Drug Dosages: Are They Effective? Are They Safe?

Introduction

OBJECTIVE: Use *Mathematica* to solve differential equations related to drug concentrations in the blood when the drug is administered by a single injection, intravenously, or by periodic injections.

You will learn how a simple mathematical model and *Mathematica* can be used to regulate the concentration of a prescribed drug in the blood. You will use *Mathematica*'s differential equation solver to obtain solutions for the drug concentration in the blood when the drug is administered by a single injection, intravenously, or by a series of periodic injections. You will also develop drug dosage regimens to keep the concentration of the drug within the safe and effective concentration limits.

Technology Guidelines

NOTE: If you have just finished a module, restart *Mathematica* or close the *Kernel* before executing a new module.
TO OPEN CELLS, put your cursor on the right cell bracket and double click.
INITIALIZATION CELLS
When asked if you want to ". . . automatically evaluate all the initialization cells in the notebook . . . ," respond by pressing the "Yes" button.
TO STOP AN EXECUTION
Select the *Kernel* pull-down menu and click on *Abort Evaluation.*
ORDER OF EXECUTION
Execute cells in the order given. Do not skip any Input cells within a given notebook.
SAVING NOTEBOOKS
You can save anytime to any directory you choose, and it is wise to save often. However, before you do your final save, delete all your output by selecting the *Delete All Output* selection under the *Kernel* pull-down menu.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, then shut down *Mathematica* and start it up again.

Part I: A Single Dose

An Initial Value Problem and its Solution

When a certain drug is administered by injection, the body absorbs it quickly so that the concentration of the drug in the blood rises to a peak value shortly after the drug is given. At the same time, the kidneys work to remove the drug from the blood, and the rate of removal is proportional to the drug concentration in the blood. This process is modeled by the initial value problem $\frac{dc}{dt}$ =-k c with c(0)=$c0$, where c(t) is the drug concentration in the blood at time t, and $c0$ is the initial concentration when the injection is given at time t=0. This model assumes that at the time of the injection, the drug is immediately absorbed into the blood.

Even though we know that c(t)=$c0$ e^{-kt} is the solution of the initial value problem, we will use *Mathematica*'s **DSolve[ ]** command to obtain this solution.

In[3]:=

```
Clear[c, k, c0];

soln = DSolve[{c'[t] == -k * c[t], c[0] == c0}, c[
```

Out[4]=

```
{{c[t] → c0 e^(-k t)}}
```

In[5]:=

```
soln = Flatten[soln]
```

Out[5]=

```
{c[t] → c0 e^(-k t)}

">  About Mathematica
```

Now we can form a function, **c[t_]**, by using the rule from the solution of the initial value problem to substitute the solution c0 e^{-kt} for the expression **c[t]**.

In[6]:=

```
c[t_] = c[t] /. soln
```

Out[6]=

$$c0\, e^{-k\,t}$$

The last command effectively replaces **c[t]** with $c0\, e^{-k\,t}$ and assigns it to a redefined function, **c[t_]**.

▪ Determining *k* and Plotting Concentrations

One hour after the drug is injected, a blood test from a particular patient shows that the drug concentration in the blood drops to 80% of its initial concentration. From this information, we can determine the value of *k* for this patient. We do this by setting **c[t]** equal to 0.80*c*0 at t=1 hour, and solving for *k*. (You may ignore the error message here.)

In[7]:=

```
ksoln = Solve[c[1] == 0.80 * c0, k]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so
   some solutions may not be found; use Reduce
   for complete solution information. More…
```

Out[7]=

```
{{k → 0.223144}}
```

```
">  About Mathematica
```

In[8]:=

```
ksoln = Flatten[ksoln]
```

Out[8]=

```
{k → 0.223144}
```

Let's redefine **c[t_]**, substituting the value for *k*.

