Computational Methods (PHYS 2030)

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

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

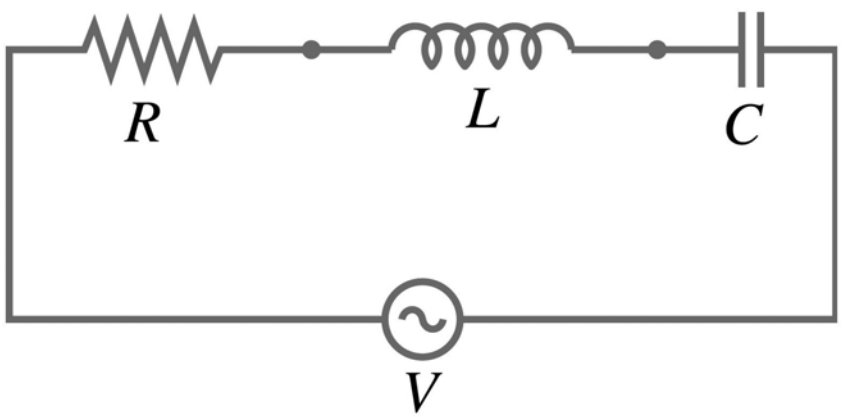
Website: http://www.yorku.ca/cberge/2030W2018.html

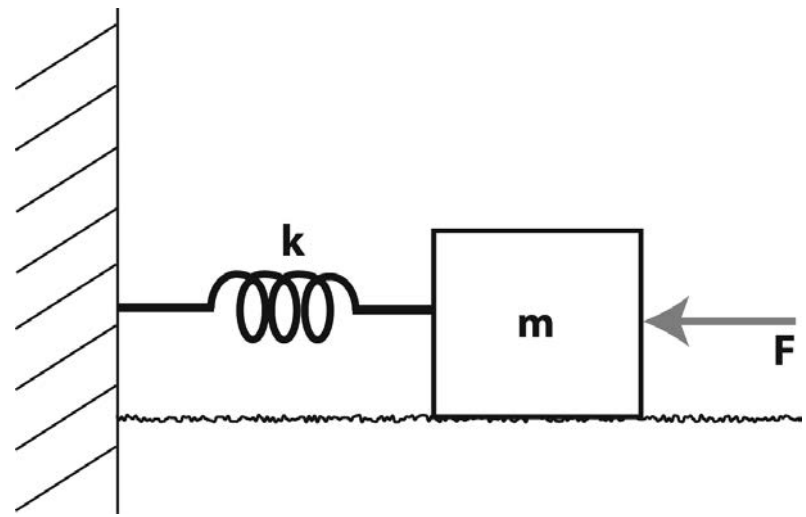
York University
Winter 2018
Lecture 8

Goal: Motivate how to deal w/ the HO

→ Recall the harmonic oscillator as our (reoccurring) example

Ex. RLC circuit, mass-on-a-spring, quantum harmonic oscillator





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

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

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

→ How do we deal numerically w/ these types of ODEs?

Systems of ODEs

- So far, we have limited ourselves to a single first order ODE. But what about 'systems' of equations?

Lorenz equations

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$

SIR model

$$\frac{dS}{dt} = -\beta IS$$

$$\frac{dI}{dt} = \beta IS - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Predator-Prey
(Lotka-Volterra equations)

$$\frac{dx}{dt} = x(\alpha - \beta y)$$

$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

- What does each term physically represent?
- Are these equations linear? Is there an exact solution?
- What is the 'atto-fox' problem?

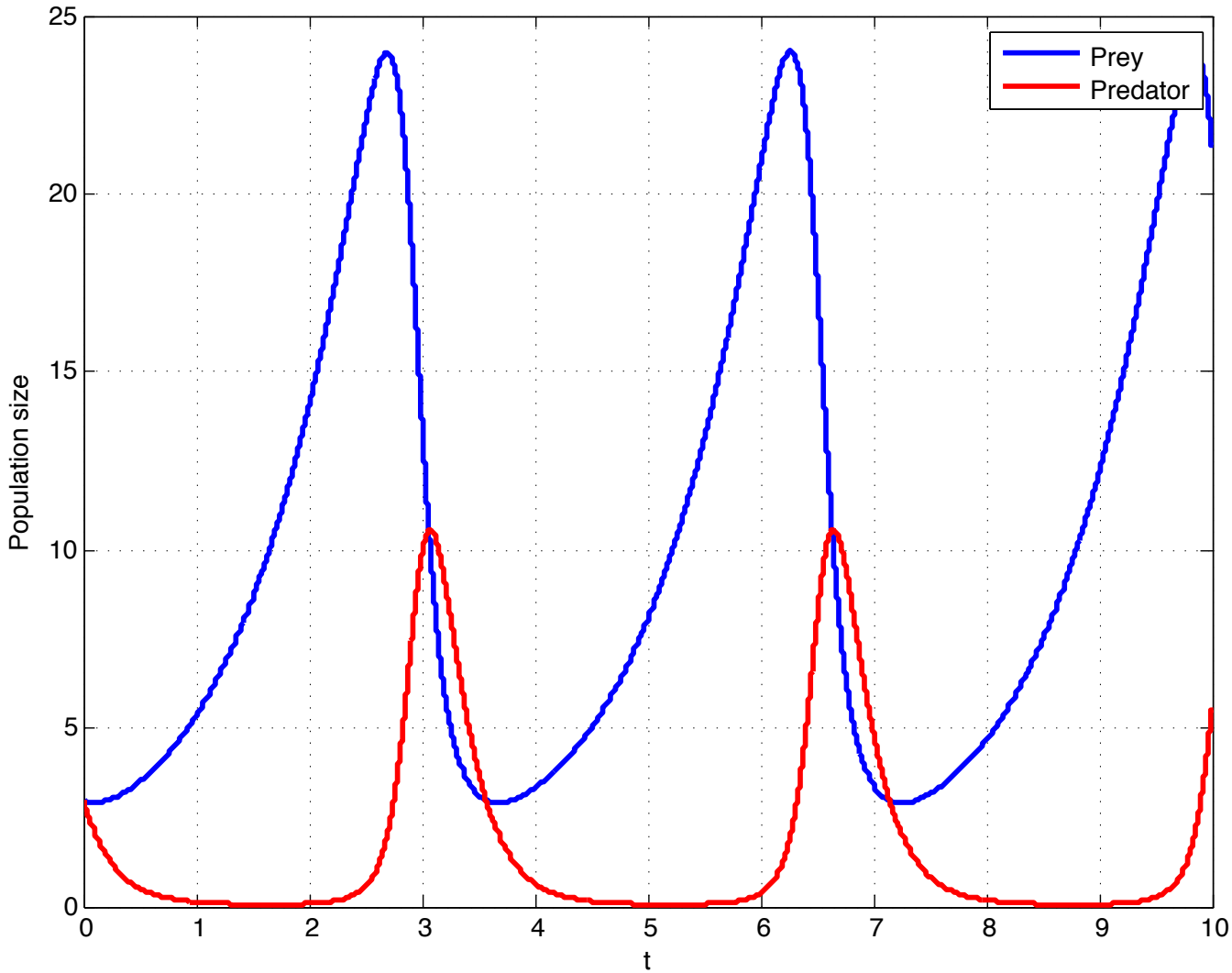
Systems of ODEs

- Solve in the exact same way as before, we just have one (or more) additional equation(s) to solve for each time step

