HSU Colloquium Series

http://www.humboldt.edu/~mathdept/Mathcolloq/colloq.html

Using Matlab in Ordinary Differential Equations

David Arnold

College of the Redwoods
Mathematics Department
email: David-Arnold@Eureka.redwoods.cc.ca.us

http://online.redwoods.cc.ca.us/instruct/darnold

Ordinary Differential Equations

- An *ordinary* differential equation is an equation involving derivatives of an unknown function of a single variable.

Ordinary Differential Equations

- An *ordinary* differential equation is an equation involving derivatives of an unknown function of a single variable.
- For example,

$$\frac{dy}{dt} = y^2 - t$$

is an ordinary differential equation.

Ordinary Differential Equations

- An *ordinary* differential equation is an equation involving derivatives of an unknown function of a single variable.
- For example,

$$\frac{dy}{dt} = y^2 - t$$

  is an ordinary differential equation.
- A differential equation having form

$$y' = f(t, y)$$

  is said to be in *normal form*.

Ordinary Differential Equations

- An *ordinary* differential equation is an equation involving derivatives of an unknown function of a single variable.
- For example,

$$\frac{dy}{dt} = y^2 - t$$

is an ordinary differential equation.

- A differential equation having form

$$y' = f(t, y)$$

is said to be in *normal form*. For example, if $ty' = y + 3t^2$, then solving for y',

$$y' = \frac{1}{t}y + 3t,$$

places the differential equation in normal form.

Solutions of Differential Equations

A *solution* of the first-order, ordinary differential equation $y' = f(t, y)$ is a differentiable function $y(t)$ such that

$$y'(t) = f(t, y(t))$$

for all t in the interval (called the *interval of existence*) where $y(t)$ is defined.

Solutions of Differential Equations

A *solution* of the first-order, ordinary differential equation $y' = f(t, y)$ is a differentiable function $y(t)$ such that

$$y'(t) = f(t, y(t))$$

for all t in the interval (called the *interval of existence*) where $y(t)$ is defined.

For example, simple substitution shows that

$$y = \frac{1}{2 - t}$$

is a solution of the differential equation

$$y' = y^2$$

on the interval $(-\infty, 2)$.

The Initial Value Problem

A first order differential equation together with initial condition,

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

is called an *initial value problem*. A *solution* of an initial value problem is a differentiable function $y(t)$ such that

The Initial Value Problem

A first order differential equation together with initial condition,

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

is called an *initial value problem*. A *solution* of an initial value problem is a differentiable function $y(t)$ such that

1. $y'(t) = f(t, y(t))$ for all t in an interval containing t_0 where $y(t)$ is defined, and

The Initial Value Problem

A first order differential equation together with initial condition,

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

is called an *initial value problem*. A *solution* of an initial value problem is a differentiable function $y(t)$ such that

1. $y'(t) = f(t, y(t))$ for all t in an interval containing t_0 where $y(t)$ is defined, and
2. $y(t_0) = y_0$.

The Initial Value Problem

A first order differential equation together with initial condition,

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

is called an *initial value problem*. A *solution* of an initial value problem is a differentiable function $y(t)$ such that

1. $y'(t) = f(t, y(t))$ for all t in an interval containing t_0 where $y(t)$ is defined, and
2. $y(t_0) = y_0$.

For example,

$$y' = y^2 - t, \qquad y(0) = 1$$

is an initial value problem.

Dfield

John Polking, of Rice University, has written a numerical solver called dfield, which is ideal for solving first order initial value problems.

- The dfield software requires that the MATLAB software is installed.

  http://www.mathworks.com

- The dfield software is available at the following url.

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

- A Java version of dfield is available.

  http://math.rice.edu/~dfield/#java

Existence—An Example

There is *no* guarantee that an arbitrary initial value problem will have a solution. For example, consider the initial value problem

$$ty' = y + 3t^2, \qquad y(0) = 1,$$

which, in normal form, becomes

$$y' = \frac{1}{t}y + 3t, \qquad y(0) = 1.$$

Does this initial value problem have a solution? Let's look at two separate approaches.

Existence—An Example

There is *no* guarantee that an arbitrary initial value problem will have a solution. For example, consider the initial value problem

$$ty' = y + 3t^2, \qquad y(0) = 1,$$

which, in normal form, becomes

$$y' = \frac{1}{t}y + 3t, \qquad y(0) = 1.$$

Does this initial value problem have a solution? Let's look at two separate approaches.

- Analyze the solution with `dfield`.

Existence—An Example

There is *no* guarantee that an arbitrary initial value problem will have a solution. For example, consider the initial value problem

$$ty' = y + 3t^2, \qquad y(0) = 1,$$

which, in normal form, becomes

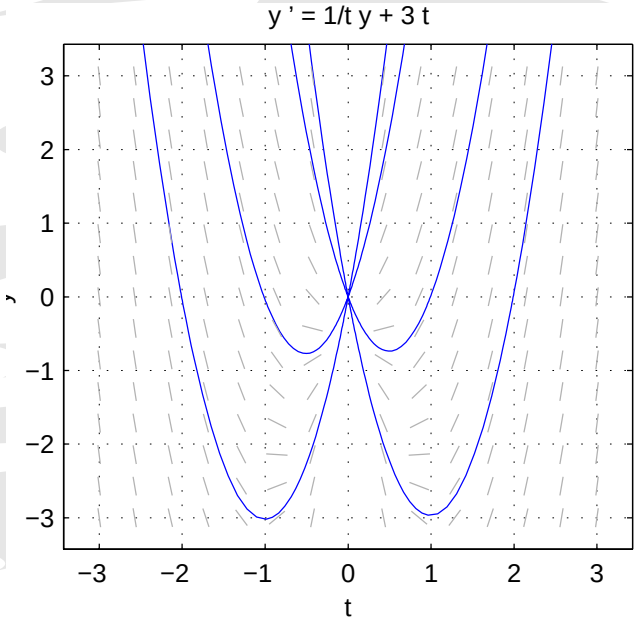
$$y' = \frac{1}{t}y + 3t, \qquad y(0) = 1.$$

Does this initial value problem have a solution? Let's look at two separate approaches.

- Analyze the solution with `dfield`.
- Provide an analytical viewpoint.

No Solution from Dfield

Using `dfield`, it would appear that there are no solutions of $y' = (1/t)y + 3t$ that pass through the point $(0, 1)$.



No Analytical Solution

The equation $y' = (1/t)y + 3t$ is *linear*, and easily solved.

$$y(t) = 3t^2 + Ct$$

No Analytical Solution

The equation $y' = (1/t)y + 3t$ is *linear*, and easily solved.

$$y(t) = 3t^2 + Ct$$

- Note that there is no value of C that will satisfy the condition $y(0) = 1$.

No Analytical Solution

The equation $y' = (1/t)y + 3t$ is *linear*, and easily solved.

$$y(t) = 3t^2 + Ct$$

- Note that there is no value of C that will satisfy the condition $y(0) = 1$.
- However, you can see that the original initial value problem has no solution without solving the equation.

$$ty'(t) = y(t) + 3t^2$$

No Analytical Solution

The equation $y' = (1/t)y + 3t$ is *linear*, and easily solved.

$$y(t) = 3t^2 + Ct$$

- Note that there is no value of C that will satisfy the condition $y(0) = 1$.
- However, you can see that the original initial value problem has no solution without solving the equation.

$$ty'(t) = y(t) + 3t^2$$
$$(0)y'(0) = y(0) + 3(0)^2$$

No Analytical Solution

The equation $y' = (1/t)y + 3t$ is *linear*, and easily solved.

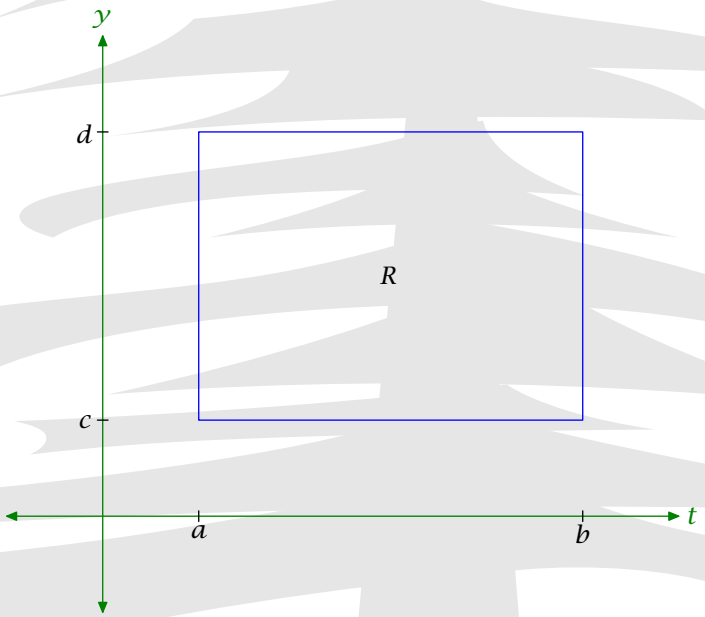
$$y(t) = 3t^2 + Ct$$

- Note that there is no value of C that will satisfy the condition $y(0) = 1$.
- However, you can see that the original initial value problem has no solution without solving the equation.

$$ty'(t) = y(t) + 3t^2$$
$$(0)y'(0) = y(0) + 3(0)^2$$
$$0 = 1 + 0$$

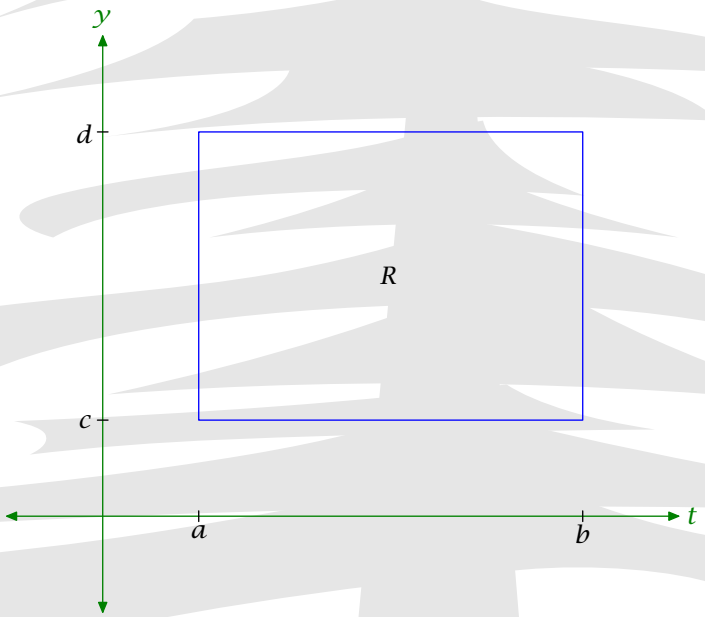
Existence on a Rectangular Region

Consider the following rectangle in the ty-plane.



Existence on a Rectangular Region

Consider the following rectangle in the ty-plane.



Question: When will solutions exist in R?

The Existence Theorem

Suppose that the function $f(t, y)$ is defined and continuous on the Rectangle R in the ty-plane. Then, given any point $(t_0, y_0) \in R$, the initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

The Existence Theorem

Suppose that the function $f(t, y)$ is defined and continuous on the Rectangle R in the ty-plane. Then, given any point $(t_0, y_0) \in R$, the initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

- has a solution $y(t)$ defined in an interval containing t_0, and

The Existence Theorem

Suppose that the function $f(t, y)$ is defined and continuous on the Rectangle R in the ty-plane. Then, given any point $(t_0, y_0) \in R$, the initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

- has a solution $y(t)$ defined in an interval containing t_0, and
- the solution will be defined at least until the solution curve $t \to (t, y(t))$ leaves the rectangle R.

The Existence Theorem

Suppose that the function $f(t, y)$ is defined and continuous on the Rectangle R in the ty-plane. Then, given any point $(t_0, y_0) \in R$, the initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

- has a solution $y(t)$ defined in an interval containing t_0, and
- the solution will be defined at least until the solution curve $t \to (t, y(t))$ leaves the rectangle R.

Question: Does the lack of solution to $y' = (1/t)y + 3t^2$, $y(0) = 1$, contradict the Existence Theorem?

The Existence Theorem

Suppose that the function $f(t, y)$ is defined and continuous on the Rectangle R in the ty-plane. Then, given any point $(t_0, y_0) \in R$, the initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0,$$

- has a solution $y(t)$ defined in an interval containing t_0, and
- the solution will be defined at least until the solution curve $t \to (t, y(t))$ leaves the rectangle R.

Question: Does the lack of solution to $y' = (1/t)y + 3t^2$, $y(0) = 1$, contradict the Existence Theorem?

- No, as $f(t, y) = (1/t)y + 3t$ is not continuous on any rectangle containing the initial condition $y(0) = 1$.

