A Dynamic Model for Analysis of Solar Energy Systems

C. L. Hamilton
TDA Planning Office

A dynamic analysis technique for evaluating the performance of solar energy systems is needed as part of the Goldstone Energy Conservation Project. This article reports progress in the construction of a model simulating prospective solar energy systems and of the computer programs in which it is embodied. The modeling approach is discussed as applied to a general baseline configuration for prospective systems, and the two programs that have been completed are described.

I. Introduction

Part of the Goldstone Energy Conservation Project involves development of an analysis technique for evaluating the performance of solar energy systems. Unlike conventional systems, which can be turned on and allowed to run at steady state as long as they are needed, solar-powered systems are largely dynamic in nature. This report describes progress in construction of a model designed to simulate prospective solar energy systems under consideration for the Goldstone Space Communications Complex (GSCC), taking into account the time dependencies inherent in their operation. The model is described together with the computer program in which it is embodied.

A primary requirement placed on the program that embodies the desired dynamic model is that it be convenient and easy to use. Since it is intended to analyze the performance of many alternative solar energy systems, the program should be modular, allowing quick and simple substitution of components in the system under study. The program documentation and code are to be written so that the functioning of the model can be understood easily even by readers unfamiliar with its development. In particular the mathematical relationships should be readily traceable throughout.

II. The Approach

We begin by postulating a baseline system designed to meet the energy needs at GSCC, powered primarily by solar sources. For the purposes of the model, this system is characterized in terms of key parameters describing the operation and interaction of all the component subsystems. The parameters of interest fall into two categories. The first kind, primary levels, is accumulated quantities

(an example would be the amount of fuel in a storage tank) describing the state of a component at a particular instant in time. The second type of parameter comprises the instantaneous rates of change for the levels at the same time. Rates depend only on the magnitude of one or more levels, a set of constants characteristic of the properties of associated subsystems, and any exogenous inputs (driving forces) that affect those subsystems. In short, the rates provide a means for coupling system components with each other and with the surroundings. The time course of these parameters is determined by stepwise computation of all variables at small time intervals, starting from pre-specified initial values for each level involved (Ref. 1).

The necessary calculations are embodied in the program in the form of several computational modules, each of which contains all the relationships necessary to describe operation of a system component (or assembly of components) as well as its interfaces with other components. These computational modules are imbedded in a program structure that ultimately will allow extensive user manipulation of inputs, outputs, and the control variables that govern the time frame over which analysis is carried out. Modification of the program to incorporate the characteristics of an alternate system component involves substitution only in the appropriate computational module; it can be accomplished in a few minutes once the characterization of the alternate component is complete.

III. The Baseline System

Figure 1 is a block diagram of the baseline GSCC energy system, showing overall energy flow. It envisions parallel conversion of the solar sources, sunlight and wind, into dc electricity. The dc output can be converted for direct use to satisfy some of the complex's ac demand or it can go to generate hydrogen and oxygen by electrolysis of water. Both gases can then be stored for use in the existing engine-generators, which will retain capability for burning petroleum fuel as well. Waste heat from the engines will satisfy some of the heating and air-conditioning requirements. Provision is made for input of commercial electricity as needed, but it is intended that only a small part of the total energy be purchased. The contribution of commercial electricity is considered to be negligible for modeling purposes.

Translation of the physical system into linked computational modules is illustrated in Fig. 2. With the exception of the blocks depicting hydrogen and oxygen production and storage, each system component in Fig. 1 is represented by a corresponding module in Fig. 2. The characterizations of electrolysis and of the storage of its products are combined into a single Hydrogen-Oxygen Module. One computational module does not appear explicitly in the physical block diagram. That is the Dispatch Module, which contains the strategy governing how the energy is routed through the various available pathways. Arrows in Fig. 2 represent the direction of information transfer occurring in the model; the corresponding labels identify the nature of the data involved.

Both the physical system and the system of computational modules divide themselves naturally into subsystems. The physical subsystems are delineated in Fig. 1 by double-lined blocks. There are two subsystems, solar and wind, that carry out initial processing of the input energy. They are separated from each other because the processes involved are physically different and because the time dependencies of the wind speed and solar radiation inputs differ markedly from each other. The central subsystem contains the components devoted to further energy conversion, storage, and distribution, while the remaining one is made up of the load to be satisfied.

Division of the system of computational modules into groups is analogous; these are surrounded by broken lines in Fig. 2. A key characteristic of the dissection is that the connections between groups are one-way. A group either delivers information to other groups or receives information from others, but does not do both. This means that the corresponding subsystems can be modeled separately, using separate computer programs. The advantages of this arrangement are illustrated in the following discussion.

