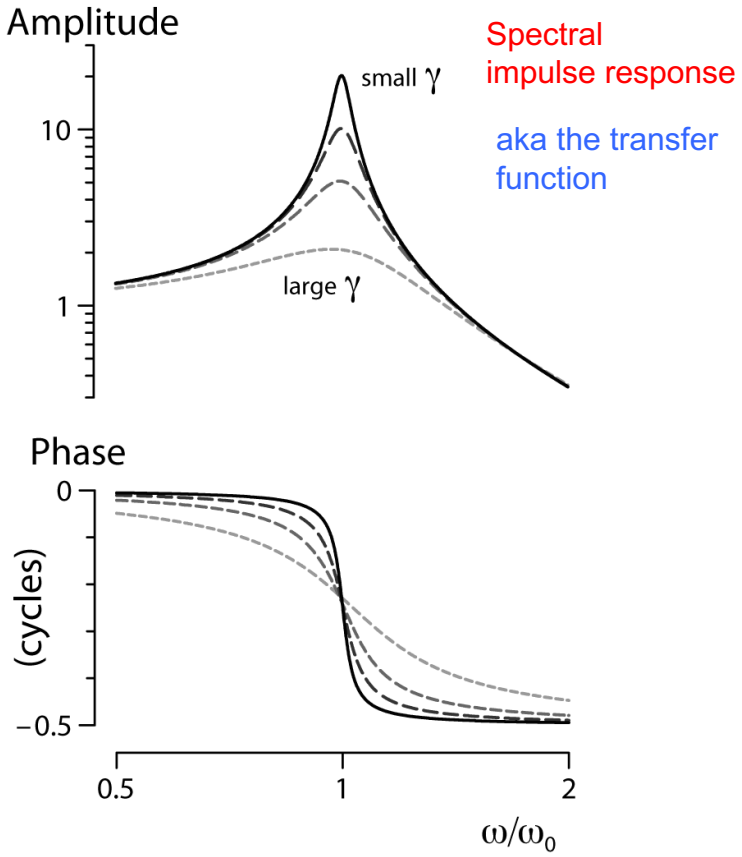

→ The “transfer function” is simply the Fourier transform of the impulse response

Temporal impulse response



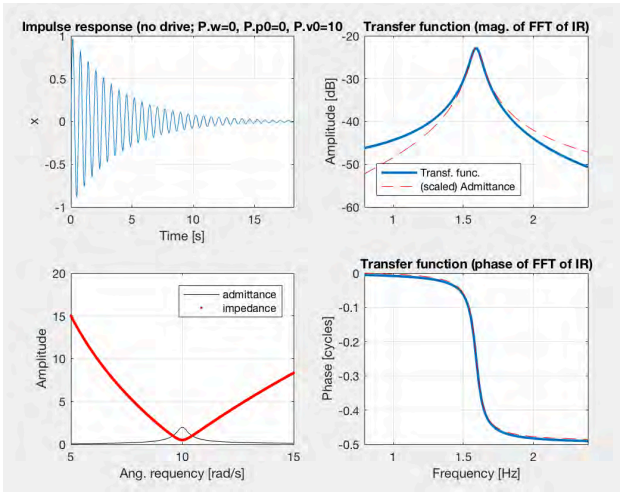

Spectral impulse response

aka the transfer function

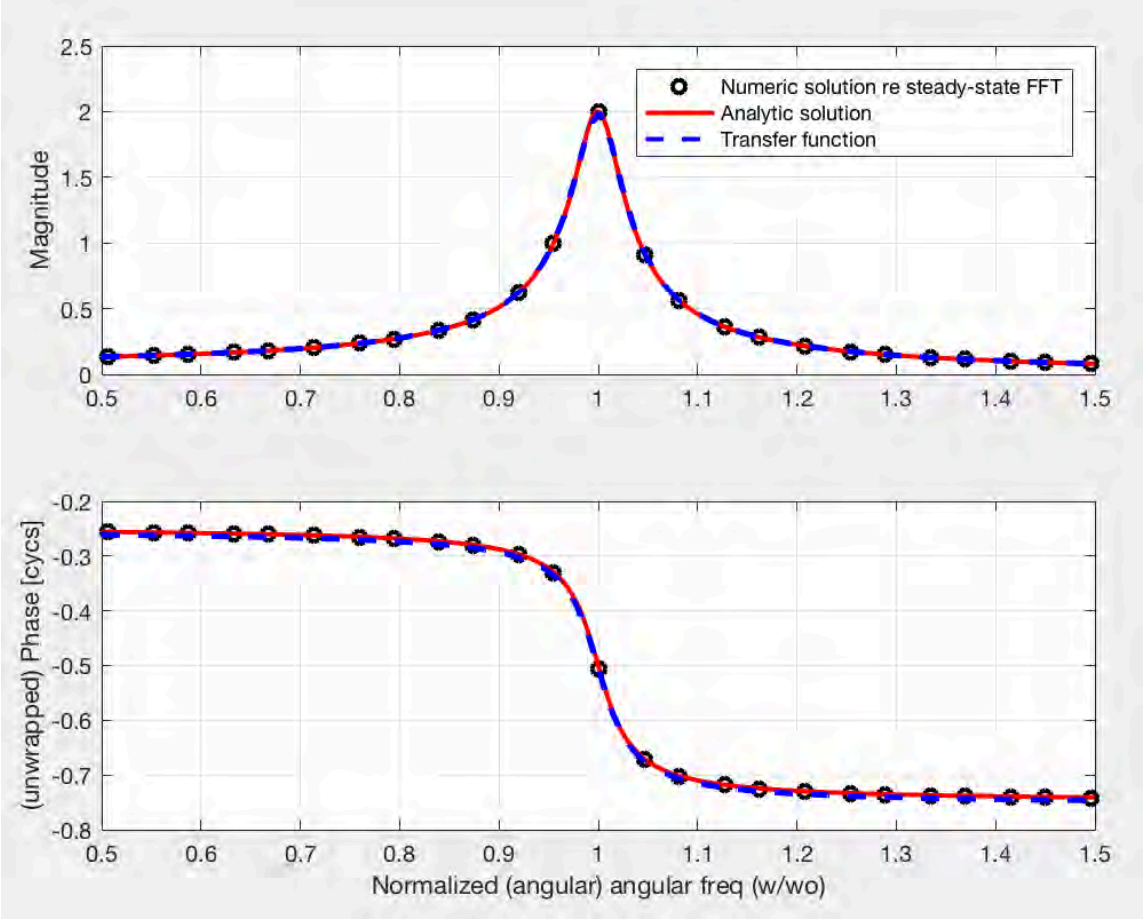



Buzug (2008)

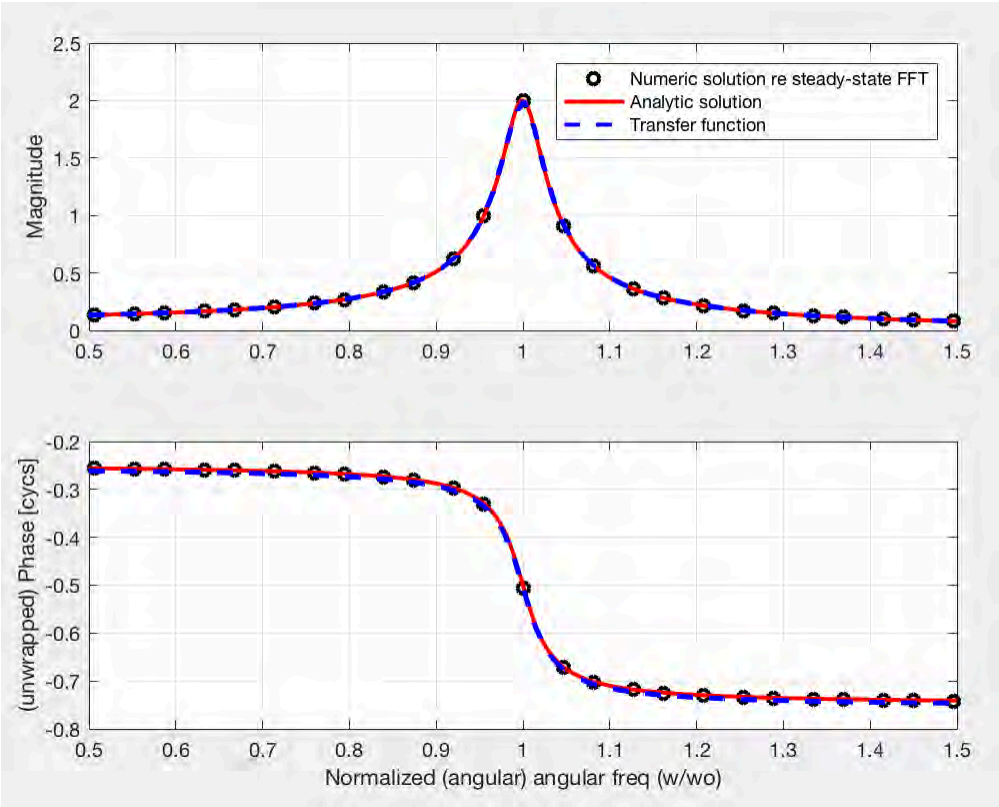
# Approach 3 – Transfer Function




→ End up w/ dashed blue curve



$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$




Three basic approaches:
(all arriving at the same answer)

1. Numerically solve the ODEs and extract the relevant magnitudes and phases (via an FFT)

2. Analytic solution (via Fourier transforms)

3. Impulse response (and associated *transfer function*)

# Basic concepts this code demonstrates....

## Physics

- ODEs (e.g., Newton's 2nd, Hooke's Law)
- Resonance
- Notion of "steady-state"

## Mathematics

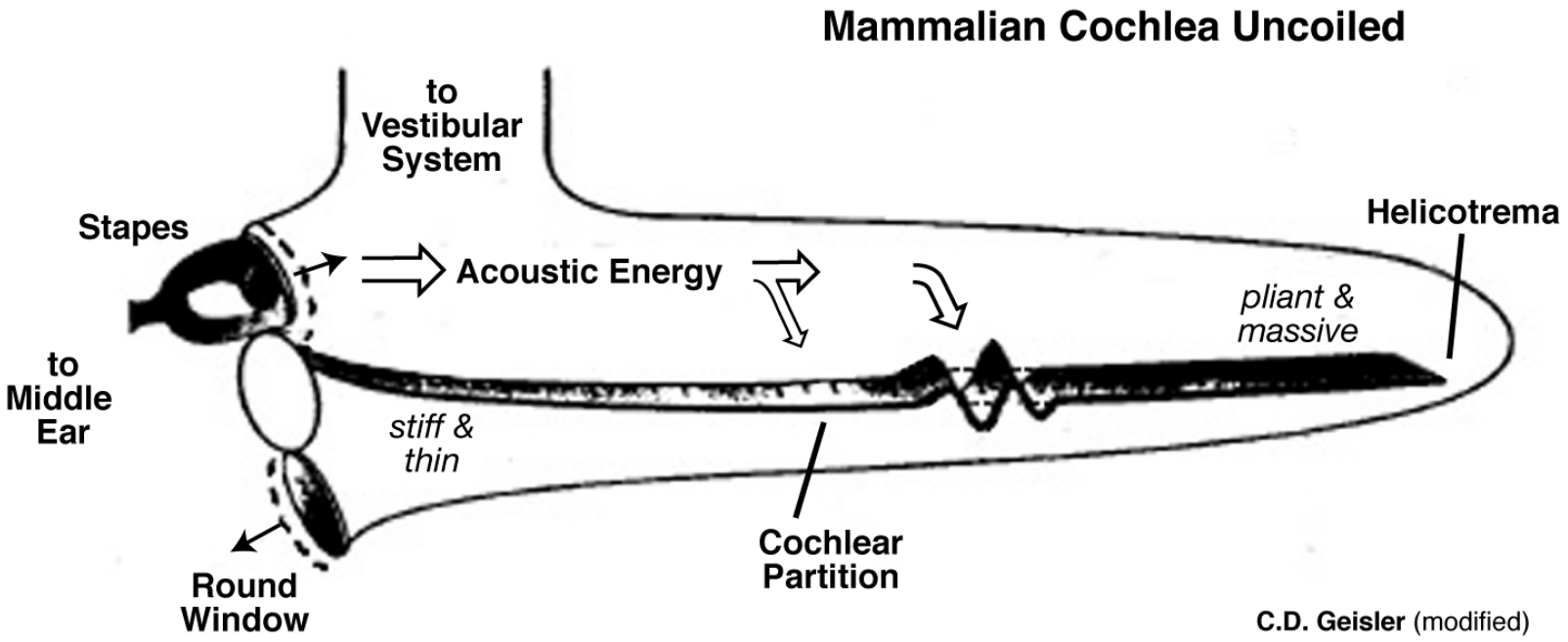
- Fourier transforms
- Complex #s
- Eigenvalues
- Phase space

