CHAPTER 6 PLOTTING DATA SAMPLED FROM COMPLEX MODELS

Often you need a plot of data sampled from arbitrary locations in a model that are not naturally grouped in a single easily plotted vector. The plot.a4l library provides models (plt_curve, plt_plot_symbol, and plt_plot_integer) that can be used with the Browser’s Display Plot button. In this chapter we see how to create such a plot using the ASCEND statement ALIASES/IS_A to sample data from a mechanical system of stretched springs, masses, anchors, and fingers. Creating plots of time series data output from ASCEND’s initial value solver LSODE is discussed in Section 12.3, ”Viewing Simulation Results,” on page 125.

Chemical engineers who can tolerate distillation models should visit the file plotcol.a4c in the models library for more complicated examples of plotting and visit the model *simple_column_profiles* in column.a4l for more complicated examples of sampling data. Reading this chapter first may be of help in interpreting those models.

6.1 THE GRAPH WE WANT

We want to plot the positions X1 to X3 of the connecting hooks h1, h2, and h3 in a mechanical system as shown in Figure 6-1. The anchor,

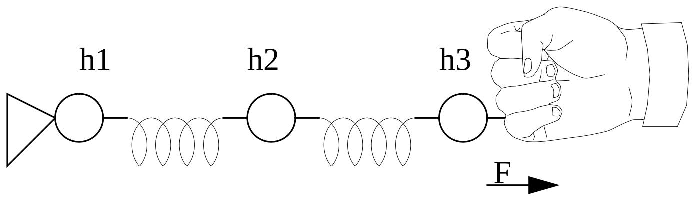

Figure 6-1 Spring test model system, st.

hooks, springs, and finger (we could replace either spring with a block mass, also) are all separate objects which we have modeled very simply. These models are given at the end of the chapter and can also be found (with improvements) in force1d.a4c, a model file in the distributed ASCEND libraries.

Plotting is usually a post-solution analysis tool, so our plots should not be entangled with the basic models or with the total mechanical system model, *st*. We might want to explain the system *st* to someone and this could be hard to do if the code is cluttered up with plot information.

6.2 CONSTRUCTING A PLOT CURVE

The plot library models follow object-oriented thinking carefully, perhaps a little too carefully. A plt_plot_integer is a plottable model built out of plt_curves which are in turn built out of arrays of data points from the user. Constructing these data arrays is the only significant challenge in using the plot models. Begin by building a new model with the system *st* as a part:

```
MODEL plot_spring_test;
   st IS_A spring_test;
   Plot_X IS_A plt_plot_integer(curve_set,curves);
END plot_spring_test;
```

We want to create a plt_curve from the array of hook numbers y_data[1..3] plotted against horizontal hook position x_data[1..3]. There are obvious problems with the model above: *curves* and *curve_set* are used without being defined and there is no mention of x_data or y_data.

Begin by using an ALIASES/IS_A statement to construct the array of positions x_data from the variables X stored in the hooks of model *st*.

```
x_data[Xset] ALIASES (st.h1.X,st.h2.X,st.h3.X) WHERE Xset
IS_A set OF integer_constant;
```

This statement creates a set, Xset, indexing a new array x_data with elements collected from st. Since the value of Xset is not specified, it becomes by default the set [1,2,3].

Now we need the hook numbers, y_data. These do not exist in st, so we create them. We will set the numeric values of these in the *default_self* method. We will include method in the final model, but do not show it here.

```
y_data[Xset] IS_A real;
```

Having both y_data and x_data, we can construct a curve from them:

```
X_curve IS_A plt_curve(Xset,y_data,x_data);
```

6.3 CONSTRUCTING THE ARRAY OF CURVES

We have a curve, but the plt_plot_integer model Plot_x expects an array of curves and the set indexing this array as input. We can make both from X_curve easily using, once again, an ALIASES/IS_A statement.

```
curves[curve_set] ALIASES (X_curve) WHERE curve_set IS_A
set OF integer_constant;
```