In[9]:=

```
c[t_] = c[t] /. ksoln
```

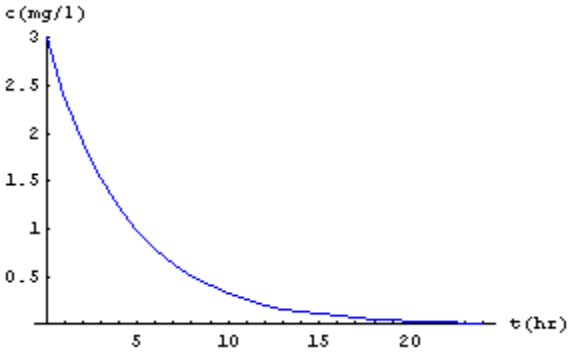
Out[9]=

$$c0\, e^{-0.223144\,t}$$

Suppose an injection of the drug raises the concentration to 3 milligrams per liter, and then the concentration decays as time progresses. Let's plot the solution, substituting 3 mg/l for $c0$.

In[10]:=

```
Plot[c[t] /. c0 → 3, {t, 0, 24}, PlotRange → {0,
 PlotStyle → {RGBColor[0, 0, 1]},
 AxesLabel → {"t(hr)", "c(mg/l)"}];
```

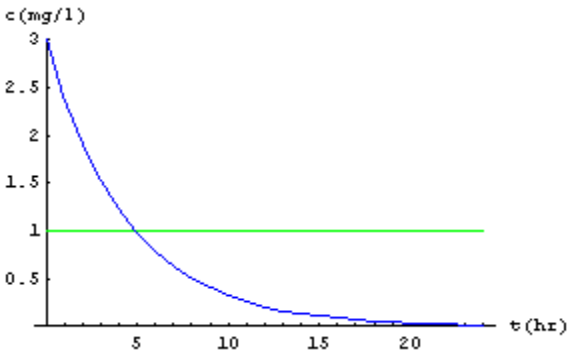


How Long is the Drug Effective?

This particular drug is effective only when the drug concentration in the blood is above 1 mg/l. Let's plot the drug concentration together with the effective threshold concentration.

In[11]:=

```
Plot[{1, c[t] /. c0 → 3}, {t, 0, 24}, PlotRange
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[0, 0
 AxesLabel → {"t(hr)", "c(mg/l)"}];
```



From the graph, can you estimate how long after the injection the drug will be effective for

this patient?

We can determine exactly how long after an injection the drug will be effective, assuming no other injections are given. We do this by finding the time when c(t) reaches 1 mg/l. (You may ignore the error message here.)

In[12]:=

```
teff = Solve[(c[t] /. c0 → 3) == 1, t]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so
   some solutions may not be found; use Reduce
   for complete solution information. More…
```

Out[12]=

```
{{t → 4.92334}}
```

This solution indicates that the drug will be effective for about 4 hours and 55 minutes after the patient receives an injection that boosts the drug concentration in the blood to 3 mg/l.

You Try It: On Another Patient

Another patient needs the same drug. A blood test on this new patient shows that after 1 hour, the drug concentration in the blood drops to 72% of the concentration right after the injection was given. (As before, you may ignore the error messages here.)

1. Determine the function that gives the drug concentration in the blood, *t* hours after the injection is given. To assist you, we have copied the commands you need in the following cell. Change the items that are highlighted in red.

In[13]:=

```
Clear[c, k, c0, soln, ksoln];

soln = Flatten[DSolve[{c'[t] == -k * c[t], c[0]

c[t_] = c[t] /. soln;

ksoln = Flatten[Solve[c[1] == 0.80 * c0, k]];
```

```
c[t_] = c[t] /. ksoln
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so
   some solutions may not be found; use Reduce
   for complete solution information. More…
```

Out[17]=

$$c0\, e^{-0.223144\, t}$$

2. For the new patient, determine the initial drug concentration, *c*0, so that the drug will remain effective for 5 hours. Assume that the effectiveness threshold is 1 mg/l.

