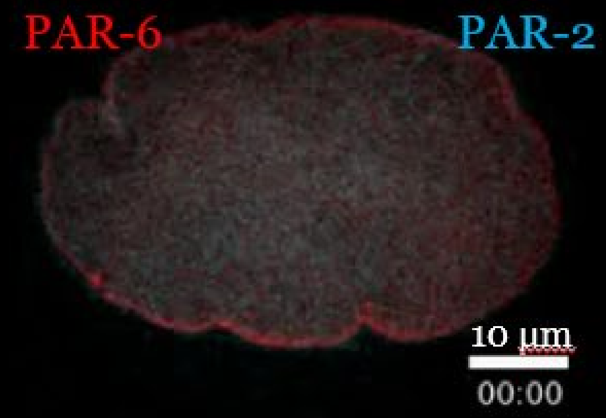
Modeling Biological Systems through Space and Time

QLS Breakfast Seminar
11 September 2024

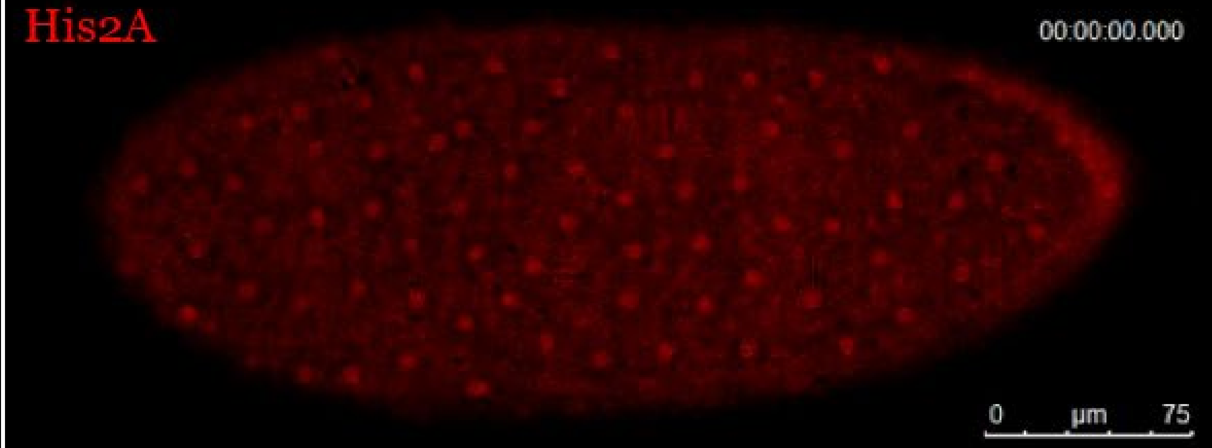
Rocky Diegmiller

cell polarization



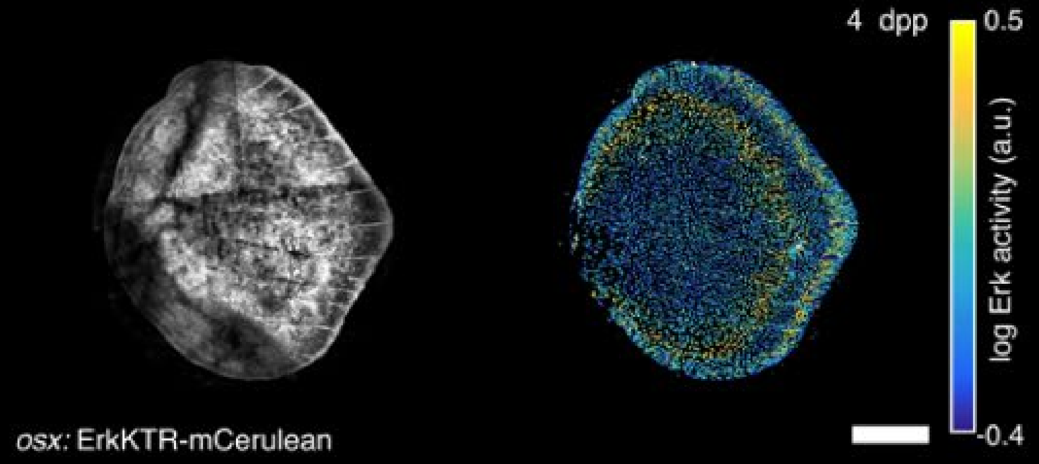
C. elegans embryo

waves of coordinated cell behavior



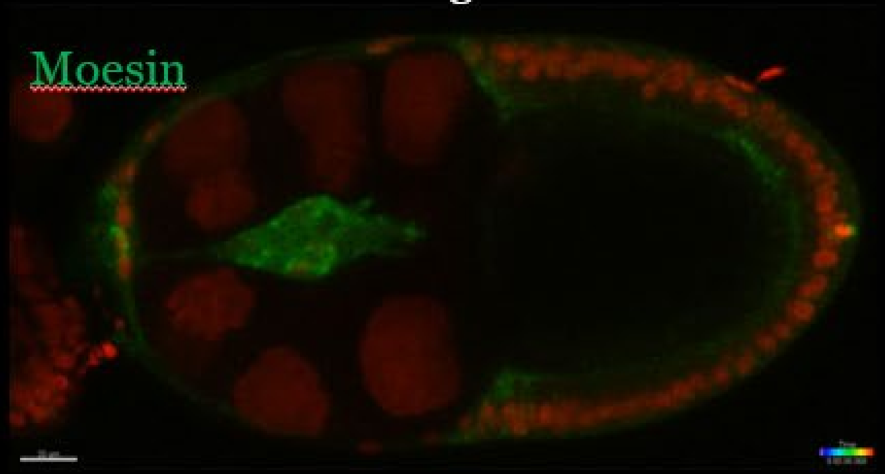
D. melanogaster embryo

appendage regeneration



D. rerio scale

cell migration



D. melanogaster egg chamber

Polarization of PAR Proteins by Advective Triggering of a Pattern-Forming System

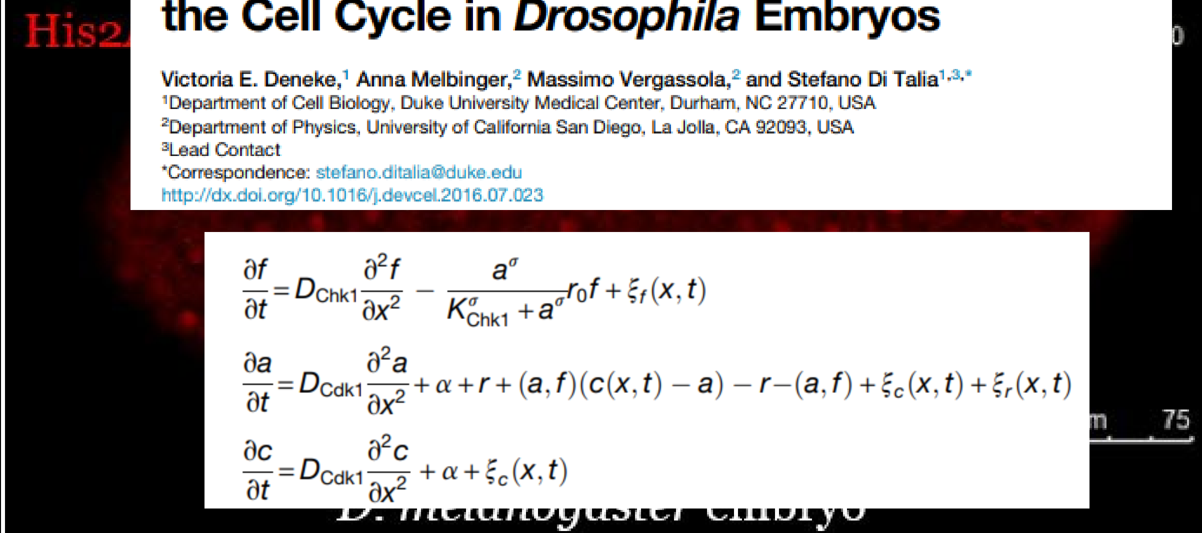
Nathan W. Goehring,[1] Philipp Khuc Trong,[2,1]* Justin S. Bois,[2,1]† Debanjan Chowdhury,[2]‡ Ernesto M. Nicola,[2]§ Anthony A. Hyman,[1] Stephan W. Grill[2,1]‖

$$\partial_t A = D_{\mathrm{A}} \partial_x^2 A - \partial_x(vA) + R_{\mathrm{A}}$$

$$\partial_t P = D_{\mathrm{P}} \partial_x^2 P - \partial_x(vP) + R_{\mathrm{P}}$$