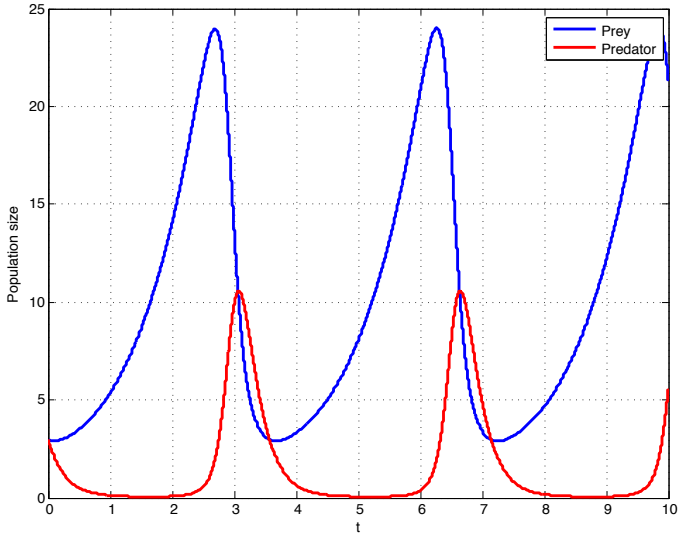
```
% ----------------------------------------------------
% User input (Note: All paramters are stored in a structure)
P.y0(1) = 3.0;   % initial prey population
P.y0(2) = 3.0;   % initial predator population
P.a= 1;     % prey pop. growth rate const.
P.b= 0.5;     % predation upon prey rate const.
P.c= 5;     % predator pop. growth rate const. (negative means loss)
P.d= 0.5;     % predator pop. growth rate const. due to prey comsumption

% Integration limits
P.t0 = 0.0;   % Start value
P.tf = 10.0;   % Finish value
P.dt = 0.001;  % time step

% ----------------------------------------------------
% +++
% use built-in ode45 to solve
[t y] = ode45('LVfunction', [P.t0:P.dt:P.tf],P.y0,[],P);

figure(1); clf;
plot(t,y(:,1)); hold on; grid on;
plot(t,y(:,2),'r');
xlabel('t');  ylabel('Population size'); legend('Prey','Predator')
figure(2); clf;         % Phase plane
plot(y(:,1), y(:,2)); hold on; grid on;
xlabel('Prey');    ylabel('Predator')
```

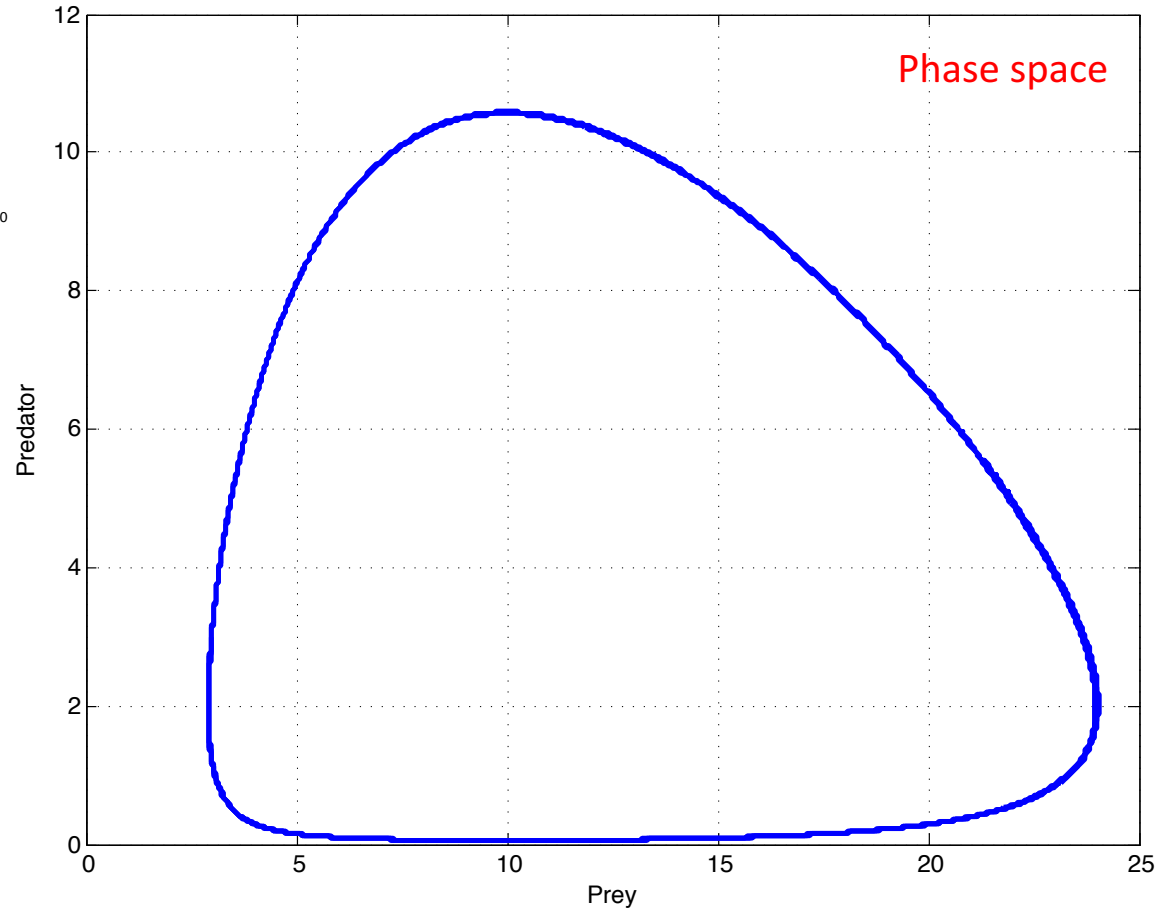
```
function [out1] = LVfunction(t,y,flag,P)
%   y(1) ... prey
%   y(2) ... predator
out1(1)= y(1)*(P.a-P.b*y(2));
out1(2)= -y(2)*(P.c-P.d*y(1));
out1= out1';
```

