The Diffusion of Photovoltaics: Background, Modeling, Calibration and Implications for Government Policy

Gary L. Lilien
May 1978

MIT ENERGY LABORATORY REPORT - MIT-EL-78-019

PREPARED FOR THE UNITED STATES
DEPARTMENT OF ENERGY
Under Contract No. EX-76-A-01-2295
Task Order 37

Note: Charles Allen, Arnold Barnett, Pat Burns, Tom McCormick, Bruce Schweitzer, and Leif Soderberg contributed significantly to the material described in this paper.
The photovoltaics simulation system is a self-contained interactive set of computer programs on the Multics system at MIT. It allows the user to exercise the PV model for different sets of parameters by providing general facilities for creating, saving, editing, using, displaying and deleting sets of parameters. The organization of these sets of parameters reflects the model: the full set of input data is kept in a model, which contains parameters which apply across sectors, such as the cost decline factor, as well as the names of its constituent sectors. A sector, in turn, contains the data that are relevant only to itself, such as average PV installation size. Finally, when a model and its associated sectors are used to perform a simulation, the results are saved in a run, which records all the model and sector input data as well. The commands in the system are used to manipulate these three types of sets of data, or records.

The actual model is defined by the following key variables and equations:

\[ X_{it} = \text{number of kwatts of government PV installed in sector } i \text{ during time } t \]

\[ Y_{it} = \text{number of Kwatts of private PV installed in sector } i \text{ during time } t \]

\[ C_t = \text{cost of one Kwatt of PV at the end of period } t \]
\( Z_i \) = average number of Kwatts per PV installation in sector \( i \)

\( N_{it} \) = cumulative Kwatts of PV installed in sector \( i \) up to time \( t \)

\( N_t \) = cumulative production of Kwatts of PV up to time \( t \)

\( N_0 \) = Kwatts of PV installed before model begins

\( C_0 \) = cost per Kwatt of PV before model begins

\( \lambda \) = decimal factor by which cost declines when PV production doubles

\( F_{ci}(\cdot) \) = cumulative normal distribution for cost acceptability, parameterized by \( \mu_{ci} \) and \( \sigma_{ci} \)

\( F_{si}(\cdot) \) = cumulative normal distribution for successful installations acceptability, parameterized by \( \mu_{si} \) and \( \sigma_{si} \)

\( h_i \) = probability of purchase by randomly selected individual in sector \( i \) given acceptability

\( P_{i0} \) = initial potential market for PV in sector \( i \)

\( P_{it} \) = total potential market for PV in sector \( i \) at time \( t \)

\( Q_i \) = potential original equipment PV installation that becomes available per time period

\( g_i \) = growth rate of existing market potential in sector \( i \)

The equations relating these variables are:

\[
N_{it} = \sum_{\tau=1}^{t} (X_{i\tau} + Y_{i\tau})
\]
\[ \begin{align*}
N_t &= N_0 + \sum_i N_{it} \\

P_{it} &= (P_{it-1} - Q_i)(1 + g_i) + Q_i \\

1 - F_{c1}(C_{t-1}) &= \text{probability that cost is acceptable at the beginning of period in sector } i. \\
F_{s1}(N_{it-1}/Z_i) &= \text{probability that installed PV is acceptable in sector } i \text{ at the beginning of period } t. \\
Y_{it} &= (P_{it} - N_{it-1}/Z_i)(1 - F_{ci}(C_{t-1})) F_{si}(N_{it-1}/Z_i) \cdot h_i \cdot Z_i \\
C_t &= C_0(N_t/N_0)^{\log_2 \lambda}
\end{align*} \]

These equations describe the model that is embodied in the simulation system. Tables I, II and III (for models, sectors and runs, respectively) show the correspondence between the variable names used here and the names as they appear in the computer programs and on the output.

In use, the computer and the user exchange information through a computer terminal. There is a wide variety of computer terminals, which have various switches that must be properly set to communicate with Multics. After turning the terminal on, a COM/LCL switch should be on COM, a REMOTE/LOCAL switch on REMOTE, a FULL/HALF switch on HALF, a CONT MSG switch ON, and a speed switch on HIGH. Once the terminal is ready, you must "dial-up" the computer, that is, establish the telephone link between
it and your terminal. The phone number is 258-8215 for IBM 2741-like terminals and 258-8313 for ASCII-type terminals. The general procedure is to dial the number; you should get a high-pitched tone in response. Then, depending on the communication system used, push the DATA button or pull up the white button in the handset cradle or insert the handset into the acoustic coupler. On 2741-like terminals, Multics will respond with a two-line message. For ASCII-type terminals, you must type a speed identification character first, followed by a line-feed. Use "s" for 110-baud terminals (11 characters/second), "y" for 150-baud (15 cps) or "o" for 330-baud (30 cps). Multics will then respond with its two-line message. Then type

```
login SMMcCormick
```

and hit carriage return for 2741, line-feed for ASCII. You will then be prompted for the password; enter it, again followed by carriage return for 2741, line-feed for ASCII. After that, Multics will print some login messages and the simulation system's opening message will print. From this point on, everything you type on either kind of terminal should be terminated with a carriage return.

The system will ask if you need the system explained (see attached sample terminal session). If you are familiar with the system, type "no", otherwise type "yes". The system expects the full word "yes" or "no" whenever it asks you a question, though it generally assumes a "no" answer if anything other than "yes" is typed. If you answered "yes!", a brief description of the commands will be printed. In either case, the system then prints "READY" as a signal that it is awaiting your first command.
You then type in the command that you want performed. If you mistype the command or type a non-existent command, the system will respond:

The command XXX does not exist - try again

with XXX replaced by the command you typed. You should then type in a valid command. After the command has finished processing (which may require that you type in information), the system again prints "READY" and awaits your next command. This cycle is terminated when you type the command:

return

which will stop processing and automatically log you off the computer. You will be informed of this by the message:

You will now be disconnected from the computer. Goodbye.

There may be circumstances when you don't want a command to finish an action it has started. For instance, you may start displaying something that is longer than you thought. In any such situation, press the "quit" button on the terminal (which may be labeled "attn" or "break" on some terminals). Whatever the command is doing will stop, "QUIT" will be printed, and then "READY" will be printed to indicate that the next command may be entered. In certain situations an additional informational message in parentheses will be printed before "QUIT" to inform you of the status of the records that were being processed. Also, should an unforeseen error arise, an error message will be printed containing information so the system maintainer can diagnose the error, followed by an artificially created quit signal to return you to command level. Any error messages should be reported promptly to the system maintainer.
There are general conventions that apply to all commands. The format in which they are typed is the command name followed by other information, separated by blanks. For instance,

```
create model example
```

would be used to create a model named example. The commands are flexible though; in this example, "create" could be abbreviated to "cr", and any word beginning with "m" could be substituted for "model." Thus

```
create mood example
```
```
create money example
```

would all have the same effect as the previous example. In general, most commands have one or two letter abbreviations, and whenever the next field entered is a type ("model," "sector" or "run"), only the first letter is significant. All names are used as is with no abbreviation allowed. Different types of records are allowed to have the same name (i.e., you can have a model and a sector both named test), but two records of the same type must have different names. The programs check for this and give you a chance to re-enter a different name or overwrite the already existing record if you try (inadvertently or not) to use a duplicate name. You can also choose to "cancel" in such circumstances, which means that you are returned to command level as if you never entered the original command. The programs check to see if you have entered a valid record type, and prompt you if you haven't. For example:

```
create x wrong
```
is answered by:

First argument must be model or sector. Reenter:

You can then type a word beginning with "m" or "s" and create the proper record. If you realize that you don't want to create anything, you can hit "quit" to return to command level. In addition, if you leave out arguments, the programs will prompt you for them. For example:

```
cr model
```

is answered by:

Enter model name:

Also several commands allow you to process several records of the same type at once. For instance:

```
cr mood test1 test2 test3
```

will create three models.

The available commands are:

- **create** - creates models or sectors
- **copy** - copies models or sectors
- **modify** - modifies models or sectors
- **delete** - deletes models, sectors or runs
- **display** - displays models, sectors or runs
- **run** - runs a model creating a run
- **index** - lists records on file of a given type
- **help** - prints a brief description of command syntax
- **return** - terminates processing, logs you off
The last two commands are self-explanatory and have no arguments. The other seven commands are described below.

**create:**

Syntax: create TYPE NAME1 NAME2 . . .

Abbreviation: cr

TYPE: can be model or sector

NAMES: are 0 or more names of records to be created.

When this command is invoked it will successively prompt you for all the data items needed to create a model or sector. When the last item has been entered, a message is issued confirming that the record was created, then the next record (if any) is processed.

**copy:**

Syntax: copy TYPE OLDNAME NEWNAME

Abbreviation: cp

TYPE: can be model or sector

OLDNAME: name of existing record which is to be copied

NEWNAME: name that the copy is to saved under

This command copies the record OLDNAME of the indicated TYPE under the name NEWNAME and issues a confirming message. Since the use of this command is frequently to have several versions of the same record with minor changes, you will then be asked if you want to modify the new record. Answering "yes" is equivalent to issuing the command "modify TYPE NEWNAME."
modify: Syntax: modify TYPE NAME1 NAME2 ...

Abbreviation: m

TYPE: can be model or sector

NAMES: are 0 or more names of records to be modified

When invoked, modify asks for your first subcommand. A dictionary of subcommands is printed by typing "help." Most subcommands are the names of variables that can be changed. To do this, enter the variable name. The program reports back the old value and asks for the new value. Then it prompts you for the next subcommand. There are also subcommands to display the modified record and to finish processing and return to command level.

delete: Syntax: delete TYPE NAME1 NAME2 ...

TYPE: can be model, run or sector

NAMES: are 0 or more names of records to be deleted

There is no abbreviation for delete, to avoid accidental deletions. For models and runs, delete checks the index to see if they have been run and displayed, respectively, and asks you to confirm whether you really want to delete these unused records or not. A confirming message is printed after each deletion.
display: Syntax: display TYPE NAME1 NAME2 ...