Waves of Cdk1 Activity in S Phase Synchronize the Cell Cycle in *Drosophila* Embryos

Victoria E. Deneke,[1] Anna Melbinger,[2] Massimo Vergassola,[2] and Stefano Di Talia[1,3,*]
[1]Department of Cell Biology, Duke University Medical Center, Durham, NC 27710, USA
[2]Department of Physics, University of California San Diego, La Jolla, CA 92093, USA
[3]Lead Contact
*Correspondence: stefano.ditalia@duke.edu
http://dx.doi.org/10.1016/j.devcel.2016.07.023



$$\frac{\partial f}{\partial t} = D_{\mathrm{Chk1}} \frac{\partial^2 f}{\partial x^2} - \frac{a^{\sigma}}{K_{\mathrm{Chk1}}^{\sigma} + a^{\sigma}} r_0 f + \xi_f(x,t)$$

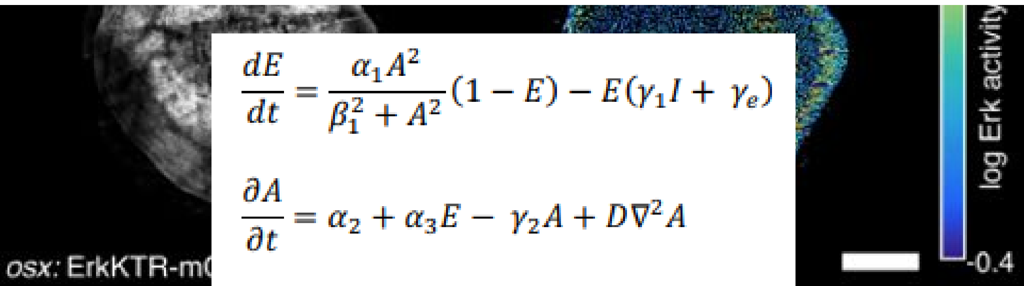
$$\frac{\partial a}{\partial t} = D_{\mathrm{Cdk1}} \frac{\partial^2 a}{\partial x^2} + \alpha + r_+(a,f)(c(x,t) - a) - r_-(a,f) + \xi_c(x,t) + \xi_r(x,t)$$

$$\frac{\partial c}{\partial t} = D_{\mathrm{Cdk1}} \frac{\partial^2 c}{\partial x^2} + \alpha + \xi_c(x,t)$$

Control of osteoblast regeneration by a train of Erk activity waves

https://doi.org/10.1038/s41586-020-03085-8
Received: 8 October 2019

Alessandro De Simone[1,2], Maya N. Evanitsky[1,2], Luke Hayden[1,2], Ben D. Cox[1,2,6], Julia Wang[1,2], Valerie A. Tornini[1,2,7], Jianhong Ou[1], Anna Chao[1,2], Kenneth D. Poss[1,2,3,4]✉ & Stefano Di Talia[1,2,5]✉



$$\frac{dE}{dt} = \frac{\alpha_1 A^2}{\beta_1^2 + A^2}(1 - E) - E(\gamma_1 I + \gamma_e)$$

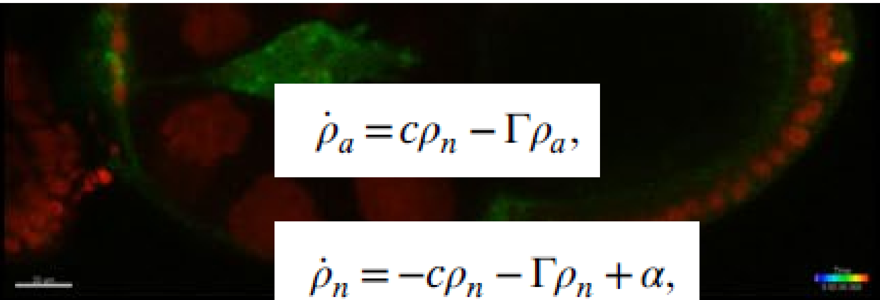
$$\frac{\partial A}{\partial t} = \alpha_2 + \alpha_3 E - \gamma_2 A + D\nabla^2 A$$

$$\frac{dI}{dt} = \gamma_3(\alpha_4 E - I)$$

Modeling and analysis of collective cell migration in an in vivo three-dimensional environment

Danfeng Cai[a,b], Wei Dai[a], Mohit Prasad[c], Junjie Luo[a,b], Nir S. Gov[d,1], and Denise J. Montell[a,b,1]

[a]Molecular, Cellular and Developmental Biology, University of California Santa Barbara, CA 93106; [b]Department of Biological Chemistry, The Johns Hopkins School of Medicine, Baltimore, MD 21205; [c]Department of Biological Sciences, Indian Institute of Science Education and Research Kolkata, West Bengal 741252, India; and [d]Department of Chemical Physics, The Weizmann Institute of Science, Rehovot 76100, Israel



$$\dot{\rho}_a = c\rho_n - \Gamma\rho_a,$$

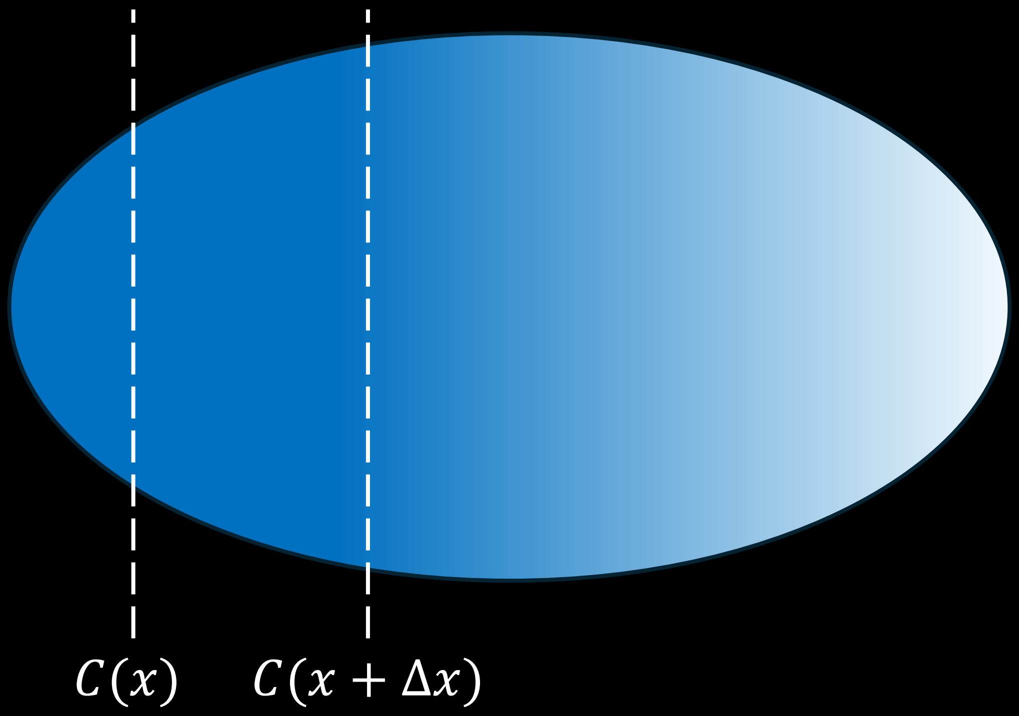
$$\dot{\rho}_n = -c\rho_n - \Gamma\rho_n + \alpha,$$

D. melanogaster egg chamber

Why bother modeling?