## Engineering

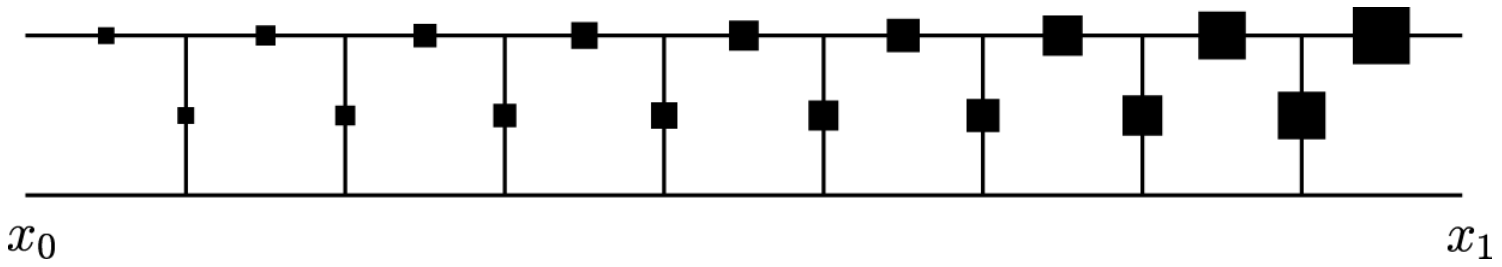
- Linear systems theory
- Convolutions
- Impedance/Admittance
- Impulse response
- Transfer functions

## Numerical

- Discrete Fourier transforms (FFT)
- Numerically solving ODEs (e.g., Euler, RK4, adaptive step-size and associated problems)
- Matlab syntax




Model: Non-uniform transmission line




→ Now in much better shape to understand this model! (see notes at end if you are curious)

Wait a sec. This thing is oscillating by itself, right?

→ Now one can do more fun (nonlinear) things....

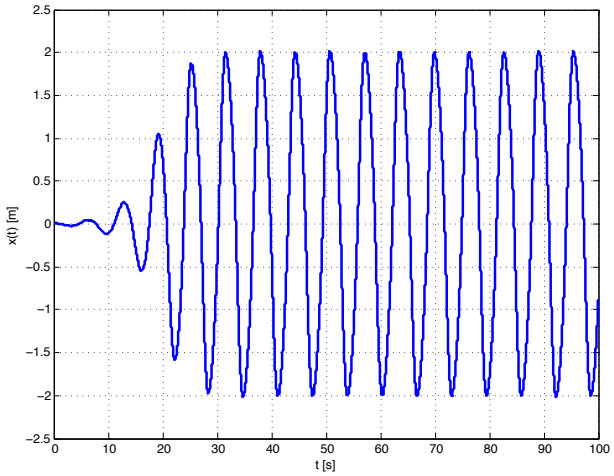
$$\ddot{x} = -\omega_o^2 x - \gamma \dot{x} + A\cos(\omega t)$$

van der Pol oscillator

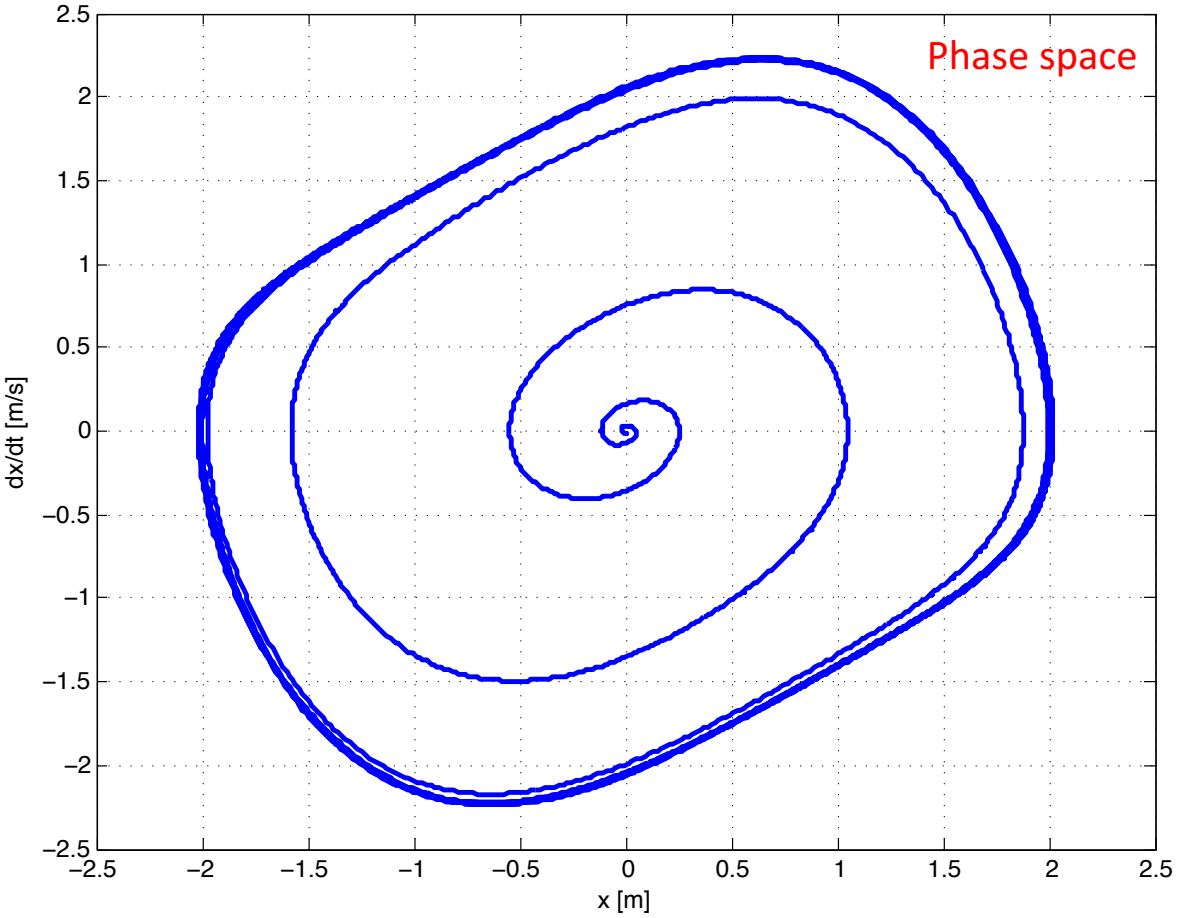
$$\ddot{x} = -x - \varepsilon(x^2 - 1)\dot{x}$$

Wikipedia

VDPode45EX.m




$$\ddot{x} = -x - \varepsilon(x^2 - 1)\dot{x}$$




## Limit cycles:

- Negative damping for small displacements injects energy into system
- Nonlinearity stabilizes
- Self-sustained oscillation!

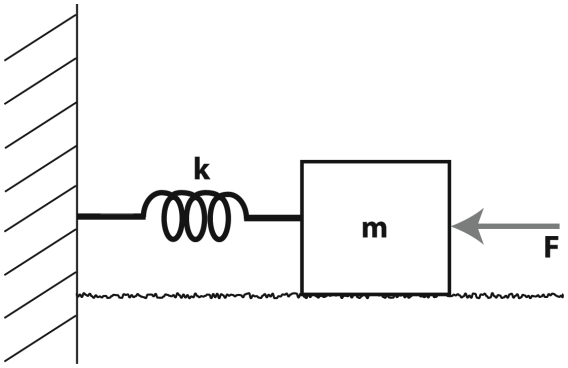
Ex. Harmonic oscillator "impulse response"

Intuitively defined in two different (but equivalent) ways:

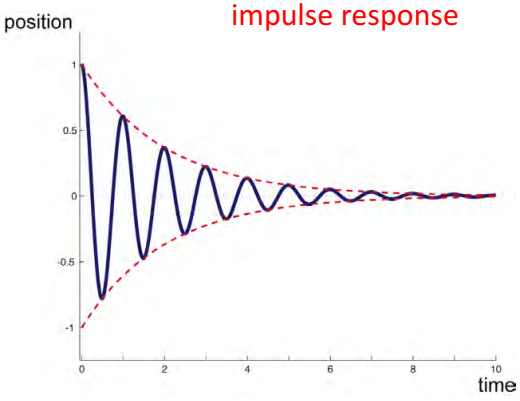
1. Time response of 'system' when subjected to an impulse
   (e.g., striking a bell w/ a hammer)

2. Fourier transform of resulting response
   (e.g., spectrum of bell ringing)

ex. Harmonic oscillator

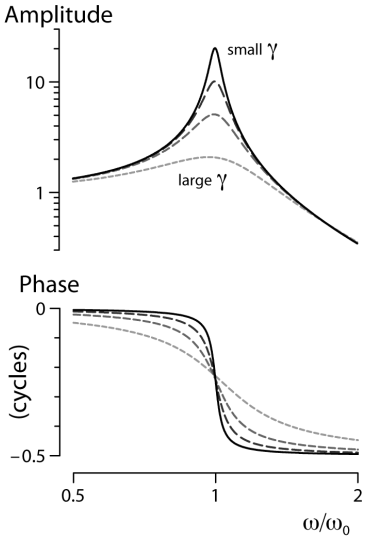



Temporal impulse response



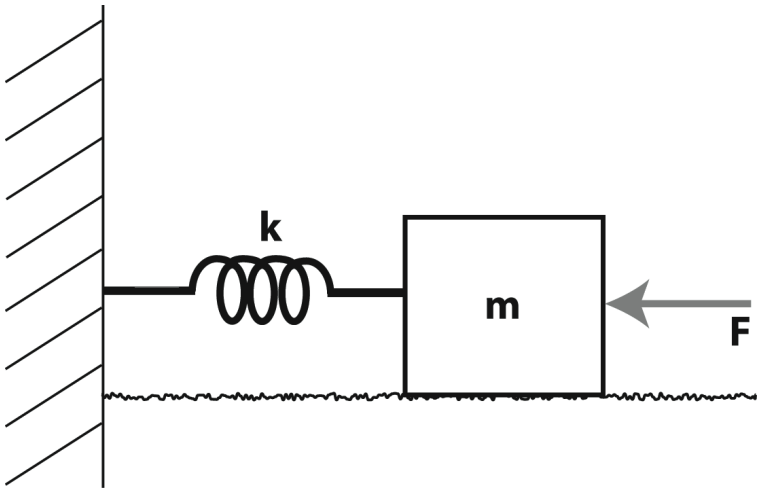

⇒ Damping causes energy loss from system

Spectral impulse response

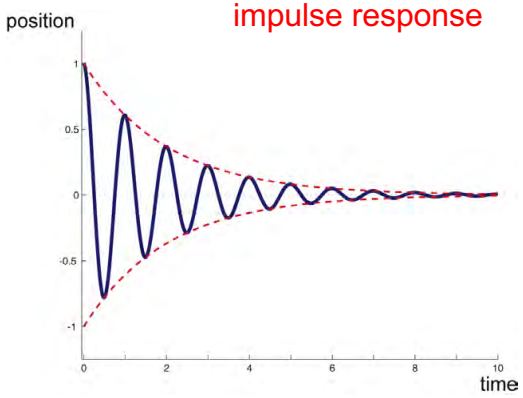



→ For linear, time-invariant systems, impulse response (don't forget the phase!!) completely characterizes the systems output for any given input (→ *Transfer Function*)