The Uniqueness Theorem

Suppose that the function $f(t,y)$ and its partial derivative $\partial f/\partial y$ are both continuous on the rectangle R in the ty-plane. Suppose that $(t_0,y_0) \in R$ and that the solutions

$$x' = f(t,x) \qquad \text{and} \qquad y' = f(t,y)$$

satisfy

$$x(t_0) = y(t_0) = y_0.$$

The Uniqueness Theorem

Suppose that the function $f(t,y)$ and its partial derivative $\partial f/\partial y$ are both continuous on the rectangle R in the ty-plane. Suppose that $(t_0,y_0) \in R$ and that the solutions

$$x' = f(t,x) \qquad \text{and} \qquad y' = f(t,y)$$

satisfy

$$x(t_0) = y(t_0) = y_0.$$

Then, as long as $(t,x(t))$ and $(t,y(t))$ stay in R, we have

$$x(t) = y(t).$$

The Uniqueness Theorem

Suppose that the function $f(t,y)$ and its partial derivative $\partial f/\partial y$ are both continuous on the rectangle R in the ty-plane. Suppose that $(t_0,y_0) \in R$ and that the solutions

$$x' = f(t,x) \qquad \text{and} \qquad y' = f(t,y)$$

satisfy

$$x(t_0) = y(t_0) = y_0.$$

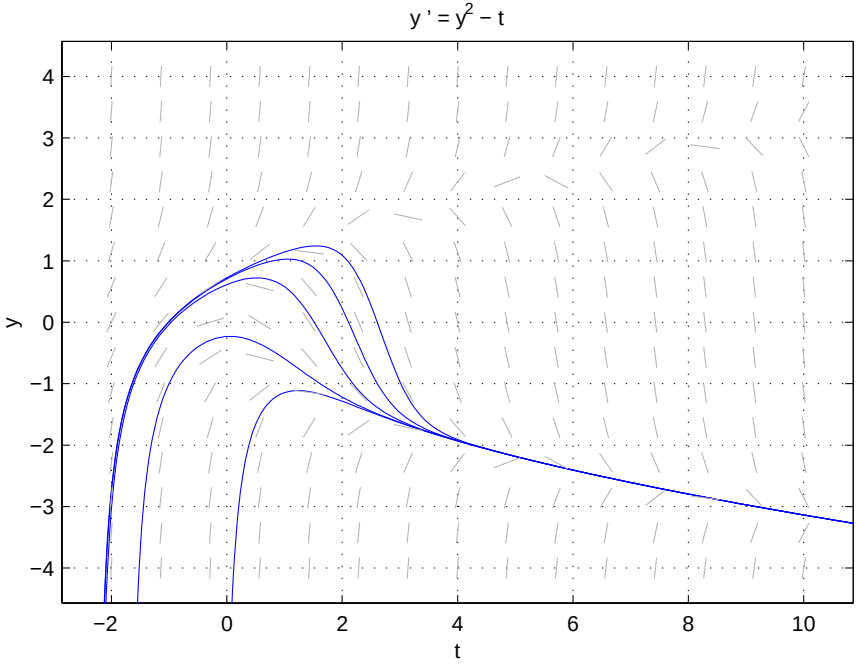
Then, as long as $(t,x(t))$ and $(t,y(t))$ stay in R, we have

$$x(t) = y(t).$$

- **Geometric Interpretation**: Stated geometrically, two distinct solution curves cannot intersect in R.

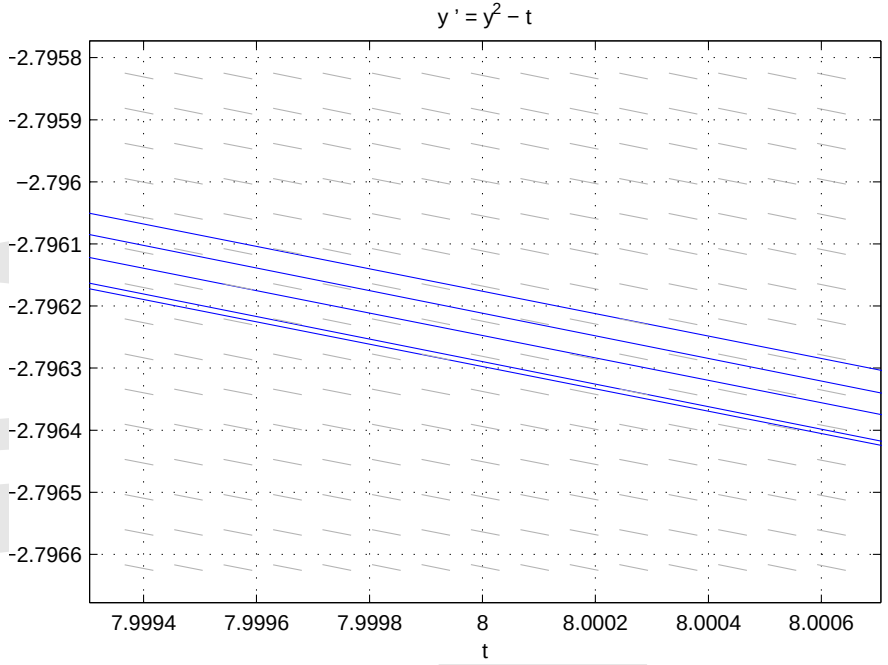
Uniqueness–An Example

Note how solutions of $y' = y^2 - t$ appear to merge into the "ponytail" on the right.



A Closer Look

Zooming in shows that the solutions don't actually intersect, so there is no contradiction of the Uniqueness Theorem.



Qualitative Analysis

The *Logistic equation*,

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P,$$

is an example of an *autonomous* differential equation. Note that the right-hand side of the equation,

$$f(P) = r\left(1 - \frac{P}{K}\right)P,$$

has no explicit dependence on the independent variable t.

Qualitative Analysis

The *Logistic equation*,

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P,$$

is an example of an *autonomous* differential equation. Note that the right-hand side of the equation,

$$f(P) = r\left(1 - \frac{P}{K}\right)P,$$

has no explicit dependence on the independent variable t. Autonomous equations lend themselves nicely to a technique called *qualitative analysis*.

Qualitative Analysis

The *Logistic equation*,

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P,$$

is an example of an *autonomous* differential equation. Note that the right-hand side of the equation,

$$f(P) = r\left(1 - \frac{P}{K}\right)P,$$

has no explicit dependence on the independent variable t. Autonomous equations lend themselves nicely to a technique called *qualitative analysis*.

- Identify the zeros of f. These are called the *equilibrium points* of the equation.

- The equilibrium points, $P = 0$ and $P = K$, lead immediately to the *equilibrium solutions*,

$$P(t) = 0 \qquad \text{and} \qquad P(t) = K,$$

It is trivial to check that these constant solutions satisfy the equation

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P.$$

- The equilibrium points, $P = 0$ and $P = K$, lead immediately to the *equilibrium solutions*,

$$P(t) = 0 \qquad \text{and} \qquad P(t) = K,$$

It is trivial to check that these constant solutions satisfy the equation

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P.$$

- A sketch of $f(P)$ versus P provides detailed information on dP/dt, the rate of change of the population with respect to time.

- The equilibrium points, $P = 0$ and $P = K$, lead immediately to the *equilibrium solutions*,

$$P(t) = 0 \qquad \text{and} \qquad P(t) = K,$$

It is trivial to check that these constant solutions satisfy the equation

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P.$$

- A sketch of $f(P)$ versus P provides detailed information on dP/dt, the rate of change of the population with respect to time.
- Where the graph of f lies *above* the horizontal axis, $dP/dt = f(P) > 0$, and the population is increasing.

- The equilibrium points, $P = 0$ and $P = K$, lead immediately to the *equilibrium solutions*,

$$P(t) = 0 \qquad \text{and} \qquad P(t) = K,$$

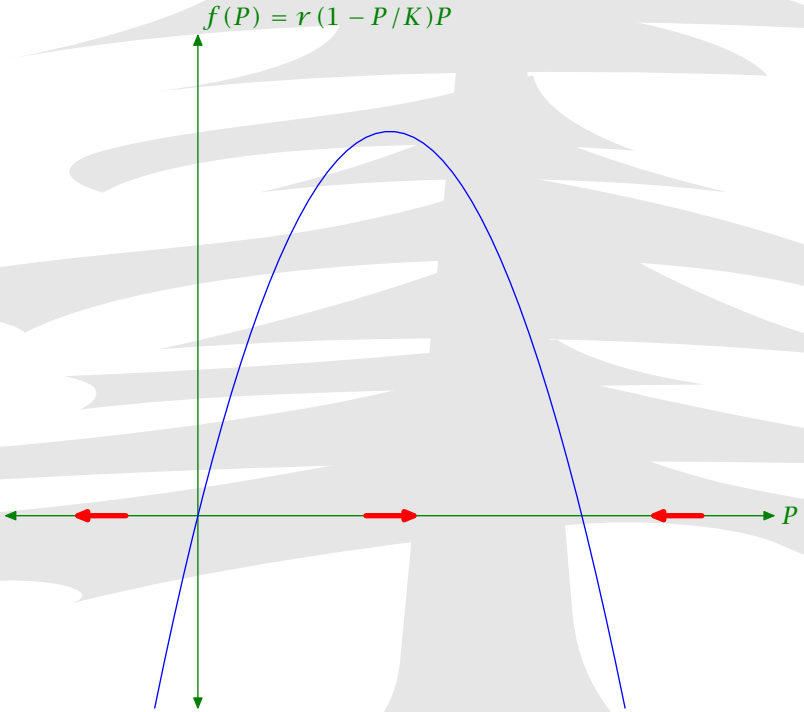
It is trivial to check that these constant solutions satisfy the equation

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right)P.$$

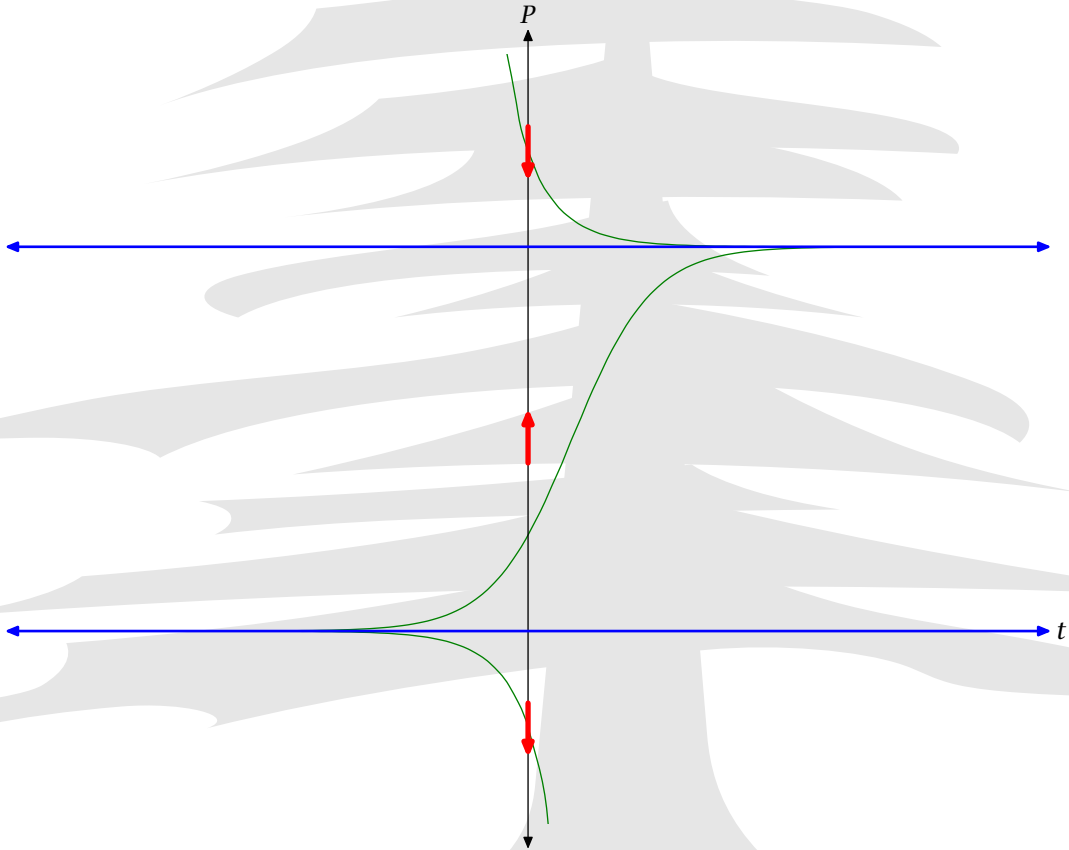
- A sketch of $f(P)$ versus P provides detailed information on dP/dt, the rate of change of the population with respect to time.
- Where the graph of f lies *above* the horizontal axis, $dP/dt = f(P) > 0$, and the population is increasing.
- Where the graph of f lies *below* the horizontal axis, $dP/dt = f(P) < 0$, and the population is decreasing.

The Phase Line

This information is summarized on the horizontal axis, which is known as the *phase line*.

