Computational Methods (PHYS 2030)

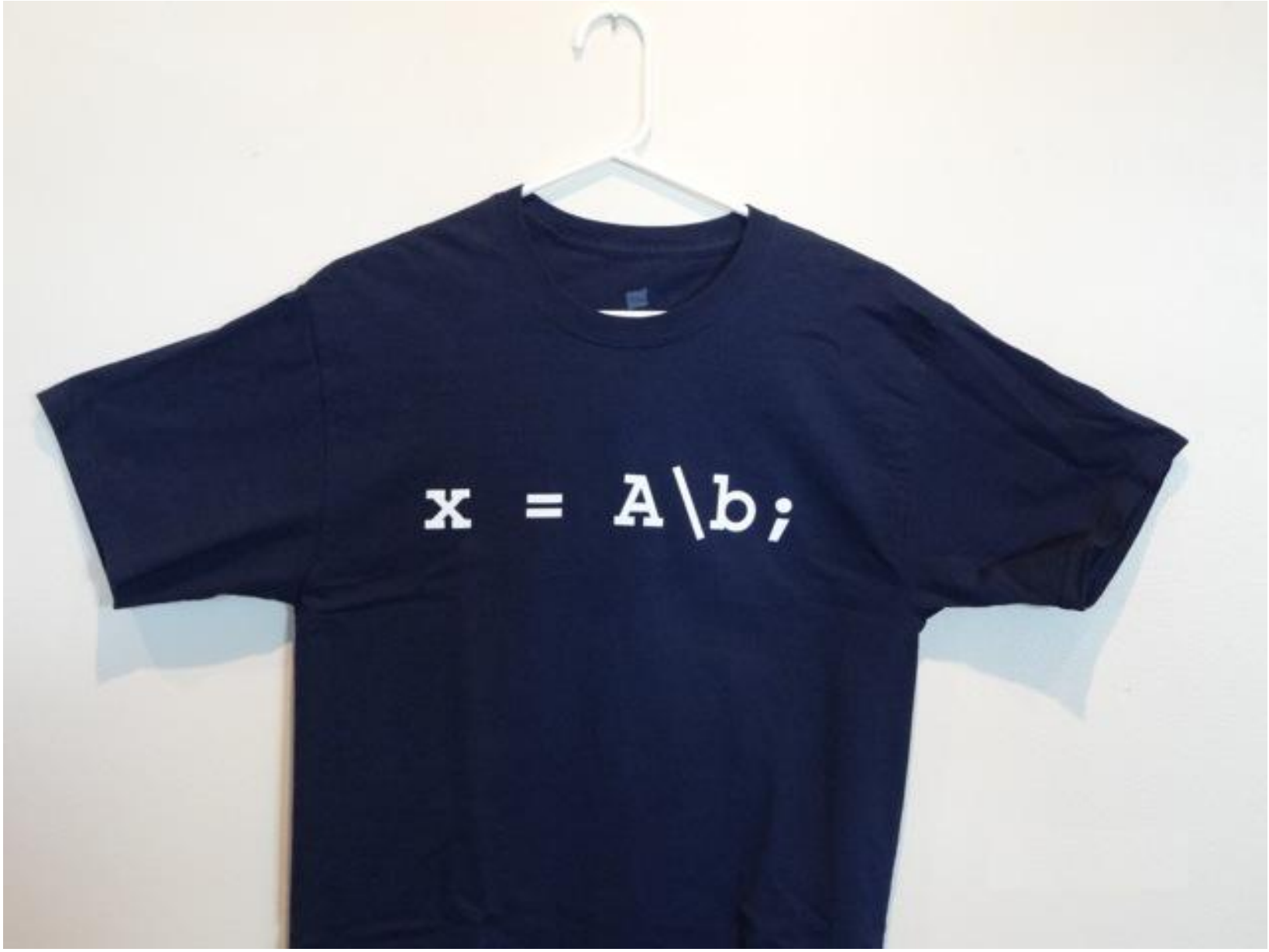
Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

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

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

York University
Winter 2018
Lecture 12

Do you speak Matlab?



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

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)

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

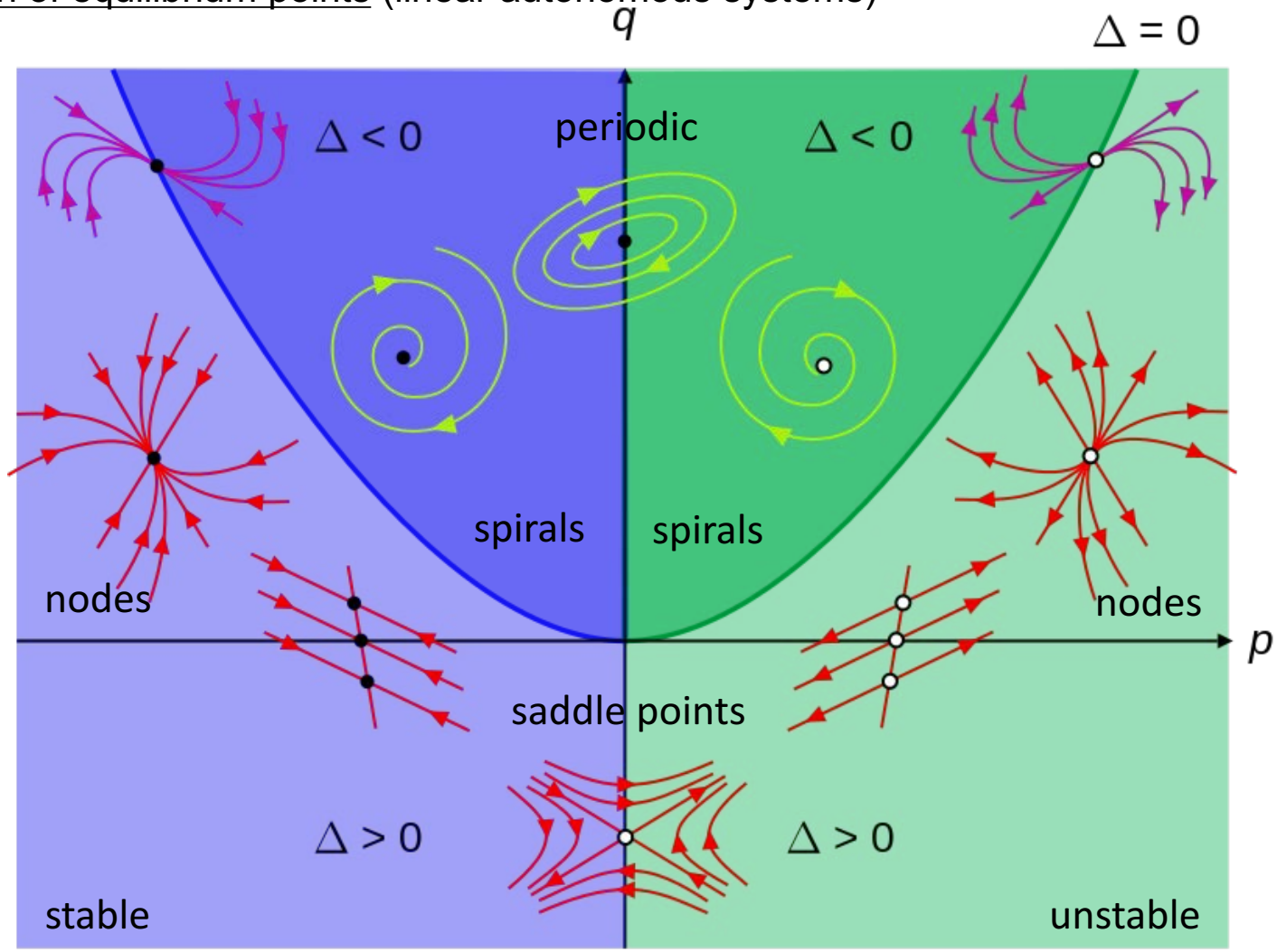
$$\lambda = \frac{(a+d) \pm \sqrt{(a+d)^2 - 4(ad-bc)}}{2}$$

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

→ Eigenvalues explicitly tell you how the solutions behave!