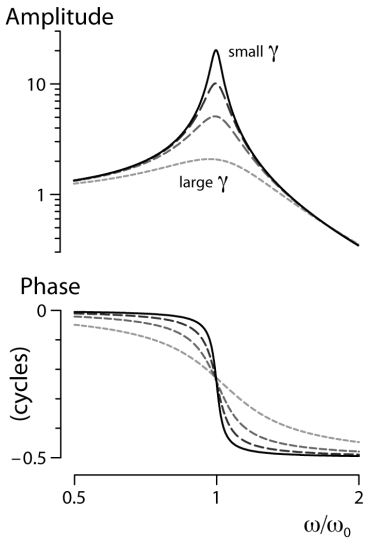
ex. Harmonic oscillator




Temporal impulse response




Spectral impulse response




If you know the impulse response (→ transfer function) for the "system" (i.e., the HO), then:

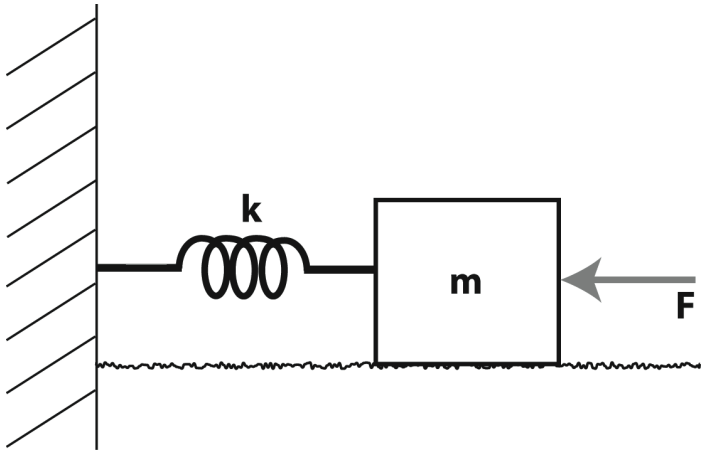
- "You tell me any sort of driving (F), I can easily tell you the response"
- Simply done by multiplying in the frequency domain, then inverse transforming (i.e., convolution!)
- Need to be cognizant of underlying assumptions (e.g., periodic boundary conds.)

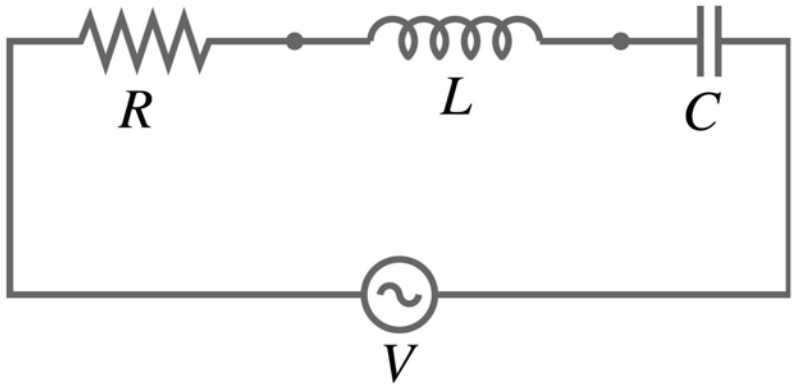
→ Convolutions are much more than just mathematical operations!

Hobbie & Roth

Ex. RLC circuit = Damped, Driven Harmonic Oscillator

| Mechanical | | Electrical | |
|---|---|---|---|
| $F$ (force) | ←→ | $V$ (potential) | state variables |
| $v$ (velocity) | ←→ | $I$ (current) | state variables |
| $x$ (position) | ←→ | $q$ (charge) | state variables |
| $m$ (mass) | ←→ | $L$ (inductance) | system properties |
| $b$ (damping) | ←→ | $R$ (resistance) | system properties |
| $k$ (spring) | ←→ | $1/C$ (capacitance) | system properties |



# Fourier analysis

Intuitive connection back to Taylor series:

$$y(x_1 + \Delta x) \approx y(x_1) + \sum_{n=1}^{N} \frac{1}{n!} \left. \frac{d^n y}{dx^n} \right|_{x_1} (\Delta x)^n. \qquad \text{(D.2)}$$

[see H&R Appendix D]

$$f(x) = f(x_o) + f'(x_o)(x - x_o) + \frac{f''(x_o)}{2!}(x - x_o)^2 + \cdots + \frac{f^{(n)}(x_o)}{n!}(x - x_o)^n + \cdots$$

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(x_o)}{n!}(x - x_o)^n$$

- Taylor series → Expand as a (infinite) sum of polynomials

- Different Idea: Fourier series → Expand as a (infinite) sum of sinusoids

# Fourier analysis

$$f(t) = a_o + a_1 \sin(\omega t) + b_1 \cos(\omega t) + \\ + a_2 \sin(2\omega t) + b_2 \cos(2\omega t) + \\ + a_3 \sin(3\omega t) + b_3 \cos(3\omega t) + \cdots$$

$$= A_0 + A_1 \sin(\omega t + \phi_1) \\ + A_2 \sin(2\omega t + \phi_2) \\ + A_3 \sin(3\omega t + \phi_3) + \cdots$$

$$= \sum_{n=0}^{\infty} A_n \sin(n\omega t + \phi_n)$$

$$= \sum_{n=0}^{\infty} B_n e^{in\omega t} \qquad \text{where } B_n \in \mathbb{C},\ i = \sqrt{-1}$$

Note:
Independent variable (t here) can represent any physical quantity, but for 1-D we typically use time

Complex #s are much more compact and easier to deal with

## Case 3: Damped, Undriven

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = 0$$

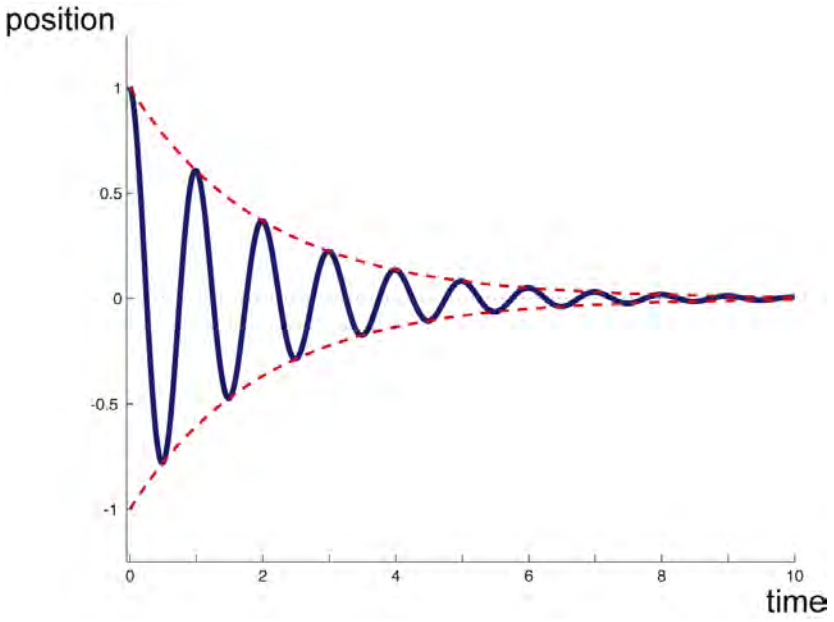
$$x(t) = Ae^{i(\omega t+\delta)}$$

$$\boxed{x(t) = Ae^{-\gamma t/2}\ e^{i(\omega t+\alpha)}}$$

[$A$ and $\alpha$ are constants of integration, depending upon initial conditions]

$$\omega^2 = \omega_o^2 - \frac{\gamma^2}{4}$$

(slightly lower frequency of oscillation due to damping)




