5 Chapter 5 Building scripts

5.1 Chapter Overview

We briefly discuss a few extra concepts useful with solve: how to use a combination of relations rather than just a single equation, and how to take apart or combine by the various forms of solve.

We then explore the concept of script: a sequence of operations useful for solving a problem. We find that it's often the case that the need for a computation is driven by its reuse -- doing the same thing but with slight alterations each time. A frequently recurring scenario is a parameterized computation: 1) use variables to assign values to the parameters and 2) have subsequent steps of the computation refer to the parametric variables. Maple is well-equipped for reuse of parameterized scripts, since it has an operation Edit → Execute → Selection or Worksheet. This makes it easy to solve different versions of a problem by editing the parameter values and re-executing the script.

5.2 The structure of information in Maple: getting information from solve

The result of the solve operation can have multiple parts if there are multiple solutions to the equation. In this case, the result of solve is a sequence, list, or set of solutions, and we can select each part by giving an index (either 1 or 2).

\[ eq1s := 3 \cdot x = x^2 - 28 \]

\[ 3 \cdot x = x^2 - 28 \]

\[ \text{solve} \]

\[ \{x = -4\} \]

\[ \text{select entry 1} \]

\[ \{x = -4\} \]

\[ eq1s \]

\[ 3 \cdot x = x^2 - 28 \]

\[ \text{solve} \]

\[ \{x = -4\}, \{x = 7\} \]

\[ \text{select entry 2} \]

If we give solve a linear equation, it has only one solution. We can still select the first entry.

\[ eq2s := 3 \cdot x = 28 \]

\[ \text{solve} \]

\[ \left\{ x = \frac{28}{3} \right\} \]
\[ x = \frac{28}{3} \]

If we do "solve for \( x \)" for the same equation, we see that the answer comes back in a slightly different form. But it still has parts.

\[ \text{solve for } x \]

\[ 3x = x^2 - 28 \]

\[ [x = -4], [x = 7] \]

\[ [x = 7] \]

Maple (as well as many other programming languages) can compute with objects that have structure. Here are four different kinds of structures that Maple can handle:

**Table 5.1: Basic data structures in Maple and operations to extract parts of them**

<table>
<thead>
<tr>
<th>Type of structure</th>
<th>What they look like</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equations, inequalities</td>
<td>What math books do for equations and inequalities. The clickable menu allows you to get at the left hand side and right hand side of an equation or inequality.</td>
<td>[ x + y = 35 ] [ x + y ] [ x \geq \sqrt{57} ] [ \sqrt{57} \leq x ] [ x ]</td>
</tr>
<tr>
<td>Sequences</td>
<td>Values or expressions separated by a comma</td>
<td>( s := 19, 47, 92 ) ( 19, 47, 92 ) [ relations := x^2 - 2 = 0, 0 &lt; x ] [ x^2 - 2 = 0, 0 &lt; x ]</td>
</tr>
</tbody>
</table>
5.3 Finding simultaneous solutions, constraining solutions.

Suppose we want to solve the system of equations $x + y = 5$ and $-3y + 7 = x$. This means finding values of $x$ and $y$ that simultaneous satisfy both equations. We can do this in Maple by typing in the first equation and then the second, separated by a comma. This is called entering a sequence of equations. Right-clicking (control-click on Macintosh) on the sequence will allow you to solve the system.

In Lab 1, you discovered that `solve` could also handle inequalities as well as equalities. You can enter a sequence of equations and inequalities to `solve`. This can be used to limit solutions to a particular range of values.

### Table 5.2: Solving simultaneous equations

<table>
<thead>
<tr>
<th>Type of structure</th>
<th>What they look like</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets</td>
<td>A sequence surrounded by curly braces { }</td>
<td>( x^2 - 2 = 0 ) (5.9)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Scores} := {3, 7, 3, 10} ) (5.10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{select entry 3} \rightarrow 10 ) (5.11)</td>
</tr>
<tr>
<td>Lists</td>
<td>A sequence surrounded by square brackets [ ]</td>
<td>( \text{MyEquations} := [x + y = 3, y - 2 \cdot x = 37] ) (5.12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{solve} \rightarrow {x = \frac{34}{3}, y = \frac{43}{3}} ) (5.13)</td>
</tr>
</tbody>
</table>

Note that the result of this `solve` is a set of equations.