In[18]:=

```
teff = Flatten[Solve[(c[t] /. c0 -> 3) == 1, t]]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so
   some solutions may not be found; use Reduce
   for complete solution information. More…
```

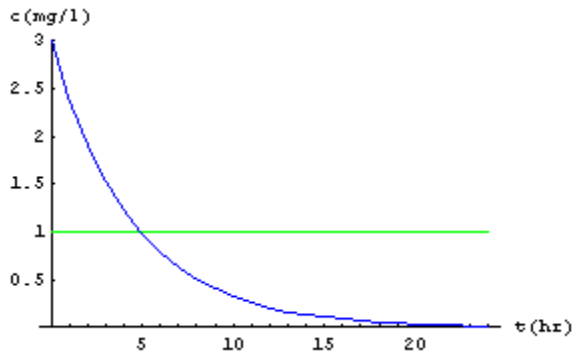
Out[18]=

```
{t → 4.92334}
```

3. Plot **c[t]** using the value for **c0** that you found in step 2.

In[19]:=

```
Plot[{1, c[t] /. c0 → 3}, {t, 0, 24}, PlotRange
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[0, 0
 AxesLabel → {"t(hr)", "c(mg/l)"}];
```



Part II: Intravenous Dosage

Sometimes it is desirable to maintain a constant concentration of the drug in the blood. This can be accomplished with intravenous (IV) injection. In this case, the drug is continuously fed into the bloodstream at a constant rate. We will modify the initial value problem in Part I to model intravenous administration of the drug.

A Model for Intravenous Dosage

The rate of change of the drug concentration in the blood is the difference between the rate at which the drug concentration increases in the blood due to the IV, and the rate at which it is removed by the kidneys. That is, $\frac{dc}{dt}$ =(rate of increase due to IV-rate of decrease due to kidneys). We assume that for IV administration of the drug, the rate of increase in the concentration of the drug due to the IV is constant and, as before, that the rate of removal is proportional to the concentration. Therefore, $\frac{dc}{dt}$ =r-k c, where *r* is the constant rate of increase in the drug concentration due to the IV and *kc* is the rate at which the kidneys reduce the concentration. For IV administration, we will assume that the initial concentration is 0, that is, c(0)=0. Now we can solve the initial value problem.

In[20]:=

```
Clear[c, k, soln];

soln =
 Flatten[DSolve[{c'[t] == r - k * c[t], c[0] == 0
   Simplify]
```

Out[21]=

$$\left\{c[t] \to \frac{r - e^{-k\,t}\,r}{k}\right\}$$

In[22]:=

```
c[t_] = Collect[soln[[1, 2]], E^(-k t)]
```

Out[22]=

$$\frac{r}{k} - \frac{e^{-k\,t}\,r}{k}$$

"> About *Mathematica*

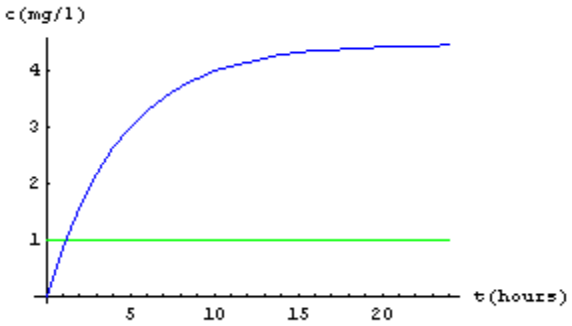
The Steady-State Concentration, c(∞)

What does c(t) approach as t→∞, if *k* is a positive constant? The limiting value is often called the steady state or equilibrium solution and is denoted by c(∞). You can also obtain this value by setting $\frac{dc}{dt}=0$ in the differential equation and solving for *c*. What do you get for the steady-state solution when you find it the second way? Is it the same result that you obtained by taking the limit of c(t) as t→∞?

Assuming that the patient is the same person as in Part I, so that the *k* value is the same, that is, k=0.223144, and assuming that the rate of increase in the drug concentration due to the IV is 1.0 $\frac{mg/l}{hour}$, we can plot the solution for c(t), substituting these values for *r* and *k*.

In[23]:=

```
Plot[{1, c[t] /. {r → 1.0, k → 0.223144}}, {t, 0
 AxesLabel → {"t(hours)", "c(mg/l)"},
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[0, 0
```



Using the rules **{r→1.0, k→0.223144}** to substitute values for the parameters *r* and *k,* as we did in the preceding plot command, is useful when we want to preserve the general form of **c[t]**. This way we can substitute other values into the general solution without having to redefine the function **c[t]**. We check to see that the general form of **c[t]** is preserved.

In[24]:=

```
c[t]
```

Out[24]=

$$\frac{r}{k}-\frac{e^{-k\,t}\,r}{k}$$

How Long Does it Take to be Effective?