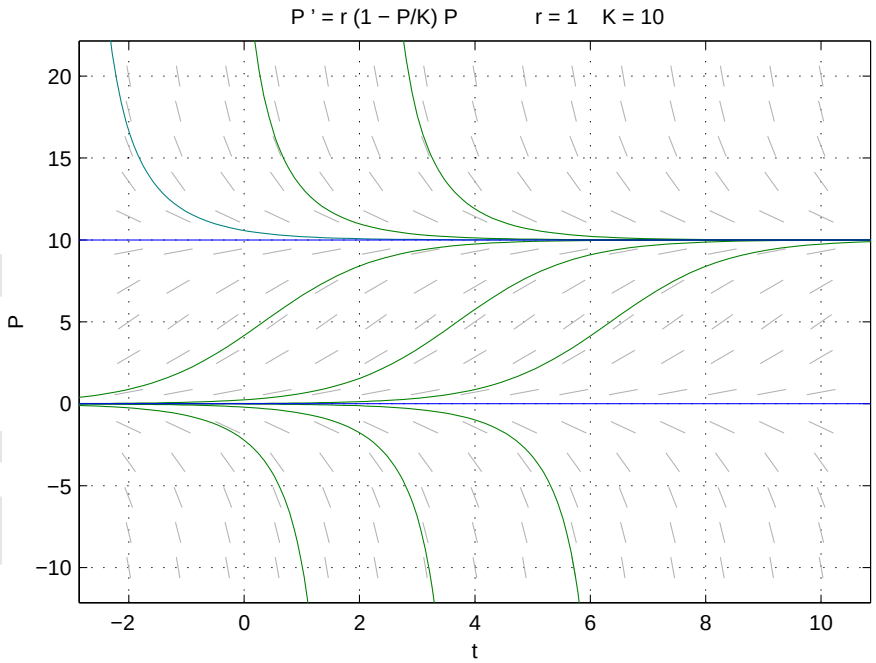


Rotating the Phase Line



Dfield Solution

Dfield easily solves the Logistic equation. The equilibrium solutions are colored blue.



MATLAB's Logical Arrays

In MATLAB, much as on a Texas Instrument's calculator, true statements evaluate as 1, false as 0.

```
>> 3>1
ans =
     1
```

```
>> 3<1
ans =
     0
```

MATLAB's Logical Arrays

In MATLAB, much as on a Texas Instrument's calculator, true statements evaluate as 1, false as 0.

```
>> 3>1
ans =
     1
```

```
>> 3<1
ans =
     0
```

Entire arrays can also be evaluated.

```
>> t=-2:2
t =
    -2    -1     0     1     2
>> t>0
ans =
     0     0     0     1     1
```

Piecewise Defined Functions

MATLAB's logical arrays are used to craft many engineering functions. For example, consider the piecewise defined *step function*.

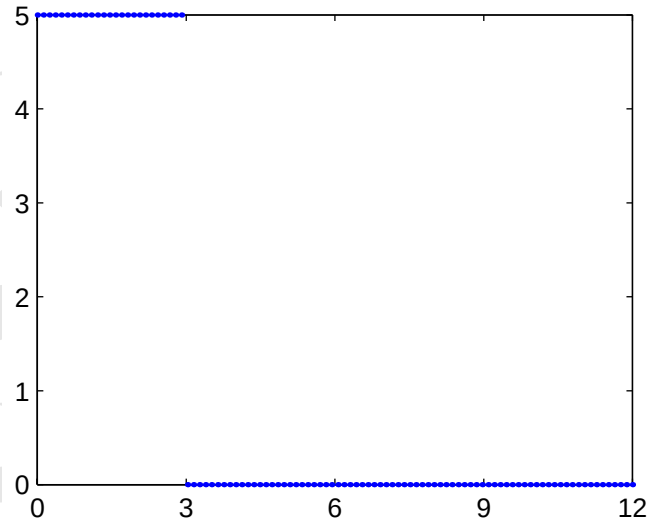
$$H(t) = \begin{cases} 5, & \text{if } t < 3, \\ 0, & \text{if } t \geq 3. \end{cases}$$

This function could model a harvesting policy where fish are taken at a rate of 5 tons per month for the first three months. After that, no harvesting is allowed for the remainder of the year. We can evaluate this function on $[0, 12]$ using logical arrays.

```
>> t=linspace(0,12);
>> y=5*(t<3);
```

A Step Function

The command `plot(t,y,'.')` will produce this image.



Note that the step function is “on” for the first 3 units, then “off” for the remainder of time.

Harvesting with a Step Function

Adjust the earlier Logistic model to take into account our harvesting function.

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right) - H(t), \qquad P(0) = 15$$

Harvesting with a Step Function

Adjust the earlier Logistic model to take into account our harvesting function.

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right) - H(t), \qquad P(0) = 15$$

- `Dfield` understands functions if they are "array smart."

Harvesting with a Step Function

Adjust the earlier Logistic model to take into account our harvesting function.

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right) - H(t), \qquad P(0) = 15$$

- `Dfield` understands functions if they are "array smart."
- Enter the equation in the `Dfield` Setup window as follows:

```
P' = r * (1 - P/K) * P - 5 * (t < 3)
```

Harvesting with a Step Function

Adjust the earlier Logistic model to take into account our harvesting function.

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right) - H(t), \qquad P(0) = 15$$

- `Dfield` understands functions if they are "array smart."
- Enter the equation in the `Dfield` Setup window as follows:

```
P' = r * (1 - P/K) * P - 5 * (t < 3)
```

- Enter parameters

r = 1 and K = 10.

Harvesting with a Step Function

Adjust the earlier Logistic model to take into account our harvesting function.

$$\frac{dP}{dt} = r\left(1 - \frac{P}{K}\right) - H(t), \qquad P(0) = 15$$

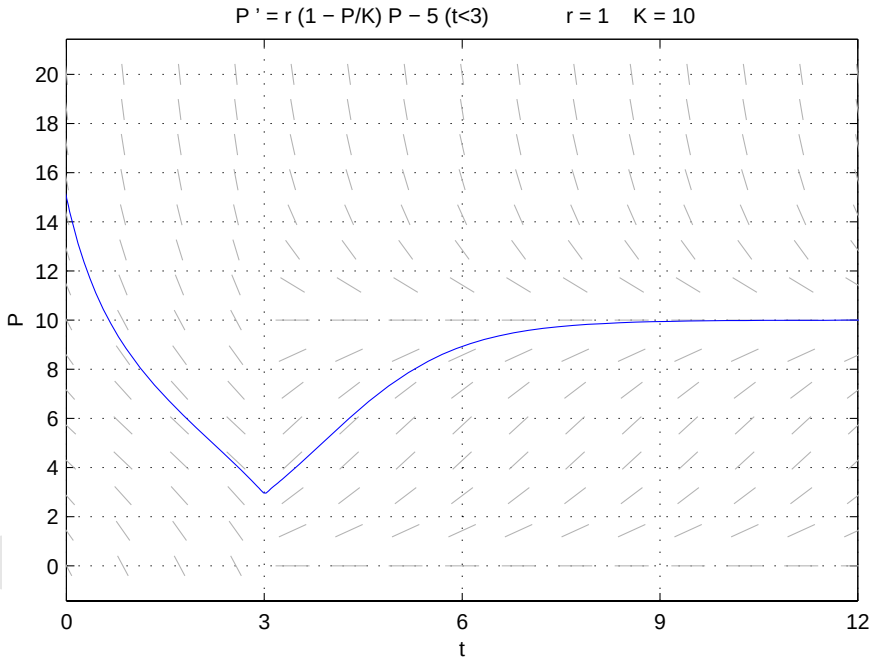
- `Dfield` understands functions if they are "array smart."
- Enter the equation in the `Dfield` Setup window as follows:

```
P' = r * (1 - P/K) * P - 5 * (t < 3)
```

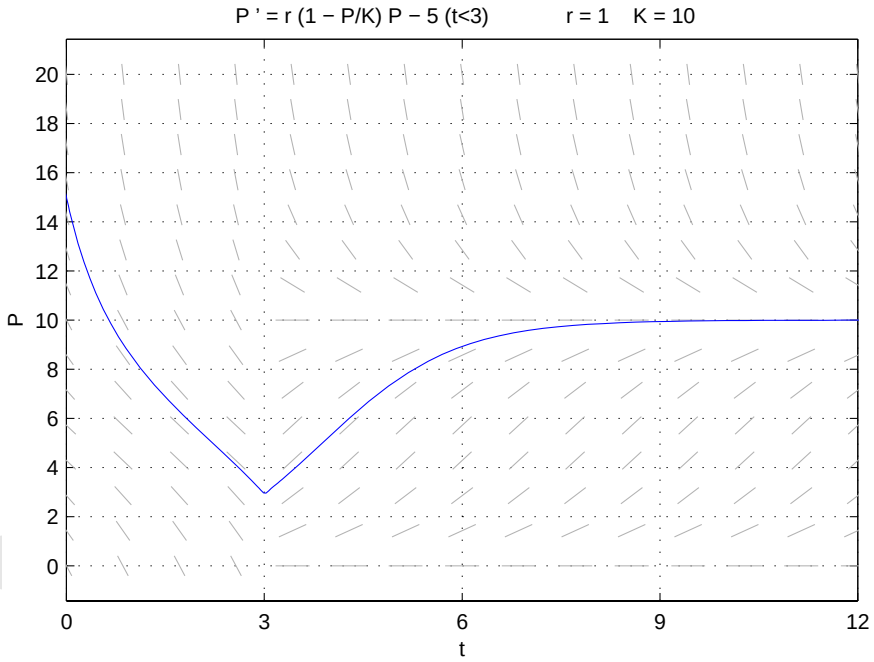
- Enter parameters

r = 1 and K = 10.

- The initial condition $P(0) = 15$ produces the following solution.



Note that the population recovers to the carrying capacity.

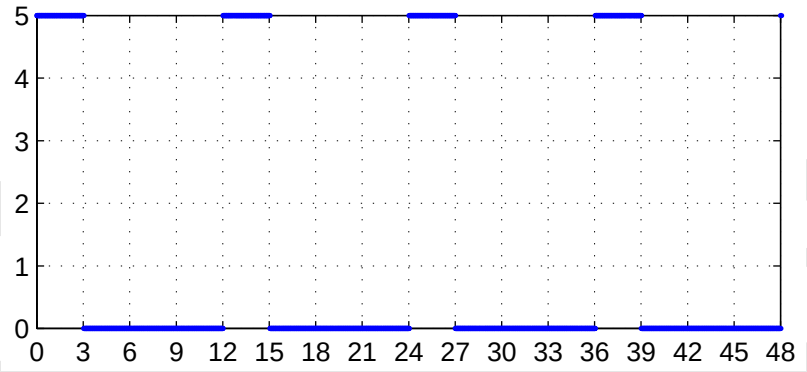


Note that the population recovers to the carrying capacity.

- What “critical” initial population will not allow the population to recover with this harvesting policy?

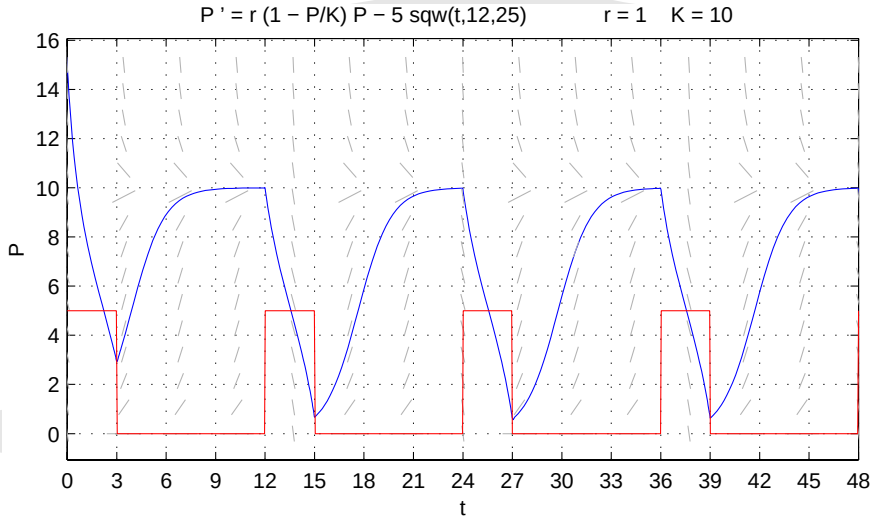
Periodic Harvesting

Suppose, instead, that we introduce a harvesting policy that captures the population at a rate of 5 tons per month, but only during the first three months of the year.



The square wave function has syntax `sqw(tspan,T,d)`, where `tspan` is the time interval, `T` is the period, and `d` is the "duty cycle," the percentage of time that the function is "on" during each period.

Adapting the Logistic equation to this harvesting policy produces the following result.



By superimposing the harvesting function in red, you can clearly discern that the population decreases during the months that the harvesting is “on,” then recovers during the months that the harvesting is “off.”

