LBL-28273

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

APPLIED SCIENCE
DIVISION

Presented at the Society for Computer Simulation
Multiconference, San Diego, CA, January 17-19, 1990,
to be published in the Proceedings

# Radiant Transfer due to Lighting: An Example of Symbolic Model Generation for the Simulation Problem Analysis Kernel

E.F. Sowell, J.-M. Nataf, and F. Winkelmann



January 1990

APPLIED SCIENCE
DIVISION

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

---

LBL--28273

DE90 013360

Presented at the Society for
Computer Simulation Multiconference,
San Diego, CA Jan 17-19, 1990

# Radiant Transfer due to Lighting: An Example of Symbolic Model Generation for the Simulation Problem Analysis Kernel

Edward F. Sowell
Department of Computer Science
California State University, Fullerton
Fullerton, CA 92634

Jean-Michel Nataf and Frederick Winkelmann
Applied Science Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

January 1990

## Abstract

The Simulation Problem Analysis Kernel (SPANK) is a simulation environment under development at Lawrence Berkeley Laboratory. A principal departure from other simulation environments is that system models are constructed from submodel objects that are defined without prescribed input or output interfaces, yielding greater modeling flexibility. Also, graph theoretic techniques are employed to determine the solution sequence, including reduction of the iterative problem size. In this paper we show one role of symbolic manipulation in SPANK processing, namely automatic generation of submodels using the MACSYMA™ package. This is demonstrated in the context of steady state radiant and convective heat transfer in a room with a fluorescent lighting system, and then in the corresponding dynamic context. Submodel definition and generation are described, followed by the solution of several interesting problems defined with the submodels.

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Buildings and Community Systems, Building Systems Division of the U. S. Department of Energy, under Contract No. DE-AC03-76SF00098.

## Introduction

Contemporary modeling environments, e.g., ACSL [MG75], require the modeler to describe a particular problem by interconnection of component models, either defined by the user or selected from a library, which have prescribed inputs and outputs. Such submodels are called "algorithmic" or "assignment-based" because the output variables are calculated from the inputs by a prescribed procedure, and each output variable is assigned a value. This approach imposes several limitations on modeling flexibility. Primary among these is the fact that the algorithmic models limit the class of problems that can be defined and solved by their interconnection. For example, assume that we have two components modeled algorithmically as:

component A: x := f(y,z)

component B: z := g(x)

Then we can easily solve the problem "given y, find x and z" by iteration on either of the unknowns. Such iteration is automatically performed by programs such as ACSL. However, if the problem statement is changed only slightly, such as "given z, find x and y," difficulties are encountered because we have an unneeded algorithm for z and no algorithm for y. As a result the modeler must either formulate a new component model by inverting A, i.e.,

component A′: y := f1(x,z),

or resort to the introduction of fictitious "implicit elements" with attendant numerical difficulties. We see that the modeler is forced to revise component models and reconstruct the system, and/or introduce non-physical "components" and become involved in the intricacies of numerical analysis.

The above difficulties are fundamentally related to the ad hoc development of current simulation environments, which were greatly influenced by analog computation methods, and to the widely held belief that physical "causality" should be reflected in the system model. More recent developments [Mat88,Mat89,SS89] suggest that a different set of model structuring principles is more appropriate. Among these principles is that component models should be *equation-based* rather than assignment-based. That is, the models above should be seen by the modeler as:

component A: $f(x,y,z) = 0$

component B: $g(x,z) = 0$

with the understanding that the environment will find a solution procedure that will enforce the equalities. Simulation environments that employ equation-based component models eliminate the difficulties seen above. The Simulation Problem Analysis Kernel (SPANK) is one such environment currently under development. Another, using an alternate approach, is IDA (formerly MODSIM) [Sah88].

The SPANK environment is based on the intuition that:

(a) there should be a single "model" for a particular component,

(b) the system model should be defined once, yet be capable of solving any well-posed problem involving the system variables, and

(c) the environment software should select the appropriate solution sequence, including necessary iteration, in a manner transparent to the modeler.

The underlying principles making this realizable for static (i.e., algebraic) systems have been described [SBEW86], and extended to handle dynamic (i.e., differential-algebraic) systems [SB88]. These ideas are described in terms of the object-oriented paradigm in [SBN89].

Briefly summarized, SPANK is a nonlinear equation solver with an object-oriented interface and with automatic equation system reduction. SPANK manipulates objects that are equations, and macro objects that are collections of equations. The task of the user is to generate the objects and their related software equivalent, and to link them together in the appropriate way to take into account the variables common to different objects.

A diagram of the current SPANK environment is shown in Fig. 1. In addition to the main SPANK program, utilities are available that include automatic object generation using MACSYMA (discussed further in the following sections), automatic library archiving of newly-created objects and functions, checking of some aspects of the well-posedness of the problem, graphical output for dynamic simulations, graphical sensitivity analysis for steady-state simulations, generation of MACSYMA code for evaluating reduced system Jacobian eigenvalues, graphical representation of the convergence of iteration variables, and automatic report generation.

Step by step, the procedure that SPANK users follow in setting up and running a simulation is as follows:

(1) Draw a schematic showing the physical components of the system and how they are connected.

(2) Write the mathematical equations, such as energy balances, mass balances, flux balances, etc., that describe the system. These equations are the basic objects that SPANK manipulates.

(3) Run the SPANK MACSYMA preprocessor to generate the **C** code and associated functions for the equations (objects) in (2).

(4) Using the SPANK Network Specification Language (NSL) or the MACSYMA preprocessor, link the objects in (3) into macro objects (sets of equations) that describe system components or processes.

(5) Using NSL, create a *problem specification file* in which macro objects are linked together into a network describing the entire system. (An interactive graphical editor is being developed that will simplify this step.) >From this file SPANK will then automatically create an executable simulation program for solving the network.

(6) Specify input values and starting values, and, for dynamic simulation, start time, stop time, and time step.

(7) Run the simulation.

(8) Plot results.

## The Need for Symbolic Manipulation

Because SPANK development is still embryonic, there have been many limitations preventing a meaningful comparison with the existing conventional environments. One such limitation is that the SPANK component models (called objects) reported in earlier publications had been "hand crafted." That is, the component modeler in need of component A above would have to write computer code such as:

```
/* SPANK Component Object Model */
/* Stored in comp_A.obj         */

    define comp_A(x, y, z)
    double x,y,z;
    {  x = f1(y, z);
       y = f2(x, z);
       z = f3(x, y);
     }
```

The understanding here is that the component model has three interface variables *x, y,* and *z.* Normally, an inverse is given for each interface variable so that the solver can select the most appropriate for a particular problem (although inverses known to be problematical can be suppressed).

Because SPANK objects represent a single equation, definition of an entire problem in terms of objects can be tedious. For convenience and hiding of unnecessary detail when defining problems, a *macro* facility is available. With this facility, for example, we could combine the preceding two objects into a macro object with the code:

```
/* SPANK macro object definition */
/* Stored in macro_A_B.obj      */

macro
   declare  comp_A  A;
   declare  comp_B  B;
   link x(A.x, B.x);
   link y(A.y);
   link z(A.z, B.z);
```

Macros are declared in problem definitions (or in other macros) exactly as objects are declared.

By "hand crafted", above, we mean that the macro definitions, object definitions, and the inverse functions *f1, f2,* and *f3* had to be manually coded as functions in the **C** computer

language. This is clearly an unnecessary burden on the component modeler, since software systems such as MACSYMA [MIT83] are available that can accept an equation such as $f(x, y, z) = 0$ and return the needed inverses as compilable code. The same program can be used to generate macro and object files of the kind shown above. The principal aim of this paper is to demonstrate this approach.

It should be noted that automatic generation of macros, objects, and inverses represents one form of symbolic manipulation in simulation. Another possibility, not explored here, is generation of derivative formulas. In general, the aim is to perform symbolic manipulation as a pre-step in order to reduce the level of numeric work during the simulation while affording maximum flexibility to the modeler in defining problems.

In the sections that follow we demonstrate the use of MACSYMA to automatically generate the SPANK macro object and object definitions, and the required inverse functions as **C** code. This is done in the context of a problem from building physics, namely determination of the transport of lighting energy by convection and multi-band radiation in a room. While the physical problem is over simplified, the model goes beyond previously published SPANK example problems [SBEW86; SBN89] in two important respects. First, its definition requires vector interface variables, where the vector length is problem dependent. That is, the number of elementary interfaces for a particular object depends on the number of other interacting objects. Since neither SPANK objects nor macro objects admit to variable interfaces, the needed flexibility is implemented at the MACSYMA input level. The second interesting aspect of the problem is that it yields a highly interconnected set of equations as a result of radiative transfer. As a result, the ratio of the number of total problem variables to the number of iteration variables (called the reduction ratio) is not as large as we find for previously studied flow system simulations [SBN89].

## Problem Description

The problem studied is shown in Fig. 2. Lighting is provided by fluorescent lamps in the plenum space of a 10,000-ft$^2$ room. A translucent ceiling lens separates the plenum from the room below. Supply air enters the room, mixes with the room air, then exhausts to the plenum through small openings in the ceiling lens. Input power leaves the lamp by shortwave (visible) and longwave (infrared) radiation and by convection to the plenum air. The radiative portion undergoes interreflection and transmission, and is ultimately absorbed by surfaces in the plenum and the room. If the plenum air temperature is greater than the room temperature, some or all of the convective portion can also escape the plenum by conduction through the transparent ceiling to the room air. Ultimately, all lamp power must be removed by the airstream after convective transfer from the various solid surfaces in the room and plenum. We wish to determine the surface and air temperatures, and the heat removal rate in the room and plenum. Naturally, these will be functions of the mass flow rate of air and the supply air temperature.

Geometric, radiative and convective data for the problem are shown in Tables 1 through 5. Lamp power (120,000 Btu/hr or 3.5 W/ft$^2$) and lamp area have been chosen so as to yield typical illumination at the floor with normal fluorescent lamp brightness. The

lamp diameter to spacing ratio is

$$\frac{D}{W} = \frac{A_{lamp}}{\pi A_{floor}} = 0.0477$$

It can be shown that the lamp-set self view factor is (approximately)

$$F(lamp, lamp) = \frac{2}{\pi}\sin^{-1}(\frac{D}{W}) = 0.0304$$

For simplicity, we assume that the dimensions in the horizontal plane are large relative to room and plenum height, thus making losses through walls negligible. The floor to ceiling view factor is then assumed to be 1.0. Other view factors can then be determined by reciprocity and conservation (see Table 4). It is assumed that the floor and ceiling are adiabatic, i.e., that no heat transfer occurs between the ceiling and the room above or the floor and the plenum below.

The convective heat transfer coefficients shown in Table 3 assume free convection [ASH89] and are taken to be constant. A straightforward improvement to the model would be to make these conductances a function of air flow rate and surface-to-air temperature difference.

## Formulation and SPANK Representation

The above problem can be formulated as an n-node network in which each node is viewed as a surface that can emit, absorb, reflect, and transmit radiant energy in the short and long wave bands. Also, nodes can interact through surface-to-air convection, and through bulk flow convection. The system variables include node temperatures, short and long wave radiosities and irradiations at each node. The basic physical laws governing the system are those of diffuse radiative transfer, convective heat transfer, and conservation of energy and mass. See [SO73] for details of this formulation.

The equations that describe the problem are as follows:

### 1. Object "blackbody"

This is the blackbody radiation equation, with $e_b$ being the blackbody radiance, $\sigma$ the Stefan-Boltzmann constant and $t$ the absolute temperature:

$$e_b = \sigma t^4$$

### 2. Object "radiosity" and macro object "radiosity_macro"

This equation gives the radiosity $J_k$ of a node $k$ in terms of the other node irradiations, $FJ_j$, the generalized transmittance matrix, $\tau^s{}_{k,j}$ and the node source radiosity $J^0{}_k$ and reflectance $r_k$.

$$J_k - J^0{}_k - r_k FJ_k = \sum_{j=1}^{7} \tau^s{}_{k,j} FJ_j$$

The above is valid for node $k$. The macro object *radiosity_macro* corresponds to the system of the above equations with $k$ running from 1 to 7.

### 3. Object "a.x" and macro object "a.x_macro"

This equation yields, for a node $k$, the irradiation $FJ_k$ in terms of all the radiosities $J_j$ of the nodes exchanging radiation with $k$ and the related shape factors $F_{k,j}$:

$$FJ_k = \sum_{j=1}^{7} F_{k,j} J_j$$

The above is valid for node $k$. The macro object *a.x_macro* corresponds to a system of the above equations with $k$ running from 1 to 7.

### 4. Object "net_radiation" and macro object "net_radiation_macro"

This equation gives the net radiant heat transfer $qr_k$ for a node $k$ in terms of all the irradiations $FJ_j$ of the nodes exchanging radiation with the considered node $k$, the generalized transmittance matrix $\tau^s{}_{k,j}$, and the $k$ node area $a_k$, radiosity $J_k$, irradiation $FJ_k$ and transmittance $\tau_k$:

$$\frac{qr_k}{a_k} = J_k - (1-\tau_k) FJ_k - \sum_{j=1}^{7} \tau^s{}_{k,j} FJ_j$$

The above is valid for node $k$. The macro object *net_radiation_macro* corresponds to a system of the above equations with $k$ running from 1 to 7.

### 5. Object "energy_balance" and macro object "energy_balance_macro"

This equation gives the energy balance at the node $k$ in terms of the node's source term $q^0{}_k$, net short wave radiant heat transfer $qr^s{}_k$, net long wave radiant heat transfer $qr^l{}_k$, the generalized conductance matrix $u^c{}_{k,j}$ (which accounts for conduction *and* convection) and the node temperatures $t_j$:

$$q^0{}_k - qr^s{}_k - qr^l{}_k = \sum_{j=1}^{7} u^c{}_{k,j} t_j$$

The above is valid for node $k$. The macroobject *energy_balance_macro* corresponds to a system of the above equations with $k$ running from 1 to 7.

The object *energy_balance* is the only one that has to be changed to switch from steady-state to dynamic simulation. All that is required is to add a derivative term with heat capacity, leading to an object *dyn_energy_balance.*

A diagram of the system without bulk air flow is shown in Fig. 3 (the complete system diagram is only slightly more complicated). The blocks in this diagram are the SPANK macro objects described above. The dashed lines show vector- or array-valued (e.g. J0 or

tau) system variables, some of which are designated as input data. Lines connecting two or more macro object interface ports indicate that a single (vector- or array-valued) system variable is identified with the corresponding macro object variables. (In addition to the equation-based macro objects there is an "adaptor" object, TAP, that allows us to "tap into" a vector in order to specify only some of its elements as problem inputs.)

Because long and short wave diffuse radiative transfer are governed by the same laws, we can use the same macro object class for both. The geometric data — areas A and view factor matrix F — are the same for both bands, but the reflectance R and transmittance tau are different. Also note that the long wave radiative transfer is coupled to the heat balance (through the *black_body* macro object) because long wave emission J0L is a function of node temperature. On the other hand, because we are ignoring the temperature dependence of short wave emission from the lamps, the short wave emission J0S is involved only in the short wave transfer; the net short wave radiation vector Qrs connects only to the energy balance.

Figure 3 represents the system model. By virtue of designation of particular system variables as "inputs," it also represents a particular "problem." One problem that can be represented (which corresponds to case (1), below) is:

**Given:**
All geometric and property data, and convection coefficients.
The short wave emission at each surface, J0S.
The source energy addition/removal rates at all surface nodes and plenum air node, Q0(1)—Q0(6).
The temperature at the room air node, T(7).

**Find:**
The temperatures at all surface nodes and plenum air node, T(1)—T(6).
The heat addition/removal rate at the room air node, Q0(7).
The short and long wave radiosities and irradiations at each node.
The short and long wave net radiant transfer rates at each node.

However, an important feature of SPANK is that different problems on the same system can be specified *without structural changes in the model.* For example, if we wished to specify a surface temperature and solve for the required heat addition/removal rate we could simply designate a different input set.

## The MACSYMA Interface to SPANK

MACSYMA is a symbolic language allowing manipulation of equations. We have written a program in this language that allows the user to enter equations or systems of equations in natural form. MACSYMA then generates all of the corresponding object files, macro object files, and function files for a SPANK problem. As an example, the object, macro object and function files (approximately 35 files in all) needed for the SPANK simulation of the above problem will be generated.

Only two types of commands are needed (others are available for general macro object generation, simulation and input file generation, and component merging):

**makespank(eq, name, list):** Creates the object *name.obj* and associated functions corresponding to equation *eq*. Equation *eq* can be piecewise defined or can be a single relationship covering the full range of its variables. Finally, *list* is the list of variables that we don't want to solve for by inverting *eq* (so-called "bad inverses" [SS89]).

**writegenericnetmacro(N, macroname, name, f1[k], f2[k,j], list):** Creates the macro *macroname.obj* that corresponds to the equation system

$$f1[k] = \sum_{j=1}^{N} f2[k,j] \text{ , for k=1 to N.}$$

Because all equations of this system are similar, **writegenericnetmacro** generates only one elementary object called *name.obj* defined as

$$f1\_1 = \sum_{j=1}^{N} f2\_1\_j$$

and then instantiates it N times, filling the variables' slots with the correct global variable name. Again, *list* is the list of variables for which we choose not to invert the above equation.