LVode45EX.m



LVode45EX.m

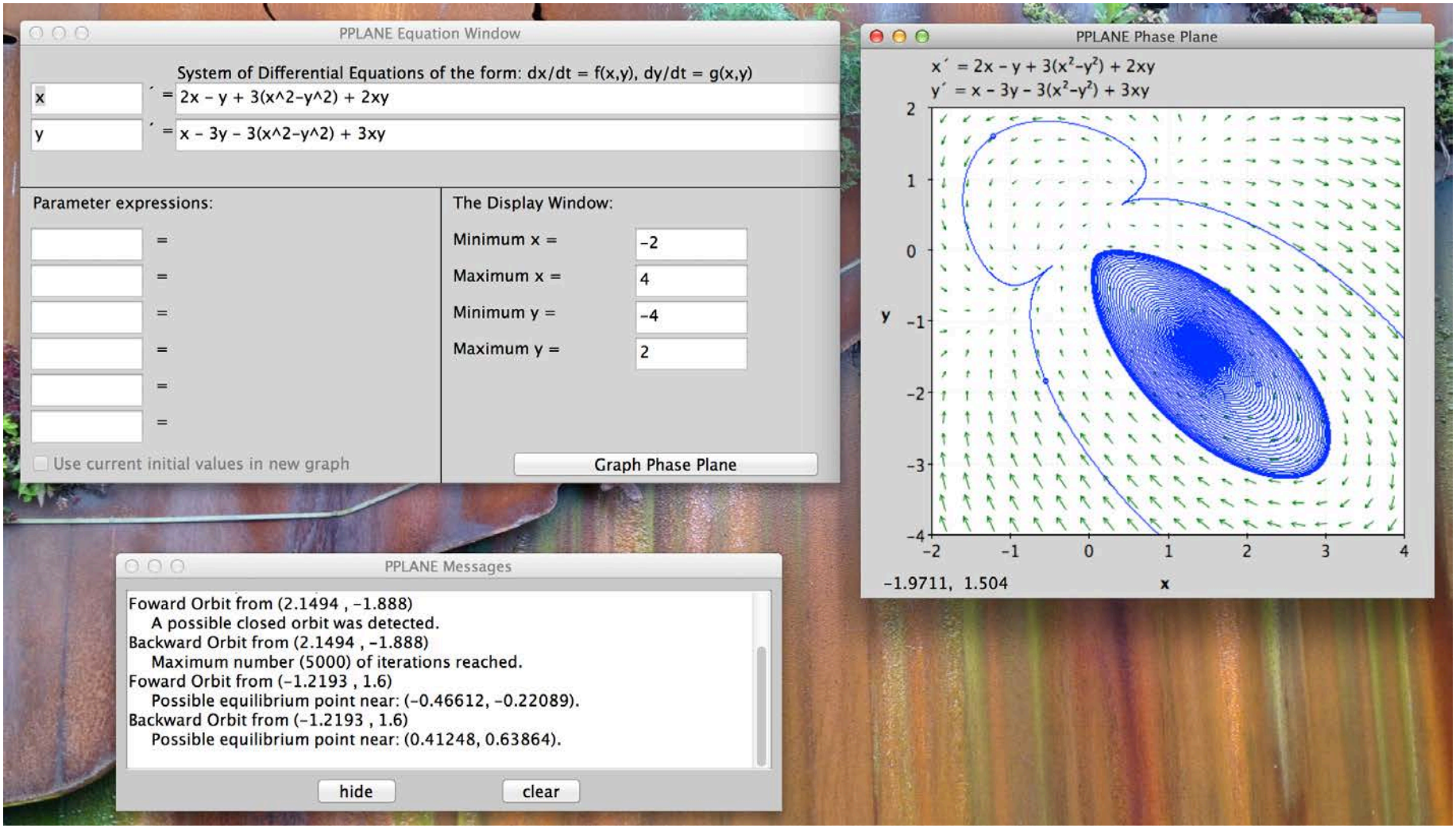

Prey
Predator
Population size
t
0
1
2
3
4
5
6
7
8
9
10
0
5
10
15
20
25


Phase space
Predator
Prey
0
5
10
15
20
25
0
2
4
6
8
10
12

Systems of ODEs

http://math.rice.edu/~dfield/dfpp.html

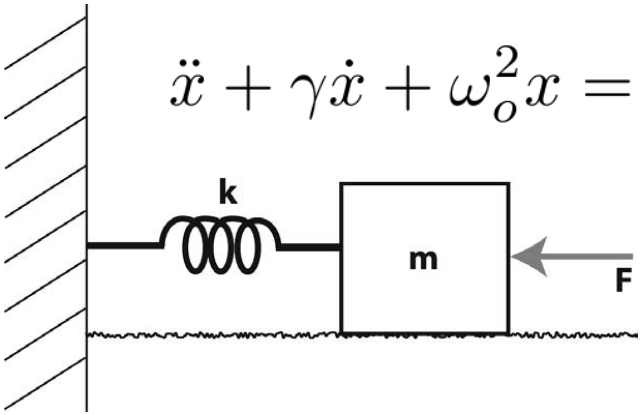
→ Phase plane analysis for (2-D) systems of ODEs



