# **Computational Methods** (PHYS 2030)

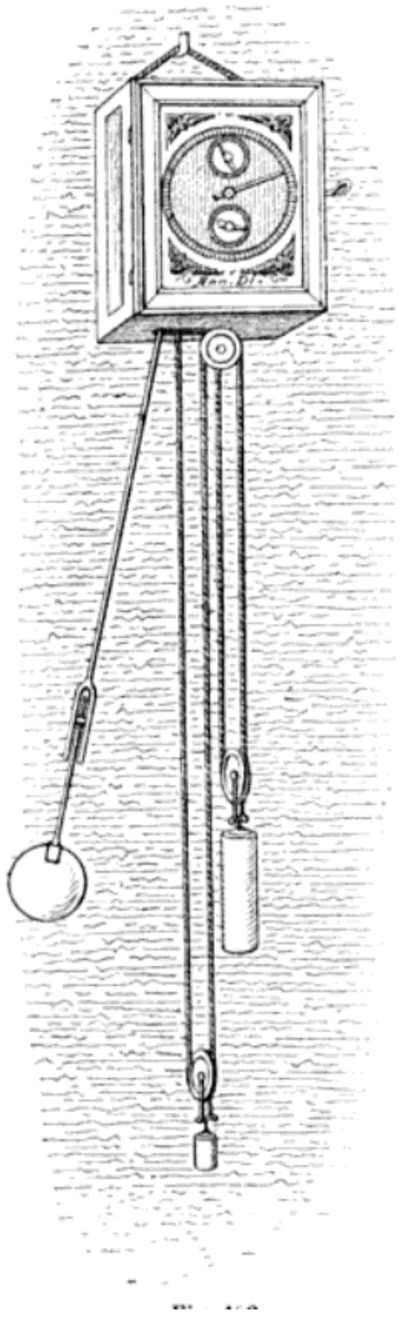
Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

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

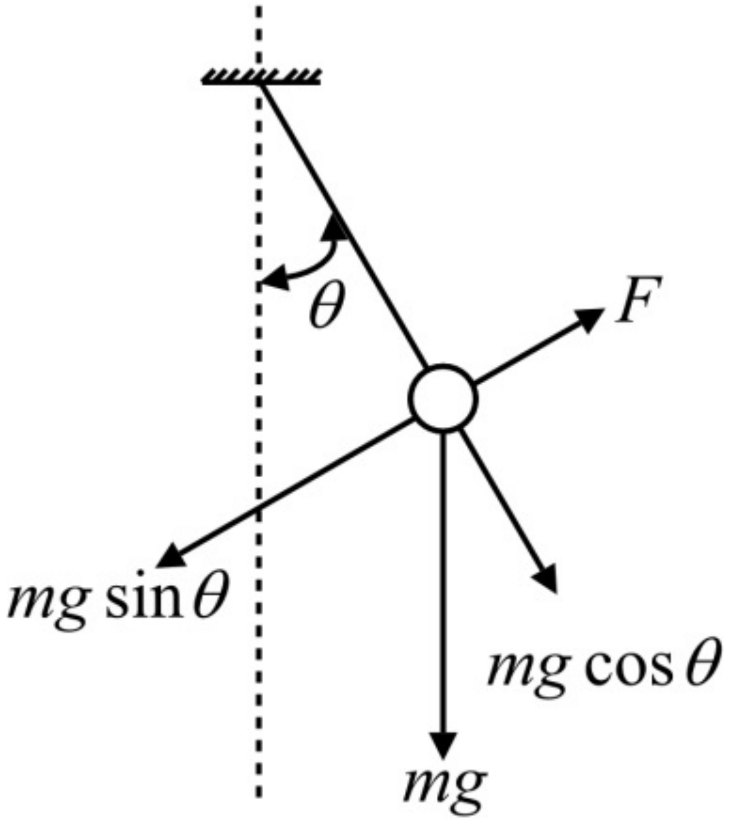
Website: http://www.yorku.ca/cberge/2030W2018.html