Using these commands we create the SPANK files for the example with the following sequence (Fig. 4):

(1) Create the blackbody emissivity object *blackbody.obj* and a Fahrenheit-to-Kelvin conversion object *degkdegf.obj*:

**makespank([eb=sigma*t^4, eb>0, sigma>0, t>0], "blackbody", [sigma]);**

**makespank(f=1.8*k-459.67, "degkdegf", []);**

Observe that above we have created *blackbody.obj* as an elementary object instead of a macro object in order to demonstrate the **makespank** tool. In Fig. 3 we use the equivalent macro that handles vectors instead of scalars.

(2) Create the radiosity macro object, *radiosity_macro.obj*, which is the system describing the radiosity vector in terms of itself, the surfaces properties, the shape factor matrix and the surfaces' source radiosities (which are zero except for the lamps). We also created the elementary object *radiosity.obj*, which is the equation relating to one surface only. It is used within the *radiosity_macro.obj* macro:

**writegenericnetmacro(7, "radiosity_macro", "radiosity",**
**'j[k]-j0[k]-r[k]*fj[k], taus[k,j]*fj[j]), [];**

As an example, Fig. 5 shows (a) the macro object file *radiosity_macro.obj*, (b) the elementary object file *radiosity.obj*, and (c) some **C** functions generated by this command. Note that in the above expression 'j means that j is to be taken literally to

represent radiosity, whereas the j in [j] is a summation index.

(3) Create the macro *ax.macro.obj*, which is a generic matrix-vector multiplication object, used here to obtain the irradiation vector from the shape factor matrix and the radiosity vector:

**writegenericnetmacro(7, "ax_macro", "ax", fj[k], f[k,j]*´j[j]), [];**

(4) Create the macro *net_radiation_macro.obj*, which gives the net radiant heat transfer vector in terms of the radiosities, irradiations, shape factors, surfaces, etc:

**writegenericnetmacro(7, "net_radiation_macro", "net_radiation",**
**qr[k]/a[k]-´j[k]+(1.-tau[k])*fj[k], -taus[k,j]*fj[j]), [];**

(5) Create the macro *energy_balance_macro.obj*, which performs an energy balance on each surface:

**writegenericnetmacro(7, "energy_balance_macro", "energy_balance",**
**q0[k]-qr_s[k]-qr_l[k], uc[k,j]*t[j]), [];**

## The SPANK Steady State Simulation

The steady state simulation program created by SPANK was used to carry out a parametric study of the effect of air flow rate on surface temperatures and heat removal rates in the plenum and room. Two cases were analyzed:

Case (1): The room air temperature was fixed at 75F; the supply air temperature and other variables were calculated. This corresponds to an air conditioning system in which the room air temperature is maintained at a constant value by varying the supply air temperature at fixed flow rate.

Case (2): The supply air temperature was fixed at 70F; the room air temperature and other variables were calculated.

The SPANK problem specification input file for case (1) is shown in Fig. 6. There are 86 equations and 86 unknowns. The SPANK reduction process [SBEW86] leads to an iteration set of 21 variables consisting of all the temperatures (except ceiling and room air), the ceiling long wave radiosity, and all of the short wave radiosities. The reduction ratio is 86/21, or 4.1. A solution was obtained after five iterations.

The problem specification file for case (2) (not shown) is obtained very simply from that of case (1) by changing the room air temperature, t_7, from an input to an unknown (i.e., to a link variable), and by changing the supply air temperature, t_0, from an unknown to an input. For case (2) there are also 86 equations and 21 iteration variables.

The simulation results are given in Fig. 7, which shows calculated temperatures and heat gains for values of air flow from 50 to 400 lb/hr/luminaire (1.0 to 8.0 cfm/ft$^2$). Case (1) results appear on the left side of this figure and case (2) results on the right. As expected, we observe a decrease in surface temperatures with increasing airflow. Also

shown is the the fraction of the heat from lights that is picked up by the air stream as it passes through the plenum.

## The SPANK Dynamic Simulation

To demonstrate that SPANK can be used for dynamic as well as steady-state simulation, we have run time-dependent simulations of cases (1) and (2) to determine the transient effects of going from a lights-off condition to a lights-on condition. This simulation takes into account the thermal lag due to the heat capacity of the floor, ceiling, and other nodes (as specified in Table 6). The dynamic SPANK problem specification file for case (1) is shown in Fig. 8. The differences between this file and its steady-state counterpart in Fig. 6 are very few and have been indicated by arrows.

The simulation results are shown in Fig. 9. For this study the air flow rate was set at 50 lb/hr/luminaire (1.0 cfm/ft$^2$), the lowest of the parametric values used in the steady-state simulation. A run period of 200 hours was chosen, with a time step of 6 minutes. Initially, all of the node temperatures are near the steady-state lights-on condition. Then, at time zero, the lights are turned off and remain off for 50 hours. The lights are then switched on with an input power of 120,000 Btu/hr (3.5 W/ft$^2$), the same value that was used in the steady-state runs.

The general behavior observed in Fig. 9 is an initial decrease in temperatures, followed by an asymptotic approach to equilibrium lights-off values, then a relatively rapid increase at 50 hours when the lights are turned on, followed by an asymptotic approach to equilibrium lights-on values. The initial decrease is due to the fact that the temperature starting values chosen for the simulation were above the equilibrium lights-off values.

As a check on the physical consistency and reasonableness of the results, we note that, as would be expected:

(a) With lights *off*, all of the surface temperatures for case (1) approach 75F, the fixed room air temperature. For case (2) they approach 70F, the fixed supply air temperature. The load approaches zero.

(b) With lights *on*, the surface temperatures for both cases approach those given by the steady-state calculation at minimum air flow (Fig. 7). The load approaches the lamp input power.

## Conclusions

We have demonstrated that complex SPANK objects and macro objects can be created automatically with available symbolic manipulation tools. The MACSYMA package was used although other less sophisticated packages would likely serve as well. Two MACSYMA-based programs were demonstrated. One, **makespank**, accepts a general equation as an argument string and generates the corresponding SPANK object, as well as the required inverses expressed as **C** function modules. The other,

**writegenericnetmacro**, creates a SPANK macro object (and all supporting macros, objects, and C functions) that represents a set of equations of the kind encountered in network modeling. This macro generator was shown to be general enough to create a vector-matrix product, a linear solver, and a complex radiative, convective and conductive heat transfer problem in a room. While space did not permit thorough description of the symbolic techniques employed, it has been shown that these techniques extend the SPANK methodology to an important class of problems in system simulation. The ease of switching inputs and unknowns to create different simulation problems without reprogramming was demonstrated. Finally, an example of SPANK dynamic simulation was presented as a natural extension of the corresponding steady state problem.

## Acknowledgment

We thank Daniel Sander of the National Research Council of Canada for for carefully reviewing this paper and providing many valuable comments.

## References

[ASH89] *Handbook of Fundamentals.* American Society of Heating, Refrigerating and Air-conditioning Engineers, Inc., Atlanta, GA., Ch 3.

[Mat88] Mattsson, S.E. 1988. "On Model Structuring Concepts," *Proceedings of the 4th IFAC Symposium on Computer Aided Design of Control Systems,* Beijing, China.

[Mat89] Mattsson, S.E. 1989. "Concepts Supporting Reuse of Models," *Proceedings of Building Simulation '89,* Vancouver, British Columbia. International Building Performance Simulation Association, P.O. Box 282, Orleans, Ontario, K1C 1S7, Canada, pp. 175-180.

[MG75] Mitchell and Gauthier Asso. 1975. *Advanced Continuous Simulation Language (ACSL) User Guide/Reference Manual.* P.O. Box 685, Concord, MA 01742.

[MIT83] MIT 1983. *MACSYMA Reference Manual, version 10,* Mathlab Group, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

[Sah88] Sahlin, P. 1988. "MODSIM, a Program for Dynamical Modeling and Simulation of Continuous Systems," Report from the Institute of Applied Mathematics. P.O. Box 26300, S-100 41 Stockholm, Sweden.

[SB88] Sowell, E.F. and W.F. Buhl 1988. "Dynamic Extension of the Simulation Problem Analysis Kernel (SPANK)," *Proceedings of the USER-1*

*Conference*, Ostend, Belgium. Society for Computer Simulation, La Jolla, CA.

[SBEW86] Sowell, E.F., W.F. Buhl, A.E. Erdem, and F.C. Winkelmann 1986. "A Prototype Object-based System for HVAC Simulation," *Proceedings of the Second International Conference on System Simulation in Buildings* (Liege, Belgium December). Univ. of Liege, Laboratory of Thermodynamics, B-4000 Liege, Belgium.

[SBN89] Sowell, E.F., W.F. Buhl, and J-M Nataf, 1989. "Object-oriented Programming, Equation-based Submodels, and System Reduction in SPANK," *Proceedings of Building Simulation '89,* Vancouver, British Columbia. International Building Performance Simulation Association, P.O. 282, Orleans, Ontario, K1C 1S7, Canada (June), pp. 141-146.

[SO73] Sowell, E.F. and P.F. O'Brien 1973. "The Transport of Lighting Energy," *ASHRAE Transactions*, Pt. 2.

[SS89] Sahlin, P. and E.F. Sowell. 1989. "A Neutral Format for Building Simulation Models," *Proceedings of Building Simulation '89,* Vancouver, British Columbia. International Building Performance Simulation Association, P.O. 282, Orleans, Ontario, K1C 1S7, Canada, pp. 147-154.

**Table 1**

**Radiation in Fluorescent Band**

0.30 to 0.80 microns

| Node | Reflectance | Transmittance |
|---|---|---|
| Ceiling | 0.7 | 0.0 |
| Lamp | 0.9 | 0.0 |
| Lens Top | 0.05 | 0.92 |
| Lens Bottom | 0.05 | 0.92 |
| Floor | 0.5 | 0.0 |

**Table 2**

**Radiation in Thermal Band**

1.0 to 200 microns

| Node | Reflectance | Transmittance |
|---|---|---|
| Ceiling | 0.05 | 0.0 |
| Lamp | 0.05 | 0.0 |
| Lens Top | 0.05 | 0.92 |
| Lens Bottom | 0.05 | 0.92 |
| Floor | 0.05 | 0.0 |

**Table 3**

**Film Coefficients/Conductances**

$\left(\frac{Btu/hr}{ft^2\,^\circ F}\right)$

| From | To | |
|---|---|---|
| Ceiling | Plenum Air | 0.26 |
| Lamp | Plenum Air | 1.09 |
| Lens Top | Plenum Air | 0.51 |
| Lens Bot | Room Air | 0.42 |
| Floor | Room Air | 0.74 |
| Lens Top | Lens Bot | 11.20 |

**Table 4**

**View Factors**

| From | To | |
|---|---|---|
| Ceiling | Lens Top | 0.92728 |
| Ceiling | Lamp | 0.07272 |
| Lens Top | Ceiling | 0.92728 |
| Lens Top | Lamp | 0.07272 |
| Lens Bot | Floor | 1.0 |
| Floor | Lens Bot | 1.0 |
| Lamp | Ceiling | 0.4848 |
| Lamp | Lens Top | 0.4848 |
| Lamp | Lamp | 0.0304 |

**Table 5**

**Area/Source Input**

| Node | Area $(ft^2)$ | $\frac{Q(Btu/hr)}{J^0_{SW}\left(\frac{Btu/hr}{ft^2}\right)}$ |
|---|---|---|
| Ceiling | 10000. | 0. / 0. |
| Lamp | 1500. | 120000. / 20. |
| Lens Top | 10000. | 0. / 0. |
| Lens Bot | 10000. | 0. / 0. |
| Floor | 10000. | 0. / 0. |

**Table 6**

**Heat Capacities**

| Node | Capacitance $(Btu/^oF)$ |
|---|---|
| Ceiling | 40000 |
| Lamps | 1000 |
| Lens Top | 1200 |
| Lens Bottom | 1200 |
| Floor | 40000 |
| Plenum Air | 1000 |
| Room Air | 1000 |

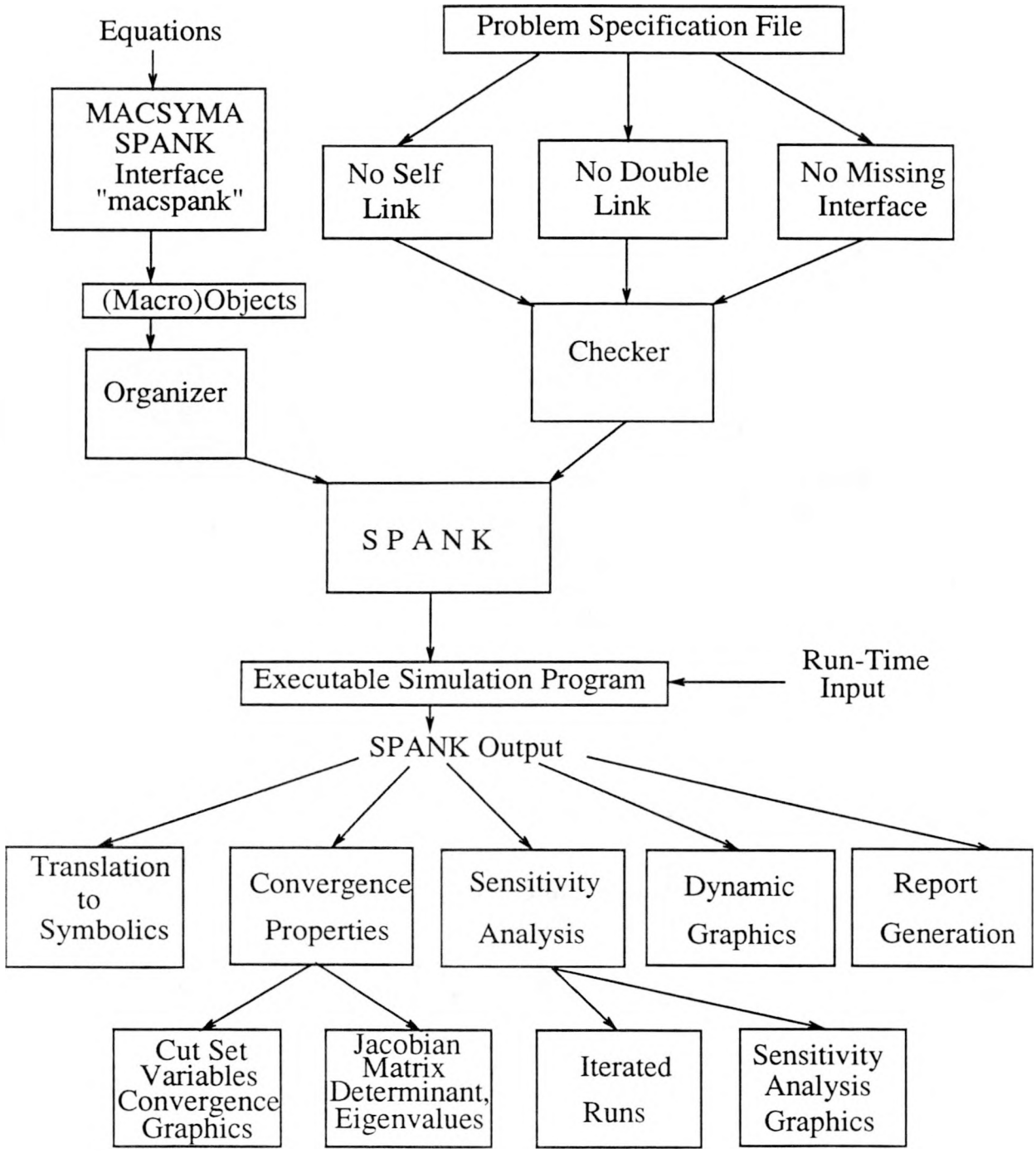



**Figure 1.** The SPANK simulation environment. From objects representing the mathematical equations of a physical system, SPANK creates an executable program that can be run to determine the steady-state or time-dependent behavior of the system. Auxiliary programs include automatic generation of C code for objects (MACSYMA/SPANK Interface), library archiving of objects (Organizer), consistency checking (Checker), results display (Dynamic Graphics), and parametric analysis (Sensitivity Analysis Graphics).