For the time being, we just want you to recognize the different kinds of structures that are output by `solve` and other functions and be able to select parts from them. Later on we will get a lot of work done by performing more sophisticated operations with them.
\[ p, x \geq 0 \]

\[
x^4 + 3x^2 - 57.5 = 0, \ 0 \leq x
\]  
(5.18)

solve

\[
\{x = 2.495959218\}
\]  
(5.19)

select entry 1

\[
x = 2.495959218
\]  
(5.20)

This gets the solver to list only the positive real solutions. We might be interested in only non-negative real solutions, if, for example, we were using the equation to model a physical situation and \( x \) was measuring weight. In this case the other values of \( x \) wouldn't be relevant to our situation.

5.4 Scripting: creating computational work in reusable form

Consider the problem you did in Lab 1, along with a solution:

**Version 1 and solution**


A breeding group of 20 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{220}{1 + 10 \cdot (0.83)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 80.

(a) Graph \( N \) versus \( t \).

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \).)

**Solution to (a)**

\[
N = \frac{220}{1 + 10 \cdot (0.83)^t}
\]

right hand side

\[
N = \frac{220}{1 + 10 \cdot 0.83^t}
\]

assign to a name

\[
sheepPopExpr
\]
Once we have the sheep population, we need to play with the plotting ranges to see when the leveling off occurs. We'd have to think about it and experiment a bit -- but the computer makes the reploting easy to do once we make our decisions about what to try.

\[ \text{sheepPopExpr} \]

\[
\frac{220}{1 + 10^{0.83t}}
\]  

\( \rightarrow \)

\[ \text{# of sheep versus time} \]

\[ N \]

\[ t \]

Solution to (b)

\[ 80 = \text{sheepPopExpr} \]

\[
80 = \frac{220}{1 + 10^{0.83t}}
\]  

\( \rightarrow \)

\[ \{ t = 9.354227718 \} \]  

We can read the leveling off point from the plot, assuming that we have figured out the appropriate time range in (b). Alternatively, we can do a little calculus and take the limit of the expression as \( t \) goes to infinity.

\[ \lim_{t \to \infty} \text{sheepPopExpr} \]

\[ 220. \]  

We can imagine ourselves working as a environmental engineer for the National Forest Service and being very pleased with ourselves for solving the problem with Maple. But now we are handed two more wildlife management problems to do, from other regions in our territory:
Version 2
A breeding group of 33 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{330}{1 + 10 \cdot (0.79)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 85.

(a) Graph \( N \) versus \( t \).
(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.
(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \)).

Version 3
A breeding group of 45 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{450}{1 + 10 \cdot (0.63)^t}
\]

and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 90.

(a) Graph \( N \) versus \( t \).
(b) How many years must the state of Montana maintain a program to care for the sheep? Show your work.
(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \)).

We have the feeling that we will shortly be handed problems for a number of other locations as well. How can we reuse our original work with minimal effort?

If we had the first solution, we could produce the second solution through *copy-paste-edit-re-execute*:

**Executing a clone of a script through copy-paste**

1. Copy and paste the original solution into a new document, or to a spot later in the same document. You do this as with a word processor -- select a region of the worksheet with the mouse, type control-C (command-C on a Mac). Then position the document cursor in the destination, and type control-V (command-V on a Mac). A duplicate of the solution should appear there.

2. Using the mouse, edit the formulae as needed. You only need to edit the black (input) formulae, not the blue ones.

3. Position the mouse at the first computation and hit enter. Continue to work your way through the sequence of the commands.

4. Alternatively, select the entire region containing the edited version of the solution and hit Edit->Execute->Selection.

5. If the region to be executed is the entire worksheet, then rather than selecting anything you can do Edit ->Execute->Worksheet.

The results of executing the edited script are??? not totally correct. We will have to change a few things by hand: the clickable operation "evaluate at a point" in step (b) will use \( N=80 \) (which is what the copy says) instead of \( N=85 \), so we will have to redo it. Also, the last plot will not show an appropriate vertical range unless we do that. We'd like to do a little less of this hand-tuning in re-execution. Also, we'd like to make it easier to remember what to change between multiple versions.

