JSC-61777-models/interactions/thermal_rider

JSC Engineering Orbital Dynamics Thermal-Rider Model

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

Package Release JEOD v5.1

Document Revision 1.0
July 2023

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC-61777-models/interactions/thermal_rider

JSC Engineering Orbital Dynamics Thermal-Rider Model

Document Revision 1.0
July 2023

Gary Turner

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

The Thermal-Rider Model is not a stand-alone model; it requires some connectivity to another Interaction model that defines an Interaction Surface (e.g. Radiation Pressure, or Aerodynamic Drag). Its connection to the Surface Model is indirect, seen only through the Interaction Surface, not the surface itself.

The Thermal-Rider Model serves to provide any Interaction Surface with thermal monitoring capabilities. Given the current environment, it integrates the thermal processes (e.g. internal power dump, skin friction, radiative absorption, thermal emission) using an R-K 4th order integrator to give the temperature evolution of the surface.

Contents

1 Introduction **1**

1.1 Purpose and Objectives of the Thermal-Rider Model . . . 1

1.2 Context within JEOD . . . 1

1.3 Documentation History . . . 2

1.4 Documentation Organization . . . 2

2 Product Requirements **3**

2.1 Requirements Traceability . . . 4

3 Product Specification **5**

3.1 Conceptual Design . . . 5

3.1.1 The Rider Hierarchy . . . 5

3.1.2 Independent Integration . . . 6

3.1.3 Handling Multiple Interaction Models . . . 6

3.2 Mathematical Formulations . . . 8

3.2.1 Basic Equations . . . 8

3.2.2 Temperature Propagation . . . 8

3.2.3 Runge-Kutta 4th Order (RK4) Integrator . . . 8

3.2.4 Emitted Power . . . 11

3.3 Detailed Design . . . 13

3.3.1 Process Architecture . . . 13

3.3.2 Functional Design . . . 21

3.4 Inventory . . . 23

4 User's Guide **24**

4.1 Analysis . . . 24

4.1.1 Model-wide Descriptors . . . 25
4.1.2 Facet-specific Descriptors . . . 25
4.1.3 Facet Parameter Descriptors . . . 26
4.2 Integration . . . 27
4.2.1 Thermal-Rider Model . . . 27
4.2.2 Thermal Facet Rider . . . 27
4.3 Extension . . . 28
4.3.1 Extending the Capability of the Thermal-Rider Model. . . . 28
4.3.2 Using the Thermal-Rider Model to Extend the Capability of an Interaction Model. . . . 28
4.3.3 Thermal-Rider Model . . . 28
4.3.4 Thermal Facet Rider . . . 28

5 Verification and Validation 30
5.1 Verification . . . 30
5.1.1 Inspections . . . 30
5.1.2 Verification of the Temperature Modeling . . . 30
5.2 Validation . . . 36
5.3 Metrics . . . 37
5.3.1 Code Metrics . . . 37

Chapter 1

Introduction

1.1 Purpose and Objectives of the Thermal-Rider Model

In some circumstances, it is desirable to have a temperature profile (both temporal and spatial) of the vehicle, while in others, it is more important to avoid the computational effort required to maintain that profile. For this reason, the thermal monitoring capabilities have been included as a separate entity, although without a surface on which to act, thermal monitoring is not possible.

Therefore, the thermal model is provided as an add-on, or a rider, to any other interaction model. So, for example, aerodynamic drag can be calculated without any reference to skin temperature (or where a reference is required, it can be assigned to a constant value without great loss of precision). Or, if the user wishes to observe the effect of drag on the skin temperature, and perhaps then on the drag profile itself, the Thermal Rider can be added to the Aerodynamic Drag model. It must be cautioned, however, that the Thermal Rider Model is not a stand-alone model; it requires some connectivity to another Interaction Model that defines an Interaction Surface (e.g. Radiation Pressure, or Aerodynamic Drag). Its connection to the Surface Model is indirect, coming only via the Interaction Surface; it cannot work on a Surface that is not already an Interaction Surface. For details on the concept of surfaces and interaction surfaces, see the Surface Model [4] documentation.

In the JEOD v5.1 release, the Thermal Rider model provides only the capabilities found in previous releases of JEOD, but now with the capability to easily add functionality as needed. JEOD v5.1 implementation includes capacity to evaluate the effect of radiation absorption (from the Radiation Pressure model), and the effect of internal vehicular energy sources and sinks. The extension section of the User's Guide contains suggestions for including effects of facet-to-facet conduction, and of heating resulting from aerodynamic drag.

The thermal evolution is provided by an internal R-K 4th-order integrator, and is independent of the dynamics integration.

1.2 Context within JEOD

The following document is parent to this document:

- *JSC Engineering Orbital Dynamics* [2]

The Thermal-Rider Model forms a component of the interactions suite of models within JEOD v5.1. It is located at models/interactions/ThermalRider. It is used primarily by the Radiation Pressure [6] model, where it can be used in conjunction with radiation pressure, or as a stand-alone model for simply monitoring temperature. It may also be used in the Aerodynamic Drag [5] model.

1.3 Documentation History

Author	Date	Revision	Description
Gary Turner	November, 2009	1.0	JEOD2.0.0 release

1.4 Documentation Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [3] and is organized into the following chapters:

Chapter 1: Introduction - This introduction contains four sections; purpose and objective, context within JEOD, document history, and document organization.

Chapter 2: Product Requirements - Describes requirements for the Thermal-Rider Model.

Chapter 3: Product Specification - Describes the underlying theory, architecture, and design of the Thermal-Rider Model in detail. It will be organized in four sections; Conceptual Design, Mathematical Formulations, Detailed Design, and Version Inventory.

Chapter 4: User's Guide - Describes how to use the Thermal-Rider Model in a Trick simulation. It is broken into three sections to represent the JEOD defined user types; Analysts or users of simulations (Analysis), Integrators or developers of simulations (Integration), and Model Extenders (Extension).

Chapter 5: Verification and Validation - Contains Thermal-Rider Model verification and validation procedures and results.

Chapter 2

Product Requirements

Requirement ThermalRider_1: Top-level Requirement

Requirement:

This model shall meet the JEOD project requirements specified in the JEOD v5.1 top-level document [2].

Rationale:

This model shall, at a minimum, meet all external and internal requirements applied to the JEOD v5.1 release.

Verification:

Inspection ThermalRider_1 on page 30

Requirement ThermalRider_2: Temperature Monitoring

Requirement:

The Thermal-Rider Model shall provide the capability to monitor the temporal and spatial profiles of the vehicle surface temperatures.

Rationale:

The temperature of a particular area of a vehicle may be of critical importance in the design of the vehicle; variation of temperature across the surface of the vehicle could affect the dynamics of the vehicle in a high fidelity simulation. Therefore, it is important to be able to identify the temporal temperature profile of certain critical areas of the vehicular surface, and hence the capability to model the spatial variation across the vehicle.

Verification:

Tests ThermalRider_1 on page 31 and ThermalRider_2 on page 33

Requirement ThermalRider_3: Minimum Functionality

Requirement:

The Thermal-Rider Model shall provide, at a minimum, the ability to accurately represent the effect of radiation absorption and emission.

Rationale:

Prior to the release of JEOD v5.1, the thermal modeling had been an intrinsic part of the radiation-pressure model, which requires knowledge of the temperature-time profile in order to accurately represent the effects of radiation-pressure. This requirement simply reflects the transfer of responsibility for these calculations from the old radiation model (which incorporated both radiation and thermal modeling) to the new Thermal-Rider Model.

Verification:

Tests ThermalRider_1 on page 31 and ThermalRider_2 on page 33

Requirement ThermalRider_4: Extended Functionality

Requirement:

The Thermal-Rider Model shall, at a minimum, provide the basic architecture to allow future extension to include representation of the effects of:

1. Power sources and sinks from within the vehicle itself.
2. Facet-to-facet thermal conduction.
3. Heating resulting from aerodynamic drag.

Rationale:

There are many methods by which the temperature of a surface can be altered. These three, in addition to the effect of radiative absorption and emission, were considered to be the most influential.

Verification:

Inspection ThermalRider_2 on page 30

2.1 Requirements Traceability

Table 2.1: Requirements Traceability

Requirement	Inspection and Testing
ThermalRider_1 - Top-level Requirements	Insp. ThermalRider_1
ThermalRider_2 - Temperature Monitoring	Test ThermalRider_1 Test ThermalRider_2
ThermalRider_3 - Minimum Functionality	Test ThermalRider_1 Test ThermalRider_2
ThermalRider_4 - Extended Functionality	Insp. ThermalRider_2

Chapter 3

Product Specification

3.1 Conceptual Design

This section describes the key concepts found in the Thermal-Rider Model. For an architectural overview, see the *Reference Manual* [1]

3.1.1 The Rider Hierarchy

The Thermal-Rider Model requires the definition, in some other model, of an Interaction Surface, which is an element of the Surface Model [4]. For details on the Surface Model, see the Surface Model Product Specification. Presented here is a very cursory overview of the Surface Model, intended to put the Thermal Rider into context.

Starting with the most general form, the Surface Model comprises a number of Facets; the facets typically have some topology and/or geometry (e.g., flat-plate with some specific area and orientation). Together, these can be used to describe the geometry of the surface, or can be left as somewhat abstract, disconnected entities. Each Facet can be further defined to become an Interaction Facet; Interaction Facets contain additional data necessary to know how to interact with some particular environmental factor.

While a generic Surface comprises a number of Facets, the associated Interaction Surface comprises the Interaction Facets associated with each of the Facets in the generic Surface. The Interaction Surface is an abstract concept that is not really implemented directly; instead specific examples of Interaction Surfaces are created, such as a radiation-pressure surface, or an aerodynamic surface, that are specific to a particular environmental interaction. Conversely, the generic Surface is simply geometric in nature and applicable to multiple situations.