Classification of equilibrium points (linear autonomous systems)

Orbits



$$\frac{dx}{dt} = Ax + By \qquad p = A + D$$

$$\frac{dy}{dt} = Cx + Dy \qquad q = AD - BC$$

$$\Delta = p^2 - 4q$$

Ex.

$$\frac{dx}{dt} = 5x - 3y$$

$$\frac{dy}{dt} = 2x - 4y$$

→ Only a single equlibrium point exists (at the origin). Stability?

$$\mathbf{A} = \begin{pmatrix} 5 & -3 \\ 2 & -4 \end{pmatrix}$$

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

$$p = \mathrm{Tr}(\mathbf{A}) = 5 + (-4) = 1$$

$$q = \det(\mathbf{A}) = 5(-4) - (-3)2 = -14$$

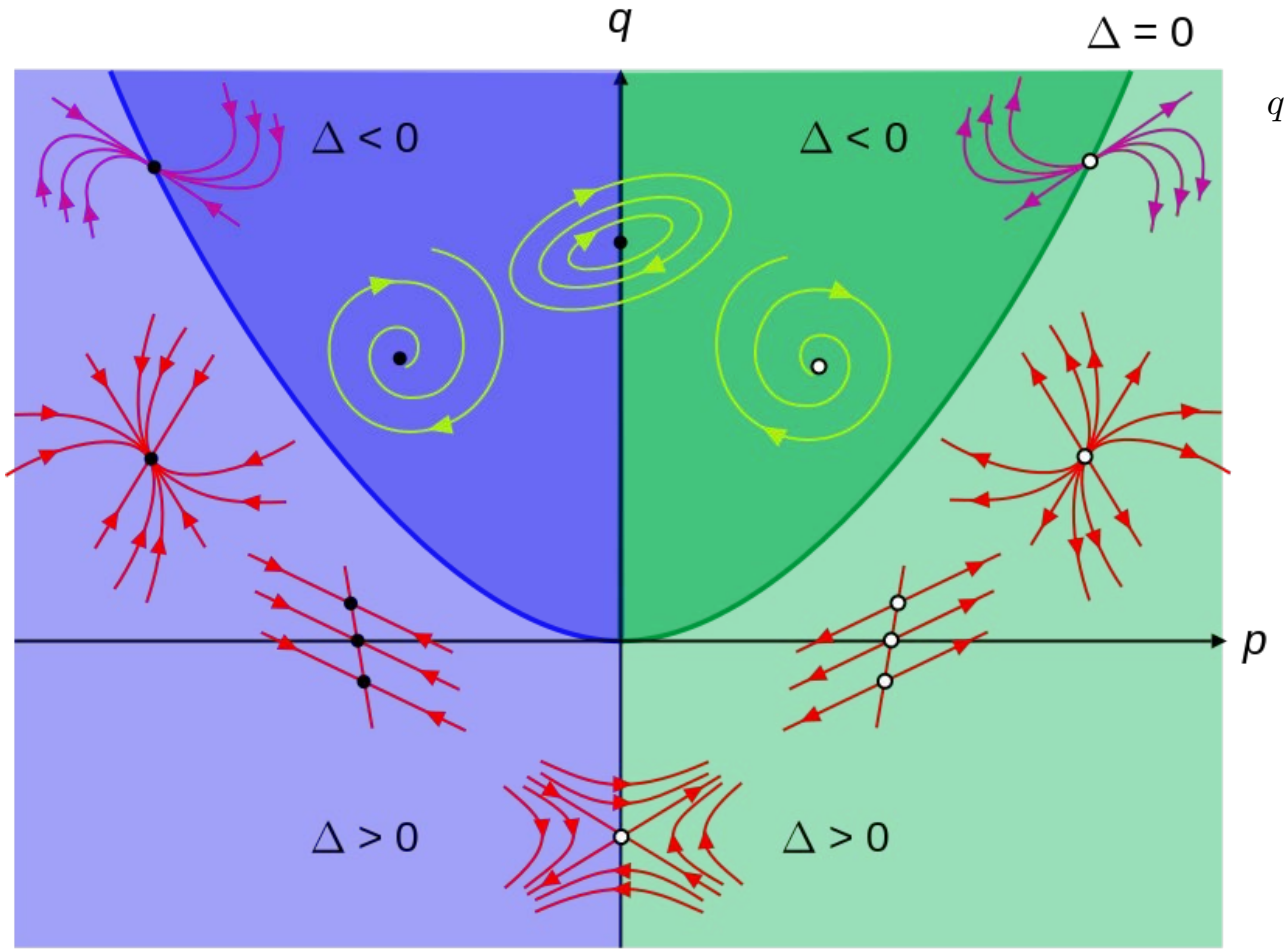
$$\lambda = \frac{1}{2}\left(1 \pm \sqrt{1 + 56}\right)$$

$$\lambda = -3.27, 4.27$$

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

→ General solution is a linear combination of a (real-valued) exponentials, one converging and one diverging

Ex. (cont.)



$$p = \mathrm{Tr}(\mathbf{A}) = 5 + (-4) = 1$$

$$q = \det(\mathbf{A}) = 5(-4) - (-3)2 = -14$$

$$\lambda = -3.27, 4.27$$

→ Solution curves approach the origin, then diverge away

→ Equilibrium point at origin (where the eigenvectors meet) is said to be a *saddle*

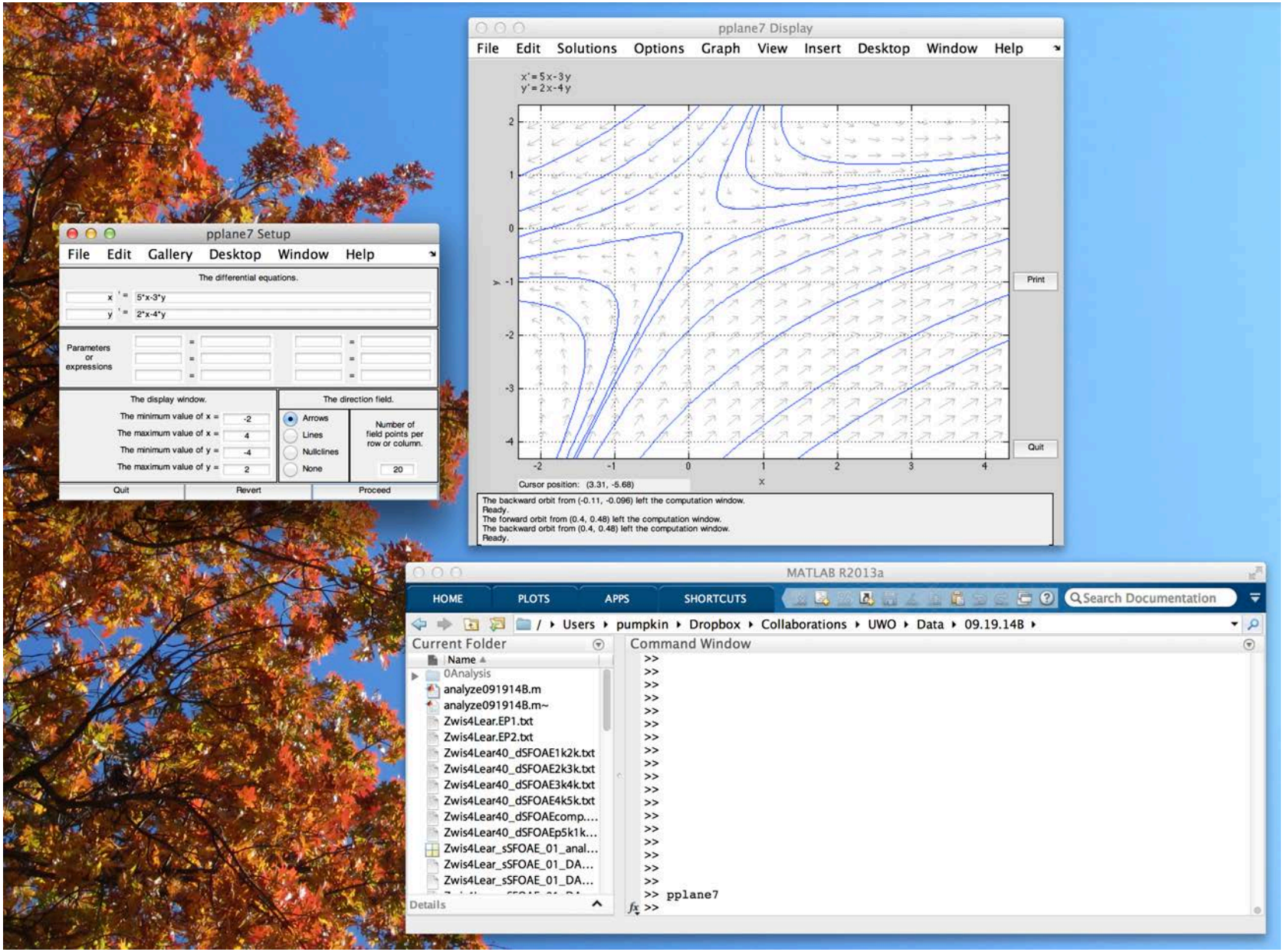
$$\frac{dx}{dt} = Ax + By \qquad p = A + D$$