*A breeding group* (page 62).
5.5 Rewriting the script using assignment

While copying and editing is probably a little faster than typing in the whole script again, we can reduce the amount of hunting around for changes by writing a script finding the parameters of the problem, and writing the script so that it assigns values to the parameters at the beginning of the problem.

You may be doing assignments at several points in your calculation. Only the ones that would need to be changed between different versions of the problem define problem parameters. You may find the other ones very useful, but they don't have parameter status.

Finding and naming parameters

First, solve at least one version of the problem. Then, imagine what would need to be changed if you were trying to solve alternative versions of the problem. You can find parameters if you have several versions of a problem by looking at what changes in the worksheet from version to version.

For example, in the sheep problem, we note the following things changing in different versions of the problem. We pick names for these.

1. the numerator of the "sheep equation" ($P$)
2. the coefficient in the denominator of the equation ($c$)
3. the value of the stable population ($s$)

The other number in the script is the original population of the sheep. We realize that it can be derived by evaluating the "sheep equation" at $t=0$. Thus this value does not need to be a parameter as we can derive it from the other information. It might be a good check though.

We then write the script to assign values to the parameters at the start of the script, and then write the other operations and expressions in terms of the symbols.

This allows us to redo the script just by changing the values at the beginning of the script. We use the word processing features to add extra directions to make the script easier to use, basically saying "here, change these things", and "this is the end of where you should stop copying".

The result of executing the script is Version 2, with use of parameters (page 64).

Having created this script, we can handle the third version of the problem by editing the values of the parameters and executing the script again by selecting the entire script with the mouse, and then doing Edit->Execute->Selection. We get this result:

5.6 Summary of script writing

Script writing is appropriate when you expect to handle several different versions of the same problem. In professional work, this is often the case -- if it's worth doing at all, it's probably going to occur more than once.

Figure out how to solve the problem first. Then write the script. There's really not much point in writing the script if you don't have some idea of the sequence of operations in it.

Once you have a worksheet of instructions for solving one version of the problem, look at it and the other versions and find the parameters. Set up a new worksheet where the first thing you do is to assign values to the parameters. Then work through the rest of the instructions and rewrite them to use the parameters instead of the fixed values from one version of the problem.
5.7 Troubleshooting scripts

Programming books have the tendency to show things that work. They say less about what to do about the things that don't work. However, it is usually the case that computer users often spend more time "getting things to work" than "working". Learning how to get out of jams is at least as valuable as knowing how to enter operations and what they mean. We offer this advice, which is "commonsense about work" as applied to interactive software development:

1. Solve one version of the problem before you try to start scripting. You can use Maple to experiment -- enter and edit snippets of operations that try out the solution technique for part of the problem. Eventually edit them together so that they solve the whole problem. If you have only fuzzy notions about the math or the operations you want to proceed, your computer work will just amplify that. Having a worksheet that solves one version of the problem can remove a lot of the fuzziness.

Where does the inspiration for solving the problem come from? If you are lucky, the solution may be told to you. Or you may find a description of a similar problem as a starter. But the big bucks, as they say, go to those who can devise the solution plan themselves.

2. Limit each step so that it is a small step. If you get into trouble, you will be able to nip it at the bud. This becomes a more prominent tactic in later work when we are tempted to construct long-winded one-line expressions that do everything at once.

3. Test pieces individually, then put them together. For example, if you don't really understand how to make a dotted line plot in teal with a title, you should try to do that with a simple plot (such as \( x^2 \) ) rather than an expression whose shape you aren't that familiar with it. Then take what works and substitute the real expression you want to plot in a copy of the \( x^2 \) plot operation.

If you think about it, this is similar to what happened in Fall 2010 ENGR 101 Lab 2, where they first had you learn how to trigger an oscilloscope with the output of the function generator, rather than the output of the camera/flash sensor. The complexity of troubleshooting is reduced if you half as many unknowns to worry about. This can be called divide and conquer troubleshooting.

4. If what you have doesn't work, find something similar which does work, and then incrementally edit it. For example, if you can't get \( a := x^2 + 3 \cdot x + 1 \) in, then first see whether you can get \( a := 1 \) to work. Once you succeed with that, edit the expression to \( a := 3 \cdot x + 1 \) and so forth.