IV. The Programs

The programs to be employed for dynamic analysis of solar energy systems are to be written in MBASIC and implemented on JPL's Univac 1108 A system, accessed through remote terminals such as Execuports. They are developed using top-down design and structured programming. The Level 1 flowchart in Fig. 3 illustrates the general structure of a representative dynamic model program. After initialization, which includes specification of starting values for all primary levels, and selection of input and output options, the program enters an integration loop. There, values for all necessary rates at time T are calculated, as well as any useful parameters that may be derived from them. Next, the values chosen for output are stored at desired intervals. Finally the primary level values are updated to correspond to time $T + DT$ and computation goes back to traverse the loop again. When integration is complete, the stored output is

printed or plotted, depending on the output medium chosen by the user.

As can be seen from Fig. 2, the model calculations depend on several exogenous inputs. These represent the influences exerted by the surroundings on the performance of various pieces of the system. Inputs are provided as data files consisting of day–hour tables containing values of the appropriate parameters for each hour in a year. These data files are themselves the result of models, and are subject to refining and upgrading as work on those models progresses. In general, however, their contents do not vary between runs of the system simulation model.

For computational stability, the integration interval *DT* must be small relative to the shortest time constant in the system being simulated. At a minimum, simulation of a year's system operation entails 8760 iterations of the program's calculation loop (DT = 1 h). It is to be expected that the system components will exhibit considerable variation in characteristic response time, and some will require integration at intervals much smaller than one hour. To increase economy of computer use, it is desirable to include as few calculations in the integration loop as possible, and to minimize the number of times the loop must be traversed. Using separate programs for computational groups helps to achieve these ends. For example, while using the dynamic model to perform analyses, it is not necessary to run the SUN or WIND groups each time the model is run if the analyses do not involve any variation in components of the solar or wind subsystems. The information output from the SUN and WIND groups is analogous to that contained in the data files for exogenous inputs, and can, in fact, be considered as pseudo-exogenous inputs to the group labeled SENSMOD2 in Fig. 2. Therefore, the programs SUN and WIND are intended to generate data files consisting of hourly electrical output rates; they will be run once whenever alteration is made in appropriate system components or the input data files are upgraded, and the resulting sets of output files will in turn be used as input whenever the program SENSMOD2 is used.

Note that, while the computational modules representing the load components are here included in the group SENSMOD2 and the corresponding program, they can be separated as SUN and WIND were. If it should prove desirable, another one or two groups (and programs) can be created to represent load functions.

At this writing, the programs SUN and WIND have been designed, coded, and tested. Data files have been generated with WIND. Work on SENSMOD2 (Solar *EN*ergy System *MOD*el, version 2[1]) is in progress. The following sections provide more detailed description of how the programs are put together.

A. SUN

Level 1 design for SUN is shown in Fig. 4. That program is intended to compute the hourly average electrical output of a solar thermal generating subsystem consisting of collector, heat storage, and heat engine. Inputs to the program include hourly average values of solar radiation intensity, hourly values for the angle of incidence of that radiation on the collector surface, and hourly average readings of ambient temperature. Output, in units of kWh/h, is stored in data files standardized for input into SENSMOD2 and represents a year's operation, hour by hour, of the subsystem, starting at 0000 GMT January 1 and ending at 2400 GMT December 31.

A Level 1 design chart illustrates gross program structure only. Examination of lower levels is necessary to see the way in which the computational modules of Fig. 2 are imbedded in that structure. The key program modules here (not to be confused with computational modules) are those labeled CALC and UPDATE. Before discussing these, however, a digression into the response times characteristic of the solar subsystem components is in order.

It was discovered early that the solar collector exhibited unusual behavior as far as time constants were concerned. The collector can be characterized as a fluid-containing structure that absorbs energy in the form of solar radiation. Energy is removed from the structure by loss mechanisms (primarily conduction and radiation) and deliberate extraction through circulation of the fluid. The loss rate shows a nonlinear temperature dependence, as it represents contributions from two processes with different temperature dependences. Extraction rate is also nonlinear with temperature, owing to the way in which temperature-dependent flow control is specified. The net result is a thermal time "constant" that is not constant, but that varies by a factor of about six over the temperature range encountered in operation. While delivering heat to storage, the collector time constant is quite short, about 0.7 h as it is presently characterized.

In contrast, the heat storage and heat engine module computations can be carried out quite satisfactorily at integration intervals of one hour. The heat storage time constant is long, and the heat engine is essentially a

[1] Version 1 was a preliminary one, covering all components of the system.

steady-state device—either on or off and delivering a nearly constant output when on.