$$q = AD - BC$$

$$\frac{dy}{dt} = Cx + Dy \qquad \Delta = p^2 - 4q$$

Ex. (cont.)

→ Computationally, use our ode45 code or pplane to explore behavior of solution curves



LINode45EX.m
LINfunction.m

```
% ### LINode45EX.m ###       01.26.16
% Numerically integrate a general 2nd order linear autonomous system (w/
% const. coefficients)
% x' = a*x + b*y
% y' = c*x + d*y

clear
% ----------------------------------------------------
% User input (Note: All paramters are stored in a structure)
P.y0(1) = 1.0;   % initial value for x
P.y0(2) = 1;   % initial value for y
P.A= [-3.9 3;      % matrix A to contain coefficients A= [a b
      -2 1];     %                                       c d]

% Integration limits
P.t0 = 0.0;   % Start value
P.tf = 10.0;   % Finish value
P.dt = 0.01;  % time step
% ---------------------------------------------------------------------
% +++
% determine some basic derived quantities
p= P.A(1,1)+ P.A(2,2);  % Tr(A)
q= P.A(1,1)* P.A(2,2)-P.A(1,2)* P.A(2,1); % det(A)
disp(['Tr(A)= ' num2str(p),' and det(A)= ',num2str(q)]);
eigV1= [0.5*(p+sqrt(p^2-4*q)) 0.5*(p-sqrt(p^2-4*q))];     % calc. eigenvalues directly
eigV2= eig(P.A);    % calculate via Matlab's built-in routine
disp(['eigenvalues= ' num2str(eigV1(1)),' and ',num2str(eigV1(2))]);

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

% ----------------------------------------------------
% visualize
% NOTE (re variable naming): x=y(1) and y=y(2)
figure(1); clf;
plot(t,y(:,1)); hold on; grid on;
xlabel('t');    ylabel('x(t)')
% Phase plane
figure(2); clf;
plot(y(:,1), y(:,2)); hold on; grid on;
xlabel('x(t)');    ylabel('y(t)')
% "solution space"
figure(3); clf;
plot(p,q,'rx','MarkerSize',9,'LineWidth',3); hold on; grid on;
if (abs(p)<1), pSpan= linspace(-1,1,100);
else    pSpan= linspace(-1.5*p-0.1,1.5*p+0.1,100); end
qSpan= pSpan.^2/4;
plot(pSpan,qSpan,'k-','LineWidth',2); %ylim([-max(qSpan) max(qSpan)])
plot(pSpan,zeros(numel(pSpan),1),'b--','LineWidth',2);
xlabel('Tr(A)');    ylabel('det(A)')
```

```
function [out1] = LINfunction(t,y,flag,P)
% ---------------------------------------------
%   y(1) ... x
%   y(2) ... y
out1(1)= P.A(1,1)*y(1) + P.A(1,2)*y(2);
out1(2)= P.A(2,1)*y(1) + P.A(2,2)*y(2);
out1= out1';
```

Finding eigenvalues via Matlab

- Matlab makes it very easy to find eigenvalues and eigenvectors via `eig.m` function

```
>> A=[5 -3;2 -4];
>> [V,D] = eig(A)
V =
          0.972012339068514          0.340833716722178
          0.234929803768182           0.94012359695167
D =
           4.27491721763537                          0
                          0          -3.27491721763537
```

- This function (or `eigs`) can be a 'black box': there is a lot going on under the hood (including root finding!), especially for larger matrices

- For some further background, see:

  http://en.wikipedia.org/wiki/Eigenvalue_algorithm

- We'll soon touch upon a general application of `eigs` in the context of *eigenfaces*

Ex. Harmonic oscillator as an eigenvalue problem

Rewrite as a system of first order ODEs

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

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

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

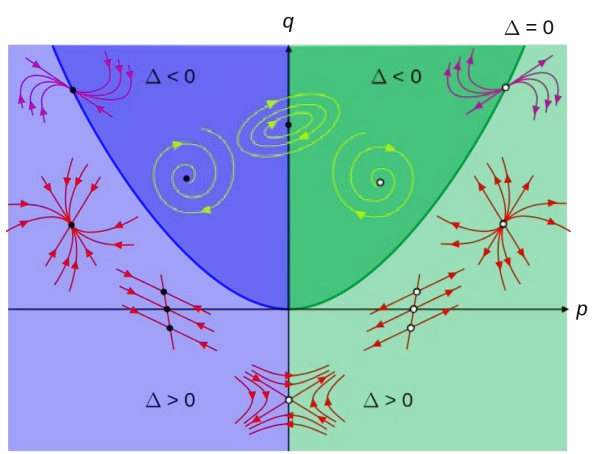
$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -\omega_o^2 & -\gamma \end{pmatrix}$$

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

- What if γ is zero? Negative?

