Rapid Thermal Analysis of Rigid Three-Dimensional Bodies with the Use of Modelica Physical Modelling Language

Corey Bolduc Chahé Adourian
Canadian Space Agency, Department of Space Technologies
6767 Route de l'Aéroport, St-Hubert, Quebec, Canada
coreybolduc@yahoo.com Chahe.Adourian@asc-csa.gc.ca

Abstract

Quick analysis of thermal systems without the use of CAD models may be very valuable for rapid-prototyping. Such a need led to the heavy modification of the Heat Transfer package found in the Modelica standard library. The library in question revolves around a three-dimensional generic block composed of a variable number of interconnected elements with individually assignable thermal and physical properties. When applying the accompanying library functions, one may create moderately complex thermal structures with non-uniform thermal properties. Furthermore, other tools available also allow the user to attach chains of blocks of different resolutions, as well as insert one block into another to create composite models with the possibility of internal heat generation.

Keywords: Thermal Analysis, Modelica, Finite Element Analysis, Matlab

1 Background

Thermal analysis software may be generally categorized into one of two groups, each with their own respective strengths and weaknesses. The first group represents packages such as the 1-D HeatFlow library in Modelica which relies on primitive nodal techniques to produce relatively crude analyses of thermal models, but in short order and with little effort. On the opposite side of the spectrum lie the other group consisting of highly developed Finite-Element Analysis software such as NASTRAN, ANSYS, and CATIA. Although the latter group offers superior fidelity, CAD models are required for analysis, which is often time consuming in itself to produce. Somewhere in the middle lie a group of hybrid tools which may be considered as rapid thermal analysis tools, and whose importance is only beginning to be recognized. They are characterized by being capable of modeling thermal models that go beyond 1-D but at the same time don't require a CAD model in order to create them. We describe in this paper the implementation of such a thermal modeling library with the use of the Modelica language.

2 Introduction

The need for a rapid thermal analysis tool arose naturally with the development of a multidisciplinary satellite simulator. Among other criteria, a means of predicting the temperature for specific sections of the spacecraft at any point in orbit became necessary. By combining multiple internal and external heat sources as those produced by solar radiation and various onboard power systems, a complete and reasonably accurate analysis may be executed for both transient and steady-state scenarios. Utilizing generic thermal models, one may quickly create specialized components which may be "snapped" together, thereby creating complex combinations quickly.

The overall objective is visualized by the components seen in figure (1). The top part of the figure shows the satellite being modeled with its two solar panels on the left and the right and the main body in the middle. These different components are connected mechanically, electrically and thermally. Below the satellite are three Modelica device blocks each containing behavioral models from each of the physical disciplines mentioned previously (and more). Their structure reflects that of the spacecraft. On the left and right are the Modelica models of the solar panels and in the middle the main body. Each device model possesses proper interfaces in each discipline to connect to its neighbors and more importantly for the purposes of this paper they each contain a *Thermal Block* model shown below the device. The *Thermal Blocks* in turns

DOI: 10.3384/ecp09430036

is composed of *Thermal Elements* which are shown as nodes on a grid at the bottom of the figure.

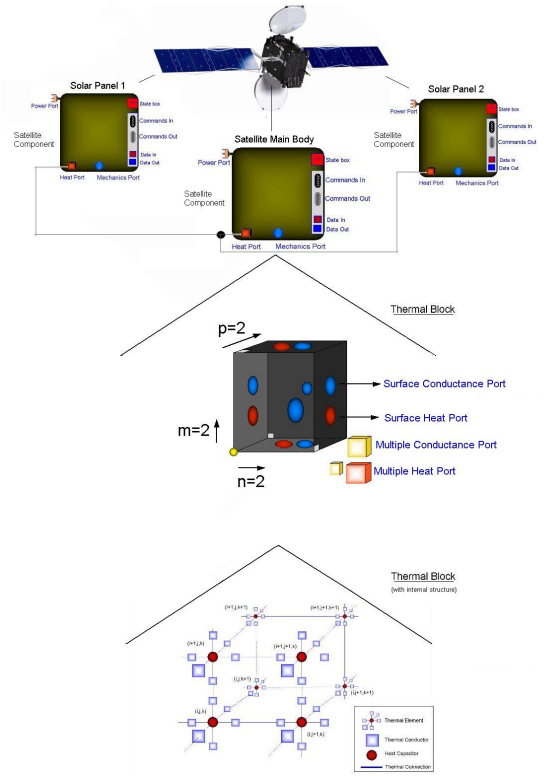

Figure 1: Example

The *Thermal Element* was designed such that it allows and simplifies the creation of hierarchically structured thermal components that fits nicely within the device models and can be representative of the hierarchical structure of real systems. This is accomplished by providing various types of ports permitting direct access to heat flow and temperature variables within *Thermal Elements* and *Thermal Blocks*. Designers have then the freedom to apply different boundary conditions (e.g. temperature, or heat flow) to individual elements and define thermally dynamic components/devices such as a battery or a heater.

