Matlab Course: day 3

Differential equations

Gabriela Capo Rangel

OIST

What is a differential equation?

An equation containing *derivatives*

The *order* of a differential equation is the order of the highest derivative in the equation.

1st order: $$\frac{dy}{dx} + xy^2 = 1$$

2nd order: $$\frac{d^2x}{dt^2} = -\frac{k}{m}x$$ (Simple harmonic motion)

nth order:

$$a_0 y + a_1 \frac{dy}{dx} + a_2 \frac{d^2y}{dx^2} + \ldots + a_{n-1}\frac{d^{n-1}y}{dx^{n-1}} + a_n \frac{d^n y}{dx^n} = b$$

What is a differential equation?

Can classify differential equations as:

linear or non-linear

Non-linear equations contain products of the function and its derivatives, or powers higher than one, i.e:

$$y\frac{dy}{dx} \qquad \left(\frac{dy}{dx}\right)^4 \qquad xy^3$$

Ordinary Differential Equations

$\longrightarrow$ contain *ordinary* derivatives

Partial Differential Equations

$\longrightarrow$ contain *partial derivatives*

Why do we need differential equations?

- To describe countless biological, physiological and chemical models
- Especially for Compartment models: One time dependent compartment an exchange between other compartments
- Examples: Predator Pray Model, Hodgkin-Huxley model, brain metabolism models,etc.

Built in Matlab ODE solvers

Non-stiff differential equations		Stiff differential equations	
ode45	Medium accuracy	**ode15s**	Low to medium
ode23	Low accuracy	**ode23s**	Low
ode113	Low to High Accuracy	**ode23t**	Low
		ode23tb	Low

More info: https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html
Or just type helpode45 and you will see all details and alternatives

ODE solver format

- **ode45** performs well with most ODE problems
- Good idea to use it as a first choice
- Stiffness: - appears when you have drastically different time scales. If ode45 returns a stiffness error or if it is extremely slow, you have to choose a stiff solver instead
- [TOUT,YOUT] = **ode45**(ODEFUN,TSPAN,Y0)

ODE solver format

- The ode45 built in solver has the following structure:
  [TOUT,YOUT] = **ode45**(ODEFUN,TSPAN,Y0)
  where:

  ✓ ODEFUN = the rhs function of your differential equation or the system of differential equations

  ✓ TSPAN = [T0 TFINAL]

  ✓ Y0 is the initial condition(s)

  ✓ TOUT = time vector output

  ✓ YOUT = your solution (vector or matrix depending on nb of differential equations)

  ✓ To visualize your solution: PLOT(TOUT,YOUT)

First example

$$\frac{dx}{dt} = 2e^{-t}$$

Which we can rewrite as:

$$\dot{x} = 2e^{-t}$$

Denote the rhs by:

$$f(t, x) = 2e^{-t}$$

Solve: $\dot{x} = f(t, x)$ with initial conditions:

$$x(0) = 0$$

STEP1: Define your rhs function

$$f(t, x) = 2e^{-t} \quad x(0) = 0$$

- How do we define a function in Matlab:
  - In the Editor open a new Script
  - Syntax:

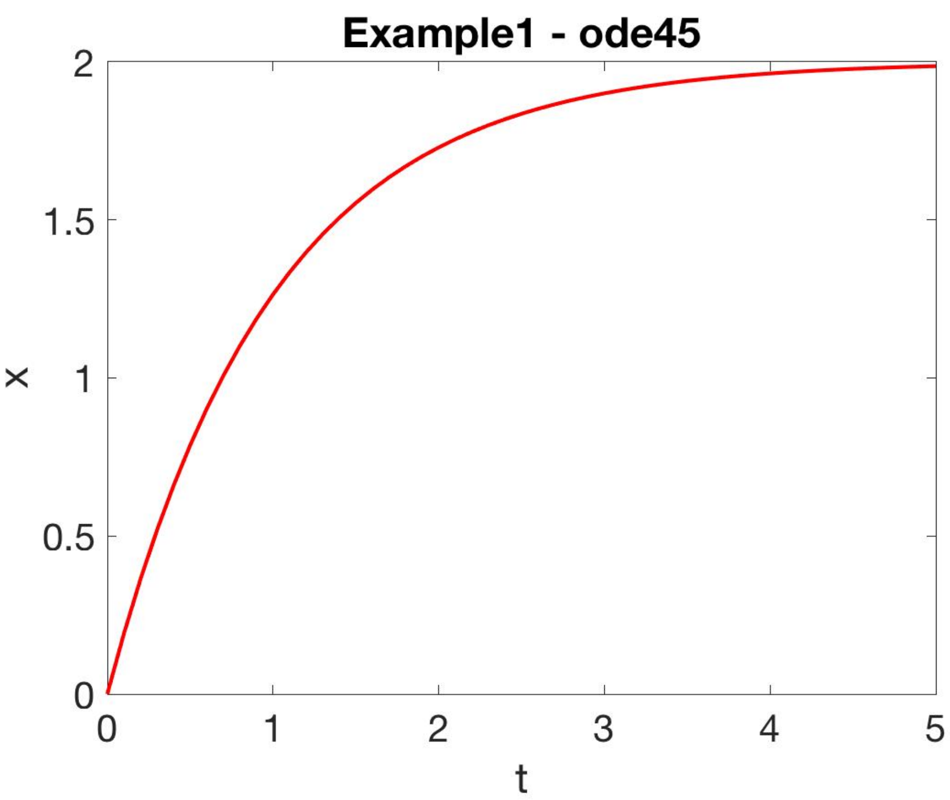
```
function dxdt = rhs(t,x)
        dxdt = 2*exp(-t);
end
```

  - Save the file with the name you chose, i.e. **rhs**.
  - Open a new Script
  - Let's use ode45 and call this function we defined

First example: Matlab code

```
%Step 1: Define time scale
t = 0:0.1:5;
%Step 2: Define initial conditions:
IC = 0;
%Step 3: Call the desired ODE solver
[t,x] = ode45(@rhs,t,IC)

%Step 4: Plot
figure(1)
plot(t,x,'LineWidth',2,'Color','r')
set(gca,'FontSize',20)
xlabel('t');
ylabel('x');
title('Example1 - ode45');
```