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

- Depending upon the sign and relative values of γ and ω_o, λ can be complex

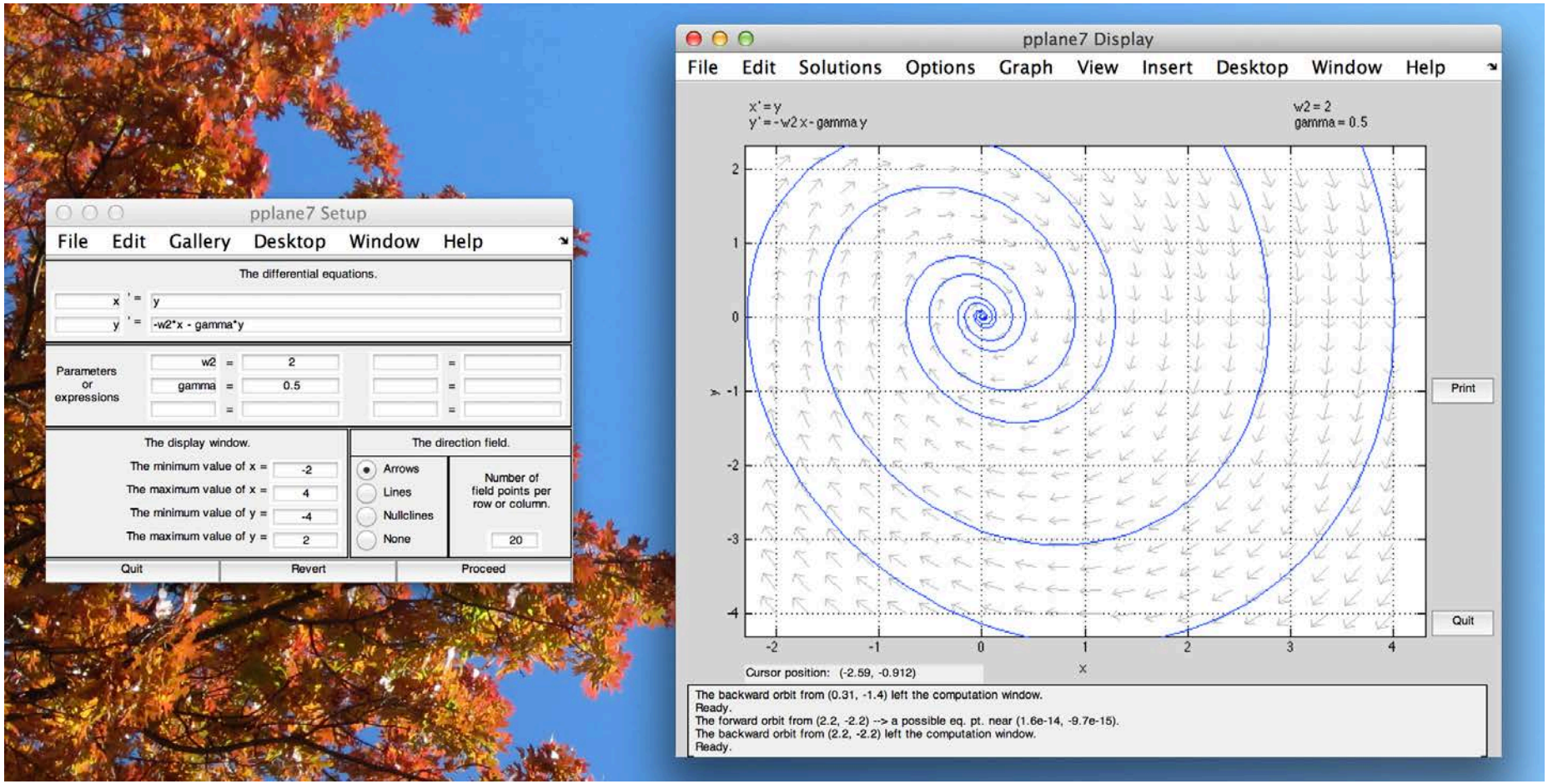


→ Eigenvalues characterize behavior of all possible solution types!

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

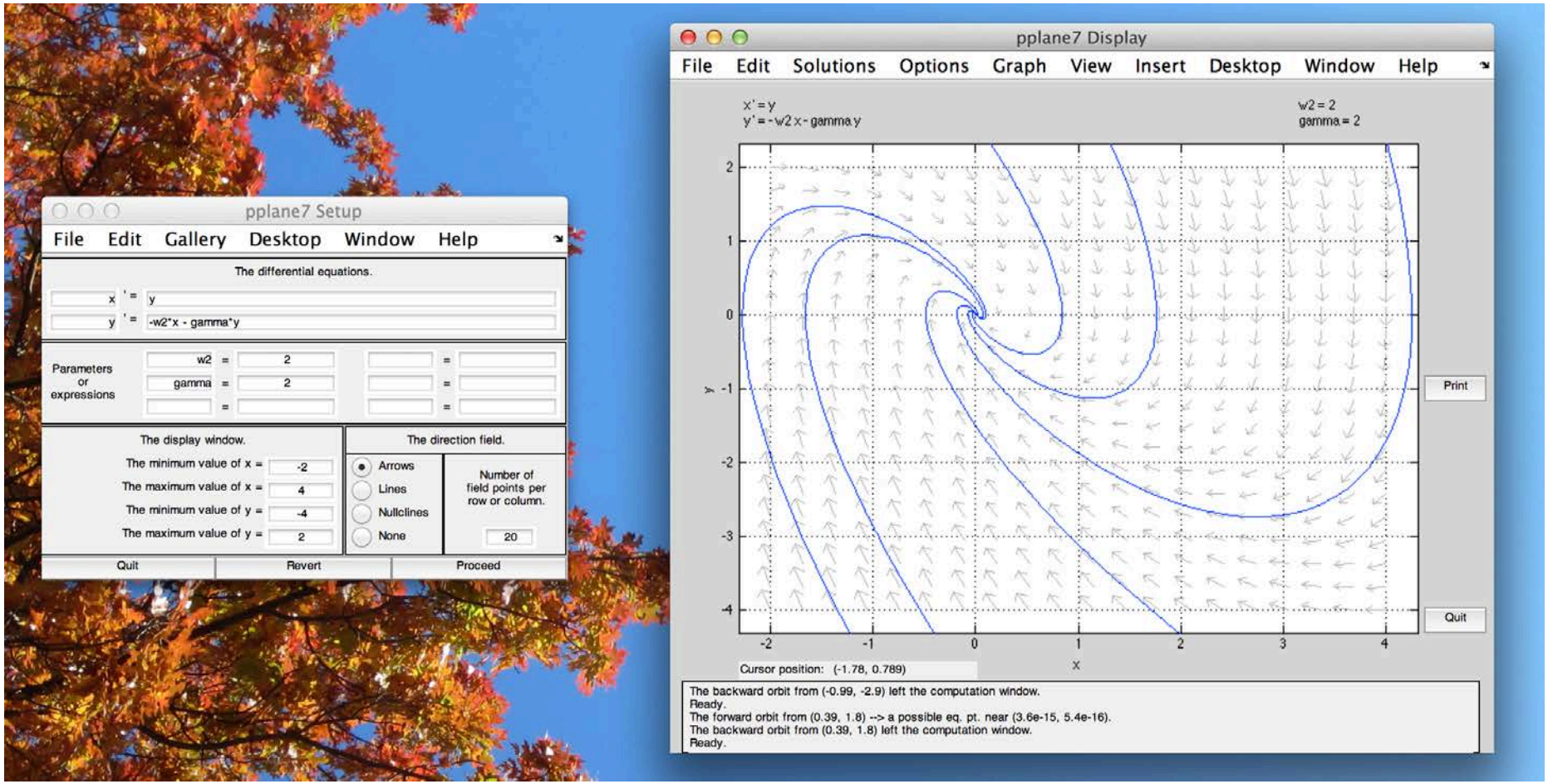
Ex. Harmonic oscillator as an eigenvalue problem