Abbreviation: dp

TYPE: can be model, sector or run

NAMES: are 0 or more names or records to be displayed

This command prints the requested records on the terminal, one at a time. For a run, display asks you if you want model, sector, both or neither input variables printed (again, only the first letter is significant, so "m", "s", "b" or "n" is sufficient), and give you the same choice for output variables. Successively answering "b", "b" will print everything, "n", "n" will print nothing, "s", "s" will print only sector-specific input and output variables (which depend on model input variables however), etc.

run: Syntax: run MODELNAME RUNNAME

Abbreviation: r

MODELNAME: name of the model to be run

RUNNAME: name that the output will be saved under

After the model is run, the program prints a confirming message and asks if you would like to have the run displayed. Answering yes is equivalent to issuing the command "display run RUNNAME".

index: Syntax: index TYPE

Abbreviation: i

TYPE: can be model, sector, run or all
If TYPE is "all" or left blank, index prints a table of all records created and still on file giving their name, their type, and for models and runs, under the heading "Used?", an indication of whether they have been run and displayed yet. If TYPE is model, sector or run, a table restricted to only that type of record is printed with no type column but retaining the "Used" column.

To clarify these ideas an annotated terminal session is attached, followed by a complete listing of the program.
TABLE I: Model Correspondences

<table>
<thead>
<tr>
<th>English Concept</th>
<th>Symbolic Notation</th>
<th>PL/I Variable Name</th>
<th>Modify Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>--</td>
<td>mname</td>
<td>name</td>
</tr>
<tr>
<td>Number of Sectors</td>
<td>--</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Number of time periods</td>
<td>--</td>
<td>tmax</td>
<td>tmax</td>
</tr>
<tr>
<td>Initial cost</td>
<td>C₀</td>
<td>initcost</td>
<td>initcost</td>
</tr>
<tr>
<td>Cost decay rate</td>
<td>λ</td>
<td>costdecay</td>
<td>costdecay</td>
</tr>
<tr>
<td>Initial installed Kwatts of PV</td>
<td>N₀</td>
<td>initwatt</td>
<td>initwatt</td>
</tr>
<tr>
<td>Name of sector i</td>
<td>--</td>
<td>sn(i)</td>
<td>secname(i)</td>
</tr>
<tr>
<td>Government investment in</td>
<td>Xᵢₜ</td>
<td>goinv(i,t)</td>
<td>goinv(i,t)</td>
</tr>
<tr>
<td>sector i at time t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English Concept</td>
<td>Symbolic Notation</td>
<td>PL/I variable name</td>
<td>Modify mnemonic</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Name</td>
<td>Z.</td>
<td>sname</td>
<td>name</td>
</tr>
<tr>
<td>Average PV installation size (Kwatts)</td>
<td>Pi</td>
<td>avgsize</td>
<td>avgsize</td>
</tr>
<tr>
<td>Initial existing potential market (Kwatts)</td>
<td>Pi0</td>
<td>initpotinst</td>
<td>initpotin</td>
</tr>
<tr>
<td>Growth rate of existing potential market</td>
<td>gi</td>
<td>buildrate</td>
<td>buildrate</td>
</tr>
<tr>
<td>Potential for original equipment installation per time period (Kwatts)</td>
<td>Qi</td>
<td>newinst</td>
<td>newinst</td>
</tr>
<tr>
<td>Purchase probability given acceptance</td>
<td>hi</td>
<td>ppurch</td>
<td>ppurch</td>
</tr>
<tr>
<td>Mean of cost distribution</td>
<td>ui</td>
<td>cmean</td>
<td>costmean</td>
</tr>
<tr>
<td>Std. dev. of cost distribution</td>
<td>ci</td>
<td>cdev</td>
<td>costdev</td>
</tr>
<tr>
<td>Mean of successes distribution</td>
<td>si</td>
<td>imean</td>
<td>instmean</td>
</tr>
<tr>
<td>Std. dev. of successes distribution</td>
<td>si</td>
<td>idev</td>
<td>instdev</td>
</tr>
</tbody>
</table>
### TABLE III: Run Correspondences

The input parameter parts of a run follow Tables I and II. Model-wide output has these correspondences.

<table>
<thead>
<tr>
<th>English Concept</th>
<th>Symbolic Notation</th>
<th>PL/I Variable Name</th>
<th>Output Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>$C_t$</td>
<td>cost(t)</td>
<td>cost</td>
</tr>
<tr>
<td>Cumulative Kwatts installed by the end of $t$</td>
<td>$N_t$</td>
<td>totwatt(t)</td>
<td>totwatt</td>
</tr>
<tr>
<td>Government investment during this period</td>
<td>$\sum_i X_{it}$</td>
<td>goinvnow(t)</td>
<td>goinv</td>
</tr>
<tr>
<td>Private investment during this period</td>
<td>$\sum_i Y_{it}$</td>
<td>privinvnow(t)</td>
<td>privinv</td>
</tr>
<tr>
<td>Cumulative government investment including this period</td>
<td>$\sum_{\tau=1}^t \sum_i X_{i\tau}$</td>
<td>allgovinv(t)</td>
<td>totginv</td>
</tr>
<tr>
<td>Cumulative private investment including this period</td>
<td>$\sum_{\tau=1}^t \sum_i Y_{i\tau}$</td>
<td>allprivinv(t)</td>
<td>totpinv</td>
</tr>
<tr>
<td>English Concept</td>
<td>Symbolic Notation</td>
<td>PL/I Variable Name</td>
<td>Output Heading</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>----------------------------------</td>
<td>--------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Government investment in i at t (this is repeated from input)</td>
<td>$X_{it}$</td>
<td>goinv (i,t)</td>
<td>goinv</td>
</tr>
<tr>
<td>Private investment in i at t</td>
<td>$Y_{it}$</td>
<td>privinv(i,t)</td>
<td>privinv</td>
</tr>
<tr>
<td>Government PV installations in i at t</td>
<td>$X_{it}/(Z_i C_{t-1})$</td>
<td>govinsti(t)</td>
<td>govinst</td>
</tr>
<tr>
<td>Private PV installations in i at t</td>
<td>$Y_{it}/(Z_i C_{t-1})$</td>
<td>privinvst(i,t)</td>
<td>privinvst</td>
</tr>
<tr>
<td>Cumulative government investment in i including t</td>
<td>$\sum_{\tau=1}^{t} X_{i\tau}$</td>
<td>totgovinv(i,t)</td>
<td>totginv</td>
</tr>
<tr>
<td>Cumulative private investment in i including t</td>
<td>$\sum_{\tau=1}^{t} Y_{i\tau}$</td>
<td>totprivinv(i,t)</td>
<td>totpivn</td>
</tr>
<tr>
<td>Cumulative total investment in i including t</td>
<td>$\sum_{\tau=1}^{t} (X_{i\tau} + Y_{i\tau})$</td>
<td>totinv(i,t)</td>
<td>totinv</td>
</tr>
<tr>
<td>Cumulative government PV installations in i including t</td>
<td>$\sum_{\tau=1}^{t} X_{i\tau}/(Z_i C_{t-1})$</td>
<td>toptgovinst(i,t)</td>
<td>totpinvst</td>
</tr>
<tr>
<td>Cumulative private PV installations in i including t</td>
<td>$\sum_{\tau=1}^{t} Y_{i\tau}/(Z_i C_{t-1})$</td>
<td>totpinvst(i,t)</td>
<td>totpinvst</td>
</tr>
</tbody>
</table>
### TABLE III: Run Correspondences (Continued)

<table>
<thead>
<tr>
<th>English Concept</th>
<th>Symbolic Notation</th>
<th>PL/I Variable Name</th>
<th>Output Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative total PV installations in i including t</td>
<td>( \sum_{\tau=1}^{t} (X_{i \tau} + Y_{i \tau})/(Z_{i \tau}C_{\tau-1}) )</td>
<td>totinst(i,t)</td>
<td>totinst</td>
</tr>
<tr>
<td>Potential PV market in i at t</td>
<td>( P_{i t} )</td>
<td>potinst(i,t)</td>
<td>potinst</td>
</tr>
<tr>
<td>Cost acceptance in i at the beginning of t</td>
<td>( 1 - F_{C_{t-1}} )</td>
<td>costaccep(i,t)</td>
<td>costacc</td>
</tr>
<tr>
<td>Successes acceptance in i at the beginning of t</td>
<td>( F_{N_{i t-1}/Z_{i}} )</td>
<td>instaccep(i,t)</td>
<td>instacc</td>
</tr>
</tbody>
</table>
APPENDIX 2

FACTOR STRUCTURE COMPARISON: AGRICULTURAL SECTOR

With Choffray and Lilien's [13] modification of the Chow test, when we compare the evaluation criteria of $A^*$ against $C^*$ we must also look at the evaluation of the combined group, $(A + C)$. Actually, we are comparing $A$ and $C$ with $(A + C)$ to determine equality; our null hypothesis is that the groups are, in fact, equal. Defining:

$$C_p = N \left( \frac{1-R^2_p}{n_1(1-R^2_p) + n_2(1-R^2_p)} - 1 \right) \frac{N-2q}{q}$$

where,

- $R^2_{p1}, R^2_{p2} =$ squared multiple correlations associated with the estimation of factor $p$ in sample 1 and 2 respectively
- $R^2_p =$ squared multiple correlation associated with factor $p$ in pooled sample
- $n_1, n_2 =$ number of responses in sample 1 and 2 respectively
- $N = n_1 + n_2$
- $q =$ number of items

When the value of $C_p$ exceeds the corresponding $F$ statistic at the appropriate level of significance, the null hypothesis of equality is rejected.

