Rain Catchers, Elevators, and Rockets

Introduction

OBJECTIVE: Gain an appreciation for the importance of the Fundamental Theorem of Calculus and the Mean Value Theorem in practical applications.

The purpose of this module is to show the wide applicability of some of the basic ideas in calculus of single variable functions. What do flood control, riding an elevator, and launching a model rocket have in common? We can describe all of these, using calculus. In addition to gaining a better understanding and appreciation for the Fundamental Theorem of Calculus and the Mean Value Theorem, you will also see the importance of finding the areas between the graphs of functions and identifying extreme values. Let's get started.

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: Rain Catchers

Background

Property owners are required by law to ensure that water running off their properties during a storm does not do harm to their neighbors. This can often be a challenge for cities, towns, and municipalities where large flows of water can run over public properties and in streets and roadways, presenting a potential threat to private property owners. As a result, public as well as private entities have to take measures to control storm runoff. Numerous examples of such efforts can be found in and around the Los Angeles basin in California.

In this module, you will explore how one would go about designing a detention basin to collect runoff water from a rainstorm and release it at a controlled flow rate.

Here Comes the Rain

During a severe 24-hour storm, water flows out of a drainage channel at the rate given by $\frac{dQ}{dt} = 10000\, t^2\, e^{-t/2}$, where t is hours and $\frac{dQ}{dt}$ is cubic feet per hour. Plot the flow rate function and determine what the maximum flow rate is and when it occurs.

In[127]:=

```
Clear[Q];
```

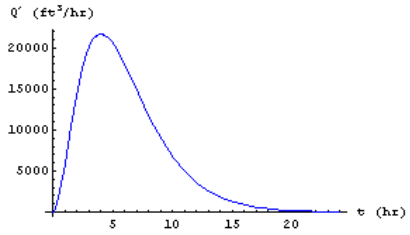
```
Q'[t_] = 10000 t^2 Exp[-t / 2]
```

Out[128]=

$$10000\, e^{-t/2}\, t^2$$

In[129]:=

```
Plot[Q'[t], {t, 0, 24},
 AxesLabel -> {"t (hr)", "Q' (ft^3/hr)"},
 PlotStyle → {RGBColor[0, 0, 1]}];
```



In[130]:=

```
Q''[t_] = D[Q'[t], t] // Simplify
```

Out[130]=

$$-5000\, e^{-t/2}\, (-4 + t)\, t$$

In[131]:=

```
soln = Solve[Q''[t] == 0, t]
```

Out[131]=

```
{{t → 0}, {t → 4}}
```

In[132]:=

```
Q'[4] // N
```

Out[132]=

```
21653.6
```

The maximum flow rate occurs at $t = 4$ hours and is 21653.6 cubic feet per hour. This is confirmed by the graph of $Q'[t]$, and by checking to see that the second derivative of $Q'[t]$ at $t = 4$ hours is negative.

In[133]:=

```
D[Q'[t], {t, 2}] /. t → 4 // N
```

Out[133]=

```
-2706.71
```

Imagine that you have been asked to design a storm drain detention basin to catch and hold all of the water that flows out the drainage channel during this severe 24-hour storm. How much water would the detention basin have to hold?

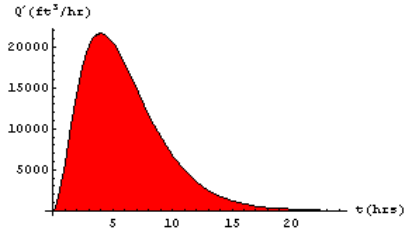
The total amount of water that flows out of the drainage channel during the 24-hour storm is the area under the graph of $Q'[t]$. You can show this area by loading a special graphics package called **Graphics`FilledPlot`** that contains the **FilledPlot[ ]** command.

In[134]:=

```
<< Graphics`FilledPlot`
```

In[135]:=

```
FilledPlot[Q'[t], {t, 0, 24},
 Fills → {RGBColor[1, 0, 0]},
 AxesLabel → {"t(hrs)", "Q'(ft^3/hr)"}];
```



The definite integral gives the area under the graph, which is the total amount of water that flows out of the drainage channel during the 24-hour storm.

In[136]:=

$$Q[24] = \int_0^{24} Q'[t]\,dt \;//\; N$$

Out[136]=

```
159916.
```

The detention basin must be able to hold about 160,000 cubic feet of water. To get an idea of how much water this is, think of a cube with this much volume. The length of each side would have to be $\sqrt[3]{160000}$ ft or

In[137]:=

$$\sqrt[3]{160000} \;//\; N$$

Out[137]=

```
54.2884
```

What is the average flow rate during the storm? You can calculate the average flow rate two different ways. In the first, calculate the average value of the flow rate function between t=0 and t=24 hours.

In[138]:=

$$\bar{Q}' = \frac{1}{24 - 0} \int_0^{24} Q'[t]\,dt \;//\; N$$

Out[138]=

```
6663.18
```

In the second, take the total amount of water that flows out of the pipe during the 24-hour period and divide it by 24 hours. That is, $\bar{Q}' = \frac{\Delta Q}{\Delta t} = \frac{Q(24) - Q(0)}{24 - 0}$, where Q(0)=0.

In[139]:=

```
Q[0] = 0;

Q'bar = (Q[24] - Q[0])/(24 - 0)
```

Out[140]=

```
6663.18
```

Therefore, the average flow is 6663.18 $\frac{ft^3}{hr}$. The fact that the two values for the average flow rate are the same should not be surprising since $\int_0^{24} Q'(t)dt = Q(24)-Q(0)=\Delta Q$ by the Fundamental Theorem of Calculus, Part II.

At what time(s) during the 24-hour event does the flow rate equal the average rate of flow?

In[141]:=

```
Solve[Q'[t] == Q̄', t]
```

```
InverseFunction::ifun :
 Inverse functions are being used. Values may be lost for multivalued inverses. More…

InverseFunction::ifun :
 Inverse functions are being used. Values may be lost for multivalued inverses. More…

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

Out[141]=

```
{{t → -0.687397}, {t → 1.06541}, {t → 10.0371}}
```

```
"> About Mathematica
```

Only the positive values for t make sense since $Q'[t]$ is only defined for $t \geq 0$.

If the detention basin is empty at $t = 0$, find a function that gives the total amount of water Q in the basin as a function of time during the 24 hours, and graph the function.

In[142]:=

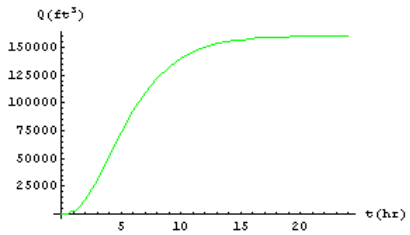
```
Q[t_] = ∫_0^t Q'[u] du + Q[0] // Simplify
  (*from the Fundamental Theorem of
    Calculus, Part 1*)
```

Out[142]=

$$10000\,(16 - 2\,e^{-t/2}\,(8 + 4\,t + t^2))$$

In[143]:=