York University
Winter 2018
Lecture 11

## Motivation: Nonlinear oscillators



"first pendulum clock" re
Christiaan Huygens

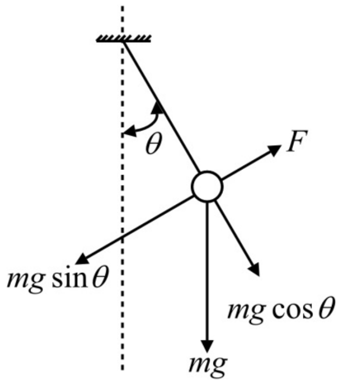


## Motivation: Nonlinear oscillators

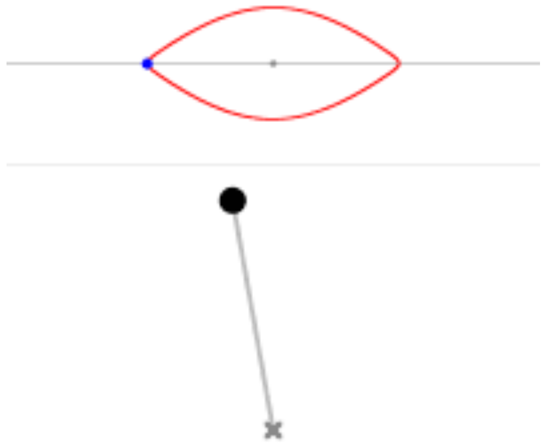
Gravity-driven pendulum

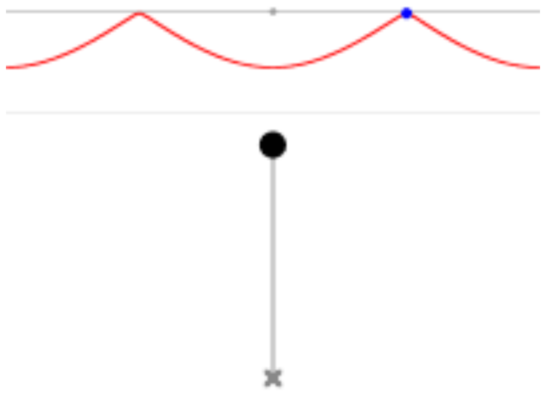
$$\frac{d^2\theta}{dt^2} = \ddot{\theta} = -\frac{g}{\ell}\sin(\theta)$$

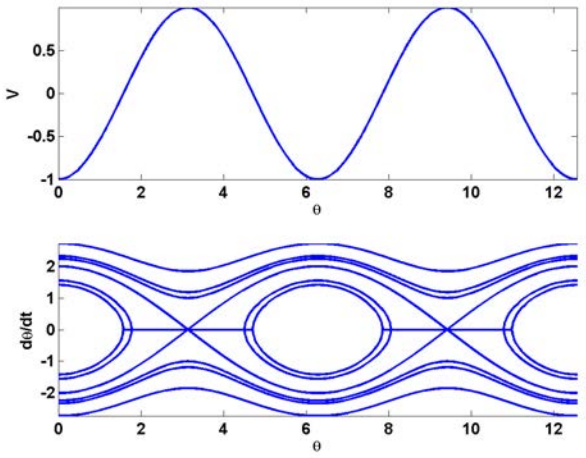
Eqn. of motion
(no damping)



Phase space







Consider that there are two equilibria w/ differing stability....

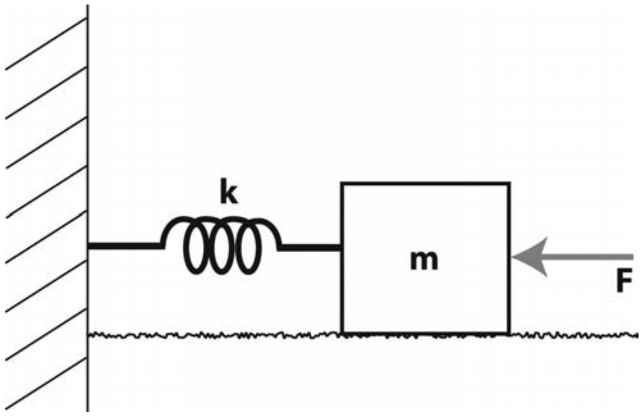
wikipedia - **Pendulum (mathematics)**

## Motivation: Nonlinear oscillators

van der Pol oscillator

- Nonlinear version of a harmonic oscillator

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



- Originally proposed to study cardiac dynamics and vacuum tubes

- Nonlinear and exhibits relatively complex behavior, thus has proven a popular model for study in mathematics, physics, and biology

- Physically, how does this differ from a linear damped harmonic oscillator?

Small displacements → Negative damping
(i.e., non-conservative system)