* Group A = Pre-test and post-test survey
Group B = pre-test survey only
Group C = post-test survey only
APPENDIX 2 (continued)

TABLE 1

For comparison of A and C:

<table>
<thead>
<tr>
<th>Factor</th>
<th>( C_p )</th>
<th>( F .90 )</th>
<th>( F .95 )</th>
<th>( F .99 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.41</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>0.42</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>1.79</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

null hypothesis is **accepted** at the .95 level

TABLE 2

For comparison of B and C:

<table>
<thead>
<tr>
<th>Factor</th>
<th>( C_p )</th>
<th>( F .90 )</th>
<th>( F .95 )</th>
<th>( F .99 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.62</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>2.53</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>0.22</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

null hypothesis is **rejected** at the .95 level
APPENDIX 3: FACTOR STRUCTURE COMPARISON: RESIDENTIAL SECTOR

TABLE 1

COMPARISON OF GROUP 1 AND GROUP 2

<table>
<thead>
<tr>
<th>Factor</th>
<th>C_p</th>
<th>F_.90</th>
<th>F_.95</th>
<th>F_.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.76</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>10.58</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>3.34</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Null Hypothesis of equality is rejected

COMPARISON OF GROUP 3 AND GROUP 4

<table>
<thead>
<tr>
<th>Factor</th>
<th>C_p</th>
<th>F_.90</th>
<th>F_.95</th>
<th>F_.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.94</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>2.42</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>1.24</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Null Hypothesis is rejected at the .95 level
APPENDIX 3 (continued): TABLE 2

MODIFIED CHOW TEST

COMPARISON OF GROUP 1 AND GROUP 4

<table>
<thead>
<tr>
<th>Factor</th>
<th>$C_p$</th>
<th>$F_{.90}$</th>
<th>$F_{.95}$</th>
<th>$F_{.99}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.44</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>3.72</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>-8.88</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Null Hypothesis of equality is rejected

COMPARISON OF GROUP 2 AND GROUP 3

<table>
<thead>
<tr>
<th>Factor</th>
<th>$C_p$</th>
<th>$F_{.90}$</th>
<th>$F_{.95}$</th>
<th>$F_{.99}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.57</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>1.92</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>0.88</td>
<td>1.72</td>
<td>2.01</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Null Hypothesis of equality is rejected
A.4 Sample Interactive Program Session
Multics sign on messa,

1) Multics login messages

Welcome to the photo-voltaics simulation model.

Do you need an explanation of how the simulation system works (yes or no)?

Input and output data reside in one of 3 kinds of data blocks - sector, for sector-specific input data, model, for model-wide input data, and run, for output data (including the inputs which it uses). The commands for manipulating these blocks are entered as follows (where TYPE is model, sector or run and the NAMES are to be replaced with user-defined names. Alternatively, any or all parameters may be left out - you will be prompted for input):

To create models or sectors:
create
or
TYPE NAME1 NAME2 ...
cr

To copy models or sectors:
copy
or
TYPE OLDNAME NEWNAME
cp

to display models, sectors or runs:
display
or
TYPE NAME1 NAME2 ...
dp

to delete models, sectors or runs:
delete TYPE NAME1 NAME2 ...
del

to modify models or sectors:
m
or
TYPE NAME1 NAME2 ...
m

To run a model:
run
or
MODEL NAME RUNNAME
r

to get an index of data on file (all blocks will be listed if TYPE is all or blank):
index
or
TYPE
i

to reprint these instructions:
help

To stop processing and disconnect from the computer:
return

READY (Cost: $1.15) The READY message signifies that the previous action is finished and the system is awaiting your next command.

Normally the cost figure would not appear; it has been included here for illustrative purposes.
create model showoff

Enter the following for model showoff:
Number of sectors: 3
Number of time periods: 5
Initial dollar cost per Kwatt: 10000

Decimal factor by which cost declines when production doubles: .70

Initial number of Kwatts installed across all sectors: 1000
The 3 sector names: farmers houses central_power

The 5 government investment values (in M$) for sector:
farmers: 40 40 50 50 60
houses: 10 10 10 10 10
central_power: 0 0 0 0 0

This model showoff has been created.

The model showoff has been created. This message confirms that the model has been put on file.

READY (Cost: $1.25) and we're ready for our next command.

enter the following for sector farmers:
Average Kwatts per installation: 100
Initial potential old installations: 200000
Decimal rate at which potential installations grow: .03
New installations per period: 0
Decimal purchase probability given acceptance: .4
Cost per Kwatt acceptance mean and std dev: 800 300
Successful installations acceptance mean and std dev: 5 3

The sector farmers has been created. The first sector is complete.

Enter the following for sector houses:
Average Kwatts per installation: 500
Initial potential old installations: 50000
Decimal rate at which potential installations grow: .02
New installations per period: 1000
Decimal purchase probability given acceptance: .2
Cost per Kwatt acceptance mean and std dev: 500 100
Successful installations acceptance mean and std dev: 5 3

The sector houses has been created.

READY (Cost: $0.81)
run show — Now we try to run our model

The sector _central_power_ in model showoff is not in the file. Check the model definition. Run aborted.

create _central_power_ We have misspelled the name.

Enter the following for sector _central_power_:

- Average Kwatts per installation: 5000
- Initial potential old installations: 1000
- Decimal rate at which potential installations grow: .01
- New installations per period: 0
- Decimal purchase probability given acceptance: .5
- Cost per (kWt acceptance mean and std dev): 200 50
- Successful installations acceptance mean and std dev: 3 1

The sector _central_power_ has been created.

_Ready (Cost $0.48)_

r showoff — Since the name was misspelled above, run still can't find the third sector

_Ready (Cost $0.48)_

modify model showoff — So we will modify "showoff" so that it points to the sector we created; we could also have modified the name of the sector itself

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: help

Name               Description

name               name of the model
n                  number of sectors
fmax               number of time periods
initcost           initial cost per KWatt of PV
costdecay          decimal fraction by which PV cost declines when production doubles
initwatt           total Kwatts of PV installed at time 0 across all sectors
secname(i)         name of the i_th sector in the model
govinv(i,j)        government investment in sector i at time j (in M$)
quit               terminates processing, saves updated model
qdisplay           terminates processing, displays and saves updated model
display            displays updated model so far

Next request: secname(3) — We accidently typed "5" instead of "3" when you notice a typing mistake, typing "#" deletes the previous character

Subscript out of range. Request ignored.

Next request: secname(3)

Old value of secname(3) is _central_power_. Enter new value: _central_power_

Next request: secname(3) — we request secname(3) again to demonstrate that the character deletion worked properly

Old value of secname(3) is _central_power_. Enter new value: _central_power_

Next request: nonsense — we try a non-existent request
There is no variable named nonsense. Try again.
Next request: initcost

Old value of initcost is 10000.00. Enter the new value: 14000
"14000" is an invalid entry. Request ignored.
Next request: initcost

Old value of initcost is 10000.00. Enter the new value 14000

Next request: display

Model showoff

Cost per Kwatt of PV at time 0 14000.00
Decimal fraction by which cost declines when production doubles 70.00
Kwatts of PV installed at time 0 across sectors 1000.00
The model will run until time = 5, and has 3 sectors.

<table>
<thead>
<tr>
<th>Sector</th>
<th>Government Investment (In M$) at Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>farmers</td>
<td>40.00</td>
</tr>
<tr>
<td>houses</td>
<td>10.00</td>
</tr>
<tr>
<td>centralpower</td>
<td>0.00</td>
</tr>
</tbody>
</table>

READY (Cost: $1.15)
r showoff

Cost decay rate is out of range. Run aborted. ← Mis-entering the cost decay rate has finally caught up with us.

READY (Cost: $0.32)
m m showoff

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: costdecay

Old value of costdecay is 70.00. Enter the new value .7

Next request: q

READY (Cost: $0.27)
r showoff

Enter the run name: output

The model showoff has been run under the name output.
Do you want to display this run? n no

If you want to delete an entire typed line, typing "@" will do it faster than a string of "#"s.

READY (Cost: $0.54)
display run output delete r output ← We stupidly try to delete this run before it's been displayed.

The run output has not been displayed. Do you still want to delete? n no ← But the program reminds us of this

READY (Cost: $0.30)
display r output

Do you want to see model, sector, both or neither input variables? s

Same choice for output variables: both ← But we abbreviatedly ask for sector inputs only as we've displayed the model inputs above

time cost totkwatt goinv privinv totgivn totprivn
<table>
<thead>
<tr>
<th></th>
<th>goinv</th>
<th>privinv</th>
<th>goinv</th>
<th>privinv</th>
<th>totginv</th>
<th>totinv</th>
<th>totginst</th>
<th>totinv</th>
<th>totginst</th>
<th>tottinl</th>
<th>potinst</th>
<th>costacc</th>
<th>instacc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6404.47</td>
<td>4571.43</td>
<td>50.00</td>
<td>0.00</td>
<td>50.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>3815.93</td>
<td>12378.48</td>
<td>50.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>200.00</td>
<td>0.00</td>
<td>200.00</td>
<td>0.00</td>
<td>200.00</td>
<td>0.00</td>
<td>200.00</td>
</tr>
<tr>
<td>3</td>
<td>2519.41</td>
<td>28020.05</td>
<td>60.00</td>
<td>0.00</td>
<td>160.00</td>
<td>0.00</td>
<td>320.00</td>
<td>0.00</td>
<td>320.00</td>
<td>0.00</td>
<td>320.00</td>
<td>0.00</td>
<td>320.00</td>
</tr>
<tr>
<td>4</td>
<td>1835.81</td>
<td>51383.13</td>
<td>60.00</td>
<td>0.00</td>
<td>220.00</td>
<td>0.00</td>
<td>440.00</td>
<td>0.00</td>
<td>440.00</td>
<td>0.00</td>
<td>440.00</td>
<td>0.00</td>
<td>440.00</td>
</tr>
<tr>
<td>5</td>
<td>1382.30</td>
<td>89970.20</td>
<td>70.00</td>
<td>0.01</td>
<td>290.00</td>
<td>0.01</td>
<td>580.00</td>
<td>0.01</td>
<td>580.00</td>
<td>0.01</td>
<td>580.00</td>
<td>0.01</td>
<td>580.00</td>
</tr>
</tbody>
</table>

Sector farmers:

Average installation size (Kwatts): 100.00
Initial potential old installations: 200000
Decimal growth rate of potential installations: 0.03
Number of new installations per period: 0.00
Decimal probability of purchase given acceptability: 0.40
Feasible number of successes mean and std dev: 800.00
Feasible cost per Kwatt mean and std dev: 5.00