Next, we determine how long it takes the drug concentration in the blood to reach the effective threshold value of 1 mg/l. (You may ignore the error message here.)

In[25]:=

```
Solve[(c[t] /. {r → 1.0, k → 0.223144}) == 1, t]
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so
   some solutions may not be found; use Reduce
   for complete solution information. More…
```

Out[25]=

```
{{t → 1.13156}}
```

It takes over an hour for the drug to become effective. One way to reduce this time is to administer a loading dose by injection at t=0. This instantly brings the drug concentration in the blood to 1mg/l, and then the IV is started. The differential equation model is the same as for IV administration without a loading dose, but the initial condition is different.

In[26]:=

```
Clear[c, k, soln];
```

```
soln =
 Flatten[DSolve[{c'[t] == r - k * c[t], c[0] == 1
   Simplify]
```

Out[27]=

$$\left\{c[t] \to \frac{e^{-k\,t}\,(k + (-1 + e^{k\,t})\,r)}{k}\right\}$$

In[28]:=

```
c[t_] = Collect[soln[[1, 2]], E^(-k t)]
```

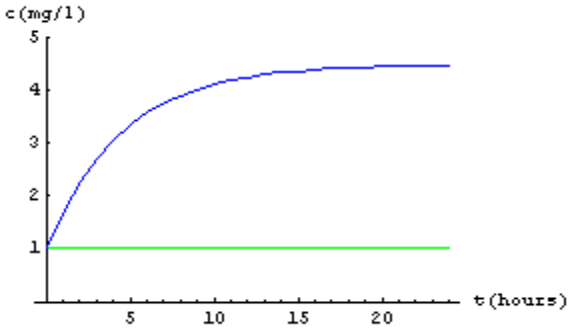
Out[28]=

$$\frac{e^{-k\,t}\,(k - r)}{k} + \frac{r}{k}$$

Let's plot **c[t]**.

In[29]:=

```
Plot[{1, c[t] /. {r → 1.0, k → 0.223144}}, {t, 0
 AxesLabel → {"t(hours)", "c(mg/l)"}, PlotRan
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[0, 0
```



You Try It: Is the Dosage Safe?

The drug is effective provided its concentration in the blood is above 1 mg/l, however, if the concentration exceeds 3 mg/l, the patient will have serious side effects from the drug. Consequently, the dosage prescriptions in Part II are unsafe for the patient.

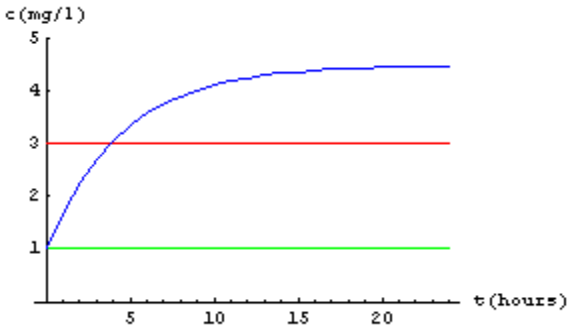
1. Modify the drug dosage prescription so that the concentration is always above the effective threshold and the steady-state concentration is halfway between the safe and effective threshold values, and plot **c[t]** over 24 hours. First use trial and error, and then use the analytic solution for the steady-state concentration to determine the correct value for *r*. To assist you we have copied the commands that you need in the following input cell. Change the items that are highlighted in red.

In[30]:=

```
Clear[c, k, soln];
rvalue = 1.0;
 kvalue = 0.223144;
 soln =
  Flatten[DSolve[{c'[t] == r - k * c[t], c[0] ==
    Simplify];
 c[t_] = Collect[soln[[1, 2]], E^(-k t)];
 Print["The drug concentration is ",
  c[t] /. {r → rvalue, k → kvalue}];
 Print["The steady-state concentration is ",
Plot[{1, 3, c[t] /. {r → rvalue, k → kvalue}}, {
  AxesLabel → {"t(hours)", "c(mg/l)"}, PlotRa
  PlotStyle → {RGBColor[0, 1, 0], RGBColor[1,
    RGBColor[0, 0, 1]}];
```