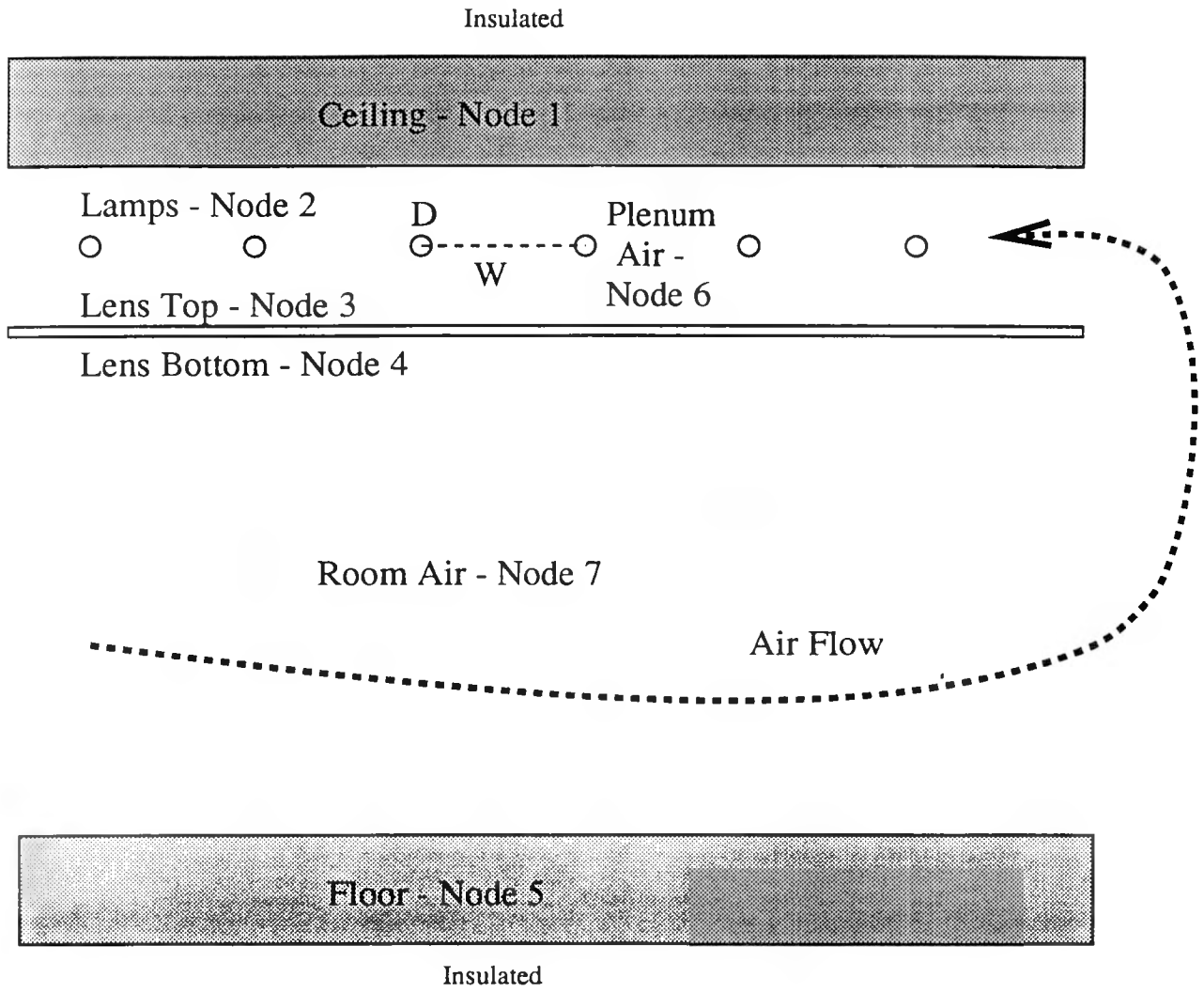



**Figure 2.** Schematic of the lighting heat transfer problem: vertical section through room and plenum.

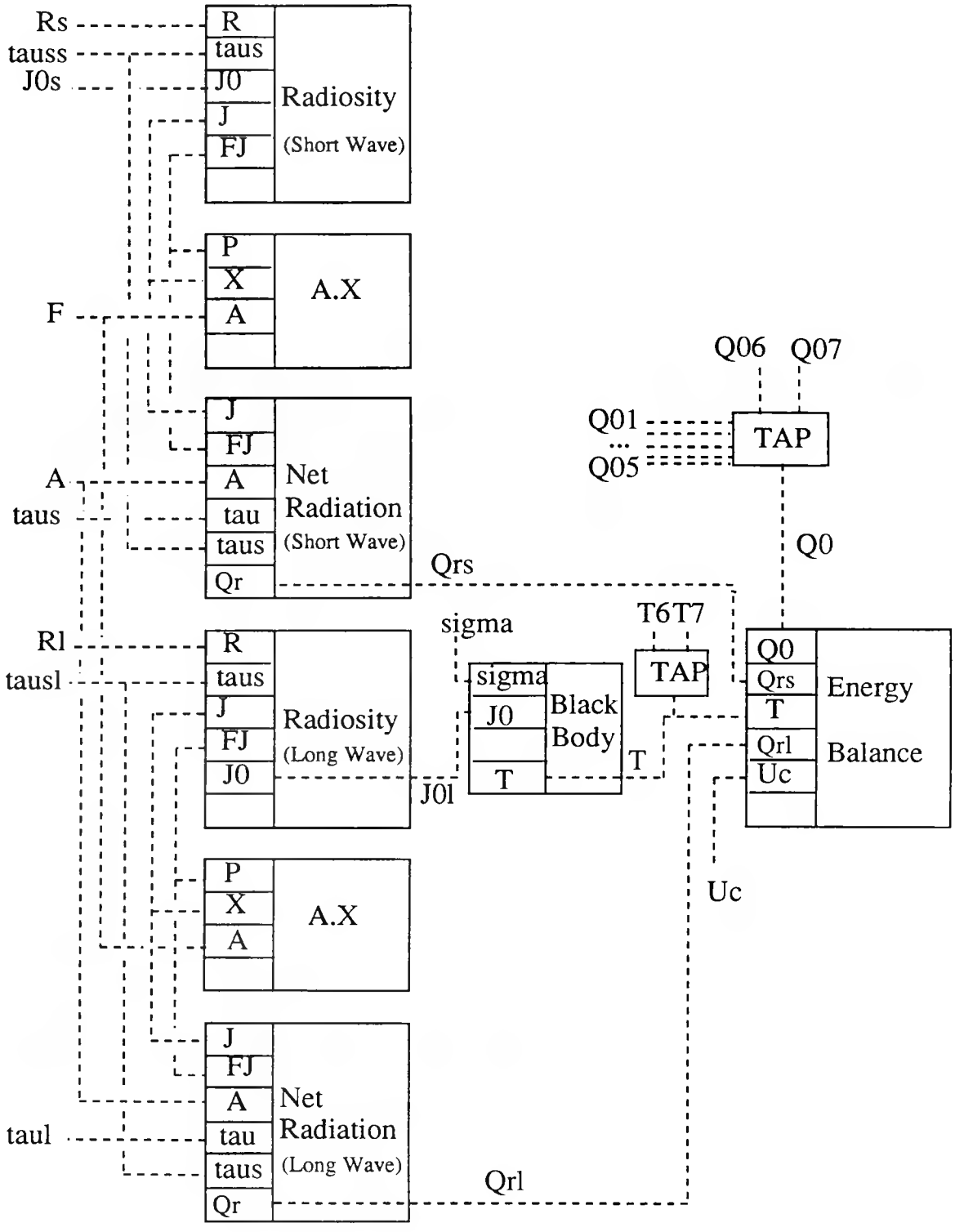



**Figure 3.** Block diagram showing macro objects. Dashed lines show either inputs or system variables shared by macro objects.

```
/*******************************************************/
/***MACSYMA COMMANDS FOR SPANK OBJECT GENERATION********/
/*******************************************************/
batch("/u1/nataf/vaxima/mysolve.mac")$
batch("/u1/nataf/vaxima/rad/radnet.mac")$

makespank([eb=sigma*t^4,eb>0,sigma>0,t>0],"blackbody",[sigma])$
makespank(f=1.8*k-459.67,"degkdegf")$
makespank(sum=x1+x2+x3,"sum3")$
writegenericnetmacro(7,"radiosity_macro",
        "radiosity",
        'j[k]-j0[k]-r[k]*fj[k],
        taus[k,j]*fj[j],[])$
writegenericnetmacro(7,"ax_macro",
        "ax",
        fj[k],
        f[k,j]*'j[j],[])$
writegenericnetmacro(7,"net_radiation_macro",
        "net_radiation",
        qr[k]/a[k]-'j[k]+(1.-tau[k])*fj[k],
        -taus[k,j]*fj[j],[])$
writegenericnetmacro(7,"energy_balance_macro",
        "energy_balance",
        q0[k]-qr_s[k]-qr_l[k],
        uc[k,j]*t[j],[])$
writegenericnetmacro(7,"lumped_radiosity_macro",
        "lumped_radiosity",
        j[k],
        ua[k,j]*j0[j],[])$
        uc[k,j]*t[j])$*/

/***DYNAMIC EXTENSION*****************************/
/*total heat balance(+ non radiative)*/
writegenericnetmacro(7,"dyn_energy_balance_macro",
        "dyn_energy_balance",
        q0[k]-qr_s[k]-qr_l[k]-'m[k]*tdot[k],
        uc[k,j]*t[j],[])$
```

**Figure 4.** Input file of MACSYMA commands for automatically generating SPANK objects.

```
/*SPANK macroobject radiosity_macro*/
macro
declare radiosity       radiosity_1;
declare radiosity       radiosity_2;
declare radiosity       radiosity_3;
declare radiosity       radiosity_4;
declare radiosity       radiosity_5;
declare radiosity       radiosity_6;
declare radiosity       radiosity_7;
link    j0_1(radiosity_1.j0_1)
link    j0_2(radiosity_2.j0_1)
link    j0_3(radiosity_3.j0_1)
link    j0_4(radiosity_4.j0_1)
link    j0_5(radiosity_5.j0_1)
link    j0_6(radiosity_6.j0_1)
link    j0_7(radiosity_7.j0_1)
link    r_1(radiosity_1.r_1)
link    r_2(radiosity_2.r_1)
link    r_3(radiosity_3.r_1)
link    r_4(radiosity_4.r_1)
link    r_5(radiosity_5.r_1)
link    r_6(radiosity_6.r_1)
link    r_7(radiosity_7.r_1)
link    j_1(radiosity_1.j_1)
link    j_2(radiosity_2.j_1)
link    j_3(radiosity_3.j_1)
link    j_4(radiosity_4.j_1)
link    j_5(radiosity_5.j_1)
link    j_6(radiosity_6.j_1)
link    j_7(radiosity_7.j_1)
link    fj_1(radiosity_1.fj_1,radiosity_2.fj_2,radiosity_3.fj_3,
        radiosity_4.fj_4,radiosity_5.fj_5,radiosity_6.fj_6,radiosity_7.fj_7)
link    fj_2(radiosity_1.fj_2,radiosity_2.fj_1,radiosity_3.fj_2,
        radiosity_4.fj_2,radiosity_5.fj_2,radiosity_6.fj_2,radiosity_7.fj_2)
link    fj_3(radiosity_1.fj_3,radiosity_2.fj_3,radiosity_3.fj_1,
        radiosity_4.fj_3,radiosity_5.fj_3,radiosity_6.fj_3,radiosity_7.fj_3)
link    fj_4(radiosity_1.fj_4,radiosity_2.fj_4,radiosity_3.fj_4,
        radiosity_4.fj_1,radiosity_5.fj_4,radiosity_6.fj_4,radiosity_7.fj_4)
link    fj_5(radiosity_1.fj_5,radiosity_2.fj_5,radiosity_3.fj_5,
        radiosity_4.fj_5,radiosity_5.fj_1,radiosity_6.fj_5,radiosity_7.fj_5)
link    fj_6(radiosity_1.fj_6,radiosity_2.fj_6,radiosity_3.fj_6,
        radiosity_4.fj_6,radiosity_5.fj_6,radiosity_6.fj_1,radiosity_7.fj_6)
link    fj_7(radiosity_1.fj_7,radiosity_2.fj_7,radiosity_3.fj_7,
        radiosity_4.fj_7,radiosity_5.fj_7,radiosity_6.fj_7,radiosity_7.fj_1)
link    taus_1_1(radiosity_1.taus_1_1)
link    taus_2_1(radiosity_2.taus_1_2)
link    taus_3_1(radiosity_3.taus_1_3)
link    taus_4_1(radiosity_4.taus_1_4)
link    taus_5_1(radiosity_5.taus_1_5)
link    taus_6_1(radiosity_6.taus_1_6)
link    taus_7_1(radiosity_7.taus_1_7)
link    taus_1_2(radiosity_1.taus_1_2)
link    taus_2_2(radiosity_2.taus_1_1)
link    taus_3_2(radiosity_3.taus_1_2)
link    taus_4_2(radiosity_4.taus_1_2)
```

**Figure 5a.** MACSYMA-generated file for the macro object *radiosity_macro.*

```
link    taus_5_2(radiosity_5.taus_1_2)
link    taus_6_2(radiosity_6.taus_1_2)
link    taus_7_2(radiosity_7.taus_1_2)
link    taus_1_3(radiosity_1.taus_1_3)
link    taus_2_3(radiosity_2.taus_1_3)
link    taus_3_3(radiosity_3.taus_1_1)
link    taus_4_3(radiosity_4.taus_1_3)
link    taus_5_3(radiosity_5.taus_1_3)
link    taus_6_3(radiosity_6.taus_1_3)
link    taus_7_3(radiosity_7.taus_1_3)
link    taus_1_4(radiosity_1.taus_1_4)
link    taus_2_4(radiosity_2.taus_1_4)
link    taus_3_4(radiosity_3.taus_1_4)
link    taus_4_4(radiosity_4.taus_1_1)
link    taus_5_4(radiosity_5.taus_1_4)
link    taus_6_4(radiosity_6.taus_1_4)
link    taus_7_4(radiosity_7.taus_1_4)
link    taus_1_5(radiosity_1.taus_1_5)
link    taus_2_5(radiosity_2.taus_1_5)
link    taus_3_5(radiosity_3.taus_1_5)
link    taus_4_5(radiosity_4.taus_1_5)
link    taus_5_5(radiosity_5.taus_1_1)
link    taus_6_5(radiosity_6.taus_1_5)
link    taus_7_5(radiosity_7.taus_1_5)
link    taus_1_6(radiosity_1.taus_1_6)
link    taus_2_6(radiosity_2.taus_1_6)
link    taus_3_6(radiosity_3.taus_1_6)
link    taus_4_6(radiosity_4.taus_1_6)
link    taus_5_6(radiosity_5.taus_1_6)
link    taus_6_6(radiosity_6.taus_1_1)
link    taus_7_6(radiosity_7.taus_1_6)
link    taus_1_7(radiosity_1.taus_1_7)
link    taus_2_7(radiosity_2.taus_1_7)
link    taus_3_7(radiosity_3.taus_1_7)
link    taus_4_7(radiosity_4.taus_1_7)
link    taus_5_7(radiosity_5.taus_1_7)
link    taus_6_7(radiosity_6.taus_1_7)
link    taus_7_7(radiosity_7.taus_1_1)
```

**Figure 5a.** (cont.)

```
/*SPANK object file radiosity.obj*/
/*equation [[-fj_1*r_1+j_1-j0_1 = fj_7*taus_1_7+fj_6*taus_1_6
 *+fj_5*taus_1_5+fj_4*taus_1_4+fj_3*taus_1_3+fj_2*taus_1_2+fj_1
 *taus_1_1]]*/
define  radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,taus_1_2,fj_3,
                  taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,
                  taus_1_6,fj_7,taus_1_7)
double j0_1;
double j_1;
double fj_1;
double r_1;
double taus_1_1;
double fj_2;
double taus_1_2;
double fj_3;
double taus_1_3;
double fj_4;
double taus_1_4;
double fj_5;
double taus_1_5;
double fj_6;
double taus_1_6;
double fj_7;
double taus_1_7;
{
        j0_1=j0_1_radiosity(j_1,fj_1,r_1,taus_1_1,fj_2,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7,
             taus_1_7);
        j_1=j_1_radiosity(j0_1,fj_1,r_1,taus_1_1,fj_2,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7,
             taus_1_7);
        fj_1=fj_1_radiosity(j0_1,j_1,r_1,taus_1_1,fj_2,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7,
             taus_1_7)*/;
        r_1=r_1_radiosity(j0_1,j_1,fj_1,taus_1_1,fj_2,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7,
             taus_1_7);
        taus_1_1=taus_1_1_radiosity(j0_1,j_1,fj_1,r_1,fj_2,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,
             fj_7,taus_1_7);
        fj_2=fj_2_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,taus_1_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7,
             taus_1_7)*/;
        taus_1_2=taus_1_2_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
             fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,
             fj_7,taus_1_7);
        fj_3=fj_3_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
             taus_1_2,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7
             taus_1_7)*/;
        taus_1_3=taus_1_3_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
             taus_1_2,fj_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,
             fj_7,taus_1_7);
        fj_4=fj_4_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
             taus_1_2,fj_3,taus_1_3,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6,fj_7
             taus_1_7)*/;
```

**Figure 5b.** MACSYMA-generated file for the elementary object *radiosity*.

```
        taus_1_4=taus_1_4_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,fj_5,taus_1_5,fj_6,taus_1_6,
            fj_7,taus_1_7);
        fj_5=fj_5_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,taus_1_5,fj_6,taus_1_6,fj_7
            taus_1_7)*/;
        taus_1_5=taus_1_5_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,fj_5,fj_6,taus_1_6,
            fj_7,taus_1_7);
        fj_6=fj_6_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,taus_1_6,fj_7
            taus_1_7)*/;
        taus_1_6=taus_1_6_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,
            fj_7,taus_1_7);
        fj_7=fj_7_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,taus_1_6
            taus_1_7)*/;
        taus_1_7=taus_1_7_radiosity(j0_1,j_1,fj_1,r_1,taus_1_1,fj_2,
            taus_1_2,fj_3,taus_1_3,fj_4,taus_1_4,fj_5,taus_1_5,fj_6,
            taus_1_6,fj_7);
}
```

**Figure 5b.** (cont.)

```
/*SPANK function file fj_1_radiosity.c*/
#include        "val.h"
#include        <math.h>
#include        <stdio.h>


#define j0_1    args[0].dval
#define j_1     args[1].dval
#define fj_1    result.dval
#define r_1     args[2].dval
#define taus_1_1        args[3].dval
#define fj_2    args[4].dval
#define taus_1_2        args[5].dval
#define fj_3    args[6].dval
#define taus_1_3        args[7].dval
#define fj_4    args[8].dval
#define taus_1_4        args[9].dval
#define fj_5    args[10].dval
#define taus_1_5        args[11].dval
#define fj_6    args[12].dval
#define taus_1_6        args[13].dval
#define fj_7    args[14].dval
#define taus_1_7        args[15].dval

VAL
fj_1_radiosity(args)
VAL args[];
{
        VAL     result;
        fj_1 = -1.0*pow(taus_1_1+r_1,-1.0)*(fj_7*taus_1_7+fj_6*taus_1_6
            +fj_5*taus_1_5+fj_4*taus_1_4+fj_3*taus_1_3+fj_2*taus_1_2-
            1.0*j_1+j0_1);
        return(result);
}
```