Predator-Prey (Lotka–Volterra)

- Suppose that the prey, in the absence of predators, grows according to the Malthusian model of population growth.

$$\frac{dF}{dt} = aF$$

Predator-Prey (Lotka–Volterra)

- Suppose that the prey, in the absence of predators, grows according to the Malthusian model of population growth.

$$\frac{dF}{dt} = aF$$

- In the presence of predators, one would not expect the reproductive rate of the prey to remain constant. Let's assume that interaction with the predators reduces the reproduction rate of the prey by an amount that is proportional to the predator population.

$$\frac{dF}{dt} = (a - bS)F$$

$$\frac{dF}{dt} = aF - bSF$$

- If we assume that the predator population depends solely upon the prey for its sustenance, it is not unreasonable to assume that the predators will decay according to the model

$$\frac{dS}{dt} = -cS.$$

- If we assume that the predator population depends solely upon the prey for its sustenance, it is not unreasonable to assume that the predators will decay according to the model

$$\frac{dS}{dt} = -cS.$$

- Again, in the presence of prey, one would not expect the reproductive rate of the predators to remain constant. Let us assume that interaction with the prey serves to increase the reproductive rate of the predators by an amount that is proportional to the prey population.

$$\frac{dS}{dt} = (-c + dF)S$$
$$\frac{dS}{dt} = -cS + dFS.$$

PPlane

John Polking, of Rice University, has written a numerical solver called pplane, which is ideal for solving planar, autonomous systems.

- The pplane software requires that the MATLAB software is installed.

http://www.mathworks.com

- The pplane software is available at the following url.

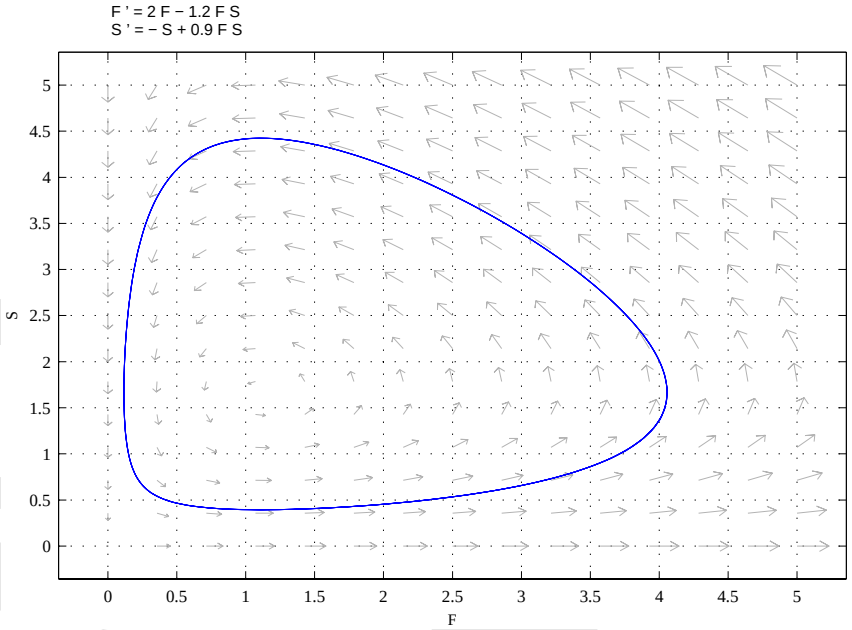
http://math.rice.edu/~dfield/

- A Java version of pplane is available.

http://math.rice.edu/~dfield/#java

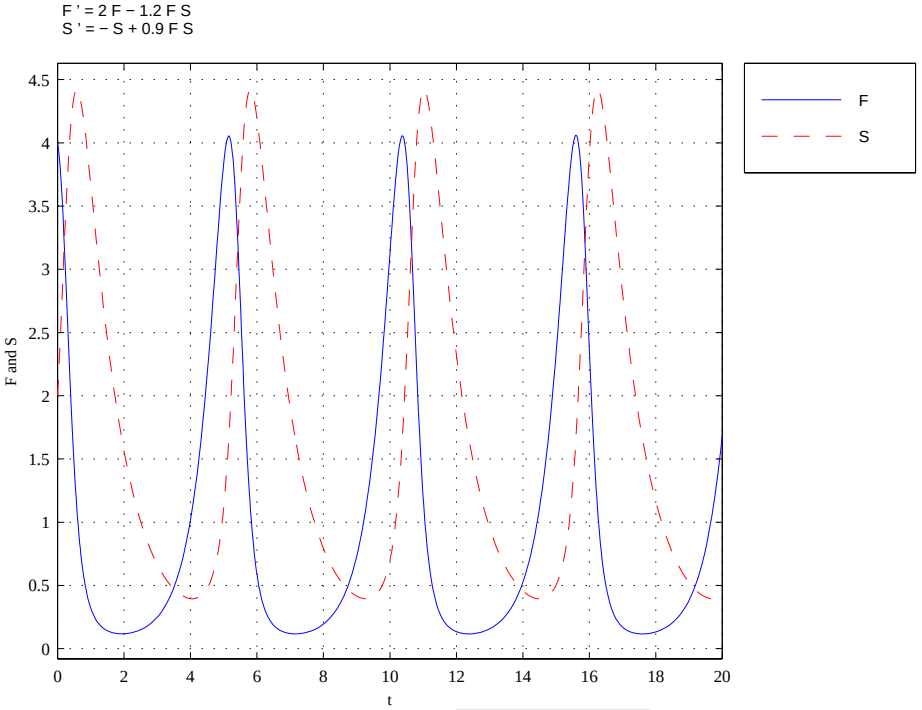
PPlane and Predator-Prey

A solution in the *phase plane*.



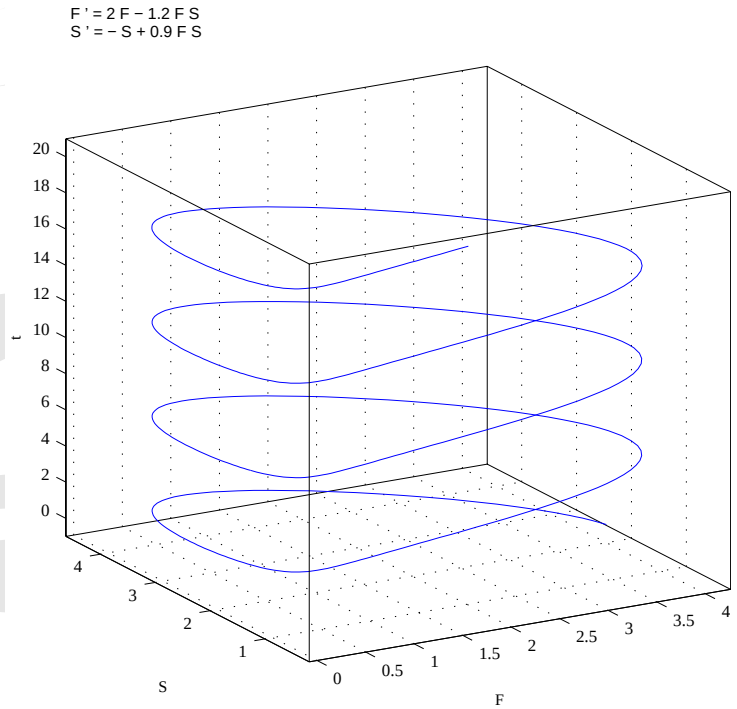
PPlane and Predator-Prey

Time plots.



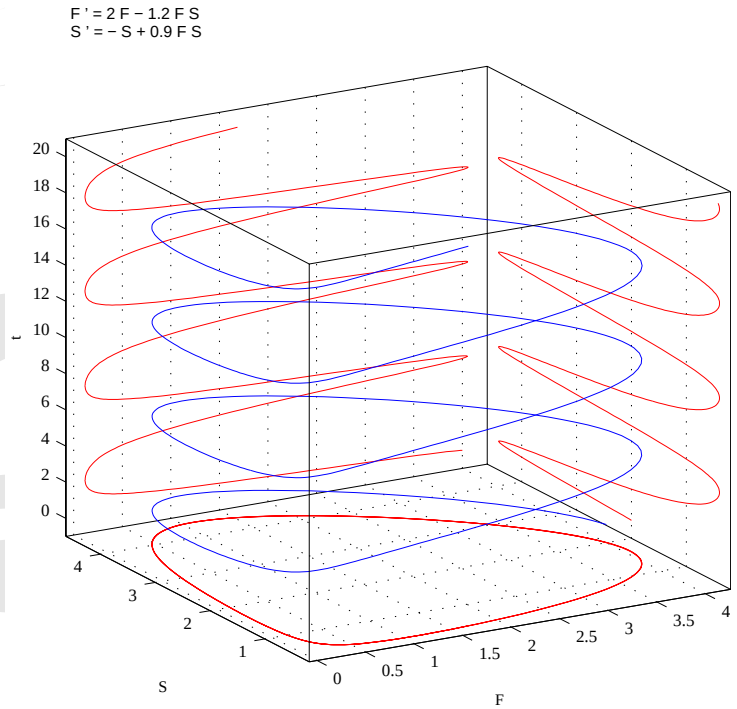
PPlane and Predator-Prey

A three dimensional plot with time on the vertical axis.



PPlane and Predator-Prey

A composite plot.



PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.







PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.
- *Nullclines* use arrows to indicate direction of motion.

PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.
- *Nullclines* use arrows to indicate direction of motion.
- You can find and classify *equilibrium points*.

PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.
- *Nullclines* use arrows to indicate direction of motion.
- You can find and classify *equilibrium points*.
- You can *linearize* your system at an equilibrium point.







PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.
- *Nullclines* use arrows to indicate direction of motion.
- You can find and classify *equilibrium points*.
- You can *linearize* your system at an equilibrium point.
- A *"zoom square"* option eases comparison of the nonlinear system with its linearization.

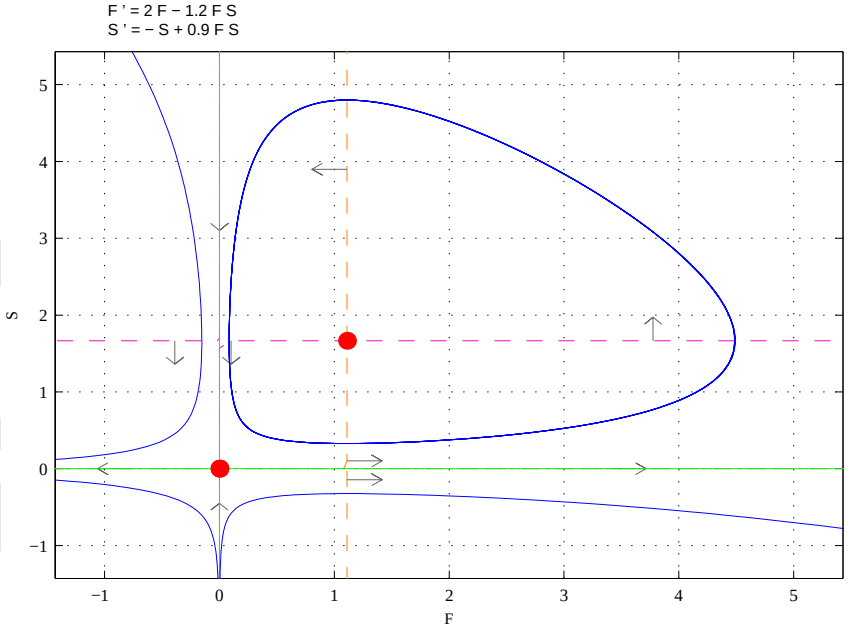
PPlane Options

PPlane has numerous options that are invaluable in an introductory differential equations class.

- Most menu items are self explanatory and easy to use.
- *Nullclines* use arrows to indicate direction of motion.
- You can find and classify *equilibrium points*.
- You can *linearize* your system at an equilibrium point.
- A *"zoom square"* option eases comparison of the nonlinear system with its linearization.
- Saddle points have stable and unstable orbits called *"separatrices."*

Predator-Prey With Options

Here is the predator-prey model with nullclines, equilibrium points, and separatrices.



Teddy Bears

Mathematicians just wanna have fun! So, here are a few "Teddy Bears" to brighten your day!

$$x' = \sin y - 2 \sin x^2 \sin 2y$$
$$y' = -\cos x - 2x \cos x^2 \cos 2y$$

Teddy Bears

Mathematicians just wanna have fun! So, here are a few "Teddy Bears" to brighten your day!

$$x' = \sin y - 2\sin x^2 \sin 2y$$
$$y' = -\cos x - 2x\cos x^2 \cos 2y$$

- The initial condition $(0, \pi/2)$ creates the "legs" of the teddy bears.