```
Plot[Q[t], {t, 0, 24},
 AxesLabel → {"t(hr)", "Q(ft^3)"},
 PlotStyle → {RGBColor[0, 1, 0]}];
```



▪ Watch it Fill

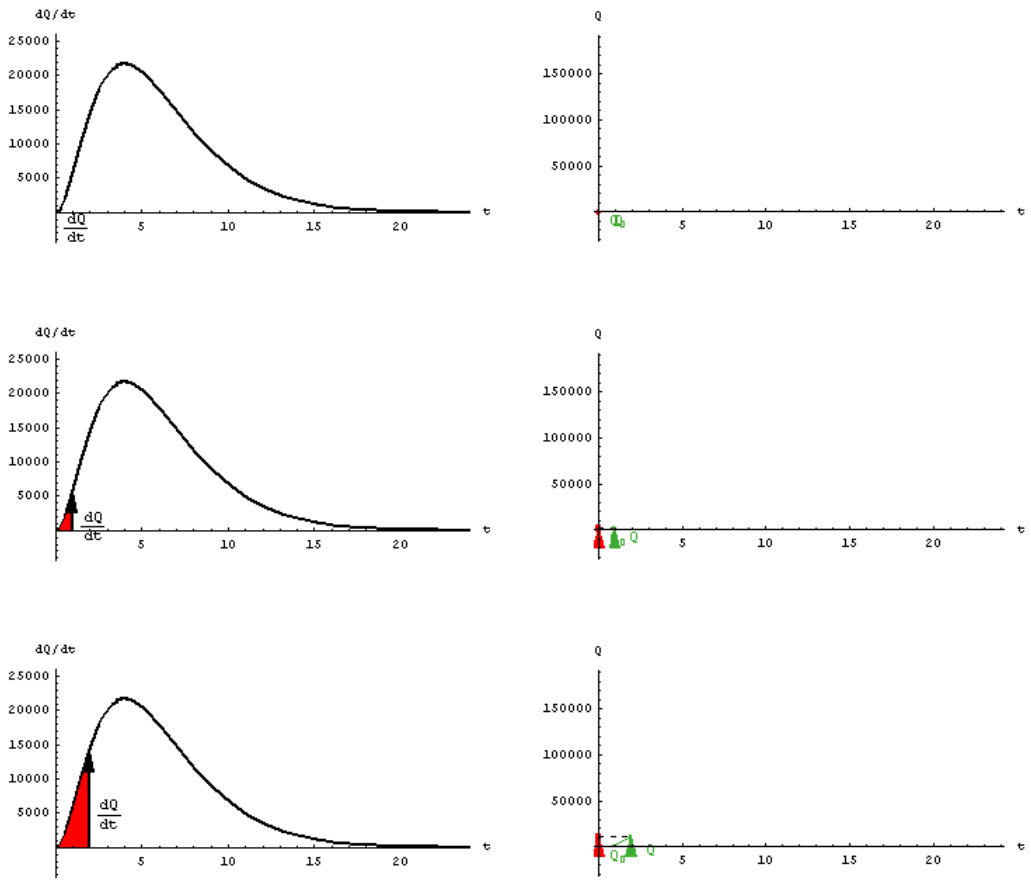
We can watch the water accumulate in the detention basin using the command, **antidifferentiate[f[t], t, a, b, f[a]]**, which was designed specially for this demonstration. The arguments of the function are: **f[t]**, the rate of change function, **t**, the independent variable, **a**, the beginning time, **b**, the ending time, and **f[a]**, the initial value of **f[t]**.

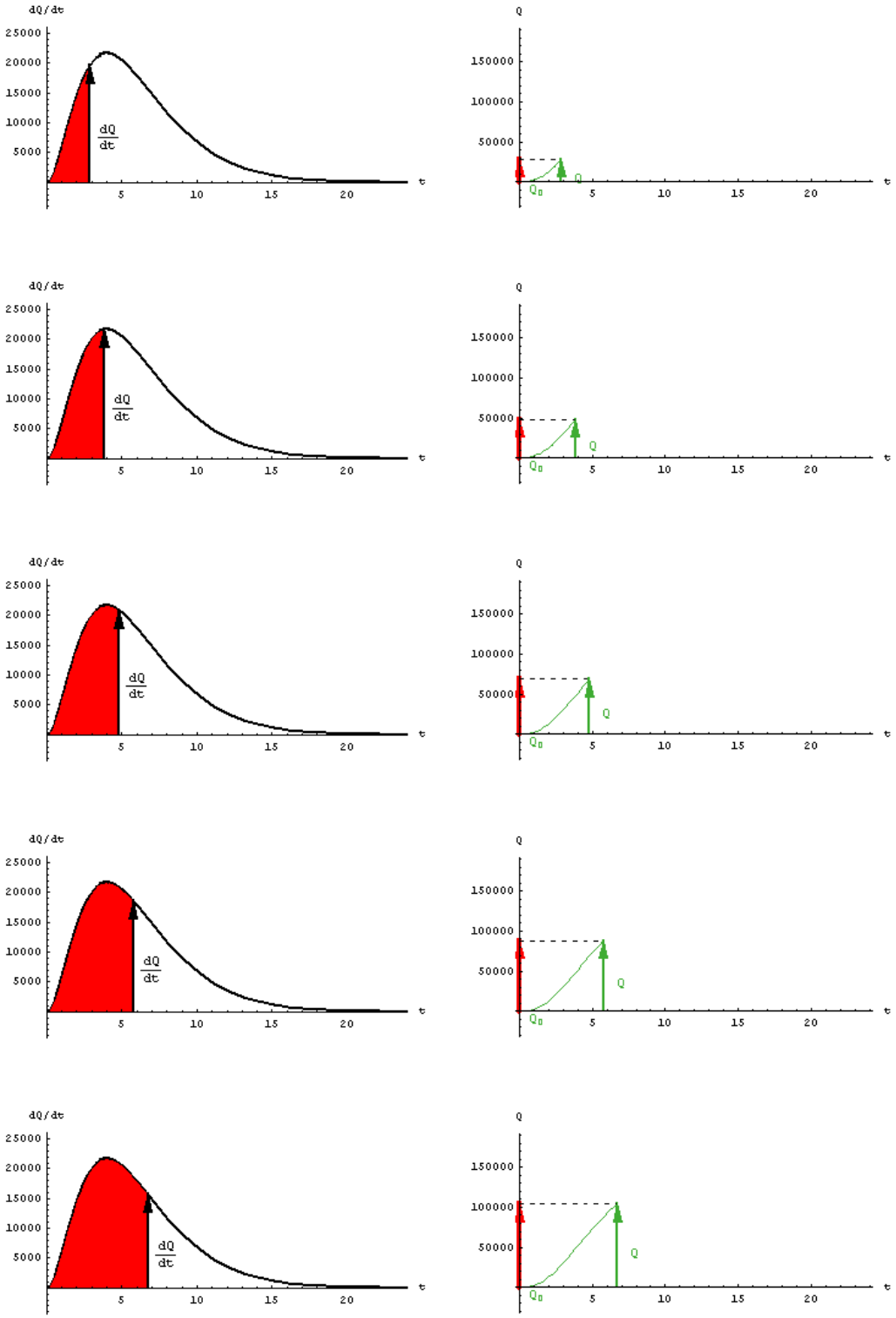
To animate the sequence of graphs:
1. If necessary, widen the notebook window so that all three graphs in each cell show across the page.
2. Put the cursor in the cell bracket that contains all the graphics cells, and double click the left mouse button. This will collapse all the graphs into one cell, displaying only the first graphics cell in the sequence.
3. Be sure the cell bracket that contains the collapsed graphics cells is selected (if not, place the cursor in the cell bracket and click once), and then press Ctrl+Y. This will play the sequence of graphics slides to generate the animation.
4. While the animation is playing, a control bar appears at the bottom of the notebook window. This bar allows you to control the speed and direction of the animation.

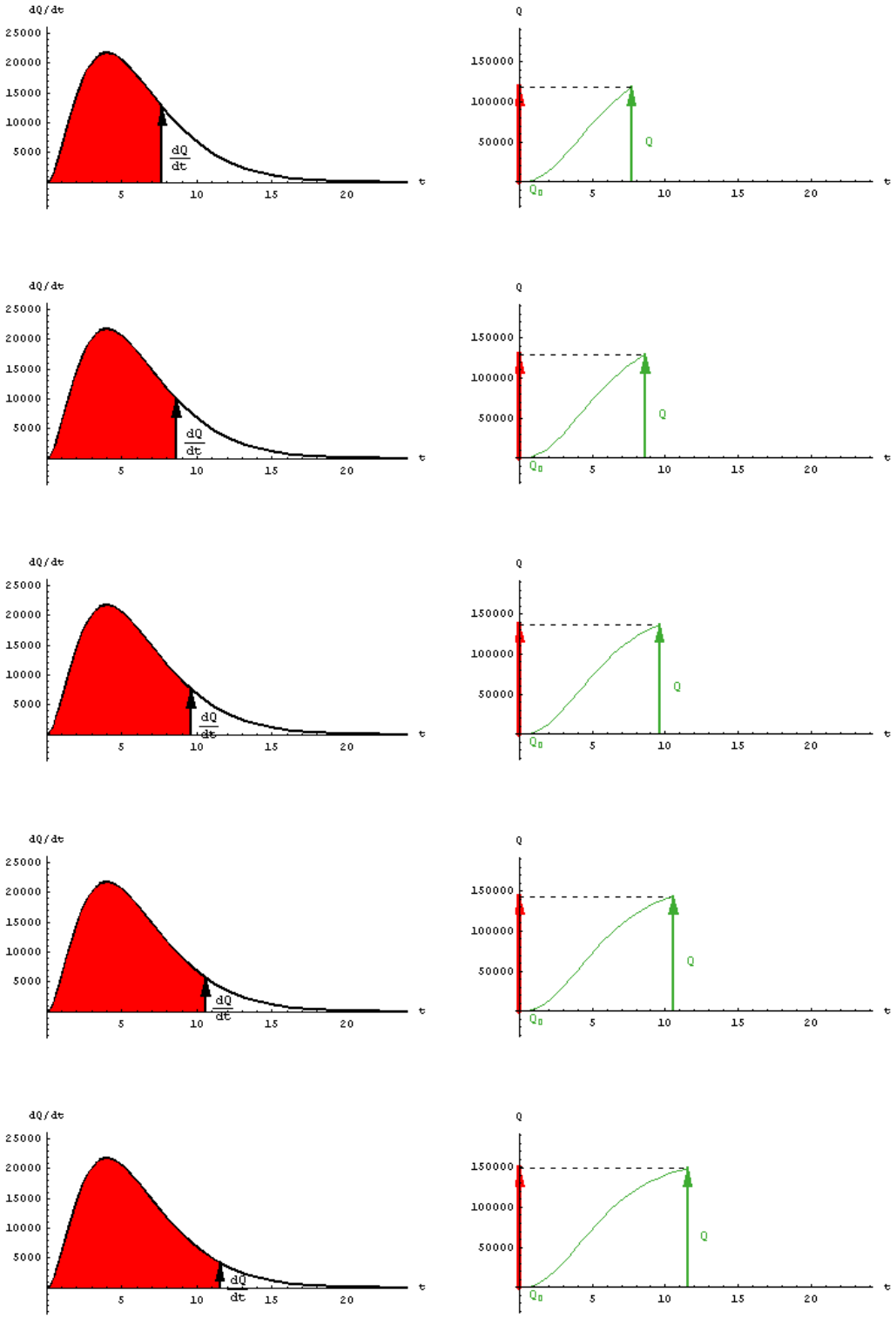
Since the next command generates a lot of output, you probably won't want to print this notebook until after you have deleted most or all of the graphs. We recommend deleting all but a few representative cells from the sequence before you print.

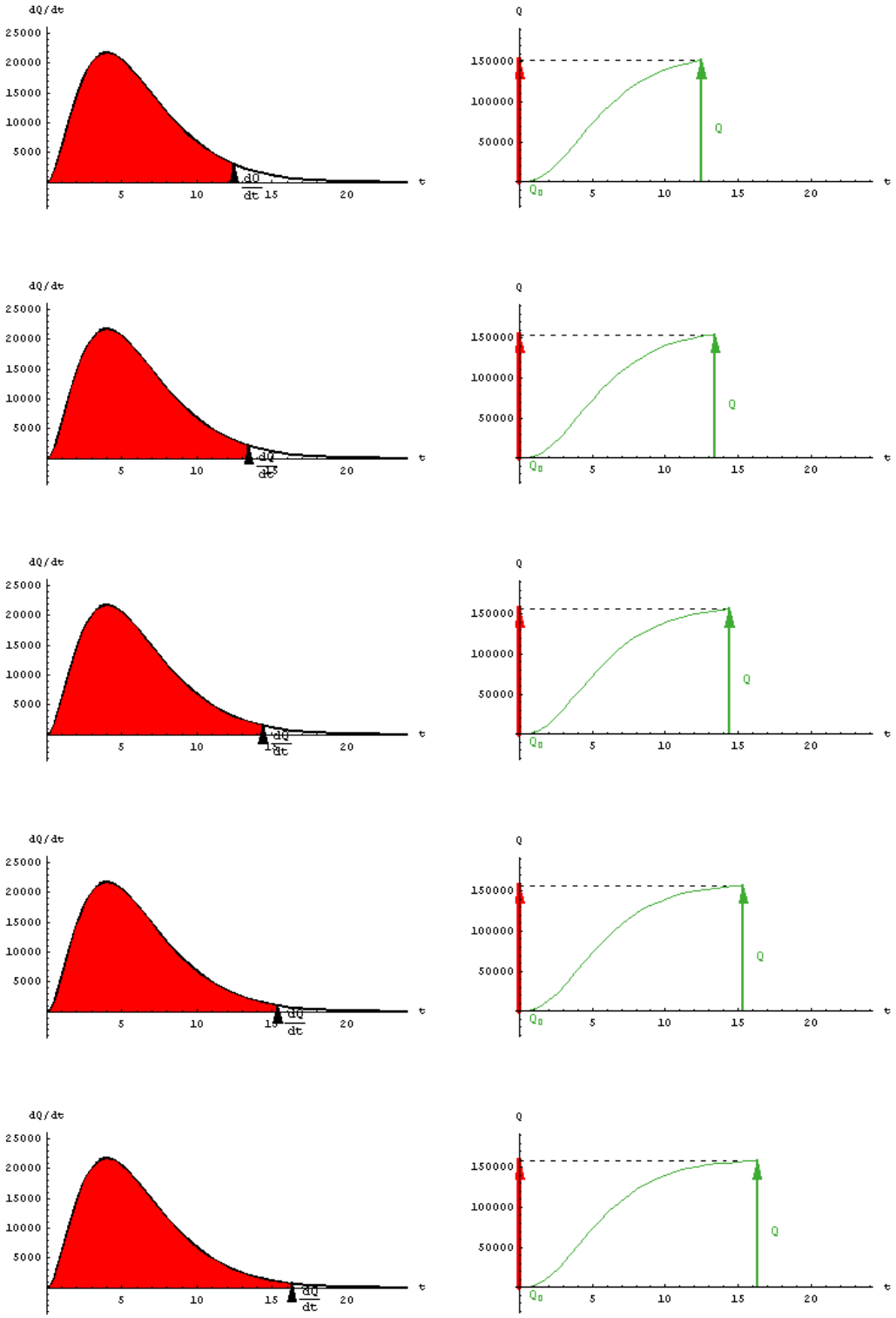
In[144]:=

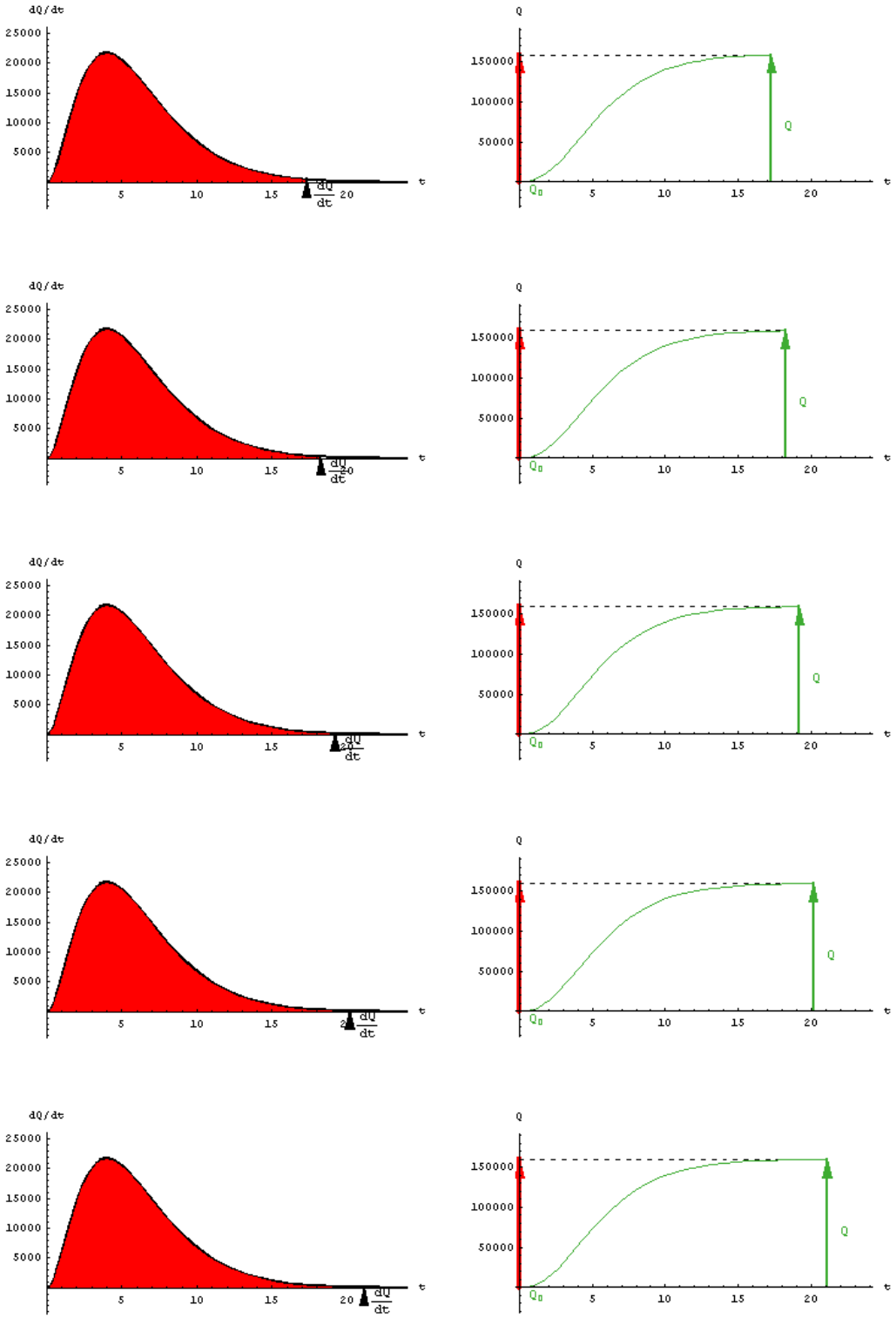
```
antidifferentiate[Q'[t], t, 0, 24, 0]
```

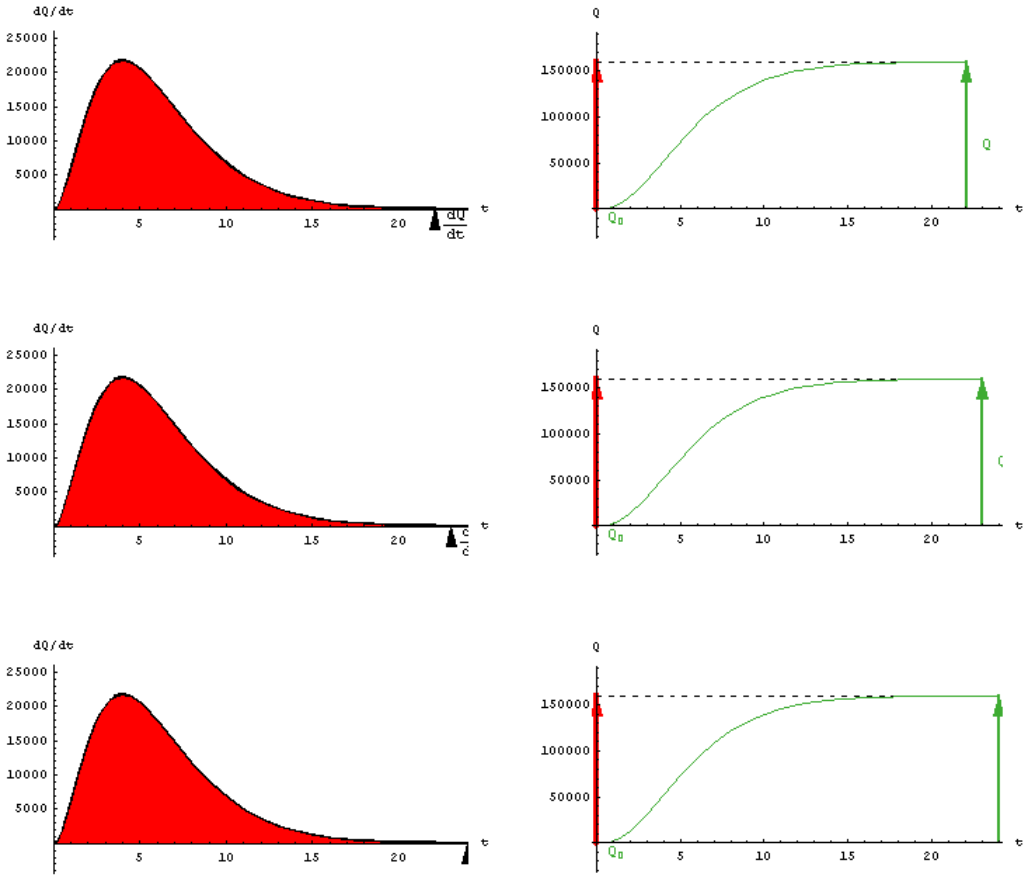



dQ/dt
25000
20000
15000
10000
5000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0
5
10
15
20
t


dQ/dt
25000
20000
15000
10000
5000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0
5
10
15
20
t


dQ/dt
25000
20000
15000
10000
5000
5
10
15
20
t
dQ/dt
Q
150000
100000
50000
Q
Q0


dQ/dt
25000
20000
15000
10000
5000
5
10
15
20
t
dQ/dt
Q
150000
100000
50000
Q
Q0



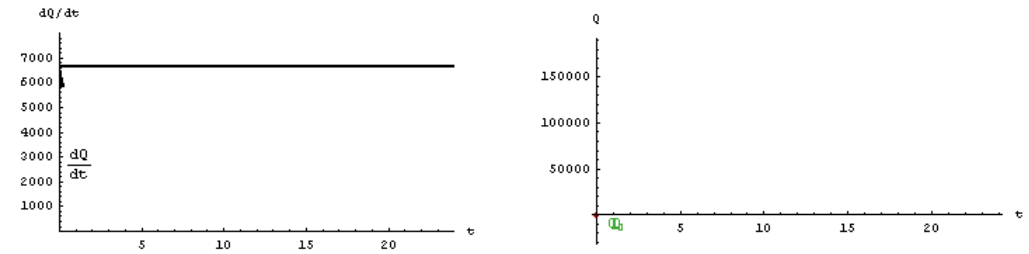
If the reservoir were to fill at a constant rate over the 24-hour period, what would the constant fill rate be? It would, of course, be the average rate of flow, $\bar{Q}'$=6663.18 $\frac{ft^3}{hr}$, that we calculated above. This is true because we know from the definition of the average value of a function that $\bar{Q}' \times (t_f - t_i) = \int_{t_i}^{t_f} Q' dt$. Let's watch the reservoir fill at the average rate instead of the actual.

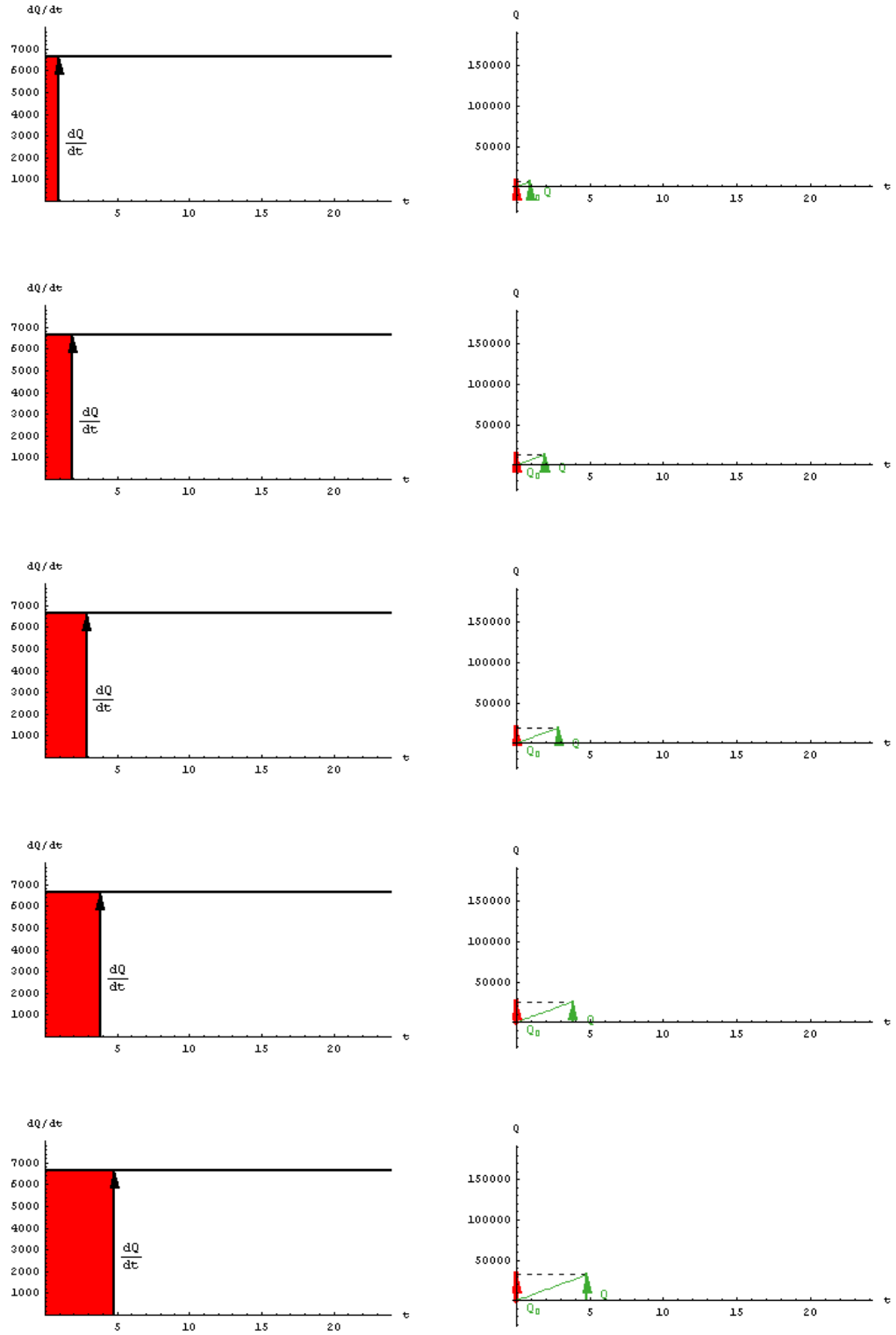
Note: The **antidifferentiate[ ]** command generates a lot of graphics, which may fill up your computer's memory. Before executing the next command, pull down the Kernel menu and select Delete All Output. All computed values are saved when you do this.

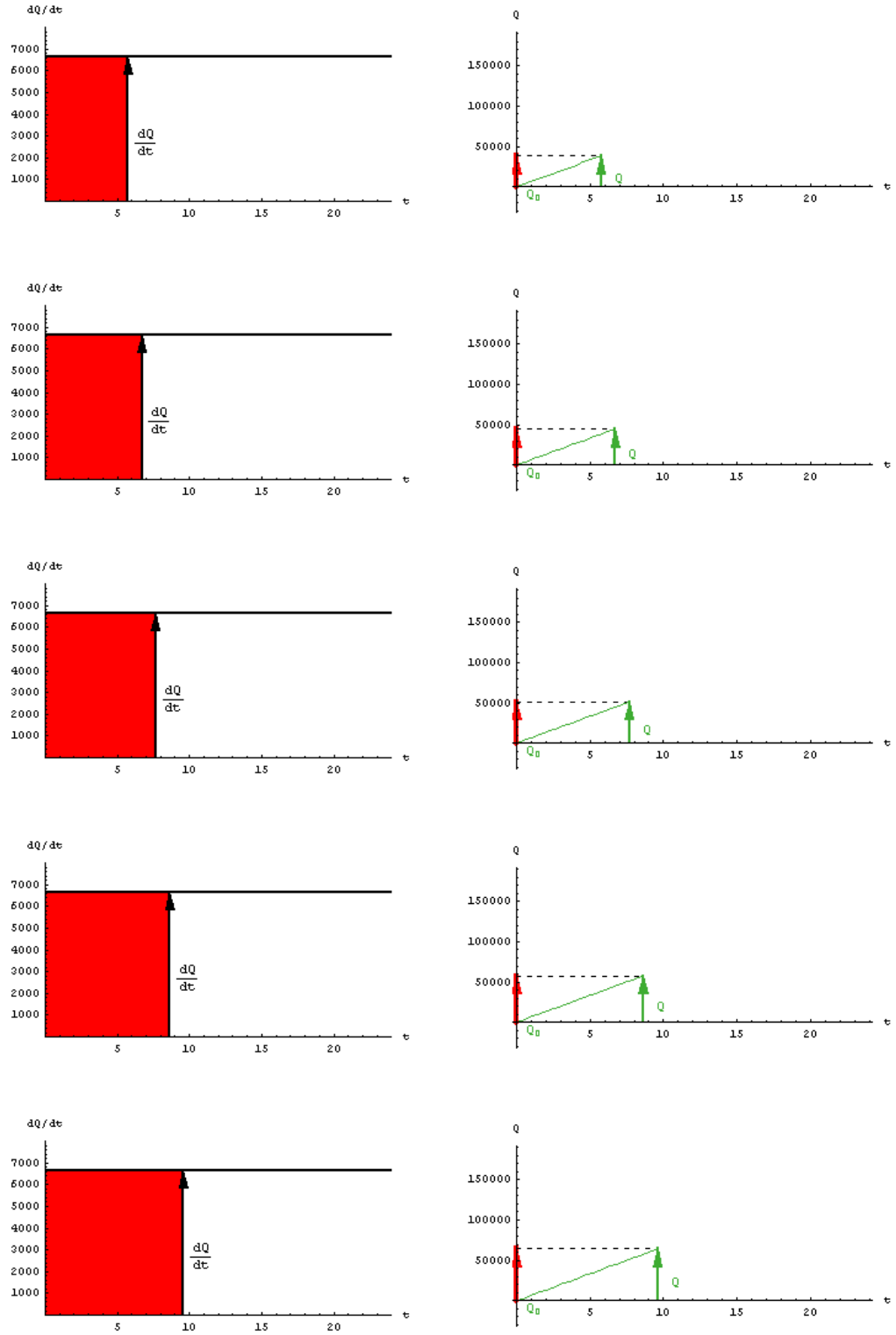
Since the next command generates a lot of output, you probably won't want to print this notebook until after you have deleted most or all of the graphs. We recommend deleting all but a few representative cells from the sequence before you print.

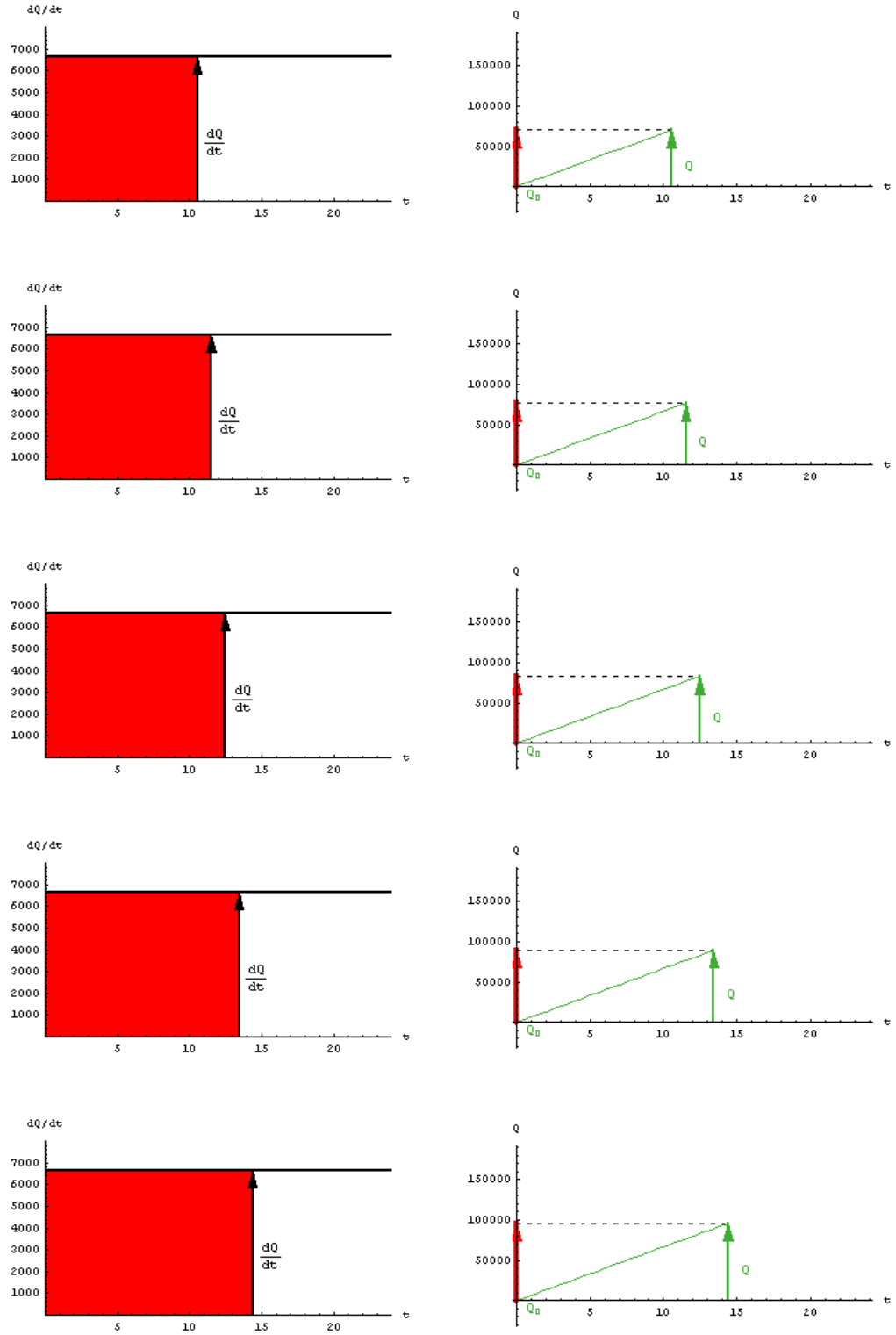
In[145]:=

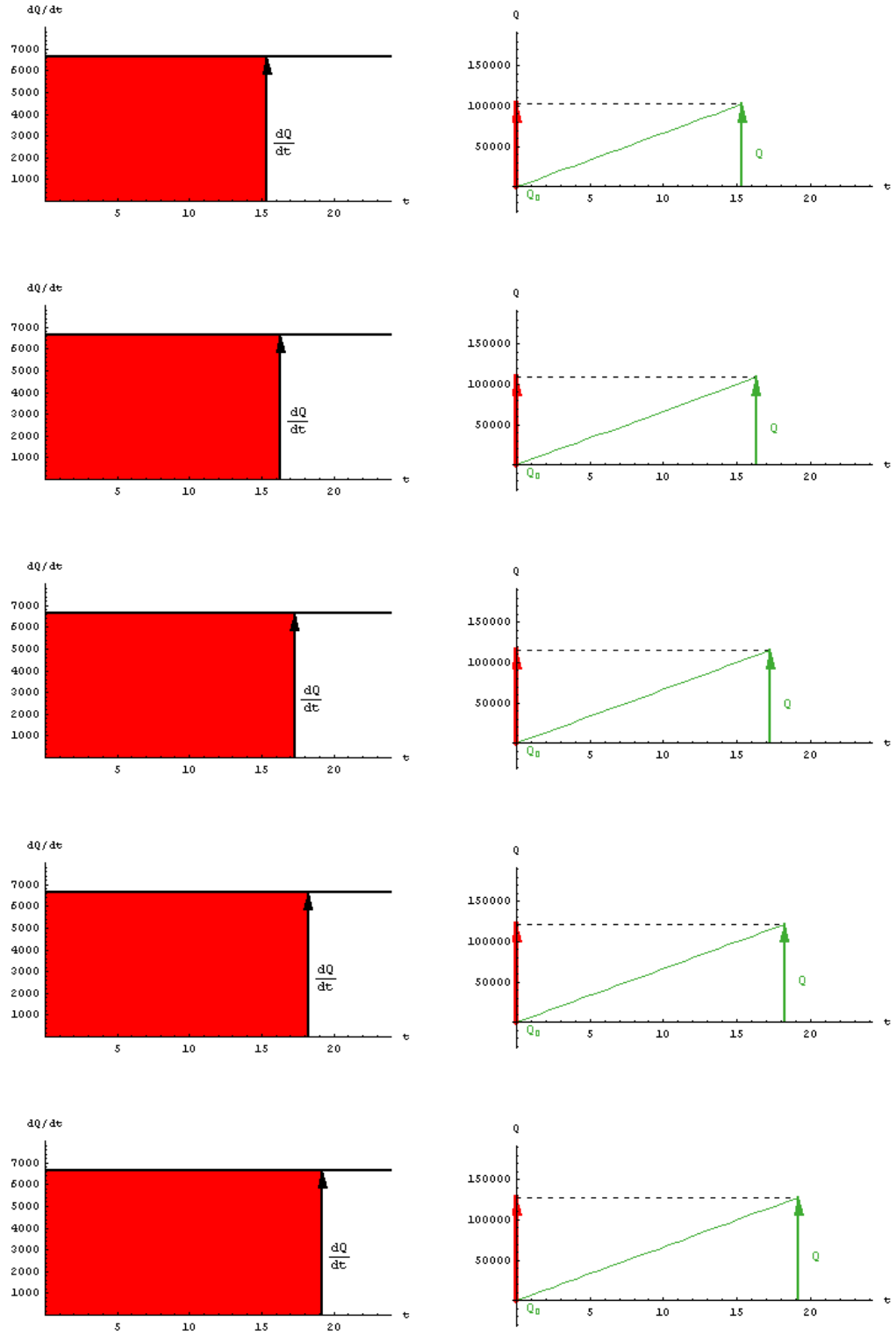
```
antidifferentiate[Q̄′, t, 0, 24, 0]
```

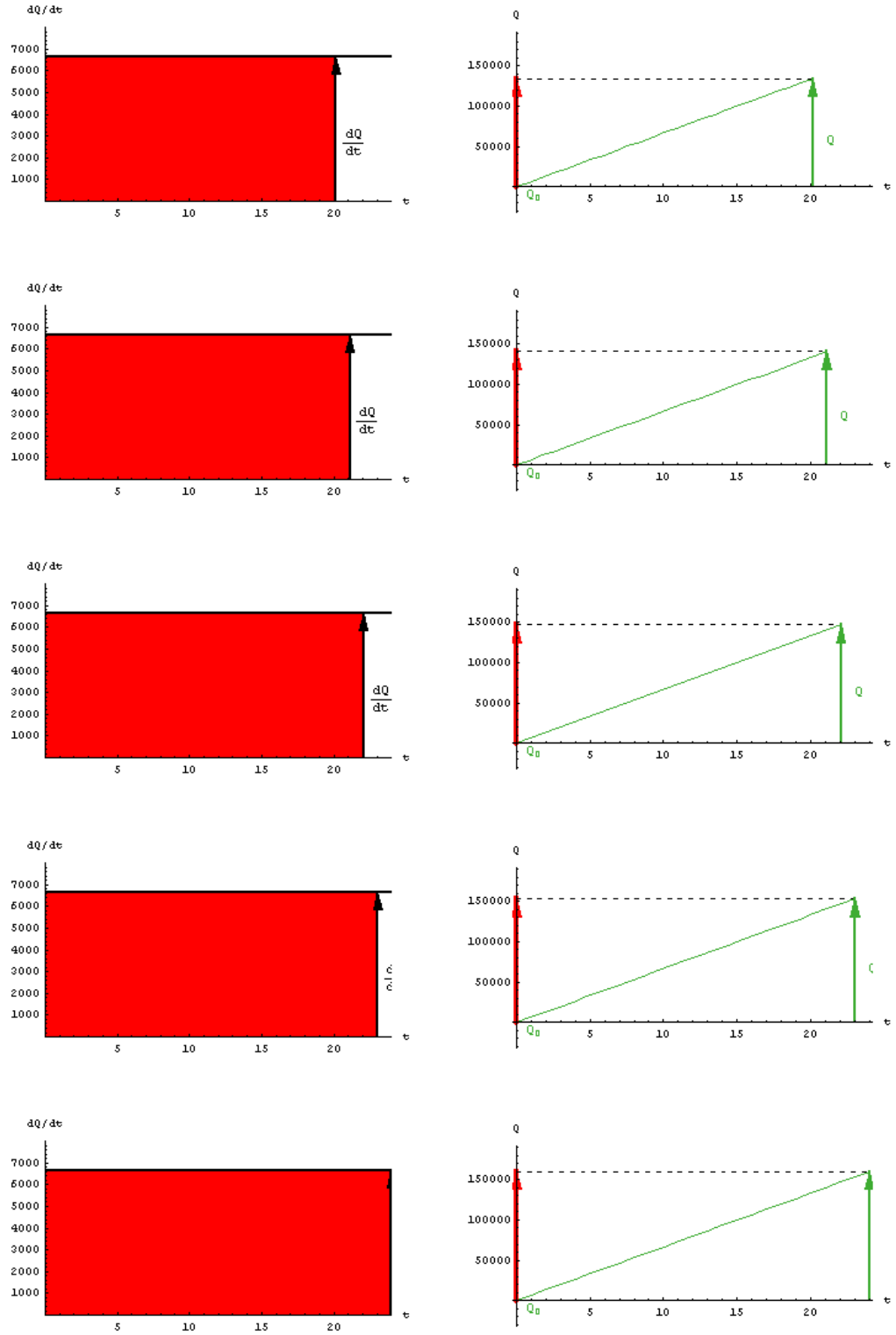



dQ/dt
7000
6000
5000
4000
3000
2000
1000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0


dQ/dt
7000
6000
5000
4000
3000
2000
1000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0


dQ/dt
7000
6000
5000
4000
3000
2000
1000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0
5
10
15
20
t


dQ/dt
7000
6000
5000
4000
3000
2000
1000
dQ/dt
5
10
15
20
t
Q
150000
100000
50000
Q
Q0
5
10
15
20
t

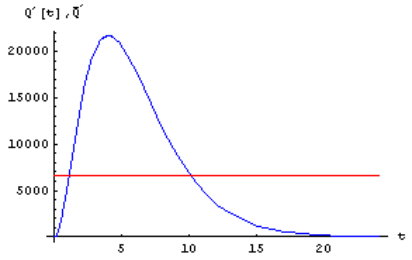


The Mean Value Theorem: Integral Form

For some additional insight, let's plot the constant average-rate-of-change function together with the actual-rate-of-change function.

In[146]:=