Each *Thermal Element* in *Thermal Block* may be assigned thermal properties different from those of its neighbors' such as specific heat (heat capacitance), thermal conductance and initial temperature. Thermal conductance is of particular interest since the ability to assign it a value of zero effectively prevents any heat flow through an element, resulting in what accurately represent a cavity. The assignment of the aforementioned properties is carried out by use of base functions (three shapes for each individual thermal property and density). For each property, these functions are available for spherical, rectangular prismatic, cylindrical distributions, and are used to assign a particular property value within the shape's boundary and another on its outside.

Other tools developed also give the user the option to interface blocks of dissimilar resolution, or to insert multiple blocks into a single block in order to represent a satellite containing interacting components of varying fidelity, and heat sources through conduction and radiation (future development).

Despite its powerful simulation capabilities, Modelica (and Dymola) leave much to be desired in terms of user interfacing and three-dimensional graphic visualization of simulation results. In order to expedite results analysis, a program was created in Matlab to display an animation of the temperature and heat flow distribution within a body (or bodies) as a function of time. It is the hope that in the future further advances of the Matlab code will lead to an independent executable which will be able to act as the interface between the user and Dymola, thereby facilitating the initial configuration and debugging of models.

3 Thermal Element

The basic building block of the thermal model is the thermal element representing the thermal behavior of a cubic shape with isotropic thermal properties. The mathematical modeling follows from finite volume methods as explained in [1]. In the following sections, we provide the details of this model.

3.1 Physics

The *Thermal Element* is comprised of six thermal conductors to represent thermal connections from any of the six faces of the cube and connected to a central heat capacitor modeling the cube capacitance via a heat port (Figure 2). Two thermal conductors are placed on each side of the heat capacitor in each direction. The thermal conductors transport heat and are normally assigned a constant thermal conductance value given by equation (3.2) where G_i is the heat conductance, k is the thermal conductance, A_i is the cross-sectional area and L_i is the length of the cube in the direction

$$i \in \{+x, -x, +y, -y, +z, -z\} \tag{3.1}$$

of heat flow.

$$G_i = k \frac{A_i}{L_i/2} \tag{3.2}$$

To calculate the heat flow through each thermal conductor equation (3.3) is used where T_i and T_c are re-

spectively the temperatures of surface A_i and the center of the cube. Heat flow into the cube has positive sign.

$$Q_{ThermalConductor_i} = G\,\frac{T_i - T_c}{L_i/2} \tag{3.3}$$

The heat capacitor modeling the heat storage capacity of the cube completes the thermal model of the single cube. Given the specific heat capacity c and the mass of the cube its thermal capacitance C is calculated using equation (3.4).

$$C = c\,m \tag{3.4}$$

Heat flow into the capacitor is given by equation (3.5) where $\frac{dT_c}{dt}$ is the time rate of change of the capacitor's (or cube's) temperature.

$$Q_{HeatCapacitor} = C\,\frac{dT_c}{dt} \tag{3.5}$$

Because a heat flow is associated with each thermal conductor and the conductors are thermally coupled to the central heat capacitor, the total heat flow into the capacitor (or equivalently the cube) is the sum of heat flows through the conductors (or cube surfaces) and is given by equation (3.6).

$$Q_{HeatCapacitor} = \Sigma Q_{ThermalConductor_i} \tag{3.6}$$

The *Thermal Element* is a single node in a rectangular grid discretisation of a physical object given by the *Thermal Block* described in section (4).

3.2 Coding

The thermal element is assembled using standard components from the Modelica library. Relevant components include thermal conductors and a heat capacitor from *Modelica.Thermal.HeatTransfer* library and constant sources from the *Blocks.Sources* library. These are arranged as shown in figure 2.

