Section 7.1: Systems of Ordinary Differential Equations

Remember that this is a supplement to the material in the book. Please read section 7.1 before looking at these course notes.

First a quick review of ordinary differential equations and why we study them.

Applied mathematics is the study of describing the natural world. In this study we are often more interested in how things change rather than how things are currently. We want to study how things change in order to predict how things will be in the future. Often it is easier to measure how things change.

In mathematics, rates of change are described by derivatives. Thus, an important field of mathematics is that of equations which have derivatives in them. This is a extremely broad field of study that includes ordinary and partial derivatives, linear and nonlinear equation, systems of equation and numerical techniques. We will cover some of the basics.

Definition: An n-th order, constant coefficient, linear ordinary differential equation has the form

$$\frac{d^n y}{d x^n} + a_{n-1} \cdot \frac{d^{n-1} y}{d x^{n-1}} + \cdots + a_1 \cdot \frac{d y}{d x} + a_0 \cdot y = f(x)$$

where a_i are constants.

First Order ODE's

Example 1: The growth rate of a population with unlimited resources is proportional to the population size. In mathematics this statement cam be modeled by the first order equation

$$\frac{dP}{dt} = k\,P.$$

This is one of the simplest ordinary differential equations (ODE's), called the Malthusian Population Model. We can solve this equation by the method of separation of variables and we get a solution of

$$P(t) = C \cdot e^{k\,t}.$$

In *Mathematica* we would use the **DSolve** command.

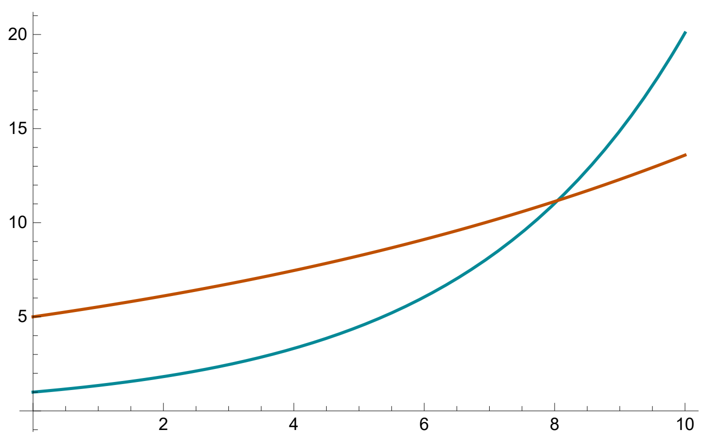
```
In[ ]:= sols = DSolve[{P'[t] == k*P[t], P[0] == P0}, P, t]
```

```
Out[ ]= {{P → Function[{t}, e^(k t) P0]}}
```

In this case the P0 is called an initial condition of the ODE while the k is called a parameter of the equation. We can graph this solution for several values of P0 and k as follows

```
In[ ]:= Plot[Evaluate[P[t] /. sols /. {{P0 → 1, k → 0.3}, {P0 → 5, k → 0.1}}], {t, 0, 10}]
```

Out[ ]=

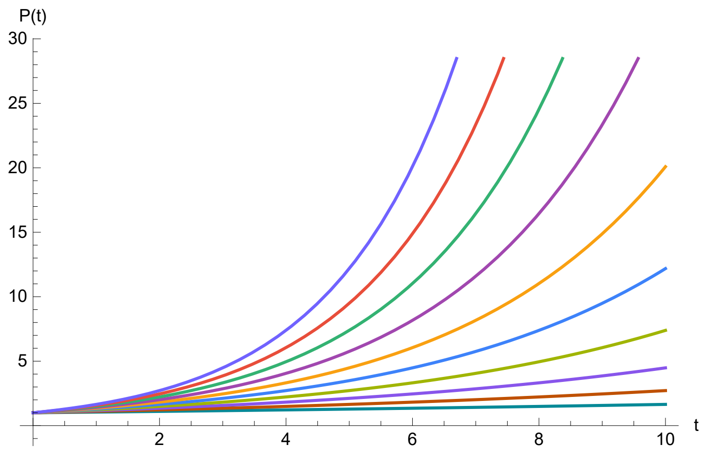


We often want to explore families of solutions with either varying values for a parameter or varying values of an initial condition. In the following we use the **Table** command to get various parameter values. The output is being suppressed by the semicolon.