Sector houses:

Average installation size (Kwatts): 500.00
Initial potential old installations: 50000
Decimal growth rate of potential installations: 0.02
Number of new installations per period: 1000.00
Decimal probability of purchase given acceptability: 0.20
Feasible number of successes mean and std dev: 500.00
Feasible cost per Kwatt mean and std dev: 6.00

Sector central power:

Average installation size (Kwatts): 500.00
Initial potential old installations: 1000
Decimal growth rate of potential installations: 0.01
Number of new installations per period: 0.00
Decimal probability of purchase given acceptability: 0.40
Feasible number of successes mean and std dev: 200.00
Feasible cost per Kwatt mean and std dev: 3.00

Input and Output for sector "farmers"
REVEAL (Cost: $0.54) commands cannot be arbitrarily abbreviated. TYPEs can be used.

mod sec farmers <

The command mod does not exist - try again. mod sec farmers

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: help

Name Description
name name of the sector
avgsizE average number of kwatts in a PV installation
inittofin number of potential installations at time t
buildrate decimal growth rate of potential installations
newinst number of new installations per time period
cpurch decimal probability of PV purchase given acceptability
costmean mean of cost feasibility distribution
costdev std dev of cost feasibility distribution
instmean mean of successful installation distribution
instdev std dev of successful installation distribution
quit terminates processing, saves updated sector
display displays updated sector so far

Next request: ppurch

Old value of ppurch is .6. Enter the new value: .8

Next request: q

READY (Cost: $0.54) Here our command is the minimum required - one letter

Enter name of the model to be run: showoff

Enter the run name: highfarm

The model showoff has been run under the name highfarm. Do you want to display this run? yes

Do you want to see model, sector, both or neither input variables? none We've already seen the inputs

Same choice for output variables: none

Type output only in a silly way

Time cost totwatt gowinn orgwinn totgwin totgwin
1 640 4.67 4571.43 50.00 0.00 50.00 0.00
2 3839.33 12378.48 50.00 0.00 100.00 0.00
3 2579.41 23920.96 60.00 0.00 160.00 0.00
4 1839.11 51835.13 60.00 0.00 220.00 0.00
5 1382.26 (8997.92) 70.00 0.02 290.00 0.02

In this case, changing probability of purchase in sector "farmers" from .5 to .8 has resulted in an increase of only 5 kwatts in period 5.

READY (Cost: $0.61) Here we ask for an index of all records on file

Index all

Name Type Used?
Aagr model yes
The "used" column has "yes" for a model if it has been run and "no" otherwise, has "yes" for a run if it has been displayed at all, "no" otherwise, and always has "n/a" for sectors.

```
Acr  model yes
Arr  model yes
agr  model yes
pb   model yes
rr   model yes
showoff model yes
highfarmpurch run yes
output run yes
ay   sector n/a
ag1  sector n/a
ag2  sector n/a
ag3  sector n/a
ag4  sector n/a
ag5  sector n/a
ag6  sector n/a
ag7  sector n/a
centralpower sector n/a
farmers sector n/a
houses sector n/a
r    sector n/a
r1   sector n/a
r2   sector n/a
r3   sector n/a
r4   sector n/a
r5   sector n/a
r6   sector n/a
r7   sector n/a
```

READY (Cost: $0.21)
copy model showoff demonstration

The model showoff has been copied under the name demonstration.
Do you want to modify model demonstration? Y yes

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: costdecay

Old value of costdecay is 0.70. Enter the new value: 0.6

Next request: q

READY (Cost: $0.51)
index mods

All models

<table>
<thead>
<tr>
<th>Name</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aagr</td>
<td>yes</td>
</tr>
<tr>
<td>Acr</td>
<td>yes</td>
</tr>
<tr>
<td>Arr</td>
<td>yes</td>
</tr>
<tr>
<td>agr</td>
<td>yes</td>
</tr>
<tr>
<td>demonstration</td>
<td>no</td>
</tr>
<tr>
<td>pb</td>
<td>yes</td>
</tr>
<tr>
<td>rr</td>
<td>yes</td>
</tr>
<tr>
<td>showoff</td>
<td>yes</td>
</tr>
</tbody>
</table>

READY (Cost: $0.22)
run demonstration fastdecay
The model demonstration has been run under the name fastdecay.
Do you want to display this run? 1: yes

Do you want to see model, sector, both or neither input variables? 1: n

Same choice for output variables: m

time  | cost  | totwatt | privinv | privinv | totginv | totginv
-----|-------|---------|---------|---------|---------|---------
1    | 4567.65| 4571.43 | 0.00    | 50.00   | 0.00    | 50.00   |
2    | 5555.77| 3517.39 | 0.00    | 50.00   | 0.00    | 50.00   |
3    | 809.24 | 47855.23| 0.00    | 160.00  | 0.00    | 160.00  |
4    | 17.35  | 8793352.25| 0.01    | 70.21   | 0.00    | 70.21   |
5    | 7.03   | 2997536.08| 0.01    | 70.21   | 0.00    | 70.21   |

Changing the cost decay rate from .7 to .6 has caused a 300-fold increase in kwatts purchased.

READY (Cost: $0.52)

delete r fastdecay  We mistyped the name, but are given a chance to redeem ourselves.
The run fastdecay is not in the file. Reenter the name (or reply "*" to cancel) 1 fast decay.
The run fast decay has been deleted.

READY (Cost: $0.43)

All models

Name          Used
Aagr   yes
Acr    yes
Arr    yes
agr    yes

demonstration  Used has changed to "yes" for "demonstration".

do      yes
rr      yes
showoff  yes

READY (Cost: $0.20)

m m demonstration

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: n

Old value of n is 3. Enter the new value: 4

Enter the name of sector 4: farmers  this is perfectly legal and has the effect of parameterizing the new fourth sector the same as the first sector.

Enter the 5 government investment values (in M$) for this sector: 10 10 10 10 10

Next request: tmax

Old value of tmax is 5. Enter the new value: 6

Enter the 1 new government investment values (in M$) for times 6 to 6 for sector... farmers: 10

houses: 10

centralpower: 0
Next request: q

Model demonstration:

Cost per watt of PV at time 0: $14000.00

Decimal fraction by which cost declines when production doubles: 0.60

Kwatts of PV installed at time 0 across sectors: $1000.00

The model will run until time = 6, and has 4 sectors.

<table>
<thead>
<tr>
<th>Sector</th>
<th>Government Investment (in M$) at time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>farmers</td>
<td>40.00</td>
</tr>
<tr>
<td>houses</td>
<td>10.00</td>
</tr>
<tr>
<td>central power</td>
<td>0.00</td>
</tr>
<tr>
<td>farmers</td>
<td>10.00</td>
</tr>
</tbody>
</table>

We have effectively turned a 3x5 government investment matrix into a 4x6 one.

READY (Cost: $0.60)

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified. p purch

There is no variable named p purch. Try again.

Next request: cost decay

Old value of cost decay is 0.60. Enter the new value: 0.7 ← We want to use the old, same value for cost decay.

Next request: q

READY (Cost: $0.41)

r demonstration 4x6

The model demonstration has been run under the name 4x6.

Do you want to display this run? y: yes

Do you want to see model, sector, both or neither input variables? n:

Same choice for output variables:

<table>
<thead>
<tr>
<th>time</th>
<th>cost</th>
<th>totwatt</th>
<th>goinv</th>
<th>privinv</th>
<th>totginv</th>
<th>totpinv</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5934.45</td>
<td>5285.71</td>
<td>60.00</td>
<td>0.00</td>
<td>60.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>3430.36</td>
<td>15380.87</td>
<td>60.00</td>
<td>0.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>2221.38</td>
<td>35786.90</td>
<td>70.00</td>
<td>0.00</td>
<td>190.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>1605.64</td>
<td>67238.88</td>
<td>70.00</td>
<td>0.00</td>
<td>260.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>1192.52</td>
<td>119877.19</td>
<td>80.00</td>
<td>4.39</td>
<td>340.00</td>
<td>4.39</td>
</tr>
<tr>
<td>6</td>
<td>339.29</td>
<td>1412037.83</td>
<td>80.00</td>
<td>1460.93</td>
<td>420.00</td>
<td>1465.32</td>
</tr>
</tbody>
</table>

The fourth sector with its government investment has caused a significant increase in purchased kwatts in period 5.

READY (Cost: $0.51)

All records

Type Used?

Name | model | yes
Aagr | model | yes
Acr | model | yes
Arr | mode|
READY (Cost: $0.06) ← ... and we get back to command level

All runs

Name Used
4x6 yes
highfarmpunch yes
output yes

READY (Cost: $0.15)
CR's faults

Enter the following for sector faults:
Average KWatts per installation (sector faults has not been created) once again, we don't really want to create this sector, so we push QUIT instead of answering the question.
and we are reminded that any partial data entry will be lost.

QUIT

READY (Cost: $0.58)
delete r 4x6 highfarmpunch output

The run 4x6 is not in the file. ← We accidently type small "x" instead of capital "X".
Reenter the name (or reply "**" to cancel) 1 4x6

The run 4x6 has been deleted.
The run highfarmpunch has been deleted.
The run output has been deleted.

READY (Cost: $0.51)

All runs

Name Used
There are currently no runs in the file ← All our runs are deleted so they aren't taking up space and costing storage charges.

