THE MODELLING OF THE THERMAL SUBSYSTEM IN SPACECRAFT REAL–TIME SIMULATORS*

Miguel Á. Carreira Perpiñán
Simulation Systems Section esa-esoc

Abstract

Two different approaches to thermal simulation of spacecraft exist in widespread use: one operates by carrying out some kind of interpolation over a finite set of selected, typical scenarios for which the thermal behaviour is known; networks of heaters and thermistors are often used in this approach. The other technique is essentially an integrator: first, the thermal system is nodalised, giving as a result a discrete network of nodes and conductors; second, the heat transfer equations are applied to this network, thus yielding a system of differential equations that can be solved using a numerical integrator (finite difference methods are very common).

To present, the thermal subsystem of the real–time satellite simulators developed at the Simulation Systems Section at ESOC, Germany, has always been an implementation of an *interpolation* method; the main reasons for this policy are the speed of computation and the somewhat loose requirements for accuracy. Examples of it are the Italsat, ISO, ERS–1, Eureca and Pastel satellite simulators. However, the computing power available has increased in such a way that *integration* methods are no longer prohibitive in real–time simulation.

In this paper, both simulation schemes are presented and criticised in terms of efficiency, accuracy, flexibility and ease of model construction. The possibility of using integrators for real–time simulation of simplified spacecraft models is discussed, putting particular emphasis on ESATAN, the standard thermal analysis code employed at the European Space Agency.

Keywords: thermal modelling, real–time simulation, spacecraft simulation.

1 Introduction

The primary functions of the real–time satellite simulators (RTS) developed at the Simulation Systems Section at ESOC (SIM) are the testing and validation of the Operations Control Centre facilities and flight control procedures and the training of the ground–network operators.

The mission of the thermal subsystem of an RTS is to provide thermal telemetry (TTM) in due time, i.e. accurate predictions of the temperature in selected points of the satellite with a certain frequency. This means that other types of calculations, like heat flows or heat imbalances, are not necessary for the thermal subsystem. Checking that the temperature of the equipment stays between certain margins (so that certain heaters turn on or off when the limit is surpassed) and rendering more or less correct temperature tendencies have traditionally been the foremost requirements.

There exist several approaches for the modelling of a thermal system. The most important for us, the methods based on integration with finite volumes and the methods based on steady–state interpolation, will be dealt with in the next sections. Another method that enjoys a considerable popularity is the finite–element method. A point in favour of it is that it is readily applicable to other problems that are governed by a set of differential equations, such as dynamic or structural modelling, and so a unique program (e.g. NASTRAN) can be used for all the different models. But this method requires quite fine

*This work has been carried out under the sponsorship of the Spanish *Centro para el Desarrollo Tecnológico e Industrial (CDTI)*.

spatial meshes (more than those of the finite–difference methods) and the resulting matrices are extremely big, what prevents one from using it for real–time thermal simulation at the moment.

Heat transfer in satellites takes place mainly through conduction and radiation, except for some specific devices in which convection is important (such as heat pipes). Convection is usually avoided in thermal models because it calls for very complex systems of equations.

The satellite payload is always considered to have constant temperature and to be uncoupled with the rest of the system, so one does not have to worry about it in the thermal subsystem of the RTS.

2 Current techniques for thermal simulation

The chief goal that one seeks when simulating a thermal system is to determine the evolution with time of its temperature distribution. Currently, two widespread approaches exist for that purpose; they are outlined in this section, as well as their pros and cons.

2.1 Integrators

Methods based in integrators consist of the following steps:

1. Discretise the system in a network of:
   - *Nodes*: isothermal volumes where heat can be stored; they are characterised by their thermal capacitance and optionally by a heat source.
   - *Links*: a link is a path between two nodes that allows heat to flow from one to the other. It can be conductive (and it is characterised by its thermal conductance) or radiative (characterised by its radiative exchange factor).[1]

   The network is called a *lumped parameter network* (LPN) because the continuous parameters of the thermal system have been "lumped" into the discrete set of the nodes and links. This phase is also called *nodalisation*.