Let's calculate the analytical solution

- Our example was: $\dfrac{dx}{dt} = 2e^{-t}$ for $x(0) = 0$
- Let's integrate:

$$x = \int 2e^{-t}dt = -2e^{-t} + c_1$$

- According to the IC:

$$x(0) = 0 \Longrightarrow -2e^{0} + c_1 = 0 \Longrightarrow c_1 = 2$$

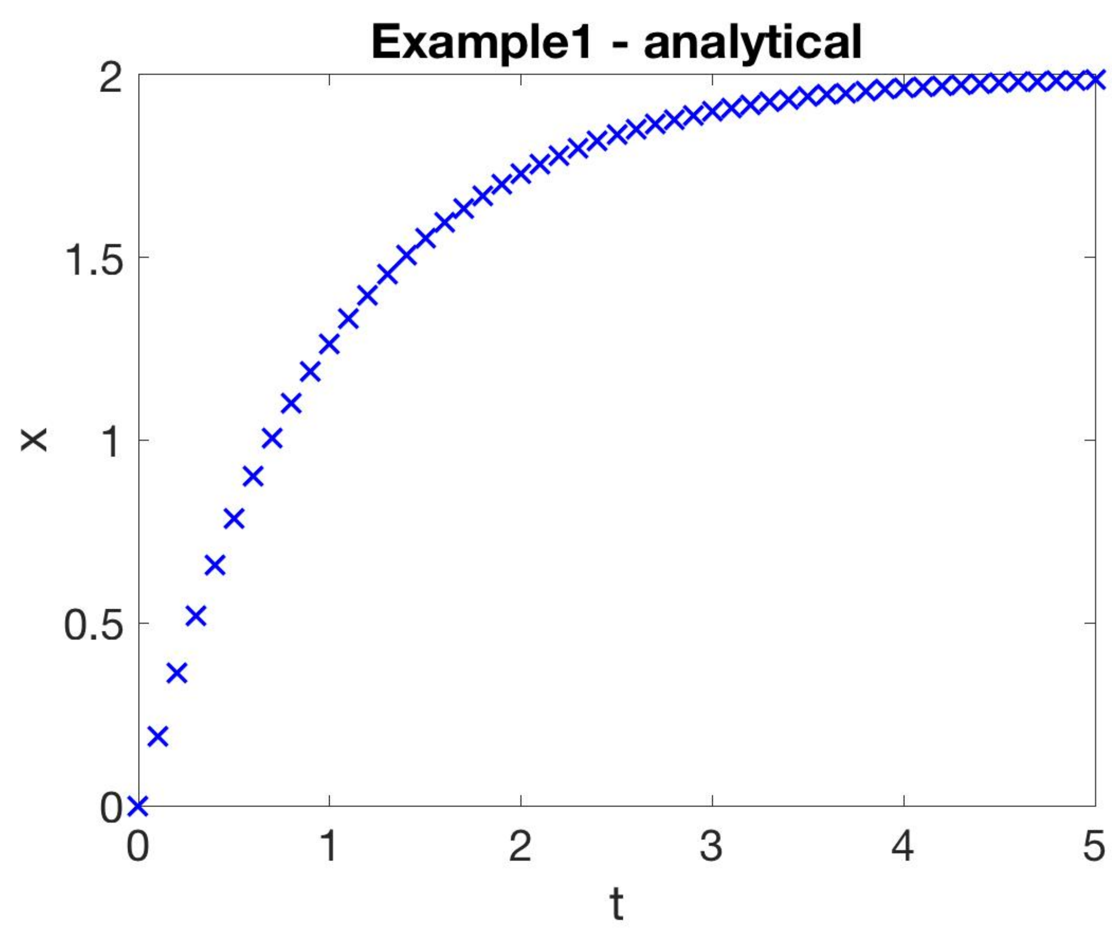
- Our solution is:

$$x(t) = -2e^{-t} + 2$$

Let's plot this!!

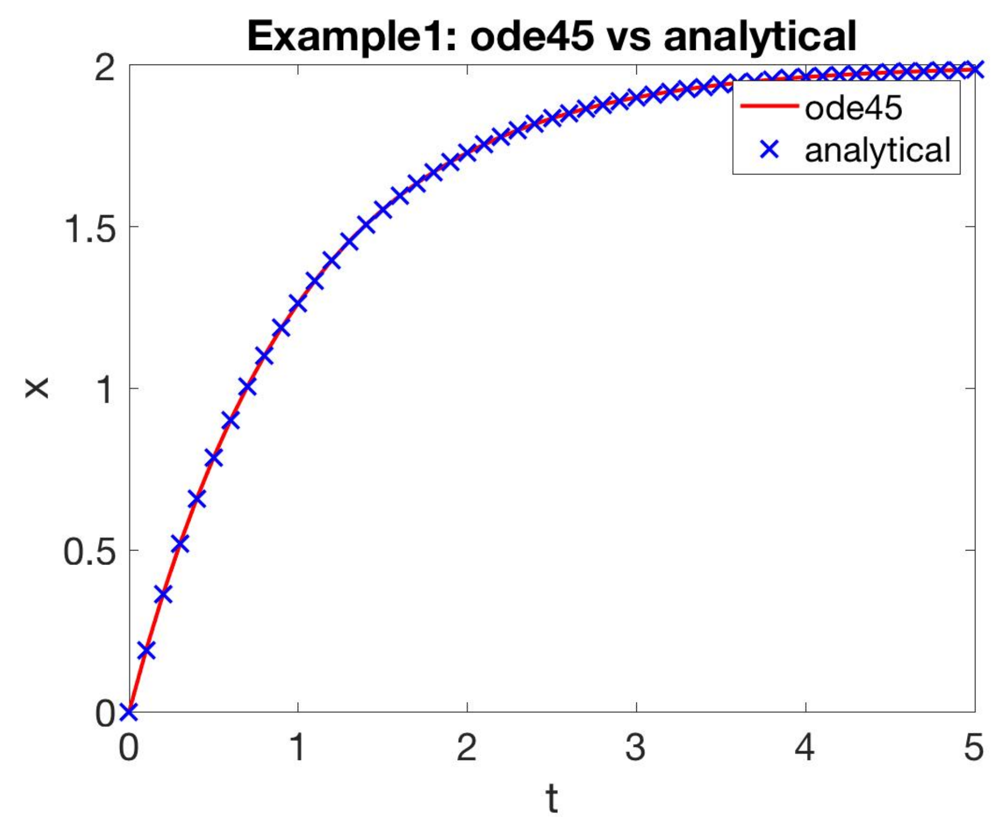
First example - analytical solution

```
%Let's plot the analytical solution:
figure(2)
plot(t,x2,'x','MarkerSize',10,'Color','b','LineWidth',2);
set(gca,'FontSize',20)
xlabel('t');
ylabel('x');
title('Example1 - analytical')
saveas(figure(1),'Example1b.jpg')
```



Plot them on the same graph:

```
%Let's plot the analytical solution:
figure(3)
plot(t,x,'LineWidth',2,'Color','r');
hold on
plot(t,x2,'x','MarkerSize',10,'Color','b','LineWidth',2);
set(gca,'FontSize',20)
xlabel('t');
ylabel('x');
legend('ode45','analytical')
title('Example1: ode45 vs analytical'
saveas(figure(3),'Example1ab.jpg')
```



Example 2: System of ODEs

- Van der Pol equation:

autonomous system
(t doesn't appear explicitly)

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

- Denote: $v = \begin{pmatrix} x \\ y \end{pmatrix}$

- Rewrite the system as:

$$\dot{v}[1] = v[2]$$
$$\dot{v}[2] = (1 - [v[1]]^2)v[2] - v[1]$$

Let's solve this for [0,20] with (v[1],v[2])=(1,0)