5. If strange things continue to happen despite your best efforts to troubleshoot, it may be that previous settings in the Maple session are interfering with your current work. Recall that some people in the ENGR101 lab couldn't get their oscilloscopes to work because of settings changed in the oscilloscopes by groups earlier in the day. This can be particularly true if you are developing scripts and are assigning parameter values, then switch to development of another script in the same document. The values you assigned will not magically unassign themselves when you start working on something new in the same worksheet.

The remedy for this is to put a restart in as the first operation in your script, then re-execute the worksheet.

5.8 Attachments

Attachment: Version 2 of sheep script without parameters

Version 2 of sheep problem, with edited script

A breeding group of 30 bighorn sheep is released in a protected area in Colorado. It is expected that with careful management the number of sheep, \( N \), after \( t \) years will be given by the formula:

\[
N = \frac{330}{1 + 10 \cdot (0.79)^t}
\]
and that the sheep population will be able to maintain itself without further supervision once the population reaches a size of 85.

(a) Graph $N$ versus $t$.

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of $t$.)

Solution to (a)

\[
N = \frac{330}{1 + 10 \cdot (0.79)^t}
\]

Solution to (b)
$85 = \textit{sheepPopExpr}$

\[
85 = \frac{330}{1 + 10^{0.79t}}
\] (5.28)

\[
\text{solve } \rightarrow \{t = 5.277302835\}
\] (5.29)

We can read the leveling off point from the plot, assuming that we have figured out the appropriate time range in (b). Alternatively, we can do a little calculus and take the limit of the expression as $t$ goes to infinity.

\[
\lim_{t \to \infty} \textit{sheepPopExpr}
\]

\[
330.
\] (5.30)

**Attachment: Version 2 of Sheep Script, with parameters**

**Version 2, with use of parameters**

Start of parameters -- change these for each version of the problem

\[
P := 330
\] (5.31)

\[
c := 0.79
\] (5.32)

We call the size of the stable population $s$.

\[
s := 85
\] (5.33)

**End of parameters**

(a) Graph $N$ versus $t$.

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of $t$.)

(a)

\[
\textit{sheepPopExpr} := \frac{P}{1 + 10^{-(c)t}}
\]

\[
\frac{450}{1 + 10^{0.83t}}
\] (5.34)

Note that this $\textit{sheepPopExpr}$ is not a parameter since assignment is always the same for all versions of the problem.
To make the graphing work all the time, we set the vertical axis to "P+30" rather than a fixed value. This will set the vertical axis so that it will be 30 more than the value of P which from the first problem we have realized is the top of the graph.

\[ s = \text{sheepPopExpr} \]

\[
\frac{330}{1 + 10^{0.79t}}
\]

\[ t = 5.277302835 \]

This is the time in years that model predicts it will take for the sheep population to reach self-sustaining status, allowing the wildlife managers to move onto another job.

(c) A little thinking reveals that the leveling off value (as indicated from the graph), is the numerator of \( \text{sheepPopExpr} \), since the denominator goes to 1 as \( t \) goes to infinity.

\[
\lim_{{t \to \infty}} \text{sheepPopExpr}
\]

\[ 330. \]
End of script

**Attachment: Version 3 of Sheep Script, with parameters**

**Version 3 with edited parameters and re-execution**

**Start of parameters -- change these for each version of the problem**

\[ P := 450 \]

450

(5.38)

\[ c := 0.83 \]

0.83

(5.39)

\[ \text{sheepEquation} := N = \frac{P}{1 + 10 \cdot (c)^t} \]

\[ N = \frac{450}{1 + 10 \cdot 0.83^t} \]

(5.40)

We call the size of the stable population \( s \).

\[ s := 100 \]

100

(5.41)

**End of parameters**

(a) Graph \( N \) versus \( t \).

(b) How many years must the state of Colorado maintain a program to care for the sheep? Show your work.

(c) How many bighorn sheep can the environment in the protected area support? (hint: examine the graph for large values of \( t \).)

(a)

\[ \text{sheepPopExpr} := \frac{P}{1 + 10 \cdot (c)^t} \]

(5.42)

Note that this sheepPopExpr is not a parameter since assignment is always the same for all versions of the problem.