⇒ Damping causes energy loss from system

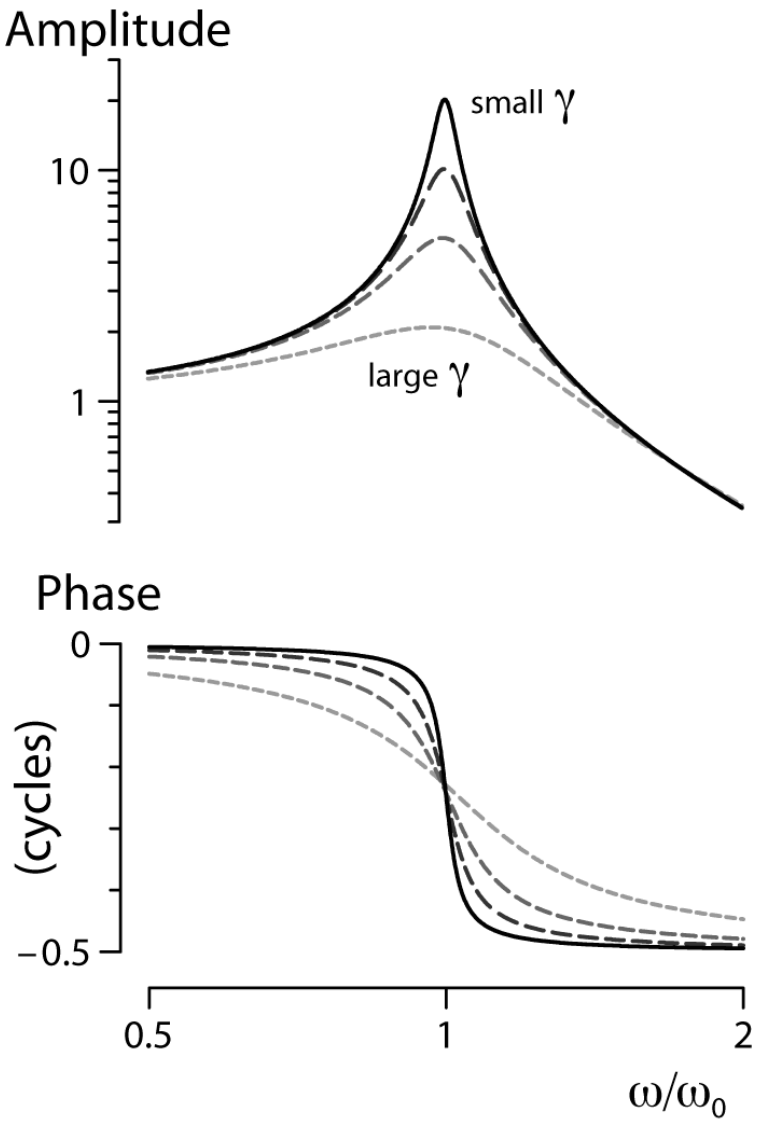
# Theoretical considerations

More realistic case: Damped, driven

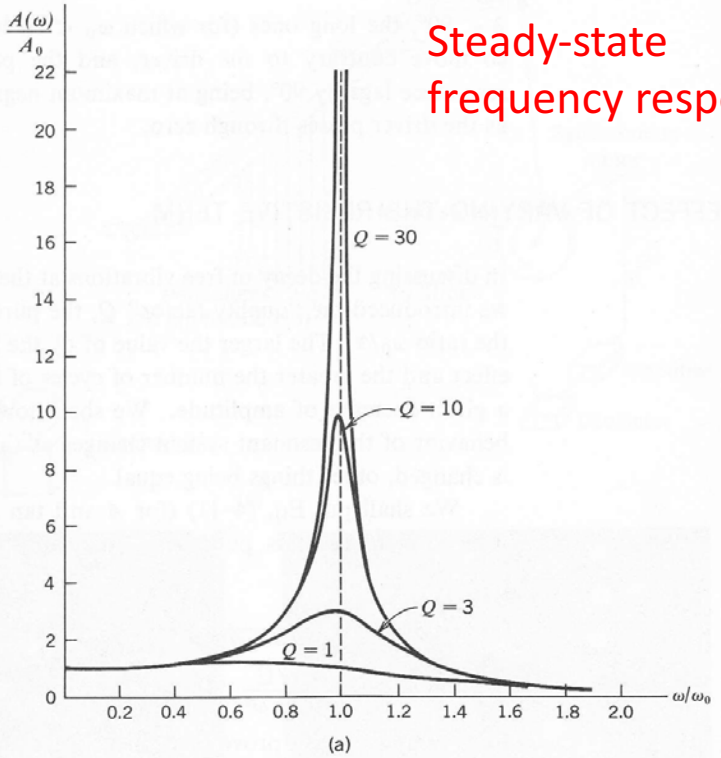
$$A(\omega) = \frac{F_o/m}{[(\omega_o^2 - \omega^2)^2 + (\gamma\omega)^2]^{1/2}}$$

$$\delta(\omega) = \arctan\left(\frac{\gamma\omega}{\omega^2 - \omega_o^2}\right)$$

⇒ Second-order oscillator behaves as as *band-pass filter*



## Steady-state frequency response



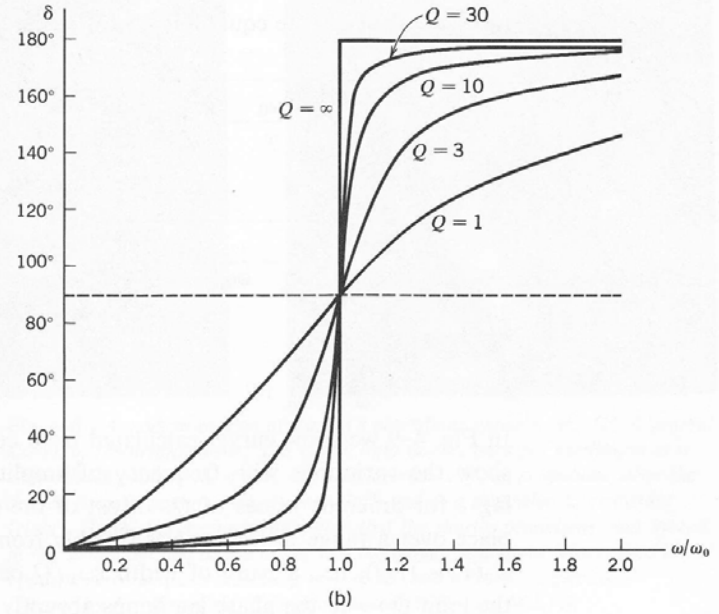




*Fig. 4–9 (a) Amplitude as function of driving frequency for different values of Q, assuming driving force of constant magnitude but variable frequency. (b) Phase difference δ as function of driving frequency for different values of Q.*

$Q$ is the 'quality factor'

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = \frac{F_o}{m} e^{i\omega t}$$

$$Q = \omega_o / \gamma$$

→ Phase information tells us something about the damping

French (1971)

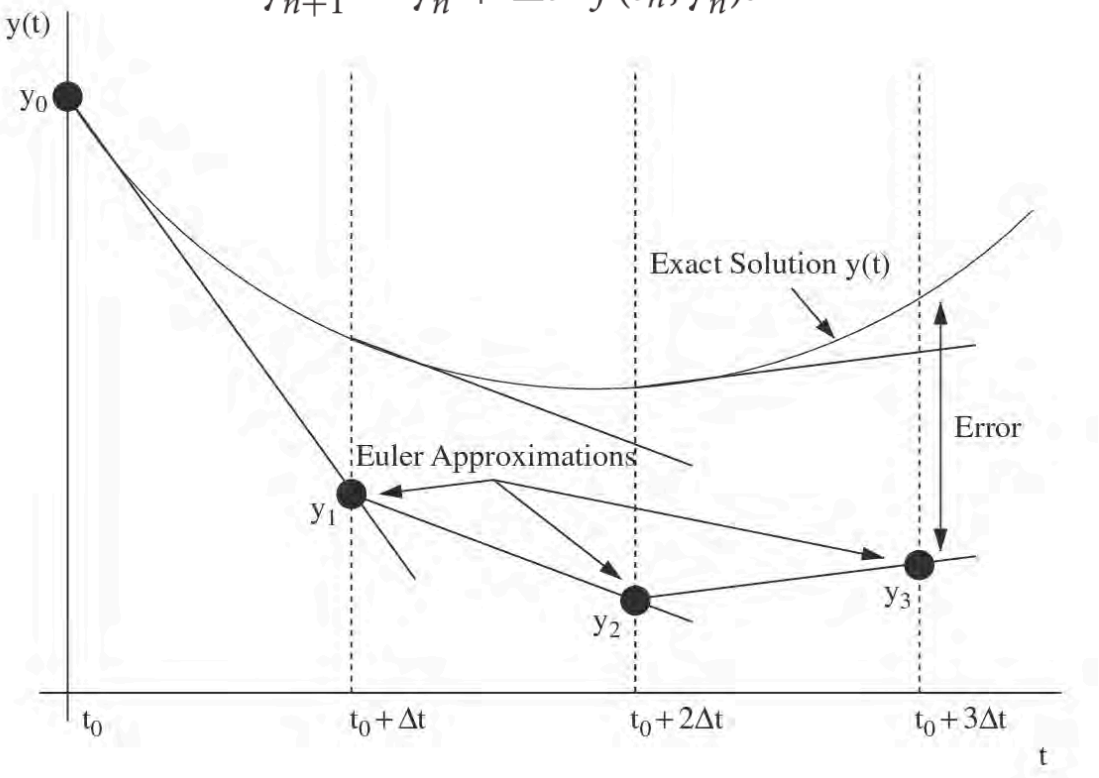
Starting point: Euler's method

Idea: Since equations tell us how things change, numerically integrate to find solution(s)

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y})$$

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) \quad \Rightarrow \quad \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{\Delta t} \approx f(t_n, \mathbf{y}_n).$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \cdot f(t_n, \mathbf{y}_n).$$




Note the 'boring' case: $f$ is a function of $t$ alone (in which case we simply integrate)

Reality check: Choice of variable names may change a bit throughout slides, but don't be confused!

$$y'(x) = f(x, y)$$

$$y(x) = \int^{x} f(\chi)\, d\chi$$

→ Note that while we can propagate a solution forward (or backward), there can be some associated ***error***

Kutz (2013)

# Error in Euler's method

$$y(x_0 + \Delta x) = y(x_0) + y'(x_0)\Delta x + \frac{1}{2!}y''(x_0)(\Delta x)^2 + \frac{1}{3!}y^{(3)}(x_0)(\Delta x)^3 + \cdots$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \cdot f(t_n, \mathbf{y}_n).$$

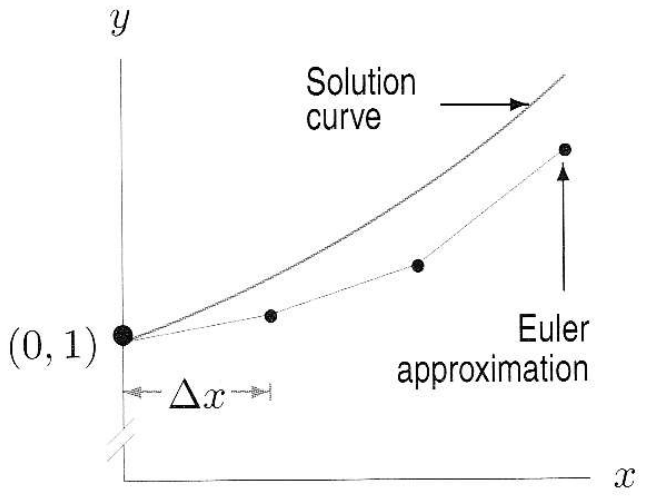



Figure 11.27: Euler's approximate solution to $dy/dx = y$

→The error in Euler's method is the difference between approximate and exact value. If the number of steps used is $n$ (think about how $n$ will be related to $\Delta t$), the error is approximately *proportional* to $1/n$

- 'higher order' methods reduce such error (e.g., Runge-Kutta) → We'll return to this

- Question: How can we know what the higher order derivatives are, since the ODE only tells us the first derivative?

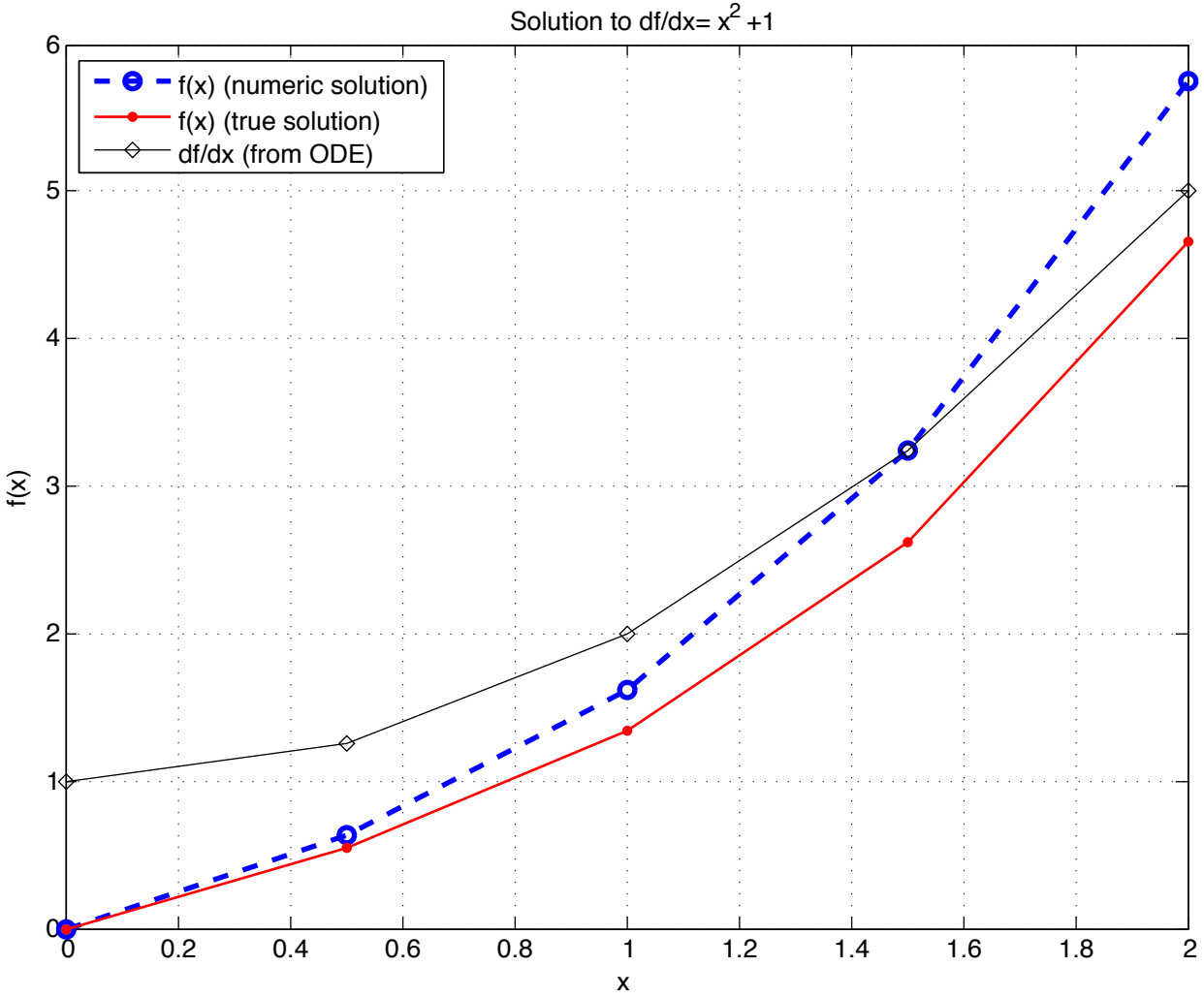
Hughes-Hallett et al. (2005)