READY (Cost: $0.19) ← We are done for now, so we "return."
return

You will now be disconnected from the computer. Goodbye. ← And get this sign-off message. We have been automatically logged off the computer.
A.5 Program Listing
algorithm proc(rptr);
{
  \textbf{include run;}
dcl \{a,b,i,t\} fixed;
dcl tsqrt2 \textbf{float init}(1/sqrt(2));
  
  \textbf{totgovinv}*\{0\}=0;
  \textbf{totprivinv}*\{0\}=0;
  \textbf{totgovinst}*\{0\}=0;
  \textbf{totprivinst}*\{0\}=0;
  \textbf{potinst}*\{0\}=\textbf{initpotinst()}*\textbf{newinst}();
  \textbf{govinvnow}()*\{0\}=0;
  \textbf{privinvnow}()*\{0\}=0;
  \textbf{allgovinv}()*\{0\}=0;
  \textbf{allprivinv}()*\{0\}=0;
  \textbf{cost}()*\{0\}=\textbf{initcost}();
  \textbf{totwatt}()*\{0\}=\textbf{initwatt}();

  \textbf{do} t=1 \textbf{to} \textbf{tmax};
  \textbf{do} i=1 \textbf{to} \textbf{n};

    \textbf{govinv}(*,t)=\textbf{max}(0,\textbf{govinv}(*,t));
    \textbf{costaccept}(*,t)=5*\{1+erf(-tsqrt2*\{cost(t-1)\-\textbf{cmean}(i)\}/\textbf{cdev}(i))\};
    \textbf{instaccept}(*,t)=5*\{1+erf(tsqrt2*\{\textbf{totinst}(i,t-1)\-\textbf{tmean}(i)\}/\textbf{tdev}(i))\};
    \textbf{privinst}(*,t)=\textbf{max}(0,\{\textbf{potinst}(i,t)\-\textbf{totinst}(i,t-1)\}*\textbf{costaccept}(i,t)\*\textbf{instaccept}(i,t)\*\textbf{ppurch}(i)\*\textbf{avsize}(i)*\textbf{cost}(t-1));
    \textbf{govinst}(i,t)=\textbf{govinv}(i,t)/\textbf{avsize}(i)*\textbf{cost}(t-1);
    \textbf{privinst}(i,t)=\textbf{privinv}(i,t)/\textbf{avsize}(i)*\textbf{cost}(t-1);
    \textbf{totgovinv}(i,t)=\textbf{totgovinv}(i,t-1)+\textbf{govinv}(i,t);
    \textbf{totprivinv}(i,t)=\textbf{totprivinv}(i,t-1)+\textbf{privinv}(i,t);
    \textbf{totgovinst}(i,t)=\textbf{totgovinst}(i,t-1)+\textbf{govinst}(i,t);
    \textbf{totprivinst}(i,t)=\textbf{totprivinst}(i,t-1)+\textbf{privinst}(i,t);
    \textbf{totinst}(i,t)=\textbf{totinst}(i,t-1)+\textbf{privinst}(i,t);
    \textbf{totprivinv}(i,t)=\textbf{totprivinv}(i,t-1)+\textbf{privinv}(i,t);
    \textbf{govinvnow}(t)=\textbf{govinvnow}(t)+\textbf{govinv}(i,t);
    \textbf{privinvnow}(t)=\textbf{privinvnow}(t)+\textbf{privinv}(i,t);
    \textbf{totwatt}(t)=\textbf{totwatt}(t)+\textbf{govinv}(i,t)+\textbf{privinv}(i,t)/\textbf{cost}(t-1);
  \textbf{end};

  \textbf{cost}(t)=\textbf{initcost}*(\textbf{totwatt}(t)/\textbf{initwatt})^{*\{\textbf{log2}(%textbf{costdecay})\}};
  \textbf{allgovinv}(t)=\textbf{allgovinv}(t-1)+\textbf{govinvnow}(t);
  \textbf{allprivinv}(t)=\textbf{allprivinv}(t-1)+\textbf{privinvnow}(t);

\textbf{end};

and;
dcl 01 model based(mptr),
   02 mname char(30) varying,
   02 n fixed,
   02 tmax fixed,
   02 initcost float,
   02 costdecay float,
   02 initwatt float,
   02 sections refer (n),
   03 id char(30) varying,
   03 goinv(tm refer (tmax)) float;

dcl getarg entry(fixed) returns(char(30) varying aligned);

dcl (key, any_other, quit, conversion) condition;

dcl (oncode, onloci builtin);

dcl 01 sector controlled,
   02 sid char(30) varying,
   02 avgsze float,
   02 initsp inst float,
   02 buildrate float,
   02 newinst float,
   02 purhch float,
   02 costp,
   03 cmean float,
   03 cdev float,
   02 instp,
   03 imean float,
   03 idev float;

dcl 01 index,
   02 iname char(31),
   02 type char(6),
   02 used char(3);

dcl (sysin, sysprint, (sectile, modfile, indfile) keyed) file;

dcl (idx, i, ns, tm) fixed;

dcl mptr ptr;

dcl aname char(30) varying aligned;

dcl rtype char(1);

dcl fnam char(30) varying aligned;

dcl nargs fixed external static;

dcl kflag bit(1);

dcl onsource builtin:

on conversion begin;
   put skip list(""""""");
   delete file(indfile) key(iname);
   go to back;
   end:

on key(indfile) kflag=","";

aname=getarg(2);

do while (aname="")
   put skip list("Enter "model" or "sector" : ");
   get list(aname);
   end;

rtype=substr(aname,1,1);

do while ("(rtype="m" "rtype="s")")
   put skip list("First argument must be model or sector. Reenter : ");
   get list(aname);
   rtype=substr(aname,1,1);
   end;

if rtype="m" then do;
```c

```
get * [tm];
alloc e model;
put skip list("Initial dollar cost per Kwatt ");
get list(initcost);
put skip list("Decimal factor by which cost declines when production doubles ");
get list(costdecay);
put skip list("Initial number of Kwatts installed across all sectors ");
get list(initkwatt);
put edit("The "+ns," sector name ") (skip,a,f(2),a);
get list(sec(*),ID);
put edit("The "+tm," government investment values (in M$) for sector") (skip,a,f(3),a);
push skp;
do i=1 to ns;
put edit(sec(i),ID," ");(skip,a,a);
got list(sec(i),govinv(*));
govinv(i,*)=govinv(i,)*le6;
end;
nname=tname;
write file(modfile) key(ename) from(model));
free model;
end;
put skip list("The "+lname1l ""llname2l has been created.");
back End;
close file(indfile),file(modfile),file(sectfile);
return;
error revert cond(quit),cond(any_other);
push skip list("l+lname1l ""llname2l has not been created ");
on key(indfile);
delete file(indfile) key(ename);
if rtype="m" then do;
on key(modfile);
delete file(modfile) key(mname);
end;
else do;
on key(sectfile);
delete file(sectfile) key(sid);
end;
signal cond(quit);
end;
copy.proc

dcl 01 sector based(mtr),
   02 sname char(30) varying,
   02 avgsize float,
   02 initpotinst float,
   02 buildrate float,
   02 newinst float,
   02 oourn float,
   02 cost0,
   03 cmean float,
   03 cdev float,
   02 instp,
   03 imean float,
   03 idev float;

dcl 01 model based(mtr),
   02 sname char(30) varying,
   02 fixed,
   02 tmax fixed,
   02 initcost float,
   02 costdec float,
   02 initwatt float,
   02 section refer(n),
   03 id char(30) varying,
   03 goto(m refer(tmax)) float;

dcl mtr otr;

dcl 01 index,
   02 iname char(31),
   02 type char(5),
   02 used char(3);

dcl (oidkey,newkey,aname) char(30) varying;

dcl file file variable;

dcl rtype char(1);

dcl (modfile,sectfile,indfile,sysin,sysprint) file;

dcl (key,quit,any_other) condition;

dcl (oncode,onloc) builtin;

dcl getarg entry(fixed) returns(char(30) varying aligned);

dcl mproc variable entry(ptr);

dcl (mmod,msec) entry(ptr);

aname=getarg(2);
if aname="m" then do
   put skip list("Is a model or a sector to be copied? ");
   get list(aname);
   rtype=substr(aname,1,1);
endif type=substr(aname,1,1);
do while("(rtype=an "r=type="s"))"
   put skip list("First argument must be model or sector. Reenter ");
   get list(aname);
   rtype=substr(aname,1,1);
end:
if rtype="m" then do
   aname="model"
   file=modfile
   used="no"
   mproc=mmod;
else do
   aname="sector"
   file=sectfile;
used="",k;
smoc=mscc;
end;
oldkey=getarg(3);
if oldkey="" then do:
put skio list("Enter the name of the "llname!!") to be copied : ");
get list(oldkey);
end;
on cond(qquit) go to error;
on cond(any_other) begin;
put skio edit("Copy error - onloc is "lllocloc!", oncode is ",oncode){a,f(i2)};
go to error;
end;
open file(filet) keyed direct update;
on key(filet) begin;
put skio list("The "llname!!") "lloldkey!!") is not in the file.");
put skio list("Reenter the name for the copy of "llname!!") "lloldkey!!") : ");
get list(oldkey);
if oldkey="" then go to back;
go to retrin;
end;
refini read file(filet) key(oldkey) sat(mptr);
newkey=getarg(4);
if newkey="" then do:
put skio list("Enter the name for the copy of "llname!!") "lloldkey!!") : ");
get list(newkey);
end;
open file(indfile) keyed direct update;
on key(indfile) begin;
put skio list("There is already a "llname!!") called "llnewkey!!") in the file.");
put skio list("Reenter the name for the copy of "llname!!") "llnewkey!!") : ");
get list(newkey);
if newkey="" then go to back;
go to retry;
end;
type=aname;
retry! iname=rtypelikeynewkey;
write file(indfile) from(index) keyfrom(iname);
if rtype="m" then do;
amanewkey;
write file(modfile) from(model) keyfrom(mname);
end;
else do:
amanewkey;
write file(seccfile) from(sector) keyfrom(sname);
end;
put skio list("The "llname!!") "lloldkey!!") has been copied under the name "llnewkey!!");
revert cond(qquit),cond(any_other);
put skio list("Do you want to modify "llname!!") "llnewkey!!") ? 1 ");
get list(oldkey);
if oldkey="yes" then call mproc(mptr);
back! close file(filet), file(indfile);
return;
error! revert cond(qquit),cond(any_other);
put skio list(""llname!!") "llnewkey!!") has not been saved");
on key(indfile);
delete file(indfile) key(iname);
delete proc

dcl (badkey, aname) char(30) varying;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl rtype char(1);
dcl (logfile, secfile, runfile, indfile, sysin, sysprint) file;
dcl file file variable;
dcl verb char(9) varying;
dcl nargs fixed external static;
dcl key condition;
dcl (1 index,
  01 index,
    02 iname char(31),
    02 type char(6),
    02 used char(3));