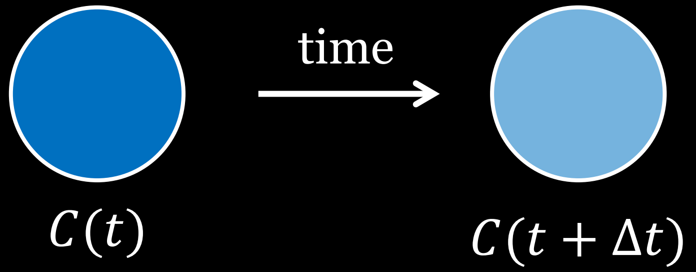
- To better understand the mechanics or biological interactions that are driving observed behaviors
- To identify common patterns or integrate ideas from other fields
- To generate testable predictions based on outcomes of modeled simulations or parameters (or to make predictions without having to generate or test live animals)

Modeling and estimating rates of change



$$\frac{\Delta C}{\Delta x} = \frac{C(x + \Delta x) - C(x)}{\Delta x}$$

$$\frac{dC(x)}{dx} = \lim_{\Delta x \to 0} \frac{C(x + \Delta x) - C(x)}{\Delta x}$$



$$\frac{\Delta C}{\Delta t} = \frac{C(t + \Delta t) - C(t)}{\Delta t}$$

$$\frac{dC(t)}{dt} = \lim_{\Delta t \to 0} \frac{C(t + \Delta t) - C(t)}{\Delta t}$$

Modeling and estimating rates of change

- Once we have a mathematical model to consider for our observed data, we will need some way to interpret it or solve to find how the prediction behaves in space, time, and with respect to other changes
- Luckily for us, MATLAB (and Python, Mathematica, and similar programs) can numerically solve differential equations, even when explicit solutions are hard or impossible
- There is also a rich amount of research that goes into both the theory of differential equations, as well as the codes that help numerically solve them with higher precision and greater efficiency

A simple example

- Consider the following ordinary differential equation (ODE):

$$\frac{dy}{dx} = xy, \qquad y(0) = 2$$

- This is a separable problem that we can explicitly solve:

$$\frac{dy}{dx} = xy \;\Rightarrow \int \frac{1}{y}\frac{dy}{dx}\,dx = \int x\,dx \Rightarrow \ln(y) = \frac{1}{2}x^2 + \mathrm{C} \;\Rightarrow y = A\exp\left(\frac{1}{2}x^2\right)$$

$$y(0) = 2 \;\Rightarrow 2 = A\exp\left(\frac{1}{2}(0)^2\right) \;\Rightarrow 2 = A \;\Rightarrow \boxed{y = 2\exp\left(\frac{1}{2}x^2\right)}$$

MATLAB makes life easier

output of integration

system of equations for numerical integration

initial/boundary condition(s)

```
[x,y] = ode45(@(x,y) x*y, [0 3], 2);
```

interval to solve over

dependent variable(s)

independent variable

MATLAB makes life easier

- `ode45` (and other numerical solvers) solves the ODE by selecting successive points inside the interval and estimating the value of the function at the new selected point

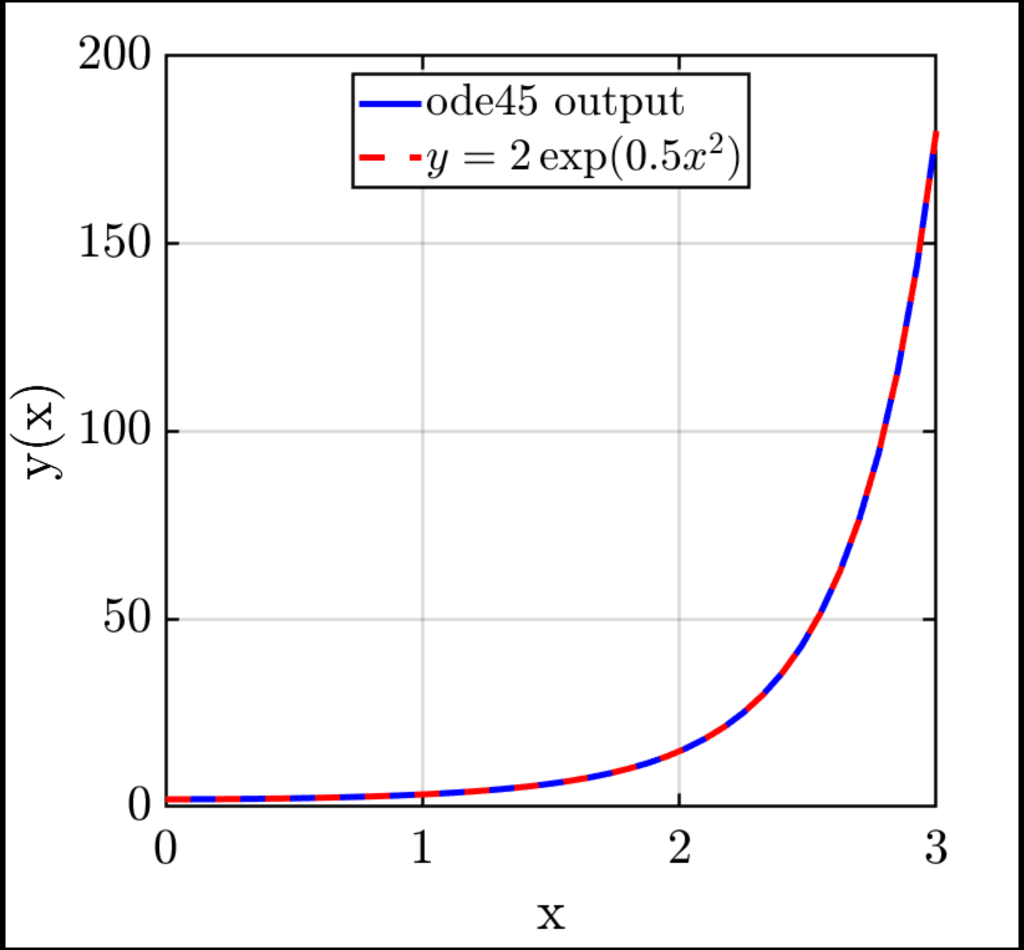
- Said another way:

From initial point x_0, pick new point x_1

Use algorithm to estimate change in function, Δf

Add to previous known value, $f(x_0)$, to get estimate of $f(x_1)$

Take $x_1, f(x_1)$ as new values. Repeat until the end of the interval



A harder example

- Consider the ODE

$$\frac{dy}{dx} = \sin(x + y), \qquad y(0) = \pi$$

- This ODE has no explicit solutions (as far as I can tell…)

- Some clever math tricks (Wolfram Alpha) can get us to the implicit solution

$$\tan(x + y) - \sec(x + y) = x + 1$$

- BUT! MATLAB is still perfectly happy to plot up a numerically-derived solution to this problem

Once again, MATLAB makes life easier

system of equations for numerical integration

initial/boundary condition(s)

output of integration

```
[x,y] = ode45(@(x,y) sin(x+y), [0 4], pi);
```

interval to solve over

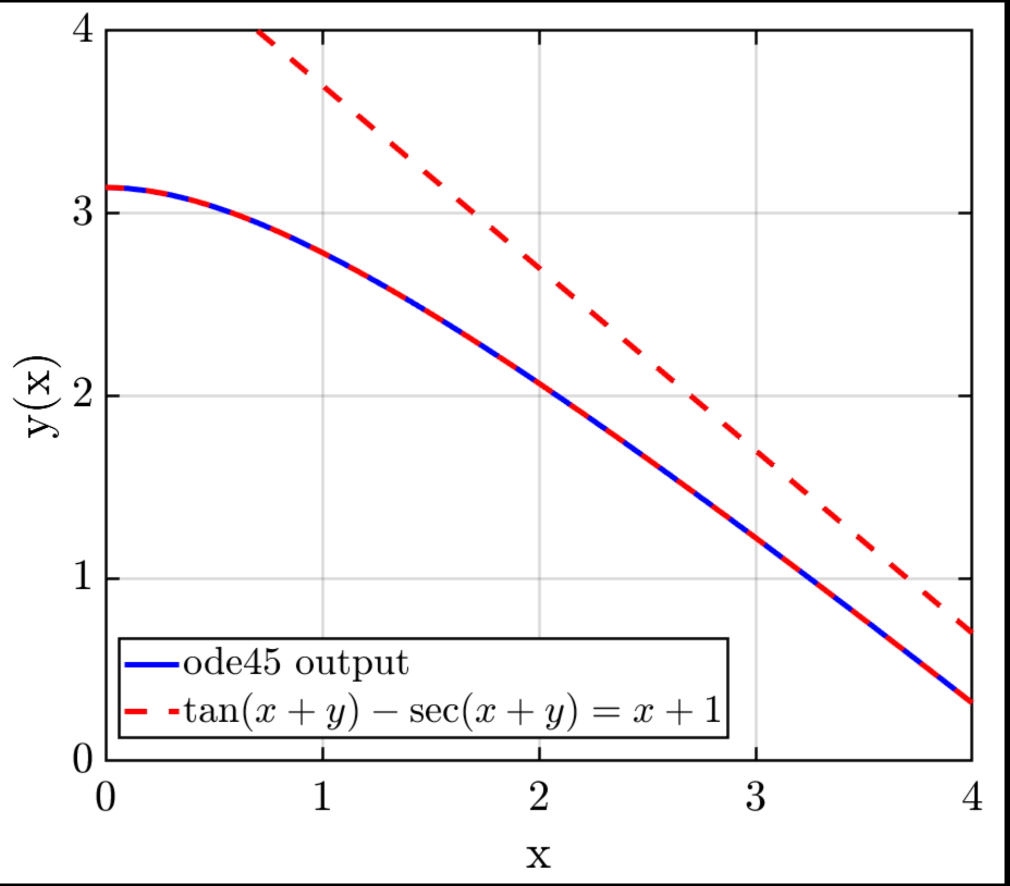
dependent variable(s)

independent variable

Once again, MATLAB makes life easier

- As before, `ode45` does a great job of matching the solution of our ODE

- In fact, it seems to do an even better job than implicitly plotting over the interval with the MATLAB `fimplicit` function