In[∘]:=
```
Clear[P, t, k, P0]
malthus = P'[t] - k*P[t];
malsols = Table[k = 0.05*n;
   DSolve[{malthus == 0, P[0] == 1.0}, P, t], {n, 10}];
```

We graph these solutions as follows:

In[∘]:=
```
malplot = Plot[Evaluate[P[t] /. malsols], {t, 0, 10},
  AxesLabel → {"t", "P(t)"}]
```

Out[∘]=


Systems of First Order ODE's

Systems of ODE's come about when you have more than one dependent variable, one independent variable and the rate of change in the dependent variables can depend on any combination of the independent and dependent variables.

Example 2: Consider the following system that model the competition between two species of animals.

$$\frac{du}{dt} = 2\cdot u - u\cdot v,$$

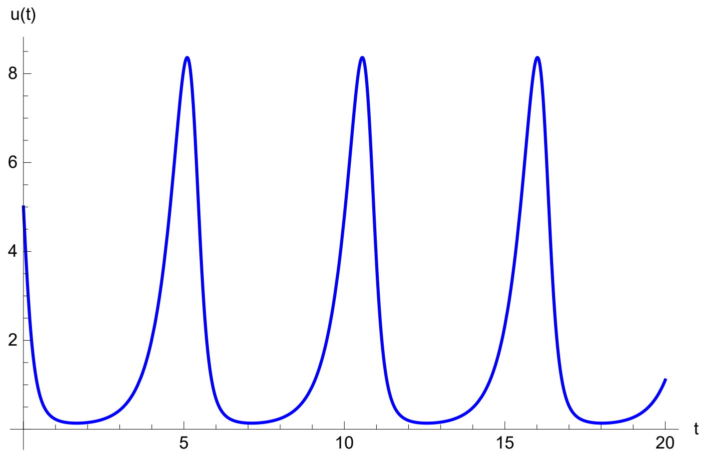
$$\frac{dv}{dt} = -v + \frac{1}{2}\,u\cdot v$$

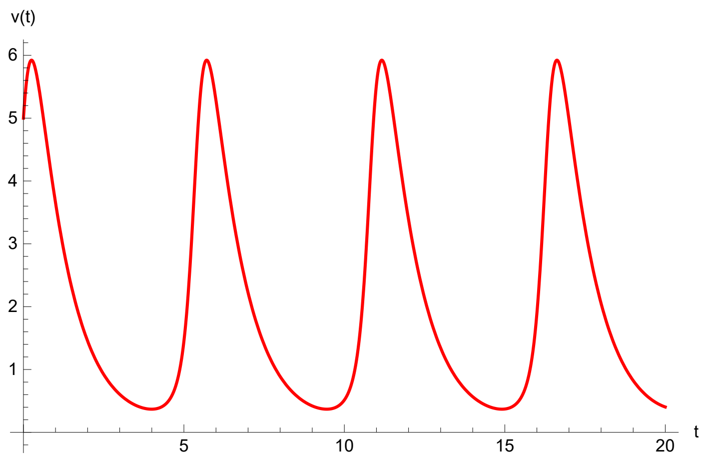
If the first species is u (the prey) and the second is v (the predator), then the first species is harmed, while the second benefits from interactions between the two species. This is a simple predator-prey model. The system is **autonomous** because the independent variable does not appear on the right hand side of the equations. These equations form a **non-linear** model because of the $u\cdot v$ terms in the equations. Non-linear problems often require special techniques to solve in closed form (if possible), so we will use commands that give us a numerical approximation to the solution. Lets see how *Mathematica* solves this system. Note in the following that the general form for the **NDSolve** command is {list of equations and initial conditions}, {list of things to solve for}, {interval to solve over} and then options which we will see a little later.

In[∘]:=
```
ppsols = NDSolve[{u'[t] == 2*u[t] - u[t]*v[t],
    v'[t] == -v[t] + 1/2*u[t]*v[t], u[0] == v[0] == 5}, {u, v}, {t, 0, 20}];
```