$$\frac{df}{dx} = x^2 + 1$$

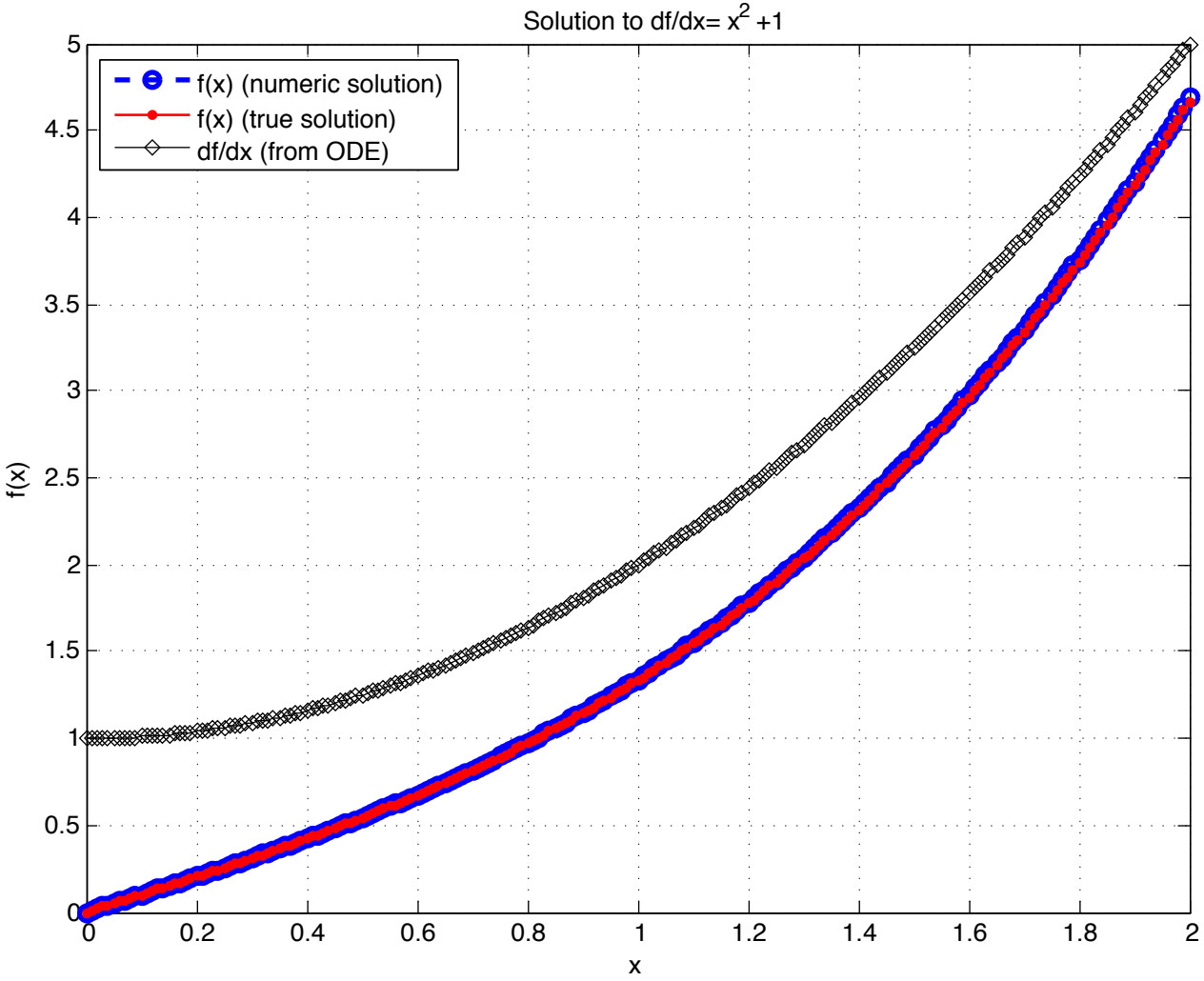
```
% York U PHYS 2030  (09.15.14)
% use basic Euler method to solve the following differential equation
% f'(x)= x^2 + 1

clear
% ************************
% User Inputs
Fprime = @(x)(x.^2+1); % function describing the ODE
Fsol= @(x,IC)(1/3*x.^3 + x + IC); % solution (known; IC is the initial condition, f0)
stepsize= 0.003;
f0= 0.0;   % initial condition at xI [i.e. f(xI)]
xB= [0 2];  % interval to solve over (i.e., boundaries)

% ************************

F(1)= f0;  %initialize first value
k=1;  % counter

for j=xB(1):stepsize:xB(2)
 if (j == xB(1)),
     F(k)= f0;  % handle initial condition
 else
     F(k)= F(k-1) + stepsize*(Fprime(j));    % apply Euler's method
 end
 x(k)= j;  % keep track of x for plotting
 k= k+ 1;  % increment counter
end

% visualize
figure(1); clf;
plot(x,F, 'o--','LineWidth',2);    grid on;    hold on
xlabel('x');    ylabel('f(x)'); title('Solution to df/dx= x^2 +1')

% now plot exact solution and derivative
plot(x,Fsol(x,f0), 'r.-','LineWidth',1);
plot(x,Fprime(x), 'kd-');
legend('f(x) (numeric solution)', 'f(x) (true solution)', 'df/dx (from ODE)','Location','NorthWest')
```

Where are the key steps occurring?

```
stepsize= 0.5;
f0= 0.0
```



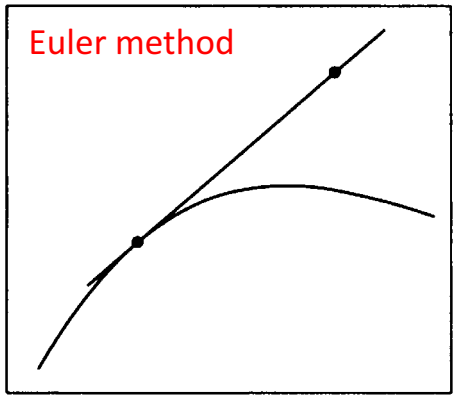
```
stepsize= 0.01;
f0= 0.0
```



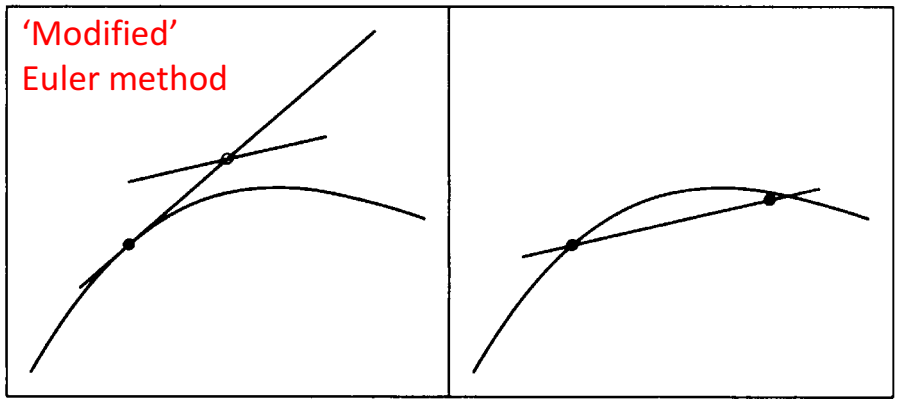
# Improve Euler's method?

- Can we 'guess' the solution at a mid-point?

$$x_{mid} = x_0 + \frac{h}{2} \qquad y(x_{mid}) = y_0 + \frac{h}{2}y_0' = y_0 + \frac{h}{2}f_0$$




$$y(x_0 + h) = y(x_0) + hf\left(x_{mid}, y_{mid}\right)$$




- Change in the associated error?
- How does this differ from changing our step-size?

$$f(x_{mid}, y_{mid}) = y'(x_{mid}) \approx \frac{y(x_0 + h) - y(x_0)}{h}$$

Devries (1994)

Runge-Kutta (RK)

- RK methods characterized by expressing solution in terms of derivative evaluated for different arguments (various Euler methods being a subset of this)

- In contrast to Taylor series solutions, which rely of successively higher order derivatives all evaluated at the same argument

- Question: How well does Euler's method and a Taylor series solution match up?

general form for all Euler method solutions

$$y(x_0 + h) = y(x_0) + h[\alpha f(x_0, y_0) + \beta f(x_0 + \gamma h, y_0 + \delta h f_0)]$$

Taylor series expansion for function of two variables

$$f(x,y) = f(x_0,y_0) + (x - x_0)\frac{\partial f(x_0,y_0)}{\partial x} + (y - y_0)\frac{\partial f(x_0,y_0)}{\partial y} + \frac{(x - x_0)^2}{2}\frac{\partial^2 f(\xi,\eta)}{\partial x^2} + (x - x_0)(y - y_0)\frac{\partial^2 f(\xi,\eta)}{\partial x \partial y} + \frac{(y - y_0)^2}{2}\frac{\partial^2 f(\xi,\eta)}{\partial y^2} + \cdots,$$

$$\textbf{where } x_0 \leq \xi \leq x \textbf{ and } y_0 \leq \eta \leq y$$

Devries (1994)

# Runge-Kutta (RK)

Use last expression to expand general Euler equation

$$y(x) = y_0 + h\alpha f(x_0, y_0) + h\beta \left[ f(x_0, y_0) + h\gamma \frac{\partial f(x_0, y_0)}{\partial x} + h\delta f(x_0, y_0) \frac{\partial f(x_0, y_0)}{\partial y} + O(h^2) \right]$$

$$= y_0 + h(\alpha + \beta) f(x_0, y_0) + h^2 \beta \left[ \gamma \frac{\partial f(x_0, y_0)}{\partial x} + \delta f(x_0, y_0) \frac{\partial f(x_0, y_0)}{\partial y} \right] + O(h^3)$$

Agrees with Taylor series

$$y(x) = y_0 + (x - x_0) f(x_0, y_0) + \frac{(x - x_0)^2}{2!} \left[ \frac{\partial f(x_0, y_0)}{\partial x} + f(x_0, y_0) \frac{\partial f(x_0, y_0)}{\partial y} \right] + \frac{(x - x_0)^3}{3!} y'''(\xi),$$

if $\alpha + \beta = 1$,
$\beta\gamma = 1/2$,
and $\beta\delta = 1/2$

→ Modified and improved Euler methods agree with Taylor series through terms of order $h^2$ and are said to be ***2nd-order RK methods***