To cope with these widely varying response time characteristics and their effect on computing time requirements, a further economizing step was taken. Advantage was taken of the fact that the only information the solar collector computational module requires from the rest of the subsystem is the temperature of the fluid returning from storage, which is relatively constant and can certainly be regarded as constant for an hour at a time. An initial value of that return temperature could also be provided as a function of initial values of primary levels within the heat storage computational module. Under these circumstances, it was possible to nest a variable-step-size integration loop for the solar collector module alone within an hour-by-hour loop performing the calculations for the other two modules. Figures 5 through 7 contain the flow charts describing insertion of the collector integration loop DT-VARY into the main one. The flow charts shown represent increasing levels of detail. For example, Chart 1.4 (Fig. 5) is an expansion of block 4 in Chart 1 (Fig. 4), Chart 1.4.2 (Fig. 6) is block 2 in Chart 1.4 expanded, etc.

Representation of the "normal" computational modules, those for heat storage and the heat engine, occurs in the UPSTREAM, COUPLE, and DOWNSTREAM blocks of CALC, and in UPDATE. Each of these program modules is represented by a subprogram that can be called from the computer and listed separately. Distribution of each computational module into the several program modules is necessitated by the two-way information flow between computational modules and by the need during execution of the program to stop and store desired information for output before primary levels are updated. For illustration, representation of the heat engine module is described below in detail.

A complete characterization of the heat engine computational module comprises the information contained in Figs. 8 and 9. Figure 8 is an indented list of the relationships defining the behavior of the heat engine under analysis and giving values to the characteristic constants incorporated therein. Figure 9 shows the relationships in the form of a calculation tree and allows visualization of the order in which computation is carried out and how information is transferred between computational modules.

A listing of the subprogram UPSTREAM will contain, in initialization statements, values for all the constants specified in Fig. 8. The heat engine module contains no primary levels; for those computational modules that do, initial values would be supplied in those statements also. The initialization statements also contain tables describing any empirical relationships needed to characterize a component. The heat engine module has no tables. Separate, identified initialization statements are included in UPSTREAM for each computational module, to facilitate substitution. UPSTREAM also contains all equations below the solid dividing line in Fig. 9 (other computational modules have more than one equation; all equations for a module are grouped into one labeled set of statements). The UPSTREAM calculations represent all steps that can be carried out in one computational module without using a value calculated in another.

COUPLE is made up of a series of labeled statements, one for each computational module, that transfer variable values computed in that module to the modules requiring them. Its purpose is to simplify keeping track of the interfaces between components. In Fig. 9 variable transfers are denoted by a large X. Their operation is easier to conceptualize if the transfer is considered to occur into a computational module as implied in Fig. 9 rather than out of it as written in the SUN code. The reverse direction is imposed in the case of SUN by the nature of the interface between the solar collector integration loop and the main one.

In the case of the heat engine module, all the equations above the line in Fig. 9 are contained in a labeled group of statements in the subprogram DOWNSTREAM. For computational modules with primary levels, DOWNSTREAM has all remaining equations except the ones updating the primary levels to $T + DT$. Those appear, after output data for time T are stored, in subprogram UPDATE.

Revision of the program to analyze the performance of an alternate solar collector involves changing statements in the subprogram ALLSOL to reflect the characteristics of the new collector. Examination of alternate versions of any of the "normal" system components entails revising clearly identified statements in each of the four subprograms discussed above, a practice that is much simpler in execution than in appearance and that helps guard against making errors in defining the interfaces between components.

B. WIND

WIND is a less complex program than SUN, representing a less complex subunit in Fig. 2. It takes as input hourly average values of wind speed and computes from them the hourly average output during a year's operation

of a wind turbine generator. The result is a set of data files standardized for input into SENSMOD2 containing hourly values of electricity generated from wind, in MWh/h.

C. SENSMOD2

Current effort is directed toward completion of the model and program for SENSMOD2. This program will be implemented first in an abbreviated version with limited capability for handling input and output options. Assembly of a complete set of operational programs and production of preliminary output data on operation of the whole energy system is a high priority goal. Concurrent and following activity will be devoted to refinement of component characterization packages and implementation of the program's full user option capability.