2. Apply the heat conduction (Fourier's law) and radiation (Stefan–Boltzmann law) equations to this network. This yields a nonlinear system of differential equations, one for each node, of the form:

$$C_i \frac{dT_i}{dt} = \sum_{i \neq j} K_{ij}(T_j - T_i) + \sum_{i \neq j} R_{ij}(T_j^4 - T_i^4) + Q_i, \quad i, j = 1, \ldots, n \tag{1}$$

   where n is the number of nodes in the network, T_i, C_i and Q_i the temperature (°C or K), capacitance (J/°C) and heat source (W) of node i respectively, K_{ij} the conductance (W/°C) of the conductive link between nodes i and j, and R_{ij} the radiative exchange factor (W/°C^4) between nodes i and j. The values of C_i, K_{ij} and R_{ij} depend on the physical properties and the geometry of the system and can be variable with the time.

3. Integrate system (1) in order to provide values for the temperature vector $\mathbf{T}(t) = (T_1(t), \ldots, T_n(t))^T$ at different times. Several methods exist for that purpose, usually based in finite–difference schemes; two very common are the explicit forward differences method and the Crank–Nicolson method.

   To obtain the steady–state, one sets the left–hand side of (1) to zero and solves for T_i.

2.1.1 Advantages

- High precision attainable (for small enough timesteps).
- They are also valid for non–nominal situations (e.g. failures).
- Any interpolation method can be simulated by an LPN with a clever election of the coefficients (see appendix A).

[1] Convection can also be considered, but it is not necessary for our purposes.

2.1.2 Disadvantages

- Time–consuming, specially for fine meshes (big n) or small timesteps.
- The development of a thermal model is a complex task that requires expertise and time.

2.1.3 Actual implementations

This approach is employed in thermal analysis tools, such as ESATAN, THERMICA, IDEAS–TM2 and others. As an example of the way they work, let us consider the sequence of actions required to obtain a simulation with ESATAN, the European Space Agency Thermal ANalyser (see refs. [1, 2, 3]):

- The input is an ASCII file called *input deck* that describes the lumped network (nodes, links and all coefficients), the simulation control parameters (simulation interval, solution method, desired accuracy, etc.) and other relevant information (such as the output format).
- ESATAN preprocesses this input deck, creating two files: a model database containing the network description and a Fortran program containing the simulation script.
- The previous Fortran program is then compiled and linked with the ESATAN object libraries (which contain the solution routines, among other things).
- The resulting executable file produces, when ran, the specified output.

The computation of the radiative couplings R_{ij} of system (1), as well as the external fluxes that affect some of the Q_i, is quite complicated. For this reason, there is usually a separate program that accomplishes this task; for instance, the companion radiative analyser of ESATAN is ESARAD. ESARAD (see refs. [4, 5, 6]) allows the user to describe the geometrical configuration of the satellite (in terms of elementary shapes, like boxes or cylinders), its thermooptical properties (e.g. emissivity) and its orbit, and outputs an ESATAN input deck with the numerical values of R_{ij} and Q_i.

2.2 Methods based in interpolation

The construction of a model using an interpolation method involves, from a very general point of view, the following steps:

1. The thermal engineers determine those parameters P_j, $j = 1, \ldots, p$, that are relevant to the thermal behaviour of the spacecraft (such as orbital coordinates, attitude angles, ON/OFF state of heaters, etc.). At any given instant, the **spacecraft (S/C) thermal context** is defined as the vector $(P_1(t), \ldots, P_p(t))$.
2. Now the range of variation of each parameter is discretised; the Cartesian product of all discretised ranges builds a p-dimensional mesh $\{\tilde{P}_{1,min}, \ldots, \tilde{P}_{1,max}\} \times \cdots \times \{\tilde{P}_{p,min}, \ldots, \tilde{P}_{p,max}\}$ of discrete contexts $\tilde{\mathbf{C}} = (\tilde{P}_1, \ldots, \tilde{P}_p)$. Notice that the tilde over a variable means that it is discrete.
3. The **S/C thermal map** is defined as a vector of m selected temperature measurements inside the S/C: $\mathbf{T} = (T_1, \ldots, T_m)^T$.
4. *Thermal preprocessing*: by means of a thermal analysis tool (such as ESATAN), one computes the steady–state S/C thermal map for each possible discrete context and stores it into a table. In other words, this table defines the mapping:

$$\tilde{\mathbf{T}}_\infty : \{\tilde{P}_{1,min}, \ldots, \tilde{P}_{1,max}\} \times \cdots \times \{\tilde{P}_{p,min}, \ldots, \tilde{P}_{p,max}\} \longrightarrow \mathbb{R}^m$$

$$\tilde{\mathbf{C}} = (\tilde{P}_1, \ldots, \tilde{P}_p) \longmapsto \tilde{\mathbf{T}}_\infty(\tilde{\mathbf{C}}) = (\tilde{T}_{1,\infty}, \ldots, \tilde{T}_{m,\infty})^T$$