Note: Such agreement requires that $\gamma=\delta$, but other parameters allow flexibility, e.g., $h^3$ is minimized when

$$\alpha = 1/3,\ \beta = 2/3, \text{ and } \gamma = \delta = 3/4$$

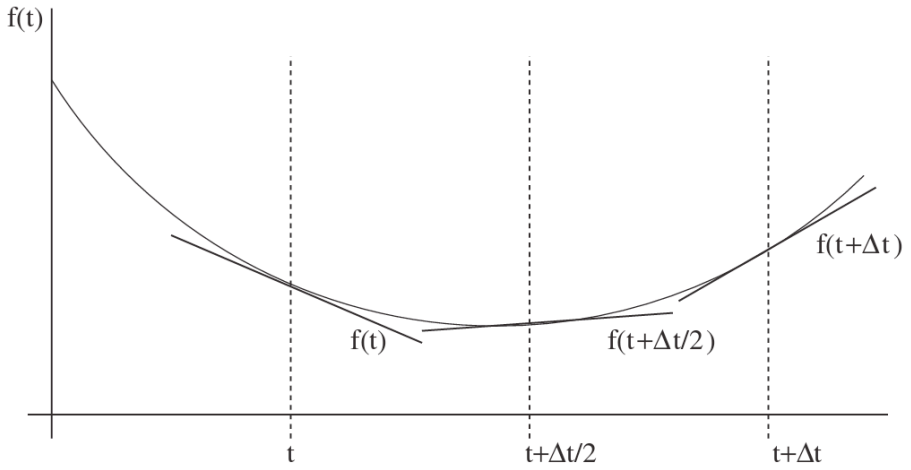
Runge-Kutta (RK)

- ‘Higher order’ methods improve in a similar fashion to Riemann sums, for example:
  - Euler → LEFT
  - Modified Euler → MID
  - Improved Euler → TRAP

- Most popular RK method is the ‘fourth order’ (RK4) and is equivalent to SIMP:

$$f_0 = f(x_0, y_0),$$
$$f_1 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} f_0),$$
$$f_2 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} f_1),$$
$$f_3 = f(x_0 + h, y_0 + h f_2).$$

$$y(x_0 + h) = y(x_0) + \frac{h}{6}(f_0 + 2f_1 + 2f_2 + f_3)$$




Devries (1994)
Kutz (2013)

# Adaptive step-size

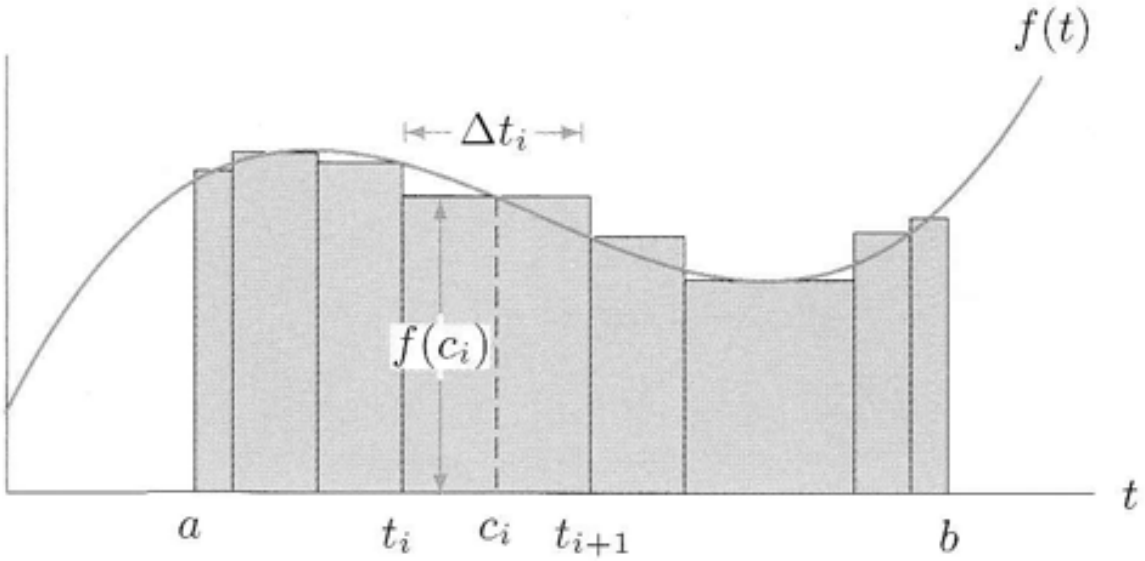
- How would you check the accuracy of your solver?

  (Easy) Brute force way: Calculate for two (or more) different step-sizes (e.g., $\Delta t$ and $\Delta t/2$) and compare → if difference is small, then the error is assumed small

- Motivates the notion of some sort of *error tolerance*

- Further motivates that step-size need not be constant

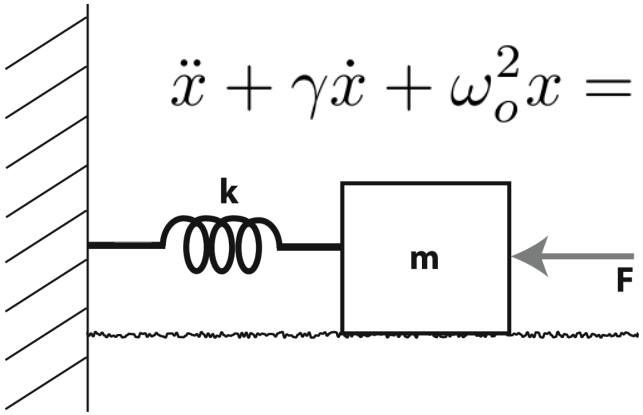
→ Just like Riemann sums, $\Delta t$ need not be constant




Hughes-Hallett et al. (2005)

# 2nd order ODEs

Harmonic oscillator




- What do you do when your ODE contains higher order derivatives?

General 1st order ODE $y' = f(x, y)$

General 2nd order ODE $y'' = f(x, y, y')$

- Simply break down into a system of 1st order ODEs, then proceed as before. This can be done simply by introducing a new variables

$\textbf{let } y_1 = y, \textbf{ and } y_2 = y'$ then $y_1' = y_2,$

$y_2' = f(x, y_1, y_2)$

e.g., harmonic oscillator $\ddot{x} + \gamma\dot{x} + \omega_o^2 x = 0$ becomes

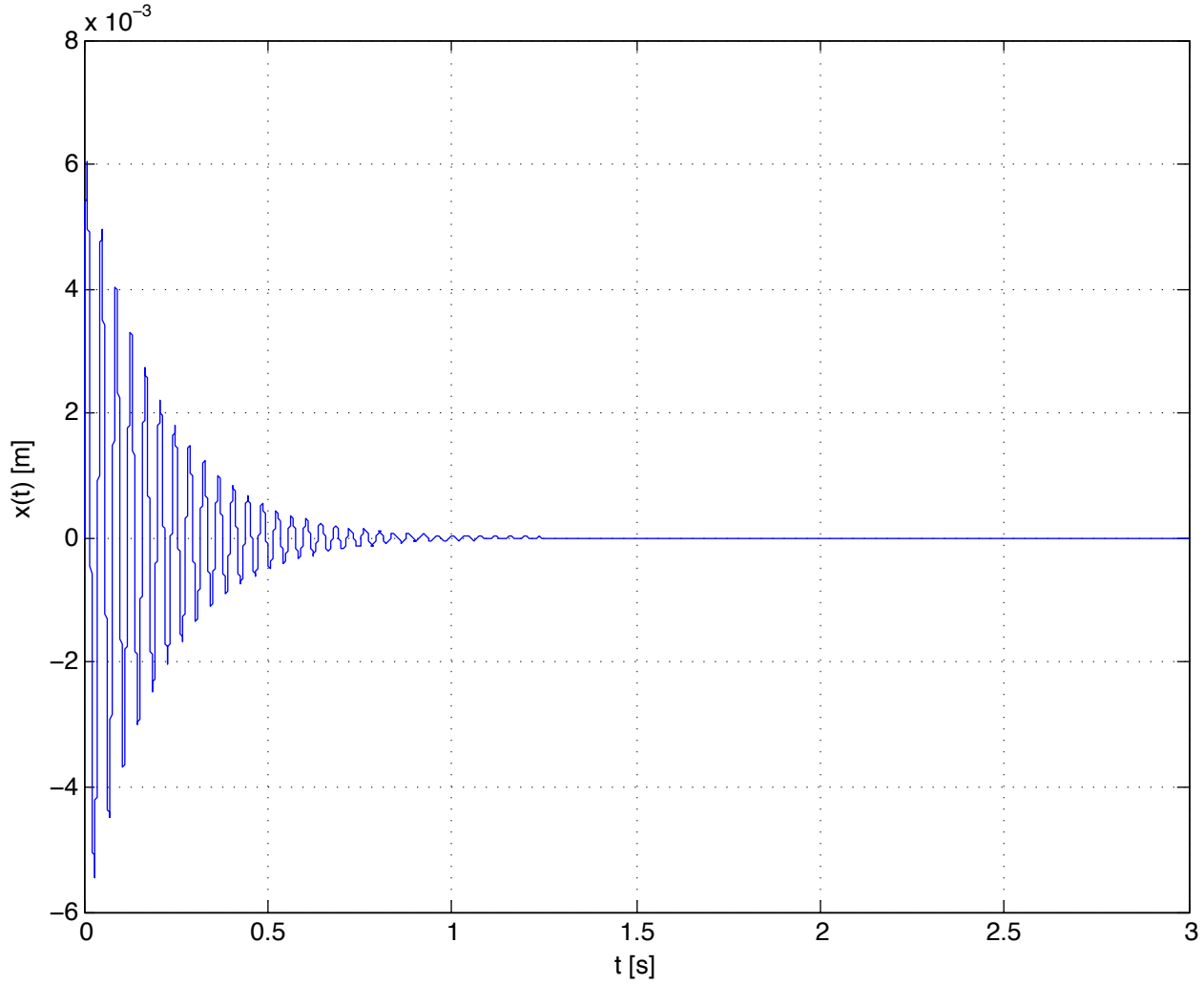
$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\omega_o^2 x - \gamma v$$