Thus, each interaction model (Radiation Pressure, Aerodynamic Drag, other user-specified models) utilizes an implementation of an interaction surface that comprises a collection of model-specific Interaction Facets (e.g., Radiation Facets). These model-specific interaction facets inherit their basic data from the Interaction Facets (i.e., those defined in the Interaction Surface Model), and add model-specific values and methods to simulate the actual interaction between the surface and the environment.

One of those model-specific members that can be added to each Interaction Facet is a collection of thermally-related values. This is called the Thermal-*Facet*-Rider; it is added to each model-specific interaction facet (e.g. Radiation Facet). To accompany the rider on each facet, there is also a *model* rider that basically instructs the non-thermal instance of the Interaction Model to include thermal effects. This is called the Thermal-*Model*-Rider (italics used to distinguish between the facet rider and the model rider).

An example of using the Rider concept can be found in the default Radiation Pressure model.

> The class *radiation_presssure* contains an instance of the *ThermalModelRider*, called *thermal*. The Radiation Pressure *update* method is called from the *S_define*, which calls the ThermalModelRider *update* method. That, in turn, calls the Radiation Surface *thermal_integrator* method, which cycles through all facets calling each facet's ThermalFacetRider's *integrate* function to integrate the rate of change of temperature.

3.1.2 Independent Integration

The standard dynamic integration that takes place behind-the-scenes requires the input of forces, which are translated into accelerations, and integrated to provide updated values of velocity and position. Similarly, torques are provided for the angular components. However, those forces and torques depend on the temperature profile. Integrating the temperature profile with the dynamics will lead to unnecessary errors; these can easily be circumvented by independently integrating the temperature prior to calculating the forces that are fed to the dynamics integrator.

Therefore, the Thermal-Rider Model provides the capability to independently integrate the temperature.

3.1.3 Handling Multiple Interaction Models

Both problems and opportunities present themselves when a simulation requires the consideration of multiple interaction models.

The Thermal Rider Model is always associated with one specific instance of an Interaction Model, which contains one specific instance of an Interaction Surface – comprising interaction-specific Interaction Facets – which are based on the geometric Facets. Since each Interaction Model carries its own Interaction Facets, it must carry its own Thermal-Rider Model for that set of Interaction Facets. If each collection of Interaction Facets is based on a common geometric surface, then it is a straightforward task to direct all thermal processes to one Interaction Model, and include a *ThermalModelRider* for only that model, and *ThermalFacetRider*s for only that set of Interaction Facets. In this case, it is recommended that the Radiation Pressure Model (if being used) be the model that is used to contain the Thermal Rider Model, since it is the most closely tied to the temperature.

A more complicated scenario occurs when the different Interaction Models use different base Facets, such as when different fidelity is required between the different models. Here again, it is recommended that only one model carry the Thermal Rider. Complications arise when the desired fidelity of the Thermal-Rider Model is not directly matched with any of the Interaction Surfaces, or is only matched to one of the non-preferred models. The solution to this usually involves changing the desired fidelity on either the Thermal-Rider or one of the preferred Interaction Models.

3.2 Mathematical Formulations

3.2.1 Basic Equations

The equation for temperature of a facet is based on an emission-absorption model. Thermal emission is modeled using a black-body baseline, modified by the emissivity of the surface. The total power, P, radiated by a surface at temperature T and with surface area A is

$$P = \epsilon \sigma A T^4 \tag{3.1}$$

where ϵ is the emissivity of the surface, and σ is the Stefan-Boltzmann constant, $\sigma = 5.6704004 \times 10^{-8}\ W\ m^{-2}\ K^{-4}$

We model the power absorbed P_{abs} as the sum of the power from external sources of radiation, thermal input from the vehicle, and as a possible future extension, conduction from other facets. The current implementation ignores conduction.

The net flow of power can be represented as a rate of change of temperature by factoring in the specific heat of the facet,

$$Q \frac{dT}{dt} = P_{abs} - \epsilon \sigma A T^4 \tag{3.2}$$

where Q is the specific heat, expressed in $J \cdot K^{-1}$ and $P_{abs}$is the rate at which energy is absorbed into a facet.

3.2.2 Temperature Propagation

The conventional JEOD approach to solving 3.2 has been to use a special 4^{th} order Runge-Kutta technique which is included in the *ThermalFacetRider* class. This technique includes a series of tests in order to avoid instabilities resulting from an integration step too large to accommodate rapidly varying temperatures.

With the introduction of first order methods in the *er7_utils* package, it is now possible to have JEOD propagate temperature just as it propagates the state of a vehicle. This approach required the introduction of the new *ThermalIntegrableObject* class which is derived from *er7_utils::IntegrableObject*. The *integrate* method of *ThermalIntegrableObject* performs each step of the integration cycle. This method also checks at each step to make sure that the equilibrium temperature is not surpassed within an integration cycle.

3.2.3 Runge-Kutta 4th Order (RK4) Integrator

The rate at which temperature changes is a strong function of temperature due to the rate of thermal emission, as described in equation 3.1. It is assumed that other contributing factors change more slowly, indeed that they are constant over an integration step. All other factors are combined into the value *power_absorb*, which is positive if there is a net power into the facet (excluding thermal emission), and negative if there is a net flow out from the surface (e.g. from a thermal source within the vehicle). With those assumptions, the RK4 integrator can be implemented as follows:

T_0 represents the initial temperature at the start of the integration step.

The first full-step approximation to the temperature change, I_1, is calculated as

$$I_1 = \frac{\Delta t}{Q}\left(P_{abs} - \epsilon\sigma A T_0^4\right)$$

The second step uses the mid-point temperature as the operating temperature, calculated as the average of T_0 and $T_0 + I_1$.

$$T_1 = T_0 + \frac{I_1}{2}$$

$$I_2 = \frac{\Delta t}{Q}\left(P_{abs} - \epsilon\sigma A\left(T_1\right)^4\right)$$

The third step uses the mid-point temperature, calculated as the average of T_0 and $T_0 + I_2$ as the operating temperature,

$$T_2 = T_0 + \frac{I_2}{2}$$

$$I_3 = \frac{\Delta t}{Q}\left(P_{abs} - \epsilon\sigma A\left(T_2\right)^4\right)$$

The fourth step uses the predicted end value, $T_0 + I_3$ as the operating temperature,

$$T_3 = T_0 + I_3$$

$$I_4 = \frac{\Delta t}{Q}\left(P_{abs} - \epsilon\sigma A\left(T_3\right)^4\right)$$

The actual change in temperature is then calculated as a weighted average of those 4 values,

$$\Delta T = \frac{I_1 + 2I_2 + 2I_3 + I_4}{6}$$

3.2.3.a Integrating close to the asymptote

If the integration time-step is large compared to the characteristic time over which the temperature can change appreciably, there is the potential that the RK4 integrator can become unstable. This is due to the asymptotic nature of the temperature$\sim$time evolution; whether the temperature is above or below the equilibrium temperature, the temperature will asymptotically approach equilibrium. The instability is observed when the integration time-step is sufficiently large that the local linearization of that asymptotic approach jumps over the asymptote in one step, and then tries to approach from the other side.

Figure 3.1 illustrates the temperature$\sim$time profile, with the asymptotic trend both above and below the asymptote at T_{eq}. Note that the two profiles are not symmetric; the upper profile is a little steeper for the same temperature difference from T_{eq}, making this instability more likely when approaching from above than when approaching from below.

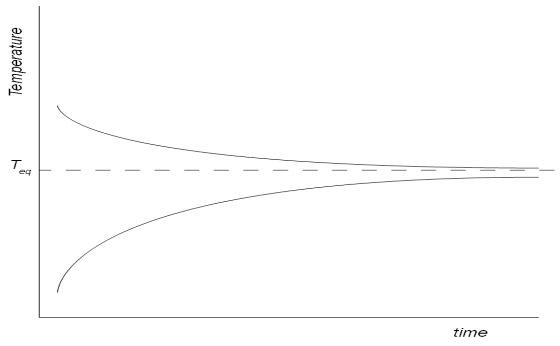

Figure 3.1: Illustration of the temperature∼time profile near the equilibrium temperature.

Aside: The rate at which the surface radiates energy is $P_{rad} \propto T^4$.
The rate at which the surface absorbs energy is some constant P_{abs}.
At equilibrium, these two are equivalent, $P_{abs} = P_{rad} \propto {T_{eq}}^4$.
The net rate at which energy increases is therefore

$$\epsilon = (P_{abs} - P_{rad}) \propto \frac{dT}{dt}.$$

Therefore,

$$\frac{dT}{dt} \propto ({T_{eq}}^4 - T^4).$$

Then for some deviation from the equilibrium temperature:

$$T = T_{eq} + \Delta T \Rightarrow \frac{dT}{dt} \propto (-)\,(T_{eq})^4 \left(4\left(\frac{\Delta T}{T_{eq}}\right) + 6\left(\frac{\Delta T}{T_{eq}}\right)^2 + 4\left(\frac{\Delta T}{T_{eq}}\right)^3 + \left(\frac{\Delta T}{T_{eq}}\right)^4\right)$$

The difference between above the asymptote and below the asymptote results from the sign change on ΔT. The first and third terms will have different signs for the two cases, whereas the second and fourth will have the same sign regardless of whether the temperature is above or below the asymptote. The consequence is that above the asymptote, all terms are additive, and below the asymptote the terms tend to cancel each other.

Figure 3.2 illustrates the problem with having an integration time-step that is too large. In the first step, the original temperature (T_0) is used to calculate the rate of change of temperature, and the predicted change to that temperature (I_1). Next, that change is used to derive an intermediate temperature ($T_1 = T_0 + 0.5I_1$), which is below the asymptote. This temperature is used to calculate a new temperature gradient which is now positive; that temperature gradient is used to calculate a new correction (I_2, not shown) to the original temperature. The next intermediate temperature ($T_2 = T_0 + 0.5I_2$) will then be higher than T_0, therefore have a steeper gradient, which leads to I_3 having a larger deviation than did I_1. Thus, the calculation of $T_3 = T_0 + I_3$ falls far below the equilibrium value, which is highly erroneous, and further leads to a ridiculously large value for I_4. The consequence is that the temperature change could be driven to non-physical values, and in the opposite direction to which it should have been evolving.