on key(indfile) begin;
  put skip list("The "llname=" "llbadkey=" is not in the file.");
  put skip list("Reenter the name (or retype "**" to cancel) :");
  get list(badkey);
  if badkey="**" then go to back;
go to retry;
end;

aname= getarg(2);
if aname="" then do;
  put skip list("Is a modal, sector or run to be deleted? : ");
  get list(aname);
end;
rtype=strrtrim(aname, 1, 1);
do while (~rtype="m" ~rtype="s" ~rtype="r") ;
  put skip list("First argument must be model/sector/run. Reenter : ");
  get list(aname);
  rtype=strrtrim(aname, 1, 1);
end;
if rtype="m" then do;
  verb="run";
  aname="model";
  file=logfile;
end;
else if rtype="s" then do;
  aname="sector";
  file=secfile;
end;
else do;
  verb="displayed";
  aname="run";
  file=runfile;
end;
open file(file) keyed direct update;
open file(indfile) keyed direct update;
for i=3 to max(3,nargs);
  badkey=getarg(1);
  if badkey="" then do;
    put skip list("Enter the name of the "llname=" to be deleted : ");
    get list(badkey);
  end;
retry: if rtype="s" then do;
  read file(indfile) into(index) key(rtype1, badkey);

if used="r" then do:
    put skip list("The "llnamemiss" "lbadkey!!" has not been "llver..."). Do you still want to delete? ": "
    get list(type)
    if type="yes" then do:
        go to back:
    end;
end;
delete file(indfile) key(type||badkey):
delete file(file?) key(badkey):
puzzle list("The "llnamemiss" "lbadkey!!" has been deleted.");
back:
end;
close file(indfile),file(file?):
end;
%include run;
dcl (list,a,b) fixed;
dcl (sysin,sysprint,indfile) file;
dcl 01 index,
  02 line char(31),
  02 type char(6),
  02 used char(3);
dcl (anso,ansl) char(10);
dcl (dmod,dscc) entry(ptr);

open file(indfile) keyed direct update;
read file(indfile) key("r" lname) into(index);
used="yes";
rewrite file(indfile) key(lname) from(index);
close file(indfile);

do skip list("Do you want to see model, sector, both or neither input variables? ")
  get list(ansl);
  ans1=str(ans1,1,1);
do skip list("Same choice for output variables!")
  get list(anso);
  ans0=str(ans0,1,1);
if ans1="s" ans1="b" then call dmod(addr(model))
if ans0="s" ans0="b" then do:
do skip list("time  cost  totwatt goinvv privinvv totginv  totpinv ");
do skip edit(t, cost(t), totwatt(t), goinvvnow(t)*1e-6, privinvvnow(t)*1e-6, allgovinv(t)*1e-6, allprivinvv(t)*1e-6
  do t=1 to tmax)(skip,(3),(6))(f(12),2));
do skip;
do list=
if ans1="s" ans1="b" then call dscc(addr(sector(1)));
if ans0="s" ans0="b" then do:
do skip list("Sector " llsname(1) " "")
do skip;
do list=
do skip edit("time  goinvv  privinvv  goinvst  privinvst  totginv  totpinv  totinv  totginst  totpinst  totinvst  totinst")
  put edit("doptinst costacc instacc")
  put skip edit(t, goinvv(t)*1e-6, privinvv(t), goinvst(t), privinvst(t), totginv(t)*1e-6, totpinv(t)*1e-6, totinv(t)*1e-6, totginst(t), totpinst(t), totinvst(t), costacc(t), instacc(t), t)
  do t=1 to tmax)(skip,(3),(11))(f(10),2),(2)(f(7),4))
do skip;
do list=
do end;
dsect proclptr;

dcl 01 sector based(sptr),
  02 sname char(30) varying,
  02 avgsize float,
  02 initpotinst float,
  02 buildrate float,
  02 newinst float,
  02 opurch float,
  02 costf,
  03 cmn mean float,
  03 cdev float,
  02 insto,
  03 cmn mean float,
  03 cdev float;

dcl sptr attr;

dcl sysprint file;

out skip list("Sector "sname1")1;

out skip;

out skip edit("Average installation size (Kwatts)",avgsize)a,col(67),f(11,2));

out skip edit("Initial potential old installations",initpotinst)a,col(67),f(11,2));

out skip edit("Decimal growth rate of potential installations",buildrate)a,col(67),f(11,2));

out skip edit("Number of new installations per period",newinst)a,col(67),f(11,2));

out skip edit("Decimal probability of purchase given acceptability",purch)af,col(67),f(11,2));

out skip edit("Feasible cost per Kwatt mean and std dev",costf)a,col(67),f(11,2)x(2));

out skip edit("Feasible number of successes mean and std dev",insto)a,col(67),f(11,2)x(2));

out skip;

return;

end: entry(mptr);

dcl (list) fixed;

dcl 01 model based(mptr),
  02 sname char(30) varying,
  02 fix fixed,
  02 fmax fixed,
  02 costf float,
  02 costdec float,
  02 initwatt float,
  02 sectins refer(n);
  03 id char(30) varying,
  01 ginvitm refer(tmax) float;

dcl mptr attr;

out skip list("Model "sname1")1;

out skip;

out skip edit("Cost per Kwatt of PV at time 0",initcost)a,col(67),f(11,2));

out skip edit("Kwatts of PV installed at time 0 across sectors",initwatt)a,col(67),f(11,2));

out skip edit("The model will run until max",fimax)af,col(3),af(3),a;

out skip(2) edit("Sector","Government investment (in HS) at time "Name","f do f=1 to fmax),
  (id(1))+(givinv(1+f)*f=0 do f=1 to fmax) do i=1 to n)
  +a,col(31),a,skp,a,(floor(fmax/10))col(35),10(f(3)x(6)),skp,skp,skp
  (min(1,mod(fmax,10))col(35),mod(fmax,10))f(3)x(6),skp,skp,skp
  (n)(a,(floor(fmax/10))col(31),10f(9,2),skp,skp,skp
  (min(1,mod(fmax,10))col(31),mod(fmax,10))f(9,2),skp));

out skip;

return;

end:
INDEX { index 11...0; 
dcl 01 index;
  01 name char(31);
  02 type char(b);
  02 used char(3);

dcl (sysin,sysprint,indfile) file;

dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl rtype char(1);
dcl arg char(30) varying aligned;
dcl flag put(1) init('"b'2);
dcl dum char(31);
dcl endfile cond;

dcl getarg(2);
if arg = "" then rtype = "a":
else rtype = substr(arg,1,1);
do while ("-" not rtype = "m" not rtype = "s")
  put skip list("First argument must be model/sector/run/all. Reenter:");
  get list(arg);
  rtype = substr(arg,1,1);
end;

donfile(indfile) keyed sequential input;
donfile(indfile) go to out;
if rtype = "a" then do:
  arg = "record";
  put skip list("All records");
  put skip edit("Name","Used?")(a,col(31),a);
  put skip;
  do while(1"b);
    read file(indfile) into(index) keyto(dum);
    put skip edit(substr(name,2,30),type,used)(a,col(31),a,col(38),a);
    flag = "b";
  end;
end;
else do:
  if rtype = "m" then arg = "model";
  if rtype = "s" then arg = "sector";
  if rtype = "r" then arg = "run";
  put skip list("All "arg11"s");
  put skip edit("Name","Used") (a,col(31),a);
  put skip;
  do while(1"b);
    read file(indfile) into(index) keyto(dum);
    if type = arg then do:
      put skip edit(substr(name,2,30),used)(a,col(31),a);
      flag = "b";
    end;
  end;
end;
out if "flag then put skip list("There are currently no "flag11"s in the file");
put skip;
close file(indfile);
end;
int buf1, pr

dcl buf(140) static char(1);
dcl tuf char(140) defined(buf);
dcl nargs fixed static external;
dcl code fixed bin(35);
dcl iptr static ptr;
dcl tab(30) fixed static;
dcl lox_find_locb entry(char(*),iptr,code);

nargs=0;
call lox_find_locb("user_input",iptr,code);
return;

new line entry;
dcl lox_get_line entry(iptr,ptr,code,fixe d bin(21),fixed bin(21),fixed bin(35));
dcl i fixed init(0);
dcl sw bit(1) init("0"b);
dcl sysprint file;
dcl i fixed;
dcl lbuf fixed bin(21);
lbuf=1;
do while(lbuf=1);
call lox_get_line(iptr,addr(buf),140,lbuf,code);
end;

buf(lbuf)=""

do i=1 to lbuf;
  if bool((buf(i)=" &buf(i)=" "") sw, "0110"b) then do:j=i+1;
  if j>30 then do:
    i=lbuf;
    j=30;
    put list("Only the first 15 fields have been recognized - the rest have been ignored.");
    put skip;
  end;
  else do:
    sw=¬sw;
    tab(j)=1;
  end;
end;

nargs=1/2;
return;

getarg: entry(argind) returns(char(30) varying aligned);
dcl argind fixed;
if argind=0 & argind<nargs
  then return (substr(tuf,tab(2*argind-1),min(30,tab(2*argind-1))));
else return ("");

afarg: entry(num);
dcl num char(*);
dcl cu_saf_return_arg entry(fixed bin(17),ptr,code,fixe d bin(17),fixed bin(35));
dcl sreturn char(is) varying based(sptr);
dcl sptr ptr;
dcl is fixed bin(17);
call cu_saf_return_arg(1,sptr,is,code);
sreturn=safarg(fixed(num));
end;
mmod pr (mtr)

dcl 01 model based(mtr),
  02 name char(30) varying,
  02 n fixed,
  02 max fixed,
  02 var(315) float,
  02 secns refer(n),
  02 sid char(30) varying,
  03 gov,nu(tm refer(tmax)) float;

dcl (mtr,newp) ptr;

dcl (modfile,indfile,sysin,syout,term) file;

dcl (i,j,k,num,ns,tm) fixed;

dcl rep char(30) varying;