Teddy Bears

Mathematicians just wanna have fun! So, here are a few "Teddy Bears" to brighten your day!

$$x' = \sin y - 2\sin x^2 \sin 2y$$
$$y' = -\cos x - 2x\cos x^2 \cos 2y$$

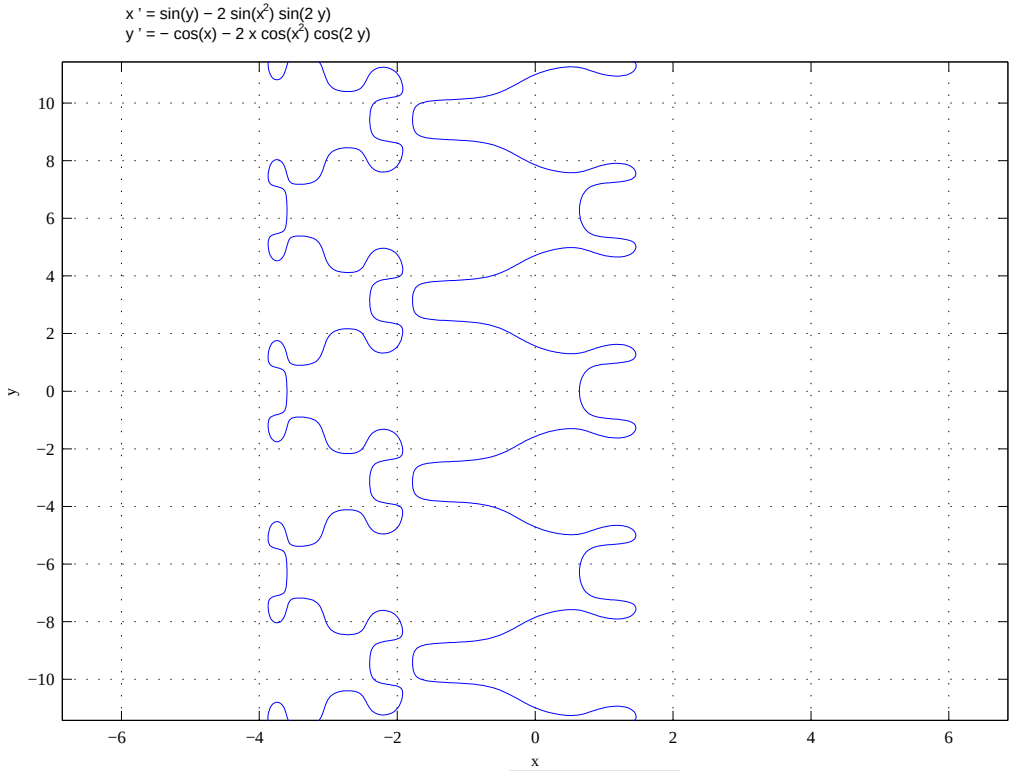
- The initial condition $(0, \pi/2)$ creates the "legs" of the teddy bears.
- The initial condition $(-3.5, -7.2)$ creates the "heads" of the teddy bears.

Teddy Bears

Mathematicians just wanna have fun! So, here are a few "Teddy Bears" to brighten your day!

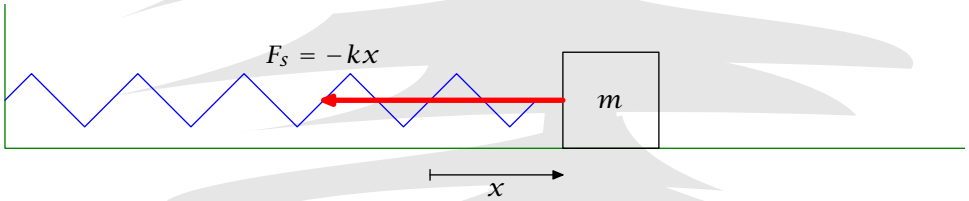
$$x' = \sin y - 2\sin x^2 \sin 2y$$
$$y' = -\cos x - 2x\cos x^2 \cos 2y$$

- The initial condition $(0, \pi/2)$ creates the "legs" of the teddy bears.
- The initial condition $(-3.5, -7.2)$ creates the "heads" of the teddy bears.
- Experiment to find the "eyes," "mouths," and "bellybuttons" of the teddy bears.


x ' = sin(y) − 2 sin(x^2) sin(2 y)
y ' = − cos(x) − 2 x cos(x^2) cos(2 y)
10
8
6
4
2
0
−2
−4
−6
−8
−10
y
−6
−4
−2
0
2
4
6
x

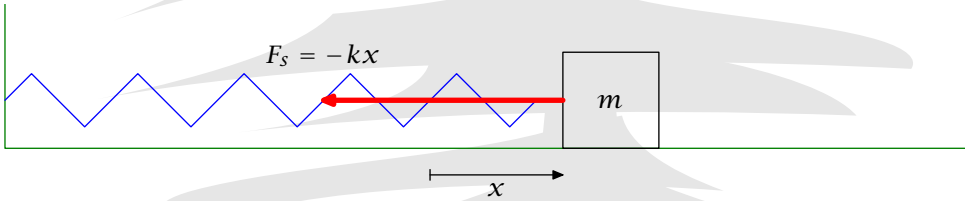
The Harmonic Oscillator

Consider the spring-mass system.



The Harmonic Oscillator

Consider the spring-mass system.

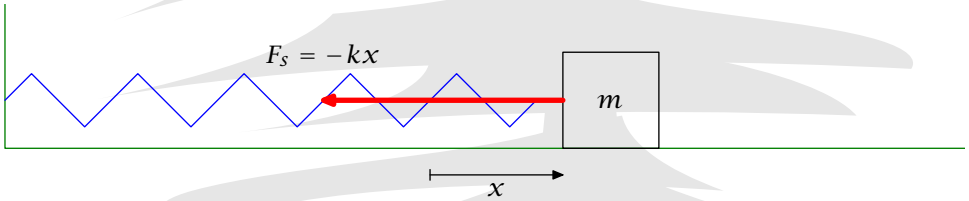


- Assume that the spring provides a restoring force *opposite the displacement* according to Hooke's Law,

$$F_s = -kx.$$

The Harmonic Oscillator

Consider the spring-mass system.



- Assume that the spring provides a restoring force *opposite the displacement* according to Hooke's Law,

$$F_s = -kx.$$

- Assume a resistive force that is *opposite the motion*, as in

$$F_r = -\mu v.$$

Newton's Second Law

According to Sir Isaac Newton, the sum of the forces acting on an object equals its mass times it acceleration. Thus,

$$ma = \sum F$$
$$ma = F_r + F_s$$
$$ma = -\mu v - kx.$$

Newton's Second Law

According to Sir Isaac Newton, the sum of the forces acting on an object equals its mass times it acceleration. Thus,

$$ma = \sum F$$
$$ma = F_r + F_s$$
$$ma = -\mu v - kx.$$

However,

$$v = x',$$
$$a = x'',$$

Newton's Second Law

According to Sir Isaac Newton, the sum of the forces acting on an object equals its mass times it acceleration. Thus,

$$ma = \sum F$$
$$ma = F_r + F_s$$
$$ma = -\mu v - kx.$$

However,

$$v = x',$$
$$a = x'',$$

so

$$mx'' = -\mu x' - kx$$
$$mx'' + \mu x' + kx = 0.$$

An Example

Suppose that we have the following constants.

$$m = 1\,\text{kg}$$
$$\mu = 0.2\,\text{kg/s}$$
$$k = 4\,\text{N/m}.$$

An Example

Suppose that we have the following constants.

$$m = 1\,\text{kg}$$
$$\mu = 0.2\,\text{kg/s}$$
$$k = 4\,\text{N/m}.$$

Thus, this particular harmonic oscillator becomes

$$mx'' + \mu x' + kx = 0$$
$$x'' + 0.2x' + 4x = 0.$$

An Example

Suppose that we have the following constants.

$$m = 1\,\text{kg}$$
$$\mu = 0.2\,\text{kg/s}$$
$$k = 4\,\text{N/m}.$$

Thus, this particular harmonic oscillator becomes

$$mx'' + \mu x' + kx = 0$$
$$x'' + 0.2x' + 4x = 0.$$

Suppose that the mass is initially displaced $1\,\text{m}$ and released from rest. Then we have the following initial conditions:

$$x(0) = 1\,\text{m}$$
$$x'(0) = 0\,\text{m/s}.$$

The Equivalent System

If we solve for the highest derivative of x,

$$x'' = -0.2x' - 4x,$$

The Equivalent System

If we solve for the highest derivative of x,

$$x'' = -0.2x' - 4x,$$

and let $v = x'$, then the *second order* equation is transformed into a *system* of first order equations,

$$x' = v,$$
$$v' = -0.2v - 4x,$$

The Equivalent System

If we solve for the highest derivative of x,

$$x'' = -0.2x' - 4x,$$

and let $v = x'$, then the *second order* equation is transformed into a *system* of first order equations,

$$x' = v,$$
$$v' = -0.2v - 4x,$$

with initial conditions

$$x(0) = 1,$$
$$v(0) = 0.$$

The Equivalent System

If we solve for the highest derivative of x,

$$x'' = -0.2x' - 4x,$$

and let $v = x'$, then the *second order* equation is transformed into a *system* of first order equations,

$$x' = v,$$
$$v' = -0.2v - 4x,$$

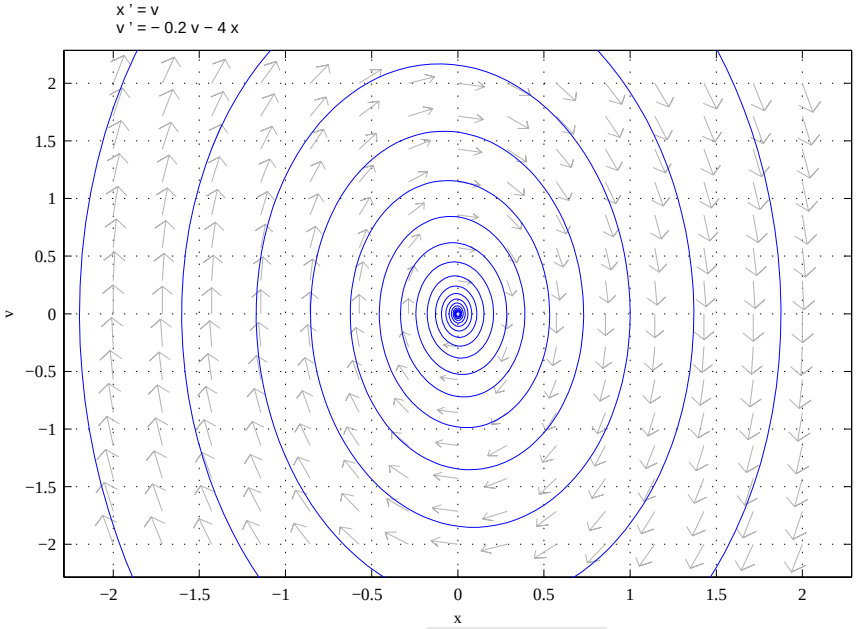
with initial conditions

$$x(0) = 1,$$
$$v(0) = 0.$$

This *planar, autonomous* system can be analyzed with *PPlane*.

PPlane Analysis

As you would expect, solutions spiral inward toward an asymptotically stable equilibrium point, commonly called a *spiral sink*.



Vector Notation

Consider the system of n first order equations in n unknowns.

$$
\begin{aligned}
x_1' &= f_1(t, x_1, x_2, \ldots, x_n) \\
x_2' &= f_2(t, x_1, x_2, \ldots, x_n) \\
&\vdots \\
x_n' &= f_n(t, x_1, x_2, \ldots, x_n)
\end{aligned}
$$

Vector Notation

Consider the system of n first order equations in n unknowns.

$$\begin{aligned} x_1' &= f_1(t, x_1, x_2, \ldots, x_n) \\ x_2' &= f_2(t, x_1, x_2, \ldots, x_n) \\ &\vdots \\ x_n' &= f_n(t, x_1, x_2, \ldots, x_n) \end{aligned}$$

In vector notation,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}' = \begin{bmatrix} f_1(t, x_1, x_2, \ldots, x_n) \\ f_2(t, x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(t, x_1, x_2, \ldots, x_n) \end{bmatrix}.$$

If we let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

If we let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

then

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}' = \begin{bmatrix} f_1(t, x_1, x_2, \ldots, x_n) \\ f_2(t, x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(t, x_1, x_2, \ldots, x_n) \end{bmatrix}.$$







If we let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

then

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}' = \begin{bmatrix} f_1(t, x_1, x_2, \ldots, x_n) \\ f_2(t, x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(t, x_1, x_2, \ldots, x_n) \end{bmatrix}.$$

becomes

$$\mathbf{x}' = \begin{bmatrix} f_1(t, \mathbf{x}) \\ f_2(t, \mathbf{x}) \\ \vdots \\ f_n(t, \mathbf{x}) \end{bmatrix}.$$

Finally, if we let

$$\mathbf{F}(t,\mathbf{x}) = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix},$$

Finally, if we let

$$\mathbf{F}(t,\mathbf{x}) = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix},$$

then

$$\mathbf{x}' = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix}.$$

Finally, if we let

$$\mathbf{F}(t,\mathbf{x}) = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix},$$

then

$$\mathbf{x}' = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix}.$$

becomes

$$\mathbf{x}' = \mathbf{F}(t,\mathbf{x}).$$

Finally, if we let

$$\mathbf{F}(t,\mathbf{x}) = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix},$$

then

$$\mathbf{x}' = \begin{bmatrix} f_1(t,\mathbf{x}) \\ f_2(t,\mathbf{x}) \\ \vdots \\ f_n(t,\mathbf{x}) \end{bmatrix}.$$

becomes

$$\mathbf{x}' = \mathbf{F}(t,\mathbf{x}).$$

Note the similarity to the *normal form* employed with ordinary first order equations. This form is especially important when writing *function M-files* to pass to MATLAB's suite of ordinary differential equation solvers.