To see these solutions we can graph the approximations as follows

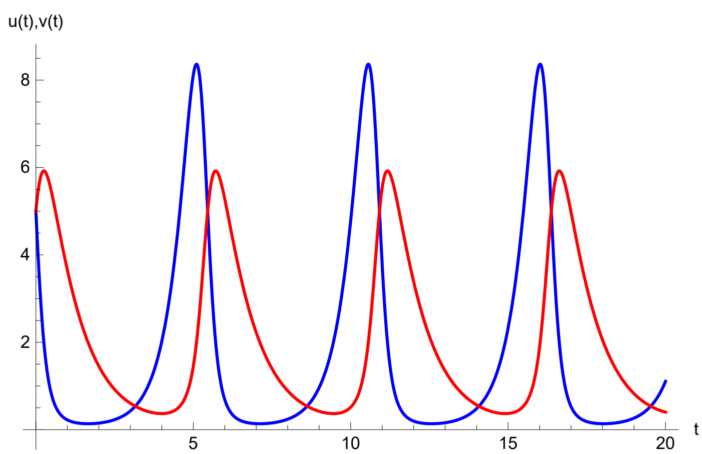
```
pu = Plot[Evaluate[u[t] /. ppsols], {t, 0, 20}, AxesLabel → {"t", "u(t)"}, PlotStyle → Blue]
pv = Plot[Evaluate[v[t] /. ppsols], {t, 0, 20}, AxesLabel → {"t", "v(t)"}, PlotStyle → Red]
```





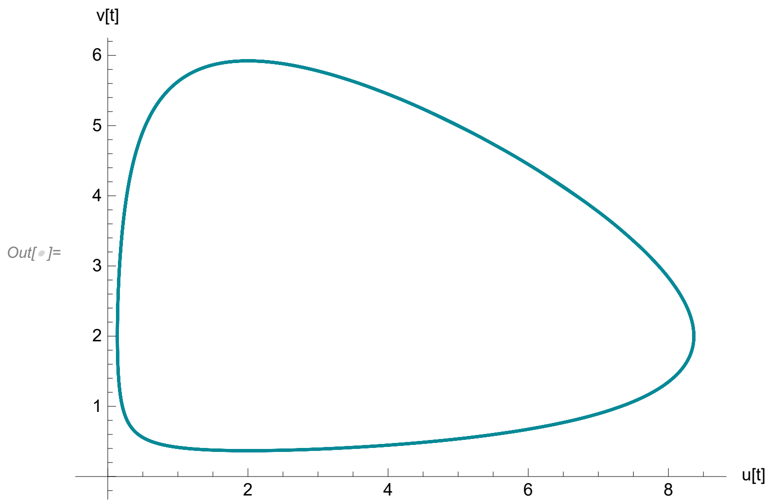
These graphs are called **state-space** graphs. This is because we are graphing the state or dependent variables versus the independent variable. To see both of these graphs at once, we can use the **Show** command:

```
Show[pu, pv, AxesLabel → {"t", "u(t),v(t)"}]
```



Another type of graph that is sometimes used is called a **phase-space** graph. Here we graph the state variables against each other and the independent variable appears parametrically:

```
In[•]:= ParametricPlot[Evaluate[{u[t], v[t]} /. ppsols], {t, 0, 20}, AxesLabel → {"u[t]", "v[t]"}]
```



Example 3: Solve the previous system of ODE's with initial conditions of (5,2), (5,4), (5,6), and (5,7). Graph all of the solutions on the same axes in phase-space along with the initial conditions.

First define a vector of initial conditions (note that the first component of each of the initial conditions is 5):

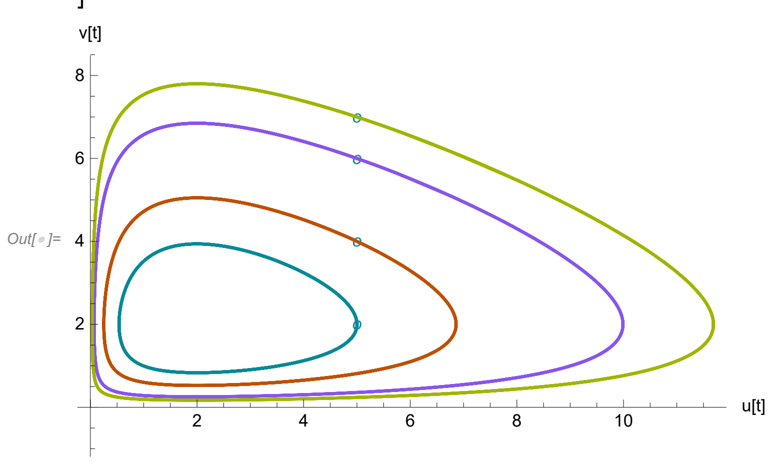
```
In[•]:= v0 = {2, 4, 6, 7}

Out[•]= {2, 4, 6, 7}
```