2nd order ODEs

Harmonic oscillator

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



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

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

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

- 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

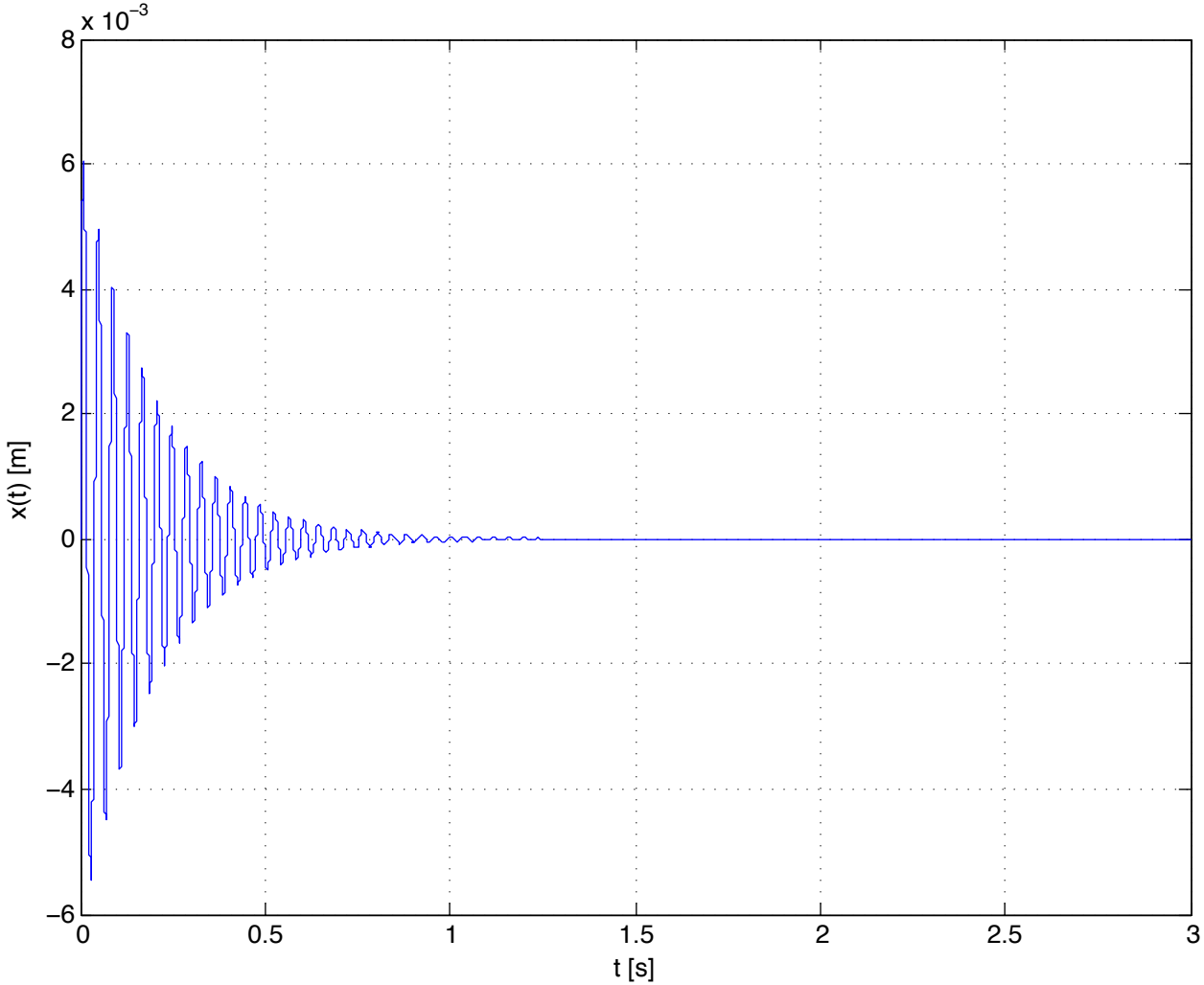
$$\frac{dx}{dt} = v$$

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

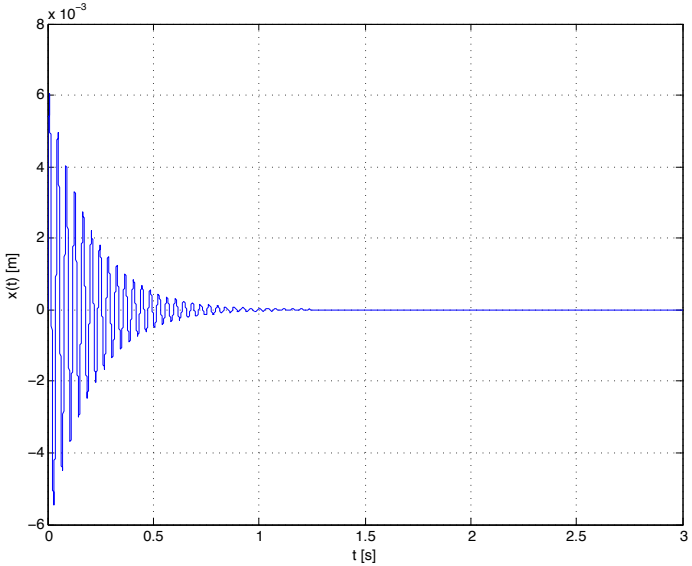
Devries (1994)