Reference

1. Forrester, J. W., *Principles of Systems*, Wright-Allen Press, Inc., Cambridge, Mass., 1968.

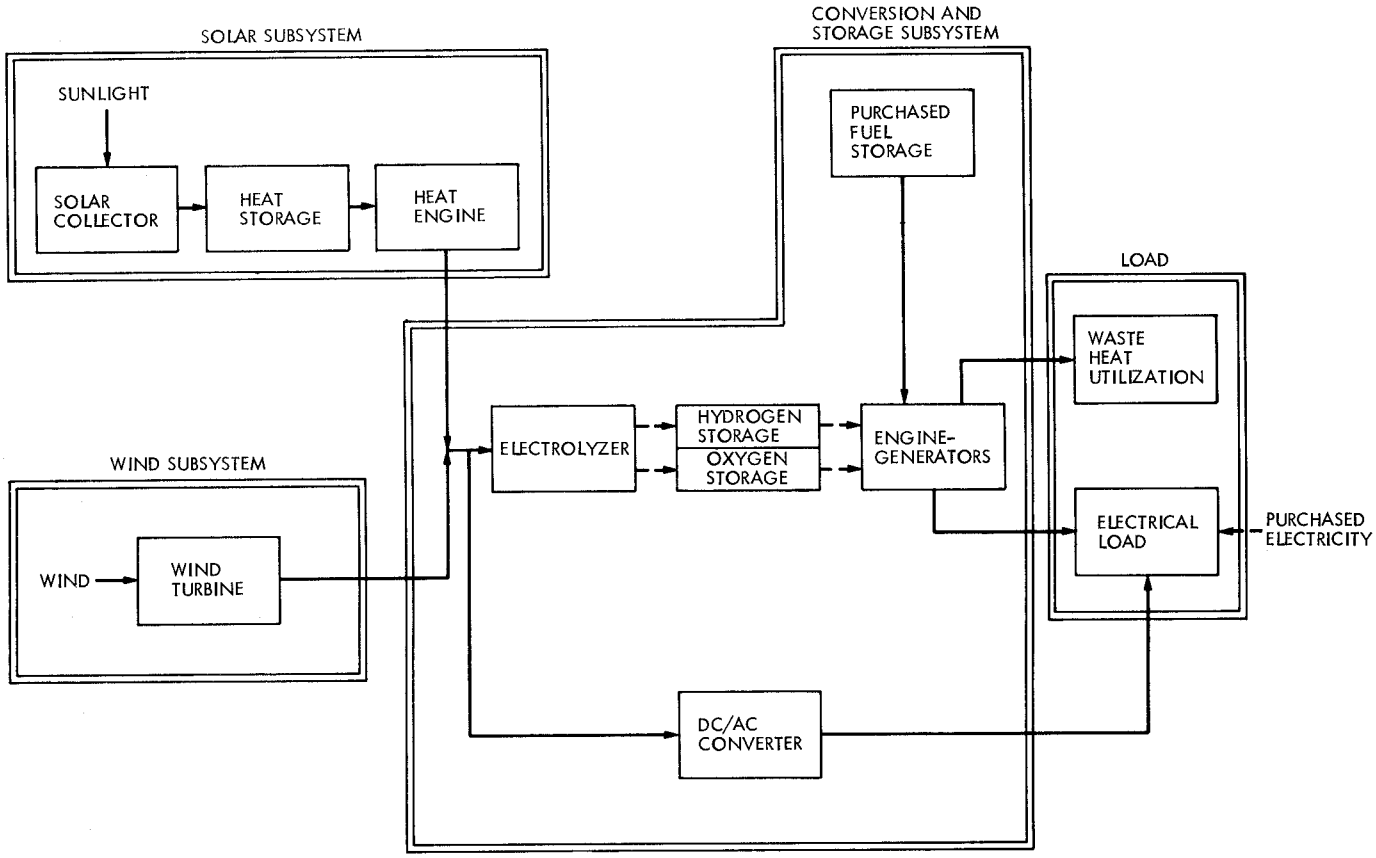

Fig. 1. Baseline solar energy system

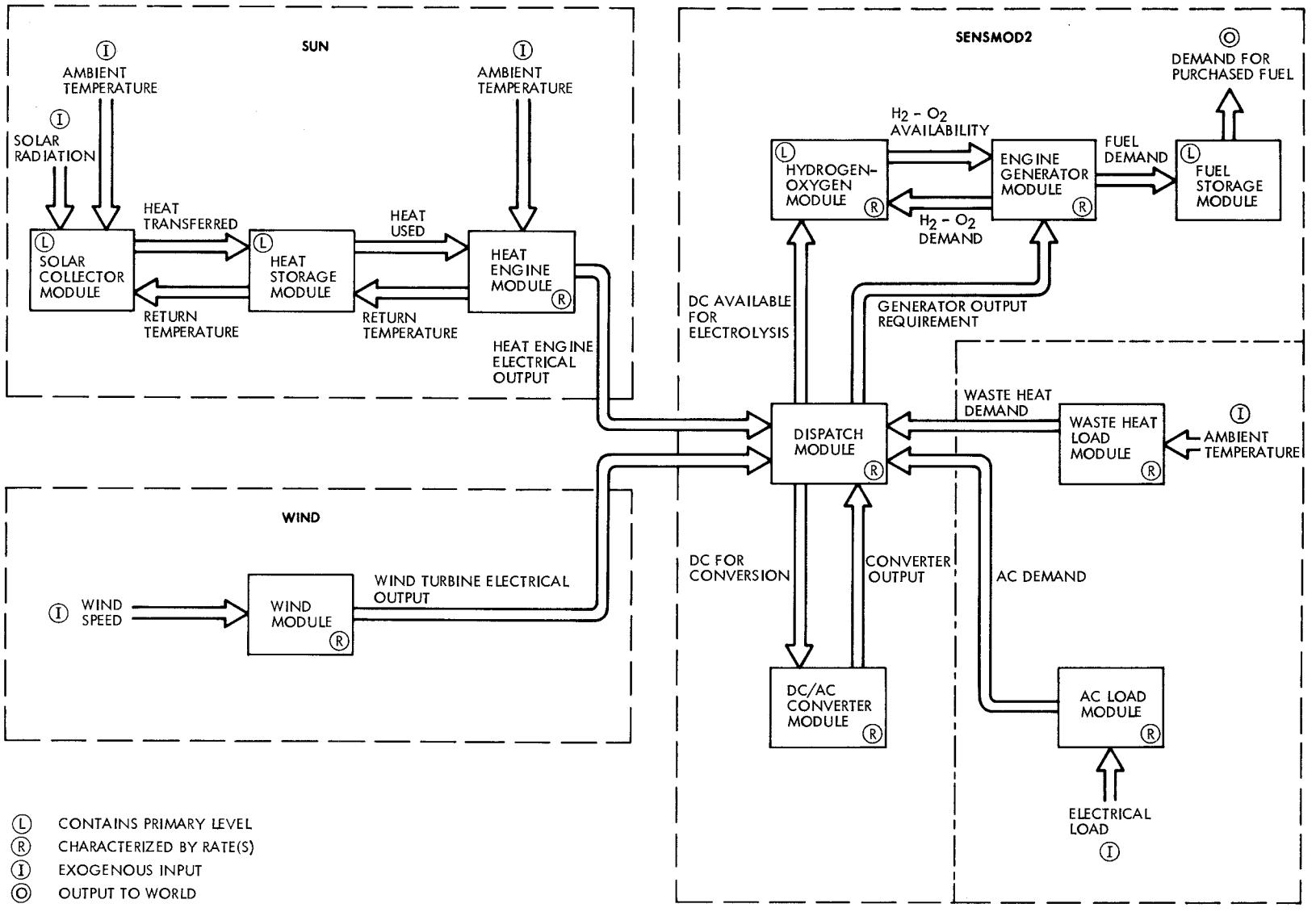

Fig. 2. Computational modules and data flow

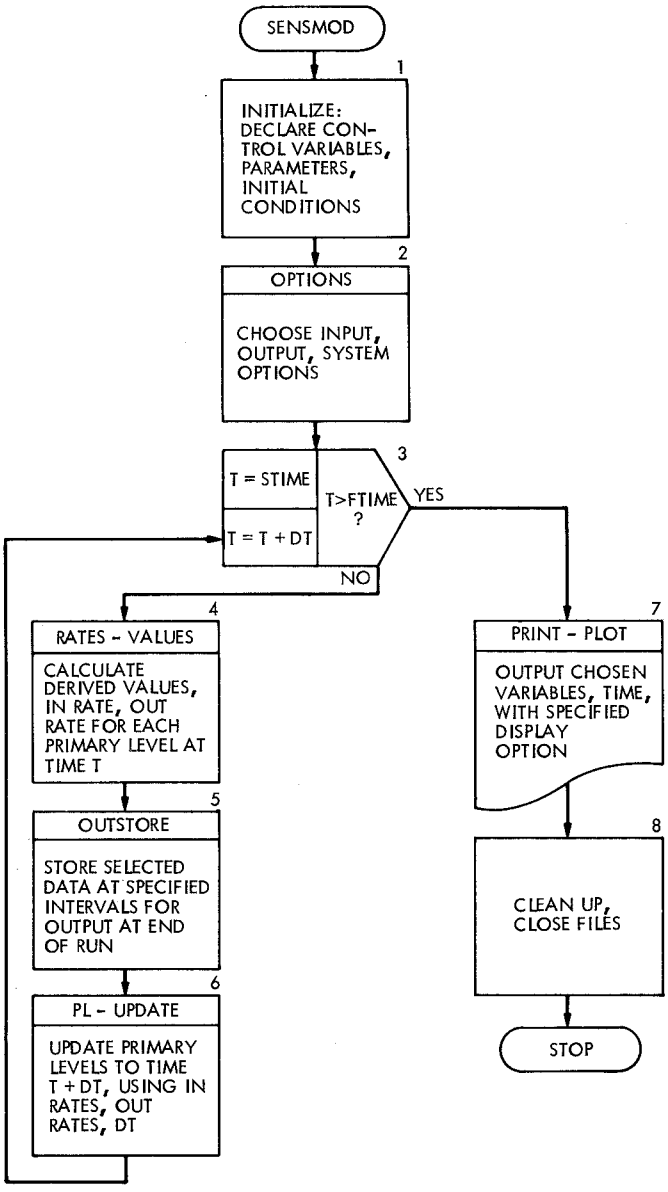