5. The steady–state thermal map of a context $\mathbf{C}$ not in the p-mesh is then interpolated (or extrapolated) as $\mathbf{T}_\infty(\mathbf{C}) = \mathcal{F}(P_1, \ldots, P_p)$. For the interpolating function $\mathcal{F} : \mathbb{R}^p \to \mathbb{R}^m$, one can:

- Use a global interpolation function, possibly obtained as a least–squares fit over the set of pairs $(\tilde{\mathbf{C}}, \tilde{\mathbf{T}}_\infty(\tilde{\mathbf{C}}))$ (see fig. 1).
- Use a local interpolation function (typically multilinear) among the k "closest" (in some sense) contexts of the p-mesh to $\mathbf{C}$ (see fig. 2).

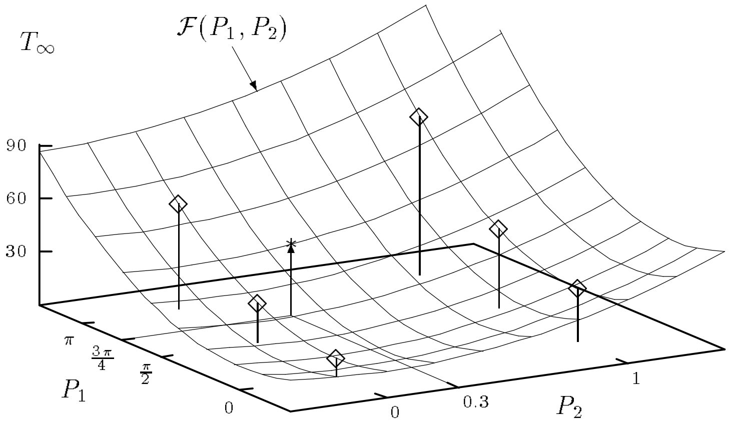

Figure 1: *Global interpolation for context* $\mathbf{C} = (\frac{3\pi}{4}, 0.3)$ *over parameters* $\tilde{P}_1 \in \{0, \frac{\pi}{2}, \pi\}$ *and* $\tilde{P}_2 \in \{0, 1\}$.

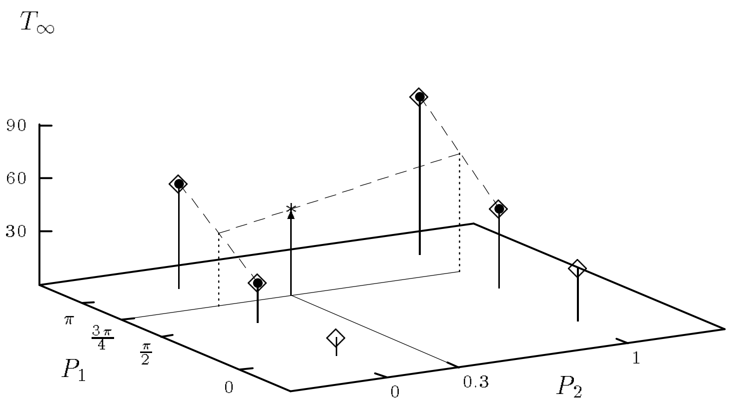

Figure 2: *Local bilinear interpolation for context* $\mathbf{C} = (\frac{3\pi}{4}, 0.3)$ *over parameters* $\tilde{P}_1 \in \{0, \frac{\pi}{2}, \pi\}$ *and* $\tilde{P}_2 \in \{0, 1\}$*, using 4 neighbours (highlighted).*

6. The temporal evolution of each single temperature component in the thermal map is considered to be an exponential drift from its current value to its steady–state value:

$$T_i(t + \Delta t) = T_{i,\infty} + (T_i(t) - T_{i,\infty})\, e^{-\frac{\Delta t}{\tau_i}}, \quad i = 1, \ldots, m \tag{2}$$

(as long as the context remains constant between t and $t + \Delta t$). The time constants τ_i have to be determined with the help of a thermal analysis tool, too.