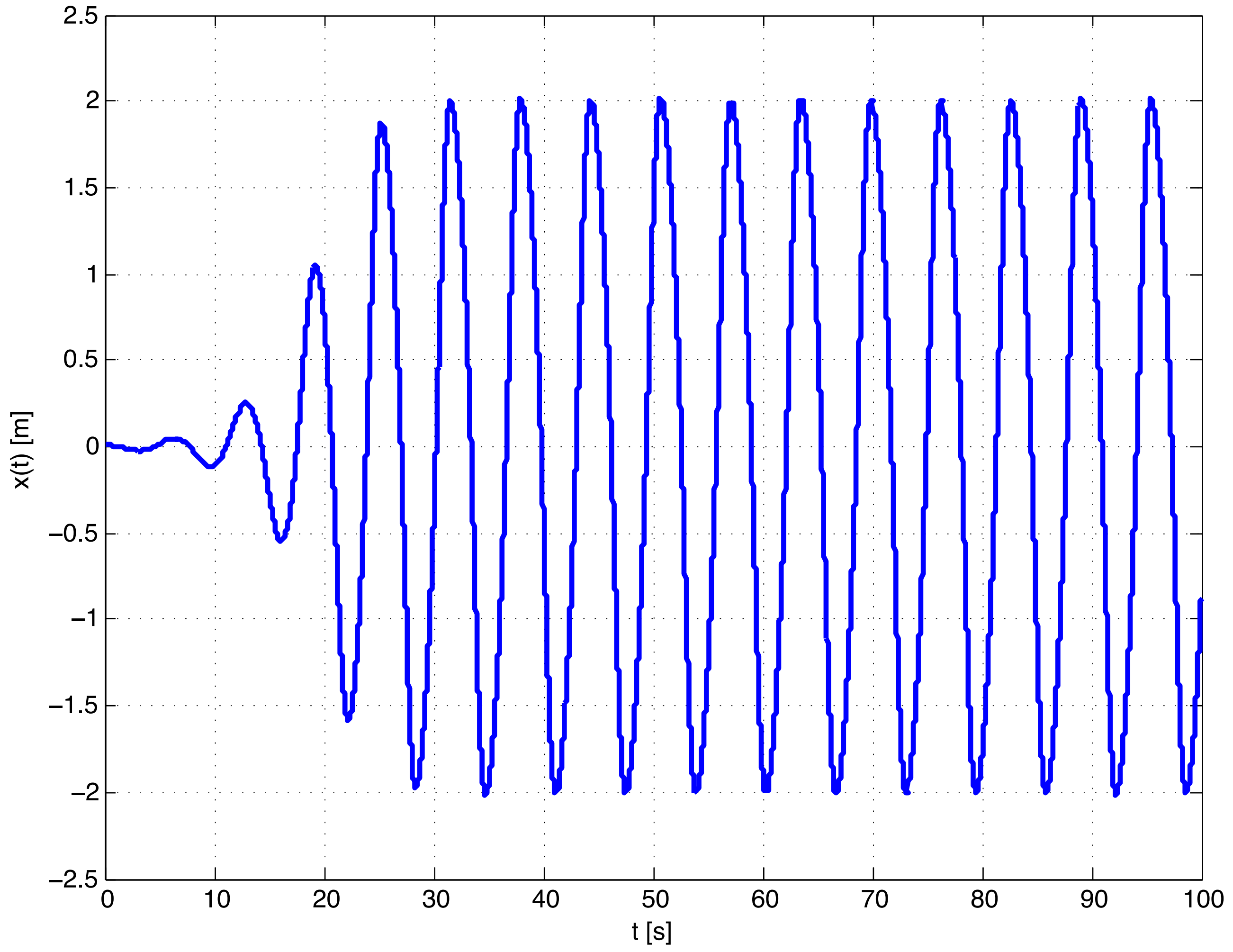
Limit cycles

VDPode45EX.m

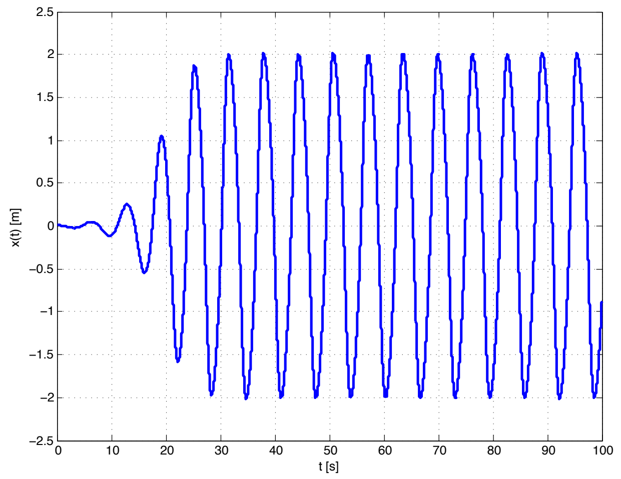
```
function [out1] = VDPfunction(t,y,flag,P)
% --------------------------------------------------
%   y(1) ... position x
%   y(2) ... velocity dx/dt
out1(1)= y(2);
out1(2)= (P.mu/P.m)*(1-y(1)^2)*y(2) - (P.k/P.m)*y(1) + (P.A/P.m)*sin(P.wr*t);
out1= out1';     % wants output as a column vector
```