Fig. 3. Level 1 design, representative dynamic model program

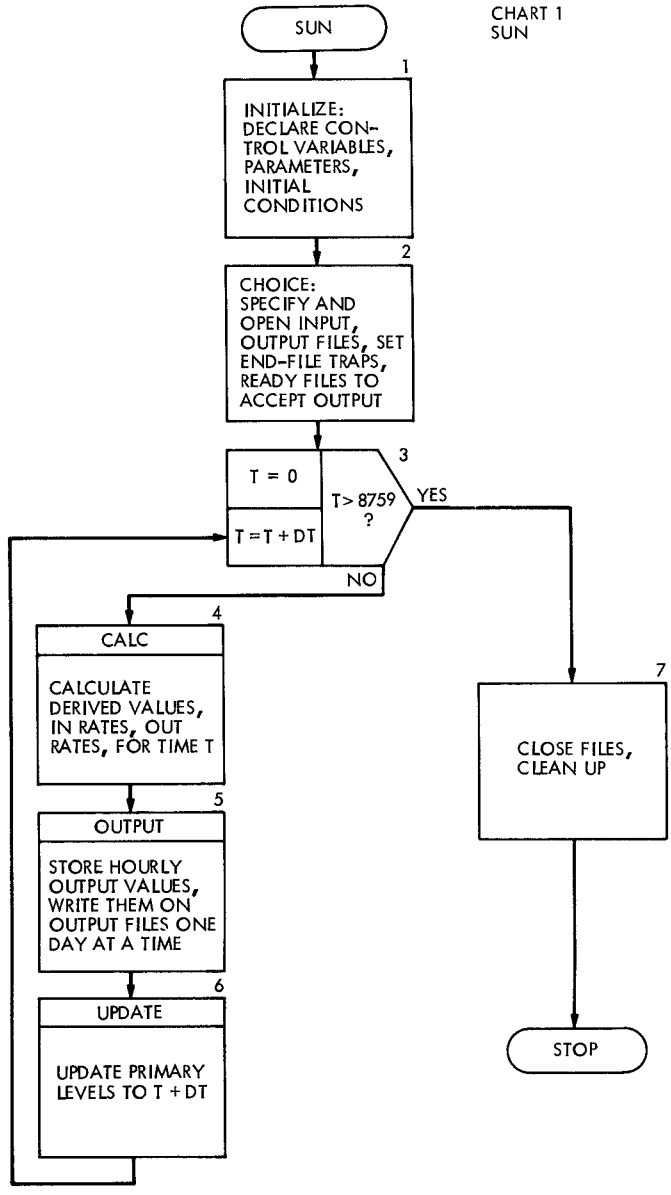

Fig. 4. Level 1 design, program SUN

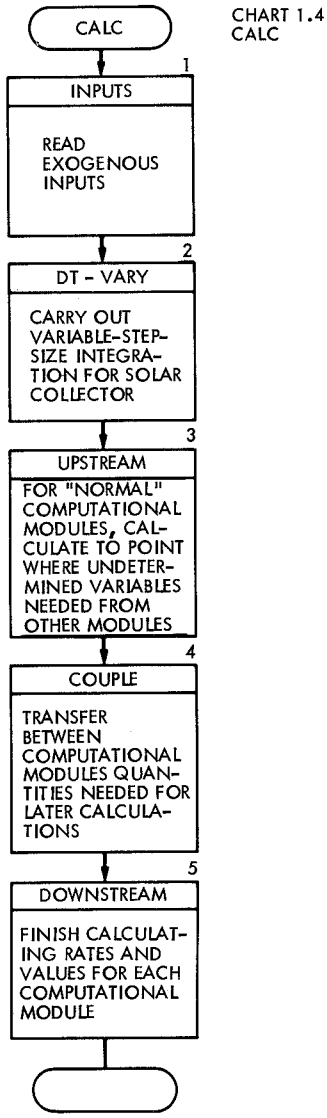

Fig. 5. Partial Level 2 design, program SUN

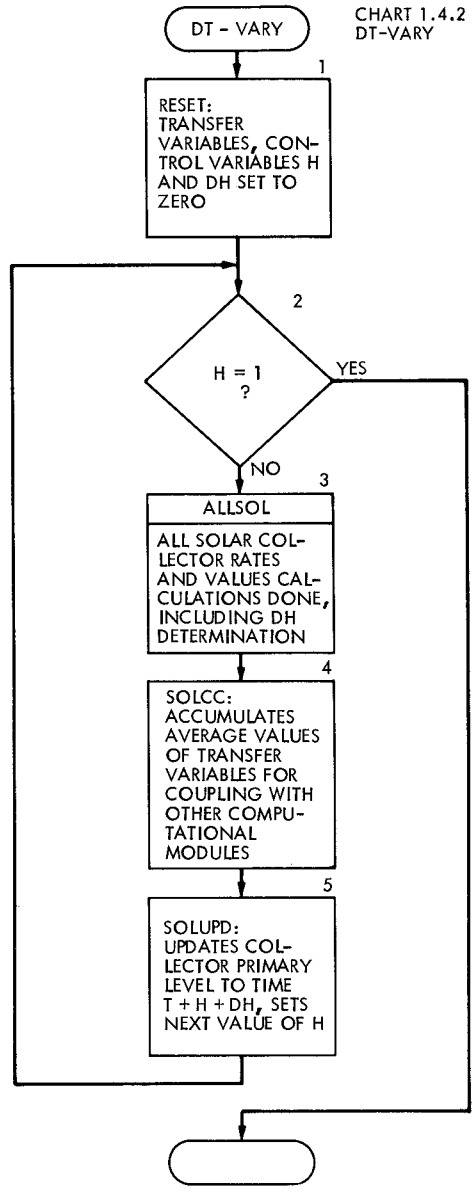

Fig. 6. Partial Level 3 design, program SUN

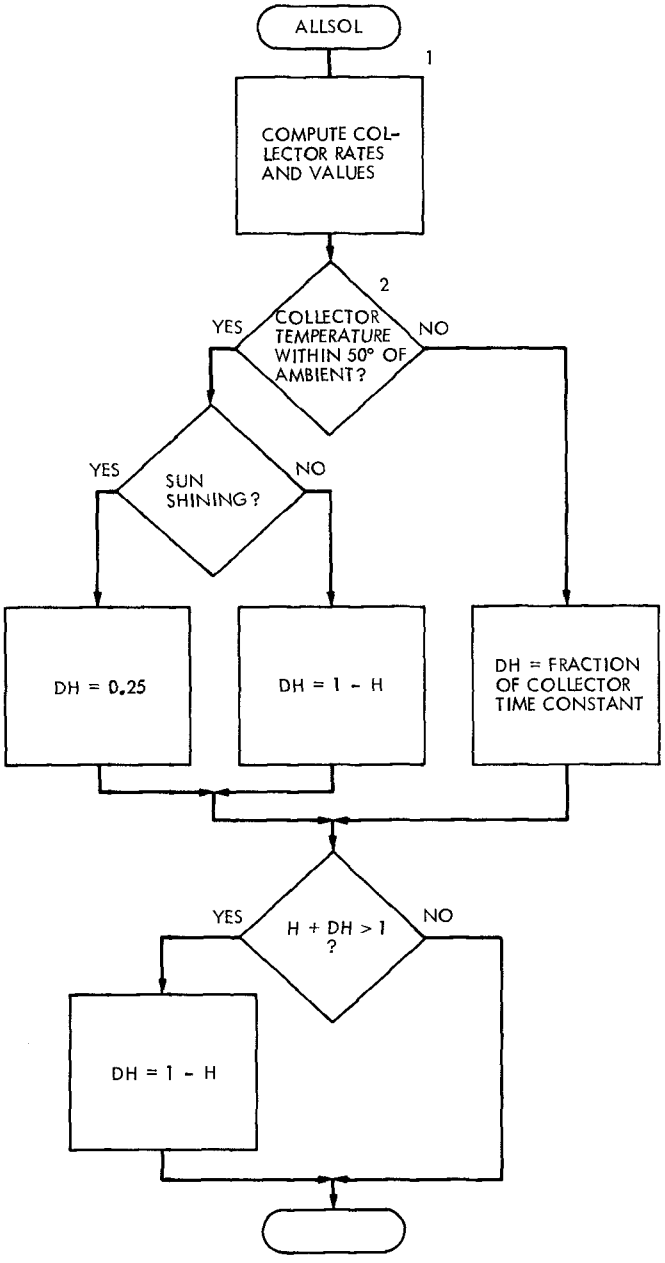