Duffing's Oscillator

"In 1918, Duffing introduced a nonlinear oscillator with a cubic stiffness term to describe the hardening spring effect observed in many mechanical systems." (Guckenheimer and Holmes)

$$x'' + \delta x' - x + x^3 = \gamma \cos \omega t$$







Duffing's Oscillator

"In 1918, Duffing introduced a nonlinear oscillator with a cubic stiffness term to describe the hardening spring effect observed in many mechanical systems." (Guckenheimer and Holmes)

$$x'' + \delta x' - x + x^3 = \gamma \cos \omega t$$

1. Solve the equation for highest derivative of x.

$$x'' = -\delta x' + x - x^3 + \gamma \cos \omega t$$







Duffing's Oscillator

"In 1918, Duffing introduced a nonlinear oscillator with a cubic stiffness term to describe the hardening spring effect observed in many mechanical systems." (Guckenheimer and Holmes)

$$x'' + \delta x' - x + x^3 = \gamma \cos \omega t$$

1. Solve the equation for highest derivative of x.

$$x'' = -\delta x' + x - x^3 + \gamma \cos \omega t$$

2. Introduce new variables representing x and x'.

$$x_1 = x$$
$$x_2 = x'$$

Duffing's Oscillator

"In 1918, Duffing introduced a nonlinear oscillator with a cubic stiffness term to describe the hardening spring effect observed in many mechanical systems." (Guckenheimer and Holmes)

$$x'' + \delta x' - x + x^3 = \gamma \cos \omega t$$

1. Solve the equation for highest derivative of x.

$$x'' = -\delta x' + x - x^3 + \gamma \cos \omega t$$

2. Introduce new variables representing x and x'.

$$x_1 = x$$
$$x_2 = x'$$

3. Differentiate.

$$x_1' = x'$$
$$x_2' = x'' = -\delta x' + x - x^3 + \gamma \cos \omega t$$

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

5. In vector form, this system is written

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t \end{bmatrix},$$

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

5. In vector form, this system is written

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t \end{bmatrix},$$

which has the form

$$\mathbf{x}' = \mathbf{F}(t, \mathbf{x}).$$

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

5. In vector form, this system is written

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t \end{bmatrix},$$

which has the form

$$\mathbf{x}' = \mathbf{F}(t, \mathbf{x}).$$

Before we can employ MATLAB's solvers, we must first write a function M-file for the right-hand side of this equation.

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

5. In vector form, this system is written

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t \end{bmatrix},$$

which has the form

$$\mathbf{x}' = \mathbf{F}(t, \mathbf{x}).$$

Before we can employ MATLAB's solvers, we must first write a function M-file for the right-hand side of this equation.

- Note that the inputs to $\mathbf{F}$ are t and $\mathbf{x} = [x_1, x_2]^T$, a *scalar* and a *vector* from $\mathbb{R}^2$.

4. Substitute $x_1 = x$ and $x_2 = x'$.

$$x_1' = x_2,$$
$$x_2' = -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t.$$

5. In vector form, this system is written

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -\delta x_2 + x_1 - x_1^3 + \gamma \cos \omega t \end{bmatrix},$$

which has the form

$$\mathbf{x}' = \mathbf{F}(t, \mathbf{x}).$$

Before we can employ MATLAB's solvers, we must first write a function M-file for the right-hand side of this equation.

- Note that the inputs to $\mathbf{F}$ are t and $\mathbf{x} = [x_1, x_2]^T$, a *scalar* and a *vector* from $\mathbb{R}^2$.
- The output of the function $\mathbf{F}$ is $\mathbf{x}' = [x_1', x_2']^T$, also a *vector* from $\mathbb{R}^2$.

The Function M-file

With $\delta = 0.15$, $\gamma = 0.3$ and $\omega = 1$,

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{F}(t, \mathbf{x}) = \begin{bmatrix} x_2 \\ -0.15x_2 + x_1 - x_1^3 + 0.3\cos t \end{bmatrix}.$$

The Function M-file

With $\delta = 0.15$, $\gamma = 0.3$ and $\omega = 1$,

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{F}(t, \mathbf{x}) = \begin{bmatrix} x_2 \\ -0.15x_2 + x_1 - x_1^3 + 0.3\cos t \end{bmatrix}.$$

Thus, it is natural to write the following function M-file.

```
function xprime=F(t,x)
xprime=zeros(2,1);
xprime(1)=x(2);
xprime(2)=-0.15*x(2)+x(1)-x(1)^3+0.3*cos(t);
```

Save the file as `F.m`.

Calling MATLAB's Solvers

The general calling syntax follows:

```
ode45(Fcn,tspan,init,options,p1,p2,...)
```

Fcn	The name of the function M-file
tspan	The solution interval
init	The initial conditions
options	Options passed to the solver
p1,p2,...	Optional parameters

Calling MATLAB's Solvers

The general calling syntax follows:

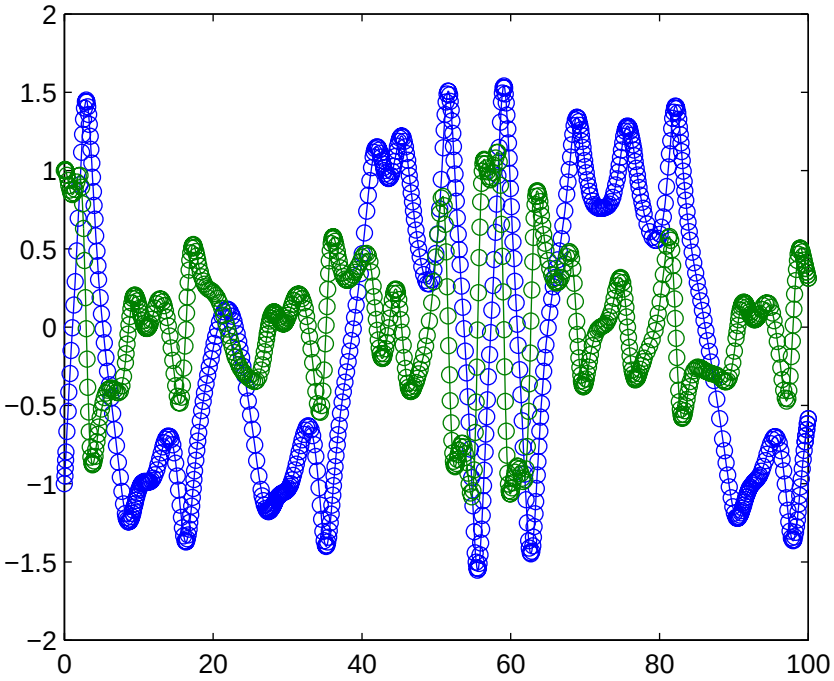
```
ode45(Fcn,tspan,init,options,p1,p2,...)
```

Fcn	The name of the function M-file
tspan	The solution interval
init	The initial conditions
options	Options passed to the solver
p1,p2,...	Optional parameters

For example, the command

```
ode45('F',[0,100],[-1;1])
```

will produce the image shown on the next slide.


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
0
20
40
60
80
100

Setting the Relative Error Tolerance

The *relative error tolerance* determines (approximately) the number of significant digits used in the computation of the solution.

Setting the Relative Error Tolerance

The *relative error tolerance* determines (approximately) the number of significant digits used in the computation of the solution.

- It is good practice to make several runs with the solver, decreasing the relative tolerance until you get two consecutive solutions that match.







Setting the Relative Error Tolerance

The *relative error tolerance* determines (approximately) the number of significant digits used in the computation of the solution.

- It is good practice to make several runs with the solver, decreasing the relative tolerance until you get two consecutive solutions that match.
- MATLAB's `odeset` command is used to set options to send to the solver.

```
options=odeset('RelTol',1e-5')
```

Setting the Relative Error Tolerance

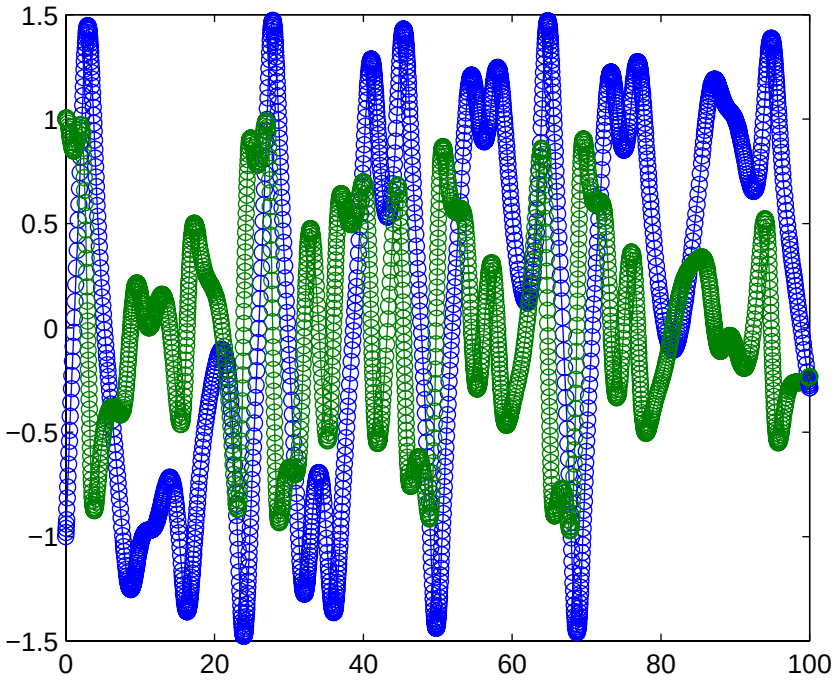
The *relative error tolerance* determines (approximately) the number of significant digits used in the computation of the solution.

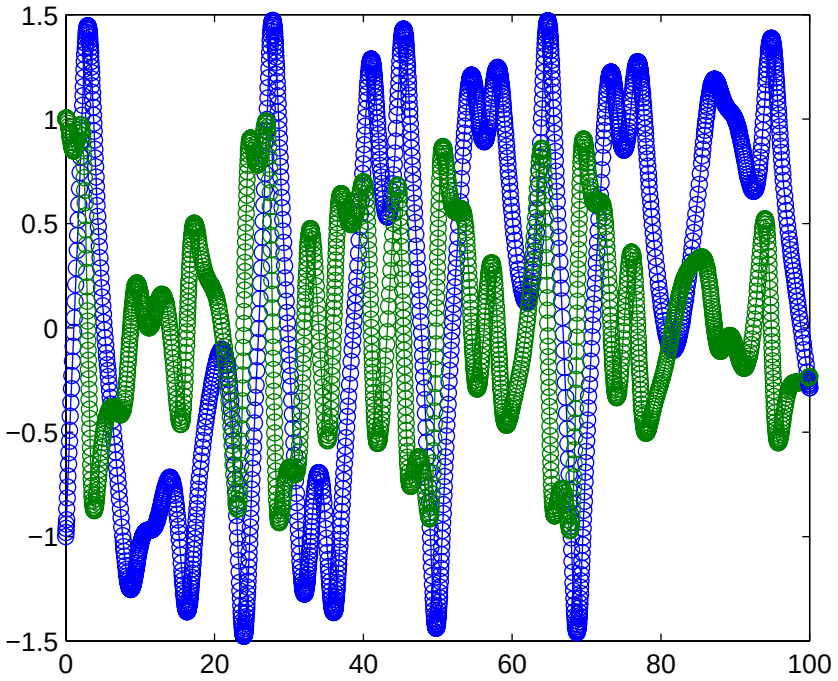
- It is good practice to make several runs with the solver, decreasing the relative tolerance until you get two consecutive solutions that match.
- MATLAB's `odeset` command is used to set options to send to the solver.

```
options=odeset('RelTol',1e-5')
```

- Another run produces the result on the next slide.