- $\gamma = 0.5$
- $\omega_0^2 = 2$



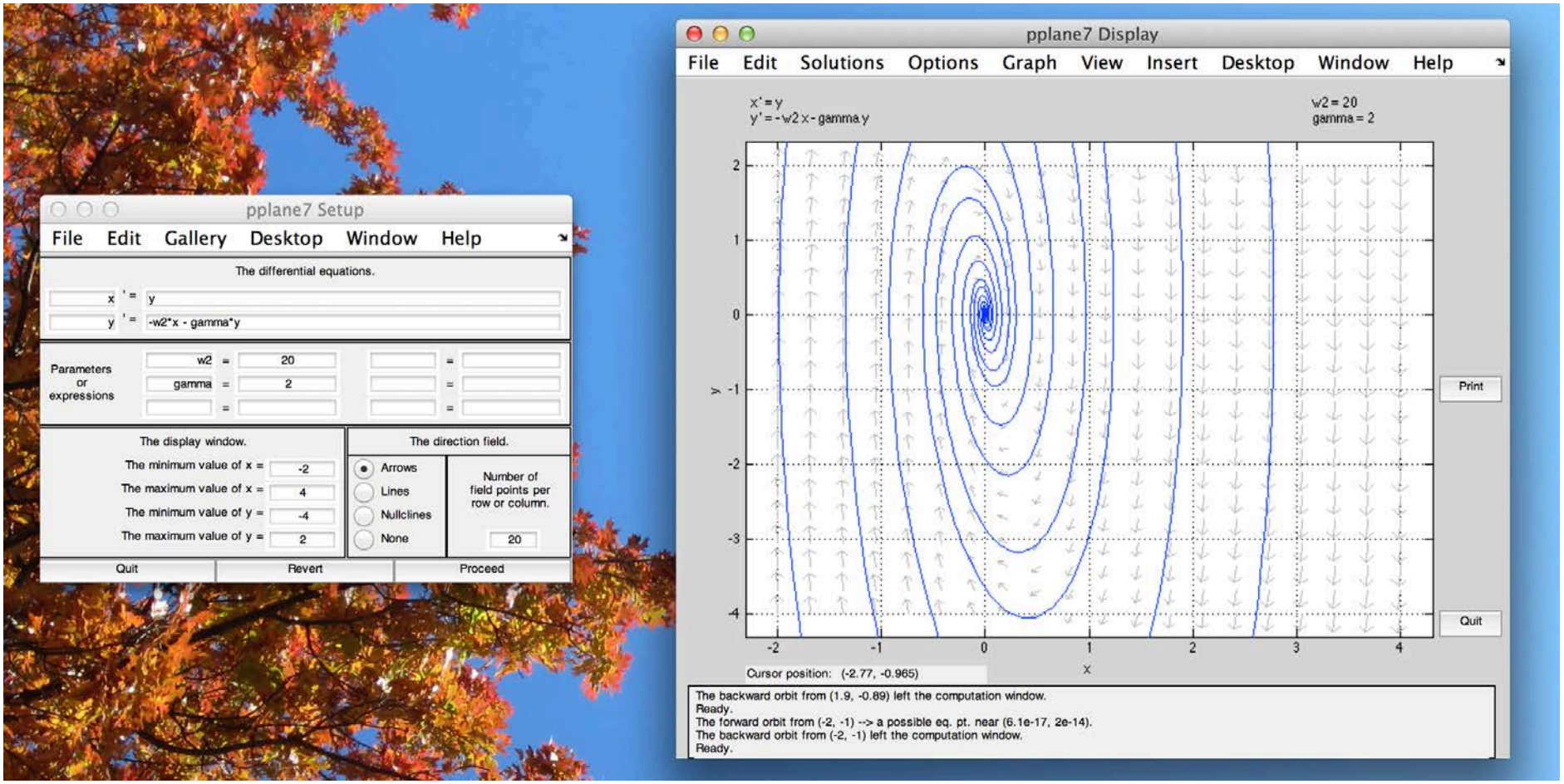
Ex. Harmonic oscillator as an eigenvalue problem

- $\gamma = 2$
- $\omega_0^2 = 2$



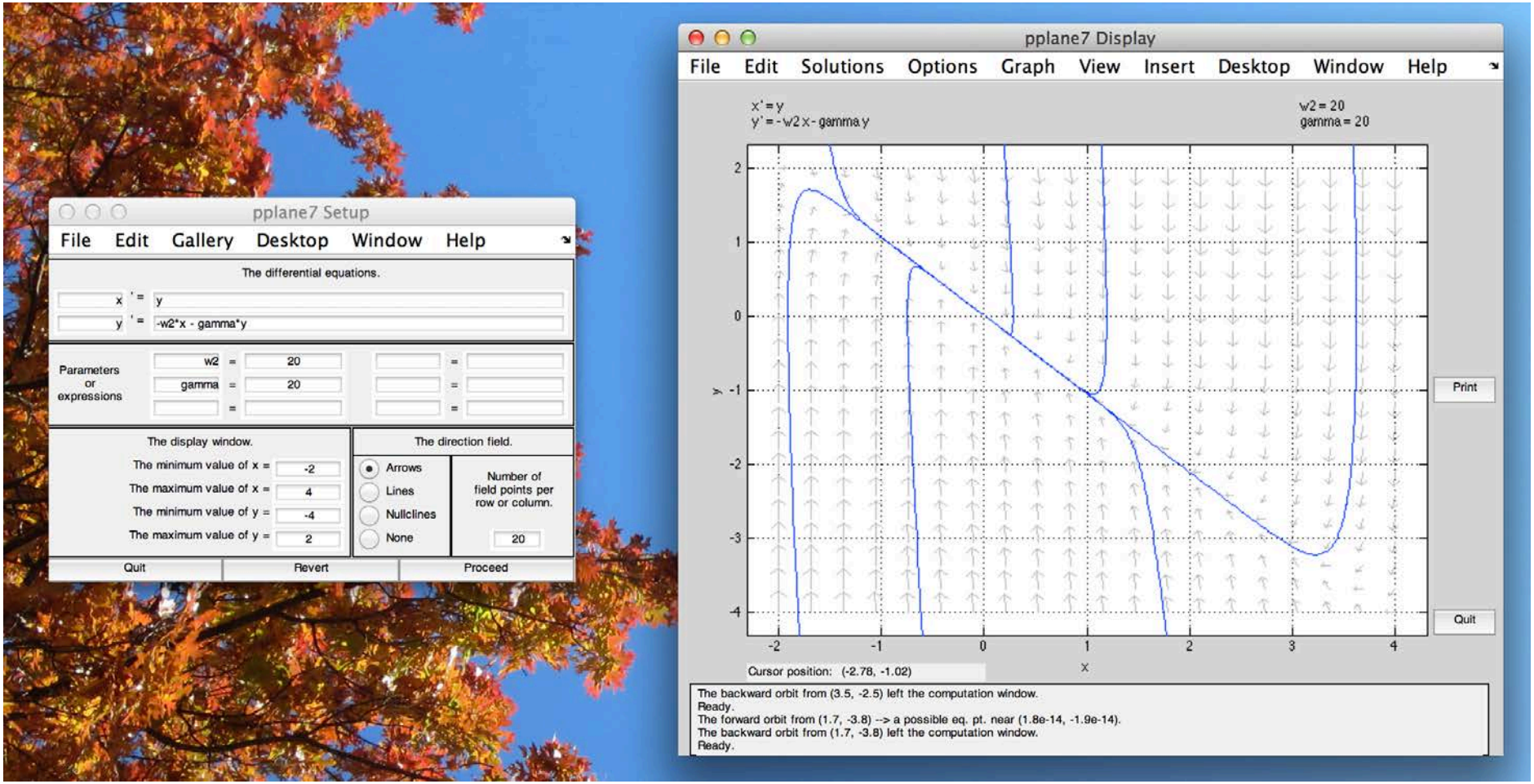
Ex. Harmonic oscillator as an eigenvalue problem

- $\gamma = 2$
- $\omega_0^2 = 20$



Ex. Harmonic oscillator as an eigenvalue problem

- $\gamma = 20$
- $\omega_0^2 = 20$



Nonlinear systems

- So what can this approach tell us about nonlinear systems (e.g., van der Pol)?

Linearize!

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

e.g., Connect pendulum and Taylor series back to simple harmonic oscillator!

1. Find the fixed points of the system

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

$$\frac{dy}{dt} = -x + \epsilon(1 - x^2)y$$

2. For a given fixed point (x_o,y_o), determine the *Jacobian* matrix

$$\dot{x} = f(x, y)$$

$$\dot{y} = g(x, y)$$

Jacobian

$$J(x_o, y_o) = \left[\begin{array}{cc} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{array} \right] \Bigg|_{x_o, y_o}$$

3. Determine the associated eigenvalues

→ Provides a snapshot in the neighborhood local to the fixed point

Ex. van der Pol

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

$$\frac{dy}{dt} = -x + \epsilon(1 - x^2)y$$

- Fixed point $(x_o, y_o) = (0,0)$
- Associated Jacobian is then:

$$J(0,0) = \begin{bmatrix} 0 & 1 \\ -1 & \epsilon \end{bmatrix}$$

$$J(x_o, y_o) = \left. \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} \right|_{x_o, y_o}$$

- The characteristic equation is then:

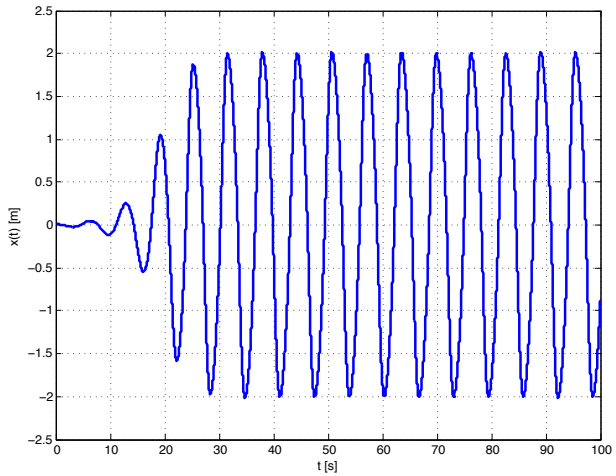
$$\left| \begin{bmatrix} 0 & 1 \\ -1 & \epsilon \end{bmatrix} - \lambda \mathbf{I} \right| = 0$$

- Associated eigenvalues:

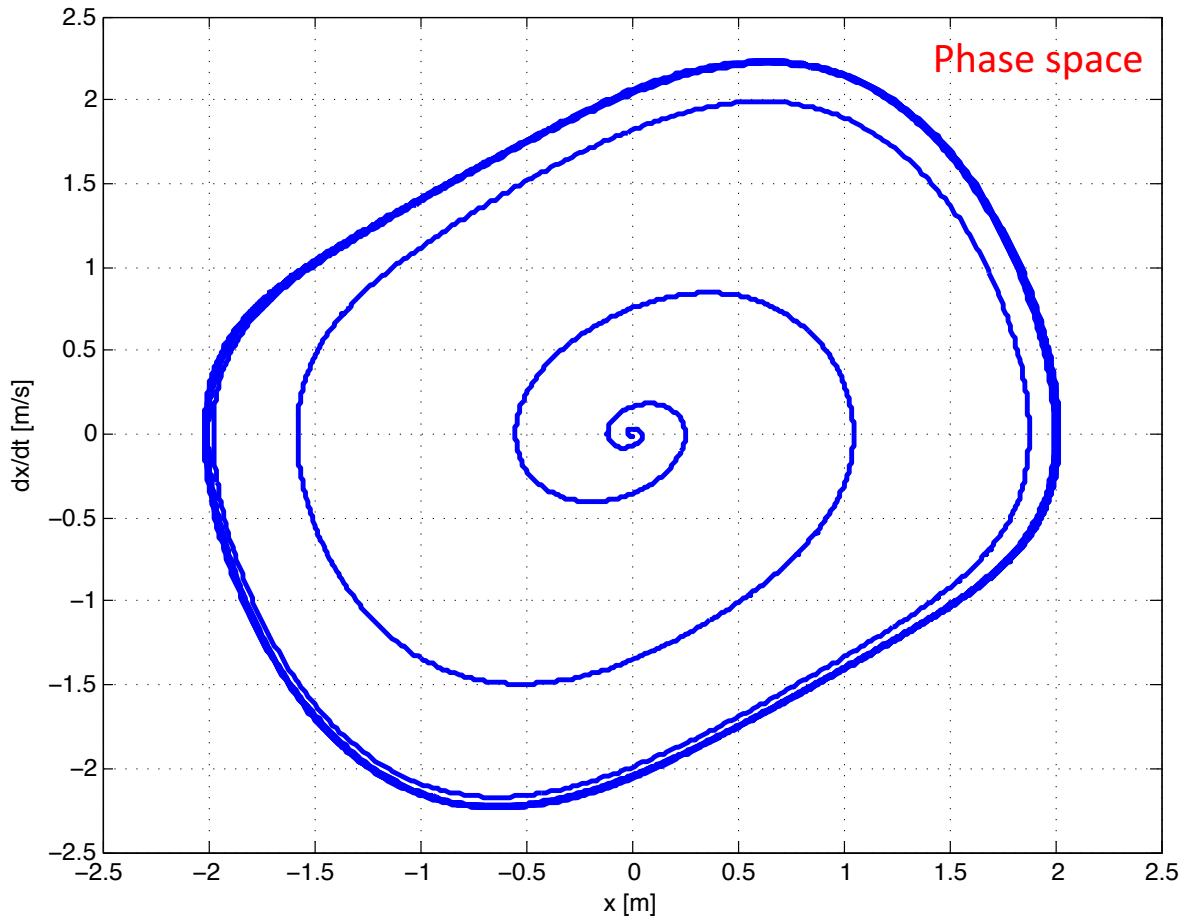
$$\lambda = \frac{\epsilon \pm \sqrt{\epsilon^2 - 4}}{2}$$

→ Assuming $\varepsilon > 0$, the real parts of both eigenvalues are positive meaning that all solutions will diverge away from the fixed point (i.e., it is unstable)

VDPode45EX.m



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



→ Our linearized analysis (so far) didn't tell us about the limit cycle, but it allows us to infer something about it (i.e., it is likely stable)

Ex. Lotka-Volterra

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

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

1. Find the fixed points of the system

- Two fixed points: $(x_o, y_o) = (0,0)$ and $(\gamma/\delta, \alpha/\beta)$

2. Find their stability

$$J = \begin{bmatrix} \alpha - \beta y & -\beta x \\ \delta y & \delta x - \gamma \end{bmatrix}$$

$$J(0,0) = \begin{bmatrix} \alpha & 0 \\ 0 & -\gamma \end{bmatrix} \qquad J\left(\frac{\gamma}{\delta}, \frac{\alpha}{\beta}\right) = \begin{bmatrix} 0 & -\beta\gamma/\delta \\ \delta\alpha/\beta & 0 \end{bmatrix}$$

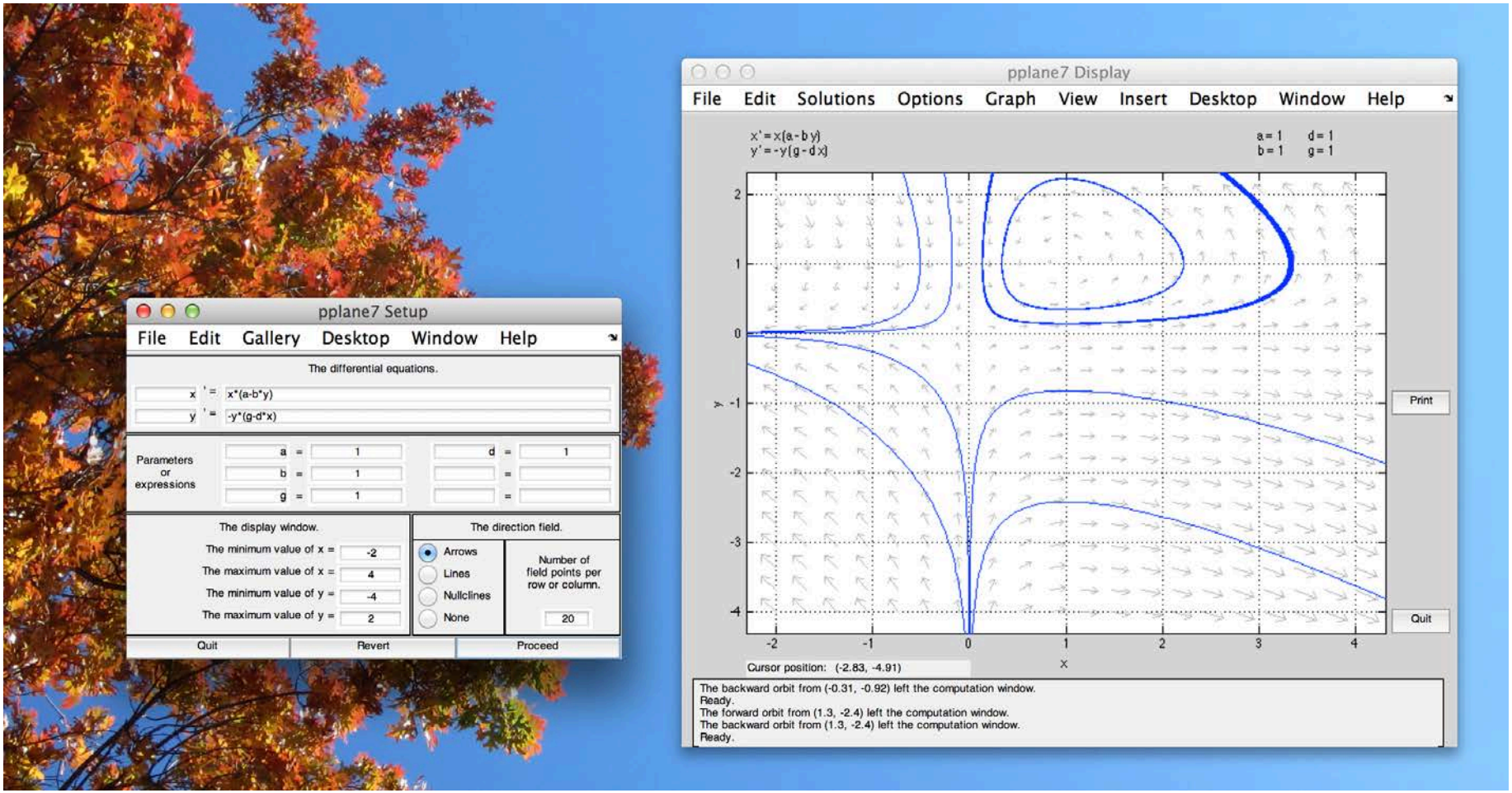
When the smoke clears (after a bit of algebra), the corresponding eigenvalues are:

$$\lambda_{(0,0)} = \alpha, -\gamma \qquad \lambda_{\left(\frac{\gamma}{\delta}, \frac{\alpha}{\beta}\right)} = \pm i\sqrt{\alpha\gamma}$$

→ So the stability depends upon the actual choice of the various parameters (much like the harmonic oscillator)

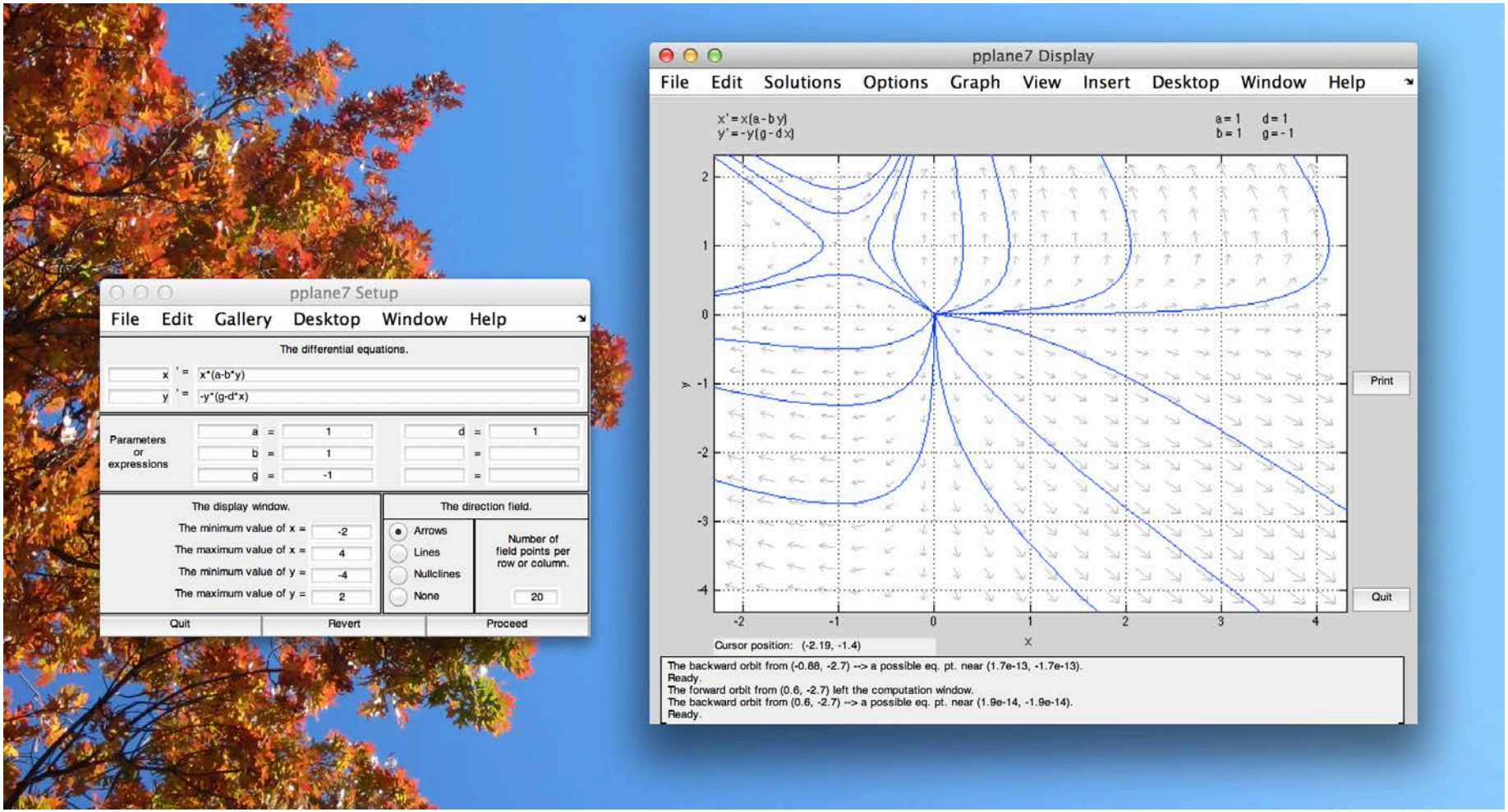
Ex. Lotka-Volterra

$$\lambda_{(0,0)} = \alpha, -\gamma \qquad \lambda_{\left(\frac{\gamma}{\delta}, \frac{\alpha}{\beta}\right)} = \pm i\sqrt{\alpha\gamma}$$



Ex. Lotka-Volterra

$$\lambda_{(0,0)} = \alpha, -\gamma \qquad \lambda_{\left(\frac{\gamma}{\delta}, \frac{\alpha}{\beta}\right)} = \pm i\sqrt{\alpha\gamma}$$



→ Clearly something different happens when γ changes sign. This is an example of a *bifurcation* (we'll come back to this later in the semester)

Extending further: Eigenfaces

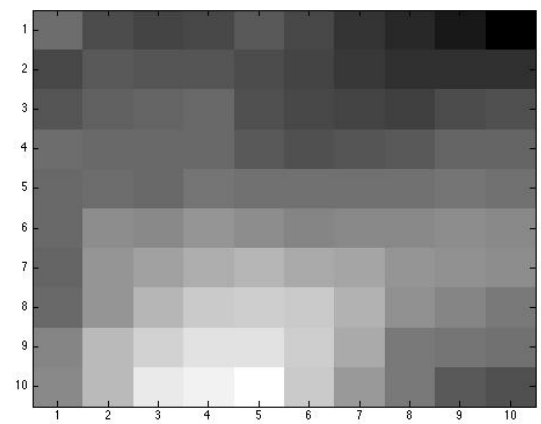
Kutz (2013)

Extending further: Eigenfaces

- Keep in mind that an image is really just an array of numbers....

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

.... much like our matrix form for linear ODEs



c =

115	108	106	107	111	107	103	100	97	91
107	111	110	110	108	106	104	102	102	102
110	112	113	114	109	107	106	105	108	109
115	114	114	114	111	109	110	111	113	113
114	115	114	117	116	116	116	116	117	116
114	122	121	124	122	120	121	121	122	121
113	124	126	129	131	128	127	124	123	122
114	124	131	135	136	135	130	123	120	118
120	132	137	140	140	136	128	118	117	116
121	132	142	144	147	135	125	118	111	109

- So why not try calculating eigenvalues and eigenvectors for an image?