**Figure 5c.** MACSYMA-generated C functions for the elementary object *radiosity*.

```
/*SPANK function file j0_1_radiosity*/
#include        "val.h"
#include        <math.h>
#include        <stdio.h>


#define j0_1    result.dval
#define j_1     args[0].dval
#define fj_1    args[1].dval
#define r_1     args[2].dval
#define taus_1_1        args[3].dval
#define fj_2    args[4].dval
#define taus_1_2        args[5].dval
#define fj_3    args[6].dval
#define taus_1_3        args[7].dval
#define fj_4    args[8].dval
#define taus_1_4        args[9].dval
#define fj_5    args[10].dval
#define taus_1_5        args[11].dval
#define fj_6    args[12].dval
#define taus_1_6        args[13].dval
#define fj_7    args[14].dval
#define taus_1_7        args[15].dval

VAL
j0_1_radiosity(args)
VAL args[];
{
        VAL     result;
        j0_1 = -1.0*fj_7*taus_1_7-1.0*fj_6*taus_1_6-1.0*fj_5*taus_1_5
            -1.0*fj_4*taus_1_4-1.0*fj_3*taus_1_3-1.0*fj_2*taus_1_2-
            1.0*fj_1*taus_1_1-1.0*fj_1*r_1+j_1;
        return(result);
}
```

**Figure 5c.** (cont.)

```
/*SPANK function file j_1_radiosity.c*/
#include        "val.h"
#include        <math.h>
#include        <stdio.h>


#define j0_1    args[0].dval
#define j_1     result.dval
#define fj_1    args[1].dval
#define r_1     args[2].dval
#define taus_1_1        args[3].dval
#define fj_2    args[4].dval
#define taus_1_2        args[5].dval
#define fj_3    args[6].dval
#define taus_1_3        args[7].dval
#define fj_4    args[8].dval
#define taus_1_4        args[9].dval
#define fj_5    args[10].dval
#define taus_1_5        args[11].dval
#define fj_6    args[12].dval
#define taus_1_6        args[13].dval
#define fj_7    args[14].dval
#define taus_1_7        args[15].dval

VAL
j_1_radiosity(args)
VAL args[];
{
        VAL     result;
        j_1 = fj_7*taus_1_7+fj_6*taus_1_6+fj_5*taus_1_5+fj_4*taus_1_4
            +fj_3*taus_1_3+fj_2*taus_1_2+fj_1*taus_1_1+fj_1*r_1+ j0_1;
        return(result);
}
```

**Figure 5c.** (cont.)

```
/*SPANK function file r_1_radiosity.c*/
#include        "val.h"
#include        <math.h>
#include        <stdio.h>


#define j0_1    args[0].dval
#define j_1     args[1].dval
#define fj_1    args[2].dval
#define r_1     result.dval
#define taus_1_1        args[3].dval
#define fj_2    args[4].dval
#define taus_1_2        args[5].dval
#define fj_3    args[6].dval
#define taus_1_3        args[7].dval
#define fj_4    args[8].dval
#define taus_1_4        args[9].dval
#define fj_5    args[10].dval
#define taus_1_5        args[11].dval
#define fj_6    args[12].dval
#define taus_1_6        args[13].dval
#define fj_7    args[14].dval
#define taus_1_7        args[15].dval

VAL
r_1_radiosity(args)
VAL args[];
{
        VAL     result;
        r_1 = -1.0*(fj_7*taus_1_7+fj_6*taus_1_6+fj_5*taus_1_5
            +fj_4*taus_1_4+fj_3*taus_1_3+fj_2*taus_1_2+fj_1*taus_1_1-1.0*
            j_1+j0_1)/fj_1;
        return(result);
}
```

**Figure 5c.** (cont.)

```
/*SPANK function file taus_1_1_radiosity.c*/
#include        "val.h"
#include        <math.h>
#include        <stdio.h>


#define j0_1    args[0].dval
#define j_1     args[1].dval
#define fj_1    args[2].dval
#define r_1     args[3].dval
#define taus_1_1        result.dval
#define fj_2    args[4].dval
#define taus_1_2        args[5].dval
#define fj_3    args[6].dval
#define taus_1_3        args[7].dval
#define fj_4    args[8].dval
#define taus_1_4        args[9].dval
#define fj_5    args[10].dval
#define taus_1_5        args[11].dval
#define fj_6    args[12].dval
#define taus_1_6        args[13].dval
#define fj_7    args[14].dval
#define taus_1_7        args[15].dval

VAL
taus_1_1_radiosity(args)
VAL args[];
{
        VAL     result;
        taus_1_1 = -1.0*(fj_7*taus_1_7+fj_6*taus_1_6+fj_5*taus_1_5
            +fj_4*taus_1_4+fj_3*taus_1_3+fj_2*taus_1_2+fj_1*r_1-1.0*
            j_1+j0_1)/fj_1;
        return(result);
}
```

**Figure 5c.** (cont.)

```
/*SPANK simulation file for lighting problem*/
/*SPANK file m_ed_light7_3.ps*/
/*Short wave are indicated by sw, long wave by lw*/
/*The problem is the following: we have a ceiling and a floor,
 *both insulated, with a plenum with lamps in it below the ceiling,
 *A translucent lens allows air flow between room and plenum*/
/*We have 5 nodes for radiative heat transfer:node 1 is ceiling,
 *2 is lamp,3 is top of lens, 4 is bottom of lens and 5 is floor*/
/*Node 6 is plenum air and node 7 is room air*/
/*Same as ed_light5, except that air nodes are put into the node
 *list*/
/*Units are Btu,hrs,lb,ft,kelvin (for tk's) and fahrenheit
 *(anywhere else)*/
/*Includes mass flow*/
/*Difference with m_ed_light7_2 is that there is no hplenum.
/*Outside temperature t_0 is an unknown, room temperature t_7 is an input*/
/*MAIN SIMULATION FOR STEADY STATE SIMULATION*/

declare radiosity_macro s,l;     /*Short wave and Long wave simulation*/
declare ax_macro        fjs,fjl;         /*FJ vectors, F shape factor, J radi
declare net_radiation_macro      fjqrs,fjqrl;     /*Net radiation in each band
declare heat_balance_macro       h;       /*Heat balance on each node*/
declare blackbody       eb1,eb2,eb3,eb4,eb5,eb6,eb7;
declare degkdegf        kf1,kf2,kf3,kf4,kf5,kf6,kf7;
declare sum             qr1,qr2,qr3,qr4,qr5,qr6,qr7;
declare mprod           mp;
declare msum6           uc1,uc2,uc3,uc4,uc5,uc6,uc7;

declare heat_add        hroom,conv76;
declare prod            rr;
declare heat_add        wroom,wplenum;
declare hdbw            hr,hp,hs;

/*Mass flow and specific heat of air*/
input   m(mp.in1,hroom.m,wroom.m,wplenum.m,conv76.m)
input   cp(mp.in2)

input   rs_1(s.r_1)
input   rs_2(s.r_2)
input   rs_3(s.r_3)
input   rs_4(s.r_4)
input   rs_5(s.r_5)
input   rs_6(s.r_6)
input   rs_7(s.r_7)

input   rl_1(l.r_1)
input   rl_2(l.r_2)
input   rl_3(l.r_3)
input   rl_4(l.r_4)
input   rl_5(l.r_5)
input   rl_6(l.r_6)
input   rl_7(l.r_7)

/*Short wave transmittances*/
input   taus_1(fjqrs.tau_1)
input   taus_2(fjqrs.tau_2)
```

**Figure 6.** Problem specification file for steady-state problem, case (1): fixed room air temperature. The network that describes the problem is formed by linking together macro objects that represent system component models and by assigning input quantities.

```
input   taus_3(fjqrs.tau_3)
input   taus_4(fjqrs.tau_4)
input   taus_5(fjqrs.tau_5)
input   taus_6(fjqrs.tau_6)
input   taus_7(fjqrs.tau_7)

/*Long wave transmittances*/
input   taul_1(fjqrl.tau_1)
input   taul_2(fjqrl.tau_2)
input   taul_3(fjqrl.tau_3)
input   taul_4(fjqrl.tau_4)
input   taul_5(fjqrl.tau_5)
input   taul_6(fjqrl.tau_6)
input   taul_7(fjqrl.tau_7)

/*Areas*/
input   a_1(fjqrs.a_1,fjqrl.a_1)
input   a_2(fjqrs.a_2,fjqrl.a_2)
input   a_3(fjqrs.a_3,fjqrl.a_3)
input   a_4(fjqrs.a_4,fjqrl.a_4)
input   a_5(fjqrs.a_5,fjqrl.a_5)
input   a_6(fjqrs.a_6,fjqrl.a_6)
input   a_7(fjqrs.a_7,fjqrl.a_7)

/*Shape factors*/
input   f_1_1(fjs.f_1_1,fjl.f_1_1)
input   f_1_2(fjs.f_1_2,fjl.f_1_2)
input   f_1_3(fjs.f_1_3,fjl.f_1_3)
input   f_1_4(fjs.f_1_4,fjl.f_1_4)
input   f_1_5(fjs.f_1_5,fjl.f_1_5)
input   f_1_6(fjs.f_1_6,fjl.f_1_6)
input   f_1_7(fjs.f_1_7,fjl.f_1_7)

input   f_2_1(fjs.f_2_1,fjl.f_2_1)
input   f_2_2(fjs.f_2_2,fjl.f_2_2)
input   f_2_3(fjs.f_2_3,fjl.f_2_3)
input   f_2_4(fjs.f_2_4,fjl.f_2_4)
input   f_2_5(fjs.f_2_5,fjl.f_2_5)
input   f_2_6(fjs.f_2_6,fjl.f_2_6)
input   f_2_7(fjs.f_2_7,fjl.f_2_7)

input   f_3_1(fjs.f_3_1,fjl.f_3_1)
input   f_3_2(fjs.f_3_2,fjl.f_3_2)
input   f_3_3(fjs.f_3_3,fjl.f_3_3)
input   f_3_4(fjs.f_3_4,fjl.f_3_4)
input   f_3_5(fjs.f_3_5,fjl.f_3_5)
input   f_3_6(fjs.f_3_6,fjl.f_3_6)
input   f_3_7(fjs.f_3_7,fjl.f_3_7)

input   f_4_1(fjs.f_4_1,fjl.f_4_1)
input   f_4_2(fjs.f_4_2,fjl.f_4_2)
input   f_4_3(fjs.f_4_3,fjl.f_4_3)
input   f_4_4(fjs.f_4_4,fjl.f_4_4)
input   f_4_5(fjs.f_4_5,fjl.f_4_5)
input   f_4_6(fjs.f_4_6,fjl.f_4_6)
input   f_4_7(fjs.f_4_7,fjl.f_4_7)
```

**Figure 6.** (cont.)

```
input    f_5_1(fjs.f_5_1,fjl.f_5_1)
input    f_5_2(fjs.f_5_2,fjl.f_5_2)
input    f_5_3(fjs.f_5_3,fjl.f_5_3)
input    f_5_4(fjs.f_5_4,fjl.f_5_4)
input    f_5_5(fjs.f_5_5,fjl.f_5_5)
input    f_5_6(fjs.f_5_6,fjl.f_5_6)
input    f_5_7(fjs.f_5_7,fjl.f_5_7)

input    f_6_1(fjs.f_6_1,fjl.f_6_1)
input    f_6_2(fjs.f_6_2,fjl.f_6_2)
input    f_6_3(fjs.f_6_3,fjl.f_6_3)
input    f_6_4(fjs.f_6_4,fjl.f_6_4)
input    f_6_5(fjs.f_6_5,fjl.f_6_5)
input    f_6_6(fjs.f_6_6,fjl.f_6_6)
input    f_6_7(fjs.f_6_7,fjl.f_6_7)

input    f_7_1(fjs.f_7_1,fjl.f_7_1)
input    f_7_2(fjs.f_7_2,fjl.f_7_2)
input    f_7_3(fjs.f_7_3,fjl.f_7_3)
input    f_7_4(fjs.f_7_4,fjl.f_7_4)
input    f_7_5(fjs.f_7_5,fjl.f_7_5)
input    f_7_6(fjs.f_7_6,fjl.f_7_6)
input    f_7_7(fjs.f_7_7,fjl.f_7_7)

/*Generalized conductance convectance matrix*/
link     uc_1_1(h.uc_1_1,uc1.msum)
input    uc_1_2(h.uc_1_2,uc1.x1)
input    uc_1_3(h.uc_1_3,uc1.x2)
input    uc_1_4(h.uc_1_4,uc1.x3)
input    uc_1_5(h.uc_1_5,uc1.x4)
input    uc_1_6(h.uc_1_6,uc1.x5)
input    uc_1_7(h.uc_1_7,uc1.x6)

input    uc_2_1(h.uc_2_1,uc2.x1)
link     uc_2_2(h.uc_2_2,uc2.msum)
input    uc_2_3(h.uc_2_3,uc2.x2)
input    uc_2_4(h.uc_2_4,uc2.x3)
input    uc_2_5(h.uc_2_5,uc2.x4)
input    uc_2_6(h.uc_2_6,uc2.x5)
input    uc_2_7(h.uc_2_7,uc2.x6)

input    uc_3_1(h.uc_3_1,uc3.x1)
input    uc_3_2(h.uc_3_2,uc3.x2)
link     uc_3_3(h.uc_3_3,uc3.msum)
input    uc_3_4(h.uc_3_4,uc3.x3)
input    uc_3_5(h.uc_3_5,uc3.x4)
input    uc_3_6(h.uc_3_6,uc3.x5)
input    uc_3_7(h.uc_3_7,uc3.x6)

input    uc_4_1(h.uc_4_1,uc4.x1)
input    uc_4_2(h.uc_4_2,uc4.x2)
input    uc_4_3(h.uc_4_3,uc4.x3)
link     uc_4_4(h.uc_4_4,uc4.msum)
input    uc_4_5(h.uc_4_5,uc4.x4)
input    uc_4_6(h.uc_4_6,uc4.x5)
```

**Figure 6.** (cont.)

```
input   uc_4_7(h.uc_4_7,uc4.x6)

input   uc_5_1(h.uc_5_1,uc5.x1)
input   uc_5_2(h.uc_5_2,uc5.x2)
input   uc_5_3(h.uc_5_3,uc5.x3)
input   uc_5_4(h.uc_5_4,uc5.x4)
link    uc_5_5(h.uc_5_5,uc5.msum)
input   uc_5_6(h.uc_5_6,uc5.x5)
input   uc_5_7(h.uc_5_7,uc5.x6)

input   uc_6_1(h.uc_6_1,uc6.x1)
input   uc_6_2(h.uc_6_2,uc6.x2)
input   uc_6_3(h.uc_6_3,uc6.x3)
input   uc_6_4(h.uc_6_4,uc6.x4)
input   uc_6_5(h.uc_6_5,uc6.x5)
link    uc_6_6(h.uc_6_6,uc6.msum)
link    uc_6_7(h.uc_6_7,mp.mproduct,uc6.x6)

input   uc_7_1(h.uc_7_1,uc7.x1)
input   uc_7_2(h.uc_7_2,uc7.x2)
input   uc_7_3(h.uc_7_3,uc7.x3)
input   uc_7_4(h.uc_7_4,uc7.x4)
input   uc_7_5(h.uc_7_5,uc7.x5)
input   uc_7_6(h.uc_7_6,uc7.x6)
link    uc_7_7(h.uc_7_7,uc7.msum)

/*Special short wave transmittance matrix*/
input   tauss_1_1(s.taus_1_1,fjqrs.taus_1_1)
input   tauss_1_2(s.taus_1_2,fjqrs.taus_1_2)
input   tauss_1_3(s.taus_1_3,fjqrs.taus_1_3)
input   tauss_1_4(s.taus_1_4,fjqrs.taus_1_4)
input   tauss_1_5(s.taus_1_5,fjqrs.taus_1_5)
input   tauss_1_6(s.taus_1_6,fjqrs.taus_1_6)
input   tauss_1_7(s.taus_1_7,fjqrs.taus_1_7)

input   tauss_2_1(s.taus_2_1,fjqrs.taus_2_1)
input   tauss_2_2(s.taus_2_2,fjqrs.taus_2_2)
input   tauss_2_3(s.taus_2_3,fjqrs.taus_2_3)
input   tauss_2_4(s.taus_2_4,fjqrs.taus_2_4)
input   tauss_2_5(s.taus_2_5,fjqrs.taus_2_5)
input   tauss_2_6(s.taus_2_6,fjqrs.taus_2_6)
input   tauss_2_7(s.taus_2_7,fjqrs.taus_2_7)

input   tauss_3_1(s.taus_3_1,fjqrs.taus_3_1)
input   tauss_3_2(s.taus_3_2,fjqrs.taus_3_2)
input   tauss_3_3(s.taus_3_3,fjqrs.taus_3_3)
input   tauss_3_4(s.taus_3_4,fjqrs.taus_3_4)
input   tauss_3_5(s.taus_3_5,fjqrs.taus_3_5)
input   tauss_3_6(s.taus_3_6,fjqrs.taus_3_6)
input   tauss_3_7(s.taus_3_7,fjqrs.taus_3_7)

input   tauss_4_1(s.taus_4_1,fjqrs.taus_4_1)
input   tauss_4_2(s.taus_4_2,fjqrs.taus_4_2)
input   tauss_4_3(s.taus_4_3,fjqrs.taus_4_3)
input   tauss_4_4(s.taus_4_4,fjqrs.taus_4_4)
input   tauss_4_5(s.taus_4_5,fjqrs.taus_4_5)
```