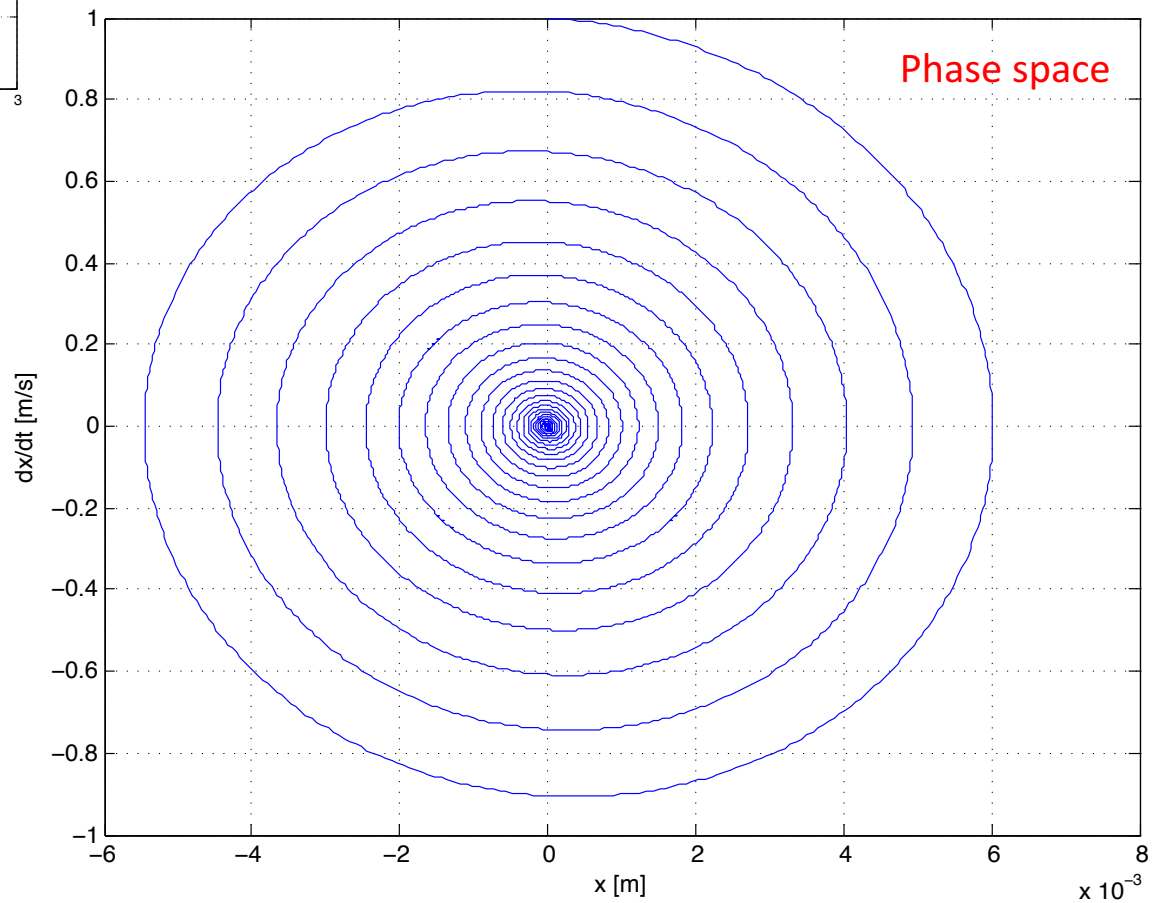
```matlab
% ### HOode45EX.m ###
% Numerically integrate the damped/driven harmonic oscillator
%   m*x''+ b*x' + k*x = A*sin(wt)
clear
% -----------------------------------------------------
% User input (Note: All paramters are stored in a structure)
P.y0(1) = 0.0;   % initial position [m]
P.y0(2) = 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]')
```

```matlab
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]

HOode45EX.m





Things to try:

- Change the damping
- Change the stiffness or mass
- Change the initial conditions
- Turn on the (non-autonomous) sinusoidal driving term
- Other types of driving terms? (e.g., an impulse)
- Other changes?

Resonance

Steady-state frequency response

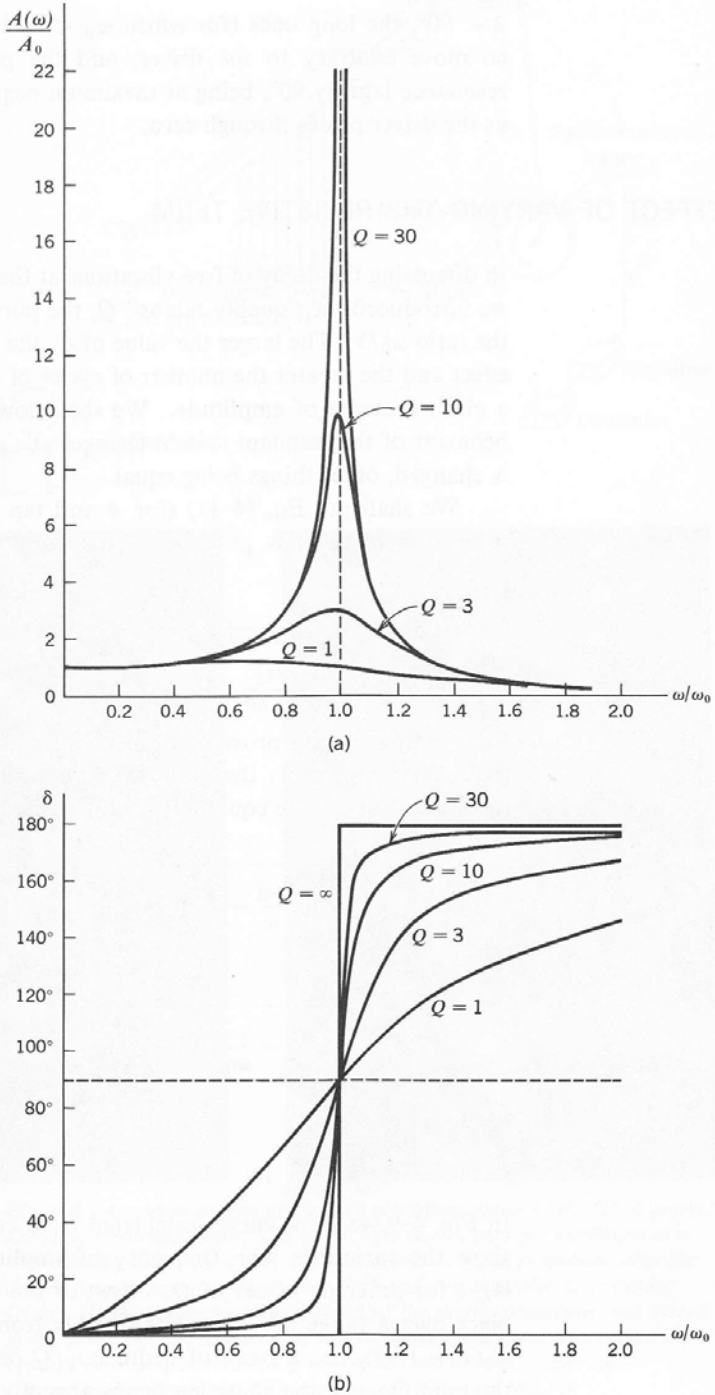


Fig. 4–9 (a) Amplitude as function of driving frequency for different values of Q, assuming driving force of constant magnitude but variable frequency. (b) Phase difference δ as function of driving frequency for different values of Q.

Consider the sinusoidally "driven" case:

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

$$Q = \omega_o / \gamma$$

Q is the 'quality factor'

→ Think about how you would go about making these plots…

French (1971)

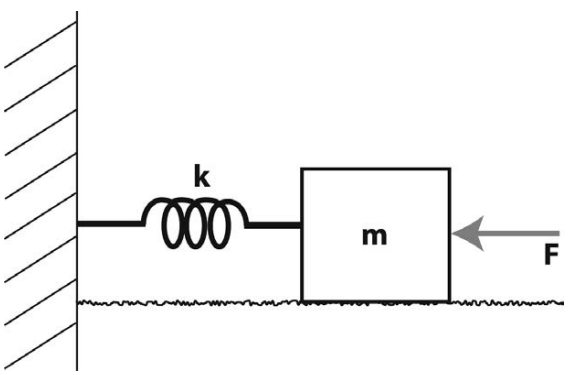
Looking ahead (re spectral analysis)

"*Impulse response*" intuitively defined in two different (but equivalent) ways:

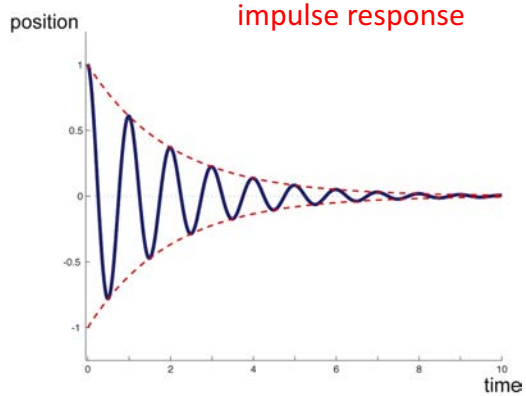
1. Time response of 'system' when subjected to an impulse
   (e.g., striking a bell w/ a hammer)

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

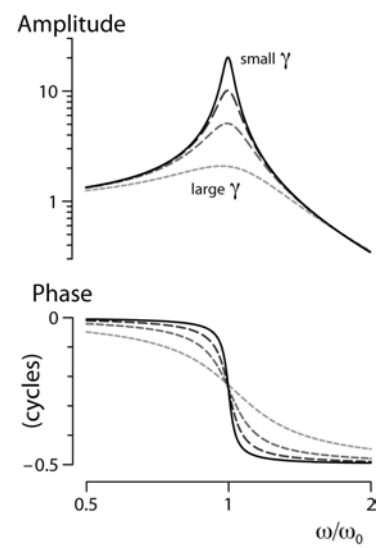
ex. Harmonic oscillator



Temporal impulse response



Spectral impulse response



⇒ Damping causes energy loss from system

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

Summary re ODEs

Stability of numeric solutions

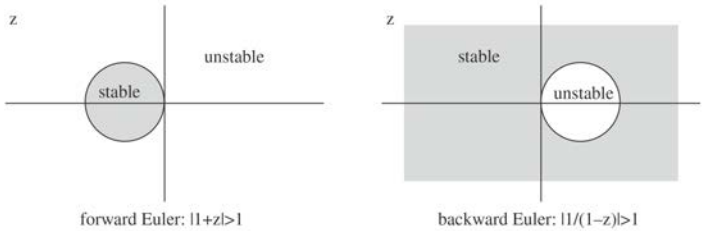


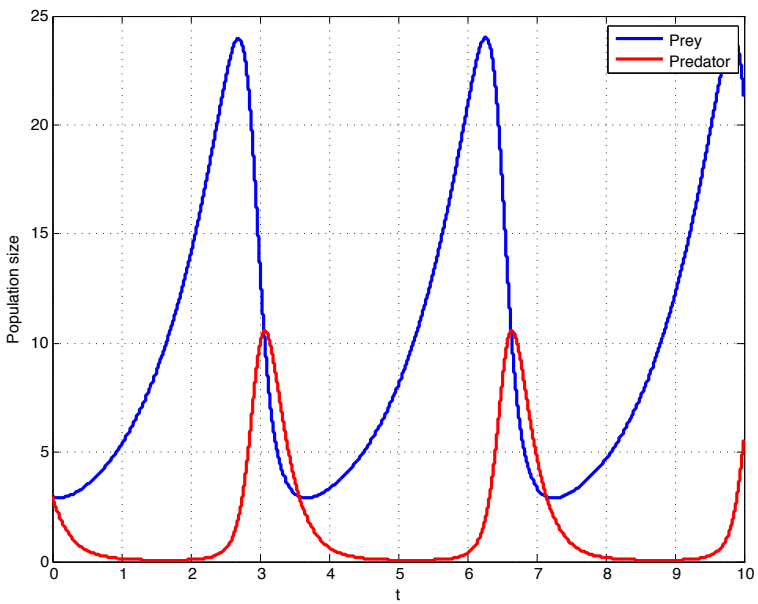
Figure 7.3: Regions for stable stepping (shaded) for the forward Euler and backward Euler schemes. The criteria for instability are also given for each stepping method.

Matlab's built-in solvers

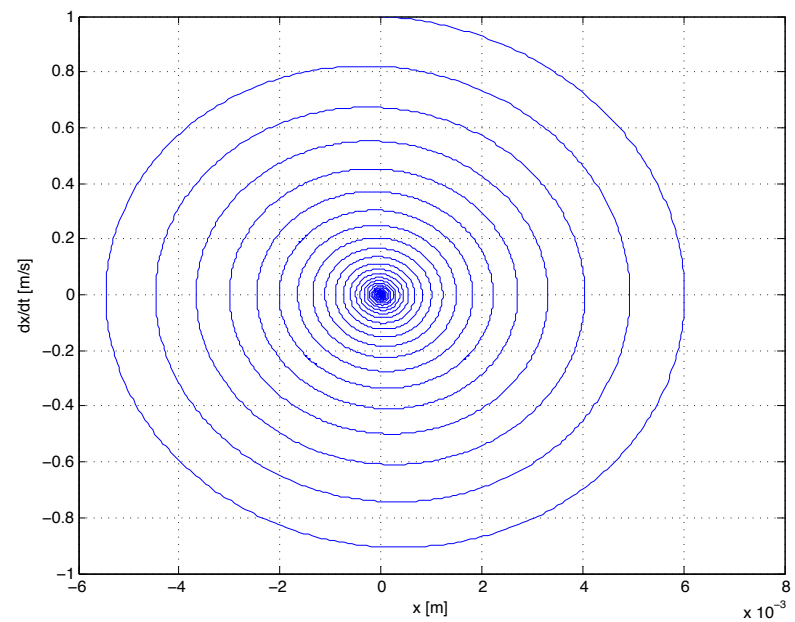
Solver	Description
`ode23`	An explicit, one-step Runge-Kutta low-order (2–3) solver. Suitable for problems that exhibit mild stiffness, problems where lower accuracy is acceptable, or problems where $f(t, y)$ is not smooth (e.g., discontinuous).
`ode45`	An explicit, one-step Runge-Kutta medium-order (4–5) solver. Suitable for nonstiff problems that require moderate accuracy. *This is typically the first solver to try on a new problem.*