Example2: Implementation

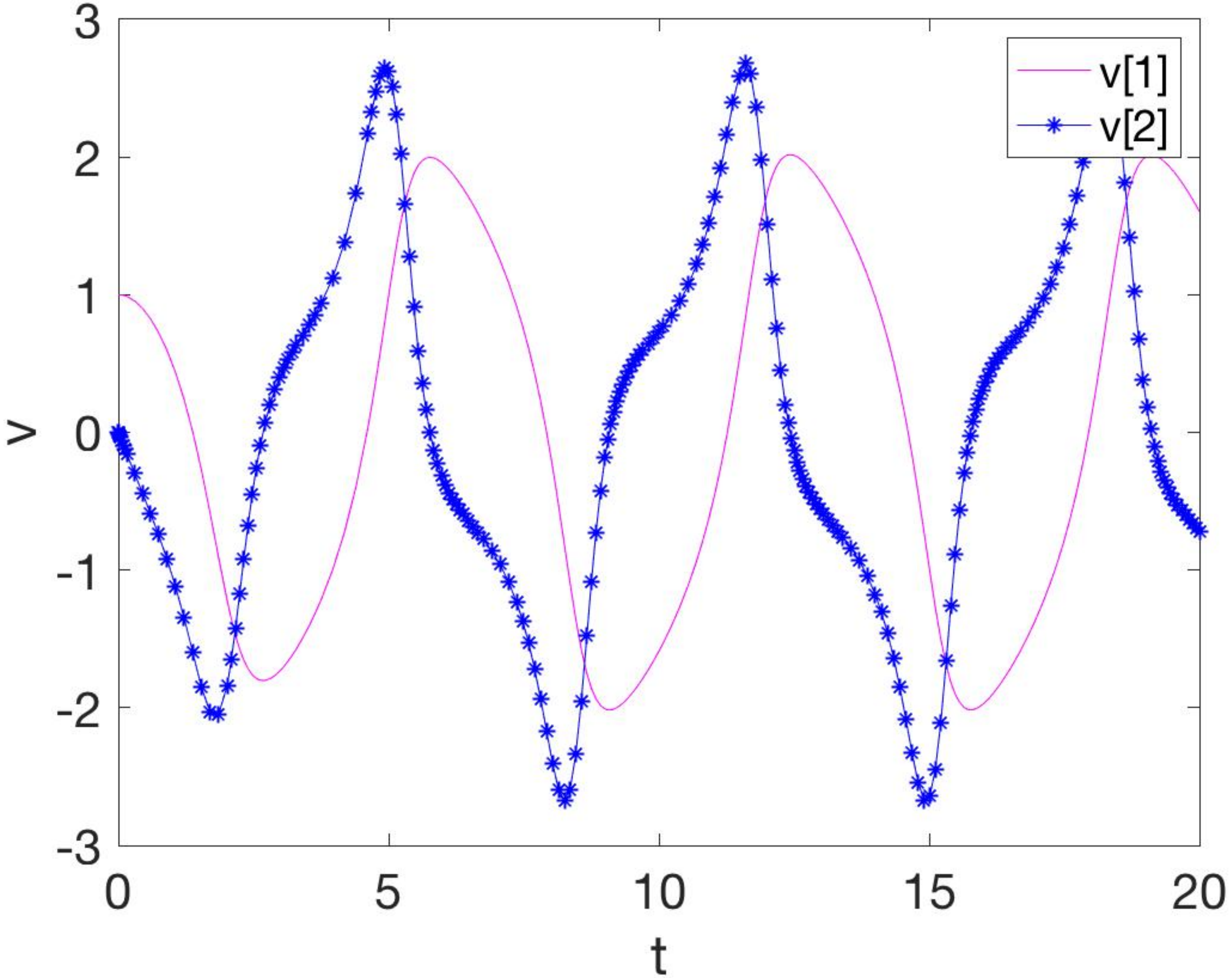
- **Step1:** Write a Matlab function for the rhs

$$\dot{v}[1] = v[2]$$
$$\dot{v}[2] = (1 - [v[1]]^2)v[2] - v[1]$$

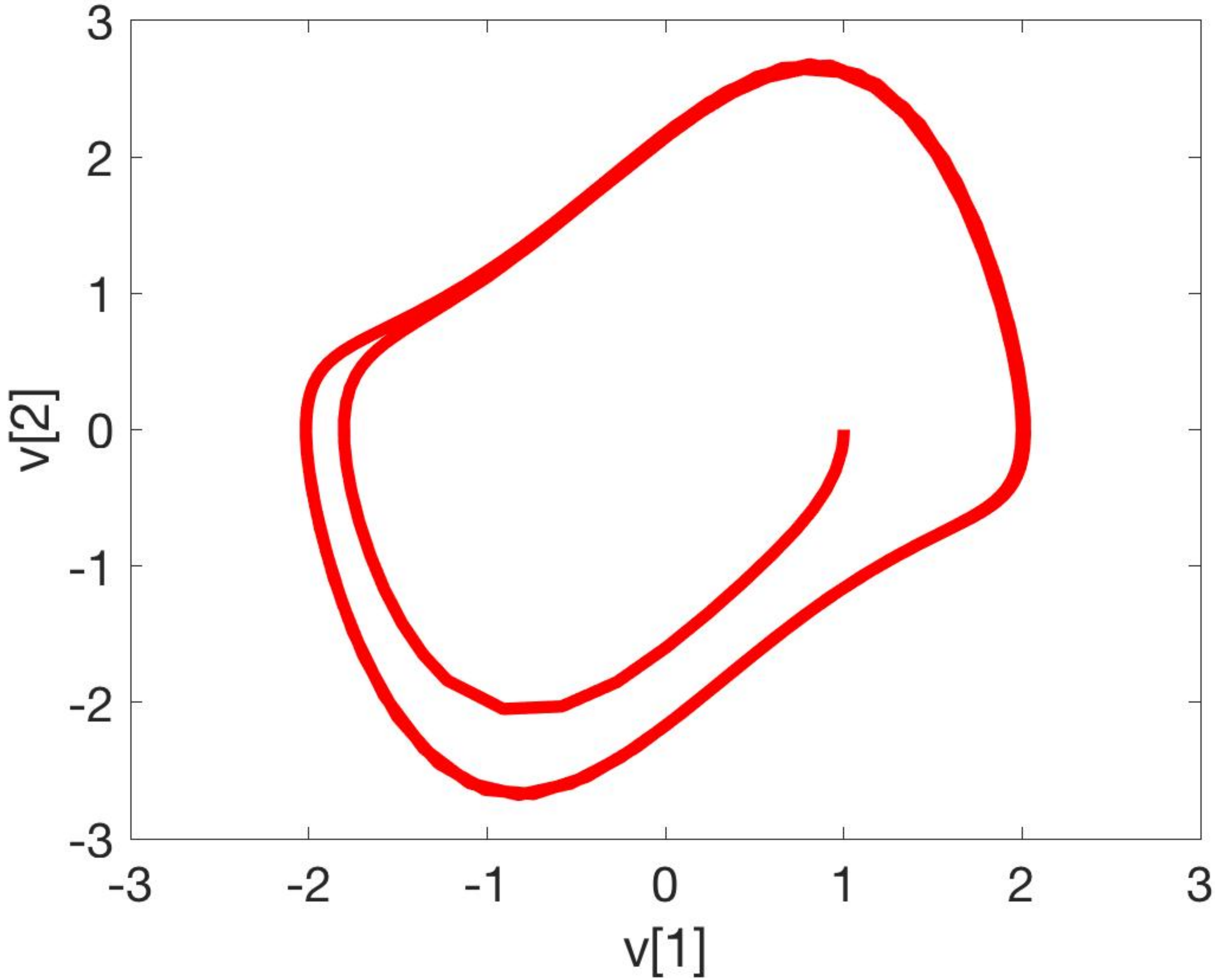
- **Step2:** Solve via ode45 for:
  TI = [0 20] and IC = [1 0]

- **Step3:** Plot your results:
  - Plot1 - plot t vs v[1] and t vs v[2]
  - Plot2 - phase plane plot: v[1] vs v[2]

Example 2: Plots



Example 2: Phase plan Plot



Example3: Let's experience stiffness

- Modify your system of differential equations:

$$\dot{v}[1] = v[2]$$
$$\dot{v}[2] = 1000(1 - [v[1]]^2)v[2] - v[1]$$

It takes forever, right?