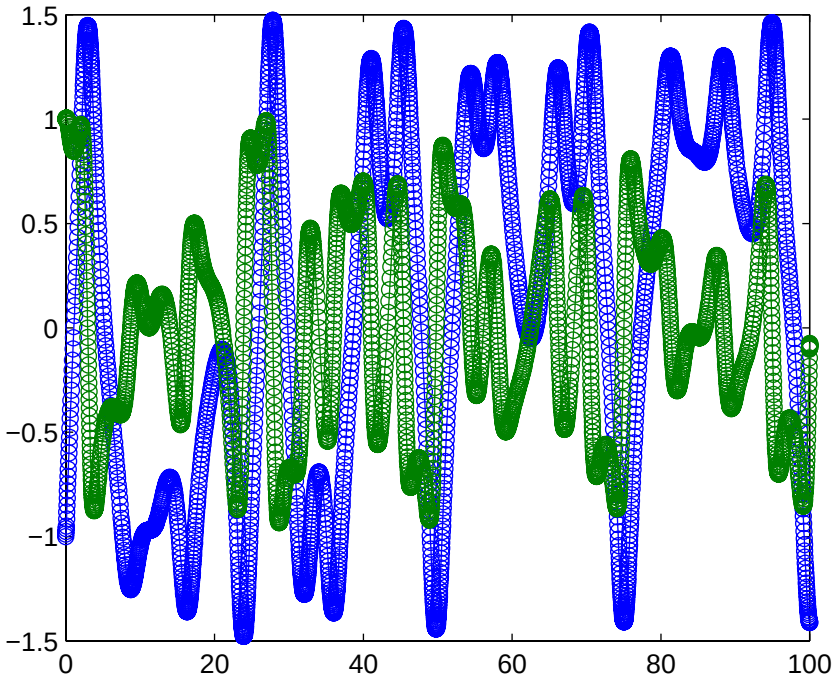
```
ode45('F',[0,100],[-1;1],options)
```

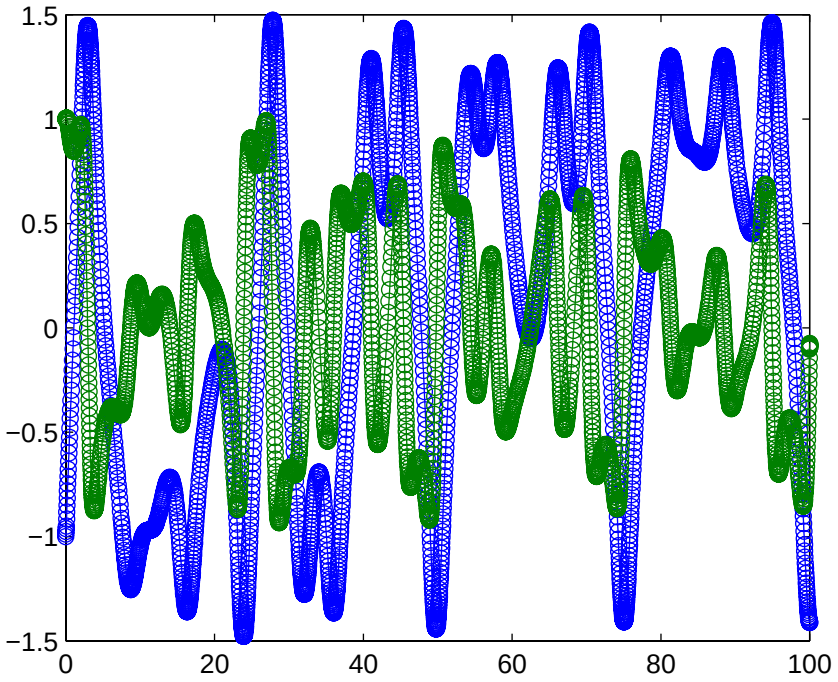

1.5
1
0.5
0
−0.5
−1
−1.5
0
20
40
60
80
100



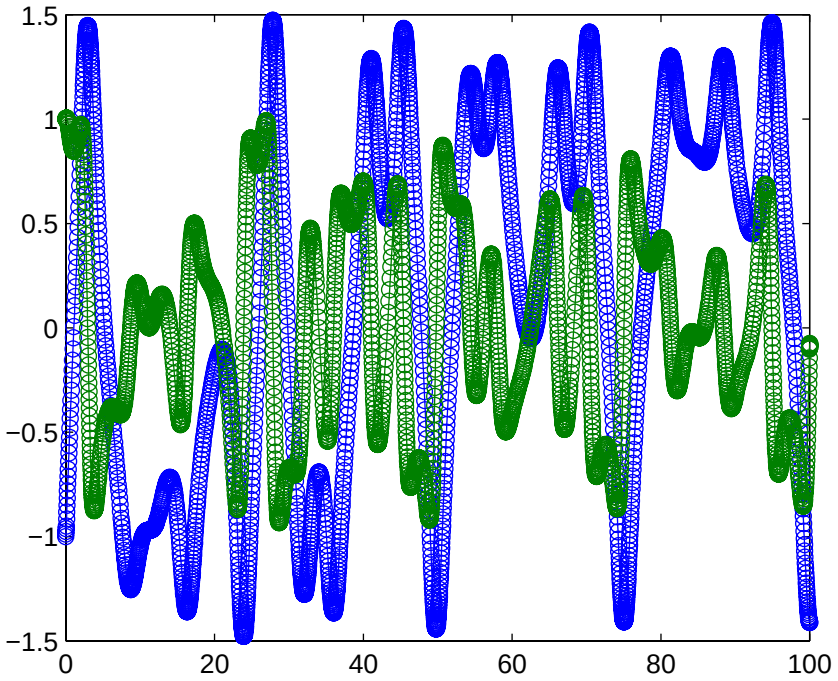
- Lower the tolerance and make another run.

```
>> options=odeset('RelTol',1e-6')
>> ode45('Fcn',[0,100],[-1;1],options)
```


1.5
1
0.5
0
−0.5
−1
−1.5
0
20
40
60
80
100



- Note that the solution continues to change with reduced tolerances.



- Note that the solution continues to change with reduced tolerances.
- Continue reducing the tolerance until you get two consecutive solutions that match.

Changing the Output Function

The previous plots provided time plots of x_1 and x_2 for Duffing's oscillator. Phase plane plots are also possible.

Changing the Output Function

The previous plots provided time plots of x_1 and x_2 for Duffing's oscillator. Phase plane plots are also possible.

- Use odeset to change the output routine used by MATLAB.

```
options=odeset('RelTol',1e-6,...
                    'OutputFcn','odephas2')
```







Changing the Output Function

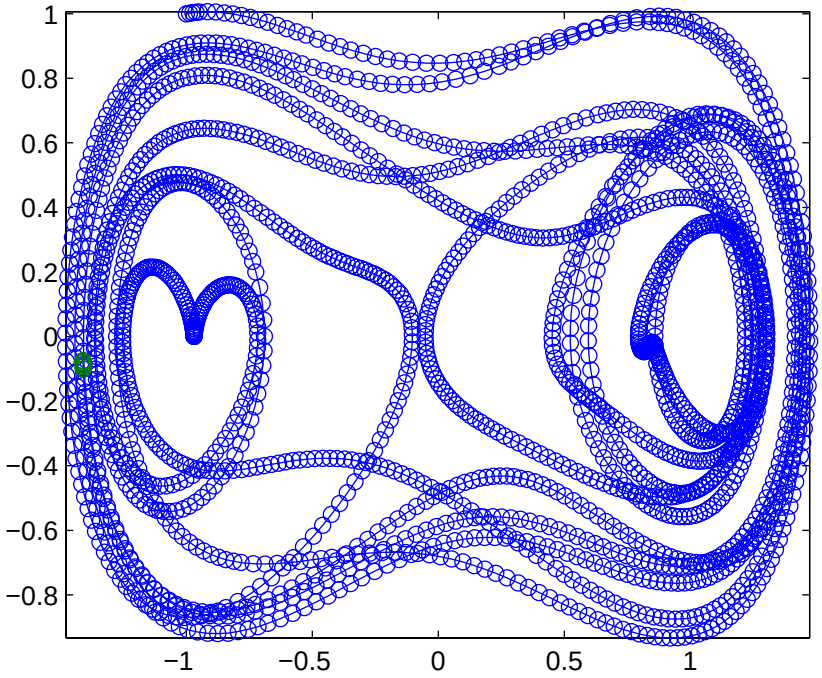
The previous plots provided time plots of x_1 and x_2 for Duffing's oscillator. Phase plane plots are also possible.

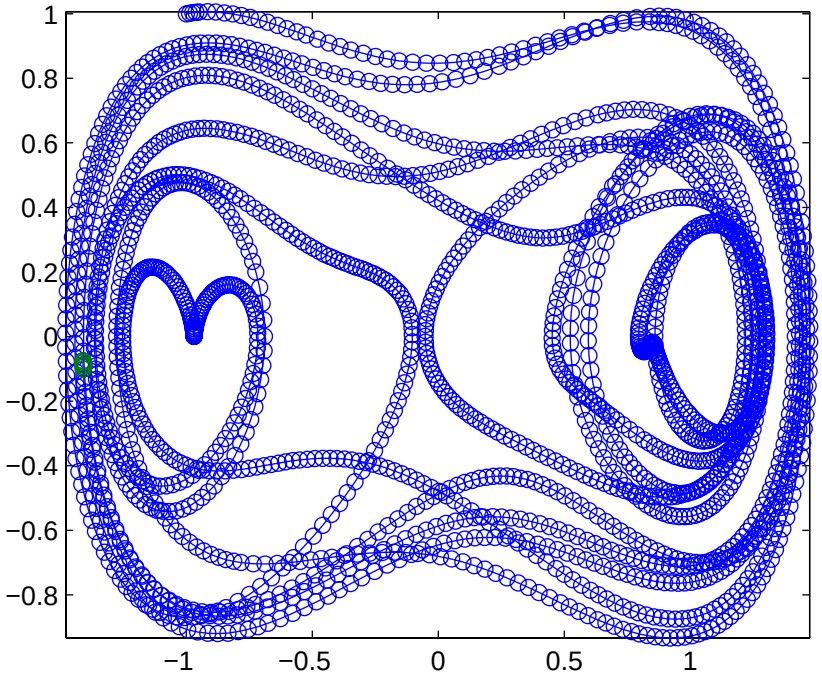
- Use odeset to change the output routine used by MATLAB.

```
options=odeset('RelTol',1e-6,...
                    'OutputFcn','odephas2')
```

- A call to ode45 will now plot the output in the phase plane, as shown on the next slide.

```
ode45('F',[0,100],[-1;1],options)
```


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−0.5
0
0.5
1



- Solutions may cross in the phase plane when the vector field

$$\mathbf{F}(t,\mathbf{x}) = \begin{bmatrix} x_2 \\ -0.15x_2 + x_1 - x_1^3 + 0.3\cos t \end{bmatrix}$$

varies with time t.

Using Parameters

You may wish to examine how a equation or system behaves as you vary one or more parameters.

Using Parameters

You may wish to examine how a equation or system behaves as you vary one or more parameters.

- First restructure the function M-file by appending the parameters as input variables.

```
function xprime=F(t,x,flag,delta,gamma,omega)
xprime=zeros(2,1);
xprime(1)=x(2);
xprime(2)=delta*x(2)+x(1)-x(1)^3+...
          gamma*cos(omega*t);
```

- Note the special input `flag` which is required and used by the solver. After this input, you can follow with your list of parameters.

Passing Parameters

- If you do not have options to pass to the solver, use an empty array for the options, this list the parameter values in the same order used in the function M-file.

```
ode45('F',[0,100],[-1;1],[],-0.15,0.3,1)
```

Passing Parameters

- If you do not have options to pass to the solver, use an empty array for the options, this list the parameter values in the same order used in the function M-file.

```
ode45('F',[0,100],[-1;1],[],-0.15,0.3,1)
```

- The numerical solution obtained by the solver can be stored in output variables for later use.

```
options=odeset('RelTol',1e-6)
[t,x]=ode45('F',[0,100],[-1;1],...
            options,-0.15,0.3,1)
```


Interpreting the Output

- Each row in the solution array `x` corresponds to a time returned in column vector `t`.

```
 [t,x]
ans =
         0   -1.0000    1.0000
    0.0126   -0.9874    1.0017
    0.0252   -0.9747    1.0032
    0.0379   -0.9621    1.0043
    0.0505   -0.9494    1.0051
    0.0837   -0.9160    1.0060
    0.1169   -0.8826    1.0052
    0.1501   -0.8492    1.0029
    0.1833   -0.8159    0.9993
    ...      ...        ...
```



- Note that the first column of x contains the solution for x_1.

```
x(:,1)
ans =
   -1.0000
   -0.9874
   ...
```

- The second column of x contains the solution for x_2.

```
x(:,2)
ans =
    1.0000
    1.0017
    ...
```

- The command `plot(t,x)` will create a plot of the columns of `x` versus `t`.

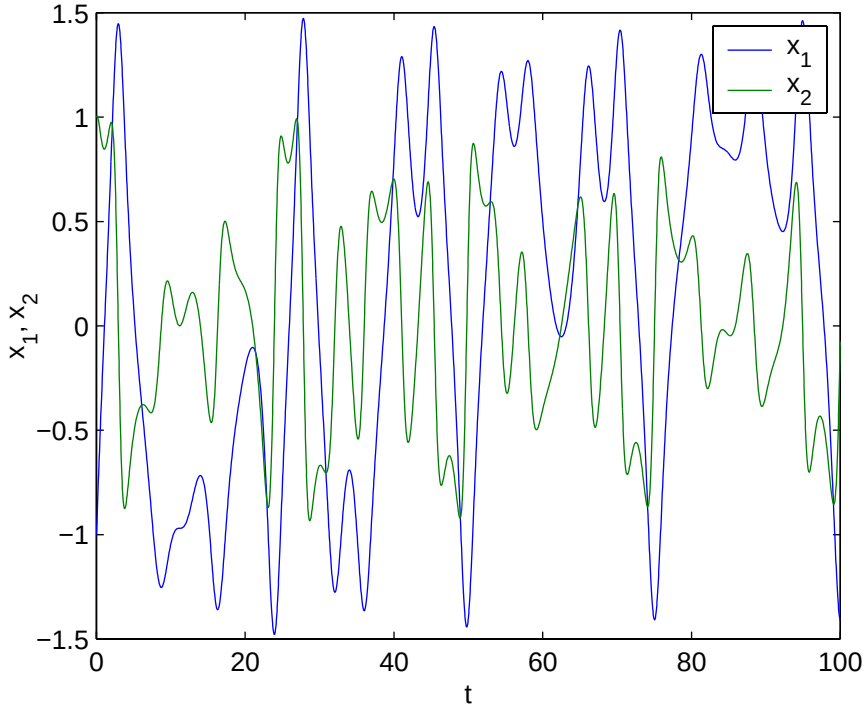
```
plot(t,x)
legend('x_1','x_2')
xlabel('t')
ylabel('x_1,x_2')
```

- The command `plot(t,x)` will create a plot of the columns of `x` versus `t`.