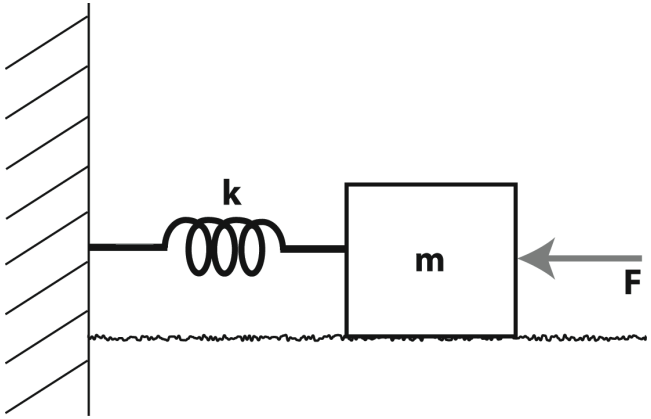
Devries (1994)

```
% ### HOode45EX.m ###
% Numerically integrate the damped/driven harmonic oscillator
%   m*x''+ b*x' + k*x = A*sin(wt)
clear
% ------------------------------------------------------
% User input (Note: All paramters are stored in a structure)
P.y0(1) = 0.0;   % initial position [m]
P.y0(2) = 1.0;   % initial velocity [m/s]
P.b= 0.1;  % damping coefficient [kg/s]
P.k= 250.0;   % stiffness [N m]
P.m= 0.01;    % mass [kg]

% sinusoidal driving term
P.A= 0.0;   % amplitude [N] (set to zero to turn off)
fD= 1.05*sqrt(P.k/P.m)/(2*pi);  % freq. (Hz) [expressed as fraction of resonant freq.]

% Integration limits
P.t0 = 0.0;   % Start value
P.tf = 3.0;   % Finish value
P.dt = 0.0001;  % time step
% ------------------------------------------------------------------------
% +++
% spit back out some basic derived quantities
P.wr= 2*pi*fD;  % convert to angular freq.
disp(sprintf('Resonant frequency ~%g [Hz]', sqrt(P.k/P.m)/(2*pi)));
Q = (sqrt(P.k/P.m))/(P.b/P.m);  % quality factor
disp(sprintf('Q-value = %g', Q));
% +++
% use built-in ode45 to solve
[t y] = ode45('HOfunction', [P.t0:P.dt:P.tf],P.y0,[],P);

% ------------------------------------------------------
% visualize
figure(1); clf;
plot(t,y(:,1)); hold on; grid on;
xlabel('t [s]');    ylabel('x(t) [m]')
% Phase plane
figure(2); clf;
plot(y(:,1), y(:,2)); hold on; grid on;
xlabel('x [m]');    ylabel('dx/dt [m/s]')
```

```
function [out1] = HOfunction(t,y,flag,P)
% -----------
%   y(1) ... position x
%   y(2) ... velocity dx/dt
out1(1)= y(2);
out1(2)= -1*(P.b/P.m)*y(2) - (P.k/P.m)*y(1)
          + (P.A/P.m)*sin(P.wr*t);
out1= out1';
```


x 10^-3
8
6
4
2
0
−2
−4
−6
x(t) [m]
0
0.5
1
1.5
2
2.5
3
t [s]

Case 1: Undamped, Undriven




$$F = ma = m\ddot{x} = -kx$$

Newton's Second Law
Hooke's Law

$$\ddot{x} + \frac{k}{m}x = 0$$

Second order ordinary differential equation
(no need worrying about how to "solve", yet...)

$$x(t) = A\cos\left(\omega_o t + \phi\right)$$

$\Rightarrow$ Solution is oscillatory!

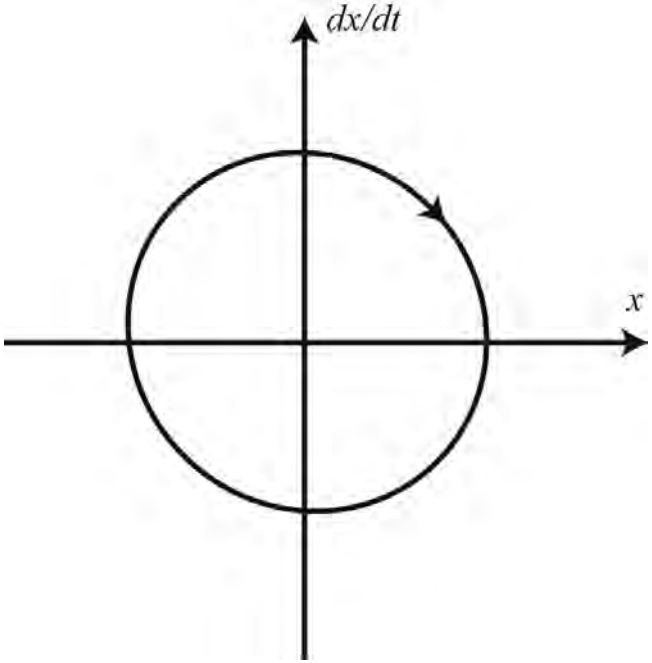
$$\omega_o = \sqrt{k/m}$$

System has a *natural* frequency

## Case 1: Undamped, Undriven (cont.)

Consider the system's energy:

$$E = T + U = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2$$




phase plane portrait

- Two means to *store* energy: mass and spring

- Oscillation results as energy transfers back and forth between these two *modes* (i.e., system is considered second-order)

## Case 2: Undamped, Driven

$$\ddot{x} + \frac{k}{m}x = F_o \cos \omega t$$

Sinusoidal driving force at frequency $\omega$

Assumption: Ignore onset behavior and that system oscillates at frequency $\omega$

$$x(t) = B \cos(\omega t + \alpha)$$

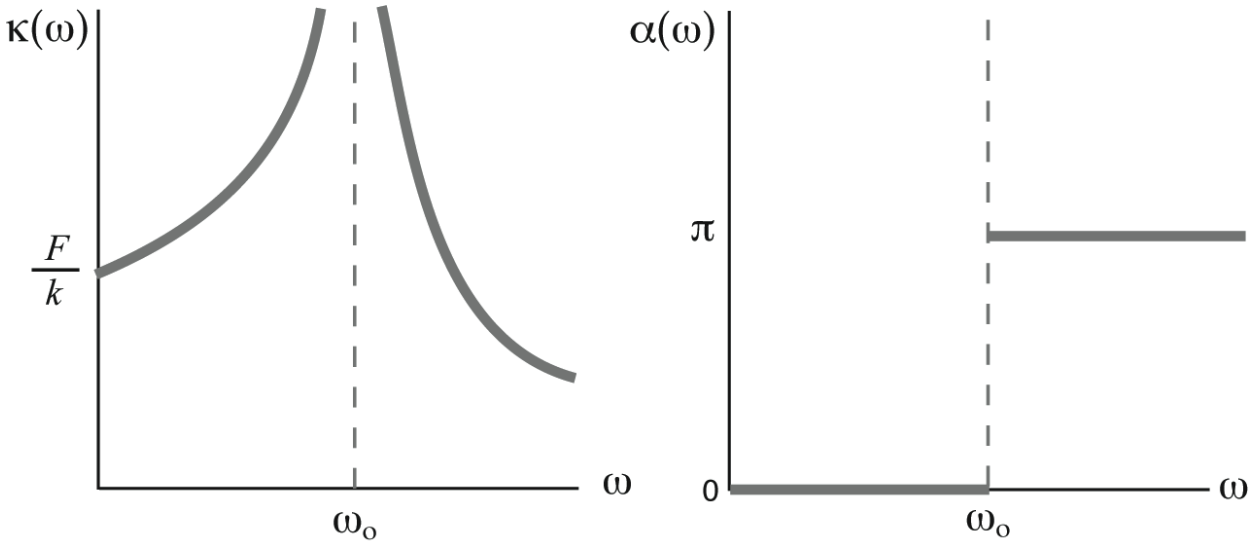
Assumed form of solution

$$-m\omega^2 B \cos \omega t + kB \cos \omega t = F_o \cos \omega t$$

$$\boxed{x(t) = \frac{F_o/m}{\omega_o^2 - \omega^2} \cos(\omega t + \alpha)}$$

## Case 2: Undamped, Driven (cont.)

$$x(t) = \frac{F_o/m}{\omega_o^2 - \omega^2}\cos(\omega t + \alpha) = \kappa(\omega)\cos(\omega t + \alpha)$$




Two Important Concepts Demonstrated Here:

- *Resonance* when system is driven at natural frequency

- Phase shift of 1/2 cycle about resonant frequency

## Case 3: Damped, Undriven

$$m\ddot{x} + b\dot{x} + kx = 0$$

Purely sinusoidal solution no longer works!

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = 0$$

Change variables

Assumption: Form of solution is a complex exponential

$$x(t) = Ae^{i(\omega t + \delta)}$$

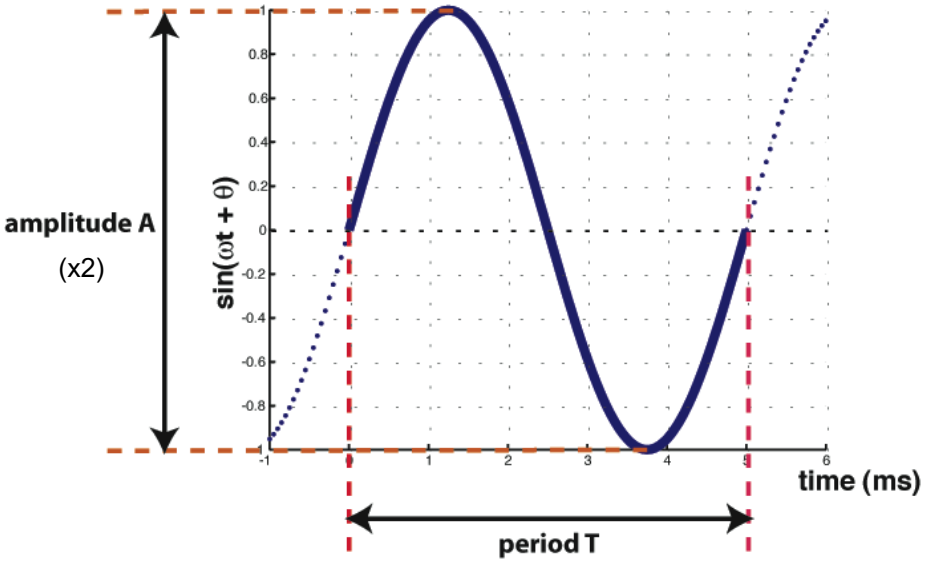
Trigonometry review ⇒ Sinusoids

Sinusoid has 3 basic properties:

i. Amplitude - height

ii. Frequency = 1/T [Hz]

iii. Phase - tells you where the peak is (needs a reference)

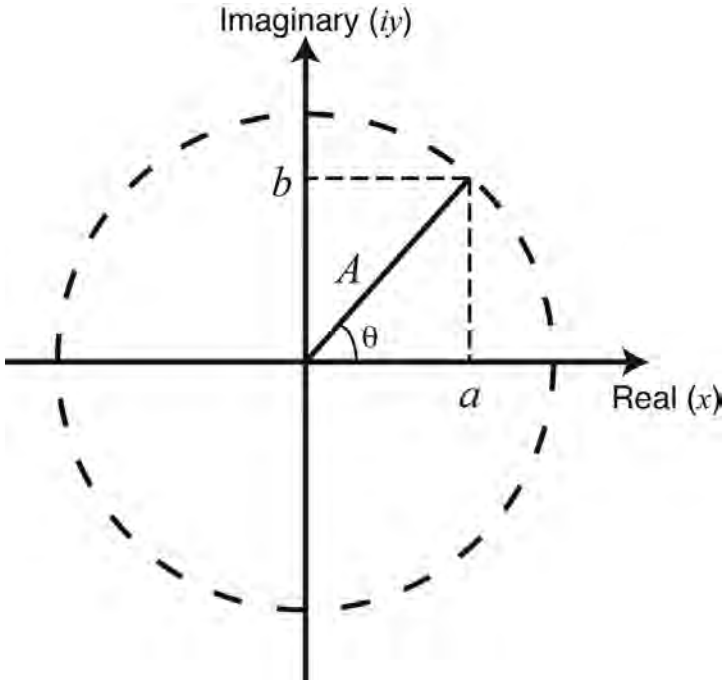



⇒ Phase reveals timing information

## Case 3: Damped, Undriven (cont.)

Motivation for complex solution:

$$a + ib = Ae^{i\theta}$$
$$= A(\cos\theta + i\sin\theta)$$




Cartesian Form

$$a = A\cos(\theta)$$
$$b = A\sin(\theta)$$

$\Leftrightarrow$

Polar Form

$$A = \sqrt{a^2 + b^2}$$
$$\theta = \tan^{-1}\left(\frac{b}{a}\right)$$

$\Rightarrow$ Complex solution contains both magnitude and phase information

## Case 3: Damped, Undriven

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = 0$$

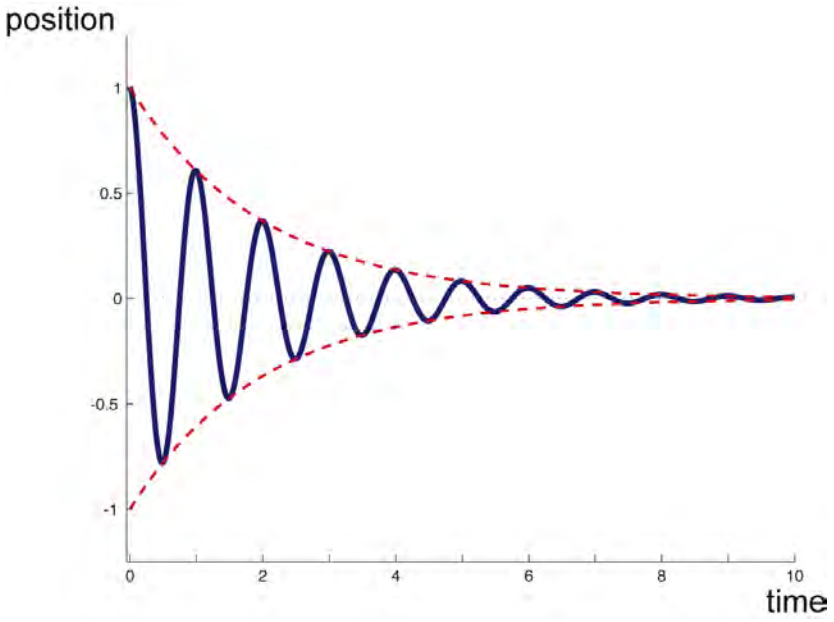
$$x(t) = Ae^{i(\omega t+\delta)}$$

$$\boxed{x(t) = Ae^{-\gamma t/2}\ e^{i(\omega t+\alpha)}}$$

[$A$ and $\alpha$ are constants of integration, depending upon initial conditions]

$$\omega^2 = \omega_o^2 - \frac{\gamma^2}{4}$$

(slightly lower frequency of oscillation due to damping)




⇒ Damping causes energy loss from system

## Case 4: Damped, Driven

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = \frac{F_o}{m} e^{i\omega t}$$

Sinusoidal driving force at frequency $\omega$

Assumption: Ignore onset behavior and that system oscillates at frequency $\omega$

$$x(t) = Ae^{-i(\omega t + \delta)}$$

Assumed form of solution

$$A(\omega) = \frac{F_o/m}{[(\omega_o^2 - \omega^2)^2 + (\gamma\omega)^2]^{1/2}}$$

(magnitude)

$$\delta(\omega) = \arctan\left(\frac{\gamma\omega}{\omega^2 - \omega_o^2}\right)$$

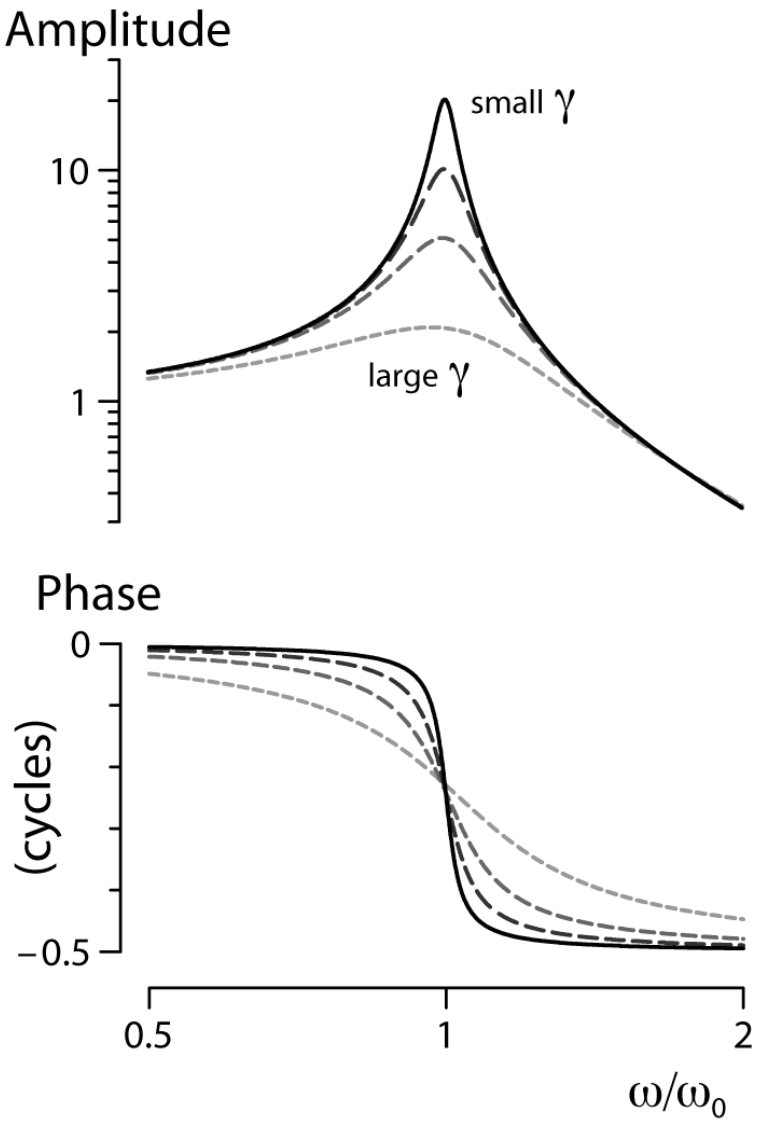
(phase)

## Case 4: Damped, Driven (cont.)

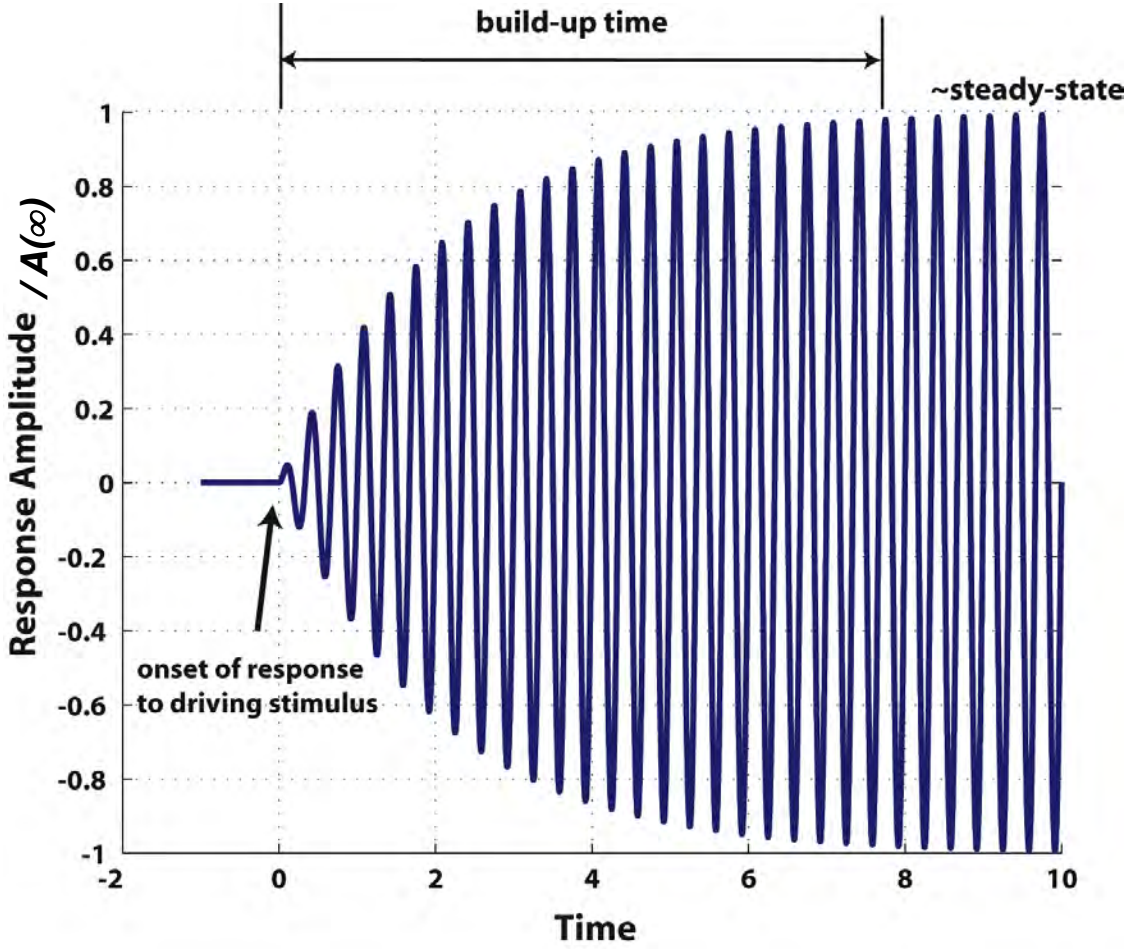
$$A(\omega) = \frac{F_o/m}{[(\omega_o^2 - \omega^2)^2 + (\gamma\omega)^2]^{1/2}}$$

$$\delta(\omega) = \arctan\left(\frac{\gamma\omega}{\omega^2 - \omega_o^2}\right)$$

⇒ Second-order oscillator behaves as as *band-pass filter*



# Basic Idea: Tuned Responses Take Time




**Second Order System**
(resonant frequency $\omega_o$)

**⇒ External driving force at frequency $\omega$**

$$x(t) = A(\infty)\,[1 - e^{(-t/\tau)}]$$

$$\tau = 1/\gamma = Q / \omega_o$$

Ex. Harmonic oscillator as an eigenvalue problem

$$\ddot{x} + \gamma\dot{x} + \omega_o^2 x = 0$$

Rewrite as a system of first order ODEs

$$\frac{dx}{dt} = y$$

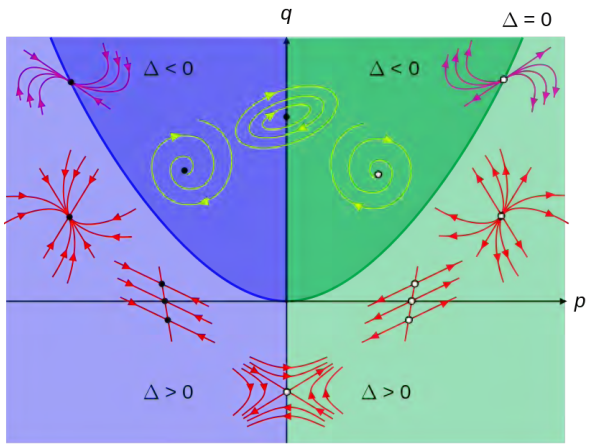
$$\frac{dy}{dt} = -\omega_o^2 x - \gamma y$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -\omega_o^2 & -\gamma \end{pmatrix}$$

$$\lambda = \frac{1}{2}\left(-\gamma \pm \sqrt{\gamma^2 - 4\omega_o^2}\right)$$

- What if $\gamma$ is zero? Negative?
- Depending upon the sign and relative values of $\gamma$ and $\omega_o$, $\lambda$ can be complex

$p = -\gamma$ $\quad$ $q = \omega_o^2$ (>0)




→ Eigenvalues characterize behavior of all possible solution types!

$$x(t) = Ae^{-\gamma t/2}\ e^{i(\omega t + \alpha)}$$