All the pieces are now in place, so we have the final model:

```
MODEL plot_spring_test;

   (* create our system model and plot. *)
   st IS_A spring_test;
   Plot_X IS_A plt_plot_integer(curve_set,curves);

   (* Gather the sampled data into an array *)
   x_data[Xset] ALIASES (st.h1.X,st.h2.X,st.h3.X)
   WHERE Xset IS_A set OF integer_constant;
   (* Create the Y coordinates *)
   y_data[Xset] IS_A real;

   (* create the curve *)
   X_curve IS_A plt_curve(Xset,y_data,x_data);
   (* Make X_curve into the array for plt_plot_integer *)
   curves[curve_set] ALIASES (X_curve) WHERE
   curve_set IS_A set OF integer_constant;

METHOD default_self;
   RUN st.default_self;
   st.s1.L0 := 0.2{m}; (* make st more interesting *)
   RUN Plot_X.default_self;
   RUN X_curve.default_self;
   FOR i IN Xset DO
      y_data[i] := i;
   END FOR;
   X_curve.legend := ‘meter’;
   Plot_X.title := ‘Hook locations’;
   Plot_X.XLabel := ‘location’;
   Plot_X.YLabel := ‘hook #’;
END default_self;
END plot_spring_test;
```

6.4 RESULTING POSITION PLOT

We can compile the plot model and obtain the graph in with the following short script.

```
READ FILE force1d.a4c;
COMPILE pst OF plot_spring_test;
BROWSE {pst};
RUN {pst.st.reset};
SOLVE {pst.st} WITH QRSlv;
PLOT {pst.Plot_X} ;
SHOW LAST;
```

We can also obtain the plot by moving to pst.Plot_X in the Browser window and then pushing the Display->Plot button or then typing "Alt-d p". We see the hooks are positioned near 0, 230, and 370 mm. We also see that xgraph sometimes makes less than pretty graphs.