The drug concentration is
$4.48141 - 3.48141\, e^{-0.223144 t}$

The steady-state concentration is 4.48141



Out[30]=

$$\frac{dc}{dt}$$

2. Repeat exercise 1 for the other patient from You Try It: Part I.

Part III: Maintenance Dosages at Discrete Time Intervals

An alternative method for administering the drug is to give periodic injections to maintain a safe and effective drug concentration in the blood. Again, we assume that the drug is completely absorbed as soon as it is administered and that the kidneys reduce the drug concentration at a rate proportional to the concentration *c*. The differential equation model is [redacted]=-k c+d(t), where d(t) is the rate of increase in the drug concentration due to administration of the drug. The initial condition is the concentration of the drug in the blood at t=0. If there is no loading dosage, then c(0)=0, but if a loading dose of *c*0 is given, then c(0) =*c*0.

First, we specify a maintenance dosage that we will administer periodically.

In[31]:=

```
dosageschedule = maintenancedose * DiracDelta[
  maintenancedose * DiracDelta[t - 8]
```

Next, we use a special mathematical function called the Dirac delta to model the periodic injections. The Dirac delta allows us to increase the drug concentration in the blood by a finite amount at specified times. In the next cell, we show how to use the Dirac delta to model giving a maintenance dosage at *t* = 4 hour and then again at t=8 hours.

In[32]:=

```
2. DiracDelta[-8 + t] + 2. DiracDelta[-4 + t]
```

Out[32]=

```
Clear[c, k] ;
```

Now we solve the initial value problem to determine the drug concentration in the blood as a function of time. We will take the loading dosage to be 2.0, the same as the maintenance dosage, and replace d(t) in the differential equation with **dosageschedule** .

In[33]:=

```
soln =
 Flatten[
  DSolve[{c'[t] == -k * c[t] + dosageschedule,
    c[0] == maintenancedose}, c[t], t]]
```

Out[34]=

```
{c[t] →
  e^(-1. k t) (2. + 2. 2.71828^(8. k) UnitStep[-8. + t] +
     2. 2.71828^(4. k) UnitStep[-4. + t])}
```

```
c[t_] = c[t] /. soln
```

The **UnitStep[ ]** function that appears in the solution is another special mathematical function. It acts like an "on-off" switch. The function **UnitStep[t, a]** has a value of 0 when t<a, and a value of 1 when t≥a. The unit step function is also called the Heaviside step function.

Next, we redefine **c[t_]** and plot the solution.

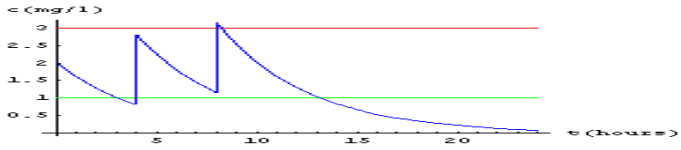
In[35]:=

```
e^(-1.k t) (2. + 2. 2.71828^(8. k) UnitStep[-8. + t] +
   2. 2.71828^(4. k) UnitStep[-4. + t])
```

Out[35]=

```
Plot[{1, 3, c[t] /. k → kvalue}, {t, 0, 24},
 AxesLabel → {"t(hours)", "c(mg/l)"},
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[1, 0
   RGBColor[0, 0, 1]}];
```

In[36]:=




```
maintenancedose = 2.5;
```

Now we use the Dirac delta and the **Sum[ ]** command to build a dosage maintenance schedule to use as an input function for the differential equation. It is apparent that the previous schedule fell outside the safe and effective range, so we will try something different. We will try 2.5 mg/l every 6 hours.

In[37]:=

```
deltat = 6;
```

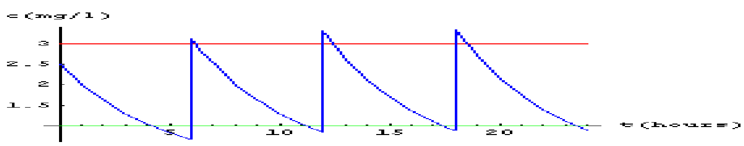
```
dosageschedule = Sum[maintenancedose * DiracDe
  {n, 1, 24 / deltat}];
```

```
Clear[c, k];
```

```
soln =
 Flatten[
  DSolve[{c'[t] == -k * c[t] + dosageschedule,
    c[0] == maintenancedose}, c[t], t]];
```

```
c[t_] = c[t] /. soln;
```

```
Plot[{1, 3, c[t] /. {r → rvalue, k → kvalue}}, {
 AxesLabel → {"t (hours)", "c (mg/l)"},
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[1, 0
   RGBColor[0, 0, 1]}];
```




```
maintenancedosage = 2.5;
```

Well, that didn't work either.

