CHAPTER 15 SOLVED SIMPLE MODELING PROBLEMS WITH ASCEND

In this chapter we present two simple modeling problems for which we then show you our ASCEND models for solving them. Modeling is a matter of style, and we will start to show you what we believe to be good styles for modeling. We assume you have not used the ASCEND system before. These problems are very generic and should be readily followed by anyone with a modest technical background.

One purpose is to show you some of the different ways you can use ASCEND. Specifically we want to show you that you can use ASCEND to setup and solve the simple types of problems that you might have solved using a spreadsheeting program. Indeed, we use ASCEND to solve homework problems quite often. When you factor in the powerful debugging tools, you might find it faster to use ASCEND, especially as the models get more complex. And no one would want (we think) to solve a 20,000 simultaneous nonlinear equation model using a spreadsheeting program.

A major advantange of using ASCEND is that once you have written, debugged and learned to solve such a model, you can interactively alter the "fixed" flags for the variables, changing which variables are to be fixed and which to be calculated. You can then immediately solve or optimize the new problem, using the previously solved problem as the initial guess.

15.1 ROOTS OF A POLYNOMIAL

In this problem you wish to find the roots of a polynomial. Assume you do not wish to keep the code. You could readily use a spreadsheet program with its "root finder" routine to solve this type of problem, but you can as readily use ASCEND.

15.1.1 Problem statement

Numerically compute the roots of *$(x-1)(x-5)(x+7)(x^2+1) = 0$*. (Given in this form the roots are obviously 1, 5, and -7. Two roots are complex, and ASCEND will not find them.)

15.1.2 Answer

You can find the roots by guessing initial points after typing in, loading and compiling the following model. You would use any text editor to enter this model into the computer. If you use a "WYSIWYG" (what you see is what you get) editor such as Word or Framemaker, be sure to save the file as a **text only** file. If possible, use a simpler text editor.

```
MODEL polynomial_roots;
   x        IS_A generic_real;
   (x-1)*(x-5)*(x+7)*(x^2+1) = 0;
END polynomial_roots;
```

This simple model is a stand-alone model. You need no other predefined libary models to support it. Load and compile an instance of this model (using tools in the LIBRARY tool set), browse it (using the BROWSER tool set) to see if it appears to have compiled correctly, and then pass it to the SOLVER tool set.

This model involves a single equation in the single unknown variable, x. The ASCEND solver treats a single equation in one unknown in a special manner when asked to solve it. The solver first attempts to rearrange the equation by simple algebraic manipulations to isolate the unknown on the left hand side of the equation in the form x = expression not involving x. In this form, solving is simply evaluating the expression on the right hand side once. Here the solver would fail as there is no way to isolate x on the left hand side as the equation is a fifth order polynomial in x. When rearrangement fails, the solver uses bisection to locate a root in the range between the lower and upper bound on the variable. You can see the bounds, x.lower and x.upper, using the BROWSER. The default values for these bounds are plus and minus 10^{20} respectively, which gives a very large range in which to look for the root. You should change the bounds[1] to more realistic ones. Selecting bounds to be -10 to 0 and then solving will find the root x=-7. Selecting other values will find the other roots.

This model illustrates that you can quickly set up and solve simple problems using ASCEND. Note that you would have been required to place bounds on x had you used a goal seeking tool in a spreadsheeting program if you wanted to control which root to locate.

15.2 NUMERICAL INTEGRATION OF TABULAR DATA

This problem is similar to the previous one in that it is very easy to set up and solve. It adds in the notion of units (e.g., ft, m, hr, atm) which ASCEND handles in a straight-forward manner, relieving modelers from thinking about converting among the many units they might use when expressing the data for a problem.

Again we are talking about producing throw-away code. All we are really concerned with here is the answer which we intend to put into a report. We are using ASCEND as a "calculator."

15.2.1 PROBLEM STATEMENT

Given the following velocity data vs. time, estimate numerically the distance one has traveled between time equal to zero and 100 seconds.

Table 1: Velocity data to be integrated

data point number	time, s	velocity, ft/min
1	0	100
2	10	120
3	20	130
4	30	135
5	40	140
6	50	160
7	60	180

1. To set the value for a variable interactively, select the variable when it is display in the right window or in the lower window with the RIGHT mouse button (the other button). A window for changing its value opens.

Table 1: Velocity data to be integrated

data point number	time, s	velocity, ft/min
8	70	210
9	80	240
10	90	220
11	100	200

15.2.2 ANSWER

(This example will be solved using variables whose types are defined in the file atoms.a4l. You must load this file first before loading the file with the code below, else you will experience a number of diagnostic messages indicating missing type definitions. See Chapter 18 for a discussion of libraries on ASCEND.)

The distance traveled is the integral of the velocity over time. We can use Simpson’s rule to carry out this integration for evenly spaced points.

$$\begin{aligned} \mathbf{d} &= \mathbf{((v[1] + 4\,v[2] + v[3]) + (v[3] + 4\,v[4] + v[5]) + \ldots.} \\ &\quad \mathbf{+(v[N\text{-}2] + 4\,v[N\text{-}1] + v[N]))*\Delta t/6} \end{aligned} \tag{15.1}$$

where *d* is the distance covered when traveling at the velocities, *V[k],* listed. This formula requires there to be an odd number of 3 or more evenly space data points, which is fine here as we have eleven velocity points evenly spaced in time. (If there had been an even number of points, we could use Simpson’s rule for all but the last time interval and use a simple trapezoidal rule to integrate it.)

An ASCEND model to evaluate this distance is as follows. The types definitions for *speed, time* and *distance* are in the file *atoms.a4l.*

```
MODEL travel_distance;
   kmax              IS_A integer_constant;
   v[1..2*kmax+1]    IS_A speed;
   delta_time        IS_A time;
   d                 IS_A distance;

   d = SUM[v[2*k-1]+4*v[2*k]+v[2*k+1] SUCH_THAT k IN
          [1..kmax]]*delta_time/6;
END travel_distance;

MODEL test_travel_distance REFINES travel_distance;
   kmax     :== 5;

METHODS
   METHOD specify;
       v[1..2*kmax+1].fixed   := TRUE;
       delta_time.fixed       := TRUE;
   END specify;

   METHOD values;
      v[1]                    := 100 {ft/min};
      v[2]                    := 120 {ft/min};
      v[3]                    := 130 {ft/min};
      v[4]                    := 135 {ft/min};
      v[5]                    := 140 {ft/min};
      v[6]                    := 160 {ft/min};
      v[7]                    := 180 {ft/min};
      v[8]                    := 210 {ft/min};
      v[9]                    := 240 {ft/min};
      v[10]                   := 220 {ft/min};
      v[11]                   := 200 {ft/min};
   delta_time                 := 10 {s};
   END values;
END test_travel_distance;
```

If you look carefully at this model, you will note that we did NOT account for the conversion factors required because velocities are in ft/min while the time increment is in seconds. ASCEND understands these units and makes all the needed conversions. When you run this model, you can ask for the distance to be displayed to you in any supported length units you would prefer (e.g., ft, mile, m, cm, angstroms, lightyears). The distance traveled, when reported using SI units, is 42.84 m.