## Limit cycles

initial non-zero displacement

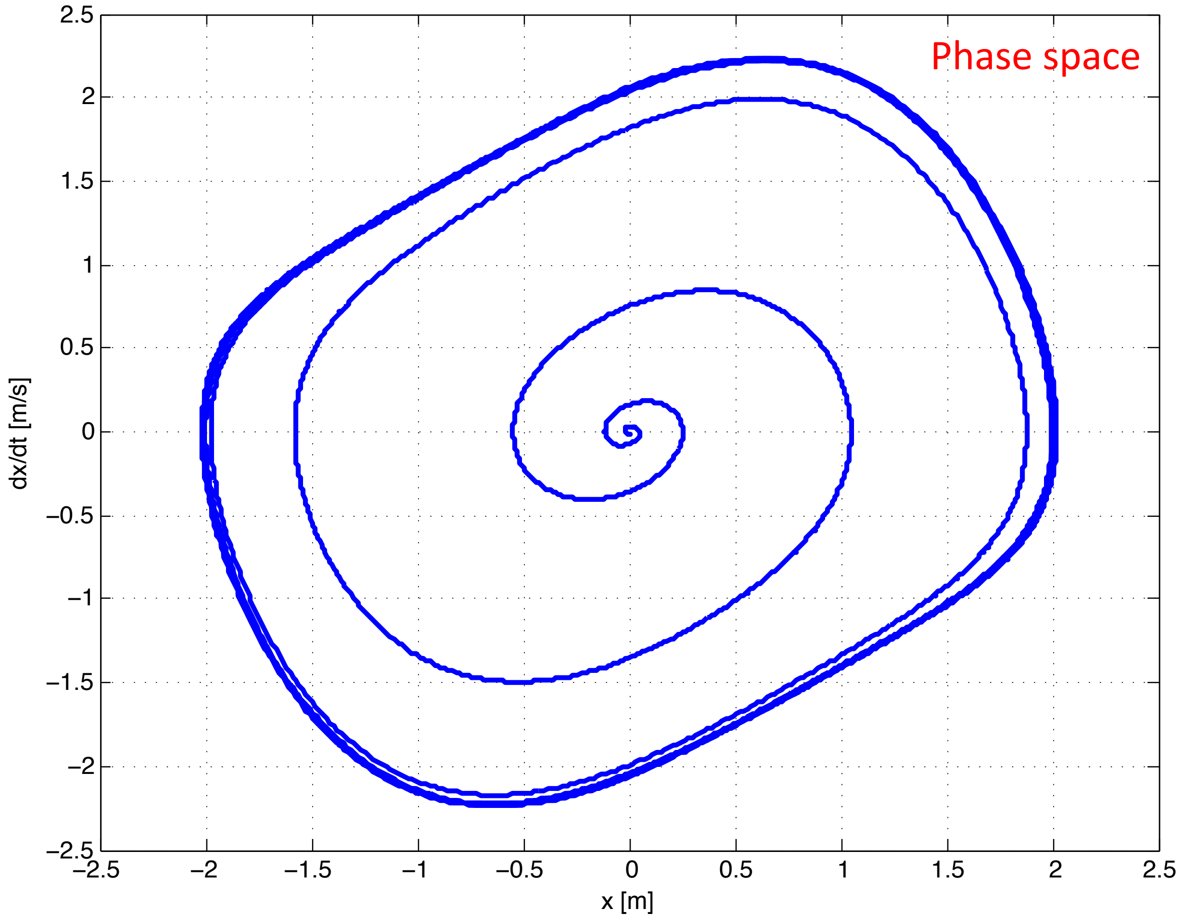


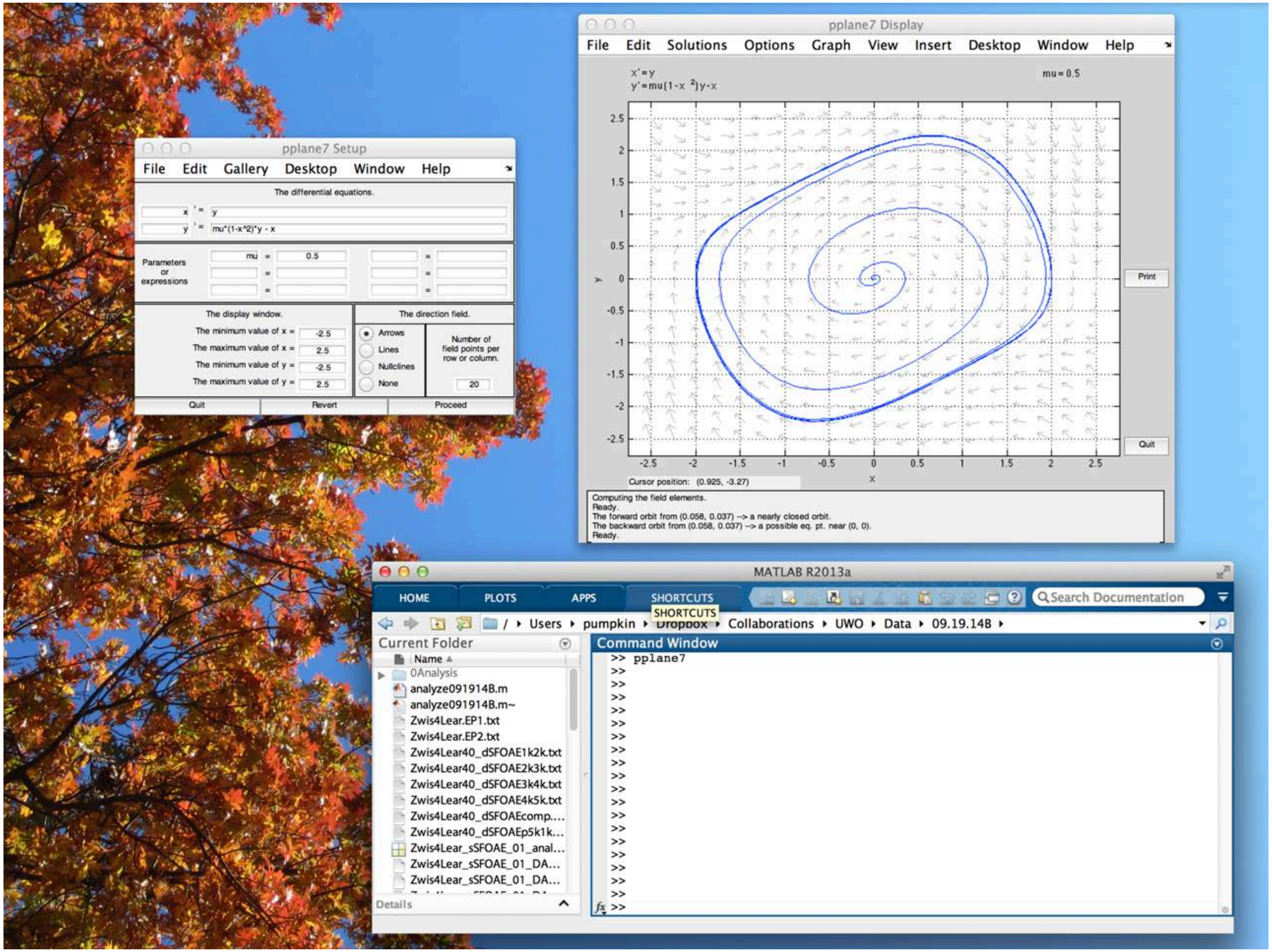
VDPode45EX.m



## Limit cycles

→ Even though there is damping, the system oscillates by itself in a stable fashion




pplane7 Setup
File Edit Gallery Desktop Window Help
pplane7 Display
File Edit Solutions Options Graph View Insert Desktop Window Help
Print
Quit
MATLAB R2013a
HOME PLOTS APPS SHORTCUTS
Search Documentation
Current Folder
Command Window
Details

pplane7 Setup

The differential equations.

x ' = y

y ' = mu*(1-x^2)*y - x

Parameters or expressions

mu = 0.5

The display window.

The minimum value of x = -2.5

The maximum value of x = 2.5

The minimum value of y = -2.5

The maximum value of y = 2.5

The direction field.

(● Arrows) Lines Nullclines None

Number of field points per row or column. 20

Quit Revert Proceed

pplane7 Display

x' = y

y' = mu(1-x $^2$)y-x

mu = 0.5

Cursor position: (0.925, -3.27)

```
Computing the field elements.
Ready.
The forward orbit from (0.058, 0.037) --> a nearly closed orbit.
The backward orbit from (0.058, 0.037) --> a possible eq. pt. near (0, 0).
Ready.
```

/ ▸ Users ▸ pumpkin ▸ Dropbox ▸ Collaborations ▸ UWO ▸ Data ▸ 09.19.14B ▸

Current Folder

```
Name
0Analysis
analyze091914B.m
