APPENDIX D

Plot and Colex:
Utility Programs for Making Graphs

The programs described in this appendix are intended to allow the user to make simple two-dimensional graphs based on information logged while running one of the simulation models in this package.

We mention first the format of the log files. We then describe how to extract information from them using colex and how to make a graph using plot.

LOG FILE CONVENTIONS

The simulation programs log information according to specifications contained in the template file. Logging occurs only if a log file has been opened using the log command. If it has, then every time the screen is updated, all templates that are displayed are candidates for logging. A template is displayed if its dievel is positive and is less than or equal to the value of the global dievel variable.1 A displayed template is logged if the template's local dievel is less than or equal to the global dievel variable.

On each update, all of the logged variables are stored on a single line, separated from each other by spaces. Templates are considered for display and for logging in the order encountered in the template file. Within a template, values of vector templates are displayed in order; values of matrix templates are displayed in order as well, with the column index changing most rapidly. Values of look templates are displayed in the order they are specified in the look file, which is read like a matrix, with the column index

---

1 Note that templates with a dievel of 0 are only updated when the display is clear, and are never logged.
changing most rapidly. In general then, for look templates, the order in which values are displayed and logged will not necessarily correspond to the order in which they are stored internally in the program.

As an example, the template list from the jets.tem file used with the iac program looks like this:

<table>
<thead>
<tr>
<th>cycleno</th>
<th>label</th>
<th>0 5 70</th>
<th>h 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycleno</td>
<td>variable</td>
<td>1 5 75 cycleno</td>
<td>4 1</td>
</tr>
<tr>
<td>unitnames</td>
<td>label_look</td>
<td>0 5 3 uname</td>
<td>h 7 14 jets.loo</td>
</tr>
<tr>
<td>extinput</td>
<td>look</td>
<td>2 5 0 extinput</td>
<td>2 100 14 jets.loo</td>
</tr>
<tr>
<td>activation</td>
<td>look</td>
<td>1 5 10 activation</td>
<td>3 100 14 jets.loo</td>
</tr>
</tbody>
</table>

The third entry in each template specification is the dlevel. If the global dlevel is 2 and the global slevel is 1, then the cycleno, extinput, and activation templates will be displayed each time the display is updated, but only the cycleno and the activation will be logged.

Suppose that the iac program is called with this template and that the dlevel is set to 2, the slevel is set to 1, and the log command has been used to open the log file jets.log after the display is first initialized. Suppose further that the stepsizesize is set to cycle, so that the display is updated at the end of each cycle of training. Then when the program is given the cycle command, it will store a line for each cycle. Each line begins with the cycle number and then has one entry for the activation of each unit that is shown in the display. Each line is, therefore, rather long, and particular elements of the line must be extracted for further perusal or plotting.

Since the activations are displayed according to the specification stored in the jets.loo file, the order of logging of activations is given by the order in which units are encountered in the look file, which is read across each row. As examples, entry 1 (the one after the cycle number) corresponds to Jets, entry 2 corresponds to Art, entry 6 corresponds to Sharks, entry 15 corresponds to in20s, and entry 31 corresponds to the name unit for Ken.

**EXTRACTING COLUMNS TO PLOT USING COLEX**

The colex command is used for extracting columns from log files for plotting. Note that colex is a separate program, run under the command interpreter, rather than a command that is executed by the simulation programs themselves. Typically it would be used after a simulation session to plot the results of a simulation run.

The colex program is called in the following way:

```
colex infile outfile [abufile] cols
```

where infile is the name of the input file from which you wish to extract columns, outfile is the name of the output file you wish to put the extracted
columns in, abutfile (optional) is the name of a file containing lines you wish to abut the new output with, and cols is a list of column numbers counting from 0 to extract from the input file and place in the output file.