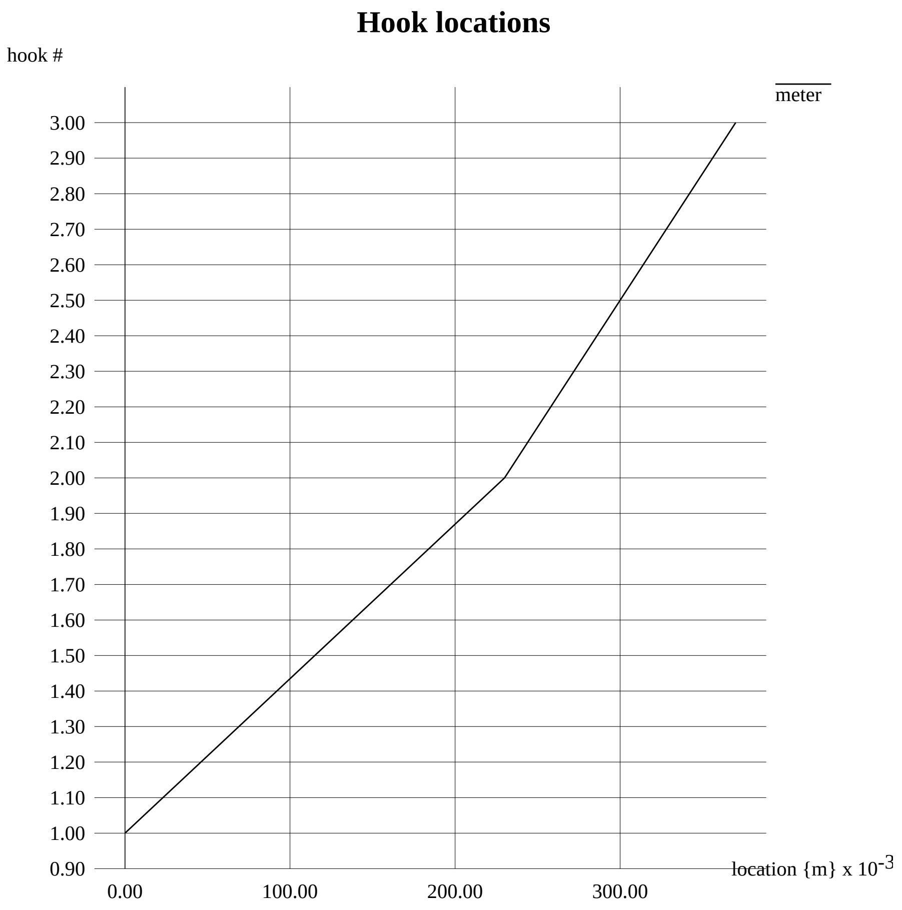

Figure 6-2 Plot_X in plot_spring_test

6.5 1-D MECHANICAL HOOK, SPRING, MASS, ANCHOR, AND FINGER MODELS

The models used in this chapter are very simple versions of masses and springs horizontally at rest, but possibly under tension, stretched between an anchor and a finger. Only the code absolutely necessary for this example is given here; the full code with methods and additional comments is given in force1d.a4c, an ASCEND modeling example in the library.