Systems of ODEs

- Consider the system of ODEs

$$\frac{dx}{dt} = 3x - 2xy, \qquad x(0) = 2$$

$$\frac{dy}{dt} = -y + 3xy, \qquad y(0) = 1$$

- This system is an example of a Lotka-Volterra system (predator-prey model)
- As we will see, systems with this form can produce oscillations

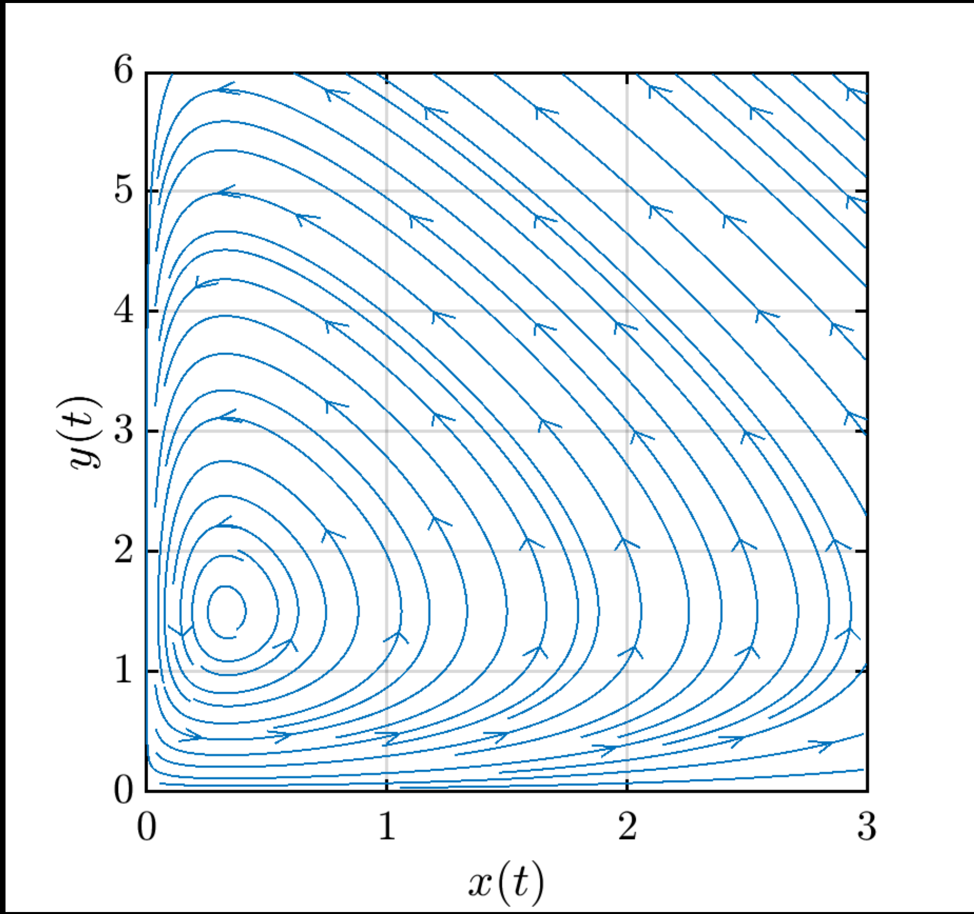
SHAMELESS PLUG!

The `streamslice` function for visualizing systems of ODEs

- MATLAB also gives us the ability to view the "flow" of a system of ODEs with the `streamslice` function:

```
[X,Y] = meshgrid(0:0.1:3,0:0.1:6);
U = 3.*X-2.*X.*Y;
V = -Y+3.*X.*Y;

figure; box on; grid on;
streamslice(X,Y,U,V)
```



Numerically solving systems of ODEs

system of equations for numerical integration

output of integration

initial conditions

```
[t,y] = ode45(@(t,y) [3*y(1)-2*y(1)*y(2); -y(2)+3*y(1)*y(2)], [0 30], [2 1]);
```