**Figure 6.** (cont.)

```
input   tauss_4_6(s.taus_4_6,fjqrs.taus_4_6)
input   tauss_4_7(s.taus_4_7,fjqrs.taus_4_7)

input   tauss_5_1(s.taus_5_1,fjqrs.taus_5_1)
input   tauss_5_2(s.taus_5_2,fjqrs.taus_5_2)
input   tauss_5_3(s.taus_5_3,fjqrs.taus_5_3)
input   tauss_5_4(s.taus_5_4,fjqrs.taus_5_4)
input   tauss_5_5(s.taus_5_5,fjqrs.taus_5_5)
input   tauss_5_6(s.taus_5_6,fjqrs.taus_5_6)
input   tauss_5_7(s.taus_5_7,fjqrs.taus_5_7)

input   tauss_6_1(s.taus_6_1,fjqrs.taus_6_1)
input   tauss_6_2(s.taus_6_2,fjqrs.taus_6_2)
input   tauss_6_3(s.taus_6_3,fjqrs.taus_6_3)
input   tauss_6_4(s.taus_6_4,fjqrs.taus_6_4)
input   tauss_6_5(s.taus_6_5,fjqrs.taus_6_5)
input   tauss_6_6(s.taus_6_6,fjqrs.taus_6_6)
input   tauss_6_7(s.taus_6_7,fjqrs.taus_6_7)

input   tauss_7_1(s.taus_7_1,fjqrs.taus_7_1)
input   tauss_7_2(s.taus_7_2,fjqrs.taus_7_2)
input   tauss_7_3(s.taus_7_3,fjqrs.taus_7_3)
input   tauss_7_4(s.taus_7_4,fjqrs.taus_7_4)
input   tauss_7_5(s.taus_7_5,fjqrs.taus_7_5)
input   tauss_7_6(s.taus_7_6,fjqrs.taus_7_6)
input   tauss_7_7(s.taus_7_7,fjqrs.taus_7_7)


/*Special long wave transmittance matrix*/
input   tausl_1_1(l.taus_1_1,fjqrl.taus_1_1)
input   tausl_1_2(l.taus_1_2,fjqrl.taus_1_2)
input   tausl_1_3(l.taus_1_3,fjqrl.taus_1_3)
input   tausl_1_4(l.taus_1_4,fjqrl.taus_1_4)
input   tausl_1_5(l.taus_1_5,fjqrl.taus_1_5)
input   tausl_1_6(l.taus_1_6,fjqrl.taus_1_6)
input   tausl_1_7(l.taus_1_7,fjqrl.taus_1_7)

input   tausl_2_1(l.taus_2_1,fjqrl.taus_2_1)
input   tausl_2_2(l.taus_2_2,fjqrl.taus_2_2)
input   tausl_2_3(l.taus_2_3,fjqrl.taus_2_3)
input   tausl_2_4(l.taus_2_4,fjqrl.taus_2_4)
input   tausl_2_5(l.taus_2_5,fjqrl.taus_2_5)
input   tausl_2_6(l.taus_2_6,fjqrl.taus_2_6)
input   tausl_2_7(l.taus_2_7,fjqrl.taus_2_7)

input   tausl_3_1(l.taus_3_1,fjqrl.taus_3_1)
input   tausl_3_2(l.taus_3_2,fjqrl.taus_3_2)
input   tausl_3_3(l.taus_3_3,fjqrl.taus_3_3)
input   tausl_3_4(l.taus_3_4,fjqrl.taus_3_4)
input   tausl_3_5(l.taus_3_5,fjqrl.taus_3_5)
input   tausl_3_6(l.taus_3_6,fjqrl.taus_3_6)
input   tausl_3_7(l.taus_3_7,fjqrl.taus_3_7)

input   tausl_4_1(l.taus_4_1,fjqrl.taus_4_1)
input   tausl_4_2(l.taus_4_2,fjqrl.taus_4_2)
input   tausl_4_3(l.taus_4_3,fjqrl.taus_4_3)
```

**Figure 6.** (cont.)

```
input   tausl_4_4(l.taus_4_4,fjqrl.taus_4_4)
input   tausl_4_5(l.taus_4_5,fjqrl.taus_4_5)
input   tausl_4_6(l.taus_4_6,fjqrl.taus_4_6)
input   tausl_4_7(l.taus_4_7,fjqrl.taus_4_7)

input   tausl_5_1(l.taus_5_1,fjqrl.taus_5_1)
input   tausl_5_2(l.taus_5_2,fjqrl.taus_5_2)
input   tausl_5_3(l.taus_5_3,fjqrl.taus_5_3)
input   tausl_5_4(l.taus_5_4,fjqrl.taus_5_4)
input   tausl_5_5(l.taus_5_5,fjqrl.taus_5_5)
input   tausl_5_6(l.taus_5_6,fjqrl.taus_5_6)
input   tausl_5_7(l.taus_5_7,fjqrl.taus_5_7)

input   tausl_6_1(l.taus_6_1,fjqrl.taus_6_1)
input   tausl_6_2(l.taus_6_2,fjqrl.taus_6_2)
input   tausl_6_3(l.taus_6_3,fjqrl.taus_6_3)
input   tausl_6_4(l.taus_6_4,fjqrl.taus_6_4)
input   tausl_6_5(l.taus_6_5,fjqrl.taus_6_5)
input   tausl_6_6(l.taus_6_6,fjqrl.taus_6_6)
input   tausl_6_7(l.taus_6_7,fjqrl.taus_6_7)

input   tausl_7_1(l.taus_7_1,fjqrl.taus_7_1)
input   tausl_7_2(l.taus_7_2,fjqrl.taus_7_2)
input   tausl_7_3(l.taus_7_3,fjqrl.taus_7_3)
input   tausl_7_4(l.taus_7_4,fjqrl.taus_7_4)
input   tausl_7_5(l.taus_7_5,fjqrl.taus_7_5)
input   tausl_7_6(l.taus_7_6,fjqrl.taus_7_6)
input   tausl_7_7(l.taus_7_7,fjqrl.taus_7_7)


/*Short wave irradiation*/
link    fjs_1(s.fj_1,fjqrs.fj_1,fjs.fj_1)[fjs1]
link    fjs_2(s.fj_2,fjqrs.fj_2,fjs.fj_2)[fjs2]
link    fjs_3(s.fj_3,fjqrs.fj_3,fjs.fj_3)[fjs3]
link    fjs_4(s.fj_4,fjqrs.fj_4,fjs.fj_4)[fjs4]
link    fjs_5(s.fj_5,fjqrs.fj_5,fjs.fj_5)[fjs5]
link    fjs_6(s.fj_6,fjqrs.fj_6,fjs.fj_6)[fjs6]
link    fjs_7(s.fj_7,fjqrs.fj_7,fjs.fj_7)[fjs7]

/*Long wave irradiation*/
link    fjl_1(l.fj_1,fjqrl.fj_1,fjl.fj_1)[fjl1]
link    fjl_2(l.fj_2,fjqrl.fj_2,fjl.fj_2)[fjl2]
link    fjl_3(l.fj_3,fjqrl.fj_3,fjl.fj_3)[fjl3]
link    fjl_4(l.fj_4,fjqrl.fj_4,fjl.fj_4)[fjl4]
link    fjl_5(l.fj_5,fjqrl.fj_5,fjl.fj_5)[fjl5]
link    fjl_6(l.fj_6,fjqrl.fj_6,fjl.fj_6)[fjl6]
link    fjl_7(l.fj_7,fjqrl.fj_7,fjl.fj_7)[fjl7]


/*Temperatures in fahrenheits*/
link    t_1(kf1.f,h.t_1)[T]
link    t_2(kf2.f,h.t_2)[T]
link    t_3(kf3.f,h.t_3)[T]
link    t_4(kf4.f,h.t_4)[T]
link    t_5(kf5.f,h.t_5)[T]
link    t_6(kf6.f,h.t_6,hp.db)[T]
```

**Figure 6.** (cont.)

```
input   t_7(kf7.f,h.t_7,hr.db)[T]

/*Temperatures in kelvins*/
link    tk_1(kf1.k,eb1.t)
link    tk_2(kf2.k,eb2.t)
link    tk_3(kf3.k,eb3.t)
link    tk_4(kf4.k,eb4.t)
link    tk_5(kf5.k,eb5.t)
link    tk_6(kf6.k,eb6.t)
link    tk_7(kf7.k,eb7.t)

/*Net short wave radiant heat transfer*/
link    qr_s_1(fjqrs.qr_1,qr1.in1,h.qr_s_1)
link    qr_s_2(fjqrs.qr_2,qr2.in1,h.qr_s_2)
link    qr_s_3(fjqrs.qr_3,qr3.in1,h.qr_s_3)
link    qr_s_4(fjqrs.qr_4,qr4.in1,h.qr_s_4)
link    qr_s_5(fjqrs.qr_5,qr5.in1,h.qr_s_5)
link    qr_s_6(fjqrs.qr_6,qr6.in1,h.qr_s_6)
link    qr_s_7(fjqrs.qr_7,qr7.in1,h.qr_s_7)

/*Net long wave radiant heat transfer*/
link    qr_l_1(fjqrl.qr_1,qr1.in2,h.qr_l_1)
link    qr_l_2(fjqrl.qr_2,qr2.in2,h.qr_l_2)
link    qr_l_3(fjqrl.qr_3,qr3.in2,h.qr_l_3)
link    qr_l_4(fjqrl.qr_4,qr4.in2,h.qr_l_4)
link    qr_l_5(fjqrl.qr_5,qr5.in2,h.qr_l_5)
link    qr_l_6(fjqrl.qr_6,qr6.in2,h.qr_l_6)
link    qr_l_7(fjqrl.qr_7,qr7.in2,h.qr_l_7)

/*Source power input = 0 except on lamp*/
input   q0_1(h.q0_1)
input   q0_2(h.q0_2,rr.in1)
input   q0_3(h.q0_3)
input   q0_4(h.q0_4)
input   q0_5(h.q0_5)
input   q0_6(h.q0_6)[q06]
link    q0_7(h.q0_7,hroom.q)[q07]

/*Total net radiant heat transfer on each node*/
link    qr_1(qr1.sum)
link    qr_2(qr2.sum)
link    qr_3(qr3.sum)
link    qr_4(qr4.sum)
link    qr_5(qr5.sum)
link    qr_6(qr6.sum)
link    qr_7(qr7.sum)

/*Short wave radiosity source vector = 0 except for lamp (node 2)*/
input   j0s_1(s.j0_1)
input   j0s_2(s.j0_2)
input   j0s_3(s.j0_3)
input   j0s_4(s.j0_4)
input   j0s_5(s.j0_5)
input   j0s_6(s.j0_6)
input   j0s_7(s.j0_7)
```

**Figure 6.** (cont.)

```
/*Long wave radiosity source vector = blackbody emission*/
link    j01_1(l.j0_1,eb1.eb)
link    j01_2(l.j0_2,eb2.eb)
link    j01_3(l.j0_3,eb3.eb)
link    j01_4(l.j0_4,eb4.eb)
link    j01_5(l.j0_5,eb5.eb)
link    j01_6(l.j0_6,eb6.eb)[eb]
link    j01_7(l.j0_7,eb7.eb)[eb]

/*Stefan Boltzmann constant*/
input   sigma(eb1.sigma,eb2.sigma,eb3.sigma,eb4.sigma,eb5.sigma,eb6.sigma,eb

/*Short wave radiosities*/
link    js_1(s.j_1,fjs.j_1,fjqrs.j_1)
link    js_2(s.j_2,fjs.j_2,fjqrs.j_2)
link    js_3(s.j_3,fjs.j_3,fjqrs.j_3)
link    js_4(s.j_4,fjs.j_4,fjqrs.j_4)
link    js_5(s.j_5,fjs.j_5,fjqrs.j_5)
link    js_6(s.j_6,fjs.j_6,fjqrs.j_6)
link    js_7(s.j_7,fjs.j_7,fjqrs.j_7)

/*Long wave radiosities*/
link    jl_1(l.j_1,fjl.j_1,fjqrl.j_1)[jl1]
link    jl_2(l.j_2,fjl.j_2,fjqrl.j_2)[jl2]
link    jl_3(l.j_3,fjl.j_3,fjqrl.j_3)[jl3]
link    jl_4(l.j_4,fjl.j_4,fjqrl.j_4)[jl4]
link    jl_5(l.j_5,fjl.j_5,fjqrl.j_5)[jl5]
link    jl_6(l.j_6,fjl.j_6,fjqrl.j_6)[jl6]
link    jl_7(l.j_7,fjl.j_7,fjqrl.j_7)[jl7]

/*Enthalpies. The strange "outs" and "ins" are due to the fact that q0_6
 *and q0_7 are source power inputs to the nodes, and thus are calculated
 *the opposite way as is usually done in heat_add object.*/
/*Thus q0_7=m(h_0-h_7)*/
link    h_0(hs.h,hroom.h_out)
link    h_6(hp.h,conv76.h_out)
link    h_7(hr.h,hroom.h_in,conv76.h_in)
link    q76(conv76.q,rr.product)
link    r(rr.in2)

/*Supply temperature*/
link    t_0(hs.db)

/*Humidities*/
input   w_0(hs.w,wroom.h_in)
link    w_6(hp.w,wplenum.h_out)
link    w_7(hr.w,wroom.h_out,wplenum.h_in)

/*Humidity source*/
input   wroom_added(wroom.q)
input   wplenum_added(wplenum.q)
```

**Figure 6.** (cont.)

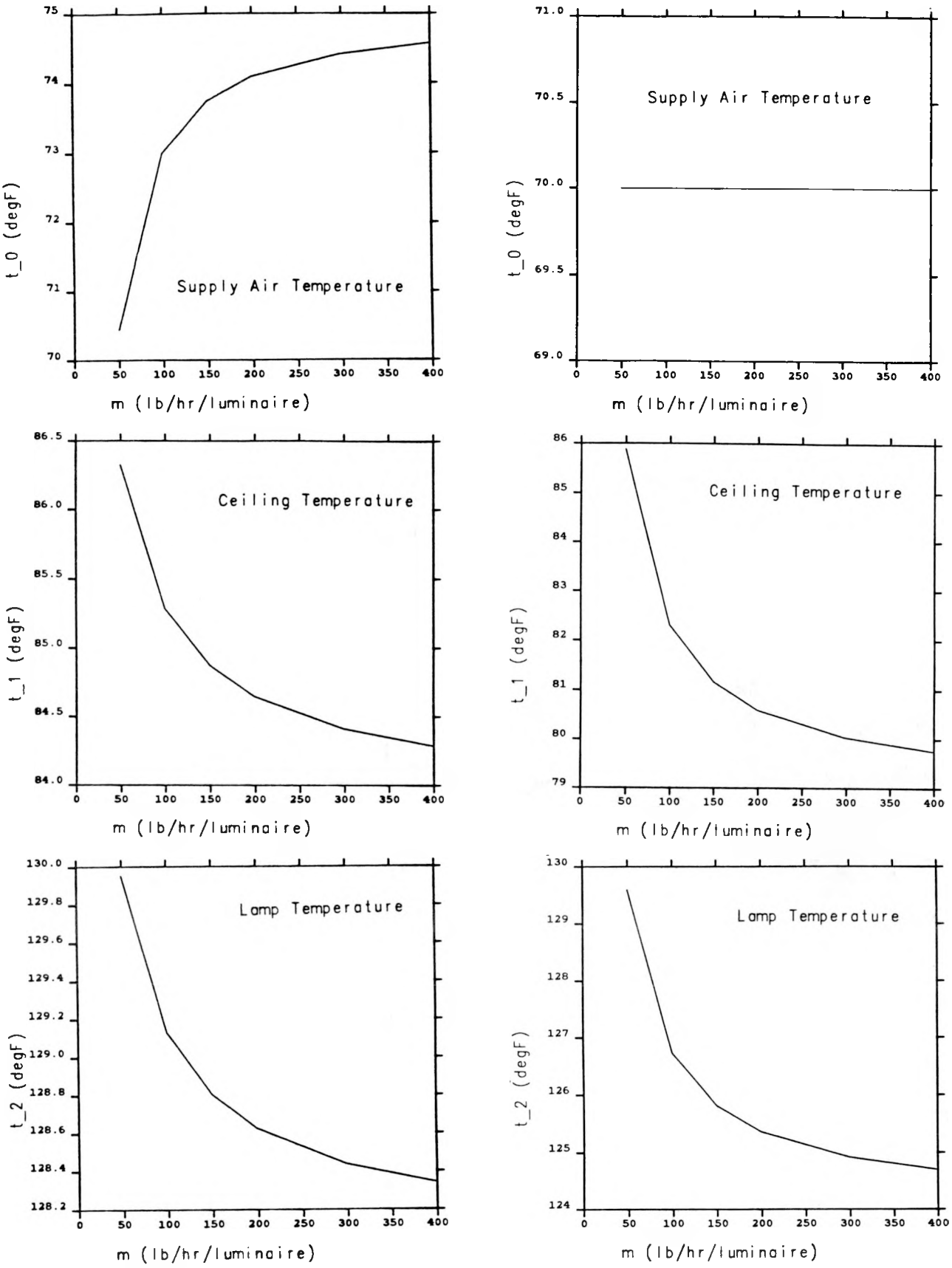



**Figure 7.** SPANK steady-state simulation results for case (1), room air temperature fixed at 75F (**left-hand graphs**), and case (2), supply air temperature fixed at 70F (**right-hand graphs**), as a function of air flow rate.