These models could easily be extended to include mass, momentum, and acceleration in two or three dimensions. Most of the methods in the force1d.a4c models are unedited from the code generated by the ASCEND Library button Edit->Suggest method. If you improve on these models, please share them with us and the rest of the ASCEND community.

```
REQUIRE “atoms.a4l”;
CONSTANT spring_constant REFINES real_constant DIMENSION M/T^2;
CONSTANT position_constant REFINES real_constant DIMENSION L;
ATOM position REFINES distance DEFAULT 0{m};
END position;

MODEL hook;
   F_left, F_right IS_A force;
   F_left = F_right;
   X IS_A position;
METHODS
METHOD default_self;
   (* ATOM defaults are fine *)
END default_self;
METHOD specify;
   F_right.fixed := TRUE;
END specify;
METHOD specify_float;
END specify_float;
END hook;

MODEL massless_spring(
   k IS_A spring_constant;
   h_left WILL_BE hook;
   h_right WILL_BE hook;
) WHERE (
   h_left, h_right WILL_NOT_BE_THE_SAME;
);
```

```
   L0, dx IS_A distance;
   h_right.X = h_left.X + L0 + dx;
   F = k * dx;
   h_left.F_right = F;
   h_right.F_left = F;
   F IS_A force;
METHODS
METHOD default_self;
   dx := 1{cm};
   L0 := 10{cm};
END default_self;
METHOD specify;
   L0.fixed := TRUE;
   RUN h_left.reset;
   RUN h_right.reset;
   h_left.F_right.fixed := FALSE;
   h_left.X.fixed := TRUE;
END specify;
METHOD specify_float;
   L0.fixed := TRUE;
   RUN h_left.specify_float;
   RUN h_right.specify_float;
END specify_float;
END massless_spring;

MODEL massless_block(
   h_left WILL_BE hook;
   h_right WILL_BE hook;
) WHERE (
   h_left, h_right WILL_NOT_BE_THE_SAME;
);
   width IS_A distance;
   h_left.F_right = h_right.F_left;
   h_right.X = h_left.X + width;
   X “center of the block” IS_A position;
   X = width/2 +  h_left.X;
METHODS
METHOD default_self;
   width := 3{cm};
END default_self;
METHOD specify;
   width.fixed := TRUE;
   RUN h_left.reset;
   h_left.F_right.fixed := FALSE;
   h_left.X.fixed := TRUE;
   RUN h_right.reset;
```

```
END specify;
METHOD specify_float;
   width.fixed := TRUE;
   RUN h_left.specify_float;
   RUN h_right.specify_float;
END specify_float;
END massless_block;

MODEL anchor(
   x IS_A position_constant;
   h_right WILL_BE hook;
);
   h_right.X = x;
   F = h_right.F_left;
   F IS_A force;
METHODS
METHOD default_self;
END default_self;
METHOD specify;
   RUN h_right.reset;
END specify;
METHOD specify_float;
END specify_float;
END anchor;

MODEL finger(
   h1 WILL_BE hook;
);
   pull IS_A force;
   h1.F_right = pull;
METHODS
METHOD default_self;
   pull := 3{N};
END default_self;
END finger;

MODEL finger_test;
NOTES ‘ascii-picture’ SELF {
                        ___     __
\\--O--/\/\/\/\/\/--O--|   |--O(_ \
                       |___|     \ \
(reference)-h1-(s1)-h2-(m1)-h3-(pinky)
}
END NOTES;
   reference IS_A anchor(0.0{m},h1);
   h1 IS_A hook;
   s1 IS_A massless_spring(100{kg/s^2},h1,h2);
   h2 IS_A hook;
   m1 IS_A massless_block(h2,h3);
   h3 IS_A hook;
   pinky IS_A finger(h3);
METHODS
METHOD default_self;
   RUN h1.default_self;
   RUN h2.default_self;
   RUN h3.default_self;
   RUN m1.default_self;
   RUN pinky.default_self;
   RUN reference.default_self;
   RUN s1.default_self;
END default_self;
METHOD specify;
   RUN m1.specify_float;
   RUN pinky.reset;
   RUN reference.specify_float;
   RUN s1.specify_float;
END specify;
END finger_test;

MODEL spring_test;
NOTES ‘ascii-picture’ SELF {
\\--O--/\/\/\/\/\/--O--\/\/\--O(\
(reference)-h1-(s1)-h2-(s2)-h3-(pinky)
}
END NOTES;
   reference IS_A anchor(0.0{m},h1);
   h1 IS_A hook;
   s1 IS_A massless_spring(100{kg/s^2},h1,h2);
   h2 IS_A hook;
   s2 IS_A massless_spring(75{kg/s^2},h2,h3);
   h3 IS_A hook;
   pinky IS_A finger(h3);
METHODS
METHOD default_self;
   RUN h1.default_self;
   RUN h2.default_self;
   RUN h3.default_self;
   RUN s2.default_self;
   RUN pinky.default_self;
   RUN reference.default_self;
   RUN s1.default_self;
END default_self;
```

```
METHOD specify;
   RUN pinky.reset;
   RUN reference.specify_float;
   RUN s1.specify_float;
   RUN s2.specify_float;
END specify;
END spring_test;

REQUIRE “plot.a4l”;
MODEL plot_spring_test;

   (* create our model *)
   st IS_A spring_test;

   (* Now gather the sampled data into an array for plotting *)
   x_data[Xset] ALIASES (st.h1.X,st.h2.X,st.h3.X)
   WHERE Xset IS_A set OF integer_constant;

   (* Now create the Y coordinates of the plot since there is no
    * natural Y coordinate in our MODEL.
    *)
   y_data[Xset] IS_A real; (* all will be assigned to 1.0 *)

   X_curve IS_A plt_curve(Xset,y_data,x_data);

   (* Make X_curve into the expected array for plt_plot *)
   curves[curve_set] ALIASES (X_curve) WHERE
   curve_set IS_A set OF integer_constant;

   Plot_X IS_A plt_plot_integer(curve_set,curves);
METHODS
METHOD default_self;
   RUN st.default_self;
   st.s1.L0 := 0.2{m};
   RUN X_curve.default_self;
   RUN Plot_X.default_self;
   FOR i IN Xset DO
      y_data[i] := i;
   END FOR;
   X_curve.legend := ‘meter’;
   Plot_X.title := ‘Hook locations’;
   Plot_X.XLabel := ‘location {m}’;
   Plot_X.YLabel := ‘hook #’;
END default_self;
END plot_spring_test;
```