analyze091914B.m~
Zwis4Lear.EP1.txt
Zwis4Lear.EP2.txt
Zwis4Lear40_dSFOAE1k2k.txt
Zwis4Lear40_dSFOAE2k3k.txt
Zwis4Lear40_dSFOAE3k4k.txt
Zwis4Lear40_dSFOAE4k5k.txt
Zwis4Lear40_dSFOAEcomp....
Zwis4Lear40_dSFOAEp5k1k...
Zwis4Lear_sSFOAE_01_anal...
Zwis4Lear_sSFOAE_01_DA...
Zwis4Lear_sSFOAE_01_DA...
```

Command Window

```
>> pplane7
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
```

→ Aside from our ode45 code, can also use pplane to explore van der Pol system

## Aside: Boundary value problems (BVPs)

- We have chiefly been considering 'initial value problems' (IVPs) up to this point, where a set of initial conditions are known

```
P.y0(1) = 0.0;    % initial position [m]
P.y0(2) = 1.0;    % initial velocity [m/s]
```

- BVPs are problems instead where values (or 'boundaries') are known for two different time points

```
P.y0(1) = 0.0;    % initial position [m]
P.yE(1) = 1.0;    % final position [m]
```

ex. General linear BVP

$$\frac{d^2y}{dt^2} = p(t)\frac{dy}{dt} + q(t)y + r(t) \qquad \begin{aligned} y(a) &= \alpha \\ y(b) &= \beta \end{aligned}$$

- Also connected mathematically to *Sturm–Liouville theory*

→ BVPs either require some degree of guessing (of initial conditions) or modified computational strategies to determine unique solutions (e.g., `bvp4c`)

Kutz (2013)