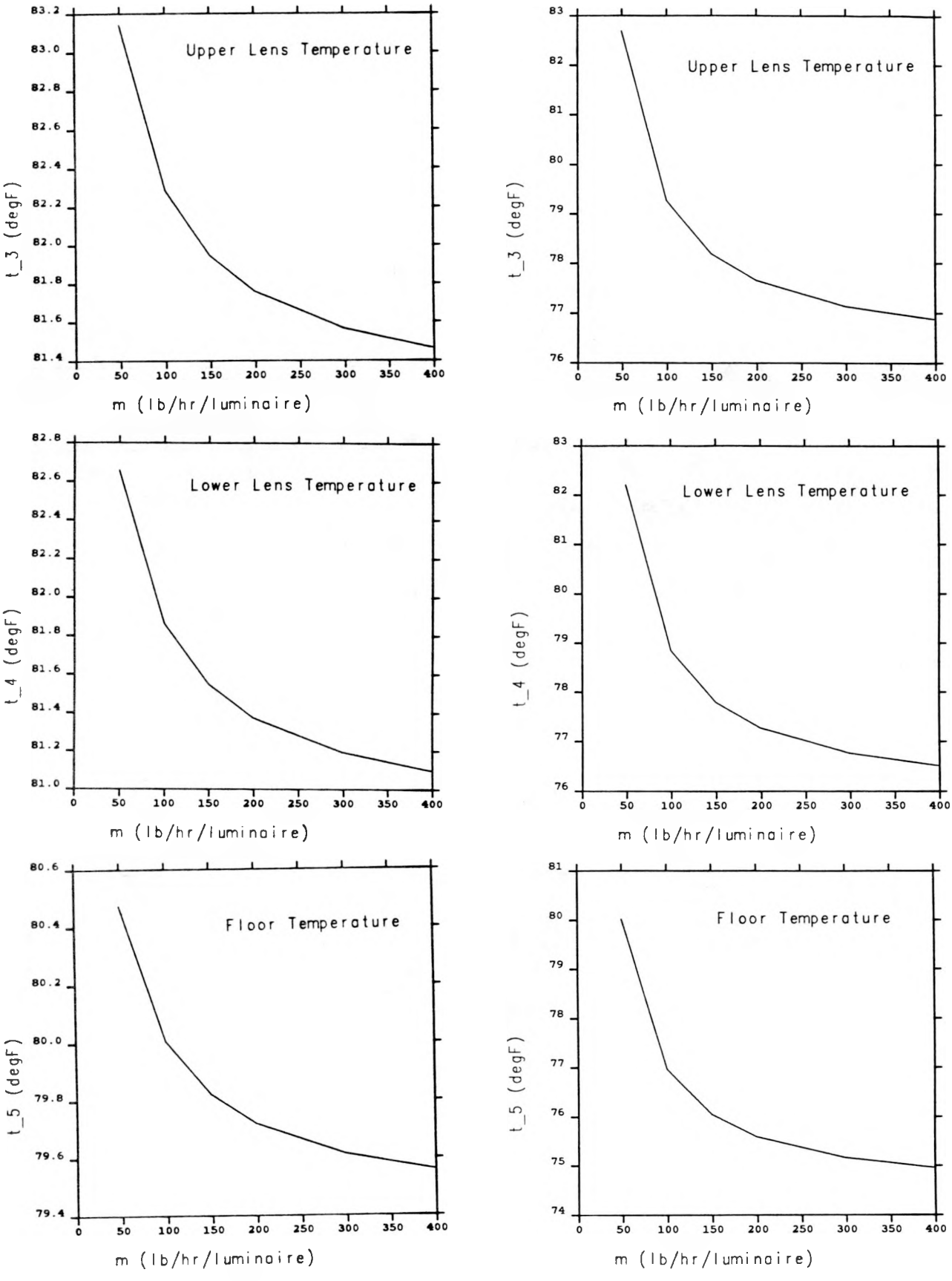
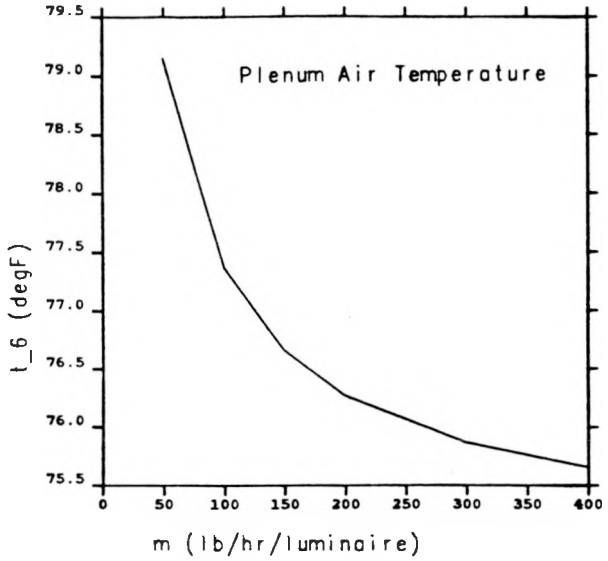
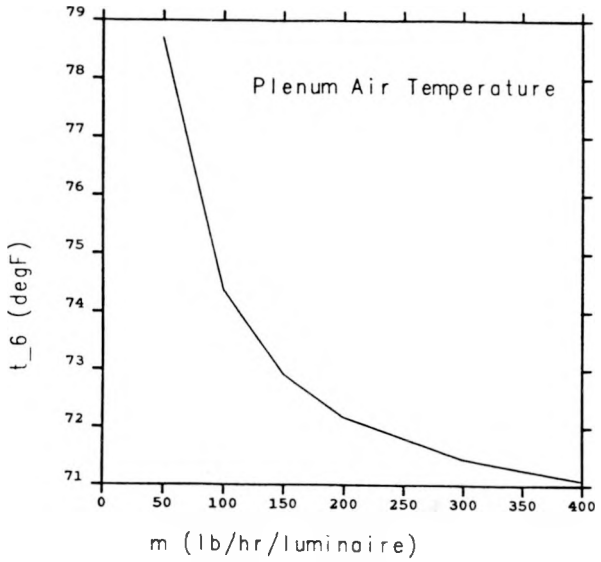
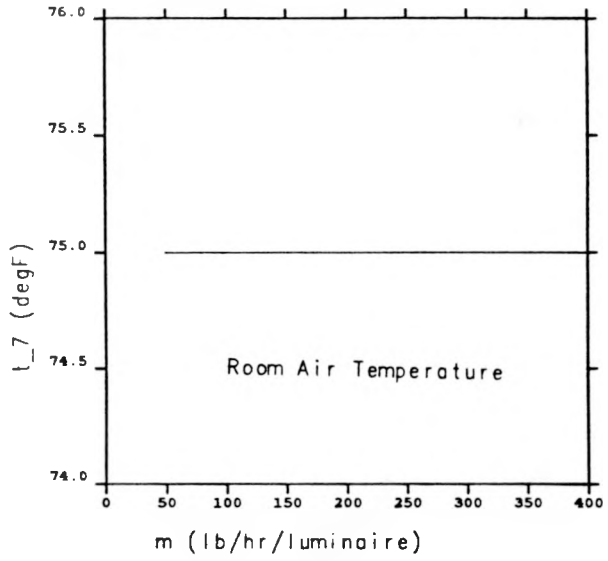
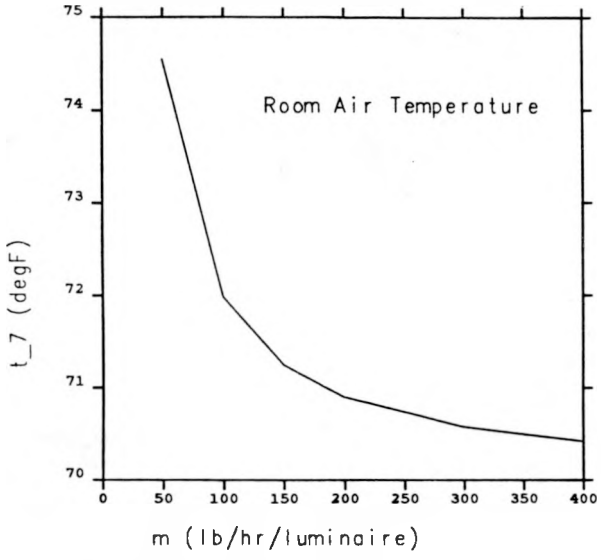


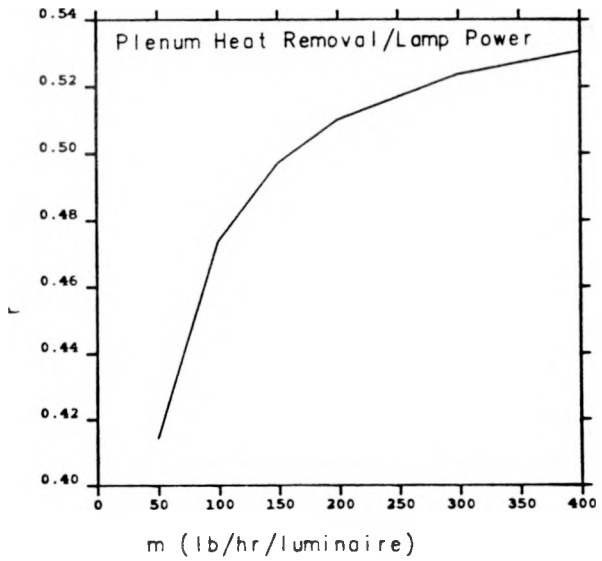

**Figure 7.** (cont.)



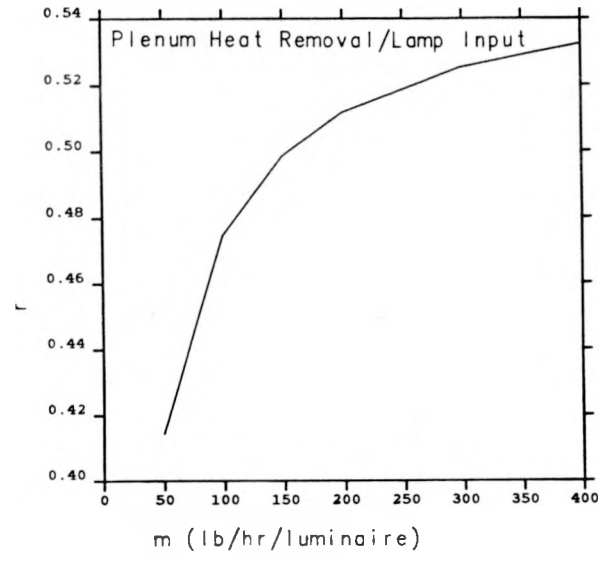




**Figure 7.** (cont.)

```
/*SPANK simulation file for lighting problem*/
/*SPANK file dyn_m_ed_light7_2.ps*/
/*Short wave are indicated by sw, long wave by lw*/
/*The problem is the following: we have a ceiling and a floor,
 *both insulated, with a plenum with lamps in it below the ceiling,
 *A translucent lens allows air flow between room and plenum*/
/*We have 5 nodes for radiative heat transfer:node 1 is ceiling,
 *2 is lamp,3 is top of lens, 4 is bottom of lens and 5 is floor*/
/*Node 6 is plenum air and node 7 is room air*/
/*Same as ed_light5, except that air nodes are put into the node
 *list*/
/*Units are Btu,hrs,lb,ft,kelvin (for tk's) and fahrenheit
 *(anywhere else)*/
/*Includes mass flow*/
/*Difference with m_ed_light7 is that it is DYNAMIC*/
/*Difference with dyn_m_ed_light7 is that the room
 *temperature is input and the
 *outside air temperature is calculated,
 *unlike in dyn_m_ed_light7 where it is the
 *contrary*/

declare radiosity_macro s,l;     /*Short wave and Long wave simulation*/
declare ax_macro        fjs,fjl;        /*FJ vectors, F shape factor, J radi
declare net_radiation_macro     fjqrs,fjqrl;    /*Net radiation in each band
declare dyn_energy_balance_macro        h;      /*Heat balance on each node*
declare blackbody       eb1,eb2,eb3,eb4,eb5,eb6,eb7;
declare degkdegf        kf1,kf2,kf3,kf4,kf5,kf6,kf7;
declare sum             qr1,qr2,qr3,qr4,qr5,qr6,qr7;
declare heat_add        q_6_7;
declare sum             q2_6_7;
declare mprod           mp;
declare msum6           uc1,uc2,uc3,uc4,uc5,uc6,uc7;

declare heat_add        hroom;
declare heat_add        wroom,wplenum;
declare hdbw            hr,hp,hs;

/*Time step*/
➤input   dt(h.dt)[TIME]

/*Heat capacitances*/
➤input   m_1(h.m_1)
➤input   m_2(h.m_2)
➤input   m_3(h.m_3)
➤input   m_4(h.m_4)
➤input   m_5(h.m_5)
➤input   m_6(h.m_6)


/*Mass flow and specific heat of air*/
input   m(mp.in1,hroom.m,wroom.m,wplenum.m,q_6_7.m)
input   cp(mp.in2)

/*Reflectances*/
input   rs_1(s.r_1)
input   rs_2(s.r_2)
```

**Figure 8.** Problem specification file for dynamic problem, case (1): fixed room air temperature. The network that describes the problem is formed by linking together macro objects that represent system component models and by assigning input quantities. Arrows indicate input lines that are different from the steady-state case, Fig. 6.

```
input   rs_3(s.r_3)
input   rs_4(s.r_4)
input   rs_5(s.r_5)
input   rs_6(s.r_6)
input   rs_7(s.r_7)

input   rl_1(l.r_1)
input   rl_2(l.r_2)
input   rl_3(l.r_3)
input   rl_4(l.r_4)
input   rl_5(l.r_5)
input   rl_6(l.r_6)
input   rl_7(l.r_7)

/*Short wave transmittances*/
input   taus_1(fjqrs.tau_1)
input   taus_2(fjqrs.tau_2)
input   taus_3(fjqrs.tau_3)
input   taus_4(fjqrs.tau_4)
input   taus_5(fjqrs.tau_5)
input   taus_6(fjqrs.tau_6)
input   taus_7(fjqrs.tau_7)

/*Long wave transmittances*/
input   taul_1(fjqrl.tau_1)
input   taul_2(fjqrl.tau_2)
input   taul_3(fjqrl.tau_3)
input   taul_4(fjqrl.tau_4)
input   taul_5(fjqrl.tau_5)
input   taul_6(fjqrl.tau_6)
input   taul_7(fjqrl.tau_7)

/*Areas*/
input   a_1(fjqrs.a_1,fjqrl.a_1)
input   a_2(fjqrs.a_2,fjqrl.a_2)
input   a_3(fjqrs.a_3,fjqrl.a_3)
input   a_4(fjqrs.a_4,fjqrl.a_4)
input   a_5(fjqrs.a_5,fjqrl.a_5)
input   a_6(fjqrs.a_6,fjqrl.a_6)
input   a_7(fjqrs.a_7,fjqrl.a_7)

/*Shape factors*/
input   f_1_1(fjs.f_1_1,fjl.f_1_1)
input   f_1_2(fjs.f_1_2,fjl.f_1_2)
input   f_1_3(fjs.f_1_3,fjl.f_1_3)
input   f_1_4(fjs.f_1_4,fjl.f_1_4)
input   f_1_5(fjs.f_1_5,fjl.f_1_5)
input   f_1_6(fjs.f_1_6,fjl.f_1_6)
input   f_1_7(fjs.f_1_7,fjl.f_1_7)

input   f_2_1(fjs.f_2_1,fjl.f_2_1)
input   f_2_2(fjs.f_2_2,fjl.f_2_2)
input   f_2_3(fjs.f_2_3,fjl.f_2_3)
input   f_2_4(fjs.f_2_4,fjl.f_2_4)
input   f_2_5(fjs.f_2_5,fjl.f_2_5)
input   f_2_6(fjs.f_2_6,fjl.f_2_6)
```