dcl (conversion,key) condition;

dcl onsource builtin;

dcl ans char(10);

dcl 01 index,
  02 name char(31),
  02 type char(6),
  02 used char(31);

dcl tabr(5) char(9) init("n","max","initcost","costdecay","initwatt");

dcl dmod entry(ptr);

dcl cu_sco entry(ptr, fixed, fixed bin(35));

dcl code fixed bin(35);

dcl line char(50) init("lo putchars user_output =sm mod,help");

dcl flag bit(1) init("1"b);

dcl bad char(30) varying;

on conversion begin;
  if onsource="" then bad=rep;
  else bad=onsource;
  put skip list(""""""""bad"""""""" is an invalid entry. Request ignored.""");
  go to endloop;
end;

on key(indfile) begin;
  put skip list("There is already a model in the file called """"1"""".");
  if ans="""" then do;
    get list(ans);
    if ans="""" then do;
      go to endloop;
    end;
    else if ans="d" then do;
      delete file(indfile) key(name);
      delete file(modfile) key(rep);
      go to retry;
    end;
    else do:
      put skip list("Enter the new name: ");
      read file(term) into(rep);
      go to retry;
  end;
end;

open file(term) environment(stringvalue) record input title("record_stream user_input");

put skip list("Enter """"help"" for a dictionary of variables, """"quit"" to return, or the name of the variable to be modified: ");

do while(flug);
  read file(term) into(rep);
  if rep="help" then call cu_sco(add(line),length(line),code);
  else if substr(rep,1,1)="q" then do;
flag = 1;
if substr(rep,2)="display" then call dmod(mtr);
end;
else if rep="display" then call dmod(mtr);
else if rep="name" then do;
    put skip list("The old name is "Ilmnameli. Enter the new name: ");
read file(term) into(repl);
open file(indfile) keyed direct update;
read file(indfile) key(\"m\"Ilmname) into(index);
retry: name="m\"Ilrep;
write file(indfile) keyfrom(index) from(index);
delete file(indfile) key(\"m\"Ilname);
close file(indfile);
delete file(modfile) key(mname);
mname=rep;
write file(modfile) keyfrom(mname) from(model);
end;
else do;
do i=1 to 5 while(mtag(i)\~rep);
end;
if i=6 then do;
    if length(rep)+1\<\min(length(rep),6)="govinv" then do;
        if length(rep)+1\<\substr(rep,7,1)="\"Ilsubstr(rep,length(rep))\"=\"\" then signal conversion;
            k=\index(substr(rep,9)\",\")
        if k=0 then signal conversion;
            i=decimal(substr(rep,\8k))
        j=decimal(substr(rep,9-k,length(rep)-9-k))
        if i11>n11+1\<max then put skip list("Subscripts out of range. Request ignored.\")
        else do;
            put skip edit("Old value of "Ilrep1 is ",govinv(i11)*1e-6," Enter new value ",(a,f(9.2),a);
             get list(govinv(i11));
             govinv(i11)=govinv(i11)*1e6;
        end;
    end;
    else if length(rep)+1\<\min(length(rep),7)="secname" then do;
        if length(rep)+1\<\substr(rep,8,1)="\"Ilsubstr(rep,length(rep))\"=\"\" then signal conversion;
            i=decimal(substr(rep,9)\")
        if i11=n11+1\<max then put skip list("Subscript out of range. Request ignored.\")
        else do;
            put skip list("Old value of "Ilrep1 is ",isid(i11)\", Enter new value ");
             get list(isid(i11));
        end;
    end;
else put skip list("There is no variable named "Ilrep1. Try again.\")
end;
else do;
    if i1\<3 then do;
        if i=1 then new=n;
        else new=tmx;
        put skip edit("Old value of "Ilrep1 is ",new," Enter the new value ",(a,f(3),a);
             get list(new);
        if i=1 then do; ns=new; tm=tmx; end;
        else do; tm=new; ns=n; end;
        allocate model set(newp);
        newp\>var(*)=var(*);
        newp\>mname=mname;
do i=1 to min(n,ns);
        newp\>sid(i)=sid(i);
do j=1 to min(tm,tm);
        newp\>govinv(i11)=govinv(i11);
    end;
end;
L

4...

0

a

L

0

E

ID

O

X

L

0

0 O E >E

4-

L

E

L

C

0.c

C

C

i

4-

A

C C

- +

(V

4,, 4- 4 .

I / C -,.

C C .

_-

4*-

L

C

X O

E~f ,-

*-

m

A

L

X O

LO .

F

O>

E

@ 0

X

4-

_ I / C-,.

4

X O

E~f ,-

*-

m

A

L

X O

LO .

F

O>

E

@ 0
modify m; display do; proc:
dcl filc file variable;
dcl procl(entryptr) variable init(dmod, mmod, dsec, msec, drun);
dcl (sysin, syout, modfile, secfile, runfile) file;
dcl (dmod, dsec, drun, mmod, msec) entry(ptr);
dcl (anam, tnam) char(30) varying;
dcl name char(30) varying;
dcl rname char(5) varying;
dcl rtype char(1);
dcl rep char(5) init("yes");
dcl narys fixed external static;
dcl sptr ptr;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl (key, suit, any_other) condition;
dcl (onloc, oncode) built in;
dcl (i, j, p) fixed
if.substr(j, getarg(1), 1, 1) = "d" then do;
   qname = "run, ";
   pname = "displayed";
   j = 0;
end;
else do:
   qname = "";
   pname = "modified";
   p = 1;
end;
aname = getarg(2);
if aname = "" then do:
   put skip list("Is a " llname1" model or sector to be "llname2"? 1");
get list(aname);
end;
rtype = substr(aname, 1, 1);
do while("" (rtype = "m" | rtype = "r" | rtype = "s") (rtype = "r" | rtype = "s"));
   put skip list("First argument must be " llname1" model or sector. Reenter: ");
get list(aname);
   rtype = substr(aname, 1, 1);
end;
if rtype = "r" then do:
   aname = "run";
   filc = runfile;
   j = 5;
else if rtype = "m" then do:
   aname = "model";
   filc = modfile;
   j = p + 1;
else do:
   aname = "sector";
   filc = secfile;
   j = p + 3;
end;
end;
on cond(quit) go to error;
on cond(any_other) begin;
    put skip ed!("Mod/disb error - onloc is "lonloc"
        ", oncode is ",oncode(a,f[12]));
go to error;
end;
open file(filet) keyed direct update;
do i=3 to max(3,nargs);
    fnam=getarg(i);
    if fnam="" then do;
        put skip list("Enter the name of the "llnamell" to be "llnamell": ");
        get list(fnam);
    end;
retry; read file(filet) key(fnam) set(sptr);
call proct4p(sptr);
end;
back; close file(filet);
return;

error revert cond(quit),cond(any_other);
if qname="" then put skip list("(modifications may not have been saved): ");
signal cond(quit);
end;
msect proc(s...

  dcl 01 sector based(sptr),
        02 name char(30) varying,
        02 var(9) float;
  dcl stab(9) char(9) init("avgsiz", "inittot", "bullrate", "newinst", "ppurch", "costmean", "costdev", "instmean", "instdev"):  
  dcl sofr ptr;  
  dcl rep char(30) varying;  
  dcl ans char(10);  
  dcl i fixed;  
  dcl (key, conversion) condition;  
  dcl (indfile, secfile, sysin, sysprint) file;  
  dcl 01 index,
        02 iname char(31),
        02 type char(6),
        02 used char(3);  
  dcl onsource builtin;  
  dcl dsec entry(ptr);  
  dcl cu__so entry(ptr, fixed, fixed bin(35));  
  dcl code fixed bin(35);  
  dcl flag bit(1) init("1"b);  
  dcl line char(50) init("io put_chars user_outout -sm sec_help");

  on conversion begin;
    put skip list("""
    "error! "lnonsource!" is an invalid numeric entry. Request ignored."");
    go to endloop;
  end;
  on key(indfile) begin;
    put skip list("There is already a sector in the file called "!rep!".");
    put skip list("Do you want to overwrite/delete it(d), enter a new name(n), or cancel(*)? ":");
    get list(ans);  
    if ans="*" then do;
      go to endloop;
    end;
    else if ans="d" then do;
      delete file(indfile) key(iname);
      delete file(secfile) key(rep);
      go to retry;  
    end;
    else if ans="n" then do;
      put skip list("Enter the new name: ");
      get list(rep);
      go to retry;  
    end;
    end;

  put skip list("Enter ""help"" for a dictionary of variables, ""quit"" to return, or the name of the variable to be modified: ");
  do while(flag);
      get list(rep);
      if rep="help" then call cu__so(address(line), length(line), code);
      else if substr(rep, 1)="d" then do;
        flag=0"b;
        if substr(rep, 2)="display" then call dsec(sptr);
      end;
      else if rep="display" then call dsec(sptr);
      else if rep="name" then do;
        open file(indfile) keyed direct update;
        read file(indfile) key("s" llname into(index));
        put skip list("The old name is "llname!. Enter the new name: ");
        get list(rep);
retry: l {e="s" || rep;
    write file(indfile) keyfrom(index) from(index);
    delete file(indfile) key("s"||sname);
    close file(indfile);
    delete file(secfile) key(sname);
    name=rep;
    write file(secfile) keyfrom(sname) from(sector);
    end:
    else do;
        do 1=1 to 11 while(stab(k)="rep");
        end:
        if 1<12 then do;
            put skip edit("Old value of "||rep||" is",var(l),". Enter the new value ") (a,f(9,2),a);
            get list(var(l));
            end:
            else put skip list("There is no variable named "||rep||". Try again.");
            end:
        endloop if flag then put skip list("Next request: ");
        end:
    rewrite file(secfile) key(sname) from(sector);
    end:
run rt
#include run;
dcl 01 mod based(mptr),
  - 02 nn char(30) varying,
    02 a fixed,
    02 b fixed,
    02 c float,
    02 d float,
    02 e float,
    02 s(ns refer(a)),
    03 sid char(30) varying,
    03 g(fm refer(b)) float;
dcl mptr otr;
dcl {runfile,modfile,secfile,indfile,srsin,srsprint} file;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl aname char(30) varying;
dcl {key,quit,any_other} condition;
dcl rep char(9);
dcl drun entry(ptr);
dcl algor entry(ptr);
dcl {onkey,ontoc,oncode} builtin;
dcl {i,t} fixed;
dcl 01 index,
  02 lname char(31),
  02 tyou char(6),
  02 used char(3);
dcl 01 rindex,
  02 rname char(31),
  02 rtype char(6),
  02 rused char(3);