`ode113`	A multistep Adams-Bashforth-Moulton PECE solver of varying order (1–13). Suitable for nonstiff problems that require moderate to high accuracy involving problems where $f(t, y)$ is expensive to compute. Not suitable for problems where $f(t, y)$ is not smooth (i.e., where it is discontinuous or has discontinuous lower-order derivatives).
`ode23s`	An implicit, one-step modified Rosenbrock solver of order 2. Suitable for stiff problems where lower accuracy is acceptable, or where $f(t, y)$ is discontinuous. *Stiff problems are generally described as problems where the underlying time constants vary by several orders of magnitude or more.*
`ode15s`	An implicit, multistep numerical differentiation solver of varying order (1–5). Suitable for stiff problems that require moderate accuracy. *This is typically the solver to try if* `ode45` *fails or is too inefficient.*

Warning: Beware the black box!

Systems of ODEs



2nd order ODEs



Exercises

- Write a code to explicitly implement RK4 for the harmonic oscillator

- Modify the harmonic oscillator code to simulate the van der Pol system

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

- Write a code to solve the gravity-driven pendulum: $$\frac{d^2\theta}{dt^2} = -\sqrt{g/\ell}\sin(\theta)$$

- <u>Challenge</u>: Consider the 1-D time-independent Schrodinger equation for an electron bound in an anharmonic potential:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x) = E\psi(x) \qquad V(x) = \alpha x^2 + \beta x^4$$

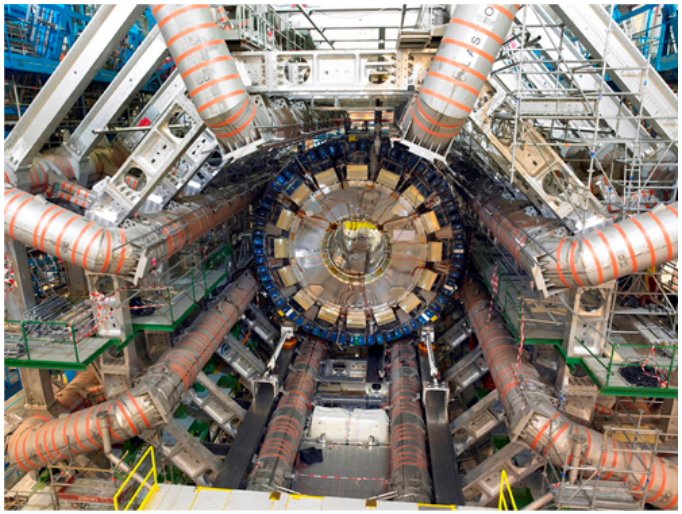
What are the initial conditions? Or do you instead have a better handle on the 'boundary conditions'?

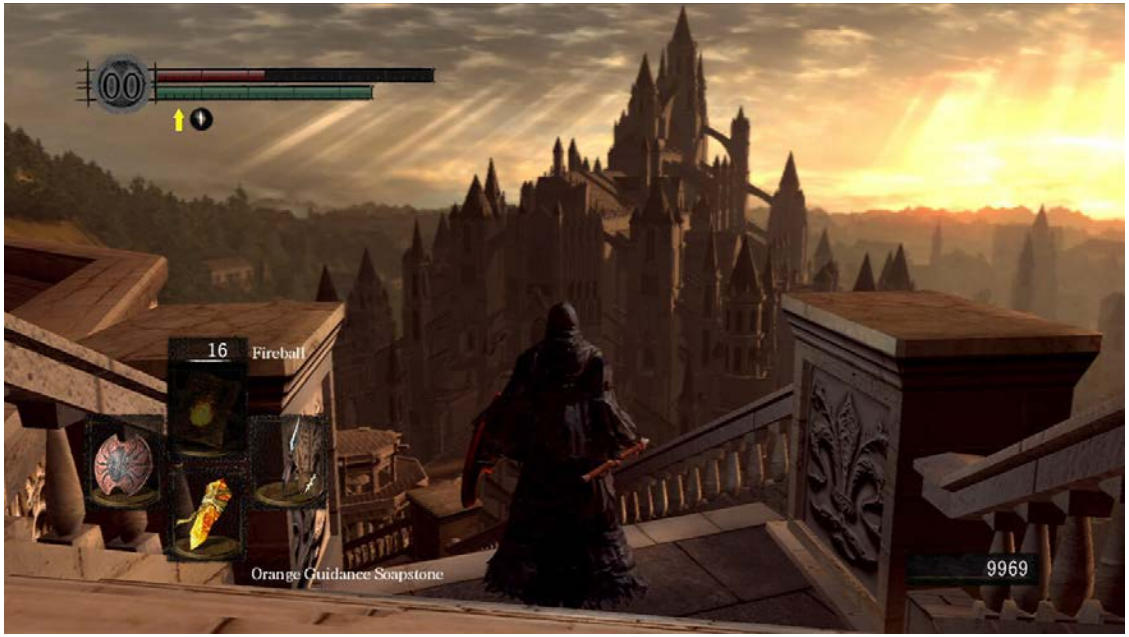
“Data Acquisition”

- We “interact” w/ various classes of devices in a variety of ways

- In essence, these devices “acquire” “data” from the external world and convert it to (???)....

ATLAS detector (CERN)





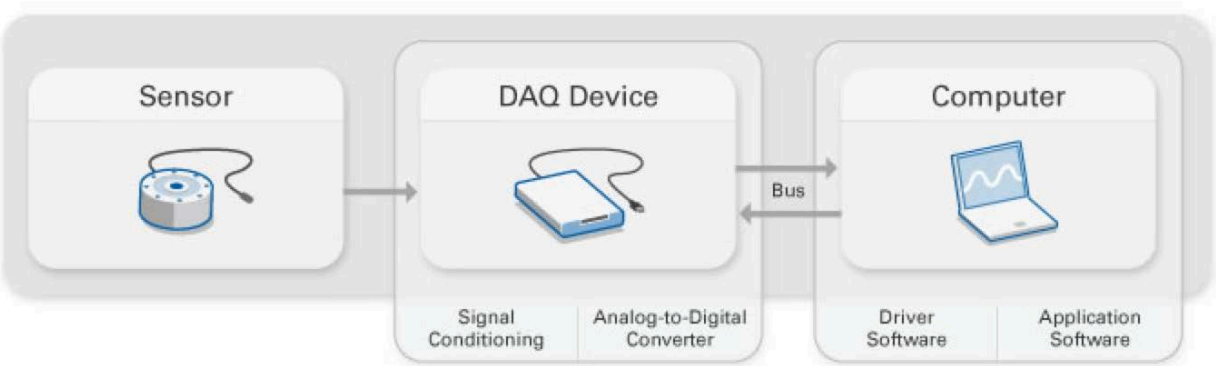
DAQ 101

- What is 'DAQ'? **Data AcQuisition**

- Basically, in a nutshell, using a computer (usually in a programmable fashion) to collect 'data'

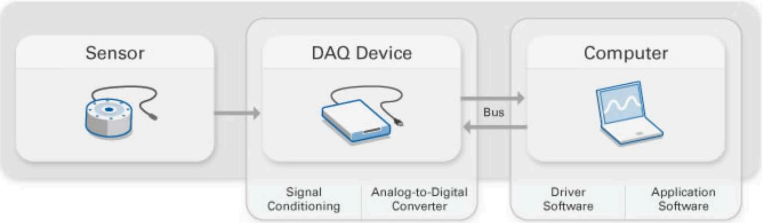
- Broadly interpreted, DAQ is a very common thing we do in the modern world. e.g.,
  - Taking a picture with a cell phone (i.e., Charged Coupled Device, or **CCD**)
  - ATLAS detector at CERN
  - Many (all?) of the measurement systems on Rosetta's landing module Philae
  - Using a computer to record your voice for a spectrogram
  - Force transducer to measure stiffness of a material
  - Voltmeter (incl. using a thermocouple to measure temperature)
  - Use an Arduino as a Theramin
  - .... (the list could go on and on)

- Most basic pieces:



http://www.ni.com/data-acquisition/what-is/

DAQ 101



What Is a Sensor?