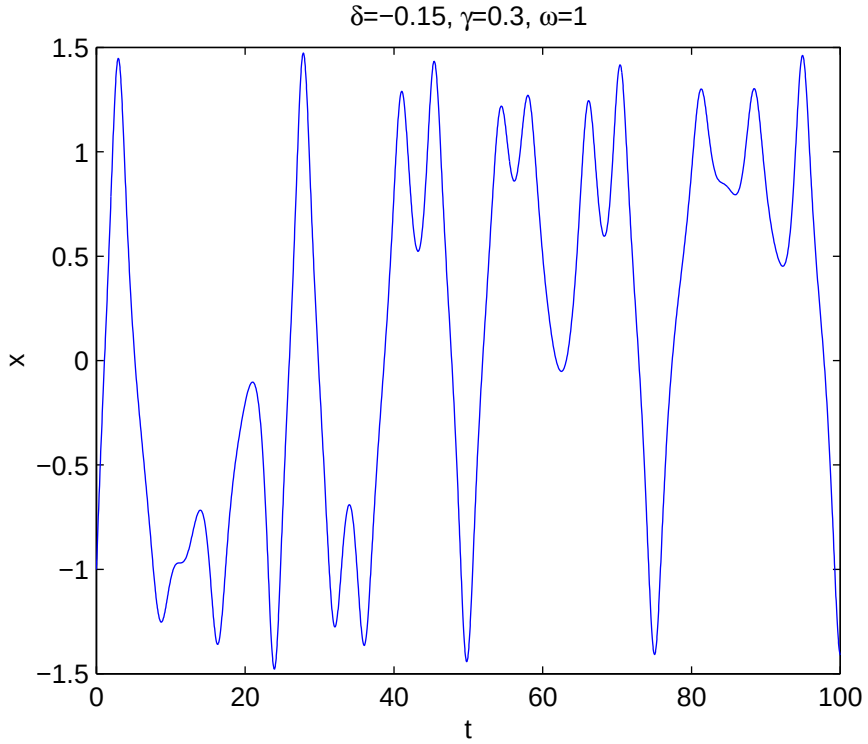
```
plot(t,x)
legend('x_1','x_2')
xlabel('t')
ylabel('x_1,x_2')
```

- Recall that $x_1 = x$, so the first column of `x` contains the solution to Duffing's Oscillator.

```
plot(t,x(:,1))
xlabel('t')
ylabel('x')
title('\delta=-0.15,\gamma=0.3,\omega=1')
```



- Note the axes labels and legend, created by the `xlabel`, `ylabel`, and `legend` commands.



- Note the title. MATLAB understands a small subset of TEX commands.

Higher Dimensions

Because the basic data structure in MATLAB ("MATrix LABoratory") is the matrix, higher dimensions are handled in exactly the same manner as two dimensional systems.

$$
\begin{aligned}
u_1' &= -au_1 + au_2 \\
u_2' &= ru_1 - u_2 - u_1u_3 \\
u_3' &= -bu_3 + u_1u_2
\end{aligned}
$$

This system of three equations was published in 1963 by the meteorologist and mathematician *E. N. Lorenz*. It represents a simplified model for atmospheric turbulence beneath a thunderhead.

Global Variables

An alternative method for handling parameters is to declare them as *global* variables and initialize them in command workspace.


```
global A B R
A=10; B=8/3; R=28;
```

Global Variables

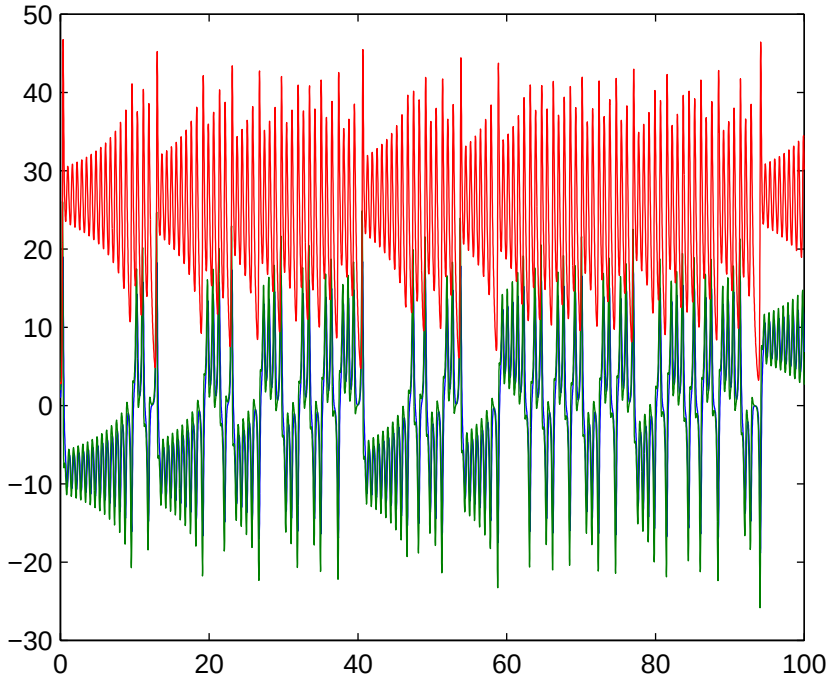
An alternative method for handling parameters is to declare them as *global* variables and initialize them in command workspace.

```
global A B R
A=10; B=8/3; R=28;
```

They must also be declared global in the function M-file.

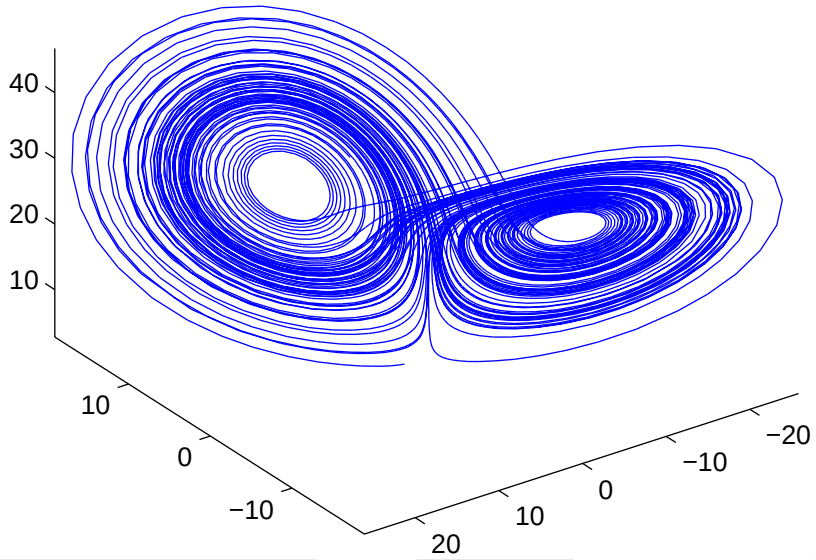
```
function uprime=lor(t,u)
global A B R
uprime=zeros(3,1);
uprime(1)=-A*u(1)+A*u(2);
uprime(2)=R*u(1)-u(2)-u(1)*u(3);
uprime(3)=-B*u(3)+u(1)*u(2);
```

Time Plots



- Produced with `[t,u]=ode45('lor',[0,100],[1;2;3])` and `plot(t,u)`.

Strange Attractor



- Produced with `plot3(u(:,1),u(:,2),u(:,3))`.

The van der Pol Oscillator

This equation,

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0,$$

was first introduced by van der Pol to model nonlinear nonlinear circuits that appeared in the first radios.







The van der Pol Oscillator

This equation,

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0,$$

was first introduced by van der Pol to model nonlinear nonlinear circuits that appeared in the first radios.

- The *nonlinear damping* term, $\mu(x^2 - 1)$ serves to dissipate energy if $|x| > 1$, but actually pumps energy *into* the circuit if $|x| < 1$.

The van der Pol Oscillator

This equation,

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0,$$

was first introduced by van der Pol to model nonlinear nonlinear circuits that appeared in the first radios.

- The *nonlinear damping* term, $\mu(x^2 - 1)$ serves to dissipate energy if $|x| > 1$, but actually pumps energy *into* the circuit if $|x| < 1$.
- As the parameter μ is increased, the equation becomes increasingly *stiff*, necessitating the application of a stiff solver to assist in the computation of the numerical solution.

- Letting $y = \dot{x}$, the initial value problem

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0, \quad x(0) = \dot{x}(0) = 1$$

is easily written as a system of two first order equations,

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

with initial conditions $x(0) = 1$, $y(0) = 1$.

- Letting $y = \dot{x}$, the initial value problem

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0, \quad x(0) = \dot{x}(0) = 1$$

is easily written as a system of two first order equations,

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

with initial conditions $x(0) = 1$, $y(0) = 1$.

- The function M-file is easily written.

```
function xprime=vdpol(t,x,flag,mu)
xprime=zeros(2,1);
xprime(1)=x(2);
xprime(2)=-x(1)+mu*(1-x(1)^2)*x(2);
```

- Let $\mu = 2$ and sketch the solution in the phase plane.

```
mu=2;
options=odeset('OutputFcn','odephas2')
ode45('vdpol',[0,9],[1;1],options,mu)
```

- Let $\mu = 2$ and sketch the solution in the phase plane.

```
mu=2;
options=odeset('OutputFcn','odephas2')
ode45('vdpol',[0,9],[1;1],options,mu)
```

- Observe the behavior for the following values of μ on the indicated solution interval.

  1. $\mu = 10$ on $[0, 20]$
  2. $\mu = 20$ on $[0, 35]$
  3. $\mu = 100$ on $[0, 250]$

- Let $\mu = 2$ and sketch the solution in the phase plane.

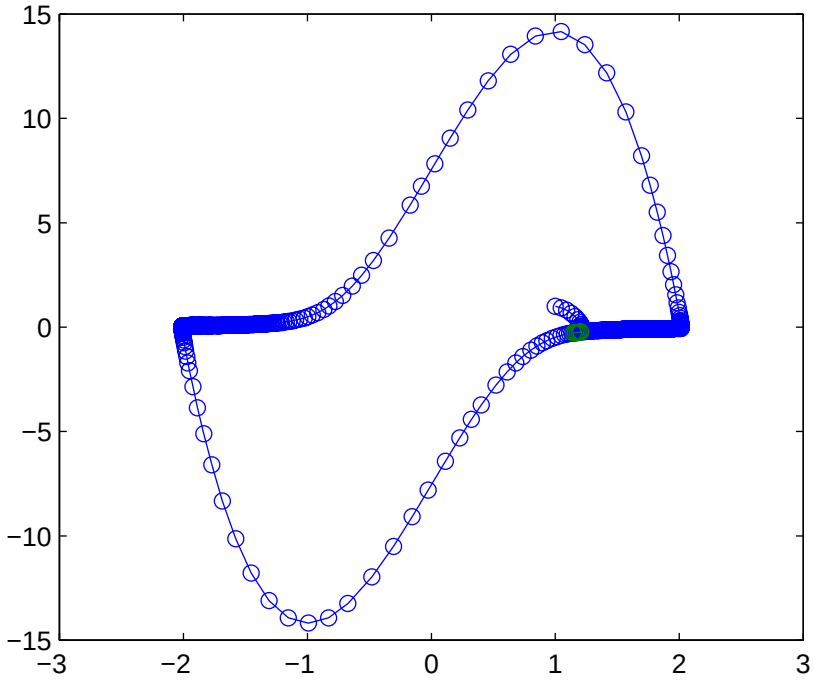
```
mu=2;
options=odeset('OutputFcn','odephas2')
ode45('vdpol',[0,9],[1;1],options,mu)
```

- Observe the behavior for the following values of μ on the indicated solution interval.

  1. $\mu = 10$ on $[0, 20]$
  2. $\mu = 20$ on $[0, 35]$
  3. $\mu = 100$ on $[0, 250]$

- *Stiff solvers* are specially designed to efficiently handle systems that are stiff.

```
ode23s('vdpol',[0,1600],[1;1],options,1000)
```


15
10
5
0
−5
−10
−15
−3
−2
−1
0
1
2
3

Event Trapping

Each of MATLAB's solvers can trap *events*.

Event Trapping

Each of MATLAB's solvers can trap *events*.

- A special coding of the *odefile* is required.







Event Trapping

Each of MATLAB's solvers can trap *events*.

- A special coding of the *odefile* is required.
- You must inform the solver that *event trapping* is "on."

```
options=odeset('Events','on')
```

Event Trapping

Each of MATLAB's solvers can trap *events*.

- A special coding of the *odefile* is required.
- You must inform the solver that *event trapping* is "on."

```
options=odeset('Events','on')
```

- Some example files:
  - `ballode`
  - `orbitode`