Steps 1-4 take place off–line. Only steps 5-6 occur during the RTS.

It is important to notice that, while integrators *predict* the temperature distribution in a strict sense, interpolators just replicate a set of precomputed thermal maps —with some interpolation— in an efficient way (but possibly not accurate for general scenarios).

If in eq. (1) one takes $R_{ij} = 0$ and considers Q_i and T_j constant during a certain time interval, then the system is not anymore coupled and can be readily integrated. The explicit solution for T_i is given by eq. (2) with

$$T_{i,\infty} = \frac{Q_i + \sum_{i\neq j} K_{ij}T_j}{\sum_{i\neq j} K_{ij}}, \quad \tau_i = \frac{C_i}{\sum_{i\neq j} K_{ij}}.$$

This introduces errors, because the solution of system (1) cannot be expressed (in general) as a single exponential term for each T_i.

2.2.1 Advantages

- The computation time of $\mathbf{T}(t + \Delta t)$ depends on the interpolation function of step 4, but it will always be very short.
- The computation of the steady states is easy.
- The model is very simple and its implementation straightforward.

2.2.2 Disadvantages

- Poor accuracy, for two reasons:
  - There is an interpolation error in $T_{i,\infty}$ for those contexts not included in the tables.
  - Eq. (2) is not exact for any context (in general).
- Extrapolation (for contexts out of those foreseen, i.e. non–nominal scenarios) will produce uncertain predictions.
- It is necessary to store the tables that define $\tilde{\mathbf{T}}_\infty$, which can be very large if there are many contexts.

2.2.3 Actual implementations

The thermal subsystem of the RTSs developed at ESOC/SIM for the ISO, Italsat, Eureca and Pastel satellites, among others, is modelled according to an approach that can be considered as an interpolation method. The model consists of a "network"[2] made up with two different kinds of nodes: *heaters* (also called *hot objects*), which are elements that produce or absorb heat, and *thermistors* (also called *thermal nodes*), which are elements that measure temperature. The number of thermistors in the RTSs already developed is about 100.

The set of heaters and thermistors, together with their coefficients, is extracted in a first step from a detailed model of the satellite. This data is then adjusted by a thermal engineer by means of a trial–and–error procedure until the results are acceptable. The final model is usually available at a very late stage of the life cycle of the simulator. Until then, the software developers work with guesses.

According to the ISO Simulator Architectural Design Document (ref. [7]), the accuracy requirements for the thermal subsystem of the ISO simulator are of ±20% for the time constants and ±5°C for the tables of steady–state temperatures; the actual accuracy of the TTM is worse than that (a detailed analysis is awaiting completion).

3 Use of integration methods in real–time satellite simulators

The main reasons in favour of accomplishing the thermal simulation with an integrator instead of an interpolation method are the greater applicability to general scenarios and the improved accuracy of the results. However, this lays severe performance requirements on the computer CPU.

The use of integrators involves two independent operations:

- The actual integration of system (1), that provides with the S/C thermal map in due time. This task is performed by the solution routines of ESATAN, whose performance is evaluated in section 3.1.

[2] Not to be confused with the LPN used by integrators.

- The update of the coefficients of system (1). We will see that the bottleneck of the simulation is here.

3.1 Performance of the ESATAN solution routines

Many integration methods substitute the time derivatives in the left–hand side of system (1) by a first–order approximation:

$$\left(\frac{dT_i}{dt}\right)_t = \frac{T_i(t+\Delta t) - T_i(t)}{\Delta t} + \mathcal{O}(\Delta t^2) \tag{3}$$

The differential system (1) turns into an implicit or explicit algebraic system that can be solved with standard numerical methods (e.g. Newton–Raphson), giving the new temperatures at time $t + \Delta t$ from those at t. Δt is the *timestep* of the method and needs not be constant (in ESATAN it is called `DTIMEU`).