## Aside: Boundary value problems (BVPs)

Shooting method

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right)$$

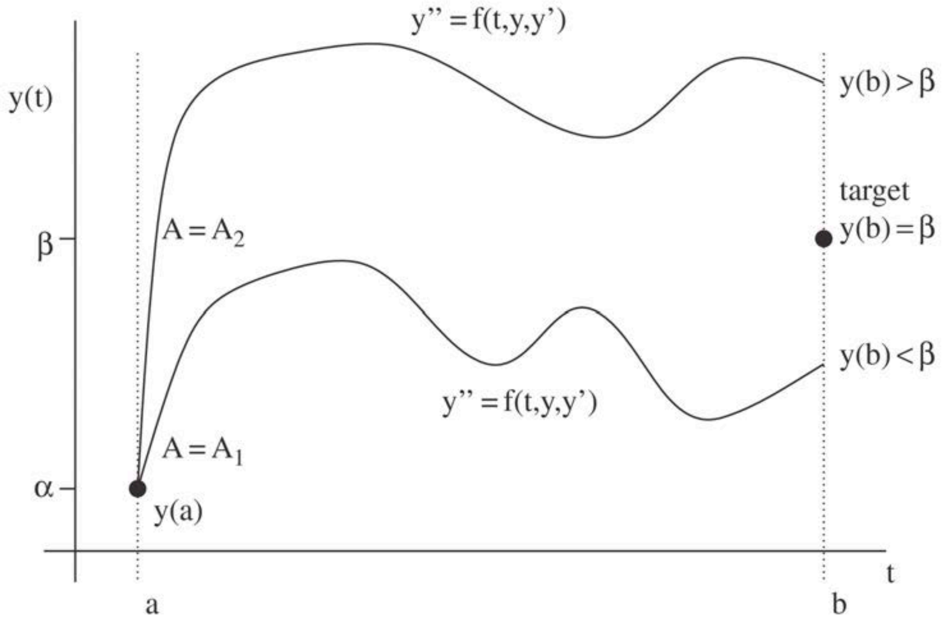
ODE to solve

$$y(a) = \alpha$$
$$y(b) = \beta$$

Boundary conditions

$$y(a) = \alpha$$
$$\frac{dy(a)}{dt} = A$$

'Guess' as to initial conditions



1 Solve the differential equation using a time-stepping scheme with the initial conditions $y(a) = \alpha$ and $y'(a) = A$.

2 Evaluate the solution $y(b)$ at $t = b$ and compare this value with the target value of $y(b) = \beta$.

3 Adjust the value of $A$ (either bigger or smaller) until a desired level of tolerance and accuracy is achieved. A bisection method for determining values of $A$, for instance, may be appropriate.

4 Once the specified accuracy has been achieved, the numerical solution is complete and is accurate to the level of the tolerance chosen and the discretization scheme used in the time-stepping.

→ 2030 will stay focused on IVPs (at least for now)

Kutz (2013)

## Linear systems analysis

- Very powerful means to study a wide class of linear systems
  - Acoustics
  - Electric circuits
  - Signal processing
  - MRI (Magnetic Resonance Imaging)
  - Neuroscience
  - Image processing and computer vision
  - etc..... (the list goes on and on)

$\rightarrow$ We will tackle numerous topics in linear systems theory throughout 2030 (e.g., Fourier transforms) and beyond

- Powerful means to study a wide class of nonlinear systems

$\rightarrow$ 'Linearizing' nonlinear systems can provide crucial insights (we'll come back to the van der Pol oscillator shortly)

## Starting Point: System of linear equations

3 (linear) equations,
3 unknowns

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_1 + 2x_2 + 4x_3 &= -1 \\ x_1 + 3x_2 + 9x_3 &= 1. \end{aligned}$$

rewrite in
matrix form

In matrix algebra form, we can rewrite this as $\mathbf{Ax} = \mathbf{b}$ with

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

Gaussian
elimination
(row reduction)

$$[\mathbf{A}|\mathbf{b}] = \left[\begin{array}{ccc|c} \underline{1} & 1 & 1 & 1 \\ 1 & 2 & 4 & -1 \\ 1 & 3 & 9 & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & -2 \\ 0 & 2 & 8 & 0 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & \underline{1} & 3 & -2 \\ 0 & 1 & 4 & 0 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

$$\begin{aligned} x_3 &= 2 & &\rightarrow & x_3 &= 2 \\ x_2 + 3x_3 &= -2 & &\rightarrow & x_2 &= -8 \\ x_1 + x_2 + x_3 &= 1 & &\rightarrow & x_1 &= 7\,. \end{aligned}$$

Kutz (2013)

## Starting Point: System of linear equations

- Can set up a computational routine to compute

1 Movement down the $N$ pivots.
2 For each pivot, perform $N$ additions/subtractions across the columns.
3 For each pivot, perform the addition/subtraction down the $N$ rows.