3.3 Interfaces

A *Thermal Element* has many different interfaces useful under different circumstances.

Default Heat Ports In the basic use case of a *Thermal Element*, the elements are packed side by side and their heat ports connected individually top-to-bottom, left-to-right and front-to-back in order to form the grid nodes of a single *Thermal Block*.

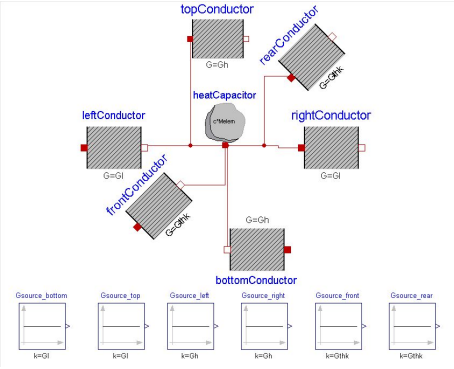

Figure 2: Thermal Element model

Multiple Heat Port In some situations that we will encounter later, it is convenient to have a single connector which combines all six heat-ports of a *Thermal Element*. A custom connector called *HeatPortMulti* was created for this purpose and is part the *Thermal Element*'s interfaces.

Conductance Ports In addition to the heat ports, there are six signal output ports that hold the conductance values G_i and are used when connecting to a resolution adapter (i.e. Mapper or Surface Interface). Under some circumstances, it is necessary for a *Thermal Element* to share with connected thermal components its conductance value for correct physical modeling.

4 Thermal Block

The *Thermal Block* pictured as a grid in figure (3) is the main physical representation which the user initializes and analyzes after simulation. It represents a block of material in the shape of a rectangular prism and is composed of *Thermal Elements* set on a cartesian grid. Setting the parameters of the individual *Thermal Elements* is the method by which the user defines the actual shape of the prism. This is equivalent to a Finite Element Model with the parameters at each grid point determining both the type or even presence of material together with its physical properties. For example, an empty cavity inside the prism can be modeled by setting the conductance to zero at grid points in the cavity. Different materials may be specified at different grid points again by varying the parameter values at these points.

4.1 Mathematical Model

The *Thermal Block* is assumed to be a rectangular prism with height H (x-direction), length L (y-

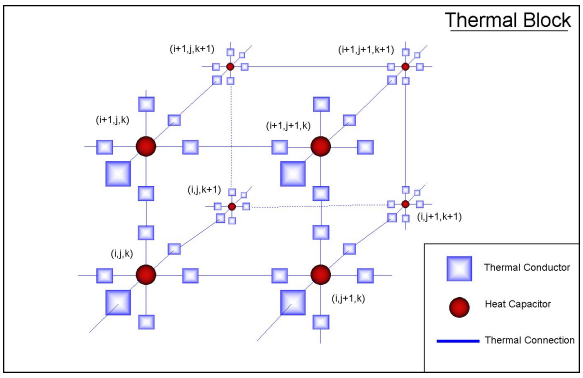

Figure 3: Thermal Block

direction) and thickness Thk (z-direction). The grid resolutions or the number of *Thermal Element* nodes in each direction are specified by integers N_x, N_y and N_z which the *Thermal Block* used to create a lattice structure of *Thermal Elements* connected by their heat ports.

Thermal conductivity, specific heat capacity, initial temperatures and mass-density distributions must also be provided and are used to initialize the *Thermal Elements*. The physical dimensions and grid resolutions are used to calculate *Thermal Element* lengths L_i and cross-sectional areas A_i in each direction i which in turn are used to calculate the individual thermal parameters C and G_i.

4.2 Physical Property Distributions

By default, the aforementioned properties are set to be filled uniformly with a default value (arbitrarily chosen to be that of aluminum) and can be changed by the user. However, uniform distributions are not very interesting and another option is required.

Components with complex geometries and non-uniform thermal properties are made possible with the aid of pre-defined functions. We defined example functions that provide the parameter distributions necessary to model cylinders, spheres and rectangular prisms of constant k, c, T_0 and ρ values. Thermal property values for both inside and outside the shape in question are specified. Moreover, the center of each shape function may be positioned anywhere within the *Thermal Block*, and superimposed on one another. For example, two create a thermal model representing two parallel cylinders side by side we take a single *Thermal Block* and apply a superposition of two Cylindrical distributions with proper offsets.