To prevent this instability, there are numerous checks made during the integration. First, the direction of anticipated temperature change is recorded as *dt_dir* (± 1 corresponding to up and down) by comparing the original temperature to the equilibrium temperature.

Next, I_1 is calculated; it is assumed that I_1 has the correct direction to it. I_1 is used to calculate T_1 and thereby I_2, the latter of which is compared against *dt_dir* to ensure that it is in the correct direction. If this is not the case, the predicted temperature is set to the equilibrium temperature and the integration ends.

The same check could be made on I_3, but it is not necessary. Since T_1 is closer than T_0 to T_{eq}, the magnitude of I_2 must be less than the magnitude of I_1. Therefore, if I_1 was insufficient to cross the asymptote, I_2 must have been likewise; hence if I_2 was in the correct direction, I_3 must also be in the correct direction.

The same is not true of I_4, since T_3 uses a full-step correction to T_0, whereas T_1 and T_2 used only half-step corrections. T_3 could, therefore, be on the wrong side of the equilibrium temperature. Incorporating I_4 in this situation would lead to a calculated temperature change that is too small; ignoring it would lead to the value being too large, and the final temperature being very close to equilibrium. Instead, one-half of the value is taken.

The four temperature deviations are combined then to give the final temperature change.

Once again, checks are made that this is sensible. If the temperature change was in the wrong direction (this may be impossible), or if the temperature change results in a new temperature that is on the wrong side of equilibrium (this, too, may be impossible) the temperature is set to the equilibrium temperature.

3.2.4 Emitted Power

The mean emitted power over the integration step can now be calculated from the resulting change to the temperature.

$$P_{emit} = Q\frac{\Delta T}{\Delta t} - P_{abs} \tag{3.3}$$

Note: a positive value of P_{emit} means that the facet gains energy by emission (physically impossible). P_{emit} is typically negative.

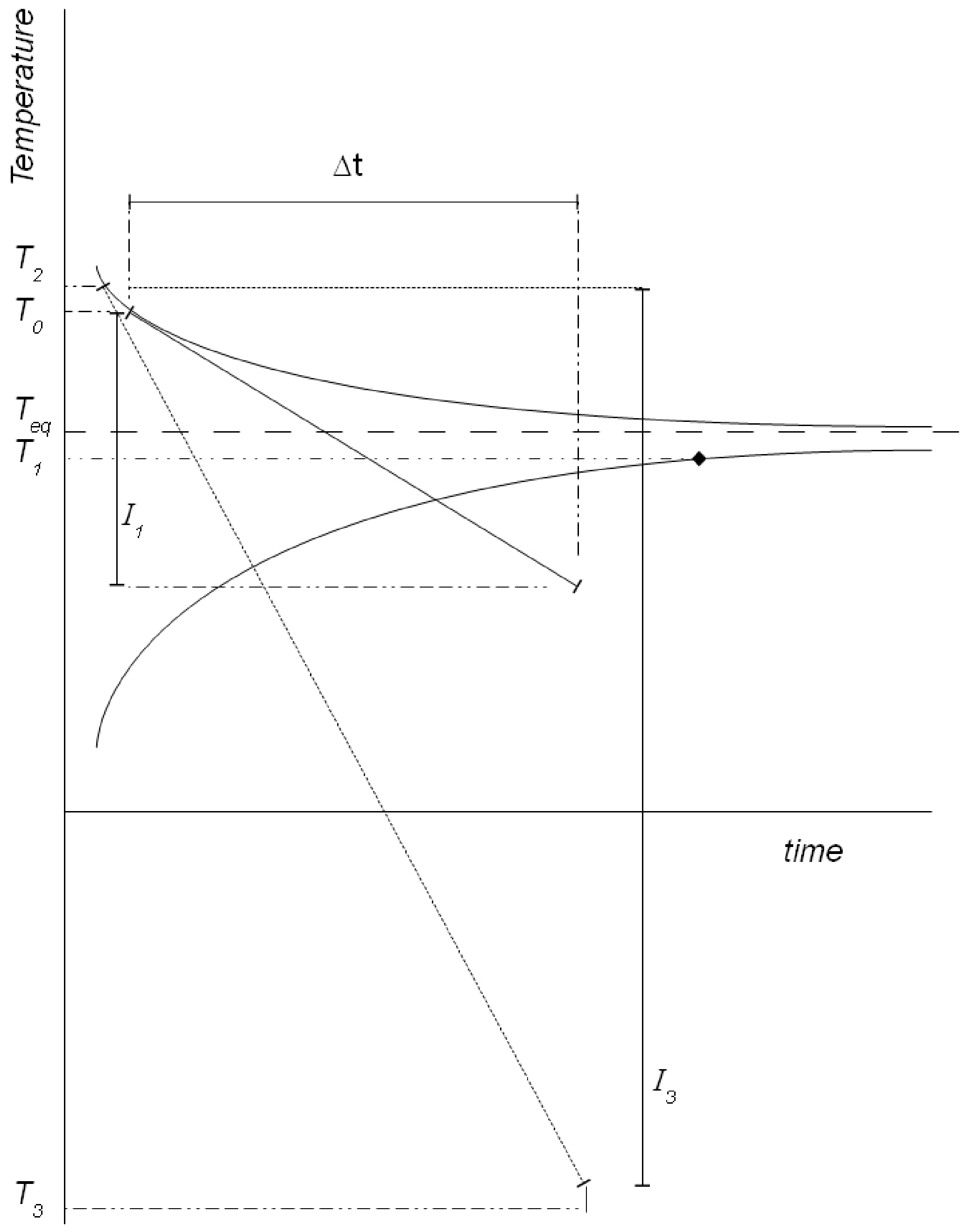

Figure 3.2: Illustration of the instability near the equilibrium temperature.

3.3 Detailed Design

This section is divided into 2 parts:

- A process flow-through description, or Process Architecture of the sequence in which the various functions are called, and the interaction between the objects.
- An alphabetized list of the objects that comprise the Thermal-Rider Model, with reference to the parent class where appropriate, and description of the functions contained within those object.

Further, the *Reference Manual* [1] contains a structural overview of the Thermal-Rider Model.

3.3.1 Process Architecture

3.3.1.a Initialization

The initialization of the Thermal-Rider Model starts with the implementation of the generic aspects of the Interaction Model (on which the Thermal-Rider Model rides):

1. The Interaction Surface is created.
2. The specific Interaction Model is initialized.

For specific information on these processes, see the appropriate documents (Surface Model[4] and, for example, Radiation Pressure[6]).

1. The initialization of the generic Interaction Surface has little direct impact on this model, it creates the surface for later population by an Interaction Model.
2. The initialization of the specific Interaction Model must include the following tasks:
   (a) The *thermal.active* flag must be set (to be *true* or *false*). If it is set to false, the Thermal-Rider Model will not be utilized. The rest of the initialization proceeds regardless of this flag setting; this allows the user to switch the flag mid-simulation and still have the Thermal-Rider Model available immediately.
   (b) The initialization method for the specific instance of the Interaction Surface (e.g. Radiation Surface) must be called.

   The initialization routine for the surface calls the initialization routine for each facet in that surface, which in turn call the *initialize* (on page 21) routine located in its attached ThermalFacetRider object.

3.3.1.b Run-time

The Thermal-Rider Model functionality is first called from the Interaction Model with which it is associated, as shown in figure 3.3 on the next page.

The update function of the Thermal-Rider Model comprises two sections (see figure 3.4 on page 16).

1. The extent of the model is tested, to identify whether it is desirable to include additional thermal sources. If so, the *accumulate_thermal_sources* method is run from the Interaction Surface (see figure 3.5 on page 17).

   This surface version of this method in turn calls the *accumulate_thermal_sources* (on page 21) method from the Thermal Facet Rider attached to each of the Interaction Facets.

   This function (see figure 3.6 on page 18) in turn simply adds the potential sources of energy for the given facet, and sets the default equilibrium condition that the power emitted and the power absorbed are equal.

2. It can be useful to accumulate the power sources, even if there is no intent to monitor the temperature profile. There is therefore another verification, that the user does intend that the temperature of the facets be variable. If so, a call is made to the *thermal_integrator* from the Interaction Surface (see figure 3.7 on page 19).

   This function in turn calls the *integrate* (on page 21) method from the Thermal Facet Riders attached to each of the Interaction Facets.

   It is this method (see figure 3.8 on page 20) that runs the RK4 integrator on the temperature. The value of power emitted from the facet is re-calculated, obviously in light of the non-equilibrium conditions now being applied. Finally, the Thermal Facet Rider on each facet stores the integrated temperature (i.e. the temperature at the end of the next integration step) for later use.