```
Plot[{Q'[t], Q̄'}, {t, 0, 24},
 PlotStyle → {{RGBColor[0, 0, 1]},
   {RGBColor[1, 0, 0]}},
 AxesLabel → {"t", "Q'[t],Q̄'"}];
```



The integral form of the Mean Value Theorem assures us that since $Q'[t]$is continuous on the 24-hour interval, there will be at least one time where the average rate of flow, $\bar{Q}'$, will equal the instantaneous rate of flow, $Q'[t]$. In this case, there are actually two. Do you see these two times on the preceding graph? What are these times? (We calculated them above.)

The Mean Value Theorem: Derivative Form

Let's plot the amount of water in the detention basin as functions of time for the two flow-rate functions shown in the graph above.

In[147]:=

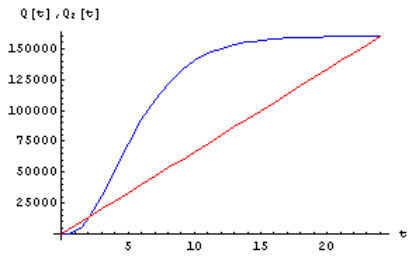
```
Q2[0] = 0;

Q2[t_] = Q̄' * t + Q2[0]
```

Out[148]=

```
6663.18 t
```

In[149]:=

```
Plot[{Q[t], Q2[t]}, {t, 0, 24},
 PlotStyle → {{RGBColor[0, 0, 1]},
   {RGBColor[1, 0, 0]}},
 AxesLabel → {"t", "Q[t],Q2[t]"}];
```



In this case, the average rate of flow over the 24-hour interval is the slope of the secant line between the to points (0, 0) and (24, 159916). The derivative form of the Mean Value Theorem tells us that since Q(t) is continuous on the interval [0, 24] and differentiable on (0, 24), there will be at least one time where the instantaneous flow rate, $Q'(t)$, will equal the average flow rate, $\frac{\Delta Q}{\Delta t}$. (Note that by the Fundamental Theorem of Calculus, Part 2

$$\frac{\Delta Q}{\Delta t} = \frac{Q(24) - Q(0)}{24 - 0} = \frac{1}{24 - 0}\int_0^{24} Q'(t)dt = \bar{Q}'.$$

This is why we call the slope of the secant line the **average rate of change**: $\frac{\Delta Q}{\Delta t} = \bar{Q}'$ truly is the average of the rate of change function over the interval.)

There are actually two points where $Q'(t) = \bar{Q}'$. Do you see them in the graph above? To highlight them further, let's draw the tangents to the curve at the two points. First, we define a new function, **L[t, a]**, that gives the equation of the line tangent to *Q(t)* at t=a; that is, it gives the linearization of *Q(t)* at the point (a, *Q(a)*).

In[150]:=

```
Clear[L, a];

L[t_, a_] = (Q[a] + Q'[a] * (t - a)) // Simplify
```

Out[151]=

$$10000\, e^{-a/2} \left(-8\, a - a^3 + 16 \left(-1 + e^{a/2}\right) + a^2 \left(-2 + t\right)\right)$$

Note that for a specific value of *a*, the **L[t, a]** command gives a linear function. For example,

In[152]:=

```
L[t, 2] // N // Expand
```

Out[152]=

```
-16582.1 + 14715.2 t
```

Recall that in a preceding calculation we found the values of *t* where the average rate of change equals the instantaneous rate of change. We repeat the calculation here.

In[153]:=

```
a = Solve[Q'[t] == Q̄', t]

InverseFunction::ifun :
 Inverse functions are being used. Values may be lost for multivalued inverses. More…

InverseFunction::ifun :
 Inverse functions are being used. Values may be lost for multivalued inverses. More…

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

Out[153]=

```
{{t → -0.687397}, {t → 1.06541}, {t → 10.0371}}
```

Now we find the equations of the tangent lines at the two points in the domain where $Q'(t) = \bar{Q}'$ and graph them together with Q(t) and $Q_2(t)$.

In[154]:=

```
tangent1 = L[t, a[[2, 1, 2]]] // Simplify
```

Out[154]=

```
-4380.74 + 6663.18 t
```

In[155]:=

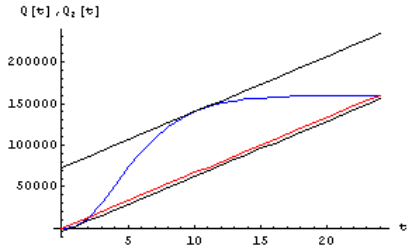
```
tangent2 = L[t, a[[3, 1, 2]]] // Simplify
```

Out[155]=

```
73425.2 + 6663.18 t
```

In[156]:=

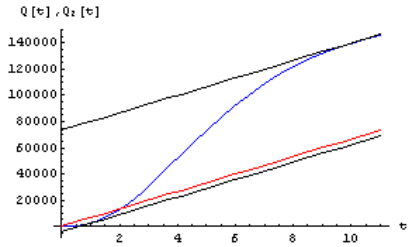
```
Plot[{Q[t], Q2[t], tangent1, tangent2},
 {t, 0, 24},
 PlotStyle → {{RGBColor[0, 0, 1]},
   {RGBColor[1, 0, 0]}, {RGBColor[0, 0, 0]},
   {RGBColor[0, 0, 0]}},
 AxesLabel → {"t", "Q[t],Q2[t]"}];
```



Let's zoom in a little.

In[157]:=

```
Plot[{Q[t], Q2[t], tangent1, tangent2},
 {t, 0, 11},
 PlotStyle → {{RGBColor[0, 0, 1]},
   {RGBColor[1, 0, 0]}, {RGBColor[0, 0, 0]},
   {RGBColor[0, 0, 0]}},
 AxesLabel → {"t", "Q[t],Q2[t]"}];
```



We see that the tangents at the two points are parallel to the secant line. That is, $Q'(t)=\frac{\Delta Q}{\Delta t}=\bar{Q}'$ at these two points.

▪ Let's Drain the Dam Thing

Now that we have a full detention basin, we need to figure out how to drain it. The design includes a pump that is capable of pumping 5000 $\frac{ft^3}{hr}$ out of the basin. This rate of outflow is acceptable and won't flood our neighbors. The pump doesn't start until the amount of water in the reservoir reaches 50,000 cubic feet, and then it runs until the reservoir is empty. Let's find a function to describe how the water accumulates in the basin under these conditions. First, we determine when the pump will kick on.

In[158]:=

```
ton = Solve[Q[t] == 50000, t]
 (*This command won't work,
  but don't give up!*)
```

```
Solve::tdep : The equations appear to involve the
   variables to be solved for in an essentially non-algebraic way. More…
```

Out[158]=

```
Solve[10000 (16 - 2 e^(-t/2) (8 + 4 t + t^2)) == 50000, t]
```

The **Solve[ ]** command doesn't work for this function, so we resort to the **FindRoot[ ]** command. From the graph of Q(t), it appears that the water will reach 50000 cubic feet at about 4 hours, so we use this as a starting value in the **FindRoot[ ]** command.

In[159]:=

```
ton = FindRoot[Q[t] == 50000, {t, 4}]
```

Out[159]=

```
{t → 3.92001}
```

"> About *Mathematica*

Now we can form the rate-of-fill function and graph it. First, we define the rate of inflow and the rate of outflow and graph each of them.

In[160]:=

```
inflowrate[t_] = 10000 t^2 Exp[-t / 2]
```

Out[160]=

```
10000 e^(-t/2) t^2
```

The outflow rate function is piecewise defined since its value is 0 for $t < 3.92$ and 5000 for $t \geq 3.92$. In *Mathematica*, the **UnitStep[ ]** command works well for constructing piecewise defined functions because *Mathematica* integrates the **UnitStep[ ]** function correctly. The **UnitStep[t]** function has a value of 0 for $t < 0$ and 1 for $t \geq 0$. See how it works to define the outflow rate function in the next cell.

In[161]:=

```
outflowrate[t_] =
 5000 * UnitStep[t - ton[[1, 2]]]
```

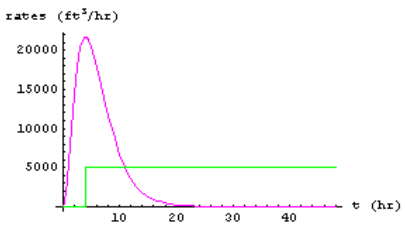
Out[161]=

```
5000 UnitStep[-3.92001 + t]
```

"> About *Mathematica*

In[162]:=

```
Plot[{inflowrate[t], outflowrate[t]},
 {t, 0, 48},
 AxesLabel -> {"t (hr)", "rates (ft^3/hr)"},
 PlotRange → All,
 PlotStyle → {{RGBColor[1, 0, 1]},
   {RGBColor[0, 1, 0]}}];
```



The rate at which water accumulates is represented as the difference between the two curves, that is, (**inflowrate[t] - outflowrate[t]**). The accumulated water can be represented as the signed area between the two graphs. To show this, we use the **FilledPlot[ ]** command from the special package, **Graphics`FilledPlot`**, that we loaded above.

First, let's figure out when the reservoir stops filling and begins to drain. This will occur when the outflow rate of 5000 ft^3/hr exceeds the inflow rate. In the graph above, it is the point where the two curves cross near t=12 hours.

In[163]:=

```
FindRoot[Q'[t] == 5000, {t, 12}]
```

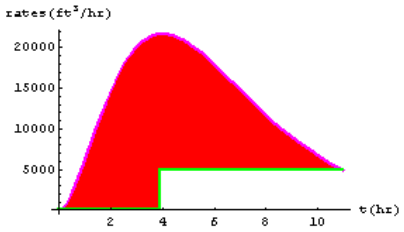
Out[163]=

```
{t → 10.9652}
```

Let's plot the curves with the net inflow area colored red and the net outflow area colored blue.