The measurement of a physical phenomenon, such as the temperature of a room, the intensity of a light source, or the force applied to an object, begins with a sensor. A sensor, also called a transducer, converts a physical phenomenon into a measurable electrical signal. Depending on the type of sensor, its electrical output can be a voltage, current, resistance, or another electrical attribute that varies over time. Some sensors may require additional components and circuitry to properly produce a signal that can accurately and safely be read by a DAQ device.

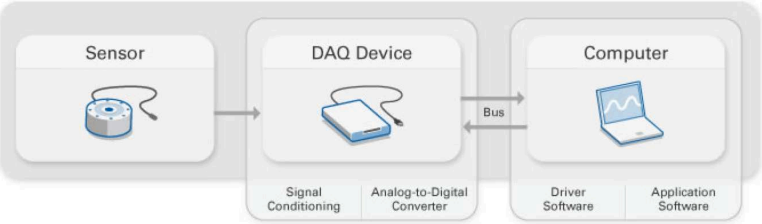
Learn the fundamentals of how common sensors work

Common Sensors

Sensor	Phenomenon
Thermocouple, RTD, Thermistor	Temperature
Photo Sensor	Light
Microphone	Sound
Strain Gage, Piezoelectric Transducer	Force and Pressure
Potentiometer, LVDT, Optical Encoder	Position and Displacement
Accelerometer	Acceleration
pH Electrode	pH

http://www.ni.com/data-acquisition/what-is/

DAQ 101



What Is a DAQ Device?

DAQ hardware acts as the interface between a computer and signals from the outside world. It primarily functions as a device that digitizes incoming analog signals so that a computer can interpret them. The three key components of a DAQ device used for measuring a signal are the signal conditioning circuitry, analog-to-digital converter (ADC), and computer bus. Many DAQ devices include other functions for automating measurement systems and processes. For example, digital-to-analog converters (DACs) output analog signals, digital I/O lines input and output digital signals, and counter/timers count and generate digital pulses.

Review five questions to ask when choosing your DAQ device

Key Measurement Components of a DAQ Device

Signal Conditioning

Signals from sensors or the outside world can be noisy or too dangerous to measure directly. Signal conditioning circuitry manipulates a signal into a form that is suitable for input into an ADC. This circuitry can include amplification, attenuation, filtering, and isolation. Some DAQ devices include built-in signal conditioning designed for measuring specific types of sensors.

Learn more about the different types of signal conditioning

Analog-to-Digital Converter (ADC)

Analog signals from sensors must be converted into digital before they are manipulated by digital equipment such as a computer. An ADC is a chip that provides a digital representation of an analog signal at an instant in time. In practice, analog signals continuously vary over time and an ADC takes periodic "samples" of the signal at a predefined rate. These samples are transferred to a computer over a computer bus where the original signal is reconstructed from the samples in software.

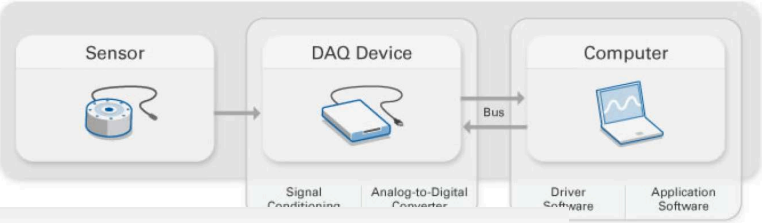
Learn the basics of analog sampling

Computer Bus