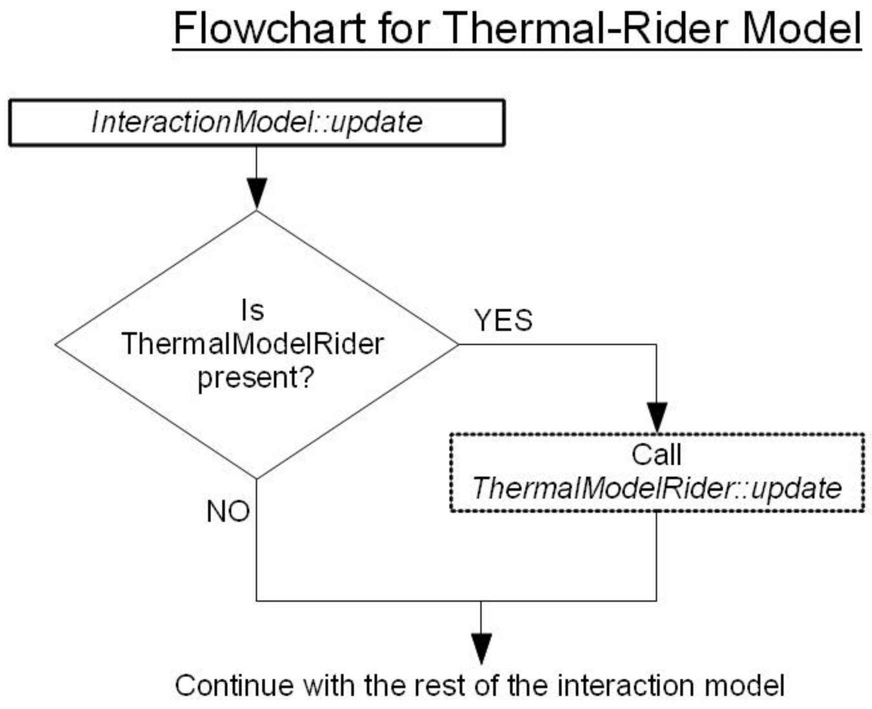
Flowchart for Thermal-Rider Model
InteractionModel::update
Is ThermalModelRider present?
YES
NO
Call ThermalModelRider::update
Continue with the rest of the interaction model

Figure 3.3: Illustration of how the Thermal-Rider Model is incorporated into the simulation.

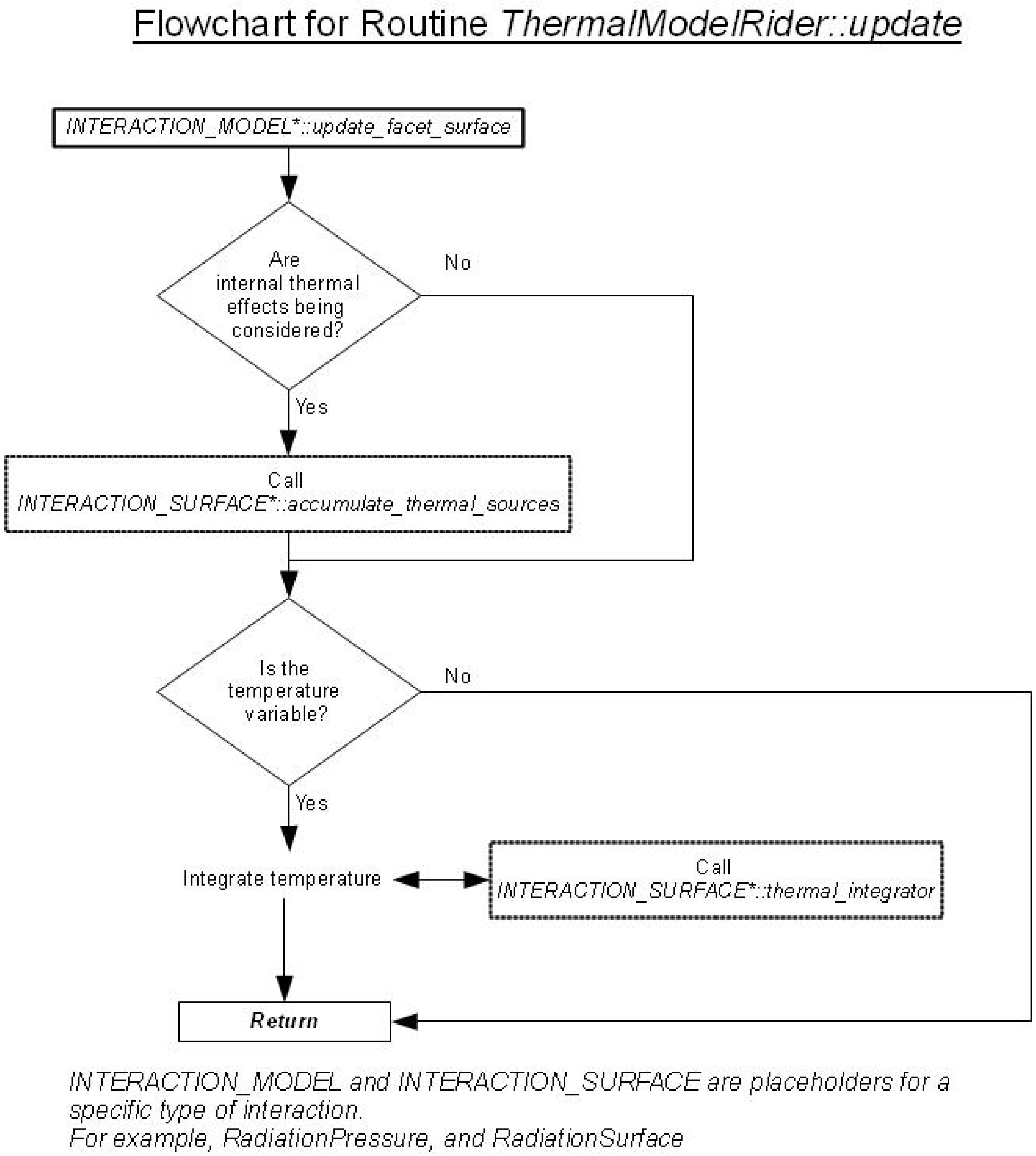
Flowchart for Routine *ThermalModelRider::update*
INTERACTION_MODEL*::update_facet_surface
Are internal thermal effects being considered?
No
Yes
Call INTERACTION_SURFACE*::accumulate_thermal_sources
Is the temperature variable?
No
Yes
Integrate temperature
Call INTERACTION_SURFACE*::thermal_integrator
Return
INTERACTION_MODEL and INTERACTION_SURFACE are placeholders for a specific type of interaction.
For example, RadiationPressure, and RadiationSurface

Figure 3.4: The Thermal-Rider Model controls the calls to the interaction surface.

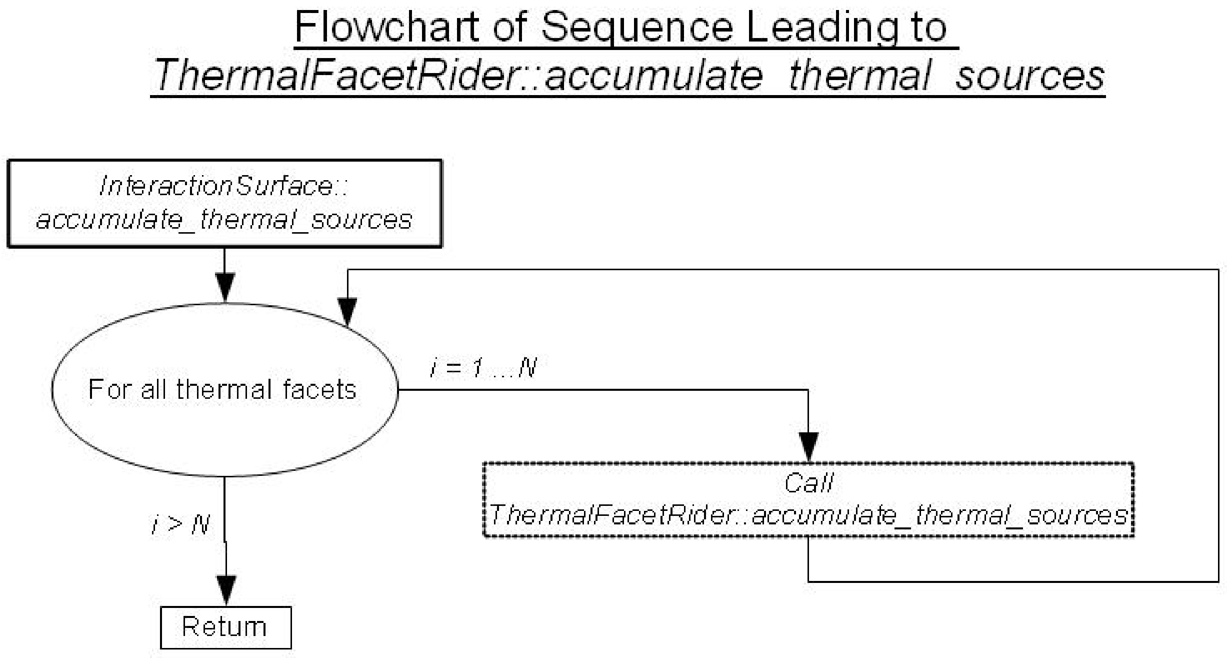
Flowchart of Sequence Leading to
ThermalFacetRider::accumulate_thermal_sources
InteractionSurface::
accumulate_thermal_sources
For all thermal facets
i = 1 ... N
Call
ThermalFacetRider::accumulate_thermal_sources
i > N
Return

Figure 3.5: The Interaction Surface controls the calls to each of the interaction facets to independently accumulate the thermal sources on that facet.