To make the graphing work all the time, we set the vertical axis to "P+30" rather than a fixed value. This will set the vertical axis so that it will be 30 more than the value of P which from the first problem we have realized is the top of the graph.

\[ \text{sheepPopExpr} \]

\[ \frac{450}{1 + 10 \cdot 0.83^t} \]

(5.43)
\[ s = \text{sheepPopExpr} \]

\[
100 = \frac{450}{1 + 10 \cdot 0.83^t}
\]

\[
\Rightarrow \{t = 5.634221548\}
\]

This is the time in years that model predicts it will take for the sheep population to reach self-sustaining status, allowing the wildlife managers to move onto another job.

(c) A little thinking reveals that the leveling off value (as indicated from the graph), is the numerator of \( \text{sheepPopExpr} \), since the denominator goes to 1 as \( t \) goes to infinity.

\[
\lim_{t \to \infty} \text{sheepPopExpr} = 450.
\]
5.9 Summary of Chapter 5 material

<table>
<thead>
<tr>
<th>Type of structure</th>
<th>What they look like</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic data structures in Maple</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Equations and inequalities | Expression related by $=, >, <, \geq, \leq$, or $\neq$. | $x + y = 0$
$x + y = 0$ (5.46)
$n^2 - 3 \cdot n > 4$
$4 < n^2 - 3 \cdot n$ (5.47) |
| Sequences | Values separated by a comma | 19, 47, 92
19, 47, 92 (5.48) |
| Lists | A sequence surrounded by square brackets $[]$ | $MyData := [1.0, x, \frac{3}{4}, a]$
$[1.0, x, \frac{3}{4}, a]$ (5.49) |
| Sets | A sequence surrounded by curly braces $\{\}$ | $Scores := \{3, 7, 3, 10\}$
$\{3, 7, 10\}$ (5.50) |

Solving simultaneous equations

- $x + y = 5, -3y + 7 = x$
  $\text{solve}$
  $(x = 4, y = 1)$ (5.52)
- $p := x^4 + 3x^2 - 57.5 = 0$
  $\text{solve}$
  $\{x = 3.0380606341, x = -3.0380606341, x = 2.495959218, x = -2.495959218\}$ (5.54)
- $p, x \geq 0$
  $x^4 + 3x^2 - 57.5 = 0, 0 \leq x$
  $\text{solve}$
  $(x = 2.495959218)$ (5.56)

The result of this solve is a set of solutions.
Solving this equation produces 4 roots. Two of them are complex numbers (since they have I in them) the others are real.
This gets the solver to list only the positive real solutions. We might be interested in only non-negative real solutions, if, for example, we were using the equation to model a physical situation and $x$ was measuring weight. In this case the other values of $x$ wouldn't be relevant to our situation.
<table>
<thead>
<tr>
<th>Scripts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Creating a script for a problem</strong></td>
<td>In a Maple worksheet, take a version of a problem and create a sequence of operations in the worksheet that solve it. Note similarities and differences between different versions of the problem. Envision what you'd have to change in the worksheet in order to solve a different version of the problem, and what would stay the same. You may have to rewrite some of the expressions to refer to the parameter rather than the value. Assign the parameters at the beginning of the script. Rework the rest of the script so that the formulas refer to the parameters by name, rather than the values used in the original version of the problem. For example, if the value 42 appears in several places in your script, define a parameter ( p := 42 ) at the start of the script and edit the other occurrences of 42 to be ( p ) instead. When you have a different version of the problem, you can edit just the single line ( p := 42 ) into say ( p := 47 ) and won't need to edit any other lines of the script.</td>
</tr>
<tr>
<td><strong>Using a script</strong></td>
<td>Copy and paste the script to a new location Edit the assignments to reflect the new version of the problem. Edit-( \rightarrow )Execute-( \rightarrow )Selection, or just hit <code>enter (return)</code> on Macintosh) multiple times to perform the operations in the new version of the script.</td>
</tr>
<tr>
<td><strong>Rationale for using scripts</strong></td>
<td>More work to do than clickable interface the first time. Saves time if you expect to want to reuse the operations on multiple versions of the same problem. Also it is less error prone.</td>
</tr>
</tbody>
</table>