**Figure 8.** (cont.)

```
input    f_2_7(fjs.f_2_7,fjl.f_2_7)

input    f_3_1(fjs.f_3_1,fjl.f_3_1)
input    f_3_2(fjs.f_3_2,fjl.f_3_2)
input    f_3_3(fjs.f_3_3,fjl.f_3_3)
input    f_3_4(fjs.f_3_4,fjl.f_3_4)
input    f_3_5(fjs.f_3_5,fjl.f_3_5)
input    f_3_6(fjs.f_3_6,fjl.f_3_6)
input    f_3_7(fjs.f_3_7,fjl.f_3_7)

input    f_4_1(fjs.f_4_1,fjl.f_4_1)
input    f_4_2(fjs.f_4_2,fjl.f_4_2)
input    f_4_3(fjs.f_4_3,fjl.f_4_3)
input    f_4_4(fjs.f_4_4,fjl.f_4_4)
input    f_4_5(fjs.f_4_5,fjl.f_4_5)
input    f_4_6(fjs.f_4_6,fjl.f_4_6)
input    f_4_7(fjs.f_4_7,fjl.f_4_7)

input    f_5_1(fjs.f_5_1,fjl.f_5_1)
input    f_5_2(fjs.f_5_2,fjl.f_5_2)
input    f_5_3(fjs.f_5_3,fjl.f_5_3)
input    f_5_4(fjs.f_5_4,fjl.f_5_4)
input    f_5_5(fjs.f_5_5,fjl.f_5_5)
input    f_5_6(fjs.f_5_6,fjl.f_5_6)
input    f_5_7(fjs.f_5_7,fjl.f_5_7)

input    f_6_1(fjs.f_6_1,fjl.f_6_1)
input    f_6_2(fjs.f_6_2,fjl.f_6_2)
input    f_6_3(fjs.f_6_3,fjl.f_6_3)
input    f_6_4(fjs.f_6_4,fjl.f_6_4)
input    f_6_5(fjs.f_6_5,fjl.f_6_5)
input    f_6_6(fjs.f_6_6,fjl.f_6_6)
input    f_6_7(fjs.f_6_7,fjl.f_6_7)

input    f_7_1(fjs.f_7_1,fjl.f_7_1)
input    f_7_2(fjs.f_7_2,fjl.f_7_2)
input    f_7_3(fjs.f_7_3,fjl.f_7_3)
input    f_7_4(fjs.f_7_4,fjl.f_7_4)
input    f_7_5(fjs.f_7_5,fjl.f_7_5)
input    f_7_6(fjs.f_7_6,fjl.f_7_6)
input    f_7_7(fjs.f_7_7,fjl.f_7_7)

/*Generalized conductance convectance matrix*/
link     uc_1_1(h.uc_1_1,uc1.msum)
input    uc_1_2(h.uc_1_2,uc1.x1)
input    uc_1_3(h.uc_1_3,uc1.x2)
input    uc_1_4(h.uc_1_4,uc1.x3)
input    uc_1_5(h.uc_1_5,uc1.x4)
input    uc_1_6(h.uc_1_6,uc1.x5)
input    uc_1_7(h.uc_1_7,uc1.x6)

input    uc_2_1(h.uc_2_1,uc2.x1)
link     uc_2_2(h.uc_2_2,uc2.msum)
input    uc_2_3(h.uc_2_3,uc2.x2)
input    uc_2_4(h.uc_2_4,uc2.x3)
input    uc_2_5(h.uc_2_5,uc2.x4)
```

**Figure 8.** (cont.)

```
input   uc_2_6(h.uc_2_6,uc2.x5)
input   uc_2_7(h.uc_2_7,uc2.x6)

input   uc_3_1(h.uc_3_1,uc3.x1)
input   uc_3_2(h.uc_3_2,uc3.x2)
link    uc_3_3(h.uc_3_3,uc3.msum)
input   uc_3_4(h.uc_3_4,uc3.x3)
input   uc_3_5(h.uc_3_5,uc3.x4)
input   uc_3_6(h.uc_3_6,uc3.x5)
input   uc_3_7(h.uc_3_7,uc3.x6)

input   uc_4_1(h.uc_4_1,uc4.x1)
input   uc_4_2(h.uc_4_2,uc4.x2)
input   uc_4_3(h.uc_4_3,uc4.x3)
link    uc_4_4(h.uc_4_4,uc4.msum)
input   uc_4_5(h.uc_4_5,uc4.x4)
input   uc_4_6(h.uc_4_6,uc4.x5)
input   uc_4_7(h.uc_4_7,uc4.x6)

input   uc_5_1(h.uc_5_1,uc5.x1)
input   uc_5_2(h.uc_5_2,uc5.x2)
input   uc_5_3(h.uc_5_3,uc5.x3)
input   uc_5_4(h.uc_5_4,uc5.x4)
link    uc_5_5(h.uc_5_5,uc5.msum)
input   uc_5_6(h.uc_5_6,uc5.x5)
input   uc_5_7(h.uc_5_7,uc5.x6)

input   uc_6_1(h.uc_6_1,uc6.x1)
input   uc_6_2(h.uc_6_2,uc6.x2)
input   uc_6_3(h.uc_6_3,uc6.x3)
input   uc_6_4(h.uc_6_4,uc6.x4)
input   uc_6_5(h.uc_6_5,uc6.x5)
link    uc_6_6(h.uc_6_6,uc6.msum)
link    uc_6_7(h.uc_6_7,mp.mproduct,uc6.x6)

input   uc_7_1(h.uc_7_1,uc7.x1)
input   uc_7_2(h.uc_7_2,uc7.x2)
input   uc_7_3(h.uc_7_3,uc7.x3)
input   uc_7_4(h.uc_7_4,uc7.x4)
input   uc_7_5(h.uc_7_5,uc7.x5)
input   uc_7_6(h.uc_7_6,uc7.x6)
link    uc_7_7(h.uc_7_7,uc7.msum)

/*Special short wave transmittance matrix*/
input   tauss_1_1(s.taus_1_1,fjqrs.taus_1_1)
input   tauss_1_2(s.taus_1_2,fjqrs.taus_1_2)
input   tauss_1_3(s.taus_1_3,fjqrs.taus_1_3)
input   tauss_1_4(s.taus_1_4,fjqrs.taus_1_4)
input   tauss_1_5(s.taus_1_5,fjqrs.taus_1_5)
input   tauss_1_6(s.taus_1_6,fjqrs.taus_1_6)
input   tauss_1_7(s.taus_1_7,fjqrs.taus_1_7)

input   tauss_2_1(s.taus_2_1,fjqrs.taus_2_1)
input   tauss_2_2(s.taus_2_2,fjqrs.taus_2_2)
input   tauss_2_3(s.taus_2_3,fjqrs.taus_2_3)
input   tauss_2_4(s.taus_2_4,fjqrs.taus_2_4)
```

**Figure 8.** (cont.)

```
input   tauss_2_5(s.taus_2_5,fjqrs.taus_2_5)
input   tauss_2_6(s.taus_2_6,fjqrs.taus_2_6)
input   tauss_2_7(s.taus_2_7,fjqrs.taus_2_7)

input   tauss_3_1(s.taus_3_1,fjqrs.taus_3_1)
input   tauss_3_2(s.taus_3_2,fjqrs.taus_3_2)
input   tauss_3_3(s.taus_3_3,fjqrs.taus_3_3)
input   tauss_3_4(s.taus_3_4,fjqrs.taus_3_4)
input   tauss_3_5(s.taus_3_5,fjqrs.taus_3_5)
input   tauss_3_6(s.taus_3_6,fjqrs.taus_3_6)
input   tauss_3_7(s.taus_3_7,fjqrs.taus_3_7)

input   tauss_4_1(s.taus_4_1,fjqrs.taus_4_1)
input   tauss_4_2(s.taus_4_2,fjqrs.taus_4_2)
input   tauss_4_3(s.taus_4_3,fjqrs.taus_4_3)
input   tauss_4_4(s.taus_4_4,fjqrs.taus_4_4)
input   tauss_4_5(s.taus_4_5,fjqrs.taus_4_5)
input   tauss_4_6(s.taus_4_6,fjqrs.taus_4_6)
input   tauss_4_7(s.taus_4_7,fjqrs.taus_4_7)

input   tauss_5_1(s.taus_5_1,fjqrs.taus_5_1)
input   tauss_5_2(s.taus_5_2,fjqrs.taus_5_2)
input   tauss_5_3(s.taus_5_3,fjqrs.taus_5_3)
input   tauss_5_4(s.taus_5_4,fjqrs.taus_5_4)
input   tauss_5_5(s.taus_5_5,fjqrs.taus_5_5)
input   tauss_5_6(s.taus_5_6,fjqrs.taus_5_6)
input   tauss_5_7(s.taus_5_7,fjqrs.taus_5_7)

input   tauss_6_1(s.taus_6_1,fjqrs.taus_6_1)
input   tauss_6_2(s.taus_6_2,fjqrs.taus_6_2)
input   tauss_6_3(s.taus_6_3,fjqrs.taus_6_3)
input   tauss_6_4(s.taus_6_4,fjqrs.taus_6_4)
input   tauss_6_5(s.taus_6_5,fjqrs.taus_6_5)
input   tauss_6_6(s.taus_6_6,fjqrs.taus_6_6)
input   tauss_6_7(s.taus_6_7,fjqrs.taus_6_7)

input   tauss_7_1(s.taus_7_1,fjqrs.taus_7_1)
input   tauss_7_2(s.taus_7_2,fjqrs.taus_7_2)
input   tauss_7_3(s.taus_7_3,fjqrs.taus_7_3)
input   tauss_7_4(s.taus_7_4,fjqrs.taus_7_4)
input   tauss_7_5(s.taus_7_5,fjqrs.taus_7_5)
input   tauss_7_6(s.taus_7_6,fjqrs.taus_7_6)
input   tauss_7_7(s.taus_7_7,fjqrs.taus_7_7)


/*Special long wave transmittance matrix*/
input   tausl_1_1(l.taus_1_1,fjqrl.taus_1_1)
input   tausl_1_2(l.taus_1_2,fjqrl.taus_1_2)
input   tausl_1_3(l.taus_1_3,fjqrl.taus_1_3)
input   tausl_1_4(l.taus_1_4,fjqrl.taus_1_4)
input   tausl_1_5(l.taus_1_5,fjqrl.taus_1_5)
input   tausl_1_6(l.taus_1_6,fjqrl.taus_1_6)
input   tausl_1_7(l.taus_1_7,fjqrl.taus_1_7)

input   tausl_2_1(l.taus_2_1,fjqrl.taus_2_1)
input   tausl_2_2(l.taus_2_2,fjqrl.taus_2_2)
```

**Figure 8.** (cont.)

```
input   tausl_2_3(l.taus_2_3,fjqrl.taus_2_3)
input   tausl_2_4(l.taus_2_4,fjqrl.taus_2_4)
input   tausl_2_5(l.taus_2_5,fjqrl.taus_2_5)
input   tausl_2_6(l.taus_2_6,fjqrl.taus_2_6)
input   tausl_2_7(l.taus_2_7,fjqrl.taus_2_7)

input   tausl_3_1(l.taus_3_1,fjqrl.taus_3_1)
input   tausl_3_2(l.taus_3_2,fjqrl.taus_3_2)
input   tausl_3_3(l.taus_3_3,fjqrl.taus_3_3)
input   tausl_3_4(l.taus_3_4,fjqrl.taus_3_4)
input   tausl_3_5(l.taus_3_5,fjqrl.taus_3_5)
input   tausl_3_6(l.taus_3_6,fjqrl.taus_3_6)
input   tausl_3_7(l.taus_3_7,fjqrl.taus_3_7)

input   tausl_4_1(l.taus_4_1,fjqrl.taus_4_1)
input   tausl_4_2(l.taus_4_2,fjqrl.taus_4_2)
input   tausl_4_3(l.taus_4_3,fjqrl.taus_4_3)
input   tausl_4_4(l.taus_4_4,fjqrl.taus_4_4)
input   tausl_4_5(l.taus_4_5,fjqrl.taus_4_5)
input   tausl_4_6(l.taus_4_6,fjqrl.taus_4_6)
input   tausl_4_7(l.taus_4_7,fjqrl.taus_4_7)

input   tausl_5_1(l.taus_5_1,fjqrl.taus_5_1)
input   tausl_5_2(l.taus_5_2,fjqrl.taus_5_2)
input   tausl_5_3(l.taus_5_3,fjqrl.taus_5_3)
input   tausl_5_4(l.taus_5_4,fjqrl.taus_5_4)
input   tausl_5_5(l.taus_5_5,fjqrl.taus_5_5)
input   tausl_5_6(l.taus_5_6,fjqrl.taus_5_6)
input   tausl_5_7(l.taus_5_7,fjqrl.taus_5_7)

input   tausl_6_1(l.taus_6_1,fjqrl.taus_6_1)
input   tausl_6_2(l.taus_6_2,fjqrl.taus_6_2)
input   tausl_6_3(l.taus_6_3,fjqrl.taus_6_3)
input   tausl_6_4(l.taus_6_4,fjqrl.taus_6_4)
input   tausl_6_5(l.taus_6_5,fjqrl.taus_6_5)
input   tausl_6_6(l.taus_6_6,fjqrl.taus_6_6)
input   tausl_6_7(l.taus_6_7,fjqrl.taus_6_7)

input   tausl_7_1(l.taus_7_1,fjqrl.taus_7_1)
input   tausl_7_2(l.taus_7_2,fjqrl.taus_7_2)
input   tausl_7_3(l.taus_7_3,fjqrl.taus_7_3)
input   tausl_7_4(l.taus_7_4,fjqrl.taus_7_4)
input   tausl_7_5(l.taus_7_5,fjqrl.taus_7_5)
input   tausl_7_6(l.taus_7_6,fjqrl.taus_7_6)
input   tausl_7_7(l.taus_7_7,fjqrl.taus_7_7)


/*Short wave irradiation*/
link    fjs_1(s.fj_1,fjqrs.fj_1,fjs.fj_1) [fjs1]
link    fjs_2(s.fj_2,fjqrs.fj_2,fjs.fj_2) [fjs2]
link    fjs_3(s.fj_3,fjqrs.fj_3,fjs.fj_3) [fjs3]
link    fjs_4(s.fj_4,fjqrs.fj_4,fjs.fj_4) [fjs4]
link    fjs_5(s.fj_5,fjqrs.fj_5,fjs.fj_5) [fjs5]
link    fjs_6(s.fj_6,fjqrs.fj_6,fjs.fj_6) [fjs6]
link    fjs_7(s.fj_7,fjqrs.fj_7,fjs.fj_7) [fjs7]
```

**Figure 8.** (cont.)

```
  /*Long wave irradiation*/
  link     fjl_1(l.fj_1,fjqrl.fj_1,fjl.fj_1) [fjl1]
  link     fjl_2(l.fj_2,fjqrl.fj_2,fjl.fj_2) [fjl2]
  link     fjl_3(l.fj_3,fjqrl.fj_3,fjl.fj_3) [fjl3]
  link     fjl_4(l.fj_4,fjqrl.fj_4,fjl.fj_4) [fjl4]
  link     fjl_5(l.fj_5,fjqrl.fj_5,fjl.fj_5) [fjl5]
  link     fjl_6(l.fj_6,fjqrl.fj_6,fjl.fj_6) [fjl6]
  link     fjl_7(l.fj_7,fjqrl.fj_7,fjl.fj_7) [fjl7]


  /*Temperatures in fahrenheits*/
  link     t_1(kf1.f,h.t_1)
  link     t_2(kf2.f,h.t_2)
  link     t_3(kf3.f,h.t_3)
  link     t_4(kf4.f,h.t_4)
  link     t_5(kf5.f,h.t_5)
  link     t_6(kf6.f,h.t_6,hp.db) [T]
  input    t_7(kf7.f,h.t_7,hr.db) [T]

  /*Temperature derivatives*/
➤ link     tdot_1(h.tdot_1)
➤ link     tdot_2(h.tdot_2)
➤ link     tdot_3(h.tdot_3)
➤ link     tdot_4(h.tdot_4)
➤ link     tdot_5(h.tdot_5)
➤ link     tdot_6(h.tdot_6)

  /*Temperature history*/
➤ history t_1_h(h.t_1_hist)
➤ history t_2_h(h.t_2_hist)
➤ history t_3_h(h.t_3_hist)
➤ history t_4_h(h.t_4_hist)
➤ history t_5_h(h.t_5_hist)
➤ history t_6_h(h.t_6_hist)

  /*Temperatures in kelvins*/
  link     tk_1(kf1.k,eb1.t)
  link     tk_2(kf2.k,eb2.t)
  link     tk_3(kf3.k,eb3.t)
  link     tk_4(kf4.k,eb4.t)
  link     tk_5(kf5.k,eb5.t)
  link     tk_6(kf6.k,eb6.t)
  link     tk_7(kf7.k,eb7.t)

  /*Net short wave radiant heat transfer*/
  link     qr_s_1(fjqrs.qr_1,qr1.in1,h.qr_s_1)
  link     qr_s_2(fjqrs.qr_2,qr2.in1,h.qr_s_2)
  link     qr_s_3(fjqrs.qr_3,qr3.in1,h.qr_s_3)
  link     qr_s_4(fjqrs.qr_4,qr4.in1,h.qr_s_4)
  link     qr_s_5(fjqrs.qr_5,qr5.in1,h.qr_s_5)
  link     qr_s_6(fjqrs.qr_6,qr6.in1,h.qr_s_6)
  link     qr_s_7(fjqrs.qr_7,qr7.in1,h.qr_s_7)

  /*Net long wave radiant heat transfer*/
  link     qr_l_1(fjqrl.qr_1,qr1.in2,h.qr_l_1)
  link     qr_l_2(fjqrl.qr_2,qr2.in2,h.qr_l_2)
```