→ Useful URL: http://web.mit.edu/18.06/www/Course-Info/Tcodes.html

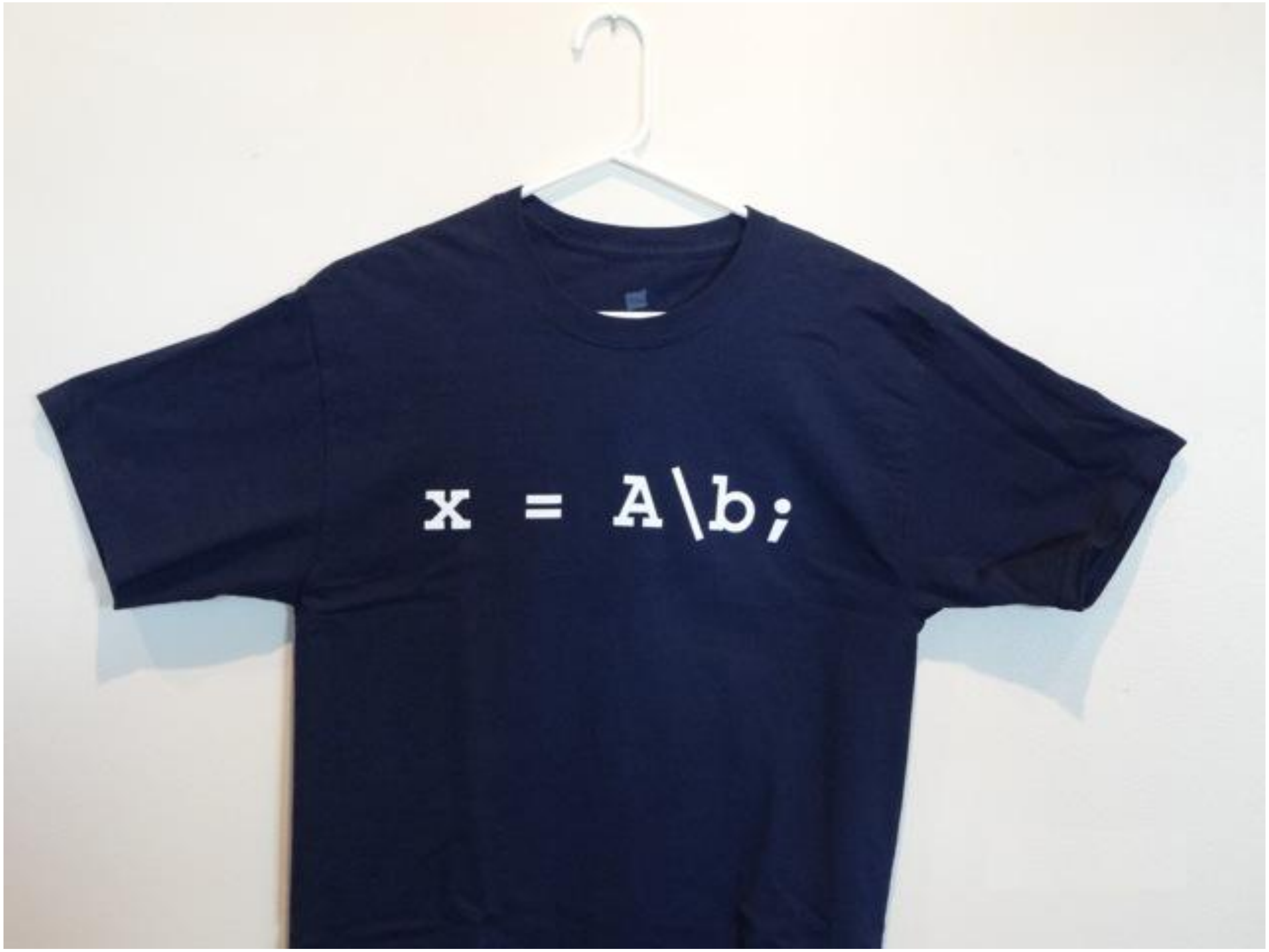
- LU ('lower upper') decomposition is more efficient

$$\mathbf{A}=\mathbf{LU} \rightarrow \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$\mathbf{Ax}=\mathbf{b} \;\rightarrow\; \mathbf{LUx}=\mathbf{b} \qquad\qquad \mathbf{Ly}=\mathbf{b} \;\text{ and }\; \mathbf{Ux}=\mathbf{y}$$

→ A bit of work to determine **L** and **U**, but easy to solve **Ax**=**b** after that

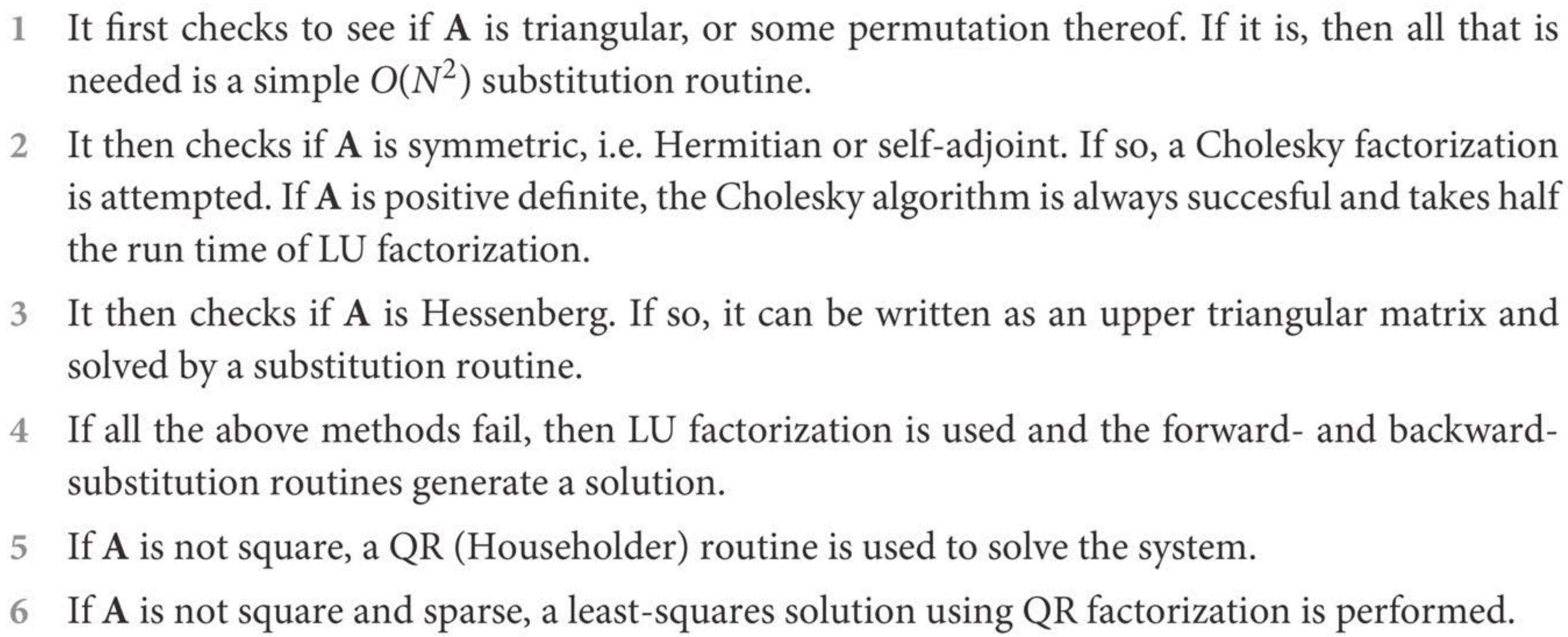
## Do you speak Matlab?