Extending further: Eigenfaces

- Averaging the faces emphasizes common features

- Basic idea is the eigenvectors (the associated eigenvalues acting as their weighting) uniquely identify a face and thereby can be used for *facial recognition* algorithms in computer vision

- Goal is to obtain a 'low-dimensional' representation of a 'face' and is related to the more general notion of Principal Component Analysis (PCA)

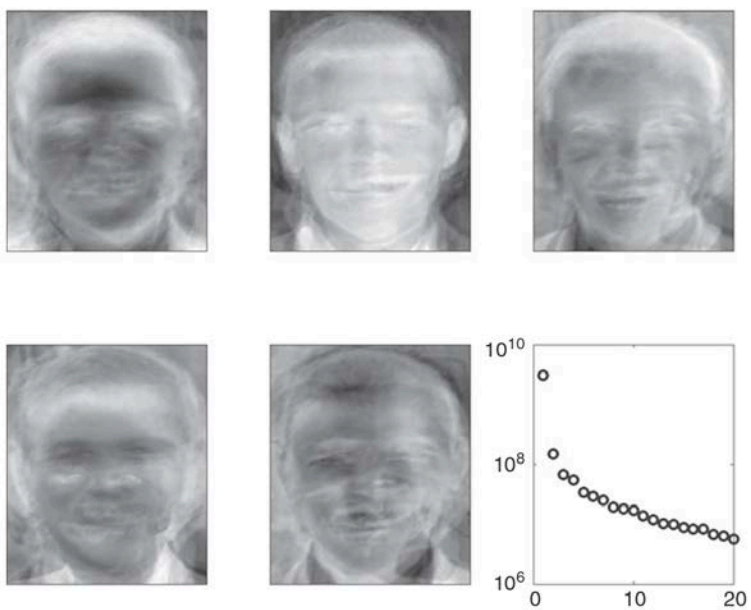
- Requires determination of an 'average face', then comparing to a 'test image' to quantify the associated 'covariance'

Kutz (2013)

Extending further: Eigenfaces

- Basic recipe to implement:

1. Prepare a training set of face images. The pictures constituting the training set should have been taken under the same lighting conditions, and must be normalized to have the eyes and mouths aligned across all images. They must also be all resampled to a common pixel resolution ($r \times c$). Each image is treated as one vector, simply by concatenating the rows of pixels in the original image, resulting in a single row with $r \times c$ elements. For this implementation, it is assumed that all images of the training set are stored in a single matrix **T**, where each column of the matrix is an image.
2. Subtract the mean. The average image **a** has to be calculated and then subtracted from each original image in **T**.
3. Calculate the eigenvectors and eigenvalues of the covariance matrix **S**. Each eigenvector has the same dimensionality (number of components) as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called eigenfaces. They are the directions in which the images differ from the mean image. Usually this will be a computationally expensive step (if at all possible), but the practical applicability of eigenfaces stems from the possibility to compute the eigenvectors of **S** efficiently, without ever computing **S** explicitly, as detailed below.
4. Choose the principal components. The $D \times D$ covariance matrix will result in D eigenvectors, each representing a direction in the $r \times c$-dimensional image space. The eigenvectors (eigenfaces) with largest associated eigenvalue are kept.

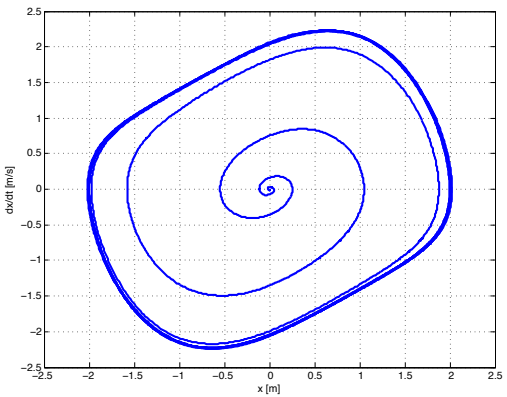


→ We will aim to return to this topic in more detail later in 2030

wikipedia (Eigenfaces)
Kutz (2013)

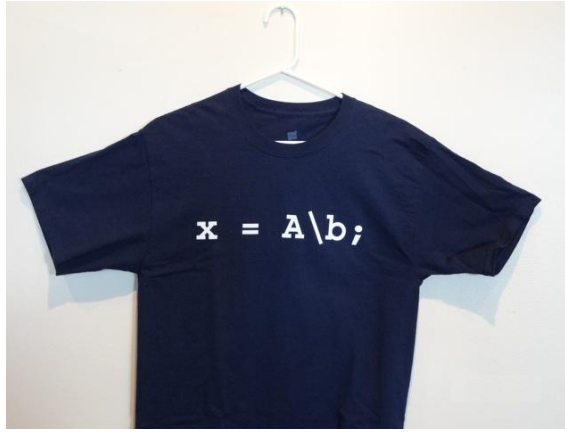
Summary

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

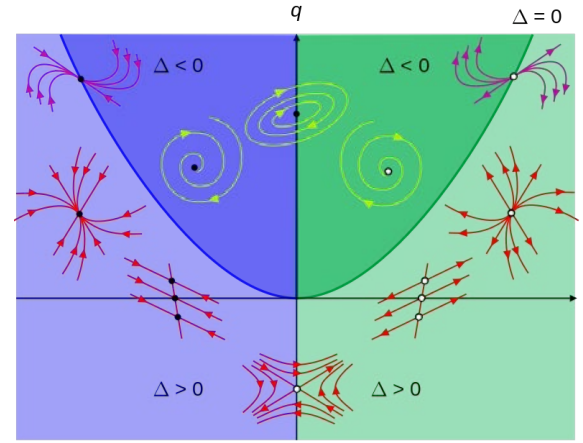


Nonlinear oscillators
& limit cycles

$$\mathbf{Ax} = \mathbf{b} \;\rightarrow\; \mathbf{LUx} = \mathbf{b}$$



Linear systems analysis

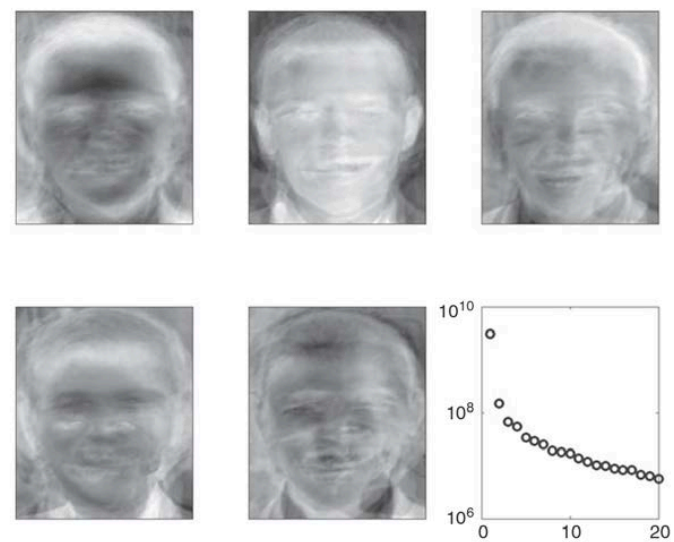


$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

'Eigen' analysis

$$J(x_o, y_o) = \left.\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}\right|_{x_o, y_o}$$

Linearizing nonlinear systems



Other applications (e.g., eigenfaces)

Post-class exercises

- For the harmonic oscillator, how do the eigenvalues relate to the system being under-, critically-, or over-damped?

- For an arbitrary 2x2 or 3x3 matrix, write a code to explicitly determine the eigenvalues (i.e., without using `eig.m`)

- Load in an image of yourself into Matlab. Calculate the associated eigenvalues and eigenvectors.
  [Hint: You may need to be careful when dealing with uint8 variables. Try converting to double.]

- Consider below the undriven Duffing oscillator. Physically, what does the nonlinear term represent? Find all equilibrium points and determine their stability via calculation of the associated Jacobian.

$$\ddot{x} + \delta\dot{x} + \beta x + \alpha x^3 = 0$$