Fig. 7. Partial Level 4 design, program SUN

3.1	HEAT ENGINE OUTPUT RATE	HEOR=HEIR*CYEFF*MEFF*GEFF	kWh/h
3.1.1	HEAT ENGINE INPUT RATE	HEIR=WFFR*[(REFT-SNKT)*WFSH+WFHV]	kWh/h
3.1.1.1	WORKING FLUID FLOW RATE	WFFR=EFFR from 2A.2.1.1	kg/h
3.1.1.2	REFERENCE TEMPERATURE	REFT=38	°C
3.1.1.3	SINK TEMPERATURE	SNKT=AMBT+10	°C
3.1.1.3.1	AMBIENT TEMPERATURE	AMBT-exogenous input	°C
3.1.1.4	WORKING FLUID SPECIFIC HEAT	WFSH=0.0007	kWh/kg°C
3.1.1.5	WORKING FLUID HEAT OF VAPORIZATION	WFHV=EFHV from 2A.2.1.5	kWh/h
3.1.2	CYCLE EFFICIENCY	CYEFF=CEFF*RCCE	n.d.
3.1.2.1	CARNOT EFFICIENCY	CEFF=(WFDT-SNKT)/(WFDT+273)	n.d.
3.1.2.1.1	WORKING FLUID DELIVERY TEMPERATURE	WFDT=EFDT from 2A.2.1.5.1	°C