By extending the use of distribution functions, one may create cavities in the material by setting the conductance of certain thermal elements to zero, thereby effectively preventing heat flow through them as well as reducing the number of equations to solve. The hollow function originally prevented the connection of heat ports between neighboring *Thermal Elements*. This was counterproductive as these connections were needed when inserting a *Thermal Block* into another. Therefore, the current cavity function works by assigning a thermal conductance value of zero to the thermal element to be isolated from its neighbors. Given the resulting "empty space" within the material, this does not forbid the possibility to connect another block to the cavity walls.

4.3 Interfaces

The *Thermal Block* has various interfaces for connecting to other thermal components. There are six interfaces to connect to the surfaces of the block. The internal grid elements of the block are also visible allowing for example to connect an external heat source to the center of a *Thermal Element* or to insert a block into another.

In order to allow for these complex interconnections, access to heat and conductance value information were devised through the creation of specialized ports. The first, referred to as a HeatPortMulti port, includes six standard heat ports (one per *Thermal Element* surface) each of which connects automatically when HeatPortMulti's are connected to one another. Such a port was implemented in order to facilitate the insertion of one block into another by the Mapper which is covered later. The second type of port, the ConductanceMultiPort port, serves the same function as the former, but for conductance.

Both types of ports were implemented as arrays of size and dimensions equal to those of the *Thermal Elements* within the *Thermal Block*, and were connected to the corresponding sides of the appropriate element before connecting to one another in sequence.

With an infrastructure established to access the internal elements of a *Thermal Block*, six standard heat port arrays and six conductance port arrays, each of dimension two, were introduced and connected to each side of the *Thermal Block* in order to facilitate surface-to-surface connections. These side ports are created conditionally only when connecting a particular side, so as to avoid the creation of mathematically problematic extra equations.

4.4 Block Insertion

Given the grid structure of the *Thermal Blocks*, it is natural to consider overlaying two such grids. Such a scenario arises when we want to model components contained within components. For example the satellite's external metallic frame contains within it a multitude of components including the computer, sensors, actuators, batteries, etc. We would like one *Thermal Block* to represent the satellite's structure with a cavity within. Another block could represent the battery which is smaller than the first block and is contained within it. Inserting one block into another can be seen as an overlaying of the two thermal grids at the intersection of the two physical objects. Thermal connections must be established at the interfacing surfaces. The Mapper component enables this by accessing the internal elements of each block and properly mapping them to each other.

5 Surface Interface

The *Surface Interface* model serves to thermally couple the external surfaces of two thermal blocks with different grid resolutions. When attempting to thermally connect two such bodies, one finds that their respective array indices cannot be correlated to each other one-to-one. Instead a more complicated linking mechanism must be implemented which requires not only connections to the surface heat ports but also knowledge of the conductance of each of the surface elements. Therefore the surface interfaces of *Thermal Blocks* contains both heat port and conductance port arrays of dimension two on each end. The array connectors of the *Surface Interface* are initialized manually by the user and must match the array sizes of the surface arrays on the adjacent bodies . Information on heat and conductance from each body is passed through both ends to a custom thermal conductor array which receives its conductance value externally through a conductance connection. In order to implement physically correct heat distribution, algorithms were implemented to distribute the heat flow from one element to the appropriate number of elements on the opposite end.
Below is a sample of the distribution algorithm for the connection between the input heat port array and a thermal conductor array:

```
for i in 1:s loop
  for j in 1:t loop
    for Mi in (1:sp) loop
      for Mj in (1:tp) loop
        M1[Mi + (i - 1)*sp, Mj + (j - 1)*tp].G
          = CPL[i, j] / (areaRatio1-1);
        connect(HPL[i, j],
          M1[Mi + (i - 1)*sp,
          Mj + (j - 1)*tp].port˙a);
      end for;
    end for;
  end for;
end for;
```

M1 is one of the external thermal conductor arrays, sp and tp are the corresponding dimension sizes to s and t and (either u or v depending on surface rotation), $areaRatio1 = u \times v$, and HPL is the heat port array on the left side.