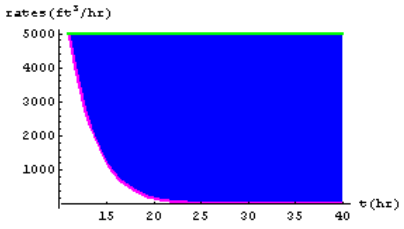
In[164]:=

```
p1 = FilledPlot[
  {inflowrate[t], outflowrate[t]},
  {t, 0, 10.9652}, PlotRange → All,
  AxesLabel -> {"t(hr)", "rates(ft^3/hr)"},
  PlotStyle →
   {{Thickness[0.010], RGBColor[1, 0, 1]},
    {Thickness[0.010], RGBColor[0, 1, 0]}},
  Fills → {RGBColor[1, 0, 0]}];
```



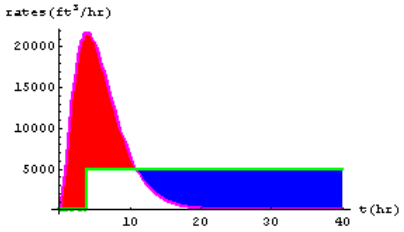
In[165]:=

```
p2 = FilledPlot[
  {inflowrate[t], outflowrate[t]},
  {t, 10.9652, 40},
  AxesLabel -> {"t(hr)", "rates(ft^3/hr)"},
  PlotStyle →
   {{Thickness[0.010], RGBColor[1, 0, 1]},
    {Thickness[0.010], RGBColor[0, 1, 0]}},
  Fills → {RGBColor[0, 0, 1]}];
```



In[166]:=

```
Show[p1, p2];
```



The red area in the preceding graph represents the amount of water that has accumulated in the reservoir, and the blue area represents the net amount that has drained out of the reservoir. From the preceding graph, can you estimate when the basin will stop filling, when it will start draining, and when it will be empty?

Let's form a function to describe the rate at which water accumulates in the basin and graph it.

In[167]:=

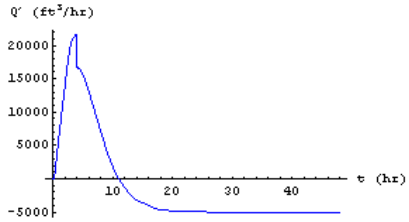
```
Clear[Q];

Q'[t_] = inflowrate[t] - outflowrate[t]
```

Out[168]=

$$10000\, e^{-t/2} t^2 - 5000\, \text{UnitStep}[-3.92001 + t]$$

In[169]:=

```
Plot[Q'[t], {t, 0, 48},
 AxesLabel -> {"t (hr)", "Q' (ft³/hr)"},
 PlotStyle → {RGBColor[0, 0, 1]}];
```



Can you tell any better from the preceding graph when the basin will stop filling, when it will start draining, and when it will be empty?

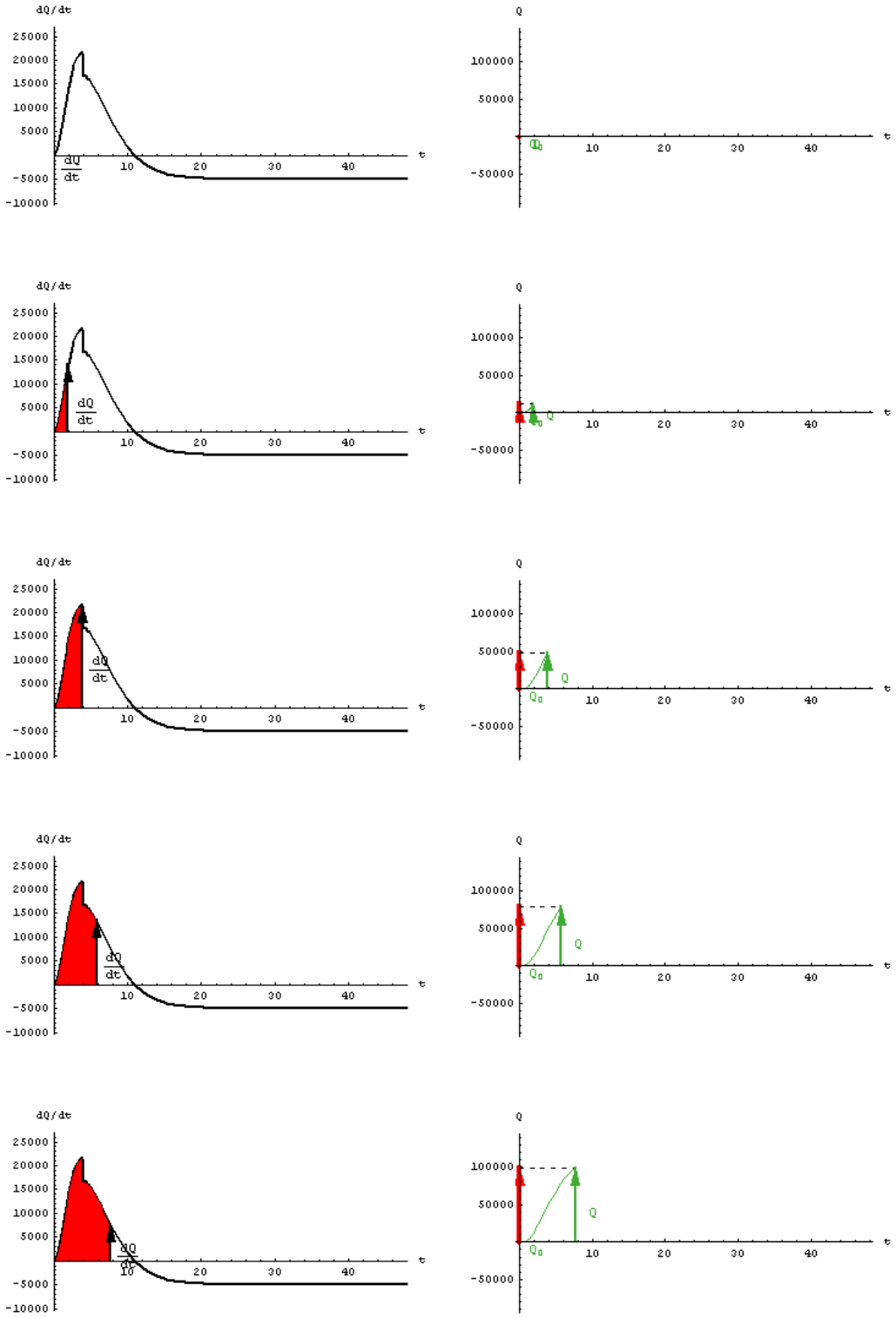
You can use the **antidifferentate[ ]** command to watch the basin fill and drain. Animate the graphics if you wish to do so.

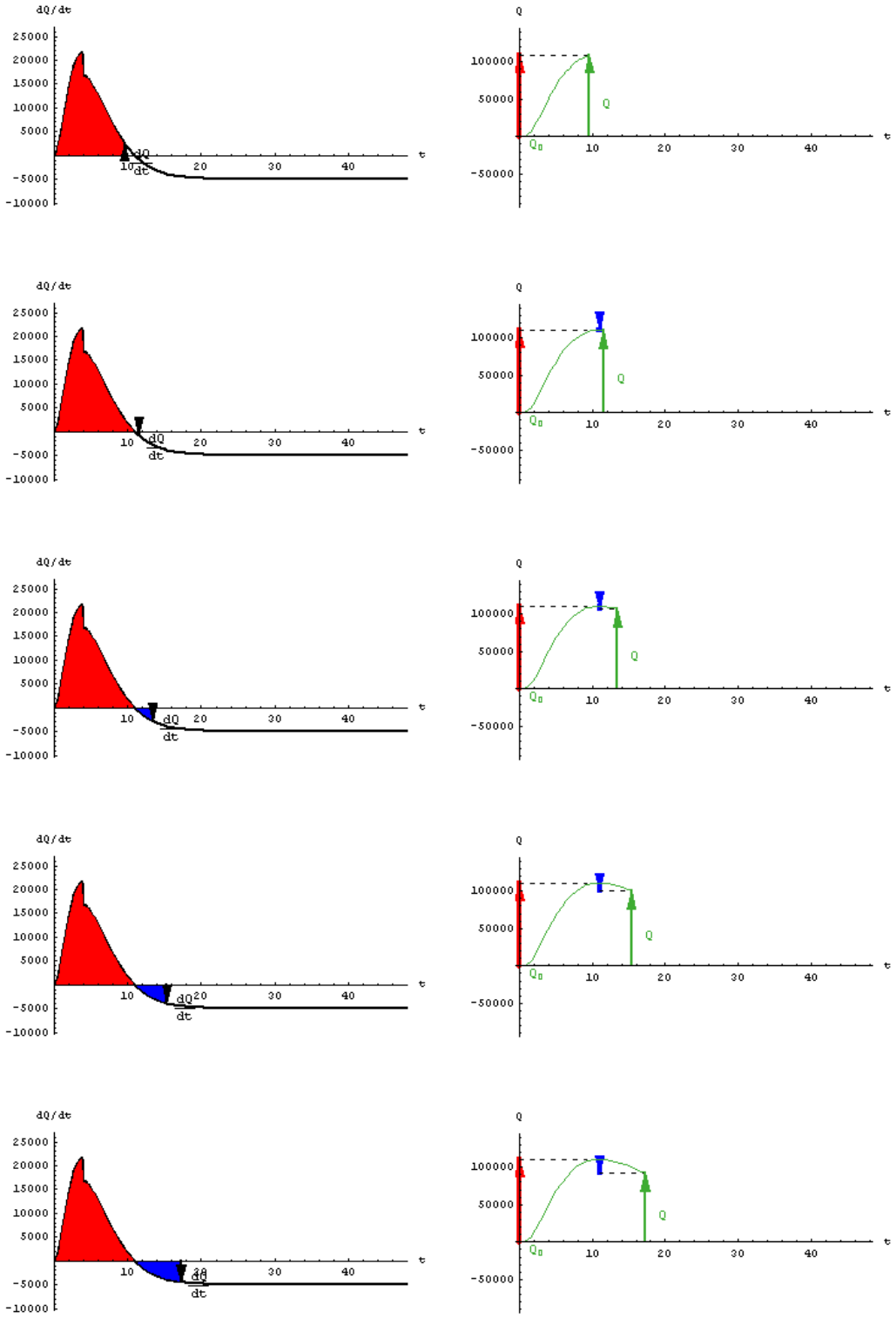
Note: The **antidifferentiate[ ]** command generates a lot of graphics, which may fill up your computer's memory. Before executing the next command, pull down the Kernel menu and select Delete All Output. All computed values are saved when you do this.

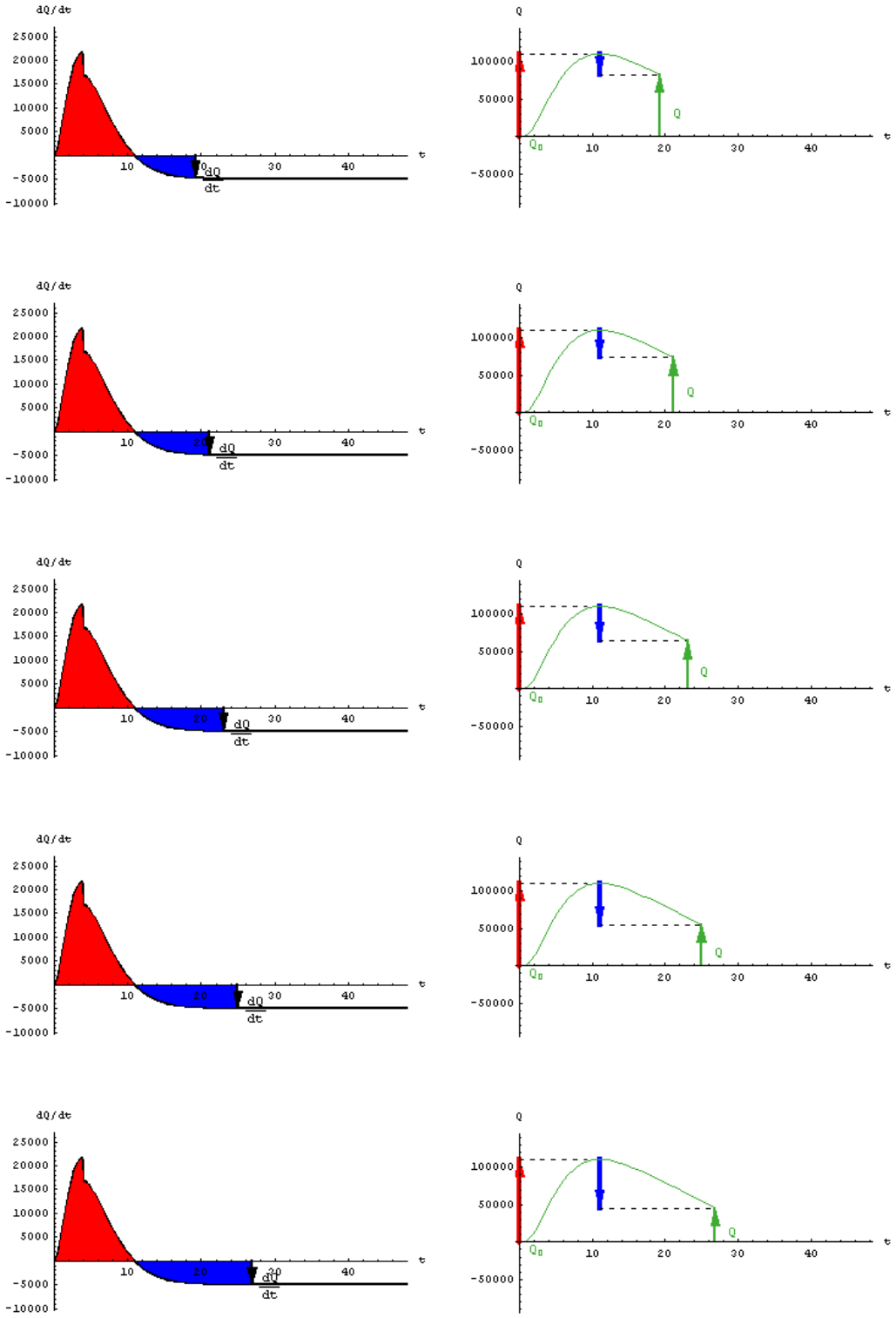
Since the next command generates a lot of output, you probably won't want to print this notebook until after you have deleted most or all of the graphs. We recommend deleting all but a few representative cells from the sequence before you print.

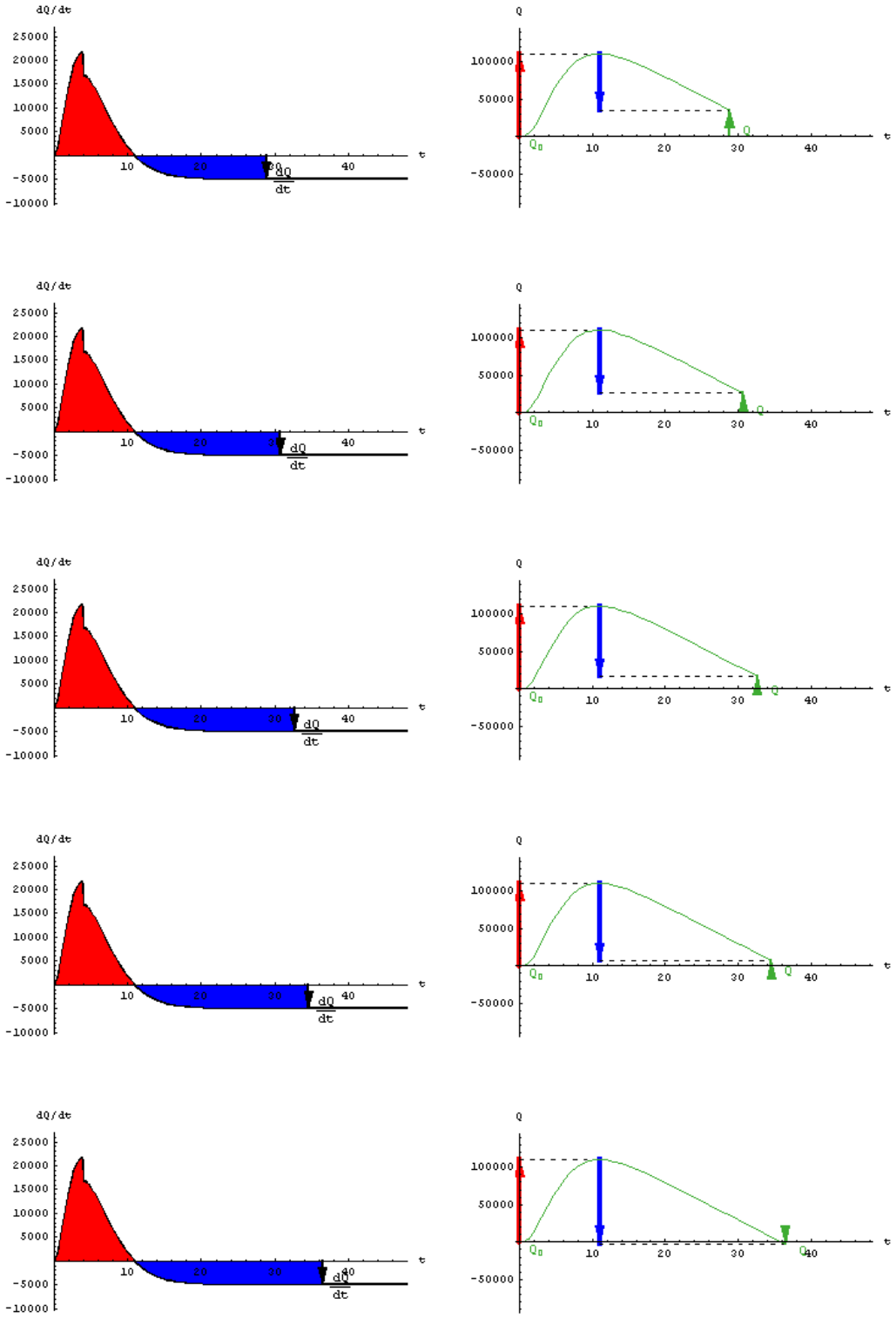
In[170]:=

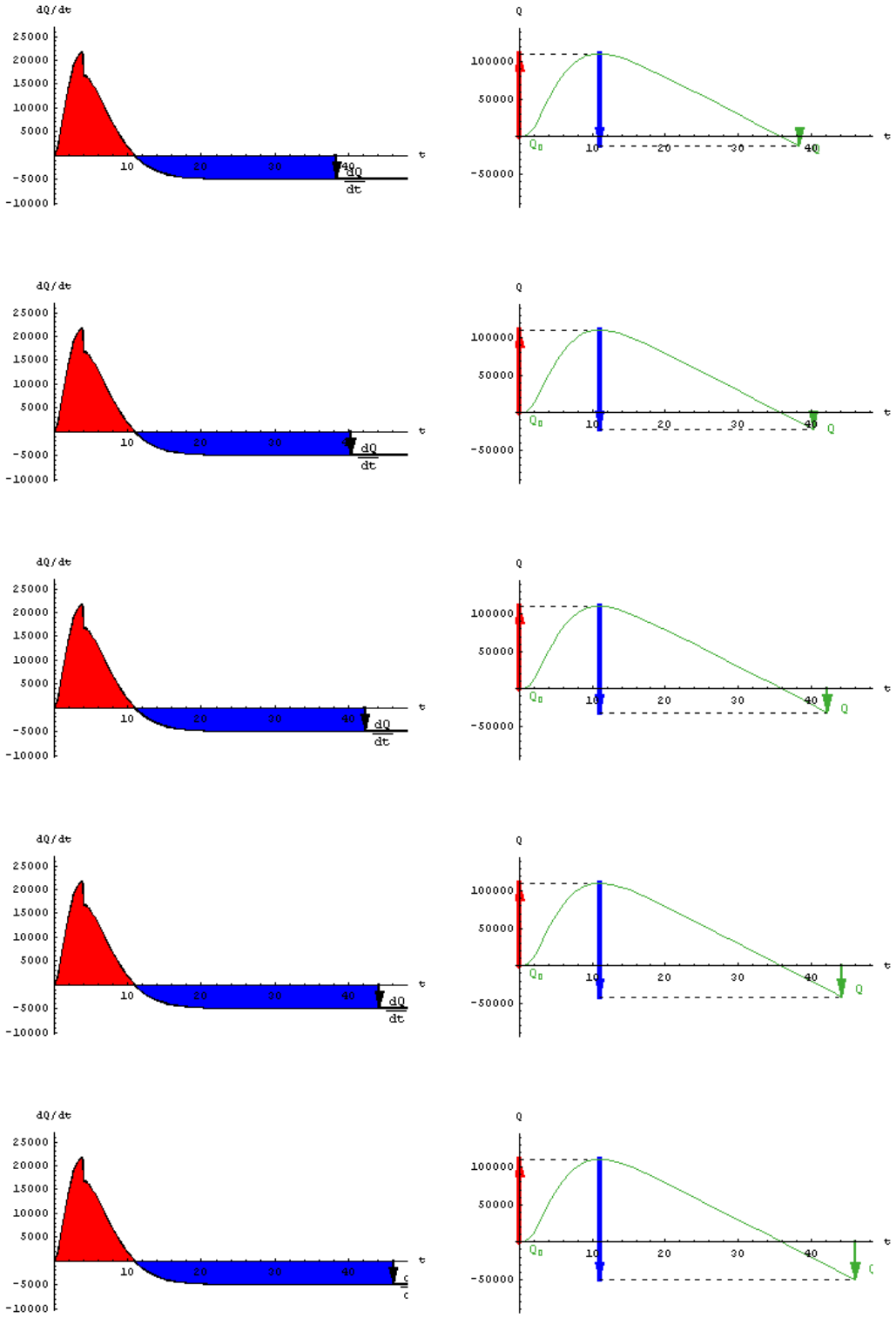
```
antidifferentiate[Q'[t], t, 0, 48, 0]
```