- Solve via ode45 for:

  TI = [0 3000] and IC = [1 0]

- Plot v[1] and v[2] in **separate** figures.

- Type size(t) to check how many time steps ode45 requires.

For ode45:
size(t)=6758557

Example3: Let's experience stiffness

- Modify your system of differential equations:

$$\dot{v}[1] = v[2]$$
$$\dot{v}[2] = 1000(1 - [v[1]]^2)v[2] - v[1]$$

- Solve via ode45 for:

  TI = [0 3000] and IC = [1 0]

- Plot v[1] and v[2] in separate figures.

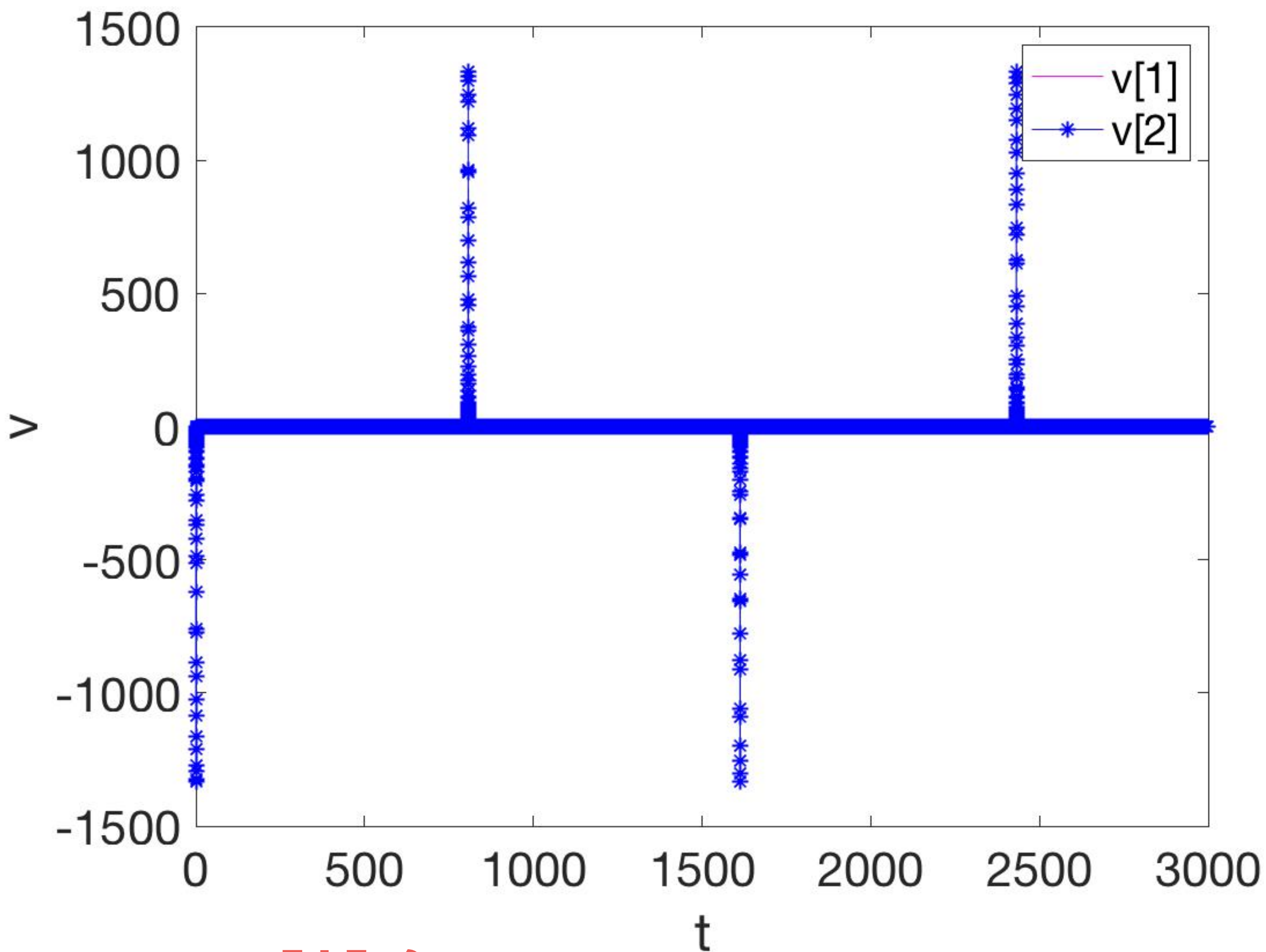
- Try an ODE solver for stiff problems: ode15s or ode23s

So much faster!!!