Interface parameters The parameters s and t represent the number of elements in each planar direction of the interacting surface of the left body (u and v for the right body). The algorithm essentially functions by finding the product of the surface dimensions and their counterpart dimensions on the opposing side (e.g. $areaRatio1 = u \times v$). The heat flow and thermal conductance values for each body surface element are individually supplied to a number of thermal conductors equal to that of the number of elements on the surface of the other body. The thermal conductor itself does not in fact utilize the received thermal conductance value immediately, but instead increases or decreases its value in order to avoid the intermediate thermal conductor arrays interfering with heat flow, which would ultimately skew the results. Included in the surface interface model is the option to rotate the right surface with respect to that of the left surface (input). Such a feature becomes necessary when constructing objects such as a cube out of four separate blocks.

Contact Resistance In the physical world, when two objects come into contact with one another, there exists a thermal resistance known as thermal contact resistance. Thermal contact resistance represents the thermal resistance created by the two materials not coming into complete contact with one another due to microscopic roughness and resulting spaces. These spaces may either contain a fluid or vacuum, each of which will inhibit the heat flow between the materials. In order to simulate the aforementioned phenomenon, a standard thermal conductor array is implemented in the interface whose thermal conductance ($G = A/Ri$) is based on the contact area and the interface resistance. Contact resistance tables exits for a variety of

materials, finishes, roughness, temperature and other conditions.

Surface Mapping In order to distribute the heat flow accurately through the surface interface from one block to another, two separate thermal conductor arrays are employed. This is shown in figure (4). The two resolution values (e.g. m and p) for each surface are multiplied by their directional counterpart to find a product ($m \times p$). The thermal conductor arrays are then established as being size $(mSide_1 \times mSide_2) \times (pSide_1 \times pSide_2)$. The intermediate thermal conductor array is based on a standard (library) thermal conductor, but fitted with a heat and conductance port, thereby allowing it to have its G inputted by the G of the connecting element in the adjacent block. With current value of G known for each element in a block and the desired resolution change, the equivalent G may be computed. Therefore, the computed G serves to increase, or decrease resistance as required across the surface interface, in order to obey heat conservation.

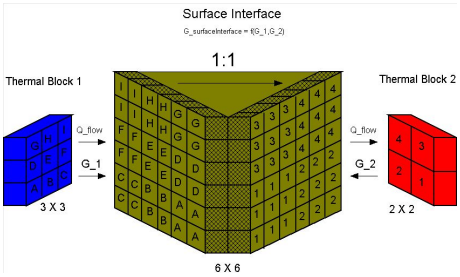

Figure 4: Surface Interface

6 Mapper

A three-dimensional extension to the surface interface is the Mapper. The Mapper allows for one body of a particular resolution, to be thermally coupled within another block of a differrent resolution. Such an option would allow one to be able to insert a power supply into a certain section of the satellite's main body.

7 Animation

Because of the somewhat limited data visualization capabilities in Dymola when working with three-dimensional data, a visualization tool was developed using Matlab. At the present moment the visualization tool accepts Dymola analysis data from the thermal model run and renders a three-dimensional animation. Visualization allow color-coded animation of the temperature scalar data as well as heat flow vector information. Figure (5) shows an example heat distribution within a *Thermal Block* at a specific point in time. The visualization capability is of great help in determining whether our model is behaving properly and for debugging purposes.

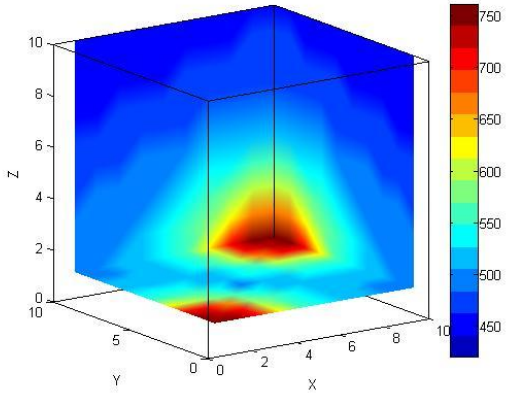

Figure 5: Temperature Distribution Snapshot

8 Conclusion

In conclusion, we have developed an experimental thermal modeling library using Modelica that lies halfway between simple 1-D modeling tools and advanced CAD-based ones. The block diagram modeling capabilities of Modelica were used to provide high-level snap-on thermal models that could represent 2-D and 3-D models with non-uniform thermal distributions. Further, mechanisms to connect these 3-D models in complex ways were developed including the capacity to insert blocks one into the other following the idea of hierarchical composition.

References

[1] Earl A. Thornton. Thermal Structures for Aerospace Applications. AIAA, 1996: 98-99.