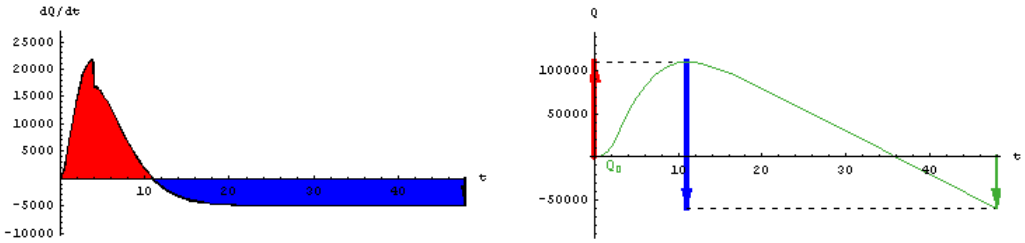

dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t


dQ/dt
25000
20000
15000
10000
5000
-5000
-10000
10
20
30
40
t
dQ
dt
Q
100000
50000
-50000
Q0


dQ/dt
25000
20000
15000
10000
5000
-5000
-10000
10
20
30
40
t
dQ/dt
Q
100000
50000
-50000
Q
Q0
t


dQ/dt
25000
20000
15000
10000
5000
-5000
-10000
10
20
30
40
t
dQ/dt
Q
100000
50000
-50000
Q0
Q


dQ/dt
25000
20000
15000
10000
5000
-5000
-10000
10
20
30
40
t
dQ
dt
Q
100000
50000
-50000
Q0



It appears that the reservoir will be fully drained at about 36 hours after the storm begins or 12 hours after the storm ends. This occurs when the graph of **Q[t]** crosses the *t* axis and when the blue area is equal to the red area on the graph of **Q′[t]**. Let's try it. Change the ending time to 36 hours in the command above, and execute it again.

How much water does the detention basin have to hold if we include the pump in the design?

You Try It: Calculus Has its Ups and Downs on Elevators

The following function describes the velocity of an elevator during a 12-second trip from the ground floor to the eighth floor, which is the top floor of the building. In this case, **Q[t]** represents the height of the elevator above the ground floor and **Q′[t]** the velocity schedule over the 12-second time interval. At first, we go through the steps with you (except step 10), and then we ask you to do one on your own.

First, we define a function that works well in *Mathematica* for piecewise defined functions because *Mathematica* integrates it correctly. It uses the **UnitStep[t]** function, which has a value of 0 for $t < 0$ and 1 for $t \geq 0$. The function **onoff[t, a, b]**, defined in the next cell, is equal to 1 for $a \leq t < b$, and is 0 otherwise. You will see how it works to define a piecewise function for the velocity of the elevator.

In[171]:=

```
Clear[a, b, t];

onoff[t_, a_, b_] =
 UnitStep[t - a] - UnitStep[t - b]
```

Out[172]=

```
UnitStep[-a + t] - UnitStep[-b + t]
```

In[173]:=

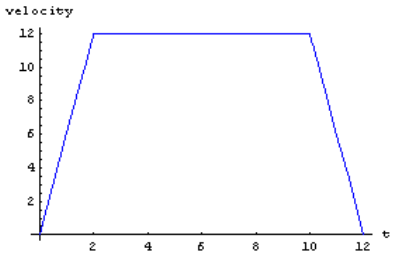
```
Clear[Q];

Q'[t_] = 6 * t * onoff[t, 0, 2] +
   12 * onoff[t, 2, 10] +
   (72 - 6 * t) * onoff[t, 10, 12]
```

Out[174]=

```
(72 - 6 t) (-UnitStep[-12 + t] + UnitStep[-10 + t]) +
 12 (-UnitStep[-10 + t] + UnitStep[-2 + t]) + 6 t (-UnitStep[-2 + t] + UnitStep[t])
```

In[175]:=

```
Plot[Q'[t], {t, 0, 12},
 PlotStyle → {RGBColor[0, 0, 1]},
 PlotRange → All,
 AxesLabel → {"t", "velocity"}];
```



Use the function **Q′[t]** to perform the following tasks.

1. Form a **Q[t_]** function that represents the height of the elevator above the ground floor, and graph the function.

In[176]:=

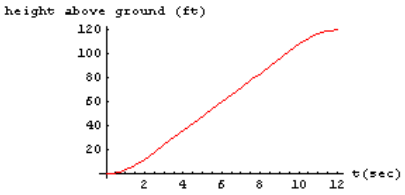
```
Q[t_] = ∫_0^t Q′[u] du
```

Out[176]=

```
3 ((-12 + t)^2 UnitStep[-12 + t] -
    (-10 + t)^2 UnitStep[-10 + t] - (-2 + t)^2 UnitStep[-2 + t] + t^2 UnitStep[t])
```

In[177]:=

```
Plot[Q[t], {t, 0, 12},
 PlotStyle → {RGBColor[1, 0, 0]},
 AxesLabel →
  {"t(sec)", "height above ground (ft)"}];
```



2. If each story of the building is the same height, determine the height of the building. (Be careful. The elevator stops at the bottom of the top floor.)

In[178]:=

```
heightofeachfloor = Q[12] / 7 // N
```

Out[178]=

```
17.1429
```

In[179]:=

```
heightofbuilding = 8 * heightofeachfloor
```

Out[179]=

```
137.143
```

3. Determine the average velocity of the elevator during the 12-second trip.

In[180]:=

$$\text{vavg} = \frac{1}{12 - 0} \int_0^{12} Q'[t]\, dt$$

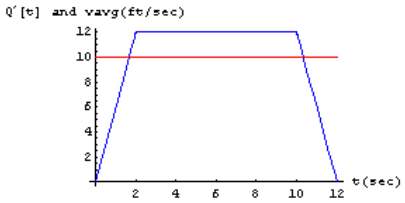
Out[180]=

```
10
```

4. Plot **Q'[t]** and the constant average-velocity function on the same graph.

In[181]:=

```
Plot[ {Q'[t], vavg}, {t, 0, 12},
 PlotRange → All,
 AxesLabel ->
  {"t(sec)", "Q'[t] and vavg(ft/sec)"},
 PlotStyle -> {{RGBColor[0, 0, 1]},
   {RGBColor[1, 0, 0]}}];
```



5. Determine the times when the instantaneous velocity is equal to the average velocity over the 12-second interval. (**Solve[ ]** won't work and **FindRoot[ ]** won't work with only one starting value. It needs two starting values.)

In[182]:=

```
FindRoot[Q'[t] == 10, {t, 1., 1.5}]
```

Out[182]=

```
{t → 1.66667}
```

In[183]:=

```
FindRoot[Q'[t] == 10, {t, 10.5, 11.5}]
```

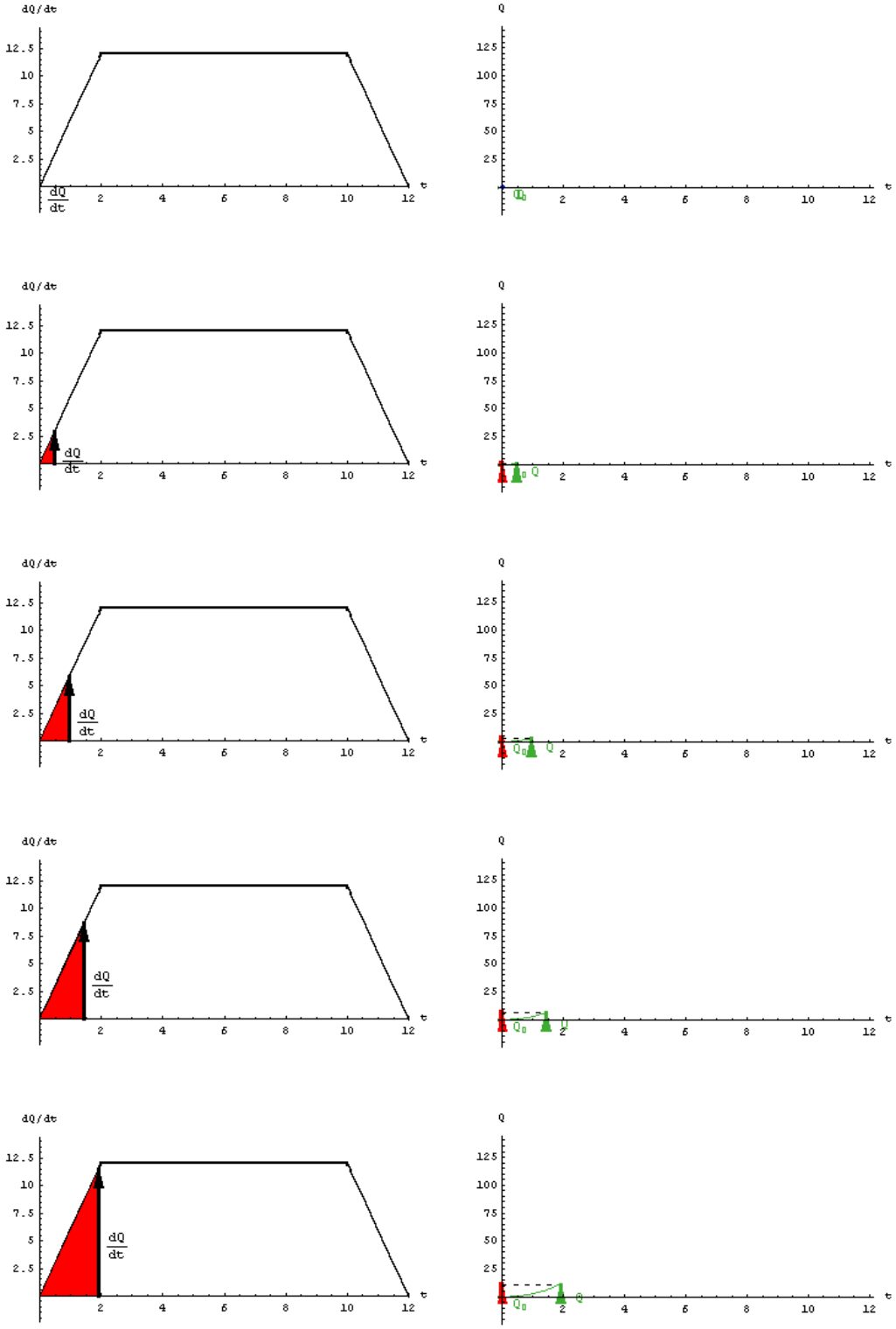
Out[183]=

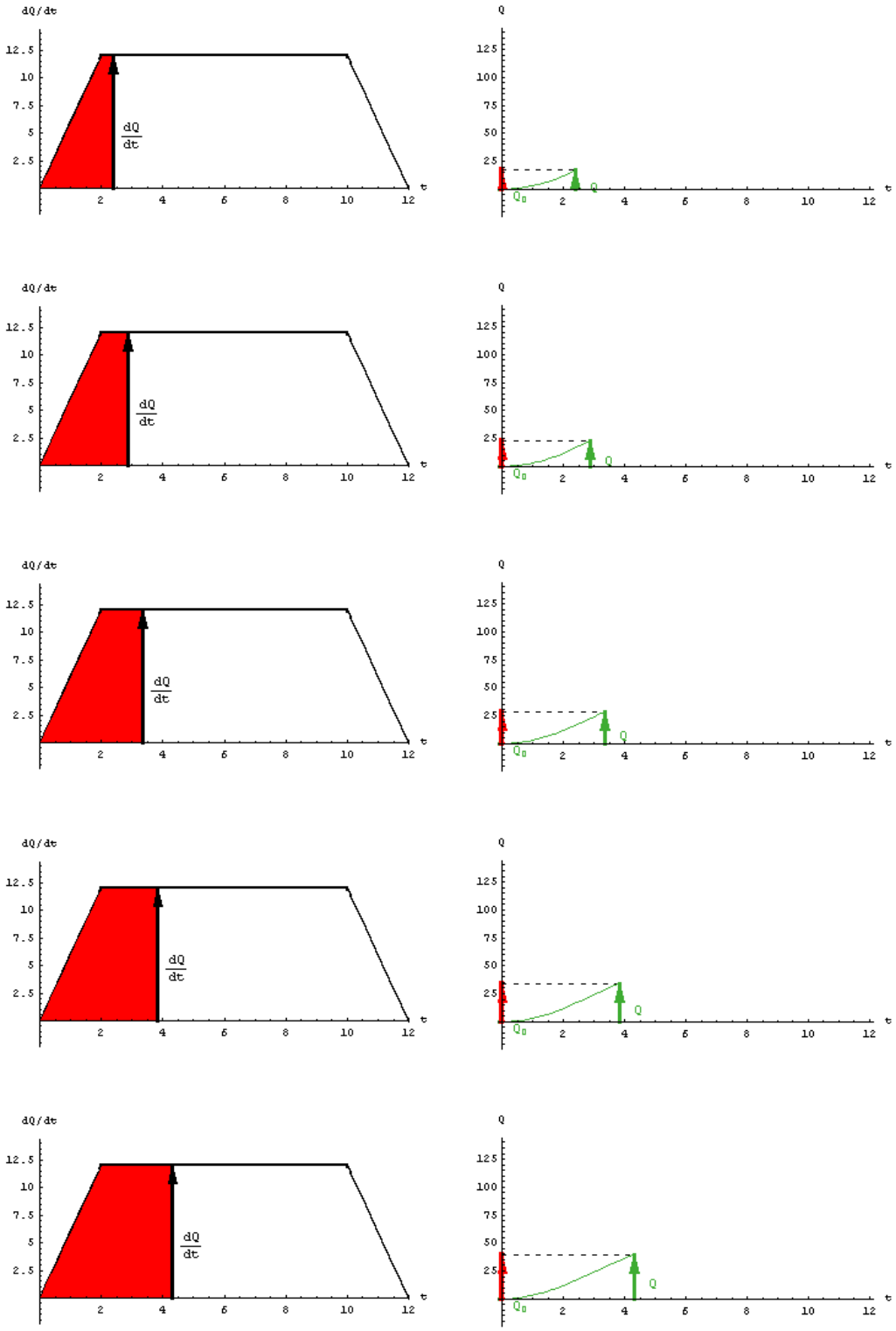
```
{t → 10.3333}
```

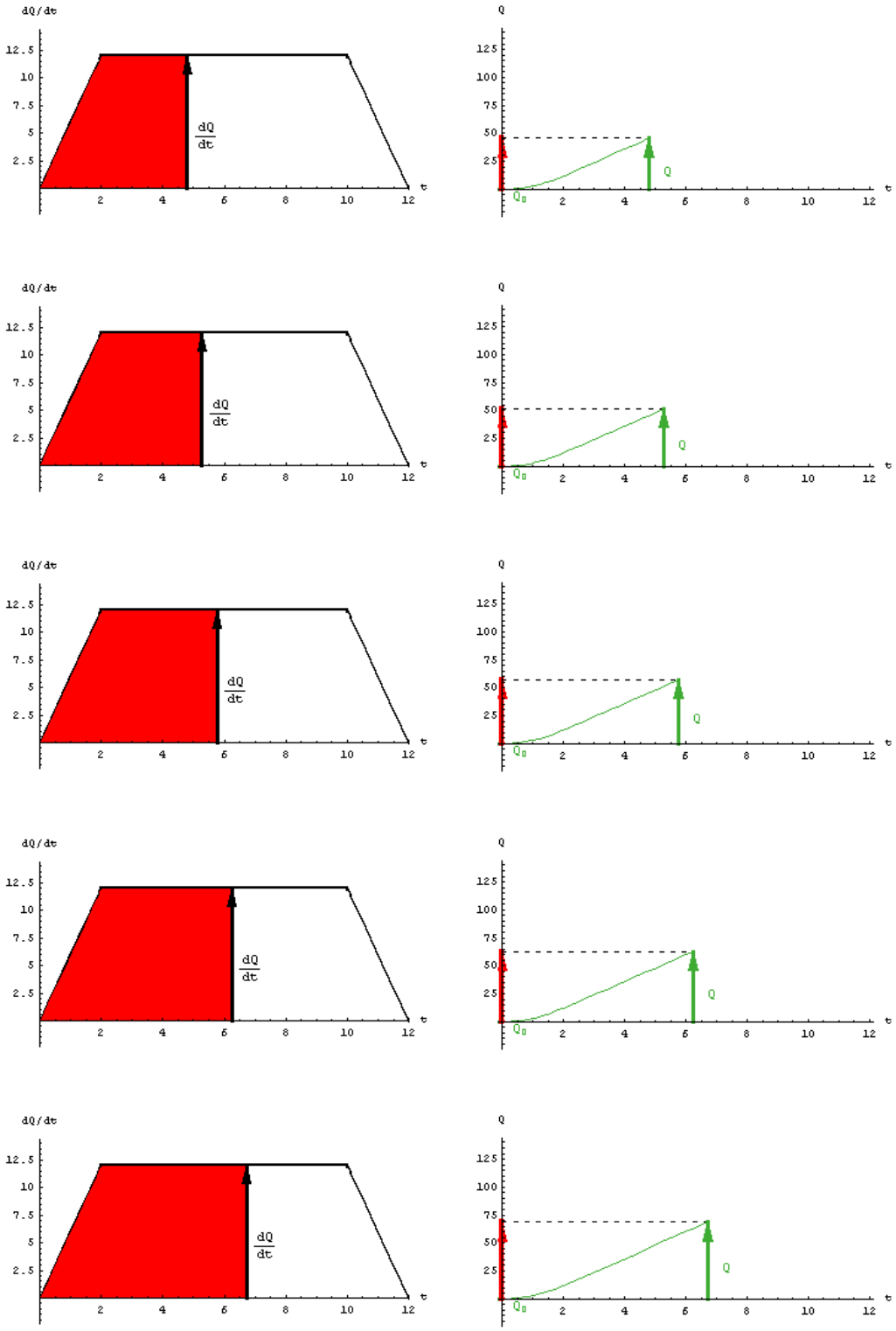
6. Use the special function **antidifferentiate[ ]** to depict the motion of the elevator as it travels to the top floor. (The **antidifferentiate[ ]** command generates a lot of graphics, which tend to fill up computer memory. Pull down the Kernel menu and select Delete All Output before executing the next command. All computed values are saved when you do this.)