x y

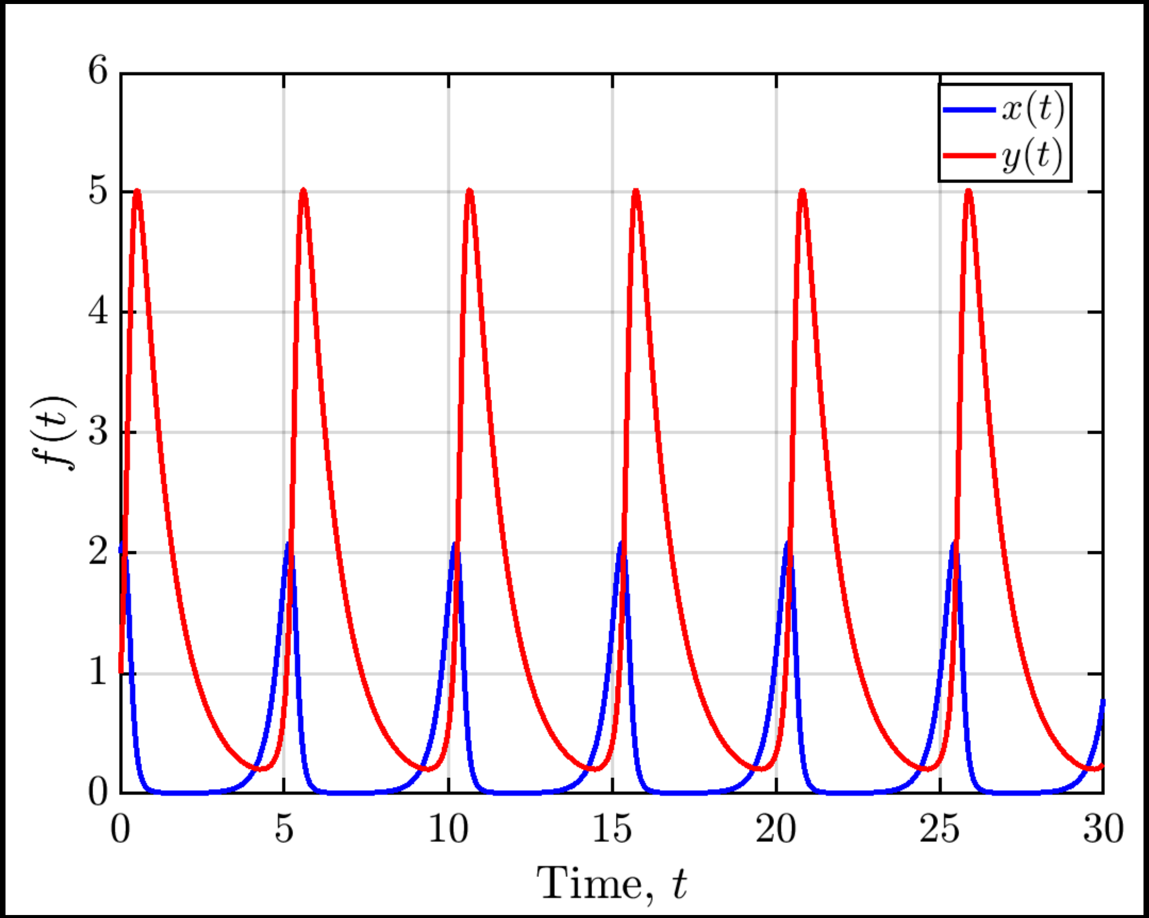
interval to solve over

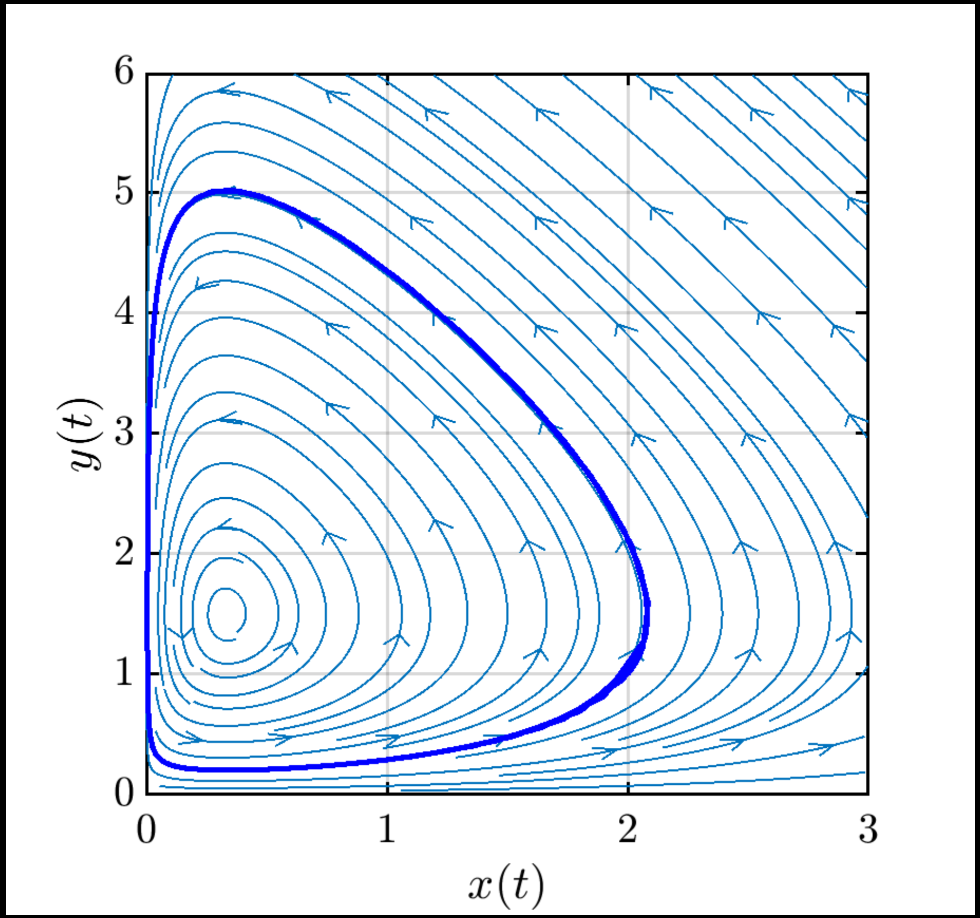
dependent variable(s)

independent variable

Here, y is now a vector of values that MATLAB is trying to solve over: $\underline{y} = \begin{pmatrix} y(1) \\ y(2) \end{pmatrix}$ $\begin{matrix} \to x \\ \to y \end{matrix}$

Numerically solving systems of ODEs





General systems of ODEs

- For a general system of ODEs

$$
\begin{aligned}
\frac{dy_1}{dt} &= f_1(y_1, y_2, \dots, y_n), & y_1(0) &= y_{1,0} \\
\frac{dy_2}{dt} &= f_2(y_1, y_2, \dots, y_n), & y_2(0) &= y_{2,0} \\
\vdots \qquad & \qquad \vdots & & \vdots \\
\frac{dy_n}{dt} &= f_n(y_1, y_2, \dots, y_n), & y_n(0) &= y_{n,0}
\end{aligned}
$$

General systems of ODEs – matrix notation

- For a general system of ODEs

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f_1(y_1, y_2, \dots, y_n) \\ f_2(y_1, y_2, \dots, y_n) \\ \vdots \\ f_n(y_1, y_2, \dots, y_n) \end{pmatrix}, \quad y_0 = \begin{pmatrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{n,0} \end{pmatrix}$$

General systems of ODEs

system of equations for numerical integration

output of integration

initial conditions
(as vector)

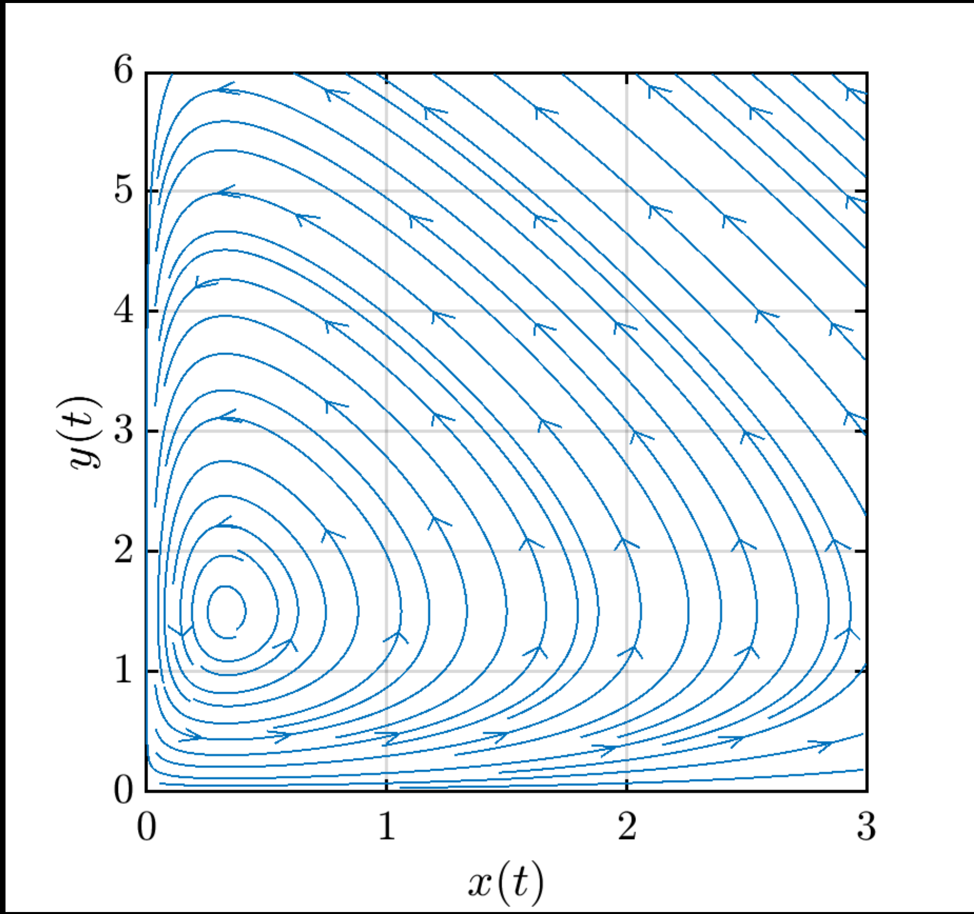
```
[t,y] = ode45(@(t,y) [f1;f2;…;fn], [t0 t1], y0);
```

interval to solve over

independent variable

dependent variables
(as vector)

ASIDE: Phase space analysis



- Phase spaces (called phase planes in 2D) are ways of visualizing characteristics of dynamical systems
- Most often used when analyzing systems in 2D, but can also be used to visualize 3D system behavior
- In 2D, they can identify fixed points, limit cycles, saddle points, or unbounded behavior (in 3D, can also observe chaotic behavior)

ASIDE: Fixed point / steady state analysis

- Of particular interest for people modeling systems like these is the long-term behavior of the components of the system

- In 2D, there are three options for the system:
  - One or both components will tend to grow exponentially (unstable)
  - Both components will tend to specific sets of values (stable)
  - The components will take on the same values periodically (neutrally stable)

ASIDE: Fixed point / steady state analysis

- A system is said to be in steady state when the change in time of all components of the system is zero

- The corresponding steady state values for each component of the system can be found by setting the time derivatives of the system all to zero and solving the corresponding algebraic equations

- From here, the stability of each fixed point (set of values at steady state) can be classified as:
  - unstable – trajectories tend to flow away from the point
  - stable – trajectories tend to flow toward the point
  - neutrally stable – trajectories tend to flow around the point (limit cycle)

ASIDE: Nullclines and trajectory analysis

- Another way to analyze how a system will behave is to consider the nullclines – that is, the curve corresponding to where the change in that component is zero (i.e., the component is not changing with time)

- To better appreciate this approach, consider the following cases:

$$\frac{dx}{dt} > 0:$$

- The value of x will increase as time increases

$$\frac{dx}{dt} < 0:$$

- The value of x will decrease as time increases

$$\frac{dx}{dt} = 0:$$

- The value of x will not change as time increases

ASIDE: Nullclines and trajectory analysis

- Therefore, if we have the following system of two variables:

$$\frac{dx}{dt} = f(x, y), \qquad x(0) = x_0$$

$$\frac{dy}{dt} = g(x, y), \qquad y(0) = y_0$$

- Then plotting the curves corresponding to $f(x, y) = 0$ and $g(x, y) = 0$ in our phase plane will give us rough insights into how the system will be predicted to behave

ASIDE: An example using phase plane analysis

- Let's take a 2D example and think about what its phase plane tells us:

$$\frac{dx}{dt} = 2 - x - y, \qquad x(0) = 2$$

$$\frac{dy}{dt} = x^2 - y, \qquad y(0) = 0$$

- First, let's find the nullclines and then use `streamslice` to see the vector field