For od15s:
size(t)=716

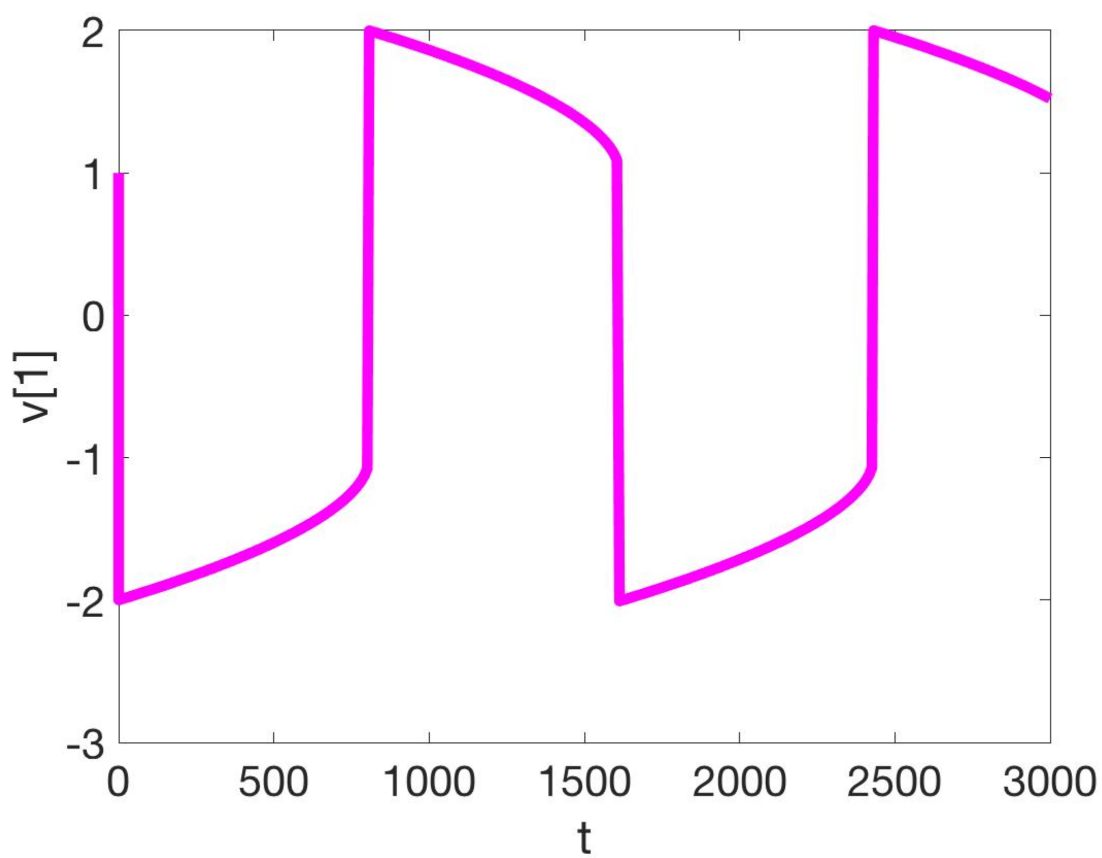
Example3 Plots

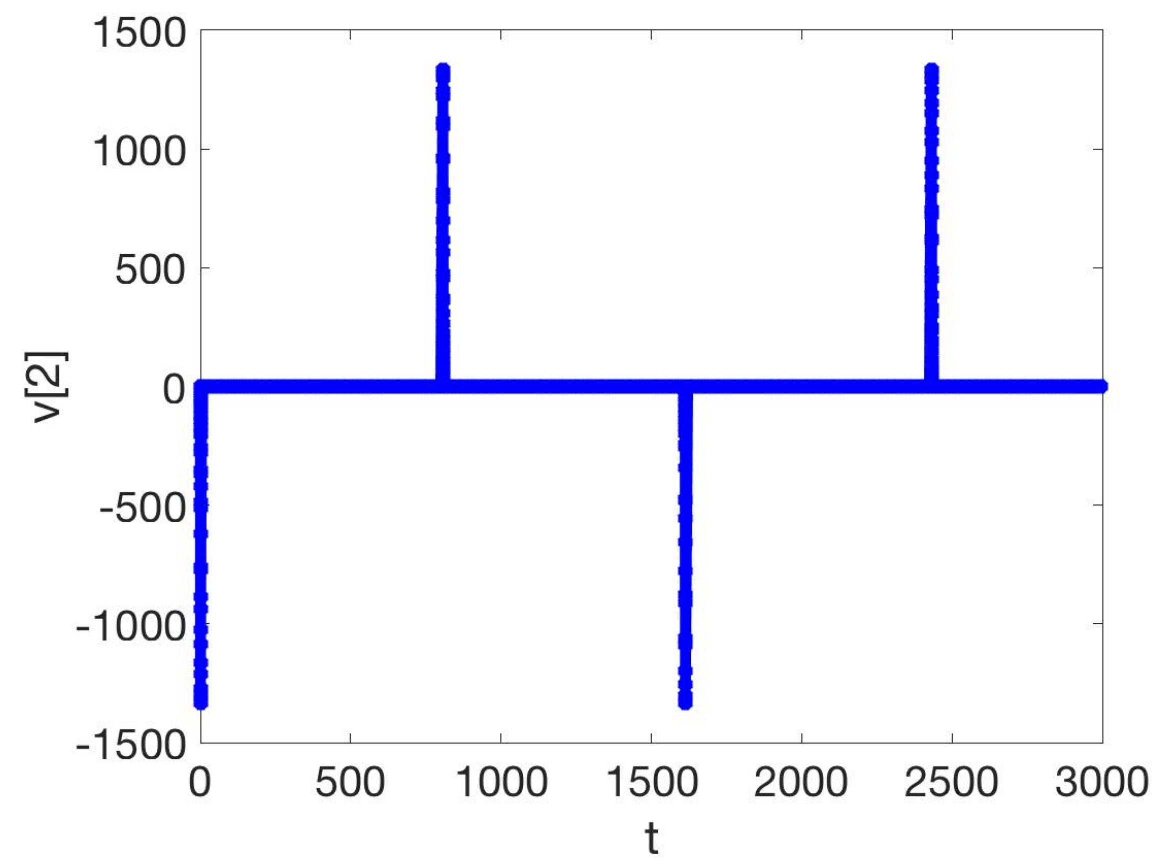
If you try to plot v[1] and v[2] on the same graph:



Where is v[1] ?

Example3 Plots





Take home messages:

A. Careful how you plot variables, they can be on very different time scales

B. Use an adequate Matlab solver, it will save you a lot of time!!

Predator - Prey model (Lotka-Volterra model)

Prey population x

Predator population y

$$\frac{dx}{dt} = Ax - Bxy$$

SOLVE

$$\frac{dy}{dt} = -Cy + Dxy$$

A - Growth rate of prey
B - Rate at which predators destroy prey
C - Rate at which predators die off
D - Growth rate of predators

Lynx vs Snowshoe Hare

Simplifying assumptions:

- The prey has unlimited supply of food
- There is no other threat to the prey
- The predator is dependent on only this type of prey

Predator - Prey model (Lotka-Volterra model)

Prey population x

Predator population y

$$\frac{dx}{dt} = Ax - Bxy$$

SOLVE

$$\frac{dy}{dt} = -Cy + Dxy$$

Let's solve this for:

$$A = B = C = D = 1$$

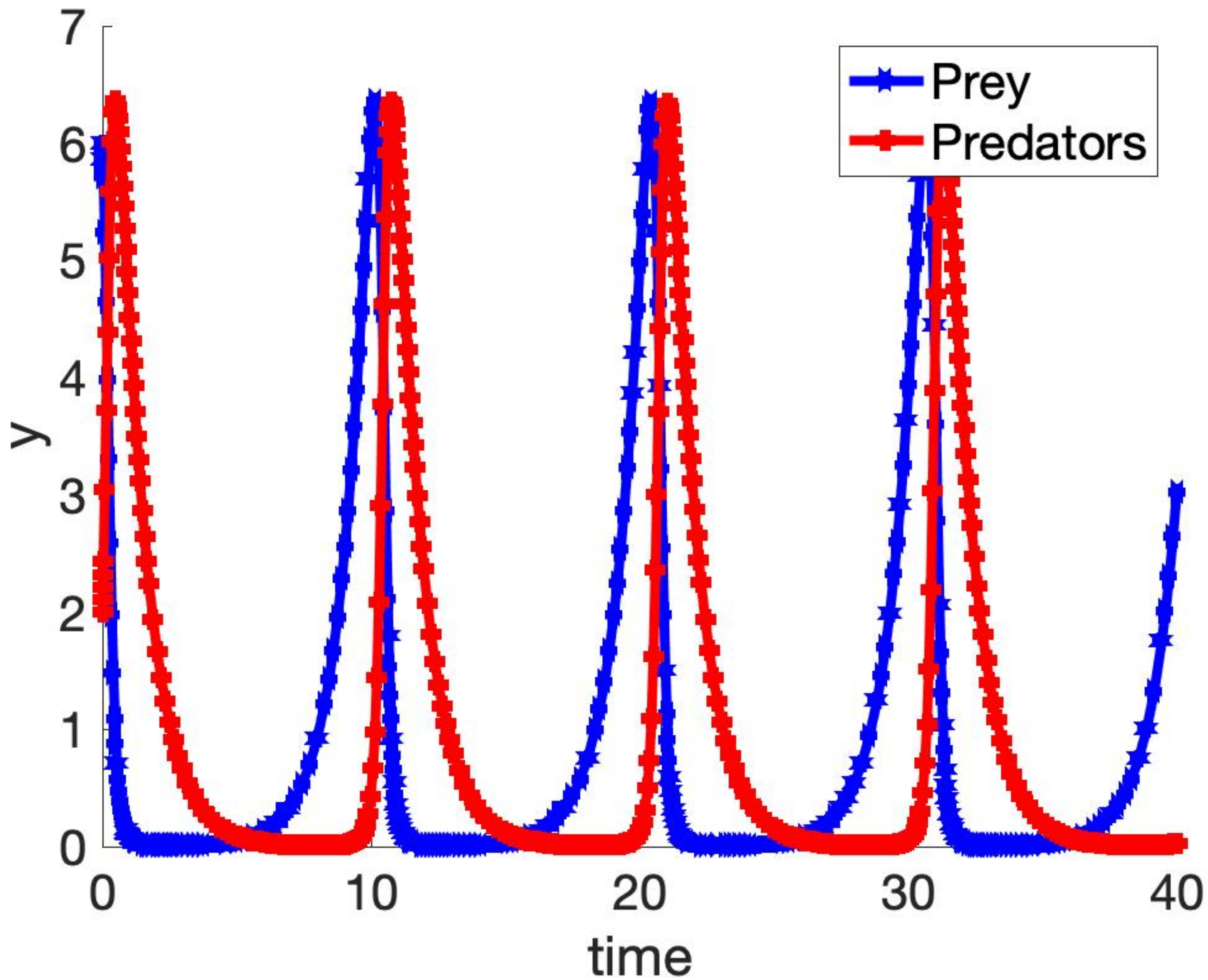
$$X(0) = 6; Y(0) = 2$$

$$TI = [0,40]$$

Lynx vs Snowshoe Hare

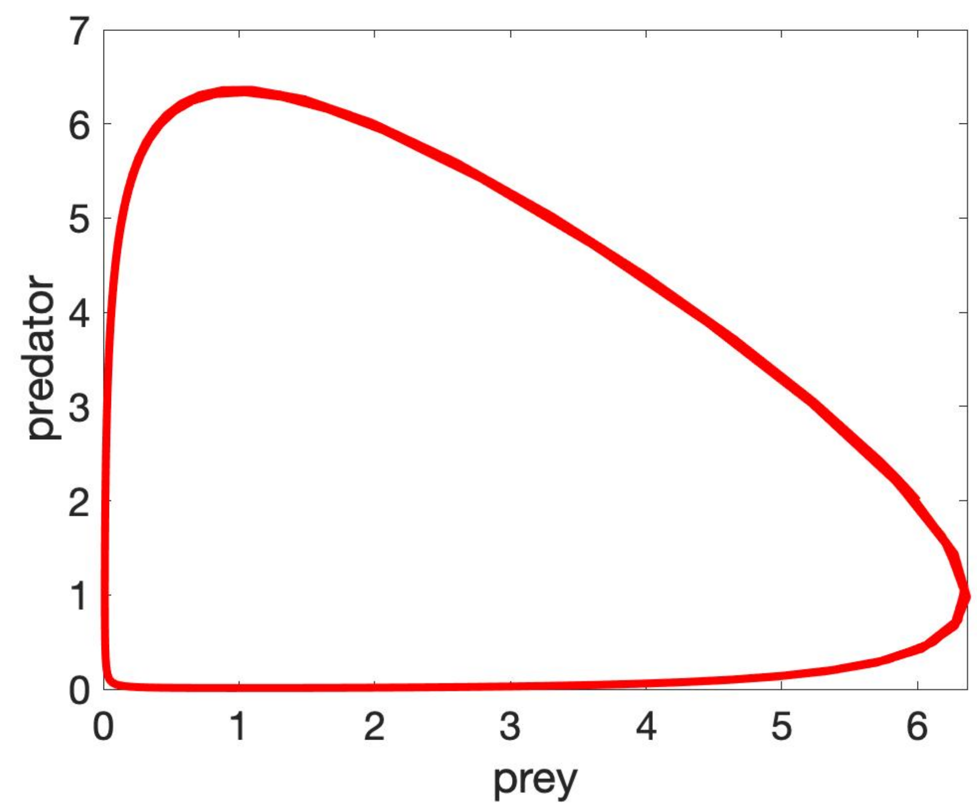
A - Growth rate of prey
B - Rate at which predators destroy prey
C - Rate at which predators die off
D - Growth rate of predators

Predator - Prey model (Lotka-Volterra model)



Result for parameter choice:

$$A = B = C = D = 1$$



Predator - Prey model (Lotka-Volterra model)

SOLVE

$$\frac{dx}{dt} = Ax - Bxy$$

$$\frac{dy}{dt} = -Cy + Dxy$$

Let's solve this for:

$$A = 0.5; B = 0.02;$$

$$C = 0.4; D = 0.004;$$

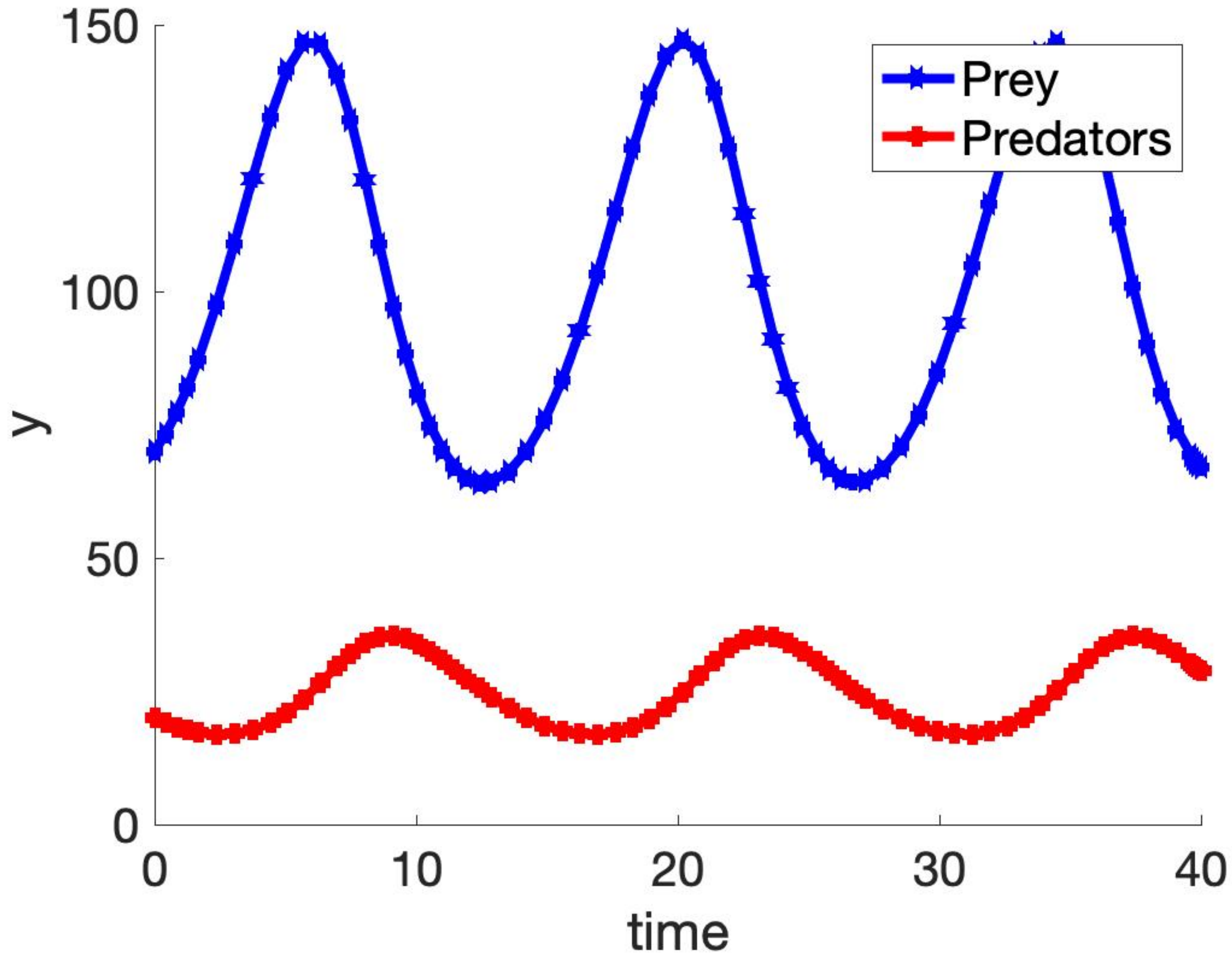
$$X(0) = 70; Y(0) = 20$$

$$TI = [0,40]$$

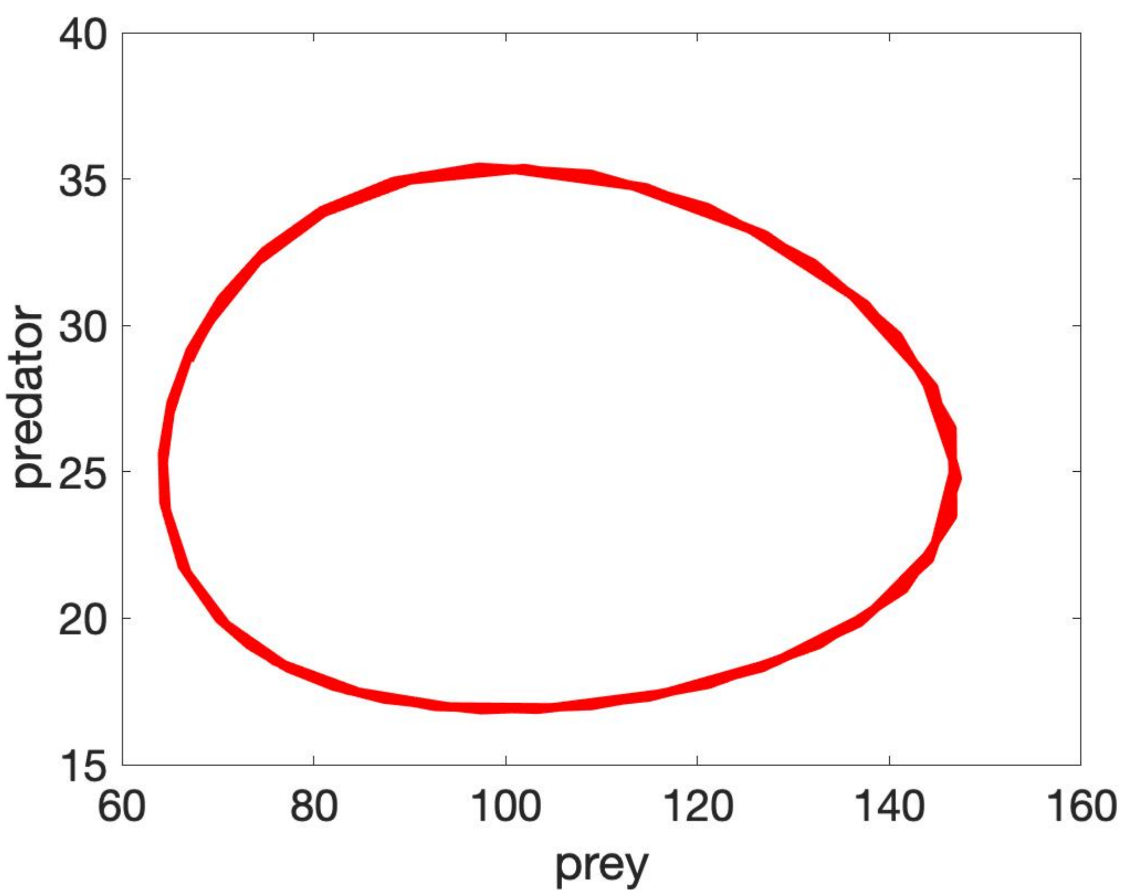
Lynx vs Snowshoe Hare

A - Growth rate of prey
B - Rate at which predators destroy prey
C - Rate at which predators die off
D - Growth rate of predators

Predator - Prey model (Lotka-Volterra model)



$$A = 0.5; B = 0.02;$$
$$C = 0.4; D = 0.004;$$



Predator - Prey model (Lotka-Volterra model)

SOLVE

$$\frac{dx}{dt} = Ax - Bxy$$

$$\frac{dy}{dt} = -Cy + Dxy$$

Let's solve this for:

$$A = 0.5; B = 0.02;$$

$$C = 0.4; D = 0.004;$$

Play with the initial conditions:

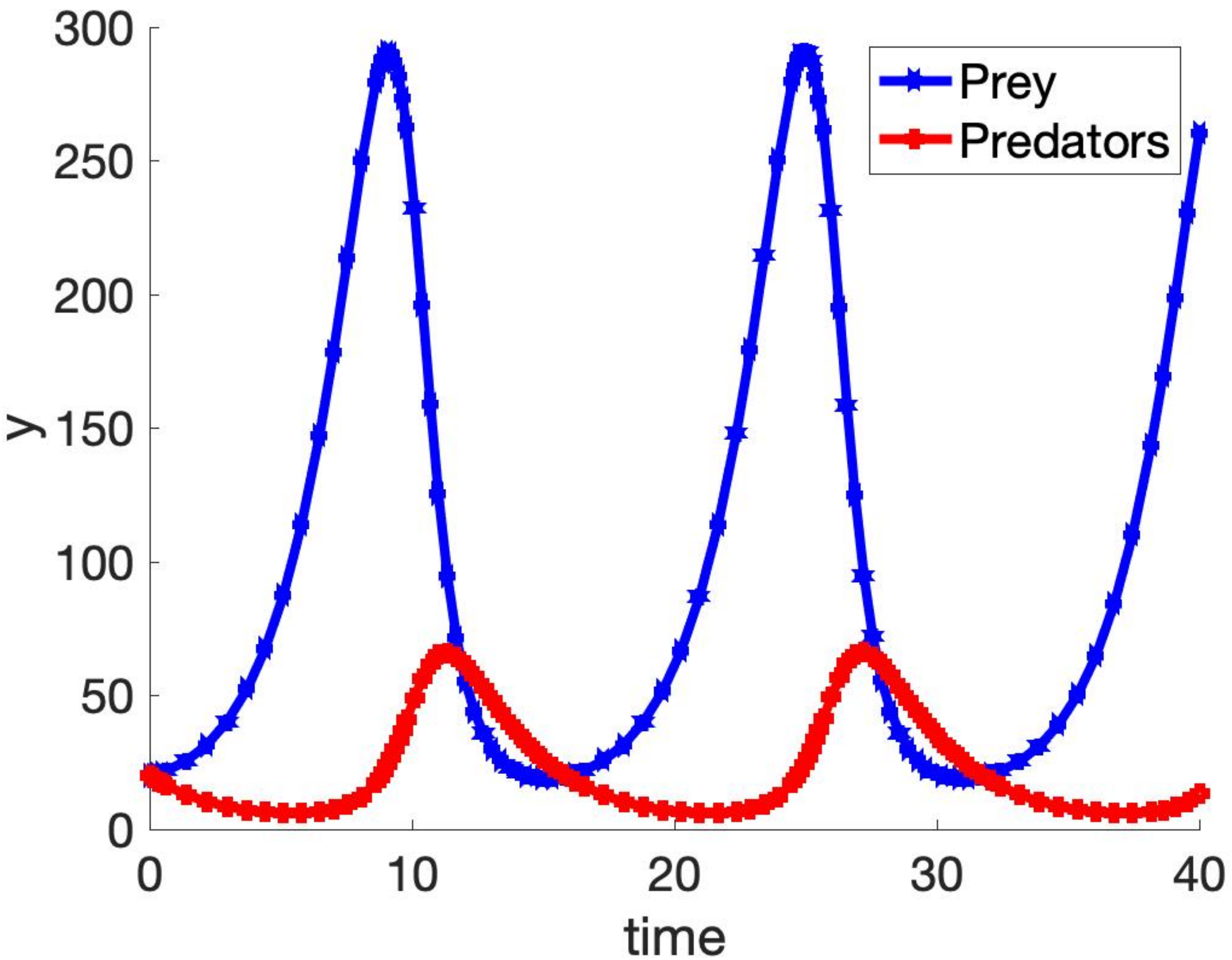
$$X(0) = 20; Y(0) = 20$$

$$TI = [0,40]$$

Lynx vs Snowshoe Hare

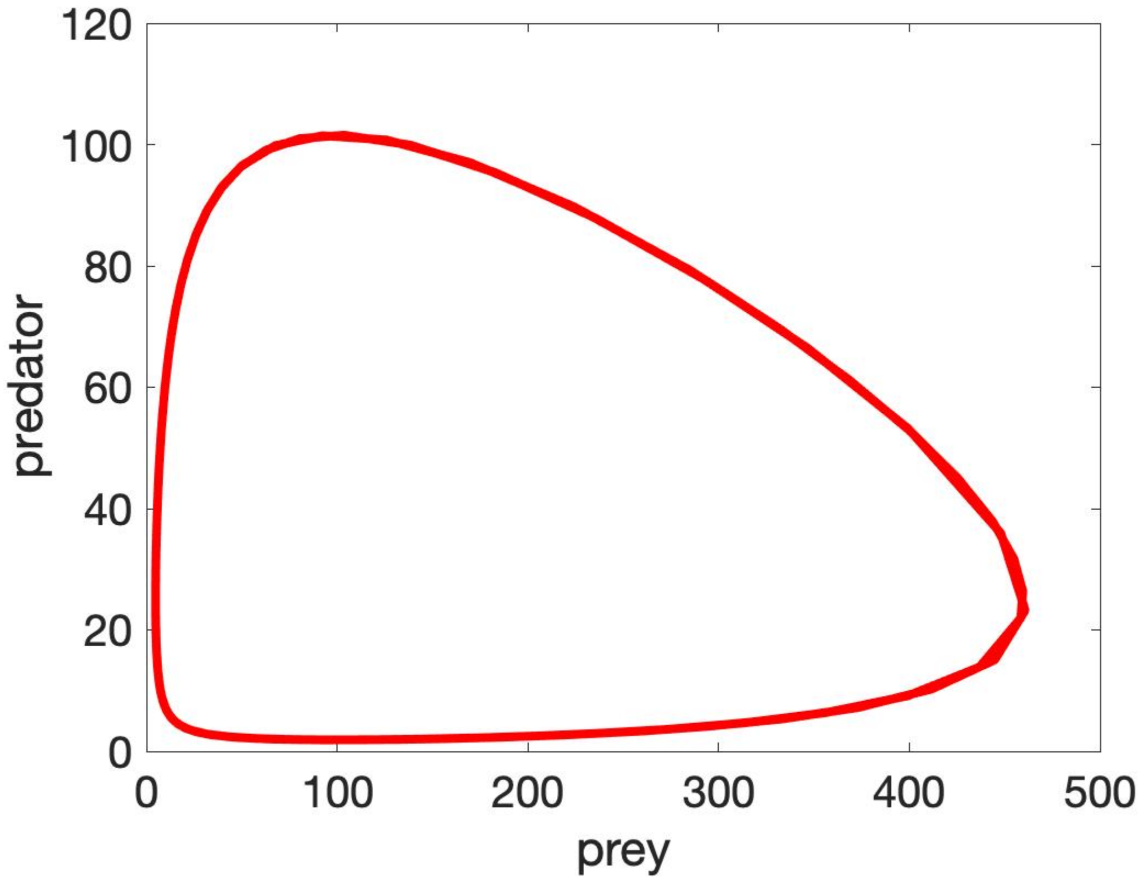
A - Growth rate of prey
B - Rate at which predators destroy prey
C - Rate at which predators die off
D - Growth rate of predators

Predator - Prey model (Lotka-Volterra model)



$$X(0) = 20; Y(0) = 20$$

$$TI = [0,40]$$



Further built-in examples

> help odeexamples

> odeexamples('ode')

→ Run examples

→ View code

Have fun!

Thank you for your attention!

Next class: Image Analysis

Tomorrow: 10am