Let's say in our lac example that we have run 100 cycles of processing, logging the results as specified above. Now let's suppose we would like to see how the activation of a few of the units changed over processing cycles, say, the Sharks unit, the in20s unit, and the name unit for Ken. Before we can plot these we must extract them from the log file, and we will use colex for this purpose. As noted above, the cycleno is in column 0, and the activations of the Sharks, in20s, and Ken units are in columns 6, 15, and 31, respectively. Thus the command to pull these columns out of the file jets.log and store then in jets.clx would be:

```
coley jets.log jets.clx 0 6 15 31
```

The optional abutfile argument to colex allows the user to specify a file containing lines to which the results of the column extraction operation are to be abutted, or appended to. As an example of the use of the abut facility, suppose that we ran the lac program twice, each time with a different value of the parameter gamma, and we wanted to see how this affected the activation of the unit for in20s; and suppose we had logged the runs in run1.log and run2.log. To plot the activation curves from the two runs on the same graph we would want to get them into the same file, say, both.clx, containing lines with the cycle number and the activation of the in20s unit from each of the two runs. We can do this with colex as follows:

```
coley run1.log run1.tmp 0 15
coley run2.log both.clx run1.tmp 15
```

MAKING GRAPHS WITH PLOT

The plot program is used to make graphs from files containing lines, each containing one x-value and one or more y-values to plot against the single x-value. The x-value is always the first one in the line. The file jets.clx is an example in which there is one x-value and three y-values per line.

The plots are displayed in an array of 23 lines, each containing 79 characters followed by a newline character. The output of the program is normally displayed on the screen, but it can be dumped to a file if an optional output filename is supplied.

---

2 Note the square brackets here are used to indicate that the abutfile argument is optional and they should not be typed when running the program.
The plot program is called with the following command format:

```
plot formatfile datafile [outputfile]
```

Here `formatfile` specifies a file containing plot format instructions, the `datafile` specifies a file containing the lines of xy-values, and the optional `outputfile` specifies a file for storage of the output.

The format file consists of a number of format specifications. Each format specification consists of a line beginning with one of the following characters, each of which has the meaning indicated:

- **t** Following argument specifies a title for the plot.

- **l** Following argument specifies a label to associate with a column of y-values. There can be as many label specifications as there are y-values on each line. They are assigned to columns of y-values in the order encountered.

- **s** Following single character specifies the symbol to be plotted at each xy-location. If no symbol specification is given, the program uses the letter *a* for the first y-value and uses successive letters of the alphabet for successive y-values. Where more than one xy-pair would fall on the same column-line position, the program plots a digit indicating the number of such collisions or "*" if there are 10 or more. If a symbol is specified, all of these features are overridden and the symbol is used for plotting every point.

- **x** The rest of the line contains a specification of a parameter of the x-dimension of the plot.

- **y** The rest of the line contains a specification of a parameter of the y-dimension of the plot.

The dimension specifications are as follows:

- **M** Following argument specifies the maximum value for the dimension.

- **m** Following argument specifies the minimum value for the dimension.

- **t** Following argument specifies a title for the corresponding axis.

- **l** Indicates that natural logarithms should be taken of the values on this dimension.
As an example, the following format specification is the one used to produce the graph shown in Figure 1:

```
t Activations_for_Ken
l Sharks
l in20s
l Name
x M 110
x m 0
y M 1.0
y m -.2
x t Cycles
y t Activation
```

If this specification is stored in the file `jets.fmt`, the graph shown in the figure can be generated using the following command:

```
plot jets.fmt jets.cbx
```

To store it in the file `jets.plo`, the command would be

```
plot jets.fmt jets.cbx jets.plo
```

Some discussion of the appearance of the figure is in order. The graph has very low resolution both horizontally and vertically; it uses only 20 rows and 60 columns for the actual graph itself. Consequently, there

![Graph](image)

**FIGURE 1.** The graph showing the activation of the Sharks, in20s, and Ken units, from the file `jets.cbx`, based on a run of the iac program with external input supplied to the Ken unit.
appears to be a succession of steps in some of the curves, though in fact the underlying data are much smoother. Note also that there are many 2s and 4s; these occur whenever more than one point is to be plotted in the same row-column location of the 20×60 array.

In spite of the low resolution, the figure is somewhat revealing; it illustrates the fact that the name unit for Ken reaches asymptote very early in processing (this is due to the fact that it receives external input in this run). The other two units rise rather gradually to their asymptotic activation values of 0.50 and 0.38.