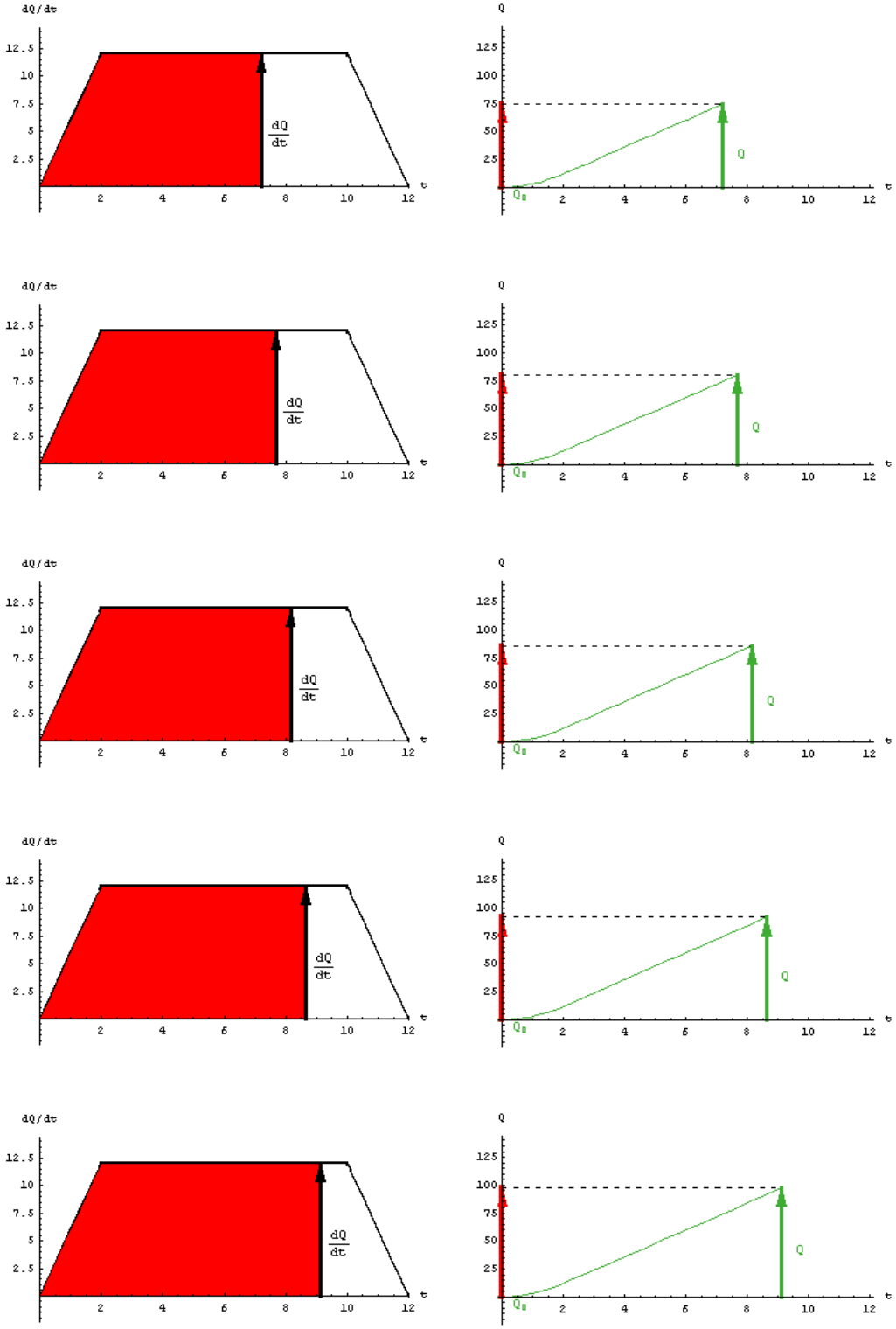
In[184]:=

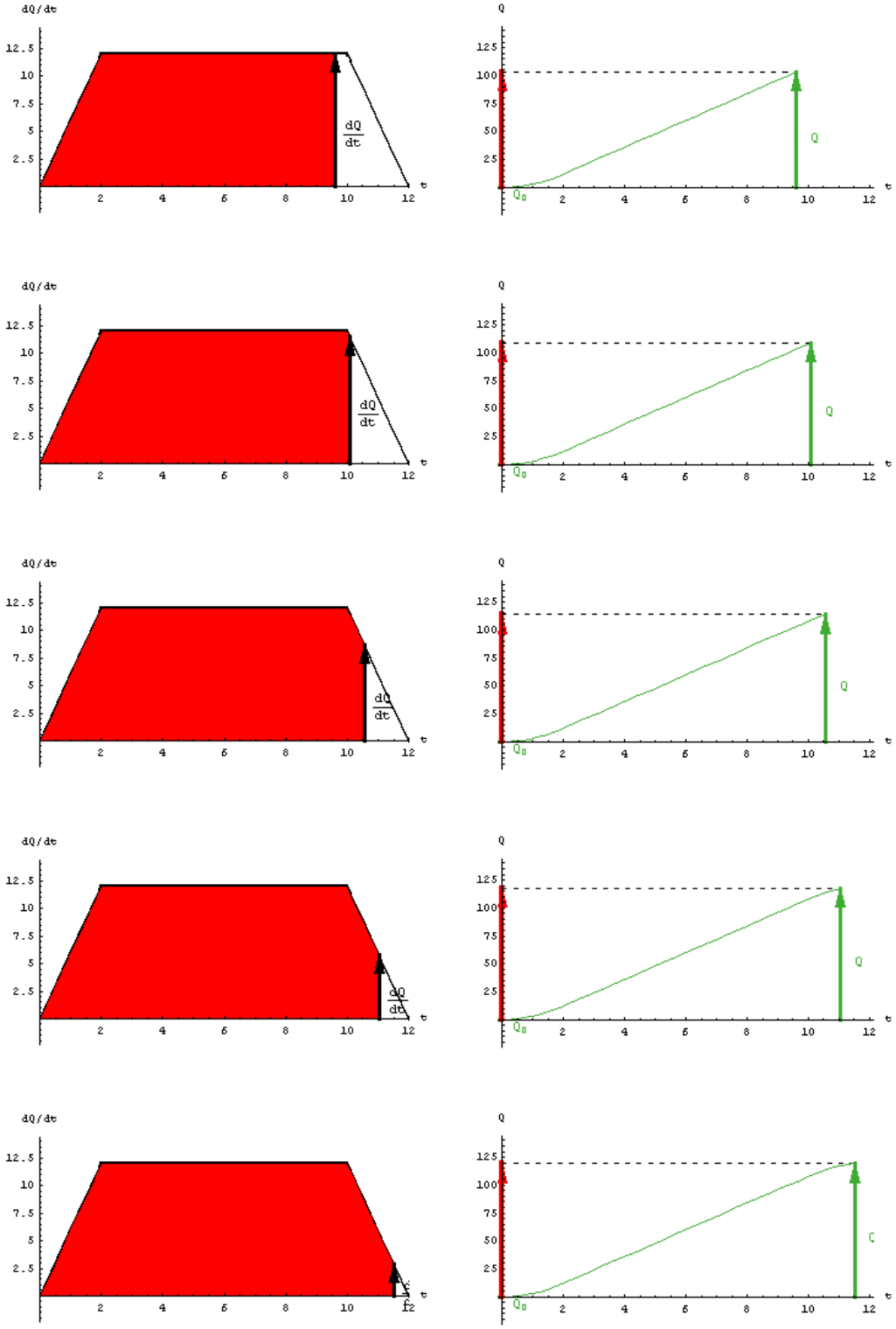
```
antidifferentiate[Q'[t], t, 0, 12, 0]
```

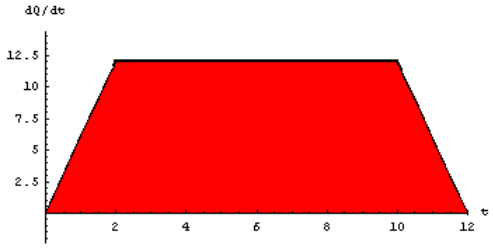

dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t

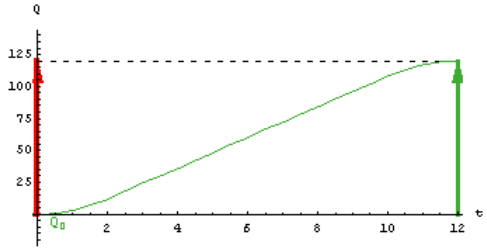

dQ/dt
dQ
dt
t
Q
Q0


dQ/dt
12.5
10
7.5
5
2.5
dQ
dt
t
2
4
6
8
10
12
Q
125
100
75
50
25
Q
Q0


dQ/dt
12.5
10
7.5
5
2.5
dQ
dt
t
2
4
6
8
10
12
Q
125
100
75
50
25
Q
Q0


dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t





7. Plot a graph of the height of the elevator if it were to travel to the eighth floor at a constant speed that is equal to its average speed during the actual trip, and then superimpose this graph on the actual graph of **Q[t]**.

In[185]:=

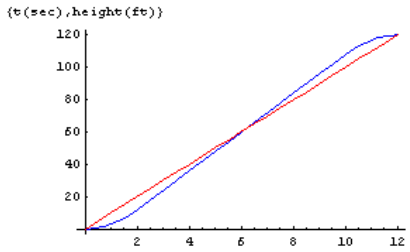
```
Qlin[t_] = vavg * t
```

Out[185]=

```
10 t
```

In[186]:=

```
p1 = Plot[{Q[t], Qlin[t]}, {t, 0, 12},
  PlotStyle -> {RGBColor[0, 0, 1],
    RGBColor[1, 0, 0]},
  AxesLabel ->
   {"t(sec)\!\(\"\,\"\)height(ft)"}];
```



8. Find the linearizations of **Q[t]** at the two times when the instantaneous velocity is equal to the average velocity, and superimpose the graphs of the two tangent lines on the graph found in step 7 above. (They should be parallel to the secant line between the two points on the graph of **Q[t]** at the beginning and end of the trip.)

In[187]:=

```
L[t_, a_] := Q[a] + Q'[a] * (t - a);
```

In[188]:=

```
tangent1 = L[t, 1.66667] // Expand
```

Out[188]=

```
-8.33337 + 10. t
```

In[189]:=

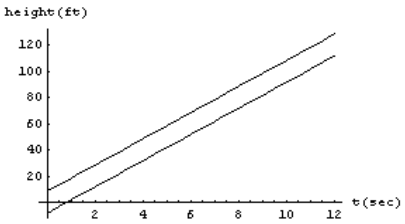
```
tangent2 = L[t, 10.33333] // Expand
```

Out[189]=

```
8.33313 + 10. t
```

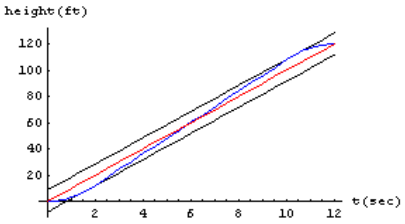
In[190]:=

```
p2 = Plot[{tangent1, tangent2}, {t, 0, 12},
  AxesLabel → {"t(sec)", "height(ft)"}];
```



In[191]:=

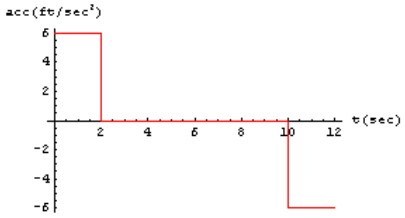
```
Show[p2, p1];
```



9. Plot the acceleration of the elevator during the trip, and describe the sensations you would feel if you were riding in the elevator as it travels from the ground floor to the top floor of the building.

In[192]:=

```
Plot[Q''[t], {t, 0, 12},
 AxesLabel → {"t(sec)", "acc(ft/sec^2)"},
 PlotStyle → {RGBColor[1, 0, 0]}];
```



10. Assuming that the elevator always accelerates and decelerates at 6 $\frac{\text{ft}}{\text{sec}^2}$ and cruises at 12 $\frac{\text{ft}}{\text{sec}}$, determine how long it would take the elevator to start at one floor, travel to the next floor, and stop there. Hint: The elevator does not reach cruising speed. It accelerates at 6 ft/sec^2 for the first half of the trip and then decelerates at -6 ft/sec^2 during the second half of the trip. Therefore, the velocity versus time function will be an isosceles triangle, and the area of this triangle is the height of one floor. From this you can figure out the duration of the trip. Answer: It takes 3.38 seconds to start at one floor and stop at the next.

11. Make up your own velocity schedule for the elevator and repeat steps 1 through 9. Here is one we made up where the elevator travels to the top floor and waits there for a three seconds, and then, after everybody gets off, the cable breaks.

In[193]:=

```
Clear[Q];
```

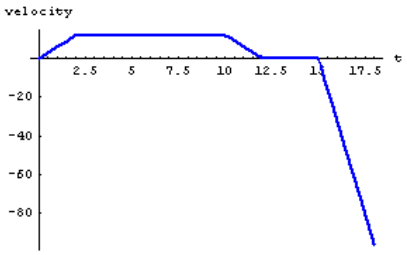
```
Q'[t_] = 6 * t * onoff[t, 0, 2] +
   12 * onoff[t, 2, 10] +
   (72 - 6 * t) * onoff[t, 10, 12] -
   32 * (t - 15) * UnitStep[t - 15]
```

Out[194]=

```
-32 (-15 + t) UnitStep[-15 + t] + (72 - 6 t) (-UnitStep[-12 + t] + UnitStep[-10 + t]) +
 12 (-UnitStep[-10 + t] + UnitStep[-2 + t]) + 6 t (-UnitStep[-2 + t] + UnitStep[t])
```

In[195]:=

```
Plot[Q'[t], {t, 0, 18},
 PlotStyle → {RGBColor[0, 0, 1],
   Thickness[0.010]}, PlotRange → All,
 AxesLabel → {"t", "velocity"}];
```



You Try It: Launching a Model Rocket

Chapter 5, Practice Exercise 1 Extension

When a model rocket is launched, the propellant burns for a few seconds, accelerating the rocket upward. The acceleration increases during this period because the mass of the rocket decreases as it burns fuel while the upward thrust of the engine and the downward pull of gravity remain essentially constant. After burnout, the rocket coasts upward for a while and then begins to fall, all the while accelerating downward at a rate equal to g (i.e., 32 $\frac{ft}{sec^2}$). A small explosive charge pops out a parachute shortly after the rocket starts down. The parachute slows the rocket to keep it from breaking when it lands. The following function describes the rate of change of velocity of the rocket (i.e., its acceleration) during the flight. In this case, **Q[t]** represents the velocity of the rocket, and **Q'[t]** represents its acceleration.

In[196]:=

```
onoff[t_, timeon_, timeoff_] =
 UnitStep[t - timeon] - UnitStep[t - timeoff]
```

Out[196]=

```
-UnitStep[t - timeoff] + UnitStep[t - timeon]
```

In[197]:=

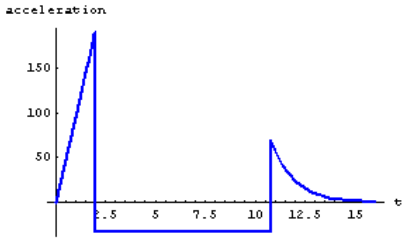
```
Clear[Q];

Q'[t_] = (95 * t) * onoff[t, 0, 2] -
   32 * onoff[t, 2, 10.8] +
   69.3 e^(-0.8664 (-10.8+t)) * UnitStep[t - 10.8]
```

Out[198]=

```
69.3 e^(-0.8664 (-10.8+t)) UnitStep[-10.8 + t] -
 32 (-UnitStep[-10.8 + t] + UnitStep[-2 + t]) + 95 t (-UnitStep[-2 + t] + UnitStep[t])
```

In[199]:=

```
Plot[Q'[t], {t, 0, 16},
 PlotStyle → {Thickness[0.010],
   RGBColor[0, 0, 1]},
 AxesLabel → {"t", "acceleration"}];
```



Use the function Q'[t] to perform the following tasks.

1. Determine the average acceleration between $t = 0$ and $t = 15$ seconds, and superimpose a graph of the constant average-acceleration function onto the graph of the actual-acceleration function.

2. Determine the times between $t = 0$ and $t = 15$ seconds, if any, where the instantaneous acceleration is equal to the average acceleration over that same time interval. If there are no times where the instantaneous acceleration is equal to the average acceleration, explain why this does not contradict the Mean Value Theorem.

3. Given that the initial velocity of the rocket is 0, form a function that gives the velocity of the rocket as a function of time, and graph it. Use the graph to estimate when the rocket reaches its maximum height, and when it lands. You may have to adjust the domain over which you plot **Q[t]** to answer the last question.

In[200]:=

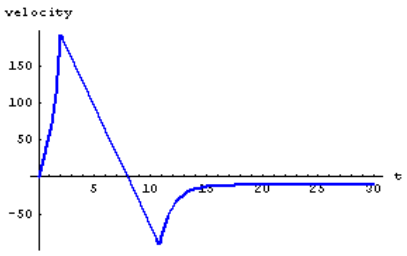
```
Q[t_] = ∫_0^t Q'[u] du
```

Out[200]=

```
e^(-0.8664 t) ((-926319. + e^(0.8664 t) (-265.614 + 32. t)) UnitStep[-10.8 + t] +
   e^(0.8664 t) (-47.5 (-2. + t) (2.67368 + t) UnitStep[-2. + t] + 47.5 t^2 UnitStep[t]))
```

In[201]:=

```
Plot[Q[t], {t, 0, 30}, PlotRange → All,
 PlotStyle → {Thickness[0.010],
   RGBColor[0, 0, 1]},
 AxesLabel → {"t", "velocity"}];
```



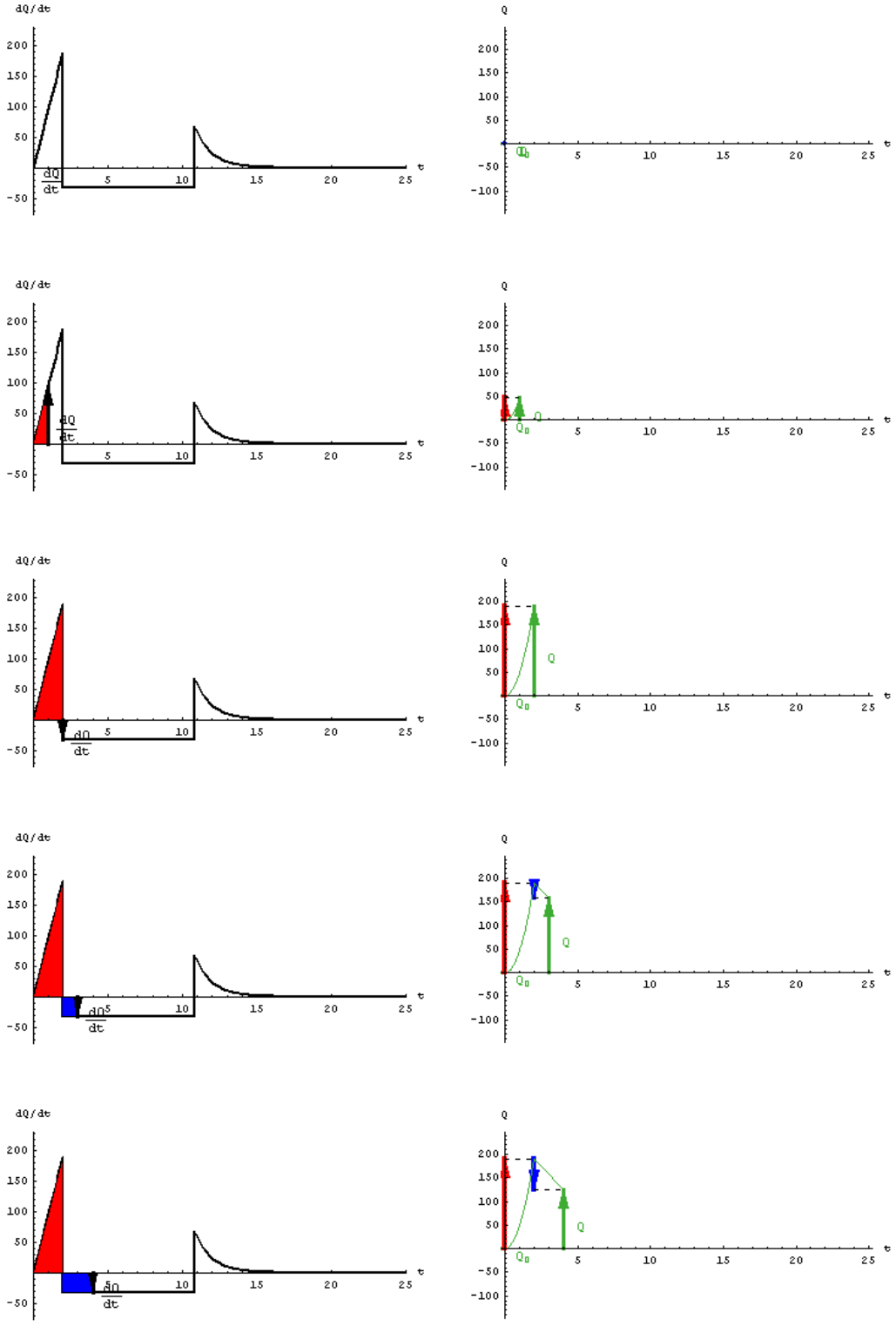
4. Find the maximum velocity and the minimum velocity of the rocket during the trip.