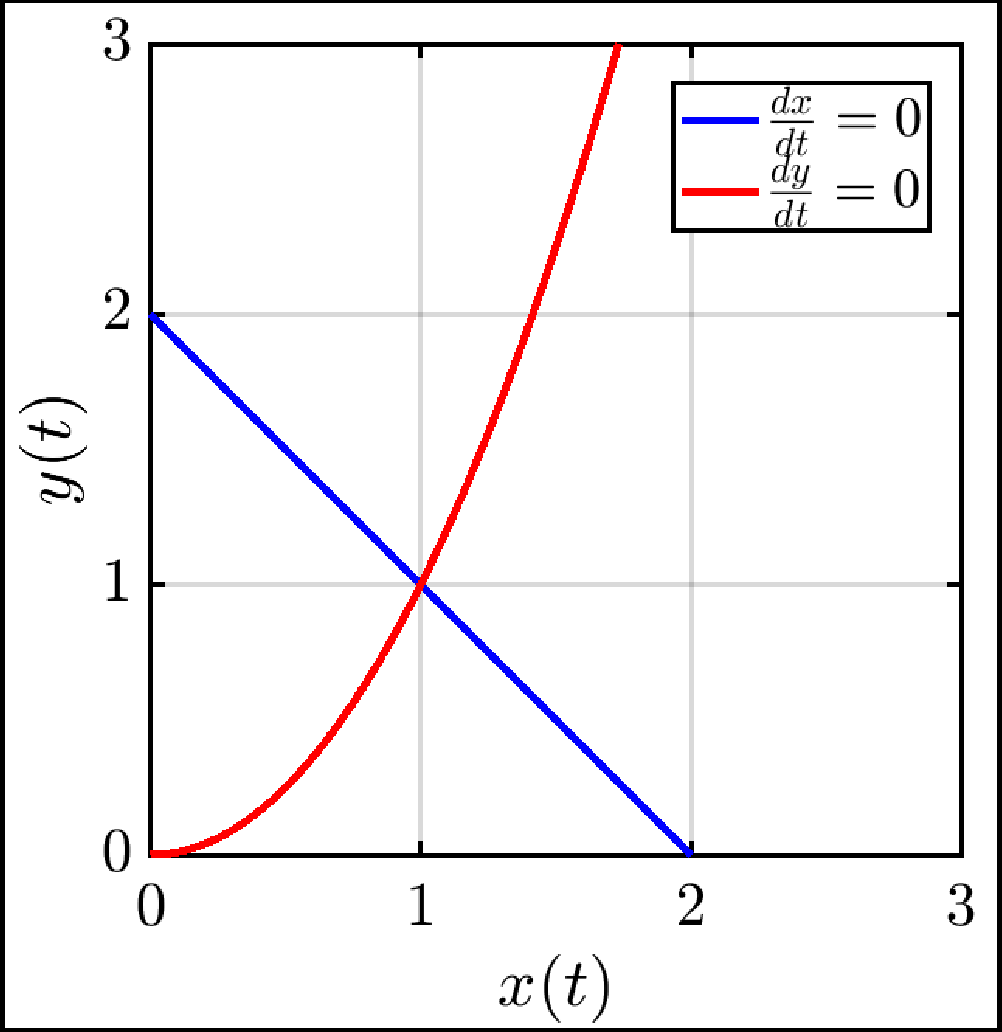
$$\frac{dx}{dt} = 0 = 2 - x - y, \Rightarrow y = -x + 2$$

$$\frac{dy}{dt} = 0 = x^2 - y, \qquad \Rightarrow y = x^2$$

ASIDE: An example using phase plane analysis

```
x = linspace(0,3,100);
y1 = 2-x;
y2 = x.^2;

figure; box on; grid on; hold on;
plot(x_1,y1,'b-','LineWidth',3)
plot(x_1,y2,'r-','LineWidth',3)
```