This is roughly the way the ESATAN routines `SOLVIT` (steady–state by Newton–Raphson), `SLFRWD` (explicit forward finite differences) and `SLFWBK` (Crank–Nicolson forward–backward finite differences) work. A least–squares fit for the CPU time in seconds employed by `SOLVIT` produced the following formula[3] (see ref. [13]):

$$\Delta t_{CPU} \approx \frac{t_i(n,c)i}{\text{MIPS}}, \quad t_i(n,c) = \alpha c + \beta n^2 + \gamma n$$

$$\alpha = 8.4 \cdot 10^{-4}\ \text{s}, \quad \beta = 3.4 \cdot 10^{-6}\ \text{s}, \quad \gamma = 6.3 \cdot 10^{-4}\ \text{s}$$

where n is the number of nodes and c the number of links of the model, i the number of iterations required to solve the algebraic system (1), $t_i(n,c)$ the time per iteration and the CPU power is expressed in MIPS (Millions of Instructions Per Second). The same formula has proven also useful for transient routines (with slightly worse accuracy); in this case, i is the number of iterations that take place during a timestep.

It is desirable to make as small as possible the real–time fraction $\frac{\Delta t_{CPU}}{\Delta t}$. The following factors have an influence on it:

- Timestep length, $\Delta t =$ `DTIMEU`: On the one hand, it should be as small as possible to minimise the approximation error[4] $\mathcal{O}(\Delta t^2)$ of eq. (3) and to update the S/C thermal map each 1-10 seconds[5]. On the other hand, a small Δt requires more computations per unit of time.

- $t_i(n,c) = \alpha c + \beta n^2 + \gamma n$ grows with n and c. The smaller the LPN the better (the faster), but simplifying a model means losing precision. Some companies have developed algorithms to reduce models, based on node grouping and node/link removal. However, a priori it is not clear how significant can be such alterations of the original model.

- MIPS: currently, the satellite simulators developed at SIM run on VAXstations ranging from model 3100 (7 MIPS) to 4000.90 (30 MIPS), which is the standard configuration. In a near future they will run on DEC/Alphas (with over 100 MIPS).

- Number of iterations accomplished, i (`LOOPCT` in ESATAN): it depends on the problem and on the desired accuracy of the iterative method (ESATAN control constants `RELXCA` —convergence criterion— and `NLOOP` —maximum number of iterations—). For the large models tested (SOHO: 640 nodes, 14500 links; ISO: 1100 nodes, 18000 links), `LOOPCT` seems to be typically around 100-130 iterations in `SOLVIT` for `RELXCA` between 0.001 and 0.01.

[3] The fit was developed running ESATAN 5.5 in a VAX 3800 with a sample of 10 small, quasi–linear models with void `$VARIABLES1`, `$VARIABLES2` and `$OUTPUT` blocks. Later, it has been applied to large models (SOHO and ISO satellites) with quite good results (maximum relative error of 15%). ESATAN 6.3 seems to run about 15% faster.

[4] Furthermore, for explicit routines (`SLFRWD`) it should not be greater than

$$\min_{i=1,\dots,n} \left\{ \frac{C_i}{\sum_{j \neq i} K_{ij}} \right\}$$

(this number is called `CSGMIN` in ESATAN); otherwise the numerical method could be unstable and diverge. For implicit routines (`SLFWBK`), Δt is unbounded.

[5] Which is the typical TTM period. However, if the actual update period is greater, intermediate values can be interpolated with a small error.

One must also take into account that there are other subsystems in the simulator that have to share the CPU with the thermal one (e.g. Attitude and Orbit Control System, Power Generation and Distribution, etc.). This imposes the restriction that the thermal routines run much faster than real–time: $\Delta t_{CPU} \ll \Delta t$.

The following table gives, for some combinations of n and c and for two CPUs of 30 and 100 MIPS, the minimum timestep length, assuming that $i = 100$ iterations and that only 10% of the CPU time is allocated to the thermal solution routine:

n	c	Δt_{min}(30 MIPS)	Δt_{min}(100 MIPS)
50	300	9.7	2.9
100	1000	31	9.4
200	3000	93	28
500	10000	320	96

Table 1: *Minimum timestep in seconds for real–time simulation of ESATAN models.*

Row one would read "the thermal subsystem (ESATAN library routine for transient solution), with a network of 50 nodes and 300 links, could be called once each 2.9 seconds consuming 10%(2.9) = 290 ms of CPU time in a 100-MIPS machine." In this case, the temperature update period would be about 3 seconds without an excessive cost of CPU time. However, one can see that large models are prohibitive.