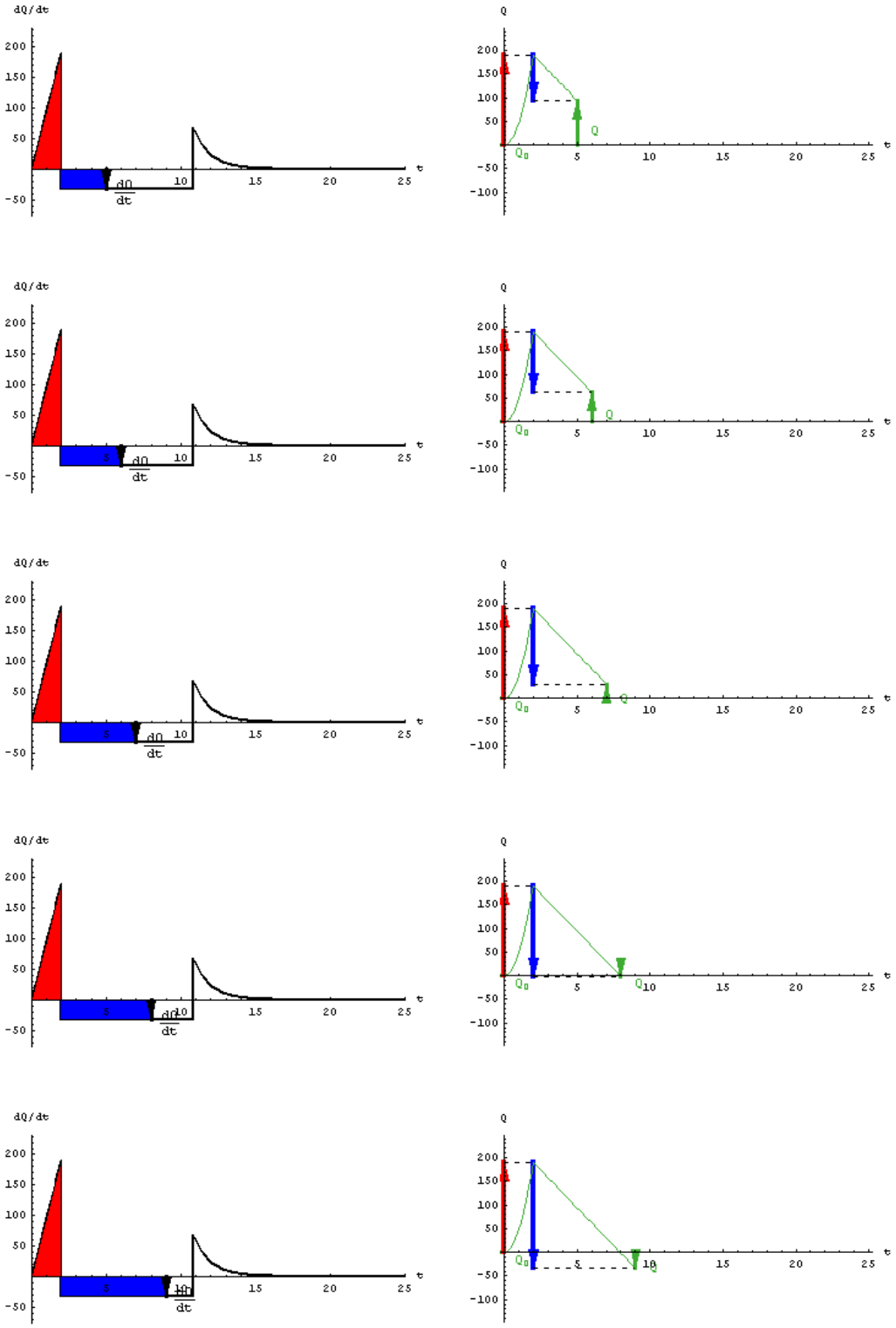
5. Use the **antidifferentiate[ ]** function to depict the accumulation of velocity between $t = 0$ and $t = 25$ seconds. (The **antidifferentiate[ ]** command generates a lot of graphics, which tend to fill up computer memory. Pull down the Kernel menu and select Delete All Output

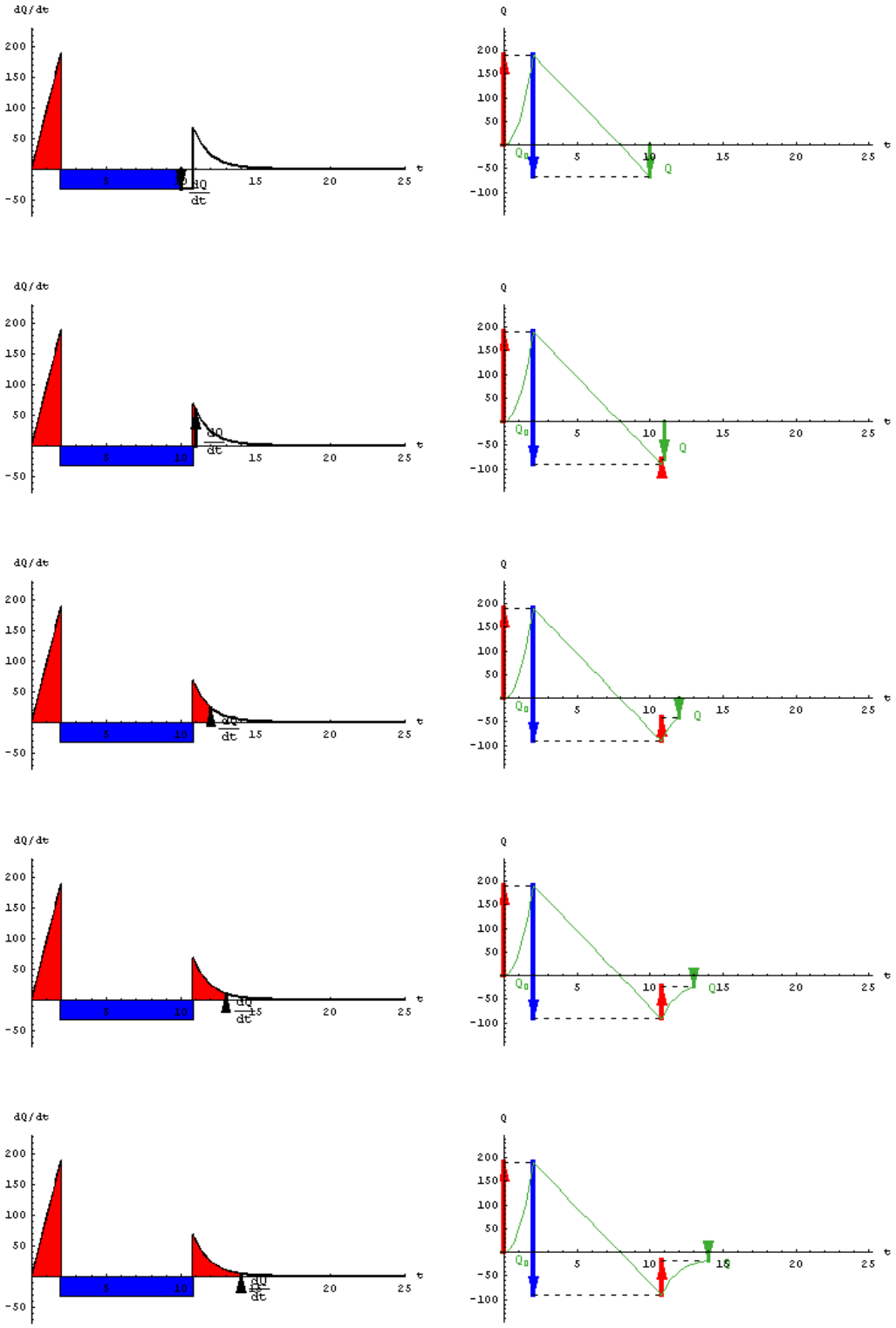
before executing the next command. All computed values are saved when you do this.)

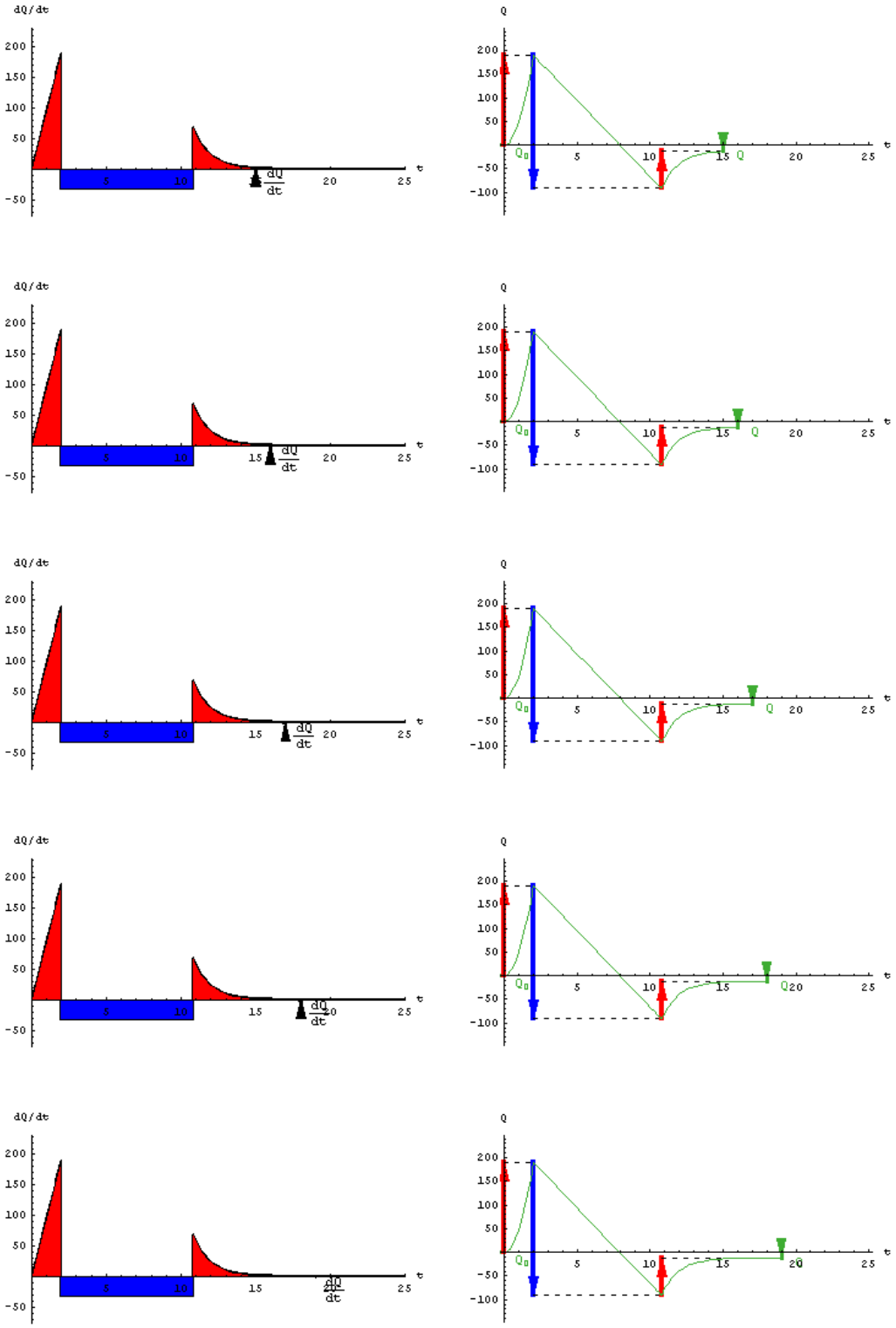
In[202]:=

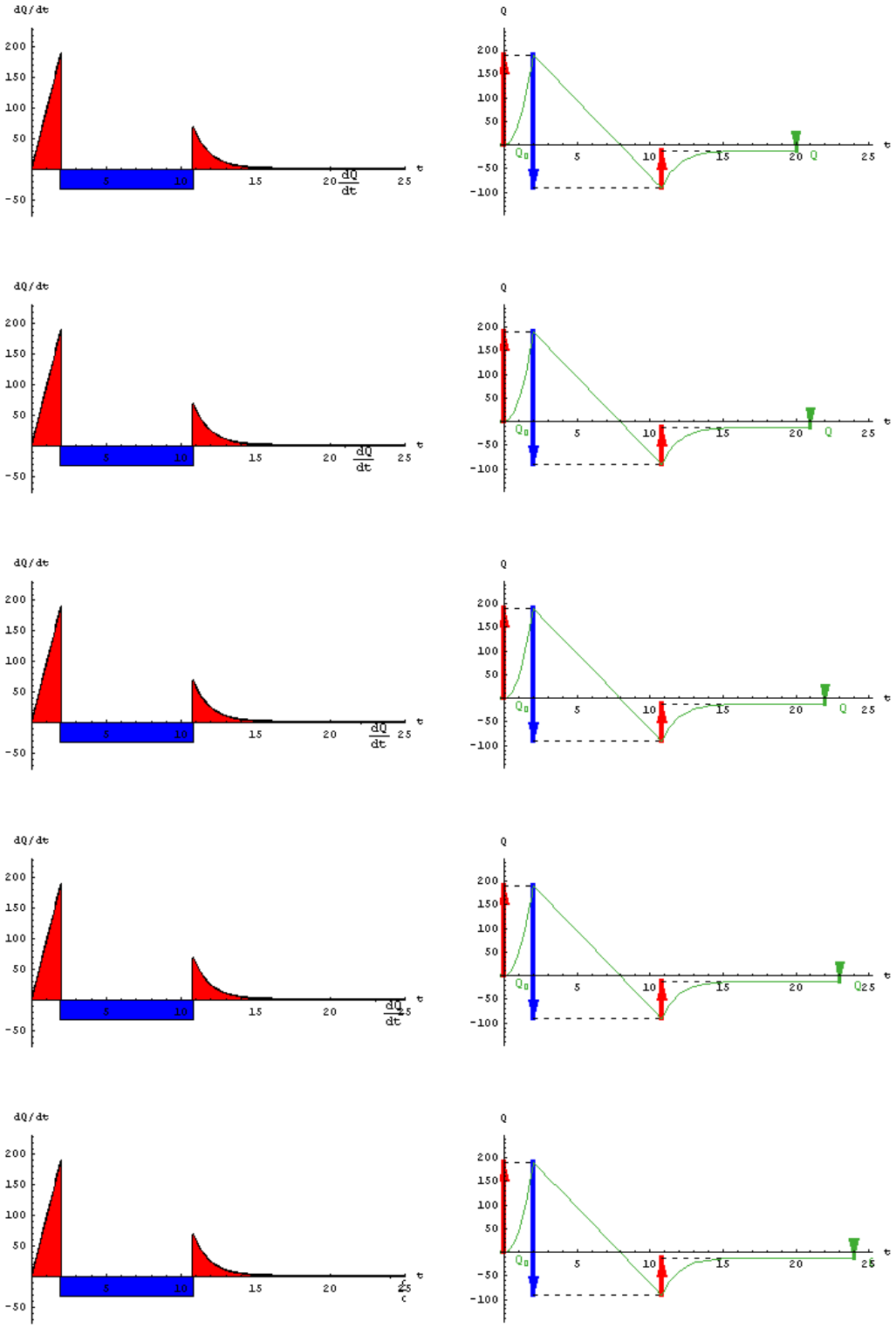
```
antidifferentiate[Q'[t], t, 0, 25, 0]
```

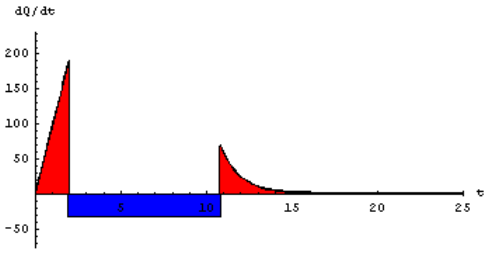


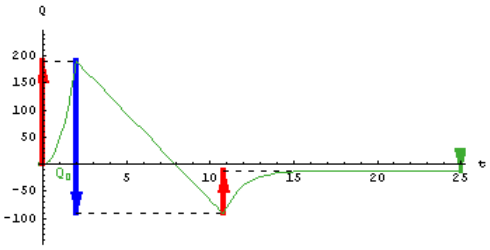

dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t


dQ/dt
Q
t
Q_0
$\frac{dQ}{dt}$


dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t


dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t
dQ/dt
Q
t





6. Redefine **Q′[t]** to be the velocity function, and use it to construct **Q[t]**, which will now represent the height of the rocket above the ground. Then graph **Q[t]**. To assign the current **Q[t]** function to **Q′[t]**, give **Q[t]** in a temporary name (e.g., **temp=Q[t]**), and then **Clear [Q]** and redefine **Q′[t_]** (e.g., **Q′[t_]=temp**).