DAQ devices connect to a computer through a slot or port. The computer bus serves as the communication interface between the DAQ device and computer for passing instructions and measured data. DAQ devices are offered on the most common computer buses including USB, PCI, PCI Express, and Ethernet. More recently, DAQ devices have become available for 802.11 Wi-Fi for wireless communication. There are many types of buses, and each offers different advantages for different types of applications.

http://www.ni.com/data-acquisition/what-is/

DAQ 101



What Is a Computer's Role in a DAQ System?

A computer with programmable software controls the operation of the DAQ device and is used for processing, visualizing, and storing measurement data. Different types of computers are used in different types of applications. A desktop may be used in a lab for its processing power, a laptop may be used in the field for its portability, or an industrial computer may be used in a manufacturing plant for its ruggedness.

Read more about how to choose the right computer for your application

What Are the Different Software Components in a DAQ System?

Driver Software

Driver software provides application software the ability to interact with a DAQ device. It simplifies communication with the DAQ device by abstracting low-level hardware commands and register-level programming. Typically, DAQ driver software exposes an application programming interface (API) that is used within a programming environment to build application software.

Read the important things you should consider when evaluating driver software

Application Software

Application software facilitates the interaction between the computer and user for acquiring, analyzing, and presenting measurement data. It is either a prebuilt application with predefined functionality, or a programming environment for building applications with custom functionality. Custom applications are often used to automate multiple functions of a DAQ device, perform signal-processing algorithms, and display custom user interfaces.

- Matlab can be used for DAQ, but there are a lot of software options (e.g., LabView, C, ...)

Note: Our focus in 2030 is on general computational methods (i.e., not software syntax specific)

http://www.ni.com/data-acquisition/what-is/

A/D

- Digitizing a signal requires sampling & converting from '***A***nalog' to '***D***igital' (A/D, or ADC)

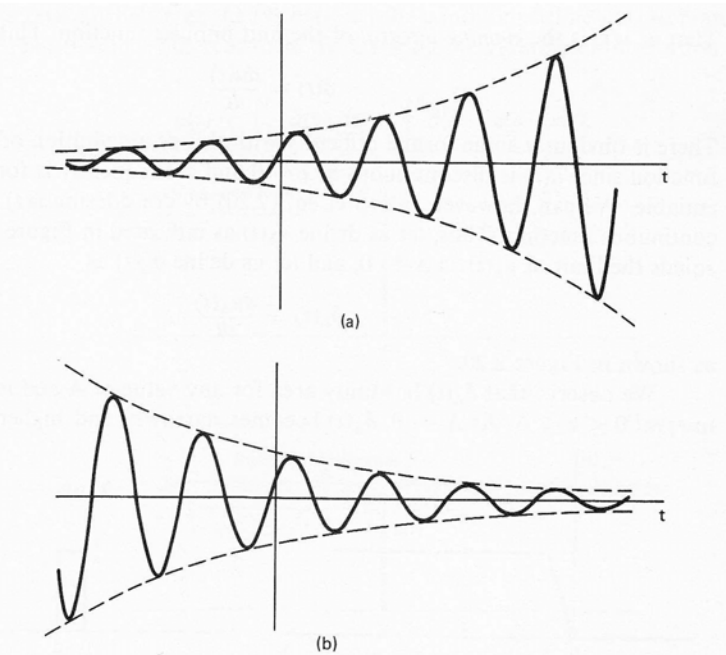


Figure 2.17 (a) Growing sinusoidal signal $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r > 0$; (b) decaying sinusoid $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r < 0$.

Continuous signal (analog)

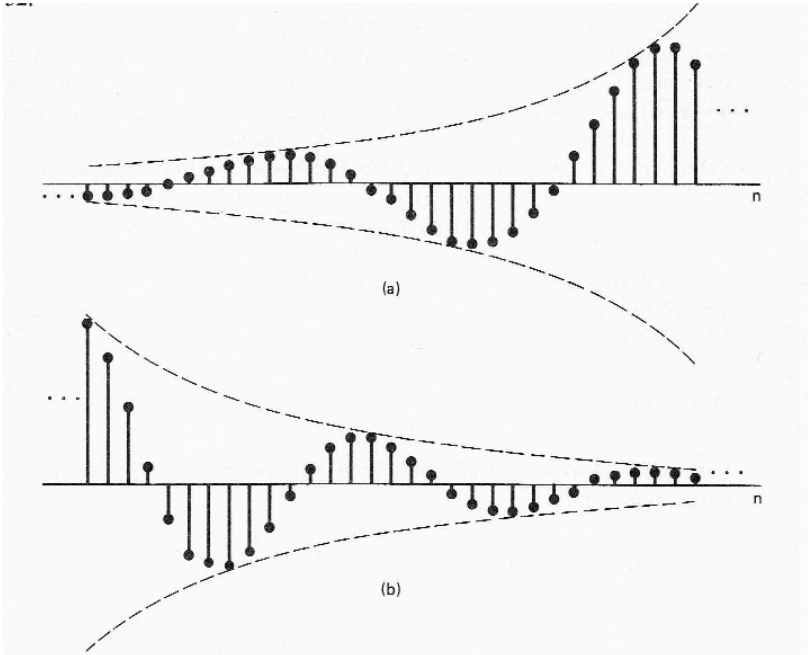


Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.

Discretized signal (digital)

- As such, there is some sample rate (SR) and bit-depth (more on this in a bit, pun!)

- Door swings both ways: Digital going 'out' requires D/A (or DCA)

Oppenheim & Willsky