3.2 Update of the LPN coefficients

There are several factors upon which the parameters of the LPN depend. Whenever these factors change, the coefficients must be recomputed and this can put a severe load on the CPU. The main factors are:

1. A **TCU (Thermal Control Unit) command** or a **telecommand sent by an operator** that actively changes one parameter. For example, a heater can be switched on, making a certain Q_i grow from 0 to 10 W. However, they happen relatively seldom and hence do not represent a threat for the RTS. Furthermore, they are not the responsibility of the simulator thermal subsystem.
2. Most coefficients (e.g. the thermal conductivity) are more or less dependant on the **temperature**, which, in turn, depends on the time. Fortunately, in most cases the temperature does not change too fast, so that one can consider the coefficients independent of T during long periods of time.
3. The **S/C geometry**: there are usually very few mobile parts in a satellite and their movements are slow: antennae, solar panels (mainly during deployment), booms or robot arms. The conductive links stay roughly the same during a movement but the radiative couplings do not, because the view factor between different nodes can undergo a dramatical variation (previously sunlit surfaces can fall inside shadows, for example). This means that the radiative couplings need to be recalculated.
4. The **S/C attitude and orbital coordinates** determine how the external fluxes (direct Sun flux, Earth albedo, Earth flux) are received by the thermal nodes. Again, this effect cannot be neglected, at least the direct Sun flux. For geostationary satellites, the fluxes coming from the Earth could be disregarded, but not for low Earth orbit satellites.

Factors 1 and 2 do not pose problems for the RTS, because the computations required by them are small and happen quite seldom. Nevertheless, the computation of the radiative couplings and the external fluxes is usually carried out by a ray–tracing algorithm (such as that used by ESARAD), which takes in the order of minutes even for medium–sized models and a powerful CPU. This completely discards its use in real–time, and the obvious solution is to update the R_{ij} and Q_j from tables by interpolation.

4 Conclusions

The performance of the computer facilities available in future to ESOC for running the real–time satellite simulators allows the use of a hybrid method in the thermal subsystem: numerical integration of a reduced ESATAN thermal model with table interpolation for the radiative couplings and external fluxes. This will improve the accuracy of the generated thermal telemetry. The inclusion of ESATAN will bring two additional benefits: the use of mathematical models designed right from the beginning by thermal engineers and the reliability of routines validated through years of use.

5 Future work

A version of ESATAN tailored for real–time (that is, optimised in speed and free of superfluous features —something like the recently released MINITAN) would be welcome. It will also be necessary to develop an interface between ESATAN and the real–time simulator (unless the integration takes place at source code level, which would be the most efficient arrangement).

As the figures of table 1 show, ESATAN demands very high CPU performance even for the smallest models. Further work should be done to find a powerful model reduction algorithm with an error estimate (specially valuable would be a reduction of the number of links, because they have —for small and medium models— a greater weight in the CPU consumption).

A Simulation of interpolation methods by means of an LPN

In this section we show how to construct a lumped network for ESATAN that will produce, for a given system, the same thermal map as an interpolation–based method, that can therefore be considered a subset of the more general LPN approach.

We make use of a special kind of node available in ESATAN: a *boundary* node. This is a node whose temperature is constant; it is usually employed to represent deep space (which is assumed to absorb any quantity of heat keeping its temperature of -273.15°C). The "normal" nodes are called *diffusion* nodes.

The physical units employed are those of the SI (*Système International d'Unités*), and °C for the temperature. These are also the default in ESATAN.

The LPN is depicted in fig. 3. There are m diffusion nodes $N_1, \ldots, N_m$ corresponding to the m components of the S/C thermal map $\mathbf{T}$, and m boundary nodes $B_1, \ldots, B_m$ that contain the steady–state thermal map $\mathbf{T}_\infty$. N_i has $C_{N_i} = \tau_i$ and is connected to B_i by one link of conductance 1. Consequently, eq. (1) particularised to fig. 3 becomes:

$$C_{N_i}\dot{T}_{N_i} = T_{B_i} - T_{N_i} \quad \Longleftrightarrow \quad \tau_i\dot{T}_i = T_{i,\infty} - T_i$$

which is eq. (2) in differential form.

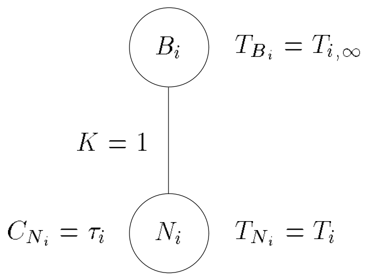

Figure 3: *One component of the fake LPN, which follows eq. (2)*

The steady–state interpolation $\mathbf{T}_\infty(\mathbf{C}) = \mathcal{F}(P_1, \ldots, P_p)$, that updates the temperature of the boundary nodes, could be carried out by a user subroutine.