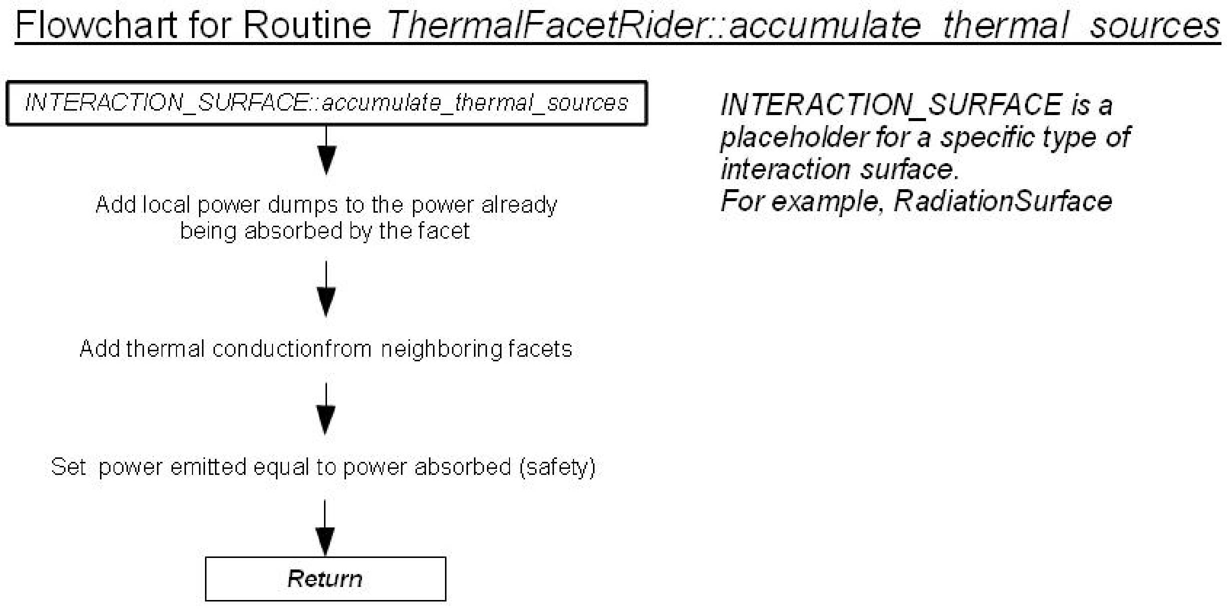
Flowchart for Routine ThermalFacetRider::accumulate_thermal_sources
INTERACTION_SURFACE::accumulate_thermal_sources
Add local power dumps to the power already being absorbed by the facet
Add thermal conductionfrom neighboring facets
Set power emitted equal to power absorbed (safety)
Return
INTERACTION_SURFACE is a placeholder for a specific type of interaction surface.
For example, RadiationSurface

Figure 3.6: Each facet uses the functionality contained within its thermal rider to collect (calculating where necessary) the thermal sources being absorbed by that surface.

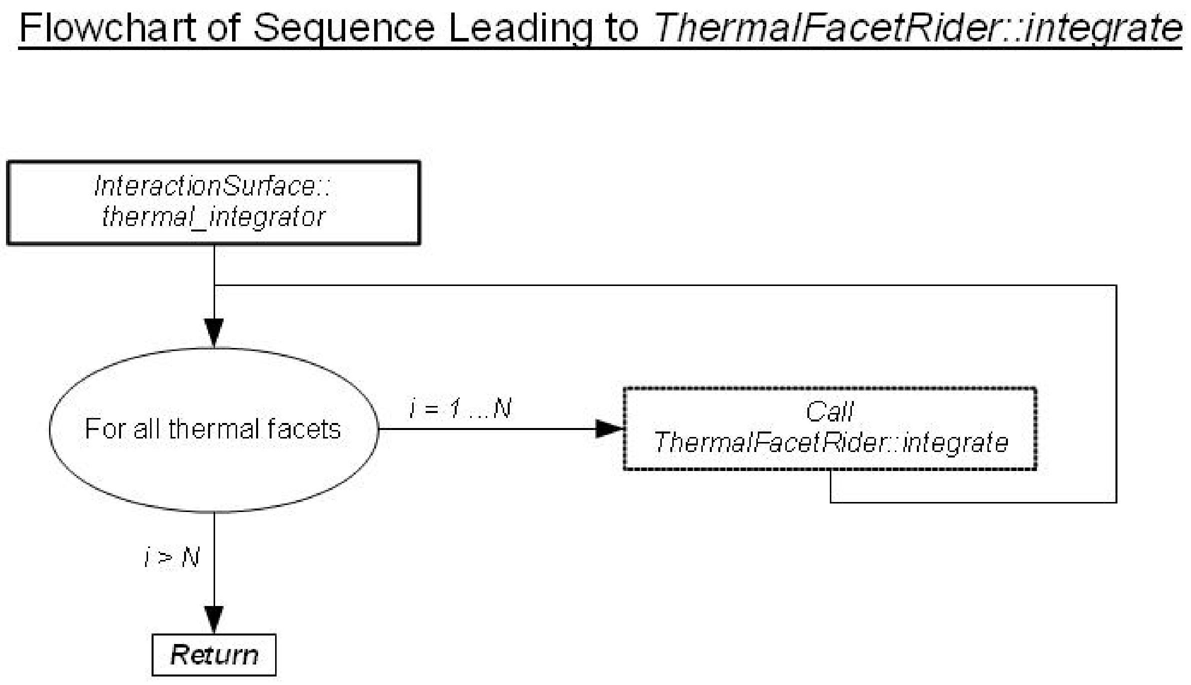
Flowchart of Sequence Leading to ThermalFacetRider::integrate
InteractionSurface:: thermal_integrator
For all thermal facets
i = 1 ...N
Call ThermalFacetRider::integrate
i > N
Return

Figure 3.7: The interaction surface controls the calls to each of the interaction facets to independently integrate the rate of change of temperature resulting from the accumulated thermal sources.

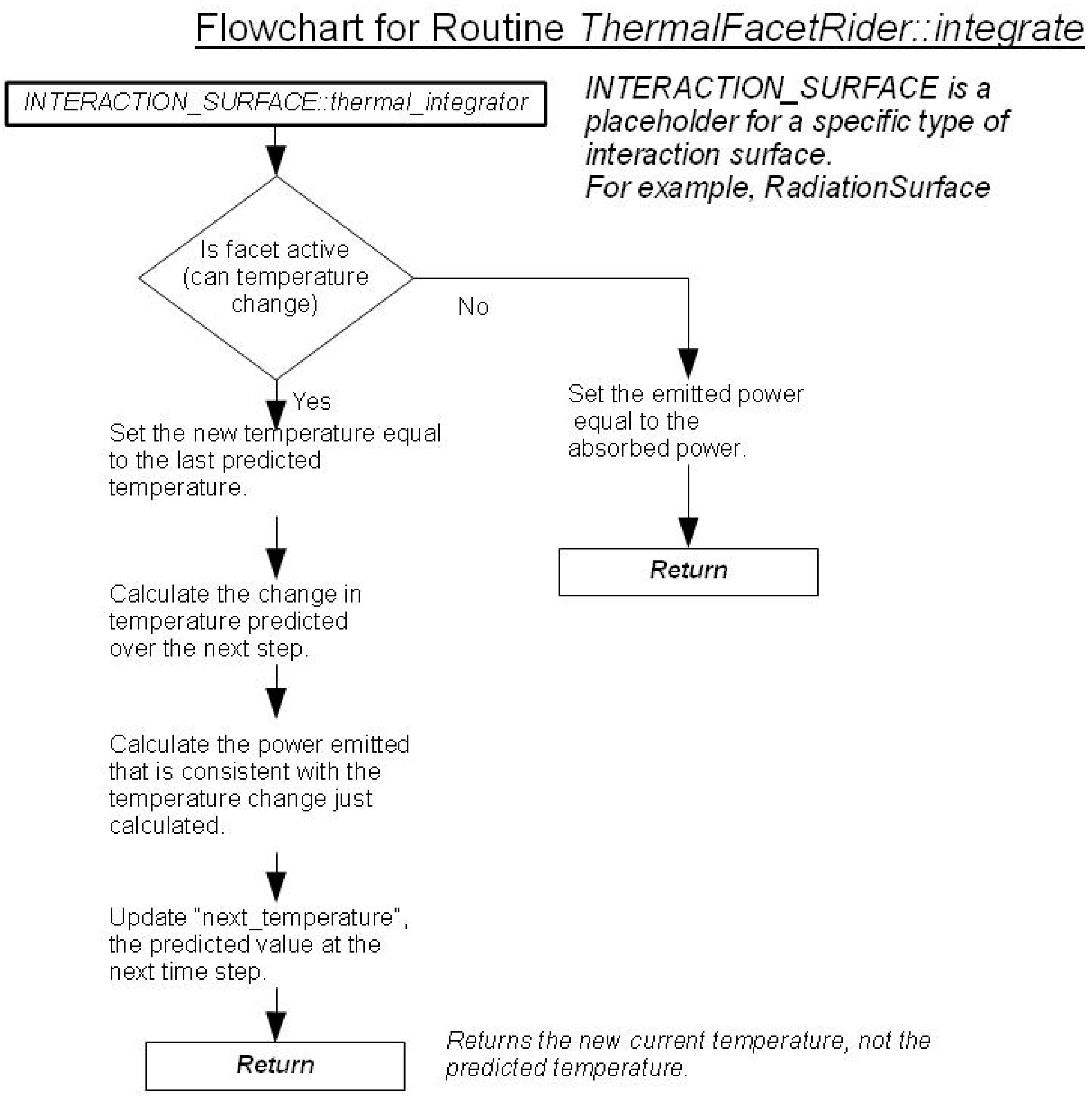
Flowchart for Routine ThermalFacetRider::integrate
INTERACTION_SURFACE::thermal_integrator
INTERACTION_SURFACE is a placeholder for a specific type of interaction surface.
For example, RadiationSurface
Is facet active (can temperature change)
No
Yes
Set the new temperature equal to the last predicted temperature.
Set the emitted power equal to the absorbed power.
Return
Calculate the change in temperature predicted over the next step.
Calculate the power emitted that is consistent with the temperature change just calculated.
Update "next_temperature", the predicted value at the next time step.
Return
Returns the new current temperature, not the predicted temperature.

Figure 3.8: Each facet uses the functionality contained within its thermal rider to integrate the temperature variation across one integration step, and from that, determine the mean rate at which thermal radiation was emitted.

3.3.2 Functional Design

See the Reference Manual[1] for a summary of member data and member methods for all classes. This section describes the functional operation of the methods in each class.

Objects are presented in alphabetical order. Methods are presented in alphabetical order within the object in which they are located.