**Figure 8.** (cont.)

```
link    qr_1_3(fjqr1.qr_3,qr3.in2,h.qr_1_3)
link    qr_1_4(fjqr1.qr_4,qr4.in2,h.qr_1_4)
link    qr_1_5(fjqr1.qr_5,qr5.in2,h.qr_1_5)
link    qr_1_6(fjqr1.qr_6,qr6.in2,h.qr_1_6)
link    qr_1_7(fjqr1.qr_7,qr7.in2,h.qr_1_7)

/*Source power input = 0 except on lamp*/
input   q0_1(h.q0_1)
input   q0_2(h.q0_2)
input   q0_3(h.q0_3)
input   q0_4(h.q0_4)
input   q0_5(h.q0_5)
input   q0_6(h.q0_6,q2_6_7.in1)[q06]
link    q0_7(h.q0_7,hroom.q,q2_6_7.in2)[q07]

/*Load*/
link    load(q_6_7.q)
link    load2(q2_6_7.sum)

/*Total net radiant heat transfer on each node*/
link    qr_1(qr1.sum)
link    qr_2(qr2.sum)
link    qr_3(qr3.sum)
link    qr_4(qr4.sum)
link    qr_5(qr5.sum)
link    qr_6(qr6.sum)
link    qr_7(qr7.sum)

/*Short wave radiosity source vector = 0 except for lamp (node 2)*/
input   j0s_1(s.j0_1)
input   j0s_2(s.j0_2)
input   j0s_3(s.j0_3)
input   j0s_4(s.j0_4)
input   j0s_5(s.j0_5)
input   j0s_6(s.j0_6)
input   j0s_7(s.j0_7)

/*Long wave radiosity source vector = blackbody emission*/
link    j0l_1(l.j0_1,eb1.eb)
link    j0l_2(l.j0_2,eb2.eb)
link    j0l_3(l.j0_3,eb3.eb)
link    j0l_4(l.j0_4,eb4.eb)
link    j0l_5(l.j0_5,eb5.eb)
link    j0l_6(l.j0_6,eb6.eb)[eb]
link    j0l_7(l.j0_7,eb7.eb)[eb]

/*Stefan Boltzmann constant*/
input   sigma(eb1.sigma,eb2.sigma,eb3.sigma,eb4.sigma,eb5.sigma,eb6.sigma,eb

/*Short wave radiosities*/
link    js_1(s.j_1,fjs.j_1,fjqrs.j_1)
link    js_2(s.j_2,fjs.j_2,fjqrs.j_2)
link    js_3(s.j_3,fjs.j_3,fjqrs.j_3)
link    js_4(s.j_4,fjs.j_4,fjqrs.j_4)
link    js_5(s.j_5,fjs.j_5,fjqrs.j_5)
link    js_6(s.j_6,fjs.j_6,fjqrs.j_6)
```

**Figure 8.** (cont.)

```
link    js_7(s.j_7,fjs.j_7,fjqrs.j_7)

/*Long wave radiosities*/
link    jl_1(l.j_1,fjl.j_1,fjqrl.j_1)[jl1]
link    jl_2(l.j_2,fjl.j_2,fjqrl.j_2)[jl2]
link    jl_3(l.j_3,fjl.j_3,fjqrl.j_3)[jl3]
link    jl_4(l.j_4,fjl.j_4,fjqrl.j_4)[jl4]
link    jl_5(l.j_5,fjl.j_5,fjqrl.j_5)[jl5]
link    jl_6(l.j_6,fjl.j_6,fjqrl.j_6)[jl6]
link    jl_7(l.j_7,fjl.j_7,fjqrl.j_7)[jl7]

/*Enthalpies. The strange "outs" and "ins" are due to the fact that
 *q0_7 is source power input to the nodes, and thus is calculated
 *the opposite way as is usually done in heat_add object.*/
/*Thus q0_7=m(h_0-h_7)*/
link    h_0(hs.h,hroom.h_out,q_6_7.h_in)
link    h_6(hp.h,q_6_7.h_out)
link    h_7(hr.h,hroom.h_in)

/*Supply temperature*/
link t_0(hs.db)

/*Humidities*/
input   w_0(hs.w,wroom.h_in)
link    w_6(hp.w,wplenum.h_out)
link    w_7(hr.w,wroom.h_out,wplenum.h_in)

/*Humidity source*/
input   wroom_added(wroom.q)
input   wplenum_added(wplenum.q)
```

**Figure 8.** (cont.)

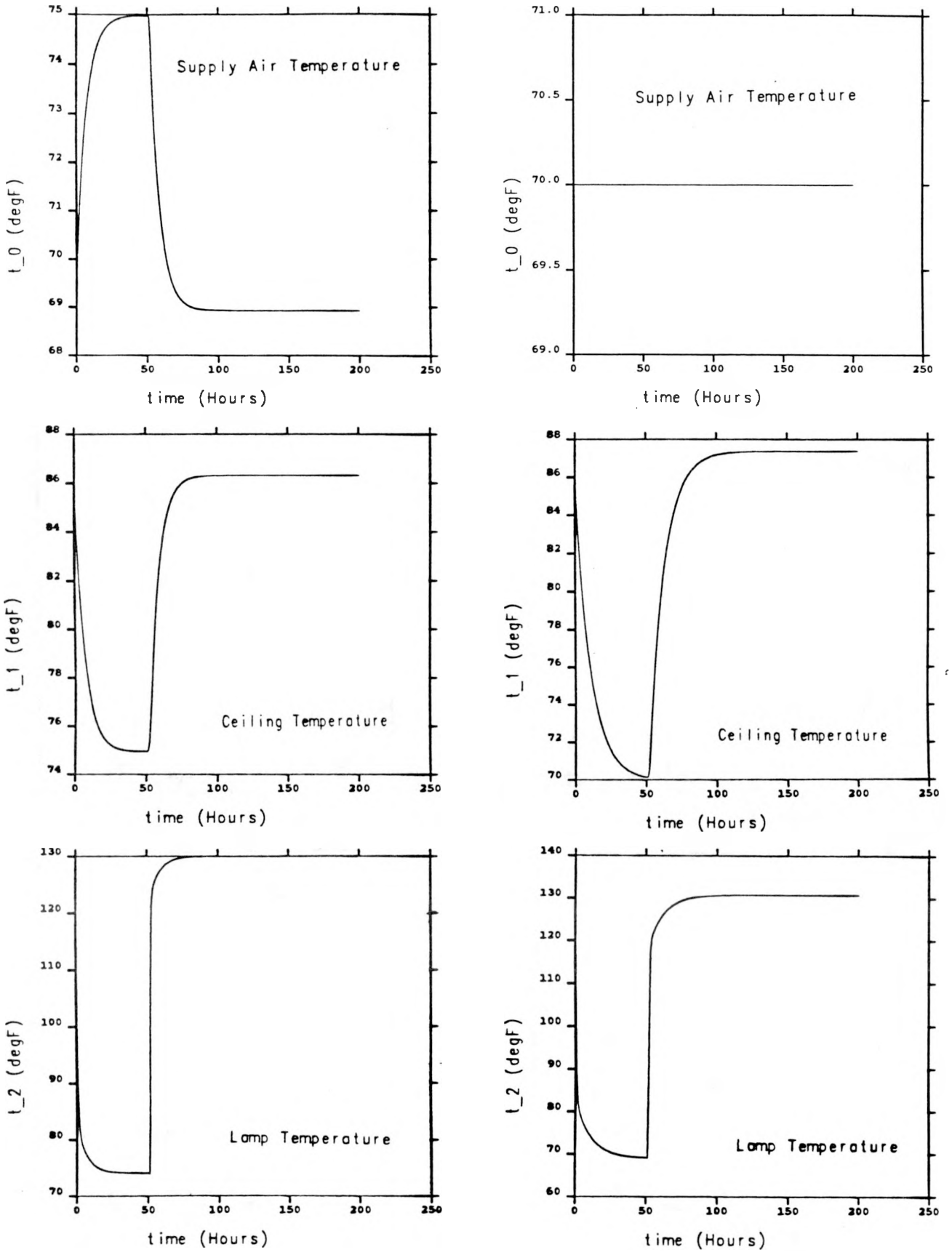



**Figure 9.** SPANK dynamic simulation results for case (1), room air temperature fixed at 75F (**left-hand graphs**), and case (2), supply air temperature fixed at 70F (**right-hand graphs**), as a function of air flow rate.

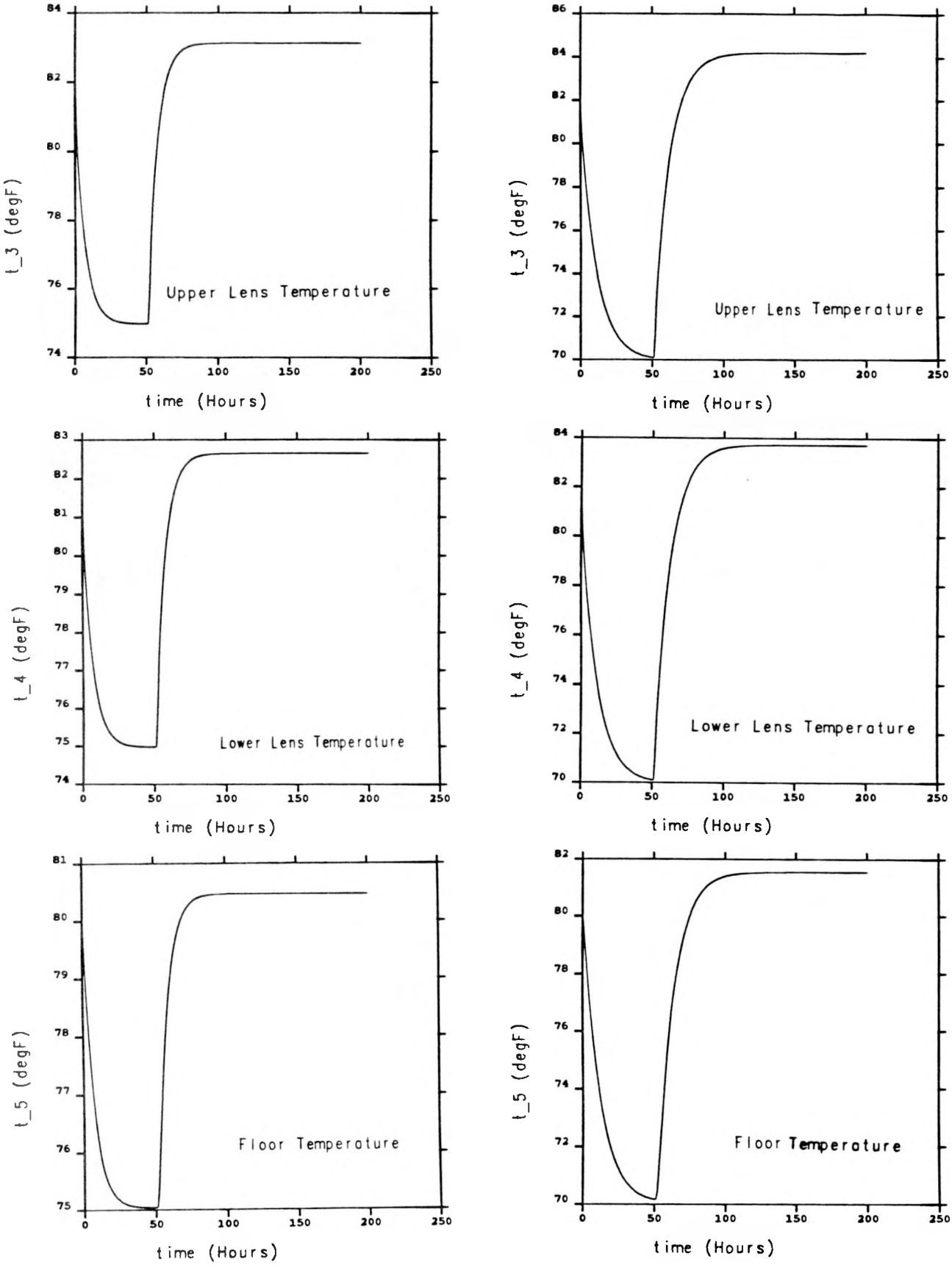



**Figure 9.** (cont.)

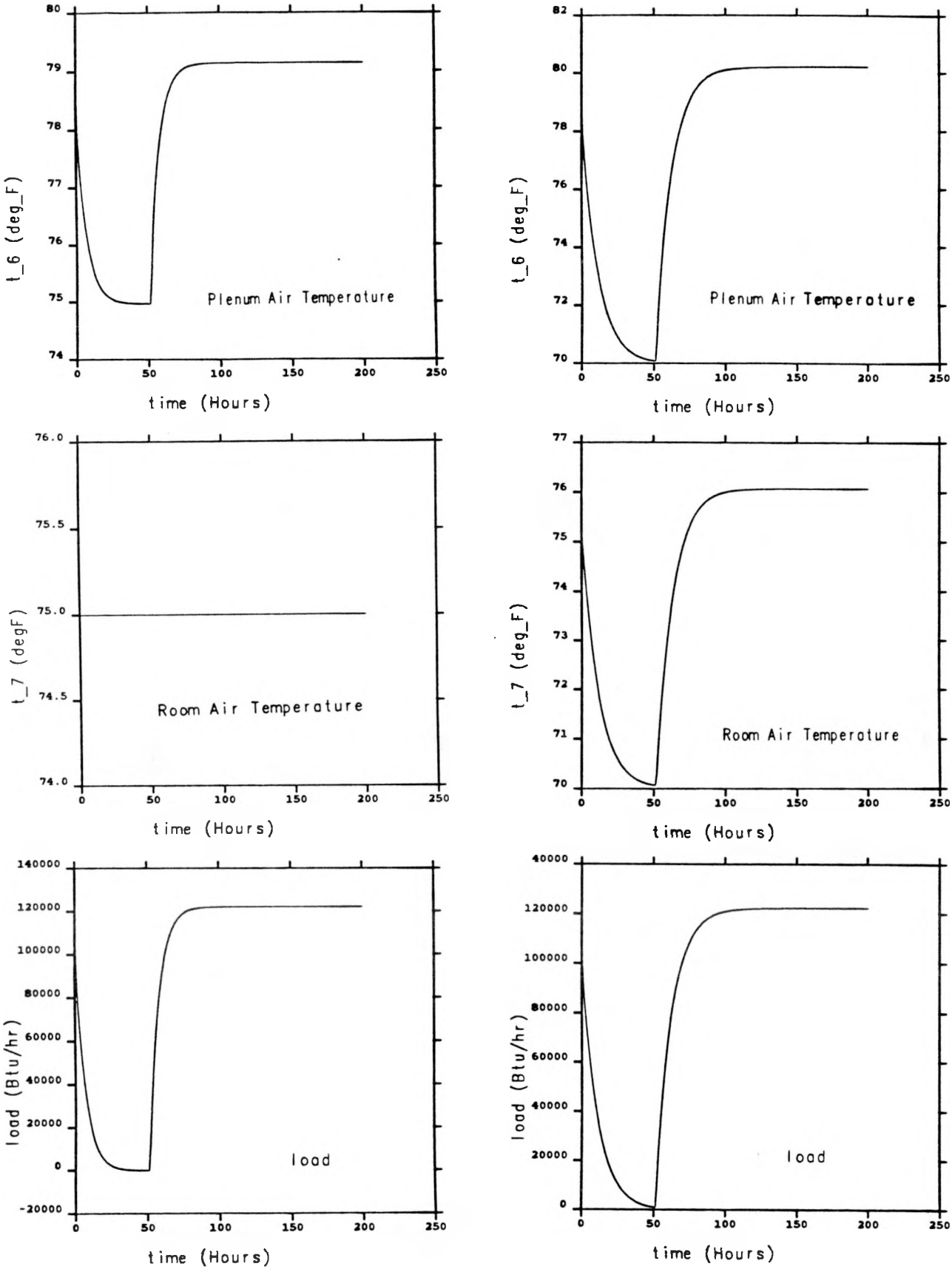



**Figure 9.** (cont.)

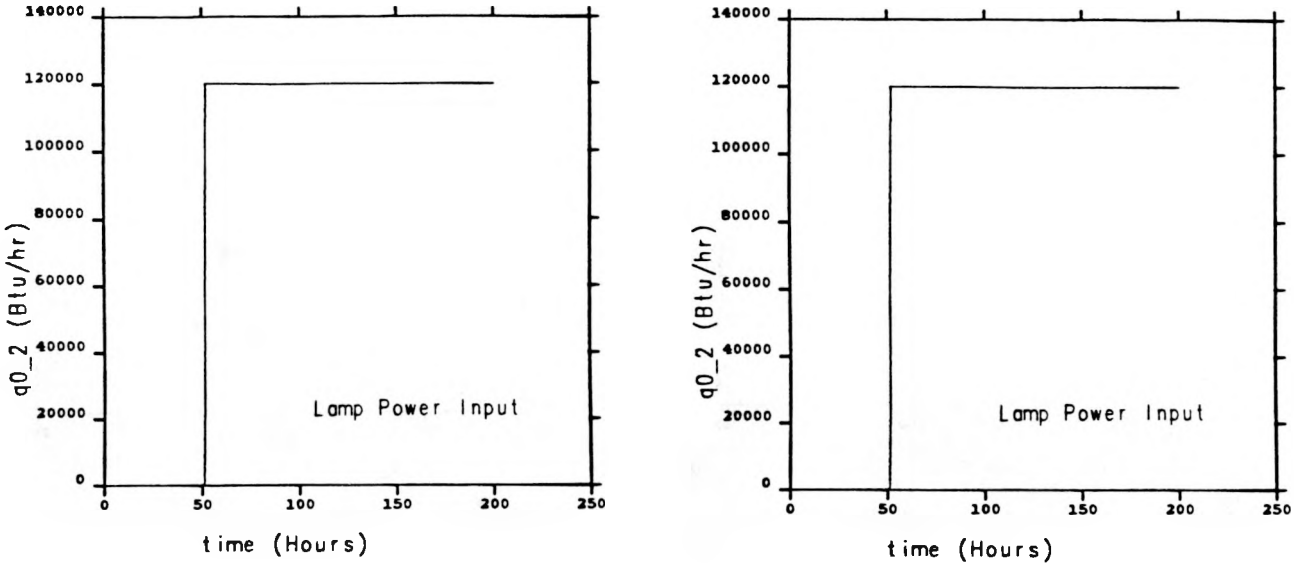



**Figure 9.** (cont.)