aname=getarg(2);
if aname="" then do;
  put skip list("Enter name of the model to be run ":");
  get list(aname);
end;

open file(indfile) keyed direct update;
on {key} indfile begin;
  if onkey="n"lname then do;
    put skip list("The model " lname"" is not in the file.");
    put skip list("Reenter the name (or reply \"\"\"\" to cancel) ":");
    get list(aname);
    if aname="" then go to back;
  go to read;
end;
else do;
  put skip list("There is already a run named " rname"" in the file.");
  put skip list("Do you wish to overwrite/delete it(d), enter a new name(n), or cancel(*)? ":");
  get list(rep);
  if rep="d" then go to back;
  if rep="n" then do;
    put skip list("Enter the new name ":");
    get list(rname);
    go to retry;
end;
read file(indfile) into(rindex) key("n"rname);
if rused="yes" then do;
  put skip list("The run \"rname\" has not been fully displayed.");
  put skip list("Do you still wish to delete it? ":");
ge \}list(rep);  
  if rep="yes" then go to back;  
end;  
rused="no";  
rewrite file(indfile) from(rindex) key("r1"rname);  
delete file(runfile) key(rname);  
go to start;  
end;  
end;

open file(modfile) keyed direct input;  
modread: read file(indfile) into(index) key("m1"laname);  
used="yes";  
read file(modfile) key(aname set(mptr));  
allocate run;  
model=mod;

open file(secfile) keyed direct input;  
on key(secfile) begin;  
  put skip list("The sector "l1onkey11" in model "l1aname11" is not in the file.");  
  put skip list("Check the model definition. Run aborted.");  
go to back;  
end;

do i=1 to n;  
read file(secfile) key(sid(i)) into(sector(i));  
  if pos(avgsize(i),"Average size")pos(initinfostat(i),"Pot. inst.");  
  rates(orpurch(i),"Purchase")rates(buildate(i),"Pot. installation")rates(cdev(i),"Cost st dev")rates(ldev(i),"Inst st dev") then go back;  
end;

if pos(initcost,"Initial cost")pos(initwatt,"Initial watts")rates(costdecay,"Cost decay") then go to back;

rates1 proc(rate,descr) returns(bit(1));  
dcl rate float;  
dcl descr char(19) varying;  
if rate<0 then return("0\"b");  
put skip list(descr"") rate is out of range. Run aborted.");  
return("1\"b");  
end;  

post proc(var,descr) returns(bit(1));  
dcl var float;  
dcl descr char(13) varying;  
if var>=0 then return("0\"b");  
put skip list(descr"") is not positive. Run aborted.");  
return("1\"b");  
end;

rname=getarg(3);  
if rname="" then do;  
  put skip list("Enter the run name : ");  
  get list(rname);  
end;  
rtype="run";  
rused="no";

on condiquit) go to error;  
on condi(any_other) begin;  
  put skip edit("Run error - onloc is "l1onloc11", oncode is ",oncode)(a,7f(12));
go to error;
end;

retry: rename="r"llrname;
write file(indfile) from(index) keyfrom(rname);
start: rewrite file(indfile) from(index) key(rname)

call algo(rptr);

open file(runfile) keyed direct update;
write file(runfile) from(-un) keyfrom(rname);
out skip list("The model "llrname" has been run under the name "llrname".");
revert cond(quit),cond(any_other);
out skip list("Do you want to display this run? (y) ");
get list(ana);
if ana="yes" then call drun(rptr);
dock: close file(indfile),file(modfile),file(secfile),file(runfile);
return;

error: revert cond(quit),cond(any_other);
out skip list("(model "llrname" has not been run)");
on key(indfile);
delete file(indfile) key(rname);
on key(runfile);
delete file(runfile) key(rname);
end;
dcl 01 ru
    
    02 rname char(30) varying,
    02 model,
        03 mname char(30) varying,
        03 n fixed,
        03 mmax fixed,
        03 inst cost float,
        03 cost decay float,
        03 ins inst float,
        03 secta refer(n),
        04 sn char(30) varying,
            04 govinv(b refer(tmax)) float,
        02 sector(a refer(n)),
            03 sn more char(30) varying,
            03 avg size float,
            03 ins inst float,
            03 build rate float,
            03 new inst float,
            03 purch float,
            03 cost p,
                04 ce mean float,
                04 cdev float,
            03 inst,
                04 lmean float,
                04 ldev float,
        02 cost(qtb refer(tmax)) float,
        02 tot inst(qtb refer(tmax)) float,
        02 govin(kw) refer(tmax)) float,
        02 prinv(b refer(tmax)) float,
        02 al.govinv(qtb refer(tmax)) float,
        02 al.prinv(qtb refer(tmax)) float,
        02 secta ref(n),
            03 prinv(b refer(tmax)) float,
            03 govinst(b refer(tmax)) float,
            03 prinst(b refer(tmax)) float,
            03 totgovinv(qtb refer(tmax)) float,
            03 totprinv(qto refer(tmax)) float,
            03 totinv(b refer(tmax)) float,
            03 totgovinst(qtb refer(tmax)) float,
            03 totprinv(qto refer(tmax)) float,
            03 totinst(b refer(tmax)) float,
            03 pat inst(qto refer(tmax)) float,
            03 cost accu(pib refer(tmax)) float,
            03 inst accu(b refer(tmax)) float;

  dcl rptr ptr1;
Input and output data reside in one of 3 kinds of data blocks – sector, for sector-specific input data, model, for model-wide input data, and run, for output data (including the inputs which it used). The commands for manipulating these blocks are entered as follows (where TYPE is model, sector or run, and the names are to be replaced with user-defined names. Alternatively, any or all parameters may be left out – you will be prompted for them):

To create models or sectors:
```
create
```
```
or TYPE NAME1 NAME2 ...
```
```
cr
```

To copy models or sectors:
```
copy
```
```
or TYPE OLDNAME NEWNAME
cp
```

To display models, sectors or runs:
```
display
```
```
or TYPE NAME1 NAME2 ...
dp
```

To delete models, sectors or runs:
```
delete TYPE NAME1 NAME2 ...
delete
```

To modify models or sectors:
```
modify
```
```
or TYPE NAME1 NAME2 ...
m
```

To run a model:
```
run
```
```
or MODELNAME RUNNAME
r
```

To get an index of data on file (all blocks will be listed if TYPE is all or blank):
```
index
```
```
or TYPE
.l
```

To reprint these instructions:
```
help
```

To stop processing and disconnect from the computer:
```
return
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name of the sector</td>
</tr>
<tr>
<td>avgsize</td>
<td>average number of Kwatts in a PV installation</td>
</tr>
<tr>
<td>initpoin</td>
<td>number of potential installations at time 0</td>
</tr>
<tr>
<td>buildrate</td>
<td>decimal growth rate of potential installations</td>
</tr>
<tr>
<td>newinst</td>
<td>number of new installations per time period</td>
</tr>
<tr>
<td>buurch</td>
<td>decimal probability of PV purchase given acceptability</td>
</tr>
<tr>
<td>costmean</td>
<td>mean of cost feasibility distribution</td>
</tr>
<tr>
<td>costdev</td>
<td>std dev of cost feasibility distribution</td>
</tr>
<tr>
<td>instmean</td>
<td>mean of successful installation distribution</td>
</tr>
<tr>
<td>instdev</td>
<td>std dev of successful installation distribution</td>
</tr>
<tr>
<td>quit</td>
<td>terminates processing, saves updated sector</td>
</tr>
<tr>
<td>qdisplay</td>
<td>terminated processing, displays and saves updated sector</td>
</tr>
<tr>
<td>display</td>
<td>displays updated sector so far</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>name</td>
<td>name of the model</td>
</tr>
<tr>
<td>n</td>
<td>number of sectors</td>
</tr>
<tr>
<td>Tmax</td>
<td>number of time periods</td>
</tr>
<tr>
<td>initcost</td>
<td>initial cost per kilowatt of PV</td>
</tr>
<tr>
<td>costdecay</td>
<td>decimal fraction by which PV cost declines when production doubles</td>
</tr>
<tr>
<td>initwatt</td>
<td>total kilowatts of PV installed at time 0 across all sectors</td>
</tr>
<tr>
<td>secname(i)</td>
<td>name of the i-th sector in the model</td>
</tr>
<tr>
<td>govinv(i,j)</td>
<td>government investment in sector i at time j (in M$)</td>
</tr>
<tr>
<td>quit</td>
<td>terminates processing, saves updated model</td>
</tr>
<tr>
<td>display</td>
<td>terminates processing, displays and saves updated model</td>
</tr>
<tr>
<td>display</td>
<td>displays updated model so far</td>
</tr>
</tbody>
</table>
&command_line off
ready_off
inbuf
line_length 130
if {user brief_bit} &then general_ready -string READY -set
else general_ready -string READY -x "(Costi" -inc_cost ")" -set
if {equal [user term_type] "2741"} &then set tty -modes tabs
else set tty -modes echo,crecho,fil,130, tabs
if {or [not [exists segment msg.data]]} {equal [user name] "Gravens"]}
&then &goto nomsg
else
&print
10 out_chars user_output -sm msg.data
if {equal [response "Do you want this message deleted? [ ] "] "yes"}
&then dl msg.data
else
&goto nomsg
&print
&goto loc_name
&print Welcome to the photovoltaic simulation model.
&print [response "Do you need an explanation of how the simulation system works (yes or no)? [ ]"]
&goto finale
&goto no
&goto no
&goto finale
if {equal [tfary ["return" ]}"return"}
&then logout -of -hold
else general_ready -revert
&quit