You Try It: Safe and Effective Drug Dosage

1. Try to build a drug dosage schedule that keeps the drug concentration in the safe and effective range, that is, between 1.0 and 3.0, and that maximizes patient comfort at the same time. We have copied the commands that you will need in the next cell. Change the items that are highlighted in red.

In[44]:=

```
deltat = 6;
```

```
dosageschedule =
 Sum[maintenancedosage * DiracDelta[t - k * del
  [k, 1, 24 / deltat]]
```

```
Clear[c, k];
```

```
soln =
 Flatten[
  DSolve[{c'[t] == -k * c[t] + dosageschedule,
    c[0] == maintenancedosage}, c[t], t]]
```

```
c[t_] = c[t] /. soln;
```

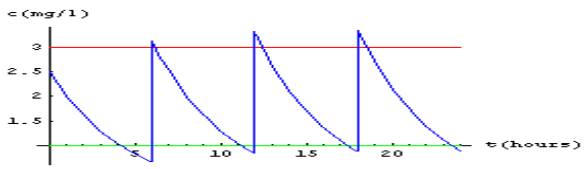
```
Plot[{3, c[t] /. {r → rvalue, k → kvalue}}, {
 AxesLabel → {"t(hours)", "c(mg/l)"},
 PlotStyle → {RGBColor[0, 1, 0], RGBColor[1, 0
   RGBColor[0, 0, 1]}};
```

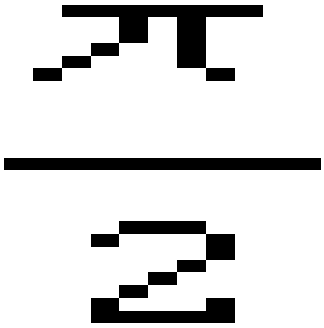
```
2.5 DiracDelta[-24 + t] +
  2.5 DiracDelta[-18 + t] +
  2.5 DiracDelta[-12 + t] +
  2.5 DiracDelta[-6 + t]
```

Out[46]=

```
{c[t] → e^(-1. k t)
   (2.5 + 2.5 2.71828^(24. k) UnitStep[-24. + t] +
     2.5 2.71828^(18. k) UnitStep[-18. + t] +
     2.5 2.71828^(12. k) UnitStep[-12. + t] +
     2.5 2.71828^(6. k) UnitStep[-6. + t])}
```

Out[48]=





2. The Dirac delta function and the Heaviside step function are very useful in many science and engineering applications. Research more about them, and write a one or two page report summarizing what you find.

About *Mathematica*

The **Flatten[ ]** command removes the inner curly brackets from a list. We do this so that the solution is in the form of a rule inside a set of single curly brackets, and we can use this for substitutions in later expressions. The double curly brackets will not work as a list of rules. To learn more about the **Flatten[ ]** command, pull down the Help menu, select the Help Browser, and type **Flatten**. To learn more about rules, refer to the Overview of *Mathematica* module included with this supplement or go to the Help Browser and type **Rule**. Go Back.

The error message that displays when you execute the **Solve[ ]** command is to warn you that whenever *Mathematica* uses inverse functions to solve an equation there may be some solutions that are missed. A good example of this is the equation sin x=1. *Mathematica* will use the inverse sine function to solve this equation, giving $x = \frac{\pi}{2}$ (with the same warning message). We all know, however, that the equation actually has infinitely many solutions. They are x = (4n+1)e^{-kt}, where n can be any integer. You might try using *Mathematica* to solve sin x=1. ">Go Back.

The **Collect[ ]** command in the preceding cell collects together all of the terms that include like powers of [ ] as factors. To learn more about the **Collect[ ]** command, pull down the Help menu, select the Help Browser, and type **Collect**. Go Back.

Created by *Mathematica* (August 28, 2005)