3.1.2.1.2	SINK TEMPERATURE	SNKT from 3.1.1.3	°C
3.1.2.2	RATIO OF CYCLE TO CARNOT EFFICIENCY	RCCE=0.9	n.d.
3.1.3	MECHANICAL EFFICIENCY	MEFF=0.8	n.d.
3.1.4	GENERATOR EFFICIENCY	GEFF=0.9	n.d.

Fig. 8. Relationships within heat engine computational module

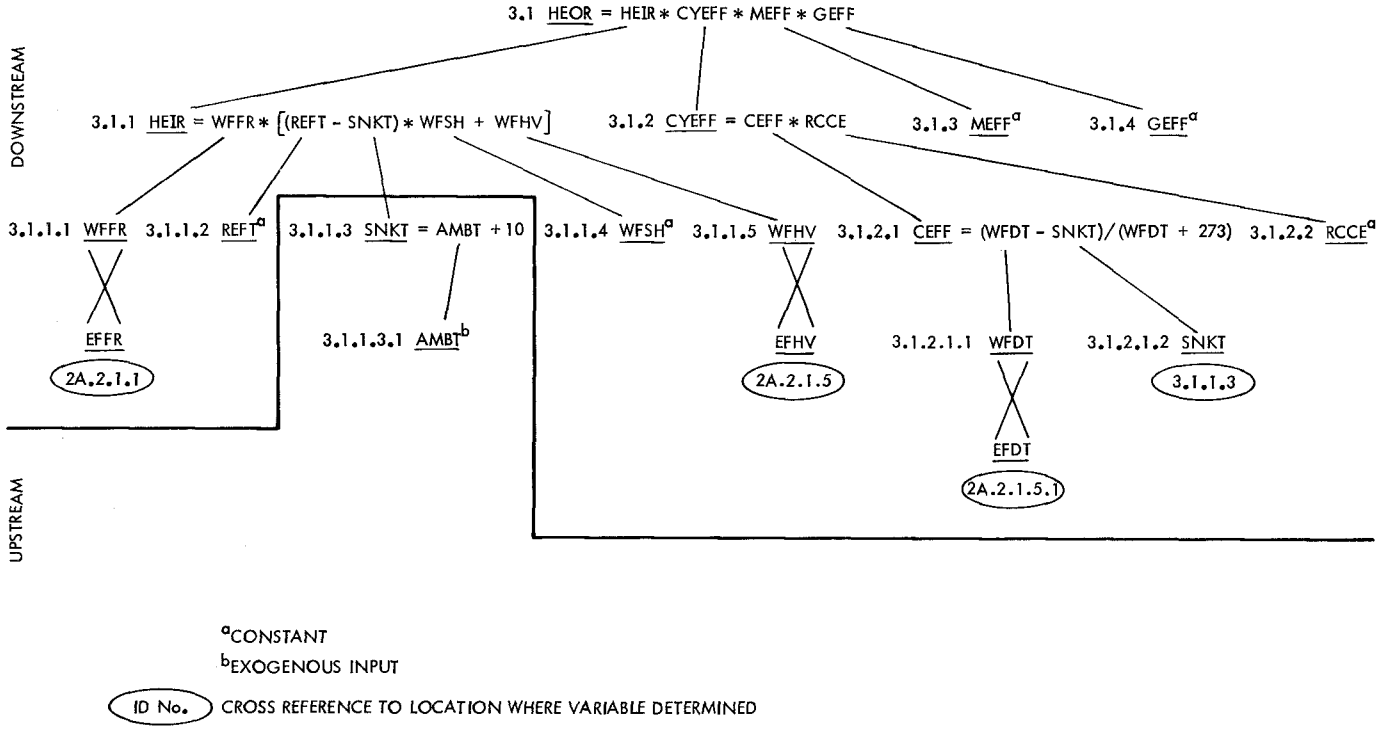

Fig. 9. Calculation tree for heat engine computational module