ASIDE: An example using phase plane analysis

$$\frac{dx}{dt} > 0 \Rightarrow 2 - x - y > 0$$

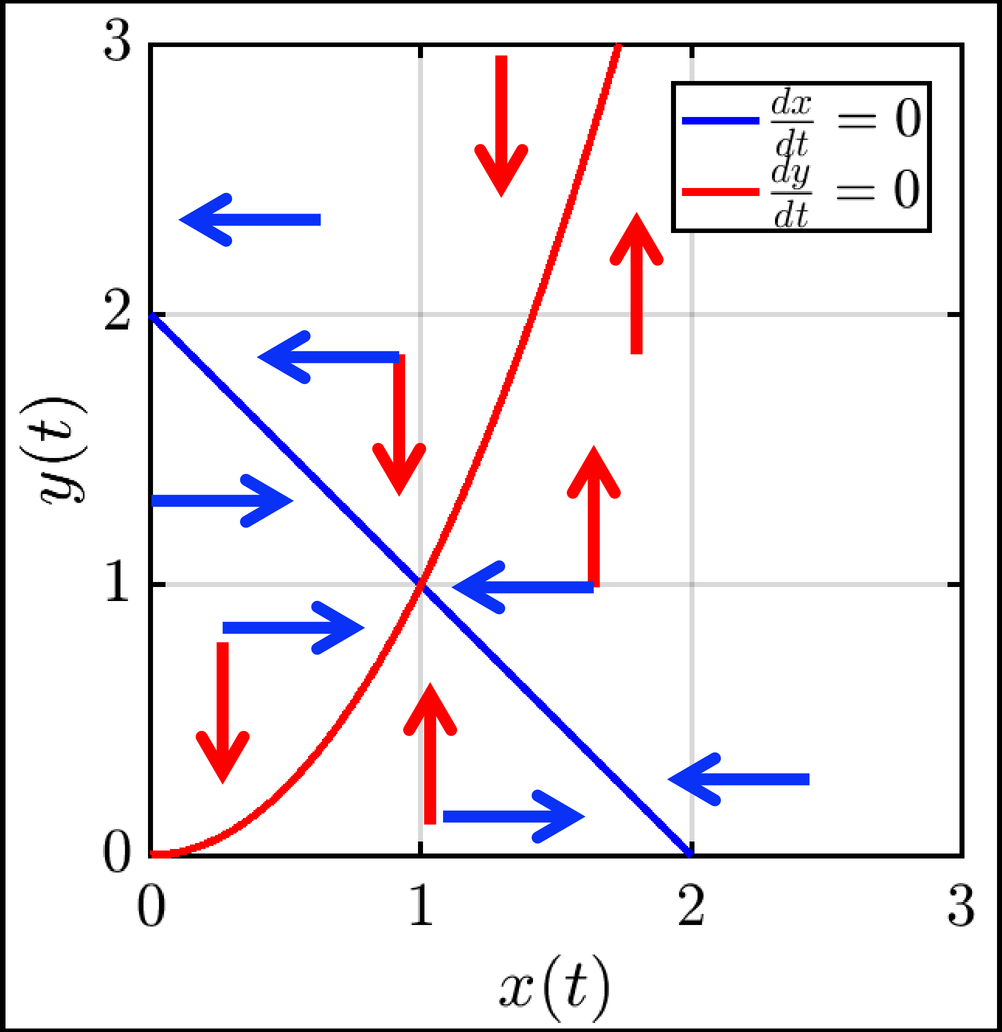
$$\Rightarrow y < 2 - x$$

$$\Rightarrow y > 2 - x$$

$$\frac{dy}{dt} > 0 \Rightarrow x^2 - y > 0$$

$$\Rightarrow y < x^2$$

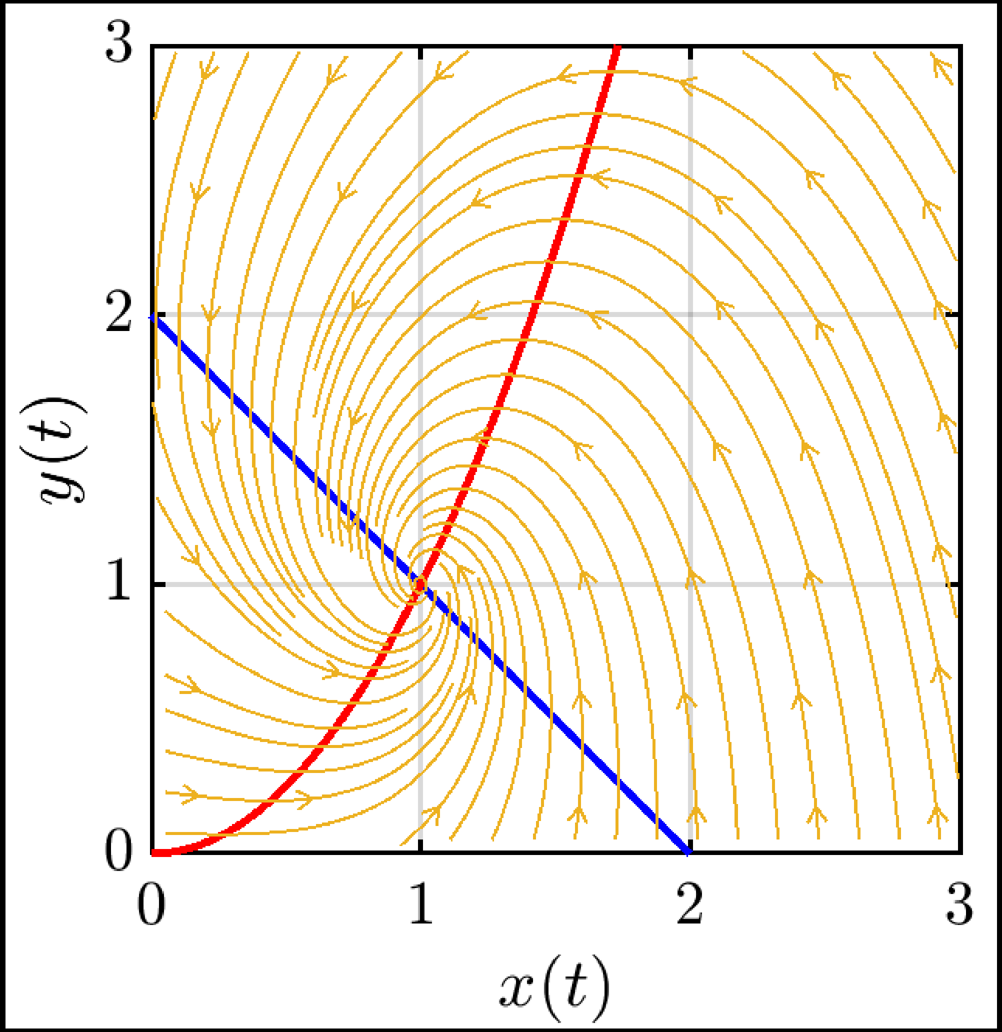
$$\Rightarrow y > x^2$$



ASIDE: An example using phase plane analysis

```
[X,Y] = meshgrid(0:0.25:3,0:0.25:3);
U = 2-X-Y;
V = X.^2-Y;

streamslice(X,Y,U,V)
```



Second order ODEs

- Consider the second order ODE

$$\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + x = 3, \qquad x(0) = 1, \quad \left.\frac{dx}{dt}\right|_{t=0} = 0$$

- So far, we've only talked about solving systems where we only are taking one derivative with respect to each variable being considered

- However, MATLAB can be persuaded to solve equations like this…

Tricking the computer into being clever

- Let $y = \dfrac{dx}{dt}$

- Then $\dfrac{dy}{dt} = \dfrac{d^2x}{dt^2} = 3 - x - 2\dfrac{dx}{dt} = 3 - x - 2y$

- This trick produces a system of ODEs – something we know how to solve:

$$\frac{dx}{dt} = y, \qquad x(0) = 1$$

$$\frac{dy}{dt} = 3 - x - 2y, \qquad y(0) = 0$$

Second order ODEs

system of equations for numerical integration

initial conditions
(as vector)

output of integration

```
[t,y] = ode45(@(t,y) [y(2);3-y(1)-2*y(2)], [0 10], [1 0]);
```

interval to solve over

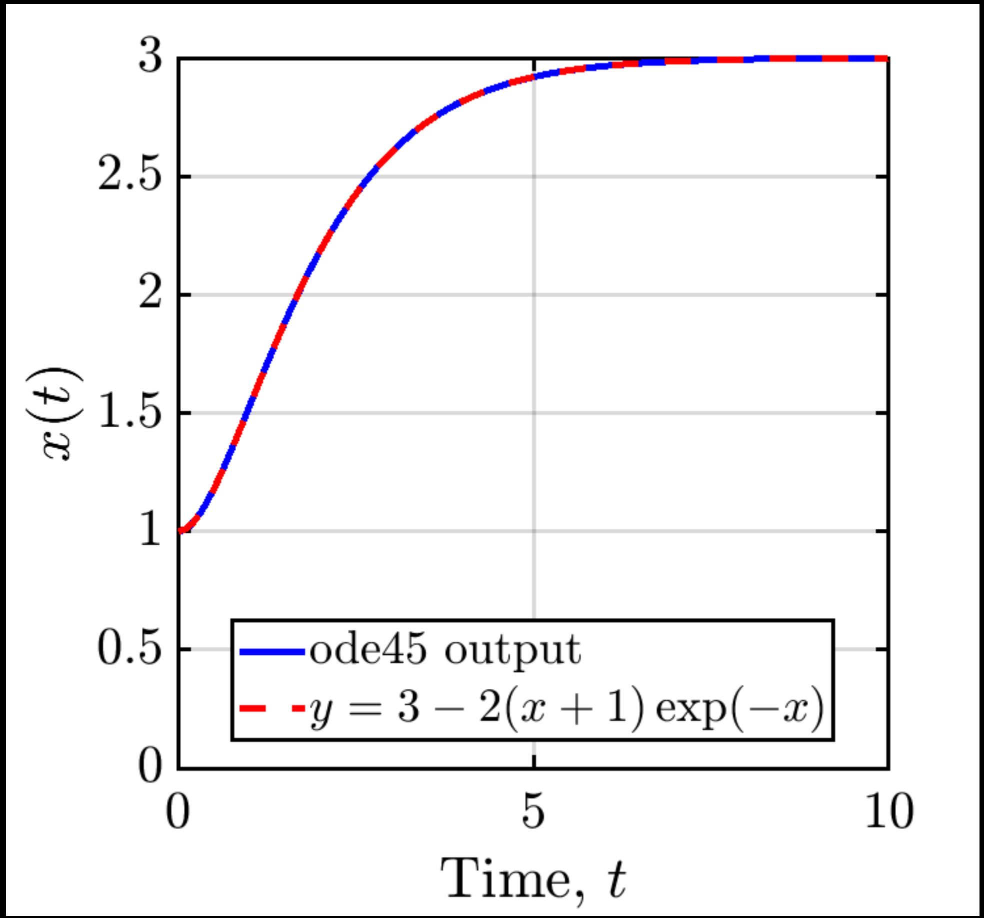
independent variable

dependent variables
(as vector)

Just as before…

- ode45 is doing all the heavy lifting for us and matching the solution to the original second order ODE

- This trick can be applied multiple times to solve for any order ODE



General n^{th} order ODEs

- Consider the general n^{th} order ODE

$$y_1(0) = y_{1,1}$$

$$\frac{d^n y_1}{dt^n} - f\left(y_1, \frac{dy_1}{dt}, \dots \frac{d^{n-1} y_1}{dt^{n-1}}\right) = 0,$$

$$\left.\frac{dy_1}{dt}\right|_{t=0} = y_{1,2}$$

$$\vdots$$

$$\left.\frac{d^{n-1} y_1}{dt^{n-1}}\right|_{t=0} = y_{1,n}$$

Being clever once again

- Let $y_2 = \frac{dy_1}{dt}, \;\; y_3 = \frac{dy_2}{dt}, \; \dots, y_n = \frac{dy_{n-1}}{dt}$

- Then $\frac{dy_n}{dt} = \frac{d^2 y_{n-1}}{dt^2} = \frac{d^n y_1}{dt^n} = f\left(y_1, \frac{dy_1}{dt}, \dots \frac{d^{n-1} y_1}{dt^{n-1}}\right)$

$$= f(y_1, y_2, y_3, \dots, y_n)$$

Being clever once again

$$\frac{dy_1}{dt} = y_2, \qquad y_1(0) = y_{1,1}$$

$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = y_{1,2}$$

$$\vdots \qquad \vdots$$

$$\frac{dy_n}{dt} = f(y_1, y_2, y_3, \dots, y_n) \qquad y_n(0) = y_{1,n}$$

General n^{th} order ODEs

system of equations for numerical integration

initial conditions
(as vector)

output of integration

```
[t,y] = ode45(@(t,y) [y(2); y(3); …; f], [t1 t2], y1);
```

interval to solve over

independent variable

dependent variables
(as vector)

Stiffness and ODE solvers

- If your system of ODEs has multiple timescales or a lot of "problem points," you can run into poor or slow performance when numerically solving
- A "stiff equation" is one where numerical methods can become numerically unstable unless the step size becomes very (sometimes arbitrarily) small
- This can lead to either rapidly growing error between the actual and numerical solution or extremely slow stepping when determining a solution