1. **Class: ThermalFacetRider** **Inheritance:** Base-class

   This is the rider that attaches to a particular instance of the generic and virtual Interaction-Facet object.

   Methods:

   (a) **accumulate_thermal_sources**
   It is assumed that any external, model-specific, heating sources have already been accumulated (e.g., as a result of radiative absorption (*radiation_pressure*), or aerodynamic drag heating (*aerodynamics*)). This method is intended to be used to accumulate vehicular sources of energy, such as a power sink/source from within the vehicle, or conduction between facets.

   (b) **initialize**
   When the facet is initialized, we set both the *dynamic_temperature* (that is the operating temperature) and the *next_temperature* (the operating temperature predicted at the end of the current integration step) to the initialization value.

   The radiative emission, known from equation 3.1 to be $P_{emit} = \epsilon\sigma AT^4$ must be calculated repeatedly. Three of those values are constants, so those are stored together in a new variable, *rad_constant* $=\epsilon\sigma A$.

   (c) **integrate**
   The integration routine is described in detail in the *integrator* (on page 8) description found in the Mathematical Formulation section of this document.

2. **Class: ThermalIntegrableObject** **Inheritance:**er7_utils::IntegrableObject

   This is a derived class of a generic object specialized for thermal integration which can be integrated by JEOD.

   Methods:

   (a) **compute_temp_dot**
   Computes the time derivative of temperature and the instantaneous rate of emission.

   (b) **create_integrators**
   Required by *er7_utils::IntegrableObject*

   (c) **destroy_integrators**
   Required by *er7_utils::IntegrableObject*

   (d) **initialize**
   Sets the initial temperature and caches a pointer to the *ThermalFacetRider* associated with the facet to be integrated.

(e) **integrate**
Implements the *integrate* method of *er7_utils::IntegrableObject*

(f) **get_temp**
Accessor method for the temperature

(g) **get_temp_dot**
Accessor method for time derivative of temperature.

(h) **reset_integrators**
Required by *er7_utils::IntegrableObject*

3. **Class: ThermalMessages** **Inheritance:** Base-class

   This is the message handler for the Thermal-Rider Model. It has no methods.

4. **Class: ThermalModelRider** **Inheritance:** Base-class

   This is the object that rides on the interaction model directly.

   Methods:

   (a) **update**
   If the model is to include thermal sources coming from within the vehicle, the *model* method *accumulate_thermal_sources* is called, which ultimately calls the *facet* method *accumulate_thermal_sources* (on the preceding page) for each of the facets.

   If the temperature of the surface is intended to be dynamic, the routine *thermal_integrator* is called which ultimately calls *integrate* (on the previous page) for each of the facets.

5. **Class: ThermalParams** **Inheritance:** Base-class

   This class contains the collection of data for a thermal facet; it contains no methods.

3.4 Inventory

All Thermal-Rider Model files are located in the directory
`${JEOD_HOME}/models/interactions/thermal_rider`. Relative to this directory,

- Header and source files are located in the model `include` and `src` subdirectories. Table **??** lists the configuration-managed files in these directories.
- Documentation files are located in the model `docs` subdirectory. See table **??** for a listing of the configuration-managed files in this directory.
- Verification files are located in the model `verif` subdirectory. See table **??** for a listing of the configuration-managed files in this directory.

Chapter 4

User's Guide

The User's Guide is divided into 3 components, one for each of three different types of user.

The Analysis section of the User's Guide is intended primarily for users of pre-existing simulations. It comprises the following elements:

- A description of how to modify Thermal-Rider Model variables after the simulation has compiled, including an in-depth discussion of the input file;
- An overview of how to interpret (but not edit) the S_define file;
- A sample of some of the typical variables that may be logged.

The Integration section of the User's Guide is intended for simulation developers. It describes the necessary configuration of the Thermal-Rider Model within an S_define file, and the creation of standard run directories. The latter component assumes a thorough understanding of the preceding Analysis section of the user guide. Where applicable, the user may be directed to selected portions of Product Specification (Chapter 3).

The Extension section of the User's Guide is intended primarily for developers needing to extend the capability of the Thermal-Rider Model. Such users should have a thorough understanding of how the model is used in the preceding Integration section, and of the model specification (described in Chapter 3).

4.1 Analysis

Determining whether or not the Thermal-Rider Model is included in a simulation is non-obvious; because the model rides on some other model, it is unlikely to appear as an *S_define* file entry. Instead, it is necessary to look for the data files that are used to define the vehicle surface. An example of this is found in the radiation pressure model, at
interactions/radiation_pressure/verif/SIM_2_SHADOW_CALC/SET_test/RUN_ten_plates/input
(here, the Thermal-Rider Model is riding on the Radiation Pressure Model). We will use this example as a guide to establishing the Thermal Rider on any model.

At some point in any simulation that utilizes a surface-interaction, the surface must be defined; that definition is typically carried out through a data file. There may also be some values that are set in the input file. This particular input file contains the surface description at the very end, and includes the data file *Modified_data/radiation_surface_v2.d*, which defines the surface. Simple models, with only one surface, may put all of the thermal characteristics into the input file (e.g., *interactions/radiation_pressure/verif/SIM_1_BASIC/SET_test/RUN_basic/input*, which includes a data file for the full surface, and just a few lines for the default spherical surface).

The Thermal-Rider Model descriptors fall into one of three categories:

- The model-wide descriptors.
- The facet-specific descriptors.
- The facet parameter descriptors (material-specific).

The facets are the sections of the overall vehicle surface into which that surface has been divided; they are defined and controlled at an individual level.

4.1.1 Model-wide Descriptors

These values are found one level below the thermal rider, which is one level below the interaction model on which the Thermal-Rider Model rides, e.g., *radiation.rad_pressure.thermal.****.

- *active*
  The *active* flag toggles the entire Thermal-Rider Model on or off. The default value is *false* (i.e., off). This flag would typically be set in an input file, or in the case of the Radiation Pressure Model (upon which we are currently riding), this flag is automatically set to true, since the Radiation Pressure Model cannot function without a Thermal Rider.
- *include_internal_thermal_effects*
  This flag allows for the inclusion of thermal flow within the vehicle, e.g., from a power source inside the vehicle, or facet-to-facet conduction. This flag also defaults to *false.*

4.1.2 Facet-specific Descriptors

These values are found one level below the thermal rider, which is one level below the facet on which it rides, which is one level below the surface, e.g., *radiation.rad_surface.facets.thermal.****

- *active*
  In addition to having the capability to turn the entire thermal model on or off, facets are equipped with an individual on/off flag to indicate whether or not to integrate the temperature. A facet that is inactive ($active = false$) will just retain a constant temperature.
- *heat_capacity*
  This can be defined for the facet at an individual level, or in the parameters list as a heat capacity per unit area (a material property), which is then converted into the facet-specific *heat_capacity* by functionality within the model-specific Interaction Facet.

- *thermal_power_dump*
  This value allows some level of thermal power to be transfered to the facet from within the vehicle if the model-wide *include_internal_thermal_effects* flag is set. A positive value represents flow into the facet, a negative value represents flow out of the facet.

4.1.3 Facet Parameter Descriptors

These values describe the material with which a facet is made. Each facet is assigned a parameter list; this avoids repetition of identical information from facet to facet. In our example, only one material is defined, *radiation_test_material*, and it is assigned to all facets.

- *emissivity*
  The emissivity measures how similar to a black-body a material is in the thermal emission that it gives off. An emissivity of 1.0 represents a perfect black-body, an emissivity of 0.0 implies that the surface cannot radiate at all.
- *heat_capacity_per_area*
  This is converted into the facet-specific *heat capacity* by functionality within the model-specific Interaction Facet. It is useful to allow for comparable behavior across facets of vastly different size without the necessity of calculating the heat capacity for each facet individually.

4.2 Integration

Including the Thermal Rider option into any existing surface is fairly straightforward, the most difficult part of using the Thermal-Rider Model is in creating the *Interaction Surface* on which it will ride. The Thermal-Rider Model comes in two parts, one attached to each of the facets, and one attached to the interaction model itself (the *thermal facet rider* and *thermal model rider* respectively). We will treat these individually. If the interaction model does not include both the model rider and the facet riders, see the section *Extension* (on the following page) for details on how to extend an interaction model to include the Thermal Rider.

4.2.1 Thermal-Rider Model

The main class for the interaction model (e.g. *RadiationPressure* for the Radiation Pressure Model) must contain an instance of the Thermal-Rider Model. Every Interaction Model must already have an *update* (or similar) method, through which the interaction forces are updated. This method must call the *thermal.update* method (*thermal* being the name of the instance of the Thermal-Rider Model). The setting of data values necessary to the Thermal-Rider Model is covered in the section *Analysis* (on page 24).

4.2.2 Thermal Facet Rider

This rider must be attached to each of the facets in the surface; ideally, the thermal rider would be added when the model-specific interaction facets are defined, based on the generic Interaction Facet. See, for an example. the class *RadiationFacet* in the Radiation Pressure Model.

It is important to ensure that all facets in the surface have thermal characteristics available to them, even if the intention is to de-activate the thermal activity in many of those facets. The code is expecting that certain methods will be available to all facets, and that will only work if they all contain the Thermal Facet Rider.

Setting the data for the each of the facets is covered in section *Analysis* (on page 24).

4.3 Extension

4.3.1 Extending the Capability of the Thermal-Rider Model.

The Thermal-Rider Model is intended to be a stand-alone sub-model, used to enhance an interaction model. There is no specific plan to extend the capability of the Thermal-Rider Model.

4.3.2 Using the Thermal-Rider Model to Extend the Capability of an Interaction Model.

The Thermal-Rider Model comes in two parts, one attached to each of the facets, and one attached to the interaction model itself (the *thermal facet rider* and *thermal model rider* respectively). Both must be added to the interaction model in order for the Thermal-Rider Model to function. We will treat these individually.