In[203]:=

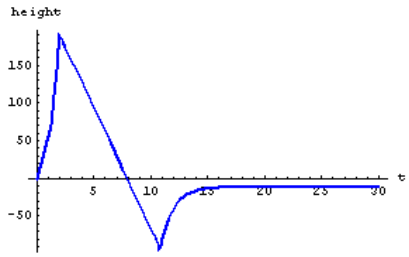
```
temp = Q[t];

Clear[Q];

Q'[t_] = Rationalize[temp];

Q[t_] = ∫_0^t Q'[u] du
```

Out[206]=

$$e^{-1083\,t/1250}\,(1.06916\times 10^{6}\,\text{UnitStep}[-10.8+t]+16.\,e^{1083\,t/1250}\,(1.\,(-11.7671+t)\,(-4.83376+t)\,\text{UnitStep}[-10.8+t]-0.989583\,(-2.+t)\,(-2.+t)\,(5.01053+t)\,\text{UnitStep}[-2.+t]+0.989583\,t^{3}\,\text{UnitStep}[t]))$$

In[207]:=

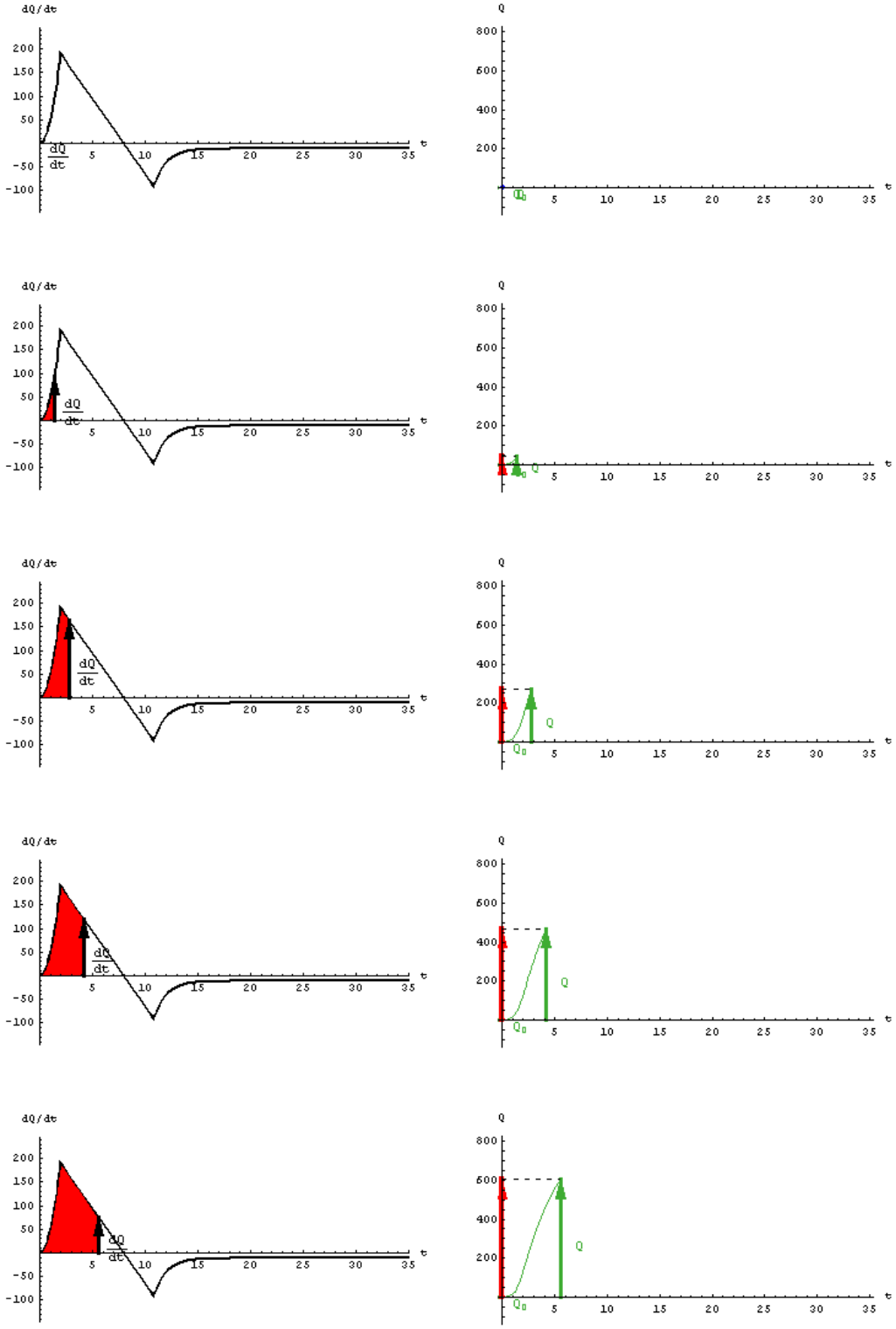
```
Plot[Q'[t], {t, 0, 30},
 PlotStyle → {Thickness[0.010],
   RGBColor[0, 0, 1]},
 AxesLabel → {"t", "height"},
 PlotRange → All];
```

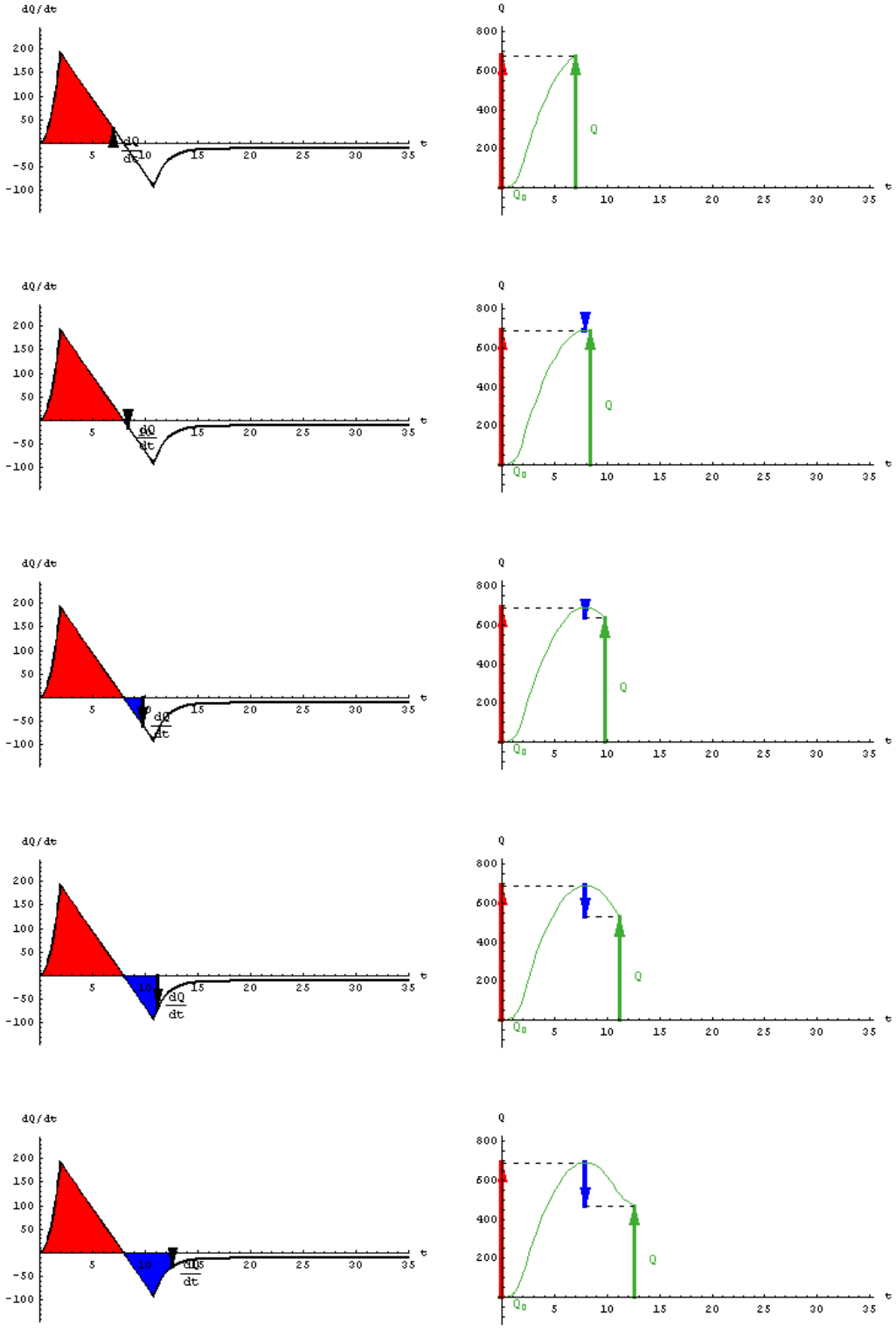


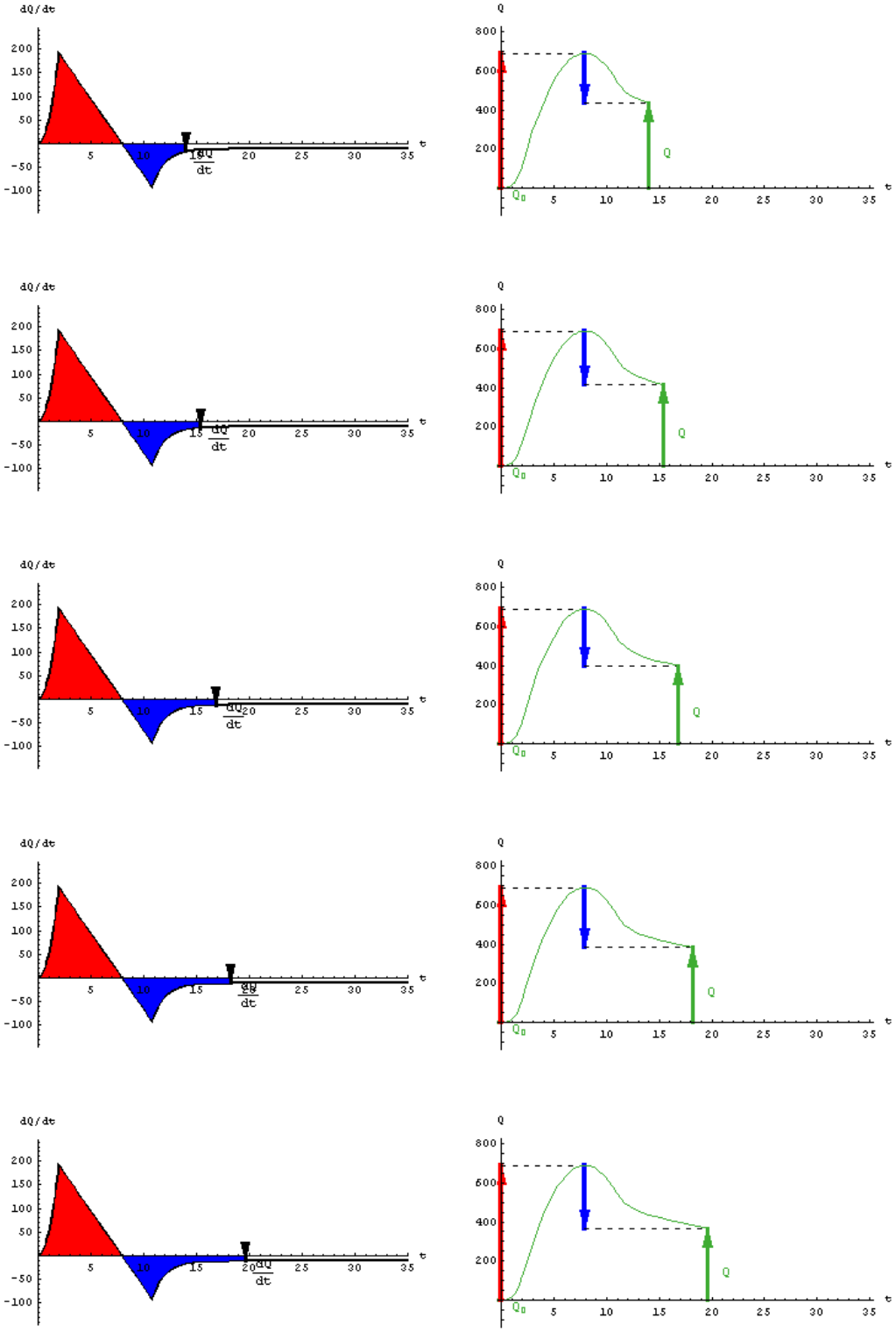
7. Use **antidifferentiate[ ]** to study the velocity and height functions as the rocket flies. (The **antidifferentiate[ ]** command generates a lot of graphics, which tend to fill up computer memory. Pull down the Kernel menu and select Delete All Output before executing the next command. All computed values are saved when you do this.)

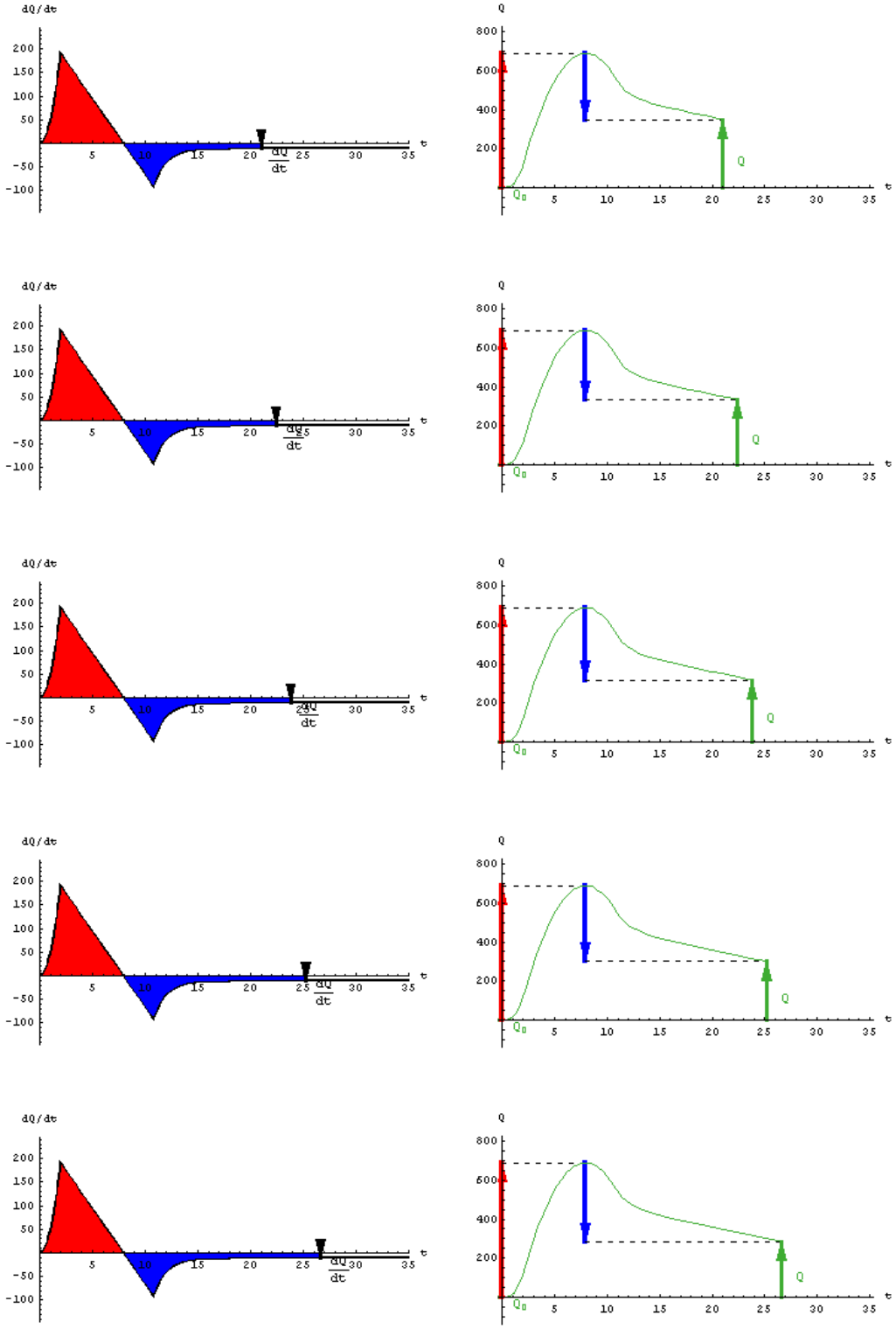
In[208]:=

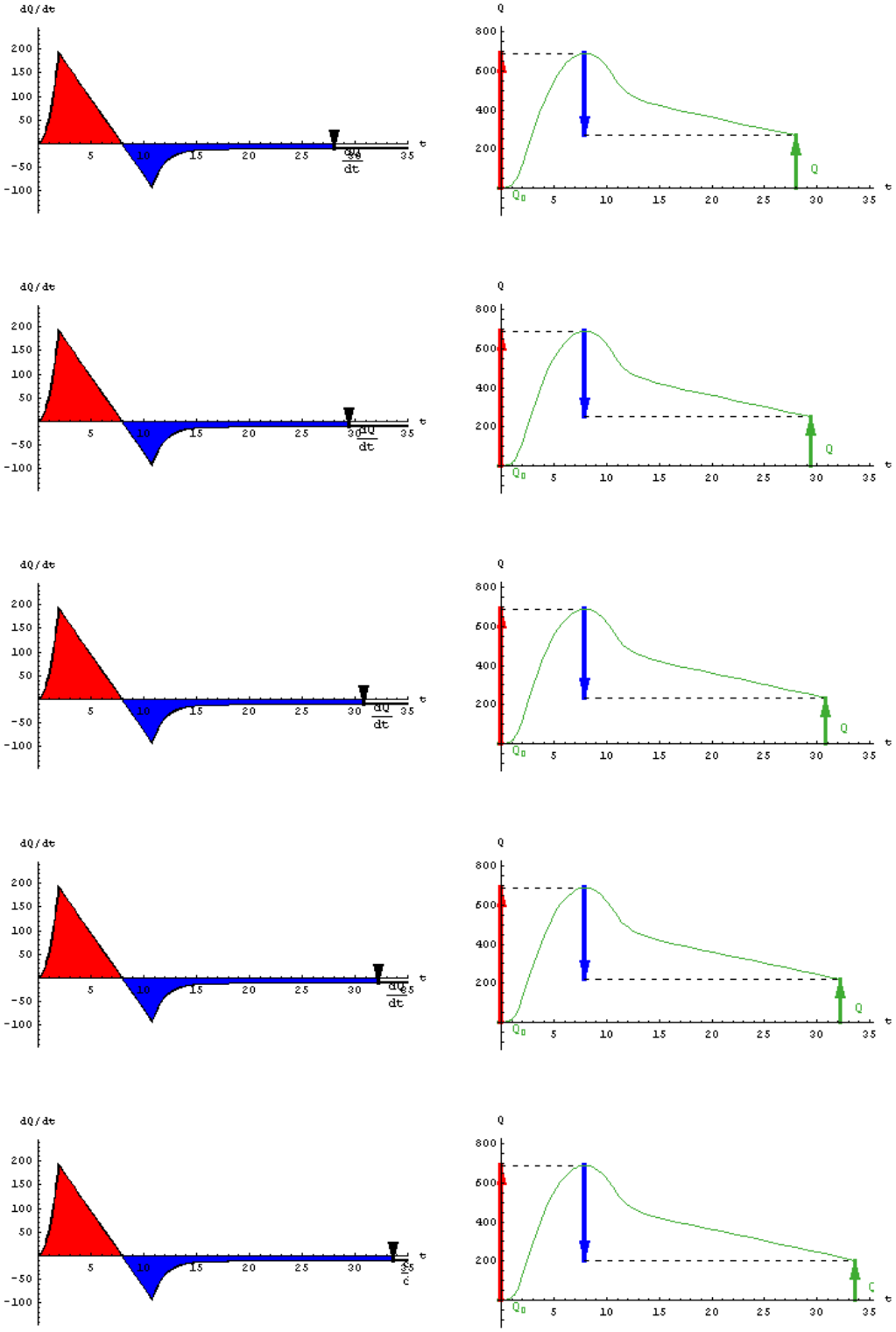
```
antidifferentiate[Q'[t], t, 0, 35, 0.0]
```

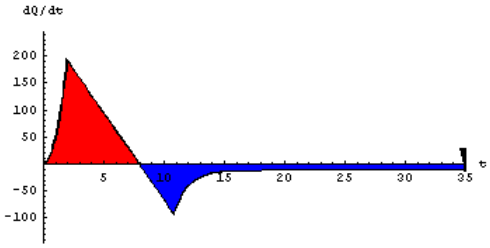

dQ/dt
Q
t
dQ/dt
Q

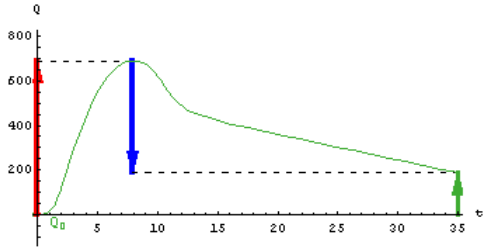

dQ/dt
Q
t
dQ/dt


dQ/dt
Q
t
Q0
dQ/dt


dQ/dt
Q
t
dQ/dt
Q
Q₀


dQ/dt
Q
t
Q_0
dQ/dt
Q





8. Determine how high the rocket goes and when it lands. To answer the latter question, you may have to adjust the time limits in the **antidifferentiate[ ]** command in the preceding cell.

9. Determine the average velocity of the rocket between the time that it is at its maximum height and when it lands. Also, find the time or times in the same interval of time that the average velocity is equal to the instantaneous velocity.

10. On the velocity graph, superimpose a graph of the constant average velocity over the interval described in step 9, and indicate on the graph the time or times that the instantaneous velocity is equal to the average velocity over the interval.

11. Find the linearizations of **Q[t]** at the times that the average velocity over the interval described in step 7 is equal to the instantaneous velocity. Also find the equation of the secant line on the graph of **Q[t]**, between the times that the rocket is at its maximum height and when it lands. Then superimpose the graphs of these lines (the secant and tangent lines) on the graph of **Q[t]**.

About *Mathematica*

The error message that appears when you execute the **Solve[ ]** command warns you that whenever *Mathematica* uses inverse functions to solve an equation, some solutions may not be provided. 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 $x = (4n+1)\frac{\pi}{2}$, where *n* can be any integer. You might try using *Mathematica* to solve sin x=1. Go Back.

FindRoot[Q[t] = = 50000, {t, 4}] searches for a numerical solution to the equation **Q[t] = = 50000**, starting with **t=4**. To find out more about the **FindRoot[ ]** command, pull down the Help menu, select the Help Browser, and type **FindRoot**. Go Back.

In *Mathematica*, the **UnitStep[t]** command is the unit step function, sometimes called the Heaviside step function. This function has a value of 0 when the value of the independent variable is less than 0, and its value is 1 when the value of the independent variable is greater than or equal to 0. To learn more about the **UnitStep[ ]** command, pull down the Help menu, select the Help Browser, and type **UnitStep**. Go Back.