Fig. 4 shows a sample ESATAN input deck for the LPN mentioned.

```
$MODEL BUBU                         # Thermal map: 100 components
$NODES
# Diffusion nodes with time constant and initial T
D   1 = 'Thermistor 1',      T = 10, C = 40;
...
D 100 = 'Thermistor 100',    T = 30, C = 35;
# Boundary nodes with steady-state T of associated diffusion node
B1000 = 'ssThermistor 1',    T = -10;
...
B1100 = 'ssThermistor 100', T =  40;
$CONDUCTORS
FOR KL1 = 1 TO 100 DO
GL(KL1,KL1+1000) = 1;               # Conductance K=1
END DO
$CONSTANTS
$CONTROL
TIMEND = 100.0;                     # Time at end of transient
OUTINT =   5.0;                     # Output interval
RELXCA = 1E-10;                     # Convergence criterion
NLOOP  = 50000;                     # Maximum no. of iterations
DTIMEI =   1.0;                     # Initial timestep
DTPMAX =   1.0;                     # Max. temperature change
$ARRAYS
# Context interpolation data could go here
$SUBROUTINES
# Steady-state interpolation could be done here
$EXECUTION
      CALL SLFWBK
$OUTPUTS
...
$ENDMODEL BUBU
```

Figure 4: *Sketch of an ESATAN input deck for a* fake *LPN*

References

[1] "ESATAN Engineering Manual," Rel. 2.3, Analysis & Verification Section of ESA/ESTEC and Engineering Research Centre of GEC Alsthom, August 1992 (EM–ESATAN–056).

[2] "ESATAN Training Manual," Rel. 4.4, Analysis & Verification Section of ESA/ESTEC and Engineering Research Centre of GEC Alsthom, October 1992 (TM–ESATAM–008).

[3] "ESATAN User's Manual," Rel. 2.1, ESA Publications Division, ESTEC, November 1991 (ESA PSS–03–105 Issue 2).

[4] "ESARAD 2.2C Reference Manual," Rel. 4.1, Analysis & Verification Section of ESA/ESTEC and EGT Engineering Research Centre, July 1994 (UM–ESARAD–083).

[5] "ESARAD 2.2C User Manual," Rel. 4.1, Analysis & Verification Section of ESA/ESTEC and EGT Engineering Research Centre, July 1994 (UM–ESARAD–024).

[6] Flett, D. C.: "ESARAD Getting Started Guide," Rel. 2.1, Mechanical Engineering Centre of European Gas Turbines Ltd., July 1994 (ESARAD–113).

[7] ISO Development Team (SPS/Sims/Marcol): "Architectural Design Document for the ISO Simulator," ESOC/ECD/SPD/Sims, July 1991.

[8] "ISO Simulator Detailed Design Document (DDD) Vol. 14: Thermal Model," ESOC/ECD/SPD/Sims, June 1992 (ISO.DD.094).

[9] Krogh, H. K.: "Italsat Simulator Detailed Design Document (DDD) Vol. 12: Thermal Model," ESOC/ECD/SPD/Sims, December 1989 (ITA.DD.061).

[10] Hocken, R. D.: "Eureca Simulator Detailed Design Document (DDD) Vol. 10: The Thermal Control System," ESOC/ECD/SPD/Sims, June 1989 (EUR.DD.0036).

[11] "Pastel Simulator Detailed Design Document: Thermal Model," ESOC/FCSD/SIM, 1994.

[12] Mastronimico, G.: "Pastel Simulator User's Manual," ESOC/FCSD/SIM/CISET, September 1994 (PAST–SST–SUM–0195–SIM).

[13] Carreira Perpiñán, M. Á.: "ESATAN: An Evaluation," ESOC/FCSD/SIM, March 1994 (DOPS–SST–TN–0215–SIM).

[14] "Spacecraft Modelling Methods in Support of Simulation of Future Missions," Matra Marconi Space, September 1994: "Survey of the State of the Art in Modelling Methods" (SMM–NT–005) and "Software Design and Implementation Issues" (SMM–NT–006f).