4.3.3 Thermal-Rider Model

1. If the main class for the interaction model (e.g. *RadiationPressure* for the Radiation Pressure Model) does not contain an instance of the Thermal-Rider Model, add one. Call it *thermal.*
2. Every Interaction Model must already have an *update* (or similar) method, through which the interaction forces are updated. This method must perform the following additional tasks:
   (a) Call the *thermal.update* method (*thermal* being the name of the instance of the Thermal-Rider Model). This may require the addition of code to the interaction model update method; if so use the Radiation Pressure Model as a template.
   (b) Obtain the integration step size that will be used by the thermal integrator. Again, use the Radiation Pressure model as a template.
3. It is assumed that if the user makes the effort to specify the Thermal-Rider Model, that the intention is to have it active. The *model-level* active flag therefore defaults to *true* (i.e., on).

4.3.4 Thermal Facet Rider

Adding the Facet Rider is more complicated than adding the Model Rider. Since the interaction surface comprises the facets, changing the facets requires changes to the interaction surface.

4.3.4.a Thermally Active Facets

1. When the facet is created, the thermal values from the parameter list must be copied over into the facet (see *RadiationFacet::define_facet_core*).
2. Within the initialization of the Interaction Facet, the *thermal.initialize* method must be called; this sets the radiation constant that is used in modeling the thermal radiation, and initializes the dynamic and next-step temperatures.

3. If the interaction force is temperature dependent, the thermal integrator must be called from some method that is called at the regular update rate in order to maintain the correct temperature profile. Note that it is the thermal integration method that updates *thermal.dynamic_temperature* to the value calculated in the previous integration step, so this method should be called before the *current* temperature is needed.

4.3.4.b Thermally Active Surface

The thermally active surface should contain the following additional methods:

1. accumulate_thermal_sources
2. thermal_integrator
3. equalize_absorption_emission

These are simple interface methods, passing control from the model through to the individual facets one at a time. They can typically be copied verbatim from the example of a Radiation Surface in the Radiation Pressure model.

Chapter 5

Verification and Validation

5.1 Verification

5.1.1 Inspections

Inspection ThermalRider_1: Top-level Inspection

This document, the code, and associated files, have been inspected, and together satisfy requirement ThermalRider_1.

Inspection ThermalRider_2: Extended Functionality

The method *ThermalFacetRider::accumulate_thermal_sources* includes the functionality to add thermal power sources or sinks (see *Verification of Intra-vehicular Thermal Transfer* (on page 35)), and a good description of how a conduction model could be implemented. The addition of heating due to aerodynamic drag has not been investigated in depth, but it is anticipated that this would be handled by the aerodynamics model by adding a Thermal Rider to the existing aerodynamics model. This satisfies requirement ThermalRider_4

5.1.2 Verification of the Temperature Modeling

The simulations for this verification procedure are found in the Radiation Pressure model; this verification test is duplicated in the Radiation Pressure model.

This section is divided into two parts, one comparing the case of the vehicle without illumination — for which an analytic solution exists — and the second comparing the response of surfaces with different characteristics in the presence of illumination.

5.1.2.a Verification of thermal processes without illumination

Test ThermalRider_1: Thermal Processes - no illumination

Purpose:
: This test is to ensure that the Runge-Kutte plate-temperature integration process is functioning normally in the case of a vehicle cooling in the absence of incident radiation.

Requirements:
: Satisfactory conclusion of this test, along with test ThermalRider_2 satisfies requirements ThermalRider_2 and ThermalRider_3.

Procedure:
: The simulation used in this test is available at *radiation_pressure/verif/SIM_2_SHADOW_CALC/RUN_shadow_cooling*. A number of facets were monitored in a flux-free environment to observe their temperature variation with time. Each plate had slightly different characteristics, allowing for the dependency on emissivity, plate area, and heat capacity to be investigated. In the absence of incident flux, an analytic solution exists for the temperature and the force, and this can be compared to the numerical solution.

Results:

1. Variation with time of temperature
   Simulation data of the variation with time of temperature matched very well to analytic solution. On any scale longer than 3 timesteps, the graphs of analytical values and simulation values were indistinguishable, with a difference less than 0.05%, and very much smaller than the observed changes in temperature. See Figure 5.1.
2. Variation with emissivity of the temporal evolution of temperature
   As the emissivity increases, the rate of change of temperature also increases, as expected. For ϵ=0, no temperature change was observed.
3. Variation with starting temperature of the temporal evolution of temperature
   With a lower starting temperature, a plate maintains a lower temperature throughout the simulation, but the temperature decreases more slowly. Temperature$\sim$time plots have identical shapes; shifting the curves along the x-axis to match temperature at some time results in matching temperatures for all (valid) time.
4. Variation with plate area of the temporal evolution of temperature
   A plate with a smaller surface area, but identical heat capacity shows a more gradual temperature drop. There is no observable difference between plates for which the heat capacity is decreased proportionally with the area (e.g. smaller plate of the same material).
5. Variation with heat capacity of the temporal evolution of temperature
   A plate with a smaller heat capacity, but identical area shows a more rapid temperature drop. There is no observable difference between plates for which the area is decreased proportionally with the heat capacity (e.g. smaller plate of the same material).

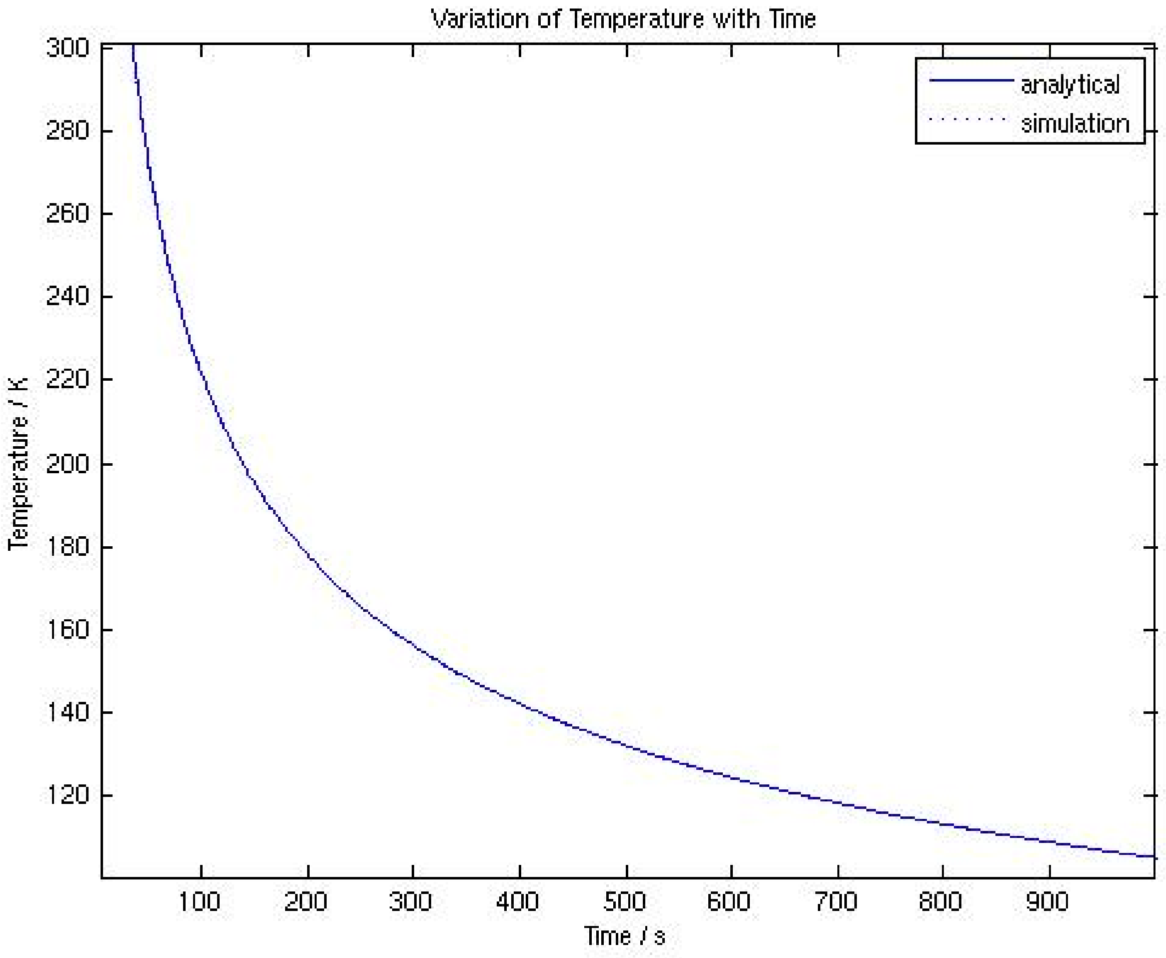

Figure 5.1: Variation of Temperature with time showing the simulation data and analytic solution overlaying.

5.1.2.b Verification of thermal processes with illumination

Test ThermalRider_2: Thermal Processes - with illumination

Purpose:

To verify that the temperature integration and thermal emission processes are functioning as expected when the vehicle is illuminated.

Requirements:

Satisfactory conclusion of this test, along with test ThermalRider_1 satisfies requirements ThermalRider_2 and ThermalRider_3.

Procedure:

The simulation used in this test is available at *SIM_2_SHADOW_CALC/RUN_ten_plates*. 10 identical plates were oriented at different angles to the flux vector, with the angle between the normal and the flux vector varying from 0 to 90 degrees. All plates were started with a preliminary temperature of 270 K, and the simulation was allowed to run until all but one plate had approached their equilibrium temperature. The force from each plate was monitored throughout the simulation.

Results:

The equilibrium temperature is calculated from the energy balance equation

$$\phi(A\cos\theta)(1-\alpha) = \epsilon\sigma A {T_{eq}}^4 \tag{5.1}$$

where

- ϕ is the radiative flux,
- θ is the angle between the surface normal and the incident radiation vector,
- α is the albedo of the surface,
- ϵ is the emissivity of the surface,
- σ is the Stefan-Boltzmann constant,
- A is the plate area, and
- T_{eq} is the equilibrium temperature.