Then get a table of solutions:

```
In[•]:= Clear[ppsols]
        ppsols = Table[
           NDSolve[{u'[t] == 2*u[t] - u[t]*v[t],
             v'[t] == -v[t] + 1/2*u[t]*v[t], u[0] == 5, v[0] == v0[[n]]}, {u, v}, {t, 0, 20}]
           , {n, 1, 4}];
```

Now use the **Show** command to combine several graphs:

```
In[•]:= Show[
         ListPlot[{Table[{5, v0[[n]]}, {n, 1, 4}]}, PlotMarkers → {"o"}],
         ParametricPlot[Evaluate[{u[t], v[t]} /. ppsols], {t, 0, 20}],
         PlotRange → {-1, 8}, AxesLabel → {"u[t]", "v[t]"},
         AxesOrigin → {0, 0}
        ]
```



OK. Note that we use the **ListPlot** command to plot the initial conditions where the list of points is generated using the **Table** command (note that the option **PlotMarkers** sets how the points appear in the graph). We then use the **ParametricPlot** command to graph the four solutions that are the output of the **NDSolve** command. In the **Show** command we list the various graphics commands followed by some options that change the vertical axis, label the axes, and setting where the axes pass through on the graph.

Converting Higher Order DE's into Systems of First Order DE's

We can convert n^{th} order ODE's into systems of first order equations as follows:

Example 4: Convert $y'' + 3y' - 4y = \text{Sin}[t]$ into a system of first order equations.

To do this we will set $u_1 = y$ and $u_2 = y'$. Then $u_1' = y' = u_2$ and $u_2' = y'' = \text{Sin}[t] - 3y' + 4y$ (then substitute in for y and y') to get:

$$u_1' = u_2,$$

$$u_2' = 4u_1 - 3u_2 + \text{Sin}[t].$$

This is a linear system, that came from a linear second order equation. However, both the original DE and the system are not autonomous. That is, the independent variable appears in the Sin[t] term of the equations. We can convert the system to an autonomous system by adding another "dependent" variable: let $u_0 = t$. Then the system becomes,

$$u_0' = 1,$$

$$u_1' = u_2,$$

$$u_2' = 4u_1 - 3u_2 + \text{Sin}[u_0].$$

This is now a non-linear, autonomous system of first order ordinary differential equations. ☺

Example 5: Convert $y''' - y'y'' + e^t y = 0$ where $y(2) = 1, y'(2) = 4, y''(2) = -1$ into an autonomous system of first order equations centered at zero. Note that this equation is homogeneous, but not linear or autonomous.

Here we set $u_0 = t$, $u_1 = y$, $u_2 = y'$ and $u_3 = y''$. Then

$$u_0' = 1,\ u_0(0) = 2,$$

$$u_1' = u_2,\ u_1(0) = 1,$$

$$u_2' = u_3,\ u_2(0) = 4,$$

$$u_3' = u_2 u_3 - e^{u_0} u_1,\ u_3(0) = -1.$$

The left hand side of the last equation comes from solving the DE for y'''. The initial conditions for the system come directly form the initial conditions of the initial value problem.

You may have noticed that an n^{th} order autonomous equation can be written as a system of n first order autonomous equations and you would add one more equation to the system if you want to write a non-autonomous ODE as an autonomous system.

Practice Problems.

Try these problems in Mathematica, if available, and let me know if you have any questions.

1. Use the **DSolve** command to try to solve the following problems. If a closed form solution is possible (and aesthetically pleasing) then show that solution. If *Mathematica* does not find a nice closed form solution, then find a numerical solution with initial conditions of y(0)=1 and graph the solution.

a.

$$\frac{dy}{dx} = 2y - 1$$

b.

$$\frac{dy}{dx} = xy$$

c.

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

2. Use the **NDSolve** command to numerically approximate solutions to the following systems of equations. Generate both state and phase space graphs of the solutions (try different ranges of the independent variable to see what works).

a.

$$\frac{dx}{dt} = x + x\,y,\ \frac{dy}{dt} = x^2 - y,\ x(0) = 1,\ y(0) = 1/2.$$

b.

$$\frac{dx}{dt} = x\,y\,\sin(t),\ \frac{dy}{dt} = x\,y\,\cos(t),\ x(0) = 1,\ y(0) = 2.$$

c.

$$\frac{dx}{dt} = 2\,x - \frac{1}{2}\,x\,y\,,$$

$$\frac{dy}{dt} = \frac{1}{5}\,x\,y - y,$$

$$x(0) = 3,\ 6,\ 9,\ 12,\ 15;\ y(0) = 1.$$