http://blogs.mathworks.com/cleve/2013/08/19/backslash/

## Matlab: Built-in optimizations to solve linear equations

➢ The Matlab command A\b actually does the following:


1 It first checks to see if **A** is triangular, or some permutation thereof. If it is, then all that is needed is a simple $O(N^2)$ substitution routine.

2 It then checks if **A** is symmetric, i.e. Hermitian or self-adjoint. If so, a Cholesky factorization is attempted. If **A** is positive definite, the Cholesky algorithm is always succesful and takes half the run time of LU factorization.

3 It then checks if **A** is Hessenberg. If so, it can be written as an upper triangular matrix and solved by a substitution routine.

4 If all the above methods fail, then LU factorization is used and the forward- and backward-substitution routines generate a solution.

5 If **A** is not square, a QR (Householder) routine is used to solve the system.

6 If **A** is not square and sparse, a least-squares solution using QR factorization is performed.

→ Matlab is fast/efficient at solving linear equations in matrix form

Kutz (2013)

## Starting Point: System of linear autonomous ODEs

- Let's consider a simple 2nd order system (all these ideas scale up for higher dimension systems)

$$\frac{dx}{dt} = ax + by$$

$$\frac{dy}{dt} = cx + dy$$

- Re-express in matrix/vector form:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}$$

- Let's make an assumption: solutions will have the form of (possibly complex) exponentials

e.g., remember what we did for the damped undriven harmonic oscillator!

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

$$x = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} c_1 e^{\lambda_1 t} + \begin{bmatrix} k_3 \\ k_4 \end{bmatrix} c_2 e^{\lambda_2 t}$$

This expression explicitly deals with the **eigenvalues** and **eigenvectors** of the system

wikipedia (phase space)

## Novel concept: Eigenvalues & Eigenvectors

Eigenvalue –
A scalar associated with a given linear transformation of a vector space and having the property that there is some nonzero vector which when multiplied by the scalar is equal to the vector obtained by letting the transformation operate on the vector; *especially* **:** a root of the characteristic equation of a matrix
[from Merriam-Webster]

- Etymology: *eigen* is German for ‘own’, ‘peculiar’
- Rich history in mathematics (Cauchy, Euler, Fourier, Hilbert, ....)
- Used in a wide variety of physics & engineering applications such as:
    - Classical mechanics (e.g., ‘principle axes’ in rigid body rotations)
    - Quantum mechanics (e.g., for solutions to Schrodinger’s equation, the eigenvalue of the wavefunction is the associated energy $E$)
    - Principal Component Analysis (PCA; we’ll come back to this later in the semester)
    - Image processing and ‘Eigenfaces’ (we’ll come back to this at the end of the lecture)
    - Harmonic oscillator (we’ll come back to this shortly)

## Eigen Decomposition

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \qquad \frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}$$

Characteristic equation:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

→ determinant (det) is scalar value associated with a square matrix

ODE as combination of eigenvalues and eigenvectors

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

'secular equation'

General solution:

$$x = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} c_1 e^{\lambda_1 t} + \begin{bmatrix} k_3 \\ k_4 \end{bmatrix} c_2 e^{\lambda_2 t}$$

→ Remember, we implicitly assume the solution has this exponential form!

## Finding eigenvalues

Characteristic equation:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

Quadratic equation w/ two roots (for a 2nd order system)

$$\lambda^2 - \lambda(a+d) + (ad - bc) = 0$$

Note that complex roots are possible

$$\lambda = \frac{(a+d) \pm \sqrt{(a+d)^2 - 4(ad-bc)}}{2}$$

$$x = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} c_1 e^{\lambda_1 t} + \begin{bmatrix} k_3 \\ k_4 \end{bmatrix} c_2 e^{\lambda_2 t}$$

$\rightarrow$ Eigenvalues explicitly tell you how the solutions behave!