Using values of $\alpha = 0.5$, $\epsilon = 0.5$, $\phi = 1400\ W\ m^{-2}$, and with θ varying from 0 to 90 degrees in 10 degree intervals, the equilibrium temperatures of the 10 plates are:

396 K, 395 K, 390 K, 382 K, 371 K, 355 K, 333 K, 303 K, 256 K, and 0 K.

Figure 5.2 shows the variation with time of the temperature of the 10 plates; the 9 plates that had closely approached equilibrium temperatures had final temperatures all within 1% of their analytical equilibrium temperature. The final plate, with an equilibrium temperature of 0 K has a temperature variation that is decreasing at a rate consistent with expectations.

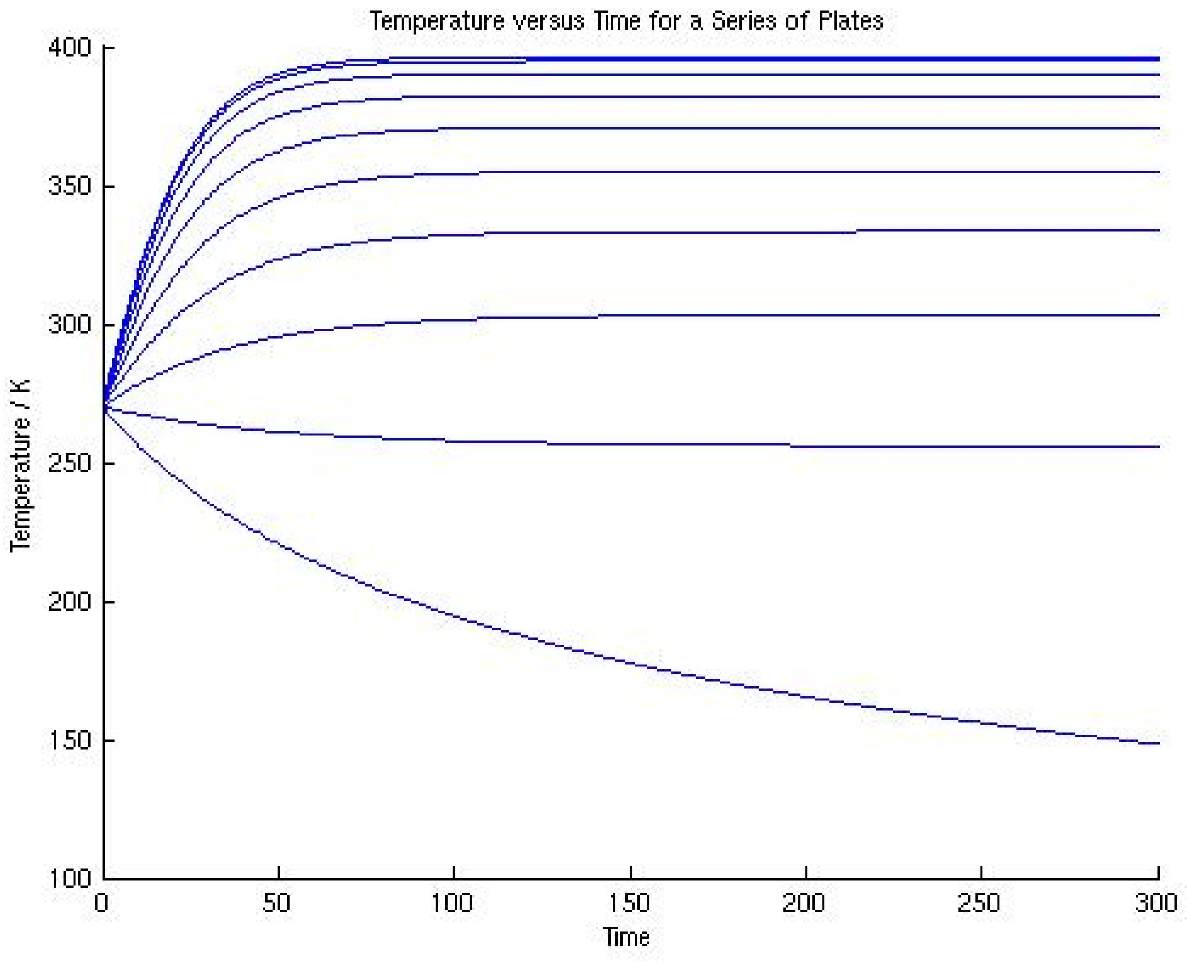

Figure 5.2: The variation of temperature with time over the 10 plates in the simulation.

Test ThermalRider_3: Verification of Intra-vehicular Thermal Transfer

Purpose:
: To test the capability of modeling thermal energy transfer from within the vehicle to the surface.

Requirements:
: Satisfactory conclusion of this test, together with inspection ThermalRider_2 satisfies requirement ThermalRider_4

Procedure:
: A test comparable to the basic Radiation Pressure test (*interactions/radiation_pressure/verif/SIM_1_BASIC/RUN_basic*) was performed (at *interactions/radiation_pressure/verif/SIM_1_BASIC/RUN_thermal_dump*). The difference being that in this test, a thermal power dump sufficient to raise the temperature of the surface by 1 degree per second was added.

Predictions:
: The temperature of the surface should initially rise by 1 degree per second above the temperature of the surface without the thermal power input. As radiative cooling increases in response, the rate of change should diminish over time; eventually a new equilibrium will be reached at some higher temperature than the case with no thermal power dump.

Results:
: The results were as expected; after 1 second, the temperature difference was 0.985 degrees; after 2 seconds, 1.94 degrees (0.97 degrees per second average); after 10 seconds, 8.6 degrees (0.86 degrees per second average); after 25 seconds, 17 degrees (0.68 degrees per second average).

5.2 Validation

There is no independent validation of the Thermal-Rider Model.

5.3 Metrics

5.3.1 Code Metrics

Table 5.1 presents coarse metrics on the source files that comprise the model.

Table 5.1: Coarse Metrics

	Number of Lines			
File Name	**Blank**	**Comment**	**Code**	**Total**
Total	0	0	0	0

Table 5.2 presents the extended cyclomatic complexity (ECC) of the methods defined in the model.

Table 5.2: Cyclomatic Complexity

Method	**File**	**Line**	**ECC**
jeod::ThermalIntegrableObject::get_temp ()	include/thermal_integrable_object.hh	121	1
jeod::ThermalIntegrableObject::get_temp_dot ()	include/thermal_integrable_object.hh	130	1
jeod::ThermalFacetRider::ThermalFacetRider (void)	src/thermal_facet_rider.cc	57	1
jeod::ThermalFacetRider::accumulate_thermal_sources (void)	src/thermal_facet_rider.cc	77	1
jeod::ThermalFacetRider::initialize (double temperature, double surface_area)	src/thermal_facet_rider.cc	132	5
jeod::ThermalFacetRider::integrate (void)	src/thermal_facet_rider.cc	179	9
jeod::ThermalFacetRider::~ThermalFacetRider (void)	src/thermal_facet_rider.cc	310	1
jeod::ThermalIntegrableObject::ThermalIntegrableObject ()	src/thermal_integrable_object.cc	44	1
jeod::ThermalIntegrableObject::~ThermalIntegrableObject ()	src/thermal_integrable_object.cc	57	1

Continued on next page

Table 5.2: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::ThermalIntegrableObject::create_integrators (const er7_utils::IntegratorConstructor & generator, er7_utils::IntegrationControls & controls, const er7_utils::TimeInterface & time_if JEOD_UNUSED)	src/thermal_integrable_object.cc	67	1
jeod::ThermalIntegrableObject::destroy_integrators (void)	src/thermal_integrable_object.cc	83	1
jeod::ThermalIntegrableObject::reset_integrators (void)	src/thermal_integrable_object.cc	94	1
jeod::er7_utils::integrate (double dyn_dt, unsigned int target_stage)	src/thermal_integrable_object.cc	105	2
jeod::ThermalIntegrableObject::initialize (double temperature, ThermalFacetRider &associated_rider)	src/thermal_integrable_object.cc	142	1
jeod::ThermalIntegrableObject::compute_temp_dot (void)	src/thermal_integrable_object.cc	159	1
jeod::ThermalModelRider::ThermalModelRider (void)	src/thermal_model_rider.cc	50	1
jeod::ThermalModelRider::update (InteractionSurface * surface_ptr)	src/thermal_model_rider.cc	60	4
jeod::ThermalModelRider::~ThermalModelRider (void)	src/thermal_model_rider.cc	80	1
jeod::ThermalParams::ThermalParams (void)	src/thermal_params.cc	44	1
jeod::ThermalParams::~ThermalParams (void)	src/thermal_params.cc	55	1

Bibliography

[1] Generated by doxygen. *Thermal Rider Reference Manual*. NASA, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, July 2023.

[2] Jackson, A., Thebeau, C. JSC Engineering Orbital Dynamics. Technical Report JSC-61777-docs, NASA, Johnson Space Center, Houston, Texas, July 2023.

[3] NASA. NASA Software Engineering Requirements. Technical Report NPR-7150.2, NASA, NASA Headquarters, Washington, D.C., September 2004.

[4] Spencer, A. Surface Model. Technical Report JSC-61777-utils/surface_model, NASA, Johnson Space Center, Houston, Texas, July 2023.

[5] Spencer, A. Aerodynamics Model. Technical Report JSC-61777-interactions/aerodynamics, NASA, Johnson Space Center, Houston, Texas, July 2023.

[6] Turner, G. Radiation Pressure Model. Technical Report JSC-61777-interactions/radiation_pressure, NASA, Johnson Space Center, Houston, Texas, July 2023.