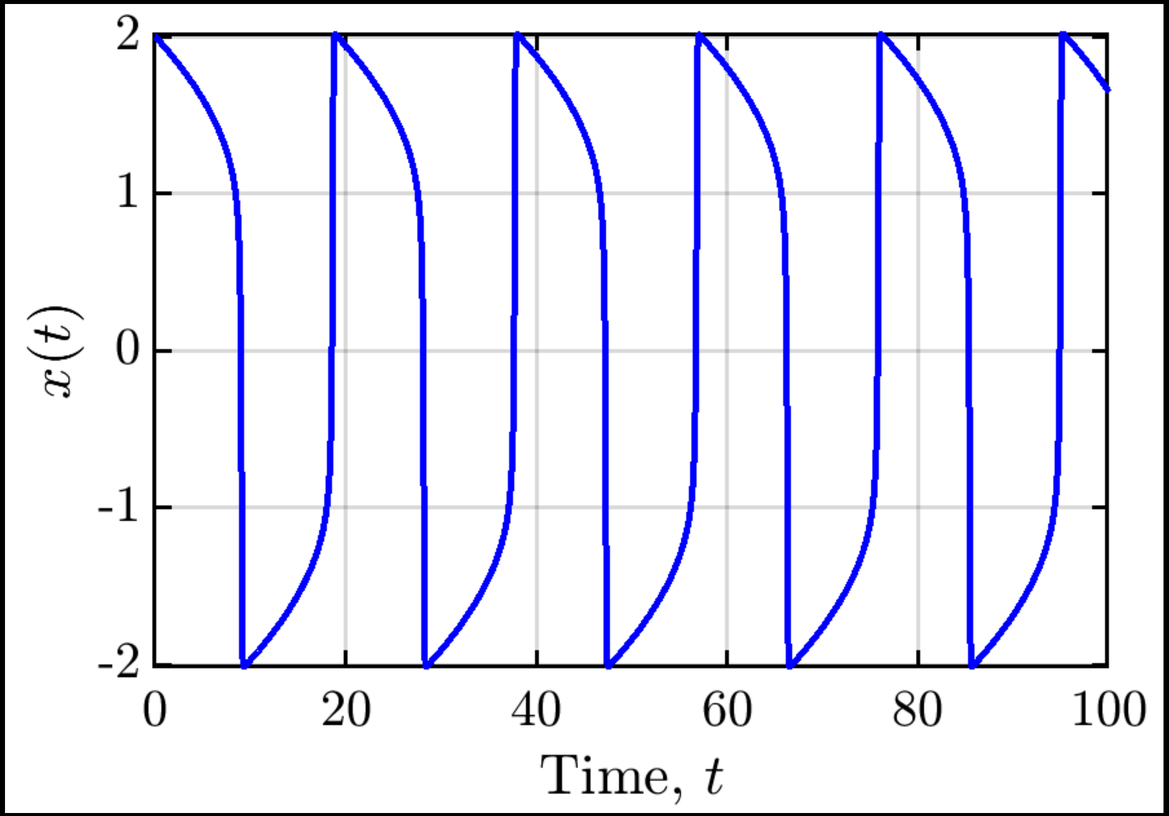
Stiffness and ODE solvers

- Example – van der Pol oscillator:

$$\frac{dx}{dt} = y, \qquad x(0) = 2$$

$$\frac{dy}{dt} = \mu(1 - x^2)y - x, \qquad y(0) = 0$$

$\mu = 10$



Stiffness and ODE solvers

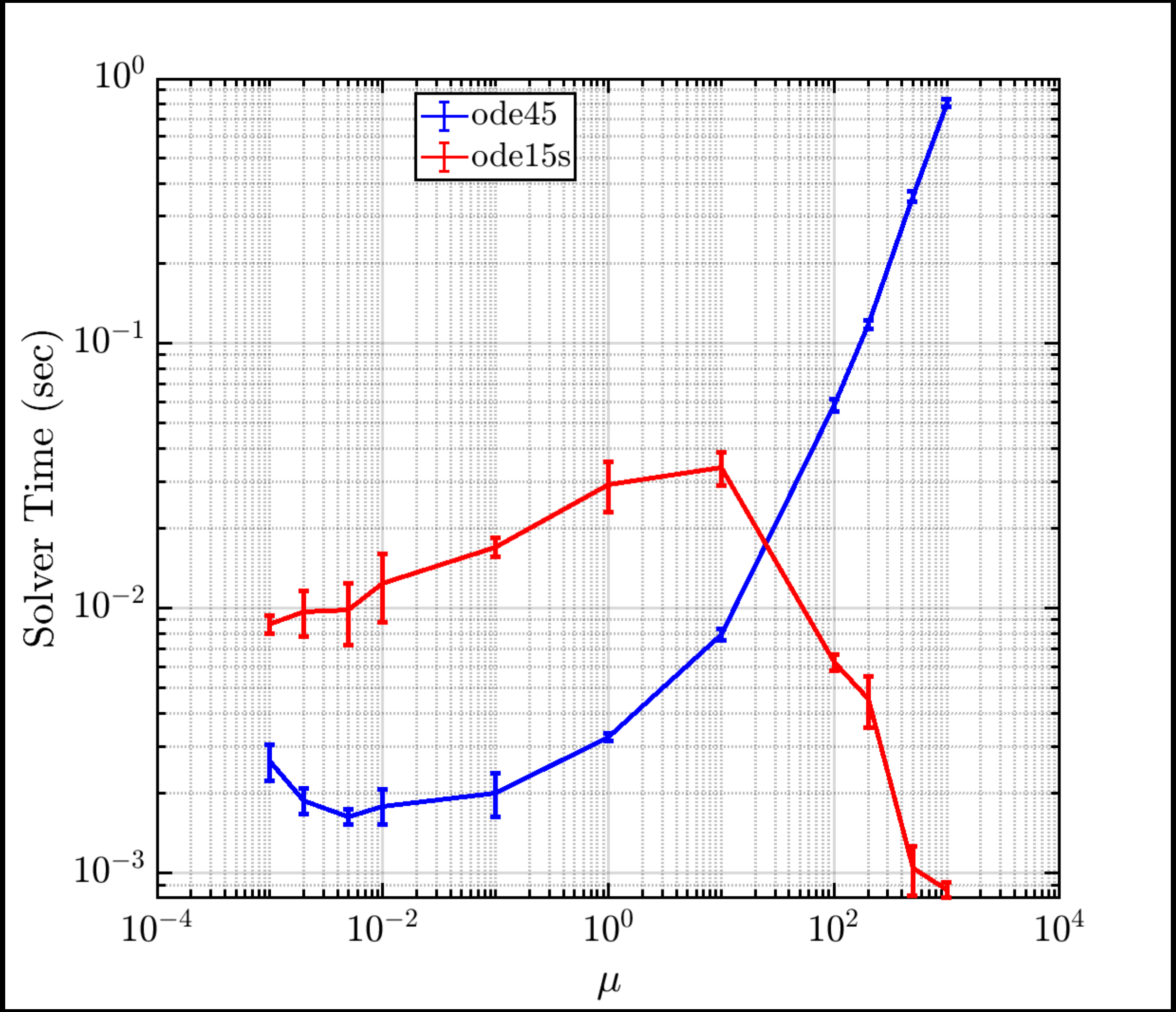
```
[t,y] = ode45(@(t,y) [y(2); mu*(1-y(1).^2).*y(2)-y(1)], [0 200], [2 0])
```

```
[t,y] = ode15s(@(t,y) [y(2); mu*(1-y(1).^2).*y(2)-y(1)], [0 200], [2 0])
```

Stiffness and ODE solvers

- As μ increases, the equation becomes more stiff

- If we use `ode15s` (a stiff ODE solver) rather than `ode45`, we find that for larger values of μ, our system is solver much faster



Stiffness and ODE solvers

- Whenever you have a system of ODEs to solve, even MATLAB recommends first using `ode45` and switching only if it becomes clear that you need a stiff equation solver

- In addition to `ode15s`, `ode23s` is a common stiff solver

- A full list (and when to use them) of ODE solvers in MATLAB can be found at:

  https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html

Summary

- MATLAB is a great tool for numerically solving systems of ODEs
- Phase plane analysis can let us analyze the behavior of our system of ODEs without having to solve the equation itself!
- A clever trick can let us solve ODEs of any order in MATLAB
- `ode45` is a versatile numerical ODE solver, but if equations are stiff, `ode15s` and `ode23s` are faster and tend to be more accurate